University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Construction and Materials, specialization within Mechanical Construction	Spring semester, 2016 Open
Writer: Thomas Engen	………………………………………… (Writer's signature)
Faculty supervisor: Bjørn Helge Hjertager External supervisor(s):	
Thesis title: CFD analysis of gas-particle flow in a scaled circulating fluidized bed	
Credits (ECTS): 30	
Key words: OpenFOAM twoPhaseEulerFoam CFD Circulating fluidized bed Multiphase flow	Pages: 56 + enclosure: 24 + memory stick Stavanger, 29.06.2016

Abstract

The hydrodynamics of gas-particle flow in a Circulating Fluidized Bed (CFB) system have been studied computationally. The numerical predictions are evaluated against experimental testing done on a 1/9th scale cold CFB boiler at Chalmers University, in 2001. The numerical model is created using the open-source CFD code OpenFOAM.

The simulations were done using two different approaches for how the flow is driven through the CFB system. First the simulations were run with a uniform plug velocity for the superficial gas velocity. This was done to see if OpenFOAM would generate the same type of turbulent regime that has been seen occurring in similar CFD codes created by previous researchers. The second approach was to set the superficial gas velocity to be calculated based on the pressure flux at the inlet boundary.

Numerical results of the pressure drop, solid concentration distribution in the riser, mean velocity and fluctuating velocity were taken for all the simulations. The numerical code is a two phase model. This was selected due to the fact that it has the capability of using the kinetic theory of granular flow. Previously work on this experimental setup were lacking when it came to predicting the correct pressure drop and solid volume concentration along the riser height.

For simulations that ran a superficial gas velocity calculated by the pressure flux the dense bottom bed were predicted. The pressure drop along the riser height compared well with experimental data when using a representative particle size of 70 µm. The velocity profiles is predicted to follow the same trends as the experimental results, but they are sensitive to the particle size. For particle size of 45 µm the axial velocity is generally overestimated, and for 70 µm the axial velocity is underestimated towards the walls, but fits well in the center of the riser.

Based on the validation against experimental data, it is assumed that the obtained results are physically correct for the case simulated with a calculated pressure flux for the superficial gas velocity. The experiment is done with a wide particle size distribution, so modeling the gas-particle flow with only one mean particle velocity will influence the comparability of the results.

Contents

List of Figures

List of Tables

Nomenclature

Latin symbols

A	Surface area	[m²]
C_D	Drag coefficient	[-]
d_s	Particle diameter	[m]
e	Restitution coefficient	[-]
e’	Internal energy	[]
F_D	Drag force	[N/m³]
F_p	Buoyancy force	[N/m³]
Fr	Frictional pressure module	[Pa]
f	Drag factor	[-]
g	Gravitational acceleration	[m/s²]
g_0	Radial distribution function	[-]
G	Relative velocity	[m/s]
I	Moment of inertia	[-]
I	Unity tensor	[-]
J	Impulsive force	[N]
K_{sg}	Drag coefficient between solid and gas phase	[-]
m	Mass	[kg]
n	Normal vector	[-]
p	Pressure	[Pa]
q	Heat flux	[J/s]
r	Radius	[m]
t	Time	[s]
S	Face-normal vector	[m²]
u_g	Gas velocity	[m/s]
u_s	Solid velocity	[m/s]
v	Translation velocity	[m/s]

Greek symbols

α_g	Gas volume fraction	[-]
α_s	Solid volume fraction	[-]
β_A	Friction coefficient	[-]
γ_i	Energy dissipation	[kg/ms³]
Θ_i	Granular temperature	[m²/s²]
κ_i	Diffusion coefficient of granular energy	[kg/ms]
λ_i	Bulk viscosity	[kg/ms]
μ_i	Shear viscosity	[kg/ms]
ν_i	Kinematic viscosity	[m²/s]
ρ_i	Density	[kg/m³]
τ_i	Stress tensor	[N/m²]
ϕ	Angle of internal friction	[-]

φ_i	Phase velocity flux	[m³/s²]
ω	Angular velocity	[-]
ζ_i	Discrete inverse diagonal coefficient	[s]

Subscripts

i,j	General index
g	Gas
s	Solid
w	Wall

1 Introduction

1.1 Background

Circulating fluidized beds (CFB) is the topic of this thesis. The application of CFB systems have increased the last decades from being used to mainly catalytic cracking, to power generation and drying of wet powders.

For now the pulverized coal combustors (PCC) is the dominant technology for the power generation sector, only a minority of the plant commissioned is based on the circulating fluidized bed combustion (CFBC) technology. One of the biggest advantages of the CFB combustors oppose to pulverized coal combustors is the utilization of low quality coal, flexibility regarding fuel, and emission performance of NO_x and SO_2. A disadvantage of the CFB is that there is possibility of small particle emissions, therefore it is necessary to use energy into separating the particles from the outflow [1].

The CFB technology have been used for power generation for over 25 years and it is still evolving. Most of the CFBC units used today is small in size, with a delivery of 330 MW, while the PCC units deliver up to 1000 MW. The lack of good design tools for scaling up the process is one of the reasons for the low power delivery [2].

Computational fluid dynamics (CFD) has proven that it is capable of obtaining precise predictions for single phase flows. The CFD models for multiphase flow simulations still needs improvements before it can be seen as reliable as single phase flow simulations.

This thesis will evaluate a multiphase CFD code using OpenFOAM. The purpose will be to point out what specific area that need improvement, and come with recommendations for improvements.

1.2 Objective

The objective of this thesis is to evaluate the possibility of using the CFD open source program OpenFOAM to simulated multiphase gas-particle flow. The software have been greatly approved over the last 10 years. The foundation for evaluating the codes used in OpenFOAM will be experimental data from the 1/9th scale cold CFB boiler at Chalmers University. The main focus of the thesis is to create a code that is able to predict the correct pressure drop and solid volume concentration along the riser height, to see what impact different particle sizes have on the two phase numerical model. And to find out what diameter size gives the best result. The results gather in this thesis will hopefully provide a basis for further discussions of the use of two phase numerical simulations for circulating fluidized beds.

1.3 Previous work

A great deal of research have been done when it comes to simulation of multiphase gas-solid models. Most work have been executed using 2-dimensional models, due to the amount of computer power required for 3-dimensional analysis. During the literature search it became clear that most work had been done on simulating bubbling fluidized beds. In a bubbling flow the particle packing is generally high, and turbulent dilute flow regimes does not exist. The results from these types of flows have given good results, one reason for this might be that the flow pattern does not change much in the riser.

The research done by Ibsen is on the 1/9th scale cold CFB boiler. The numerical simulations were done by using kinetic theory of granular flow and LES for turbulence modeling of the solid and gas phase, respectively. This give a good description of the random motion of the particle and behavior of the gas phase. Using the mean volume length scale to determine the diameter size for the particle gave a satisfying velocity profile, but the flux and pressure drop in the riser were far from the experimental data [1].

Hansen, Madsen [3] simulated the behavior of the same boiler with k-e turbulence model for the gas phase, and either k-e turbulence model or fixed particle viscosity for the solid phase. They manage to simulate the overall behavior of the circulation of the particles, something that was not done in the work by Ibsen. The axial velocity profiles for the solid phase did not fit well with the experimental data, but they concluded that fixed particle viscosity of the solid phase gave better agreement than using a k-e turbulence model [3].

Hansen, Solberg and Hjertager [4] did a study on distribution of particles in a scaled CFB boiler. Previous work with one solid phase using a mean diameter did not give satisfying results for the pressure drop. Their solution to solve this was to use more particle phases. This way the particle with high diameter would stay in the lower part of the riser while particles with lower diameter would circulate. These high diameter particle would then create a steeper pressure gradient at the bottom of the riser [4].

1.4 Thesis layout

This thesis is divided into 8 chapters. Theory regarding fluidization, fluid-particle, particle-particle and kinetic theory of granular flow is found in chapter 2.

The governing equations is found in chapter 3. Chapter 4 gives description of the solver, numerical discretization schemes, solution controls and turbulence modelling theory.

Chapter 5 gives a description of the experimental setup of the scaled CFB boiler.

Chapter 6 gives a description of the modelling approach. This includes information of the mesh, boundary conditions, modelling of the turbulence and the phase properties etc. Chapter 7 contains the result and discussion, and the conclusion is given in chapter 8.

2 Theory

In this chapter the principle of a CFB is explained together with the basic hydrodynamics that occur in a fluidized bed. A more detailed description is given for the kinetic theory of granular flow, due to the importance of its implementation in the numerical model.

2.1 Principle of the CFB

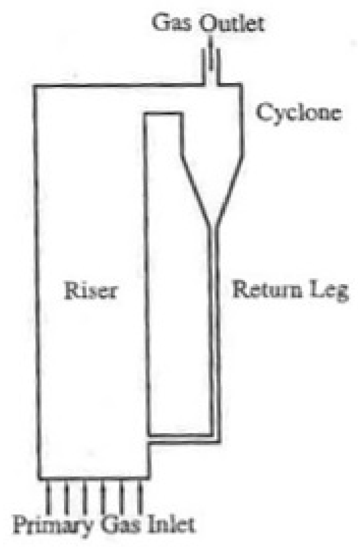


Figure 1 Sketch of a CFB system [1]

In the simplest way a CFB system consist of three functional objects, a riser, a cyclone and a return leg. Figure 1 shows a sketch of CFB system. Particles are introduced to the riser, creating a particle bed in the bottom of the riser. The particle bed is kept in free motion by an upward flowing gas, or liquid, hence the term “fluidized bed”. By increasing the velocity of the upwards flowing gas the particles will rise farther up the riser. When the velocity is high enough the particles will exit the riser and enter the cyclone. In the cyclone the particles gets separated from the outflowing gas, and fall down the return leg. The return leg does as the name suggests, it returns the particle to the bottom of the riser, hence the term “circulating” [1].

In this thesis different types of flows will be described, Figure 2 shows how the superficial gas velocity will influence what type of flow regime occur in the riser.

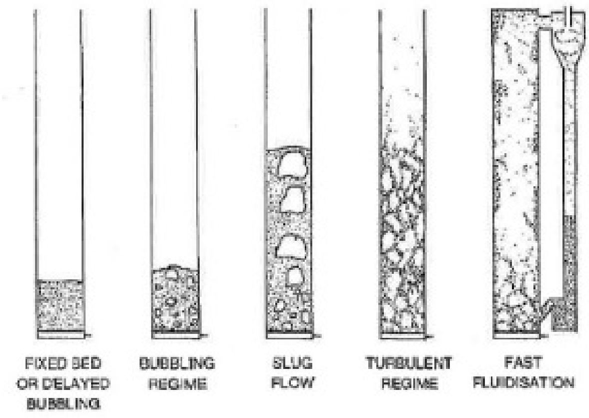


Figure 2 Fluidization regimes with increasing velocity [1]

2.2 Fluidization for gas-solid flow

Fluidization is what occurs when gas, or liquid, is forced vertically through a packed particle bed, with a flow rate great enough to support the weight of the particle by the drag force imposed by the fluid. When the particles are supported by the drag they get a fluid like appearance, and some of the properties of a fluid, hence the term fluidization [5].

When the fluid is forced through the packed particle bed the flow will cause a pressure drop across the particle bed. The pressure drop can be described by the Ergun equation. When fluidization occurs, the pressure drop is high enough to support the weight of the particle, this is known as the state of minimum fluidization. This minimum fluidization state will then have a corresponding minimum fluidization velocity [6]

To estimate the minimum fluidization a moment balance equation for the buoyancy to equal the drag on the particle can be derived. If the gas-wall friction and the solid stresses transmitted by the particle is ignored the balance equation can be written as [6]:

$$(1-\alpha_g)(\rho_p-\rho_g)g=\frac{\beta_A}{\alpha_g}(u_g-u_s) \tag{1}$$

From the Ergun equation (1952) the friction coefficient β_A can be obtained for dense beds

$$\beta_A=150\frac{(1-\alpha_g)^2\mu_g}{\alpha_g(d_s\vartheta)^2}+1.75\frac{\rho_g|u_g-u_s|(1-\alpha_g)}{d_s\vartheta} \tag{2}$$

Where is $\alpha_g$the gas volume fraction, ρ_g and ρ_s is the gas and particle densities respectively, g is the gravitational acceleration force, u_g and u_s is the gas and particle velocities respectively, d_s is the particle diameter, μ_g is the viscosity of the gas phase, and ϑ is the spherity of the particle. The spherity of a particle is defined as the ratio of the surface area of a sphere with the same volume as the particle, and the surface area of the particle [6].

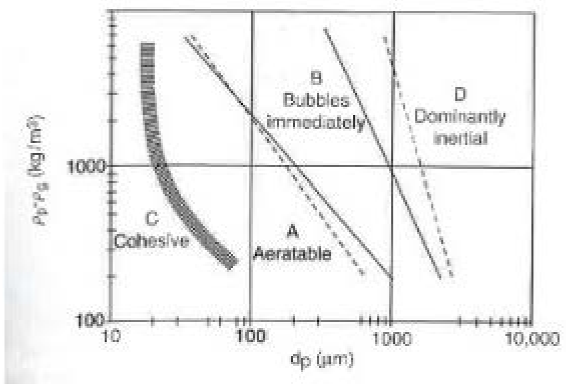


Figure 3 Geldart powder groups [5]

Geldart (1973) created a classification scheme for particles fluidized by air in atmospheric condition. The scheme divided the fluidization into four powder groups, in order to incorporate the broad behavior of particles with different densities and diameter size. The Geldart diagram is shown in Figure 3 [5]

The particles used in the 1/9 scale CFB boiler is made out of bronze, so the density difference between the gas and particles is very high. Based on the mean diameter of the particle size distribution, $60\mu m$, the particles in this thesis fit group B on Geldart's diagram. Most of the characteristics described for flows within group B is for particles with densities 1400kg/m^3 to 4000kg/m3. Based on the experimental data represented in by Ibsen [1], the expansion of the bed is small and it happens very fast. This description fits for particles within group B. The high density difference makes it so that behavior like spouting will occur in the bed, resulting in low flux with high velocities out of the riser. This type of behavior is not mentioned for powder group B, so expecting results based on powder group seems difficult for the experimental case looked at in this thesis.

2.3 Fluid-particle interaction

The movement of a particle interacting with a fluid can be described by combining four types of simple motion of a sphere, with the assumption that the particle does not interact with other particles [7]. (1) A particle moving with a constant velocity in a uniform flow field, (2) a particle accelerating in a uniform flow field, (3) a particle moving with a constant velocity in a non-uniform flow field, and (4) a particle rotating with a constant angular velocity in a uniform flow field. The four types of simple motions represents the drag force, Basset force, Staffman force and the Magnus force respectively [7].

