Friedrich Holzinger, BSc

Modification of Filtered Drag Models for Polydisperse Flows

MASTER'S THESIS

to achieve the university degree of

Master of Science

Master's degree programme: Chemical and Process Engineering

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Stefan Radl

Institute of Process and Particle Engineering

in cooperation with
Dipl.-Ing. Dr. techn. Günter Gronald, Andritz Energy and Environment

Graz, Oktober 2015

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

31.10.2015 — Date

Friedrich Holzinger — Signature

Abstract

Circulating fluidized beds are difficult to investigate due to instantaneous formation of particle clusters and streamers. Detailed simulations can help to quantify these effects, and can be also used to develop so-called “filtered” drag models. Previously, filtered drag models for monodisperse systems have been widely used and verified. However, within the last few years only first steps to establish a rigorous filtered drag models for bidisperse or polydisperse systems have been made.

This thesis should close this gap by studying the effect of a variety of (classical and filtered) drag models for polydisperse particle mixtures in an industrially-relevant system. The particle population considered reflected the system in a typical flue gas cleaning application. First, the effect of drag models and grid resolution have been analyzed in a fully periodic box. This setup allowed us to study the clustering behavior of a freely sedimenting gas-particle suspension in an infinitely large domain. The effect of particle clustering was quantified by computing the dimensionless (domain-averaged) slip velocity. Since the typical particle Reynolds number of the gas-particle system was less than unity, the inverse of this dimensionless slip velocity is a typical drag correction. A verification of the predictions when using the advanced (i.e., filtered) drag model was done by comparing the results with predictions that used classical drag models. Furthermore, parcel-based simulations have been performed. These simulations revealed that a smoothing of the exchange fields (i.e., the particle volume fraction and the volumetric coupling forces) has a profound effect on the flow predictions. Thus, applying such a smoothing operation is essential, not only for the correct prediction of sedimentation rates, but also to stabilize the simulation in case of large volumetric coupling forces typical for industrial applications. Finally, the developed drag and smoothing models, as well as a novel coupling scheme, has been applied to study turbulent gas-particle-droplet flow in a full-scale riser. The simulations revealed that the filtered drag model has a small effect. This is because the sedimentation velocity of the particles is much smaller than a typical rate of turbulent dispersion. It was shown that the injection velocity of the droplets has a significant effect on the overall flow structure. Also, the simulations revealed that the injected particle cloud is able to penetrate the flow vertically downwards. This vertical particle jet penetration can lead to unwanted downflow in the nozzle region, which was also observed in industrial practice. A simple mechanistic model was proposed that can help to avoid downflow in the nozzle region via a future modification of the riser design.

Kurzfassung

Zirkulierende Wirbelschichten sind wegen spontaner Ausbildung von Strähnen schwierig zu untersuchen. Detaillierte Simulationen können bei der Quantifizierung dieser Effekte helfen, und können auch zur Entwicklung sogenannter "gefilterter" Strömungswiderstandsmodelle verwendet werden. Gefilterte Strömungswiderstandsmodelle sind für monodisperse Systeme weit verbreitet und verifiziert. Jedoch wurden in den letzten Jahren erste Schritte zu belastbaren gefilterten Strömungswiderstandsmodellen für bi-disperse und polydisperse Systeme unternommen.

Diese Arbeit soll diese Lücke durch die Untersuchung einer Vielzahl von (klassischen und gefilterten) Strömungswiderstandsmodellen für polydisperse Partikelmischungen in einem industriell relevanten System schließen. Die betrachtete Partikelpopulation stammt aus einer typischen Rauchgasreinigungsanwendung. Zuerst wurden die Auswirkungen der Strömungswiderstandsmodelle und der Gitterauflösung untersucht. Der Effekt der Strähnenbildung wurde durch Berechnung der dimensionslosen (räumlich gemittelten) Schlupfgeschwindigkeit quantifiziert. Der reziproke Wert dieser Geschwindigkeit entspricht einer typischen Strömungswiderstandskorrektur. Die Vorhersagen mit anspruchsvolleren (d.h. gefilterten) Strömungswiderstandsmodellen wurden anhand klassischer Strömungswiderstandsmodelle überprüft. Darüber hinaus wurden sogenannte „parcel“-basierte Simulationen durchgeführt. Diese zeigten, dass eine Glättung der Austauschfelder (d.h. die Partikelvolumenfraktion und die volumetrischen Kopplungskräfte) eine starke Wirkung auf die Strömungsvorhersagen hat. Somit ist die Anwendung einer solchen Glättung wesentlich zur Stabilisierung der Simulation im Fall großer volumetrischer Kopplungskräfte, typisch für industrielle Anwendungen. Schließlich wurden die entwickelten Strömungswiderstands- und Glättungsmodelle sowie ein neuartiges Kopplungsschema angewendet, um eine turbulente Gas-Partikel-Tröpfchen-Strömung in einem Steigrohr mit realitätsgetreuen Abmessungen zu untersuchen. Die Simulationen zeigten eine geringe Wirkung des gefilterten Strömungswiderstandsmodells, da die Sedimentationsgeschwindigkeit der Partikel wesentlich kleiner ist als die einer typischen Dispersionsrate aufgrund Turbulenz. Die Einspritzgeschwindigkeit der Tröpfchen hat einen signifikanten Effekt auf die Gesamtströmungsstruktur. Eingebrachte Partikel können die Strömung ungewollt vertikal nach unten verlassen, was auch in der industriellen Praxis beobachtet wurde. Daher wurde eine Änderung der Steigrohrausführung vorgeschlagen.

Acknowledgements

The author (F. Holzinger) appreciates all those who enabled this work and assisted him in doing that.

First of all the author is beholden to the industrial partner Andritz Energy & Environment GmbH for their cooperation. They supported him financially and provided exemplary data, which underlines the importance of this thesis in industry.

The author owes thanks to his advisor Dr. Stefan Radl for his excellent support. Regular meetings kept them focused on their common goal and ensured progress with limited resources. Moreover, he provided the author with plenty genius ideas, permanent good advice and technical service.

The author gratefully appreciates the Institute of Process and Particle Engineering for the place of work, necessary work equipment and very good administration cooperation. At this point he thanks the work colleagues for pleasant working conditions and encouragement, which merged the author into work.

The author gratefully acknowledges support from NAWI Graz by providing access to dcluster.tugraz.at.

The evaluation committee is gratefully appreciated by the author and he looks forward for a positive examination.

Last but not least the author thanks classmates, friends and his family for their encouragement, compassion and motivation.

Table of Contents

List of Figures

List of Tables

Abbreviations

2D, 3D	two-dimensional, three-dimensional
CFB	circulating fluidized bed
CFD	computational fluid dynamics
DEM	discrete/distinct element method (Cundall and Strack, 1979), for collision dominated flows
DNS	direct numerical simulation
DPM	discrete particle (or parcel) method
EE	Euler-Euler, i.e., approach where the continuous and the dispersed phase are modeled using a continuum approach
EL	Euler-Lagrange, i.e., approach where the continuous phase is modeled using a continuum approach, and the dispersed phase is modeled using a discrete approach
KTGF	kinetic theory of granular flow
LBM	Lattice-Boltzmann method, a meso-scale approach to model fluids
LES	large eddy simulation
MP-PIC	multiphase particle-in-cell method (Andrews and O'Rourke, 1996; Snider et al., 1998)
PSD	particle size distribution
RANS	Reynolds averaged Navier-Stokes equation
SGS	sub grid scale
URANS	unsteady (i.e., transient) RANS
TFM	two fluid model, i.e., an EE-based model

Nomenclature

Latin symbols

a	volume-specific surface area	[m^2/m^3]
A	total cross-sectional flow area	[m^2]
c_p	mass-specific heat capacity at constant pressure	[J/(kgK)]
CF	coupling factor	
Co	Courant number	
d	diameter	[m]
	particle diameter	[m]
D	diameter, depth	[m]
	diffusion coefficient	[m^2/s]
d_{32}	Sauter mean diameter	[m]
d_P	parcel diameter	[m]
E	Young's modulus (per atom type)	[Pa]
e_{pp}	coefficient of restitution (for particle-particle collisions)	
F_{ij}	interphase momentum transfer coefficient (btw. phases i and j)	[kg/(m^{3}s)]
$\mathbf{f}$	force per unit volume of the suspension	[N/m^3]
$\mathbf{F}$	force acting on a single particle	[N]
$\mathbf{g}$	gravitational acceleration	[m/s^2]
g_0	radial distribution function at contact	
h	mass-specific enthalpy	[J/kg]

H	enthalpy	[J]
	height	[m]
k	mass-specific turbulent (or SGS) kinetic energy	[m^2/s^2]
Kn	Knudsen number	
l	length, distance, spacing	[m]
L	length	[m]
M	mass	[kg]
N	number (e.g. of particles)	
p	pressure	[Pa]
q	mass-specific heat	[J/kg]
Q	heat	[J]
q_3	probability density function (mass based)	[1/m]
Q_3	cumulative probability function (mass based)	
ΔQ_3	mass fraction of particles in a certain class	
R	universal gas constant	[J/(kmolK)]
Re	Reynolds number	
R_i	gas constant of species i	[J/(kgK)]
$\dot{S}$	volumetric source term	[$kg/(m^3s)$]
$\mathbf{S}$	strain rate tensor	[1/s]
Sc	Schmidt number	
Sh	Sherwood number	
t	time	[s]
T	temperature	[K]

t_{Hertz}	particle collision time according to Hertz	[s]
t_{Rayleigh}	particle collision time according to Rayleigh	[s]
Δt	time step	[s]
$\mathbf{u}$	(local) fluid velocity	[m/s]
$\mathbf{U}$	average fluid velocity	[m/s]
$\mathbf{u}_{\mathrm{slip}}$	slip velocity of particles	[m/s]
V	volume (e.g. of particles)	[m³]
V_{P}	parcel volume	[m³]
$\mathbf{v}$	(local) particle velocity	[m/s]
$\mathbf{V}$	average particle velocity	[m/s]
w	mass fraction	
W	width	[m]
y_i	dimensionless particle size	

Greek symbols

α	coarse graining ratio	
α_{Z}	inclination angle	[°]
β	drag coefficient	[kg/(m^{3}s)]
	mass transfer coefficient	[m/s]
β_{ij}	fluid-mediated particle-particle drag friction coefficient	[kg/(m^{3}s)]
γ	transferred quantity (via smoothing)	
Δ_{filter}	filter size	[m]

ε	hold up	[kg/m^3]
	turbulent energy dissipation rate	[m^2 /s^3]
λ	lubrication cut-off distance	[m]
μ	mass loading	
μ_f	dynamic viscosity of the fluid phase	[Pa·s]
μ_{PP}	particle-particle friction coefficient (per atom type pair)	
ν_f	kinematic viscosity of the fluid phase	[m^2/s]
ν_P	Poisson ratio (per atom type)	
φ	volume fraction	
φ_P	total particle volume fraction	
ϕ	flux quantity across a surface	[m/s, kg/(m^{2}s)]
$\mathbf{\Phi}$	volumetric fluid-particle coupling force	[N/m^3]
ρ	density	[kg/m^3]
σ_N	standard deviation	
τ	characteristic time scale	[s]
	residence time	[s]
$\boldsymbol{\tau}$	turbulent stress tensor	[m^2/s^2]

Subscripts

0	at the initial state
B	buoyancy
BN	before nozzle

calibr	calibration
char	characteristic
cell	cell
CG	clean gas
CZ	cleaning zone
coll	collision
corr	correction
crit	critical
cross	cross-sectional
d	droplet
D	drag
domain	domain
eff	effective
est	estimated
exp	expected
f	fluid
	filter
grid	grid containing a cell
FG	flue gas (raw)
filter	filter
fixed	fixed bed
flux – avg.	flux-averaged, i.e., weighted with the local flux
g	gaseous

G	gravity
	gas
i	individual of discrete elements, e.g. particles or cells
	inner
i, *j*	class *i*, class *j*
inject	injection
inlet	inlet
K	Kolmogorov scale
l	liquid
LS	light scattered
m	mean
max	maximum
min	minimum
mix	mixture
mono	monodisperse
N	number of elements
O	upper class boundary
o	outer
P	parcel
poly	polydisperse
prim	primary particle
PP	particle-particle interaction
quench	quenching

R	recirculate
ref	reference
$relax$	relaxation
sat	saturated steam
sf	superficial
SGS	sub grid scale
sim	parameter for simulation
$smooth$	smoothened
t	terminal
	turbulent
tot	total
U	lower class boundary
vap	evaporation
W	water
X,Y,Z	spatial directions (in a Cartesian coordinate system)

Superscripts

$*$	dimensionless
$'$	fluctuating component (e.g., of turbulent flow)
$\bar{}$	time-averaged
$\tilde{}$	filtered (on filter size)

Other

$\lvert x \rvert$	absolute value of scalar x
$\lvert \mathbf{x} \rvert, x$	norm of vector $\mathbf{x}$
$\langle x \rangle$	ensemble-averaged quantity x
$\mathbf{x}$	vector or tensor quantity

1 Introduction

1.1 Motivation

Size-polydisperse gas-solid flows are of key importance for a number of industrial applications, such as blast furnaces, fluidized beds, or classifiers. These flows are often characterized by a high mass loading (i.e., the ratio of the particle to the fluid mass is large) and a wide spectrum of the local particle concentration. Unfortunately, these flows are inherently unstable and spontaneously form clusters and streamers (i.e., regions of high particle concentration) that feature a wide range of length and time scales (González, 2013, p. 1; Igci et al., 2008, p. 1431). These meso-scale structures can have dramatic hydrodynamic effects (Ozel et al., 2013, p. 43), e.g. on the average slip velocity (Radl et al., 2012, p. 1), on the fluid-particle drag force, or on the segregation rate (González, 2013, p. iii). In order to predict the effect of meso-scale structures, models have been developed and validated in the past.

Experimental measurements are often limited to space- or time-averaged quantities, or cannot be done since dense gas-solid flows are opaque. In contrast, simulations can be used to predict local quantities (González, 2013, p. 3), and hence can help to unveil the physics that dictate the formation of meso-scale structures. In simulations, material and flow conditions can be perfectly controlled, which is often not the case in experiments (Beetstra et al., 2007, p. 490). Fully resolved simulations (i.e., simulations that directly predict meso-scale structures) are computationally expensive, since the size of the meso-scale structures is in the order of the particle size. Typically, simulations on an industrial scale cannot resolve these meso-scale structures (Igci et al., 2008, p. 1432). In order to account for effects of clustering that are not resolved on the coarse grid scale, filtered models have been developed (Igci et al., 2008). Drag is the predominant interaction force in gas-particle systems, and hence filtered drag models are of key importance for industrial-scale simulations of gas-solid flows (Igci et al., 2008, p. 1432).

1.2 Goals

It is now well accepted that filtered drag models are required for the reliable prediction of gas-particle flows in industrial-scale fluidized beds. While most of the recent developments of filtered drag models (Milioli et al., 2013; Ozel et al., 2013; Parmentier et al., 2012) have focused on monodisperse Geldart A systems, the recent study of Holloway et al. (2011) was the first step towards a systematic development of a filtered drag model for bi-disperse systems. In a follow-up work (Holloway and Sundaresan, 2014), a model for filtered simulations involving polydisperse gas-particle suspensions was presented.

This thesis will attempt to investigate the reliability of a filtered drag model by an ad-hoc modification of a filtered drag model for monodisperse systems.

The goals of this thesis are

(i) to provide data and insight of the most important parameters that impact certain flow phenomena in an industrial-scale fluidized bed (e.g., the segregation state, or the rate of fines elutriation), as well as

(ii) to support the implementation of new filtered models into ANSYS® Fluent® software (specifically in the DDPM sub-package).

In this work, also effects due to a non-isothermal temperature distribution should be taken into account.

1.3 Tasks and Thesis Outline

This thesis focuses on parcel-based methods (similar to Patankar and Joseph (2001), and O'Rourke and Snider (2012)), which have been extensively used to simulate circulating fluidized beds (CFBs). The following tasks were identified to be of relevance:

1) Generation of fully-resolved reference data for a freely sedimenting polydisperse gas-particle suspension (at least three particle size fractions, particle size distribution to be defined by Andritz Energy & Environment) in a periodic box simulation setup. This should be done by performing large-scale simulations using the software CFDEM® (i.e., a combination of OpenFOAM® and LIGGGHTS®) on one of the clusters of the TU Graz. The domain-averaged slip velocity (for each

size fraction), stress, and the particle-phase viscosity should be recorded for a range of particle volume fractions.

2) A sensitivity study with respect to the parcels size and grid resolution should be performed in a periodic box. This data will constitute the basis for the comparison with predictions made by filtered models. Furthermore, a study of the effect of temperature gradients (caused by the evaporating water droplets) should be performed.

3) A large set of filtered simulations (using the filtered drag model of Radl and Sundaresan (2014), as well as of Holloway and Sundaresan (2014) for a polydisperse (and a monodisperse system with identical Sauter mean diameter d_{32}) in a periodic domain should be performed. The drag correction factor for each size fraction should be investigated in order to match the reference slip velocity calculated from the fully-resolved simulations. Similarly, a prefactor for the drag correction factor should be determined such that the (mean) slip velocity in case one uses a monodisperse particle population having the same d_{32} is matched. These modifications of the filtered drag law should be repeated for a range of particle volume fractions (typical for CFBs) to obtain a complete filtered drag model for a polydisperse gas-particle suspension.

4) We should generate the geometry and computational mesh of one of Andritz' fluidized beds (non-isothermal, non-reacting, particles should be assumed to be non-cohesive), and simulate the flow in this FB using

   (i) a polydisperse particle population with and without the modified drag model, as well as

   (ii) a monodisperse particle population with and without the modified drag model.

   Non-isothermal conditions should be taken into account by introducing a model for quenching by injected water droplets.

5) We should then analyze and compare the results of these simulations with experimental data (if made available by Andritz) in order to assess which drag models yields the most realistic results.
6) Finally, we should transfer the most suitable filtered model to a User Defined Function (UDF) to be used in the DDPM model of the ANSYS® Fluent® software. Testing of the implementation should be supported by Dr. Gronald (Andritz to provide the computational resources and the software license for the time span of the implementation and testing).

The thesis starts with a description of state of the art, followed by the relevant theoretical background and the models used (see Section 3). The case setup for the first three tasks is discussed in Section 04. Section 5 discusses the case setup needed in task four, and the results of all tasks are discussed in Section 6.

2 State of the Art

2.1 Simulation Approaches for Gas-Particle Systems

Classical models for gas-particle flows rely on an Euler-Eulerian (EE) approach, i.e., the particle cloud is considered to be a continuum that interpenetrates the continuous (i.e., gas) phase. Such models can be found in literature, and are often referred to as two-fluid models (TFM). In contrast, an Euler-Lagrangian (EL) approach is capable (in principle) to account for forces that act on individual particles. A list of fluid-particle forces and relevant governing equations used within this work can be found in González' thesis (2013, sec. 2). Algorithmic details for the coupling of Eulerian (fixed) and Lagrangian (particle tracking) frames of reference relevant for the current work can be found in González' thesis (2013, sec. 2), as well as in the work of Zhu et al. (2007, p. 3387).

It should be noted that models for particle-particle interactions play an important role when predicting particle clustering phenomena. These models are (i) collisional models that predict the effect of inelastic collisions, as well as (ii) fluid-mediated particle-particle interactions. The latter are only of importance for polydisperse systems characterized by a small to moderate particle-to-fluid density ratio (González, 2013, p. 55). Hence, fluid-mediated particle-particle interaction models will be neglected in the current work, since the particle-to-fluid density ratio is $O(2300)$.

2.2 Reduction of Computational Cost

With the aim to model fluid-particle flow in industrial-scale equipment, the relative size of the computational domain size with respect to the smallest flow scales is typically large. Hence, the required number of computational grid cells and particles (when attempting to resolve all flow phenomena) is beyond our resources. There are two strategies to counteract this: First, a virtual "agglomeration" (i.e., grouping) of the particles to computational parcels is often used. Second, the grid resolution will be chosen sufficiently coarse (typically in the order of 10 cm). Unfortunately, both strategies are leading to additional modeling efforts related to unresolved meso-scale structures.

Radl et al. (2011, p. 124/2) describe pros and cons of three parcel-based approaches. An important difference is the model employed to account for particle collisions within the parcel, and to prevent particles from becoming close-packed. The model used is described, and is based on the so-called "Multi-Phase Particle-In-Cell" (MP-PIC) approach proposed by Andrews and O'Rourke (1996). The MP-PIC approach has been widely applied in industry, and is part of the software package "Barracuda®". Particle collisions within the parcels are considered by a simple particle pressure model.

Filtered drag models can be found in the outline section (see Section 1.2) for both monodisperse and bidisperse systems. The filtered drag model intended to be used in the current work is described in detail in Section 3.1.4, and the parcel approach is detailed in Section 3.1.5. It is employed in drag models of Beetstra et al. (2007) and Holloway et al. (2010). González (2013, p. 1) highlights that the model is only valid for monodisperse gas-particle systems.

2.3 Microscopic Drag Models

Most of the drag models published are based on experimental data, and since millennium turn by direct numerical simulation. Beetstra et al. (2007) give a general overview and Di Felice (1995) gives an overview on experimental data. Both studies focused on monodisperse systems only.

The drag force in polydisperse systems can be quantified by (i) measuring the terminal settling velocity of each individual class of particles in a sedimentation experiment, or (ii) by measuring the segregation behavior. Much of the data available considers segregation, often based on the experimental data of Goldschmidt et al. (2003). With respect to the experimental measurement of sedimentation velocities, Beetstra et al. state the following:

> *"The problem with these kind of experiments is the particles segregate while falling, so that locally the mass fraction of the species […] is not constant. Also, the experiments only give indirect information on the drag force. Although several methods have been developed to measure the drag force on a particle directly, […] these are all limited to single particles or particles that are surrounded by only a few*

> *others, which cannot be representative of a bi- or polydisperse system."* (Beetstra et al., 2007, p. 493)

Many authors use a combination of the models of Ergun (1952) and Wen and Yu (1966) for monodisperse systems. Such a combined model has been widely used in the fluidized bed community (Gidaspow, 1994; van der Hoef et al., 2005; Zhu et al., 2007)

Holloway et al. use a drag model for polydisperse gas-particle systems which is derived from direct numerical simulation (DNS) instead of experiments since

> "*Beetstra et al. (2006) found that drag models derived from direct numerical simulations provided the best agreement with experimental observations of segregation in polydisperse fluidized bed simulations*." (Holloway et al., 2011, p. 4406)

Radl et al. (2014, p. 5) have shown in their article that the monodisperse formulation of the drag model provided by Beetstra et al. (2010) gave the best agreement with the experimental results for the polydisperse systems of Goldschmidt et al. (2003). Consequently, in this thesis the models of Gidaspow (1994), Beetstra et al. (2007) and Holloway et al. (2010) will be used.

3 Theory and Model

3.1 Models for Fluid-Particle Drag Forces

Classical drag models consider drag on sedimenting particles in gas-solid suspensions at steady-state conditions (Gidaspow, 1994; van der Hoef et al., 2005; Zhu et al., 2007). The drag coefficient in these models is typically a function of the particle Reynolds number and the particle volume fraction. These are also known as standard drag models, and are not able to take certain particle-particle interactions into account. Advanced drag models do take more complex interactions of particles into account, for example the so-called fluid-mediated particle-particle drag. Hence, these models are capable to predict the drag force in dense fluid-particle suspensions with higher confidence (González, 2013; Radl and Sundaresan, 2014). Fluid-mediated particle-particle interactions are, however, only significant in systems where the fluid density is comparable to the particle density.

The monodisperse models of Gidaspow (1994) and Beetstra et al. (2007) will be explained in the following section in detail, followed by the models of Beetstra et al. (2007) and Holloway et al. (2010) for polydisperse suspensions.

3.1.1 Key Quantities when Predicting Fluid-Particle Drag

The particle (or parcel) volume fraction of a class is defined as the ratio of the total volume of particles (or parcels) of that class and the total volume of the suspension:

$$\varphi_{\mathrm{P},i} = \frac{\sum V_i}{V_{\mathrm{tot}}} . \tag{3.1}$$

$$\varphi_P = \sum \varphi_{P,i} \tag{3.2}$$

The slip velocity of a single particle is used to measure its settling behavior and is defined as:

$$\mathbf{u}_{\mathrm{slip},i} = \Delta \mathbf{u}_i = \mathbf{v}_i - \mathbf{u} . \tag{3.3}$$

The terminal settling velocity of an isolated particle is the velocity of that particle at steady-state conditions. For Stokes flow the terminal settling velocity is:

$$\mathbf{u}_{\mathrm{t},i} = \frac{d_i^2(\rho_{\mathrm{P},i} - \rho_{\mathrm{f}})\mathbf{g}}{18\mu_{\mathrm{f}}} . \tag{3.4}$$

In order to verify the resulting settling behavior of a particulate system, the dimensionless domain-averaged slip velocity is typically used. As a reference velocity scale the terminal settling velocity of a single particle that represents the particulate system must be used. This can be done by choosing a particle with a diameter equal to the Sauter mean diameter of the particulate system:

$$\langle d \rangle = d_{32} = \frac{\sum N_i d_i^3}{\sum N_i d_i^2} = \left[\frac{1}{\varphi_{\mathrm{P}}} \sum \frac{\varphi_{\mathrm{P},i}}{d_i} \right]^{-1} . \tag{3.5}$$

According to Radl et al. (2012, p. 5), the domain-average slip velocity is defined as the difference of the Favre-averaged fluid and particle velocity:

$$\langle \mathbf{u}_{\mathrm{slip}} \rangle = \frac{\langle \mathbf{u}(1-\varphi_{\mathrm{P}}) \rangle}{\langle 1-\varphi_{\mathrm{P}} \rangle} - \frac{\langle \sum \varphi_{\mathrm{P},i} \mathbf{v}_i \rangle}{\langle \varphi_{\mathrm{P}} \rangle} \tag{3.6}$$

The domain-averaged Reynolds number for polydisperse particulate systems is adapted from the particle Reynolds number of a corresponding monodisperse system:

$$\langle Re \rangle = \frac{\left(1 - \langle \varphi_{\mathrm{P}} \rangle\right) \langle \mathbf{u}_{\mathrm{slip}} \rangle \langle d \rangle}{\nu_{\mathrm{f}}} \tag{3.7}$$

A simple estimate for the above Reynolds number is based on Stokes drag law and an infinitely dilute system:

$$\langle Re \rangle \approx \frac{\langle u_{\mathrm{t}} \rangle \langle d \rangle}{\nu_{\mathrm{f}}} \tag{3.8}$$

3.1.1.1 Drag Force and Friction Coefficient

Beetstra et al. (2007) begin the derivation of a drag force with the well-known expression for Stokes drag force acting on a single, isolated sphere.

$$\mathbf{F}_{\mathrm{D}} = 3\pi\,\mu_{\mathrm{f}}\,d\,\mathbf{U} \tag{3.9}$$

One might use this drag force as a reference force for particulate systems. Thus, Beetstra et al. (2007) normalized the drag force via

$$F_{\mathrm{D}}^{*} = \mathbf{F}_{\mathrm{D}} / 3\pi\,\mu_{\mathrm{f}}\,d\,\mathbf{U}_{\mathrm{sf}}\,, \tag{3.10}$$

where the average fluid velocity has been replaced by the superficial fluid velocity:

$$\mathbf{U}_{sf} = (1-\varphi_{\mathrm{P}})\mathbf{U}\,. \tag{3.11}$$

One might incorrectly view the dimensionless drag force as a correction term to Stokes drag. Strictly speaking this normalization is only valid for non-moving particles (as in fixed beds), and high Stokes number flows.

The drag force of a class of particles can also be described by a volume-specific friction coefficient β_i, as well as the slip velocity between the (average) particle and fluid velocity:

$$\mathbf{f}_{\mathrm{D},i} = -\beta_i\left(\mathbf{V}_i - \mathbf{U}\right). \tag{3.12}$$

3.1.2 Particle Drag in Monodisperse Gas-Solid Flow

In addition to the particle volume fraction, the particle Reynolds number is needed in many models to characterize the flow in particulate systems. Beetstra et al. (2007, p. 491) define this number for monodisperse systems using:

$$Re = \frac{\left|\mathbf{U}_{\mathrm{sf}}\right| d}{\nu_{\mathrm{f}}} = \frac{\left|\mathbf{U}\right|(1-\varphi_{\mathrm{P}})d}{\nu_{\mathrm{f}}}\,. \tag{3.13}$$

Gidaspow (1994, p. 37), Holloway et al. (2010, p. 1996) and Radl et al. (2012, p. 4) simply replace the gas velocity by the slip velocity

$$Re = \frac{(1-\varphi_\mathrm{P})\left|\mathbf{U}-\mathbf{V}\right| d}{\nu_\mathrm{f}} \tag{3.14}$$

3.1.2.1 The Model of Gidaspow (1994)

One of the earliest models for monodisperse systems was proposed by Gidaspow (1994, pp. 35–37), which is a combination of the model of Ergun (1952) and the one of Wen and Yu (1966). The implemented model, i.e., the cited model of Zhu et al. (2007) in the documentation of CFDEM®, reads:

$$\begin{aligned}
\beta &= 150\frac{\mu_\mathrm{f}}{d^2(1-\varphi_\mathrm{P})} + 1.75\frac{\rho_\mathrm{f}}{d}\left|\mathbf{u}-\mathbf{v}\right| \text{ for } \varphi_\mathrm{P} > 0.2 \\
\beta &= \frac{3}{4}c_\mathrm{D}\frac{(1-\varphi_\mathrm{P})\left|\mathbf{u}-\mathbf{v}\right|\rho_\mathrm{f}}{d}(1-\varphi_\mathrm{P})^{-2.65} \text{ for } \varphi_\mathrm{P} < 0.2 \\
c_\mathrm{D} &= \frac{24}{Re}(1+0.15Re^{0.687}) \text{ for } Re < 1000 \\
c_\mathrm{D} &= 0.44 \text{ for } Re > 1000
\end{aligned} \tag{3.15}$$

3.1.2.2 The Model of Beetstra et al. (2007)

The dimensionless drag force proposed by Beetstra et al. (2007, p. 497) for monodisperse fixed beds consists of a term dependent on the particle volume fraction, and a term to account for a finite particle Reynolds number:

$$\begin{aligned}
F^*_\text{D-fixed} &= \frac{10\varphi_\mathrm{P}}{(1-\varphi_\mathrm{P})^2} + (1-\varphi_\mathrm{P})^2(1+1.5\sqrt{\varphi_\mathrm{P}}) \\
&+ \frac{0.413Re}{24(1-\varphi_\mathrm{P})^2}\left[\frac{(1-\varphi_\mathrm{P})^{-1} + 3\varphi_\mathrm{P}(1-\varphi_\mathrm{P}) + 8.4Re^{-0.343}}{1+10^{3\varphi_\mathrm{P}}Re^{-(1+4\varphi_\mathrm{P})/2}}\right]
\end{aligned} \tag{3.16}$$

Also, most of the models for polydisperse particulate systems are based on this equation.

3.1.3 Particle Drag in Polydisperse Gas-Solid Flow

To establish a drag model for polydisperse systems, each class of particles needs to be characterized by a class-specific particle volume fraction $\varphi_{\mathrm{P},i}$ and a dimensionless diameter

$$y_i = \frac{d_i}{\langle d \rangle} . \tag{3.17}$$

3.1.3.1 The Model of Beetstra et al. (2007)

Beetstra et al. (2007, p. 493) do not assume that "*a particle experiences the same normalized drag force as it would in a monodisperse system of equal overall [particle volume fraction], with the Reynolds number [...] replaced by the individual value*". The latter is according to Beetstra et al. (2007, p. 498):

$$Re_i = \frac{|\mathbf{U}| d_i}{\nu_\mathrm{f}} . \tag{3.18}$$

Instead, Beetstra et al. make use of their dimensionless drag force for monodisperse systems mentioned in the previous section by replacing the particle Reynolds number with an average (mixture) Reynolds number defined as:

$$\langle Re \rangle = \frac{|\mathbf{U}| \langle d \rangle}{\nu_\mathrm{f}} \tag{3.19}$$

They then propose the following equation for calculating the dimensionless drag force of a single class of particles in a polydisperse suspension:

$$F^*_{\mathrm{D},i} = \left[(1-\varphi_\mathrm{P}) y_i + \varphi_\mathrm{P} y_i^2 + 0.064(1-\varphi_\mathrm{P}) y_i^3 \right] F^*_\mathrm{D}(\varphi_\mathrm{P}, \langle Re \rangle) \tag{3.20}$$

3.1.3.2 The Model of Holloway et al. (2010)

This model is based on the work of Yin and Sundaresan (2009). They (2009, p. 1355) account for particle movement by redefining the Reynolds number

$$Re_i = \frac{|\mathbf{V}_i - \mathbf{U}| d_i}{\nu_\mathrm{f}}. \tag{3.21}$$

They (2009, p. 1354) normalize the drag force of a particle (of one class) in analogy to Beetstra et al. (2007) as:

$$F^*_{\mathrm{D},i} = \mathbf{F}_{\mathrm{D},i} / \left[3\pi\mu_\mathrm{f} d_i (1-\varphi_\mathrm{P})(\mathbf{V}_i - \mathbf{U}) \right] \tag{3.22}$$

In such a way the dimensionless drag force can be viewed as a correction factor to Stokes drag. It is easy to relate the dimensionless drag force to the volume-specific friction coefficient by combining Eqn. (3.12) and Eqn. (3.22).

$$\beta_i = \frac{18\varphi_{\mathrm{P},i}(1-\varphi_\mathrm{P})\mu_\mathrm{f}}{d_i^2} F^*_{\mathrm{D},i} \tag{3.23}$$

They (2009, pp. 1358–1359) proposed the following expressions for the dimensionless drag force of one class of particles in a fixed bed:

$$F^*_{\mathrm{D},i\text{-fixed}} = \frac{1}{1-\varphi_\mathrm{P}} + \left(F^*_{\text{D-fixed}} - \frac{1}{1-\varphi_\mathrm{P}} \right)\left(a y_i + (1-a) y_i^2 \right)$$
$$a = 1 - 2.66\varphi_\mathrm{P} + 9.096\varphi_\mathrm{P}^2 - 11.338\varphi_\mathrm{P}^3, \tag{3.24}$$

where the dimensionless drag force for monodisperse systems (i.e., $F^*_{\text{D-fixed}}$) is that of Beetstra et al. (2007) mentioned in the previous section. The more advanced drag model published in the same paper (2009, pp. 1355, 1365) accounts for indirect (so-called “fluid-mediated”) particle-particle interactions by additional terms.

$$\mathbf{f}_{D,i} = -\beta_i \left(\mathbf{V}_i - \mathbf{U} \right) - \sum_{j \neq i} \beta_{ij} (\mathbf{V}_j - \mathbf{V}_i) \tag{3.25}$$

González (2013, p. 20) denotes β_{ij} a fluid-mediated particle-particle drag friction coefficient. To estimate their values, Yin and Sundaresan (2009, p. 1364) propose

$$\beta_{ij} = \frac{2\alpha_{ij}\varphi_{\mathrm{P},i}\varphi_{\mathrm{P},j}}{\dfrac{\varphi_{\mathrm{P},i}}{\beta_i} + \dfrac{\varphi_{\mathrm{P},j}}{\beta_j}} \quad (3.26)$$

where α_{ij} is a logarithmic function of the distance at which the lubrication force between particles begins to saturate:

$$\alpha_{ij} = 1.313\log_{10}(\min(d_i, d_j)/\lambda) - 1.249\,. \quad (3.27)$$

The dimensionless lubrication cutoff length λ / d ranges from 10^{-3} to 10^{-2} in their simulations. Holloway et al. (2010, p. 1997) proposed to replace the particle Reynolds number in the equation for the dimensionless drag force for monodisperse systems of Beetstra et al. (2007) by

$$Re_{\mathrm{mix}} = \frac{\left|\Delta\mathbf{U}_{\mathrm{mix}}\right|(1-\varphi_{\mathrm{P}})\left\langle d\right\rangle}{\nu_{\mathrm{f}}}$$

$$\Delta\mathbf{U}_{\mathrm{mix}} = \frac{\sum \varphi_{\mathrm{P},i}\Delta\mathbf{U}_i}{\sum \varphi_{\mathrm{P},i}} \quad (3.28)$$

3.1.4 Filtered Drag Model

Radl and Sundaresan (2014, pp. 420–421) proposed the following model for a filtered friction coefficient, which is based on simulations of a freely sedimenting suspension in a periodic domain.

$$\frac{\tilde{\beta}_{\mathrm{P}}}{\beta_{\mathrm{P}}} = 1 - f\left(\frac{\Delta_{\mathrm{filter}}}{L_{\mathrm{char}}}, \tilde{\varphi}_{\mathrm{P}}\right) h(\tilde{\varphi}_{\mathrm{P}}) \quad (3.29)$$

$$Fr_{\mathrm{prim}} = u_{\mathrm{t}}^2 / (d_{\mathrm{prim}} g) \quad (3.30)$$

$$L_{\mathrm{char}} = \frac{u_{\mathrm{t}}^2}{g} Fr_{\mathrm{prim}}^{-2/3} \quad (3.31)$$

$$f\left(\frac{\Delta_{\text{filter}}}{L_{\text{char}}}, \tilde{\varphi}_{\text{P}}\right) = \frac{1}{\dfrac{L_{\text{char}}}{\Delta_{\text{filter}}} a(\tilde{\varphi}_{\text{P}}) + 1} \tag{3.32}$$

For Δ_{filter} a characteristic length of the fluid grid cells should be used, i.e., $\Delta_{\text{filter}} = \sqrt[3]{V_{\text{cell}}}$ (Radl and Sundaresan, 2014, p. 418). The function $a(\tilde{\varphi}_{\text{P}})$ is given by the following spline function

$$a(\tilde{\varphi}_{\text{P}}) = a_{0,i} + a_{1,i}(\tilde{\varphi}_{\text{P}} - \varphi_{\text{P,a},i}) + a_{2,i}(\tilde{\varphi}_{\text{P}} - \varphi_{\text{P,a},i})^2 + a_{3,i}(\tilde{\varphi}_{\text{P}} - \varphi_{\text{P,a},i})^3 \tag{3.33}$$

where the coefficients $a_{0,i}$ to $a_{3,i}$, $\varphi_{\text{P,a},i}$ and $h(\tilde{\varphi}_{\text{P}})$ are defined in the same work (2014, p. 420).

The above expressions have been implemented into CFDEM® previously (Radl and Sundaresan, 2014, pp. 420–421). Because the above model has been obtained by filtering data obtained with the drag model of Beetstra et al. (2007), it should be used only in conjunction with this drag model (Radl and Sundaresan, 2014, p. 418). Also, the above model requires tracking all individual particles, i.e., it only compensates for grid size effects. Next, a model that can be applied to parcel-based simulations is introduced.

3.1.5 Filtered Drag Model including Parcel Effects

One computational parcel shares the same volume as the primary physical particles it represents. The parcel diameter normalized by the diameter of the primary particles defines the coarse graining ratio:

$$\alpha = \frac{d_{\text{P}}}{d_{\text{prim}}} \tag{3.34}$$

Radl and Sundaresan (2013, p. 6) propose a correction factor in the drag model mentioned before to account for the contribution of unresolved particle clustering (i.e., particle clustering within a parcel):

$$\frac{\tilde{\beta}_{\mathrm{P}}}{\beta_{\mathrm{P}}} = c_{\mathrm{corr}}(\alpha)\left[1 - f\left(\frac{\Delta_{\mathrm{filter}}}{L_{\mathrm{char}}}, \tilde{\varphi}_{\mathrm{P}}\right) h(\tilde{\varphi}_{\mathrm{P}})\right] \tag{3.35}$$

$$c_{\mathrm{corr}} = a + (1-a)\exp\left[-k(\alpha - 1)\right] \tag{3.36}$$

with the recommended parameters being $a = 0$, and $k = 0.05$. The above expressions have been implemented into CFDEM® as well (Radl and Sundaresan, 2014, pp. 420–422).

3.2 Models for Turbulence

In order to assess the influence of unresolved fluid velocity fluctuations (i.e., "turbulence") on the predictions, a variety of turbulence models are available. Crowe and Group (2006, pp. 13.34–13.36) provide some guidance for particle-laden turbulent flow and also provide a classification map shown in Figure 3.1. Within this thesis the cases are two-way coupled and collisions were tracked for dense suspensions. The Kolmogorov time scale and length scale characterize the smallest, dissipative eddies according to Kolmogorov's theory and are defined there as follows (Pope, 2000, sec. 6.1.2):

$$\tau_{\mathrm{K}} = (\nu / \varepsilon)^{1/2} \text{ and} \tag{3.37}$$

$$l_{\mathrm{K}} = (\nu^3 / \varepsilon)^{1/4}, \tag{3.38}$$

where ν is the fluid molecular viscosity. Larger eddies involve larger viscosity, as well as time and length scales according to the above equations replacing the index K.

ε is the turbulent energy dissipation rate according to the turbulent energy cascade model of Richardson (Pope, 2000, sec. 6.1.1). Richardson considers turbulence as eddies of different sizes with characteristic velocities and time scales. Large eddies are unstable and break up until the smallest eddies are stable enough to be dissipated by molecular viscosity.

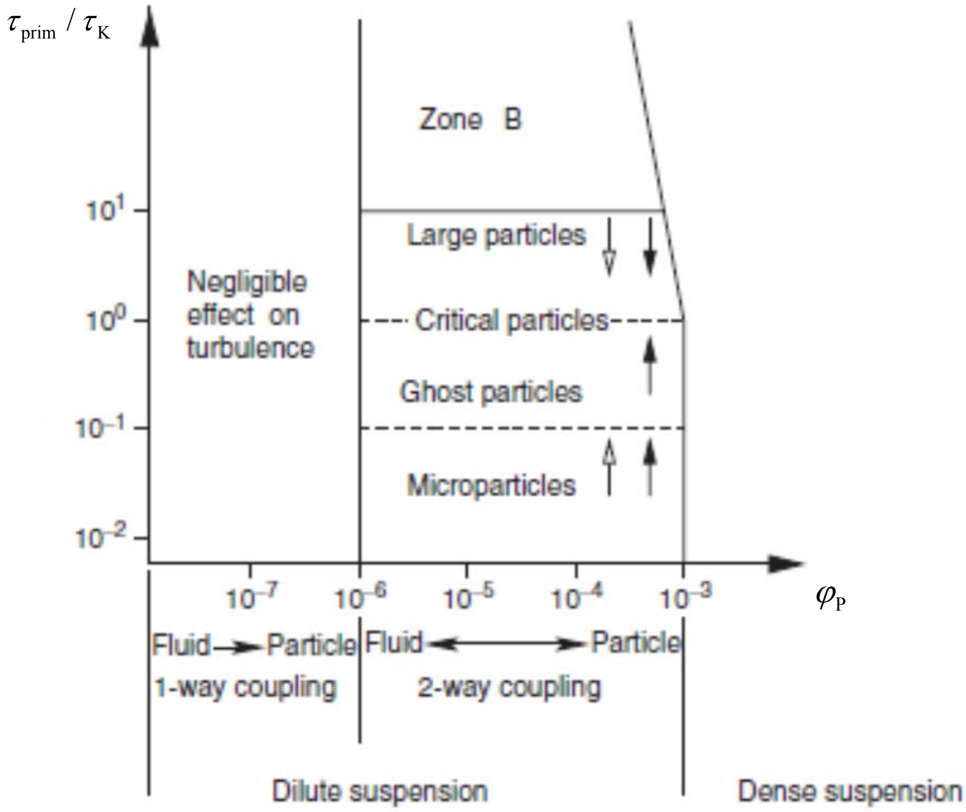

Figure 3.1 Classification map of particles laden turbulent flow (Crowe and Group, 2006, Figure 13.20). For definition of variables see Eqn. (3.1), (3.37) and (4.4).

Finally, the turbulent kinetic energy is defined (Pope, 2000, p. 88) as

$$k = \frac{1}{2}\overline{\mathbf{u}'\cdot\mathbf{u}'} \tag{3.39}$$

and can be interpreted as mean over all directions of velocity fluctuations' kinetic energy.

The Reynolds Averaged Navier-Stokes (RANS) equations are ensemble-averaged mass and momentum balances, and hence they cannot directly predict the instantaneous effect of turbulent eddies. Large Eddy Simulation (LES) resolves eddies larger than the grid scale, and Direct Numerical Simulations (DNS) are capable of resolving the whole spectrum as it solves the Navier-Stokes Equations without simplifications. Turbulence models describe turbulent stress terms, which appear when averaging the Navier-Stokes equations. As particles might influence turbulence locally, LES is the preferred option, and is recommended to be used for large grid-size simulations typical for industrial applications. The remaining sub grid scale turbulent fluid agitation has to be modeled. The models mentioned within this thesis assume isotropic turbulence, i.e., a turbulent field sharing the same turbulent stresses in all directions. De Villiers (2006, pp. 64–66) describes these Smagorinsky-like models in general. Penttinen (2011) describes the implementation of LES models in OpenFOAM®. Additionally, Unsteady-RANS (URANS) should be applied in order to understand the sensitivity with respect to treatment of turbulence in 2D particulate flow. Hence a simple k-ε-model is introduced below as well.

3.2.1 Realizable k-ε-Model

k-ε models solve one equation for the turbulent kinetic energy, and one for the dissipation rate. Moradnia (2010, p. 33) introduces the realizable k-ε model, where realizable refers to the fact that "*A turbulence model is realizable if the normal stresses remain positive*". Moradnia (2010, pp. 33–34) sums up the equations used and the corresponding coefficients, for the original model as well as the implemented one.

Appropriate initial conditions of turbulent quantities must be defined. (N.N. (OpenFOAM Foundation), 2015, sec. 2.1.8.1) provide in their cavity tutorial some guidance, where they suggest using $\mathbf{u'} = 5 \cdot 10^{-2}\,\mathbf{U}$ for the turbulent kinetic energy, and the following relationship for the dissipation rate:

$$\varepsilon = C_\mu^{0.75} k^{1.5} / l\,, \tag{3.40}$$

where $C_\mu = 9 \cdot 10^{-2}$, and the length scale $l = 0.2\,L$, where L is the box with, i.e., the characteristic length.

3.2.2 Smagorinsky Model

Penttinen (2011, p. 20) describes "*In Chapter 3.8.2 of An Introduction to Computational Fluid Dynamics it is stated that Smagorinsky assumed the local SGS stresses [...] to be proportional to the local rate of strain of the resolved flow.*" (Versteeg and Malalasekera, 2007, p. 102). The (filtered) strain rate tensor is (de Villiers, 2006, p. 65):

$$\tilde{\mathbf{S}} = \frac{1}{2}(\nabla \cdot \tilde{\mathbf{u}} + (\nabla \cdot \tilde{\mathbf{u}})^{\mathrm{T}}) \tag{3.41}$$

However, averaging is done differently in RANS and LES. Thus, sub grid scale turbulent kinetic energy and turbulent kinetic energy differ in their values. In OpenFOAM® the scalar quantity k is used for both depending on the turbulence model used. Although the Smagorinsky model typically models only the sub grid scale viscosity (de Villiers, 2006, p. 65), OpenFOAM® models the sub grid scale turbulent kinetic energy also using (Penttinen, 2011, p. 20):

$$k = 2\frac{c_\mathrm{k}}{c_\mathrm{e}}\Delta_\mathrm{filter}^2\left|\tilde{\mathbf{S}}\right|^2 \tag{3.42}$$

$$\nu_\mathrm{SGS} = c_\mathrm{k}\sqrt{k}\Delta_\mathrm{filter} \tag{3.43}$$

with the parameters $c_\mathrm{k} = 7 \cdot 10^{-2}$ and $c_\mathrm{e} = 1.05$ as suggested by de Villiers (2006, p. 68). Note that OpenFOAM® uses as default values $c_\mathrm{k} = 9.4 \cdot 10^{-2}$ and $c_\mathrm{e} = 1.048$ (Penttinen, 2011, p. 20).

3.2.3 One Equation Turbulence Model

This LES-based model solves a transport equation for the sub grid scale turbulent kinetic energy, and hence does not rely on an algebraic relationship between the (filtered) shear rate and the SGS viscosity. Its derivation is detailed in de Villiers' thesis (2006, p. 66)

$$\frac{\partial}{\partial t}k + \nabla\cdot\left(k\tilde{\mathbf{u}}\right) - \nabla\cdot\left[\left(\nu_\mathrm{f} + \nu_\mathrm{SGS}\right)\nabla k\right] = -\varepsilon - \boldsymbol{\tau} : \tilde{\mathbf{S}} \tag{3.44}$$

The last term on the right hand side of this equation represents the decay of turbulence from the resolved scales to the sub grid scales via the energy cascade (de Villiers, 2006, p. 123). Following Penttinen (2011), the same numerical values for constants c_k and c_e are used by default in OpenFOAM®, i.e. $c_\mathrm{k} = 9.4 \cdot 10^{-2}$ and $c_\mathrm{e} = 1.048$.

3.3 *Model for Quenching*

The gas in the fluidized bed is quenched by adding a water spray. Hence, the effect of the quench water on the local water droplet and vapor concentration, as well as on the gas temperature distribution in the riser must be modeled. In case we assume that the injected water evaporates completely, the average gas temperature at the exit of the riser (assuming a perfectly mixed gas) can be calculated using an enthalpy balance (see also Eqn. (3.48)):

$$\begin{aligned}\dot{Q}_{\mathrm{quench}} &= \overline{c_{p,\mathrm{f}}}\,|_{T_{\mathrm{CZ}}}^{T_{\mathrm{FG}}}\,(T_{\mathrm{FG}} - T_{\mathrm{CZ}})\rho_{\mathrm{f}}\dot{V}_{\mathrm{FG}} \\ &= \left[\Delta h_{\mathrm{W,vap}}\,|_{T_0} - c_{p,\mathrm{W}}\,|_{T_{\mathrm{W}}}\,(T_{\mathrm{W}} - T_0) + c_{p,\mathrm{W,vap}}\,|_{T_{\mathrm{CZ}}}\,(T_{\mathrm{CZ}} - T_0)\right]\dot{M}_{\mathrm{W}} \\ &+ \overline{c_{p,\mathrm{P}}}\,|_{T_{\mathrm{P}}}^{T_{\mathrm{CZ}}}\,(T_{\mathrm{CZ}} - T_{\mathrm{P}})\dot{M}_{\mathrm{P}}\end{aligned} \qquad (3.45)$$

The above equation neglects the gas mass entering the riser via the particle injection ports, which is expected to be small. Genuine CFDEM® can handle only a single fluid phase, and particles dispersed therein. In this work only the motion of the process gas (without the water vapor) is modeled, and the influence of water droplets and steam is assumed to be negligible for the flow simulation. Such an assumption is justified for small mass loadings of droplets and steam. This is true (at least in a global sense) for the conditions considered in this study. In order to predict the local temperature, as well as the water vapor and droplet content of the gas, a model was added to the CFDEM® implementation. Specifically, the model was developed as an add-on library for OpenFOAM® to solve the following three transport equations (see Eqn. (3.46) to Eqn. (3.48)) assuming

- a low volume concentration of liquid and evaporated water such that the water does not influence the flow,
- a low volume concentration of particles,
- particles and droplets are in thermal equilibrium with the surrounding fluid, i.e., particles, gas, and droplet phase share the same temperature. We hence consider the mixture enthalpy transport equation for the gas-particle-droplet mixture,
- the gas and the droplet phase share the same flow speed,
- turbulent dispersion is characterized with the same effective diffusivity for heat, water vapor, and droplet transport,
- constant material and transport properties.

Formulating the differential mass and enthalpy balances in terms of the mass loadings μ_i, we get:

$$\frac{\partial \rho_{\mathrm{G}} \mu_{\mathrm{W,vap}}}{\partial t} + \nabla \cdot (\mathbf{u} \rho_{\mathrm{G}} \mu_{\mathrm{W,vap}}) - \nabla \cdot (D_{\mathrm{eff}} \rho_{\mathrm{G}} \nabla \mu_{\mathrm{W,vap}}) = \dot{S}_{\mathrm{vap}} \tag{3.46}$$

$$\frac{\partial \rho_{\mathrm{G}} \mu_{\mathrm{W,l}}}{\partial t} + \nabla \cdot (\mathbf{u} \rho_{\mathrm{G}} \mu_{\mathrm{W,l}}) - \nabla \cdot (D_{\mathrm{eff}} \rho_{\mathrm{G}} \nabla \mu_{\mathrm{W,l}}) = -\dot{S}_{\mathrm{vap}} + \dot{S}_{\mathrm{inject}} \tag{3.47}$$

$$\frac{\partial \left(\rho_{\mathrm{mix}} c_{p,\mathrm{mix}} T \right)}{\partial t} + \nabla \cdot (\mathbf{u} \rho_{\mathrm{mix}} c_{p,\mathrm{mix}} T) - \nabla \cdot (D_{\mathrm{eff}} \rho_{\mathrm{mix}} c_{p,\mathrm{mix}} \nabla T) = \dot{S}_{\mathrm{vap}} \Delta h_{\mathrm{W,vap}} \tag{3.48}$$

$$\dot{S}_{\mathrm{vap}} = \varphi_{\mathrm{W,l}} a_{\mathrm{d}} \beta \Delta \rho_{\mathrm{W,vap}} \text{ where}$$

$$\begin{aligned}
\varphi_{\mathrm{W,l}} &= \frac{\mu_{\mathrm{W,l}}}{\mu_{\mathrm{W,l}} + \mu_{\mathrm{W,vap}} \rho_{\mathrm{W,l}} / \rho_{\mathrm{W,vap}} + \rho_{\mathrm{W,l}} / \rho_{\mathrm{G}}}, \\
a_{\mathrm{d}} &= 6 / d_{\mathrm{d}}, \\
\beta &= Sh \frac{D_{\mathrm{vap}}}{d_{\mathrm{d}}}, \\
\Delta \rho_{\mathrm{W,vap}} &= (\rho_{\mathrm{W,sat}} - \rho_{\mathrm{G}} \mu_{\mathrm{W,vap}}), \\
\rho_{\mathrm{W,sat}} &= \frac{p_{\mathrm{W,sat}}}{R_{\mathrm{W}} T}, \\
p_{\mathrm{W,sat}} &= 133 \cdot 10^{A - \frac{B}{C+T}} \text{ (N.N. (DDBST), 2015)}
\end{aligned} \tag{3.49}$$

$$\dot{S}_{\mathrm{inject}} = \begin{cases} \dot{M}_{\mathrm{W}} / V_{\mathrm{inject}} & \text{within } V_{\mathrm{inject}} \\ 0 & \text{else} \end{cases} \tag{3.50}$$

The local mixture density and the volumetric heat capacity can be calculated from

$$\rho_{\mathrm{mix}} = \frac{\mu_{\mathrm{W,l}} + \mu_{\mathrm{W,vap}} + \mu_{\mathrm{P}} + 1}{\mu_{\mathrm{W,l}} / \rho_{\mathrm{W,l}} + \mu_{\mathrm{W,vap}} / \rho_{\mathrm{W,vap}} + \mu_{\mathrm{P}} / \rho_{\mathrm{P}} + 1 / \rho_{\mathrm{G}}} \tag{3.51}$$

$$\rho_{\mathrm{mix}} c_{p,\mathrm{mix}} = \frac{\mu_{\mathrm{W,l}} c_{p,\mathrm{W,l}} + \mu_{\mathrm{W,vap}} c_{p,\mathrm{W,vap}} + \mu_{\mathrm{P}} c_{p,\mathrm{P}} + c_{p,\mathrm{G}}}{\mu_{\mathrm{W,l}} / \rho_{\mathrm{W,l}} + \mu_{\mathrm{W,vap}} / \rho_{\mathrm{W,vap}} + \mu_{\mathrm{P}} / \rho_{\mathrm{P}} + 1 / \rho_{\mathrm{G}}} \tag{3.52}$$

The mass loading of the particles is

$$\mu_{\mathrm{P}} = \frac{\varphi_{\mathrm{P}} \rho_{\mathrm{P}}}{\left(1 - \varphi_{\mathrm{P}}\right) \rho_{\mathrm{G}}} \tag{3.53}$$

The volume-specific injection source term $\dot{S}_{\mathrm{inject}}$ is defined according to Eqn. (3.50) in a predefined injection region in the riser. The injection source term is named “quenchMuLiq” in the current implementation when using the “specific volume” mode. Note that in case the “absolute volume” mode is used, “quenchMuLiq” equals the quenching water mass flow rate. Also, the dispersion coefficient will be taken to be equal to the effective kinematic viscosity (instead of the effective diffusion coefficient) in case no turbulent Schmidt number is specified. The turbulent Schmidt number relates the effective diffusion coefficient to the effective viscosity, and its value was chosen to be 0.7 in the current study (see Eqn. (3.54) and (Radl and Khinast, 2010, p. 2426)).

$$Sc_{\mathrm{t}} = \frac{\nu_{\mathrm{eff}}}{D_{\mathrm{eff}}} \tag{3.54}$$

The following settings are required in the input dictionaries:

- $\rho_{\mathrm{G}}, \rho_{\mathrm{W,l}}, \rho_{\mathrm{W,vap}}, \rho_{\mathrm{W,P}}$ (rhoGas, rhoLiq, rhoVap, rhoParticle) being the gas density, liquid water density, water vapor density and particle density respectively,
- $c_{p,\mathrm{G}}, c_{p,\mathrm{W,l}}, c_{p,\mathrm{W,vap}}, c_{p,\mathrm{W,P}}$ (cpGas, cpLiq, cpVap, cpParticle) being the gas heat capacity, liquid water heat capacity, water vapor heat capacity and particle heat capacity respectively,
- $\Delta h_{\mathrm{W,vap}}$ (deltaHEvap) being the evaporation enthalpy,
- $t_{\mathrm{vap}} = 1/(a_{\mathrm{d}}\beta)$ (tEvap) being the reciprocal of the specific mass transfer surface area a_{d} of the droplets and the mass transfer coefficient β (see also Eqn. (3.49)).

t_{vap} is a typical time scale for droplet evaporation, and was hold constant in the current study.

