Simon Rustige, BSc

Development and Implementation of Temperature Distribution Simulation Code for Stirred Industrial Bioreactors based on Lattice Boltzmann Methods

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree program: Chemical and Process Engineering

submitted to

Graz University of Technology

Advisor:

Univ.-Prof. Dipl.-Ing. Dr. techn. Johannes Khinast
Institute of Process and Particle Engineering

Co-Advisors:

Dipl.-Ing. Dr. Christian Witz
Dipl.-Ing. Philipp Eibl, BSc
Institute of Process and Particle Engineering

Graz, April 2019

AFFIDAVIT

Abstract

The Institute of Particle and Process Engineering of the TU Graz is developing a lattice Boltzmann method (LBM) based CFD simulation and modeling framework with fully parallel computation on GPUs for aerated stirred industrial scale fedbatch bioreactors. The simulation of the liquid temperature is a feature that is defining for the modeling of liquid viscosity, biological growth, production rate as well as oxygen solubility and mass transfer kinetics in the reactor. The work of this master's thesis includes the implementation and testing of a parallelizable LBM based algorithms for the calculation of transient and locally resolved temperature distribution that can be integrated in the existing simulation framework. The single relaxation time double distribution function SRT-DDF algorithm proposed by Peng et al. in 2003 resulted in numerical instabilities for the heat transfer in water at typical Peclet numbers of $Pe \approx 10^5$ and lattice Mach number of $\mathrm{Ma} < 0.02$. Since the numerically stable region of this approach lies at unphysically high Prandtl numbers the alternative LBM based algorithm for purely advective transport of scalars proposed by Osmanlic et al. in 2016 was investigated. An analytical derivation of the numerical diffusivity of the highly stable and conservative algorithm for hot bulks advected by constant fluid velocity shows that the numerical diffusivity is proportional to the velocity in lattice units in the system. The numerical diffusivity is in the range of the physical thermal diffusivity of water substance at about 0.1% of the stirrer's blade tip speed. For the application of the simulation tool for bioreactors heating jackets and tube bundle heat exchangers can be initialized as constant heat sources or sinks. The code structure introduces the heat due to reaction by microorganisms which are represented by a Lagrangian parcel approach. Moreover, the heat due to viscous dissipation calculated by the LES Smagorinsky model is taken into account. The simulation of temperature was coupled to the fluid field simulation by applying the Vogel-Tamman-Fulcher temperature-viscosity model for pure water substance.

Kurzfassung

Das Institut für Prozess und Partikeltechnik der TU Graz entwickelt eine auf der Lattice-Boltzmann-Methode basierende CFD Modell- und Simulationsumgebung für begaste, gerührte, industrielle Fed-Batch Bioreaktoren, dessen Rechenschritte voll parallelisiert auf GPUs operieren. Hierbei ist die Simulation der Flüssigkeitstemperatur ein wichtiger Bestandteil für die Modellierung von Viskosität, biologischem Wachstum, Produktionsraten sowie Sauerstoffsättigung und Stoffübergangskinetik. Diese Masterarbeit enthält das Implementieren und Testen von parallelisierbaren, auf der Lattice-Boltzmann-Methode basierenden Algorithmen für die Berechnung von instationärer, lokal aufgelöster Temperaturverteilung, welche in die existierende Simulationsumgebung integriert werden können. Die sogenannte Single-Relaxation-Time-Double-Distribution-Function SRT-DDF Methode von Peng et al. von 2003 führt zu numerischer Instabilität bei gegebener Pecletzahl um $Pe = 10^5$ and Lattice-Machzahlen von $Ma < 0.02$. Da der stabile Bereich dieses Ansatzes bei unpyhysikalisch hohen Prandtlzahl liegt, wurde der von Osmanlic et al. im Jahr 2016 vorgeschlagene alternative, auf LBM basierende Algorithmus für den rein advektiven Transport von Skalaren untersucht. Eine analytische Herleitung der numerischen Diffusivität des stabilen und konservativen Algorithmus für Warmzonen bei konstanter Strömungsgeschwindigkeit zeigt, dass die numerische Diffusivität proportional zur Geschwindigkeit in Latticeeinheiten im System ist. Die numerische Diffusivität liegt im Bereich der physikalischen Temperaturleitfähigkeit von Wasser bei etwa 0,1% der Schaufelspitzengeschwindigkeit des Rührwerks. Für den Einsatz des Simulationstools für Bioreaktoren können Heizmäntel und Rohrbündelwärmetauscher als konstante Wärmequellen oder Senken initialisiert werden. Die Codestruktur ermöglicht die Einbringung von Wärme durch Reaktion von Mikroorganismen, die durch den Parcel-Ansatz bei Lagrangescher Betrachtungsweise repräsentiert werden. Darüber hinaus wird die mit dem LES Smagorinsky-Modell berechnete Wärme aufgrund der viskosen Dissipation berücksichtigt. Die Temperatursimulation wurde mit der Fluidsimulation gekoppelt, indem das Vogel-Tamman-Fulcher Temperatur-Viskositätsmodell für Wasser angewendet wurde.

Acknowledgement

Hi-Suk, Franz, Lara, Pascal & Pablo Rustige

Daniela Fiedler

Andreas Lacek

Christian Witz, Philipp Eibl & Johannes Khinast

All members of the *Institute of Process and Particle Engineering*

University of Technology Graz

Nomenclature

Abbreviations	
AD, ADE	advection-diffusion (equation)
API	application programming interface
BC	boundary condition
BGK	Bhatnagar, Gross and Krook
CFD	computational fluid dynamics
CPU	central processing unit
CUDA	compute unified device structure by *NVIDIA*®
DDF	double distribution function
DNS	direct numerical simulation
GPU	graphics processing unit
IE	internal energy, internal energy formulation
LB, LBM	lattice Boltzmann (method)
LHS	left hand side of an equation
LES	large eddy simulation
LU	lattice units
MRT	multi relaxation time
NS, NSE	Navier-Stokes (equations)
RANS, RANSE	Reynolds-averaged Navier-Stokes (equations)
RHS	right hand side of an equation
SRT	single relaxation time
TBHE	tube bundle heat exchanger
TLBM	thermal lattice Boltzmann method

Legend of Symbols	
a	scalar
$\boldsymbol{a}$	vector (bold)
$\boldsymbol{A}$	matrix (capital bold)
$a(b)$	quantity a is a function of quantity b

$\frac{Dx}{Dt}$	material or substantial derivative of x
$\frac{\partial x}{\partial t}$	partial derivative of x
$\boldsymbol{a} \cdot \boldsymbol{b}$	scalar or dot product of vector a and b
$\boldsymbol{a} \otimes \boldsymbol{b}$	tensor or outer product of vector a and b

Latin Symbols

A		amplitude
a	$\left[\frac{m}{s^2}\right]$	acceleration
c	$\left[\frac{m}{s}\right]$	velocity
c_v	$\left[\frac{J}{kg \cdot K}\right]$	mass specific heat capacity for constant volume
C	$[m], [s], [kg]$	conversion factor or constant
d	$[m]$	diameter
e	$[LU]$	lattice velocity vector
Ec	$[-]$	Eckert number
f	$\left[\frac{kg \cdot s^3}{m^6}\right], [LU]$	(particle) density distribution function, “f-value”
g	$[LU]$	energy distribution function
k	$\left[\frac{W}{mK}\right]$	heat conductivity
M	$[kg]$	fluid mass
m	$[-]$	total number of lattice velocity vectors
Ma	$[-]$	Mach number
N	$[-], \left[\frac{rad}{s}\right]$	number, rotational speed
n	$[-]$	order of spatial dimension
O	$[-]$	order of magnitude
P	$[W]$	power
Pe	$[-]$	Peclet number
Pr	$[-]$	Prandtl number

Q	$[LU]$	momentum flux
$\dot{q}$	$\left[\frac{W}{m^3}\right]$	volume specific heat source
rd	$[-]$	relative error
Re	$[-]$	Reynolds number
s	$[-]$	solid fraction
T	$[K]$	temperature
t	$[s]$	time
u	$\left[\frac{m}{s}\right]$	macroscopic fluid velocity
V	$[m^3]$	volume
x	$[m]$	spatial position

Greek Symbols

α	$\left[\frac{m^2}{s}\right]$	thermal diffusivity
ε	$\left[\frac{J}{kg}\right]$	internal energy
ξ	$\left[\frac{m}{s}\right]$	microscopic particle velocity
μ	$[Pa \cdot s], [-]$	dynamic viscosity, mean value of distribution
ν	$\left[\frac{m^2}{s}\right]$	kinematic viscosity
ρ	$\left[\frac{kg}{m^3}\right]$	density
σ	$[-]$	standard deviation
τ	$[s], [Pa]$	relaxation time, shear stress
Φ	$\left[\frac{1}{s^2}\right]$	viscous dissipation term
Ω	$\left[\frac{kg}{m^3 s}\right]$	collision operator

Subscripts

*		updated value

0	initial value
2D	in two spatial dimensions
3D	in three spatial dimensions
act	actual, true value
d	dissipation
D	Dirichlet condition
f	density distribution function
g	energy distribution function
i	index of discrete LBM velocity set
l	length
m	mass
P	pellets
p	constant pressure, isobaric
R	tank reactor
rot	rotation
s	sound
SM	Smagorinsky
stirr	stirrer of the tank reactor
t	time
tot	total
T	turbulence
v	constant volume, isochoric
x	spatial cartesian x-coordinate
y	spatial cartesian y-coordinate
z	spatial cartesian z-coordinate

Superscripts/ Accents

*	physical quantity with units
^	post (after) collision
.	rate (change of quantity per time)
~	dimensionless units

‘	normalized value
B	body
eq	equilibrium
HJ	heating jacket
Os	Osmanlic (algorithm)
S	schemes

Table of Contents

Abstract III

Kurzfassung IV

Acknowledgement V

Nomenclature VI

Table of Contents XI

1 Introduction 1

1.1 Bioreactor Simulation 1

1.2 Introduction to the Lattice Boltzmann Method 2

1.3 Turbulence Model 9

1.4 Energy Equation 12

1.5 Reactor Geometries 13

2 Energy Transport Algorithm 16

2.1 Double Distribution Function Method 17

2.2 Direct Advection of Internal Energy by Density Distribution Function 29

3 Bioreactor Implementation 43

3.1 Heating Jacket 43

3.2 Tube Bundle Heat Exchanger 48

3.3 Viscous Dissipation 50

3.4 Microorganisms 52

3.5 Fluid-Phase Coupling 56

4 Summary and Outlook 61

5 List of Figures 63

6 List of Tables 68

7 Literature 69

8 Appendix i

8.1 MATLAB Source Code of Double Distribution Function Thermal LBM D2Q9: Square Cavity Problem i

8.2 Derivation: Numerical Diffusion of Osmanlic Transport Algorithm xi

1 Introduction

1.1 Bioreactor Simulation

In chemical and process engineering the use of computational fluid dynamics (CFD) tools have emerged to be a superior approach to experimental testing and optimizing in terms of financial costs, expenditure of time and often quality of results. On top CFD tools allow for an insight into spatial and time dependent physical quantities such as temperature or soluble compound concentrations that are often inaccessible via measurement in the industrial apparatus.

In the field of bioprocess engineering the aerated stirred batch bioreactor is an exemplary industrial apparatus that combines complex physical phenomena such as multiscale turbulent fluid flow, heat transfer, mass transfer and biochemical reactions. Intensive experimentation would be required to vary geometrical and operational parameters that influence these phenomena to be able to optimize them. Still, the decision to use simulation is associated with challenges: The large volume of industrial scale bioreactors ($> 10m^3$) with multiscale turbulent fluid flow caused by the rotating stirrer leads to the necessity of highly resolved computational domains (grids) which in turn requires the usage of powerful but cost intensive CPU clusters. Moreover, simultaneous simulation of moving gas bubbles and microorganisms further increase these computational costs. The alternative use of coarser grids in combination with the broadly applied Reynolds-averaged Navier-Stokes (RANS) approach cannot satisfy the need for the description of small scale anisotropic turbulence. In this conflict of computational costs and quality of the simulation results of the bioreactor the *lattice Boltzmann method* (LBM) makes a reasonable alternative to the common Navier-Stokes equations (NSE) solvers. The crucial reason is that the highly parallelizable LBM algorithm is very suitable for the large number of computational units (threads) on graphic processing units (GPUs).

The Institute of Particle and Process Engineering of the TU Graz is developing an LBM based modeling and simulation framework that is able to describe the mentioned phenomena in an aerated stirred batch bioreactor. In the current state the tool is developed via the C++ based CUDA interface and operates on *NVIDIA* GPUs. Regarding CFD the tool applies the large eddy simulation (LES) turbulence models with a sub-grid model for isotropic turbulences and covers the anisotropic turbulence of the flow field directly by simulation (see Section 1.3). Furthermore, the movement of gas bubbles or microorganisms is determined by an Euler-

Lagrange approach and the spatial and time dependent evolution of soluble compounds such as oxygen or substrate are simulated. These simulation units are complemented with the modeling of gas bubble breakage & coalescence, mass transfer models and bioreaction kinetic models.

The aim of this master's thesis is to add a simulation unit to the existing modeling and simulation framework that describes the spatial and time dependent distribution of the fluid temperature in the reactor. For this two LBM based algorithms have been implemented and investigated (Chapter 2). Basic features such as the initialization of a heating jacket or the coupling of temperature and fluid field calculation were added to the simulation code (Chapter 3).

1.2 Introduction to the Lattice Boltzmann Method

1.2.1 Boltzmann Kinetic Theory

To describe the behavior of a fluid different approaches exist. In the first, so called macroscopic approach the fluid is described as a continuum without the insight into molecular interactions. On this coarse level, macroscopic quantities such as thermal conductivity, density, viscosity or heat capacity are used to solve the equations of fluid mechanics. The microscopic approach aims to gain fluid properties by describing the mass, position and velocity of each molecule. On top the internal state by vibration or rotation and the intra molecule interactions are modeled. Since both approaches describe the same physical system there are ways to link them such as the Boltzmann kinetic theory. Without further derivation at this point, the Boltzmann kinetic theory is based on following *particle density distribution function* f for gases:

$$\rho(\boldsymbol{x},t) = \int f(\boldsymbol{x},\boldsymbol{\xi},t)\, d^3\xi$$

$$\rho \ldots macroscopic\ density\ at\ position\ \boldsymbol{x}\ and\ time\ t\ \left[\frac{kg}{m^3}\right] \qquad \text{Eq. 1-1}$$

$$\boldsymbol{\xi} \ldots microscopic\ particle\ velocity\ \left[\frac{m}{s}\right]$$

As stated in Eq. 1-1 the density can be calculated by a total integral over the velocity space and thereby considering the contribution of all possible microscopic velocities of the particles $\boldsymbol{\xi}$ at position $\boldsymbol{x}$ and time t. If the integration is weighted by the particles' velocity $\boldsymbol{\xi}$, the moment of the distribution function f will lead to the momentum density $\rho(\boldsymbol{x},t)\boldsymbol{u}(\boldsymbol{x},t)$ (Eq. 1-2). The mean velocity $\boldsymbol{u}(\boldsymbol{x},t)\left[\frac{m}{s}\right]$ at position $\boldsymbol{x}$ and time t equals the macroscopic fluid velocity.

$$\rho(\boldsymbol{x},t)\boldsymbol{u}(\boldsymbol{x},t) = \int \boldsymbol{\xi} \cdot f(\boldsymbol{x},\boldsymbol{\xi},t)\, d^3\xi \qquad \textit{Eq. 1-2}$$

Theoretically, the knowledge about the position and momentum of all particles in the fluid at a certain time should allow for the prediction of all future states. The evolution of the particle density distribution function f over time is given by the Boltzmann equation (Eq. 1-3) which is very similar to an advection equation (Eq. 1-4).

$$\frac{df}{dt} = \Omega(f)$$

$$\Omega(f) \ldots collision\ operator \qquad \textit{Eq. 1-3}$$

$$\frac{df}{dt} \ldots total\ differential\ of\ f\ in\ time$$

$$\underbrace{\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial f}{\partial \boldsymbol{x}}}_{advection} + \underbrace{\boldsymbol{a^B} \cdot \frac{\partial f}{\partial \boldsymbol{\xi}}}_{body\,force} = \underbrace{\Omega(f)}_{source\ term} \qquad \textit{Eq. 1-4}$$

$$a^B \ldots\ acceleration\ by\ body\ force \left[\frac{N}{kg}\right]$$

The collision operator $\Omega(f)$ contains the information about the outcomes of all two-particle collisions in the gas. However, this complex integro-differential term makes the Boltzmann equation unsuitable for CFD calculation [1]–[3].

1.2.2 Lattice Boltzmann Method Algorithm

In this section a brief introduction to the LBM is given that links the Boltzmann kinetic theory from previous Section 1.2.1 to the final algorithm suitable for CFD.

The BGK Approximation

In 1954 Bhatnagar, Gross and Krook (BGK) introduced a simplified model for the collision operator $\Omega(f)$ [4]. It determines the time of relaxation of the density distribution function f towards the equilibrium function f^{eq}.

$$\Omega(f) = -\frac{1}{\tau_f} \cdot (f - f^{eq})$$

Eq. 1-5

$\tau_f \ldots relaxation\ time$

$f^{eq} \ldots equilibrium\ or\ Maxwell - Boltzmann\ distribution\ function$

Inserting the BGK collision term into Eq. 1-4 and dropping the body force term leads to Eq. 1-6.

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial f}{\partial \boldsymbol{x}} = -\frac{1}{\tau_f} \cdot (f - f^{eq})$$

Eq. 1-6

In LBM the equation is discretized along specific velocity directions $\boldsymbol{\xi_i}$ (or $\boldsymbol{e_i}$ in standard LBM terminology) in the 2D or 3D simulation domain which leads the topic to the lattice arrangements.

$$\frac{\partial f_i}{\partial t} + \boldsymbol{e_i} \cdot \frac{\partial f_i}{\partial \boldsymbol{x}} = -\frac{1}{\tau_f} \cdot \left(f_i - f_i^{eq}\right)$$

Eq. 1-7

Lattice Arrangement

Similar to conventional NSE solvers the simulation domain is split into discrete cells which is called the lattice. In LBM the common terminology is to refer to the dimension n of the domain and to the number of discrete velocity directions (including one memory term) m with *DnQm*. Examples for the two and three-dimensional case respectively are D2Q9 and D3Q19 (Figure 1-1).

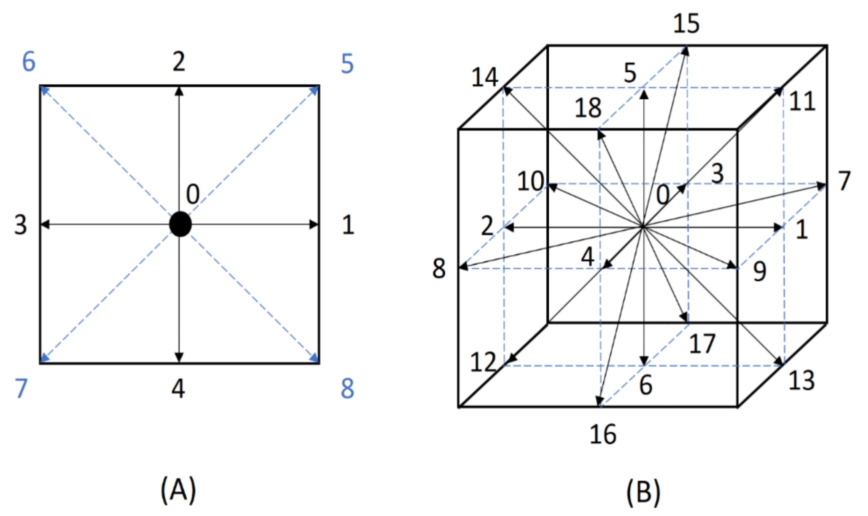


Figure 1-1: A) D2Q9 - a quadratic cell with center node and 9 discrete velocity vectors e_i
B) D3Q19 – a cubic cell with 19 discrete velocity vectors e_i either pointing to the center of a side or to the center of an edge

Table 1-1: discrete velocity set (D3Q19) – the lattice velocity vector $\boldsymbol{e_i}$ for the respective index i

i	0	1-2	3-4	5-6	7-10	11-14	15-18
$\boldsymbol{e_i}$	(0,0,0)	(±1,0,0)	(0,±1,0)	(0,0,±1)	(±1,±1,0)	(±1,0,±1)	(0,±1,±1)

So, in LBM the particle density distribution function $f_i(\boldsymbol{x}, t)$ is discretized in m directions for each lattice node at position $\boldsymbol{x}$ and time t.

Collision and Streaming

One remaining quantity to discretize is the Maxwell-Boltzmann distribution f^{eq}. The applied method is to use the Gauss-Hermite quadrature to approximate the distribution f^{eq} after a Taylor expansion of second order. This leads to:

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

$w_i \ldots weights\ of\ the\ discrete\ f_i\ for\ the\ given\ lattice$ *Eq. 1-8*

$c_s \ldots lattice\ speed\ of\ sound, c_s^2 = \frac{1}{3}$

The sum of all weights for the discrete velocity set equals unity.

Table 1-2: weights w_i for the respective index i (D3Q19)

i	0	1-6	7-18
w_i	1/3	1/18	1/36

Finally, the evolution of f-values in time stated by Eq. 1-7 can be written in a discrete LBM manner:

$$\underbrace{f_i(\boldsymbol{x}+\boldsymbol{e_i}\Delta t, t+\Delta t)}_{advected\ f_i} - f_i(\boldsymbol{x},t) = \underbrace{-\frac{\Delta t}{\tau_f}\cdot\left(f_i - f_i^{eq}\right)}_{collision\ (diffusive\ effects)} \qquad \textit{Eq. 1-9}$$

The Eq. 1-9 contains two essential steps - *collision* (1) and *streaming* (2). These distinct steps are implemented and processed separately in the simulation code (Eq. 1-10, Eq. 1-11).

1)
$$\widehat{f_i}(\boldsymbol{x},t) = f_i - \frac{\Delta t}{\tau_f}\cdot\left(f_i - f_i^{eq}\right) \qquad \textit{Eq. 1-10}$$

$\widehat{f_i}(\boldsymbol{x},t)$ *... post collision density distribution function*

2)
$$f_i(\boldsymbol{x}+\boldsymbol{e_i}\Delta t, t+\Delta t) = \widehat{f_i}(\boldsymbol{x},t) \qquad \textit{Eq. 1-11}$$

$f_i(\boldsymbol{x}+\boldsymbol{e_i}\Delta t, t+\Delta t)$

... post streaming density distribution function

In the collision step (1) the density distribution function values f_i are calculated for the current time t with the equilibrium distribution function values f_i^{eq} from Eq. 1-8. For the BGK approximation the relaxation time τ_f is defined by the kinematic viscosity of the fluid ν (Eq. 1-12). Since the τ_f is the same for every direction of the lattice velocity set the BGK-LBM is also called a single relaxation time lattice Boltzmann method (SRT-LBM).

$$\tau_f = 3\cdot\nu + \frac{1}{2} \qquad \textit{Eq. 1-12}$$

In the streaming step (2) the post collision density distribution $\widehat{f}_i(\boldsymbol{x}, t)$ values at position $\boldsymbol{x}$ are advected (moved) to the neighboring nodes along their specific lattice velocity vector $\boldsymbol{e_i}$ (see Figure 1-2). After collision and streaming boundary conditions for e.g. steady or moving solid boundaries are applied which results in one simulated time step of the LBM.

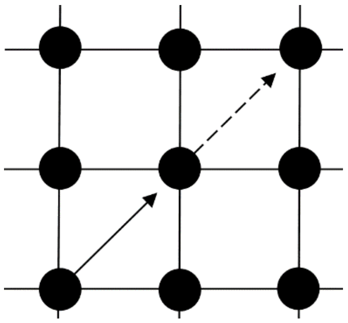


Figure 1-2: D2Q9 - Scheme of streaming step of a post collision $\widehat{f}_5(\boldsymbol{x}, t)$-value (arrow with solid line), streamed $\widehat{f}_5(\boldsymbol{x}, t)$-value equals post streaming $f_5(\boldsymbol{x} + \boldsymbol{e}_5 \Delta t, t + \Delta t)$-value (arrow with dashed line)

Calculate velocity and density

Analogous to Eq. 1-1 and Eq. 1-2 the fluid density $\rho(\boldsymbol{x}, t)$ and macroscopic velocity $\boldsymbol{u}(\boldsymbol{x}, t)$ can be calculated straightforward:

$$\rho(\boldsymbol{x}, t) = \sum_{i=0}^{m} f_i(\boldsymbol{x}, t) \qquad \textit{Eq. 1-13}$$

$$\rho(\boldsymbol{x}, t) \cdot \boldsymbol{u}(\boldsymbol{x}, t) = \sum_{i=0}^{m} \boldsymbol{e_i} \cdot f_i(\boldsymbol{x}, t) \qquad \textit{Eq. 1-14}$$

LBM code implementation

The implementation of an LBM CFD code requires following steps:

1) Initialization of the simulation domain (fluid properties, spatial domain, …)
2) Simulation via LBM for given time steps N_t
    a. Collision $(\rho(\mathbf{x}, \mathrm{t}), \mathbf{u}(\mathbf{x}, \mathrm{t}), \mathrm{f_i}(\mathbf{x}, \mathrm{t}), \tau_\mathrm{f})$
    b. Streaming $\left(\mathrm{f_i}(\mathbf{x}, \mathrm{t})\right)$
    c. Boundary conditions $\left(\mathbf{u_{BC}}(\mathbf{x_{BC}}, \mathrm{t}), \mathrm{f_i}(\mathbf{x}, \mathrm{t})\right)$

d. Calculate velocity and density $\left(\mathrm{f_i}(\mathbf{x}, \mathrm{t})\right)$

e. $\mathrm{t} = \mathrm{t} + \Delta\mathrm{t}$, repeat a. – e. if $\mathrm{t} < \mathrm{N_t}$

3) Output and visualization

Between step 1) and 2) a lattice unit conversion is carried out that converts the physical units of quantities e.g. $\nu\left[\frac{m^2}{s}\right]$ to non-dimensional lattice units to be used for example in Eq. 1-12.

1.2.3 Lattice Unit Conversion

Before the computation of the flow field by the LBM the units of physical quantities that define the system must be converted to so called lattice units LU. The reason is that the numerical solving of the LBM equation system is processed with discrete time and space steps. For simplicity, the arbitrarily chosen value of the steps is unity and in turn the conversion factors C of time C_t $[s]$, space C_l $[m]$ and mass C_m $[kg]$ are introduced to fulfill the chosen unity of time and space steps. One restriction to the unit conversion is that according to the law of similarity the dimensionless characteristic values that define the solution to the problem must stay the same. Using the dimensionless form of the NSE without body forces one can see that the transient solution of the flow field $\boldsymbol{u}(\boldsymbol{x}, t)$ solely depends on the Reynolds number $Re = \frac{u d_R}{\nu}$ and the geometric ratio/ratios. Concerning the solution to the internal energy equation without viscous dissipation the Prandtl number $Pr = \frac{\nu}{\alpha}$ must be added to the set of characteristic numbers. Moreover, the stability of LBM constrains the Mach number Ma to be $Ma < 0.35$ [1]. The accuracy requirement of hydrodynamic simulations often force the Mach number to be $Ma < 0.05$ to avoid compressibility effects that scale with Ma^2 [1] [5]. In the following, one procedure for LB unit conversion is carried out. Note that all quantities with physical units are signed with *.

$$C_l = \frac{d^*}{N_l} \qquad \begin{array}{l} d \ldots characteristic\ length\ e.g.\ diameter\ [m] \\ N_l \ldots number\ of\ nodes\ for\ length\ d\ [-] \end{array} \qquad \textit{Eq. 1-15}$$

Typically, a certain Mach number Ma or a certain relaxation time τ_f is predefined to calculate a conversion factor of time. Due to *diffusive scaling* it is reasonable to obtain $C_t \cong C_l^2$.

$$u = c_s \cdot Ma \qquad u \ldots\ characteristic\ velocity\ [LU] \qquad \textit{Eq. 1-16}$$

$$u^* = u \cdot \frac{C_l}{C_t} \qquad \textit{Eq. 1-17}$$

The conversion factor of mass C_m is obtained by choosing the value to the lattice density ρ arbitrarily, which is typically unity for single-phase flow and not unity for the continuous phase in multi-phase flow simulations.

$$\rho^* = \rho \cdot \frac{C_m}{C_l^3} \qquad \textit{Eq. 1-18}$$

With the given conversion factors of time C_t, length C_l and mass C_m further mechanical quantities such as kinematic viscosity $\nu^* \left[\frac{m^2}{s}\right]$, pressure $p^*[Pa]$ or internal energy $\varepsilon^* \left[\frac{J}{kg}\right]$ can be converted to lattice units (Eq. 1-19, Eq. 1-20). The value of the relaxation time $\tau_f(\nu)$ is dependent on the physical value of the kinematic viscosity $\nu^*$and the conversion factor of time and space (Eq. 1-20). So, the stability of SRT-LBM affected by τ is strongly coupled to the choice of numerical parameters such as node number N_l and Mach number Ma as well as system parameters such as the Reynolds number $Re = \frac{d^*u^*}{\nu^*}$.

$$\varepsilon = \varepsilon^* \cdot \frac{C_t^2}{C_x^2} \qquad \textit{Eq. 1-19}$$

$$\tau = \frac{1}{3} \cdot \left(\nu^* \cdot \frac{C_t}{C_l^2} - 0.5\right) \qquad \textit{Eq. 1-20}$$

Since the discrete time step in lattice units equals unity ($\Delta t = 1$), the total simulated time equals:

$$t_{sim} = N_t \cdot C_t \qquad N_t \, ... \, total\ number\ of\ timesteps\ [-] \qquad \textit{Eq. 1-21}$$

1.3 Turbulence Model

Since the NSE can be recovered using the *Chapman-Enskog theory*, it is proven that the physics of hydrodynamic phenomena can be described mechanistically by the LBM. As shown in Section 1.2.2, the velocity (flow) field $\boldsymbol{u}(\boldsymbol{x},t)$ can be obtained for the given lattice with the discrete three-dimensional position vector $\boldsymbol{x} = (x, y, z)$. If the lattice is quasicontinuous ($C_l \rightarrow$

0), the temporal and spatial dependencies of the flow field can be described exactly. In the field of CFD this method is referred to as *direct numerical simulation* (DNS). As this method cannot be applied at scales of industrial bioreactors due to immense computational costs a coarser grid is applied and a sub-grid model is used for the detailed smaller structural elements of the flow field that are not directly simulated.