The velocity of the particle u_s and the fluid velocity u_g generally differ. The velocity difference $u_g - u_s$ create an unbalance in pressure distribution and generate viscous stresses on the surface of the particle. This unbalance results in a force called drag. It can be expressed by Eq.3

$$F_D = \frac{1}{2}\rho_g C_D A|u_g - u_s|(u_g - u_s) \tag{3}$$

Where ρ_g is the density of the gas phase, A is the frontal area of the particle to the direction of the incoming flow, and C_D is the drag coefficient. The drag coefficient is a function of the particle Reynolds number, Re_s, and the turbulent intensity of the continuous gas phase close to the particle. The particle Reynolds number is defined as [7]:

$$Re_s = \rho_g d_s \frac{u_g - u_s}{\mu_g} \tag{4}$$

Experimental data for a drag coefficient corresponding to a wide range of Re_s of a single sphere where gathered into a standard curve, Schlichting (1979), as shown in Figure 4.

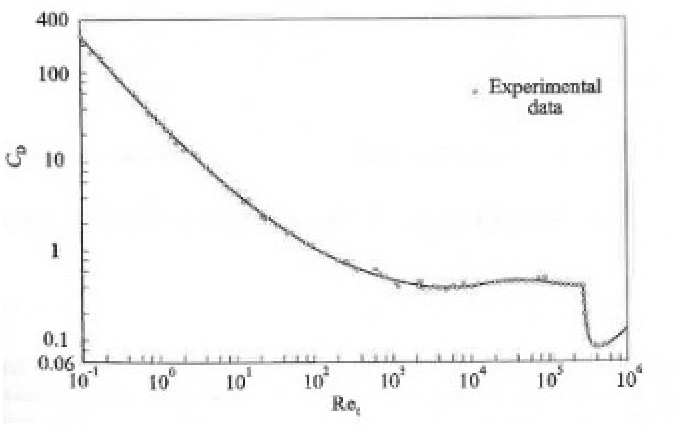


Figure 4 Drag coefficient for spheres as function of Re [7]

Figure 4 illustrates that the $C_D = 0.44$ value is stable in the high Reynold regime, 700 to10^5, this was determined from Newton's experiments in 1710. The reasoning for the stable drag coefficient was that the inertia effect on the particle where dominant. For lower Reynolds numbers, known as creeping flow, the viscous effects is the dominant parameter, and inertia is negligible. Stokes reviled in 1850 that the drag in the creeping flow regime could be expressed as

$$C_D = \frac{24}{Re_s} \tag{5}$$

For very high Reynold number, $3 \cdot 10^5$, a transition occurs from laminar to turbulent boundary layer over the particle, the transition is due to the surface pressure distribution around the particle, caused by change in the wake structure behind the particle [7].

The local pressure gradient gives rise to a force in the opposite direction to the gravity, buoyancy force. This force is given by the Eq. 6.

$$F_p = \rho_g g V_s \tag{6}$$

This means that the force acting against gravity is equal to the weight of the fluid displaced, known as the Archimedes principle. The equation of motion for the particle including buoyancy effect is

$$m\frac{du_s}{dt} = 3\pi\mu_g d_s f\left(u_g - u_s\right) + mg - \rho_g g V_s \tag{7}$$

Or

$$\frac{du_s}{dt} = \frac{f}{\tau_v}\left(u_g - u_s\right) + g(1 - \frac{\rho_g}{\rho_s}) \tag{8}$$

$$f = \frac{C_D}{C_{D_{Stokes}}} = \frac{C_D Re_s}{24} \tag{9}$$

$$\tau_v = \frac{\rho_p d_p^2}{18\mu_c} \tag{10}$$

Where$f$ is the drag factor, explained as the ratio of drag to Stokes drag, and τ_v is the particle relaxation time. The particle relaxation time represent the characteristic time scale it takes for the particle to react to changes in the surrounding flow field [8].

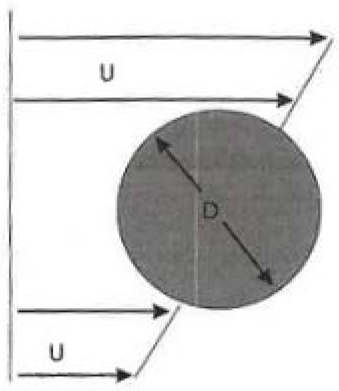
For a particle surrounded by an unsteady flow field additional forces will effect the particle trajectory. One equation that sums up all the forces acting on a droplet, or particle, in an unsteady environment is known as the Basset-Boussinesq-Oseen equation, and is given below in Eq. 11.

$$\begin{aligned} m\frac{du_s}{dt} &= 3\pi\mu_g d_s\left(u_g - u_s\right) + V_s(-\nabla p + \nabla\cdot\tau) + \frac{\rho_g V_s}{2}\left(\dot{u_g} - \dot{u_s}\right) \\ &+ \frac{3}{2}d_s^2\sqrt{\pi\rho_g\mu_g}\left[\int_0^t\left(\dot{u_g} - \dot{u_s}\right)\left(\frac{dt'}{\sqrt{t-t'}}\right) + \left(u_g - u_s\right)_0/\sqrt{t}\right] \\ &+ mg \end{aligned} \tag{11}$$

By dividing the whole Eq.11 by the droplet/particle mass and rearranging the term for virtual mass will give an equation where it is easily seen what the density difference between the particle and gas phases can be justifiably simplified. [8]

$$\begin{aligned} &\left(1 + \frac{1}{2}\frac{\rho_g}{\rho_s}\right)\frac{du_s}{dt} \\ &= \frac{1}{\tau_v}\,(u - u_s) + \frac{1}{\rho_s}(-\nabla p + \nabla\tau_i) + \frac{1}{2}\frac{\rho_g}{\rho_s}\,\dot{u} \\ &+ \sqrt{\frac{9}{2\pi}}\,\left(\frac{\rho_g}{\rho_s}\right)^{\frac{1}{2}}\frac{1}{\sqrt{\tau_v}}\left[\int_0^t\frac{\dot{u} - \dot{u_s}}{\sqrt{t-t'}}dt' + \frac{(u - u_s)_0}{\sqrt{t}}\right] + g \end{aligned} \tag{12}$$

Rotation of particles results in lift forces. The rotation is caused by a velocity gradient or other sources, such as particle collision or particle wall interaction. The force induced by a velocity gradient is called Staffman force. The pressure distribution on the particle give rise to this force. The high velocity on the top side of the particle results in a low pressure, and the high pressure on the low velocity side of the particle results in the lift force. Figure 5 illustrates how the Staffman force occur. When the rotation of the particle is due to another source it is called Magnus lift force. This occurs due to a pressure differential that is created by the rotation. Figure 6 illustrates the Magnus lift on a rotating particle [8].



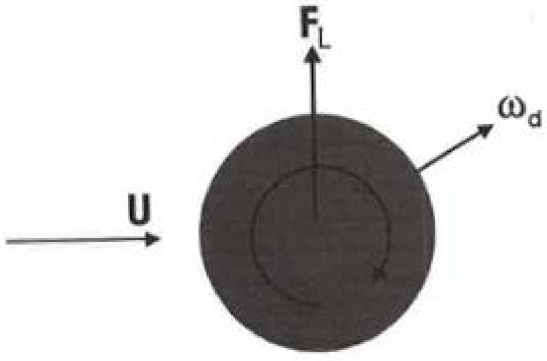


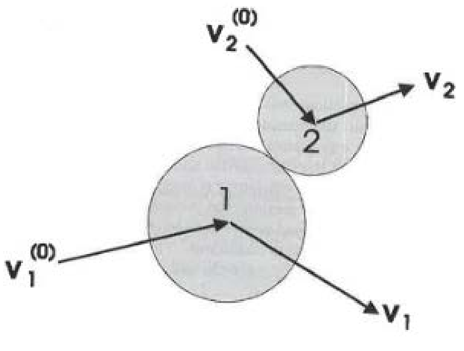
Figure 5 Particle in shear flow [8] *Figure 6 Magnus lift on particle rotating in a fluid [8]*

The terms describing virtual mass, Staffman lift force and Magnus lift force included in the Basset-Boussinesq-Oseen equation is proportional to the density ratio between the gas and particle phase. If the density ratio is less then (10^{-3}) the terms in the BBO equation can be simplified to [8]:

$$\frac{du_s}{dt} = \frac{1}{\tau_v}\left(u_g - u_s\right) + g \tag{13}$$

2.4 Particle-particle interaction

In dilute gas-particle flow the particle-particle interaction is often neglected due to the low probability of collision. For higher solid volume concentration the loss in kinetic energy due to particle collision cannot be neglected. There is two well known models for describing the collision, the hard-sphere and soft-sphere model. For this thesis the particle is described by using the hard-sphere model. The hard-sphere model is described using the same notations as used by Crow [8]. Figure 7 and Figure 8 is illustrations used to understand the notations used.



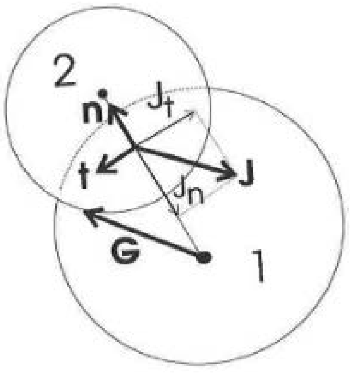


Figure 7 Particle-particle collision [8] *Figure 8 Relative motion of two spheres [8]*

$$m_1(v_1 - v_1^0) = \boldsymbol{J} \tag{14}$$

$$m_2(v_2 - v_2^0) = -\boldsymbol{J} \tag{15}$$

$$I_1(\omega_1 - \omega_1^0) = r_1 n \times J \tag{16}$$

$$I_2(\omega_2 - \omega_2^0) = r_2 n \times J \tag{17}$$

The subscripts 1 and 2 refers to the two particles that is colliding. n is the unit normal vector from particle 1 to particle 2 and the moment of impact, and J is the impulsive force that acts on particle 1, which means it is also the reaction force on particle 2. I is the moment of inertia, and the subscript (0) refers to the values before the collision. r and m is the particle radius and mass, respectively. The velocities v and ω are the translation and angular velocities, respectively.

$$G^0 = v_1^0 - v_2^0 \tag{18}$$

$$G = v_1 - v_2 \tag{19}$$

G^0 and G are the relative velocities between the particle centers before and after impact. The restitution coefficient, e, is used to relate the relative velocity after collision with the pre-collisional velocity. Eq.20 is by Crow [8] regarded as the definition of the restitution coefficient.

$$e = -\frac{n \cdot G}{n \cdot G^0} \tag{20}$$

2.5 Kinetic Theory of Granular Flow

The Kinetic theory of granular flow provides a set mathematical models for the kinetic and collisional regimes using the concept of a granular temperature as a variable to form the solid phase stress tensor [6], [9]. Granular temperature represents the energy of random particle motion in granular flow, and is defined to be one third of the mean square of the instantaneous particle velocity [8]:

$$\Theta = \frac{1}{3}|u_s'|^2 \tag{21}$$

There are two ways the kinetic theory can be approached in OpenFOAM. One method is to assume a local equilibrium between the production and dissipation of granular temperature. The granular temperature is then described with an algebraic model. The other approach is to solve a transport equation for the granular temperature, which is derived by using the

Boltzmann integral differential conservation equation for the probability distribution of random particle motion, see Eq. 22. For more detailed description of this derivation see the work done by Gidaspow [8].

$$\frac{3}{2}\left[\frac{\partial}{\partial t}(\alpha_s\rho_s\theta_s) + \nabla\cdot(\alpha_s\rho_s U_s\theta_s)\right] = (-p_s\boldsymbol{I} + \boldsymbol{\tau}_s):\nabla U_s + \nabla\cdot(\kappa_s\nabla\theta_s) - \gamma_s + J_{vis} \tag{22}$$

The equation for solid pressure is given in Eq. 23. This equation have been generally accepted in the literature and was given by Lun [10]. The solids pressure represent the normal force due to particle interaction in the solid phase. The first term represent the effect of kinetic component, the second represent the collisional effect.

$$p_s = p_{s,col} + p_{s,kin} \tag{23}$$

$$p_{s,kin} = \rho_s\alpha_s\Theta_s \tag{24}$$

$$p_{s,col} = 2\rho_s\alpha_s^2 g_0\left(1 + e_p\right)\Theta_s \tag{25}$$

The solids shear viscosity is given in Eq. 26, it includes both the kinetic and collisional components. Eq. 29 gives the bulk viscosity of the solid phase. The bulk viscosity is represented in a similar way as shear viscosity due to collision [6].

$$\mu_s = \mu_{s,col} + \mu_{s,kin} \tag{26}$$

$$\mu_{s,kin} = \frac{10\rho_s d_s\sqrt{\Theta_s\pi}}{96 g_0(1 + e_p)}\left[1 + \frac{4}{5}(1 + e_p)\alpha_s g_0\right]^2 \tag{27}$$

$$\mu_{s,col} = \frac{4}{5}\rho_s\alpha_s^2 d_s g_0\left(1 + e_p\right)\left(\frac{\Theta_s}{\pi}\right)^{\frac{1}{2}} \tag{28}$$

$$\lambda_s = \frac{4}{3}\rho_s\alpha_s^2 d_s g_0\left(1 + e_p\right)\left(\frac{\Theta_s}{\pi}\right)^{\frac{1}{2}} \tag{29}$$

All these equations contain the restitution coefficient, e, and the radial distribution function, g_0. The use of the radial distribution is to describe the probability of particle collision. When a maximum packing limit is implemented, the radial distribution function becomes infinite when the particle concentration is reached. This is implemented in the last term in the solid pressure equation and limits the packing. Eq. 30 is the radial distribution function by Sinclair and Jacksons description.

$$g_0 = \frac{1}{(1 - \left(\frac{\alpha_s}{\alpha_{s,lim}}\right)^{\frac{1}{3}}} \tag{30}$$

The diffusion of granular energy flux can be explained as conduction of heat. κ_s is the conductivity of the granular energy and is given in Eq. 31.

$$\kappa_s = \rho_s d_s \sqrt{\Theta}\left[\frac{2\alpha_s^2 g_0(1+e_p)}{\sqrt{\pi}} + \frac{9}{16}\sqrt{\pi}\alpha_s^2 g_0\left(1+e_p\right) + \frac{15}{16}\sqrt{\pi}\alpha_s + \frac{25\sqrt{\pi}g_0}{64(1+e_p)}\right] \tag{31}$$

The dissipation of granular energy due to inelastic particle-particle collision is described by Jenkins and Savage (1983) in Eq. 32. Eq. 33 is the dissipation caused by the viscous fluid effects.

$$\gamma_s = 3(1-e_s^2)\alpha^2\rho_s g_0\Theta\left(\frac{4}{d_p}\sqrt{\frac{\Theta}{\pi}} - \nabla\cdot u_s\right) \tag{32}$$

$$J_{vis} = 3K_{sg}\Theta \tag{33}$$

Where K_{sg} is the drag coefficient between the solid and gas phase.