All quantities are given in SI units. Constants implemented are $R_W = 462$ J/(kgK) as in (N.N. (VDI), 2006), and the Antoine constants are $A = 8.07$, $B = 1{,}730$ K, and C = -39.7 K as in (N.N. (DDBST), 2015).

3.4 Approximate Particle-to-Fluid Coupling Algorithm

In order to reduce the computation time for the riser simulations, the coupling algorithm has been modified in the latest release of CFDEM®coupling and OpenFOAM® 2.3 such that:

- particle velocity updates due to drag forces (caused by the surrounding gas) are performed implicitly using the Crank-Nicholson scheme (i.e., the fix “couple/cfd/force/integrateImp” in LIGGGHTS®).
- Due to the tight coupling of gas and particle motion, i.e., the extremely small particle size, particles and gas can be assumed to move with almost the same speed. Hence, the momentum balance equation of the mixture (and not that of the gas) has been solved, and coupling forces must not be considered. The treatment as a mixture affects the inertial term in Eqn. (3.55), as well as the gravity term in Eqn. (3.56).

$$\partial_t(\rho_{mix}\mathbf{u}) + \nabla\cdot(\rho_{mix}\mathbf{uu}) = \partial_t(\rho_f\varphi_f\mathbf{u}) + \nabla\cdot(\rho_f\varphi_f\mathbf{uu}) + \partial_t(\rho_P\varphi_P\mathbf{u}) + \nabla\cdot(\rho_P\varphi_P\mathbf{uu}) \qquad (3.55)$$

$$\rho_{mix}\mathbf{g} = \rho_f\varphi_f\mathbf{g} + \rho_P\varphi_P\mathbf{g} \qquad (3.56)$$

Subtracting the hydrostatic pressure (which for a dilute system is $\rho_f\mathbf{g}$), we arrive at the following expression for the gravitational term:

$$\mathbf{f}_g = (\rho_P - \rho_f)\varphi_P\mathbf{g}\,. \qquad (3.57)$$

This term, together with the inertial term above, models the effect of the particles on gas flow in case of a tight fluid-particle coupling. Thus, the effect of fluid-particle drag forces on the gas flow have been approximated, and there is no need to map interaction forces onto the grid. It has been found that such an approach avoids unphysical oscillations in the gas-phase flow field, and allows us to use significantly large time steps on the CFD side.

3.5 Summary of Key Assumptions

1. The fluid phase consists of flue gas and vapor and is considered to be incompressible.
2. Collisions of particles are modeled using a spring-dashpot model assuming soft-sphere interactions with a coefficient of restitution of 0.9.
3. Phases and phase interactions
   3.1. The liquid quenching water volume is considered to be negligible compared to the vapor plus flue gas volume. Hence particles do not absorb liquid (i.e., no direct liquid-particle interaction takes place). Compared to the flue gas, vapor and liquid mass fractions are considered to be low; hence fluid mixture properties are approximated by flue gas properties.
   3.2. A change of droplet size is not explicitly accounted for, i.e., the initial droplet size distribution is considered. However, the droplets' surface area decreases with the local droplet concentration
   3.3. Vapor and liquid water droplets travel with the speed of the gas phase, i.e., inertial effects of the droplet cloud are neglected.
4. The current CFDEM® model does not account for tangentional stresses due to walls, since we use a slip boundary condition for the gas phase. Slip boundary conditions are used to avoid unphysical particle behavior near walls caused by incorrect interpolation of the local fluid velocity.

3.6 Particle Size Distribution

Size distribution data of the particles as provided by the industrial partner is summarized in

$$q_3^* = \frac{\Delta Q_3}{\Delta d / d_{32}} \qquad (3.58)$$

Table 3.1, and illustrated in Figure 3.2. These particles can be classified as Geldart C particles (Geldart, 1973). Since these particles will be cohesive, particle size classes consisting of extremely small particles were merged into one class having a class mean diameter of $5 \cdot 10^{-6}$ m. This also enables a more efficient simulation of the system, since an excessive amount of small particles would have to be used. The modeled particle size distribution is summarized in Table 3.2, and illustrated in Figure 3.2 as well. The new particle size distribution results in the Sauter mean diameter reported in Table 4.2 and Table 5.3. The probability density (reported in

$$q_3^* = \frac{\Delta Q_3}{\Delta d / d_{32}} \tag{3.58}$$

Table 3.1 and Table 3.2) as well as the abscissa in Figure 3.2 is normalized by that Sauter mean diameter according to Eqn. (3.58).

$$q_3^* = \frac{\Delta Q_3}{\Delta d / d_{32}} \tag{3.58}$$

Table 3.1 Original particle size distribution provided by the industrial partner.

i	d_U [µm]	d_O [µm]	d_m [µm]	Q_3 [%]	ΔQ_3 [%]	q_3^* [%]
1	0	1	0.5	4	4	0.272
2	1	2.5	1.75	18	14	0.634
3	2.5	5	3.75	37	19	0.517
4	5	7.5	6.25	51	14	0.381
5	7.5	10	8.75	62	11	0.299
6	10	15	12.5	78	16	0.217
7	15	20	17.5	88	1	0.136
8	20	25	22.5	94	6	$8.16 \cdot 10^{-2}$

9	25	30	27.5	97	3	$4.08 \cdot 10^{-2}$
10	30	40	35	100	3	$2.04 \cdot 10^{-2}$
11	40	50	45	100	0	0
Total					100	

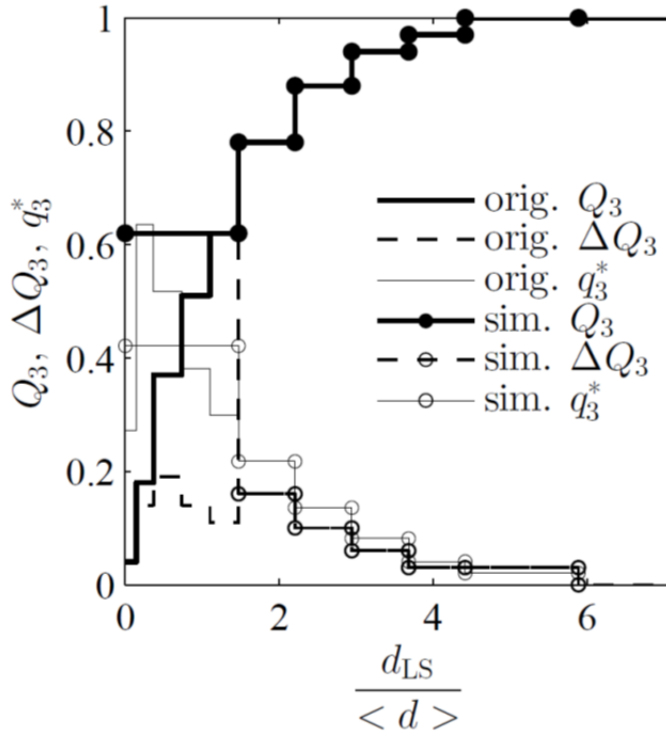

Figure 3.2 Particle size distributions.

Table 3.2 Modeled particle size distribution used in the simulation.

i	d_U [µm]	d_O [µm]	d_m [µm]	Q_3 [%]	ΔQ_3 [%]	q_3^* [%]
1	0	10	5	62	62	0.421
2	10	15	12.5	78	16	0.217
3	15	20	17.5	88	10	0.136
4	20	25	22.5	94	6	$8.16 \cdot 10^{-2}$

5	25	30	27.5	97	3	$4.08 \cdot 10^{-2}$
6	30	40	35	100	3	$2.04 \cdot 10^{-2}$
7	40	50	45	100	0	0
Total					100	

4 Sedimentation in an Unbounded Domain

4.1 Simulation Setup

The simulations of sedimenting gas-particle systems were based on the CFDEM® tutorial example "cfdemSolverPimpleImEx/sedimentationPeriodicBoxBiDisperse". Specifically, the sedimentation of a poly- and mono-disperse particle cloud was investigated in a periodic box. Thus, a computational domain without bounding walls was considered, and particles that move out of the domain are injected at the opposite boundary. Since the domain-averaged slip velocity was considered as the key output of the simulation, the domain size corresponds to the filter size (Radl and Sundaresan, 2014, p. 418).

4.1.1 Simulation Parameters

In the current work the grid spacing was chosen to be twice the largest parcel diameter. In case smoothing of the exchange fields was used, the smoothing length was chosen to be two times[(1)] the largest parcel diameter (see also Radl et al., 2014, p. 3). Initial investigations showed the need for an adjustment of the smoothing length, which is described in Section 4.3. During the variation of parameters, such as particle size distribution, particle volume fraction, drag model or the coarse graining ratio, the key dimensionless length parameters summarized in Table 4.1 were kept constant.

Table 4.1 Dimensionless length parameters for the periodic box simulations.

Parameter	Value
Domain size l_{domain}	$30 l_{\text{grid}}$
Grid spacing l_{grid}	$2 d_{\text{P,max}}$
Smoothing length l_{smooth}	$2 d_{\text{P,max}}$ (1)

[1] Initial guess. During the investigation of coarse graining the scaling law described in Section 4.3 was derived and the smoothing length was verified for each coarse graining used.

The simulation parameters for the CFD part were chosen to be identical to settings in the CFDEM® tutorial "cfdemSolverPimpleImEx/sedimentationPeriodicBoxBiDisperse". Key physical and numerical parameters are summarized in Table 4.2 and Table 4.3, respectively. Discretization schemes, solver settings and other dictionaries detailing solver settings are summarized in the appendix, Section 9.2.2.

Table 4.2 Key physical parameters for the periodic box simulations.

Parameter	Value
Pressure p	0.929 bar
Temperature T	160 °C
Sauter mean diameter $\langle d \rangle$	$6.80 \cdot 10^{-6}$ [m]
Particle density ρ_P	2,250 [kg/m^3]
Fluid density ρ_f	0.804 kg/m^3[2]
Fluid kinematic viscosity ν_f	$3.04 \cdot 10^{-5}$ m^2/s[2]

In numerical computations the Courant number has to be considered, which is defined as (Courant et al., 1928):

$$Co = \frac{U\,\Delta t_{\mathrm{CFD}}}{l_{\mathrm{grid}}} \qquad (4.1)$$

Many books for numerical computation give limits for this number, also known as CFL number, to ensure stability of the solver used (Hirsch, 1988, p. 287). In polydisperse cases the Courant number should not exceed 0.1. This was ensured by choosing an appropriately small (CFD) time step. Surprisingly, the local mean Courant number yielded a maximum of 0.488, which indicates stability issues on a local level as the domain-averaged mean Courant number remained below 0.1.

According to Radl et al. (2012, p. 3), a spring-dashpot model is used in CFDEM® to model collisions of frictional, inelastic spheres as described in the article of Luding (2008).

[2] Air at 160°C and 1 bar (N.N. (VDI), 2006).

The Hertzian model implemented in CFDEM® has been used in the current work. Hence, material properties are chosen for each type of particle-particle contact similarly as reported in Table 4.3.

In order to track particle collisions accurately, the time step for DEM should not exceed a specified collision time scale. CFDEM® issues a warning in case the DEM time step exceeds a predefined fraction (i.e., five percent in this thesis) of both the Hertz and Rayleigh collision time scale. Table 4.3 summarizes both time scales for monodisperse and polydisperse suspensions having a Sauter mean diameter of the modeled size distribution. These values are calculated manually, where the highest velocity of the smallest particle is estimated based on the particle's terminal settling velocity.

Table 4.3 Numerical parameters used in the periodic box simulations.

Parameter	Monodisperse	Polydisperse
Solver	cfdemSolverPimpleImEx[3]	
Young's modulus E	$5 \cdot 10^{7}$ N/m²	
Poisson ratio ν_{P}	0.45	
coefficient of friction μ_{PP}	0.5	
coefficient of restitution e_{PP}	0.9	
Terminal settling velocity of the smallest particle $u_{t,min}$	$2.32 \cdot 10^{-3}$ m/s	$1.25 \cdot 10^{-3}$ m/s
Hertz time scale t_{Hertz}	$1.11 \cdot 10^{-6}$ s	$9.27 \cdot 10^{-7}$ s
Rayleigh time scale $t_{Rayleigh}$	$1.75 \cdot 10^{-7}$ s	$1.29 \cdot 10^{-7}$ s

[3] Here, discrete particles are considered. Hence a setup using the pisoFoam solver (see appendix, Section 9.2.2) turns into a cfdem type solver.

4.1.2 Pre-Processing

A key reference time scale in sedimenting polydisperse suspensions is the acceleration time of an isolated particle having the Sauter mean diameter (Radl et al., 2012, p. 6):

$$t_{\text{ref}} = u_{\text{t}} / g \quad (4.2)$$

After initiation of the simulation with zero fluid and particle velocity, the fluid-particle slip velocity will increase and fluctuate around a pseudo-steady-state value. The time for this initial transient phase was found to be a few multiples of t_{ref}. Thus, approximately 25 times the above reference time scale were sufficient to collect statistically relevant data of the sedimentation behavior of the particle cloud.

The cases studied were split into a base case, and cases that varied one of the parameters of interest. Specifically, the particle volume fraction was varied between $7.78 \cdot 10^{-5}$ and $5 \cdot 10^{-2}$ to cover relevant conditions in the riser. The largest feasible number of particles (or parcels in case coarse graining was used) that can be simulated was limited to $5 \cdot 10^{6}$. Given these preconditions, the domain size was chosen based on the maximal feasible particle number and the highest particle volume fraction. It can be shown that a unit volume of polydisperse suspension (having the modeled size distribution) contains 1.59 more particles than a unit volume of a monodisperse suspension with particles having the corresponding Sauter mean diameter. Thus, the overall domain size can be calculated from the monodisperse suspension and a correction factor. Considering the volume of a single parcel, the total volume of the simulation box containing $N_{\text{P,poly}}$ particles is:

$$V_{\text{tot}} = \frac{\pi d_{\text{P,mono}}^3}{6} \frac{N_{\text{P,poly}}}{1.59} \frac{1}{\langle \varphi_{\text{P}} \rangle} \quad (4.3)$$

This equation can also be used to predict the amount of particles when varying the particle volume fraction for a fixed size of the simulation domain.

In Table 4.4 the terminal settling velocity and the domain-averaged Reynolds number for an isolated particle having the Sauter mean diameter of the modeled size distribution is summarized. Also, the numerical values for the reference time and the total simulation time are displayed in this table.

Table 4.4 Estimation of the total simulation time.

Parameter	Value
Terminal settling velocity u_t	$2.32 \cdot 10^{-3}$ m/s
Reynolds number $\langle Re \rangle$	$5.18 \cdot 10^{-4}$
Reference time t_{ref}	$2.36 \cdot 10^{-4}$ s
Total simulation time t_{sim}	$7 \cdot 10^{-3}$ s

According to Radl et al. (2014, p. 4) the particle relaxation time sets the upper limit for the particle time step. The particle relaxation time is the time a particle needs to accelerate to some fraction of the fluid velocity by means of drag forces. Thus, at most 30 % of the particle relaxation time (of the smallest particle) was chosen as the particle time step. Since $\langle Re \rangle$ << 0.1, Stokes drag law is valid, and we get:

$$\tau_{\mathrm{prim}} = \frac{d_{\mathrm{prim}}^2 \rho_{\mathrm{prim}}}{18 \mu_{\mathrm{f}}} \tag{4.4}$$

The fluid relaxation time sets the upper limit for the fluid time step. The fluid relaxation time is the time the fluid needs to accelerate to a certain velocity by means of drag forces exerted by the suspended particles. At most 5 % of the fluid relaxation time for a particle having the Sauter mean diameter was chosen in cases of explicit force coupling.

$$\tau_{\mathrm{f}} = \frac{d_{\mathrm{prim}}^2 \rho_{\mathrm{f}}}{18 \mu_{\mathrm{f}}} \frac{1 - \varphi_{\mathrm{P}}}{\varphi_{\mathrm{P}}} \tag{4.5}$$

For those dense particulate suspensions in which collisions need to be tracked, the minimum of (i) the Hertzian time scale, (ii) the Rayleigh time scale, and (iii) the particle relaxation time was used as the particle time step. In dilute particulate suspensions collisions occur infrequently, and hence the collision tracking was deactivated for suspensions having a particle volume fraction below 1 %. Consequently, only the particle relaxation time sets the particle time step.

Table 4.5 Particle and fluid relaxation time.

Parameter	$\langle \varphi_P \rangle$	Monodisperse	Polydisperse
particle relaxation time τ_{prim}		$2.36 \cdot 10^{-4}$ s	$1.28 \cdot 10^{-4}$ s
fluid relaxation time τ_f	$5 \cdot 10^{-2}$	$1.61 \cdot 10^{-6}$ s	
	$2 \cdot 10^{-2}$	$4.14 \cdot 10^{-6}$ s	
	10^{-2}	$8.37 \cdot 10^{-6}$ s	
	$5 \cdot 10^{-3}$	$1.68 \cdot 10^{-5}$ s	
	$2 \cdot 10^{-3}$	$4.22 \cdot 10^{-5}$ s	
	10^{-3}	$8.44 \cdot 10^{-5}$ s	
	$5 \cdot 10^{-4}$	$1.69 \cdot 10^{-4}$ s	
	$2 \cdot 10^{-4}$	$4.22 \cdot 10^{-4}$ s	
	$7.78 \cdot 10^{-5}$	$1.10 \cdot 10^{-3}$ s	

Table 4.5 summarizes the particle relaxation time and the fluid relaxation time, Table 4.8 and Table 4.9 summarizes the time steps chosen. In the last two tables, the ratio of the fluid and particle time step is denoted as the coupling factor:

$$CF = \Delta t_{CFD} / \Delta t_{DEM} \quad (4.6)$$

An overview of all considered cases is provided in Table 4.6 to Table 4.9. Note that for dense particulate suspensions (i.e., for $\varphi_P > 10^{-2}$) one case was split into two cases: one for filling the simulation domain with particles, and one for the sedimentation simulation with the particle arrangement from the filling simulations. The logic behind folder and file naming is explained in the appendix, Section 9.2.1.

Table 4.6 Overview of cases (periodic domain simulations using coarse graining $\alpha = 10$).

Model	PSD	$\langle \varphi_P \rangle$	l_{domain}	N_P
Beetstra	poly	$5 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$ m	$4.47 \cdot 10^{6}$
Beetstra		$2 \cdot 10^{-2}$		$1.79 \cdot 10^{6}$
Beetstra		10^{-2}		$8.94 \cdot 10^{5}$
Beetstra		$5 \cdot 10^{-3}$		$4.47 \cdot 10^{5}$
Beetstra		$2 \cdot 10^{-3}$		$1.79 \cdot 10^{5}$
Beetstra		10^{-3}		$8.94 \cdot 10^{4}$
Gidaspow		$5 \cdot 10^{-4}$		$4.47 \cdot 10^{4}$
Gidaspow		$2 \cdot 10^{-4}$		$1.79 \cdot 10^{4}$
Gidaspow		$7.78 \cdot 10^{-5}$		6,900
Beetstra	mono	$5 \cdot 10^{-2}$	$4.2 \cdot 10^{-3}$ m	$2.25 \cdot 10^{4}$
Beetstra		$2 \cdot 10^{-2}$		9,000
Beetstra		10^{-2}		4,500
Beetstra		$5 \cdot 10^{-3}$		2,250
Beetstra		$2 \cdot 10^{-3}$		900
Beetstra		10^{-3}		450
Gidaspow		$5 \cdot 10^{-4}$		225
Gidaspow		$2 \cdot 10^{-4}$		90
Gidaspow		$7.78 \cdot 10^{-5}$		35

Table 4.7 Overview of cases (periodic domain simulations without coarse graining, i.e., $\alpha = 1$).

Model	PSD	$\langle \varphi_P \rangle$	l_{domain}	N_P
Beetstra	poly	$5 \cdot 10^{-3}$	$2.1 \cdot 10^{-3}$ m	$4.47 \cdot 10^5$
Beetstra		$2 \cdot 10^{-3}$		$1.79 \cdot 10^5$
Beetstra		10^{-3}		$8.94 \cdot 10^4$
Gidaspow		$5 \cdot 10^{-4}$		$4.47 \cdot 10^4$
Gidaspow		$2 \cdot 10^{-4}$		$1.79 \cdot 10^4$
Gidaspow		$7.78 \cdot 10^{-5}$		6,900
Beetstra	mono	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$ m	3,800
Beetstra		$2 \cdot 10^{-3}$		1,520
Beetstra		10^{-3}		759
Gidaspow		$5 \cdot 10^{-4}$		380
Gidaspow		$2 \cdot 10^{-4}$		152
Gidaspow		$7.78 \cdot 10^{-5}$		59

Table 4.8 Overview of key time scales used in periodic domain simulations with coarse graining ($\alpha = 10$).

Model	PSD	$\langle \varphi_P \rangle$	Δt_{DEM}	Δt_{CFD}	CF	t_{sim}
Beetstra	poly	$5 \cdot 10^{-2}$	10^{-8} s	10^{-6} s	100	$7 \cdot 10^{-3}$ s
Beetstra		$2 \cdot 10^{-2}$	10^{-8} s	10^{-6} s	100	$7 \cdot 10^{-3}$ s
Beetstra		10^{-2}	10^{-8} s	10^{-5} s	1,000	$7 \cdot 10^{-3}$ s
Beetstra		$5 \cdot 10^{-3}$	10^{-5} s	10^{-5} s	1	0.5 s
Beetstra		$2 \cdot 10^{-3}$	10^{-5} s	10^{-5} s	1	0.5 s
Beetstra		10^{-3}	10^{-5} s	10^{-4} s	10	0.5 s
Gidaspow		$5 \cdot 10^{-4}$	10^{-5} s	10^{-4} s	10	0.5 s
Gidaspow		$2 \cdot 10^{-4}$	10^{-5} s	10^{-4} s	10	0.5 s
Gidaspow		$7.78 \cdot 10^{-5}$	10^{-5} s	10^{-3} s	100	0.5 s
Beetstra	mono	$5 \cdot 10^{-2}$	10^{-8} s	10^{-6} s	100	$7 \cdot 10^{-3}$ s
Beetstra		$2 \cdot 10^{-2}$	10^{-8} s	10^{-6} s	100	$7 \cdot 10^{-3}$ s
Beetstra		10^{-2}	10^{-8} s	10^{-5} s	1,000	$7 \cdot 10^{-3}$ s
Beetstra		$5 \cdot 10^{-3}$	10^{-5} s	10^{-5} s	1	0.5 s
Beetstra		$2 \cdot 10^{-3}$	10^{-5} s	10^{-5} s	1	0.5 s
Beetstra		10^{-3}	10^{-5} s	10^{-4} s	10	0.5 s
Gidaspow		$5 \cdot 10^{-4}$	10^{-5} s	10^{-4} s	10	0.5 s
Gidaspow		$2 \cdot 10^{-4}$	10^{-5} s	10^{-4} s	10	0.5 s
Gidaspow		$7.78 \cdot 10^{-5}$	10^{-5} s	10^{-3} s	100	0.5 s

Table 4.9 Overview of key time scales used in periodic domain simulations without coarse graining (i.e., $\alpha = 1$).

Model	PSD	$\langle \varphi_\mathrm{P} \rangle$	Δt_DEM	Δt_CFD	CF	t_sim
Beetstra	poly	$5 \cdot 10^{-3}$	10^{-5} s	10^{-5} s	1	0.5 s
Beetstra		$2 \cdot 10^{-3}$		10^{-5} s	1	
Beetstra		10^{-3}		10^{-4} s	10	
Gidaspow		$5 \cdot 10^{-4}$		10^{-4} s	10	
Gidaspow		$2 \cdot 10^{-4}$		10^{-4} s	10	
Gidaspow		$7.78 \cdot 10^{-5}$		10^{-3} s	100	
Beetstra	mono	$5 \cdot 10^{-3}$	10^{-5} s	10^{-5} s	1	0.5 s
Beetstra		$2 \cdot 10^{-3}$		10^{-5} s	1	0.5 s
Beetstra		10^{-3}		10^{-4} s	10	0.5 s
Gidaspow		$5 \cdot 10^{-4}$		10^{-4} s	10	0.5 s
Gidaspow		$2 \cdot 10^{-4}$		10^{-4} s	10	0.5 s
Gidaspow		$7.78 \cdot 10^{-5}$		10^{-3} s	100	0.5 s

4.1.3 Post-Processing

For each case the domain-averaged slip velocity, domain-averaged dimensionless momentum error and the maximal overlap of particles was recorded. The domain-averaged slip velocity was calculated according to Eqn. (3.6) and time-averaged over the last 90 % of the simulated time. The domain-averaged dimensionless momentum error is the ratio of the integral momentum of the system and a reference momentum. The momentum error was time-averaged over the last 60 % of the simulation time. The integral momentum is the sum of the particle and the fluid momentum, and should become zero in periodic box simulations (González, 2013, p. 33). However, due to implicit force coupling, Newton's Third Law is not strictly enforced and the integral momentum slowly drifts, which is counteracted with a momentum control algorithm (Radl et al., 2014, p. 5). The reference momentum is the product of the domain-averaged slip velocity and the particles mass. The maximum overlap is 1 minus the ratio of the minimal distance between the particles and the smallest particle diameter.

4.2 Initial Spatial Particle Distribution

Exploratory simulations yielded very small slip velocities, and the formation of meso-scale structures was not observed within a feasible simulation time. Thus, it was concluded that the instability that causes the meso-scale structures propagates very slowly, resulting in infeasible long simulation times. Hence, the initial spatial distribution of the particles in the simulation domain was varied in order to "kick" the instability to form meso-scale structures more rapidly. Thus, the domain was bisected and quadrisected and the particle concentration in each subdomain was set to a different value while keeping the domain-average concentration unchanged. Specifically, in the bi- and quadrisected cases one region contained three times more particles than the other one.

Table 4.10 Base case for initial effects.

Parameter	Value
Domain size l_{domain}	$4.20 \cdot 10^{-3}$ m
Grid spacing l_{grid}	$1.40 \cdot 10^{-4}$ m
Particle volume fraction $\langle \varphi_{\text{P}} \rangle$	10^{-3}
Coarse graining ratio α	1 (off)
Smoothing length l_{smooth}	$7 \cdot 10^{-5}$ m
Drag model	Beetstra
DEM time step Δt_{DEM}	10^{-5} s[(4)]
CFD time step Δt_{CFD}	10^{-4} s
Coupling factor CF	1,000
Simulation time t_{sim}	$7 \cdot 10^{-3}$, 1 s

[4] (Smallest) particles were lost using this time step. Lowering to 10^{-7} s solved this problem.

Key parameters for the simulations to investigate the effect of the initial spatial particle distribution are summarized in Table 4.10.

4.2.1 Results

For monodisperse cases, homogeneous, bisected and quadrisected cases have been investigated. In Quadrisected cases the pseudo-steady-state conditions are expected to be satisfied earlier than in bisected cases. Hence, the bisected case was not conducted for polydisperse cases. Figure 4.1 to Figure 4.5 provide an illustration of key flow features at the beginning of the simulation and after the pseudo-steady-state conditions were satisfied. The domain-averaged slip velocities at pseudo-steady-state conditions are summarized in Table 4.11.

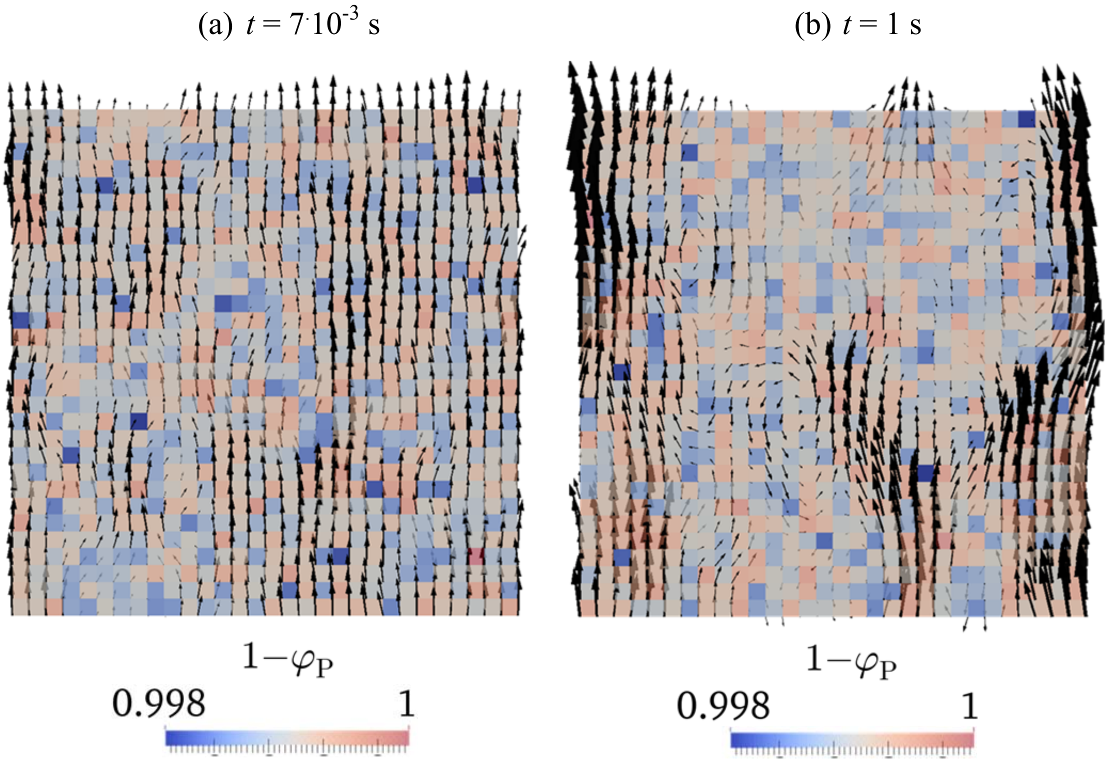

Figure 4.1 Void fraction distribution after (a) $t = 7 \cdot 10^{-3}$ s and (b) $t = 1$ s (monodisperse case, $\varphi_P = 10^{-3}$, particles are initially homogeneously distributed) in a vertical slice located at the center of the domain. Arrows indicate the local gas velocity.

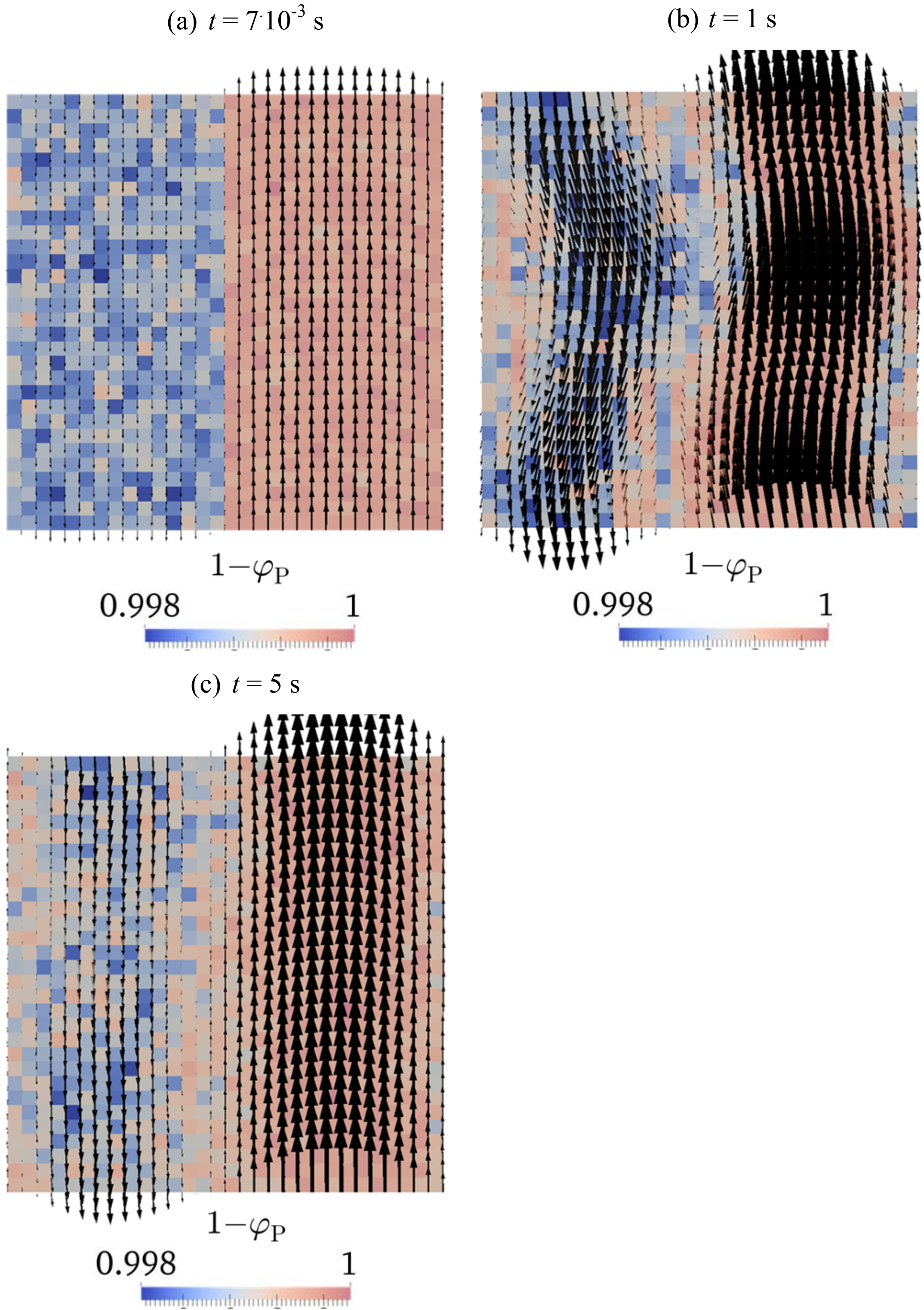

Figure 4.2 Void fraction distribution after (a) $t = 7 \cdot 10^{-3}$ s, (b) $t = 1$ s and (c) $t = 5$ s (monodisperse case, $\varphi_P = 10^{-3}$, particles in each of the two regions are initially homogeneously distributed with one region containing three times the particles of the other one) in a vertical slice located at the center of the domain. Arrows indicate the local gas velocity.

Figure 4.1 indicates that in case the particles are initially homogeneously distributed, they will remain homogeneously dispersed in monodisperse gas-solid suspensions for long times. They are well dispersed, and do not form meso-scale structures.

Figure 4.2 indicates that in the bisected case mixing takes place and streamers are formed, but lateral mixing is very slow leading to high slip velocities at the pseudo-steady state.

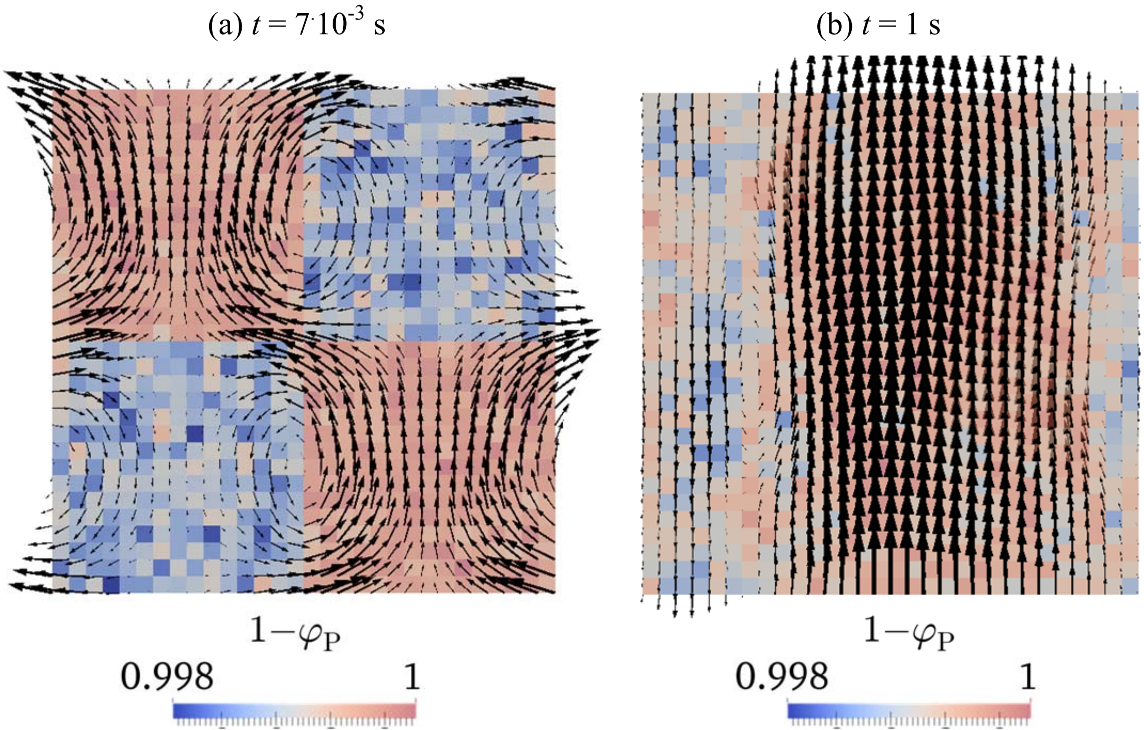

Figure 4.3 Void fraction distribution after (a) $t = 7 \cdot 10^{-3}$ s and (b) $t = 1$ s (monodisperse case, $\varphi_P = 10^{-3}$, particles in each of the four regions are initially homogeneously distributed with two regions containing three times the particles of the other two) in a vertical slice located at the center of the domain. Arrows indicate the local gas velocity.

Figure 4.3 indicates faster mixing in the quadrisected case than in the bisected case. However, the particles tend to form streamers as in the bisected case. Both effects counteract each other, leading to a slip velocity between the bisected and the homogeneous case.

We now compare the results for the polydisperse gas-solid suspensions with that of the monodisperse system. Again, we begin with an initially homogeneously spatially distributed particle cloud. We observe that the domain-averaged slip velocity is increased due to the particle size distribution. However, again the system remains spatially homogeneously distributed even for long times as can be seen in Figure 4.4.

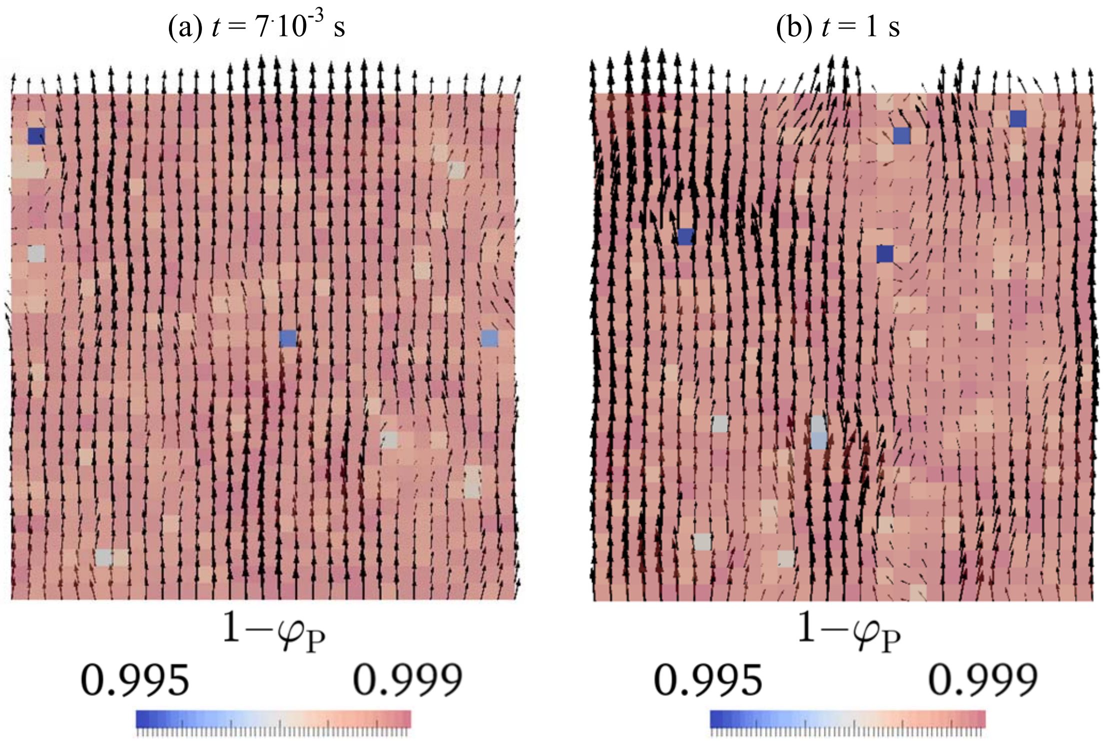

Figure 4.4 Void fraction distribution after (a) $t = 7 \cdot 10^{-3}$ s and (b) $t = 1$ s (polydisperse case, $\varphi_P = 10^{-3}$, particles are initially homogeneously distributed) in a vertical slice located at the center of the domain. Arrows indicate the local gas velocity.

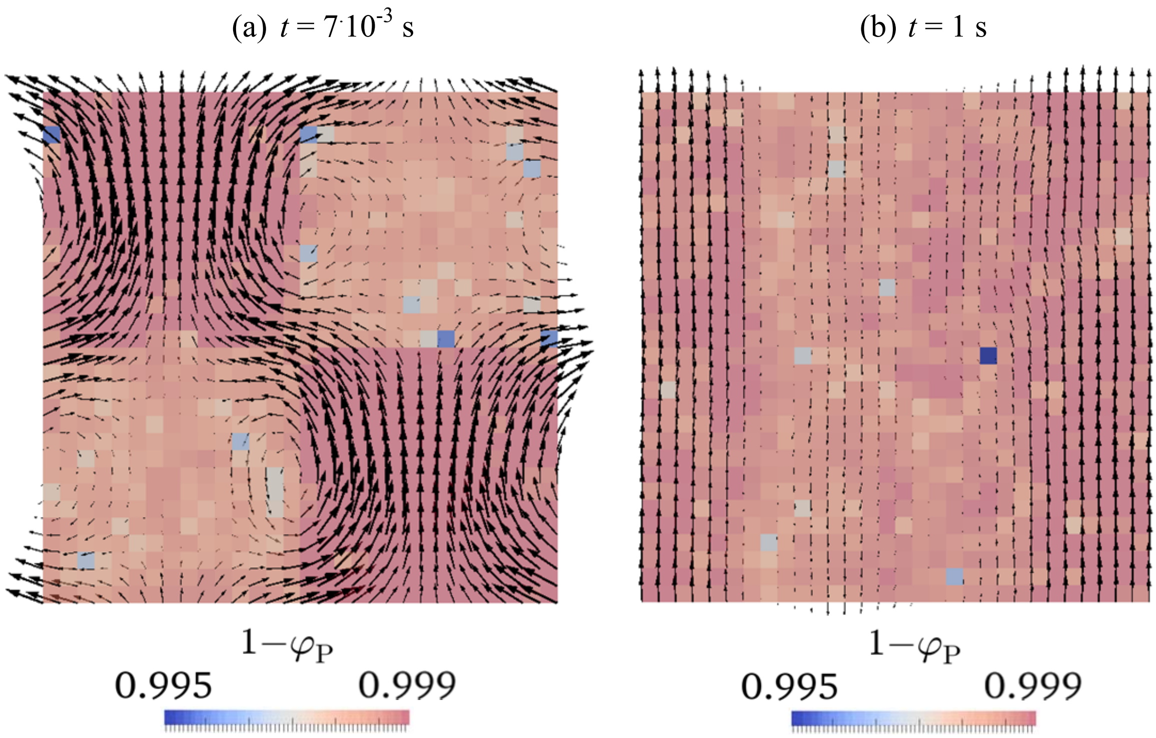

Figure 4.5 Void fraction distribution after (a) $t = 7{\cdot}10^{-3}$ s and (b) $t = 1$ s (polydisperse case, $\varphi_P = 10^{-3}$, particles in each of the four regions are initially homogeneously distributed with two regions containing three times the particles of the other two) in a vertical slice located at the center of the domain. Arrows indicate the local gas velocity.

Figure 4.5 indicates that in the quadrisected polydisperse case the particles segregate only insignificantly, and the system is well mixed due to the agitation provided by big particles.

Table 4.11 Results of initial effects.

$\langle u_{slip} \rangle / u_t$	Monodisperse	Polydisperse
Bisected	6.94	-
Quadrisected	2.48	3.84
Homogeneous	0.995	3.33

The resulting domain-averaged slip velocities at pseudo-steady-state conditions indicate significant dependence on the initial spatial particle distribution for monodisperse suspensions. Specifically, the results for monodisperse gas-solid suspensions show that in case of a homogeneous initial spatial particle distribution the slip velocity is within 0.5 % of the terminal settling velocity. In contrast, polydisperse suspensions are blended well by large particles, leading to a smaller dependency on the initial spatial particle distribution. Specifically, in the polydisperse system a well-mixed pseudo-steady state is achieved independent from start with an error of about 15 %. Also, the higher slip velocity (of a sedimenting polydisperse particle cloud) compared to the monodisperse case with particles having the Sauter mean diameter indicates an effect of the particle size distribution on the average drag force acting on the particles. Hence, polydisperse suspensions were investigated further to eliminate artifacts due to initial conditions.

4.3 Smoothing of Coupling Fields

Parcels with large coarse graining ratios lead to a locally concentrated coupling force, since only the center of mass position of the particles within a parcel is tracked. Locally concentrated coupling forces destabilize the flow, can lead to an unphysical agitation of the fluid, and hence to unphysically large fluid velocities. Thus, coupling forces need to be distributed over a certain region that is affected by the particles. Such a coupling force re-distribution can be realized with a smoothing operation, which should be performed depending on the coarse graining ratio considered in the simulation.

Explorative simulations of coarse-grained particulate systems showed a dramatic increase in the slip velocity due locally concentrated coupling forces. Smoothing is realized by solving a diffusion equation for each transferred quantity γ. Such a smoothing step has been already used in literature (Capecelatro and Desjardins, 2013, pp. 9–10; Pirker et al., 2011, pp. 2481–2483). In the current work, the diffusion coefficient D was chosen based on the CFD time step Δt_{CFD} to realize smoothing with a characteristic length scale l_{smooth} (Radl et al., 2014, p. 3).

$$\frac{\partial \gamma}{\partial t} = D\nabla^2 \gamma \tag{4.7}$$

$$D = l_{\text{smooth}}^2 / \Delta t_{\text{CFD}} \tag{4.8}$$

Radl et al. (2014, p. 3) used $l_{\text{smooth}} / d_{\text{P}} = 3$ in simulations of monodisperse gas-particle suspensions, which was motivated by the work of Capecelatro and Desjardins (2013, p. 10). For polydisperse systems with wide size distribution the latter (2013, p. 10) recommend $l_{\text{smooth}} = d_{\text{P,max}}$.

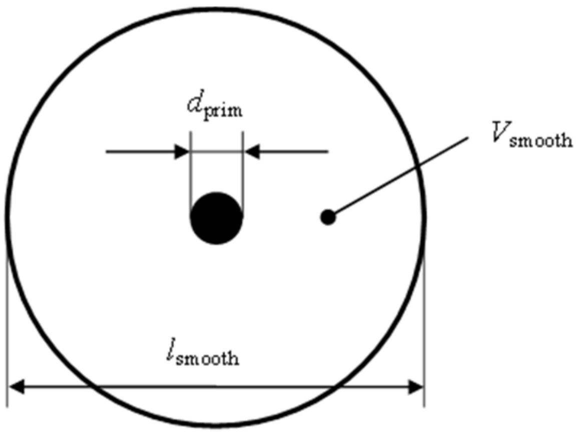

Figure 4.6 Illustration of the sphere of influence around each parcel.

The following derivation is based on the assumption of a spherical region of influence with volume V_{smooth} around each parcel. The particle volume fraction inside this spherical region is assumed to be identical to the domain-averaged particle volume fraction. Hence, one can derive the following equation for the diameter of the sphere of influence:

$$\frac{l_{\text{smooth}}}{d_{\text{prim}}} \approx \frac{\alpha}{\sqrt[3]{\langle \varphi_{\text{P}} \rangle}} \tag{4.9}$$

This law can be used to scale the smoothing length in simulations with different particle concentration. The law was verified by varying the smoothing length until the coarse-grained and not-coarse-grained cases shared the same slip velocity. Key parameters of the simulations to investigate the effect of the smoothing length are summarized in Table 4.12 and Table 4.13.

Table 4.12 Base case for the investigation of the smoothing length effect.

Parameter	Value
Domain size l_{domain}	$4.20 \cdot 10^{-3}$ m
Grid spacing l_{grid}	$1.40 \cdot 10^{-4}$ m
Particle volume fraction $\langle \varphi_{\mathrm{P}} \rangle_{\mathrm{ref}}$	10^{-3}
Drag model	Beetstra
DEM time step Δt_{DEM}	10^{-5} s[(5)]
CFD time step Δt_{CFD}	10^{-4} s
Coupling factor CF	10
Simulation time t_{sim}	$7 \cdot 10^{-2}$ s

Table 4.13 Overview of smoothing length variations.

PSD	$\frac{l_{\mathrm{smooth}}}{d_{\mathrm{P,max}}}$	α	N_{P}
mono	1, 2, 5, 8, 10	1	4,510
mono	1, 2, 5, 8, 10	10	450
poly	1, 2, 5, 8, 10	1	$7.17 \cdot 10^{5}$
poly	1, 2, 5, 8, 10	10	716
poly	1, 2, 5, 8, 10	25	45
poly	1, 2, 5, 8, 10	33	19
poly	1, 2, 5, 8, 10	50	5

[5] (Smallest) particles were lost using this time step in some cases. Lowering to 10^{-7} s solved this problem.

4.3.1 Results

The normalized domain-averaged slip velocities resulting from each case are summarized in Table 4.14 and illustrated in Figure 4.7. The Sauter mean diameter has been chosen as the reference length to enable a comparison between monodisperse and polydisperse suspensions. A correctly applied smoothing length should eliminate the effect of coarse graining on the predicted domain-averaged slip velocity. Thus, we have extracted the recommended smoothing length by intersecting the curves of coarse-grained cases with the predicted slip velocity of a polydisperse particle cloud (and for $\alpha = 1$, i.e., a non-coarse-grained simulation). The results of this analysis are summarized in Table 4.15.

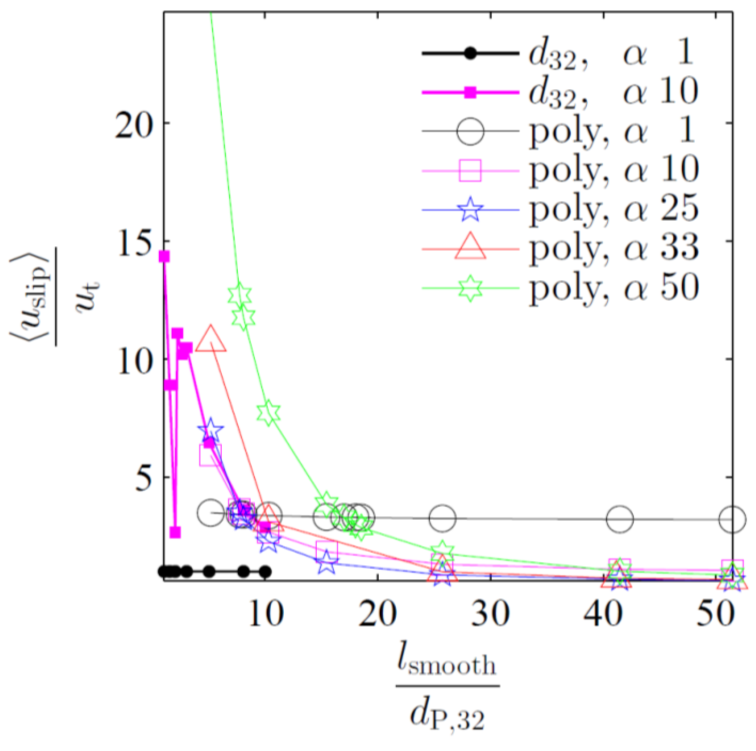

Figure 4.7 Effect of the smoothing length on the predicted slip velocity. Filled symbols refer to monodisperse cases, blank symbols refer to polydisperse cases. The shape of a symbol refers to the coarse graining ratio α. $\varphi_{P,ref} = 10^{-3}$.