To distinguish between the simulated and modelled elements in flow fields the concept of anisotropic and isotropic turbulence is used (Figure 1-3).

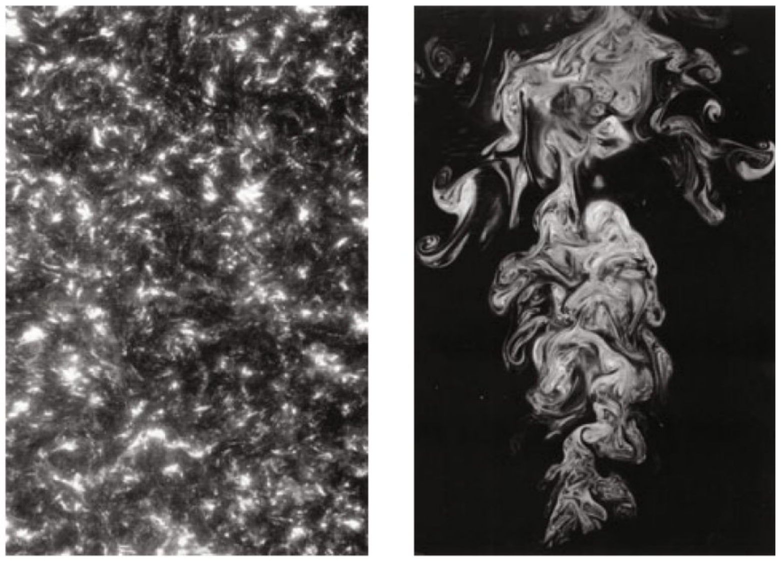


Figure 1-3: left) isotropic turbulence – the depicted flow field is independent from any direction or macroscopic geometries, right) anisotropic turbulence – the depicted flow field shows large-scale vortexes (eddies) which are dependent on the upward direction and are influenced by the macroscopic oblong geometry [6]

The basic idea of the *large eddy simulation* (LES) turbulence model is that the anisotropic turbulence (large eddies) are simulated while the isotropic small-scale eddies are modelled. The premise of LES is that the lattice (grid) spacing and time step in the simulation must be small enough to cover the large eddies and their transient behavior over time. The small-scale eddies do have a dissipative characteristic and only marginally influence the anisotropic flow field which justifies the approach to model all sub-grid isotropic turbulences [4] [5].

The *Smagorinsky model* [8] is the most prominent approach in LES because of its simplicity and numerical stability. This method is used in the current simulation framework [9] and its main goal is to model the dissipation energy to be withdrawn from the kinetic energy of the large-scale fluid field by increasing the physical fluid viscosity $\nu_0(\tau_{f,0})$ by a turbulent (eddy) viscosity ν_T. The calculation of this eddy viscosity is based on the momentum flux scalar Q_{tot}:

$$\boldsymbol{Q} = \sum_{i=1}^{m} \left(\boldsymbol{e_i} \otimes \boldsymbol{e_i} \cdot (f_i - f_i^{eq}) \right) \qquad \textit{Eq. 1-22}$$

$\boldsymbol{Q} \ldots 3x3\ tensor\ for\ three\ dimensions\ (n = 3)$

$$Q_{tot} = \sqrt{2 \cdot \sum_{j=1}^{n} \sum_{k=1}^{n} Q_{jk} Q_{jk}} \qquad \textit{Eq. 1-23}$$

$$\nu_T = \frac{1}{6}\left(\sqrt{\tau_{f,0}^2 + 2\sqrt{2}(C_{SM})^2 \cdot \frac{9Q_{tot}}{\rho}} - \tau_{f,0}\right) \text{ with}$$

$$\tau_{f,0} = 3 \cdot \nu_0 + \frac{1}{2} \qquad \textit{Eq. 1-24}$$

$$\tau_{f*} = 3 \cdot (\nu_0 + \nu_T) + \frac{1}{2} \qquad \textit{Eq. 1-25}$$

The Smagorinsky constant C_{SM} is typically set to 0.1 as proposed in [10]. The updated relaxation time τ_{f*} for each respective node in the lattice is used in the collision step.

The energy dissipation rate $\dot{\varepsilon}_d$ can be calculated by the strain rate tensor $\boldsymbol{S}$ [10]:

$$\boldsymbol{S} = -\frac{1}{2\,\rho_0\,\tau_{f*}\,c_s^2} \cdot \boldsymbol{Q} \qquad \textit{Eq. 1-26}$$

$$\dot{\varepsilon}_d = 2\,(\nu_0 + \nu_T) \cdot \sum_{j=1}^{n} \sum_{k=1}^{n} S_{jk} S_{jk} \qquad \textit{Eq. 1-27}$$

Using the definition of $\boldsymbol{S}$ from Eq. 1-26 in Eq. 1-27 leads to:

$$\dot{\varepsilon}_d = \frac{9\,(\nu_0 + \nu_T)}{2\,\rho_0^2\,\tau_{f*}^2} \cdot \sum_{j=1}^{n} \sum_{k=1}^{n} Q_{jk} Q_{jk} \qquad \textit{Eq. 1-28}$$

1.4 Energy Equation

The internal energy equation for a compressible homogeneous medium can be derived by an energy balance of a moving, infinitesimal small, fluid volume element. This evolution equation of the scalar internal energy $\varepsilon = \int_0^T c_v dT$ is an advection-diffusion equation ADE and contains following terms [6] [11]:

$$\begin{bmatrix} Rate\ of\ change\ of \\ energy\ inside\ the \\ volume\ element \end{bmatrix} = \begin{bmatrix} Net\ flux\ of \\ heat\ into \\ the\ element \end{bmatrix} + \begin{bmatrix} Rate\ of\ work\ applied\ on \\ the\ element\ due\ to\ body \\ and\ surface\ forces \end{bmatrix} \qquad Eq.\ 1\text{-}29$$

$$A = B + C$$

Similar to the continuity equation the term A can be expressed mathematically as:

$$A = \frac{D(\rho\,\varepsilon)}{Dt} = \frac{\partial(\rho\,\varepsilon)}{\partial t} + \boldsymbol{\nabla}\cdot(\rho\,\varepsilon\,\boldsymbol{u}) \qquad Eq.\ 1\text{-}30$$

The term B covers the heat flux due to thermal conduction and volumetric heating such as absorption of radiation or heat of chemical reaction:

$$B = \dot{q} + \boldsymbol{\nabla}\cdot(k\,\boldsymbol{\nabla}\,T)$$

$\dot{q}$ ... heat source $\left[\frac{W}{m^3}\right]$ *Eq. 1-31*

k ... heat conductivity $\left[\frac{W}{mK}\right]$

The term C includes the compression and dissipation term. Note that body forces are not considered because they do not influence the internal energy of the volumetric element.

$$C = -p\,(\boldsymbol{\nabla}\cdot\boldsymbol{u}) + \rho\,\nu\,\Phi$$

Φ ... dissipation term $\left[\frac{1}{s^2}\right]$ *Eq. 1-32*

$$C = \underbrace{-p\,(\boldsymbol{\nabla}\cdot\boldsymbol{u})}_{=0\ for\ \rho=const} + \rho\,\dot{\varepsilon}_d \qquad Eq.\ 1\text{-}33$$

Firstly, the viscous dissipation $\dot{\varepsilon}_d = \nu\,\Phi$ has to be modelled as explained previously in Section Turbulence Model1.3 (see Eq. 1-28) and can be put together with all other sources $\dot{q}$. Secondly, the compression term $-p\,(\boldsymbol{\nabla}\cdot\boldsymbol{u})$ can be neglected for incompressible flows. Inserting the three terms A, B and C into Eq. 1-29 leads to:

$$\frac{\partial(\rho\,\varepsilon)}{\partial t} + \underbrace{\boldsymbol{\nabla}\cdot(\rho\,\varepsilon\,\boldsymbol{u})}_{advection} = \underbrace{\boldsymbol{\nabla}\cdot(k\,\boldsymbol{\nabla}\,T)}_{diffusion} + \underbrace{\rho\,\dot{\varepsilon}_d + \dot{q}}_{source} \quad \text{with}$$
$$\varepsilon = \int_0^T c_v(T)\,dT \xrightarrow{c_v = const} \varepsilon = c_v\cdot T \qquad \text{Eq. 1-34}$$

The final equation (Eq. 1-34) is very similar to the NSE and has following dimensionless form:

$$\frac{\partial(\tilde{\rho}\,\tilde{T})}{\partial\tilde{t}} + \tilde{\boldsymbol{\nabla}}\cdot(\tilde{\rho}\,\tilde{T}\,\tilde{\boldsymbol{u}}) = \frac{1}{Pr\cdot Re}\,\tilde{\boldsymbol{\nabla}}\cdot(\tilde{\boldsymbol{\nabla}}\,\tilde{\boldsymbol{T}}) + \frac{Ec}{Re}\,\tilde{\Phi} + \tilde{\dot{q}} \qquad \text{Eq. 1-35}$$

The dimensionless form shows that with high Peclet numbers $Pe = Re\cdot Pr$ the advective effects outweigh diffusive (conductive) effects in the thermal fluid flow. Moreover, with low Eckert $Ec = \frac{u^2}{c_v\cdot\Delta T}$ and high Reynolds number Re the advection outweighs the viscous dissipation.

1.5 Reactor Geometries

As mentioned in Section 1.2.3 the Reynolds number and geometric ratios are the only parameters that influence the solution of the single-phase flow field. Except for the kinematic viscosity all other information for the solution must be given by the reactor geometries that define the moving or steady solid parts of the computational domain. Figure 1-4 depicts an example of a stirred tank reactor whose parts are described in the following.

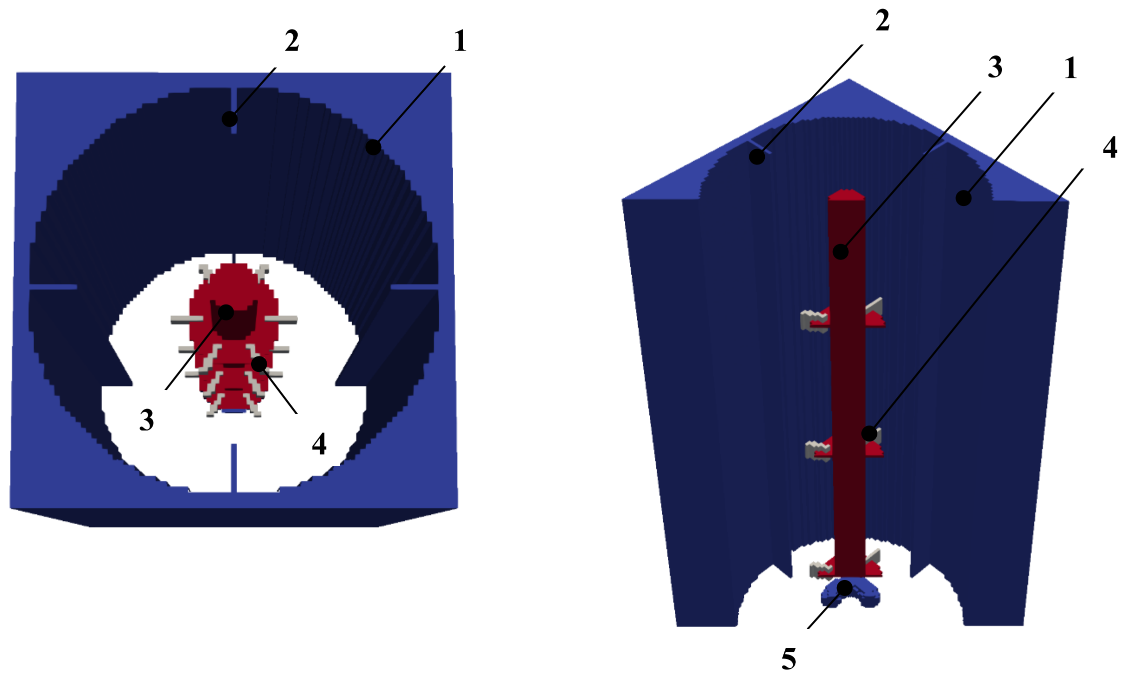


Figure 1-4: left) top view of the stirred tank reactor with Rushton stirrer, right) isometric front view, blue) steady solid parts of reactor, red) steady solid parts of stirrer, grey) moving solid parts of stirrer 1) wall of the reactor, 2) baffles, 3) shaft of the stirrer with rotating disks, 4) blades of the stirrer, 5) gas sparger

1) *Wall of the reactor*

A cylinder with certain diameter and height/ volume defines the reactor wall. The staircase approximation is used for the curved surface.

2) *Baffles*

Typically, four upright rectangular baffles are installed in the reactor with equal spacings. The purpose is to partially convert the tangential flow produced by the rotation of the stirrer blades into an axial flow.

3) *Stirrer Shaft*

The cylindric stirrer shaft simply transfers the torque of the rotating motor to the stirrer blades.

4) *Blades*

Figure 1-4 shows in total 3 Rushton stirrers with 6 blades each. Currently the simulation framework supports different blade shapes such as the depicted Rushton, propeller or elephant ear blades. Moreover, the number of the blades can vary both by the number on each level as well as by the number of stirrers on the shaft. The axial distance of the stirrers on the shaft can be varied too.

5) Gas Sparger

The torus-shaped gas sparger at the bottom of the reactor flushes and disperses a gas phase such as air or pure oxygen into the reactor. This is necessary to provide the required oxygen for the metabolism of the microorganisms.

2 Energy Transport Algorithm

In the field of thermal lattice Boltzmann methods (TLBM) three methods to simulate the evolution of the internal energy or temperature distribution in fluids by solving the ADE are common [1][12][13].

In the *multi-speed* approach, the very same density distribution function f is used to solve both the NSE and the ADE of the internal energy. This is physically valid since the Boltzmann equation can as well describe the macroscopic energy in a system by moments of the density distribution. However, there are several drawbacks of this approach such as the requirement of a large set of discrete velocities (e.g. D3Q40) which requires a significant change to the current fluid flow calculation and large computational storage costs. Moreover, the method suffers from stability issues and require a difficult implementation of boundary conditions for the large set of discrete density distribution values. Larger temperature differences and an adaptable Prandtl number Pr are not easily achievable.

The second more popular *hybrid* method solves the NSE by LBM and the internal energy ADE by direct finite volume discretization of the ADE. On the one hand the method is more stable than the multi-speed approach, but on the other hand the calculation is less parallelizable especially on GPUs. One major advantage to the pure LB approach is that the pressure can be coupled with the temperature explicitly which is not vital in liquid flows though.

The third so called *double distribution function* DDF method uses a separate second distribution function g that is used to calculate the temperature, internal or total energy distribution in the system. Advantages of the approach are that the Prandtl number can be easily adapted due to the explicit and decoupled algorithm and the better stability compared to the multi-speed approach. Moreover, the algorithm can be implemented and verified more independently to the existing implemented and already tested parts of the simulation framework. In the literature of the recent 10 years most TLBM applications and development are based on DDF-TLBM [14]–[17]. As the LBM for the flow field the DDF-TLBM algorithm can be parallelized the same way. Due to the given advantages and shortcomings of each respective method the DDF was chosen to be implemented as an energy transport algorithm to describe the temperature distribution in a stirred tank reactor. Further explanation, showcase of implementation and verification cases are given in following Section 2.1 and Appendix 8.1.

2.1 Double Distribution Function Method

The DDF approach is based on the work of He et al. 1998 [18] who introduced a coupled model that takes the compression and dissipation energy into account by making them a function of spatial velocity derivatives. This term however restricts the simple *bounce-back boundary condition* [19] implementation and is known to cause numerical instabilities [12]. An often used adaption of this first approach is the simplification by Peng et al. 2003/2004 [19] [20] who neglected the compression/dissipation term which often marginally influences the flow and temperature field. A dissipation energy can be added via a source term in the collision step though (see Eq. 2-2). The method calculates the temperature distribution in the domain by simulating the evolution of the internal energy IE by its distribution g. The following scheme shall clarify the simulation procedure including the lattice unit conversion (see Section 1.2.3) where quantities with physical units are signed with superscript *.

$$\boxed{T^*_{init} \xrightarrow{c_v(T)} \varepsilon^*_{init} \xrightarrow{C_l, C_t} \varepsilon_{init} \xrightarrow{TLBM\ simulation} \varepsilon_{out} \xrightarrow{C_l, C_t} \varepsilon^*_{out} \xrightarrow{c_v(T)} T^*_{out}}$$

Collision and Streaming

The SRT-DDF-IE-TLBM by Peng et al. [20] follows an analogous sequence as the SRT-LBM approach with the BGK collision operator for the flow field (see Section 1.2.2). The following formulas are based on the D3Q19 lattice arrangement.

$$g_0^{eq} = \rho\, \varepsilon\, w_0 \cdot \left(-\frac{3\,\boldsymbol{u}^2}{2}\right) \qquad for\ i = 0$$

$$g_i^{eq} = \rho\, \varepsilon\, w_i \cdot \left(1 + \frac{\boldsymbol{e_i u}}{c^2} + \frac{9(\boldsymbol{e_i u})^2}{2c^2} - \frac{3\boldsymbol{u}^2}{2c^2}\right) \qquad for\ i = 1, 2, \dots 6 \qquad Eq.\ 2\text{-}1$$

$$g_i^{eq} = \rho\, \varepsilon\, w_i \cdot \left(2 + \frac{4\,\boldsymbol{e_i u}}{c^2} + \frac{9(\boldsymbol{e_i u})^2}{2c^4} - \frac{\boldsymbol{3u}^2}{2c^2}\right) \qquad for\ i = 7, 8, \dots 18$$

1) $$\widehat{g_i}(\boldsymbol{x},t) = g_i - \frac{\Delta t}{\tau_g} \cdot \left(g_i - g_i^{eq}\right) + w_i \cdot \underbrace{\rho\dot{\varepsilon}_d + \dot{q}}_{source} \cdot \Delta t$$ *Eq. 2-2*

$\widehat{g_i}(\boldsymbol{x},t)$ *... post collision energy distribution function*

2) $$g_i(\boldsymbol{x} + \boldsymbol{e_i}\Delta t, t + \Delta t) = \widehat{g_i}(\boldsymbol{x},t)$$ *Eq. 2-3*

$g_i(\boldsymbol{x} + \boldsymbol{e_i}\Delta t, t + \Delta t)$

... post streaming energy distribution function

The thermal relaxation time τ_g is linked to the thermal diffusivity $\alpha = \frac{k}{\rho c_v}$ of the fluid:

$$\tau_{g,0} = \frac{9}{5}\alpha_0 + \frac{1}{2}$$ *Eq. 2-4*

The Prandtl number is $Pr = \frac{\nu_0}{\alpha_0} = \frac{3}{5} \cdot \frac{(\tau_{f,0} - 0.5)}{(\tau_{g,0} - 0.5)}$. The effective thermal relaxation time $\tau_{g*} = \frac{9}{5}(\alpha_0 + \alpha_t) + 0.5$ can be made a function of the turbulent Prandtl number $Pr_t = \frac{\nu_t}{\alpha_t}$, the effective relaxation time τ_{f*} and the initial relaxation time $\tau_{f,0}$ [21]:

$$\tau_{g*} = \frac{3}{5} \cdot \left[\frac{1}{Pr}\left(\tau_{f,0} - 0.5\right) + \frac{1}{Pr_t}\left(\tau_{f*} - \tau_{f,0}\right)\right] + 0.5$$ *Eq. 2-5*

Calculate internal energy

Analogous to the density (mass per volume) calculation by f-values, the internal energy times the density (energy per volume) can be calculated as the sum of all distribution values g_i for the given node at position $\boldsymbol{x}$.

$$\rho(\boldsymbol{x},t)\varepsilon(\boldsymbol{x},t) = \sum_{i=0}^{18} g_i(\boldsymbol{x},t)$$ *Eq. 2-6*

TLBM code implementation

In DDF-TLBM the calculation of the thermal field $\varepsilon(\boldsymbol{x},t)$ is carried out after the flow field $\boldsymbol{u}(\boldsymbol{x},t)$ in the same explicit manner (see Section 1.2.2.)

1) Initialization of the simulation domain (fluid properties, spatial domain, …)
2) Simulation via TLBM for given time steps N_t
   a. Flow field: Collision, Streaming, Boundary condition, Calculate velocity and density
   b. Thermal Collision
   c. Thermal Streaming
   d. Thermal Boundary condition
   e. Calculate internal energy
   f. $\mathrm{t = t + \Delta t,\ \ repeat\ a. - f.\ \ if\ t < N_t}$
3) Output and visualization

2.1.1 Boundary Handling

Since the DDF-TLBM approach is based on a separate probability function g the algorithm requires a separate handling of (thermal) boundary conditions (BC). In LBM generally, after the streaming step g_i or f_i-values of fluid nodes next to a solid node (e.g. reactor wall) are missing as no g-values are streamed from solid nodes to the fluid boundary nodes in the streaming step. Since a complete set of g-values for each node in the fluid domain is required in the collision step of the next timestep, a boundary condition is used to calculate the missing g-values after the streaming step (see Section 2.1 'TLBM code implementation ').

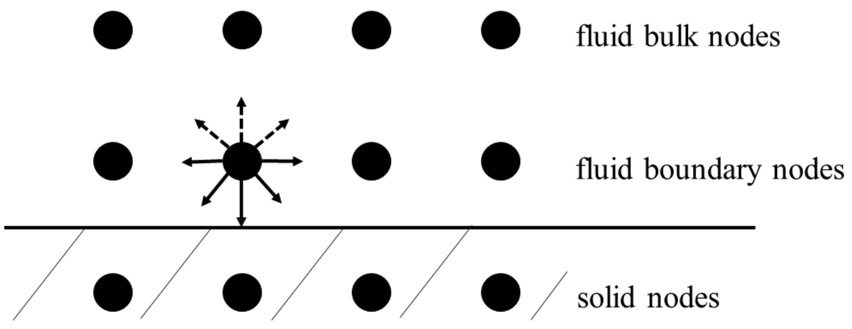


Figure 2-1: Example of g-distribution set after streaming step, the 'fluid boundary nodes' are missing g-values (dashed arrows), all streamed values (solid arrows) were streamed from neighboring fluid nodes, missing g-values must be calculated by boundary condition

In the field of TLBM three common methods have emerged to handle boundary conditions:

1) Bounce back alike BC (introduced in [22])
2) Simple weighted splitting method (studied in e.g. [16])
3) Splitting method with taking streamed values into account (introduced in e.g. [23])

The first approach is based on a calculation of missing g-values by post collision g-values of the local node itself and neighboring nodes plus the value of the *Dirichlet* condition ε_D. In the simple second approach all g-values of boundary nodes are set according to their weighting factor by $g_i = w_i \cdot \rho\varepsilon_D$. In the similar third method the amount that is left to fulfill the *Dirichlet* condition is determined and distributed to the missing values according to their weighting factor. The already streamed g-values are retained. The basic formula is given in Eq. 2-7. Different 'schemes' are given in literature [23] which uses the base value g_i^S for the calculation of the missing g-values $g_{i,missing}$.

$$g_{i,missing} = g_{i,missing}^S + \frac{w_{i,missing}}{\sum_{i=1}^m w_{i,missing}} \cdot \left(\rho\varepsilon_D - \sum_{i=1}^{m}(g_{i,streamed} + g_{i,missing}^S) - g_0\right)$$

$g_i, g_0 \ldots post\ streaming\ values$ *Eq. 2-7*

$Scheme\ C: g_{i,missing}^S = g_i^{eq}$

$Scheme\ D: g_{i,missing}^S = 0$

Neumann boundary conditions can be easily converted to *Dirichlet* conditions by finite difference method. For example, if the gradient is known at the beginning of a domain:

$$\varepsilon_D(x=0) = \frac{2}{3}\frac{\partial\varepsilon(x=0)}{\partial x} + \frac{4}{3}\,\varepsilon(x=1) - \frac{1}{3}\varepsilon(x=2)$$

The third method was implemented in the simulation framework since it is a simple local node method but still remains more information of the thermal field compared to method 2).

2.1.2 Implementation in MATLAB

The LBM algorithm for fluid flow and temperature distribution explained in Section 1.2.2 and 2.1 respectively, have been implemented via the *MATLAB* computing environment. The DDF method for the temperature distribution and the LBM algorithm for fluid flow were tested using a D2Q9 lattice arrangement by the thermal Couette and lid driven cavity flow experiment. The MATLAB source code for the lid driven cavity simulation is given in the Appendix 8.1. For the boundary conditions of the fluid flow field the common *Halfway Bounce Back* method, *Zhou & He* method and periodic boundary condition were used. These methods can be looked up in detail in section 4.4 'Boundary Conditions' of [24].

2.1.2.1 Thermal Couette Flow

The incompressible Couette flow equals a flow of a fluid that is embedded between two walls. The top wall moves with a constant velocity that sets in at the beginning of the experiment. Additionally, in the thermal Couette flow experiment the two walls are set to constant temperature. The top wall is hotter than the bottom wall.

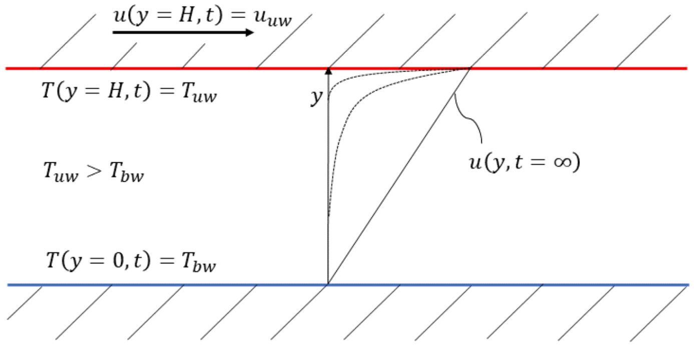


Figure 2-2: Scheme of the thermal Couette flow experiment, the hot top wall is constantly moving, the temperature difference between hot top wall and cool bottom wall is constant over time

The simulation code can be tested for different Prandtl numbers Pr: If the Prandtl number equals $Pr = 1$, the normalized solution of the flow field and the temperature distribution shall coincide. In addition to that the solution can be compared to the transient analytic solution and to the steady state analytic solution for $t \rightarrow \infty$ (Eq. 2-8 and Eq. 2-9). Note that the given transient solution in Eq. 2-8 does not hold the no slip condition at the bottom wall and deviates from the correct solution when $\left.\frac{du(y,t)}{dy}\right|_{y=0} > 0$.

transient solution:

$$\frac{u(y,t)}{u_{uw}} = \left(1 - \operatorname{erf}\left(\frac{H-y}{2\sqrt{\nu \cdot t}}\right)\right) \; or \qquad \textit{Eq. 2-8}$$

$$\frac{T(y,t) - T_{bw}}{T_{uw} - T_{bw}} = \left(1 - \operatorname{erf}\left(\frac{H-y}{2\sqrt{\alpha \cdot t}}\right)\right)$$

steady state solution $t \rightarrow \infty$:

$$\frac{u(y,t)}{u_{uw}} = \frac{y}{H} \; or \qquad \textit{Eq. 2-9}$$

$$\frac{T(y,t) - T_{bw}}{T_{uw} - T_{bw}} = \frac{y}{H}$$

T_{uw} … *temperature of the top (upper) wall*

T_{bw} … *temperature of the bottom wall*

H … *distance of the both plates (walls)*

The performance of the simulation of the thermal Couette flow is depicted in Figure 2-3 and Figure 2-4 and compared to the transient and steady state solution respectively.

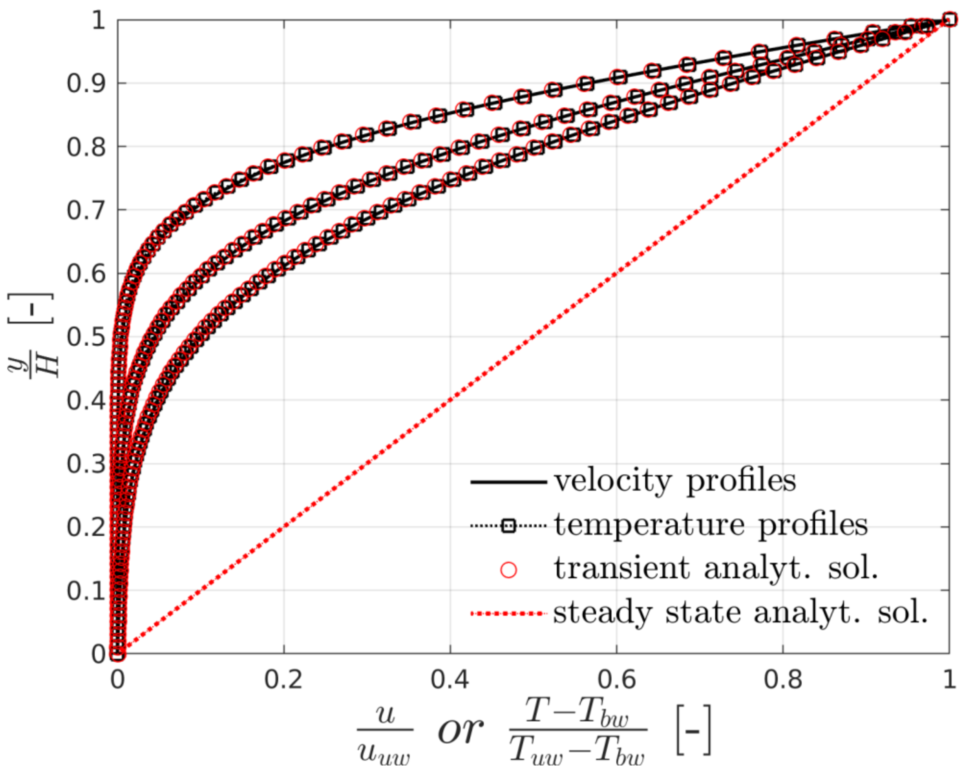


Figure 2-3: Thermal Couette flow field and temperature distribution for Re=1, Pr=1, nodes in y-coordinate: 100, temperature and velocity profiles after t=0.15s, t=0.30s and t=0.45s, comparison to transient solution of the thermal Couette flow

The simulation slightly overpredicts the viscous effects and thereby leads to a positive deviation to the analytic solution. The simulation results are in a good agreement with the transient analytic solution though. The velocity and temperature profiles coincide for the Prandtl number of $Pr = 1$ and verify the correct implementation of the temperature distribution and flow field calculation.