2.6 Frictional stress

When solid volume concentration is high the particle collision are no longer instantaneous, as it is assumed in kinetic theory. The resulting frictional stress due to sustained contact between the particles needs to be put in consideration for the description of the solid phase stress. This is done by adding a frictional stress component to the solid pressure equation, and the viscous stress equation. This additional stress is only implemented when the solid volume concentration is greater than the specified minimum solid volume fraction value [10].

$$p_s = p_{kin} + p_{fric} \tag{34}$$

$$\mu_s = \mu_{kin} + \mu_{fric} \tag{35}$$

A semi-empirical equation for the normal frictional stress, p_f, were proposed by Johnson and Jackson (1987), see Eq. 36.

$$p_f = Fr\frac{\left(\alpha_s - \alpha_{s,min}\right)^n}{\left(\alpha_{s,max} - \alpha_s\right)^p} \tag{36}$$

Fr, n, p are empirical material constant, they are material dependent. $\alpha_{s,min}$ refers to the solid packing limit where the frictional stress is used, $\alpha_{s,max}$ is the maximum solid packing limit. Johnson and Jackson then relates the frictional shear viscosity to the frictional normal stress in Eq. 37, using the law proposed by Coulomb (1776) [10].

$$\mu_f = p_f sin\Phi \tag{37}$$

Φ is the angle of internal friction of the particle.

3 Governing equations

The governing equation for the multiphase gas-solid model is presented in this chapter. For more detailed description of the equations see the work of Gidaspow [6]

3.1 Mass and moment balance

The mass and momentum equation for the solid phase are:

$$\frac{\partial}{\partial t}(\alpha_s \rho_s) + \nabla \cdot (\alpha_s \rho_s \boldsymbol{u_s}) = 0 \tag{38}$$

$$\frac{\partial}{\partial t}(\alpha_s \rho_s u_s) + \nabla \cdot (\alpha_s \rho_s u_s u_s) \\ = \nabla \cdot (\alpha_s \tau_s) - \alpha_s \nabla p - \nabla p_s + \alpha_s \rho_s g + K_{sg}(u_g - u_s) \tag{39}$$

$$\tau_s = \mu_s[\nabla \boldsymbol{u_s} + \nabla u_s^T] + \left(\lambda_s - \frac{2}{3}\mu_s\right)(\nabla \cdot u_s)\boldsymbol{I} \tag{40}$$

The mass and momentum equation for the gas phase are:

$$\frac{\partial}{\partial t}(\alpha_g \rho_g) + \nabla \cdot (\alpha_g \rho_g \boldsymbol{u_g}) = 0 \tag{41}$$

$$\frac{\partial}{\partial t}(\alpha_g \rho_g u_g) + \nabla \cdot (\alpha_g \rho_g u_g u_g) \\ = \nabla \cdot (\alpha_g \tau_g) - \alpha_g \nabla p + \alpha_g \rho_g g + K_{sg}(u_s - u_g) \tag{42}$$

$$\tau_g = \mu_g[\nabla \boldsymbol{u_g} + \nabla \boldsymbol{u_g^T}] - \frac{2}{3}\mu_g(\nabla \cdot \boldsymbol{u}_g)\boldsymbol{I} \tag{43}$$

Where α, ρ and u are the volume fraction, density and velocity, respectively.

And p ,g, and τ are pressure, gravitational acceleration and stress respectively.

The Sub Grid Scale model proposed by Smagorinsky is used for modelling the gas phase turbulence.

$$\mu_{eff,g} = \mu_{lam,g} + u_{turb,g} \tag{44}$$

$$\mu_{turb,g} = \rho_g (C_s \Delta)^2 |\boldsymbol{S}| \tag{45}$$

$\boldsymbol{S}$ is the characteristic filtered rate of strain and $\Delta = \left(\Delta_i \Delta_j \Delta_k\right)^{\frac{1}{3}}$ is the filter width. C_s is the dimensionless Smagorinsky model constant which is set to 0.19 in OpenFOAM [11].

3.1.1 Drag model

The drag model that is used in this thesis is Gidaspow and it defines the drag coefficient as [1]:

$$K_{sg} = \begin{cases} \dfrac{\frac{3}{4} C_D \alpha_g \rho_g \left|u_g - u_s\right|}{d_s} \alpha_g^{-2.65} & if\ \alpha_s < 0.2 \\ \dfrac{150 \mu_g \alpha_s^2}{\alpha_g^2 d_s^2} + \dfrac{1.75 \rho_g \alpha_s}{\alpha_g d_s} \left|u_g - u_s\right| & if\ \alpha_s \geq 0.2 \end{cases} \tag{46}$$

Drag coefficient C_D is defined as:

$$C_D = \begin{cases} \dfrac{24}{Re}\left[1 + 0.15(Re)^{0.687}\right] & Re < 1000 \\ 0.44 & Re \geq 1000 \end{cases} \tag{47}$$

The particle Reynolds number is defined as:

$$Re = \frac{\rho_g d_p \left|U_g - U_s\right|}{\mu_g} \tag{48}$$

The Gidaspow drag model uses a combination of the Wen and Yu drag model and the Ergun equation. The Ergun equation is used when the solid volume is 0.2 or higher. Then when the solid volume fraction is lower than 0.2 it switches to the Wen and Yu model.

3.1.2 Energy balance

Eq. 49 describes the conservation of energy on a differential form with internal energy as the dependent variable.

$$\frac{\partial(\rho e')}{\partial t}+\frac{\partial(u_j e')}{\partial x_j}=-\frac{\partial q_j}{\partial x_j}-p\cdot\frac{\partial u_i}{\partial x_i}+\tau_{ij}\frac{\partial u_i}{\partial x_j}+\dot{S} \tag{49}$$

The equation express that the increase of energy must be equal to the rate of heat added in addition to the rate of work done on the fluid. In the equation e' is the internal energy, q is the heat flux and $\dot{S}$ is the source or sink term [12].

3.1.3 Equation of state

The equation of state for an ideal gas is written as:

$$pV = nRT \tag{50}$$

Where p, V is the pressure and volume respectively, n is the number of moles, R is the universal gas constant and T is the temperature. The equation of states makes it possible to link different state variables of the fluid, such as density variation as a result of pressure and temperature variation [12].

4 Computational fluid dynamics

4.1 CFD introduction

All main CFD codes are contains three main elements: a pre-processor, a solver and a post-processor.

Pre-processing is where input of the flow is defined. Firstly the geometry of the fluid flow region needs to be defined, known as the computational domain. Then division of the domain into a number of smaller sub-domains, known as a grid or mesh. Followed by defining fluid properties, and select the physical and chemical phenomena that need to be modelled.

The solver can compute a solution when the problem has been defined in the pre-processor. There are three well known numerical solution technique that can be used in the CFD analysis: finite element, finite difference and spectral methods. OpenFOAM and most well-established CFD codes uses finite volume method for the analysis, finite volume method is a formulation of finite differencing. The finite volume algorithm consist of the three following steps [12]:

- Integration of the governing equations of fluid flow over all the control volumes inside the computational domain
- Discretising the resulting integral equations into a system of algebraic equations
- Solution of the algebraic equations by use of iterative methods

In the post-processor the results can be visualized by vector plots, 2D and 3D surface plots, particle tracking, and view manipulation can be done.

The accuracy of the solution is governed by the number of cells in the computational domain. For most cases the larger the number of cells the more accurate the result will be, this will come with a computational cost, increase in computational time and hardware power. Since the physics involved in solving fluid problem is complex, it is important that the analyst have knowledge of the underlying physics that are involved. This knowledge is important when results are visually good, but might be physically incorrect. To validate the results it should be compared to experimental data, or similar problems that are reported fs experimental data does not exist [12].

4.2 OpenFOAM software

Open Source Filed Operation And Manipulation (OpenFOAM) is an open source C++ library that primarily creates executables, known as applications. Users have access to a large set of precompiled applications and the ability to create their own, or modify existing ones, the users C++ knowledge is the only restriction. The applications is divided into two categories, solvers and utilities. Solvers are designed to solve specific problems in computational continuum mechanics. Utilities is used for pre-and post-processing tasks involving data manipulation and algebraic calculations [13].

OpenFOAM is an Open source software that has gotten a lot of attention from academic and research purposes. Open source gives unrestricted access to the code/algorithm giving a limitless customization possibilities for the user.

4.2.1 Case setup

Cases is build up of one main folder, called the directory. Within the directory three subdirectories are stored. The subdirectory is named system, constant and 0/time directories, see Figure 9.

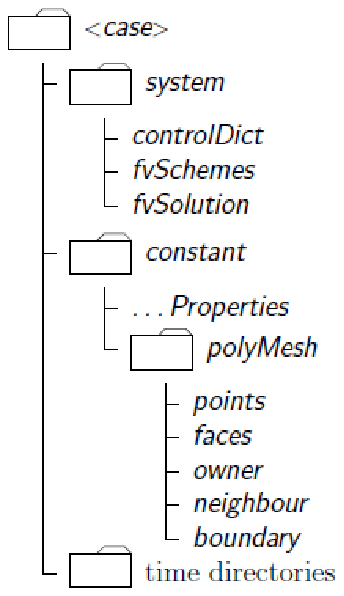


Figure 9 General OpenFOAM case structure [14]

A description of the minimum set of files that are required to run an application.

The system directory is where the specific parameters associated with the solution procedure in specified. Under the system directory the run control is given in the controlDict file. Here the start and end time is given, time step and parameters for data output. Discretization schemes used for the solution is given in the fvSchemes file. The equation solvers and other algorithm control are given in the fvSolution file.

In the constant directory a subdirectory called polyMesh gives a description of the case mesh. Specification for physical properties for the fluid flow is found in the transportProperties and thermoProperties files.

The time directory contains files for data for particular fields. This is data that describe initial values and boundary conditions that the user have specified to define the flow problem. When the case run resulting output files will create new time directories [14].

4.3 Eulerian cfd modeling

In computational fluid dynamics computers are used to numerically solve the equations that describes a flow. The equations are based on conservation of mass, momentum and energy of a system. The equations that are the classic Navier-Stokes equations taught in fluid dynamics are believed to be useful to describe any kind of single-phase fluid flow [15].

Each of these conservation equations could be solved directly if there is enough spatial and time resolution [15], in practice due to the limited computational power computers have today, this would be impossible for most applications. Simplifications of the equations needs to be implemented. The most common is to leave out less important terms and use averaging of the equation in time and space. The additional fluctuation terms due to averaging requires modelling. In a normal single-phase flow these fluctuations is known as Reynold Stresses. The average equations are called Reynolds Average Navier-Stokes (RANS) equation.

For the lagrangian approach each particle is individual tracked and the interaction between fluid, walls and other particles is described using the single particle model. This is described in chapter 2. For the fluid the modeling is done the same way as for single-phase flow, with the implementation of fluid-particle interaction. The computer power required for Lagrangian makes it a valid approach for low particle fraction simulations, but for fluidization the time required makes it not too time consuming for industrial applications.

The current standard approach to model fluidized beds is the Eularian method. In the Eularian method both the solid and fluid phase is described as interpenetrating fluids. The local average properties is described locally for the flow and particle clouds. This means that the number of the equations only depends on the amount of phases present in the analysis. The particle cloud does not always behave like a fluid wish is a disadvantage for the model. To describe the particle-particle interaction the Kinetic Theory of Granular flow is used [15]. This is the approach used in this thesis.

4.4 Solver

twoPhaseEulerFoam is a transient multiphase flow solver for 2 compressible fluid phases, where one phase is dispersed, for example gas bubbles in liquid, or as in the case of this thesis solid particles in gas. The solver was developed based on a previous solver called bubbleFoam, developed by Rusche [16].

The solver had a major change from OpenFOAM version 2.2 to 2.3. Not much changed from 2.3 to 3.0, which is used in this thesis. The most important change is that the phases is now modelled using OpenFOAMs thermophysical models.

Since the new version of twoPhaseEulerFoam uses thermo-physical models for the phases, the user most specify initial boundary conditions for temperature, for both phases. Even if temperature is not something that is looked at in this thesis, it needs to be added to the time directories to make the solver work [17].

4.4.1 Solver capabilities

All the phases are treated as compressible. The behavior of the phases can be specified in file thermophysicalProperties. The solver solves an energy equation for all the phases, this cannot be turned off. A number of phase interaction models have been added specifically for gas-liquid systems. twoPhaseEulerFoam used to be hard coded to only use kEpsilon turbulence model, but now a number of turbulence models have been added [17].

4.4.2 Numerical treatment

In OpenFOAM the equations are solved using the finite volume method, and with the PIMPLE algorithm for the pressure-velocity coupling [9]:

$$\frac{\partial \alpha_g}{\partial t} + \nabla \cdot \left(\alpha_g u_g\right) = 0 \tag{51}$$

$$\frac{\partial}{\partial t}(\alpha_s u_s) + \nabla \cdot (\alpha_s u_s u_s) = \frac{1}{\rho_s}\nabla \cdot (\alpha_s \tau_s) - \frac{\alpha_s}{\rho_s}\nabla p - \frac{1}{\rho_s}\nabla p_s + \alpha_s g + \frac{K_{sg}}{\rho_s}(u_g - u_s) \tag{52}$$

$$\frac{\partial \alpha_g}{\partial t} + \nabla \cdot (\alpha_g u_g) = 0 \tag{53}$$

$$\frac{\partial}{\partial t}(\alpha_g u_g) + \nabla \cdot (\alpha_g u_g u_g) = \frac{1}{\rho_g}\nabla \cdot (\alpha_g \tau_g) - \frac{\alpha_g}{\rho_g}\nabla p + \alpha_g g + \frac{K_{sg}}{\rho_g}(u_s - u_g) \tag{54}$$

When trying to solve Eq. 52 and 54 one of the main problems is that the volume fraction is present in all the terms. This will create a null equation when one of the phases is not present. Venier et al. [9] then proposed a conservative formulation to increase the robustness of the solver, and dealing with phase fractions reaching zero. Phase fractions reaching zero is dealt with by avoiding the solution in cells where the phase fractions is smaller than a certain cutoff value. The Semi-discrete form of the momentum equations are:

$$A_s u_s = H_s - \frac{\alpha_s}{\rho_s}\nabla p - \frac{1}{\rho_s}\nabla p_s + \alpha_s g + \frac{K_{sg}}{\rho_s}(u_g - u_s) \tag{55}$$

$$A_g u_g = H_g - \frac{\alpha_g}{\rho_g}\nabla p + \alpha_g g + \frac{K_{sg}}{\rho_g}(u_s - u_g) \tag{56}$$

Where H_i includes the off-diagonal contribution and A_i condensates the diagonal coefficients. This can then be used to generate two new coefficients:

$$\zeta_s = \frac{1}{A_s + \frac{K_{sg}}{\rho_s}} \tag{57}$$

$$\zeta_g = \frac{1}{A_g + \frac{K_{sg}}{\rho_g}} \tag{58}$$

Now the phase velocities can be expressed by Eq. 59 and 60.

$$u_s = \zeta_s(H_s + \frac{K_{sg}}{\rho_s}H_g - \frac{\alpha_s}{\rho_s}\nabla p - \frac{1}{\rho_s}\nabla p_s + \alpha_s g) \tag{59}$$

$$u_g = \zeta_g(H_g + \frac{K_{sg}}{\rho_g}H_s - \frac{\alpha_g}{\rho_g}\nabla p + \alpha_g g) \tag{60}$$

Using Eq. 59 and Eq. 60 the phase velocity fluxes can be expressed by Eq. 61 and 62.

$$\varphi_s = \sum_f \left\{ \zeta_{s,f} \left(H_s + \frac{K_{sg}}{\rho_s} H_g + \alpha_s g \right)_f \cdot S - \zeta_{s,f} \left(\frac{\alpha_s}{\rho_s} \nabla p \right)_f \cdot S - \zeta_{s,f} \left(\frac{1}{\rho_s} \nabla p_s \right)_f \cdot S \right\} \tag{61}$$

$$\varphi_g = \sum_f \left\{ \zeta_{g,f} \left(H_g + \frac{K_{sg}}{\rho_g} H_s + \alpha_g g \right)_f \cdot S - \zeta_{g,f} \left(\frac{\alpha_g}{\rho_g} \nabla p \right)_f \cdot S \right\} \tag{62}$$

Where S is the face normal vector.