Table 4.14 Results of smoothing length variations. $\varphi_{P,ref} = 10^{-3}$.

PSD	$\frac{l_{smooth}}{d_{P,32}}$	$\frac{\langle u_{slip} \rangle}{u_t}$ for $\alpha =$				
		1	10	25	33	50
Mono	1.00	0.980	14.4			
	1.50	0.979	8.89			
	2.00	0.979	2.63			
	2.20		11.1			
	2.60		10.2			
	3.00	0.978	10.5			
	5.00	0.975	6.44			
	8.00	0.970	3.56			
	10.0	0.967	2.87			
Poly	5.15	3.46	5.91	6.94	10.7	24.7
	7.72	3.40	3.60	3.49		12.7
	8.06	3.39	3.42	3.21		11.8
	10.3	3.35	2.65	2.25	3.09	7.70
	15.4	3.29	1.81	1.34		3.81
	17.0	3.28				3.26
	18.0	3.27				2.97
	18.5	3.27				2.84
	25.7	3.23	1.28	0.828	0.960	1.73
	41.2	3.19	1.07	0.644	0.696	0.989
	51.5	3.18	1.02	0.602	0.634	0.823

Table 4.15 Recommended smoothing length for sedimenting polydisperse suspensions ($\varphi_{P,ref} = 10^{-3}$).

α	10	25	33	50
$\frac{l_{smooth}}{d_{P,32}}$	8.06	8.06	10.3	17.0

Also, the recommended smoothing length was adapted and applied to cases of different particle volume fractions. This was done by a proportional correction according to the smoothing law in Eqn. (4.9). Thus, the adapted smoothing law reads:

$$\frac{l_{smooth}}{d_{P,32}} = \left(\frac{l_{smooth}}{d_{P,32}}\right)_{\varphi_{P,ref}} \frac{\sqrt[3]{\langle \varphi_{P,ref} \rangle}}{\sqrt[3]{\langle \varphi_P \rangle}} = \left(\frac{l_{smooth}}{d_{P,32}}\right)_{\varphi_{P,ref}} \frac{1}{10\sqrt[3]{\langle \varphi_P \rangle}} \tag{4.10}$$

Specifically, a linear function has been fitted to the data reported in Table 4.15 (and shown in Figure 4.8) to determine an appropriate smoothing length for the simulations requiring a large coarse graining ratio.

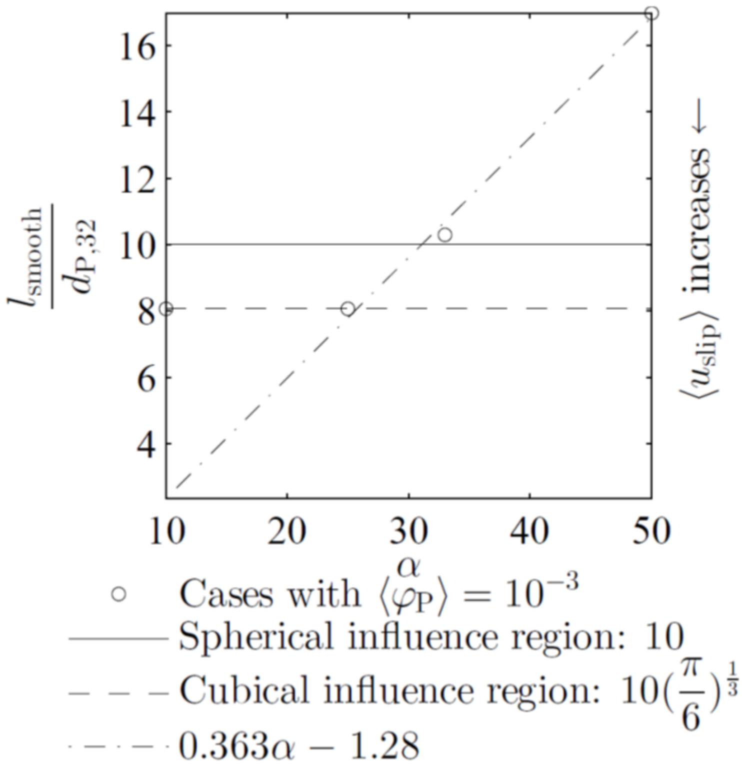

Figure 4.8 Approximation of smoothing length as a function of the coarse graining ratio using a linear function.

As can be seen, a constant relationship between the smoothing length and the parcel diameter fits the data for $\alpha < 25$ reasonably well, supporting our idea of the influence region of each parcel.

The linear function was an interpolation excluding the case of the low coarse graining ratio (i.e., α = 10). Extrapolation of the above linear function is not reasonable for very large coarse graining ratios as explained by the following example: Considering the setup of the 2D riser simulation (see Section 5.4.6), the coarse graining ratio with 1,020 is much larger than in the cases of the unbounded domain simulations performed within this study. If the 2D riser simulation ends up in the range of the reference particle volume fraction of this section (10^{-3}), the required smoothing length would be about 2.5 m. Using such a large influence region of a single parcel, meso-scale structures are expected to be completely suppressed. Hence, the scaling law according to Eqn. (4.9) was applied to the 2D model and the 3D model of the riser (reported in Table 5.14 and Table 5.18). Finally, Figure 4.8 indicates an increase in the domain-averaged slip velocity in case of using too small smoothing length for parcels of a particular size (see also Figure 4.7).

4.4 Simulations in Large Domains

In order to apply the results of the periodic domain simulations to the 2D and 3D cases of the riser, cases with larger domain size were performed. These cases are intended to mimic average cells of the riser, i.e., the domain size is now taken to be equal to the average size of a single cell in the 2D and 3D model of the riser. Hence, the domain size is calculated from the average grid volume of the riser CFD model. Next, the maximum number of parcels was set to $2 \cdot 10^6$ by adjusting the coarse graining ratio. Also, the smoothing length was calculated utilizing the correction in Eqn. (4.10) to Eqn. (4.9), which also accounts for the different particle volume fractions. 2D quantities are normalized by the mesh depth (see Section 5.4.1: 0.532 m). Table 4.16 summarizes the parameters of the simulations featuring larger domain sizes.

$$\alpha = \sqrt[3]{\frac{N_{\mathrm{prim}}}{N_{\mathrm{P,max}}}} \qquad (4.11)$$

Table 4.16 Overview of cases with a larger domain size.

Parameter	3D	2D
Riser: total volume V_{tot}, total area A_{tot},	3,870 m^3	367 m^2
Number of cells in riser simulation	$1.11 \cdot 10^6$	$2.62 \cdot 10^4$
Domain size l_{domain}	0.151 m	0.118 m
Grid spacing l_{grid}	$4.20 \cdot 10^{-3}$ m	
Grid resolution l_{domain} / l_{grid}	36	28
Expected particle volume fraction $\langle \varphi_P \rangle_{exp}$	$2.22 \cdot 10^{-3}$	$1.11 \cdot 10^{-3}$
Drag model	Beetstra	
Coarse graining ratio α	33	33
Smoothing length l_{smooth}	$1.77 \cdot 10^{-3}$ m	$2.23 \cdot 10^{-3}$ m
DEM time step Δt_{DEM}	10^{-5} s(6)	
CFD time step Δt_{CFD}	10^{-5} s	10^{-4} s
Coupling factor CF	100	1,000
Simulation time t_{sim}	$7 \cdot 10^{-2}$ s	

4.4.1 Results

The resulting domain-averaged slip velocity and the domain-averaged dimensionless momentum error are summarized in Table 4.17. The slip velocity is larger than in similar cases with smaller domains (shown in Figure 4.9). Figure 4.9 reveals no clear effect of the domain size on the slip velocity for polydisperse cases of equal coarse graining ratio in the first place. For clarification of the effects, relevant parameters are displayed in the figure as well. First, the low number of parcels in the case of the previously used (i.e., small) unbounded domain may not give a meaningful slip velocity. Furthermore, the domain size

[6] (Smallest) particles were lost using this time step. Lowering to 10^{-7} s solved this problem.

and the mean particle volume fraction affect the results to some extent. Overall, we observe domain-averaged slip velocities up to ca. 8 times the terminal settling velocity.

Table 4.17 Results for simulations using a larger domain size.

Variable	3D	2D
$\langle u_{slip} \rangle / u_t$	5.82	7.68
$\frac{\textit{integral momentum error}}{\textit{reference momentum}}$	$-9.52 \cdot 10^{-2}$	-0.482

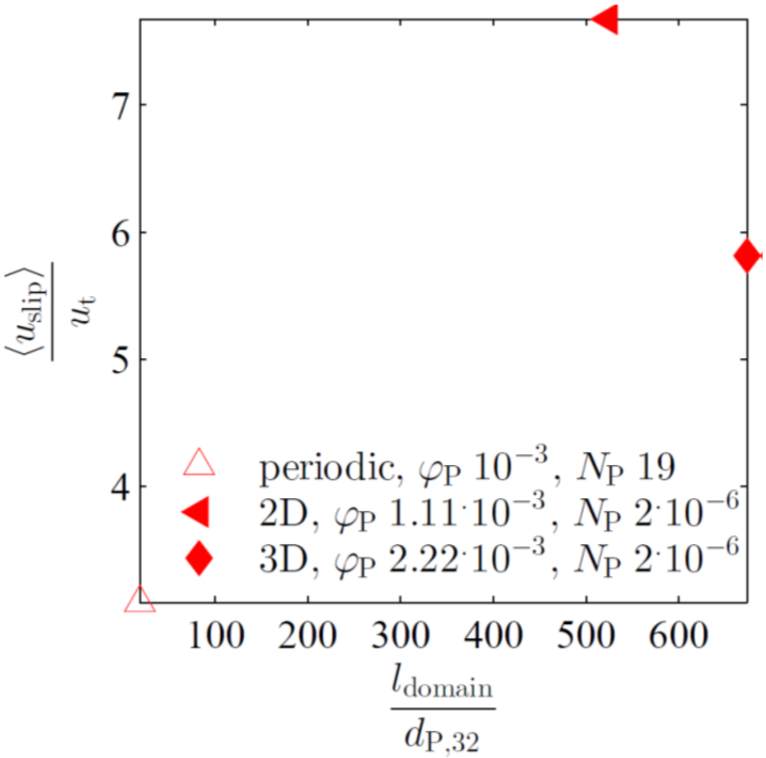

Figure 4.9 Effect of the domain size on the predicted slip velocity (polydisperse, *α* = 33). The blank triangle refers to a case with unbounded domain (size $4.2 \cdot 10^{-3}$ m). Filled symbols refer to cases with domains of an average cell in 2D (domain size 0.118 m, triangle) and 3D riser models (domain size 0.151 m, diamond).

4.5 Drag Correction

In contrast to the previous cases, the cases for the determination of the drag correction factor should be setup in accordance with Table 4.6 to Table 4.9. This means keeping the dimensionless grid resolution constant instead of the domain size. The domain-averaged slip velocity was monitored for a broad range of particle volume fractions and coarse graining ratios in preliminary cases. Unfortunately, the number of cases had to be reduced as only the model of Beetstra et al. (2007) showed meaningful results for the polydisperse gas-particle suspension. This means, the simulations using this model yielded domain-averaged slip velocities with magnitudes in the order of the terminal settling velocity of a single particle having the Sauter mean diameter. In what follows, the cases were setup using the smoothing length found in Section 4.3.1. The number of cases was reduced to simulate two coarse graining ratios and two particle volume fractions. The domain-averaged slip velocities and domain-averaged dimensionless momenta are summarized in Table 4.18. It can be seen that as the particle volume fraction increases, the domain-averaged slip velocity also increases. Also, the ratio of integral momentum error to reference momentum is still large due to the relative grid size of 2 (as reported in Table 4.1). A larger relative grid size could reduce the error. Unfortunaltely, the coarse graining ratio in addition has a strong effect on the measured slip velocity. This indicates that still a large uncertainty is associated when using a parcel-based approach and large coarse graining ratios.

Table 4.18 Results of drag model investigation.

Model	PSD	$\langle \varphi_P \rangle$	α	$\langle u_{slip} \rangle / u_t$	$\frac{integral\ momentum\ error}{reference\ momentum}$
Beetstra	poly	$2 \cdot 10^{-3}$	10	8.77	-0.317
Beetstra	poly	$9.99 \cdot 10^{-4}$	10	8.53	-0.346
Beetstra	poly	$2 \cdot 10^{-3}$	1	3.24	-0.520
Beetstra	poly	$9.9 \cdot 10^{-4}$	1	3.12	-0.665

5 Full-Scale Fluidized Bed Setup

5.1 The Fluidized Bed Setup

Here a brief description of the fluidized bed provided by the industrial partner shall be given. It is used to clean flue gas by recirculating particles that adsorb pollutants (see Reissner et al., 2003). Figure 5.1 illustrates the riser; all dimensions can be found in the dimensional drawing provided in the appendix, Section 9.1.

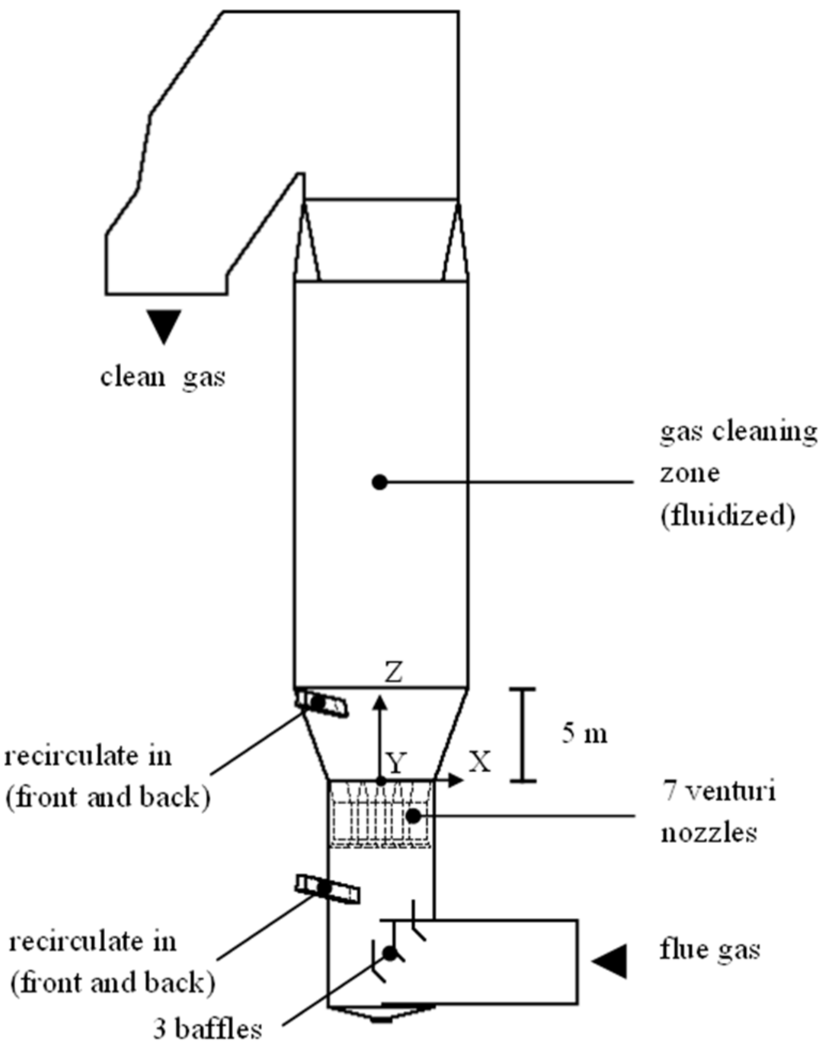

Figure 5.1 Frontal view of the riser.

Recirculate enters either above or below the venturi nozzles via two of the four chutes. Cooling water enters via an additional inlet (not shown in Figure 5.1) at the same height as the recirculate chutes above the venturi nozzles. Flow rates and sectional dimensions are summarized in Table 5.1.

Table 5.1 Riser dimensions and operating conditions.

Section	Dimensions	Temperature, pressure	Flow rate	Velocity
Overall dimensions	24.8 x 53.3 x 14.6 m W x H x D			
Flue gas inlet (FG)	4.37 x 5.56 m H x D	160 °C, 0.929 bar	G $1.6 \cdot 10^6$ m^3/h	G 18.3 m/s
Clean gas outlet (CG)	6.4 x 8.13 m W x D		G + R + W $1.6 \cdot 10^6$ m^3/h	8.54 m/s
Diameter of cleaning zone (CZ)	9.14 m			G 6.77 m/s
Diameter below nozzles (BN)	5.56 m			G 18.3 m/s
each nozzle	1.27 x 1.68 x 3.51 m D_i x D_o x H		G $2.28 \cdot 10^5$ m^3/h	G 50 m/s
each chute below nozzles	0.532 x 0.749 x 5.63 m W x H x L		-	-
each chute above nozzles (R)	0.532 x 0.749 x 4.23 m W x H x L	86 °C	R $2.8 \cdot 10^5$ kg/h (per chute, 2 chutes total)	G 2 m/s R 2 m/s
Quench water inlet	$1.8 \cdot 10^{-2}$ m D	20 °C	W $4.6 \cdot 10^4$ kg/h	W 50 m/s

Dimensions and operating conditions were provided by the industrial partner (Gronald, 2014a). Flow rates were converted to average velocities and imposed at the inlets. In case the velocity was provided in addition to the flow rate by the industrial partner, a volume equivalent cross-sectional diameter was calculated (for the quench water inlet). It is assumed that the conveying air velocity and the velocity of the recirculated particles equilibrate in the chutes. Hence, the average velocity of recirculate plus the conveying air phase was considered for the particle injection calculations in Section 5.2. To convert mass rates and volume rates, the pure density of water and of clean air at given temperatures were assumed. These values, the given pure density of recirculate and other assumed physical properties at given operating conditions used in this thesis are summarized in Table 5.2. The values are based on the reference book “VDI Wärmeatlas” (2006).

Table 5.2 Physical properties.

Property	Value
Particle density ρ_P	2,250 kg/m^3
Water density ρ_W	1,000 kg/m^3
Fluid density ρ_f	0.804 kg/m^3
Fluid kinematic viscosity ν_f	$3.04 \cdot 10^{-5}$ m^2/s

According to Reissner et al. (2003, p. 66) the riser should operate in the circulating fast fluidization regime. However, it is operating in the dilute pneumatic conveying regime according to Crowe and Group (2006, pp. 5.4–5.5, 5.9) due to the high gas velocity.

In order to assess the particulate flow, additional parameters might be of interest. Table 5.3 summarizes them, where the Sauter mean diameter and the terminal settling velocity of a single particle having that diameter are calculated. The Reynolds number, the particle volume fraction, and the mass loading are calculated based on the inlet flow rates (see equations below). $\langle \varepsilon_P \rangle_{exp}$ and $\langle \mu_P \rangle_{exp}$ are the expected particle hold up and mass loading in the cleaning zone, respectively, and were provided by the industrial partner (Gronald, 2014b) based on experience from the operation of the fluidized bed. Thus, the expected hold up is ca. 14 times larger than that calculated from the inlet flow rates, indicating that (i) there is a substantial (mean) slip velocity between particles and gas, and/or (ii) that particles sediment in the cleaning zone (CZ) of the fluidized bed. Clearly, the terminal settling velocity u_{t} is much smaller than the mean flow velocity in the CZ. This indicates that clustering phenomena and an inhomogeneous gas velocity distribution in the CZ must be present, resulting in a substantial (mean) slip velocity in the riser.

$$Re_{\mathrm{CZ}} = \frac{\mathbf{U}_{\mathrm{CZ}} D_{\mathrm{CZ}}}{\nu_{\mathrm{f}}} \quad (5.1)$$

$$\langle \varphi_{\mathrm{P}} \rangle = \frac{\sum \dot{M}_{\mathrm{R}} / \rho_{\mathrm{P}}}{\dot{V}_{\mathrm{CG}}} \quad (5.2)$$

$$\langle \mu_{\mathrm{P}} \rangle = \frac{\sum \dot{M}_{\mathrm{R}}}{\dot{V}_{\mathrm{FG}} \rho_{\mathrm{f}}} \quad (5.3)$$

Table 5.3 Riser flow characteristics.

Parameter	Value
Sauter mean diameter $\langle d \rangle$	$6.80 \cdot 10^{-6}$ m
Terminal settling velocity u_{t}	$2.32 \cdot 10^{-3}$ m/s
Reynolds number Re_{CZ}	$2.04 \cdot 10^{6}$
Particle volume fraction $\langle \varphi_{\mathrm{P}} \rangle$	$1.56 \cdot 10^{-4}$
Mass loading $\langle \mu_{\mathrm{P}} \rangle$	0.436
Expected hold up $\langle \varepsilon_{\mathrm{P}} \rangle_{\mathrm{exp}}$	5 kg/m^3
Expected mass loading $\langle \mu_{\mathrm{P}} \rangle_{\mathrm{exp}}$	6.2
Velocity ratio $U_{\mathrm{R}} / U_{\mathrm{CZ}}$	0.295

5.2 Particle Injection Parameters

In Figure 5.2 a sketch illustrates the relevant geometry for particle injection. In 3D cases the particles enter via both chutes above the nozzles, and in 2D cases via one chute (Holzinger, 2014a). The velocity has been assumed to vary between 0.165 m/s (conveying air velocity) and 3.18 m/s (i.e., the free fall velocity) (Gronald, 2014b). The velocity of free fall has been calculated for the above chute having a length of 4.23 m inclined 83° against the vertical wall using Eqn. (5.4).

$$U_{\text{inject}} = \sqrt{2\left|\mathbf{g}\right| l \cos(\alpha_Z)} \tag{5.4}$$

It has been assumed that conveying air and particles move with the same speed in the chute, and that gravity acts in z-direction, as can be seen in Figure 5.2.

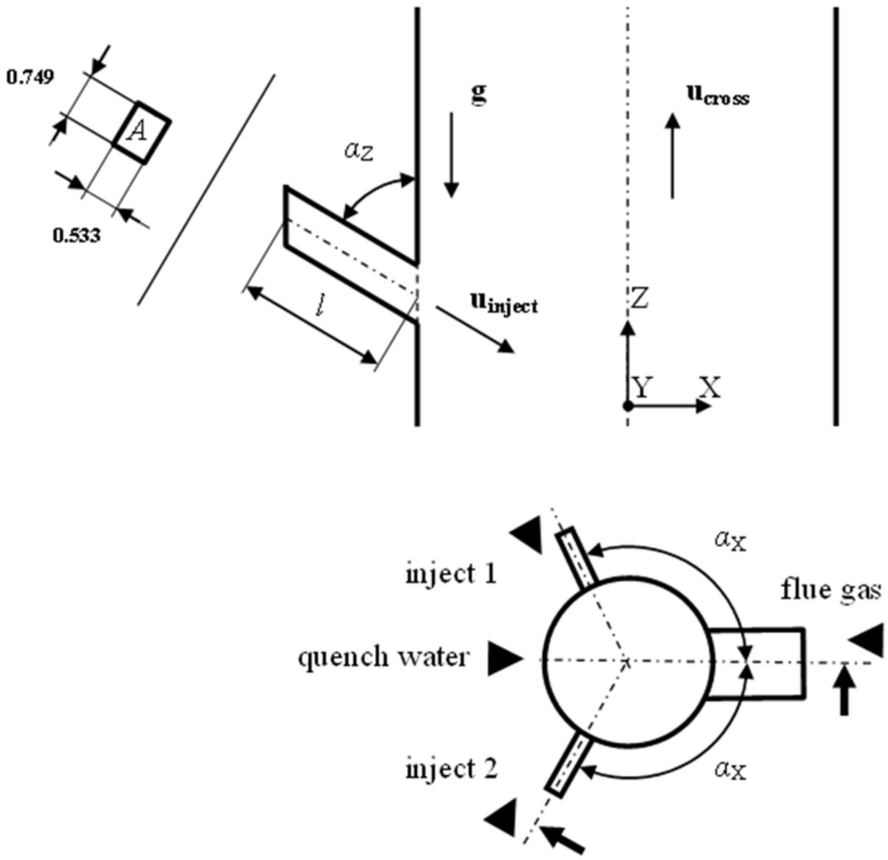

Figure 5.2 Sketch of particle injection.

One might be interested in dimensionless sizes, so $U_{\text{inject}} / U_{\text{cross}}$ has been calculated, where U_{cross} = 6.76 m/s is the average vertical gas velocity in the riser main body.

The average velocity is needed in CFD component-by-component, which can be computed for 2D cases from:

$$U_{\text{X,inject,2D}} = U_{\text{inject}} \sin(\alpha_{\text{Z}}) \text{, and} \tag{5.5}$$

$$U_{\text{Z,inject,2D}} = -U_{\text{inject}} \cos(\alpha_{\text{Z}}) \text{;} \tag{5.6}$$

and for 3D cases from:

$$U_{\text{X,inject,3D}} = -U_{\text{inject}} \sin(\alpha_{\text{Z}}) \cos(\alpha_{\text{X}}) \text{,} \tag{5.7}$$

$$U_{\text{Y,inject,3D}} = \pm U_{\text{inject}} \sin(\alpha_{\text{Z}}) \sin(\alpha_{\text{X}}) \text{, and} \tag{5.8}$$

$$U_{\text{Z,inject,3D}} = -U_{\text{inject}} \cos(\alpha_{\text{Z}}) \text{.} \tag{5.9}$$

Since the particle volume fraction cannot exceed ca. 0.5, one can compute a minimum injection velocity of particles and gas assuming that the inlet chute is completely filled with particles:

$$U_{\text{inject,min}} \approx 2\dot{M}_{\text{P,inject}} / \left(A\rho_{\text{P}} \right) \tag{5.10}$$

Here $\dot{M}_{\text{P}}$ = 77.8 kg/s is the particle feed rate per chute and A = 0.749·0.533 m^2 is the cross-sectional area of the injection chute, shown in Figure 5.2. Recirculate is fed into the riser either via the two upper chutes, or the two lower chutes. This gives a minimum injection velocity of 0.173 m/s.

Since both phases have been assumed to share the same velocity, the mass flow rate for each phase can be computed from:

$$\dot{M}_{\text{P,inject}} = U_{\text{inject}} A \varphi_{\text{P,inject}} \rho_{\text{P}} \text{, and} \tag{5.11}$$

$$\dot{M}_{\text{f,inject}} = U_{\text{inject}} \, A\left(1-\varphi_{\text{P,inject}}\right)\rho_{\text{f}} \, . \tag{5.12}$$

Thus, the total injected mass rate is:

$$\dot{M}_{\text{inject}} = U_{\text{inject}} \, A\left[\varphi_{\text{P,inject}} \, \rho_{\text{P}} + (1-\varphi_{\text{P,inject}})\,\rho_{\text{f}}\right]. \tag{5.13}$$

Here $\rho_{\text{P}} = 2{,}250$ kg/m^3 is the particle density and $\rho_{\text{f}} = 0.804$ kg/m^3 is the fluid density. Now, the mass flow rates can be calculated easily using Eqn. (5.11) and (5.12). One might be interested in volumetric flow rates of each phase, which can be calculated from:

$$\dot{V}_i = \dot{M}_i \, / \, \rho_i \, . \tag{5.14}$$

Furthermore, the mass loading is defined as

$$\mu_{\text{P},i} = \frac{\dot{M}_{\text{P},i}}{\dot{M}_{\text{f},i}} = \frac{\rho_{\text{P}}\varphi_{\text{P},i}}{\rho_{\text{f}}(1-\varphi_{\text{P},i})} \tag{5.15}$$

and the particle concentration (in [kg/m^3])

$$\varepsilon_{\text{P},i} = \varphi_{\text{P},i}\rho_{\text{P}} \, , \tag{5.16}$$

where i indicates the chosen particle class. Considering now the flow in the riser, mass rate and particle volume fraction have been calculated assuming a purely vertical gas cross flow. Thus, it follows that:

$$\dot{M}_{\text{f,cross}} = \dot{V}_{\text{cross}}\rho_{\text{f}} \tag{5.17}$$

$$\left\langle \varphi_{\text{P}} \right\rangle = \frac{\sum \dot{V}_{\text{P,inject}}}{\dot{V}_{\text{cross}} + \sum(\dot{V}_{\text{f,inject}} + \dot{V}_{\text{P,inject}})} \tag{5.18}$$

Table 5.4 summarizes parameters, flow rates and recirculate quantities at the inlet and the subsequent Table 5.5 summarizes recirculate quantities that describe recirculate within the riser. All quantities in both tables are calculated for the chosen injection velocities.

Table 5.4 Calculated quantities for recirculate injection.

Parameter / Variable	Unit	Value			
$\dot{M}_{\mathrm{P,inject}}$	[kg/s]	77.8			
U_{inject}	[m/s]	0.173	1	2	3.18
α_{X}	[°]	120			
α_{Z}	[°]	83			
$\dot{V}_{\mathrm{cross}}$	[m^3/s]	444			
$U_{\mathrm{inject}} / U_{\mathrm{cross}}$		$2.56 \cdot 10^{-2}$	0.148	0.296	0.471
$U_{\mathrm{X,inject,2D}}$	[m/s]	0.172	0.993	1.99	3.16
$U_{\mathrm{X,inject,3D}}$	[m/s]	$8.59 \cdot 10^{-2}$	0.496	0.993	1.58
$U_{\mathrm{Y,inject,3D}}$	[m/s]	±0.149	±0.860	±1.72	±2.73
$U_{\mathrm{Z,inject}}$	[m/s]	$-2.11 \cdot 10^{-2}$	-0.122	-0.244	-0.388
$\varphi_{\mathrm{P,inject}}$	[m³/m³]	0.5	$8.66 \cdot 10^{-2}$	$4.33 \cdot 10^{-2}$	$2.72 \cdot 10^{-2}$
$\dot{V}_{\mathrm{P,inject}}$	[m³/s]	$3.46 \cdot 10^{-2}$			
$\dot{V}_{\mathrm{f,inject}}$	[m³/s]	$3.46 \cdot 10^{-2}$	0.364	0.763	1.23
$\dot{M}_{\mathrm{f,inject}}$	[kg/s]	$2.78 \cdot 10^{-2}$	0.293	0.614	0.992
$\dot{M}_{\mathrm{f,cross}}$	[kg/s]	357			
$\mu_{\mathrm{P,inject}}$	[kg/kg]	2,800	265	127	78.4
$\varepsilon_{\mathrm{P,inject}}$	[kg/m³]	1,130	195	97.5	61.3

Table 5.5 Calculated quantities for recirculate within the riser.

Variable	Unit	Value			
$\langle \varphi_P \rangle_{2D}$	[m³/m³]	$7.79 \cdot 10^{-5}$	$7.78 \cdot 10^{-5}$	$7.78 \cdot 10^{-5}$	$7.77 \cdot 10^{-5}$
$\langle \mu_P \rangle_{2D}$	[kg/kg]	0.218	0.218	0.218	0.217
$\langle \varepsilon_P \rangle_{2D}$	[kg/m³]	0.175	0.175	0.175	0.175
$\langle \varphi_P \rangle_{3D}$	[m³/m³]	$1.56 \cdot 10^{-4}$	$1.56 \cdot 10^{-4}$	$1.56 \cdot 10^{-4}$	$1.55 \cdot 10^{-4}$
$\langle \mu_P \rangle_{3D}$	[kg/kg]	0.436	0.436	0.436	0.434
$\langle \varepsilon_P \rangle_{3D}$	[kg/m³]	0.35	0.35	0.35	0.35

5.3 Drag Correction

Similar to Parmentier et al. (2012, pp. 1087–1088), drag is easily corrected by an additional factor to the domain-averaged slip velocity. This is the reciprocal of the normalized domain-averaged slip velocity as in Eqn. (5.19). The terminal settling velocity of a single particle having the Sauter mean diameter of the particle cloud is an approximation to a homogeneous dilute suspension.

$$\mathbf{f}_{\mathrm{corr}} = \frac{\mathbf{u}_{\mathrm{t}}}{\langle \mathbf{u}_{\mathrm{slip}} \rangle} \qquad (5.19)$$

The drag model of Beetstra et al. (2007) implemented in CFDEM® has been extended to account for this correction factor. Its value was 0.13 (being the lower limit, i.e., the largest correction) motivated by the values for the dimensionless slip velocity reported in Table 4.18.

5.4 2D Model of the Fluidized Bed

A 2D model is calculated faster than a 3D one. The effect of a number of parameters can be assessed before applying the simulation to a 3D model. The approach for assessing the effect of a number of simulation parameters was as follows: First, single-phase steady-state gas flow (without a turbulence model) was computed. Second, a turbulence model was considered, and the simulation continued until the turbulent energy approached its pseudo-steady-state value. Third, particles were being injected until the particle hold up in the domain has approached its pseudo-steady-state value. Calculations for particle injection were described in the Section 5.2 above.

5.4.1 Geometry

The origin of the coordinate system is located at the center of the nozzle outlet as shown in Figure 5.3. Within this thesis the 2D plane intersects the cylindrical section of the 3D model longitudinally (i.e., from the center along its height). The industrial partner recommended keeping velocities constant and applying only one nozzle (Holzinger, 2014a). Hence, the cross-sectional areas have been adapted. The depth has been chosen to be the width of the chute, and the dimensions in flow direction are identical to that of the 3D case. Comparing the height-to-diameter ratio of the riser with the one in the tutorial example “pitzDaily” of OpenFOAM® the flow is expected to be fully developed in the main flow direction. In order to keep the large turbulent vortices within the riser, the outlet has been modified to span only half of the height of the vertical cross-section in the top section. Since the cross-sectional area, and hence the gas mass flow rate, has been adopted in the 2D configuration, the particle mass rate has been reduced proportionally. Thus, the cross-sectional area has been modified (summarized in Table 5.6 and Table 5.7) following Eqn. (5.20). Note, that Eqn. (5.20) applies to all flow rates using the cross-sectional area in the cleaning zone as the reference.

$$\dot{M}_{2\mathrm{D}} = \dot{M}_{3\mathrm{D}} \frac{A_{2\mathrm{D,ref}}}{A_{3\mathrm{D,ref}}} = \dot{M}_{3\mathrm{D}} \frac{Y_{2\mathrm{D}}}{D_{\mathrm{CZ,3D}} \pi / 4} \qquad (5.20)$$

Table 5.6 Adapted bottom and middle region dimensions of the 2D riser model.

Parameter	3D	2D
2D depth Y_{2D}		0.532 m
Particle rate per chute $\dot{M}_{P,inject}$	77.8 kg/s	5.77 kg/s
	Cleaning zone (CZ)	
Width X	9.14 m	9.14 m
Cross-sectional area A	65.7 m2	4.87 m2
Average velocity U_Z	6.76 m/s	
	All nozzles	
Width X	1.27 m	1.23 m
Cross-sectional area A	8.87 m^2	0.658 m^2
Average velocity U_Z	50 m/s	
	Before nozzles (BN)	
Width X	5.56 m	3.38 m
Cross-sectional area A	24.3 m^2	1.80 m^2
Average velocity U_Z	18.3 m/s	

Table 5.7 Adapted top region dimensions of the 2D riser model.

Parameter	3D	2D
	Top region (horizontal)	
Width X	8.13 m	9.21 m
Cross-sectional area A	66.1 m^2	4.9 m^2
Average velocity U_Z	6.71 m/s	
	Top region (vertical)	
Height Z	8.59 m	9.73 m
Cross-sectional area A	69.8 m^2	5.18 m^2
Average velocity U_X	-6.35 m/s	
	Outlet	
Height Z	4.29 m	4.86 m
Cross-sectional area A	34.9 m^2	2.59 m^2
Average velocity U_X	-12.7 m/s	

5.4.2 Mesh

The bounding geometry was created via CAD software whereas the base mesh was created by utilizing tools of OpenFOAM®: A raw mesh created by blockMesh was subsequently modified in snappyHexMesh. Using snappyHexMesh, two surface layers were added at the side walls. Extruding the mesh turned out to be a safe way for a good mesh quality. In order to get a boundary that perfectly attaches to the snapped mesh, the command "foamToSurface –constant boundary.stl" was used. This was followed by manually deleting unused patches to arrive at an STL file that contains the side walls where particles bounce off in LIGGGHTS®. The resulting mesh is shown in Figure 5.3. The grid length in z-direction was 0.155 m in the cleaning zone, similar to a previous 3D mesh consisting of ca. 1 million grid cells. The grid length in x-direction was shortened to about 0.1 m in case the edges between nozzles and shell of an intersected 3D geometry should be resolved. This has led to $2.62 \cdot 10^4$ cells within 367 m^2, i.e., an average grid length of 0.118 m.

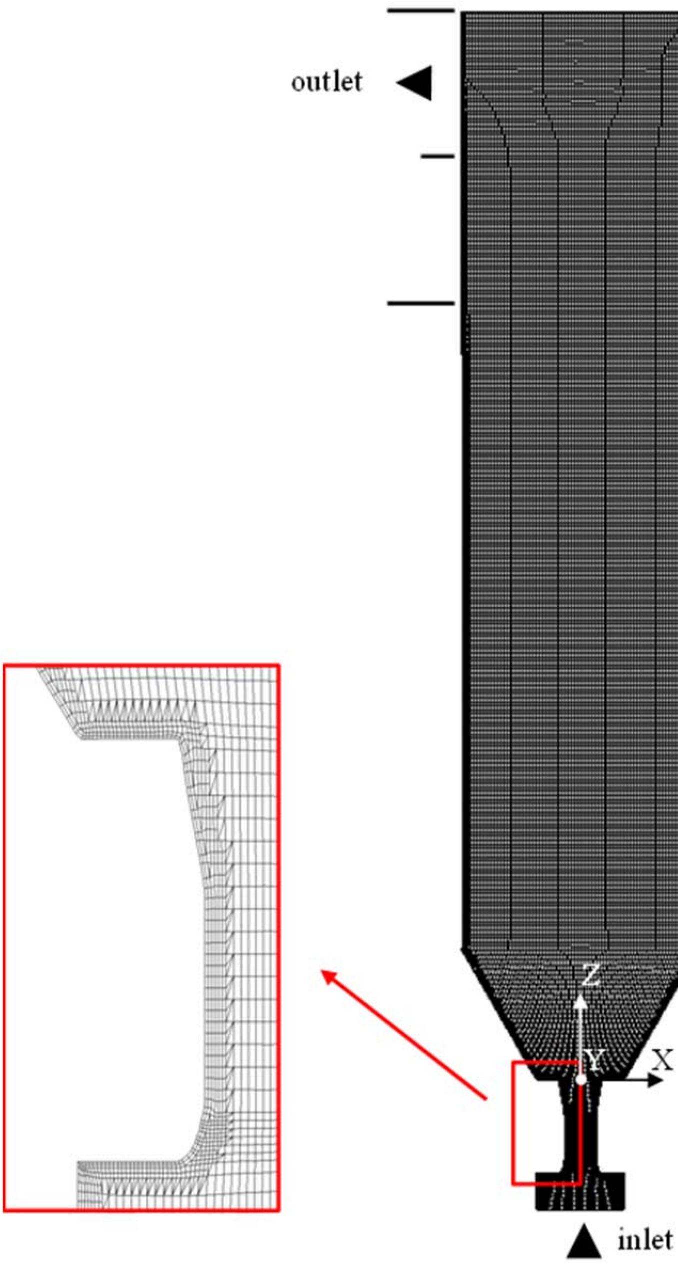

Figure 5.3 2D mesh.

5.4.3 Steady-State Gas Flow (No Turbulence Model)

In this simulation was assumed, that a steady-state flow can be achieved, and no turbulence model has been used. The setup has been based on a combination of CFDEM® tutorial examples "cfdemSolverPimpleImEx/crossFlowSalzman3D" and on "cfdemSolverPimpleImEx/sedimentationPeriodicBoxBiDisperse".

Since no particles were injected, only the injected gas at the recirculate inlet was considered here. The injected gas has been modeled using a "pressureGradientExplicitSource" in the recirculate inlet region. This region was modeled with a rectangular box of cells. The vertical location of the chute should be chosen above the cone-shaped region (Holzinger, 2014b). These settings are summarized in Table 5.8.

Table 5.8 Injection settings of (injection variants in) 2D riser simulations.

Parameter	Unit	Value		
Gas velocity below nozzles U_{BN}	[m³/s]	18.3		
Recirculate inlet gas velocity U_{R}	[m/s]	1	2	3.18
Recirculate inlet gas velocity $\begin{pmatrix} U_{\mathrm{X}} \\ U_{\mathrm{Y}} \\ U_{\mathrm{Z}} \end{pmatrix}_{\mathrm{R}}$	[m/s]	0.993 0 -0.122	1.99 0 -0.244	3.16 0 -0.388
(left front bottom) corner of inlet box	[m]	(-4.57 -0.266 6.5)		
(right back top) corner of inlet box	[m]	(-4 0.266 7.25)		

The number of iterations has been increased successively until steady-state conditions have been satisfied. Numerical settings have been based on the sedimentation cases (see Section 4.1), and are summarized in Table 5.9. Discretization schemes, solver settings and other dictionaries are available in the appendix, Section 9.2.3.2.

By monitoring extremes of pressure and gas velocity the reliability of the case was assessed. In order to determine whether a steady state has been achieved or not, several probes have been used to monitor local values of the pressure and gas velocity. Time-averaged (mean) quantities, e.g. gas velocity, have been collected. Sampling along the central axis, as well as along the radius at various heights has been performed to extract gas velocity profiles.

Table 5.9 Numerical parameters for 2D steady-state gas flow.

Parameter	Value
Turbulence model	laminar
Solver	simpleFoam
Time derivative scheme	steadyState
Divergence schemes	upwind

5.4.4 Turbulent Gas Flow

These simulations were started from the steady-state solution reported in Section 6.1.1. In the simulation setup now turbulent stresses and running in a transient mode were considered. Thus, a LES or unsteady-RANS (i.e., URANS) approach has been adopted. These cases were based on a base case in which turbulent flow was developed. Depending on the turbulence model used, a vortical flow structure was expected in the riser. The simulations were expected to last longer than in the previous section, i.e. five to seven times the residence time estimated in Eqn. (5.21).

$$\tau = V \,/\, \dot{V} \qquad \textbf{(5.21)}$$

Wall models for turbulent stresses have had to be defined as well. This was done by using wall functions at the fluid side with boundary conditions either from the results of the previous case or equal to the initial conditions. Initial conditions of turbulent quantities have been estimated according to Section 3.2.1 (in the cleaning zone) and were used at the inlet as boundary conditions. Settings of the base case were based on the simulations in the previous section and were adjusted during the simulation run in order to stabilize the solution. The settings for the simulations are summarized in Table 5.10. Detailed settings are available in the appendix, Section 9.2.3.3.

Table 5.10 Numerical parameters for 2D turbulent gas flow.

Parameter	Value
Basis for initial conditions	Steady state of previous case
Time step	adjustable
Courant number Co	< 0.8 base case, < 0.5 otherwise
Estimated residence time τ_{est}	5.93 s
Simulation time t_{sim}	35 s
Turbulence model	realizableKE oneEqEddy Smagorinsky
(specific) turbulent energy k_0, k_{FG}, sub grid scale and kinematic	0.171 m^2/s^2 (kqRWallFunction)
Dissipation rate ε_0, ε_{FG}	$6.38 \cdot 10^{-3}$ m^2/s^3 (epsilonWallFunction)
sub grid scale kinematic viscosity $\nu_{SGS,0}$, $\nu_{SGS,FG}$	$3.44 \cdot 10^{-3}$ m^2/s (nutUSpaldingWallFunction)
Solver	pimpleFoam
Time derivative scheme	Euler
Divergence schemes	limitedLinear

Extremes monitoring and post-processing settings have been extended in order to assess whether turbulence quantities have arrived at pseudo-steady-state conditions.

5.4.5 Quenching

In this simulation setup the injection of quenching water was considered. The simulation was started from the pseudo-steady-state solution reported in Section 6.1.2 which predicts the turbulent gas flow appropriately. The industrial partner informed that the inlet is located at the same height as the chutes, and centered between them. Quench water enters perpendicular to the center line of the riser as can be seen in Figure 5.2 (Holzinger, 2014b, 2014c). The quenching region has been modeled to be cylindrical (Holzinger, 2014c). The length of this cylinder has been assumed to be ten percent of the wall distance at that height of the riser. According to correspondence with the industrial partner (Radl, 2015), the modeled diameter is equal to the length of the cylinder as illustrated in Figure 5.4.

In order to assess the effect of the water injection velocity on the gas flow, the quench water velocity was enforced in the quenching region. Since the gas velocity near the wall is much lower compared to the quench water velocity, the injection volume has been moved closer to the center to prevent the simulation from divergence. The exact position is summarized in Table 5.11 and illustrated in Figure 5.4.

Table 5.11 Geometry of the quenching region in the 2D riser simulation.

Parameter	Value
Horizontal position of the base area closest to the wall	-4 m
Vertical position of the cylinder's center line	6.9 m
Cylinder dimensions D x L	0.914 x 0.914 m
Volume of the quenching region V_{inject}	0.45 m^3

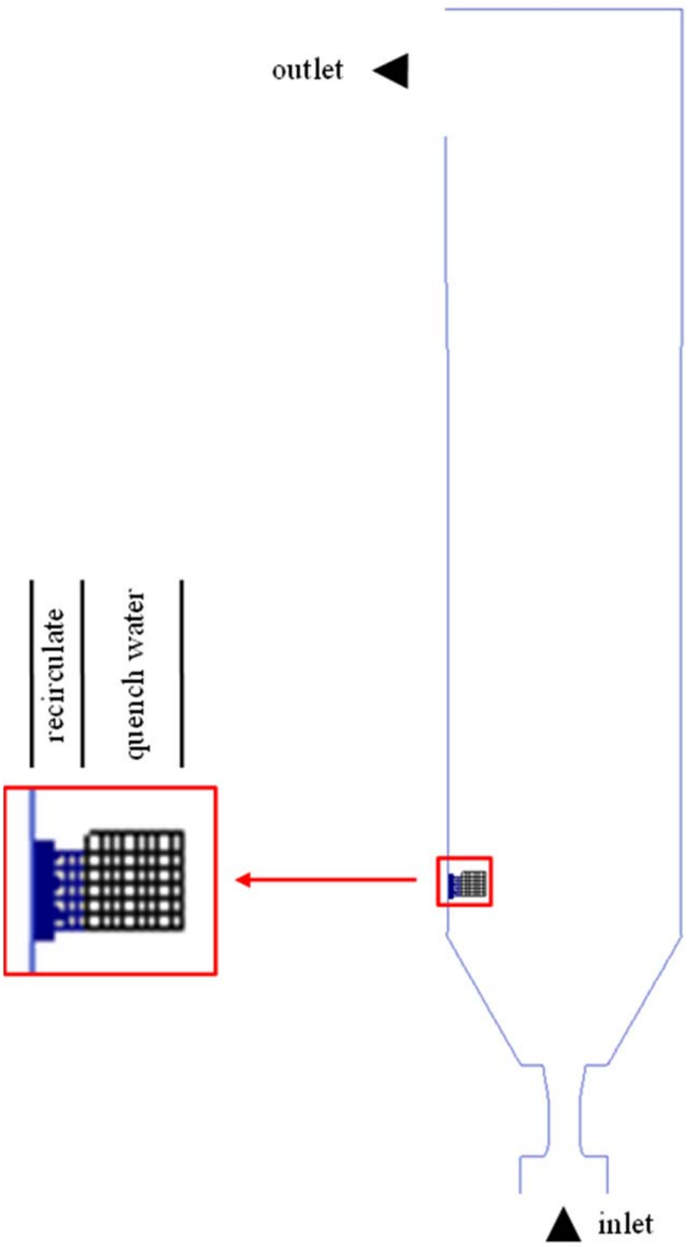

Figure 5.4 2D injection regions.

Next, all relevant quench water (and vapor) properties were collected. The molar vapor diffusion coefficient can be calculated using (Nellis and Klein, 2008, p. 11):

$$D_{\text{vap}} = -2.78 \cdot 10^{-6} + 4.48 \cdot 10^{-8} T_{\text{CZ}} + 1.66 \cdot 10^{-10} T_{\text{CZ}}^2 \qquad (5.22)$$

where an average temperature in the cleaning zone has been assumed. Eqn. (5.22) is a polynomial fit to data of Bolz and Tuve (1976), and is thus more precise than using the Chapman-Enskog equation.

Droplet relaxation time and the droplet relaxation length have been calculated according to Eqn. (3.4) and are as follows:

$$\tau_{\text{d}} = \frac{d_{\text{d}}^2 \rho_{\text{W,l}}}{18 \mu_{\text{f}}} \qquad (5.23)$$

$$\lambda_{\text{d}} = \frac{d_{\text{d}}^2 \rho_{\text{W,l}} u_{\text{t,d}}}{18 \mu_{\text{f}}} \qquad (5.24)$$

Following the quenching model in Section 3.3, all relevant quantities have been calculated using material properties found in literature (Eichlseder, 2008 No. 15; Kelley and Moore, 1944; Khinast et al., 2009, p. 90; N.N. (DDBST), 2015; N.N. (VDI), 2006; Radl and Khinast, 2010, p. 2426) and the droplet Sauter mean diameter reported by the industrial partner (Gronald, 2014a). The results of the droplet calculations are summarized in Table 5.12 (note that key physical properties have been already reported in Table 4.2).

Table 5.12 Quenching water droplet parameters (of the 2D and the 3D riser simulation).

Parameter	Value
Estimated average temperature in the cleaning zone T_{CZ}	373 K (100 °C)
Liquid water density $\rho_{W,l}\|_{T_W}$	998 kg/m^3
Droplet Sauter mean diameter $d_{d,32}$	2.33·10^{-4} m
Droplet relaxation time τ_d	0.123 s
Droplet relaxation length λ_d	0.149 m
Vapor diffusion coefficient (water in air) D_{vap}	3.7·10^{-5} m^2/s
Droplet Sherwood number Sh_{min}	2
Droplet evaporation time scale t_{vap} (tEvap)	1.22·10^{-4} s

Supplementary parameters and results of calculations relevant for the quenching model are summarized in Table 5.13 (note that key physical properties have been already reported in Table 4.2). Particles mainly consist of semi-hydrated calcium sulfite (Gronald, 2014a), hence the heat capacity was estimated considering such a material. The estimate for the average temperature in the cleaning zone has been estimated based on the evaporation temperature of water; the estimate for the outlet temperature has been based on the overall enthalpy balance of the fluidized bed, i.e., Eqn. (3.45).

Table 5.13 Quenching model parameters of the 2D riser simulation.

Parameter	Value
Quenching water injection velocity	50 m/s
Water rate $\dot{M}_{\mathrm{W}}$ (absolute quenchMuLiq)	0.947 kg/s
Water mass loading $\langle \mu_{\mathrm{W}} \rangle$	$3.58 \cdot 10^{-2}$
Estimated temperature at the outlet T_{CG}	347 K (74 °C)
Mean gas heat capacity $\overline{c_{p,\mathrm{f}}}\big\|_{T_{\mathrm{CZ}}}^{T_{\mathrm{FG}}}$	1,040 J/kgK
Liquid water heat capacity $c_{p,\mathrm{W,l}}$	4,190 J/kgK
Water vapor heat capacity $c_{p,\mathrm{W,vap}}$	1,850 J/kgK
Mean particle heat capacity $\left(\overline{c_{p,\mathrm{P}}}\big\|_{T_{\mathrm{P}}}^{T_{\mathrm{CZ}}}\right)_{\mathrm{est}}$	999 J/kgK
Heat of evaporation $\Delta h_{\mathrm{W,vap}}\big\|_{T_0}$ (deltaHEvap)	$2.5 \cdot 10^{6}$ J/kg
Water vapor density $\rho_{\mathrm{W,vap}}\big\|_{T_{\mathrm{CZ}}}$ (saturated)	0.598 kg/m^3

The setup has been based on the cases for turbulent flow, except for the divergence scheme. It has been set to “upwind” in order to ensure a stable simulation at the expense of a lower precision. Discretization schemes, solver settings and other dictionaries are available in the appendix, Section 9.2.3.4. The simulation was run until a pseudo-steady-state temperature profile was obtained.

Monitoring and post-processing settings have been extended for assessing the temperature profile. At the outlet quantities have been flux-averaged, i.e., considering the velocity component perpendicular to the outlet surface in the weighing process. In case of backflow flux-averaging is not applicable. Hence ranges of values were gathered from cell data at the outlet instead. Flux-averaging, e.g., of the time-averaged outlet temperature was done using:

$$T_{\mathrm{m,CG,flux\text{-}avg.}} = \frac{\sum_{i \in \mathrm{CG}} \phi_i T_{\mathrm{m},i}}{\sum_{i \in \mathrm{CG}} \phi_i}, \tag{5.25}$$

where i represents a cell adjacent to the outlet surface and ϕ_i the flux orthogonal to that surface directing outwards.

5.4.6 Particle Injection

In this simulation setup the injection of particles into the fully developed turbulent gas flow was considered. The simulation was started from that pseudo-steady-state solution reported in Section 6.1.2 which predicts the turbulent gas flow appropriately. The smoothing length has been calculated according to the law proposed in Eqn. (4.9). Other settings have been adapted from the CFDEM® tutorial “cfdemSolverPimpleImEx/crossFlowSalzman3D”.

The gas velocity at the particle position will be underestimated near walls when using a (linear) interpolation of the gas velocity. This is because the velocity profile in the (turbulent) boundary layer near walls cannot be resolved in full-scale simulations. Hence, the slip boundary condition for the gas has been applied at walls. This allows a (linear) interpolation of gas flow quantities at the particle position, and removes unphysically low estimates of gas flow quantities near walls. Furthermore, particles have been kept at a certain distance from the walls by reflecting particles at a certain distance from the walls. In addition, the (Hertzian) elastic soft sphere model was active, in order to model enduring particle-wall contacts and to prevent particles from penetrating the walls.

In order to improve the stability of the simulation, the CFD time step has been decreased, the SGS model has been changed from oneEqEddy to Smagorinsky, and the discretization scheme has been changed from limited linear to upwind. In order to reduce computation time, the coupling algorithm described in Section 3.4 has been used.

Table 5.14 Numerical parameters for 2D particulate flow.

Parameter	Value
Particle rate $\dot{M}_{\mathrm{P,inject}}$	5.77 kg/s
Basis for initial conditions	see Section 6.1.2 predicting appropriate turbulent gas flow for pseudo-steady-state solution
DEM time step Δt_{DEM}	$2 \cdot 10^{-5}$ s(7)
CFD time step Δt_{CFD}	10^{-4} s
Courant number Co	< 0.1
Coupling factor CF	5
Expected particle volume fraction $\langle \varphi_{\mathrm{P}} \rangle_{\mathrm{exp}}$	$1.11 \cdot 10^{-3}$
Coarse graining ratio α	1,020
Smoothing length l_{smooth}	$6.69 \cdot 10^{-2}$ m
Drag model	Beetstra
Drag correction factor f_{corr}	0.130
Turbulence model	Smagorinsky
Solver	cfdemSolverPimpleImEx
Time derivative scheme	Euler
Divergence schemes	upwind
Wall-collision particle scale-up factor	7.50
Young’s modulus	$2 \cdot 10^{6}$ N/m²

7 Small amount of particles allowed to be lost

The setup has been based on the previous setup. The simulation was run until the particle hold up has arrived at a pseudo-steady-state value. The 2D mass rate calculated in the geometry section has been used (reported in Table 5.6 and Table 5.14). Also, the nominal number of parcels was set to $2 \cdot 10^6$ by adjusting the coarse graining ratio. The settings for the simulation are summarized in Table 5.14. Detailed settings are available in the appendix, Section 9.2.3.5. An explanation for the drag correction factor, used for the subsequent filtered (modified) case, is given in Section 5.3.