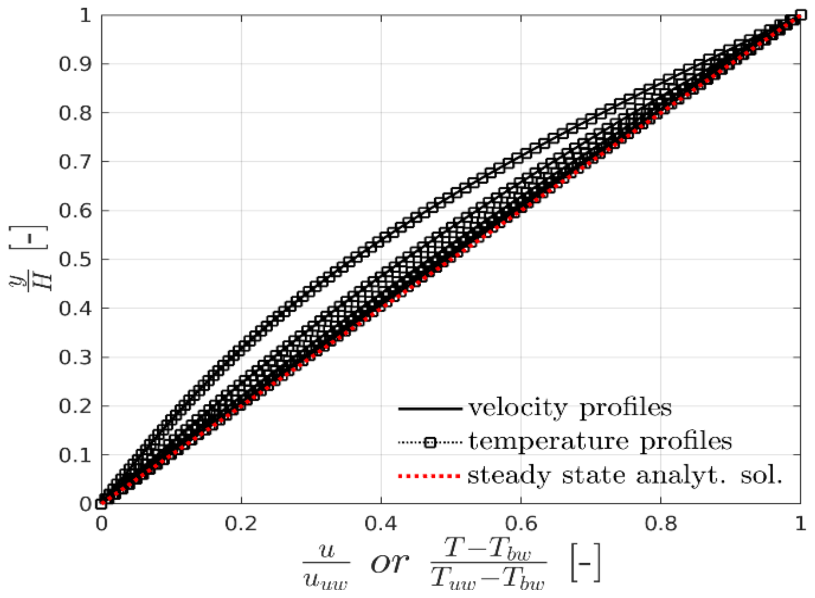


Figure 2-4: Thermal Couette flow field and temperature distribution for Re=1, Pr=1, nodes in y-coordinate: 100, temperature and velocity profiles after t=1.5s, t=2.25s, t=3.0s, t=3.75s and t=4.5s, comparison to steady state solution of the thermal Couette flow

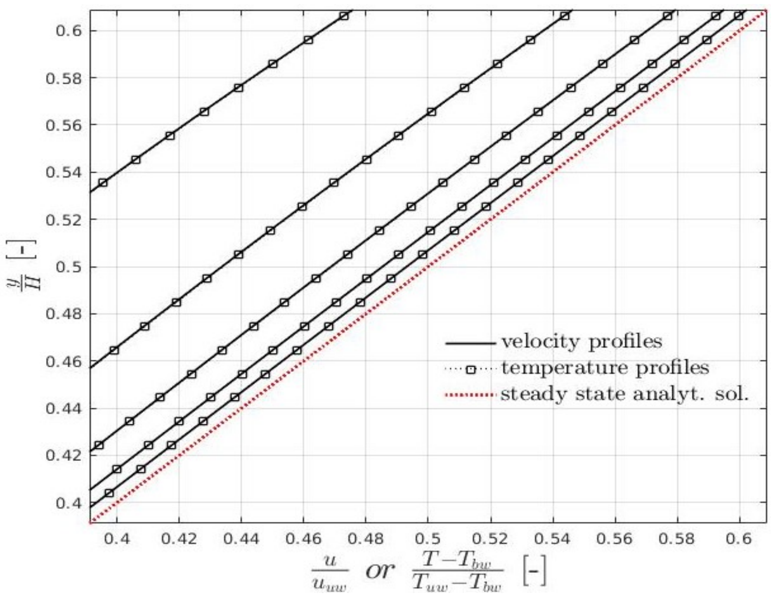


Figure 2-5: Square scope from 0.4 to 0.6 of Figure 2-4

The temperature and velocity profiles converge to the expected steady state analytic solution for longer simulated time.

2.1.2.2 Lid Driven Cavity Flow

Although the thermal Couette flow experiment was simulated in a two dimensional domain with periodic boundary conditions for the left and right domain boundaries respectively, the physical problem is one dimensional. For this reason, the common lid driven cavity flow experiment was chosen as a 2D benchmark problem for the simulation code. The lid driven cavity flow experiment equals a fluid embedded in a solid square. The top wall is moving with a constant velocity. The movement is started at the beginning of the experiment (Figure 2-6). The solution of the simulation can be verified by benchmark data given in the literature [25].

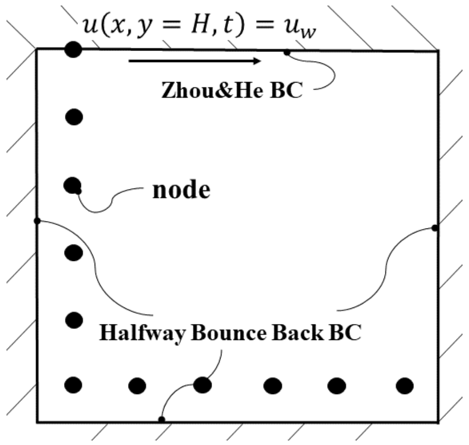


Figure 2-6: Scheme of the lid driven cavity experiment, the top wall (wall) is constantly moving, the three remaining walls are stationary

There are two common approaches to verify the correctness of the simulation:

1) Vector plot of the 2D flow field with vortex center $\boldsymbol{u}(x_{vc}, y_{vc}) = \boldsymbol{0}$ (see Figure 2-7)
2) Velocity profile of x-component velocity at the horizontal middle of the cavity $u\left(x = \frac{1}{2}H, y\right)$ (see Figure 2-8)

The applied input and simulation parameters are given in Table 2-1

Table 2-1: Input and simulation parameter of the lid driven cavity flow simulation

Physical Quantity	Test Case	Physical Quantity	Case
velocity of lid u_w	$0.01 \frac{m}{s}$	number of nodes	$41 \; x \; 41 = 1681$
side length H	$0.01 \; m$	relaxation time τ	0.623
kin. viscosity ν	$1 \cdot 10^{-4} \frac{m^2}{s}$	time step Δt^*	$2.439 \cdot 10^{-5} s$
Reynolds number Re	1	total simulated time t_{sim}	0.2439 s
density ρ	$1000 \; \frac{kg}{m^3}$		

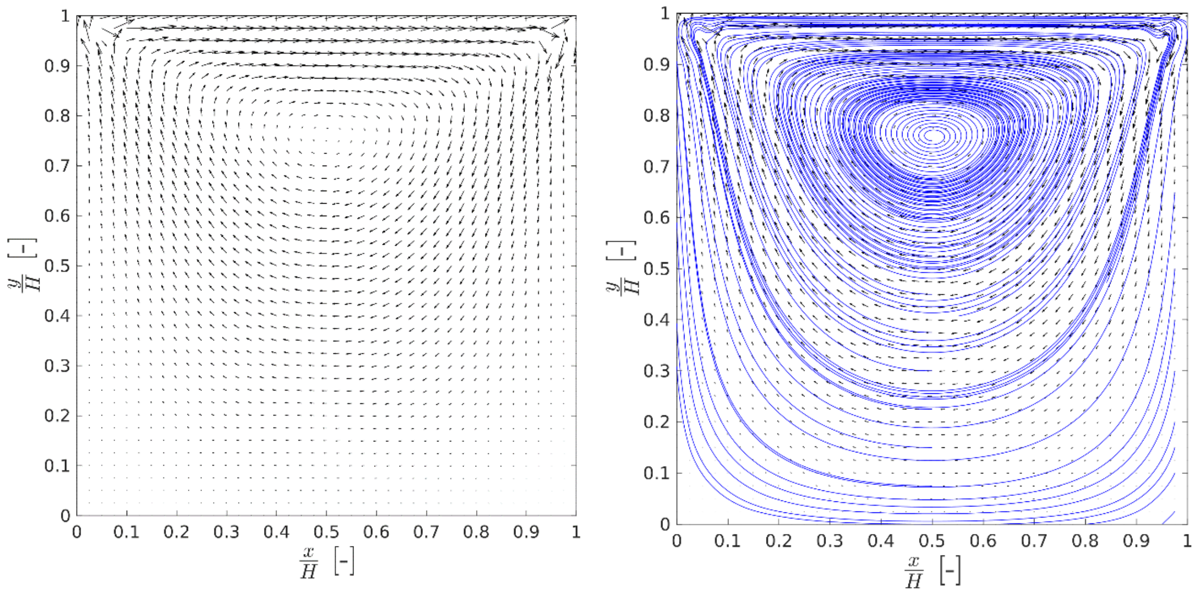


Figure 2-7: Vector plot of the lid driven cavity flow problem (left) and streamline plot (right), the vortex center location $(x_{vc}, y_{vc}) = (0.5, 0.75)$ *agrees with the collection of literate data for Re* ≤ 1 [25]*, numerical artifacts remain at the upper corner where bounce back and Zhou & He BC are adjacent*

The center location of the vortex agrees with the data given in the literature [25]. The solution of the flow field contains artifacts at the upper corner of the cavity. The reason might be the interaction of Bounce Back BC and Zhou & He BC at these corners.

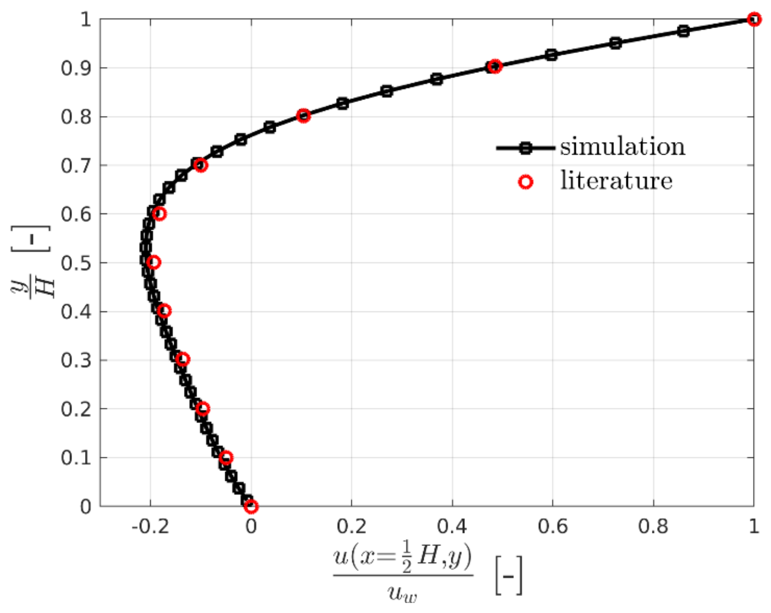


Figure 2-8: X-component velocity u plotted along the height of the cavity at the horizontal middle of the cavity

The steady state velocity profile at the horizontal middle of the cavity is in good agreement with the literature data [25].

2.1.3 Implementation in CUDA

The SRT-DDF-IE-TLBM was ported to the C++ based CUDA API simulation framework and in terms of LB adapted to the D3Q19 lattice arrangement. The coded energy transport algorithm shall be able to correctly simulate the transient advection and diffusion of energy with given local and transient energy sources.

2.1.3.1 Verification: Heat Diffusion in a Cylinder with Steady Fluid

For the verification of the method a heat diffusion in a cylinder with constant hot wall temperature and resting water is simulated. This case was chosen to independently check for stability issues that may arise by the value of the relaxation time τ_g that is linked to the thermal diffusivity α of the fluid (see Eq. 2-4). Note that the viscosity ν of the water is set to $\nu = 1 \cdot 10^{-6} \frac{m^2}{s}$ and the true physical Prandtl number is about $Pr = \frac{\nu}{\alpha} \sim 5$. The following overview of the simulation setup (Figure 2-9), simulation parameters and graphs (Figure 2-10 - Figure 2-13) illustrate the stability limit of the method.

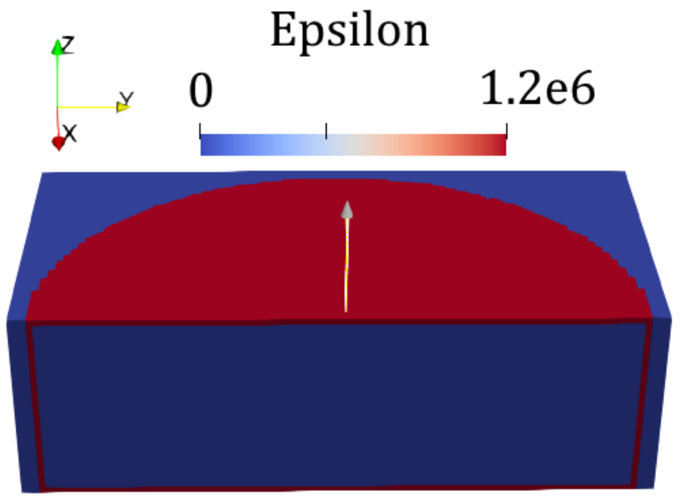


Figure 2-9: Initial setup of the verification case, the boundary nodes of the cylinder are set to a constant temperature of 300K ($\varepsilon = 1.2 \cdot 10^6 \frac{J}{kg}$, $c_p = 4000 \frac{J}{kg \cdot K}$),the bulk nodes are set to a temperature of 0 K, the heat diffusion is investigated along a plot axis (white arrow in z-coordinate), diameter: 1 m, height: 0.332 m, 100 nodes per meter

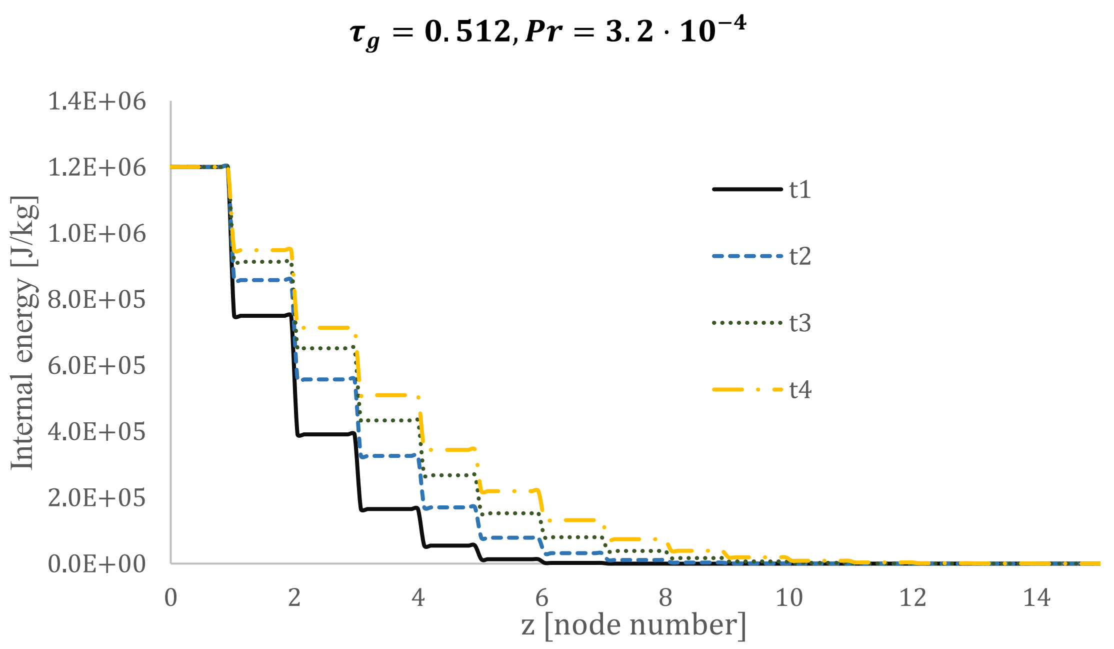


Figure 2-10: $t1 = 250,\ t2 = 500,\ t3 = 750,\ t4 = 1000$ ***timesteps***, *the algorithm shows a stable evolution of the heat distribution in the system*

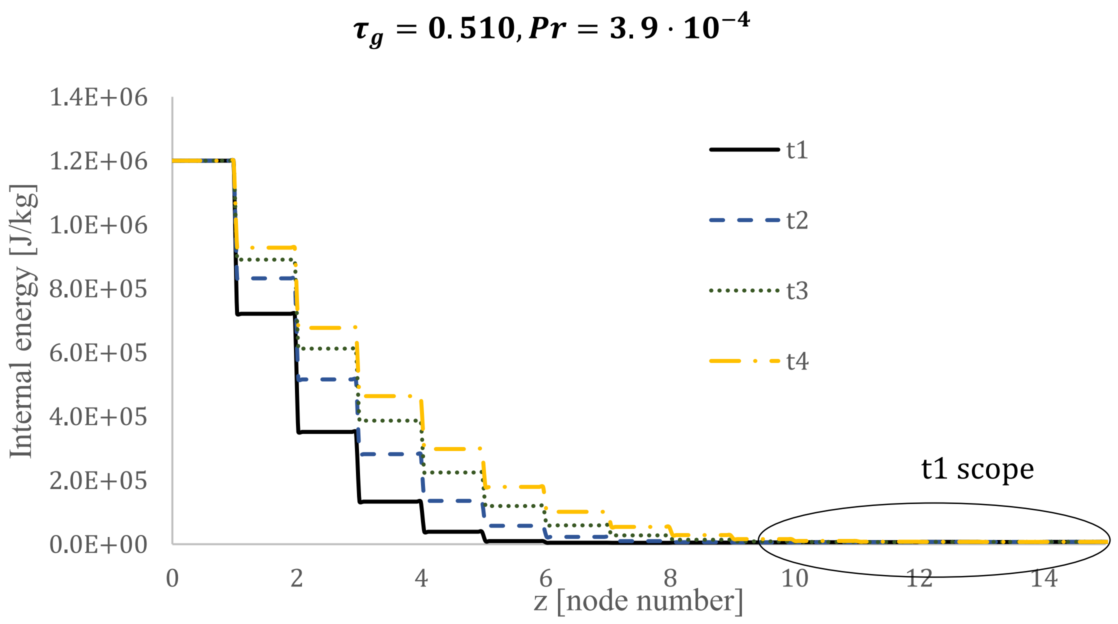


Figure 2-11: the stability limit is reached – a closer view of the data reveals slight instabilities (see Figure 2-12)

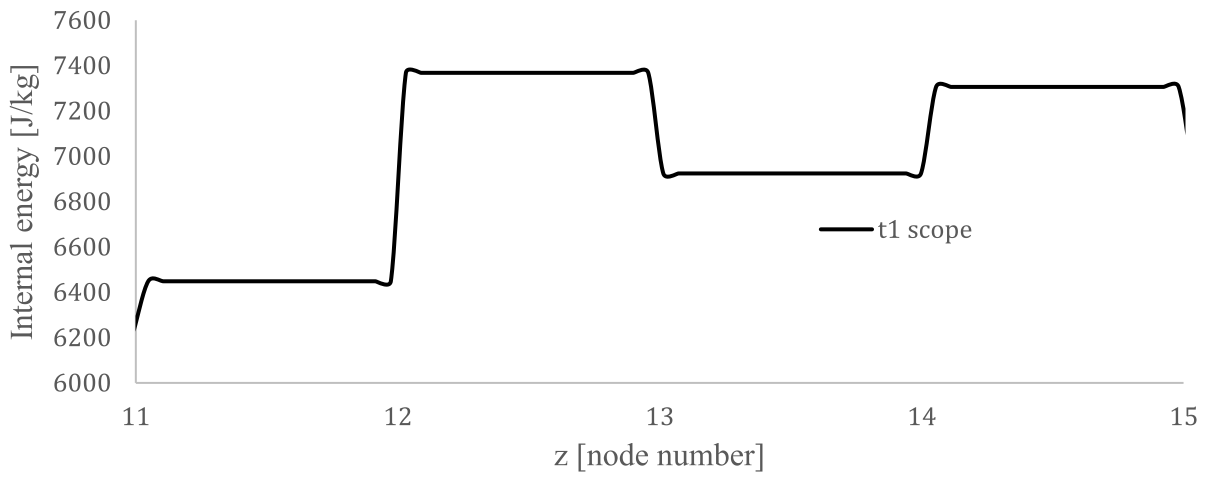


Figure 2-12: Nonphysical results of the method – towards the middle of the cylinder the temperature rises, although the heat gradient should be negative towards the middle (hotter wall, cold bulk)

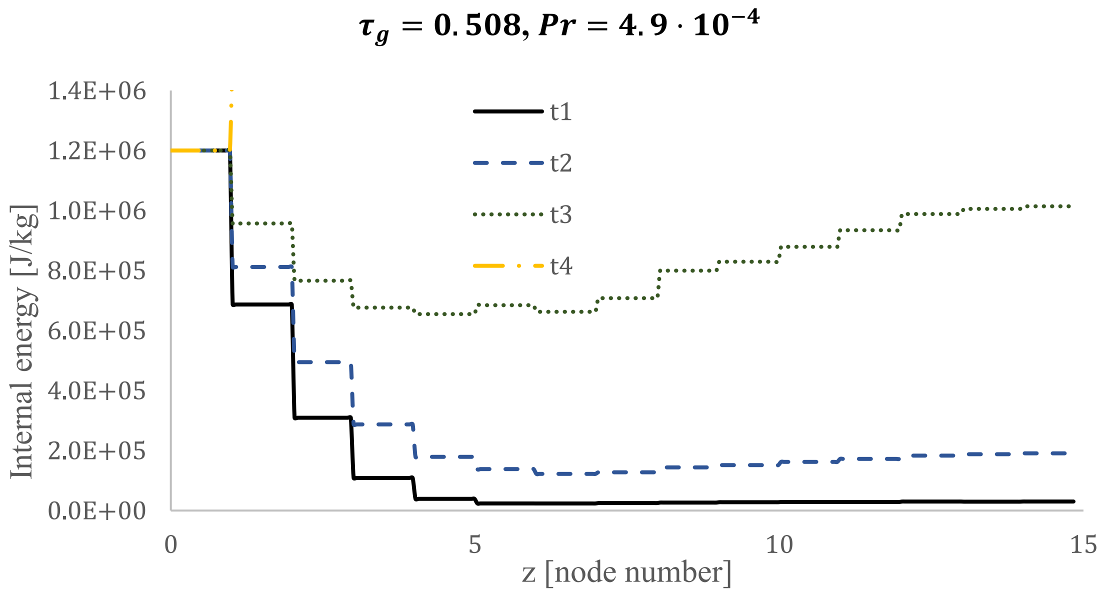


Figure 2-13: The stability limit is exceeded – the temperature starts to oscillate and reaches huge values after $\boldsymbol{t4 = 1000\ timesteps}$

The Figure 2-10 to Figure 2-13 show that the method is unstable for a relaxation time τ_g below 0.512. This limit equals a Prandtl number of $Pr = 3.2 \cdot 10^{-4}$ that is four magnitudes lower than the physical Prandtl number of common fluids such as water, which in turn implies that the method is only stable if the heat diffusivity α is four magnitudes larger than the physical one. It stands to reason that this considerable mismatch demands for an alternative or adapted method to describe the temperature distribution correctly.

2.1.3.2 Stability Issues

In the current literature the stability of the BGK-SRT-LBM for the flow field is well studied and guidelines are given to choose a stable pair of u_{max} and τ_f which is well summarized in Section 4.4 of [1]. However, the literature lacks in number-based guidelines for u_{max} and $\tau_g = f(Pe, Ma, N_d)$ which determine the stability of the (SRT-) TLBMs.

One crucial factor is the choice of the boundary condition handling [12]. The chosen method for boundary handling (see Section 2.1.1) leads to negative equilibrium values g^{eq} if the thermal relaxation time τ_g reaches values of about 0.52. At least for the fluid flow BGK-SRT-LBM the non-negativity of all equilibrium values f^{eq} is a sufficient criterion for the stability.

Papers from year 2013-2018 such as [13], [26], [27] propose more complex methods which are often based on the multi relaxation time lattice Boltzmann method MRT-LBM that attempt to overcome the known limited stability of the SRT-TLBM.

Still for the given problem of a stirred tank reactor with aqueous solution a purely advective LB based algorithm can be a reasonable choice to describe the transient temperature distribution which is explained and investigated in the following Section 2.2.

2.2 Direct Advection of Internal Energy by Density Distribution Function

Because of the instability issues and the large mismatch between the achievable Prandtl number and the real Prandtl number of liquids an alternative approach to the DDF-TLBM must be used.

In fact, the thermal diffusion modelled in the collision step of the g-distribution caused the numerical instabilities and it is questionable whether the advective effects may outweigh the diffusive effects in a stirred tank with aqueous liquids. As stated before the Peclet number $Pe = Re \cdot Pr$ provides information whether the diffusive term in the internal energy equation is negligible or not (see Eq. 1-35). The mixing Reynolds number of stirred tanks is defined as $Re = \frac{N_{rot}[\frac{rad}{s}] \cdot D_{stirr}^2}{\nu}$ and is typically in the range of $10^5 \div 10^6$. With a Prandtl number of about $10^{0.5}$ the Peclet number is larger than 10^5 which makes it reasonable to cancel diffusion in the temperature distribution simulation of the stirred bioreactor (Eq. 2-10). This implies that all effects such as external cooling/heating must be modelled as sinks/sources and the heat in the reactor is only advected by the flow field.

$$\frac{\partial(\tilde{\rho}\,\tilde{T})}{\partial\tilde{t}} + \tilde{\boldsymbol{\nabla}}\cdot(\tilde{\rho}\,\tilde{T}\,\tilde{\boldsymbol{u}}) = \underbrace{\cancel{\frac{1}{Pr\cdot Re}\,\tilde{\boldsymbol{\nabla}}\cdot(\tilde{\boldsymbol{\nabla}}\tilde{\boldsymbol{T}})}}_{\ll 1} + \frac{Ec}{Re}\,\tilde{\Phi} + \tilde{\rho}\,\tilde{\dot{q}} \qquad \textit{Eq. 2-10}$$

In 2016 Osmanlic et al. [28] introduced a purely advective transport equation that can as well be used to transport scalars in the fluid field such as the temperature T in the equation above [29]. The model is based on the calculation of a net flux into or out of a cell at $\boldsymbol{x} = (x, y, z)$ by a balance of mass flow between the node and all its neighboring nodes i at $\boldsymbol{x} + \boldsymbol{e_i}$ (Eq. 2-11).

$$\Delta m_i = \hat{f}_{-i}(\boldsymbol{x} + \boldsymbol{e_i}\Delta t, t) - \hat{f}_i(\boldsymbol{x}, t) \qquad \textit{Eq. 2-11}$$

$\hat{f}_i \ldots post\ collision\ density\ distribution\ function$

$\hat{f}_{-i} \ldots value\ in\ the\ opposite\ direction\ of\ \hat{f}_i$

The internal energy is either advected into the node by a neighboring node i ($\Delta m_i > 0$) or advected from the node to the corresponding neighboring node i ($\Delta m_i \leq 0$). Summing up all energy balances $\frac{\Delta m_i \cdot \varepsilon}{\rho}$ and adding the energy of the last timestep plus the heat source/sinks gives the updated post streaming internal energy of the node $\varepsilon(\boldsymbol{x}, t + \Delta t)$:

$$\varepsilon(\boldsymbol{x}, t+\Delta t) = \varepsilon(\boldsymbol{x}, t) + \left(\dot{\varepsilon}_d(\boldsymbol{x}, t) + \frac{1}{\rho}\dot{q}(\boldsymbol{x}, t)\right)\Delta t + \sum_{i=1}^{m} \Delta m_i \cdot \begin{cases} \dfrac{\varepsilon(\boldsymbol{x} + \boldsymbol{e_i}\Delta t, t)}{\rho(\boldsymbol{x} + \boldsymbol{e_i}\Delta t, t+\Delta t)}, & \Delta m_i > 0 \\ \dfrac{\varepsilon(\boldsymbol{x}, t)}{\rho(\boldsymbol{x}, t+\Delta t)}, & \Delta m_i \leq 0 \end{cases} \quad \ldots \quad \textit{Eq. 2-12}$$

The major advantage of the method is that the calculation of the net flux ensures that no energy can be created or lost by numerical calculation of the energy itself by Eq. 2-12. However, the numerical diffusion or/and oscillation cause deviations from the analytical solution.

Regarding computational costs the calculation is processed in one step compared to the two steps of streaming and collision and concerning storage costs the algorithm does not require extra computation fields like the g-distribution in the case of DDF-LBM.

2.2.1 Verification: Advection of Gauss Distribution

In this test case the implemented (energy) transport algorithm shall be verified for its correct simulation of a 1D Gauss distribution advection with constant fluid velocity. The four critical factors that are observed are

1) the correct shift of the peak of the distribution (mean of distribution),
2) the conservation of the overall integral,
3) the existence of numerical oscillations and retaining the *non-negativity constraint* and
4) the degree of numerical diffusion (reduction of amplitude, increase of standard deviation).

These factors are used to assess the applicability of the transport model with respect to the physical solution. The first constraint is that the advection velocity of the transported quantity coincides with the fluid velocity. Secondly, mass has to be conserved and thirdly, no negative values shall arise due to instabilities. Lastly the degree of numerical diffusion is evaluated and compared to the physical diffusion and other transport models.