For the continuity equation $\sum_i \alpha_i = 1$ must be verified locally at each time step. This makes it a possibility to solve one phase continuity equation and then derive the volume fraction of the remaining phase. The semi-discrete form of the continuity equation can be written as:

$$\frac{\partial \alpha_s}{\partial t} + \sum_f \alpha_{s,f} \varphi_s = 0 \tag{63}$$

The phase fractions needs to be bounded between zero and one, and this is done using a MULES limiter implemented in OpenFOAM. For more information about MULES limiter see [18]. By defining the mixture and relative fluxes a final semi-discrete form of the phase continuity equation can be derived [9].

$$\varphi = \alpha_{g,f} \varphi_g + \alpha_{s,f} \varphi_s \tag{64}$$

$$\varphi_{r,s} = \varphi_s - \varphi_g \tag{65}$$

Where:

$$\varphi' = \varphi + \frac{1}{\rho_s} \zeta_{s,f} \left(\frac{\partial p_s}{\partial \alpha_s} \right)_f |S| \nabla \alpha_s \tag{66}$$

$$\varphi'_{r,s} = \varphi_{r,s} + \frac{1}{\rho_s} \zeta_{s,f} \left(\frac{\partial p_s}{\partial \alpha_s} \right)_f |S| \nabla \alpha_s \tag{67}$$

With this representation of the relative fluxes, a final semi-discrete form of the phase continuity equation can be written as:

$$\frac{\partial \alpha_s}{\partial t} + \sum_f \ (\alpha_s \varphi')_f + \sum_f \left(\alpha_s \alpha_g \varphi'_{r,s} \right)_f - \frac{1}{\rho_s} \alpha_{s,f} \zeta_{s,f} \left(\frac{\partial p_s}{\partial \alpha_s} \right)_f |S| \nabla \alpha_s = 0 \tag{68}$$

By summing up the solid and gas phase continuity equation we obtain Eq. 69

$$\nabla \cdot \varphi = 0 \tag{69}$$

Applying divergence on the phase velocity fluxes in each term in Eq. 61 and Eq. 62, and consider Eq. 69, an expression for the pressure equation is obtained, see Eq. 70:

$$\nabla \cdot \left\{ \left[\alpha_{s,f}\zeta_{s,f}\left(\frac{\alpha_s}{\rho_s}\right)_f + \alpha_{g,f}\zeta_{g,f}\left(\frac{\alpha_g}{\rho_g}\right)_f \right] (\nabla \cdot S) \right\} = \nabla \cdot \varphi^0 \tag{70}$$

$$\varphi^0 = \alpha_{s,f}\varphi_s^0 + \alpha_{g,f}\varphi_{g,f}\varphi_g^0 \tag{71}$$

$$\varphi_s^0 = \sum_f \left\{ \zeta_{s,f}\left[H_s + \frac{K_{sg}}{\rho_s}H_g + \alpha_s g \right]_f \cdot S - \zeta_{s,f}\left[\frac{1}{\rho_s}\nabla p_s\right]_f \cdot S \right\} \tag{72}$$

$$\varphi_g^0 = \sum_f \left\{ \zeta_{g,f}\left[H_g + \frac{K_{sg}}{\rho_g}H_s + \alpha_g g \right]_f \cdot S \right\} \tag{73}$$

4.4.3 The solution algorithm

The solution algorithm used in twoPhaseEulerFOAM used to be a PISO algorithm explained in the work done by Rusche 2002 [16]. Now the solver has evolved its algorithm from PISO to PIMPLE. PIMPLE is an algorithm that uses a combination of PISO and SIMPLE to couple the pressure field. By doing so it allows the implementation of an under-relaxation factor to enforce the convergence of the iterative procedures and outer iterations in order to enhance the coupling between mass and momentum conservation equations. The sequence of the algorithm is taken from the work done by Venier 2014 [9].

1. Start the continuity equation loop
   a. Solve Eq.(68) without the particle pressure contribution, using MULES limiter.
   b. Calculate $\frac{\partial p_s}{\partial \alpha_s}$ and correct the continuity equation to re-obtain α_s. If the kinetic theory is used, solve the granular temperature Θ_s (Eq.(22) in order to obtain the kinetic particle pressure p_s (Eq.(23))
   c. Calculate $\alpha_g = (1 - \alpha_s)$
   d. Iterate from a. until a convergence criterion is reached
2. Calculate the drag coefficient K_{sg}
3. Update the momentum equation coefficients
4. Solve the predicted velocity from the phase momentum equation with the previously stored pressure
5. Start the pressure equation loop
   a. Obtain the interpolated values of the phase fraction α_i and momentum coefficients $\boldsymbol{H}_i$ and z$_i$ on the cell faces.
   b. Construct the partial phase fluxes given by Eq.(72) and (73)
   c. Construct the pressure flux contribution coefficient
   d. Solve the pressure equation and correct for non-orthogonal meshes
   e. Correct the face fluxes with the pressure flux contribution
   f. Correct the cell centred phase velocities

6. Iterate from 1. Until a convergence criterion is reached

4.5 Numerical schemes

When the governing equations are integrated over the control volume, the resulting integral equations needs to be discretized into algebraic equations. The discretization method that is used in OpenFOAM is specified in the fvScheme file in the system directory. OpenFOAM provides several different discretization schemes., the terms that needs to be specified by numerical schemes are divided into categories listed in Table 1 [19].

Table 1 Group of terms

Group	**Description**
interpolationSchemes	Point-to-point interpolations of values
snGradeSchemes	Component of gradient normal to cell face
gradSchemes	Gradient ∇
divSchemes	Divergence $\nabla \cdot$
laplacianSchemes	Laplacian ∇^2
timeScheme	First and second time derivatives $\frac{\partial}{\partial t}, \frac{\partial^2}{\partial^2 t}$
fluxRequired	Fields that require generation of a flux

The schemes applied in this thesis is shown in Table 2. The different schemes are based on tutorial files from twoPhaseEulerFoam that comes with OpenFOAM 3.0.1x, and Liu's Ph.D thesis [20].

Table 2 Selected numerical discretization schemes in fvScheme dictionary

Group	**twoPhaseEulerFoam**
ddtSchemes	Euler
gradSchems	Gauss linear
divSchemes	div(phi,alpha) : Gauss vanLeer div(phir,alpha) : Gauss vanLeer div(alphaRhoPhi,U) : Gauss limitedLinearV 1 div(phi,U) : Gauss limitedLinearV 1 div(alphaRhoPhi, h | e): Gauss limitedLinear 1 div(alphaRhoPhi, K) : Gauss limitedLinear 1 div(alphaPhi,p) : Gauss limitedLinear 1 div(alphaRhoPhi.particles,Theta) : Gauss limitedLinear 1
laplacianSchemes	Default : Gauss linear uncorrected Bounded : Gauss linear uncorrected
interpolationSchemes	Default : linear
snGradSchemes	Default : uncorrected Bounded : uncorrected
fluxRequired	Default : no

Detailed description of the discretization schemes will not be done in this thesis. Multiphase flow modeling require long computational time, so everything that can be done to speed up the simulation should be incorporated, such as a coarse grid. Liu [20] looked into the use of Total Variation Diminishing (TVD) schemes for gas-solid fluidized bed with a central jet, and generated good results. This type of discretization schemes was made to capture sharp shock predictions without any misleading oscillation on a coarse grid. The TVD in this thesis is the vanLeer for the discretization of the solid volume convection terms, and the limitedLinearV for the phase velocity convection terms. They both were tested up against other TVD by Liu [20], and gave similar satisfying results.

4.6 Solution and algorithm control

When numerical simulation is run it is important to make sure that convergence of the solution is reached. A converged solution is a good indicator to identify if the simulation is physically correct. The best way to ensure that the simulation is physically correct is to validate the result with experimental data. By monitoring the residual and continuity error the user can evaluate if the simulation has reached convergence or not. Another parameter that can be looked at is the Courant number, and ensure that the value is below one [21].

4.6.1 Residual and Courant number

Residuals are the imbalance, or error, that occur in the equations for each solved variable. It can be seen as a measurement of the error in the solution, the smaller it gets the more accurate the solution will be. The residuals can be calculated by substituting the current solution for a time step into the equations and taking the absolute value of the left and right hand side. To make the result independent of the scale of the problem the residual is normalized [21].

The Courant number is defined as:

$$Co = \frac{\Delta t \cdot |U|}{\Delta x} \tag{74}$$

Δt is the length of the time step, U is the flow velocity and Δx is the length of the cell. The Courant number represent the fraction of the cell that the flow is able to advance through during a time step. This will indicate if the time step that is used is small enough for a good time discretization. The max Courant number should be smaller than 1, the lower the value the more stable the solution will be. This means that for a low Courant number the simulation time will increase. In the cases run the max Courant number is set 0,9. The mean Courant number for the cases stays below 0.07 with this upper limit [22].

To ensure that the PIMPLE solution is reasonably converging Hjertager [22] has pointed out that the most important factors are; the continuity error stays small for the last loop. Courant number should not become large. Even though they can be larger than with the PISO method. The initial residual is not as important as for SIMPLE simulations, but the initial residual of the last outer loop should be relatively small.

4.6.2 fvSolution

For each discretized equations a solver needs to be specified. The solvers that are used is described in the fvSolution file. This files contain two dictionaries. First the control for the solvers are specified, then the controls for the solution algorithm needs to be specified. The solver control used for all the cases run in this thesis is given in Table 3. Solution methods, tolerances, preconditioner etc., is described in the solver control. The solution algorithm used in this thesis is described in Table 4.

The initial residual is evaluated based on the current field values before equations of a particular field is solved. After each iteration is done by the solver the residual gets re-evaluated. The solver stops if one of the following conditions are reached; the residual falls below the solver tolerance. The relative tolerance is reached, this is the ratio between current and initial residual. The number of iterations exceeds a maximum number of iterations [21].

The solvers and tolerances used for all the cases are based on the tutorial cases that comes OpenFOAM. Since the simulations done is this thesis is transient the relative tolerance is set to 0. This will force the solution to converge to the solver tolerance in each time step [21].

Table 3 Solvers selected in fvSolution dictionary

Equation	**twoPhaseEulerFoam**

Alpha	solver smoothSolver;
	smoother symGaussSeidel;
	nAlphaCorr 1;
	nAlphaSubCycles 2;
	smoothLimiter 0.1;
	implicitPhasePressure yes;
	tolerance 1e-9;
	relTol 0;
	minIter 1;

p_rgh	solver GAMG;
	smoother DIC;
	nPreSweeps 0;
	nPostSweep 2;
	nFinestSweeps 2;
	cacheAgglomeration true;
	nCellsInCoarsestLevel 10;
	agglomerator faceAreaPair;
	mergeLevels 1;
	tolerance 1e-8;
	relTol 0;

p_rgh_Final	solver GAMG;
	smoother DIC;
	nPreSweeps 0;
	nPostSweep 2;
	nFinestSweeps 2;
	cacheAgglomeration true;
	nCellsInCoarsestLevel 10;
	agglomerator faceAreaPair;
	mergeLevels 1;
	tolerance 1e-8;
	relTol 0;

U	solver smoothSolver;
	smoother symGaussSeidel;
	tolerance 1e-5;
	relTol 0;
	minIter 1;

h | e	solver smoothSolver;
	smoother symGaussSeidel;
	tolerance 1e-6;
	relTol 0;
	minIter 1;

Theta	solver smoothSolver;
	smoother symGaussSeidel;
	tolerance 1e-6;
	relTol 0;
	minIter 1;

Table 4 Settings for solution algorithm control

PIMPLE	
nOuterCorrectors	3;
nCorrectors	1;
nNonOrthogonalCorrectors	0;
pRefCell	0;
pRefValue	0;

twoPhaseEulerFoam uses smooth solvers with the Gauess seidel smoother, this is the most used solver and is known to be very reliable. For the pressure equations a Generalised geometric-algebraic multi-grid (GAMG) solver is used with a diagonal incomplete cholesky preconditioner (DIC) [21].

5 Experimental setup of the 1/9th scale cold CFB boiler

There is a lack in experimental data for gas-solid flow in fluidized bed boilers, so validation data for the CFD simulation is scarce. Most laboratory studies are done in high and narrow risers, often circular cross section with operation conditions that are not relevant for boilers. CFB boilers, and the one looked at in this thesis, typically have a rectangular or square cross section, low solid flux, and within Geldart powder group B.