Extremes monitoring and post-processing settings have been extended for assessing particle (mass) rate, particle volume fraction, particle size distribution, and sub grid scale viscosity. Subsequently, a modified simulation was performed, which uses drag correction based on the drag correction factor.

5.5 3D Model of the Fluidized Bed

Simulations involving a 3D model of the riser were performed similar to the 2D cases. First, a single-phase steady-state gas flow (without turbulence model) was performed. Second, turbulence was added until the turbulent energy remained constant, and third the particles have been injected until particles have left the geometry.

5.5.1 Geometry

The geometry "Modell_Kentucky_rev2.sat" of the riser has been obtained from the industrial partner as standard ACIS text file. A front view is shown in Figure 5.1. Inlet and outlet are rectangular, while the vertical section is cylindrical. The coordinate origin is at the center of the riser and located in the outlet plane of the nozzles. A detailed dimensional drawing is available in the appendix, Section 9.1.

5.5.2 Mesh

The mesh "2014_Kentucky_TUG_G00.msh" has been created by the industrial partner. This was done after several unsuccessful attempts to prepare a mesh with the utility "snappyHexMesh" of OpenFOAM® and the "Cubit" toolkit. By using the command "importMesh", the mesh has been imported into the case and the bounding wall "wall.stl" has been extracted. By running "AllrunPar" on the local computer, bounding patches have been merged prior to the simulation. In addition to hexahedrons, the final mesh consisted of tetrahedrons and pyramids in regions where a rectangular and cylindrical geometry merge (i.e., below the nozzles, and after the main cylindrical region of the riser). The mesh consisted of $1.11 \cdot 10^6$ cells with a volume of 3,870 m^3, i.e., an average grid size of 0.152 m. A longitudinal cross-section of the riser's computational mesh is shown in Figure 5.5.

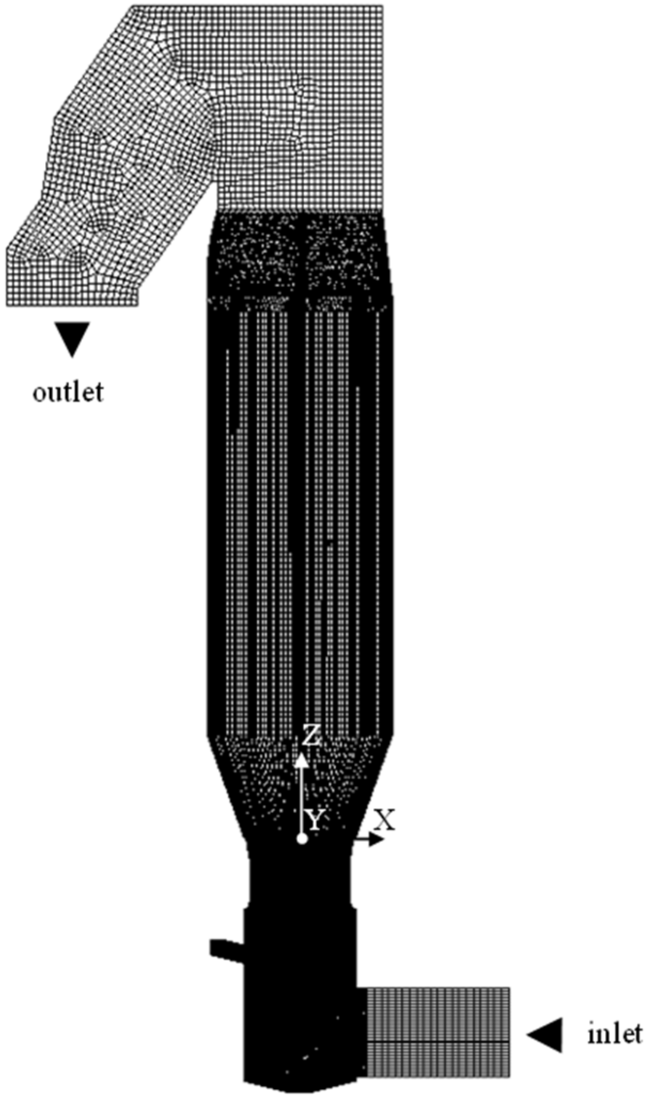

Figure 5.5 3D mesh of the riser.

5.5.3 Steady-State Gas Flow (No Turbulence Model)

In this simulation a steady-state flow and no turbulent stresses were assumed. The setup has been based on the 2D steady-state gas flow setup. The gas entering through the particle inlet was considered in the simulation.

Due to the small tetrahedral mesh elements above the cleaning zone, the divergence scheme had to be changed to the more stable “upwind” scheme. Numerical settings are summarized in Table 5.15. Discretization schemes, solver settings and other dictionaries are available in the appendix, Section 9.2.4.1.

Table 5.15 Numerical parameters for 3D steady-state gas flow.

Parameter	Value
Turbulence model	laminar
Solver	simpleFoam
Time derivation scheme	steadyState
Divergence schemes	upwind

Extremes monitoring and post-processing settings followed the settings for the 2D steady-state gas flow.

5.5.4 Turbulent Gas Flow

This simulation followed the settings for the 2D turbulent gas flow considering only the one-equation model. The simulation was started from the 3D steady-state solution reported in Section 6.2.1. Numerical settings are summarized in Table 5.16. Solver settings and other dictionaries are available in the appendix, Section 9.2.4.2.

Table 5.16 Numerical parameters for 3D turbulent gas flow.

Parameter	Value
Basis for initial conditions	Steady state of previous case
Time step	adjustable
Courant number Co	< 0.8 base case, < 0.5 otherwise
Estimated residence time τ_{est}	8.72 s
Simulation time t_{sim}	50 s
Turbulence model	oneEqEddy
(specific) turbulent energy k_0, k_{FG}, sub grid scale and kinematic	0.171 m^2/s^2 (kqRWallFunction)
Dissipation rate ε_0, $\varepsilon_{\mathrm{FG}}$	$6.38 \cdot 10^{-3}$ m^2/s^3 (epsilonWallFunction)
sub grid scale kinematic viscosity $\nu_{\mathrm{SGS,0}}$, $\nu_{\mathrm{SGS,\,FG}}$	$3.44 \cdot 10^{-3}$ m^2/s (nutUSpaldingWallFunction)
Solver	pimpleFoam
Time derivative scheme	Euler
Divergence schemes	upwind

Extremes monitoring and post-processing settings followed the settings for the 2D turbulent gas flow.

5.5.5 Quenching

This simulation followed the settings for the 2D quenching simulation. The simulation was started from the 3D turbulent flow solution reported in Section 6.2.2. The shape of the water droplet injection region was again considered to be cylindrical, with the cylinder diameter equal to the length of the cylinder. The setup basically merges the 2D quenching setup and the previous 3D turbulent flow simulation setup. Geometry and settings for the quenching model different to the 2D quenching case are summarized in Table 5.17.

Table 5.17 Parameters of quenching in the 3D riser simulation.

Parameter	Value
Water rate $\dot{M}_{\mathrm{W}}$ (absolute quenchMuLiq)	12.8 kg/s
Horizontal position	-3.9 m
Vertical position	3.83 m
Diameter	0.834 m
Length	0.834 m
Volume of quenching region V_{inject}	0.426 m^3

Extremes monitoring and post-processing settings followed the settings for the 2D quenching simulation.

5.5.6 Particle Injection

These simulations followed the settings for the 2D particulate flow simulations. The simulations were started from the 3D turbulent flow solution reported in Section 6.2.2. The particulate flow was considered to be relevant only above the nozzles. Hence, the domain has been truncated at the particle side of the simulation. In addition, parcels leaving the domain at the outlet and down below the nozzles have been recorded. The setup basically merges the 2D particulate flow setup and the previous 3D quenching simulation setup. Numerical settings different from the latter are summarized in Table 5.18. Detailed settings are available in the appendix, Section 9.2.4.4.

Table 5.18 Numerical parameters for 3D particulate flow.

Parameter	Value
Particle rate per injection chute $\dot{M}_{\mathrm{P,inject}}$	77.8 kg/s
Basis for initial conditions	Pseudo-steady state of turbulent case
DEM time step Δt_{DEM}	$6 \cdot 10^{-5}$ s
CFD time step Δt_{CFD}	$1.2 \cdot 10^{-4}$ s
Courant number Co	< 0.1
Coupling factor CF	2
Expected particle volume fraction $\langle \varphi_{\mathrm{P}} \rangle_{\mathrm{exp}}$	$2.22 \cdot 10^{-3}$
Coarse graining ratio α	3,460
Smoothing length l_{smooth}	0.18 m
Drag model	Beetstra
Drag correction factor f_{corr}	0.130
Turbulence model	Smagorinsky
Solver	cfdemSolverPimpleImEx
Time derivation scheme	Euler
Divergence schemes	upwind
Wall-collision particle scale-up factor	5.00
Young's modulus E	$2 \cdot 10^{6}$ N/m²

Extremes monitoring and post-processing settings has been based on the 2D particulate gas flow. In a correct setup, time-averaged data of water mass loadings should be available as soon as pseudo-steady-state conditions were satisfied in order to display their profiles. Unfortunately, that data is not available for such an early time step. In order to reduce the amount of data, field data of certain centered cross-sections have been recorded during the simulation instead of the full 3D field data.

6 Results for the Full-Scale Fluidized Bed

6.1 2D Model of the Fluidized Bed

In the following simulations the pseudo-steady-state flow needs to be determined. This is done first by probing the quantities of interest at constant locations. Second, the deviations of their values from one time step to the next are computed. Computation continues using a constant interval of five seconds until the quantities of interest share a deviation lower than 5 % at all probes. Then their corresponding fields are considered to satisfy the pseudo-steady-state conditions. In case of turbulence, or the formation of meso-scale structures, the probed velocity (or voidfraction), is allowed to deviate 25 % from their pseudo-steady-state conditions within the same interval.

In the following figures the bounding wall is represented by a bold blue line, and the injection regions are indicated by grey and black lines for the recirculate and the quench zone, respectively. This is in accordance with the regions shown in Figure 5.4. For comparison reasons the scale for velocities is kept constant rather than representing true minima and maxima. The largest value on a logarithmic scale is the maximum value occurring in the time step under consideration.

6.1.1 Steady-State Gas Flow (No Turbulence Model)

Figure 6.1 illustrates the single-phase gas flow after 3,000 iterations at steady state. This flow builds the basis for subsequent simulations.

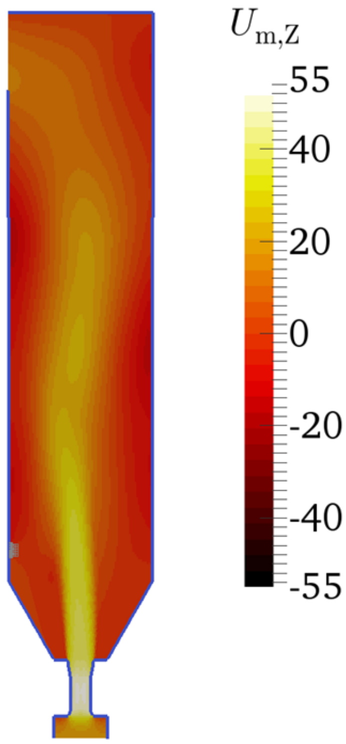

Figure 6.1 Time-averaged gas velocity in the 2D riser model (3,000 iterations, no turbulence model).

6.1.2 Turbulent Gas Flow

Figure 6.2 illustrates the turbulent single-phase gas flow at pseudo-steady state using the One-Equation-Eddy model. The approach to pseudo-steady state was monitored by considering the transients of the time-averaged gas velocity and the time-averaged turbulent energy. Pseudo-steady-state conditions were satisfied after 35 s.

As the turbulent energy increases, the effective viscosity increases too, which leads to a comparably smooth velocity field. From the gas velocity component in the z-direction (shown in Figure 6.2) one can ascertain two vortices. The instantaneous sub grid scale viscosity (shown in Figure 6.3) ranges from 10^{-3} to 0.139 m^2/s, which is many orders of magnitude larger than the molecular viscosity of the fluid, $3.04 \cdot 10^{-5}$ m^2/s.

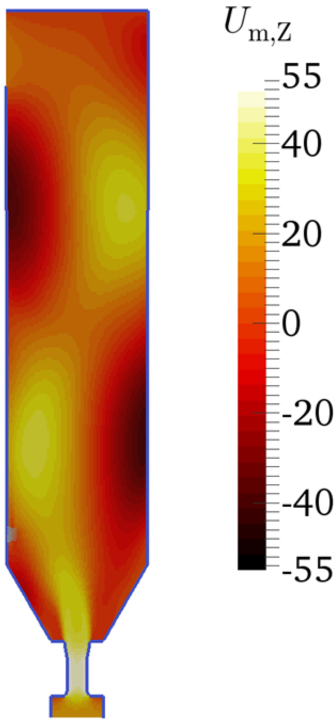

Figure 6.2 Time-averaged gas velocity in the turbulent (One-Equation-Eddy) 2D riser model (*t* = 35 s).

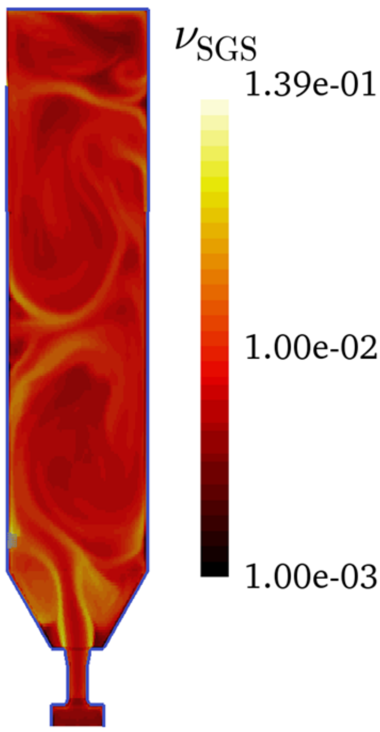

Figure 6.3 Instantenous sub grid scale viscosity in the turbulent (One-Equation-Eddy) 2D riser model (*t* = 35 s).

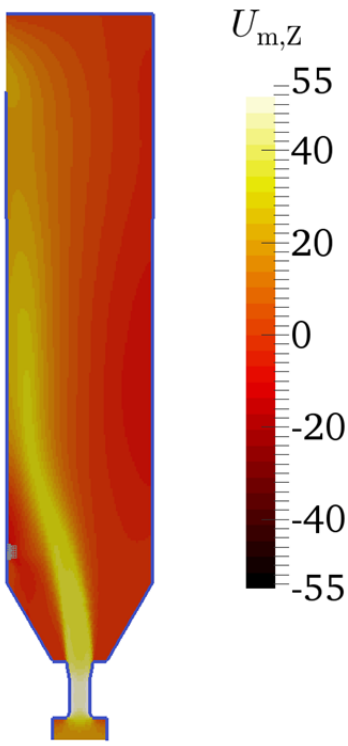

Figure 6.4 Time-averaged gas velocity in the turbulent (realizable k-ε) 2D riser model (*t* = 35 s).

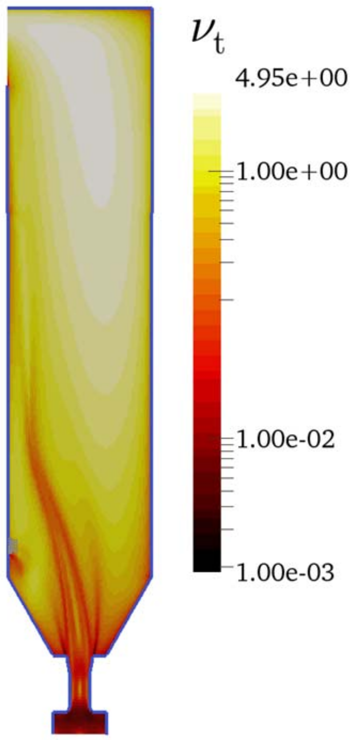

Figure 6.5 Instantenous turbulent viscosity in the turbulent (realizable k-ε) 2D riser model (*t* = 35 s).

Figure 6.4 illustrates the turbulent single-phase gas flow at pseudo-steady state using the realizable-k-ε model. Pseudo-steady-state conditions were satisfied after 35 s (note that this case started from the final state of the LES case). The flow field is free of vortices and smooth due to the relatively high turbulent viscosity. The turbulent viscosity (shown in Figure 6.5) ranges from 10^{-3} to 4.95 m^2/s, which results in a larger effective viscosity than in the LES case. As such high viscosity appears to be wrong in regions of low gas velocity, the simulation was further analyzed. First structures with high turbulent viscosity formed during the transient state using the initial state (i.e., the state from the previous LES). As prolonged simulation revealed, the solution remains almost the same. Hence the source of high turbulent viscosity is still unclear. Perhaps the outlet acts as a source of turbulence as prevention measures (boundary condition at the outlet) were not successful in repeated simulations.

Disagreement between URANS and LES was also reported in literature: Ammour (2013, pp. 198, 202, 216) compared two URANS models against LES and observed good agreement only in case a limiter is applied to the URANS model. Otherwise the URANS model was observed to overpredict the turbulent viscosity in regions of high strain rate. She recommended using the realizable k-ε model as done in the current study. The implementation in OpenFOAM® does not limit the viscosity as in the work of Ammour. Hence, the LES approach was chosen for all subsequent simulations. One should note that LES results are known to depend on the grid size, and hence a fine mesh (as used here) is obligatory.

6.1.3 Quenching

Figure 6.6 illustrates the turbulent single-phase gas flow including quenching at pseudo-steady state using the One-Equation-Eddy model. The approach to pseudo-steady state was monitored by considering the transients of the time-averaged gas velocity, the time-averaged turbulent energy and the time-averaged temperature. Pseudo-steady-state conditions were satisfied after 60 s.

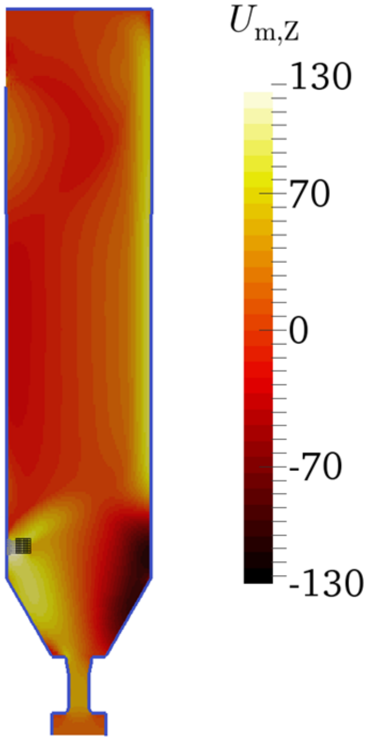

Figure 6.6 Time-averaged gas velocity in the 2D riser model (including quenching, *t* = 60 s). The water injection region is depicted with black lines.

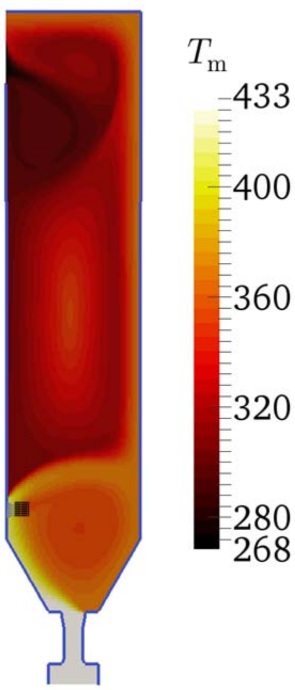

Figure 6.7 Time-averaged temperature profile in the 2D riser model (including quenching, *t* = 60 s). The water injection region is depicted with black lines.

The vortex at the bottom is observed to be significantly larger in the simulation with quenching. This is because the quench water inlet generates a low pressure region, which leads to a shift of the incoming gas jet towards this region. Also, the vertical flow structure in the upper section of the riser was not observed in the simulations with quenching. The gas velocity of the vortex is larger than the water injection velocity. This is due to the interaction of the incoming gas jet (from the nozzles) with the injected quench water droplets.

As (part of) the quenching water droplets evaporate the gas gets immediately cooled. Hence, the local vapor concentration and temperature strongly correlate. Flue gas below the quenching region was in contact with it repeatedly, heating up the quenching region and lowering the temperature in the region of the bottom vortex. The (velocity of the) vortex at the bottom indicates a well-mixed zone with almost homogeneous temperature distribution. Backflow at the outlet injects cool air leading to additional (unphysical) cooling of the flue gas. This indicates that a longer outflow region would be necessary to correctly picture the flow in this region. The low temperature of that gas mixture (flue gas plus air of backflow) alters the liquid-vapor equilibrium of quench water such that less water evaporates. These conclusions are based on the profiles of time-averaged temperature, time-averaged liquid water and time-averaged water vapor mass loadings, which are illustrated in Figure 6.7 and Figure 6.8.

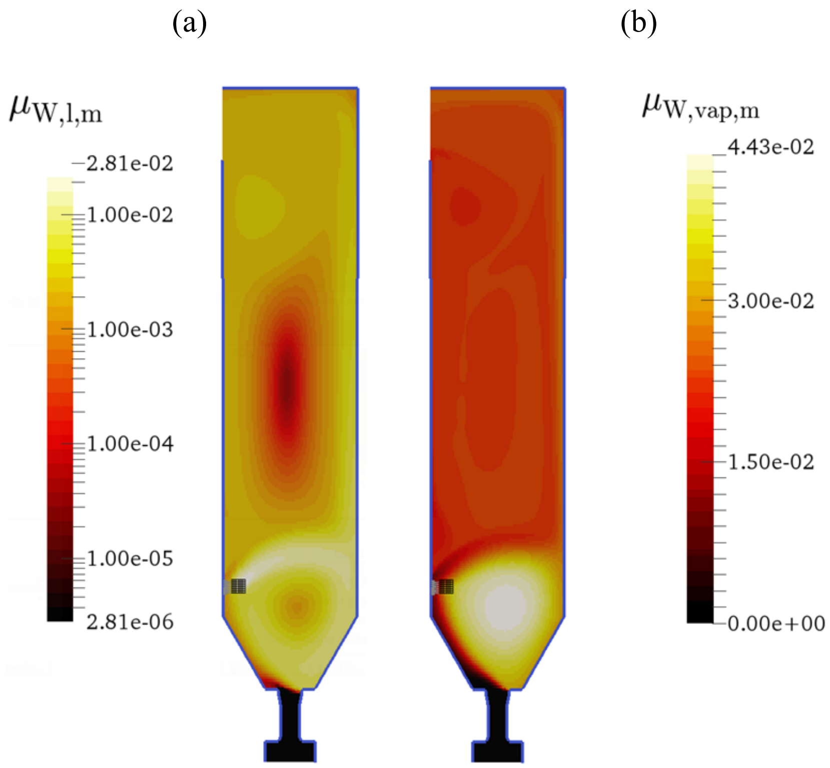

Figure 6.8 Time-averaged water mass loadings, i.e. (a) liquid and (b) vapor, in the 2D riser model (including quenching, *t* = 60 s). The water injection region is depicted with black lines.

Table 6.1 Time-averaged outlet quantities obtained from the 2D riser simulation (including quenching, *t* = 60 s).

Variable	Value
Area-averaged velocity $U_{CG,m}$	(-16.5 0 -1.81) m/s
Temperature $T_{CG,m}$	268 … 366 K
Liquid water mass loading $\mu_{CG,W,l,m}$	$1.24 \cdot 10^{-3}$ … $2.40 \cdot 10^{-3}$
Vapor mass loading $\mu_{CG,W,vap,m}$	$1.69 \cdot 10^{-2}$ … $2.57 \cdot 10^{-2}$

In addition to time- and area-averaged gas velocity at the outlet, ranges of certain quantities are summarized in Table 6.1 at pseudo-steady state. Ranges of values were gathered from cell data underlying the above figures, as flux-averaging is not applicable due to backflow. The insignificantly larger gas velocity indicates that added momentum due to quench water injection distributes partly to the main gas flow, partly to the vortex flow. The estimated temperature is within the range. The estimated water mass loading (i.e., the sum of liquid and vapor content) is reported in Table 5.13 as well, and is insignificantly larger than the upper range limit due to backflow. One should note that the estimate is based on flow rates whereas the reported range is based on cell data.

6.1.4 Particle Injection

Figure 6.9 illustrates the gas velocity field of the simulation considering particle injection at pseudo-steady state. The approach to pseudo-steady state was monitored by considering the transients of the time-averaged gas velocity, the time-averaged turbulent viscosity and the time-averaged void fraction. Pseudo-steady-state conditions were obtained after 30 s.

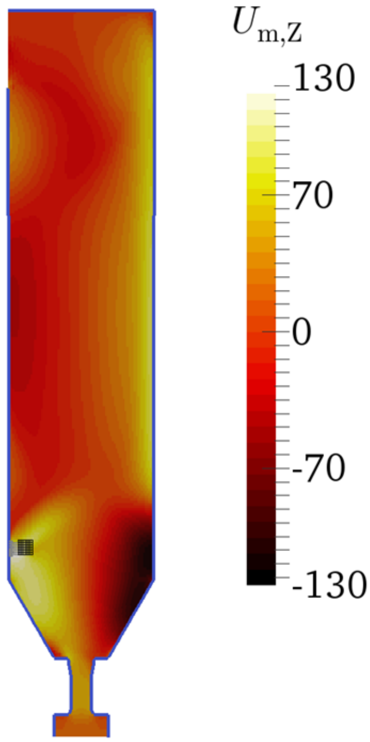

Figure 6.9 Time-averaged gas velocity in the 2D riser model (full model with particles, *t* = 30 s). Injection regions are depicted with lines, i.e. water (black) and recirculate (grey).

Clearly, particles do not affect the main flow significantly, since the general features of the gas flow remain almost unchanged (see also Figure 6.6). Note, that two particle classes comprising the largest (but most rare) particle class have not been injected due to their low number frequency. It can be expected, that these particle classes have a low effect on the flow features, such that the general conclusions drawn in the section are not affected.

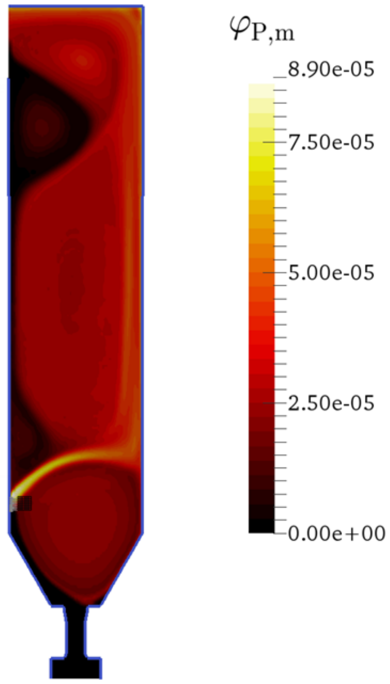

Figure 6.10 Time-averaged particle volume fraction in the 2D riser model (full model with particles, *t* = 30 s). Injection regions are depicted with lines, i.e. water (black) and recirculate (grey) .

At the beginning parcels were redirected upwards from their initial vertical direction forming a vortex located near the center (in vertical direction) of the cylindrical region of the riser. After collision with the opposite wall of the riser, parcels partially moved downwards and accumulated in the lower vortex. Gas backflow from the outlet leads to particle accumulation in a third (upper) vortex. Large particles follow the fast gas stream near the wall towards the outlet, since they cannot follow the gas downflow near the wall. The smaller the particles, the better they are blended with the incoming gas. These conclusions are based on recorded parcels positions and the time-averaged void fraction as illustrated in Figure 6.10.

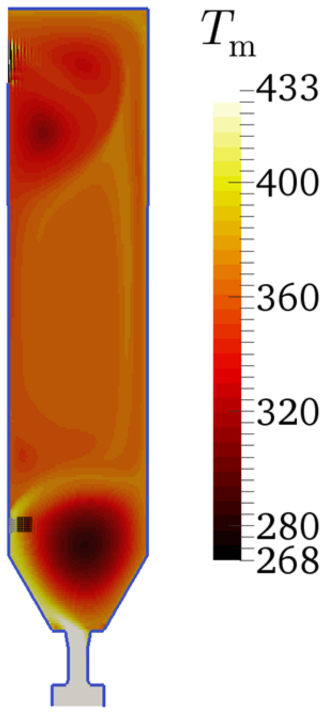

Figure 6.11 Time-averaged temperature profile in the 2D riser model (full model with particles, *t* = 30 s). Injection regions are depicted with lines, i.e. water (black) and recirculate (grey) .

The temperature profile and also the water mass loadings differ quantitatively from simulation results in the previous section (see Section 6.1.3) and are illustrated in Figure 6.11 and Figure 6.12. The mean temperature of backflow fluctuates within a broader range at the outlet. Hence, less cooling takes place in the region above the water injection point.

Near the inlet, the temperature in the center of the vortex is very low, causing water vapor to condense. The exact reason for this very low temperature is unclear, and might be caused by the combined use of the CFDEM® solver and the quenching model.

Furthermore, a simulation run for a total duration of 60 [s] resulted in no significant changes compared to the data reported in Figure 6.11.

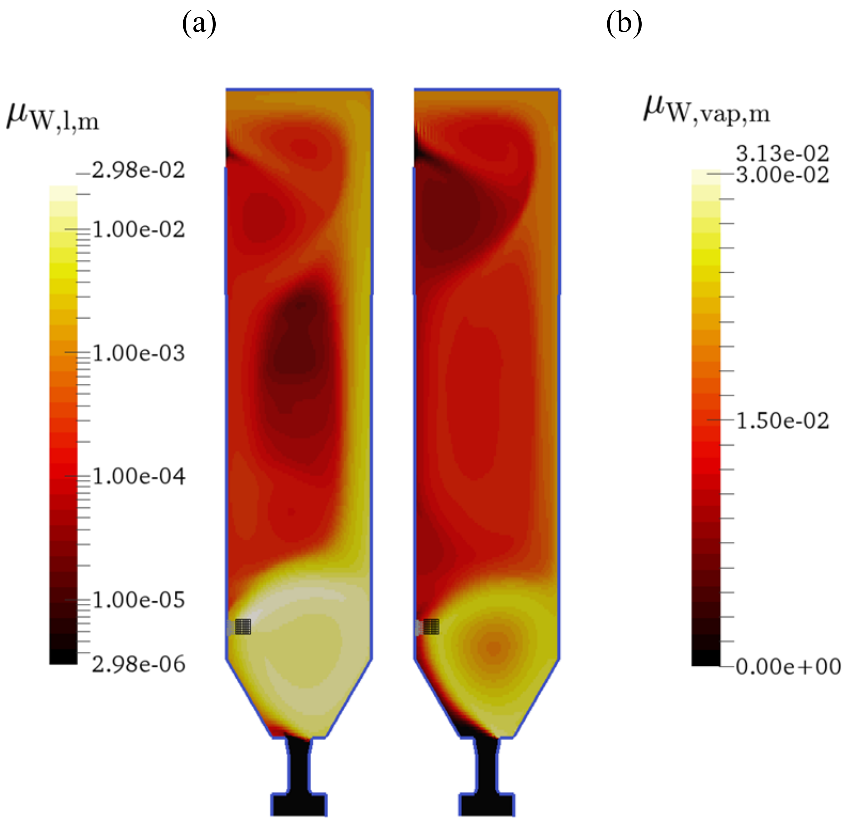

Figure 6.12 Time-averaged water mass loadings, i.e. (a) liquid and (b) vapor, in the 2D riser model (full model with particles, *t* = 30 s). Injection regions are depicted with lines, i.e. water (black) and recirculate (grey).

Table 6.2 Time-averaged outlet quantities obtained from the 2D riser simulation (full model with particles, *t* = 30 s).

Variable	Value
Area-averaged gas velocity $U_{CG,m}$	(8.47 0 -4.57) m/s
Area-averaged particle volume fraction $\varphi_{P,CG,m}$	$1.95 \cdot 10^{-5}$
Temperature $T_{CG,m}$	271 … 389 K
Liquid water mass loading $\mu_{W,l,CG,m}$	0 … $1.16 \cdot 10^{-3}$
Vapor mass loading $\mu_{W,vap,CG,m}$	0 … $2.03 \cdot 10^{-2}$

In addition to the time- and area-averaged velocity at the outlet, ranges of certain quantities are summarized in Table 6.2 at pseudo-steady state gathered in the same manner as in Section 6.1.3 due to backflow. An average gas velocity directing into the riser indicates that backflow at the outlet into the riser is important. The estimated temperature is within the range. The estimated water mass loading (reported in Table 5.13) is insignificantly larger than the upper range limit due to backflow now void of water.

Domain-averaged quantities at pseudo-steady state are summarized in Table 6.3.

Table 6.3 Time- and domain-averaged quantities obtained from the 2D riser simulation (full model with particles, *t* = 30 s).

Variable	Value
Particle volume fraction $\overline{\langle \varphi_P \rangle}$	$2.74 \cdot 10^{-5}$
Hold up $\overline{\langle \varepsilon_P \rangle}$	$6.17 \cdot 10^{-2}$ kg/m^3
Average particle insertion rate $\left.\overline{\dot{M}_P}\right\|_0^{2s}$	4.88 kg/s

The insertion rate was verified by gathering the total mass in the riser at certain time steps, and is 15.4 % lower than the desired input value of 5.77 kg/s (reported in Table 5.14). This is mainly due to particle loss as described in the next paragraph.

Hold-up is significantly lower than the estimate of 0.175 kg/m3 reported in Table 5.5 although both consider a single chute. This is because

(i) the estimate is based on flow rates,

(ii) the nozzle and the region below are assumed to be stagnant zones, i.e. without particles, increasing the hold-up approximately by 3 %,

(iii) fragments of parcels have not been inserted decreasing the mass rate approximately by 3 % after 2 seconds of insertion,

(iv) big particles leave the riser very quickly, i.e., they have a residence time smaller than the mean residence time of the gas,

(v) small particles have been lost in the simulation due to way too large DEM time step decreasing the hold-up approximately by 12 % after 2 seconds of insertion

Hence segregation effects leading to particle-size-dependent residence time and sparse regions are suggested to be the main reason for the low hold-up predicted by the simulation.

The dependency of residence time on parcel size can be examined qualitatively in Figure 6.13, which plots the parcel size distribution in the domain advancing over time. In conjunction with Figure 6.14 a quantitative picture can be provided.

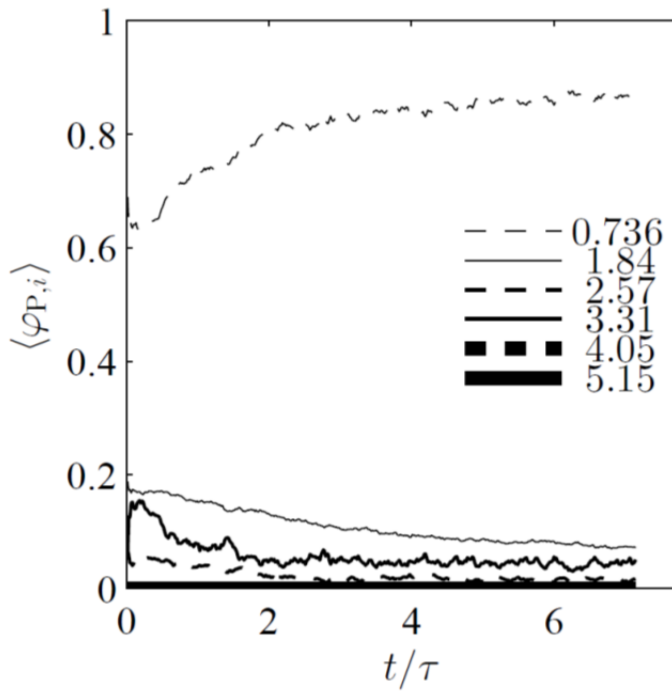

Figure 6.13 Domain-averaged parcel size distribution in the 2D riser simulation versus time. Domain-averaged particle volume fraction of individual classes normalized for a total over classes of 1. Class mean particle diameters displayed in the legend are normalized with the Sauter mean diameter $d_{32} = 6.80 \cdot 10^{-6}$ m. The mean gas residence time is $\tau = 5.63$ s.

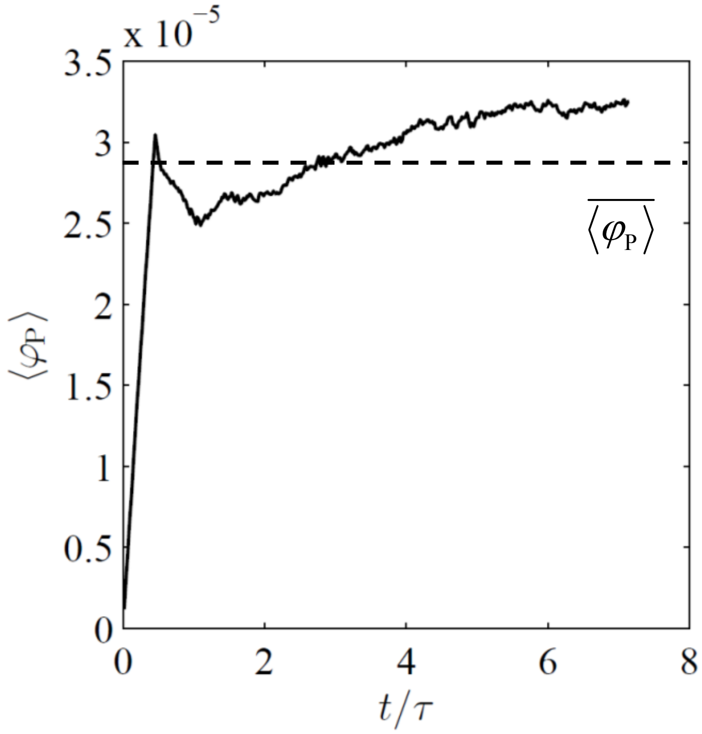

Figure 6.14 Domain-averaged particle volume fraction in the 2D riser simulation versus time. The mean gas residence time is τ = 5.63 s.

6.2 3D Model of the Fluidized Bed

The approach to pseudo-steady state is the same as for 2D simulations. Also, the design of the figures is based on the one for the 2D cases.

6.2.1 Steady-State Gas Flow (No Turbulence Model)

Figure 6.15 illustrates the single-phase gas flow after 9,500 iterations. Gas injected via the upper particle inlets does not influence the main flow. This flow constitutes the basis for subsequent simulations.

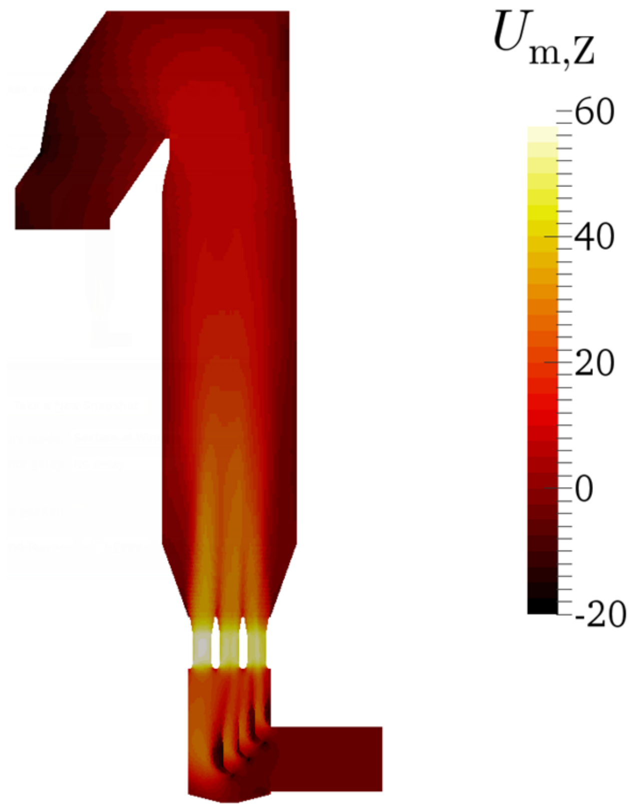

Figure 6.15 Time-averaged gas velocity in a longitudinal section of the 3D riser model (9,500 iterations, no turbulence model)

6.2.2 Turbulent Gas Flow

Figure 6.16 illustrates the turbulent single-phase gas flow after it satisfied the pseudo-steady-state conditions at $t = 65$ s using the One-Equation-Eddy model. The approach to pseudo-steady state was monitored by considering the transients of the time-averaged gas velocity and the time-averaged turbulent energy. The time-averaged gas velocity is quite similar to the one in Section 6.2.1.

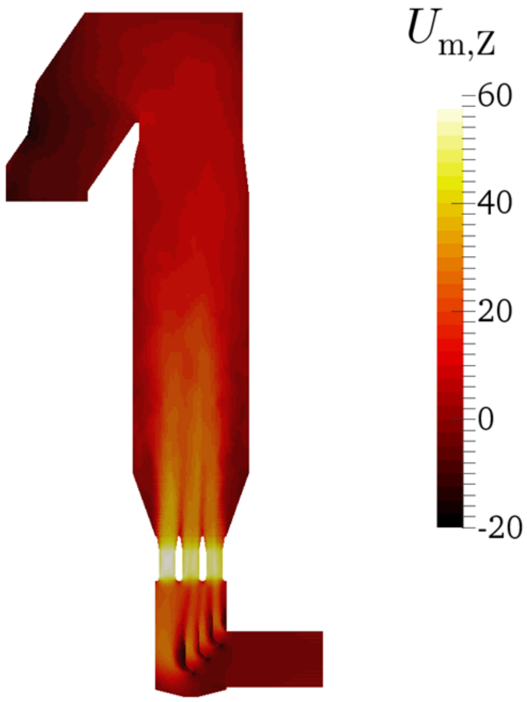

Figure 6.16 Time-averaged gas velocity in a longitudinal section of the turbulent (One-Equation-Eddy) 3D riser model (t = 65 s).

Figure 6.17 illustrates the instantaneous sub grid scale viscosity at $t = 65$ s. It ranges from 10^{-3} to 0.142 m^2/s, which is similar to the solution in the turbulent 2D riser model using the same turbulence model (shown in Figure 6.3).

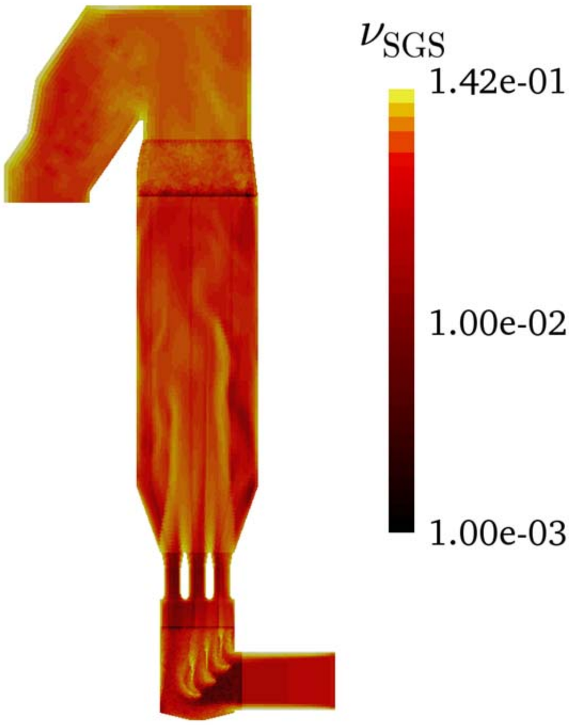

Figure 6.17 Instantaneous sub grid scale viscosity in a longitudinal section of the turbulent (One-Equation-Eddy) 3D riser model (*t* = 65 s).

6.2.3 Quenching

Figure 6.18 illustrates the turbulent single-phase gas flow including quenching after it satisfied the pseudo-steady-state conditions at *t* = 32 s. The approach to pseudo-steady state was monitored by considering the transients of the time-averaged gas velocity, the time-averaged turbulent energy and the time-averaged temperature. In contrast to the 2D case, the high-velocity gas jet caused by the injected water droplets does not significantly influence the flow below the injection region (i.e., below the water injection no vortex structure was formed. The reason is that the flue gas can bypass the high-velocity gas jet in the third dimension). Also, no backflow occurs into the flow domain due to the longer exit region.

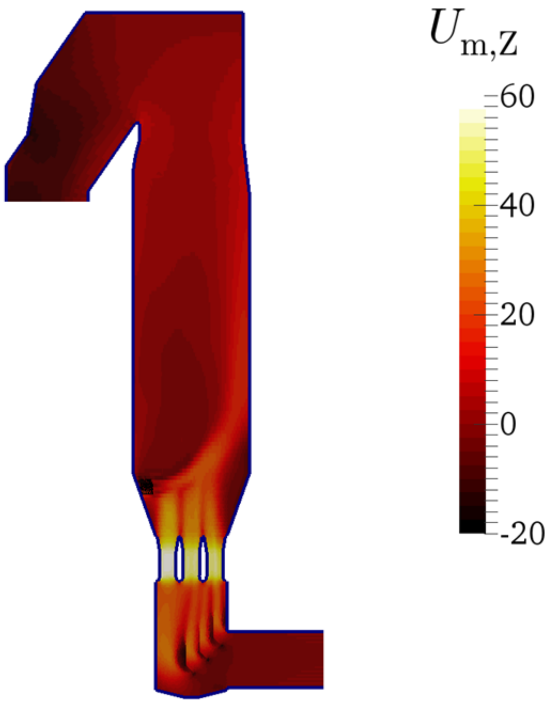

Figure 6.18 Time-averaged gas velocity in a longitudinal section of the 3D riser model (including quenching, *t* = 32 s). The water injection region is depicted with black lines.

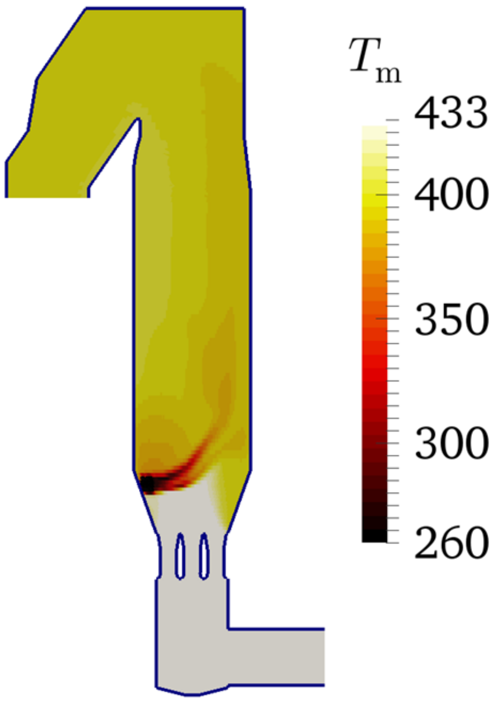

Figure 6.19 Time-averaged temperature profile in a longitudinal section of the 3D riser model (including quenching, *t* = 32 s). The water injection region is depicted with black lines.

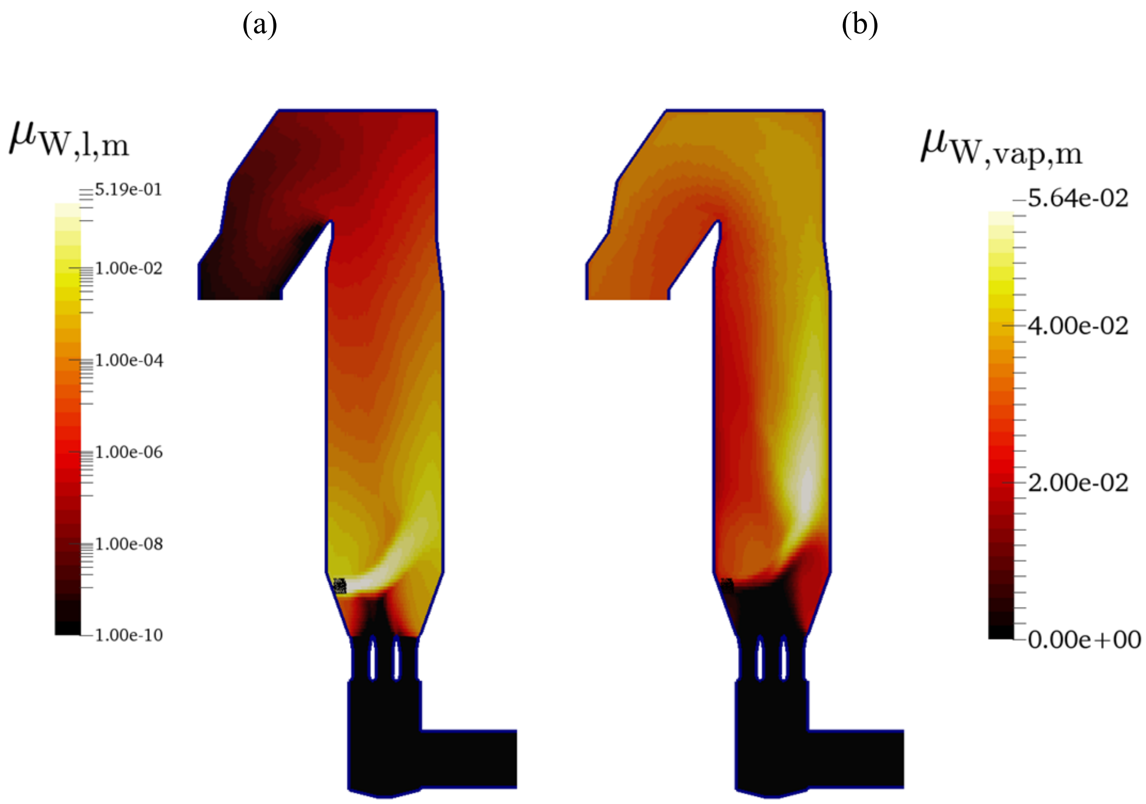

Figure 6.20 Time-averaged water mass loadings, i.e. (a) liquid and (b) vapor, in a longitudinal section of the 3D riser model (including quenching, t = 32 s). The water injection region is depicted with black lines.

In contrast to the 2D case, the flue gas below the quenching region is not in contact with the injected water droplets. This leads to locally low temperatures in the quenching region and almost no cooling of the flue gas below the water injection point. These conclusions are supported by the time-averaged temperature, the time-averaged liquid water mass loading and the time-averaged water vapor mass loading illustrated in Figure 6.19 and Figure 6.20.

Time-averaged quantities at the outlet are summarized in

Table 6.4. Different to the 2D riser simulation, the range of a quantity is reported if neither area-averaged nor flux-averaged data was available.

Table 6.4 Time-averaged outlet quantities obtained from the 3D riser simulation (including quenching, t = 32 s).

Variable	Value
Area-averaged gas velocity $U_{CG,m}$	(-1.13 0 -8.55) m/s
Temperature $T_{CG,m}$	398 … 401 K
Liquid water mass loading $\mu_{W,l,CG,m}$	0
Vapor mass loading $\mu_{W,vap,CG,m}$	$2.89 \cdot 10^{-2}$ … $3.34 \cdot 10^{-2}$

The gas velocity equals the expected one (reported in Table 5.1). The gas was not cooled down below 373 K, i.e. the evaporation temperature of water. The water mass loading (the sum of liquid and vapor content) is approximately the expected one (reported in Table 5.13).

As in the 2D model of the riser, the temperature, the liquid and evaporated water mass loadings have limits in the quenching model used, where the temperature in the quench water injection region constantly has been at the lowest limit. This limitation procedure on average might have caused the increased gas temperature.

6.2.4 Particle Injection

Figure 6.21 illustrates the gas velocity field of the simulation considering particle injection after it satisfied the pseudo-steady-state conditions at t = 30 s. The approach to pseudo-steady state was monitored by considering the transients of the time-averaged gas velocity, the time-averaged temperature and the time-averaged void fraction. Same as in the simulation in Section 6.2.3, the gas flow is dominated by the high gas velocity at the nozzles outlets, as well as the quench water injection velocity.

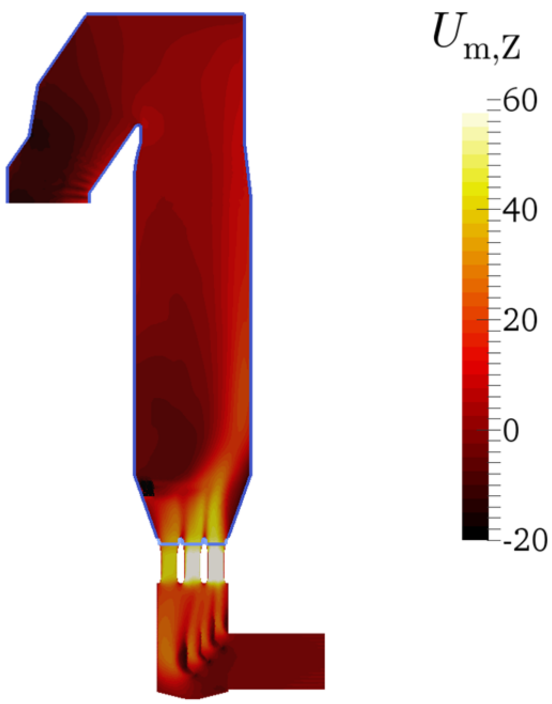

Figure 6.21 Time-averaged gas velocity in a longitudinal section of the 3D riser model (full model with particles, *t* = 30 s). The water injection region is depicted with black lines.

Same as in the 2D case, two size classes containing the largest (but most rare) particles have not been injected during the duration of the simulation. Immediately after injection, particles move downwards in a rather dense rope, approach the nozzle region, and are accelerated upwards rapidly by the flue gas exiting from the nozzles. Injected quench water leads to some mixing of the particles and the flue gas. No visible size-based segregation of particles could be observed, however, no attempt has been made to quantify segregation in the riser. In summary, a rather dense particle rope can only be observed in the region between the particle injection region (i.e., the exits of the chutes) and the nozzle outlet. Recordings of the particle positions reveal that it is possible for particles to move vertically downwards of the nozzles, despite the high (time-averaged) flow velocity in the nozzles. These conclusions are supported by parcels positions as illustrated in Figure 6.22.