The setup of the test case was adopted from a comparison of scalar transport algorithms in the field of LBM by Küng et al [29]. The 3D simulation domain equals a cuboid with only fluid nodes and periodic boundary conditions in all spatial directions. The initial Gauss distribution is given in Eq. 2-13:

$$\varepsilon_{Gauss}(z) = A \cdot \exp\left(-\frac{1}{2}\frac{(z-\mu_z)^2}{\sigma^2}\right) = 1 \cdot \exp\left(-\frac{1}{2}\frac{(z-500)^2}{50^2}\right) \qquad \textit{Eq. 2-13}$$

$A \ldots amplitude, \mu_z \ldots mean\ value, \sigma \ldots standard\ deviation$

This initial energy distribution is advected along the z-coordinate with constant lattice velocity of $u_{z,0} = 0.01$ by setting all f-values in the domain as their equilibrium distribution values $f_i(x,t) = f_i^{eq}\left(u = \left(0,0,u_{z,0}\right)\right)$. The choice of $u_{z,0} = 0.01$ in the test case is appropriate for the stirred bioreactor simulation since it equals the chosen lattice tip velocity of the stirrer and thus represents the highest velocity magnitude in the simulation. The Figure 2-14 depicts the result of the verification case:

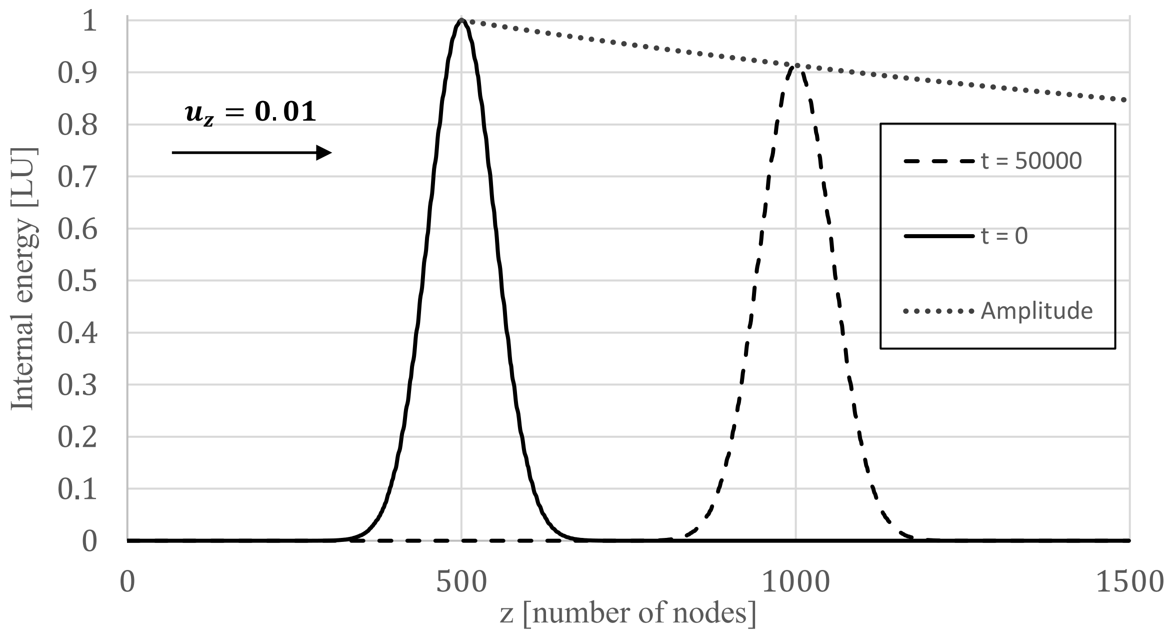


Figure 2-14: A 1D Gauss distribution (amplitude of 1, standard deviation of 50 and mean value of 500) is advected by a constant fluid velocity of $u_z = 0.01$ *for 50000 timesteps. The dashed curve shows the Gauss distribution after the simulation which has a reduced amplitude of 0.9136 and an increased standard deviation due to numerical diffusion.*

1) The implemented Osmanlic transport algorithm moves the initial distribution without relative shift.

2) The energy in the system represented by an overall integral in z-coordinate stays the same:

Table 2-2: Calculation of the overall integral for different timesteps. The integral provides a measure for the total energy in the system. The value of the integral stays about the same. The error may be caused by numerical integration (trapezoidal rule, 1000 increments from z=0 to z=1500)

$timesteps$	0	50 000	100 000
$\int_{z=0}^{1500} \varepsilon_{Gauss}(z)\, dz$	125.3314	125.3315	125.3303
$error[\%]$	-	$0.91 \cdot 10^{-4}$	$0.93 \cdot 10^{-3}$

3) The algorithm does not oscillate and does not give negative results.

4) Compared to other solvers such as the common Lax-Wendroff method the numerical diffusion is less for the set fluid velocity of $u_z = 0.01$. The decay of the Gauss amplitude can be calculated by Eq. 2-14 and Eq. 2-15 (see [29], Eq. 16, 17). For the 1D Osmanlic algorithm

the numerical diffusion coefficient D^{Os} is dependent on the lattice velocity of the fluid and is a parabola with zeroes at $u = 0.0$ and $u = 1.0$ and maximum at $u = 0.5$:

$$D^{Os} = \frac{1}{2}(1-u) \cdot u = 0.00495 \qquad \textit{Eq. 2-14}$$

$$A(t) = A_0 \sqrt{\frac{\sigma_0^2}{\sigma_0^2 + 2D^{Os}t}} = 1 \cdot \sqrt{\frac{50^2}{50^2 + 2 \cdot 0.00495 \cdot 50000}} = 0.9136 \qquad \textit{Eq. 2-15}$$

$A_0 \ldots$ *initial amplitude*, $\sigma_0 \ldots$ *initial standard deviation*,
$D^{Os} \ldots$ *numerical diffusion constant*, $t \ldots$ *time in timesteps*

The respective Gauss amplitude of the simulation and the decayed amplitude given by the above analytic equations at timestep $t = 50000$ are exactly the same. The results of Küng et al depicted in Fig 2 of paper [29] show an even higher decay of the amplitude which contradicts with the results of the stated equations given in the same paper (see [29], Eq. 16, 17).

All in all, the result of the verification test provides physically meaningful results. The value of the simulated numerical diffusion can be verified by analytic formulas.

2.2.2 Comparison of Numerical and Physical Diffusion

As mentioned in Section 2.2.1 the numerical diffusion coefficient of the 1D Osmanlic transport algorithm can be described by a second order polynomial (parabola) of fluid velocity in lattice units. The numerical diffusivity is solely driven by fluid velocity and therefore does not exist for a steady fluid. Currently there are no references given in the literature that describe the numerical diffusion in a 3D fluid domain whose magnitude and (an)isotropy shall be compared to the true physical diffusion of heat in a fluid. For following flow and LB conditions, the numerical diffusion coefficient D^{Os} and additional error terms of the algorithm are derived. The detailed derivation can be looked up in the Appendix (Section 8.2). The derivation is valid for following assumptions:

- Steady-state fluid flow → $f_i = f_i^{eq}$
- $u_x \geq u_y \geq u_z \geq 0$
- BGK collision operator with second order equilibrium distribution function (see Eq. 1-5 and Eq. 1-8)

- D3Q19 lattice arrangement
- Node at $\boldsymbol{x} = (x, y, z)$ whose 18 neighboring cells have the same component velocities $\boldsymbol{u}(\boldsymbol{x}) = (u_x, u_y, u_z)$

With the assumption that the fluid density ρ and heat capacity c_v are constant the physical advection-diffusion equation for the internal energy ε can be written as follows:

$$\frac{\partial(\rho\,\varepsilon)}{\partial t} + \underbrace{\boldsymbol{\nabla}\cdot(\rho\,\varepsilon\,\boldsymbol{u})}_{advection} = \underbrace{\boldsymbol{\nabla}\cdot(k\,\boldsymbol{\nabla}\,T)}_{diffusion} \qquad with\ \varepsilon = c_v T\ and\ \alpha = \frac{k}{\rho c_v}$$

$$\frac{\partial T(\boldsymbol{x},t)}{\partial t} + u_x\cdot\frac{\partial T(\boldsymbol{x},t)}{\partial x} + u_y\cdot\frac{\partial T(\boldsymbol{x},t)}{\partial y} + u_z\cdot\frac{\partial T(\boldsymbol{x},t)}{\partial z} = \alpha\left(\frac{\partial^2 T(\boldsymbol{x},t)}{\partial x^2} + \frac{\partial^2 T(\boldsymbol{x},t)}{\partial y^2} + \frac{\partial^2 T(\boldsymbol{x},t)}{\partial z^2}\right) \qquad Eq.\ 2\text{-}16$$

The derivation of the numerical diffusion of the Osmanlic transport algorithm leads to following form of the advection diffusion equation:

$$\frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial t} + u_x\cdot\frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial x} + u_y\cdot\frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial y} + u_z\cdot\frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial z} = \\ = \underbrace{\begin{pmatrix} -\frac{1}{2}u_x^2 + \frac{1}{2}u_x \\ -\frac{1}{2}u_y^2 + \frac{1}{3}u_y + \frac{1}{6}u_x \\ -\frac{1}{2}u_z^2 + \frac{1}{6}u_z + \frac{1}{6}u_x + \frac{1}{6}u_y \end{pmatrix}}_{\boldsymbol{D^{Os}}} \cdot \begin{pmatrix} \frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial x^2} \\ \frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial y^2} \\ \frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial z^2} \end{pmatrix} + \underbrace{\begin{pmatrix} \frac{1}{3}u_y \\ \frac{1}{3}u_z \\ \frac{1}{3}u_z \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial x\partial y} \\ \frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial x\partial z} \\ \frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial y\partial z} \end{pmatrix}}_{additional\ error\ term}$$

$$\ldots \qquad Eq.\ 2\text{-}17$$

The numerical diffusion is anisotropic with stronger diffusion towards the larger spatial component velocity u_x, u_y or u_z which is in this case the x-component. The x-component of the numerical diffusion coefficient remains the same as in the 1D case (see Eq. 2-14). The bigger spatial component velocities affect the diffusion components of smaller component velocities but not vice versa. This is for example represented by the $\frac{1}{6}\mathrm{u_x} + \frac{1}{6}\mathrm{u_y}$ term in the z-component of the diffusion coefficient vector $\boldsymbol{D^{Os}}$.

The description of the numerical diffusion for a random fluid node in a stirred bioreactor is obviously more complex, depending on 36 different f-values of the fluid node (18) and its neighboring nodes (+18) instead of three constant spatial component velocities. The derived diffusion coefficient vector $\boldsymbol{D^{Os}}$ allows for a comparison of order of magnitudes of physical and numerical diffusion though (Figure 2-15).

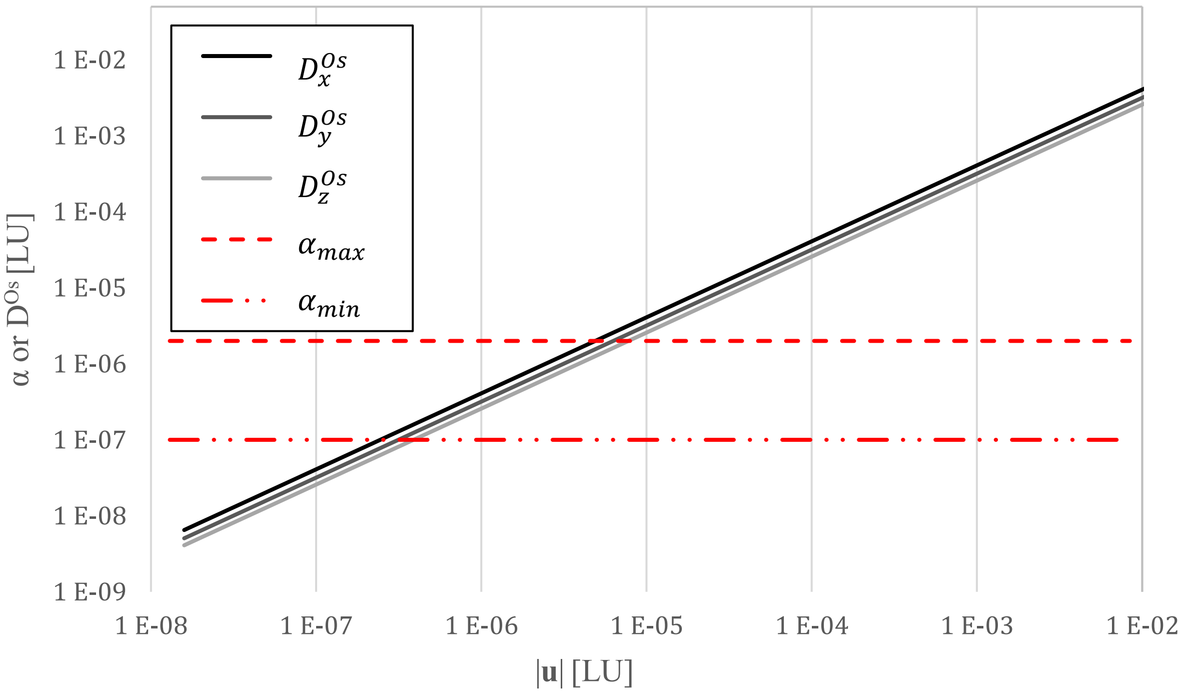


Figure 2-15: Comparison between physical and numerical diffusion of the Osmanlic transport algorithm. The physical diffusion coefficient is given by $\alpha[LU] = \frac{C_t}{Pr \cdot C_l^2} \cdot \nu^* \left[\frac{m^2}{s}\right]$ *which is for water in a plausible range of* $[\alpha_{min}[LU] = 0.1 \cdot 10^{-6}, \alpha_{max}\,[LU] = 2 \cdot 10^{-6}]$. *For reasons of simplicity of the comparison the y and z component velocity are given by the x component velocity* $u_y = \frac{2}{3}u_x, u_z = \frac{1}{3}u_x$. *The x-axis of the plot is given by the magnitude of the velocity vector.*

The plot shows that the numerical diffusion is already greater than the physical diffusion at order of magnitudes of $|\mathbf{u}| > 10^{-6} = 0.01\% \cdot \mathrm{u_{max}}$. As previously mentioned the maximum velocity $\mathrm{u_{max}}$ is given by the tip-velocity of the stirrer blades and is set to 0.01 in the lattice unit conversion.

So, although the advective algorithm does not formulate physical diffusion its numerical diffusion is larger than the physical diffusion for a wide range of fluid velocities. For the given derivation and assumptions above the numerical diffusion can be about four magnitudes larger if the fluid velocity is equal to the tip-velocity of the stirrer. If the fluid is steady, the numerical diffusion vanishes.

2.2.3 Verification: Numerical Diffusion Normal to Fluid Velocity Vector

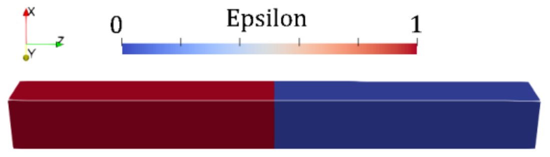
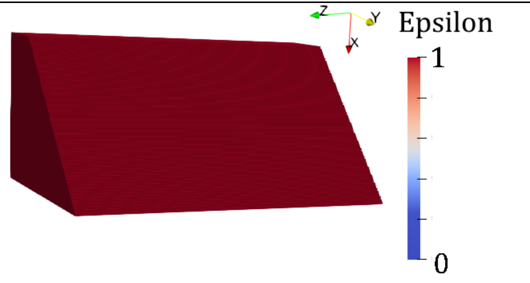
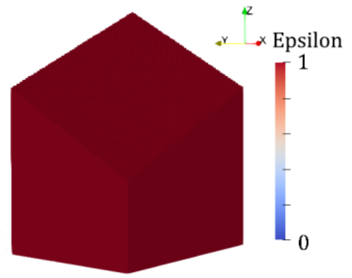
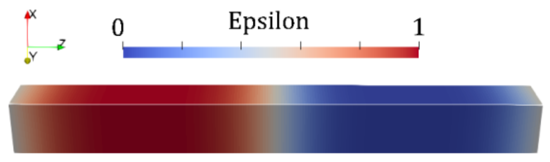
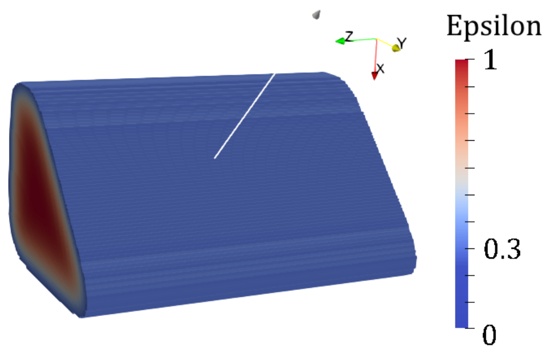
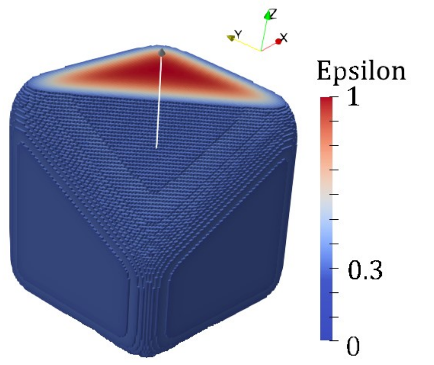
In previous Section 2.2.3 the diffusion *along* the fluid velocity vector $\boldsymbol{u}$ was investigated for a Gauss distribution. Since the numerical diffusion appears normal to the fluid velocity vector as well (see Eq. 2-17), three further verification cases were carried out to confirm this quantitatively.

The first setup is a 1D top hat function of the internal energy $\varepsilon(z)$ that is advected in the y- or x-coordinate. The numerical diffusion of the initial function is investigated along the z-axis. In this simple case the velocity vector aligns with the cubic discretized cells.

The second setup is analogous to the 1D top hat function. The step from $\varepsilon = 0$ to $\varepsilon = 1$ is set at a 2D plane E_{2D} along the $(1,1,0)$ square diagonal.

The third setup is a 3D case again analogous to the 1D top hat function. This time the step from $\varepsilon = 0$ to $\varepsilon = 1$ is set at a 3D plane E_{3D} along the $(1,1,1)$ space diagonal.

In the following, the results and setup of the 1D verification case is shown on the left, the 2D verification case in the middle column while the 3D verification case is shown on the right.

Initial distribution setup (left 1D, middle 2D, right 3D)		
 *Figure 2-16: Initial setup for investigation of 1D numerical diffusion in z-axis, fluid flow towards x-axis (aligned with lattice structure)*	 *Figure 2-17: Initial setup (set image threshold at* $\varepsilon = 1.0$) *for investigation of 2D numerical diffusion normal to plane* E_{2D}*, fluid flow towards the square diagonal* (1,1,0)	 *Figure 2-18: Initial setup (set image threshold at* $\varepsilon = 1.0$) *for investigation of 3D numerical diffusion normal to plane* E_{3D}*, fluid flow towards the space diagonal* (1,1,1)
Final distribution after simulation		
 *Figure 2-19: Numerical diffusion visible after 50000 timesteps at initial step in the middle of the domain (z=100 nodes)*	 *Figure 2-20: Numerical diffusion visible after 9000 timesteps (set image threshold at* $\varepsilon = 0.3$)*, diffusion is observed along the normal vector of the plane (white line normal to plane* E_{2D}*)*	 *Figure 2-21: Numerical diffusion visible after 9000 timesteps (set image threshold at* $\varepsilon = 0.3$)*, diffusion is observed along the normal vector of the plane (white line normal to plane* E_{3D}*)*

Velocity vector, initial energy distribution, numerical diffusion coefficient, analytical solution		
$\lvert u\rvert = 0.01, \quad \boldsymbol{u} = (0.01,0,0)$	$\lvert u\rvert = 0.01, \quad \boldsymbol{u} = \left(\frac{0.01}{\sqrt{2}}, \frac{0.01}{\sqrt{2}}, 0\right)$	$\lvert u\rvert = 0.01, \quad \boldsymbol{u} = \left(\frac{0.01}{\sqrt{3}}, \frac{0.01}{\sqrt{3}}, \frac{0.01}{\sqrt{3}}\right)$
$\varepsilon(z,t=0) = \begin{cases} 0, & \mathrm{z} > \frac{1}{2} z_{max} \\ 1, & \mathrm{z} \leq \frac{1}{2} z_{max} \end{cases}$	$E_{2D} = \{(x,y,z) \in \mathbb{R} \mid -x+y=0\}$ $\varepsilon(x,y,z,t=0) = \begin{cases} 0, & 0 \leq -x+y \\ 1, & 0 > -x+y \end{cases}$	$E_{3D} = \left\{(x,y,z) \in \mathbb{R} \mid -\frac{1}{2}x - \frac{1}{2}y + z = 0\right\}$ $\varepsilon(x,y,z,t=0) = \begin{cases} 0, & 0 \leq -\frac{1}{2}x - \frac{1}{2}y + z \\ 1, & 0 > -\frac{1}{2}x - \frac{1}{2}y + z \end{cases}$
$D_z^{Os} = \frac{1}{6} u_x = 0.00167$	$D_{xy}^{Os} = -\frac{1}{2} u_{xy}^2 + \frac{1}{2} \mathrm{u_{xy}} = 0.00351$	$D_{xyz}^{Os} = -\frac{1}{2} u_{xyz}^2 + \frac{1}{2} \mathrm{u_{xyz}} = 0.00287$
$\varepsilon(z,t) = \frac{1}{2} \cdot \left(1 - \mathrm{erf}\left(\frac{z - \frac{1}{2} z_{max}}{2 \cdot \sqrt{D_z^{Os} t}}\right)\right)$	n_{2D} *... plot axis normal to the plane* E_{2D} $\varepsilon(n_{2D},t) = \frac{1}{2} \cdot \left(1 - \mathrm{erf}\left(\frac{n_{2D} - n_{plane}}{2 \cdot \sqrt{D_{xy}^{Os} t}}\right)\right)$	n_{3D} *... plot axis normal to the plane* E_{3D} $\varepsilon(n_{3D},t) = \frac{1}{2} \cdot \left(1 - \mathrm{erf}\left(\frac{n_{3D} - n_{plane}}{2 \cdot \sqrt{D_{xyz}^{Os} t}}\right)\right)$

Analytical vs. simulated solution of numerical diffusion (top: 1D, 2D, 3D case; bottom: 2D, 3D case diffusion in x-coordinate)

Internal energy [LU]

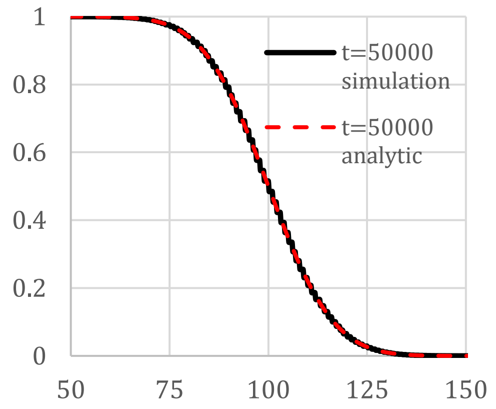


Figure 2-22: 1D

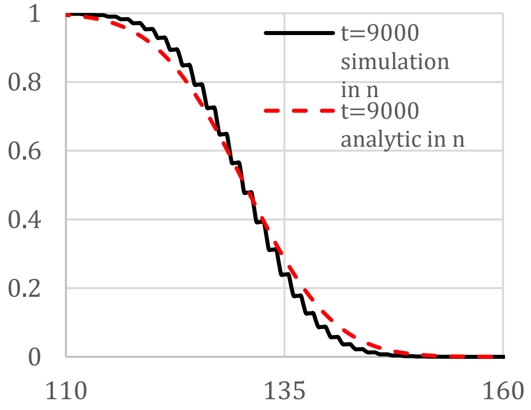


Figure 2-23: 2D diffusion normal to E_{2D}

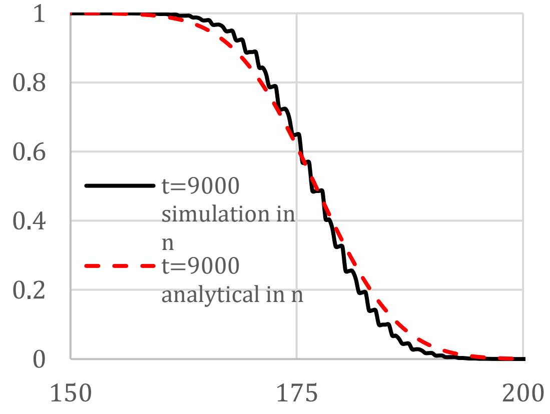


Figure 2-24: 3D diffusion normal to E_{3D}

-

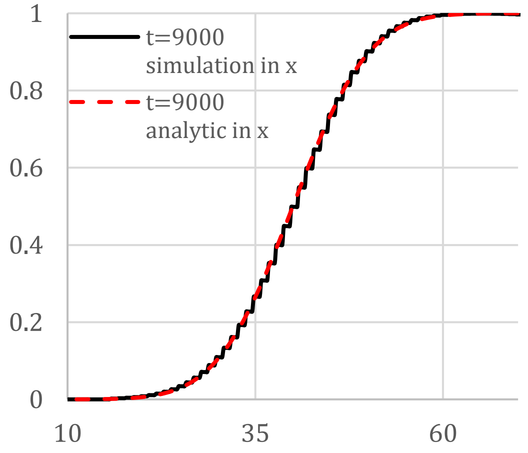


Figure 2-25:2D diffusion in x-coordinate

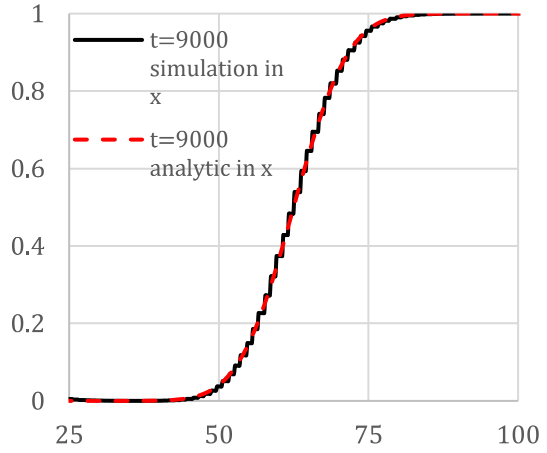


Figure 2-26: 3D diffusion in x-coordinate

z [number of nodes]

The 1D case simulation (see left column) results in a perfect agreement with the analytical solution of the numerical diffusion which is depicted in the plot at the bottom of the column (Figure 2-22). This further proves the correct implementation of the algorithm and the correct derivation of the numerical diffusion.

Compared to the analytical solution of the numerical diffusion both the 2D and 3D case (see middle and right column) show a reduced diffusion perpendicular to the plane that aligns with the fluid flow vector but not with the cubic lattice structure. This disparity can be seen in the upper line of the plots at the bottom of the column (Figure 2-23 and Figure 2-24) as well as in the thermal image of the simulation output (Figure 2-27). Still, the diffusion towards the space coordinate which aligns with the cubic lattice structure coincides with the analytical solution (Figure 2-25 and Figure 2-26). The data for Figure 2-25 and Figure 2-26 was taken from a vertical plane that aligns with the lattice structure (see for 2D case left arrow Figure 2-27).

Seemingly, there is a dependency whether the direction of the diffusion aligns with the lattice structure. If the diffusion direction aligns with lattice structure, the solution can be verified by an analytical solution which can be proven for the 1D, 2D and 3D velocity vectors respectively. If the direction of the diffusion does not align with the cubic lattice structure the diffusion is reduced which is shown in the 2D and 3D test case simulations.

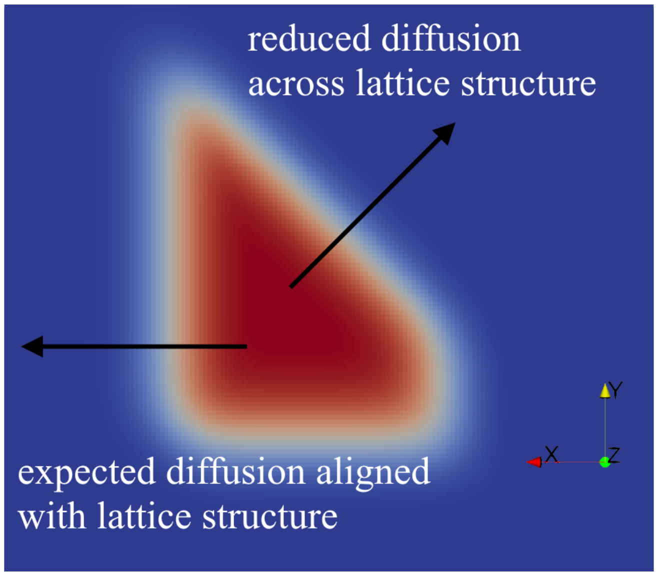


Figure 2-27: Numerical Diffusion for 2D Case at 7000 timesteps, the numerical diffusion can be determined for diffusions along the cubic lattice structure, the diffusion across the lattice structure (e.g. square diagonal) is reduced

One straightforward attempt at an explanation for the reduced diffusion across the lattice structure is that the diffusion does only behave as derived if the direction of the diffusion aligns with the lattice structure. Thus, the diffusion across the lattice grid such as in square diagonal or space diagonal direction is solely a result from diffusion that aligns with the lattice structure. See Figure 2-28 for a schematic explanation.