The dimensions of the riser in the scaled model shown in Figure 10 is, 1.5m x 0.19m x 0.17m which corresponds to Height (z) x Depth (x) x Width (y). The cyclone is located 1.2m above the primary air distributor. The cyclones job is to separate the solids from the fluid, and create the recirculating pattern. The solids that are separated by the cyclone goes down to the particle seal designed as a bubbling bed. When the solids passes the bubbling bed they are introduced to the riser. The amount of solids used in the experimental setup was adjusted to give a pressure drop in the riser of 2.7kPa. 2.7kPa in the scaled down boiler is corresponding to the pressure drop of 8.0kPa in the full scale boiler. To simplify the flow pattern there was not used any secondary air in the bubble bed. Figure 10 shows a photo of the 1/9th scale model of the Chalmers 12MW CFB boiler.[1]

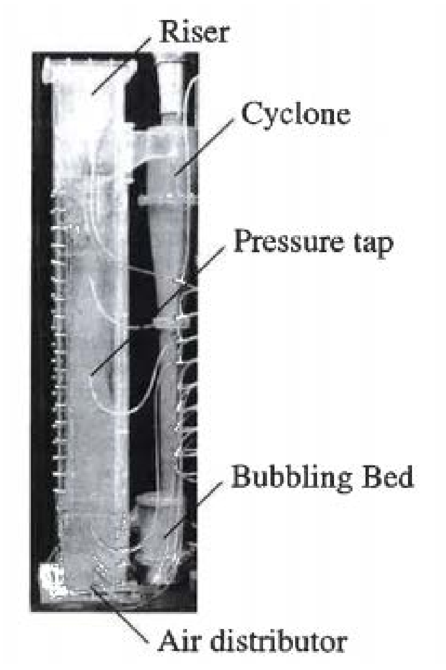


Figure 10 Photo of the 1/9th scale CFB model [1]

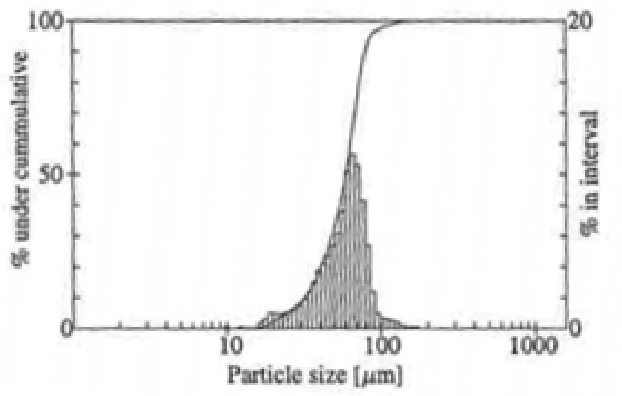


Figure 11 Particle size distribution for the bronze particles used [1]

Using Honeywell solid state pressure sensors monitoring of the pressure drop was done. Placement of the pressure taps is seen in Figure 10. From the pressure measures by the sensors, an average solid volume concentration was derived. For measuring particle velocities a two-component laser doppler anemometer was used. This anemometer provides mean and RMS-velocities within a measurement volume. By traversing the data from the measurement volume in the horizontal x- and y-direction, horizontal velocity profiles were obtained for the particles. For more detailed description of how the experimental data was gathered see Ibsen [1].

This thesis is looking at one of the 3 cases described by Ibsen [1]. The experimental case that is used to validate the results of the computational simulation is referred to as case 2 in [1]. For this case bronze particles are used with a diameter range of $d_p = 10 - 100\ \mu m$ and a mean diameter of $d_p = 60\ \mu m$, these values corresponds to the mean diameter of 310 mu in the full-scale boiler. Figure 11 shows the particle size distribution of the case. For the inlet a superficial gas velocity of $U_{sup} = 1\ m/s$ was set, which corresponds to $U_{sup} = 3\ m/s$ is the full-scaled boiler.

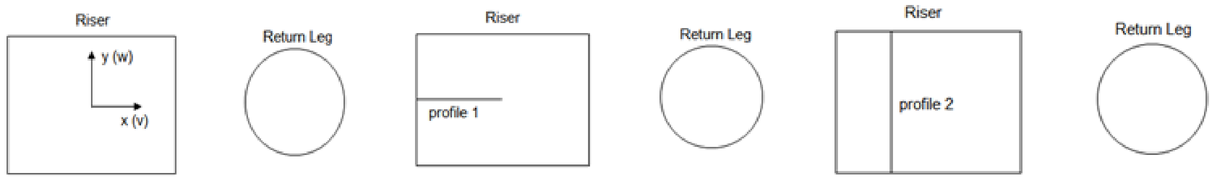


Figure 12 Location of the origin in the x,y-plane and location of Profile 1 & 2 [1]

Measurement of two-dimensional particle velocity, RMS-velocity and correlation terms $< u_p' v_p' >$ were performed. Measurements were obtain along profile 1 and 2 which is located at a dimensionless height z/H=0.5. z/H=0.5 corresponds to half the riser height. Figure 12 shows the location of the origin in the x,y-plane, and the location for profile 1 and 2. In the results the velocity will be referred to as axial and spanwise, which is in the z (u) and y (w) respectively. Positive z-direction in upward the riser [1].

6 Modelling approach

In this chapter the case setup of the simulations will be described. This includes information on mesh, boundary conditions and phase properties used for the different cases. There is a lot of things to consider when choosing parameters used in a circulating fluidized bed, values for particle wall interaction is taken from results gathered from similar case studies done [24].

6.1 Transient case setup

Doing multiphase analysis of fluidized beds requires a significant amounts of computational power. Simulations was first done using a laptop that could run 6 processors effectively at the time. The analysis needs to run for 20-30sec real time, to get steady mean values of the circulation in the system. Doing so long simulations on a laptop, would not be sustainable for all the cases needed to be run. OpenFOAM was then uploaded to the universities PC cluster that made it possible to run more simulations at a time, and not overheat the laptop.

The first cases run were done based on previous work. This was done to see how OpenFOAM would compare to simulations done by Ibsen with FLOTRACE-MO-3D [1]. The particle diameter range of the case were $D_p = 10 - 100\ \mu m$. The flow is driven by a uniform velocity input at the inlet and the amount of particles in the riser is adjusted to give the correct pressure drop over the riser. Cases were run with 4 different types of particle size diameters, 45 μm, 60 μm, 70 μm and 100 μm. A secondary air inlet in the bubble bed was also used, due the increase in stability it added to the simulation. The most interesting factors for this cases is to see if OpenFOAM also predicts a turbulent regime in the bottom of the riser.

Based on the results of previous work a uniform inlet velocity flow will have difficulties creating the dense bottom bed that is observed in the experiment. Therefore a different approach to the flow parameters are suggested in case 2.45, 2.60, 2.70. The flow is here driven by a total pressure condition for the inlet and a uniform flow velocity at the outlet corresponding to the same flow rate that used in the experimental setup. In the same way the amount of particles needs to be adjusted to give the correct pressure drop over the riser. The cases run three different particle size diameters, 45 μm, 60 μm and 70 μm.

Both cases were simulated for 20 second real time. The averaging values are gathered from 5 sec to 20 sec. This is done so all the values are taken after circulation has occurred. More details on the different boundary conditions used in the two approaches is given in chapter 6.3. The cases is separated into groups based on what type of approach is used for the superficial gas velocity, see Table 5.

Table 5 Case groups

Group A	**Group B**
Case 1.45	Case 2.45
Case 1.60	Case 2.60
Case 1.70	Case 2.70
Case 1.100	

6.2 Mesh generation

OpenFOAM has its own blockmesh utility that can generate simple meshes of blocks with hexahedral cells. This is done by defining vertices with coordinated that will define the blocks. Defining the number of cells in each direction, and their refinements, and last define the boundary patches. For a complex geometry the coordinate point tracking is challenging and can be very time consuming. In this thesis a CAD software, Inventor, was used to create the geometry. The geometry was then exported to another CAD software called Salome to create and name all the STL-files that made up the geometry. Salome was also used to extrude the outlet pipe so the cyclone could work as intended. OpenFOAM can read the STL-files created in Salome and convert them into a mesh.

There is two different meshes used in this thesis, both of them is created using the snappyHexMesh utility in OpenFoam. The work flow by the snappyHexMesh utility can be described by the two following steps [17]

1. The background of the mesh needs to be generated using blockMesh. Without the background mesh the later work of the snappy utility will not work. If possible the background mesh should consist of all-hex cell with preferably aspect ratio of 1, cube-shaped cells. The more of the backgrounds cell-edges that intersects the tri-surfaces the better it will be.
2. SnappyHexMesh will perform three basic steps:
    a. Castellating, The tri-surface gets approximated by removing or splitting cells outside/inside of the tri-surfaces, depends if the mesh is generated outside or inside the tri-surfaces.
    Splitting will refine the mesh near the surfaces.
    Removal is simply to get rid of cells inside/outside of the object.
    b. Snapping. The remaining mesh is modified to reconstruct the surfaces of the object.
    c. Layer addition. Hexahedral cells is added to the boundary surfaces to ensure good mesh quality.

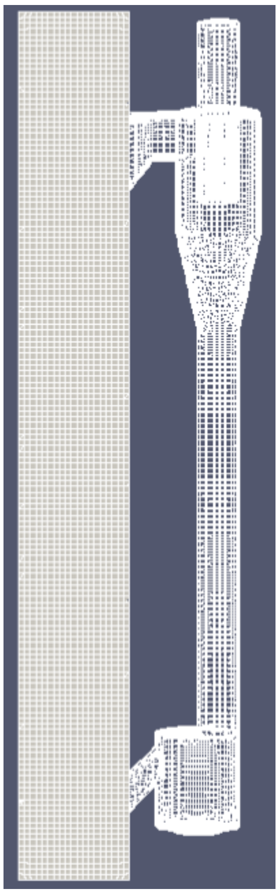


Figure 13 Mesh used for Group B

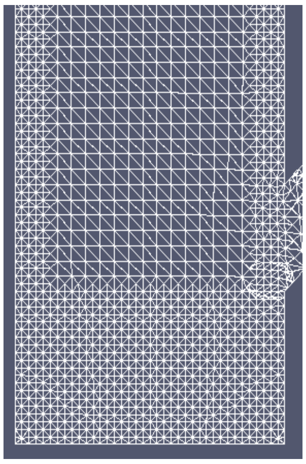


Figure 14 Zoomed in on refinement for Group A

The mesh used for group A with a uniform inlet velocity if built of a coarse cells in the core of the riser and fine mesh along the walls and in the bottom bed. This was done to see if a fine mesh would be able to predict a denser bottom bed than previous work done with this type of inlet boundary condition.

For group B with a total pressure condition at the inlet a mesh with medium sized cells was used for the whole riser. In chapter 6.2.1 a mesh independent test was done for this mesh, and a more detailed description of the cell size is given.

The mesh quality is crucial to the accuracy of the result. The checkMesh command can be used to list up specific mesh quality parameters. When running the checkMesh command for both the meshes it resulted in a “mesh ok” from OpenFoam. According to Hjertager (2009) [22] this will indicate that there is no major problems with the mesh. The checkMesh output for both meshes is given in Appendix A.1 and A.2.

6.2.1 Mesh independence test

For numerical simulations it is important that a grid independence test is done to verify that the results are independent of the mesh size. Previous work done by Zhang and Vander-Heyden (2001) [23] defined a grid independent result when the flux in their simulation matched the flux of the experiment. They simulated a CFB with dimensions 0.2m x 2.0m x 0.2m, depth x height x width. The size of the mean particles used were 120 µm. From their work grid independence were reached when the ratio between the particle diameter and the width of the cell, d_s/x_{cell} is equal 1.8 %.

In this thesis the mean diameter of the particle distribution is 60 µm. This means that it would need a substantial smaller cell size to reach 1.8% particle to cell ratio. It is unachievable to run several simulations due to computer time it will take with so many cells. A test is run for three different cell sizes for case 2.70, which predicts particle fluxes well. The particle to cell width ratio for the three cases are given in Table 6.

Table 6 Mesh types and their width

Mesh type	$\boldsymbol{d_s/x_{cell}}$
Coarse	0.92%
Medium	1.14%
Fine	1.40%

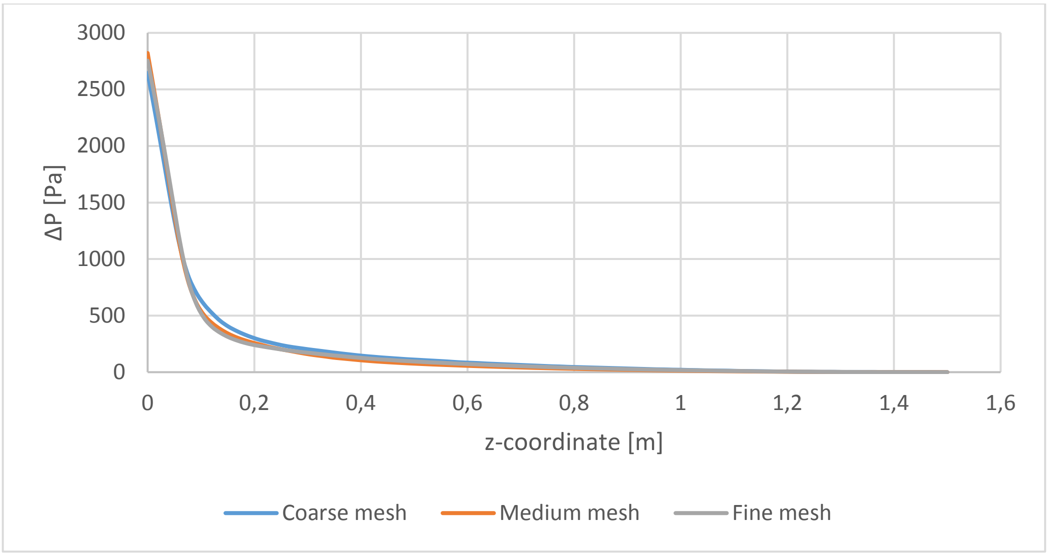


Figure 15 Pressure drop along the riser height

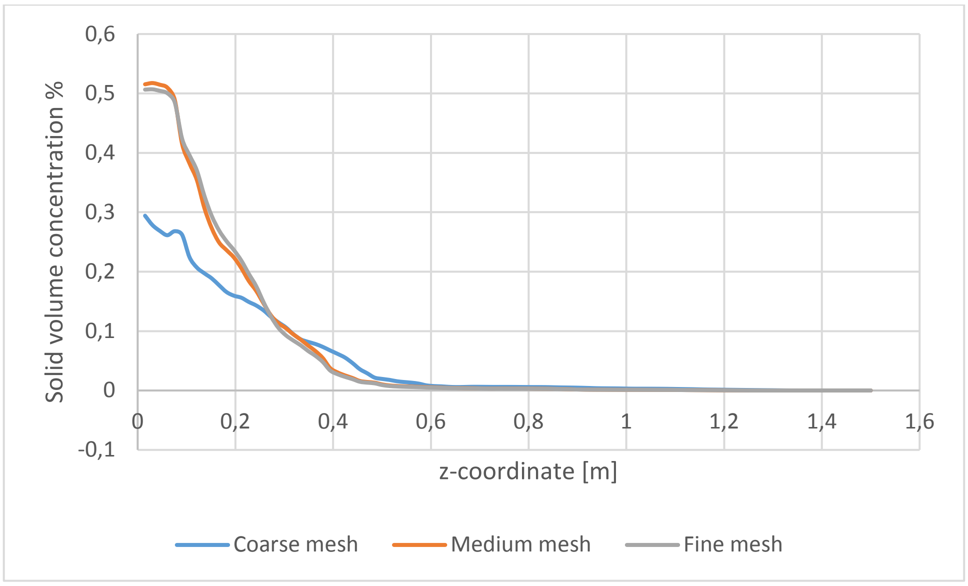


Figure 16 Average solid volume concentration along the riser height

The case is run for 10 sec and comparisons is done for the pressure drop and solid volume concentration along the centerline of the riser. Figure 15 and Figure 16 shows the pressure drop and solid volume concentration respectively. The pressure drop for all the cases shows similar trends not much difference can be seen. For the solid volume concentration the coarse mesh is not able to predict the same density in the bottom bed as medium and fine mesh is able to predict. The highest difference in the average solid volume concentration difference is only ≈ 2% between the medium and fine mesh. Total mesh independence is not achieved, but the medium mesh is adequate to predict valid results. The medium mesh is therefore selected for case 2.45, 2.60 and 2.70.