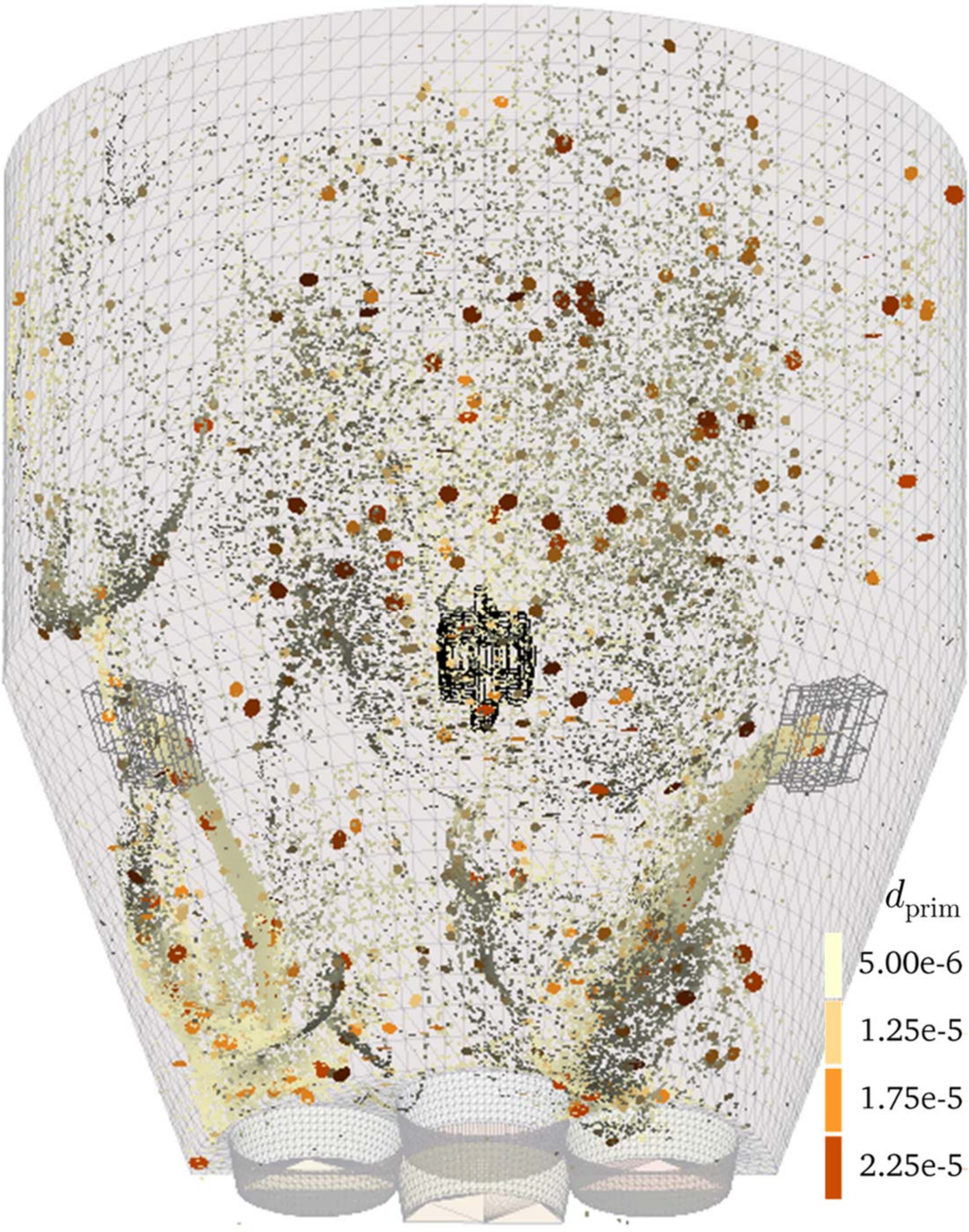

Figure 6.22 Snapshot of the particle cloud near the chute exit and the nozzles after *t* = 30 s. Parcels (filled circles) are magnified by a factor of 2.5 with face normals directing into the parcel flow direction. Injection regions are depicted with lines, i.e. water (black) and recirculate (grey).

In order to estimate the conditions, among them it is possible to have vertical downflow of particles through a nozzle, we have used a simple force balance over the height of the nozzle (see the illustration in Figure 6.23). The force balance considers the pressure before and after the nozzles, as well as the hydrostatic pressure due to a mean gas-particle mixture density ρ_{mix} in a single nozzle. Consequently, one can estimate a critical mean density in a single nozzle that would lead to vertical downflow:

$$\Delta p = \rho_{\mathrm{mix}}\, g\, h = \rho_{\mathrm{f}}\, U^2 / 2 \tag{6.1}$$

$$\rho_{\mathrm{mix}} = \varphi_{\mathrm{P}}\, \rho_{\mathrm{P}} + (1 - \varphi_{\mathrm{P}})\, \rho_{\mathrm{f}} \tag{6.2}$$

Given a nozzle height of h = 3.51 m, the average nozzle velocity from Table 5.1, and the fluid density from Table 5.2, we compute a critical density of the gas-particle mixture of ρ_{mix} = 29.2 kg/m^3. This corresponds to a particle volume fraction of $1.26 \cdot 10^{-2}$, which is larger than the largest particle volume fraction observed in the riser. However, it is not unrealistic that such a high mixture density might occur in the chutes for the recirculate injection. Also, the injected particles are initially accelerated downward by the flue gas, i.e., they gain momentum in the negative z-direction before they approach the nozzle region. This explains the observed downflow of particles through the nozzle region in the simulation.

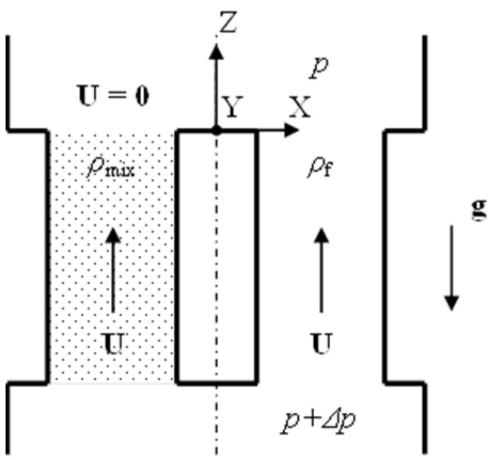

Figure 6.23 Sketch of a blocked nozzle.

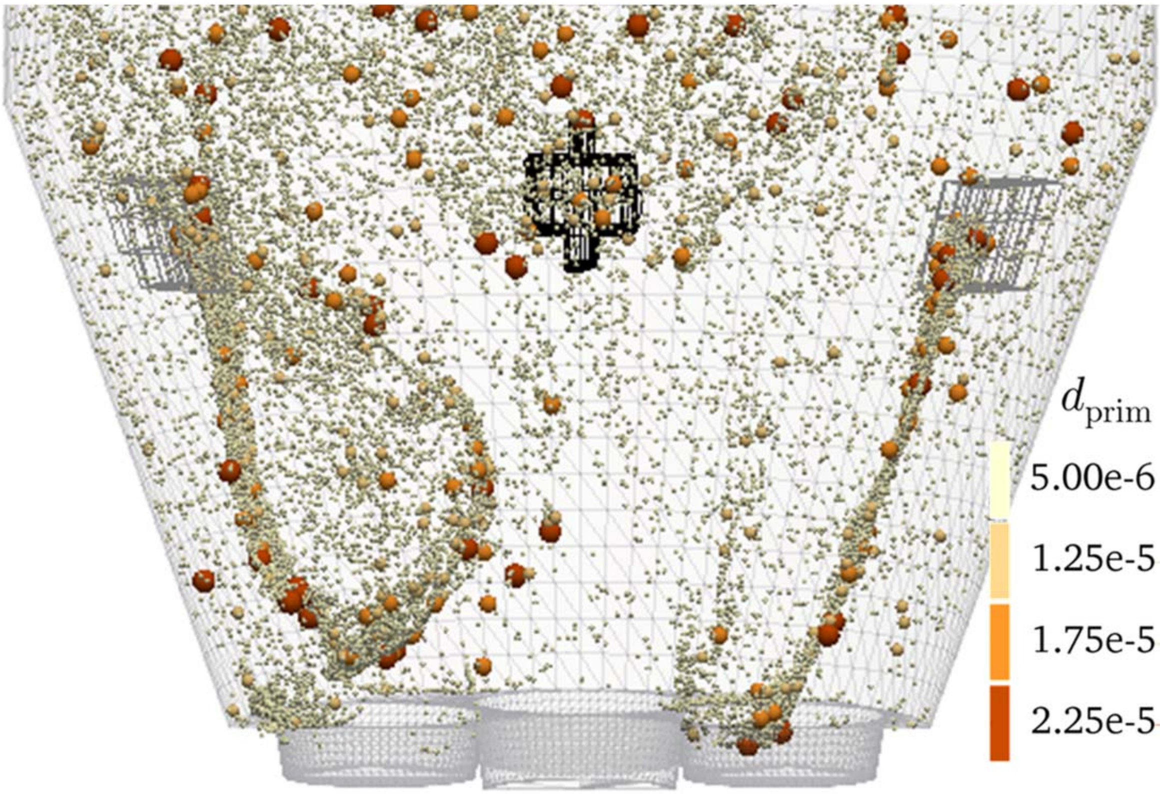

Figure 6.24 Snapshot of the particle cloud approaching the nozzle region after *t* = 16.2 s. Parcels are magnified by a factor of 2.5. Injection regions are depicted with lines, i.e. water (black) and recirculate (grey).

Figure 6.24 illustrates a situation in which particles enter the nozzle section vertically downwards. The result of an analysis of parcels passing the bottom outlet of the simulation domain is summarized in Table 6.6. The time-averaged particle volume fraction in a cross-section through all nozzles was computed as follows:

$$\varphi_{\mathrm{P}}\Big|_0^{t_{\mathrm{sim}}} = \frac{M_{\mathrm{P,N}} / \rho_{\mathrm{P}}}{M_{\mathrm{P,N}} / \rho_{\mathrm{P}} + \dot{V}_{\mathrm{FG}} t_{\mathrm{sim}}} \tag{6.3}$$

Given simulation time, particle mass passing the bottom outlet (both reported in Table 6.6), the flow rate of flue gas (reported in Table 5.1), and the fluid density (reported in Table 5.2), the time-averaged particle volume fraction near the nozzles is $\varphi_{\mathrm{P}}\Big|_0^{t_{\mathrm{sim}}} = 7.81 \cdot 10^{-6}$. This corresponds to a mixture density of $\rho_{\mathrm{mix}} = 0.822$ kg/m^3, i.e., close to that of the flue gas. This indicates that downflow cannot occur in case time-averaged quantities are considered. As one can see in Figure 6.22 and Figure 6.24, a dense rope of particles arrives at the nozzles. A movie illustrating the motion of the dense rope of particles indicates fast dynamics and short periods of time in which the rope is focused on exactly one nozzle. The particle volume fraction at the injection point (reported in Table 5.4) indicates that such a dense rope of particles can indeed induce a local downflow through the nozzles. Moreover, downflow conditions may occur for the whole range of available injection velocities considered in this work. Thus, an obvious solution, i.e., a strategy to prevent downflow in the nozzle region, is to lower the particle volume fractions at the particle injection point.

Profiles of time-averaged temperature and water mass loadings are illustrated in Figure 6.25 and Figure 6.26. The temperature oscillated in the backflow region near the outlet. Its origin can be traced back to the oscillations in the solution of the vapor mass loadings in Figure 6.26, similarly to the 2D particulate case. The time-averaged temperature is almost homogeneous above the injection regions, indicating a sufficiently fast mixing of quench water and the particle-laden gas.

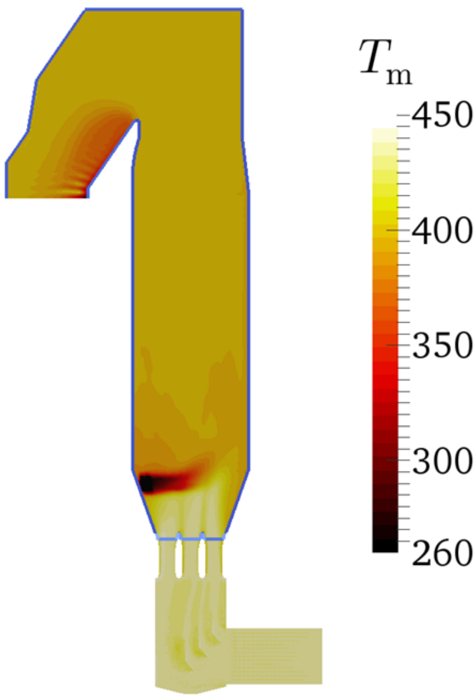

Figure 6.25 Time-averaged temperature profile in a longitudinal section of the 3D riser model (full model with particles, *t* = 30 s). The water injection region is depicted with black lines.

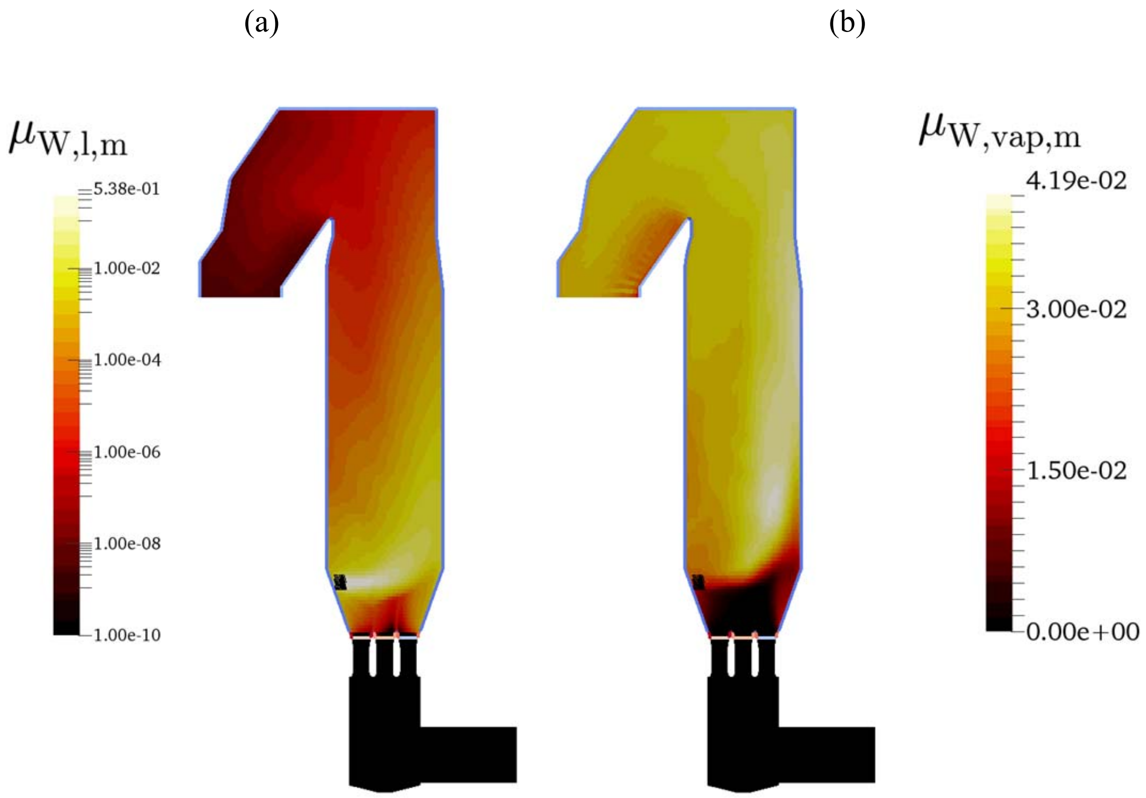

Figure 6.26 Time-averaged water mass loadings, i.e. (a) liquid and (b) vapor, in a longitudinal section of the 3D riser model (full model with particles, t = 42.6 s). The water injection region is depicted with black lines.

As flux-averaging is not applicable to a surface with backflow, ranges of certain quantities on the outlet are summarized in Table 6.5. The estimated average gas velocity at the outlet (reported in Table 5.1) is within the range. The estimated temperature at the outlet (reported in Table 5.13) is within the range (reported in the table above). The estimated water mass loading (reported in Table 5.13) is insignificantly larger than the upper range limit due to backflow from the outlet that is void of water. The time-averaged particle volume fraction at the outlet ranges below the calculated one (reported in Table 5.3 and Table 5.5). Note, that the calculation is based on the mass flow rates of flue gas and injected particles.

Table 6.5 Time-averaged outlet quantities obtained from the 3D riser simulation (full model with particles, t = 42.6 s).

Variable	Value
Gas velocity $U_{CG,m}$	(-3.3 … 9.38 -2.98 … 0.738 -14.8 … 15.1) m/s
Particle volume fraction $\varphi_{P,CG,m}$	0 … $1.42 \cdot 10^{-4}$
Temperature $T_{CG,m}$	278 … 401 K
Liquid water mass loading $\mu_{W,l,CG}$	0
Vapor mass loading $\mu_{W,vap,CG}$	$1.74 \cdot 10^{-2}$ … $3.11 \cdot 10^{-2}$

Table 6.6 Integral mass balance of the 3D riser simulation (full model with particles, t = 85.2 s)

Variable	Value
Total target particle insertion rate $\overline{\dot{M}_{P,target}}$	156 kg/s
Average particle insertion rate $\left.\overline{\dot{M}_{P}}\right\|_0^{2s}$	122 kg/s
Simulated duration t_{sim}	85.2 s
Total particle mass $\sum_{domain} M_P$	1,440 kg
Particle mass passing through the outlet $\sum_{CG} M_P$	4,000 kg
Particle mass passing the bottom outlet $\sum_{N_1}^{N_7} M_P$	665 kg
Total injected particle mass M_P	6,110 kg
Effective average particle insertion rate $\overline{\dot{M}_P}$	71.7 kg/s

An integral particle mass balance was conducted for a limited region of the riser ranging from above the nozzles to the outlet, as summarized in Table 6.6. Therefore, the total mass of all recorded particles (leaving and remaining in the domain) is computed. This value should equal the mass of the inserted particles since the simulation was started. Next, the effective average integral insertion rate can be computed as follows:

$$\overline{\dot{M}_{\mathrm{P}}} = \frac{\sum M_{\mathrm{P}}}{t_{\mathrm{sim}}} = \frac{\sum_{\mathrm{domain}} M_{\mathrm{P}} + \sum_{\mathrm{out}} M_{\mathrm{P}}}{t_{\mathrm{sim}}} \tag{6.4}$$

The effective average particle insertion rate gathered from that integral mass balance is 41 % lower than the one gathered from the first two seconds. The exact reason for this much lower insertion rate is unclear, and could be caused by parcels penetrating through the wall. Unfortunately, parcels penetrating the wall could not be avoided, and parcels penetrating the wall have not been tracked. Time- and domain-averaged quantities at pseudo-steady state are summarized in Table 6.7.

Table 6.7 Time- and domain-averaged quantities obtained from the 3D riser simulation (full model with particles, t = 30 s).

Variable	Value
Particle volume fraction $\overline{\langle \varphi_{\mathrm{P}} \rangle}$	$1.45 \cdot 10^{-4}$
Hold up $\overline{\langle \varepsilon_{\mathrm{P}} \rangle}$	0.326 kg/m^3

Particle hold-up is not as expected by the industrial partner as it is approximately the calculated one (reported in Table 5.3 and Table 5.5) based on the flue gas and injected particle mass flow rates. Note that time-averaging of (discrete) particle data was conducted differently to (continous) field data: First, a domain-averaging was performed, utilizing a built-in function of OpenFOAM®. Next, time-averaging was performed after the simulation by computing the mean over the recorded data.

In contrast to the 2D riser simulation, the parcel size distribution remains almost constant as illustrated in Figure 6.27. Thus, large particles do not segregate in such well-mixed particulate flow leading to approximately the mass-flow-based hold-up.

(b)

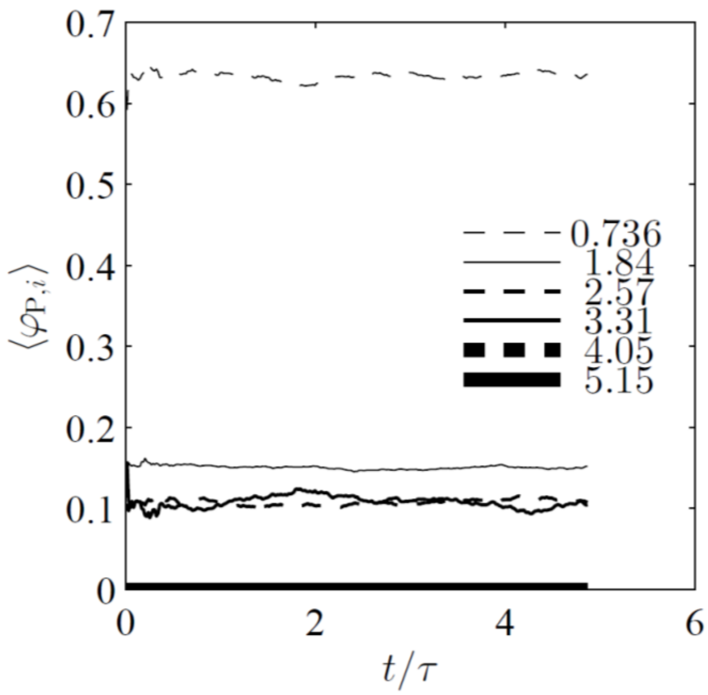

Figure 6.27 Domain-averaged parcel size distribution in the 3D riser simulation versus time. Domain-averaged particle volume fraction of individual classes normalized for a total over classes of 1. Class mean particle diameters displayed in the legend are normalized with the Sauter mean diameter $d_{32} = 6.80 \cdot 10^{-6}$ m. The mean gas residence time is $\tau = 8.72$ s.

As in Section 6.1.4 for the 2D riser simulation, a quantitative picture of the domain-averaged parcel size distribution in the 3D riser can be provided in conjunction with Figure 6.28.

(a)

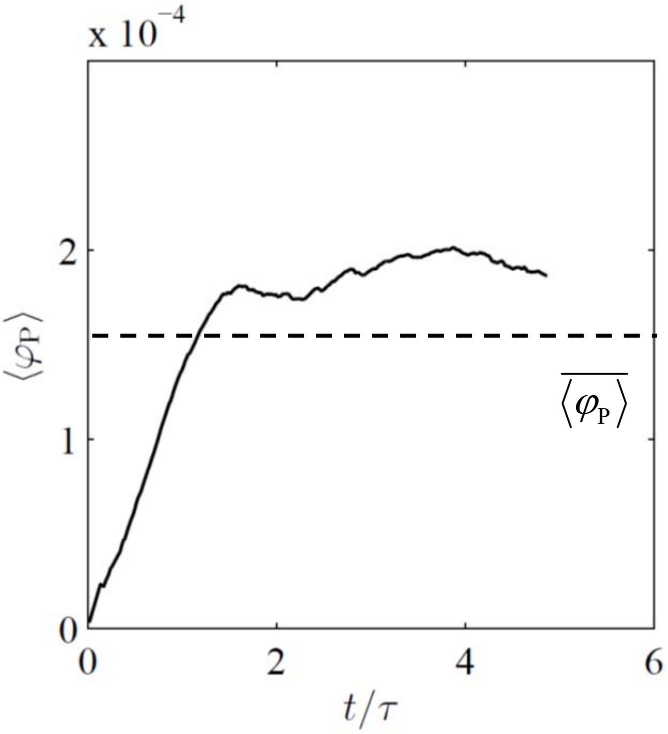

Figure 6.28 Domain-averaged particle volume fraction in the 3D riser simulation versus time. The mean gas residence time is τ = 8.72 s.

6.3 Variations of the Simulation Setup

For the 2D simulations only the drag correction factor has been varied. For the 3D simulations, however, the drag correction factor and the distribution of the particle cloud (i.e., polydisperse or monodisperse) has been varied.

Table 6.8 compares 2D with 3D polydisperse cases. They achieve pseudo-steady state at approximately the same time. The time-averaged outlet temperature ranges between similar limits. Hold up is lower than expected. This might be due to the fact that cohesion, and hence the agglomeration of primary particles, was not included in the model.

Table 6.8 Comparison of key results for polydisperse particulate flow in the riser.

Parameter / Variable	Value			
	2D		3D	
Average grid size l_{grid} [m]	0.118		0.152	
Grid resolution $l_{\text{grid}} / d_{\text{P,max}}$	3.31		1.25	
Drag correction factor	1.00	0.130	1.00	0.130
Time required to achieve pseudo-steady state [s]	30	40	30	25
Time-averaged outlet temperature $T_{\text{CG,m}}$ [K]	271 … 389	270 … 386	278 … 401[(8)]	274 … 403[(8)]
Domain-averaged particle volume fraction $\overline{\langle \varphi_{\text{P}} \rangle}$	$2.74 \cdot 10^{-5}$	$1.78 \cdot 10^{-5}$	$1.45 \cdot 10^{-4}$	$1.38 \cdot 10^{-4}$
Hold up $\overline{\langle \varepsilon_{\text{P}} \rangle}$ [kg/m^3]	$6.17 \cdot 10^{-2}$	$4.01 \cdot 10^{-2}$	0.326	0.310

A reduced drag coefficient leads to increased slip velocity (see Eqn. (3.12)), and consequently to a larger hold up. In the jet-driven flow studied, however, particles may on average move faster than the average gas velocity. This leads to a lower hold up as expected from the mass flow rates. Also, the sedimentation velocity is much smaller than the average jet velocity. This explains the fact that a drag correction does not affect the hold up in the 3D model. Interestingly, the drag correction affects the hold up only in the 2D model where the domain-averaged particle volume fraction is lower than calculated due to segregation effects (parcel-size-dependent residence time).

[8] Not available for minimum time required for pseudo-steady-state conditions. Shown at later times as follows: 42.6 s (polydisperse, no correction), 30.8 s (polydisperse, drag correction), 51.8 s (monodisperse, no correction), 51.9 s (monodisperse, drag correction), 31.1 s (polydisperse, zero quench water velocity)

Table 6.9 compares 3D cases, monodisperse against polydisperse with and without drag correction. Also, a polydisperse simulation including quench injection velocity (base case) is compared with the results of a simulation without quench injection velocity.

Table 6.9 Comparison of key results for the 3D riser simulations.

Parameter / Variable	Value				
Particle size distribution	Mono		Poly		Poly
Drag correction factor	1.00	0.130	1.00	0.130	1.00
Quench water velocity [m/s]	50				0
Time required to achieve pseudo-steady state [s]	50	50	30	25	25
Outlet temperature $T_{\mathrm{CG,m}}$ [K]	262 … $397^{(8)}$	261 … $397^{(8)}$	278 … $401^{(8)}$	274 … $403^{(8)}$	330 … $415^{(8)}$
Domain-averaged particle volume fraction $\overline{\langle \varphi_{\mathrm{P}} \rangle}$	$2.01 \cdot 10^{-4}$	$1.96 \cdot 10^{-4}$	$1.45 \cdot 10^{-4}$	$1.38 \cdot 10^{-4}$	$1.06 \cdot 10^{-4}$
Hold up $\overline{\langle \varepsilon_{\mathrm{P}} \rangle}$ [kg/m^3]	0.452	0.441	0.326	0.310	0.240

Hold up is not significantly affected by drag correction or particle size distribution. Lowering the quench velocity to zero decreases the holdup by ca. 23 %. Constant hold up arises from well-mixed parcels discussed in the previous section (see domain-averaged parcel size distribution as a function of time in Figure 6.27. Monodisperse riser cases achieve pseudo-steady state later than polydisperse ones. One can conclude that injected equal-sized particles follow similar trajectories, while a polydisperse particle population follows multiple trajectories, leading to more efficient dispersion. We speculate that this might have caused an early approach to the pseudo-steady state. The time-averaged outlet temperature is only in one case larger, the one where quench water enters with (almost) no velocity. This is due to the less intense dispersion of water droplets, leading to less efficient heat and mass transfer. The temperature profile corresponds with the gas velocity profile, i.e., a region with a low temperature forms near the riser wall.

Comparing water droplet locations with parcel locations by overlaying the following figures, one can identify (qualitatively) regions with a high probability of droplet-particle interactions. These regions are located close to the injection points of the recirculate and the water droplets. In these regions particles might be significantly more cohesive, and agglomerates might form. One should note that the present quench model relies on the assumption that particles are not in contact with liquid water.

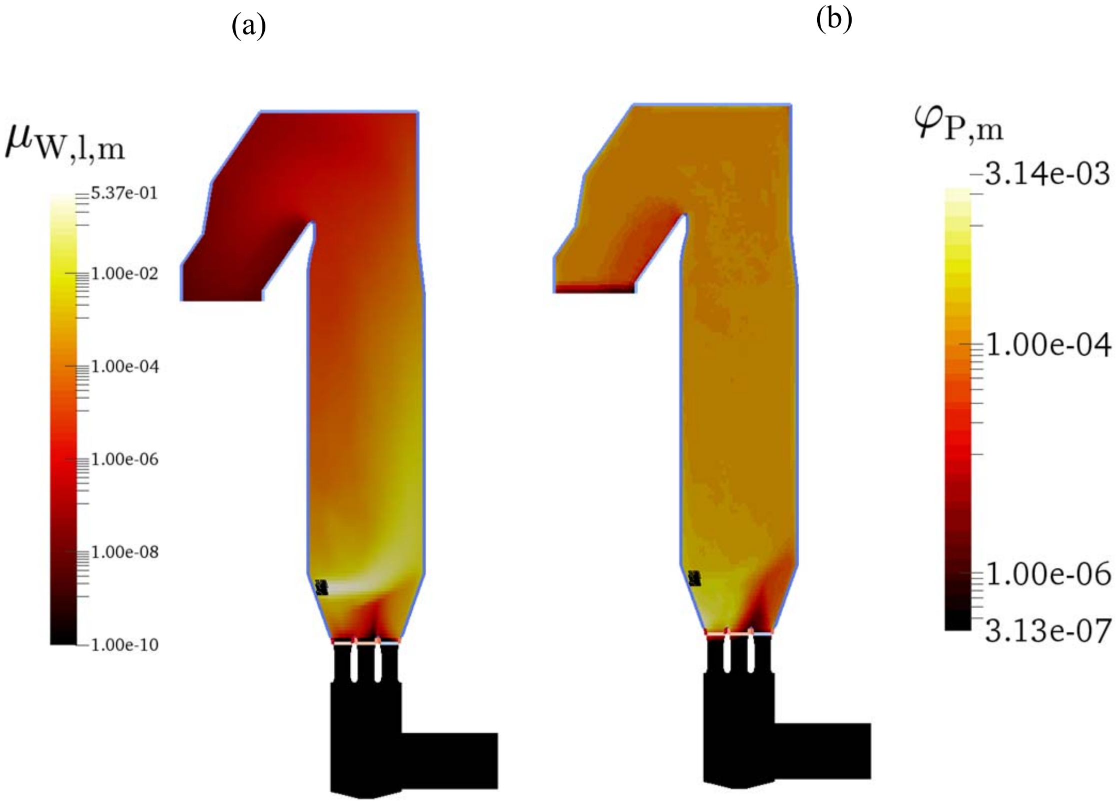

Figure 6.29 Time-averaged liquid water mass loading (a) and void fraction (b) in a longitudinal section of the 3D riser model (monodisperse without drag correction, *t* = 51.8 s). The water injection region is depicted with black lines.

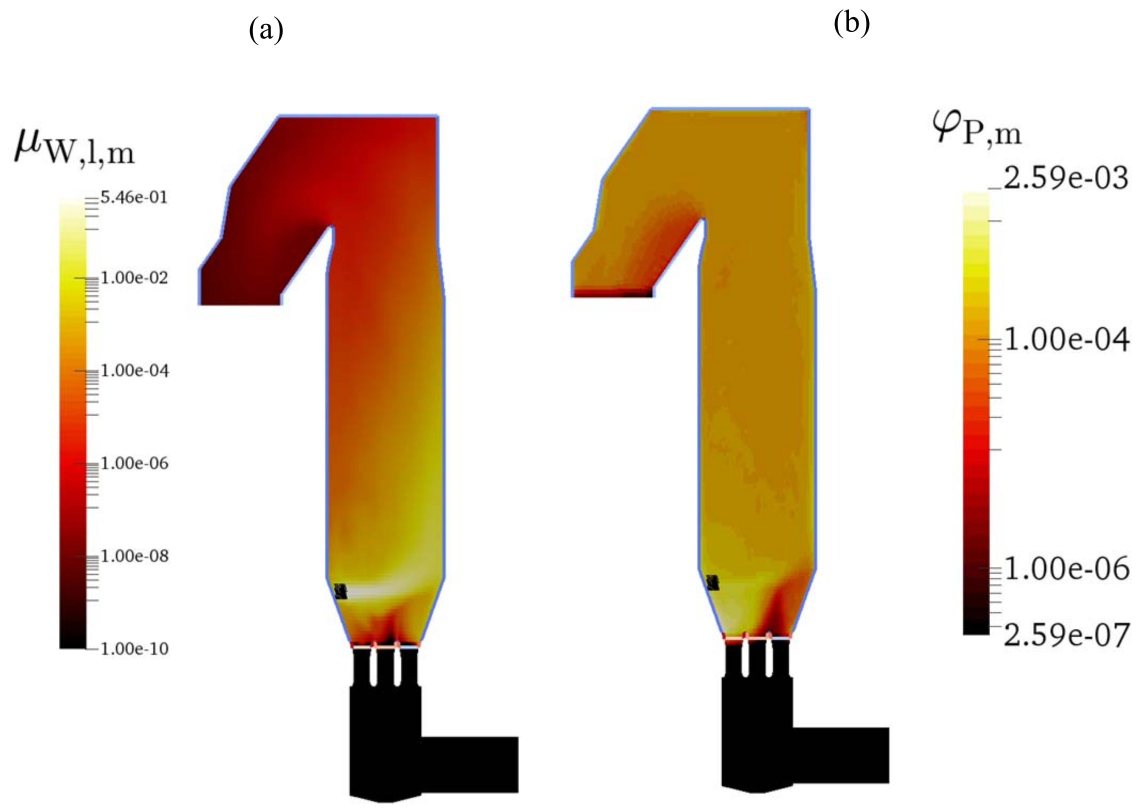

Figure 6.30 Time-averaged liquid water mass loading (a) and void fraction (b) in a longitudinal section of the 3D riser model (monodisperse with drag correction, t = 51.9 s). The water injection region is depicted with black lines.

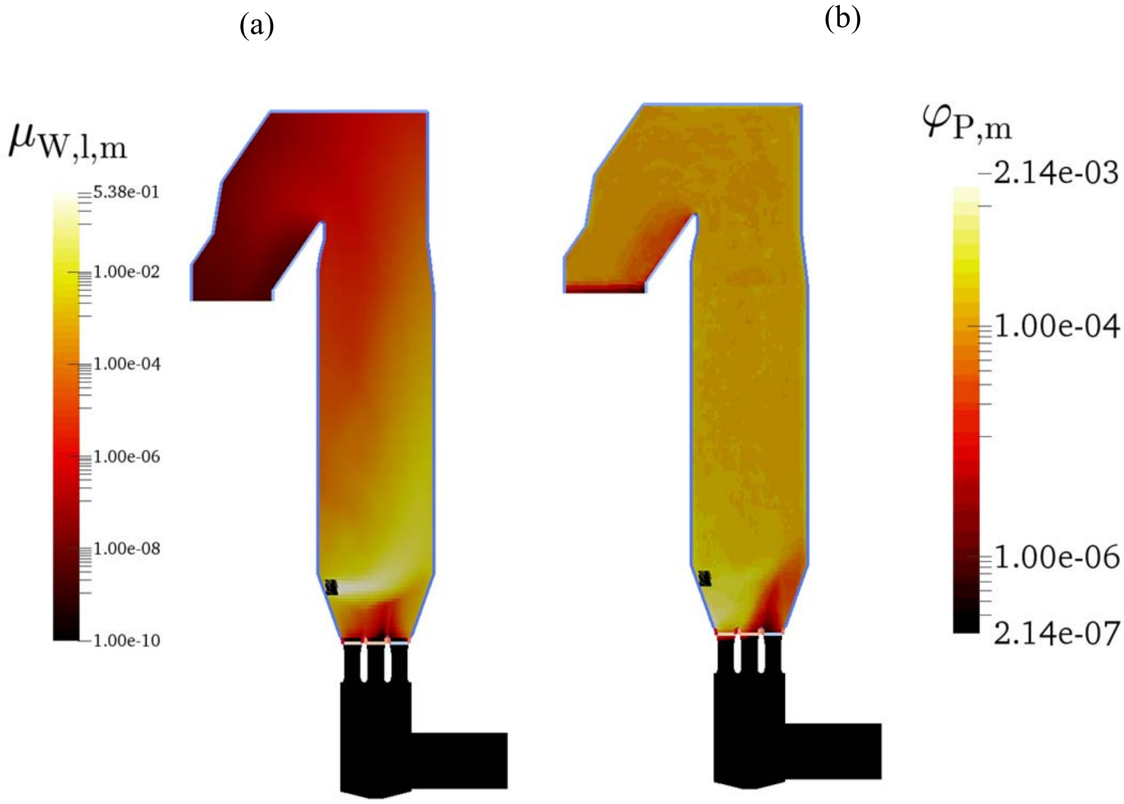

Figure 6.31 Time-averaged liquid water mass loading (a) and void fraction (b) in a longitudinal section of the 3D riser model (polydisperse without drag correction, *t* = 42.6 s). The water injection region is depicted with black lines.

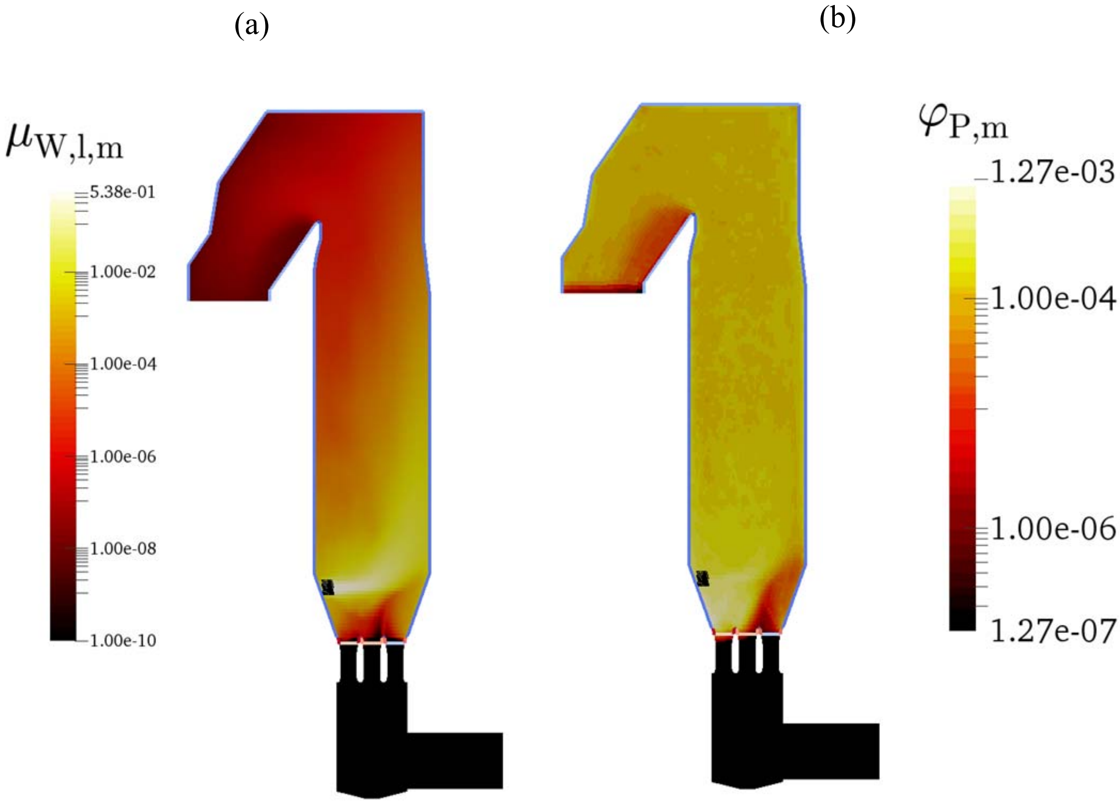

Figure 6.32 Time-averaged liquid water mass loading (a) and void fraction (b) in a longitudinal section of the 3D riser model (polydisperse with drag correction, *t* = 30.8 s). The water injection region is depicted with black lines.

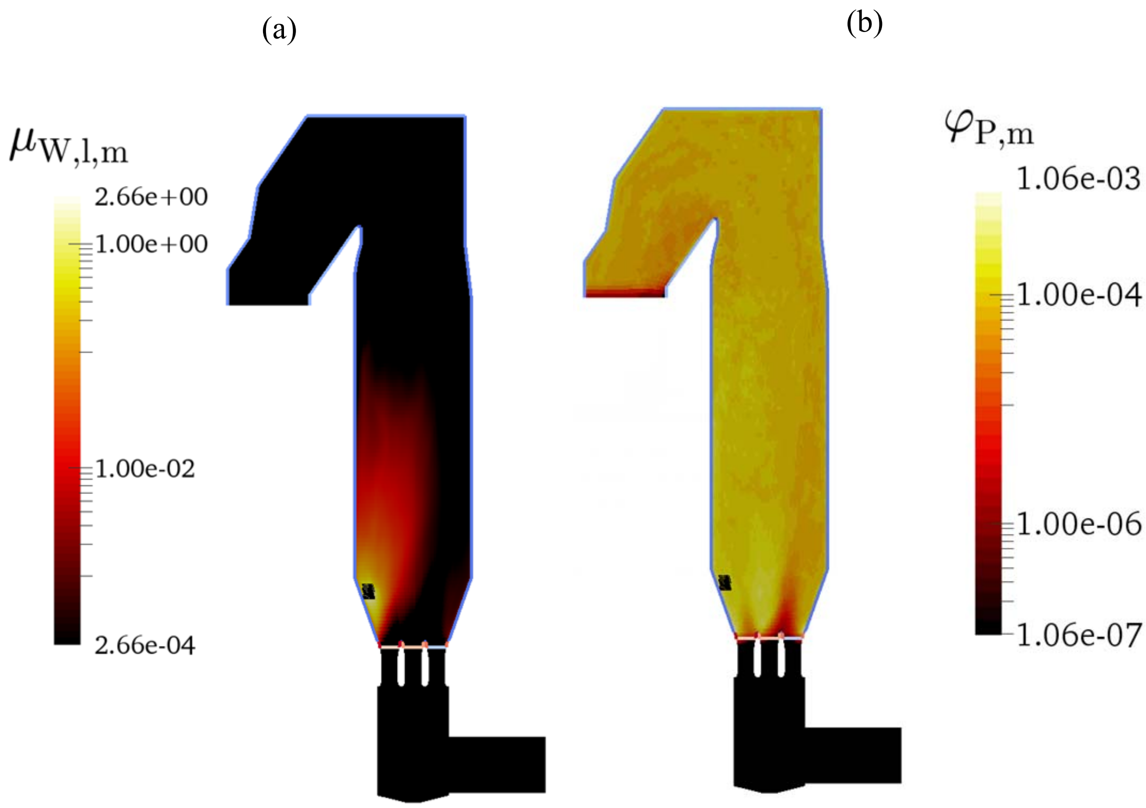

Figure 6.33 Time-averaged liquid water mass loading (a) and void fraction (b) in a longitudinal section of the 3D riser model (polydisperse without drag correction and zero quench injection velocity, *t* = 31.1 s). The water injection region is depicted with black lines.

In case particles are not influenced by the quench injection velocity, they distribute more evenly in the region below the injection point and are mainly influenced by the nozzle jets (shown in Figure 6.33 b). In this case water is significantly slower dispersed across the riser cross-section, and a large region with a high droplet mass loading forms in the vicinity of the water injection point (shown in Figure 6.33 a). Hence, the total liquid surface area is reduced, and the evaporation rate drops. Thus, we conclude that such a situation is unwanted, since particles entering the region of high droplet concentration would become sticky and consequently agglomerate or stick to the riser walls.

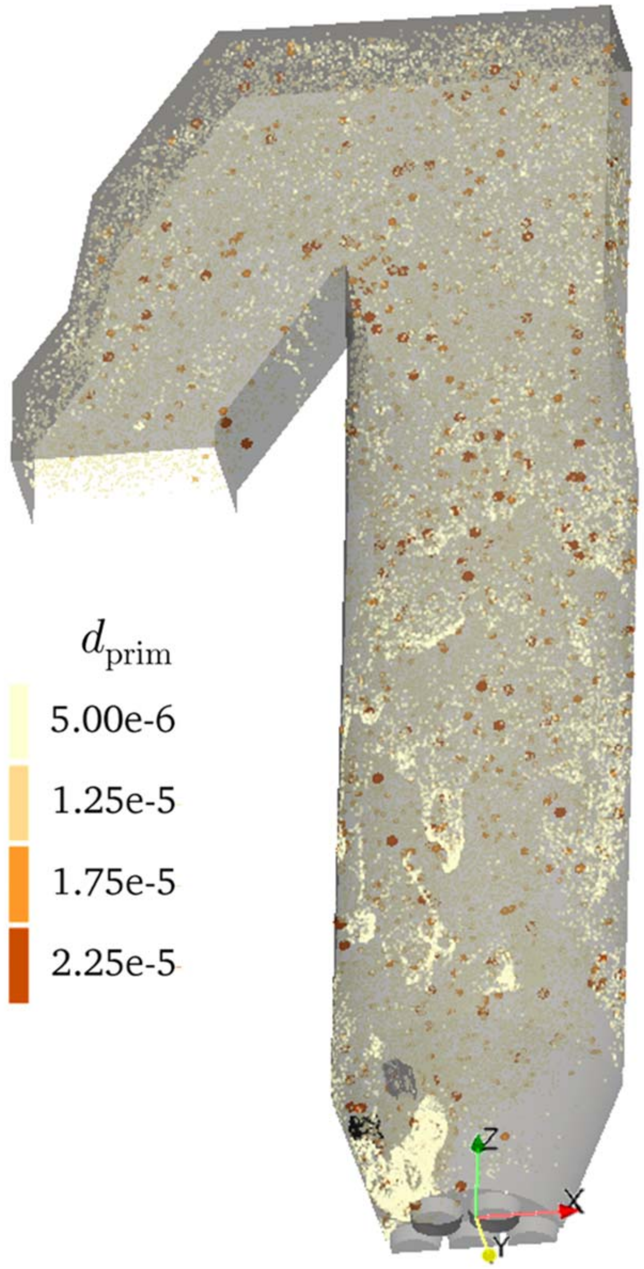

Figure 6.34 Snapshot of the polydisperse particle cloud in the 3D riser simulation (without drag correction, *t* = 30 s). Parcels (filled circles in XZ-plane) are magnified by a factor of 5. Injection regions are depicted with lines, i.e. water (black) and recirculate (grey).

Figure 6.34 provides a snapshot of the parcel cloud within the riser. Here one can see that small particles tend to form comparably large clusters, and that there is (qualitatively) no particle segregation in the riser. Also, particles appear to be well-dispersed across the riser cross-section.

7 Conclusions and Recommendations

This thesis made an attempt to better understand the effect of drag models on the predictions for polydisperse particle flow in a full-scale riser. First, a drag model was investigated by performing simulations of a freely sedimenting particle cloud in a periodic box for a limited range of particle volume fractions. These simulations were performed both with a highly resolved CFD grid, as well as using coarse-grained simulations. This was done in order to find proper settings for the drag model, as well as the smoothing algorithm. It was found that the smoothing length has a significant effect on the predicted gas-particle slip velocity. Hence, a model for estimating the appropriate smoothing length as a function of the domain-average particle concentration and the coarse graining ratio was developed. This model helps to avoid unphysical fluid agitation that is introduced in case large coarse graining ratios are used. Finally, a drag correction factor was calculated in order to account for the effect of unresolved meso-scale structures in simulations of the full-scale riser.

7.1 Review of Goals

The models of Gidaspow (1994) and Holloway et al. (2010) showed no meaningful results in preliminary studies. Hence, in this thesis only the monodisperse drag model of Beetstra et al. (2007) was investigated in higher detail in the periodic domain and full-scale riser simulations. This model gave best qualitative agreement to experimental data (Goldschmidt et al., 2003) of a dense, bi-disperse (particle ratio 1.64) gas-solid fluidized bed (González, 2013, p. 75; Radl et al., 2014, p. 5). Also, Capecelatro and Desjardins used a drag law (2013, p. 8) that has been designed for monodisperse particle beds (Radl et al., 2014, p. 5). The contribution due to fluid-mediated particle-particle drag (suggested by Holloway et al. (2010)) was not used, since the effect of fluid-mediated drag is expected to be small (González, 2013, p. 51).

7.1.1 Periodic Box Simulations

Filter parameters were fitted and added to the implementation of the monodisperse drag model of Beetstra et al. (2007) in CFDEM® already in a previous work by Radl and

Sundaresan (2014, p. 420). Unfortunately, the filtered model of Holloway and Sundaresan (2014, p. 74,77) does not provide any constitutive formulae for the filtered drag coefficient, but only trend curves highlighting the need for a correction of the drag force. Hence, the model of Holloway and Sundaresan (2014) was not incorporated into CFDEM® at this point of time. None of the filtered drag models described in Section 3.1.4 and 3.1.5 was used. Instead, the drag model of Beetstra et al. (2007) was extended by a simple filter model in CFDEM®, which enables the application of a constant drag correction factor.

In summary, it was confirmed that the exact details of the drag model (e.g., with or without a correction for particle clustering) have little effect on the predicted flow of the particles. This is because of the extremely small particle size (Geldart A and C), and the fact that the flow in the riser is characterized by a high-velocity region at the inlet. Particle-fluid drag models available in literature have been developed for larger particles (i.e., Geldart A and B group) and for freely sedimenting suspensions. The application of literature models hence has to be treated with care for the riser studied in this work.

To begin with the comparison to the initially planned tasks, a sensitivity study with respect to the parcels size and grid resolution was performed in a periodic box in a different manner. The parcel size was varied during the investigation of the effect of smoothing on the coupling fields. The grid resolution has been changed for the large periodic boxes to match the average cell in the riser model. Instead of a sensitivity study with respect to temperature gradients, the temperature field has only been considered in those riser simulations which use a quenching model.

In the periodic box simulations the domain-averaged slip velocity and the drag correction factor for each size fraction, stress, and the particle-phase viscosity have not been recorded. Stress and particle-phase viscosity are considered to be negligible in the present dilute system. Recording the drag correction factor for each size fraction is only recommended in case larger particles are considered, and the flow situation is closer to that of a freely-sedimenting suspension. In case of a flow that is agitated with a high-velocity jet, the effect of particle clustering can only be evaluated by performing highly-resolved simulation of a relevant flow configuration. Such a situation could be the injection of particles into a turbulent cross flow. Unfortunately, the extraction of a drag correction model is then possible only using advanced filtering tools, since a periodic flow cannot be used.

Next, a drag correction factor was calculated for fully-resolved cases and coarse-grained cases (with coarse grid size). The factor is based on the computed slip velocity and the terminal settling velocity of an isolated particle with the same d_{32} as the particle cloud. The drag correction factor was calculated for four cases, focusing on dilute suspensions with particle volume fraction from 10^{-3} to $2 \cdot 10^{-3}$.

7.1.2 Riser Simulations

Also, the flow in a full-scale riser was modeled using a 2D and 3D model. Successively, the effect of LES and URANS turbulence models, as well as the quench water injection on the flow, and subsequently, on the particulate flow, have been investigated. Simulations using the 2D model provided some insight on how key simulation parameters (e.g., the velocity of the injected quench water) affect the flow. This insight guided the 3D simulations. Finally, unfiltered cases were modified by applying the largest drag correction determined (i.e., applying the worst case scenario). This was done for 2D, 3D, polydisperse and monodisperse cases in order to account for the effect of unresolved meso-scale structures in simulations of the full-scale riser.

Moreover, the initial plan was to quantify the segregation state and the rate of fines elutriation. Segregation effects are only visible in simulations of polydisperse gas-solid suspensions using the 2D model. The state of segregation rates were not quantified for the 3D domain, since qualitatively no segregation was observed. The rate of fines elutriation was quantified by considering the domain-averaged particle hold up in each size fractions as a function of time. The results show that in the 3D riser simulations the size distribution is rather constant compared to the 2D riser simulations. In the latter, large particles leave the domain earlier than the smaller ones. Unfortunately, experimental data was not available to assess whether the drag model yields realistic results.

Finally, a User Defined Function (UDF) for the smoothing model to be used in the DDPM solver of the ANSYS® Fluent® software was implemented. This model can be only applied to the particle concentration field, since the fields for the coupling forces could not be accessed (Holzinger, 2014c).

7.2 Results Summary

Key parameters found in simulations using the 2D model are as follows:

- The author suggests to use an LES-type models and a sufficiently fine grid rather than an URANS turbulence models. This is in accordance with literature.
- Quenching has an effect on the particulate flow due to the high quench water injection velocity (i.e., the quench water velocity is in the order of the velocity of the incoming gas jet, and hence both flows dominate)

In full-scale riser simulations using 2D and 3D models, the resulting holdup is much lower than expected. This might be due to the fact that cohesive forces, and hence agglomeration, was not modelled. Such effects can be expected, since the majority of the particles in the riser are Geldart C particles. As (the small) particles in the distribution used are likely to agglomerate, it may be a good idea to use the size distribution of particle agglomerates or aggregates in future studies.

Surprisingly, drag correction affected the hold up only in the 2D model.

7.3 Recommendations

Strong (force) coupling (CF approx. 100 or even lower) is essential in CFDEM®, since preliminary cases did not converge. In cases where the communication with the CFD side was too loose, particle forces and velocities increased, leading to divergence of the algorithm.

7.3.1 Smoothing of Coupling Fields

Since the predicted sedimentation velocity of the monodisperse particle population depended on the initial spatial distribution, further investigations were based on a polydisperse particle population.

It was found that without coarse graining the smoothing length has no influence on the domain-averaged slip velocity. Subsequently, the smoothing length to parcel diameter ratio was determined such that the domain-averaged slip velocity of a coarse-grained case equals the uncoarsened case. For low coarse graining ratios up to 25, a theoretically-derived smoothing law can be used when adapted slightly. The observed smoothing length to parcel diameter ratio is by a factor of $\left(\frac{\pi}{6}\right)^{1/3}$ smaller than the theoretical smoothing law. This suggests that the theoretical smoothing law has to be adapted to a cubic reference volume (for the case of small coarse graining ratios). In case of larger coarse graining ratios, more smoothing is needed to account for the intra-parcel distribution of particles. Consequently, an additional smoothing function based on linear interpolation is proposed within this thesis. In case of extreme coarse graining, the smoothing length scale approaches the dimensions of the domain, i.e., the formation of structures will be completely suppressed. Thus, a smoothing law without adaption to the coarse graining ratio is a reasonable choice to picture at least some meso-scale structures. Clearly, future investigations (using expensive 3D simulations with different coarse graining ratios) are needed to sharpen the picture on the correct smoothing law.

7.3.2 Future Improvement of the Riser Design

As the single radial injection of quenching water forces particles to hit the opposite wall, more quenching inlets distributed along the perimeter are recommended for a future design of the riser. The single mean jet length is estimated to be of the order of the riser diameter with regard to liquid water. Hence, designing three or more quench water inlets would lead to an improved quench water distribution.

The droplet diameter has a significant effect on the evaporation rate; hence atomizers that generate smaller droplets should be preferred to reduce the droplet concentration.

In order to operate safely without downflow due to a large concentration of particles next to the nozzles, more recirculate chutes distributed along the perimeter are recommended. Given the operating injection velocity, four times more chutes, or four times wider chutes would be required. In order to keep the amount of fluidization air needed for the recirculate injection chutes as small as possible, an idea would be to use the incoming flue gas to fluidize and disperse the recirculated particles.