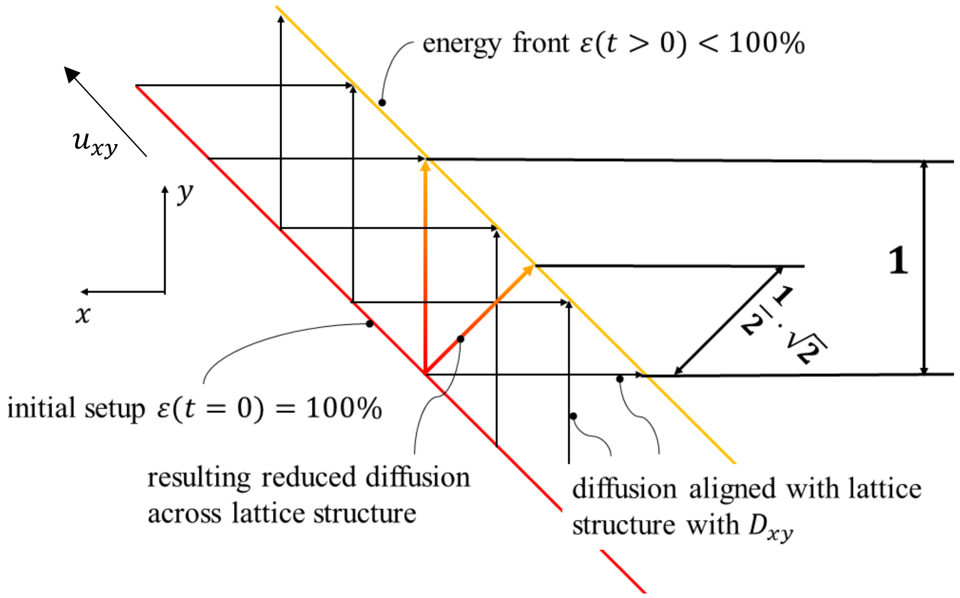

energy front $\varepsilon(t > 0) < 100\%$
u_{xy}
y
x
1
$\frac{1}{2} \cdot \sqrt{2}$
initial setup $\varepsilon(t = 0) = 100\%$
resulting reduced diffusion across lattice structure
diffusion aligned with lattice structure with D_{xy}

Figure 2-28: Diffusion that aligns with the lattice structure is represented by the vertical or horizontal black arrows, the diffusion across the lattice structure in square diagonal direction does only result from diffusion that aligns with lattice structure in x and y coordinate

According to Figure 2-28 the effective diffusion across the lattice grid can be described by the x and y coordinate in the 2D case:

$$D_{2D,(-1,1,0)} = \sqrt{\left(\frac{1}{2} D_{xy}^{Os}\right)^2 + \left(\frac{1}{2} D_{xy}^{Os}\right)^2} = \frac{1}{2}\sqrt{2} \cdot D_{xy} \qquad \textit{Eq. 2-18}$$

Analogous to the 2D case the corrected diffusion can be stated for the 3D case:

$$D_{3D,\left(-\frac{1}{2},-\frac{1}{2},1\right)} = \sqrt{\left(\frac{1}{2}\frac{1}{2} D_{xyz}^{Os}\right)^2 + \left(\frac{1}{2}\frac{1}{2} D_{xyz}^{Os}\right)^2 + \left(\frac{1}{2} D_{xyz}^{Os}\right)^2} \qquad \textit{Eq. 2-19}$$

The analytic solutions can be corrected by inserting the effective diffusion coefficients:

$$\varepsilon(n_{2D},t) = \frac{1}{2}\cdot\left(1-\operatorname{erf}\left(\frac{n_{2D}-n_{plane}}{2\cdot\sqrt{D_{2D,(-1,1,0)}t}}\right)\right) \quad or$$

$$\varepsilon(n_{3D},t) = \frac{1}{2}\cdot\left(1-\operatorname{erf}\left(\frac{n_{3D}-n_{plane}}{2\cdot\sqrt{D_{3D,\left(-\frac{1}{2},-\frac{1}{2},1\right)}t}}\right)\right) \qquad \textit{Eq. 2-20}$$

Finally, the comparison previously shown by Figure 2-23 and Figure 2-24 is depicted in Figure 2-29 and Figure 2-30 with the corrected effective analytic solutions given by Eq. 2-20.

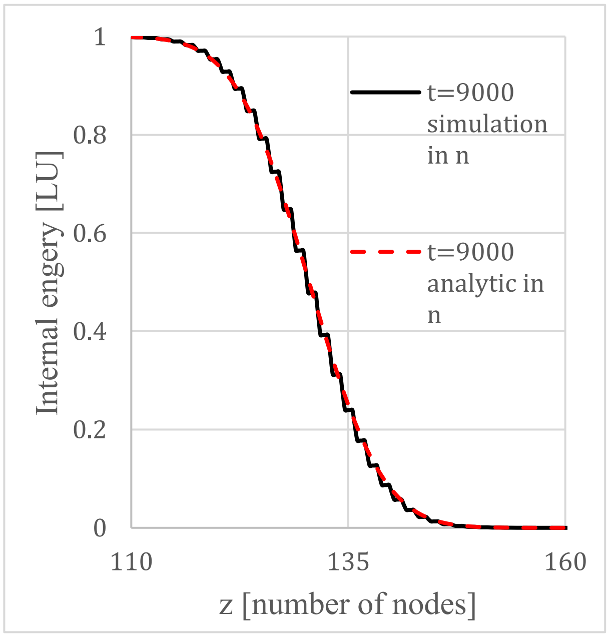


Figure 2-29: 2D numerical diffusion across lattice structure compared to analytical solution with corrected effective diffusion coefficient

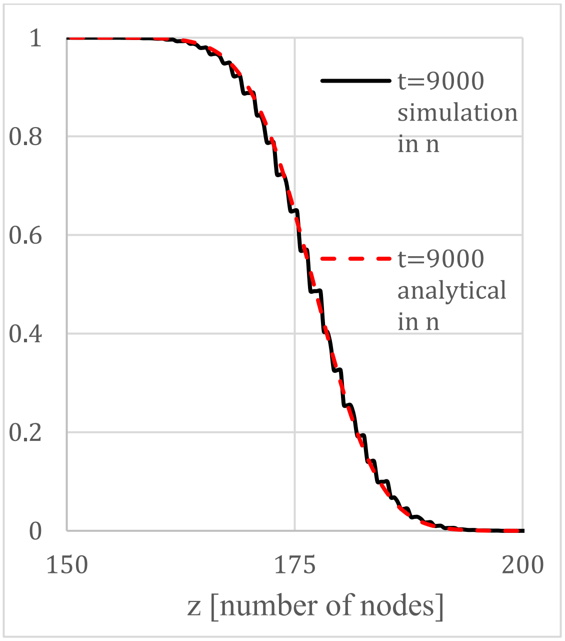


Figure 2-30: 3D numerical diffusion across lattice structure compared to analytical solution with corrected effective diffusion coefficient

The deviation of analytical and simulation results is less compared to the deviation depicted in Figure 2-23 and Figure 2-24 for the 2D and 3D case respectively.

3 *Bioreactor Implementation*

The Osmanlic transport algorithm (Section 2.2) for scalars allows to implement conditions that describe the thermal distribution problem:

- Set **absolute internal energy value** to a certain value at certain position and time (e.g. initial temperature, temperature measurement)
- Set arbitrarily chosen or physically modelled values of **internal energy sources or sinks** (e.g. external heating or cooling devices, heat generated by biosynthesis)

In the following Figure 3-1 an overview is of the code structure is given.

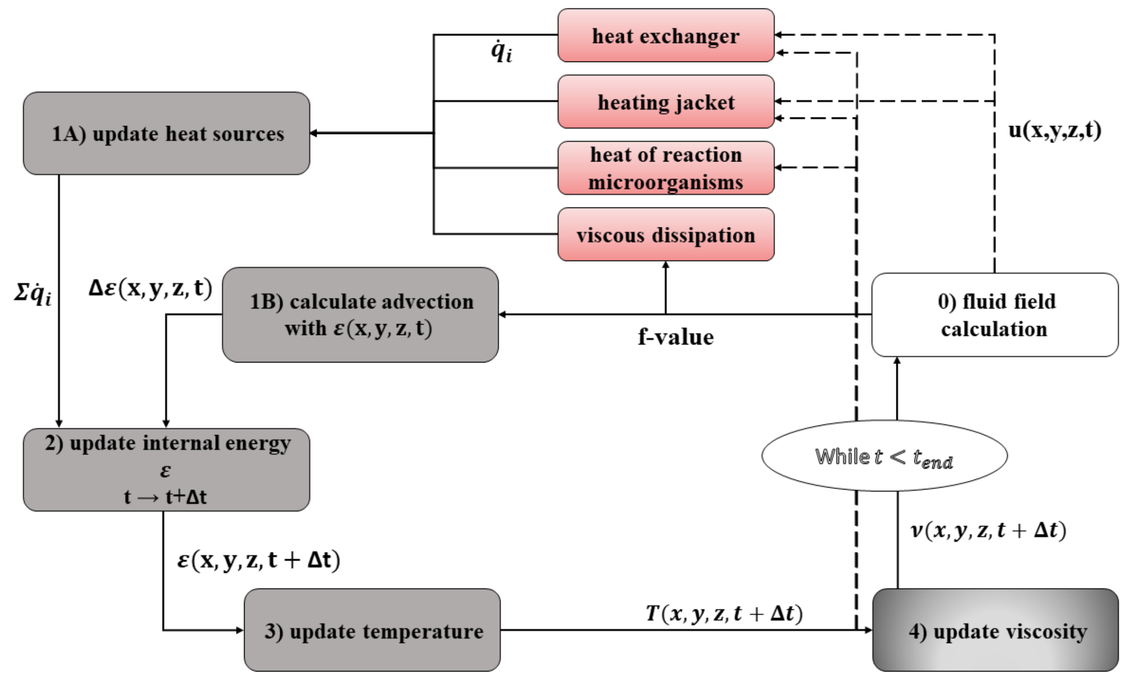


Figure 3-1: Overview of the thermal distribution calculation, solid lines represent implemented features, dashed lines represent possible coupled models

The respective subsections of this chapter explain the implementation of different features to describe the temperature distribution in an industrial stirred tank bioreactor.

3.1 *Heating Jacket*

3.1.1 *Implementation Method*

The general task is to implement a heating jacket with a predetermined total heat power $\dot{Q}_{HJ}[W]$ at a certain vertical position and height. The approach is to model the heating jacket as a heat source to the fluid nodes that it encloses. The implementation was split into following three steps with short description below:

1) Define heating jacket nodes that have heat source $\dot{q}_{HJ}\left[\frac{W}{m^3}\right]$ due to the heating jacket
2) Count the number N_{HJ} of these nodes
3) Uniformly split the total heat power $\dot{Q}[W]$ to these nodes and add the heat source $\dot{q}_{HJ}\left[\frac{W}{m^3}\right]$ to the overall heat source field $\dot{\boldsymbol{q}}(x, y, z, t)$ for the given heat up time t_{hu}

The heating jacket nodes are the boundary fluid nodes that are in in direct contact with the tank wall of the fluid and limited vertically by an upper limit $z_{HJ,uL}$ and a lower limit $z_{HJ,lL}$. Both limits can be adapted in the input parameter sheets and are defined by the ratio of the vertical length from the bottom to respective limit divided by the total height of the reactor. Accordingly, a heating jacket from the middle of the reactor to its top is initialized by $z_{HJ,uL} = 1$ and $z_{HJ,lL} = 0.5$. The boundary fluid nodes are determined in the initialization procedure of solid tank nodes. The given total power $\dot{Q}_{HJ}[W]$ is split to all heating jacket nodes as a uniform heat source $\dot{q}_{HJ}\left[\frac{W}{m^3}\right]$ that contributes to the overall heat source field $\dot{q}(\boldsymbol{x}, t)$ in the set heat up duration t_{hu}.

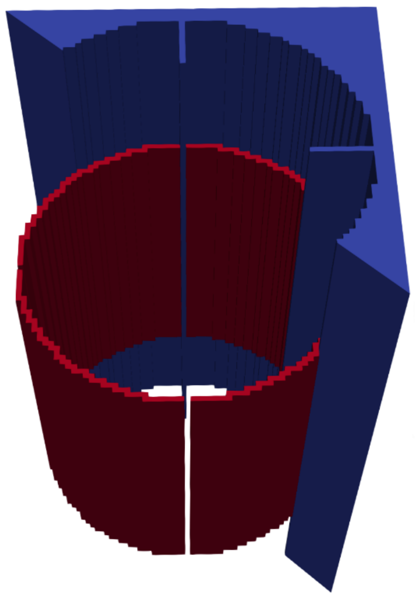


Figure 3-2: Simulation setup with active heating jacket, red cells represent the heat jacket nodes with heat source $\dot{q}_{HJ}\left[\frac{W}{m^3}\right]$ that form a ring with set vertical position and height. The nodes are in contact with the tank wall (blue cells).

3.1.2 Verification Method

The correct implementation is verified by calculating the average fluid temperature in the simulation of the stirred tank reactor with set heating jacket power and compare it to the analytic solution of the temperature of an ideally mixed system with the very same heat source (Eq. 3-2).

$$\frac{d\left(\rho c_v V_{fluid} T\right)}{dt} = \dot{Q}_{HJ}$$

$$\begin{aligned} &with\ c_v = const \\ &with\ T(t=0) = T_0 \\ &with\ T \ldots temperature\ of\ an\ ideally\ mixed\ fluid \end{aligned} \qquad \textit{Eq. 3-1}$$

$$T(t) = \frac{\dot{Q}_{HJ}}{\rho c_v V_{fluid}} \cdot t + T_0 \qquad \textit{Eq. 3-2}$$

The heating jacket will be active from the beginning of the simulation for the set heat up duration and meanwhile provides a set constant thermal energy to the system. The verification should prove that the average temperature in the system is the same as the one of an ideally mixed system and that the energy is the system stays constant after the heat up duration. The latter approves that the general evolution of the energy is correctly implemented in the coded algorithm and the former approves the correct initialization of the heating jacket.

Following input parameter were chosen in the two verification cases 1 and 2:

Table 3-1: Set input parameter for the test cases with constant heat jacket power, 1:due to staircase approximation of the circular cross-section of the cylindric reactor and installation volume (e.g. baffles) the actual mass is reduced

Physical Quantity	**Case 1**	**Case 2**	**Physical Quantity**	**Case 1 & Case 2**
fluid volume V	$0.050\ m^3$	$0.035\ m^3$	spec. heat capacity c_v	$4000\ \frac{kg}{m^3}$
actual fluid volume[1] V_{act}	$0.046241\ m^3$	$0.032708\ m^3$	total heating jacket power $\dot{Q}_{HJ}$	$1\ 000\ 000\ W$
upper limit $z_{HJ,uL}$	0.75	1.0	stirrer speed N_{stirr}	$190\ \frac{rev}{min}$
lower limit $z_{HJ,lL}$	0.2	0.0	initial temperature T_0	$300\ K$
nodes per meter	200	150	density ρ	$1000\ \frac{kg}{m^3}$
diameter d_R	$0.30\ m$	$0.35\ m$	viscosity ν	$1 \cdot 10^{-6}\ \frac{m^2}{s}$
total heat up time t_{hu}	$8\ s$	$10\ s$		

The results of the verification case are depicted in Figure 3-3 and Figure 3-4.

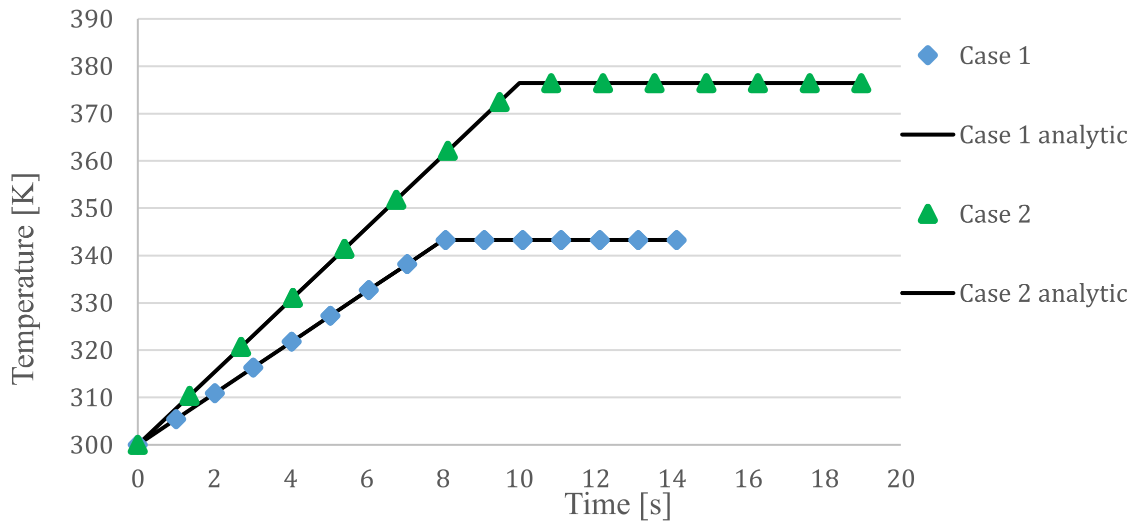


Figure 3-3: Heating curves of both verification cases with constant heating power over set time compared to the respective analytical solution

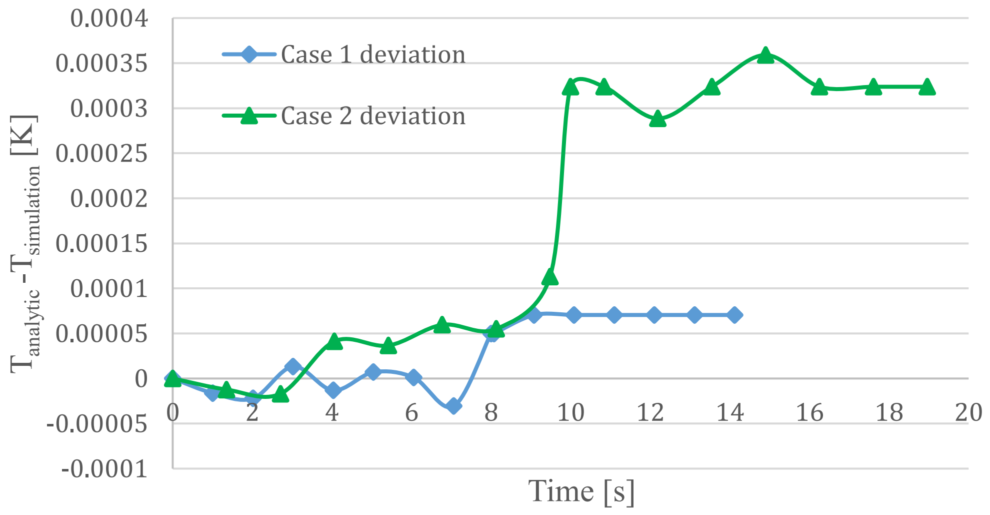


Figure 3-4: Absolute difference of the analytic and the simulated temperature for both verification cases 1 and 2 with constant heating power over time

For both test cases 1 and 2 the average temperature in the system during the heat up duration of $t_{hu} = 8\ s$ and $t_{hu} = 10\ s$ is in exact agreement with the analytical solution and stays constant afterwards. The slight oscillation of the temperature in case 1 at about $12\ s$ and $15\ s$ is caused by floating point error. This does not affect the calculation of the thermal distribution because the internal energy ε in the system which is initialized with *double precision* instead of *single precision* stays the same. After this oscillation the temperature settles and does not diverge. The maximum relative error rd of both cases equals

$$rd = \frac{\max(|T_{analytic} - T_{simulation}|)}{T(t = t_{hu}) - T(t = 0)} = \frac{0.0003589\ K}{76.433\ K} = 4.7 \cdot 10^{-4}\ \%.$$

Overall, both cases verify the correct implementation of the heating jacket as a heat source in the thermal simulation framework.

3.1.3 Thermal Distribution Example Heating Jacket

In the overview Figure 3-5 and its subfigures an example of a stirred tank with heating jacket is depicted. The arbitrarily set large heating jacket power $\dot{Q}_{HJ}$ is chosen to obtain obvious temperature gradients in the stirred fluid.

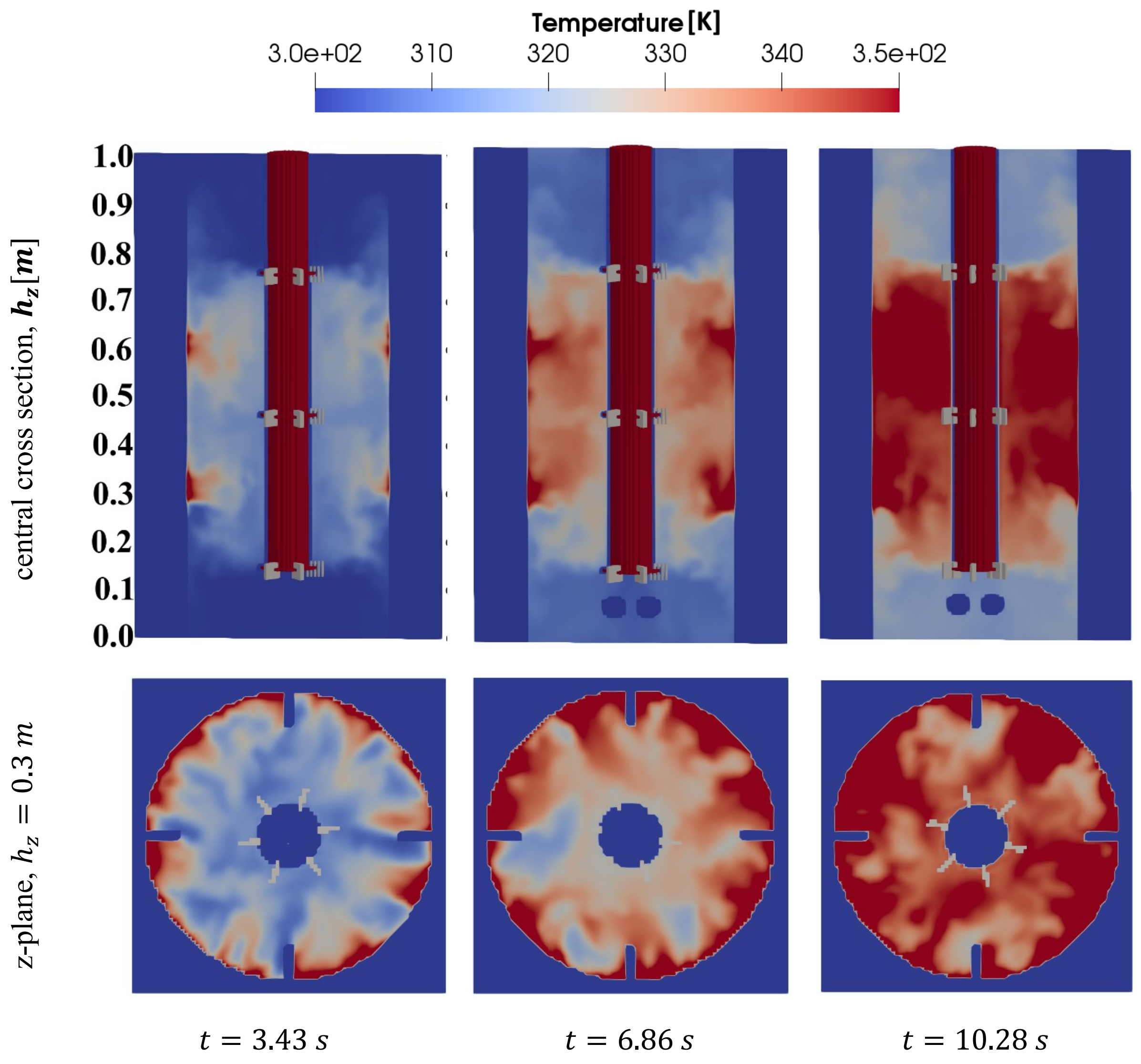


Figure 3-5: Example of a stirred tank with initialized heating jacket , significant input parameters: initial temperature $T_0 = 300K$, *heating jacket power* $\dot{Q}_{HJ} = 2000\ kW$, *actual fluid volume* $m_{act} = 142.1\ kg$, $h_{HJ,uL} = 0.761\ m$, $h_{HJ,lL} = 0.254\ m$, *diameter* $d_R = 0.44\ m$, *tank height* $h_R = 1.015\ m$, *kinematic viscosity* $\nu = 1 \cdot 10^{-6} \frac{m^2}{s}$, *stirrer speed* $N_{stirr} = 190 \frac{rev}{min}$

Over time a heat up of the overall liquid is observable. The typical double vortexes between the three vertical arranged Rushton impellers allow for an effective horizontal heat distribution but inhibit the vertical distribution in the tank. This is recognisable by the cold bottom and top liquid compartments.

3.2 Tube Bundle Heat Exchanger

In the current simulation framework tube bundle heat exchanger TBHE can be initialized in the simulation. The individual tubes of a bundle cannot be represented individually since their dimensions are of smaller or similar scale as the lattice width $\Delta x^* = C_l$. This is the reason why the TBHEs are initialized as a porous media with given solid tube fraction s_t that partially stems the flow through them. The principle of the method is a partial or probabilistic bounce back of poststreamed density distribution function. The details of the used method are given by McCloskey/Dardis [30] and Sukop et al. [24]. As a first simple approach the identified nodes of the vertical TBHE(s) are set to a uniform chosen cooling or heating rate $\dot{q}_{TBHE}\left[\frac{W}{m^3}\right]$ that is determined by the set heat rate of a TBHE in the reactor $\dot{Q}_{TBHE}[W]$. An example of four THBEs with constant 90° circumferential shift is depicted in Figure 3-6.

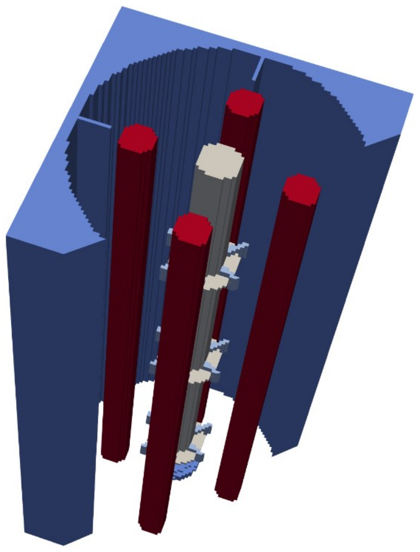


Figure 3-6: Initialization of four tube bundle heat exchanger (red) as set heat sources or sinks in the stirred tank bioreactor

3.2.1 Thermal Distribution Example TBHE

In the overview of Figure 3-7 an example of fluid flow and the respective thermal image is depicted for two TBHEs left and right of the stirrer. For each snapshot of the thermal state an image of the velocity magnitudes is given underneath. All sectional views of the stirred tank are given at a height of the second (middle) Rushton impeller $h_z = 0.46\ m$.

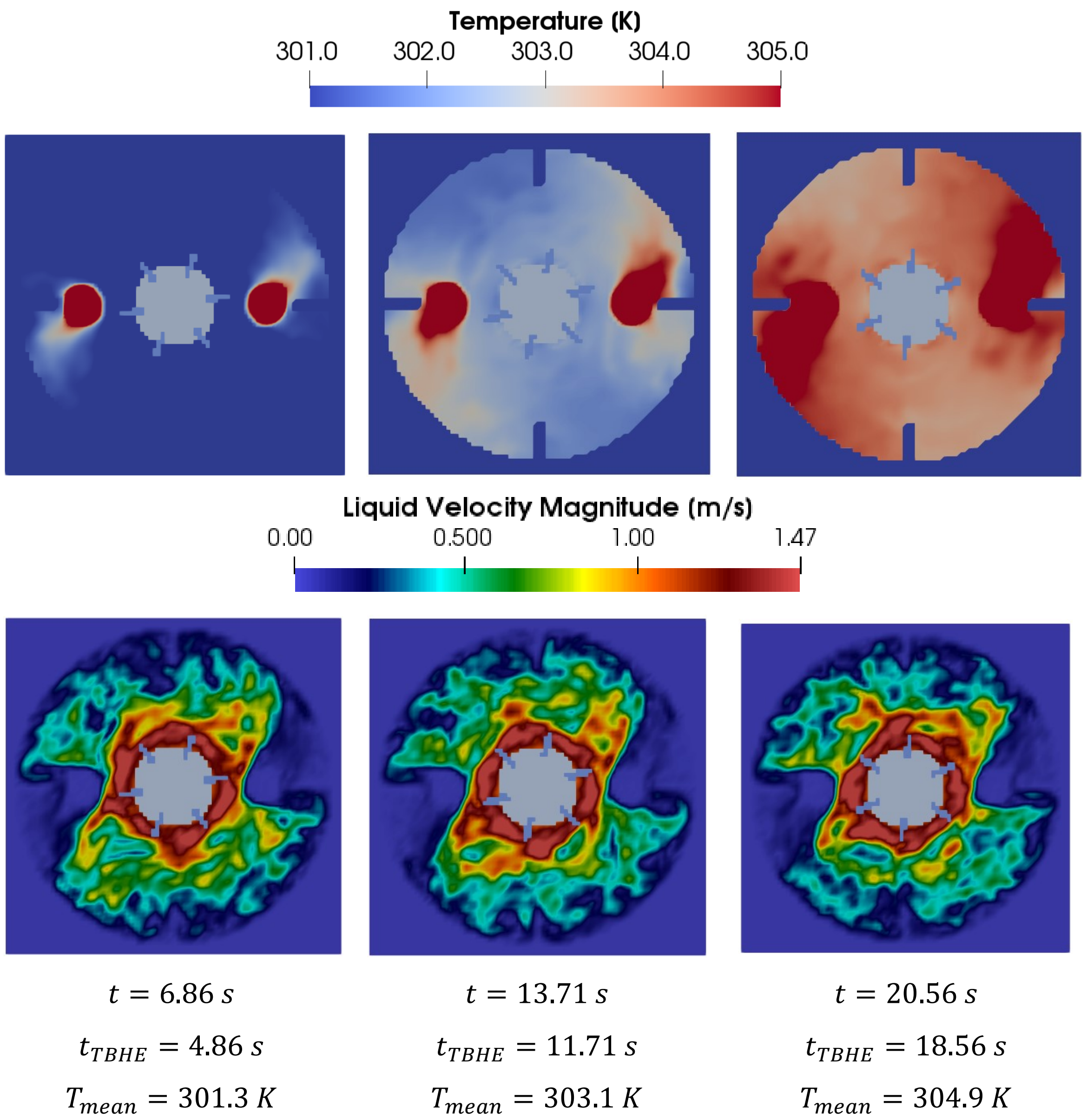


Figure 3-7: Example of a stirred tank with two initialized heat exchangers , significant input parameters: initial temperature $T_0 = 300K$, actual fluid volume $m_{act} = 142.1\ kg$, heat exchanger power $\dot{Q}_{HEs} = 2 \cdot 75\ kW$, TBHEs active after 2s, TBHE diameter $d_{TBHE} = 0.05\ m$, solid tube fraction $s_{TBHE} = 0.05$, tank diameter $d_R = 0.44\ m$, tank height $h_R = 1.015\ m$, kinematic viscosity $\nu = 1 \cdot 10^{-6} \frac{m^2}{s}$, stirrer speed $N_{stirr} = 190 \frac{rev}{min}$

3.3 Viscous Dissipation

The irreversible energy dissipation of kinetic fluid energy into heat caused by inner viscous friction of the fluid is in the field of fluid dynamics referred to as *viscous dissipation*. This depletion of kinetic energy occurs in all large and small vortexes (eddies) of the fluid field and is formulated in Section 1.3 *Turbulence Model*. The viscous dissipation that takes place in the small eddies below the resolved lattice width is considered in the applied Smagorinsky model by increasing the physical viscosity of the fluid ν by a turbulence viscosity ν_T. The resulting energy dissipation rate $\dot{\varepsilon}_d(\boldsymbol{x},t)\left[\frac{W}{kg}\right] = f(\boldsymbol{f_i},\nu,\nu_T,\boldsymbol{u})$ can be added to the overall heat source field $\dot{q}(\boldsymbol{x},t)\left[\frac{W}{m^3}\right] = \rho_0\dot{\varepsilon}_d(\boldsymbol{x},t)$.