Case 1.45, 1.60, 1.70 and 1.100 is run with refinement along the walls and in the bottom bed. For these cases the refinement regions have a particle to cell width ratio of 1.40% and the center of the riser have ratio 0.92%, see Figure 14 for illustration.

Table 7 Amount of cells in each mesh type

Mesh Type	**Total number of cells**
Coarse	146374
Medium	201358
Mesh group A	209422
Fine	400522

6.3 Boundary conditions

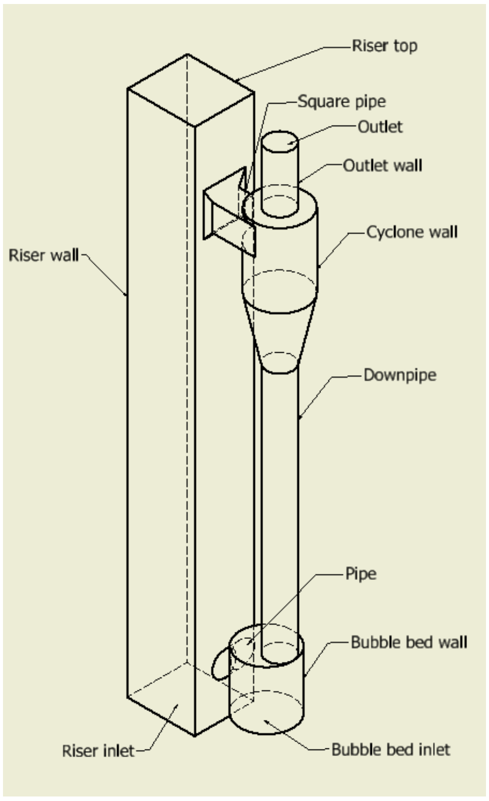


Figure 17 Patch names for the geometry

To see the difference between the two case setups they are presented in separate sub chapters. Each of the wall patches have the same boundary conditions, so they are referred to as only walls in the boundary condition tables. Figure 17 shows the names of all the patches. The new OpenFOAM have greatly improved how the case files is set up for two phase systems. Each phase have now their own boundary condition files in the 0 folder. Making the conditions for each phase easier to specify and to read for the users.

6.3.1 Group A

The boundary conditions for the continuous gas phase is listed in Table 8. At the inlet, the velocity is known and it is described by the boundary condition interstitialInletVelocity. This is a velocity condition that can take into account that more than one phase is present as the inlet input. For this case the velocity input is 1.0 m/s and only gas phase is present. The same boundary condition is used for the bubble-bed inlet with velocity 0.1 m/s. The dynamic pressure uses a fixedFluxPressure condition for both the inlets, and the wall surface a prghPressure condition is used at the outlet. The prghPressure is a newly adapted dynamic pressure condition used to define the pressure at the outlet, here set to 100000 Pa. The static pressure is calculated based on the dynamic pressure. The velocity at the outlet is calculated based on the pressure by using the pressureInletOutletVelocity. The gas temperature is set to 300 K.

Table 8 Boundary conditions for gas phase, group A

Patch	**Boundary Condition**			
	U.air	**T.air**	**alpha.air**	**p_rgh**
riser inlet	interstitialInletVelocity; inletVelocity (0 0 1.0); alpha alpha.air;	fixedValue; uniform 300;	zeroGradient;	fixedFluxPressure;
bubblebed inlet	interstitialInletVelocity; inletVelocity (0 0 0.1); alpha alpha.air;	fixedValue; uniform 300;	zeroGradient;	fixedFluxPressure;
outlet	pressureInletOutletVelocity; uniform (0 0 0);	zeroGradient;	zeroGradient;	prghPressure; p uniform 100000;
walls	fixedValue; uniform (0 0 0);	zeroGradient;	zeroGradient;	fixedFluxPressure;
internalField	uniform (0 0 0);	uniform 300;	Uniform 0;	uniform 100000;

The boundary condition for the particle phase is listed in Table 9. The particle velocity at the riser inlet is set to 0, since only gas phase is present. Some particles are able to escape the cyclone and go through the outlet. To make up for the loss of particles a flowRateInletVelocity is used at the bubble-bed inlet to introduce particles to the system, this way the amount of particles are kept stable during the whole simulation. For the particle-wall interaction a JohnsonJacksonParticlesSlip condition is used for the walls. This is a partial-slip wall condition. For the granular temperature a JohnsonJacksonParticleTheta condition is used, this condition can specify a restitution coefficient for the particle wall collision. The temperature is set to 300K for the particles as well.

Table 9 Boundary condition for the particle phase, group A

Patch	**Boundary Condition**			
	U.particles	**T.particles**	**alpha.particles**	**Theta.particles**
riser inlet	fixedValue; value uniform (0 0 0);	zeroGradient;	zeroGradient;	fixedValue; value uniform 1e-4;
bubblebed inlet	flowRateInletVelocity; phi phi.particles; massFlowRate 0.05; rhoInlet 7800;	zeroGradient;	zeroGradient;	fixedValue; value uniform 1e-4;
outlet	fixedValue; value uniform (0 0 0);	inletOutlet; phi phi.particles; inletValue uniform 300; value uniform 300;	zeroGradient;	zeroGradient;
walls	JohnsonJacksonParticlesSlip; specularityCoefficient 0.01; value uniform (0 0 0);	zeroGradient;	zeroGradient;	JohnsonJacksonParticleTheta; specularityCoefficient 0.01; restitutionCoefficient 0.85;
internalField	uniform (0 0 0);	uniform 300;	uniform 0;	Uniform 0;

Reason for having uniform 0 for both the alpha phases is that the setFieldDict from the system file is run before the simulation starts to distribute the volume concentration of both the phases, this is shown later in Figure 18.

6.3.1.1 *Johnson and Jackson*

The no-slip boundary condition has been applied for the gas-phase velocity, this is the Dirichlet condition and needs no explanation. Johnson and Jackson proposed a partial slip boundary condition for granular temperature, and the solid phase velocity. The normal velocity for the solid phase velocity is set to 0 m/s at the wall surface. The tangential velocity and granular temperature of the solid phase can be written as [24]:

$$u_{st,w} = -\frac{6\mu_s \alpha_{s,max}}{\phi \pi \rho_s \alpha_s g_0 \sqrt{3\Theta}} \frac{\partial u_{st,w}}{\partial n} \tag{75}$$

$$\Theta_w = -\frac{k_s \Theta}{\gamma_w} \frac{\partial \Theta_w}{\partial n} + \frac{\sqrt{3}\pi \phi \rho_s \alpha_s u_{s,slip}^2 g_0 \Theta^{\frac{3}{2}}}{6\gamma_w \alpha_{s,max}} \tag{76}$$

With

$$\gamma_w = \frac{\sqrt{3}\pi(1 - e_w^2)\alpha_s \rho_s g_0 \Theta^{\frac{3}{2}}}{4\alpha_{s,max}} \tag{77}$$

Where $e_w$the particle-wall restitution coefficient and ϕ is the specularity coefficient.

6.3.2 Group B

The boundary conditions for the continuous gas phase is listed in Table 10. At the inlet the total pressure is set to be 102700 Pa. The velocity is then calculated based on the pressure using the pressureInletVelocity condition. At the outlet the velocity is known to be 7.3 m/s, to get the correct mass flow, and the pressure is set to zeroGradient. The dynamic pressure is set to fixedFluxPressure for the walls and bubble-bed inlet. When using the pressureInletVelocity at the inlet it is important that the phase is specified. Since gas is the only phases wanted as the input it need to be specified as uniform 1 in the alpha.air file. The temperature is set to 300K.

Table 10 Boundary condition for the gas phase, group B

Patch	Boundary Condition			
	U.air	**T.air**	**alpha.air**	**p_rgh**
riser inlet	pressureInletVelocity; phi phi.air; alpha alpha.air; value uniform (0 0 0);	fixedValue; uniform 300;	fixedValue; uniform 1;	totalPressure; p0 uniform 102700; U U.air; gamma 0.0; phi phi.air; uniform 102700;
bubblebed inlet	fixedValue; uniform (0 0 0);	zeroGradient;	zeroGradient;	fixedFluxPressure;
outlet	fixedValue; uniform (0 0 7.3);	zeroGradient;	fixedValue; uniform 1;	zeroGradient;
walls	uniform (0 0 0);	zeroGradient;	zeroGradient;	fixedFluxPressure;
internalField	uniform (0 0 0);	uniform 300;	Uniform 0;	uniform 100000;

The boundary conditions for the particle phase is listed in Table 11. The only difference between the particle phase boundary condition for group A and B is that the particle-wall interactions could not be described in the same way. With the JohnsonJackson partial-slip conditions the simulations had problems with exceeding the maximum packing limit given. Since it is important that the maximum packing number is not exceeded the case was run with no-slip velocity conditions for the wall. The temperature is set to 300K.

Table 11 Boundary conditions for the particle phase, group B

Patch	Boundary Condition			
	U.particles	**T.particles**	**alpha.particles**	**Theta.particles**
riser inlet	fixedValue; value uniform (0 0 0);	zeroGradient;	fixedValue; uniform 0;	fixedValue; value uniform 1e-4;
bubblebed inlet	flowRateInletVelocity; phi phi.particles; massFlowRate 0.05; rhoInlet 7800;	fixedValue; value uniform 300;	zeroGradient;	fixedValue; value uniform 1e-4;
outlet	fixedValue; value uniform (0 0 0);	inletOutlet; phi phi.particles; inletValue uniform 300; value uniform 300;	zeroGradient;	zeroGradient;
walls	fixedValue; value uniform (0 0 0);	zeroGradient;	zeroGradient;	zeroGradient;
internalField	uniform (0 0 0);	uniform 300;	uniform 0;	uniform 0;

6.4 SetFields

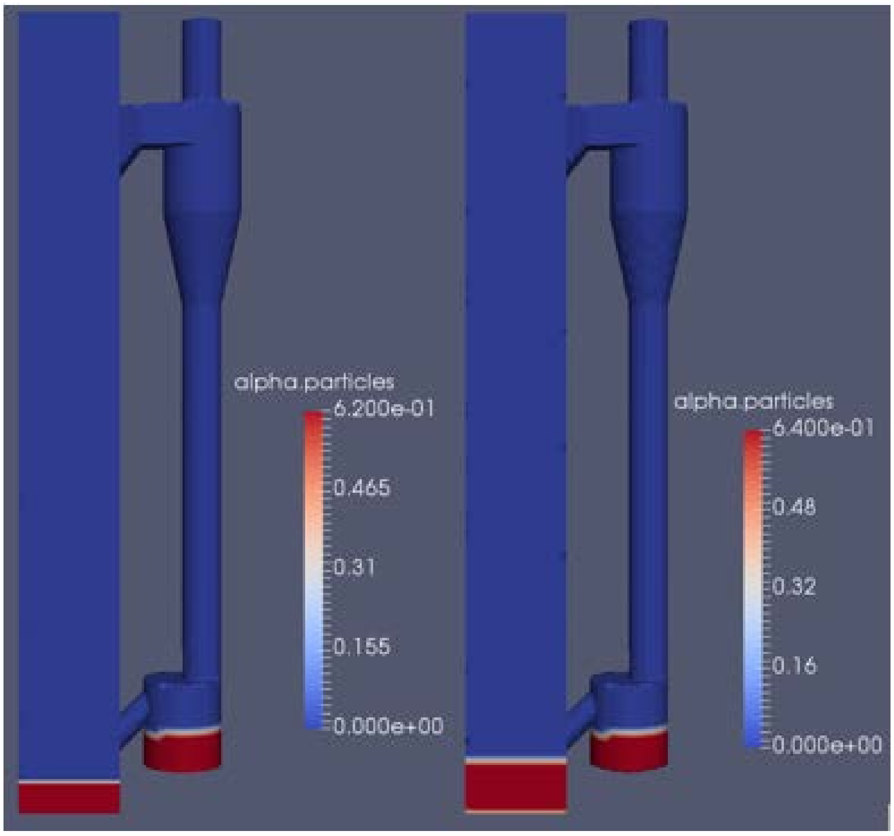


Figure 18 Amount of particles added using setfield for group A (left) and group B (right)

Particles are added to the cases using setFields utility. Figure 17 illustrates the amount that was needed to get the correct pressure drop of 2.7 kPa for group A and group B. Group A used about 30% less particles to have the same pressure drop as group B.

6.5 Thermophysical modelling

The thermophysical properties is the same for cases within group A and B. Even though the thesis is done on a cold CFB boiler, the thermophysical properties could not be neglected by the solver. To keep the temperature constant in the system a value of 300 K was used for the gas and particles phase. For the twoPhaseEulerFoam solver the appropriate thermophysical properties is described in separate dictionaries for each phase, the files is located in the constant folder. Table 12 lists the models used for the gas phase, and Table 13 lists the models used for the particle phase.

Table 12 Thermophysical properties for the gas phase

Keyword	Selected model	Description
Type	heRhoThermo	General thermophysical for compressibility solvers
Mixture	pureMixture	Calculation for passive gas mixtures
Transport	Const	Assume constant dynamic viscosity and Prandtl number
equationOfState	perfectGas	Ideal gas equation of state
thermo	hConst	Assume constant specific heat capacity and heat of fusion
Energy	sensibleInternalEnergy	Specifies that internal energy is the form used, does not include heat of formation

Table 13 Thermophysical properties for the partcle phase

Keyword	Selected model	Description
Type	heRhoThermo	General thermophysical for compressibility solvers
Mixture	pureMixture	Calculation for passive gas mixtures
Transport	Const	Assume constant dynamic viscosity and Prandtl number
equationOfState	rhoConst	Constant density
thermo	hConst	Assume constant specific heat capacity and heat of fusion
Energy	sensibleInternalEnergy	Assume constant specific heat capacity and heat of fusion

The simulations is run with a gas density, $\rho_g = 1.20\ kg/m^3$. The solid particle density is, $\rho_s = 7800\ kg/m^3$. The laminar gas viscosity, $\mu_g = 1.84 \cdot 10^{-5}\ kg/ms$.