Also, the injection height and the direction of the recirculate injection should be re-thought. Since the vertical downflow of flue gas near the recirculate injection point accelerates the particles in the negative vertical direction (i.e., downwards), a lower recirculate injection point, as well as a tangential injection direction could lead to a faster dispersion of the particles. An optimal injection strategy would ensure that each nozzle is approached by a similar mass flow rate of particles. For example, this could be realized by a swirling motion of the incoming particle jets. This could reduce the tendency for particles to move vertically downwards through the nozzles, improve the distribution of the particles in the lower section of the riser, increase the particle holdup, and hence increase the gas-particle mass transfer characteristics of the riser.

In order to prevent the recirculated particles from agglomeration, the particle injection region should be separated from the quench water injection point. Hence, the injection chutes should be placed next to the jet nozzles for maximum distance to the quenching zone. Another strategy would be to inject the quench water before the nozzles.

In summary the following recommendations for a future improvement of the riser design are:

- inject the same rate of quenching water at three locations distributed along the perimeter of the riser at angles between nozzle positions for good angular distribution and prevention of particles sticking to walls,
- inject the same rate of recirculate at eight locations distributed along the perimeter of the riser next to the nozzles to prevent downflow in nozzles. Alternatively, re-locating and re-directing the recirculate injection chutes to induce a swirling motion for a more homogeneous distribution of the injected particle cloud among the nozzles,
- us a side stream of the incoming flue gas to fluidize and disperse the injected particles,

- investigate the option of injecting the quench water before the nozzles to minimize the contact of particles with water droplets.

8 References

Ammour, D., 2013. Highly Resolved LES and Tests of The Effectiveness of Different URANS Models for The Computation of Challenging Natural Convection Cases. University of Manchester.

Andrews, M.J., O'Rourke, P.J., 1996. The Multiphase Particle-in-Cell (MP-PIC) Method for Dense Particle Flows. Int. J. Multiph. Flow 22, 379–402.

Beetstra, R., Hoef, M.A. Van Der, Kuipers, J.A.M., 2007. Drag Force of Intermediate Reynolds Number Flow Past Mono- and Bidisperse Arrays of Spheres. AIChE J. 53, 489–501. doi:10.1002/aic

Bolz, R.E., Tuve, G.L., 1976. Handbook of Tables for Applied Engineering Science, 2nd ed. CRC Press.

Capecelatro, J., Desjardins, O., 2013. An Euler–Lagrange strategy for simulating particle-laden flows. J. Comput. Phys. 238, 1–31. doi:10.1016/j.jcp.2012.12.015

Courant, R., Friedrichs, K., Lewy, H., 1928. Über die partiellen Differenzengleichungen der mathematischen Physik (About partial difference equations in mathematical physics). Math. Ann. 100, 32–74.

Crowe, C.T., Group, F., 2006. Multiphase Flow Handbook.

Cundall, P.A., Strack, O.D.L., 1979. A discrete numerical model for granular assemblies. Géotechnique 29, 47–65. doi:10.1680/geot.1979.29.1.47

De Villiers, E., 2006. The Potential of Large Eddy Simulation for the Modeling of Wall Bounded Flows. Imperial College of Science, Technology and Medicine.

Di Felice, R., 1995. Hydrodynamics of liquid fluidisation. Chem. Eng. Sci. 50, 1213–1245.

Eichlseder, H., 2008. Thermodynamik Studienblätter (Thermodynamics Study Sheets).

Ergun, S., 1952. Fluid Flow Through Packed Columns. Chem. Eng. Prog. 48, 89–94.

Geldart, D., 1973. Types of Gas Fluidization. Powder Technol. 7, 285–292.

Gidaspow, D., 1994. Multiphase Flow and Fluidization: Continuum and Kinetic Theory Descriptions. Academic Press, California.

Goldschmidt, M.J. V, Link, J.M., Mellema, S., Kuipers, J. a M., 2003. Digital image analysis measurements of bed expansion and segregation dynamics in dense gas-fluidised beds. Powder Technol. 138, 135–159. doi:10.1016/j.powtec.2003.09.003

González, B.C., 2013. Advanced Drag Models for Bi-Disperse Bubbling Fluidized Beds. Graz University of Technology.

Gronald, G., 2014a. RE: DA Wirbelschicht Polydisperse. Personal email dated 03/05/2014.

Gronald, G., 2014b. DA Fragen. Personal email dated 06/30/2014.

Hirsch, C., 1988. Numerical Calculation of Internal and External Flows. Volume 1: Fundamentals of Numerical Discretization.

Holloway, W., Benyahia, S., Hrenya, C.M., Sundaresan, S., 2011. Meso-scale structures of bidisperse mixtures of particles fluidized by a gas. Chem. Eng. Sci. 66, 4403–4420. doi:10.1016/j.ces.2011.05.037

Holloway, W., Sundaresan, S., 2014. Filtered models for bidisperse gas-particle flows. Chem. Eng. Sci. 108, 67–86. doi:10.1016/j.ces.2013.12.037

Holloway, W., Yin, X., Sundaresan, S., 2010. Fluid-Particle Drag in Inertial Polydisperse Gas – Solid Suspensions. AIChE J. 56, 1995–2004. doi:10.1002/aic

Holzinger, F., 2014a. Minutes of Meeting dated 08/28/2014.

Holzinger, F., 2014b. Minutes of Meeting dated 05/05/2014.

Holzinger, F., 2014c. Minutes of meeting dated 06/30/2014.

Igci, Y., Iv, A.T.A., Sundaresan, S., Brien, T.O., 2008. Filtered Two-Fluid Models for Fluidized Gas-Particle Suspensions. AIChE J. 54, 1431–1448. doi:10.1002/aic

Kelley, G.E., Moore, K.K., 1944. Specific Heats at Low Temperatures of Calcium Sulfite, Sodium Sulfite, and Manganese Dithionate Dihydrate. J. Am. Chem. Soc. 66, 293–295.

Khinast, J., Brenn, G., Pflügl, M., 2009. Lecture notes mass transfer. Institute of Process and Particle Engineering, Graz University of Technology.

Luding, S., 2008. Cohesive, frictional powders: contact models for tension. Granul. Matter 10, 235–246. doi:10.1007/s10035-008-0099-x

Milioli, C., Milioli, F., Holloway, W., 2013. Filtered two-fluid models of fluidized gas-particle flows: New constitutive relations. AIChE J. 59, 3265–3275. doi:10.1002/aic

Moradnia, P., 2010. CFD of Air Flow in Hydro Power Generators. Chalmers University of Technology.

N.N. (DDBST), 2015. Vapor Pressure Calculation by Antoine Equation [WWW Document]. URL http://ddbonline.ddbst.de/AntoineCalculation/AntoineCalculationCGI.exe?component=Water (accessed 2.24.15).

N.N. (OpenFOAM Foundation), 2015. OpenFOAM® User Guide v2.2.1 [WWW Document]. URL http://www.openfoam.org/archive/2.2.1/docs/user/ (accessed 4.2.15).

N.N. (VDI), 2006. D Stoffeigenschaften (material properties), in: VDI Wärmeatlas. Springer.

Nellis, G., Klein, S., 2008. Mass Transfer, in: Heat Transfer. Cambridge University Press, pp. 1–64. doi:10.1016/S0920-4083(99)80010-3

O'Rourke, P.J., Snider, D.M., 2012. Inclusion of collisional return-to-isotropy in the MP-PIC method. Chem. Eng. Sci. 80, 39–54. doi:10.1016/j.ces.2012.05.047

Ozel, A., Fede, P., Simonin, O., 2013. Development of filtered Euler–Euler two-phase model for circulating fluidised bed: High resolution simulation, formulation and a priori analyses. Int. J. Multiph. Flow 55, 43–63. doi:10.1016/j.ijmultiphaseflow.2013.04.002

Parmentier, J.-F., Simonin, O., Delsart, O., 2012. A functional subgrid drift velocity model for filtered drag prediction in dense fluidized bed. AIChE J. 58, 1084–1098. doi:10.1002/aic.12647

Patankar, N., Joseph, D., 2001. Modeling and numerical simulation of particulate flows by the Eulerian–Lagrangian approach. Int. J. Multiph. Flow 27, 1659–1684.

Penttinen, O., 2011. A pimpleFoam tutorial for channel flow, with respect to different LES models. Chalmers University of Technology.

Pirker, S., Kahrimanovic, D., Goniva, C., 2011. Improving the applicability of discrete phase simulations by smoothening their exchange fields. Appl. Math. Model. 35, 2479–2488. doi:10.1016/j.apm.2010.11.066

Pope, S.B., 2000. Turbulent Flows. Cambridge University Press. doi:10.1017/CBO9780511840531

Radl, S., 2015. RE: abstimmung quench zone. Personal email dated 04/17/2015.

Radl, S., Girardi, M., Sundaresan, S., 2012. Effective Drag for Parcel-Based Approaches - what can we learn from CFD-DEM ?, in: Europen Congress on Computational Methods in Applied Sciences and Engineering. Vienna.

Radl, S., Gonzales, B.C., Goniva, C., Pirker, S., 2014. State of the Art in Mapping Schemes for dilute and dense Euler-Lagrange Simulations, in: 10th International Conference on CFD in Oil & Gas, Metallurgical and Process Industries. pp. 226–234.

Radl, S., Khinast, J.G., 2010. Multiphase Flow and Mixing in Dilute Bubble Swarms. AIChE J. 56, 2421–2445. doi:10.1002/aic.12154

Radl, S., Radeke, C., Khinast, J.G., Sundaresan, S., 2011. Parcel-Based Approach for the Simulation of Gas-Particle Flows, in: CFD in Oil & Gas, Metallurgical and Process Industries. pp. 124/1–124/10.

Radl, S., Sundaresan, S., 2014. A drag model for filtered Euler-Lagrange simulations of clustered gas-particle suspensions. Chem. Eng. Sci. 117, 416–425. doi:10.1016/j.ces.2014.07.011

Radl, S., Sundaresan, S., 2013. Coarse-Grid Simulations Using Parcels: An Advanced Drag Model based on Filtered CFD-DEM Data, in: The 14th International Conference on Fluidization – From Fundamentals to Products. ECI Symposium Series.

Reissner, H.K., Brunner, C., Bärnthaler, K., 2003. TURBOSORP - Emission limits after 17th BimSch V (German Federal Imission Act) at lowest costs in a simple dry process - Comparison of dry / semi dry processes and results of mercury and dioxin separation in a one step process, in: 11th North American Waste to Energy Conference. pp. 65–72.

Snider, D.M., Rourke, P.J.O., Andrews, M.J., 1998. Sediment flow in inclined vessels calculated using a multiphase particle-in-cell model for dense particle flows. Int. J. Multiph. Flow 24, 1359–1382.

Van der Hoef, M. a., Beetstra, R., Kuipers, J. a. M., 2005. Lattice-Boltzmann simulations of low-Reynolds-number flow past mono- and bidisperse arrays of spheres: results for the permeability and drag force. J. Fluid Mech. 528, 233–254. doi:10.1017/S0022112004003295

Versteeg, H.K., Malalasekera, W., 2007. An Introduction to Computational Fluid Dynamics: The Finite Volume Method, 2nd ed. Pearson Education Limited, Essex, England.

Wen, C.Y., Yu, Y.H., 1966. Mechanics of fluidization. Chem. Eng. Progress, Symp. Ser. 62, 100–111.

Yin, X., Sundaresan, S., 2009. Fluid-Particle Drag in Low-Reynolds-Number Polydisperse Gas – Solid Suspensions. AIChE J. 55, 1352–1368. doi:10.1002/aic

Zhu, H.P., Zhou, Z.Y., Yang, R.Y., Yu, a. B., 2007. Discrete particle simulation of particulate systems: Theoretical developments. Chem. Eng. Sci. 62, 3378–3396. doi:10.1016/j.ces.2006.12.089

9 Appendix

9.1 3D Model Dimensional Drawing

The drawing on the next page is a printout of the CAD drawing file "Modell_Kentucky_rev2_print.dwg" which refers to the standard ACIS text file "Modell_Kentucky_rev2.sat" obtained from the industrial partner. Some minor changes have been applied to the parts and assembly files (generated during import), e.g. merging the parts such that edges on plain joined faces of connected parts vanish. Hence the additional files are also provided on the attached disc.

There are a few but important details. One should consider that the nozzles project a bit into the cone-shaped section of the riser exterior and are modeled as separate parts not well connected to the riser exterior parts assembly. Hence the origin is defined at the center of the centered nozzle outlet (the exterior shell alone cannot reflect that).

The drawing is defined for A0 paper size format hence some details on A4 paper size may need more resolution. This can be achieved by opening the drawing file in a viewer.


8128,00
53315,31
5562,60
A
A
1675,44
683,07
3824,32
12359,17
5461,70
3986,81
2392,77
2801,82
304,80
320,13
□8128,00
5274,24
1371,60
783,66
6400,80
14478,00
9956,80
4267,20
419,10
304,80
Ø9144,00
20720,45
4944,00
Ø1682,75
1201,74
1905,00
Ø1270,00
136,53
269,88
Ø1412,88
Ø1682,75
7368,25
Ø5562,60
7556,50
□4368,80
10337,80
564,42
177,40
224,08
Ø1035,05
A-A (1 : 100)
B
7384,51
650,88
1066,80
1066,80
1522,68
610,53
6,35
610,53
B (1 : 5)
Ø1682,75
13,63
7,94
532,88
C
60°
532,88
C
C-C (1 : 100)
748,77
5627,85
748,77
4225,30
83°
83°
6071,98
3830,00
dimension drawing
M1:100
Modell_Kentucky_rev2

9.2 Cases

9.2.1 File Structure

Files representing a case are organized for use with OpenFOAM® by following an immanent folder structure. Its set-up content is listed in Table 9.1 to Table 9.3; after finishing a simulation including manual post-processing the case consists of more files as listed in Table 9.4 to Table 9.9. '#' indicates that the actual content is a subunit of the above folder. Cases are provided in detail on the attached disc. Geometry, mesh and ParaView state files (*.stl, *.msh, *.pvsm) are described in root folder but may reside elsewhere.

Table 9.1 Case contents at setup. Page 1 of 3.

Filename / Foldername	Description
Allrun(Par)(.sh)	Script to run the simulation (in parallel on more processors)
cleanCase(.sh), clearRun(.sh)	Script to clean the case after simulation
startCFDEM	Script for use on the dcluster of the TU Graz Usage: qsub startCFDEM
probeProc(.sh), postProc(.sh)	Scripts for post-processing, postProc(.sh) most recent
importMesh(.sh), parCFDDEMrun(.sh), …	More scripts
*.gnuplot	Scripts for use with gnuplot for various plots Usage: gnuplot file
*.msh	ANSYS® Fluent® mesh file
*.stl	Geometry file defining a surface by triangles, created via OpenFOAM® commands or CAD software
*.pvsm	Saved Paraview configuration states for quick reproduction of figures created using Paraview

Table 9.2 Case contents at setup. Page 2 of 3.

Filename / Foldername	Description
CFD	Folder consisting of CFD-specific files
#m.foam	Visualizing a case in Paraview, open/select this file Creation: touch m.foam
#runMe(Dcluster).sh	Most recent scripts to run the simulation
#0, 0.org	CFD folder consisting of initial values
##epsilon, f(Smooth), k, Ksl, nuSgs, p, quenchT, rho, sSmoothField, U, Us, voidfraction, vSmoothField, …	One file per variable providing initial data
#constant	CFD folder consisting of constant-kept configuration, parameters and properties
##couplingProperties	CFDEM® configuration, parameters and properties
##g	Acceleration of gravity
##liggghtsCommands	Configuration of CFDEM® communication with LIGGGHTS®
##scalarTransportProperties	Configuration of transport equations for scalar field quantities e.g. heat that
##turbulenceProperties, LESProperties, RASProperties	Configuration of turbulence
##transportProperties	Transport properties, e.g. viscosity
##polyMesh	CFD folder consisting of the mesh
###blockMeshDict	Initial configuration of mesh
(###boundary, faces, neighbor, owner, points, …)	Already created mesh

Table 9.3 Case contents at setup. Page 3 of 3.

Filename / Foldername	Description
#system	CFD folder consisting of configuration and parameter files
##controlDict	Main CFD configuration file
##createPatchDict	Patches configuration. Clear or create some based on sets or patches
##decomposeParDict	Configuration of data split for parallel processing
##fvSchemes, fvSolution, fvOptions	Discretization schemes, solvers and options configuration for finite volume solved applications
##quenchAverages, quenchFunctionObject	Configuration of quenching model and of its monitoring and post-processing
##sampleDict	Configuration of 2D profile sets along geometric lines
##sliceData	Configuration of 2D profiles via cutting planes
##topoSetDict	Configuration of topographic subsets
DEM	Folder consisting of DPM-specific files
#in.liggghts_init, in.jet	DPM configuration
#particlePreparation	If system is dense ($\langle \varphi_P \rangle$>$10^{-2}$), DPM folder of a preparation case filling particles into the box which will be read in for sedimentation
##in.liggghts_init	DPM configuration
##post	Folder for DPM-specific data
#post	Folder for DPM-specific data

Table 9.4 Additional contents of post-processed case. Page 1 of 6.

Filename / Foldername	Description
startCFDEM.e*, startCFDEM.o*	Error and output log files of a run on dcluster
*.pdf	Various plots created by MATLAB and gnuplot for documentation purpose
*.png	Snapshots created using ParaView for documentation purpose
*.res, *.dat	Data condensed by MATLAB and shell scripts
CFD	Folder consisting of CFD-specific files
#jetRestart.*, liggghts.restart.*	Created during run. In case of a crash, LIGGGHTS® will restart again using one of these states
#log.*	Log files of a run
#m.foam	Viewing a case in Paraview, open/select this file Creation: touch m.foam
#mean.dat	Log file for total particle mass and domain-averaged particle velocity opposed to sedimentation direction for each CFD time step
#reportOverlap.dat	Log file for total particle overlap (in %) for each CFD time step
#0-	CFD folders consisting of combined evaluated data of all processors for each written time step

Table 9.5 Additional contents of post-processed case. Page 2 of 6.

Filename / Foldername	Description
##ddtVoidfraction, dSauter, expParticleForces, epsilon, f(Error |Next |Prev |Smooth), impParticleForces, k(Mean |Prime2Mean), Ksl(Next |Prev), nuSgs(Mean), nut, p, phi, phi_0, phiP1, phiP2, quenchEvapRate, quenchMuLiq(Mean), quenchMuVap(Mean), quenchrhoSat, quenchT(Mean), rho, sourceField, U(Mean |Prime2Mean), U_0, uP1, uP2, Us(Next |Prev), UsWeightField_, voidfraction(Mean |Next |Prev |Prime2Mean), voidfraction_0, …	One file per evaluated variable providing evaluated data
##lagrangian	CFD folder consisting of DPM data
###particleCloud	CFD folder consisting of particle data
####positions, r, v	One file per evaluated variable providing evaluated data
##uniform	CFD folder consisting of cell- and particle-independent data
###fieldAveragingProperties, momentumSource*Properties, time	One file per evaluated variable providing evaluated data
#averageProps	CFD folder consisting of domain-averaged data for each CFD time step…
##0	… starting with time step 0

Table 9.6 Additional contents of post-processed case. Page 3 of 6.

Filename / Foldername	Description
###integralMomentum	Log file for domain-averaged reference momentum and domain-averaged integral momentum. Definition see case setup (chapter 3.4)
###uSlip	Log file for domain-averaged particle volume fraction and domain-averaged slip velocity according to Eqn. (3.1) and Eqn.(3.5) respectively
###velStats	Log file for domain-averaged fluid velocity and domain-averaged particle velocity
#constant	CFD folder consisting of constant-kept configuration, parameters and properties
##polyMesh	CFD folder consisting of the mesh
###boundary, faces, neighbor, owner, points, …	Resulting mesh
###sets	CFD folder consisting of pointsets, facesets, cellsets, e.g. summarizing erroneous points, faces, cells or created by topoSet
#particleProbes	CFD folder consisting of individual particles data per force model probed…
##0	… starting with time step 0 …
###beetstraDrag.logDat.*, gradP.logDat.*, visc.logDat.*	… logging at each configurable time step. One file per processor.
#postProcessing	CFD folder consisting of post-processed cell data
##fieldMinMax*	Minimum and maximum of specified field …
###0	… starting with time step 0

Table 9.7 Additional contents of post-processed case. Page 4 of 6.

Filename / Foldername	Description
####fieldMinMax.dat	Value and location of minimum and maximum for each configurable time step
##outletAverage_1	Average fluid velocity (and/or turbulent energy, sub-grid-scale viscosity, void-fraction) over outlet surface …
###0	… starting with time step 0 …
####faceSource.dat	… for each configurable time step
##outletAverage_quench	Flux-averaged temperature and water mass loadings over outlet surface …
###0	… starting with time step 0 …
####faceSource.dat	… for each configurable time step
##probes	CFD folder consisting of probed data
###0	First folder consists of one file per normal variable probed
####k, p, quenchT, U	Probed data at each location for each configurable time step
###1	Second folder consists of one file per mean variable probed
####kMean, quenchMuLiqMean, quenchMuVapMean, quenchTMean, UMean, voidfractionMean	Probed data at each location for each configurable time step
#### kMean.txt, nuSgsMean.txt, TMean.txt, Uxyz.txt, voidfractionMean.txt	Probed data without header and parenthesis ready for import as delimited file consisting only of numbers

Table 9.8 Additional contents of post-processed case. Page 5 of 6.

Filename / Foldername	Description
##sliceCenter*	DPM folder holding for a centered cutting plane …
###*vtk	.. data files of fields ready to visualize in ParaView for each quantity and time step
##sets	CFD folder consisting of line sets for 2D profile plots for each writing time step …
###0	… starting with time step 0
####*.xy	Raw line set ready for plotting the profile along the line by gnuplot
##volAvU	Volume-averaged fluid velocity, weigthed by voidfraction …
###0	… starting with time step 0 …
####cellSource.dat	… for each configurable time step
#processor*	CFD folder consisting of evaluated data of one processor…
##0-	… for each written time step (contents see #0-)
##constant	… operating with this fraction of constant values taken from CFD/constant
###polyMesh	CFD folder consisting of the mesh
####*ProcAddressing	Cells, boundaries, faces and points are reassigned in split space so these files contain both addresses
DEM	Folder consisting of DPM-specific files
#cleanScript(.sh)	Script to clean the DPM part after simulation
#liggghts.restartCFDEM	Created during run. In case of a crash, LIGGGHTS® will restart again using this state

Table 9.9 Additional contents of post-processed case. Page 6 of 6.

Filename / Foldername	Description
#massoutlet(Bottom).dat.*	Cumulatively tracked particles penetrating through (bottom or) outlet per processor
#particlePreparation	DPM folder of a preparation case filling particles into the box on dense systems ($\langle \varphi_P \rangle > 10^{-2}$) which will be read in for sedimentation
##post	DEM folder consisting of processed data
###CFDEMdump*.liggghts	Dump providing data of filled in particles. Format ready for copy to particle.data for use in CFD
###dump*.liggghts	Dump providing more data for each filled in particle.
###liggghts*(_boundingBox).vtk	VTK files manually created from particle data files using lpp with dump files herein. Can be read by ParaView to display particle data graphically
###particle.data	Data of filled in particles for use in CFD
#post	DEM folder consisting of processed data
##dump*.part	One file per dump time step. Provides most of the data for each particle
#VTK	Folder consisting of …
##*.vtk	… mesh subsets created by topoSet converted by foamToVTK to files ready for visualization in ParaView

The following Table 9.10 and Table 9.12 list alphabetically which parameters have been changed for the various cases and where they can be found. Some physical properties might change in future adaptations, which are enlisted too. Initial conditions describing files are excluded. The void-fraction will be overwritten by the complement of the particle volume fraction, from DPM mapped to CFD. DEM/in.jet replaces DEM/in.liggghts_init in 2D and 3D cases of the riser with particle injection using the same parameters except the particle volume fraction. The particle volume fraction is replaced by the particle injection mass rate. All values are given in SI units.

Table 9.10 Parameter location. Page 1 of 3.

Parameter	Location
CFD time step Δt_{CFD}	CFD/system/controlDict, DEM/in.liggghts_init
Coarse-graining ratio α	DEM/in.liggghts_init, DEM/particlePreparation/in.liggghts_init
Coupling factor CF	CFD/constant/couplingDict
Courant number Co	CFD/system/controlDict
DEM time step Δt_{DEM}	DEM/in.liggghts_init, DEM/particlePreparation/in.liggghts_init
Divergence schemes	CFD/system/fvSchemes
Domain size l_{domain}	CFD/constant/polyMesh/blockMeshDict, DEM/in.liggghts_init, DEM/particlePreparation/in.liggghts_init
Drag model	CFD/constant/couplingDict
Drag correction factor f_{corr}	CFD/constant/couplingDict
Fluid density ρ_{f}	CFD/system/quenchFunctionObject
Fluid kinematic viscosity ν_{f}	CFD/constant/transportProperties
(Mean) gas heat capacity $\overline{c_{p,\mathrm{f}}}\Big\|_{T_{\mathrm{CZ}}}^{T_{\mathrm{FG}}}$	CFD/system/quenchFunctionObject

Table 9.11 Parameter location. Page 2 of 3.

Parameter	Location
Grid resolution $l_{\text{domain}} / \Delta x$	CFD/constant/polyMesh/blockMeshDict
Heat of evaporation $\Delta h_{\text{W,vap}} \|_{T_0}$ (deltaHEvap)	CFD/system/quenchFunctionObject
Injection velocities (U_{X} U_{Y} U_{Z}), recirculate / quenching water	CFD/system/fvOptions
Injection volumes V_{inject}, position and dimensions, recirculate / quenching water	CFD/system/topoSetDict
Number of parallel processors	Allrun(Par)(.sh), CFD/system/decomposePar, parCFDDEMrun(.sh), startCFDEM
Particle density ρ_{P}	CFD/constant/couplingDict, CFD/system/quenchFunctionObject, DEM/in.liggghts_init, DEM/particlePreparation/in.liggghts_init
(Mean) particle heat capacity $\left(\overline{c_{p,\text{P}}} \|_{T_{\text{P}}}^{T_{\text{CZ}}}\right)_{\text{est}}$	CFD/system/quenchFunctionObject
Particle (injection mass) rate $\dot{M}_{\text{P}}$	DEM/in.jet
Particle volume fraction $\langle \varphi_{\text{P}} \rangle$	CFD/0/voidfraction as complement, DEM/in.liggghts_init, DEM/particlePreparation/in.liggghts_init
Particle size distribution $d_{\text{m},i}$, $\Delta Q_{3,i}$	DEM/in.liggghts_init, DEM/particlePreparation/in.liggghts_init
Primary (Sauter mean) diameter $\langle d \rangle$	CFD/constant/couplingDict

Table 9.12 Parameter location. Page 3 of 3.

Parameter	Location
Turbulent Schmidt number Sc_t	CFD/system/quenchFunctionObject
Simulation time span t_sim	CFD/system/controlDict
Solver	CFD/system/controlDict
Smoothing length l_smooth	CFD/constant/couplingDict
Time derivation schemes	CFD/system/fvSchemes
Turbulence model	CFD/constant/couplingDict, CFD/constant/turbulenceProperties, CFD/constant/RASProperties, CFD/constant/LESProperties,
t_vap (tEvap)	CFD/system/quenchFunctionObject
Wall-collision particle scale-up factor	DEM/in.liggghts_init
Water densities, liquid $\rho_\mathrm{W,l}\|_{T_\mathrm{W}}$ and vapor (saturated) $\rho_\mathrm{W,vap}\|_{T_\mathrm{CZ}}$	CFD/system/quenchFunctionObject
Water heat capacities, liquid $c_{p,\mathrm{W,l}}$ and vapor $c_{p,\mathrm{W,vap}}$	CFD/system/quenchFunctionObject
Water (injection mass) rate $\dot{M}_\mathrm{W}$ (absolute quenchMuLiq)	CFD/system/quenchFunctionObject
Young's modulus E	DEM/in.liggghts_init, DEM/particlePreparation/in.liggghts_init

9.2.2 Investigation of Drag Models

The following printed file contents consist of code which has not been discussed in detail in the setup of cases with periodic boundaries and are listed in alphabetical order. All correspond to the case "sedimentation2_phiP_0.001_PSDAEEsim_domain_4.2x4.2x4.2mm_grid_30x30x30_implicitMapping_Beetstra_cg_10_smooth_2dPmax_tspan_250tref_CFDEM140613" except the particle preparing file, which refers to "sedimentation_phiP_0.020_PSDmonoAEEsim_domain_4.2x4.2x4.2mm_grid_30x30x30_implicitMapping_Beetstra_couplingDict_smooth_CFDEM140613/DEM/particlePreparation/in.liggghts_init". ParaView state files (*.pvsm) are not printed. Post-processed case directories contents can be viewed on the attached disk.

CFD/constant/couplingProperties

CFD/system/controlDict

CFD/system/fvSchemes

CFD/system/fvSolution

DEM/in.liggghts_init

DEM/particlePreparation/in.liggghts_init (for $\langle\varphi_P\rangle > 10^{-2}$)

For collision tracking, in DEM/in.liggghts_init the first line has to be replaced by the second one.

1. neigh_modify exclude type 1 1 #do not perform collision tracking for situations with phiP < 1 Vol%!
2. neigh_modify delay 0 one 1000

In dense particulate suspensions, i.e. $\langle \varphi_P \rangle > 10^{-2}$, the particles are filled into the box in a preparation case before sedimentation. Hence in the file DEM/in.liggghts_init the first line has to be replaced by the second line and the sections labeled "#particle distribution" and "#option 1 for insertion" have to be deleted.

1. create_box 1 reg
2. read_data ../DEM/particlePreparation/post/particle.data

Instead “liggghts <in.liggghts_init” has to be run in the DEM/particlePreparation directory before the case main script, e.g. startCFDEM.

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.4                                   |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/


FoamFile
{
    version         2.0;
    format          ascii;

    root            "";
    case            "";
    instance        "";
    local           "";

    class           dictionary;
    object          couplingProperties;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//===========================================================================//
// sub-models & settings
solveFluidFlow  true;
imExSplitFactor 1.0;    //implicit forces will be considered implicitly (1) or explicitly (0)
treatVoidCellsAsExplicitForce true;

modelType A; // A or B

couplingInterval 1000;

//skipBiDisperseUpdates; //will skip update of quantities specific for bi-disperse clouds

clockModel off; //standardClock;

locateModel engine;//standard;//

meshMotionModel noMeshMotion;

regionModel allRegion;

IOModel basicIO;

probeModel particleProbe;

dataExchangeModel twoWayMPI;//twoWayFiles;//oneWayVTK;//
```

```

voidFractionModel dividedBiDi; //weightedNeighbor;//centre;//bigParticle;//

averagingModel denseBiDi; //dense; //dilute;//

smoothingModel constDiffSmoothing; //relevant in case of particles in the range of the cell

forceModels
(
    //GidaspowDrag
    //DiFeliceDrag
    //Archimedes
    //SchillerNaumannDrag
    //KochHillDrag
    //MeiLift
    BeetstraDrag
    //HollowayDrag
    //virtualMassForce
    gradPForce
    viscForce
    //solidsPressureForce
    periodicPressure //BC
    averageSlipVel   //postProc
);

momCoupleModels
(
    implicitCouple
    explicitCoupleSource  // enables setSourceField to superpose an additional source momentum to explicitCouple
);

turbulenceModelType RASProperties;//LESProperties;//

//===========================================================================//
// sub-model properties

engineProps
{
    treeSearch true;
}

particleProbeProps
{
    particleIDsToSample (0 2 4);
    verboseToFile;  //main switch
//    verbose;  //currently not used
    printEvery  1;      //print every this many CFD time steps
//    sampleAll;          //Activate sampling for all particles

    probeDebug;  //probes additional fields
    includePosition;  //will include particle position in the output file
    writePrecision 4;           //number of significant digits to print
}

dividedProps            //dividedProps instead of dividedBiDiProps needed for voidFractionModel dividedBiDi
{
    alphaMin 0.3;       //minimum limit for voidfraction
```

```
    //interpolation;    //interpolate voidfraction to particle positions (normally off)
    weight 1.0;         //occupied in CFD domain: Vparticle=dsphere^3*pi/6*weight
    porosity 1.0;       //similar to scaleUpVol, diameter artificially increased by Vparticle*scaleUpVol, volume unaltered
}

constDiffSmoothingProps
{
    lowerLimit 0.0;
    upperLimit 1e99;
    smoothingLength 7.e-4; //smoothingLength 10*10*35e-6*0.2=CG*phiP^(-1/3)*dPrimMax*0.2
//    verbose;
}

twoWayMPIProps
{
    maxNumberOfParticles 10100;
    liggghtsPath "../DEM/in.liggghts_init"; //resume";
}

GidaspowDragProps
{
    velFieldName "U";
    granVelFieldName "Us";
    densityFieldName "rho";
    voidfractionFieldName "voidfraction";
    phi 1;
}

DiFeliceDragProps
{
    //verbose ;
    interpolation;
    splitImplicitExplicit;
    velFieldName "U";
    granVelFieldName "Us";
    densityFieldName "rho";
    voidfractionFieldName "voidfraction";
}

ArchimedesProps
{
    densityFieldName "rho";
    gravityFieldName "g";
    treatDEM;
}

SchillerNaumannDragProps
{
    velFieldName "U";
    densityFieldName "rho";
}

KochHillDragProps
{
    velFieldName "U";
    densityFieldName "rho";
    rhoParticle 2250;
```

```
    voidfractionFieldName "voidfraction";
    interpolation ;
}

MeiLiftProps
{
    velFieldName "U";
    densityFieldName "rho";
}

BeetstraDragProps
{
//    verbose ;
    velFieldName "U";
    granVelFieldName "Us";
    densityFieldName "rho";
    gravityFieldName "g";
//    dPrim           75e-6;          // only used in octave postproc!!!
    voidfractionFieldName "voidfraction";
    interpolation ;
//    useFilteredDragModel ;
//    useParcelSizeDependentFilteredDrag ; //and forces switch useFilteredDragModel to "on"
    rhoParticle    2250.;
    dPrim           6.7968e-6;
/*    k       0.05;*/
/*    aLimit  0.0;*/
/*    aExponent   1.0;*/
    splitImplicitExplicit;
}

HollowayDragProps
{
//    verbose;
//    verboseToFile;
//    useFluidMediatedDrag;
    velFieldName "U";
    densityFieldName "rho";
    voidfractionFieldName "voidfraction";
    gravityFieldName "g";
    interpolation ;
//    interpolationParticleAverages ;
    UpFieldName1 "uP1";
    UpFieldName2 "uP2";
    dSauterFieldName "dSauter";
    phiP1FieldName  "phiP1";
    phiP2FieldName  "phiP2";
    lambda            1e-6;
    voidfractionLimit 1;
//    useFilteredDragModel ;
//    useParcelSizeDependentFilteredDrag; //and forces switch useFilteredDragModel to "on"
    rhoParticle    2250.;
    dPrim           6.7968e-6;
/* only relevant if useParcelSizeDependentFilteredDrag is on */
/*    k                   0.05; */
/*    aLimit            0.0; */
/*    aExponent      1.0; */
//    treatExplicit ; //Switch to activate explicit mapping
}

```

```
virtualMassForceProps
{
    velFieldName "U";
    densityFieldName "rho";
}

gradPForceProps
{
    pFieldName "p";
    densityFieldName "rho";
    voidfractionFieldName "voidfraction";
    velocityFieldName "U";
    interpolation;
}

viscForceProps
{
    velocityFieldName "U";
    densityFieldName "rho";
    interpolation;
}

solidsPressureForceProps
{
    verbose;
    rhoParticle 2250.;
    pStar 0;
    exponent 2;
    volumefractionSwitchOff 0.58;
    volumefractionMax 0.6;
    voidfractionFieldName "voidfraction";
    //interpolation;
}

periodicPressureProps
{
    rhoParticle    2250.;
    gravityFieldName "g";
    rhoFluidName "rho";
    fluidVelFieldName "U";
    particleVelFieldName "Us";
    voidfractionFieldName "voidfractionNext";
    mode "controlled";
    referenceMomentum_x 0.0;
    referenceMomentum_y 0.0;
    referenceMomentum_z 0.0;
    momentumCorrFactor 1e3;
    //verbose ;
}

averageSlipVelProps
{
    rhoParticle             2250.;
    outputDirName           "averageProps";
    fluidVelFieldName       "U";
    particleVelFieldName    "Us";
    voidfractionFieldName   "voidfraction";
    rhoFluidName "rho";
}

implicitCoupleProps
```

```
{
    velFieldName "U";
    granVelFieldName "Us";
    voidfractionFieldName "voidfraction";
}

explicitCoupleProps     //explicitCoupleProps instead of
explicitCoupleSourceProps needed for momCoupleModel explicitCoupleSource
{
    //fLimit (0 0 0);
}

//
************************************************************************
//

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pisoFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.07;    //7e-3; /*0.1;*/

deltaT          1e-4;    /*1e-4;*/

writeControl    adjustableRunTime;

writeInterval   0.01;    //1e-3; /*0.01;*/

purgeWrite      5;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  no;

maxCo 0.1;

//libs ( "libgroovyBC.so" );

```

```
functions
(

/*    probes
    {
        type        probes;
        // Where to load it from
        functionObjectLibs ( "libsampling.so" );
        // Name of the directory for probe data
        name        probes;
        probeLocations
        (
            (0 0 0)
        );

        // Fields to be probed
        fields ( avUslipX avUslipY avUslipZ );

        // Write at same frequency as fields
        outputControl   timeStep;//outputTime;
        outputInterval  1;
    }
/*/

    volAvU
    {
        type            cellSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   outputTime;//timeStep;//
        log             false;
        valueOutput     false;
        source          all;
        operation       weightedAverage;
        weightField     "voidfraction";
        fields
        (
            U
        );
    }

   /*pressureDrop
   {
     type patchAverage;
     functionObjectLibs
     (
         "libsimpleFunctionObjects.so"
     );
     verbose true;
     patches
     (
         inlet
         outlet
     );
     fields
     (
         p
     );
     factor  1;
   }*/
);
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         backward;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         Gauss linear;
    div(phi,U)      Gauss limitedLinearV 1;
    div(phi,k)      Gauss limitedLinear 1;
    div(phi,epsilon) Gauss limitedLinear 1;
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((viscousTerm*dev(grad(U).T()))) Gauss linear;
    div((nu*dev(grad(U).T()))) Gauss linear;
    div((nuEff*dev(grad(U).T()))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
    laplacian(viscousTerm,U) Gauss linear corrected;
    laplacian(nu,U) Gauss linear corrected;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian((voidfraction2|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
```

```
    laplacian(DREff,R) Gauss linear corrected;
    laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-9;
        relTol          1e-04;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }

    pFinal
    {
        solver          GAMG;
        tolerance       1e-10;
        relTol          0;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }

    U
    {
        solver          PCG;    // PBiCG;
        preconditioner  DIC;    // DILU;
```

```
        tolerance       1e-05;
        relTol          0;
    }

    UNext
    {
        $U
    }

    UsNext
    {
        $U
    }

    UFinal
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0;
    }

    k
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       5e-06;
        relTol          0;
    }

    epsilon
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0;
    }

    R
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0;
    }

    nuTilda
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0;
    }

    voidfraction
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-09;
        relTol           1e-06;
    }
```

```

    voidfractionNext
    {
        $voidfraction
    }

    Ksl
    {
        $voidfraction
    }

    f
    {
        $voidfraction
    }

    fSmooth
    {
        $voidfraction
    }

    phiP1
    {
        $voidfraction
    }

    phiP2
    {
        $voidfraction
    }

    uP1
    {
        $voidfraction
    }

    uP2
    {
        $voidfraction
    }

    Us
    {
        $voidfraction
    }

    dSmoothing
    {
        $voidfraction
    }
}

PIMPLE
{
    nOuterCorrectors 1;
}
relaxationFactors
{
    fields
    {
            p   1.0;
```

```
    }
    equations
    {
        "U.*"           1.0;
        "k.*"           1.;
        "epsilon.*"     1.;
    }
}

PISO
{
    nCorrectors     3;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}


// ************************************************************************* //
```

```
// ************************************************************************* //

```

```
### variables declaration ###
variable phiP            equal   1e-3
variable rhoP            equal 2250
variable youngsModulus   equal 5.e7
variable poissonsRatio   equal 0.45
variable coeR            equal 0.9
variable coeF            equal 0.5
variable timeStepDEM     equal 1e-7
variable timeStepCFD     equal 1e-4      #must result in a time step multiplicator that is an integer
variable timeSpan        equal 0.07      #must result in time step multiplicators that are integers
variable Dumps           equal 5 #must result in time step multiplicators that are integers
variable coarseGrainingRatio equal 10. # coarse graining ratio = computational parcel size / original particle size . wird in Berechnungen verwendet. Daher 1 setzen wenn coarsegraining inaktiv

## Particle Size Distribution
variable d1 equal  5e-6
variable d2 equal  12.5e-6
variable d3 equal  17.5e-6
variable d4 equal  22.5e-6
variable d5 equal  27.5e-6
variable d6 equal  35.0e-6
variable dmax equal  35.0e-6
variable vfrac1 equal 0.62 #volume fraction of particles
variable vfrac2 equal 0.16
variable vfrac3 equal 0.10
variable vfrac4 equal 0.06
variable vfrac5 equal 0.03
variable vfrac6 equal 0.03
## Domain
variable boxSize         equal 4.2e-3 # region volume !>= 1e-10
variable sizeOrigin      equal 0.

## INPUT CALCULATIONS ##
variable rad1 equal      ${d1}/2.
variable rad2 equal      ${d2}/2.
variable rad3 equal      ${d3}/2.
variable rad4 equal      ${d4}/2.
variable rad5 equal      ${d5}/2.
variable rad6 equal      ${d6}/2.
variable neighborDist    equal 1e-3*${coarseGrainingRatio}*${dmax}
variable timeStepMultiplicatorCFD       equal ${timeStepCFD}/${timeStepDEM}
variable timeStepMultiplicatorSpan      equal ${timeSpan}/${timeStepDEM}
variable timeStepMultiplicatorDump      equal ${timeSpan}/(${Dumps}*${timeStepDEM})
variable timeStepMultiplicatorPrint     equal ${timeStepMultiplicatorCFD}
####################################

coarsegraining      ${coarseGrainingRatio}

atom_style               granular
atom_modify              map array
communicate              single vel yes

boundary         p p p
newton           off

units            si
```

```
processors      * * *

region          reg block ${sizeOrigin} ${boxSize} ${sizeOrigin} ${boxSize} ${sizeOrigin} ${boxSize} units box
create_box      1 reg

neighbor        ${neighborDist} bin # nsq if too many neighbor atoms
neigh_modify exclude type 1 1 #do not perform collision tracking for situations with phiP < 1 Vol%!

#Material properties required for new pair styles
fix             m1 all property/global youngsModulus peratomtype ${youngsModulus}
fix             m2 all property/global poissonsRatio peratomtype ${poissonsRatio}
fix             m3 all property/global coefficientRestitution peratomtypepair 1 ${coeR}
fix             m4 all property/global coefficientFriction peratomtypepair 1 ${coeF}

#pair style
pair_style gran model hertz      #Hertzian without cohesion
pair_coeff      * *

#timestep, gravity
timestep        ${timeStepDEM}
fix tscheck all check/timestep/gran 100 0.1 0.1 # warns if timestep exceeds Rayleigh or Hertz (fractioned) time

fix             gravi all gravity 9.81 vector 0.0 0.0 -1.0

#walls

#particle distribution
variable minVolumeLimit equal 1e-40 #minimum individual particle limit
fix      pts1 all particletemplate/sphere 1 atom_type 1 density constant ${rhoP} radius constant ${rad1} volume_limit ${minVolumeLimit}
fix      pts2 all particletemplate/sphere 1 atom_type 1 density constant ${rhoP} radius constant ${rad2} volume_limit ${minVolumeLimit}
fix      pts3 all particletemplate/sphere 1 atom_type 1 density constant ${rhoP} radius constant ${rad3} volume_limit ${minVolumeLimit}
fix      pts4 all particletemplate/sphere 1 atom_type 1 density constant ${rhoP} radius constant ${rad4} volume_limit ${minVolumeLimit}
fix      pts5 all particletemplate/sphere 1 atom_type 1 density constant ${rhoP} radius constant ${rad5} volume_limit ${minVolumeLimit}
fix      pts6 all particletemplate/sphere 1 atom_type 1 density constant ${rhoP} radius constant ${rad6} volume_limit ${minVolumeLimit}
fix      pdd1 all particledistribution/discrete 1 6 pts1 ${vfrac1} pts2 ${vfrac2} pts3 ${vfrac3} pts4 ${vfrac4} pts5 ${vfrac5} pts6 ${vfrac6}

#option 1 for insertion
fix             ins all insert/pack seed 101 distributiontemplate pdd1 vel constant 0 0 0 insert_every once overlapcheck yes all_in yes volumefraction_region ${phiP} region reg

#cfd coupling
fix             cfd all     couple/cfd couple_every ${timeStepMultiplicatorCFD} mpi
fix             cfd2 all    couple/cfd/force

#insert the particles
```

```
run 1

#apply nve integration to all particles that are inserted as single particles
fix             integr all nve/sphere

# calculate average velocity
variable          particleMomentumZ atom mass*vz
variable          myMass atom mass
compute           myMomentumPz all reduce sum v_particleMomentumZ
compute           totalMassP   all reduce sum v_myMass
variable          myMassPZVar equal c_totalMassP
variable          myMomentumPZVar equal c_myMomentumPz/(c_totalMassP+1e-99)
variable          currTime equal step*${timeStepDEM}
fix               printmyMomentum all print ${timeStepMultiplicatorPrint} "${currTime} ${myMassPZVar} ${myMomentumPZVar}" file mean.dat screen no

# calculate overlapping pairs in %
compute         PartDia all property/atom diameter
compute         minPartDia all reduce min c_PartDia
compute         myPair all pair/local dist
compute         myPairMin all reduce min c_myPair
variable        maxoverlap equal ((1.)-c_myPairMin/(c_minPartDia))*100
fix             reportOverlap all print ${timeStepMultiplicatorPrint} "${currTime} ${maxoverlap}" file reportOverlap.dat title "time maxoverlap[%]" screen no

#screen output
compute         centerOfMass all com
compute         1 all erotate/sphere
thermo_style    custom step atoms ke c_centerOfMass[1] c_centerOfMass[2] c_centerOfMass[3]
thermo          ${timeStepMultiplicatorDump}
thermo_modify   lost ignore norm no
compute_modify  thermo_temp dynamic yes

restart         ${timeStepMultiplicatorCFD} liggghts.restart.1 liggghts.restart.2      # run only in CFDEM
dump            dmp all custom ${timeStepMultiplicatorDump} ../DEM/post/dump*.part id type type x y z ix iy iz vx vy vz fx fy fz omegax omegay omegaz radius
run             1

```

```
## MAIN INPUT PARAMETERS ##
variable phiP equal   0.02
variable rhoP equal 2250.0
variable youngsModulus  equal 5.e7
variable poissonsRatio  equal 0.45
variable coeR   equal 0.9
variable coeF   equal 0.5
variable timeStepDEM equal 1e-8 # not important here (particle generation)
variable coarseGrainingRatio equal 10. # coarse graining ratio = computational parcel size / original particle size . wird in Berechnungen verwendet. Daher 1 setzen wenn coarsegraining inaktiv
variable dumpInterval equal 100000 # amount of timesteps DEM beetween dumps

## Particle Size Distribution
variable d1 equal  6.7968e-6 #type 1
variable vfrac1 equal 1.0 #volume fraction of particles

## Domain
variable boxSize        equal 4.2e-3 # region volume !>= 1e-10
variable sizeOrigin     equal 0.

## INPUT CALCULATIONS ##
variable rad1 equal     ${d1}/2.
variable neighborDist         equal 0.01*${coarseGrainingRatio}*${d1}
####################################

#echo both
coarsegraining ${coarseGrainingRatio}

atom_style      granular
atom_modify     map array
communicate     single vel yes

boundary    p p p
newton          off

units           si


region          reg block ${sizeOrigin} ${boxSize} ${sizeOrigin} ${boxSize} ${sizeOrigin} ${boxSize} units box
create_box      1 reg

neighbor            ${neighborDist} bin # nsq if too many neighbor atoms
neigh_modify    delay 0 one 1000

#Material properties required for new pair styles
fix             m1 all property/global youngsModulus peratomtype ${youngsModulus}
fix             m2 all property/global poissonsRatio peratomtype ${poissonsRatio}
fix             m3 all property/global coefficientRestitution peratomtypepair 1 ${coeR}
fix             m4 all property/global coefficientFriction peratomtypepair 1 ${coeF}

#pair style
pair_style      gran model hertz tangential history #Hertzian without cohesion
pair_coeff      * *

```

```
#timestep, gravity
timestep        ${timeStepDEM}
fix tscheck all check/timestep/gran 1 0.1 0.1 # warns if timestep exceeds Rayleigh or Hertz (fractioned) time

#particle distribution
variable minVolumeLimit equal 1e-40 #minimum individual particle limit
fix      pts1 all particletemplate/sphere 1 atom_type 1 density constant ${rhoP} radius constant ${rad1} volume_limit ${minVolumeLimit}
fix      pdd1 all particledistribution/discrete 1 1 pts1 ${vfrac1}

#option 1 for insertion
fix             ins all insert/pack seed 101 distributiontemplate pdd1 vel constant 0 0 0 insert_every once overlapcheck yes all_in yes volumefraction_region ${phiP} region reg


# calculate particle pairs that are in contact 
compute         PartDia all property/atom diameter
compute         minPartDia all reduce min c_PartDia
compute         myPair all pair/local dist
compute         myPairMin all reduce min c_myPair
variable        maxoverlap equal ((1.)-c_myPairMin/(c_minPartDia))*100
#calculate the total mass of all particles
variable myMass         atom mass
compute totalMass all   reduce sum v_myMass
variable varPhi         equal c_totalMass/vol/${rhoP}

fix             printMass all print 1 "${varPhi}" file myData.dat screen no
fix             reportOverlap all print 1 "${maxoverlap}" file reportOverlap.dat title "time maxoverlap[%]"

#screen output
thermo_style    custom step atoms vol c_totalMass
thermo          ${dumpInterval}
thermo_modify   lost ignore norm no
compute_modify  thermo_temp dynamic yes

# Daten für paraview
dump            dmp all custom 1 post/dump*.liggghts id type x y z vx vy vz fx fy fz omegax omegay omegaz radius
# Daten für CFDEM
dump            CFDEMdmp all custom 1 post/CFDEMdump*.liggghts id type diameter density x y z
#insert the particles so that dump is not empty
run             1
```

9.2.3 2D Model

9.2.3.1 Meshing

The following printed file contents consist of code which has not been discussed in detail in the meshing of the 2D riser model and are listed in alphabetical order. Geometry, mesh, ParaView state files and executables (*.stl, *.msh, *.pvsm, *(.sh)) have not been printed. The whole post-processed “JICF_1N_out3_meshing” case directories contents can be viewed on the attached disk.

system/changeDictionaryDict

system/controlDict

system/extrudeMeshDict

system/fvSchemes

system/fvSolution

system/snappyHexMeshDict.castellate

system/snappyHexMeshDict.snap

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      changeDictionaryDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dictionaryReplacement
{
    boundary
    {
        defaultFaces
        {
            type            wall;
        }
    }
}


// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     chtMultiRegionFoam;

startFrom       latestTime;

startTime       0; //0.001;

stopAt          endTime;

endTime         75;

deltaT          1; //0.001;

writeControl    adjustableRunTime;

writeInterval   15;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

maxCo           0.3;

maxDi           10.0;

adjustTimeStep  yes;