3.3.1 Floating Point Error Handling

If considering small heat sources such as the viscous dissipation heat or heat produced by microorganisms in the thermal distribution simulation kernel, the following floating point error occurs. This subsection should clarify the approach to diminish this error.

The initialisation of internal energy in the system is given by $\varepsilon(\boldsymbol{x},t=0) = \int_{T_{zp}=273.15K}^{T_0} c_v^* dT^* \cdot \frac{C_l^2}{C_t^2}$ and is typically of order of magnitude $O\big(\varepsilon(\boldsymbol{x},t=0)\big) = 1 \div 3$. As stated in Eq. 2-12 the internal energy is updated every timestep of the LBM fluid field calculation with heat sources/sinks and the advection term. If heat sources $\frac{\dot{q}(x,t)}{\rho_0}$ at certain time t_o and position $\boldsymbol{x_o}$ are of order

$$O\left(\frac{\dot{q}(\boldsymbol{x_o},t_o)}{\rho_0}\right) \leq O\big(\varepsilon(\boldsymbol{x},t)\big) - 16 \qquad \textit{Eq. 3-3}$$

16 … significant numbers of double precision floating point numbers

and are added to the internal energy with *double precision,* the heat source energy $\frac{\dot{q}(\boldsymbol{x_o},t_o)}{\rho_0} \cdot \Delta t$ will actually not be added to the internal energy due to the limited accuracy of the floating point number. A straightforward solution to this floating point error problem would be the initiation of *quadruple* or *long double precision* for the internal energy ε. In the currently used GPU architecture there is no full *long double precision* support though. One typical handling to this problem is to add the floating point numbers in ascending order. The currently applied method follows an analogous principle: The heat sources are sorted in two classes by a order of magnitude threshold value th.

$$th = 10^{s+O(\varepsilon(\boldsymbol{x},t))-16} \qquad \textit{Eq. 3-4}$$

$s \ldots effective\ significant\ number\ of\ small\ class\ heat\ source$

$$\dot{q}_L(\boldsymbol{x},t) \geq th > \dot{q}_S(\boldsymbol{x},t) \qquad \textit{Eq. 3-5}$$

The heat source energy of the larger class $\frac{\dot{q}_L(\boldsymbol{x},t)}{\rho_0} \cdot \Delta t$ is considered to be large enough to be added to the internal energy of the respective node at each timestep of the LBM simulation without the floating point error. The number of effective significant digits of $\frac{\dot{q}_L(\boldsymbol{x},t)}{\rho_0}$ is at least $s+1$. The heat source of the smaller class is not added to the internal energy at each timestep of the LBM simulation but stored in a buffer and summed up for all timesteps until its sum for the given node $\boldsymbol{x}$ at time t is larger than the set threshold value th. If this is the case, the buffer value is added to the internal energy of the respective node and reset to zero. This approach ensures that the small heat sources $\dot{q}_S(\boldsymbol{x},t)$ are added to the internal energy with $s+1$ significant digits with the drawback that the individual heat sources that contribute to this buffer are released at a delayed instant of time.

3.4 Microorganisms

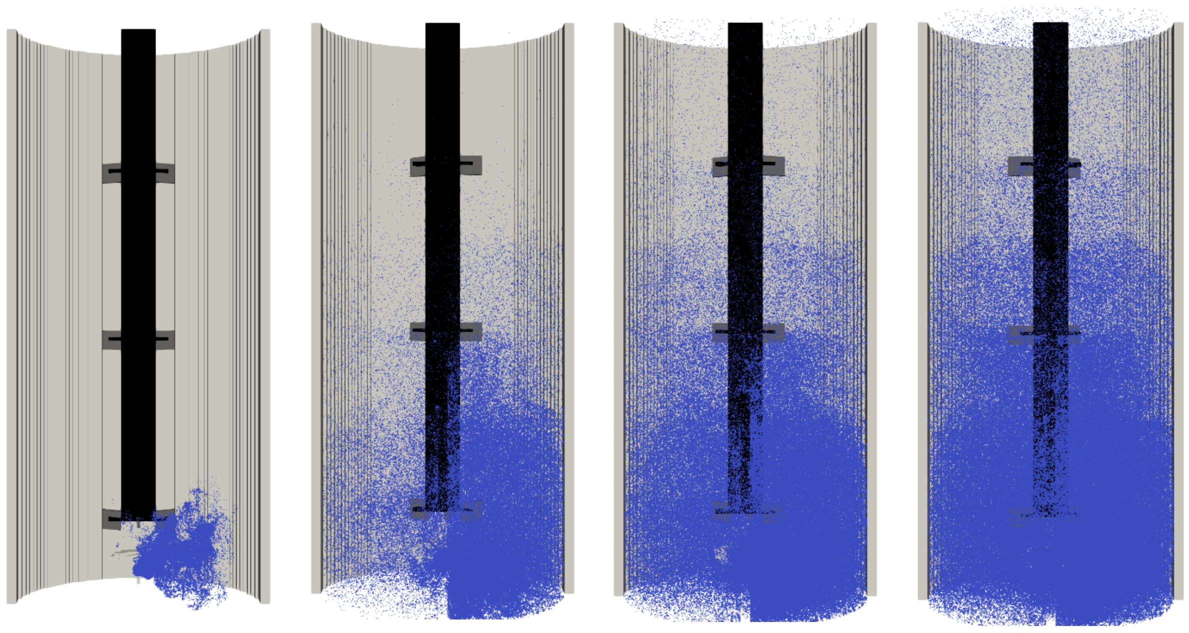
In the current simulation framework, the microorganisms in the liquid are represented by so called pellets. The number of pellets that can be simulated is of order 10^6 which means that this approach assumes that the number of pellets is sufficient to represent the transport of microorganisms in the reactor whose number is by far larger. The pellets are passively carried by the fluid flow field, which means that they do not affect the flow field by any force such as for example a drag force. Each individual initialized pellet is trackable due to the use of the *Lagrangian* method. Accordingly, the heat produced due to product generation or metabolism processes can as well be modelled for the representative tracked pellets and passed to the *Eulerian* field of heat sources $\dot{q}(\boldsymbol{x},t)$. So, at each timestep the pellets are located with Cartesian coordinates and their local heat power is passed to Eulerian calculation of temperature.

As a simple verification case all pellets in the system are initialized with a constant heat source power $P_p[W]$ over their total lifespan. In the discharge period the pellets are continuously inserted into the system at a certain rate until the set maximum number of pellets is reached (Figure 3-8). After this discharge period each pellet releases its heat source power $P_P[W]$. The occurrence of death or proliferation of microorganisms (pellets) is not considered. In Table 3-2 the key parameters of verification case are summarized.

Table 3-2: Set input parameter for the test cases with heat release by pellets (microorganisms), 1:due to staircase approximation of the circular cross-section of the cylindric reactor and installation volume (e.g. baffles) the actual mass is reduced

Physical Quantity	Case	Physical Quantity	Case
fluid volume V	$0.15433\ m^3$	spec. heat capacity c_v	$4000\ \frac{kg}{m^3}$
actual fluid volume[1] V_{act}	$0.14401\ m^3$	initial temperature T_0	$300\ K$
nodes per meter	179	pellet power $P_P[W]$	0.1
diameter d_R	$0.44\ m$	number of discharged pellets $N_{P,tot}$	500 000
stirrer speed N_{stirr}	$300\ \frac{rev}{min}$	actual discharge rate	$51870\ pellets/s$
density ρ	$1000\ \frac{kg}{m^3}$	actual discharge period $t_{d,tot}$	$9.64\ s$
kin. viscosity ν	$1 \cdot 10^{-6} \frac{m^2}{s}$	effective heat up period $t_{h,tot}$	$16.41\ s$
		total simulation time	$26.05\ s$

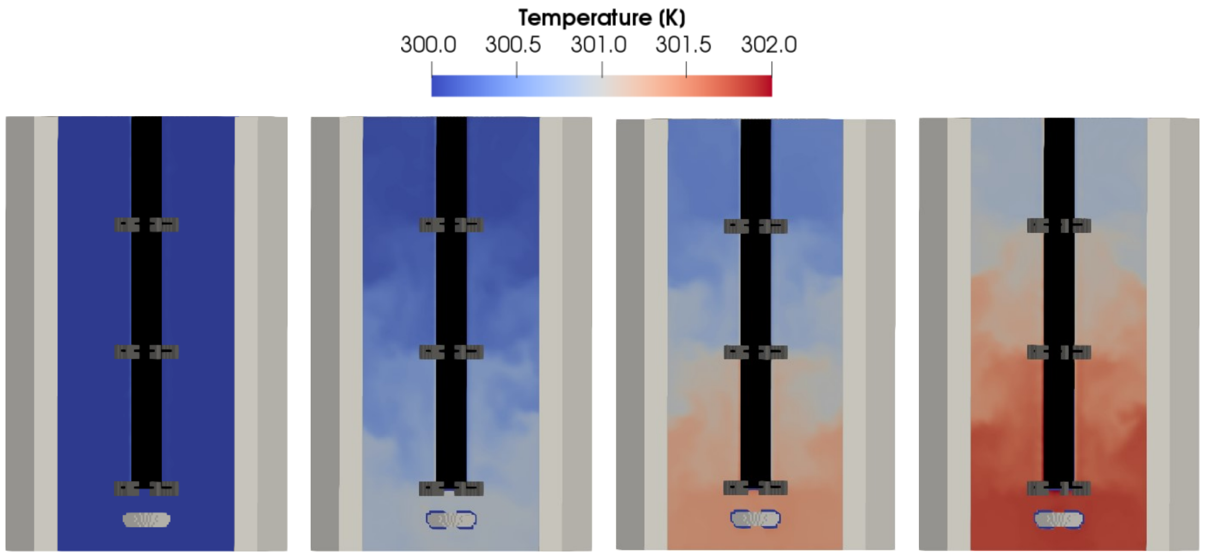
In the following overview (Figure 3-8) the discharge period is depicted: The pellets are released at a set point in the reactor and distributed by the turbulence caused by the stirrer blades. After the total amount of pellets of $N_{P,tot} = 500000$ is reached at $t = 9.64\ s$ each pellet releases a heat source power of $P_P = 0.1\ W$.



$t = 0.72\ s$ $N_P = 37610$ | $t = 3.62\ s$ $N_P = 187539$ | $t = 6.51\ s$ $N_P = 337655$ | $t = 9.41\ s$ $N_P = 487941$

Figure 3-8: In the discharge period (9.64 s) the pellets (blue) are continuously added to the bioreactor at a set position. The pellets are distributed in the reactor by the fluid flow.

The overview of Figure 3-9 depicts the heat up of the fluid caused by the pellets.



$t_h = 0.00\ s$ $t = 9.41\ s$ | $t_h = 4.83\ s$ $t = 14.47\ s$ | $t_h = 9.90\ s$ $t = 19.54\ s$ | $t_h = 14.96\ s$ $t = 24.60\ s$

Figure 3-9: Temperature distribution for a total heat source power of $P = N_{P,tot} \cdot P_P = 50000\ W$ *released by the microorganisms represented by pellets in the reactor.*

The energy that is given to the system can be verified by comparing the temperature of an ideal mixed system with same energy source $\mathrm{P} = \mathrm{N}_{\mathrm{P,tot}} \cdot \mathrm{P}_{\mathrm{P}}$ to the average temperature in the reactor (Figure 3-10).

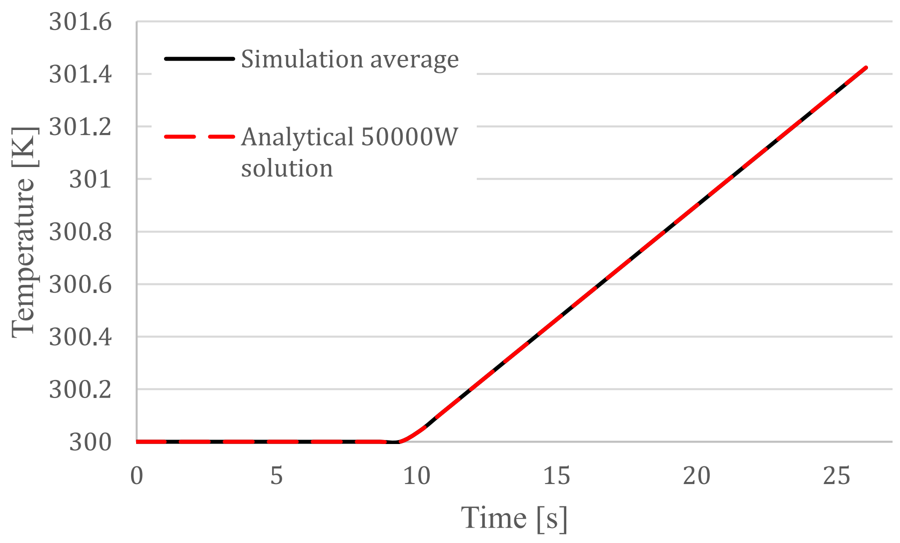


Figure 3-10: Analytical solution of ideal mixed temperature vs. average temperature in the reactor. The solution of the simulation seems to be in good agreement with the analytical solution.

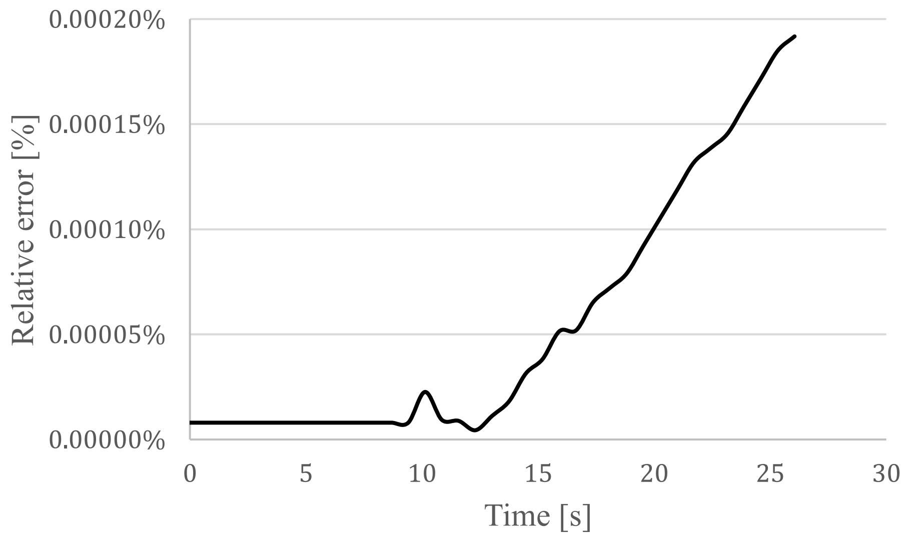


Figure 3-11: Relative error $rd = \frac{|T_{sim} - T_{analytical}|}{T_{analytical}}$ *over time. The linear error indicates an offset error in the heat source term.*

Figure 3-10 and Figure 3-11 show that the simulation slightly deviates from the correct solution. During the simulation test cases the Eulerian heat source field of the pellets $\dot{\boldsymbol{q}}_{\boldsymbol{P}}(\boldsymbol{x}, t)$ and the

Lagrangian pellet number were consistent. However, the overall heat source field $\dot{\boldsymbol{q}}(\boldsymbol{x}, t)$ deviated from the heat source field of the pellets $\dot{\boldsymbol{q}}_{\boldsymbol{P}}(\boldsymbol{x}, t)$ if solid nodes (e.g. tank wall or stirrer shaft) were excluded while passing the Eulerian data of $\dot{\boldsymbol{q}}_{\boldsymbol{P}}(\boldsymbol{x}, t)$ (see step 1) in Figure 3-1). This deviation suggests that single Pellets may enter the solid boundary for certain configurations for a short amount of time and is not caused by the thermal energy transport algorithm itself.

3.5 Fluid-Phase Coupling

3.5.1 Liquid Viscosity

For constant fluid density the kinematic viscosity is the only relevant factor except the given geometric boundaries and boundary velocities that influences the solution of the fluid flow field. In general, the viscosity of liquids can be dependent on shear rate, temperature and pressure.

$$\tau = \mu(p, T, \dot{\gamma}) \cdot \dot{\gamma} \text{ with}$$

$$\tau \ldots shear\ stress\ [Pa]$$

$$\mu \ldots dynamic\ viscosity\ [Pa \cdot s] \qquad \textit{Eq. 3-6}$$

$$\dot{\gamma} \ldots shear\ rate\ \left[\frac{1}{s}\right]$$

While the effect of pressure below $p < 1000\ bar$ is negligible [31] in a common stirred bioreactor with aqueous broth the effect of temperature is often prevailing. As an example, the viscosity of pure water at 20°C differs from the one at 50°C by more than 40% (see Figure 3-14). If the liquid broth in the bioreactor is a non-Newtonian fluid the local shear stress $\dot{\gamma}$ simultaneously affects the viscosity of the fluid which is common for aqueous broths with microorganisms (Figure 3-12 and Figure 3-13).

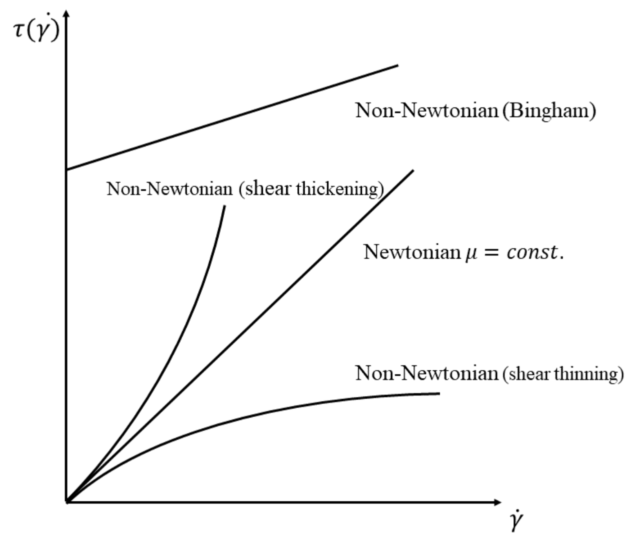


Figure 3-12: Shear stress over shear rate, for non-Newtonian fluids the viscosity is dependent on the local shear rate $\dot{\gamma}$

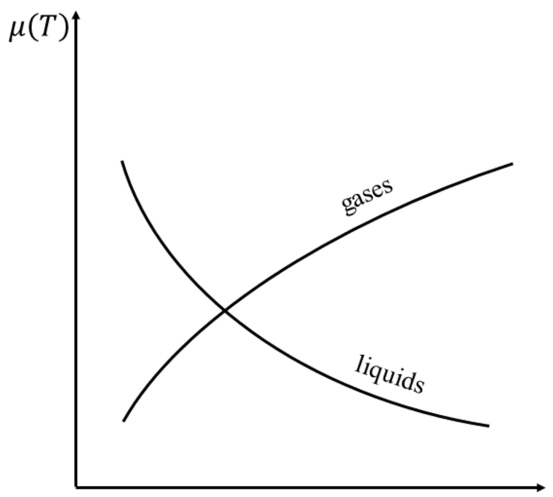


Figure 3-13: Temperature dependency of fluids, for liquids the viscosity decreases with higher temperature

The objective is to take the effect of local temperature and local shear stress on the liquid viscosity into account by coupling the solution of the temperature distribution with the fluid field calculation with a model.

3.5.2 Temperature and Shear Rate Models for Liquid Viscosity

Viscosity Model: Shear Rate

One common method is to assume a simple power law relation which covers the shear thickening or thinning behaviour of the non-Newtonian fluid.

$$\tau = \mu_0 \cdot \dot{\gamma}^n \quad \text{or} \quad \tau = \mu_{eff} \cdot \dot{\gamma} \\ \text{with} \quad \mu_{eff} = \mu_0 \dot{\gamma}^{n-1} \qquad \textit{Eq. 3-7}$$

For LBM based CFD the local shear rate $\dot{\gamma}$ is represented by the characteristic filtered rate of shear S and is calculated by the mean filtered momentum flux Q_{tot} (Eq. 1-23 and Eq. 3-8).

$$\dot{\gamma} \cong S = \frac{1}{2\,\rho_0\,\tau_{f*}\,c_s^2} \cdot Q_{tot} \qquad \textit{Eq. 3-8}$$

Viscosity Model: Temperature

The temperature dependency can be modelled by an additional factor $\mu(T,\dot{\gamma}) = \mu_0 \dot{\gamma}^{n-1} \cdot f(T)$ since temperature and shear rate influence the viscosity independently. Both the power law and the temperature function $f(T)$ have to be based on the same coefficient μ_0.

There is a variety of viscosity-temperature models for liquids which can be compared for pure water substance to the current benchmark model published by the *International Association for the Properties of Water and Steam* IAPWS revised in year 2008 [32]. For mixtures of chemical compounds physical and group contribution methods exist [33]. For pure liquids between freezing point and boiling temperature it is recommended to use empirical fitted *Arrhenius* like correlations [33] (Eq. 3-10 and Eq. 3-11). Moreover, a simple exponential decay ("Reynolds") approach is tested (Eq. 3-9). It has been shown that the choice between these models has great influence on the solution of the fluid field [34]. In Figure 3-14 and Figure 3-15 the performance of following models are compared to the IAPWS benchmark data for pure water.

Exponential: $\mu(T) = \mu_0 \cdot \exp(-BT) \quad or \quad \mu_0 \cdot 10\hat{}(-BT)$ *Eq. 3-9*

$$\mu(T[K]) = 0.62625 \cdot \exp(-0.02180\, T)\, Pa \cdot s$$

Andrade: $\mu(T) = \mu_0 \cdot \exp\left(\frac{B}{T}\right) \quad or \quad \mu_0 \cdot 10\hat{}\left(\frac{B}{T}\right)$ *Eq. 3-10*

$$\mu(T[K]) = 1.69132 \cdot 10^{-6} \cdot \exp\left(\frac{1877}{T}\right)\, Pa \cdot s$$

Vogel-Tamman-Fulcher: $\mu(T) = \mu_0 \cdot \exp\left(\frac{B}{T+C}\right) \quad or \quad \mu_0 \cdot 10\hat{}\left(\frac{B}{T+C}\right)$ *Eq. 3-11*

VTF Fit1 [35] $\mu(T[K]) = 2.414 \cdot 10^{-5} \cdot 10\hat{}\left(\frac{247.8}{T-140}\right)\, Pa \cdot s$

VTF Fit2 [36] $\mu(T[K]) = 2.4263 \cdot 10^{-5} \cdot \exp\left(\frac{578.919}{T-137.546}\right)\, Pa \cdot s$

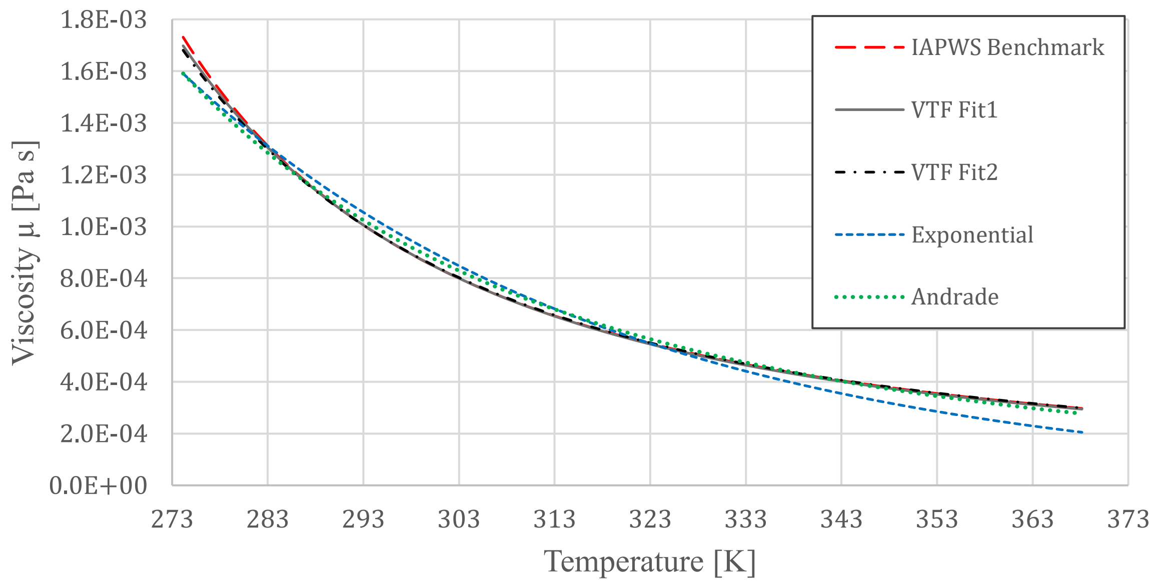


Figure 3-14: Empirical models vs. IAPWS benchmark model of dynamic viscosity of pure water

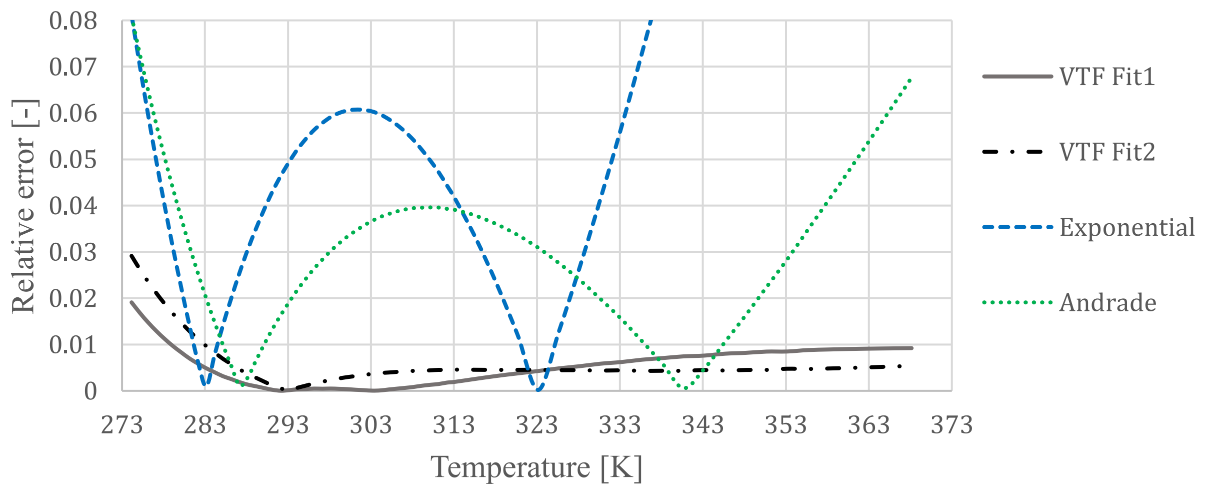


Figure 3-15: Relative error $\frac{|\mu_{IAPWS}-\mu_{model}|}{\mu_{IAPWS}}$ *of temperature-viscosity models compared to IAPWS standard*

Figure 3-14 and Figure 3-15 show that the exponential decay (Eq. 3-9) and the Andrade model (Eq. 3-10) deviate from the benchmark data by more than 8% for the temperature range of $T = [273K; 373K]$. Both parameter sets for the VTF model (Eq. 3-11) deviate by up to 3% for the same temperature range. For smaller range of $T = [283K; 373K]$ the relative error stays below 1%. The first parameter set of the VTF model (Figure 3-14: “VTF Fit1”) is a in a good agreement with the benchmark data and will be used as a model in the simulation code (Eq. 3-12).

$$\nu(\dot{\gamma}) = \frac{\mu_0 \dot{\gamma}^{n-1}}{\rho} \rightarrow \nu(\dot{\gamma}, T) = \frac{\mu_0 \dot{\gamma}^{n-1}}{\rho} \cdot \frac{A}{\mu_0} 10^{\frac{B}{T+C}} \qquad \textit{Eq. 3-12}$$

3.5.3 Verification: Viscosity Distribution for Spatial Linear Increase of Temperature

The implementation is tested for a linear increase of the temperature from the bottom to the top of the reactor from $T(z = 0) = 280K$ to $T(z = H_R) = 370K$. The implementation of temperature dependent kinematic viscosity calculation in the simulation can be verified (Figure 3-16 and Figure 3-17).