The Prandtl number needs to be specified for the gas phase, it is described as the ratio between the momentum diffusivity and the thermal diffusivity. A constant value of 0.7 was selected.

6.6 Phase properties

For the interaction between the particle phase and the continuous gas phase a single file named phaseProperties takes care of it. In the properties file the diameter model and size of the solid and gas phase are specified. The maximum packing limits of particles, what type of blending that is occurring in the analysis, stresses between the phase, drag model, virtual mass, heat transfer, lift and wall lubrication can be each given.

For all the cases run the diameter model for the particles is constant. The maximum packing limit of the solid phase is set to 0.64. The drag model that is used is Gidaspow. Which is explained in chapter 3. Ranz Marshall is used for heat transfer, since it is needed to run the solver. Virtual mass, lift and wall lubrication is all neglected, due to the fact that the density difference between the phases is high, it will have close to zero impact. The phaseProperties file is found in Appendix B.6.

6.7 Turbulence modelling

In the turbulenceProperties.air file in the constant folder the gas-phase turbulence model can be specified. For the turbulence of the gas-phase a large eddy simulation (LES) technique is used. The LES is modeled by subgrid-scale (SGS) model proposed by Smagorinsky. Details for the Smagorinsky model was given in chapter 3.

The conservation equations for the solid phase are based on the kinetic theory of granular flow (KTGF). The models used in the KTGF can be found in the turbulenceProperties.particle file, and is listed in Table 14. The full description of the file is in Appendix B.7.

Table 14 Models used for the kinetic theory of granular flow

Models used in OpenFOAM	
Kinetic particle pressure model	Lun
Frictional particle pressure models	Johnson and Jackson
Radial distribution model	Sinclair and Jackson
Kinetic viscosity model	Gidaspow
Granular conductivity model	Gidaspow
Frictional viscosity model	Johnson and Jackson

7 Results and discussion

In this chapter the most important results from the simulations are presented. Validation of the result is given by comparing the numerical predictions with the experimental data.

7.1 Results for Group A

7.1.1 Pressure drop

For each case presented in Figure 19 the riser was filled with 9kg of bronze particle to get the equivalent pressure drop to the full scale boiler. The diameter distribution of the particles will impact the flow pattern in the CFB. In the two phase simulation only a mean diameter of the particle can be described. Figure 19 shows the pressure drop along the height of the riser for all cases within group A.

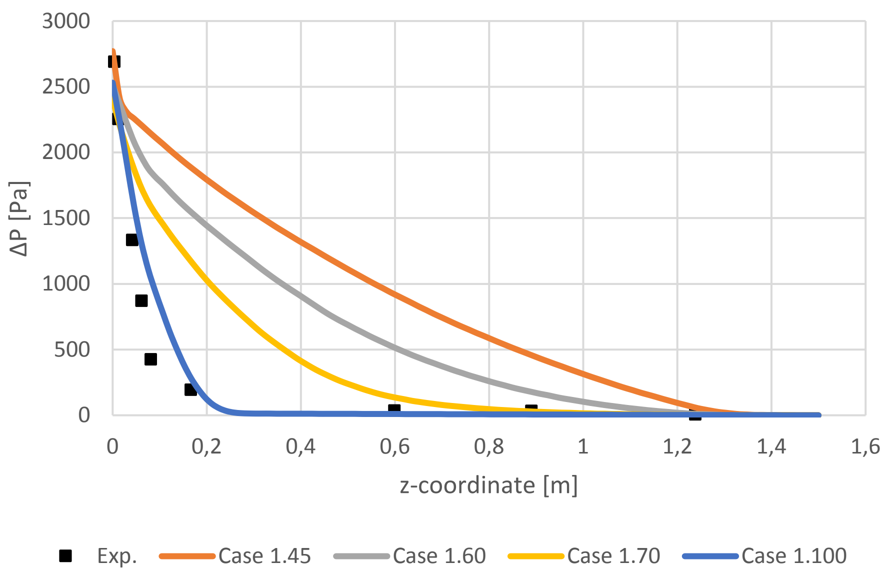


Figure 19 Pressure drop along the height of the riser, group A

To be able to evaluate the numerical result the solid flux out of the riser needs to be compared to the measured net solid flux. Table 15 shows the flux out of the riser for each case in group A.

Table 15 Calculated fluxes out of the riser exit. Experimental value is from [1]

Group A	**[kg/m²s]**
Case 1.45	305
Case 1.60	43,7
Case 1.70	2.0
Case 1.100	0
Experimental	2.0

The mean particle diameter size have a clear impact on the pressure drop in the riser. When the diameter increase the pressure drop trends towards experimental data. The net solid flux out of the riser are overestimated for cases with a smaller diameter size of 70 µm. For case 1.70 the predicted flux is the same as the measured one. Figure 20 shows the average solid volume concentration along the height of the riser.

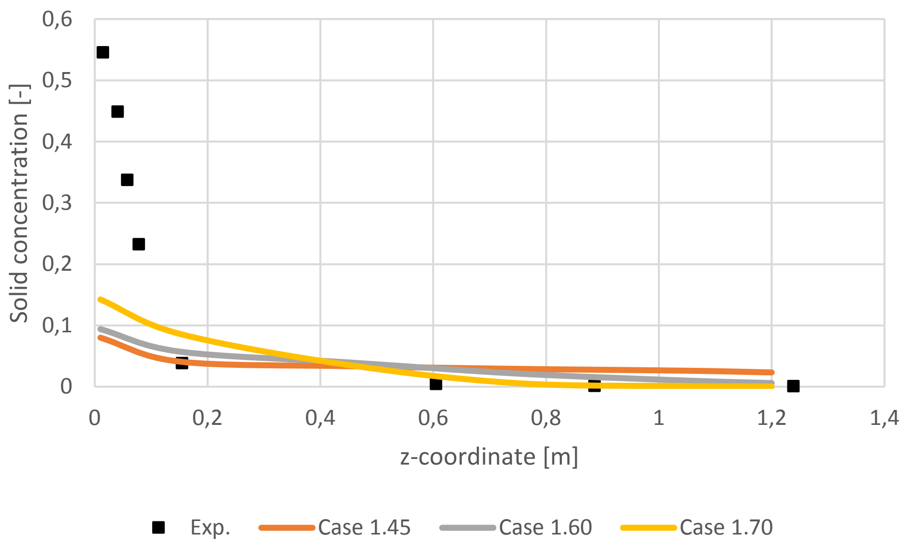


Figure 20 Average solid volume concentration along the riser height, group A

For case 1.45 the flux of circulating particles gives higher solid concentration in the upper part of the riser, therefore the dense bottom bed that are visualized during the experiments is not present in the simulation. This high solid concentration in the upper part of the riser fits with the overestimated flux shown in Table 15. With particle size 70 μm the bottom bed is observed to have about twice the density of solid particles compared with the case run with particle size 45 μm.

Figure 21 shows the particles axial velocity along the horizontal profil.1. Location of profile 1 and 2 is described in chapter 5.

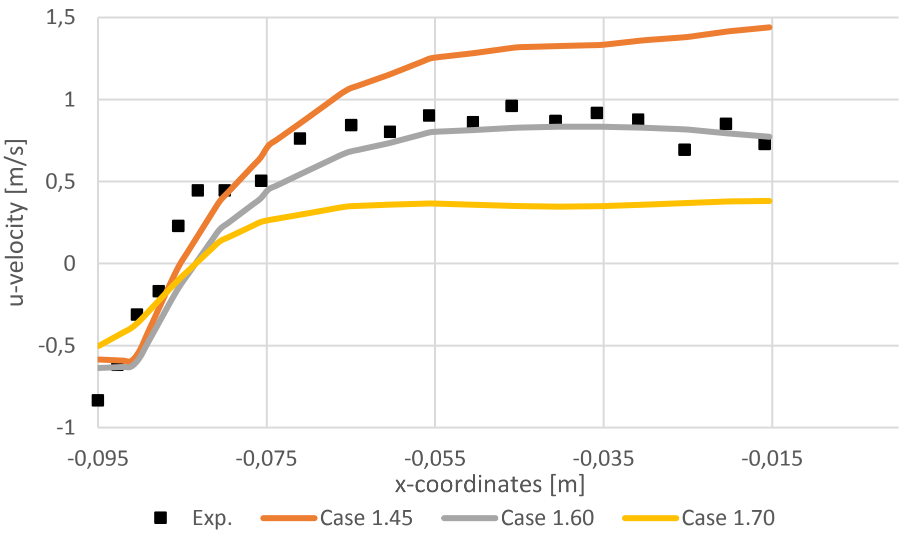


Figure 21 Axial particle velocity along profile 1, group A

Each case follow the same trends as the experimental data. Particle diameter of 45μm over estimates the velocity in the center of the riser, but fits the experimental profile well close to the wall. The mean diameter from the diameter distribution of the experimental setup is 60 μm, and for the representative case the axial velocity fits the experimental data well along the whole profile. Case 1.70 has agreement with the net solid flux out of the riser, but since the diameter is high the drag on the particle is low and the axial velocity will decrease as expected.

Figure 22 shows the spanwise particle velocity along profile 1.

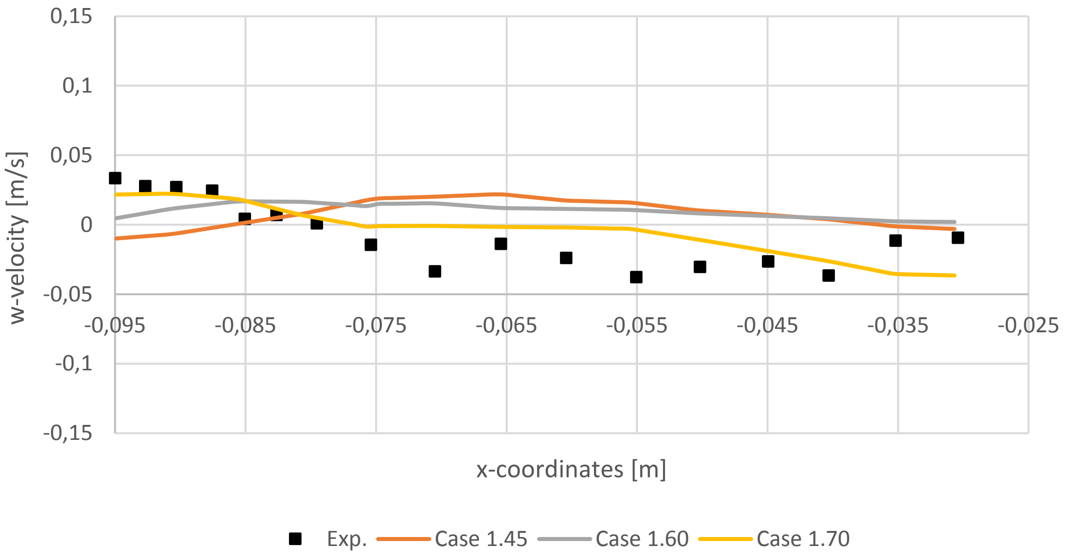


Figure 22 Spanwise particle velocity along profile 1, group A

The spanwise particle velocity compares relatively well with the experimental findings. The particle diameter does not seem to effect the spanwise velocity the same way as for the axial velocity.

Figure 23 shows the axial particle velocity along profile 2.

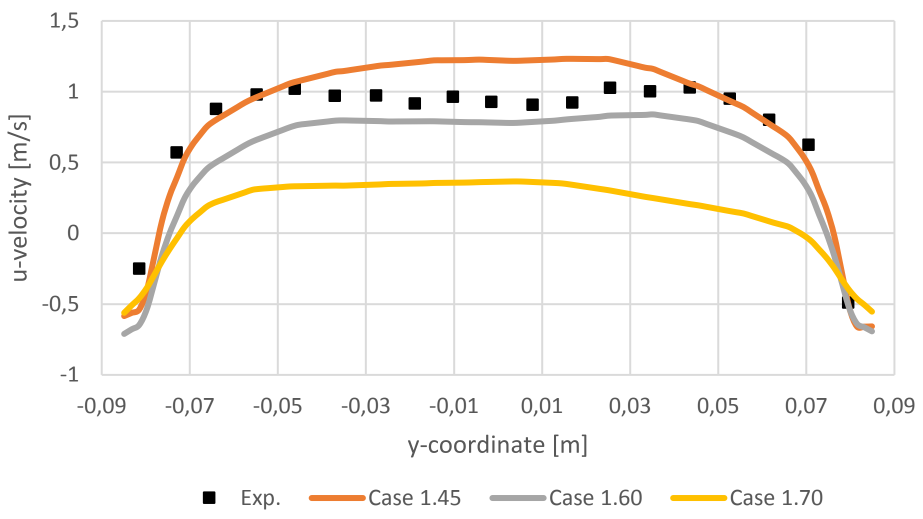


Figure 23 Axial particle velocity along profile 2, group A

The axial velocity along profile 2 follow the same trends as the experimental data. The particle dimeter size effects the magnitude of the axial velocity the same way for profile 1 and profile 2. Symmetry along the centerline of the riser can be seen, this indicates that the averaging time for the simulation were long enough. For case 1.60 and case 1.70 the velocity profile is flat in the center of the riser as is seen in the experiment. Figure 24 shows the spanwise particle velocity along profile 2.

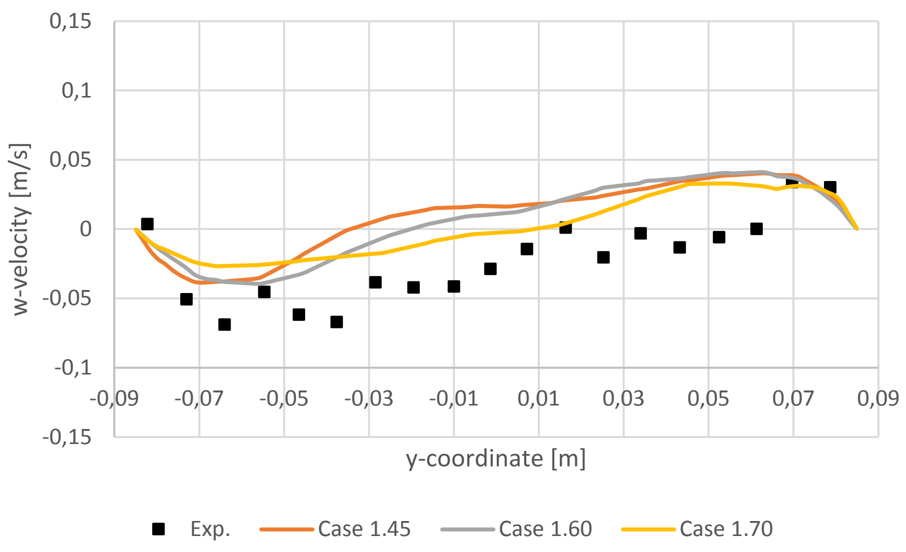


Figure 24 Spanwise particle velocity along profile 2, group A

The spanwise velocity along profile 2 shows that the diameter size have little effect on the horizontal motion in the riser. Negative values can be seen for both the simulation and the experimental data. This can be explained by a swirling motion that was visually observed during the experiment. From the simulated results the swirling motion is more symmetric along the center than the experimental data.