```

```
// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      extrudeProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

constructFrom patch;
sourceCase ".";
sourcePatches (minY);
exposedPatchName    maxY;

flipNormals true;

extrudeModel         linearNormal;
/*extrudeModel         linearDirection;*/

nLayers              1;
expansionRatio       1.0;

linearNormalCoeffs
{
    thickness        0.533;
}


mergeFaces false;

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
}

gradSchemes
{
}

divSchemes
{
}

laplacianSchemes
{
}

interpolationSchemes
{
}

snGradSchemes
{
}

fluxRequired
{
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

PIMPLE
{
    nOuterCorrectors 1;
}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      autoHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


// Which of the steps to run
castellatedMesh true;
snap            false;
addLayers       false;


// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    riser_nozzle_2D_rev5_clean.stl
    {
        type triSurfaceMesh;
        name riser;
    }
}

// Settings for the castellatedMesh generation.
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~~~~~~~~~~~~~~~~~

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 9000000; //100000;

    // Overall cell limit (approximately). Refinement will stop immediately
    // upon reaching this number so a refinement level might not complete.
    // Note that this is the number of cells before removing the part which
    // is not 'visible' from the keepPoint. The final number of cells might
```

```
    // actually be a lot less.
    maxGlobalCells 90000000; //2000000;

    // The surface refinement loop might spend lots of iterations
    // refining just a few cells. This setting will cause refinement
    // to stop if <= minimumRefine are selected for refinement. Note:
    // it will at least do one iteration (unless the number of cells
    // to refine is 0)
    minRefinementCells 2;

    // Allow a certain level of imbalance during refining
    // (since balancing is quite expensive)
    // Expressed as fraction of perfect balance (= overall number of cells /
    // nProcs). 0=balance always.
    maxLoadUnbalance 0.05; //0.0; //0.10

    // Number of buffer layers between different levels.
    // 1 means normal 2:1 refinement restriction, larger means slower
    // refinement.
    nCellsBetweenLevels 6;



    // Explicit feature edge refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies a level for any cell intersected by its edges.
    // This is a featureEdgeMesh, read from constant/triSurface for now.
    features
    (
          {
              file "riser_nozzle_2D_rev5_clean.eMesh";
             levels ( ( 0.0 0) ); //specify refinement levels near the feature
          }
    );

    // Surface based refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies two levels for every surface. The first is the minimum level,
    // every cell intersecting a surface gets refined up to the minimum level.
    // The second level is the maximum level. Cells that 'see' multiple
    // intersections where the intersections make an
    // angle > resolveFeatureAngle get refined up to the maximum level.

    refinementSurfaces
    {
        riser
        {
            // Surface-wise min and max refinement level
            level (1 1); //level (1 1);
        }

    }

    // Resolve sharp angles
    resolveFeatureAngle 30;
```

```


    // Region-wise refinement
    // ~~~~~~~~~~~~~~~~~~~~~~

    // Specifies refinement level for cells in relation to a surface. One of
    // three modes
    // - distance. 'levels' specifies per distance to the surface the
    //   wanted refinement level. The distances need to be specified in
    //   descending order.
    // - inside. 'levels' is only one entry and only the level is used. All
    //   cells inside the surface get refined up to the level. The surface
    //   needs to be closed for this to be possible.
    // - outside. Same but cells outside.

    refinementRegions
    {
        //refinementBox
        //{
        //    mode inside;
        //    levels ((1E15 4));
        //
    }


    // Mesh selection
    // ~~~~~~~~~~~~~~

    // After refinement patches get added for all refinementSurfaces and
    // all cells intersecting the surfaces get put into these patches. The
    // section reachable from the locationInMesh is kept.
    // NOTE: This point should never be on a face, always inside a cell, even
    // after refinement.
    locationInMesh (0 0 0); //locationInMesh (0.01 0.01 0.01);


    // Whether any faceZones (as specified in the refinementSurfaces)
    // are only on the boundary of corresponding cellZones or also allow
    // free-standing zone faces. Not used if there are no faceZones.
    allowFreeStandingZoneFaces false;
}



// Settings for the snapping.
snapControls
{
    //- Number of patch smoothing iterations before finding correspondence
    //  to surface
    nSmoothPatch 7; //3;

    //- Relative distance for points to be attracted by surface feature point
    //  or edge. True distance is this factor times local
    //  maximum edge length.
    tolerance 1.0;

    //- Number of mesh displacement relaxation iterations.
    nSolveIter 150; //30;
```

```

    //- Maximum number of snapping relaxation iterations. Should stop
    //  before upon reaching a correct mesh.
    nRelaxIter 12; //5;

    // Feature snapping

        //- Highly experimental and wip: number of feature edge snapping
        //  iterations. Leave out altogether to disable.
        //  Of limited use in this case since faceZone faces not handled.
        nFeatureSnapIter 16; //10;

    //- Detect (geometric only) features by sampling the surface
    //  (default=false).
    implicitFeatureSnap false;

    //- Use castellatedMeshControls::features (default = true)
    explicitFeatureSnap true;

    //- Detect points on multiple surfaces (only for explicitFeatureSnap)
    multiRegionFeatureSnap false;

}



// Settings for the layer addition.
addLayersControls
{
    relativeSizes true;

    // Per final patch (so not geometry!) the layer information
    layers
    {
//          maxY
//          {
//              nSurfaceLayers 3;
//          }
//          "(riser).*"
//          {
//              nSurfaceLayers 2;
//          }
    }

    // Expansion factor for layer mesh
    expansionRatio 1.1;

    // Wanted thickness of final added cell layer. If multiple layers
    // is the thickness of the layer furthest away from the wall.
    // Relative to undistorted size of cell outside layer.
    finalLayerThickness 0.7; //1;

    // Minimum thickness of cell layer. If for any reason layer
    // cannot be above minThickness do not add layer.
    // Relative to undistorted size of cell outside layer.
    minThickness 0.1; // ..0.1..0.15..;

    // If points get not extruded do nGrow layers of connected faces that
    are
    // also not grown. This helps convergence of the layer addition process
    // close to features.
```

```
    // Note: changed(corrected) w.r.t 17x! (didn't do anything in 17x)
    nGrow 1; //0;

    // Advanced settings

    // When not to extrude surface. 0 is flat surface, 90 is when two faces
    // are perpendicular
    featureAngle 270; // 110; // 30;

    // At non-patched sides allow mesh to slip if extrusion direction makes
    // angle larger than slipFeatureAngle.
    slipFeatureAngle 30;

    // Maximum number of snapping relaxation iterations. Should stop
    // before upon reaching a correct mesh.
    nRelaxIter 3;

    // Number of smoothing iterations of surface normals
    nSmoothSurfaceNormals 1;

    // Number of smoothing iterations of interior mesh movement direction
    nSmoothNormals 3;

    // Smooth layer thickness over surface patches
    nSmoothThickness 10; // 2;

    // Stop layer growth on highly warped cells
    maxFaceThicknessRatio 0.9; // 0.5;

    // Reduce layer growth where ratio thickness to medial
    // distance is large
    maxThicknessToMedialRatio 0.9; // 1;

    // Angle used to pick up medial axis points
    // Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130 in 17x.
    minMedianAxisAngle 270; // 130; // 90;

    // Create buffer region for new layer terminations
    nBufferCellsNoExtrude 0;

    // Overall max number of layer addition iterations. The mesher will exit
    // if it reaches this number of iterations; possibly with an illegal
    // mesh.
    nLayerIter 50;
}



// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
{
    //- Maximum non-orthogonality allowed. Set to 180 to disable.
    maxNonOrtho 65;

    //- Max skewness allowed. Set to <0 to disable.
    maxBoundarySkewness 20;
    maxInternalSkewness 2.5; //4;

```

```
    //- Max concaveness allowed. Is angle (in degrees) below which concavity
    //  is allowed. 0 is straight face, <0 would be convex face.
    //  Set to 180 to disable.
    maxConcave 80;

    //- Minimum pyramid volume. Is absolute volume of cell pyramid.
    //  Set to very negative number (e.g. -1E30) to disable.
    minVol 1e-16; //0;

    //- Minimum quality of the tet formed by the face-centre
    //  and variable base point minimum decomposition triangles and
    //  the cell centre.  Set to very negative number (e.g. -1E30) to
    //  disable.
    //     <0 = inside out tet,
    //      0 = flat tet
    //      1 = regular tet
    minTetQuality 1e-30;

    //- Minimum face area. Set to <0 to disable.
    minArea -1;

    //- Minimum face twist. Set to <-1 to disable. dot product of face normal
    //- and face centre triangles normal
    minTwist 0.02;

    //- minimum normalised cell determinant
    //- 1 = hex, <= 0 = folded or flattened illegal cell
    minDeterminant 0.001;

    //- minFaceWeight (0 -> 0.5)
    minFaceWeight 0.02;

    //- minVolRatio (0 -> 1)
    minVolRatio 0.01;

    //must be >0 for Fluent compatibility
    minTriangleTwist -1;


    // Advanced

    //- Number of error distribution iterations
    nSmoothScale 6; //4;
    //- amount to scale back displacement at error points
    errorReduction 0.82; //0.75;

      relaxed
    {
        //- Maximum non-orthogonality allowed. Set to 180 to disable.
        maxNonOrtho 75;
    }
}


// Advanced

// Flags for optional output
// 0 : only write final meshes
// 1 : write intermediate meshes
```

```
// 2 : write volScalarField with cellLevel for postprocessing
// 4 : write current intersections as .obj files
debug 0;


// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1e-6;


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      autoHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Which of the steps to run
castellatedMesh false;
snap            true;
addLayers       true;


// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    riser_nozzle_2D_rev5.stl
    {
        type triSurfaceMesh;
        name riser;
    }
}

// Settings for the castellatedMesh generation.
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~~~~~~~~~~~~~~~~~

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 1000000; //100000;

    // Overall cell limit (approximately). Refinement will stop immediately
    // upon reaching this number so a refinement level might not complete.
    // Note that this is the number of cells before removing the part which
    // is not 'visible' from the keepPoint. The final number of cells might
```

```
    // actually be a lot less.
    maxGlobalCells 10000000; //2000000;

    // The surface refinement loop might spend lots of iterations
    // refining just a few cells. This setting will cause refinement
    // to stop if <= minimumRefine are selected for refinement. Note:
    // it will at least do one iteration (unless the number of cells
    // to refine is 0)
    minRefinementCells 10;

    // Allow a certain level of imbalance during refining
    // (since balancing is quite expensive)
    // Expressed as fraction of perfect balance (= overall number of cells /
    // nProcs). 0=balance always.
    maxLoadUnbalance 0.05; //0.0; //0.10

    // Number of buffer layers between different levels.
    // 1 means normal 2:1 refinement restriction, larger means slower
    // refinement.
    nCellsBetweenLevels 2;



    // Explicit feature edge refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies a level for any cell intersected by its edges.
    // This is a featureEdgeMesh, read from constant/triSurface for now.
    features
    (
          {
              file "riser_nozzle_2D_rev5.eMesh";
             levels ( ( 0.0 1) ); //specify refinement levels near the feature
          }
    );

    // Surface based refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies two levels for every surface. The first is the minimum level,
    // every cell intersecting a surface gets refined up to the minimum level.
    // The second level is the maximum level. Cells that 'see' multiple
    // intersections where the intersections make an
    // angle > resolveFeatureAngle get refined up to the maximum level.

    refinementSurfaces
    {
        riser
        {
            // Surface-wise min and max refinement level
            level (0 0); //level (1 1);
        }

    }

    // Resolve sharp angles
    resolveFeatureAngle 30;
```

```


    // Region-wise refinement
    // ~~~~~~~~~~~~~~~~~~~~~~

    // Specifies refinement level for cells in relation to a surface. One of
    // three modes
    // - distance. 'levels' specifies per distance to the surface the
    //   wanted refinement level. The distances need to be specified in
    //   descending order.
    // - inside. 'levels' is only one entry and only the level is used. All
    //   cells inside the surface get refined up to the level. The surface
    //   needs to be closed for this to be possible.
    // - outside. Same but cells outside.

    refinementRegions
    {
        //refinementBox
        //{
        //    mode inside;
        //    levels ((1E15 4));
        //
    }


    // Mesh selection
    // ~~~~~~~~~~~~~~

    // After refinement patches get added for all refinementSurfaces and
    // all cells intersecting the surfaces get put into these patches. The
    // section reachable from the locationInMesh is kept.
    // NOTE: This point should never be on a face, always inside a cell, even
    // after refinement.
    locationInMesh (0 0 0); //locationInMesh (0.01 0.01 0.01);


    // Whether any faceZones (as specified in the refinementSurfaces)
    // are only on the boundary of corresponding cellZones or also allow
    // free-standing zone faces. Not used if there are no faceZones.
    allowFreeStandingZoneFaces false;
}



// Settings for the snapping.
snapControls
{
    //- Number of patch smoothing iterations before finding correspondence
    //  to surface
    nSmoothPatch 6; //8; //3;

    //- Relative distance for points to be attracted by surface feature point
    //  or edge. True distance is this factor times local
    //  maximum edge length.
    tolerance 0.5;

    //- Number of mesh displacement relaxation iterations.
    nSolveIter 10; //100; //30;
```

```

    //- Maximum number of snapping relaxation iterations. Should stop
    //  before upon reaching a correct mesh.
    nRelaxIter 10; //5;

    // Feature snapping

        //- Highly experimental and wip: number of feature edge snapping
        //  iterations. Leave out altogether to disable.
        //  Of limited use in this case since faceZone faces not handled.
        nFeatureSnapIter 10; //10;

    //- Detect (geometric only) features by sampling the surface
    //  (default=false).
    implicitFeatureSnap false;

    //- Use castellatedMeshControls::features (default = true)
    explicitFeatureSnap true;

    //- Detect points on multiple surfaces (only for explicitFeatureSnap)
    multiRegionFeatureSnap false;

}



// Settings for the layer addition.
addLayersControls
{
    relativeSizes true;

    // Per final patch (so not geometry!) the layer information
    layers
    {
         "(defaultFaces).*"
         {
             nSurfaceLayers 2;
         }
         "(riser).*"
         {
             nSurfaceLayers 2;
         }
         "(jet).*"
         {
             nSurfaceLayers 2;
         }
    }

    // Expansion factor for layer mesh
    expansionRatio 1.15;

    // Wanted thickness of final added cell layer. If multiple layers
    // is the thickness of the layer furthest away from the wall.
    // Relative to undistorted size of cell outside layer.
    finalLayerThickness 0.45; //1;

    // Minimum thickness of cell layer. If for any reason layer
    // cannot be above minThickness do not add layer.
    // Relative to undistorted size of cell outside layer.
    minThickness 0.1; // ..0.1..0.15..;

```

```
    // If points get not extruded do nGrow layers of connected faces that are
    // also not grown. This helps convergence of the layer addition process
    // close to features.
    // Note: changed(corrected) w.r.t 17x! (didn't do anything in 17x)
    nGrow 0; //0;

    // Advanced settings

    // When not to extrude surface. 0 is flat surface, 90 is when two faces
    // are perpendicular
    featureAngle 90; // 110; // 30;

    // At non-patched sides allow mesh to slip if extrusion direction makes
    // angle larger than slipFeatureAngle.
    slipFeatureAngle 30;

    // Maximum number of snapping relaxation iterations. Should stop
    // before upon reaching a correct mesh.
    nRelaxIter 3;

    // Number of smoothing iterations of surface normals
    nSmoothSurfaceNormals 10; //4;

    // Number of smoothing iterations of interior mesh movement direction
    nSmoothNormals 15; //6;

    // Smooth layer thickness over surface patches
    nSmoothThickness 10; // 2;

    // Stop layer growth on highly warped cells
    maxFaceThicknessRatio 0.5; // 0.5;

    // Reduce layer growth where ratio thickness to medial
    // distance is large
    maxThicknessToMedialRatio 0.9; // 1;

    // Angle used to pick up medial axis points
    // Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130 in 17x.
    minMedianAxisAngle 270; // 130; // 90;

    // Create buffer region for new layer terminations
    nBufferCellsNoExtrude 1;

    // Overall max number of layer addition iterations. The mesher will exit
    // if it reaches this number of iterations; possibly with an illegal
    // mesh.
    nLayerIter 70;
}



// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
{
    //- Maximum non-orthogonality allowed. Set to 180 to disable.
    maxNonOrtho 75;

```

```
    //- Max skewness allowed. Set to <0 to disable.
    maxBoundarySkewness 10;
    maxInternalSkewness 2.5; //4;

    //- Max concaveness allowed. Is angle (in degrees) below which concavity
    //  is allowed. 0 is straight face, <0 would be convex face.
    //  Set to 180 to disable.
    maxConcave 80;

    //- Minimum pyramid volume. Is absolute volume of cell pyramid.
    //  Set to very negative number (e.g. -1E30) to disable.
    minVol 1e-16; //0;

    //- Minimum quality of the tet formed by the face-centre
    //  and variable base point minimum decomposition triangles and
    //  the cell centre.  Set to very negative number (e.g. -1E30) to
    //  disable.
    //     <0 = inside out tet,
    //      0 = flat tet
    //      1 = regular tet
    minTetQuality 1e-30;

    //- Minimum face area. Set to <0 to disable.
    minArea -1;

    //- Minimum face twist. Set to <-1 to disable. dot product of face normal
    //- and face centre triangles normal
    minTwist 0.02;

    //- minimum normalised cell determinant
    //- 1 = hex, <= 0 = folded or flattened illegal cell
    minDeterminant 0.001;

    //- minFaceWeight (0 -> 0.5)
    minFaceWeight 0.02;

    //- minVolRatio (0 -> 1)
    minVolRatio 0.01;

    //must be >0 for Fluent compatibility
    minTriangleTwist -1;


    // Advanced

    //- Number of error distribution iterations
    nSmoothScale 8; //4;
    //- amount to scale back displacement at error points
    errorReduction 0.9; //0.75;

    relaxed
    {
        //- Maximum non-orthogonality allowed. Set to 180 to disable.
        maxNonOrtho 75;
    }
}


// Advanced
```

```

// Flags for optional output
// 0 : only write final meshes
// 1 : write intermediate meshes
// 2 : write volScalarField with cellLevel for postprocessing
// 4 : write current intersections as .obj files
debug 0;


// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1e-6;


// ************************************************************************* //

```

9.2.3.2 Steady-State Gas Flow

The following printed file contents consist of code which has not been discussed in detail in the setup for the single-phase gas flow in the 2D riser model and are listed in alphabetical order. From the previous post-processed case, the mesh has had to be copied to constant/polyMesh. Geometry, mesh, ParaView state files and executables (*.stl, *.msh, *.pvsm, *(.sh)) have not been printed. The whole "JICF_1N_in3_out3_PSD_1P_fvOptions_Ujet_2_Upjet_2_5000" post-processed case directories contents can be viewed on the attached disk.

CFD/system/controlDict

CFD/system/fvOptions

CFD/system/fvSchemes

CFD/system/fvSolution

CFD/system/topoSetDict

Unfortunately, the injection velocity directs upwards instead of downwards. Surprisingly, the gas velocity at the recirculate injection point has shown negligible effects on the main flow as depicted in Figure 6.1 due to its low momentum.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         500;

deltaT          0.1;

writeControl    adjustableRunTime;

writeInterval   10;

purgeWrite      50;

writeFormat     ascii;

writePrecision  6;

writeCompression compressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           0.1;

maxDeltaT       1;

```

```
z_nozzle        -1.7;   //acc. to paraview
z_jet           6.9;    //acc. to paraview
z_oldOutlet     25;     //acc. to paraview

libs ( "libfiniteVolumeCFDEM.so" );

functions
(
    probes
    {
        // Where to load it from
        functionObjectLibs ( "libsampling.so" );
        type probes;
        // Name of the directory for probe data
        name            probes;
        // Write at same frequency as fields
        outputControl   outputTime; //timeStep
        outputInterval  1;

        probeLocations
        (
            //estimate time for steady-state by comparing calc. mean velocity with average velocity over probes
            (  0    0 $z_nozzle )       // middle of one of the nozzles, z acc. to paraview
            (  0    0 $z_jet    )       // middle of the big pipe, z at jet inlet
            ( -2.29 0 $z_jet    )       // big pipe r=4.57/2 angle=  180°
            (  0    0 $z_oldOutlet )    // middle of the old outlet
            ( -2.29 0 $z_oldOutlet )    // big pipe r=4.57/2 angle=  180°
            ( -3.9  0 38.6      )       // middle of the left outlet
            ( -3.9  0 33.3      )       // lower half of left outlet
        );
        // Fields to be probed
        fields
        (
            p U UMean //voidfractionMean kMean
        );
    }

    outletAverage_1
    {
        type            faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        log                  yes;
        outputControl   outputTime; //timeStep
        outputInterval   20;
        valueOutput     true;
        surfaceFormat   null;
        source               patch;
        sourceName      outlet;
        operation          areaAverage;
//          weightField       voidfraction;
        fields
        (
            U UMean //voidfractionMean kMean
        );
    }

//      fieldMinMaxK
//      {
```

```
//         type fieldMinMax;
//         functionObjectLibs ("libfieldFunctionObjects.so");
//         write        yes;
//         log          yes;
//         outputControl        timeStep;
//         outputInterval       1;
//         mode         magnitude;
//         fields
//         (
//             k
//         );
//     }

    fieldMinMaxU
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write           yes;
        log             yes;
        outputControl   timeStep;
        outputInterval  1;
        mode            magnitude;
        fields
        (
            U
        );
    }

    fieldMinMaxP
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write           yes;
        log             yes;
        outputControl   timeStep;
        outputInterval  1;
        mode            magnitude;
        fields
        (
            p
        );
    }

    fieldAverage1
    {
        type            fieldAverage;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   outputTime;

        fields
        (
            U
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }

//             voidfraction
//             {
```

```
//                 mean        on;
//                 prime2Mean  on;
//                 base        time;
//             }
//
//             k
//             {
//                 mean        on;
//                 prime2Mean  on;
//                 base        time;
//             }
        );
    }

);


// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

momentumSource
{
    type            pressureGradientExplicitSource;
    active          on;            //on/off switch
    selectionMode   cellSet;       //cellSet // points //cellZone
    cellSet         inlet;

    pressureGradientExplicitSourceCoeffs
    {
        fieldNames  (U);
        Ubar        (1.99 0 .244);      // 2 m/s inclined 83° from upright
    }
}


// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState; //Euler; //steadyState; <-- start with steady!
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none; //Gauss linear;
    div(phi,U)      Gauss upwind; //limitedLinearV 1; //<-- start with upwind
    div(phi,k)      Gauss upwind; //limitedLinear 1; //<-- start with upwind
    div(phi,epsilon) Gauss upwind; //limitedLinear 1; //<-- start with upwind
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss upwind; //limitedLinear 1;
    div((viscousTerm*dev(grad(U).T()))) Gauss linear;
    div(((nu*rho)*dev(grad(U).T()))) Gauss linear;
    div((nu*dev(grad(U).T()))) Gauss linear;
    div((nuEff*dev(grad(U).T()))) Gauss linear;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
    laplacian(viscousTerm,U) Gauss linear corrected;
    laplacian(nu,U) Gauss linear corrected;
```

```
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian((voidfraction2|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DREff,R) Gauss linear corrected;
    laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-7;
        relTol          1e-04;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }

    pFinal //Only relevant if Pimple-type solver used!!
    {
        solver          GAMG;
        tolerance       1e-7;
        relTol          0;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
```

```
        tolerance       1e-05;
        relTol          0;
    }

    UFinal //Only relevant if Pimple-type solver used!!
    {
        $U
        tolerance       1e-05;
        relTol          0;
    }

}

SIMPLE       //<-- start with simpleFoam solver!*/
{
    nNonOrthogonalCorrectors 1;
    pRefCell        0;
    pRefValue       0;
}

//Only relevant if Pimple-type solver used!!
PISO
{
    nCorrectors                 2;
    nNonOrthogonalCorrectors    1;
}
PIMPLE
{
    nOuterCorrectors            2;
}

relaxationFactors
{
    fields
    {
        p       0.8;
    }
    equations
    {
        "U.*"           0.2; //0.7;    //start with 0.1 here
        "k.*"           0.001; //1.0;    //start with 0.001 here
        "epsilon.*"     0.001; //0.1;    //start with 0.001 here
    }
}



// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    {
        name    inlet;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
          box (-4.572 -0.266 6.4) (-4 0.266 7.3); //inlet from z = 6.5 to 7.25 equals one cell height
        }
    }
);

// ************************************************************************* //

```

9.2.3.3 Turbulent Gas Flow

The following printed file contents consist of code which has not been discussed in detail in the setup for the single-phase turbulent gas flow in the 2D riser model and are listed in alphabetical order. From the post-processed case “JICF_1N_out3_meshing”, mesh has had to be copied to constant/polyMesh. From the previous post-processed case “JICF_1N_in3_out3_PSD_1P_fvOptions_Ujet_2_Upjet_2_5000”, the contents of the folder holding its steady-state values have had to be copied to 0.org holding the initial values for the turbulent case “JICF_1N_in3_out3_PSD_1P_fvOptions_Ujet_2_Upjet_2_1EqEddy”. Its steady-state values have been used as initial values for the turbulent case “JICF_1N_in3_out3_PSD_1P_fvOptions_Ujet_2_Upjet_2_URANS”. The files of the case using the turbulence model One-Equation-Eddy have been printed as the changes for URANS are well-documented. Geometry, mesh, ParaView state files and executables (*.stl, *.msh, *.pvsm, *(.sh)) have not been printed. The whole post-processed cases directories contents can be viewed on the attached disk.

CFD/system/controlDict

CFD/system/sampleDict

Unfortunately, the injection velocity directs upwards instead of downwards in both cases. The main flow has been directed to the injection region as depicted in Figure 6.2 and Figure 6.4, thus a turbulent injection with the mean velocity of the laminar case may be sufficient to attract the main flow.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

myOutputIntervall 30;

application     pimpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         500;

deltaT          0.1;

writeControl    adjustableRunTime;

writeInterval   1.0;

purgeWrite      100;

writeFormat     ascii;

writePrecision  6;

writeCompression compressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           0.8;

```

```
maxDeltaT       1;

z_nozzle        -1.7;    //acc. to paraview
z_jet           6.9;     //acc. to paraview
z_oldOutlet     25;      //acc. to paraview

libs ( "libfiniteVolumeCFDEM.so" );

functions
(
    probes
    {
        // Where to load it from
        functionObjectLibs ( "libsampling.so" );
        type probes;
        // Name of the directory for probe data
        name            probes;
        // Write at same frequency as fields
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;

        probeLocations
        (
            //estimate time for steady-state by comparing calc. mean velocity with average velocity over probes
            (  0    0 $z_nozzle )       // middle of one of the nozzles, z acc. to paraview
            (  0    0 $z_jet    )       // middle of the big pipe, z at jet inlet
            ( -2.29 0 $z_jet    )       // big pipe r=4.57/2 angle=  180°
            (  0    0 $z_oldOutlet )    // middle of the old outlet
            ( -2.29 0 $z_oldOutlet )    // big pipe r=4.57/2 angle=  180°
            ( -3.9  0 38.6      )       // middle of the left outlet
            ( -3.9  0 33.3      )       // lower half of left outlet
        );
        // Fields to be probed
        fields
        (
            p U UMean k kMean //voidfractionMean
        );
    }

    outletAverage_1
    {
        type            faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        log                 yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        valueOutput     false;
        surfaceFormat   null;
        source              patch;
        sourceName      outlet;
        operation          areaAverage;
//          weightField        voidfraction;
        fields
        (
            U UMean k kMean//voidfractionMean
        );
    }

```

```
    fieldMinMaxK
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write   yes;
        log             yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode            magnitude;
        fields
        (
            k
        );
    }

    fieldMinMaxKMean
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write   yes;
        log             yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode            magnitude;
        fields
        (
            kMean
        );
    }

    fieldMinMaxU
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write           yes;
        log             yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode            magnitude;
        fields
        (
            U
        );
    }

    fieldMinMaxP
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write           yes;
        log             yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode            magnitude;
        fields
        (
            p
        );
    }

    fieldAverage1
```

```
    {
        type            fieldAverage;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   outputTime;

        fields
        (
            U
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }

            k
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }

//              voidfraction
//              {
//                  mean        on;
//                  prime2Mean  on;
//                  base        time;
//              }
        );
    }

);


// ************************************************************************* //
```

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.0                                   |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version         2.0;
    format          ascii;

    root            "/home/penfold/mattijs/foam/mattijs2.1/run/icoFoam";
    case            "cavity";
    instance        "system";
    local           "";

    class           dictionary;
    object          sampleDict;
}


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

pointCount 100; //number of sampling points


// interpolationScheme : choice of
//      cell            : use cell-centre value onlx; constant over cells
//      cellPoint       : use cell-centre and vertex values
//      cellPointFace   : use cell-centre, vertex and face values.
// 1] vertex values determined from neighbouring cell-centre values
// 2] face values determined using the current face interpolation scheme
//    for the field (linear, gamma, etc.)
interpolationScheme cellPointFace;


// writeFormat : choice of
//      xmgr
//      jplot
//      gnuplot
//      raw
setFormat       raw;


surfaceFormat       raw;
// sampling definition:
//
// Dictionary with fields
//      type : type of sampling method
//      name : name of samples. Used e.g. as filename
//      axis : how to write point coordinate
```

```
//      ...  : depending on method
//
//
// sample: choice of
//      uniform             evenly distributed points on line
//      face                one point per face intersection
//      midPoint            one point per cell, inbetween two face intersections
//      midPointAndFace     combination of face and midPoint
//
//      curve               specified points, not nessecary on line, uses
//                          tracking
//      cloud               specified points, uses findCell
//
//
// axis: how to write point coordinate. Choice of
// - x/y/z: x/y/z coordinate only
// - xyz: three columns
//  (probably does not make sense for anything but raw)
// - distance: distance from start of sampling line (if uses line) or
//             distance from first specified sampling point
//
// type specific:
//      uniform, face, midPoint, midPointAndFace : start and end coordinate
//      uniform: extra number of sampling points
//      curve, cloud: list of coordinates

//2D length scale differs from 3D to ensure constant mean velocities
x_outlet        -3.9;
z_min_outlet    35.7;   //outlet=vertHead/2
z_max_outlet    40.6;
z_oldOutlet     25;     //acc. to paraview
x_min_oldOutlet -4.6;   //big pipe r=4.57
x_max_oldOutlet  4.6;   //big pipe r=4.57
z_jet            6.9;   //acc. to paraview
x_min_jet       $x_min_oldOutlet;
x_max_jet       $x_max_oldOutlet;
z_nozzle        -1.7;   //acc. to paraview
x_min_nozzle    -0.62;
x_max_nozzle     0.62;
z_inlet         -4.8;   //acc. to paraview

sets
(
    line_axis
    {
        type        uniform;
        name        line_axis;
        axis        z;
        start       (0 0 $z_inlet);
        end         (0 0 $z_max_outlet);
        nPoints     $pointCount;
    }
    line_nozzle
    {
        type        uniform;
        name        line_nozzle;
        axis        x;
        start       ($x_min_nozzle 0 $z_nozzle);
        end         ($x_max_nozzle 0 $z_nozzle);
        nPoints     $pointCount;
```

```
    }
    line_jet
    {
        type        uniform;
        name        line_jet;
        axis        x;
        start       ($x_min_jet 0 $z_jet);
        end         ($x_max_jet 0 $z_jet);
        nPoints     $pointCount;
    }
    line_oldOutlet
    {
        type        uniform;
        name        line_outlet;
        axis        x;
        start       ($x_min_oldOutlet 0 $z_oldOutlet);
        end         ($x_max_oldOutlet 0 $z_oldOutlet);
        nPoints     $pointCount;
    }
    line_outlet
    {
        type        uniform;
        name        line_axis;
        axis        z;
        start       ($x_outlet 0 $z_min_outlet);
        end         ($x_outlet 0 $z_max_outlet);
        nPoints     $pointCount;
    }
);

surfaces
();


// Fields to sample.
fields
(
    //timeAverage_voidfraction
    UMean
//     UMean.component(1)
//     UMean.component(3)
//     voidfractionMean
    k
    kMean
);


// ************************************************************************* //

```

9.2.3.4 Quenching

The following printed file contents consist of code which has not been discussed in detail in the setup of the 2D riser simulation including quenching and are listed in alphabetical order. From the post-processed case “JICF_1N_out3_meshing”, the mesh has had to be copied to constant/polyMesh. From the post-processed case “JICF_1N_in3_out3_PSD_1P_fvOptions_Ujet_2_Upjet_2_1EqEddy”, the contents of the folder holding its steady-state values have had to be copied to 0 holding the initial values. Geometry, mesh, ParaView state files and executables (*.stl, *.msh, *.pvsm, *(.sh)) have not been printed. The whole post-processed “JICF_1N_in3_out3_optimized_quench” case directories contents can be viewed on the attached disk.

CFD/system/controlDict

CFD/system/fvSchemes

CFD/system/fvSolution

CFD/system/quenchAverages

CFD/system/quenchFunctionObject

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

myOutputIntervall 300;

application     pimpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         150;

deltaT          0.1;

writeControl    adjustableRunTime;

writeInterval   2;

purgeWrite      75;

writeFormat     ascii;

writePrecision  6;

writeCompression compressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           0.5;

```

```
maxDeltaT       1;

z_nozzle        -1.7;   //acc. to paraview
z_jet           6.9;    //acc. to paraview
z_oldOutlet     25;     //acc. to paraview

//libs ( "libfiniteVolumeCFDEM.so" );

functions
(
//Quench Settings
#include  "quenchFunctionObject"
#include  "quenchAverages"

    probes
    {
        // Where to load it from
        functionObjectLibs ( "libsampling.so" );
        type probes;
        // Name of the directory for probe data
        name            probes;
        // Write at same frequency as fields
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;

        probeLocations
        (
            //estimate time for steady-state by comparing calc. mean velocity with average velocity over probes
            (   0    0 $z_nozzle )       // middle of one of the nozzles, z acc. to paraview
            (   0    0 $z_jet    )       // middle of the big pipe, z at jet inlet
            ( -2.29 0 $z_jet    )       // big pipe r=4.57/2 angle=  180°
            (   0    0 $z_oldOutlet )    // middle of the old outlet
            ( -2.29 0 $z_oldOutlet )    // big pipe r=4.57/2 angle=  180°
            ( -3.9   0 38.6      )       // middle of the left outlet
            ( -3.9   0 33.3      )       // lower half of left outlet
        );
        // Fields to be probed
        fields
        (
            p U UMean k kMean quenchT quenchTMean quenchMuLiqMean quenchMuVapMean
        );
    }

    outletAverage_1
    {
        type            faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        log                  yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        valueOutput     false;
        surfaceFormat   null;
        source          patch;
        sourceName      outlet;
        operation         areaAverage;
//        weightField       phi;
        fields
```

```
        (
            U UMean k kMean //voidfractionMean
        );
    }

    fieldMinMaxK
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write   yes;
        log             yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode            magnitude;
        fields
        (
            k
        );
    }

    fieldMinMaxKMean
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write   yes;
        log             yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode            magnitude;
        fields
        (
            kMean
        );
    }

    fieldMinMaxU
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write           yes;
        log             yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode            magnitude;
        fields
        (
            U
        );
    }

    fieldMinMaxP
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write           yes;
        log             yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode            magnitude;
        fields
        (
```

```
            p
        );
    }

    fieldAverage1
    {
        type            fieldAverage;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   outputTime;

        fields
        (
            U
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }

            k
            {
                mean        on;
                prime2Mean  off;
                base        time;
            }

//             voidfraction
//             {
//                 mean        on;
//                 prime2Mean  on;
//                 base        time;
//             }
        );
    }

);


// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler; //steadyState; <-- start with steady!
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none; //Gauss linear;
    div(phi,U)                              Gauss upwind; // limitedLinearV 1; //<-- start with upwind
    div((phi*interpolate(particleMass)),U) Gauss upwind; // limitedLinear 1; //upwind; //linear;
    div(phi,k)                              Gauss upwind; //limitedLinear 1; //<-- start with upwind
    div(phi,epsilon) Gauss upwind; //limitedLinear 1; //<-- start with upwind
    div(phi,R)      Gauss upwind; //limitedLinear 1;
    div(R)          Gauss upwind; //linear;
    div(phi,nuTilda) Gauss upwind; //limitedLinear 1;
    div((viscousTerm*dev2(grad(U).T()))) Gauss linear;
    div(((nu*rho)*dev(grad(U).T()))) Gauss linear;
    div((nu*dev(grad(U).T()))) Gauss linear;
    div((nuEff*dev(grad(U).T()))) Gauss linear;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
    div(phi,quenchT)  Gauss  upwind; //linear;
}

laplacianSchemes
{
```

```
    default         Gauss linear corrected;
    laplacian(viscousTerm,U) Gauss linear corrected;
    laplacian(nu,U) Gauss linear corrected;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian((voidfraction2|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DREff,R) Gauss linear corrected;
    laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //
```

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

myMaxIterations         10;
myMaxIterationsPressure 10;

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-6;
        relTol          1e-3;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        maxIter         $myMaxIterationsPressure;
    }

    pFinal //Only relevant if Pimple-type solver used!!
    {
        solver          GAMG;
        tolerance       1e-6;
        relTol          0;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        maxIter         $myMaxIterationsPressure;
    }
```

```

    "(U|epsilon)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
        maxIter         $myMaxIterations;
    }

    "(U|epsilon)Final" //Only relevant if Pimple-type solver used!!
    {
        $U
        tolerance       1e-06;
        relTol          0;
        maxIter         $myMaxIterations;
    }

   "(k|kFinal)"
    {
        $U
        tolerance       1e-10;
        relTol          0;
        maxIter         $myMaxIterations;
    }

    "(quenchT)"
    {
        $k
    }

    "(quenchT)Final" //Only relevant if Pimple-type solver used!!
    {
        $kFinal
    }

}

// SIMPLE       //<-- start with simpleFoam solver!*/
// {
//     nNonOrthogonalCorrectors 1;
//     pRefCell        0;
//     pRefValue       0;
// }

//Only relevant if Pimple-type solver used!!
PISO
{
    nCorrectors                 2;
    nNonOrthogonalCorrectors    1;
}
PIMPLE
{
    nOuterCorrectors            2;
    nCorrectors                 1;
    nNonOrthogonalCorrectors    1;
}

//FINALLY DISABLE ALL RELAXATION FACTORS
relaxationFactors
{
```

```
    fields
    {
//        p       0.8;
    }
    equations
    {
//        "U.*"           0.2; //0.7;    //start with 0.1 here
//        "k.*"           0.3; //1.0;    //start with 0.001 here
//        "epsilon.*"     0.3; //0.1;    //start with 0.001 here
    }
}



// ************************************************************************* //

```

```
outletAverage_quench
{
    type            faceSource;
    functionObjectLibs ("libfieldFunctionObjects.so");
    log                  yes;
    outputControl   timeStep;
    outputInterval  $myOutputIntervall;
    valueOutput     false;
    surfaceFormat   null;
    source               patch;
    sourceName      outlet;
    operation          weightedAverage;
    weightField      phi;
    fields
    (
         quenchMuLiq  quenchMuVap  quenchT
    );
}

fieldMinMaxquenchMuLiq
{
    type fieldMinMax;
    functionObjectLibs ("libfieldFunctionObjects.so");
    write   yes;
    log             yes;
    outputControl   timeStep;
    outputInterval  $myOutputIntervall;
    mode            magnitude;
    fields
    (
        quenchMuLiq
    );
}

fieldMinMaxquenchMuVap
{
    type fieldMinMax;
    functionObjectLibs ("libfieldFunctionObjects.so");
    write   yes;
    log             yes;
    outputControl   timeStep;
    outputInterval  $myOutputIntervall;
    mode            magnitude;
    fields
    (
        quenchMuVap
    );
}

fieldMinMaxquenchT
{
    type fieldMinMax;
    functionObjectLibs ("libfieldFunctionObjects.so");
    write   yes;
    log             yes;
    outputControl   timeStep;
    outputInterval  $myOutputIntervall;
    mode            magnitude;
    fields
    (
        quenchT
```

```
        );
    }

    fieldAverage_quench
    {
        type            fieldAverage;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   outputTime;

        fields
        (
            quenchMuLiq
            {
                mean        on;
                prime2Mean  off;
                base        time;
            }

            quenchMuVap
            {
                mean        on;
                prime2Mean  off;
                base        time;
            }

            quenchT
            {
                mean        on;
                prime2Mean  off;
                base        time;
            }
        );
    }

```

```
quench
{
    type       evaporationCloud;
    functionObjectLibs ( "libutilityIPPTFunctionObjects.so");
    name            quench;
    phiName         "phi";
    ScT             0.7;
    deltaHEvap      deltaHEvap  [0  2 -2 0 0 0 0] 2.50e6;  //units: J/kg; deltaH_v
    tEvap           tEvap       [0  0 1  0 0 0 0] 1.22e-4; //units: s; tEvap_ = d_d^2 / (6*Sh*D_Vapor)

    rhoGas          rhoGas      [1 -3 0  0 0 0 0] 0.804;  //units: kg/m³
    rhoVap          rhoVap      [1 -3 0  0 0 0 0] 0.598;  //units: kg/m³
    rhoLiq          rhoLiq      [1 -3 0  0 0 0 0] 998;    //units: kg/m³
    rhoParticle     rhoParticle [1 -3 0  0 0 0 0] 2.25e3; //units: kg/m³

    cpGas           cpGas      [0  2 -2 -1 0 0 0] 1.04e3; //units: J/kg/K
    cpVap           cpVap      [0  2 -2 -1 0 0 0] 1.85e3; //units: J/kg/K
    cpLiq           cpLiq      [0  2 -2 -1 0 0 0] 4.19e3; //units: J/kg/K
    cpParticle      cpParticle [0  2 -2 -1 0 0 0] 999;    //units: J/kg/K particle represented by main component CaSO3-0.5H2O

    Tmax            Tmax       [0  0 0 1 0 0 0] 450;  //units: K, optional
    Tmin            Tmin       [0  0 0 1 0 0 0] 260;  //units: K, optional

    resetOnStartUp  false;
    autoSchemes     false;
    fvOptionsT   { };   //no extra source for temperature
    fvOptionsVap { };   //no extra source for vapor
    fvOptionsLiq
    {
        liquidInjection
        {
            type            scalarSemiImplicitSource;
            active          true;
            timeStart       0.0;
            duration        9999;
            selectionMode   cellSet;
            cellSet         quench; //needs to be generated with topoSet

            scalarSemiImplicitSourceCoeffs
            {
                volumeMode      absolute; // specific;
                injectionRateSuSp
                {
/*                          quenchMuLiq     (1 0); //units: kg/m³/s (if volumeMode = specific, this is the volume-specific quenching rate)*/
                    quenchMuLiq     (0.947 0); //units: kg/s    (if volumeMode = absolute, this is the absolute quenching rate)
                }
            }
        }
    };
}
```

9.2.3.5 Particle Injection

The following printed file contents consist of code which has not been discussed in detail in the setup for the particulate flow in the 2D riser model and are listed in alphabetical order. From the post-processed case "JICF_1N_out3_meshing", the mesh has had to be copied to constant/polyMesh and the manually created boundary side walls file wall.stl (see Section 5.4.2) to DEM. From the post-processed case "JICF_1N_in3_out3_optimized_quench", the contents of the folder holding its steady-state values have had to be copied to 0 holding the initial values. The files of the case "JICF_1N_in3_out3_optimized_dragCorrection" have been printed since drag correction is simply switched off by deleting the relevant line. Geometry, mesh, ParaView state files and executables (*.stl, *.msh, *.pvsm, *(.sh)) have not been printed. The post-processed case directories contents can be viewed on the attached disk.

CFD/constant/couplingProperties

CFD/system/fvSolution

DEM/in.jet

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.4                                   |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/


FoamFile
{
    version         2.0;
    format          ascii;

    root            "";
    case            "";
    instance        "";
    local           "";

    class           dictionary;
    object          couplingProperties;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//===========================================================================//
// sub-models & settings

myOutputIntervall 30;
myOutputIntervallParticleProbes 3e99;

solveFlow  true; //false;

treatVoidCellsAsExplicitForce false;

modelType "A";

couplingInterval 5;

IOModel "basicIO";

clockModel off;

locateModel engine; //turboEngine//standard;//

regionModel       allRegion;//differentialRegion;//

meshMotionModel   noMeshMotion;

voidFractionModel divided; //centre; //bigParticle;//

averagingModel dense;//dilute;//

```

```
/*smoothingModel off;*/
smoothingModel  constDiffSmoothing;
//smoothingModel  localPSizeDiffSmoothing;

momCoupleModels
(
    buoyantWeightCouple
    buoyantWeightCouple
    //explicitCoupleSource
);

//forceExplicitForceMapping; //force the use of explicit Lagr-To-Euler force Mapping
                                              //only relevant for Pimple-based solvers

forceModels
(
    //WenYuDrag
    //KochHillDrag
    BeetstraDrag
    gradPForce
    viscForce
    //Archimedes
//    fieldTimeAverage
//    volWeightedAverage
    averageSlipVel   //postProc
);

turbulenceModelType LESProperties; //"LESProperties";//"RASProperties";//

probeModel off; //particleProbe;

dataExchangeModel twoWayMPI;//twoWayFiles;//oneWayVTK;//

//===========================================================================//
// sub-model properties
engineProps
{
    treeSearch true;
}

dividedProps
{
    alphaMin 0.1;
    scaleUpVol 1.0;
}

constDiffSmoothingProps
{
    lowerLimit 0.0;
    upperLimit 1e99;
    smoothingLength 6.69e-2; //smoothingLength = CG*dPrim32*phiP^(-1/3)
/*    verbose;*/
}


implicitCoupleProps
{
```

```
    velFieldName "U";
    granVelFieldName "Us";
    voidfractionFieldName "voidfraction";
}

explicitCoupleProps
{
    //fLimit (0 0 0);
}

buoyantWeightCoupleProps
{
    rhoParticle 2250;
}

WenYuDragProps
{
    velFieldName "U";
    densityFieldName "rho";
    voidfractionFieldName  "voidfraction";
//    dPrim       100e-6;
//    dParcelRef  100e-6;
    interpolation true;
//    useEMMSDragModel;
//    verbose;
}

KochHillDragProps
{
    velFieldName "U";
    densityFieldName "rho";
//    dPrim       100e-6;
//    dParcelRef  100e-6;
    voidfractionFieldName "voidfraction";
    interpolation ;
}

BeetstraDragProps
{
//    verbose ;

    granVelFieldName "Us";
    velFieldName     "U";
    densityFieldName "rho";
    gravityFieldName "g";
    voidfractionFieldName "voidfraction";
    interpolation true;

    //switches for force handling
    forceSubModels
    (
        ImExCorr
    );
    explicitInterpCorr  true;
    implForceDEM        true;
}

gradPForceProps
{
    pFieldName "p";
    densityFieldName "rho";
```

```
    velocityFieldName "U";
    interpolation true;
}

viscForceProps
{
    velocityFieldName "U";
    densityFieldName "rho";
    interpolation true;
}

ArchimedesProps
{
    densityFieldName "rho";
    gravityFieldName "g";
}

fieldTimeAverageProps
{
    startTime 0.1;

    scalarFieldNames
    (
        "voidfraction"
    );

    vectorFieldNames
    (
        "Us"
    );
}

volWeightedAverageProps
{
    scalarFieldNames
    (
        voidfraction
    );
    vectorFieldNames
    (
    );
    upperThreshold 0.999;
    lowerThreshold 0;
//    verbous;
}

averageSlipVelProps
{
    rhoParticle             2250.;
    outputDirName           "averageProps";
    fluidVelFieldName       "U";
    particleVelFieldName    "Us";
    voidfractionFieldName   "voidfraction";
    rhoFluidName "rho";
}

particleProbeProps
{
    particleIDsToSample (0);
    verboseToFile;  //main switch
//    verbose;  //currently not used
```

```
    printEvery  $myOutputIntervallParticleProbes;      //print every this many CFD time steps
//    sampleAll;          //Activate sampling for all particles
    probeDebug;  //probes additional fields
    includePosition;  //will include particle position in the output file
    writePrecision 4;           //number of significant digits to print
}

twoWayMPIProps
{
    maxNumberOfParticles 10100;
    liggghtsPath "../DEM/in.jet";
}

totalMomentumExchangeProps
{
    implicitMomExFieldName "Ksl";
    explicitMomExFieldName "none";
    fluidVelFieldName "U";
    granVelFieldName "Us";
    densityFieldName "rho";
}
// ************************************************************************* //

```

```
/*---------------------------------*- C++ -*---------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

myMaxIterations         10;
myMaxIterationsPressure 10;

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-6;
        relTol          1e-3;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        maxIter         $myMaxIterationsPressure;
    }

    pFinal //Only relevant if Pimple-type solver used!!
    {
        solver          GAMG;
        tolerance       1e-6;
        relTol          0;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        maxIter         $myMaxIterationsPressure;
    }
```

```

    "(U|epsilon)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-06;
        relTol          0;
        maxIter         $myMaxIterations;
    }

    "(U|epsilon)Final" //Only relevant if Pimple-type solver used!!
    {
        $U
        tolerance       1e-06;
        relTol          0;
        maxIter         $myMaxIterations;
    }

   "(k|kFinal)"
    {
        $U
        tolerance       1e-10;
        relTol          0;
        maxIter         $myMaxIterations;
    }

    "(vSmoothField|sSmoothField|uP.|phiP.|voidfractionNext)"
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-07;
        relTol          1e-04;
        maxIter         $myMaxIterations;
    }

    "(vSmoothField|sSmoothField|uP.|phiP.|voidfractionNext)Final" //Only
    relevant if Pimple-type solver used!!
    {
        $voidfractionNext
        tolerance       1e-06;
        relTol          1e-3;
            maxIter             $myMaxIterations;
    }

    "(vSmoothField)"
    {
        $sSmoothField
            maxIter             0;
    }

    "(vSmoothField)Final" //Only relevant if Pimple-type solver used!!
    {
        $sSmoothField
            maxIter             0;
    }

    "(quenchT)"
    {
        $k
    }

```

```
    "(quenchT)Final" //Only relevant if Pimple-type solver used!!
    {
        $kFinal
    }

}

// SIMPLE       //<-- start with simpleFoam solver!*/
// {
//     nNonOrthogonalCorrectors 1;
//     pRefCell        0;
//     pRefValue       0;
// }

//Only relevant if Pimple-type solver used!!
PISO
{
    nCorrectors                 2;
    nNonOrthogonalCorrectors    1;
}
PIMPLE
{
    nOuterCorrectors            2;
    nCorrectors                 1;
    nNonOrthogonalCorrectors    1;
}

//FINALLY DISABLE ALL RELAXATION FACTORS
relaxationFactors
{
    fields
    {
//        p       0.8;
    }
    equations
    {
//        "U.*"           0.2; //0.7;    //start with 0.1 here
//        "k.*"           0.3; //1.0;    //start with 0.001 here
//        "epsilon.*"     0.3; //0.1;    //start with 0.001 here
    }
}



// ************************************************************************* //

```

```
## MAIN INPUT PARAMETERS ##
variable youngsModulus  equal 2.e6
variable poissonsRatio  equal 0.45
variable coeR           equal 0.9
variable coeF           equal 0.5
variable timeStepDEM    equal 20e-6
variable timeSpan       equal 200       #must result in time step multiplicators that are integers
variable Dumps          equal 2000      #must result in time step multiplicators that are integers

## Particle Size Distribution
variable d1 equal  5e-6 #type 1
variable d2 equal  12.5e-6 #type 2
variable d3 equal  17.5e-6 #type 3
variable d4 equal  22.5e-6 #type 4
variable d5 equal  27.5e-6 #type 5
variable d6 equal  35.0e-6 #type 6
variable dmax equal  ${d6}
variable d32  equal  6.8e-6
variable vfrac1 equal 0.62 #volume fraction of particles
variable vfrac2 equal 0.16
variable vfrac3 equal 0.10
variable vfrac4 equal 0.06
variable vfrac5 equal 0.03
variable vfrac6 equal 0.03
variable coarseGrainingRatio equal 1.02e3       #5.97e3 for 1e4, 1.02e3 for 2e6 Parcels @ phiP=1.11e-3; 5.05e3 for 1e4, 864 for 2e6 Parcels @ phiP=1e-3

## Geometry
variable wDomain        equal 0.532     #width of injection chute - must be equal to the height in y of  insertion_face.stl
variable dJet           equal ${wDomain}        #jet diameter
variable HInject        equal 0.748     #height of injection domain (approx. domain)
variable zInject        equal 6.9       #height of injection - must be equal to the position in y of  insertion_face.stl
variable zStart         equal -4.9      #beginning of simulation domain
variable zEnd           equal 40.5      #end of simulation domain
variable LDomain        equal 4.61      #half length of simulation domain little larger than riser diameter 9.144
variable LInject        equal 0.61
variable yDepth         equal 0.5*${wDomain}    #half width of simulation domain - pseudo 2D case
variable volRiser2D     equal 195       #

## Riser Operating Parameters
variable vInject        equal 2 #vertical injection velocity (of particles
variable mRate          equal 5.77      #(1.55e2/2)*(A2D/A3D), A3D/A2D=13.5; keep mass load constant (mInject/mInjectFluid)
variable angleInject    equal 83        #injection angle of particles
variable rhoF           equal 0.804     #fluid density
variable rhoP           equal 2250      #particle density

## Constants
variable piBy4          equal 0.78540   #constant
variable pi43           equal 4.1888    #constant
variable piBy180        equal 0.017453  #constant

## INPUT CALCULATIONS ##
variable rad1   equal   ${d1}/2.
```

```
variable rad2   equal   ${d2}/2.
variable rad3   equal   ${d3}/2.
variable rad4   equal   ${d4}/2.
variable rad5   equal   ${d5}/2.
variable rad6   equal   ${d6}/2.
variable radmax equal   ${dmax}/2.
variable VPart1 equal ${pi43}*${rad1}*${rad1}*${rad1}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable VPart2 equal ${pi43}*${rad2}*${rad2}*${rad2}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable VPart3 equal ${pi43}*${rad3}*${rad3}*${rad3}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable VPart4 equal ${pi43}*${rad4}*${rad4}*${rad4}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable VPart5 equal ${pi43}*${rad5}*${rad5}*${rad5}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable VPart6 equal ${pi43}*${rad6}*${rad6}*${rad6}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable zHeight        equal ${zEnd}-${zStart}         #height of simulation domain
variable zInjectLo      equal ${zInject}-${HInject}/2   #bottom of injection region
variable zInjectHi      equal ${zInject}+${HInject}/2   #bottom of injection region
variable xInject        equal -(${LDomain}-${LInject})  #inside end of injection region
variable tInject        equal ${timeSpan}       #99     #time interval for injection
variable mToInject      equal ${tInject}*${mRate}
variable nStepInj       equal ${coarseGrainingRatio}*${radmax}/${vInject}/${timeStepDEM}
variable vXInject       equal ${vInject}*sin(${angleInject}*${piBy180})
variable vYInject       equal 0
variable vZInject       equal -${vInject}*cos(${angleInject}*${piBy180})
variable neighborDist   equal ${coarseGrainingRatio}*${radmax}*0.05
variable timeStepMultiplicatorSpan      equal ${timeSpan}/${timeStepDEM}
variable timeStepMultiplicatorDump      equal ${timeSpan}/(${Dumps}*${timeStepDEM})
variable timeStepMultiplicatorPrint     equal ${timeStepMultiplicatorDump}
####################################


echo        both
coarsegraining ${coarseGrainingRatio}
atom_style      granular
boundary        f p f #walls
atom_modify     map array
newton          off
communicate     single vel yes

units           si
#processors      * * *

variable        minVolumeLimit equal 1e-40 #minimum individual particle limit
```

```
region          reg block -${LDomain} ${LDomain} -${yDepth} ${yDepth} ${zStart} ${zEnd} units box
create_box      7 reg #Must use 7 types since the wall must be a separate material!

neighbor        ${neighborDist} bin # nsq if too many neighbor atoms
neigh_modify exclude type 1 1 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 1 2 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 1 3 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 1 4 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 1 5 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 1 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 2 2 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 2 3 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 2 4 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 2 5 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 2 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 3 3 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 3 4 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 3 5 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 3 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 4 4 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 4 5 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 4 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 5 5 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 5 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 6 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
#DO NOT exlcute interactions with type 7 (wall type)

#Material properties required for new pair styles
fix             m1 all property/global youngsModulus peratomtype ${youngsModulus} ${youngsModulus} ${youngsModulus} ${youngsModulus} ${youngsModulus} ${youngsModulus} ${youngsModulus}
fix             m2 all property/global poissonsRatio peratomtype ${poissonsRatio} ${poissonsRatio} ${poissonsRatio} ${poissonsRatio} ${poissonsRatio} ${poissonsRatio} ${poissonsRatio}
fix             m3 all property/global coefficientRestitution peratomtypepair 7 ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR}
```

```
${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR}  ${coeR} ${coeR}  ${coeR}  ${coeR} ${coeR}
fix             m4 all property/global coefficientFriction peratomtypepair 7 ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF}

#pair style
pair_style gran model hertz      #Hertzian without cohesion
pair_coeff      * *
fix tscheck all check/timestep/gran 100 0.1 0.1 # warns if timestep exceeds Rayleigh or Hertz (fractioned) time

fix             gravi all gravity 9.81 vector 0.0 0.0 -1.0

timestep        ${timeStepDEM}

#wall
fix cad    all mesh/surface file ../DEM/wall.stl type 7
fix yWall  all wall/gran model hertz tangential none mesh n_meshes 1 meshes cad
fix myWall all wall/reflect/mesh mesh n_meshes 1 meshes cad  scaleUpFactor 7.5 coeffRestitution ${coeR}


#group particles according to their type (=size)
group group1 type 1
group group2 type 2
group group3 type 3
group group4 type 4
group group5 type 5
group group6 type 6

#particle distribution
fix      pts1 group1 particletemplate/sphere 1 atom_type 1 density constant ${rhoP} radius constant ${rad1} volume_limit ${minVolumeLimit}
fix      pts2 group2 particletemplate/sphere 1 atom_type 2 density constant ${rhoP} radius constant ${rad2} volume_limit ${minVolumeLimit}
fix      pts3 group3 particletemplate/sphere 1 atom_type 3 density constant ${rhoP} radius constant ${rad3} volume_limit ${minVolumeLimit}
fix      pts4 group4 particletemplate/sphere 1 atom_type 4 density constant ${rhoP} radius constant ${rad4} volume_limit ${minVolumeLimit}
fix      pts5 group5 particletemplate/sphere 1 atom_type 5 density constant ${rhoP} radius constant ${rad5} volume_limit ${minVolumeLimit}
fix      pts6 group6 particletemplate/sphere 1 atom_type 6 density constant ${rhoP} radius constant ${rad6} volume_limit ${minVolumeLimit}
fix      pdd1 all particledistribution/discrete 1 6 pts1 ${vfrac1} pts2 ${vfrac2} pts3 ${vfrac3} pts4 ${vfrac4} pts5 ${vfrac5} pts6 ${vfrac6}

#particle insertion
region  myInjet block -${LDomain} ${xInject} -${yDepth} ${yDepth} ${zInjectLo} ${zInjectHi} units box
fix ins all insert/rate/region seed 1001 distributiontemplate pdd1 mass ${mToInject} massrate ${mRate} overlapcheck no insert_every ${nStepInj} vel constant ${vXInject} ${vYInject} ${vZInject} region myInjet
```

```
#cfd coupling
fix             cfd  all couple/cfd couple_every 999999 mpi
fix             cfd2 all couple/cfd/force/integrateImp # MUST NOT use when using the integrateImp integrator!

#insert the particles
run 0


#screen output
compute         rke all erotate/sphere
compute         keAtom  all ke/atom
compute         keG all reduce sum c_keAtom
compute         centerMass  all com
thermo_style    custom step atoms c_keG c_centerMass[1] c_centerMass[2] c_centerMass[3]  pxx pyy pzz
thermo          ${timeStepMultiplicatorDump}
thermo_modify   format float %g lost ignore norm no
compute_modify  thermo_temp dynamic yes

#calculate particle mass and volume fraction for each group
variable        currTime equal step*${timeStepDEM}
compute         mall all property/atom mass
compute         m1 group1 property/atom mass
compute         m2 group2 property/atom mass
compute         m3 group3 property/atom mass
compute         m4 group4 property/atom mass
compute         m5 group5 property/atom mass
compute         m6 group6 property/atom mass
compute         smt all reduce sum c_mall
compute         sm1 group1 reduce sum c_mall
compute         sm2 group2 reduce sum c_mall
compute         sm3 group3 reduce sum c_mall
compute         sm4 group4 reduce sum c_mall
compute         sm5 group5 reduce sum c_mall
compute         sm6 group6 reduce sum c_mall
variable        vsmt equal c_smt
variable        vsm1 equal c_sm1
variable        vsm2 equal c_sm2
variable        vsm3 equal c_sm3
variable        vsm4 equal c_sm4
variable        vsm5 equal c_sm5
variable        vsm6 equal c_sm6
variable        n1 equal c_sm1/(${rhoP}*${VPart1})
variable        n2 equal c_sm2/(${rhoP}*${VPart2})
variable        n3 equal c_sm3/(${rhoP}*${VPart3})
variable        n4 equal c_sm4/(${rhoP}*${VPart4})
variable        n5 equal c_sm5/(${rhoP}*${VPart5})
variable        n6 equal c_sm6/(${rhoP}*${VPart6})
variable        nt equal c_sm1/(${rhoP}*${VPart1})+c_sm2/(${rhoP}*${VPart2})+c_sm3/(${rhoP}*${VPart3})+c_sm4/(${rhoP}*${VPart4})+c_sm5/(${rhoP}*${VPart5})+c_sm6/(${rhoP}*${VPart6})
variable        currPhiPt equal (c_smt/${rhoP})/${volRiser2D}
variable        currPhiP1 equal c_sm1/(c_smt+1e-64)
variable        currPhiP2 equal c_sm2/(c_smt+1e-64)
variable        currPhiP3 equal c_sm3/(c_smt+1e-64)
variable        currPhiP4 equal c_sm4/(c_smt+1e-64)
variable        currPhiP5 equal c_sm5/(c_smt+1e-64)
variable        currPhiP6 equal c_sm6/(c_smt+1e-64)
fix             phiPprint all print ${timeStepMultiplicatorPrint}
```

```
"${currTime} ${vsmt} ${nt} ${currPhiPt} ${n1} ${currPhiP1} ${n2} ${currPhiP2} ${n3} ${currPhiP3} ${n4} ${currPhiP4} ${n5} ${currPhiP5} ${n6} ${currPhiP6}" file currPhiP.dat screen no title currTime-massP-nTotal-phiPt-n1-phiP1-n2-phiP2-n3-phiP3-n4-phiP4-n5-phiP5-n6-phiP6
fix             massPprint all print ${timeStepMultiplicatorPrint} "${currTime} ${vsmt}  ${vsm1}  ${vsm2}  ${vsm3}  ${vsm4}  ${vsm5}  ${vsm6}" file currMass.dat screen no title currTime-massP-massP1-massP2-massP3-massP4-massP5-massP6

# calculate average velocity
variable        particleMomentumZ atom mass*vz
variable        myMass atom mass
compute         myMomentumPz all reduce sum v_particleMomentumZ
compute         totalMassP   all reduce sum v_myMass
variable        myMassPZVar equal c_totalMassP
variable        myMomentumPZVar equal c_myMomentumPz/(c_totalMassP+1e-99)
variable        currTime equal step*${timeStepDEM}
fix             printmyMomentum all print ${timeStepMultiplicatorPrint} "${currTime} ${myMassPZVar} ${myMomentumPZVar}" file mean.dat screen no

# calculate overlapping pairs in %
compute         PartDia all property/atom diameter
compute         minPartDia all reduce min c_PartDia
compute         myPair all pair/local dist
compute         myPairMin all reduce min c_myPair
variable        maxoverlap equal ((1.)-c_myPairMin/(c_minPartDia))*100
fix             reportOverlap all print ${timeStepMultiplicatorPrint} "${currTime} ${maxoverlap}" file reportOverlap.dat title "time maxoverlap[%]" screen no

#Dumps
dump            dmp  all custom ${timeStepMultiplicatorDump} ../DEM/post/dump*.part id type x y z vx vy vz fx fy fz radius f_Ksl f_uf[1] f_uf[2] f_uf[3] f_dragforce[1] f_dragforce[2] f_dragforce[3]

#SETTLE
restart         50000 jetRestart.1 jetRestart.2
run             1
```