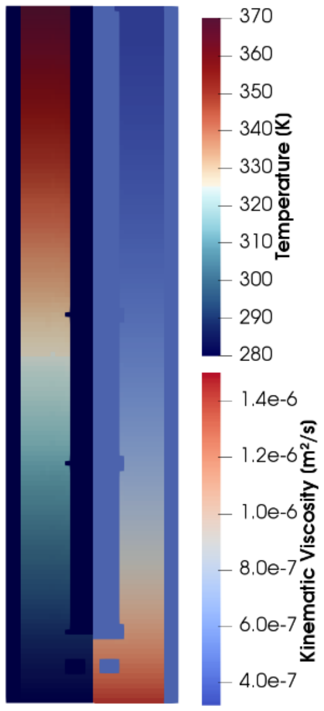


Figure 3-16: Reactor with constant spatial temperature gradient in z-coordinate, left: temperature, right: kinematic viscosity calculated with VTF model

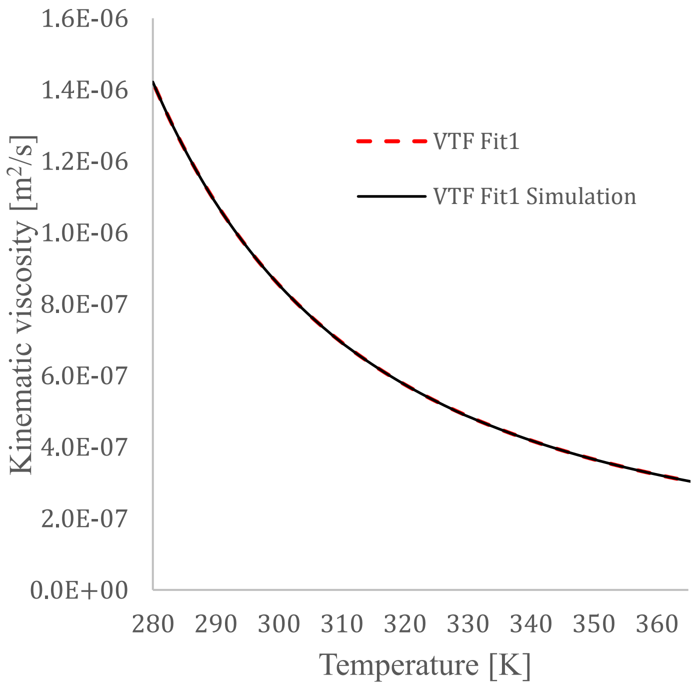


Figure 3-17: Simulation output of kinematic viscosity, the calculation can be verified

4 Summary and Outlook

The distribution of temperature in a stirred bioreactor can be described by solving the ADE of internal energy for the liquid phase. The *double distribution function* approach DDF proposed by Peng et al. in 2003 was chosen as a first common algorithm to solve the ADE of internal energy because of its fast computation and better stability compared to alternative algorithms given in the literature. For simple two dimensional problems such as the (thermal) Couette flow or lid driven cavity flow this algorithm provided reasonable results. The 2D domain algorithm using the CPU calculated the flow field and temperature distribution which were successfully verified by analytical or benchmark data. The ported 3D domain code based on parallel calculation on GPU was tested by a purely diffusive problem in resting water. The lower limit of the thermal diffusivity that led to a loss of stability was four orders of magnitude higher than the true thermal diffusivity of water. In addition to this limited stability the algorithm requires for complex boundary handling that often leads to unexpected energy loss in the system. In literature the use of multi relaxation time DDF or the use of varying boundary conditions to handle stability issues is proposed. The alternative advective transport algorithm for scalars based on LB methods proposed by Osmanlic et al. in 2016 is a stable and conservative method that does not require complex boundary condition handling. The method is based on the balance of the scalar transport between the cell and neighboring cells by advection. This method was investigated by a simple advection of a 1D Gauss distribution in the 3D computational domain. The algorithm led to an exact transport of the peak of the distribution without numerical oscillations and with conservation of energy. The numerical diffusion of the simulation perfectly agrees with a 1D analytical derivation of the numerical diffusivity that depends on the velocity of the fluid in lattice units. The thermal diffusivity for a hot bulk that is advected without turbulence in a certain spatial direction was derived for the 3D case. This derivation was tested in simulations by the advection of hot liquid bulks and can be perfectly verified when the direction of the numerical diffusion aligns with the lattice of the domain. If the direction of the diffusion is across the lattice, the diffusion is smaller than the analytical solution which can be mostly recovered by geometric correction factors. A comparison of the numerical diffusion and the true physical thermal diffusivity of water shows that for the currently applied range of lattice unit conversion factors and Mach number the numerical diffusion is at least three orders of magnitude higher than the true thermal diffusivity if the velocity in lattice units equals the stirrer’s blade tip speed. At 0.01%-0.1% of the stirrer’s blade tip speed the numerical diffusivity is in the range of the thermal diffusivity. If the fluid is steady the numerical diffusion is absent.

For the application of the transport algorithm for stirred industrial bioreactor a heating jacket with variable height, vertical position, heating time and heating power can be initialized. Moreover, the already initialized tube bundle heat exchanger (TBHE) can release a constant heat power as a spatially uniform heat source for the identified TBHE cells. The viscous dissipation in the fluid due to the shear stress caused by the rotating stirrer is calculated in the LES turbulence modeling process and coupled to the thermal simulation unit. Pellets that represent the microorganisms in the system are initialized and attributed with a heat power whose evolution over time can be determined for each individual pellet via the Lagrange approach and passed as Eulerian data to the thermal simulation unit. The simulation of the temperature distribution was successfully coupled to the calculation of the fluid field by making the viscosity a function of temperature. For this the Vogel-Tamman-Fulcher viscosity model was chosen as an empirical model that is able to correctly recover the benchmark data given by the IAWPS for the correlation of viscosity and temperature for pure water substance.

For the further development of the thermal simulation unit for industrial stirred bioreactors the integration of biokinetic reaction models that estimate the heat generated by microorganisms is necessary. The Lagrangian temperature, shear rate, oxygen and substrate data can be recorded for each simulated parcel of microorganisms over time and passed as input parameter to the reaction models. The implementation of technical cooling or heating units such as heating jackets does not yet make use of locally modelled heat sources. For the modeling of these transient and locally resolved heat sources or sinks the local temperature differences and local fluid velocities can be made use of with forced convection heat transfer models. Moreover, additional features such as the cooling of the liquid by heat of evaporation due to humidification of the gas bubbles or by heat up of the gas bubbles can be taken into account by kinetic models. The given temperature distribution can be used for accurate description of oxygen solubility in the broth and oxygen transfer kinetics.

5 List of Figures

Figure 1-1: A) D2Q9 - a quadratic cell with center node and 9 discrete velocity vectors ei B) D3Q19 – a cubic cell with 19 discrete velocity vectors ei either pointing to the center of a side or to the center of an edge 5

Figure 1-2: D2Q9 - Scheme of streaming step of a post collision $f5x, t$-value (arrow with solid line), streamed $f5x, t$-value equals post streaming $f5x + e5\Delta t, t + \Delta t$-value (arrow with dashed line) 7

Figure 1-3: left) isotropic turbulence – the depicted flow field is independent from any direction or macroscopic geometries, right) anisotropic turbulence – the depicted flow field shows large-scale vortexes (eddies) which are dependent on the upward direction and are influenced by the macroscopic oblong geometry [6] 10

Figure 1-4: left) top view of the stirred tank reactor with Rushton stirrer, right) isometric front view, blue) steady solid parts of reactor, red) steady solid parts of stirrer, grey) moving solid parts of stirrer 1) wall of the reactor, 2) baffles, 3) shaft of the stirrer with rotating disks, 4) blades of the stirrer, 5) gas sparger 14

Figure 2-1: Example of g-distribution set after streaming step, the 'fluid boundary nodes' are missing g-values (dashed arrows), all streamed values (solid arrows) were streamed from neighboring fluid nodes, missing g-values must be calculated by boundary condition 19

Figure 2-2: Scheme of the thermal Couette flow experiment, the hot top wall is constantly moving, the temperature difference between hot top wall and cool bottom wall is constant over time 21

Figure 2-3: Thermal Couette flow field and temperature distribution for Re=1, Pr=1, nodes in y-coordinate: 100, temperature and velocity profiles after t=0.15s, t=0.30s and t=0.45s, comparison to transient solution of the thermal Couette flow 22

Figure 2-4: Thermal Couette flow field and temperature distribution for Re=1, Pr=1, nodes in y-coordinate: 100, temperature and velocity profiles after t=1.5s, t=2.25s, t=3.0s, t=3.75s and t=4.5s, comparison to steady state solution of the thermal Couette flow 23

Figure 2-5: Square scope from 0.4 to 0.6 of Figure 2-4...23

Figure 2-6: Scheme of the lid driven cavity experiment, the top wall (wall) is constantly moving, the three remaining walls are stationary 24

Figure 2-7: Vector plot of the lid driven cavity flow problem (left) and streamline plot (right), the vortex center location $(xvc\,, yvc) = (0.5, 0.75)$ agrees with the collection of literate data for $Re \leq 1$ [25], numerical artifacts remain at the upper corner where bounce back and Zhou & He BC are adjacent 25

Figure 2-8: X-component velocity u plotted along the height of the cavity at the horizontal middle of the cavity 25

Figure 2-9: Initial setup of the verification case, the boundary nodes of the cylinder are set to a constant temperature of 300K ($\varepsilon = 1.2 \cdot 106Jkg, cp = 4000Jkg \cdot K$),the bulk nodes are set to a temperature of 0 K, the heat diffusion is investigated along a plot axis (white arrow in z-coordinate), diameter: 1 m, height: 0.332 m, 100 nodes per meter 26

Figure 2-10: $t1 = 250,\ t2 = 500,\ t3 = 750,\ t4 = 1000\ timesteps$, the algorithm shows a stable evolution of the heat distribution in the system..................................27

Figure 2-11: the stability limit is reached – a closer view of the data reveals slight instabilities (see Figure 2-12) 27

Figure 2-12: Nonphysical results of the method – towards the middle of the cylinder the temperature rises, although the heat gradient should be negative towards the middle (hotter wall, cold bulk) 28

Figure 2-13: The stability limit is exceeded – the temperature starts to oscillate and reaches huge values after $t4 = 1000\ timesteps$ 28

Figure 2-14: A 1D Gauss distribution (amplitude of 1, standard deviation of 50 and mean value of 500) is advected by a constant fluid velocity of $uz = 0.01$ for 50000 timesteps. The dashed curve shows the Gauss distribution after the simulation which has a reduced amplitude of 0.9136 and an increased standard deviation due to numerical diffusion...32

Figure 2-15: Comparison between physical and numerical diffusion of the Osmanlic transport algorithm. The physical diffusion coefficient is given by $\alpha LU = CtPr \cdot Cl2 \cdot \nu * m2s$ which is for water in a plausible range of $\alpha minLU = 0.1 \cdot 10 - 6, \alpha max\ LU\ = 2 \cdot 10 - 6$. For reasons of simplicity of the comparison the y and z

component velocity are given by the x component velocity $uy = 23ux, uz = 13ux$. The x-axis of the plot is given by the magnitude of the velocity vector. 35
Figure 2-16: Initial setup for investigation of 1D numerical diffusion in z-axis, fluid flow towards x-axis (aligned with lattice structure) 37
Figure 2-17: Initial setup (set image threshold at $\varepsilon = 1.0$) for investigation of 2D numerical diffusion normal to plane E_{2D}, fluid flow towards the square diagonal $(1, 1, 0)$....37
Figure 2-18: Initial setup (set image threshold at $\varepsilon = 1.0$) for investigation of 3D numerical diffusion normal to plane E_{3D}, fluid flow towards the space diagonal $(1, 1, 1)$....37
Figure 2-19: Numerical diffusion visible after 50000 timesteps at initial step in the middle of the domain (z=100 nodes)....37
Figure 2-20: Numerical diffusion visible after 9000 timesteps (set image threshold at $\varepsilon = 0.3$), diffusion is observed along the normal vector of the plane (white line normal to plane E_{2D})....37
Figure 2-21: Numerical diffusion visible after 9000 timesteps (set image threshold at $\varepsilon = 0.3$), diffusion is observed along the normal vector of the plane (white line normal to plane E_{3D})....37
Figure 2-22: 1D 39
Figure 2-23: 2D diffusion normal to $E2D$ 39
Figure 2-24: 3D diffusion normal to $E3D$ 39
Figure 2-25:2D diffusion in x-coordinate 39
Figure 2-26: 3D diffusion in x-coordinate 39
Figure 2-27: Numerical Diffusion for 2D Case at 7000 timesteps, the numerical diffusion can be determined for diffusions along the cubic lattice structure, the diffusion across the lattice structure (e.g. square diagonal) is reduced....40
Figure 2-28: Diffusion that aligns with the lattice structure is represented by the vertical or horizontal black arrows, the diffusion across the lattice structure in square diagonal direction does only result from diffusion that aligns with lattice structure in x and y coordinate....41

Figure 2-29: 2D numerical diffusion across lattice structure compared to analytical solution with corrected effective diffusion coefficient 42

Figure 2-30: 3D numerical diffusion across lattice structure compared to analytical solution with corrected effective diffusion coefficient 42

Figure 3-1: Overview of the thermal distribution calculation, solid lines represent implemented features, dashed lines represent possible coupled models 43

Figure 3-2: Simulation setup with active heating jacket, red cells represent the heat jacket nodes with heat source $qHJWm3$ that form a ring with set vertical position and height. The nodes are in contact with the tank wall (blue cells). 44

Figure 3-3: Heating curves of both verification cases with constant heating power over set time compared to the respective analytical solution 46

Figure 3-4: Absolute difference of the analytic and the simulated temperature for both verification cases 1 and 2 with constant heating power over time 47

Figure 3-5: Example of a stirred tank with initialized heating jacket , significant input parameters: initial temperature $T0 = 300K$, heating jacket power $QHJ = 2000\ kW$, actual fluid volume $mact = 142.1\ kg$, $hHJ, uL = 0.761\ m, hHJ, lL = 0.254\ m$, diameter $dR = 0.44\ m$, tank height $hR = 1.015\ m$, kinematic viscosity $v = 1 \cdot 10 - 6m2s$, stirrer speed $Nstirr = 190revmin$ 48

Figure 3-6: Initialization of four tube bundle heat exchanger (red) as set heat sources or sinks in the stirred tank bioreactor..49

Figure 3-7: Example of a stirred tank with two initialized heat exchangers , significant input parameters: initial temperature $T0 = 300K$, actual fluid volume $mact = 142.1\ kg$, heat exchanger power $QHEs = 2 \cdot 75\ kW$, TBHEs active after 2s, TBHE diameter $dTBHE = 0.05\ m$, solid tube fraction $sTBHE = 0.05$, tank diameter $dR = 0.44\ m$, tank height $hR = 1.015\ m$, kinematic viscosity $v = 1 \cdot 10 - 6m2s$, stirrer speed $Nstirr = 190revmin$ 50

Figure 3-8: In the discharge period (9.64 s) the pellets (blue) are continuously added to the bioreactor at a set position. The pellets are distributed in the reactor by the fluid flow. 54

Figure 3-9: Temperature distribution for a total heat source power of $P = NP, tot \cdot PP = 50000\ W$ released by the microorganisms represented by pellets in the reactor. 54

Figure 3-10: Analytical solution of ideal mixed temperature vs. average temperature in the reactor. The solution of the simulation seems to be in good agreement with the analytical solution. 55

Figure 3-11: Relative error $rd = Tsim - TanalyticalTanalytical$ over time. The linear error indicates an offset error in the heat source term. 55

Figure 3-12: Shear stress over shear rate, for non-Newtonian fluids the viscosity is dependent on the local shear rate γ 57

Figure 3-13: Temperature dependency of fluids, for liquids the viscosity decreases with higher temperature 57

Figure 3-14: Empirical models vs. IAPWS benchmark model of dynamic viscosity of pure water 59

Figure 3-15: Relative error $\mu IAPWS - \mu model \mu IAPWS$ of temperature-viscosity models compared to IAPWS standard 59

Figure 3-16: Reactor with constant spatial temperature gradient in z-coordinate, left: temperature, right: kinematic viscosity calculated with VTF model 60

Figure 3-17: Simulation output of kinematic viscosity, the calculation can be verified ...60

6 List of Tables

Table 1-1: discrete velocity set (D3Q19) – the lattice velocity vector ei for the respective index i..5

Table 1-2: weights wi for the respective index i (D3Q19) 6

Table 2-1: Input and simulation parameter of the lid driven cavity flow simulation.......24

Table 2-2: Calculation of the overall integral for different timesteps. The integral provides a measure for the total energy in the system. The value of the integral stays about the same. The error may be caused by numerical integration (trapezoidal rule, 1000 increments from z=0 to z=1500) 32

Table 3-1: Set input parameter for the test cases with constant heat jacket power, 1:due to staircase approximation of the circular cross-section of the cylindric reactor and installation volume (e.g. baffles) the actual mass is reduced 46

Table 3-2: Set input parameter for the test cases with heat release by pellets (microorganisms), 1:due to staircase approximation of the circular cross-section of the cylindric reactor and installation volume (e.g. baffles) the actual mass is reduced 53

7 Literature

[1] T. Krüger, H. Kusumaatmaja, O. Shardt, G. Silva, A. Kuzmin, and E. M. Viggen, *The Lattice Boltzmann Method: Principles and Practice*. 2017.

[2] A. Mohamad, *Lattice Boltzmann Method: Fundamentals and Engineering Applications with Computer Codes*. 2011.

[3] E. . Viggen, *The lattice Boltzmann method: Fundamentals and acoustics*, no. February. 2014.

[4] P. L. Bhatnagar, E. P. Gross, and M. Krook, “A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems,” *Phys. Rev.*, vol. 94, no. 3, pp. 511–525, 1954.

[5] J. Latt, “Choice of units in lattice Boltzmann simulations,” *Lattice Boltzmann How tos*, no. April, pp. 1–6, 2008.

[6] H. Örtel, M. Böhle, and T. Reviol, “Grundgleichungen der Strömungsmechanik,” in *Strömungsmechanik für Ingenieure und Naturwissenschaftler*, 2015.

[7] E. Paul, V. Atiemo-Obeng, and S. Kresta, *Handbook of Industrial Mixing*. 2004.

[8] J. Smagorinsky, “General Circulation Experiments with the Primitive Equations,” 1963.

[9] C. Witz, D. Treffer, T. Hardiman, and J. Khinast, “Local gas holdup simulation and validation of industrial-scale aerated bioreactors,” *Chem. Eng. Sci.*, vol. 152, pp. 636–648, 2016.

[10] H. Yu, S. S. Girimaji, and L. S. Luo, “DNS and LES of decaying isotropic turbulence with and without frame rotation using lattice Boltzmann method,” *J. Comput. Phys.*, vol. 209, no. 2, pp. 599–616, 2005.

[11] J. Wendt and J. Anderson, “Governing Equations of Fluid Dynamics,” in *Computational Fluid Dynamics*, 2009.

[12] Z. Guo, “LBE for Low Speed Flows with Heat Transfer,” in *Lattice Boltzmann Method and Its Applications in Engineering*, vol. 3, 2013, pp. 145–196.

[13] F. M. Elseid, S. W. J. Welch, and K. N. Premnath, “A cascaded lattice Boltzmann model for thermal convective flows with local heat sources,” *Int. J. Heat Fluid Flow*, vol. 70, no. February, pp. 279–298, 2018.

[14] E. Attar and C. Körner, “Lattice Boltzmann model for thermal free surface flows with liquid-solid phase transition,” *Int. J. Heat Fluid Flow*, vol. 32, no. 1, pp. 156–163, 2011.

[15] T. Zhang, B. Shi, Z. Guo, Z. Chai, and J. Lu, “General bounce-back scheme for concentration boundary condition in the lattice-Boltzmann method,” *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 85, no. 1, pp. 1–14, 2012.

[16] S. Karimi and K. B. Nakshatrala, “Do Current Lattice Boltzmann Methods for Diffusion and Advection-Diffusion Equations Respect Maximum Principle and the Non-Negative Constraint?,” *Commun. Comput. Phys.*, vol. 20, no. 2, pp. 374–404, 2016.

[17] S. Bartlett, “A Non-Isothermal Chemical Lattice Boltzmann Model Incorporating Thermal Reaction Kinetics and Enthalpy Changes,” *Computation*, vol. 5, no. 3, p. 37, 2017.

[18] X. He, S. Chen, and G. D. Doolen, “A Novel Thermal Model for the Lattice Boltzmann Method in Incompressible Limit,” *J. Comput. Phys.*, vol. 146, no. 1, pp. 282–300, 1998.

[19] Y. Peng, C. Shu, and Y. T. Chew, “Simplified thermal lattice Boltzmann model for incompressible thermal flows,” *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.*, vol. 68, no. 2, p. 8, 2003.

[20] Y. Peng, C. Shu, and Y. T. Chew, “A 3D incompressible thermal lattice Boltzmann model and its application to simulate natural convection in a cubic cavity,” *J. Comput. Phys.*, vol. 193, no. 1, pp. 260–274, 2003.

[21] T. M. Liou and C. S. Wang, “Large eddy simulation of rotating turbulent flows and heat transfer by the lattice Boltzmann method,” *Phys. Fluids*, vol. 30, no. 1, 2018.

[22] L. Li, R. Mei, and J. F. Klausner, “Boundary conditions for thermal lattice Boltzmann equation method,” *J. Comput. Phys.*, vol. 237, no. March, pp. 366–395, 2013.

[23] C. H. Liu, K. H. Lin, H. C. Mai, and C. A. Lin, “Thermal boundary conditions for thermal

lattice Boltzmann simulations," *Comput. Math. with Appl.*, vol. 59, no. 7, pp. 2178–2193, 2010.

[24] M. C. Sukop and D. T. Thorne, *Lattice Boltzmann Modeling*, vol. 79, no. 1 Pt 2. 2006.

[25] M. Aydin and R. T. Fenner, "Boundary element analysis of driven cavity flow for low and moderate Reynolds numbers," *Int. J. Numer. Methods Fluids*, vol. 37, no. 1, pp. 45–64, 2001.

[26] I. Ginzburg, "Multiple anisotropic collisions for advection-diffusion Lattice Boltzmann schemes," *Adv. Water Resour.*, vol. 51, pp. 381–404, 2013.

[27] Z. Chen, C. Shu, and D. Tan, "A Truly Second-Order and Unconditionally Stable Thermal Lattice Boltzmann Method," *Appl. Sci.*, vol. 7, no. 3, p. 277, 2017.

[28] F. Osmanlic and C. Körner, "Lattice Boltzmann method for Oldroyd-B fluids," *Comput. Fluids*, vol. 124, pp. 190–196, 2016.

[29] V. E. Küng, F. Osmanlic, M. Markl, and C. Körner, "Comparison of passive scalar transport models coupled with the Lattice Boltzmann method," *Comput. Math. with Appl.*, pp. 1–11, 2018.

[30] O. Dardis and J. McCloskey, "Permeability porosity relationships from numerical simulations of fluid flow," *Geophys. Res. Lett.*, vol. 25, no. 9, pp. 1471–1474, 1998.

[31] H. Siegloch, *Technische Fluidmechanik, 7. Auflage*. Springer-Verlag Berlin Heidelberg, 2009.

[32] M. L. Huber *et al.*, "New international formulation for the viscosity of H2O," *J. Phys. Chem. Ref. Data*, vol. 38, no. 2, pp. 101–125, 2009.

[33] B. E. Poling, J. M. Prausnitz, and J. P. O'Conell, *The Properties of Gases and Liquids*. 2000.

[34] P. Mora and D. A Yuen, "Comparison of convection for Reynolds and Arrhenius temperature dependent viscosities," *Fluid Mech. Res. Int. J.*, vol. 2, no. 3, pp. 99–107, 2018.

[35] T. Al-Shemmeri, *Engineering Fluid Mechanics*. 2012.

[36] “Liquid Dynamic Viscosity - Calculation by Vogel Equation.” [Online]. Available: http://ddbonline.ddbst.de/VogelCalculation/VogelCalculationCGI.exe?component=Water. [Accessed: 22-Feb-2019].