9.2.4 3D Model

9.2.4.1 Meshing and Steady-State Gas Flow

The following printed file contents consist of code which has not been discussed in detail in the setup for the single-phase gas flow in the 3D riser model and are listed in alphabetical order. Geometry, mesh, ParaView state files and executables (*.stl, *.msh, *.pvsm, *(.sh)) have not been printed. The whole post-processed “riser_singlePFlow_22_full_AEE00_50to250s” case directories contents can be viewed on the attached disk.

system/controlDict

system/createPatchDict

system/fvSchemes

system/fvSolution

system/sampleDict

On the one hand, monitoring the field values by probing and sampling was done wrong since too much probes were set up, which generates lots of data. On the other hand, this does not deteriorate the results. Subsequent cases demonstrate the correct setup of the probes and the samples.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

libs
(
    "libOpenFOAM.so"
    "libincompressibleTurbulenceModel.so"
    "libincompressibleRASModels.so"
);

application     simpleFoam; //pimpleFoam for 'steady'-LES;

startFrom       latestTime; //startTime;

startTime       50;

stopAt          endTime;

endTime         250.;

deltaT          1e-2;

writeControl    adjustableRunTime;

writeInterval   1;

purgeWrite      20;

writeFormat     ascii;

writePrecision  6;

writeCompression compressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;
```

```

adjustTimeStep  yes;

maxCo           0.1;

x_inlet         10;
z_before_nozzle -5;     //acc. to paraview
z_nozzle        -1.7;   //acc. to paraview
z_after_nozzle   1.5;
z_jet            6.9;   //z at jet inlet
z_riser         16;     //acc. to paraview
z_head_hor      31;     //acc. to paraview
x_head_vert     -4.2;   //acc. to paraview
z_outlet        26;     //acc. to paraview

functions
{
    probes
    {
        // Where to load it from
        functionObjectLibs ( "libsampling.so" );

        type            probes;

        // Name of the directory for probe data
        name            probes;

        // Write at same frequency as fields
        outputControl   outputTime;
        outputInterval  1;

        // Fields to be probed
        fields
        (
            /*kMean*/ p U UMean /*voidfractionMean*/
        );

        probeLocations
        (
            //estimate time for steady-state by comparing calc. mean velocity with average velocity over probes
            //angle spins math. pos acc. to cross section in flow direction starting with 0 at the pos Y-coordinate
            (   $x_inlet   0    -9.59)   // approx. the middle of the inlet (-11.83 -2.8)..(-7.35 2.8)
            (   $x_inlet   1.4  -8.47)   // inlet (-11.83 -2.8)..(-7.35 2.8) angle=  45°
            (   $x_inlet  -1.4  -8.47)   // inlet (-11.83 -2.8)..(-7.35 2.8) angle= 135°
            (   $x_inlet  -1.4 -10.71)   // inlet (-11.83 -2.8)..(-7.35 2.8) angle=-135°
            (   $x_inlet   1.4 -10.71)   // inlet (-11.83 -2.8)..(-7.35 2.8) angle= -45°
            (   0     0    $z_before_nozzle)     // in the middle between baffles and nozzles, z acc. to paraview
            (   0     1.39 $z_before_nozzle)     // between baffles and nozzles r=2.78/2 angle=  0°
            (   1.39  0    $z_before_nozzle)     // between baffles and nozzles r=2.78/2 angle= 90°
            (   0    -1.39 $z_before_nozzle)     // between baffles and nozzles r=2.78/2 angle=180°
```

```
                    ( -1.39  0     $z_before_nozzle)     // between baffles and nozzles r=2.78/2 angle=-90°
                    (  0     0     $z_nozzle)    // approx. the middle of one of the nozzles, z acc. to paraview
                    (   .93  1.61 $z_nozzle)    // approx. the middle of one of the nozzles angle=  30°
                    (  1.86  0     $z_nozzle)    // approx. the middle of one of the nozzles angle=  90°
                    (   .93 -1.61 $z_nozzle)    // approx. the middle of one of the nozzles angle= 150°
                    (  -.93 -1.61 $z_nozzle)    // approx. the middle of one of the nozzles angle=-150°
                    ( -1.86  0     $z_nozzle)    // approx. the middle of one of the nozzles angle= -90°
                    (  -.93  1.61 $z_nozzle)    // approx. the middle of one of the nozzles angle= -30°
                    (  0     0     $z_after_nozzle)      // z acc. to paraview
                    (  1.11  1.93 $z_after_nozzle)      // distances proportional to nozzles distances angle=  30°
                    (  2.23  0     $z_after_nozzle)      // distances proportional to nozzles distances angle=  90°
                    (  1.11 -1.93 $z_after_nozzle)      // distances proportional to nozzles distances angle= 150°
                    ( -1.11 -1.93 $z_after_nozzle)      // distances proportional to nozzles distances angle=-150°
                    ( -2.23  0     $z_after_nozzle)      // distances proportional to nozzles distances angle= -90°
                    ( -1.11  1.93 $z_after_nozzle)      // distances proportional to nozzles distances angle= -30°
                    (  0     0     $z_jet)       // cross section near jet outlet, z acc. to paraview
                    (  0     2.29 $z_jet)       // r=6.66/2 angle=   0°
                    (  1.62  1.62 $z_jet)       // r=6.66/2 angle=  45°
                    (  2.29  0     $z_jet)       // r=6.66/2 angle=  90°
                    (  1.62 -1.62 $z_jet)       // r=6.66/2 angle= 135°
                    (  0    -2.29 $z_jet)       // r=6.66/2 angle= 180°
                    ( -1.62 -1.62 $z_jet)       // r=6.66/2 angle=-135°
                    ( -2.29  0     $z_jet)       // r=6.66/2 angle= -90°
                    ( -1.62  1.62 $z_jet)       // r=6.66/2 angle= -45°
                    (  0     0     $z_riser)     // in the middle middle of the big pipe, z acc. to paraview
                    (  0     2.29 $z_riser)     // big pipe r=4.57/2 angle=   0°
                    (  1.62  1.62 $z_riser)     // big pipe r=4.57/2 angle=  45°
                    (  2.29  0     $z_riser)     // big pipe r=4.57/2 angle=  90°
                    (  1.62 -1.62 $z_riser)     // big pipe r=4.57/2 angle= 135°
                    (  0    -2.29 $z_riser)     // big pipe r=4.57/2 angle= 180°
                    ( -1.62 -1.62 $z_riser)     // big pipe r=4.57/2 angle=-135°
                    ( -2.29  0     $z_riser)     // big pipe r=4.57/2 angle= -90°
                    ( -1.62  1.62 $z_riser)     // big pipe r=4.57/2 angle= -45°
                    (  0     0     $z_head_hor)  // middle of horizontal head section (quadratic l=8.128), z acc. to paraview
                    (  2.03  2.03 $z_head_hor)  // horizontal head section (quadratic l=8.128) angle=  45°
                    (  2.03 -2.03 $z_head_hor)  // horizontal head section (quadratic l=8.128) angle= 135°
                    ( -2.03 -2.03 $z_head_hor)  // horizontal head section (quadratic l=8.128) angle=-135°
                    ( -2.03  2.03 $z_head_hor)  // horizontal head section (quadratic l=8.128) angle=- 45°
                    (  $x_head_vert  0    36.3) // approx. the middle of the vertical head section (-4.1 32)..(4.1 40.6), x acc. to
```

```
                paraview
                (  $x_head_vert  2.05 38.5) // vertical head section (-4.1 32)..(4.1 40.6) angle=  45°
                (  $x_head_vert -2.05 38.5) // vertical head section (-4.1 32)..(4.1 40.6) angle= 135°
                (  $x_head_vert -2.05 34.1) // vertical head section (-4.1 32)..(4.1 40.6) angle=-135°
                (  $x_head_vert  2.05 34.1) // vertical head section (-4.1 32)..(4.1 40.6) angle=- 45°
                (-11.28  0    $z_outlet)    // approx. the middle of the outlet (-14.48 -4.1)..(-8.08 4.1), z acc. to paraview
                (-12.88  2.05 $z_outlet)    // approx. the middle of the outlet (-14.48 -4.1)..(-8.08 4.1) angle=  45°
                (-12.88 -2.05 $z_outlet)    // approx. the middle of the outlet (-14.48 -4.1)..(-8.08 4.1) angle= 135°
                ( -9.68 -2.05 $z_outlet)    // approx. the middle of the outlet (-14.48 -4.1)..(-8.08 4.1) angle=-135°
                ( -9.68  2.05 $z_outlet)    // approx. the middle of the outlet (-14.48 -4.1)..(-8.08 4.1) angle=- 45°
        );
    }

    fieldMinMaxU
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write yes;
        log yes;
        mode magnitude;
        fields
        (
            U
        );
    }

    fieldMinMaxP
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write yes;
        log yes;
        mode magnitude;
        fields
        (
            p
        );
    }

//     fieldMinMaxK
//     {
//         type fieldMinMax;
//         functionObjectLibs ("libfieldFunctionObjects.so");
//         write yes;
//         log yes;
//         mode magnitude;
//         fields
//         (
//             k
//         );
//     }

```

```
    fieldAverage1
    {
        type            fieldAverage;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   outputTime;

        fields
        (
//              k
//              {
//                  mean        on;
//                  prime2Mean  on;
//                  base        time;
//              }

            U
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }

//              voidfraction
//              {
//                  mean        on;
//                  prime2Mean  on;
//                  base        time;
//              }
        );
    }

}

// ************************************************************************* //
```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      createPatchDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Do a synchronisation of coupled points after creation of any patches.
// Note: this does not work with points that are on multiple coupled patches
//       with transformations (i.e. cyclics).
pointSync false;

// Patches to create. An empty patch list just removes patches with zero
// faces from $FOAM_CASE/constant/polyMesh/boundary.
patches
(
    {
        // Name of new patch
        name wall;

        // Dictionary to construct new patch from
        patchInfo
        {
            type wall;
        }

        // How to construct: either from 'patches' or 'set'
        constructFrom patches;

        // If constructFrom = patches : names of patches. Wildcards allowed.
        patches (inlet_rezi1 inlet_rezi2 wall.3 wall.4);

//          // If constructFrom = set : name of faceSet
//          set fin;
    }
);

// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState; //Euler; //steadyState; <-- start with steady!
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none; //Gauss linear;
    div(phi,U)      Gauss upwind; //limitedLinearV 1; //<-- start with upwind
    div(phi,k)      Gauss upwind; //limitedLinear 1; //<-- start with upwind
    div(phi,epsilon) Gauss upwind; //limitedLinear 1; //<-- start with upwind
    div(phi,omega)  Gauss upwind; //limitedLinear 1; //<-- start with upwind
    div(phi,R)      Gauss limitedLinear 1;
    div(R)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
    div((nu*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         none;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
```

```
    laplacian(DREff,R) Gauss linear corrected;
    laplacian(DnuTildaEff,nuTilda) Gauss linear corrected;
    laplacian(nu,U) Gauss linear corrected;
    laplacian(DomegaEff,omega) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
    interpolate(U)  linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
}


// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.01;
        smoother        GaussSeidel;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        maxIter         10;
    }

    pFinal
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0;
        smoother        GaussSeidel;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        maxIter         15;
    }

    "(U|k|epsilon|omega)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0.1;
    }

```

```
    "(U|k|epsilon|omega)Final"
    {
        $U;
        tolerance       1e-05;
        relTol          0;
    }
}

SIMPLE       //<-- start with simpleFoam solver!
{
    nNonOrthogonalCorrectors 1;
    pRefCell        0;
    pRefValue       0;
}

//Only relevant if Pimple-type solver used!!
PISO
{
    nCorrectors                 2;
    nNonOrthogonalCorrectors    1;
}
PIMPLE
{
    nOuterCorrectors            2;
}

relaxationFactors
{
    fields
    {
        p       0.8;
    }
    equations
    {
        "U.*"           0.2; //0.7;    //start with 0.1 here
        "k.*"           0.001; //1.0;    //start with 0.001 here
        "epsilon.*"     0.001; //0.1;    //start with 0.001 here
    }
}


// ************************************************************************* //

```

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.0                                   |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version         2.0;
    format          ascii;

    root            "/home/penfold/mattijs/foam/mattijs2.1/run/icoFoam";
    case            "cavity";
    instance        "system";
    local           "";

    class           dictionary;
    object          sampleDict;
}


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

pointCount 100; //number of sampling points


// interpolationScheme : choice of
//      cell            : use cell-centre value onlx; constant over cells
//      cellPoint       : use cell-centre and vertex values
//      cellPointFace   : use cell-centre, vertex and face values.
// 1] vertex values determined from neighbouring cell-centre values
// 2] face values determined using the current face interpolation scheme
//    for the field (linear, gamma, etc.)
interpolationScheme cellPointFace;


// writeFormat : choice of
//      xmgr
//      jplot
//      gnuplot
//      raw
setFormat       raw;


surfaceFormat       raw;
// sampling definition:
//
// Dictionary with fields
//      type : type of sampling method
//      name : name of samples. Used e.g. as filename
//      axis : how to write point coordinate
```

```
//      ...  : depending on method
//
//
// sample: choice of
//      uniform             evenly distributed points on line
//      face                one point per face intersection
//      midPoint            one point per cell, inbetween two face intersections
//      midPointAndFace     combination of face and midPoint
//
//      curve               specified points, not nessecary on line, uses
//                          tracking
//      cloud               specified points, uses findCell
//
//
// axis: how to write point coordinate. Choice of
// - x/y/z: x/y/z coordinate only
// - xyz: three columns
//  (probably does not make sense for anything but raw)
// - distance: distance from start of sampling line (if uses line) or
//              distance from first specified sampling point
//
// type specific:
//      uniform, face, midPoint, midPointAndFace : start and end coordinate
//      uniform: extra number of sampling points
//      curve, cloud: list of coordinates

z_outlet          26;    //acc. to paraview
x_min_outlet     -14.5;  //14478
x_max_outlet      -8;    //8077.2
x_head_vert       -4.2;  //acc. to paraview
z_min_head_vert   32;    //32.013525
z_max_head_vert   40.5;  //40.598725
z_head_hor        31;    //acc. to paraview
x_min_head_hor    -4.1;  //quadratic l=8.128
x_max_head_hor     4.1;  //quadratic l=8.128
z_riser           16;    //acc. to paraview
x_min_riser       -4.6;  //big pipe r=4.57
x_max_riser        4.6;  //big pipe r=4.57
z_jet              6.9;  //z at jet inlet
x_min_jet                $x_min_riser;
x_max_jet                $x_max_riser;
z_after_nozzle     1.5;
x_min_after_nozzle       -3.32;  //r=2.78+z_after_nozzle/tan(70.1°)
x_max_after_nozzle        3.32;  //r=2.78+z_after_nozzle/tan(70.1°)
z_nozzle          -1.7;  //acc. to paraview
x_min_nozzle      -2.78; //r=2.78
x_max_nozzle       2.78; //r=2.78
z_before_nozzle   -5;    //acc. to paraview
x_min_before_nozzle      $x_min_nozzle;
x_max_before_nozzle      $x_max_nozzle;
x_inlet           10;    //10337.8
z_min_inlet      -11.8;  //-11.750675
z_max_inlet       -7.3;  //-7.381875

sets
(
    line_inlet
    {
        type        uniform;
        name        line_inlet;
```

```
        axis        z;
        start       ($x_inlet 0 $z_min_inlet);
        end         ($x_inlet 0 $z_max_inlet);
        nPoints     $pointCount;
    }
    line_before_nozzle
    {
        type        uniform;
        name        line_before_nozzle;
        axis        x;
        start       ($x_min_before_nozzle 0 $z_before_nozzle);
        end         ($x_max_before_nozzle 0 $z_before_nozzle);
        nPoints     $pointCount;
    }
    line_nozzle
    {
        type        uniform;
        name        line_nozzle;
        axis        x;
        start       ($x_min_nozzle 0 $z_nozzle);
        end         ($x_max_nozzle 0 $z_nozzle);
        nPoints     $pointCount;
    }
    line_after_nozzle
    {
        type        uniform;
        name        line_after_nozzle;
        axis        x;
        start       ($x_min_after_nozzle 0 $z_after_nozzle);
        end         ($x_max_after_nozzle 0 $z_after_nozzle);
        nPoints     $pointCount;
    }
    line_jet
    {
        type        uniform;
        name        line_jet;
        axis        x;
        start       ($x_min_jet 0 $z_jet);
        end         ($x_max_jet 0 $z_jet);
        nPoints     $pointCount;
    }
    line_riser
    {
        type        uniform;
        name        line_riser;
        axis        x;
        start       ($x_min_riser 0 $z_riser);
        end         ($x_max_riser 0 $z_riser);
        nPoints     $pointCount;
    }
    line_head_hor
    {
        type        uniform;
        name        line_head_hor;
        axis        x;
        start       ($x_min_head_hor 0 $z_head_hor);
        end         ($x_max_head_hor 0 $z_head_hor);
        nPoints     $pointCount;
    }
    line_head_vert
    {
```

```
        type        uniform;
        name        line_head_vert;
        axis        z;
        start       ($x_head_vert 0 $z_min_head_vert);
        end         ($x_head_vert 0 $z_max_head_vert);
        nPoints     $pointCount;
    }
    line_outlet
    {
        type        uniform;
        name        line_outlet;
        axis        x;
        start       ($x_min_outlet 0 $z_outlet);
        end         ($x_max_outlet 0 $z_outlet);
        nPoints     $pointCount;
    }
);

surfaces
();


// Fields to sample.
fields
(
    //timeAverage_voidfraction
//    kMean
    UMean
//     UMean.component(1)
//     UMean.component(3)
    voidfractionMean
    //U.component(1)
    //R.component(0)
);


// ************************************************************************* //

```

9.2.4.2 Turbulent Gas Flow

From now on, the probes and the samples have been set up correctly. From the post-processed case “riser_singlePFlow_22_full_AEE00_50to250s”, the mesh has to be copied to constant/polyMesh and the contents of the folder holding its steady-state values have had to be copied to 0 holding the initial values. Geometry, mesh, ParaView state files and executables (*.stl, *.msh, *.pvsm, *(.sh)) have not been printed. The whole post-processed “riser_singlePFlow_22_full_AEE00_1EqEddy” case directories contents can be viewed on the attached disk.

CFD/system/controlDict

CFD/system/fvSolution

CFD/system/sampleDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

myOutputIntervall 30;

application     pimpleFoam; //pimpleFoam for 'steady'-LES;

startFrom       latestTime; //startTime;

startTime       0;

stopAt          endTime;

endTime         250.;

deltaT          0.1;

writeControl    adjustableRunTime;

writeInterval   1;

purgeWrite      100;

writeFormat     ascii;

writePrecision  6;

writeCompression compressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           0.5;

```

```
z_before_nozzle -6.07;  //acc. to paraview
z_nozzle        -1.7;   //acc. to paraview
z_jet            3.83;  //z at jet inlet
z_riser         16;     //acc. to paraview
x_head_vert     -4.2;   //acc. to paraview

libs ( "libfiniteVolumeCFDEM.so" );

functions
{
    probes
    {
        // Where to load it from
        functionObjectLibs ( "libsampling.so" );
        type            probes;
        // Name of the directory for probe data
        name            probes;
        // Write at same frequency as fields
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;

        // Fields to be probed
        fields
        (
            p U UMean k kMean /*voidfractionMean*/
        );

        probeLocations
        (
            //estimate time for steady-state by comparing calc. mean velocity with average velocity over probes
            //angle spins math. pos acc. to cross section in flow direction starting with 0 at the pos X-coordinate
            (  0     0    $z_nozzle)    // approx. the middle of one of the nozzles, z acc. to paraview
            (  1.84  0    $z_nozzle)    // approx. the middle of one of the nozzles angle=   0°
            (  0.92  1.59 $z_nozzle)    // approx. the middle of one of the nozzles angle=  60°
            ( -0.92  1.59 $z_nozzle)    // approx. the middle of one of the nozzles angle= 120°
            ( -1.84  0    $z_nozzle)    // approx. the middle of one of the nozzles angle= 180°
            ( -0.92 -1.59 $z_nozzle)    // approx. the middle of one of the nozzles angle=-120°
            (  0.92 -1.59 $z_nozzle)    // approx. the middle of one of the nozzles angle=- 60°
            (  0     0    $z_jet)       // cross section near jet outlet, z acc. to paraview
            ( -1.06  1.84 $z_jet)       // r=8.49/4 angle= 120°
            ( -1.06 -1.84 $z_jet)       // r=8.49/4 angle=-120°
            (  0     0    $z_riser)     // in the middle middle of the big pipe, z acc. to paraview
            (  $x_head_vert  0    36.3) // approx. the middle of the vertical head section (-4.1 32)..(4.1 40.6), x acc. to paraview
        );
    }

    outletAverage_1
    {
```

```
        type            faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        log                  yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        valueOutput     false;
        surfaceFormat   null;
        source               patch;
        sourceName      outlet;
        operation          areaAverage;
//         weightField        voidfraction;
        fields
        (
            U UMean k kMean//voidfractionMean
        );
    }

    fieldMinMaxU
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write yes;
        log yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode magnitude;
        fields
        (
            U
        );
    }

    fieldMinMaxP
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write yes;
        log yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode magnitude;
        fields
        (
            p
        );
    }

    fieldMinMaxK
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write yes;
        log yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode magnitude;
        fields
        (
            k
        );
    }
```

```

    fieldMinMaxKMean
    {
        type fieldMinMax;
        functionObjectLibs ("libfieldFunctionObjects.so");
        write yes;
        log yes;
        outputControl   timeStep;
        outputInterval  $myOutputIntervall;
        mode magnitude;
        fields
        (
            kMean
        );
    }

    fieldAverage1
    {
        type            fieldAverage;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   outputTime;

        fields
        (
            k
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }

            U
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }

//              voidfraction
//              {
//                  mean        on;
//                  prime2Mean  on;
//                  base        time;
//              }
        );
    }

}

// ************************************************************************* //

```

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

myMaxIterations          3;
myMaxIterationsPressure 10;

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-9;
        relTol          1e-9;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        maxIter         $myMaxIterationsPressure;
    }

    pFinal //Only relevant if Pimple-type solver used!!
    {
        solver          GAMG;
        tolerance       1e-9;
        relTol          0;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        maxIter         $myMaxIterationsPressure;
    }
```

```

    "(U|epsilon)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-012;
        relTol          0;
        maxIter         $myMaxIterations;
    }

    "(U|epsilon)Final" //Only relevant if Pimple-type solver used!!
    {
        $U
        tolerance       1e-12;
        relTol          0;
        maxIter         $myMaxIterations;
    }

   "(k|kFinal)"
    {
        $U
        tolerance       1e-14;
        relTol          0;
        maxIter         $myMaxIterations;
    }

}

// SIMPLE       //<-- start with simpleFoam solver!*/
// {
//     nNonOrthogonalCorrectors 1;
//     pRefCell        0;
//     pRefValue       0;
// }

//Only relevant if Pimple-type solver used!!
PISO
{
    nCorrectors                 2;
    nNonOrthogonalCorrectors    1;
}
PIMPLE
{
    nOuterCorrectors            2;
    nCorrectors                 1;
    nNonOrthogonalCorrectors    1;
}

//FINALLY DISABLE ALL RELAXATION FACTORS
relaxationFactors
{
    fields
    {
        p       0.8;
    }
    equations
    {
        "U.*"           0.7; //0.2; //0.7;    //start with 0.1 here
//        "k.*"           0.3; //1.0;    //start with 0.001 here
//        "epsilon.*"     0.3; //0.1;    //start with 0.001 here
    }
```

```
}



// ************************************************************************* //

```

```
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.0                                   |
|   \\  /    A nd           | Web:      http://www.openfoam.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version         2.0;
    format          ascii;

    root            "/home/penfold/mattijs/foam/mattijs2.1/run/icoFoam";
    case            "cavity";
    instance        "system";
    local           "";

    class           dictionary;
    object          sampleDict;
}


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

pointCount 100; //number of sampling points


// interpolationScheme : choice of
//      cell            : use cell-centre value onlx; constant over cells
//      cellPoint       : use cell-centre and vertex values
//      cellPointFace   : use cell-centre, vertex and face values.
// 1] vertex values determined from neighbouring cell-centre values
// 2] face values determined using the current face interpolation scheme
//    for the field (linear, gamma, etc.)
interpolationScheme cellPointFace;


// writeFormat : choice of
//      xmgr
//      jplot
//      gnuplot
//      raw
setFormat       raw;


surfaceFormat       raw;
// sampling definition:
//
// Dictionary with fields
//      type : type of sampling method
//      name : name of samples. Used e.g. as filename
//      axis : how to write point coordinate
```

```
//      ...  : depending on method
//
//
// sample: choice of
//      uniform             evenly distributed points on line
//      face                one point per face intersection
//      midPoint            one point per cell, inbetween two face intersections
//      midPointAndFace     combination of face and midPoint
//
//      curve               specified points, not nessecary on line, uses
//                          tracking
//      cloud               specified points, uses findCell
//
//
// axis: how to write point coordinate. Choice of
// - x/y/z: x/y/z coordinate only
// - xyz: three columns
//  (probably does not make sense for anything but raw)
// - distance: distance from start of sampling line (if uses line) or
//             distance from first specified sampling point
//
// type specific:
//      uniform, face, midPoint, midPointAndFace : start and end coordinate
//      uniform: extra number of sampling points
//      curve, cloud: list of coordinates

z_max            40.5;  //40.598725
x_head_vert      -4.2;  //acc. to paraview
z_min_head_vert  32;    //32.013525
z_max_head_vert         $z_max;
z_riser          16;    //acc. to paraview
x_min_riser      -4.6;  //big pipe r=4.57
x_max_riser       4.6;  //big pipe r=4.57
z_jet             3.83; //z at jet inlet
x_min_jet               $x_min_riser;
x_max_jet               $x_max_riser;
z_after_nozzle    1.5;
x_min_after_nozzle      -3.32;  //r=2.78+z_after_nozzle/tan(70.1°)
x_max_after_nozzle       3.32;  //r=2.78+z_after_nozzle/tan(70.1°)
z_nozzle         -1.7;  //acc. to paraview
x_min_nozzle     -2.78; //r=2.78
x_max_nozzle      2.78; //r=2.78
z_before_nozzle  -6.07; //acc. to paraview
x_min_before_nozzle     $x_min_nozzle;
x_max_before_nozzle     $x_max_nozzle;
z_min_inlet     -11.8;  //-11.750675
z_min                   $z_min_inlet;

sets
(
    line_axis
    {
        type        uniform;
        name        line_inlet;
        axis        z;
        start       (0 0 $z_min);
        end         (0 0 $z_max);
        nPoints     $pointCount;
    }
    line_before_nozzle
```

```
    {
        type            uniform;
        name            line_before_nozzle;
        axis            x;
        start           ($x_min_before_nozzle 0 $z_before_nozzle);
        end             ($x_max_before_nozzle 0 $z_before_nozzle);
        nPoints         $pointCount;
    }
    line_nozzle
    {
        type            uniform;
        name            line_nozzle;
        axis            x;
        start           ($x_min_nozzle 0 $z_nozzle);
        end             ($x_max_nozzle 0 $z_nozzle);
        nPoints         $pointCount;
    }
    line_after_nozzle
    {
        type            uniform;
        name            line_after_nozzle;
        axis            x;
        start           ($x_min_after_nozzle 0 $z_after_nozzle);
        end             ($x_max_after_nozzle 0 $z_after_nozzle);
        nPoints         $pointCount;
    }
    line_jet
    {
        type            uniform;
        name            line_jet;
        axis            x;
        start           ($x_min_jet 0 $z_jet);
        end             ($x_max_jet 0 $z_jet);
        nPoints         $pointCount;
    }
    line_riser
    {
        type            uniform;
        name            line_riser;
        axis            x;
        start           ($x_min_riser 0 $z_riser);
        end             ($x_max_riser 0 $z_riser);
        nPoints         $pointCount;
    }
    line_head_vert
    {
        type            uniform;
        name            line_head_vert;
        axis            z;
        start           ($x_head_vert 0 $z_min_head_vert);
        end             ($x_head_vert 0 $z_max_head_vert);
        nPoints         $pointCount;
    }
);

surfaces
();


// Fields to sample.
fields
```

```
(
    //timeAverage_voidfraction
    kMean
    UMean
//     UMean.component(1)
//     UMean.component(3)
    voidfractionMean
    //U.component(1)
    //R.component(0)
);


// ************************************************************************* //

```

9.2.4.3 Quenching

The code for the setup of the 2D riser simulation including quenching was discussed in detail since the settings of the case “JICF_1N_in3_out3_optimized_quench” have just had to be merged with those of the case “riser_singlePFlow_22_full_AEE00_1EqEddy”. Hence no file content has been printed. From the post-processed case “riser_singlePFlow_22_full_AEE00_50to250s”, the mesh has had to be copied to constant/polyMesh. From the post-processed case “riser_singlePFlow_22_full_AEE00_1EqEddy”, the contents of the folder holding its steady-state values have had to be copied to 0 holding the initial values. Geometry, mesh, ParaView state files and executables (*.stl, *.msh, *.pvsm, *(.sh)) have not been printed. The whole post-processed “riser_singlePFlow_22_full_AEE00_quench” case directories contents can be viewed on the attached disk.

9.2.4.4 Particle Injection

The following printed file contents consist of code which has not been discussed in detail in the setup for the particulate flow in the 3D riser model and are listed in alphabetical order. From the post-processed case “riser_singlePFlow_22_full_AEE00_50to250s”, the mesh has had to be copied to constant/polyMesh and the manually truncated boundary walls file wall_new.stl to DEM. From the post-processed case “riser_singlePFlow_22_full_AEE00_quench”, the contents of the folder holding its steady-state values have had to be copied to 0 holding the initial values. Files from the case “riser_PFlow_22_full_AEE00_largeDeltaT_elasticWall_CG3460_dragCorrection” have been printed since (i) the drag correction can be switched off simply by deleting the relevant line in CFD/system/couplingProperties, (ii) the case “riser_PFlow_22_full_AEE00_largeDeltaT_elasticWall_CG3460_slowQuench” can be realized by deleting the momentumQuench section in CFD/system/fvOptions, and (iii) the monodisperse cases can be realized by replacing the particle size distribution with the Sauter mean diameter in DEM/in.jet. Geometry, mesh, ParaView state files and executables (*.stl, *.msh, *.pvsm, *(.sh)) have not been printed. The post-processed cases directories contents can be viewed on the attached disk.

CFD/system/sliceData

DEM/in.jet

```
sliceCenterX
{
          type surfaces;
          functionObjectLibs ("libsampling.so");
          interpolationScheme cell;

          outputControl   timeStep;
          outputInterval  2000;

           surfaceFormat vtk;
           fields ( U UMean p voidfraction quenchMuLiq quenchMuVap quenchT quenchTMean);

           surfaces
           (
               slice_parallel_x
               {
                   type        cuttingPlane;
                   planeType   pointAndNormal;
                   pointAndNormalDict
                   {
                       basePoint    (0 0 0);
                       normalVector (1 0 0);
                   }
                   interpolate true;
               }
            );
}

sliceCenterY
{
          type surfaces;
          functionObjectLibs ("libsampling.so");
          interpolationScheme cell;

          outputControl   timeStep;
          outputInterval  2000;

           surfaceFormat vtk;
           fields ( U UMean p voidfraction quenchMuLiq quenchMuVap quenchT quenchTMean);

           surfaces
           (
               slice_parallel_y
               {
                   type        cuttingPlane;
                   planeType   pointAndNormal;
                   pointAndNormalDict
                   {
                       basePoint    (0 0 0);
                       normalVector (0 1 0);
                   }
                   interpolate true;
               }
            );
}

sliceCenterZ
{
          type surfaces;
```

```
        functionObjectLibs ("libsampling.so");
        interpolationScheme cell;

        outputControl   timeStep;
        outputInterval  2000;

         surfaceFormat vtk;
         fields ( U p voidfraction quenchMuLiq quenchMuVap quenchT );

         surfaces
         (
             slice_parallel_z
             {
                 type        cuttingPlane;
                 planeType   pointAndNormal;
                 pointAndNormalDict
                 {
                     basePoint    (0 0 3.6);
                     normalVector (0 0 1);
                 }
                 interpolate true;
             }

             slice_parallel_z10
             {
                 type        cuttingPlane;
                 planeType   pointAndNormal;
                 pointAndNormalDict
                 {
                     basePoint    (0 0 10);
                     normalVector (0 0 1);
                 }
                 interpolate true;
             }

             slice_parallel_z20
             {
                 type        cuttingPlane;
                 planeType   pointAndNormal;
                 pointAndNormalDict
                 {
                     basePoint    (0 0 20);
                     normalVector (0 0 1);
                 }
                 interpolate true;
             }

          );
}



```

```
## MAIN INPUT PARAMETERS ##
variable youngsModulus  equal 2.e6
variable poissonsRatio  equal 0.45
variable coeR           equal 0.9
variable coeF           equal 0.5
variable timeStepDEM    equal 60e-6
variable timeSpan       equal 300       #must result in time step multiplicators that are integers
variable Dumps          equal 2000      #must result in time step multiplicators that are integers

## Particle Size Distribution
variable d1 equal  5e-6 #type 1
variable d2 equal  12.5e-6 #type 2
variable d3 equal  17.5e-6 #type 3
variable d4 equal  22.5e-6 #type 4
variable d5 equal  27.5e-6 #type 5
variable d6 equal  35.0e-6 #type 6
variable dmax equal  ${d6}
variable d32  equal  6.8e-6
variable vfrac1 equal 0.62 #volume fraction of particles
variable vfrac2 equal 0.16
variable vfrac3 equal 0.10
variable vfrac4 equal 0.06
variable vfrac5 equal 0.03
variable vfrac6 equal 0.03
variable coarseGrainingRatio equal 3.46e3 #COARSE : 5.97e3, FINE: 3.46e3 #3.46e3 for 2e6 Parcels @ phiP=2.22e-3

## Geometry
variable wDomain        equal 0.532     #width of injection chute - must be equal to the height in y of  insertion_face.stl
variable dJet           equal ${wDomain}        #jet diameter
variable HInject        equal 0.748     #height of injection domain (approx. domain)
variable zInject1       equal 3.70      #height of injection - must be equal to the position in y of  insertion_face.stl
variable zInject2       equal ${zInject1}       #height of injection - must be equal to the position in y of  insertion_face.stl
variable zStart         equal -11.8     #beginning of simulation domain
variable zEnd           equal 40.5      #end of simulation domain
variable LDomain        equal 15        #half length of simulation domain little larger for outlet
variable LInject        equal 0.61      #length of recirculate inlet; riser radius is 4.61
variable yDepth         equal 4.61      #half width of simulation domain little larger than riser diameter
variable volRiser3D     equal 3.87e3    #

## Riser Operating Parameters
variable vInject        equal 2 #vertical injection velocity (of particles
variable mRate          equal 77.8      #1.55e2/2 for one inlet; keep mass load constant (mInject/mInjectFluid)
variable angleInjectZ1  equal 83        #inclination angle from z-direction of a particle injection
variable angleInjectX1  equal 120       #inclination angle from x-direction of a particle injection
variable angleInjectZ2  equal 83        #inclination angle from z-direction of a particle injection
variable angleInjectX2  equal -120      #inclination angle from x-direction of a particle injection
```

```
variable rhoP           equal 2250      #particle density

## Constants
variable piBy4          equal 0.78540   #constant
variable pi43           equal 4.1888    #constant
variable piBy180        equal 0.017453  #constant

## INPUT CALCULATIONS ##
variable rad1   equal   ${d1}/2.
variable rad2   equal   ${d2}/2.
variable rad3   equal   ${d3}/2.
variable rad4   equal   ${d4}/2.
variable rad5   equal   ${d5}/2.
variable rad6   equal   ${d6}/2.
variable radmax equal   ${dmax}/2.
variable VPart1 equal ${pi43}*${rad1}*${rad1}*${rad1}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable VPart2 equal ${pi43}*${rad2}*${rad2}*${rad2}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable VPart3 equal ${pi43}*${rad3}*${rad3}*${rad3}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable VPart4 equal ${pi43}*${rad4}*${rad4}*${rad4}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable VPart5 equal ${pi43}*${rad5}*${rad5}*${rad5}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable VPart6 equal ${pi43}*${rad6}*${rad6}*${rad6}*${coarseGrainingRatio}*${coarseGrainingRatio}*${coarseGrainingRatio}
variable zHeight        equal ${zEnd}-${zStart}         #height of simulation domain
variable zInjectLo1     equal ${zInject1}-${HInject}/2  #bottom of injection region
variable zInjectHi1     equal ${zInject1}+${HInject}/2  #bottom of injection region
variable zInjectLo2     equal ${zInject2}-${HInject}/2  #bottom of injection region
variable zInjectHi2     equal ${zInject2}+${HInject}/2  #bottom of injection region
variable xInjectOut1    equal ${yDepth}*cos(${angleInjectX1}*${piBy180}) #inside end of injection region
variable yInjectOut1    equal ${yDepth}*sin(${angleInjectX1}*${piBy180}) #inside end of injection region
variable xInjectIn1     equal (${yDepth}-${LInject})*cos(${angleInjectX1}*${piBy180}) #inside end of injection region
variable yInjectIn1     equal (${yDepth}-${LInject})*sin(${angleInjectX1}*${piBy180}) #inside end of injection region
variable xInjectOut2    equal ${yDepth}*cos(${angleInjectX2}*${piBy180}) #inside end of injection region
variable yInjectOut2    equal ${yDepth}*sin(${angleInjectX2}*${piBy180}) #inside end of injection region
variable xInjectIn2     equal (${yDepth}-${LInject})*cos(${angleInjectX2}*${piBy180}) #inside end of injection region
variable yInjectIn2     equal
```

```
(${yDepth}-${LInject})*sin(${angleInjectX2}*${piBy180}) #inside end of injection region
variable tInject        equal ${timeSpan}       #99     #time interval for injection
variable mToInject      equal ${tInject}*${mRate}
variable nStepInj       equal ${coarseGrainingRatio}*${radmax}/${vInject}/${timeStepDEM}
variable vXInject1      equal -${vInject}*sin(${angleInjectZ1}*${piBy180})*cos(${angleInjectX1}*${piBy180})
variable vYInject1      equal -${vInject}*sin(${angleInjectZ1}*${piBy180})*sin(${angleInjectX1}*${piBy180})
variable vZInject1      equal -${vInject}*cos(${angleInjectZ1}*${piBy180})
variable vXInject2      equal -${vInject}*sin(${angleInjectZ2}*${piBy180})*cos(${angleInjectX2}*${piBy180})
variable vYInject2      equal -${vInject}*sin(${angleInjectZ2}*${piBy180})*sin(${angleInjectX2}*${piBy180})
variable vZInject2      equal -${vInject}*cos(${angleInjectZ2}*${piBy180})
variable neighborDist   equal ${radmax}*1.0 #MUST NOT consider coarsegrainingratio, will be done automatically!
variable timeStepMultiplicatorSpan      equal ${timeSpan}/${timeStepDEM}
variable timeStepMultiplicatorDump      equal ${timeSpan}/(${Dumps}*${timeStepDEM})
variable timeStepMultiplicatorPrint     equal ${timeStepMultiplicatorDump}
####################################


echo        both
coarsegraining ${coarseGrainingRatio}
atom_style      granular
boundary        f f f #walls
atom_modify     map array
newton          off
communicate     single vel yes

units           si
processors      2 2 *

variable        minVolumeLimit equal 1e-40 #minimum individual particle limit
region          reg block -14.5 10.4 -7.5 7.5 -0.51 40.7 units box
create_box      7 reg #Must use 7 types since the wall must be a separate material!

neighbor        ${neighborDist} bin # nsq if too many neighbor atoms
neigh_modify delay 0
neigh_modify one  1000
#neigh_modify page 600000
neigh_modify exclude type 1 1 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 1 2 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 1 3 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 1 4 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 1 5 #do not perform collision tracking for situations with phiP < 1 Vol%!
```

```
neigh_modify exclude type 1 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 2 2 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 2 3 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 2 4 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 2 5 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 2 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 3 3 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 3 4 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 3 5 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 3 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 4 4 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 4 5 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 4 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 5 5 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 5 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
neigh_modify exclude type 6 6 #do not perform collision tracking for situations with phiP < 1 Vol%!
#DO NOT exlcute interactions with type 7 (wall type)

#Material properties required for new pair styles
fix             m1 all property/global youngsModulus peratomtype ${youngsModulus} ${youngsModulus} ${youngsModulus} ${youngsModulus} ${youngsModulus} ${youngsModulus} ${youngsModulus}
fix             m2 all property/global poissonsRatio peratomtype ${poissonsRatio} ${poissonsRatio} ${poissonsRatio} ${poissonsRatio} ${poissonsRatio} ${poissonsRatio} ${poissonsRatio}
fix             m3 all property/global coefficientRestitution peratomtypepair 7 ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR} ${coeR}  ${coeR} ${coeR}  ${coeR}  ${coeR} ${coeR}
fix             m4 all property/global coefficientFriction peratomtypepair 7 ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF} ${coeF}

#pair style
pair_style gran model hertz      #Hertzian without cohesion
pair_coeff      * *
fix tscheck all check/timestep/gran 100 0.1 0.1 # warns if timestep exceeds Rayleigh or Hertz (fractioned) time
```

```

fix             gravi all gravity 9.81 vector 0.0 0.0 -1.0

timestep        ${timeStepDEM}

#wall
fix cad    all mesh/surface file ../DEM/wall_new.stl type 7 heal auto_remove_duplicates curvature 1e-6
fix yWall  all wall/gran model hertz mesh n_meshes 1 meshes cad
fix myWall all wall/reflect/mesh mesh n_meshes 1 meshes cad  scaleUpFactor 5.0 coeffRestitution ${coeR}

#Outlets
fix outlet     all mesh/surface file ../DEM/outlet.stl type 7
fix massoutlet all massflow/mesh mesh outlet vec_side 0. 0. -1 count once file ../DEM/massoutlet.dat  delete_atoms yes screen no

fix outletBottom     all mesh/surface file ../DEM/outletBottom.stl type 7
fix massoutletBottom all massflow/mesh mesh outletBottom vec_side 0. 0. -1 count once file ../DEM/massoutletBottom.dat  delete_atoms yes screen no

#group particles according to their type (=size)
group group1 type 1
group group2 type 2
group group3 type 3
group group4 type 4
group group5 type 5
group group6 type 6

#particle distribution
fix      pts1 group1 particletemplate/sphere 1 atom_type 1 density constant ${rhoP} radius constant ${rad1} volume_limit ${minVolumeLimit}
fix      pts2 group2 particletemplate/sphere 1 atom_type 2 density constant ${rhoP} radius constant ${rad2} volume_limit ${minVolumeLimit}
fix      pts3 group3 particletemplate/sphere 1 atom_type 3 density constant ${rhoP} radius constant ${rad3} volume_limit ${minVolumeLimit}
fix      pts4 group4 particletemplate/sphere 1 atom_type 4 density constant ${rhoP} radius constant ${rad4} volume_limit ${minVolumeLimit}
fix      pts5 group5 particletemplate/sphere 1 atom_type 5 density constant ${rhoP} radius constant ${rad5} volume_limit ${minVolumeLimit}
fix      pts6 group6 particletemplate/sphere 1 atom_type 6 density constant ${rhoP} radius constant ${rad6} volume_limit ${minVolumeLimit}
fix      pdd1 all particledistribution/discrete 1 6 pts1 ${vfrac1} pts2 ${vfrac2} pts3 ${vfrac3} pts4 ${vfrac4} pts5 ${vfrac5} pts6 ${vfrac6}

#particle insertion
region  injet3 sphere -1.8 -3.1 ${zInject2} 0.2 units box
fix ins1 all insert/rate/region seed 1001 distributiontemplate pdd1 mass ${mToInject} massrate ${mRate} overlapcheck no insert_every ${nStepInj} vel constant ${vXInject2} ${vYInject2} ${vZInject2} region injet3

region  injet4 sphere -1.8 3.1 ${zInject2} 0.2 units box
fix ins2 all insert/rate/region seed 1001 distributiontemplate pdd1 mass ${mToInject} massrate ${mRate} overlapcheck no insert_every ${nStepInj} vel constant ${vXInject1} ${vYInject1} ${vZInject1} region injet4


#cfd coupling
fix             cfd  all couple/cfd couple_every 999999 mpi
fix             cfd2 all couple/cfd/force/integrateImp # MUST NOT use an INTEGRATION FIX when using the integrateImp integrator!
```

```

#insert the particles
run 0


#screen output
compute         rke all erotate/sphere
compute         keAtom  all ke/atom
compute         keG all reduce sum c_keAtom
compute         centerMass  all com
thermo_style    custom step atoms c_keG c_centerMass[1] c_centerMass[2] c_centerMass[3]  pxx pyy pzz
thermo          ${timeStepMultiplicatorDump}
thermo_modify   format float %g lost ignore norm no
compute_modify  thermo_temp dynamic yes

#calculate particle mass and volume fraction for each group
variable        currTime equal step*${timeStepDEM}
compute         mall all property/atom mass
compute         m1 group1 property/atom mass
compute         m2 group2 property/atom mass
compute         m3 group3 property/atom mass
compute         m4 group4 property/atom mass
compute         m5 group5 property/atom mass
compute         m6 group6 property/atom mass
compute         smt all reduce sum c_mall
compute         sm1 group1 reduce sum c_mall
compute         sm2 group2 reduce sum c_mall
compute         sm3 group3 reduce sum c_mall
compute         sm4 group4 reduce sum c_mall
compute         sm5 group5 reduce sum c_mall
compute         sm6 group6 reduce sum c_mall
variable        vsmt equal c_smt
variable        vsm1 equal c_sm1
variable        vsm2 equal c_sm2
variable        vsm3 equal c_sm3
variable        vsm4 equal c_sm4
variable        vsm5 equal c_sm5
variable        vsm6 equal c_sm6
variable        n1 equal c_sm1/(${rhoP}*${VPart1})
variable        n2 equal c_sm2/(${rhoP}*${VPart2})
variable        n3 equal c_sm3/(${rhoP}*${VPart3})
variable        n4 equal c_sm4/(${rhoP}*${VPart4})
variable        n5 equal c_sm5/(${rhoP}*${VPart5})
variable        n6 equal c_sm6/(${rhoP}*${VPart6})
variable        nt equal c_sm1/(${rhoP}*${VPart1})+c_sm2/(${rhoP}*${VPart2})+c_sm3/(${rhoP}*${VPart3})+c_sm4/(${rhoP}*${VPart4})+c_sm5/(${rhoP}*${VPart5})+c_sm6/(${rhoP}*${VPart6})
variable        currPhiPt equal (c_smt/${rhoP})/${volRiser3D}
variable        currPhiP1 equal c_sm1/(c_smt+1e-64)
variable        currPhiP2 equal c_sm2/(c_smt+1e-64)
variable        currPhiP3 equal c_sm3/(c_smt+1e-64)
variable        currPhiP4 equal c_sm4/(c_smt+1e-64)
variable        currPhiP5 equal c_sm5/(c_smt+1e-64)
variable        currPhiP6 equal c_sm6/(c_smt+1e-64)
fix             phiPprint all print ${timeStepMultiplicatorPrint} "${currTime} ${vsmt} ${nt} ${currPhiPt} ${n1} ${currPhiP1} ${n2} ${currPhiP2} ${n3} ${currPhiP3} ${n4} ${currPhiP4} ${n5} ${currPhiP5} ${n6} ${currPhiP6}" file currPhiP.dat screen no title currTime-massP-nTotal-phiPt-n1-phiP1-n2-phiP2-n3-phiP3-n4-phiP4-n5-phiP5-n6-phiP6
```

```
fix             massPprint all print ${timeStepMultiplicatorPrint} "${currTime} ${vsmt}  ${vsm1}  ${vsm2}  ${vsm3}  ${vsm4}  ${vsm5}  ${vsm6}" file currMass.dat screen no title currTime-massP-massP1-massP2-massP3-massP4-massP5-massP6

# calculate average velocity
variable        particleMomentumZ atom mass*vz
variable        myMass atom mass
compute         myMomentumPz all reduce sum v_particleMomentumZ
compute         totalMassP   all reduce sum v_myMass
variable        myMassPZVar equal c_totalMassP
variable        myMomentumPZVar equal c_myMomentumPz/(c_totalMassP+1e-99)
variable        currTime equal step*${timeStepDEM}
fix             printmyMomentum all print ${timeStepMultiplicatorPrint} "${currTime} ${myMassPZVar} ${myMomentumPZVar}" file mean.dat screen no

# calculate overlapping pairs in %
compute         PartDia all property/atom diameter
compute         minPartDia all reduce min c_PartDia
compute         myPair all pair/local dist
compute         myPairMin all reduce min c_myPair
variable        maxoverlap equal ((1.)-c_myPairMin/(c_minPartDia))*100
fix             reportOverlap all print ${timeStepMultiplicatorPrint} "${currTime} ${maxoverlap}" file reportOverlap.dat title "time maxoverlap[%]" screen no

#Dumps
dump            dmp  all custom ${timeStepMultiplicatorDump} ../DEM/post/dump*.part id type x y z vx vy vz fx fy fz radius f_Ksl f_uf[1] f_uf[2] f_uf[3] f_dragforce[1] f_dragforce[2] f_dragforce[3]

#SETTLE
restart         50000 jetRestart.1 jetRestart.2
run             1
```