8 Appendix

8.1 MATLAB Source Code of Double Distribution Function Thermal LBM D2Q9: Square Cavity Problem

8.1.1 Main Script

```
%------------------------------
%------------------------------
% Thermal Square Simulation,  D2Q9 LBM DDF SRT, internal energy formulation
% Simon Rustige
% May 2018, IPPT, TU Graz
%------------------------------
%------------------------------

% 01.) calc equilibrium distributions
% 02.) calc f post collision
% 03.) calc f post stream
% 04.) add boundary conditions
% 05.) calc density
% 06.) calc velocities in x-axis and y-axis
% 07.) calc g equilibrium distributions
% 08.) calc g post collision
% 09.) calc g post stream
% 10.) add  g boundary conditions
% 11.) calc internal energy in x-axis and y-axis
%------------------------------

% clc
% close all
clear variables



%Definitions
Ny=MC.Ny;
Nx=MC.Nx;
uw=MC.uw;
timesteps=MC.timesteps;
mytitle=['tsim(sec)=' int2str(MC.t_sim) ' - iter=' int2str(timesteps) ' - grid y=' int2str(Ny) ',
x=' int2str(Nx) ' - Re=' int2str(MC.Re) ' - Pr=' int2str(MC.Pr)];
t_Start=tic; % to optain elapsed time in script, see toc command

%% ------------------------------
% ----Init-----%
% -----------------------------

% x ... counter in x-axis 1,2,3 ... Nx
% y ... counter in y-axis 1,2,3 ... Ny
% i ... counter in discrete directions of lattice 1,2,3 ... 9 for D2Q9

%-----------------MassFlow---------------------
% u, v, rho
u=zeros(Nx,Ny);
v=zeros(Nx,Ny);
rho=ones(Nx,Ny);

% Init feq(u,v)
feq=zeros(Nx,Ny,MC.Dir);
for x=1:Nx
    for y=1:Ny
        for i=1:MC.Dir
            feq(x,y,i)=fun_feq(u(x,y),v(x,y),rho(x,y),i);
        end
    end
end

% Init f
```

```
f=feq;

%-----------------ThermalFlow---------------------
% epsilon (internal energy)
eps=MC.eps_init.* ones(MC.Nx,MC.Ny);
eps(1:round(Nx/2),1:round(Ny/2))=MC.eps_init*1.2;
% % eps(round(Nx/2)-round(Nx/4):round(Nx/2)+round(Nx/4),1:round(Ny/2))=MC.eps_init/2;
% eps(1:round(Nx/2),:)=MC.eps_init/2;
Tmean_init=mean(mean(eps))  *MC.C_x^2 ./ MC.C_t^2/MC.cv_real ;


%colormap for initial setup
figure();
imagesc([0 1],[0 1],rot90(fliplr(eps.*MC.C_x^2 ./ MC.C_t^2./MC.cv_real)));
set(gca,'YDir','normal')
colorbar
title(['INITIAL SETUP: ' mytitle],'Fontsize',12,'Interpreter','latex');
xlabel(gca,' $x$' ,'Fontsize',12,'Interpreter','latex');
ylabel(gca,' $y$ ' ,'Fontsize',12,'Interpreter','latex');


% Init geq(eps,u,v)
geq=zeros(Nx,Ny,MC.Dir);
for x=1:Nx
    for y=1:Ny
        for i=1:MC.Dir
            geq(x,y,i)=fun_geq(eps(x,y),u(x,y),v(x,y),rho(x,y),i);
        end
    end
end


%Init g
g=geq;


%% ------------------------------
% ----MainLoop-----%
%------------------------------

fig_snap=figure('Name','u and eps profiles'); hold all

snapinterval=0.05; %sec ...interval for plots in mainloop (simulated time)
pic=1:snapinterval/MC.C_t:timesteps; % snapshots (plots) for different timestep with constant
time intervalls


for timestep=1:timesteps
%     pause()
%     eps
%     var(eps)
%     timestep

    % Plot u and eps-profiles for different timesteps
    if (any(round(pic)==timestep))
        figure(fig_snap)
        plot(u(round(Nx/2),:)./uw,  0:1/(Ny-1):1)
        plot(eps(round(Nx/2),:)./MC.eps_ref, 0:1/(Ny-1):1,':')
        timestep
    end

    %-----------------MassFlow---------------------
    % --Calc feq(u,v)--
    for x=1:Nx
        for y=1:Ny
            for i=1:MC.Dir
                feq(x,y,i)=fun_feq(u(x,y),v(x,y),rho(x,y),i);
            end
        end
    end

    % --Collision--
    for x=1:Nx
```

```
        for y=1:Ny
            for i=1:MC.Dir
                f(x,y,i)=fun_coll(feq(x,y,i),f(x,y,i),MC.tau);
            end
        end
    end


    % --Streaming--
    f_prestream=f; % save old values (before streaming) for boundary conditions
    f=fun_stream(f_prestream); %boundary nodes x=1 and x=Nx, y=1 and y=Ny are "missing" (not
streamed) 3 or 5 (corner) f-values

    % --Boundary Conditions--

    % Calculate nodes that are missing
    %        C7  C3  C6       ^ y
    %          \ | /          |
    %        C4-C1-C2         |
    %          / | \          |
    %        C8  C5  C9         -----> x

    % bottom: Bounce Back BC
    f(:,1,3)=f_prestream(:,1,5);
    f(:,1,6)=f_prestream(:,1,8);
    f(:,1,7)=f_prestream(:,1,9);

    % left side: Bounce Back BC
    f(1,:,2)=f_prestream(1,:,4);
    f(1,:,6)=f_prestream(1,:,8);
    f(1,:,9)=f_prestream(1,:,7);

    %right side: Bounce Back BC
    f(Nx,:,4)=f_prestream(Nx,:,2);
    f(Nx,:,7)=f_prestream(Nx,:,9);
    f(Nx,:,8)=f_prestream(Nx,:,6);

%      %top side: Bounce Back BC
%     f(:,Ny,5)=f_prestream(:,Ny,3);
%     f(:,Ny,8)=f_prestream(:,Ny,6);
%     f(:,Ny,9)=f_prestream(:,Ny,7);

     % top: Zhou He BC: u=uw, v=0,rho=sum(f), f3_neq=f5_neq -> solve rho, f5, f8, f9
     for x=1:Nx
          if x==1
              %remain BB for corners
              f(x,Ny,5)=f_prestream(x,Ny,3);
              f(x,Ny,8)=f_prestream(x,Ny,6);
              f(x,Ny,9)=f_prestream(x,Ny,7);
          elseif x==Nx
              f(x,Ny,5)=f_prestream(x,Ny,3);
              f(x,Ny,8)=f_prestream(x,Ny,6);
              f(x,Ny,9)=f_prestream(x,Ny,7);
          else
             Zhou He
             f(x,Ny,9)=f_prestream(x,Ny,7);
             rho(x,Ny)=f(x,Ny,1) + f(x,Ny,4) + f(x,Ny,2) + 2 .* ( f(x,Ny,3) + f(x,Ny,6) +
f(x,Ny,7) );
             f(x,Ny,5)=f(x,Ny,3);
             f(x,Ny,8)=f(x,Ny,6) + 0.5.*(f(x,Ny,2)-f(x,Ny,4)) -  0.5.* rho(x,Ny) .* uw ;
             f(x,Ny,9)=f(x,Ny,7) + 0.5.*(f(x,Ny,4)-f(x,Ny,2)) +  0.5.* rho(x,Ny) .* uw ;
          end
     end


    % --Calculate density--
    rho=sum(f,3);


    % --Calculate u,v --
    for x=1:Nx
        for y=1:Ny
            sum_u=0;
            sum_v=0;
            for i=1:MC.Dir
                sum_u=sum_u + MC.cx(i)*f(x,y,i);
                sum_v=sum_v + MC.cy(i)*f(x,y,i);
            end
            u(x,y)=sum_u/rho(x,y);
            v(x,y)=sum_v/rho(x,y);
        end
    end

    %-----------------ThermalFlow---------------------

    % --Calc geq(u,v)--
    for x=1:Nx
        for y=1:Ny
            for i=1:MC.Dir
            geq(x,y,i)=fun_geq(eps(x,y),u(x,y),v(x,y),rho(x,y),i);
            end
        end
    end


    % --CollisionEps--
    for x=1:Nx
        for y=1:Ny
            for i=1:MC.Dir
                g(x,y,i)=fun_coll_wSource(geq(x,y,i),g(x,y,i),MC.tau_g,i);
            end
        end
    end

    % --StreamingEps--
    g_prestream=g; % save old values (before streaming) for boundary conditions
    g=fun_stream(g_prestream); %boundary nodes x=1 and x=Nx, y=1 and y=Ny are "missing" (not
streamed) 3 or 5 (corner) f-values


    % --Boundary ConditionsEps--

    % Calculate nodes that are missing
    %        C7  C3  C6       ^ y
    %          \ | /          |
    %        C4-C1-C2         |
    %          / | \          |
    %        C8  C5  C9         -----> x



    %top:  thermal boundary condition see Liu2010
    for x=1:Nx
%        g(x,Ny,:)=g(x,Ny-1,:);

         eps_top=3./3 .* sum(g(x,Ny-1,:))/sum(f(x,Ny-1,:));%-1./3*sum(g(x,Ny-2,:))/sum(f(x,Ny-
2,:)) ; %adiabatic Neummann;
 %         eps_top=3./3 .* eps(x,Ny-1); %-1./3*eps(x,Ny-2) ; %adiabatic Neummann;
 %         eps_top=0;

g(x,Ny,:)=fun_ThermalBCDirichlet_D2Q9_Liu2010_wCorners(eps_top,rho(x,Ny),g(x,Ny,:),zeros(9),x,Ny)
;
    end


    %bottom:  thermal boundary condition see Liu2010
    for x=1:Nx
%        g(x,1,:)=g(x,2,:);
         eps_bottom= 3./3 .* sum(g(x,2,:))/sum(f(x,2,:));%-1./3*sum(g(x,3,:))/sum(f(x,3,:)) ;
%adiabatic Neummann;
 %         eps_bottom= 3./3 .* eps(x,2); %-1./3*eps(x,3) ; %adiabatic Neummann;
 %         eps_bottom=0;

g(x,1,:)=fun_ThermalBCDirichlet_D2Q9_Liu2010_wCorners(eps_bottom,rho(x,1),g(x,1,:),zeros(9),x,1);
    end
    %left side:  thermal boundary condition see Liu2010
    for y=1:Ny
%        g(1,y,:)=g(2,y,:);
         eps_left=3./3 .* sum(g(2,y,:))/sum(f(2,y,:));%-1./3*sum(g(3,y,:))/sum(f(3,y,:)) ;
%adiabatic Neummann;
 %         eps_left=3./3 .* eps(2,y); %-1./3*eps(3,y)
 %         eps_left=0;

g(1,y,:)=fun_ThermalBCDirichlet_D2Q9_Liu2010_wCorners(eps_left,rho(1,y),g(1,y,:),zeros(9),1,y);
    end

    %right side:  thermal boundary condition see Liu2010
    for y=1:Ny
%        g(Nx,y,:)=g(Nx-1,y,:);
         eps_right=3./3 .* sum(g(Nx-1,y,:))/sum(f(Nx-1,y,:));%-1/3*sum(g(Nx-2,y,:))/sum(f(Ny-
2,y,:)) ; %adiabatic Neummann;
 %         eps_right=3./3 .* eps(Nx-1,y); %-1./3*eps(Nx-2,y)
 %         eps_right=0

g(Nx,y,:)=fun_ThermalBCDirichlet_D2Q9_Liu2010_wCorners(eps_right,rho(Nx,y),g(Nx,y,:),zeros(9),Nx,
y);

    end



    % --Calculate eps --
    for x=1:Nx
        for y=1:Ny
            sum_eps=0;
            for i=1:MC.Dir
                sum_eps=sum_eps + g(x,y,i);
            end
            eps(x,y)=sum_eps/rho(x,y);
        end
    end


end
%% ------------------------------
% ----Plots-----%
%------------------------------
% plot for last timestep

figure(fig_snap)
uplot=plot(u(round(Nx/2),:)./uw, 0:1/(Ny-1):1) ;
epsplot=plot(eps(round(Nx/2),:)/MC.eps_ref, 0:1/(Ny-1):1,':');



% plot cosmetics
figure(fig_snap)
xlabel(gca,' $\frac{u}{u_{w}}~or~\frac{eps}{eps_{w}}$' ,'Fontsize',12,'Interpreter','latex');
ylabel(gca,' $\frac{y}{H}$ ' ,'Fontsize',12,'Interpreter','latex');
grid on; box on;
legend([uplot, epsplot],'u','eps');
title(mytitle,'Fontsize',12,'Interpreter','latex');

% colormap plot for thermal
figure('Name','eps');
imagesc([0 1],[0 1],rot90(fliplr(eps.*MC.C_x^2 ./ MC.C_t^2./MC.cv_real)));
set(gca,'YDir','normal')
colorbar
title(mytitle,'Fontsize',12,'Interpreter','latex');
xlabel(gca,' $x$' ,'Fontsize',12,'Interpreter','latex');
ylabel(gca,' $y$ ' ,'Fontsize',12,'Interpreter','latex');

% vector plot for velocity and streamline plot
[x,y] = meshgrid(0:1/(Nx-1):1,0:1/(Ny-1):1);
figure();
quiver(x,y,rot90(fliplr(u)),rot90(fliplr(v)))
```

```
yStart=0.1:0.1:0.9;
xStart=0.5*ones(size(yStart));
streamline(x,y,rot90(fliplr(u)),rot90(fliplr(v)),xStart,yStart)
axis([-0.05 1.05 -0.05 1.05])
title(mytitle,'Fontsize',12,'Interpreter','latex');
xlabel(gca,' $x$' ,'Fontsize',12,'Interpreter','latex');
ylabel(gca,' $y$ ' ,'Fontsize',12,'Interpreter','latex');

% colormap plot for rho
figure('Name','rho');
imagesc([0 1],[0 1],rot90(fliplr(rho*MC.C_m./MC.C_x.^3)));
set(gca,'YDir','normal')
colorbar
title(mytitle,'Fontsize',12,'Interpreter','latex');
xlabel(gca,' $x$' ,'Fontsize',12,'Interpreter','latex');
ylabel(gca,' $y$ ' ,'Fontsize',12,'Interpreter','latex');

%interesting values
epsmean=mean(mean(eps));
Tmean=mean(mean(eps))  *MC.C_x^2 ./ MC.C_t^2/MC.cv_real ; % K
% -------------TheEnd------------
t_Elapsed=toc(t_Start);
```

8.1.2 Input Script

```
classdef MC
   properties (Constant)
    % ----------D2Q9------------------
    w=[4./9,1./9,1./9,1./9,1./9,1./36,1./36,1./36,1./36];
    cx=[0.0,1.0,0.0,-1.0,0.0,1.0,-1.0,-1.0,1.0];
    cy=[0.0,0.0,1.0,0.0,-1.0,1.0,1.0,-1.0,-1.0];
    Dir=9;

    % ----------Domain------------------
    Ny=20;
    Nx=20;
    timesteps=3000;

    % ----------MassFlow------------------
    uw=0.001; %chosen for low Ma  .. in lu
    rho=1; %in lu

    H_real=0.1 %m
    nue_real=1e-3 %mÂ²/s
    uw_real=0.01 %m/s
    rho_real=1000 %kg/mÂ³

    Re=MC.H_real*MC.uw_real/MC.nue_real
    nue=MC.Ny*MC.uw/MC.Re;
    tau=3*MC.nue+0.5;



    %Unit Conversion
    C_x=MC.H_real/MC.Ny; %  m
    C_t=MC.uw*MC.C_x/MC.uw_real % s
    C_m=MC.C_x.^3*MC.rho_real./MC.rho; % kg

    % ----------ThermalFlow------------------


    Pr=1;
    alpha=MC.nue./MC.Pr
    tau_g=3/2*MC.alpha+0.5 %Liu2010;
    tau_g_passive=3*MC.alpha+0.5

    cv_real=4.19e3 %J/(kg K)
    T_ref_real=273.15 % K
    eps_ref=MC.T_ref_real*MC.cv_real.* MC.C_t^2 ./ MC.C_x^2

    T_init_real=1.0*273.15 % K
    eps_init_real = MC.cv_real .* MC.T_init_real  % J/kg 12282985.5 equals energy of water with
293.15 K (cv=const=4.19kJ/(kg K))
```

```
    eps_init = MC.eps_init_real .* MC.C_t^2 ./ MC.C_x^2 ;

    Qdot_real=0*4.19e7 % J/(mÂ³s)... 4.19e7 changes the temperature of water by 10K each second
    Qdot=MC.Qdot_real*MC.C_x*MC.C_t^3/(MC.C_m)


    %------------------------
    t_sim=MC.timesteps*MC.C_t % seconds
    t_calc=965/3600*MC.Ny*MC.Nx/900*MC.timesteps/600 % hours estimated time of calculation

   end
end
```

8.1.3 Functions

8.1.3.1 Equilibrium Distribution Function

```
function [feq] = fun_feq(u,v,rho,i)
% i... counter for dirctions e.g. 1,2...9 for D2Q9
% Calculates one feq in one direction of one node
t1= u.^2+v.^2;
t2= u.*MC.cx(i)+v.*MC.cy(i);

feq=rho*MC.w(i).*(1.0 + 3*t2 + 4.5.*t2.^2 - 1.5*t1);
end
```

8.1.3.2 Collision

```
function [f_postcoll] = fun_coll_wSource(feq,f,tau,i)
f_postcoll = f-(1./tau)*(f-feq) + MC.w(i) *MC.Qdot;
end
```

8.1.3.3 Streaming

```
function [f_poststream] = fun_stream(f)
    %Streaming of the fluid nodes (1:Ny , 1:Ny)
    %        C7  C3  C6       ^ y
    %          \ | /          |
    %        C4-C1-C2         |
    %          / | \          |
    %        C8  C5  C9         -----> x
    f_poststream=NaN(size(f));

    % C1
    f_poststream(:,:,1)=f(:,:,1);

    % C2
    for x=1:MC.Nx-1
        for y=1:MC.Ny
            f_poststream(x+1,y,2)=f(x,y,2);
        end
    end

    % C3
    for x=1:MC.Nx
        for y=1:MC.Ny-1
            f_poststream(x,y+1,3)=f(x,y,3);
        end
    end

    % C4
    for x=2:MC.Nx
        for y=1:MC.Ny
            f_poststream(x-1,y,4)=f(x,y,4);
        end
    end

    % C5
    for x=1:MC.Nx
        for y=2:MC.Ny
            f_poststream(x,y-1,5)=f(x,y,5);
        end
    end

    % C6
    for x=1:MC.Nx-1
        for y=1:MC.Ny-1
            f_poststream(x+1,y+1,6)=f(x,y,6);
        end
    end

    % C7
    for x=2:MC.Nx
        for y=1:MC.Ny-1
            f_poststream(x-1,y+1,7)=f(x,y,7);
        end
    end

    % C8
    for x=2:MC.Nx
        for y=2:MC.Ny
            f_poststream(x-1,y-1,8)=f(x,y,8);
        end
    end

    % C9
    for x=1:MC.Nx-1
        for y=2:MC.Ny
            f_poststream(x+1,y-1,9)=f(x,y,9);
        end
    end
end
```

8.1.3.4 Boundary Handling

```
function [g_bound] = fun_ThermalBCDirichlet_D2Q9_Liu2010_wCorners(BC_eps, rho, g, g_Scheme,x,y)


% Calculates missing prob values at boundary nodes for given condition and location and
calculation scheme (by Liu2010)
%
%------OUTPUT------
% g_bound [vector] ... calculated  prob values for boundary node (for D2Q9 a vector with 9
values)
%
%------INPUT------
%
% BC_eps ... [scalar] value of internal energy at given boundary
% rho ... [scalar] density at boundary node
% g ... [vector] propability values for the boundary node  (for D2Q9 a vector with 9 values, with
missing values)
% g_Scheme ... [vector] prob values needed to calc g_st (see Liu2010)   (for D2Q9 a vector with 9
values)
% x,y ... cooradinates to determine/distinguish Edge or Corner
% Date: 29.05.2018


    %        C7  C3  C6       ^ y
    %          \ | /          |
    %        C4-C1-C2         |
    %          / | \          |
    %        C8  C5  C9         -----> x

% abc ... index of missing prob values
% d1,d2 etc. ... index of known prob values

% ---------Check if Corner---------

if x==MC.Nx && y==MC.Ny %upper right corner
    Corner=1;
    a1=4;
    a2=5;
    a3=7;
    a4=8;
    a5=9;
    d1=2;
    d2=3;
    d3=6;
    d4=1;
elseif x==1 && y==1 % lower left corner
    Corner=1;
    a1=2;
    a2=3;
    a3=6;
    a4=7;
    a5=9;
    d1=4;
    d2=5;
    d3=8;
    d4=1;
elseif x==1 && y==MC.Ny % upper left corner
    Corner=1;
    a1=2;
    a2=5;
    a3=6;
    a4=8;
    a5=9;
    d1=3;
    d2=4;
    d3=7;
    d4=1;
elseif x==MC.Nx && y==1 % lower right corner
    Corner=1;
    a1=3;
    a2=4;
    a3=6;
    a4=7;
    a5=8;
    d1=2;
    d2=5;
    d3=9;
    d4=1;
else
    Corner=0;
end

    %        C7  C3  C6       ^ y
    %          \ | /          |
    %        C4-C1-C2         |
    %          / | \          |
    %        C8  C5  C9         -----> x

% ---------Check if Edge---------

if Corner==0

if x==MC.Nx
        Edge=1;
        a=4;
        b=7;
        c=8;
        d1=1;
        d2=2;
        d3=3;
        d4=5;
        d5=6;
        d6=9;
elseif y==MC.Ny
        Edge=1;
        a=5;
        b=8;
        c=9;
        d1=1;
        d2=2;
        d3=3;
        d4=4;
        d5=6;
        d6=7;
```

```
elseif x==1
        Edge=1;
        a=2;
        b=6;
        c=9;
        d1=1;
        d2=3;
        d3=4;
        d4=5;
        d5=7;
        d6=8;
elseif  y==1
        Edge=1;
        a=3;
        b=6;
        c=7;
        d1=1;
        d2=2;
        d3=4;
        d4=5;
        d5=8;
        d6=9;

else
    Edge=0;
end

end

g_bound=zeros(size(MC.w)); %init


if Corner %BC for Corner
    eps_st= 1 ./ rho .*  ( g(d1) + g(d2) + g(d3) +g(d4)  + g_Scheme(a1) + g_Scheme(a2) +
g_Scheme(a3) + g_Scheme(a4) + g_Scheme(a5) );

    G_c=rho.* ( BC_eps - eps_st )./( MC.w(a1) + MC.w(a2) + MC.w(a3) + MC.w(a4) + MC.w(a5));

    g_bound(a1) = g_Scheme(a1) + MC.w(a1).*G_c ;
    g_bound(a2) = g_Scheme(a2) + MC.w(a2).*G_c ;
    g_bound(a3) = g_Scheme(a3) + MC.w(a3).*G_c ;
    g_bound(a4) = g_Scheme(a4) + MC.w(a4).*G_c ;
    g_bound(a5) = g_Scheme(a5) + MC.w(a5).*G_c ;
    g_bound(d1) = g(d1);
    g_bound(d2) = g(d2);
    g_bound(d3) = g(d3);
    g_bound(d4) = g(d4);

elseif Edge %BC for Edge


    eps_st= 1 ./ rho .*  ( g(d1) + g(d2) + g(d3) + g(d4) + g(d5) + g(d6) +  g_Scheme(a) +
g_Scheme(b) + g_Scheme(c) );

    G_c=rho.* ( BC_eps - eps_st )./( MC.w(a) + MC.w(b) + MC.w(c) );

    g_bound(a) = g_Scheme(a) + MC.w(a).*G_c ;
    g_bound(b) = g_Scheme(b) + MC.w(b).*G_c ;
    g_bound(c) = g_Scheme(c) + MC.w(c).*G_c ;
    g_bound(d1) = g(d1);
    g_bound(d2) = g(d2);
    g_bound(d3) = g(d3);
    g_bound(d4) = g(d4);
    g_bound(d5) = g(d5);
    g_bound(d6) = g(d6);
else
    error('Node is not a boundary node')

end
end
```

8.2 Derivation: Numerical Diffusion of Osmanlic Transport Algorithm

The numerical diffusion is developed for a D3Q19 lattice arrangement and the Osmanlic transport algorithm for scalars. The derivation is carried out for a cell node $\boldsymbol{x} = (x, y, z)$ and its neighbouring nodes $\boldsymbol{x_i} = \boldsymbol{x} + \boldsymbol{e_i}$ with fixed velocity vector

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}(\boldsymbol{x} + \boldsymbol{e_i}) = \left(u_x, u_y, u_z\right), u_x > u_y > u_z > 0.$$

The density distribution values f_i are set equal to the corresponding BGK equilibrium density distribution $f_i = f_i^{eq}$.

The normalized mass flow balances $\Delta m_i{}' = \frac{1}{\rho}\left(f_{-i}^{eq}(\boldsymbol{x} + \boldsymbol{e_i}\Delta t, t) - f_i^{eq}(x,t)\right)$ are:

$\Delta m_1{}' = -6/18\ u_x$	$\Delta m_7' = -6/36\ (u_x + u_y)$	$\Delta m_8{}' = 6/36\ (u_x + u_y)$
$\Delta m_2{}' = 6/18\ u_x$	$\Delta m_9{}' = -6/36\ (u_x - u_y)$	$\Delta m_{10}{}' = 6/36\ (u_x - u_y)$
$\Delta m_3{}' = -6/18\ u_y$	$\Delta m_{11}{}' = -6/36\ (u_x + u_z)$	$\Delta m_{12}{}' = 6/36\ (u_x + u_z)$
$\Delta m_4{}' = 6/18\ u_y$	$\Delta m_{13}{}' = -6/36\ (u_x - u_z)$	$\Delta m_{14}{}' = 6/36\ (u_x - u_z)$
$\Delta m_5{}' = -6/18\ u_z$	$\Delta m_{15}{}' = -6/36\ (u_y + u_z)$	$\Delta m_{16}{}' = 6/36\ (u_y + u_z)$
$\Delta m_6{}' = 6/18\ u_z$	$\Delta m_{17}{}' = -6/36\ (u_y - u_z)$	$\Delta m_{18}{}' = 6/36\ (u_y - u_z)$

With the balances above the Osmanlic transport algorithm (without sources) is given by:

$$\varepsilon(\boldsymbol{x}, t + \Delta t) = \varepsilon(\boldsymbol{x}, t) + \sum_{i=1}^{18} \Delta m_i{}' \cdot \begin{cases} \varepsilon(\boldsymbol{x} + \boldsymbol{e}_i, t), & \Delta m_i > 0 \\ \varepsilon(\boldsymbol{x}, t), & \Delta m_i \leq 0 \end{cases} \qquad \textit{Eq. 8-1}$$

$$\begin{aligned} \varepsilon(\boldsymbol{x}, t+1) = \varepsilon(\boldsymbol{x}, t) \quad & + 6/18\ u_x \cdot \left[\underbrace{-\varepsilon(\boldsymbol{x}, t)}_{i=1} + \underbrace{\varepsilon(x-1, y, z, t)}_{i=2}\right] + \cdots \\ & + 6/18\ u_y \cdot [-\varepsilon(\boldsymbol{x}, t) + \varepsilon(x, y-1, z, t)] + \cdots \\ & + 6/18\ u_z \cdot [-\varepsilon(\boldsymbol{x}, t) + \varepsilon(x, y, z-1, t)] + \cdots \\ & + 6/36\ (u_x + u_y) \cdot \left[\underbrace{-\varepsilon(\boldsymbol{x}, t)}_{i=7} + \underbrace{\varepsilon(x-1, y-1, z, t)}_{i=8}\right] + \cdots \\ & + 6/36\ (u_x - u_y) \cdot [-\varepsilon(\boldsymbol{x}, t) + \varepsilon(x-1, y+1, z, t)] + \cdots \\ & + 6/36\ (u_x + u_z) \cdot [-\varepsilon(\boldsymbol{x}, t) + \varepsilon(x-1, y, z-1, t)] + \cdots \\ & + 6/36\ (u_x - u_z) \cdot [-\varepsilon(\boldsymbol{x}, t) + \varepsilon(x-1, y, z+1, t)] + \cdots \\ & + 6/36\ (u_y + u_z) \cdot [-\varepsilon(\boldsymbol{x}, t) + \varepsilon(x, y-1, z-1, t)] + \cdots \end{aligned}$$

$$+6/36\ (u_y - u_z) \cdot \left[-\varepsilon(\boldsymbol{x},t) + \underbrace{\varepsilon(x,y-1,z+1,t)}_{i=18} \right]$$

The forward in time and backward/forward in space values are approximated by a *Taylor polynomial* of second order:

$$\varepsilon(\boldsymbol{x},t+1) = \varepsilon(\boldsymbol{x},t) + \frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial t^2}\Delta t^2, \Delta t = +1 \qquad \textit{Eq. 8-2}$$

Example for mixed backward/forward in space approximation:

$$\begin{aligned} \varepsilon(x-1,y+1,z,t) = \varepsilon(\boldsymbol{x},t) &+ \frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial x}\Delta x + \frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial y}\Delta y + \cdots \qquad \textit{Eq. 8-3}\\ &+ \frac{1}{2}\frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial x^2}\Delta x^2 + \frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial x\partial y}\Delta x\Delta y + \frac{1}{2}\frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial y^2}\Delta y^2\\ &\Delta x = -1 \textit{ and } \Delta y = +1 \end{aligned}$$

Inserting the Taylor polynomials (see Eq. 8-3 and Eq. 8-2) into the Osmanlic transport algorithm (Eq. 8-1) leads to following term:

Note that LHS refers to the left-hand side and RHS to the right-hand side of Eq. 8-1. The terms are sorted by derivatives of internal energy starting with the derivatives that are multiplied by the velocity in the same spatial direction. The nine $\varepsilon(\boldsymbol{x},t)$ values on the RHS of Eq. 8-1 in the squared brackets are already cancelled out.

$$LHS = \cancel{\varepsilon(\boldsymbol{x},t)} + \frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial t} + \frac{1}{2}\frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial t^2} =$$

$$RHS = \cancel{\varepsilon(\boldsymbol{x},t)} \qquad + u_x \cdot \frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial x} \cdot \underbrace{\left(-\frac{6}{18} - 4\cdot\frac{6}{36}\right)}_{-1} + \cdots \qquad \text{Term 1}$$

$$+ u_y \cdot \frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial y} \cdot \underbrace{\left(-\frac{6}{18} - 4\cdot\frac{6}{36}\right)}_{-1} + \cdots \qquad \text{Term 2}$$

$$+ u_z \cdot \frac{\partial\varepsilon(\boldsymbol{x},t)}{\partial z} \cdot \underbrace{\left(-\frac{6}{18} - 4\cdot\frac{6}{36}\right)}_{-1} + \cdots \qquad \text{Term 3}$$

$$+ u_x \cdot \frac{\partial^2\varepsilon(\boldsymbol{x},t)}{\partial x^2} \cdot \underbrace{\left(+\frac{3}{18} + \frac{3}{36} + \frac{3}{36} + \frac{3}{36} + \frac{3}{36}\right)}_{+\frac{1}{2}} + \cdots \qquad \text{Term 4}$$

$$+u_y \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial y^2} \cdot \underbrace{\left(+\frac{3}{18}+\frac{3}{36}-\frac{3}{36}+\frac{3}{36}+\frac{3}{36}\right)}_{+1/3} + \cdots$$ Term 5

$$+u_z \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial z^2} \cdot \underbrace{\left(+\frac{3}{18}+\frac{3}{36}-\frac{3}{36}+\frac{3}{36}-\frac{3}{36}\right)}_{+1/6} + \cdots$$ Term 6

$$+u_x \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial y^2} \cdot \underbrace{\left(+\frac{3}{36}+\frac{3}{36}\right)}_{+\frac{1}{6}} + \cdots$$ Term 7

$$+u_x \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial z^2} \cdot \underbrace{\left(+\frac{3}{36}+\frac{3}{36}\right)}_{+\frac{1}{6}} + \cdots$$ Term 8

$$+u_y \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial x^2} \cdot \underbrace{\left(+\frac{3}{36}-\frac{3}{36}\right)}_{0} + \cdots$$ Term 9

$$+u_y \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial z^2} \cdot \underbrace{\left(+\frac{3}{36}+\frac{3}{36}\right)}_{+\frac{1}{6}} + \cdots$$ Term 10

$$+u_z \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial x^2} \cdot \underbrace{\left(+\frac{3}{36}-\frac{3}{36}\right)}_{0} + \cdots$$ Term 11

$$+u_z \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial y^2} \cdot \underbrace{\left(+\frac{3}{36}-\frac{3}{36}\right)}_{0} + \cdots$$ Term 12

$$+u_x \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial x \partial y} \cdot \underbrace{\left(+\frac{6}{36}-\frac{6}{36}\right)}_{0} + \cdots$$ Term 13

$$+u_x \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial x \partial z} \cdot \underbrace{\left(+\frac{6}{36}-\frac{6}{36}\right)}_{0} + \cdots$$ Term 14

$$+u_y \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial x \partial y} \cdot \underbrace{\left(+\frac{6}{36}+\frac{6}{36}\right)}_{+1/3} + \cdots$$ Term 15

$$+u_y \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial y \partial z} \cdot \underbrace{\left(+\frac{6}{36}-\frac{6}{36}\right)}_{0} + \cdots$$ Term 16

$$+u_z \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial x \partial z} \cdot \underbrace{\left(+\frac{6}{36}+\frac{6}{36}\right)}_{+1/3} + \cdots$$ Term 17

$+u_z \cdot \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial y \partial z} \cdot \underbrace{\left(+\frac{6}{36}+\frac{6}{36}\right)}_{+1/3}$	Term 18

Terms 1-3 on the RHS are shifted to the LHS. The $\frac{1}{2}\frac{\partial^2 \varepsilon(x,t)}{\partial t^2}$ term of the LHS can be replaced according to the advection equation (time derivative of advection equation) and is shifted to the RHS:

$$\frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial t^2} = \mathrm{u}_\mathrm{x}^2 \cdot \frac{\partial^2 \varepsilon(\mathbf{x},\mathrm{t})}{\partial \mathrm{x}^2} + \mathrm{u}_\mathrm{y}^2 \cdot \frac{\partial^2 \varepsilon(\mathbf{x},\mathrm{t})}{\partial \mathrm{y}^2} + \mathrm{u}_\mathrm{z}^2 \cdot \frac{\partial^2 \varepsilon(\mathbf{x},\mathrm{t})}{\partial \mathrm{z}^2}$$

Note that Term 9, 11, 12, 13, 14 and 16 are zero as well as the six “mixed” terms of component velocity and first order space derivatives (e.g. $\mathrm{u}_\mathrm{z} \cdot \frac{\partial \varepsilon(\mathbf{x},\mathrm{t})}{\partial \mathrm{y}}$) which are **not listed** on the RHS.

This leads to:

$$\boxed{\begin{aligned} &\frac{\partial \varepsilon(\boldsymbol{x},t)}{\partial t} \underbrace{+ u_x \cdot \frac{\partial \varepsilon(\boldsymbol{x},t)}{\partial x} + u_\mathrm{y} \cdot \frac{\partial \varepsilon(\boldsymbol{x},t)}{\partial \mathrm{y}} + u_\mathrm{z} \cdot \frac{\partial \varepsilon(\boldsymbol{x},t)}{\partial \mathrm{z}}}_{Term\ 1-3} = \\ &= \underbrace{\begin{pmatrix} -\frac{1}{2}u_x^2 + \underbrace{\frac{1}{2}u_x}_{Term\ 4} \\ -\frac{1}{2}u_y^2 + \underbrace{\frac{1}{3}u_y}_{Term\ 5} + \underbrace{\frac{1}{6}u_x}_{Term\ 7} \\ -\frac{1}{2}u_z^2 + \underbrace{\frac{1}{6}u_z}_{Term\ 6} + \underbrace{\frac{1}{6}u_x}_{Term\ 8} + \underbrace{\frac{1}{6}u_y}_{Term\ 10} \end{pmatrix}}_{\boldsymbol{D}^{\boldsymbol{Os}}} \cdot \begin{pmatrix} \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial x^2} \\ \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial y^2} \\ \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial z^2} \end{pmatrix} + \underbrace{\begin{pmatrix} \underbrace{\frac{1}{3}u_y}_{Term\ 15} \\ \underbrace{\frac{1}{3}u_z}_{Term\ 17} \\ \underbrace{\frac{1}{3}u_z}_{Term\ 18} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial x \partial y} \\ \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial x \partial z} \\ \frac{\partial^2 \varepsilon(\boldsymbol{x},t)}{\partial y \partial z} \end{pmatrix}}_{additional\ error\ term} \\ &\qquad \ldots \quad \textit{Eq. 8-4} \end{aligned}}$$

Despite the stated nomenclature of this thesis (see page VI) the capital bold $\boldsymbol{D}^{\boldsymbol{Os}}$ refers to a vector.