7.1.2 Particle RMS velocity

In the multiphase gas-particle flow model a turbulence velocity can be obtained. To get the total particle RMS velocity the large scale mean fluctuation velocity from the LES model, and the small scale fluctuation of the granular temperature needs to be added. The small and large scale fluctuations are assumed to be statistically independent. From Eq. 78 the total particle RMS velocity can be obtained [25]:

$$U_{i,RMS,tot} = \sqrt{\bar{\Theta} + U_{i,RMS}} \tag{78}$$

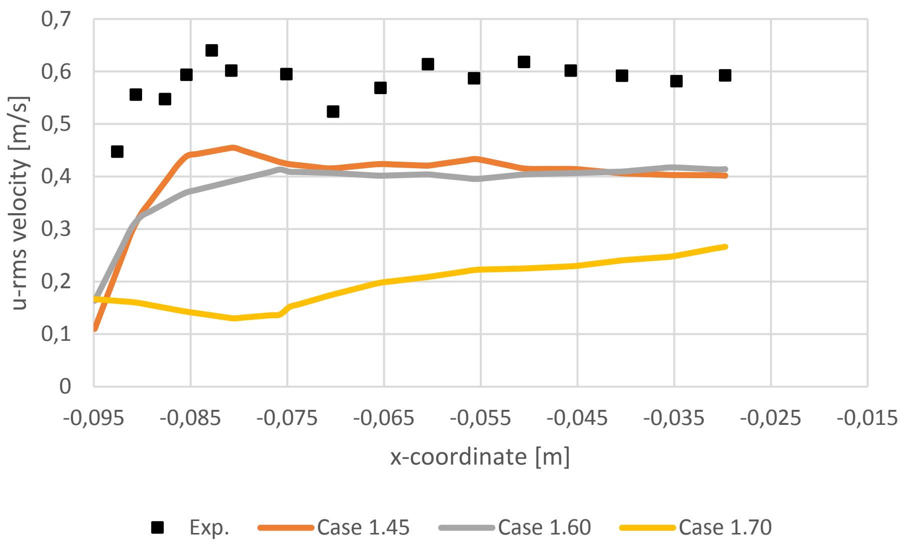


Figure 25 Axial RMS velocity along profile 1, group A

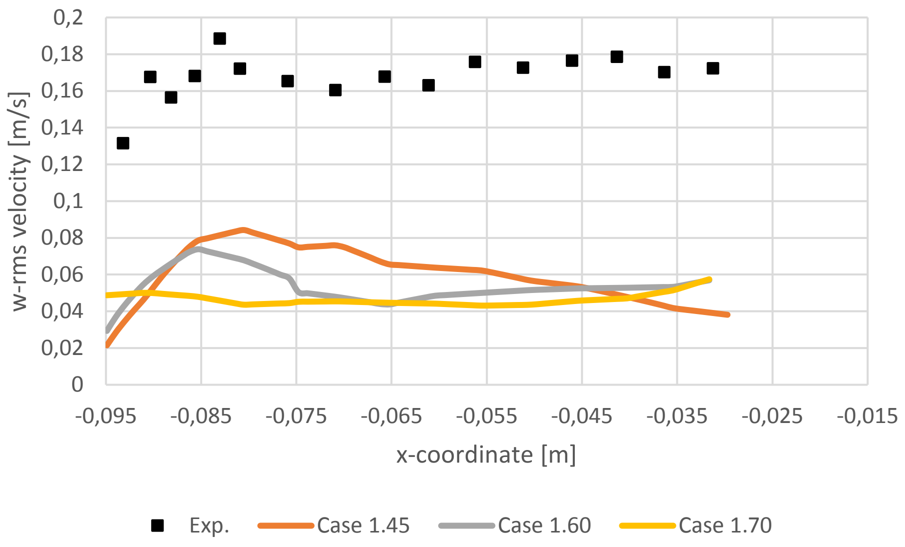


Figure 26 Spanwise RMS velocity along profile 1, group A

Figure 25 and Figure 26 shows the total RMS velocity for axial and spanwise velocity along profile 1, respectively. The numerical prediction of the fluctuating axial velocity is underestimated compared to the experimental data. The fluctuation towards the center of the riser is constant, which corresponds well with the numerical prediction for small particle size. The spanwise fluctuation is underestimated compared to the experimental data, but the values follow the same trend towards the center.

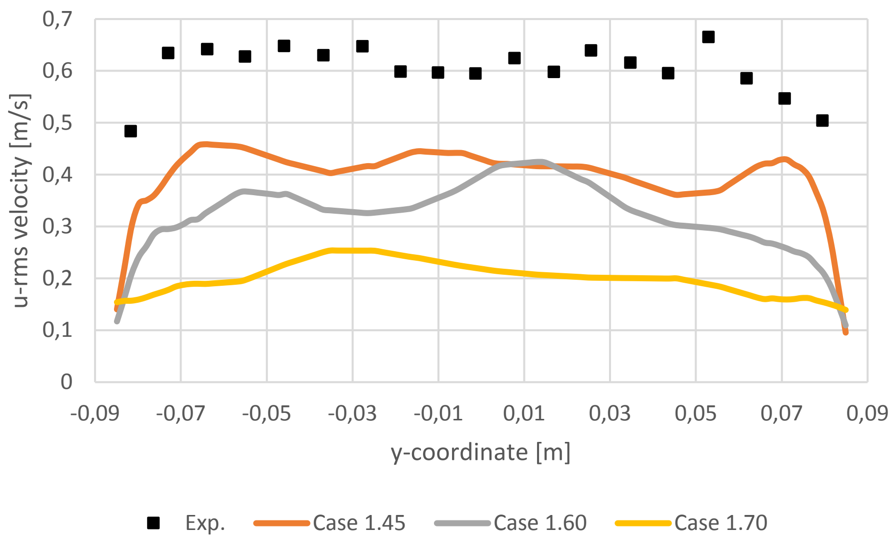


Figure 27 Axial RMS velocity along profile 2, group A

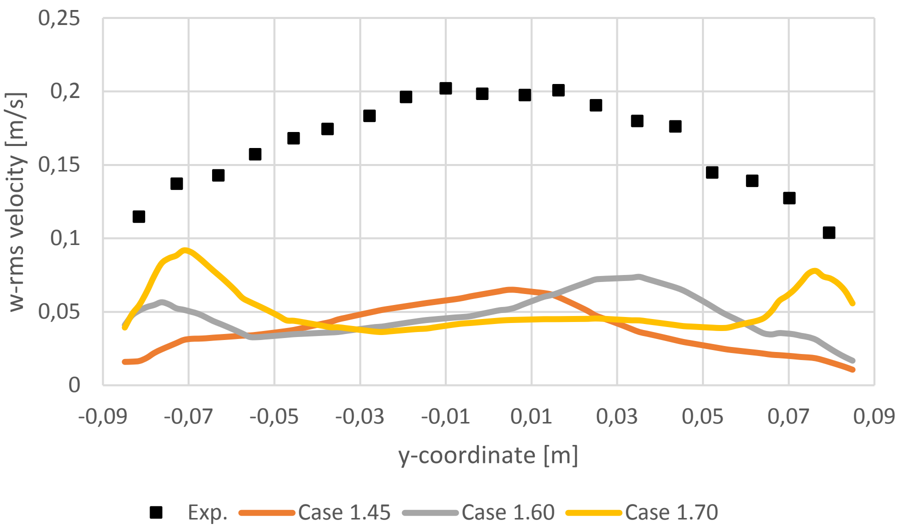


Figure 28 Spanwise RMS velocity along profile 2, group A

Figure 27 and Figure 28 shows the total RMS velocity for axial and spanwise velocity along profile 2, respectively. In the same way as for profile 1 the fluctuation is underestimated compared to the experimental data. The numerical results predicts a considerable lower fluctuation along the wall, were the experimental data suggests a higher value. For the spanwise fluctuation the simulations predict an almost constant value toward the center of the riser, while the experimental data suggest a more parabolic behavior.

This low values close to the wall may have something to do with the wall boundary conditions that are used in the numerical simulation. This indicates that the specularity coefficient of 0.01 chosen for the particle wall interaction is too high. Figure 29 shows the difference in fluctuation close to the wall for Case 1.45 with a no-slip and a specularity coefficient along profile 1.

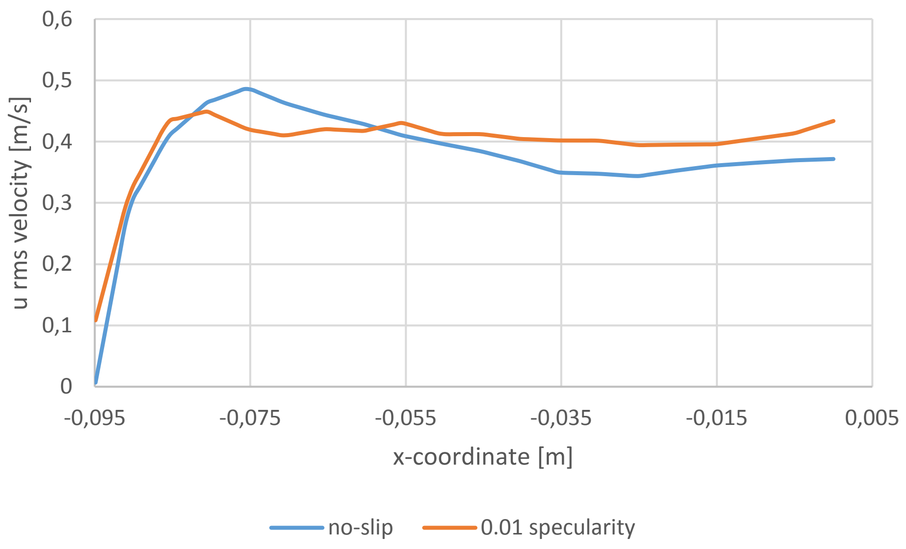


Figure 29 Axial RMS velocity compared for no-slip and johnsonJackson condition, group A

Figure 29 shows that the velocity fluctuation along the wall increases with the use of a specularity coefficient. This is as expected since a specularity coefficient of 0.01 is close to a slip condition.

7.1.3 Summary discussion

With a uniform inlet velocity the dense bottom bed, and the steep pressure drop that occur due to the bottom bed is not predicted in the numerical simulation. By visualizing the particle behavior in the bottom of the riser it is seen that a turbulent regime occur very fast. This is the same type of behavior that Ibsen got during his work using the same type of flow conditions[1].The numerical predictions were found to be sensitive to the representative particle diameter. Case 1.70 were able to predict the correct flux out of the riser, but the velocity is underestimated along both the profiles. Case 1.60 with a diameter representation of 60 µm is seen to compare very well to the experimental data, for both the axial and spanwise velocity.

Overall the simulations with a uniform plug velocity for the superficial gas velocity is corresponding very well between the OpenFOAM code used in this thesis, and the Euler/Euler code used in the previous FLOTRACE simulations done by Ibsen [1]. This is an indicator that OpenFOAM can contest the commercial software’s used for multiphase flows.

7.2 Results for Group B

The prediction done for group A when it comes to pressure drop and solid volume concentration along the riser height does not fit well with the experimental results. The fast turbulent regime that occurs makes the prediction of the dense bed difficult, which results again results in a pressure drop does not fit the experimental data. By changing the superficial gas velocity from being uniform to be calculated from the pressure flux on the inlet, hopefully this will be able to predict the bubbling eruption through the bottom as it is seen in the experiment.

7.2.1 Pressure drop and solid volume concentration

The Figure 30 shows the pressure drop along the riser height for three different particle diameters used.

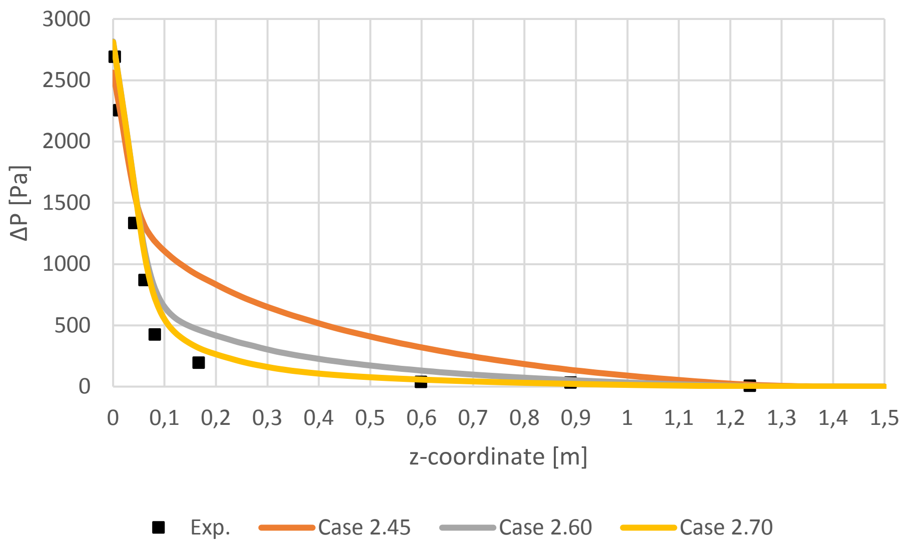


Figure 30 Pressure drop along the height of the riser, group B

Figure 30 shows that a steep pressure drop is predicted for all the cases. This indicates that there is a higher solid volume concentration in the bottom of the riser. The flux out of the riser is presented in Table 16. The flux for the cases is not as overestimated as for cases in group A, but fluxes for small particles are still too high. Case 2.70 with particle diameter of 70 µm compares well with the experimental data for the particle flux out of the riser. Numerical prediction with representative diameter of 70 µm predicts the solid flux out of the riser best for both group A and group B. Figure 31 shows the average solid volume concentration along the height of the riser.

Table 16 Calculated flux out of the riser exit. Experimental value taken from [1]

Group B	**[Kg/M^2s]**
Case 2.45	55.84
Case 2.60	11.30
Case 2.70	2.22
Experimental	2.00

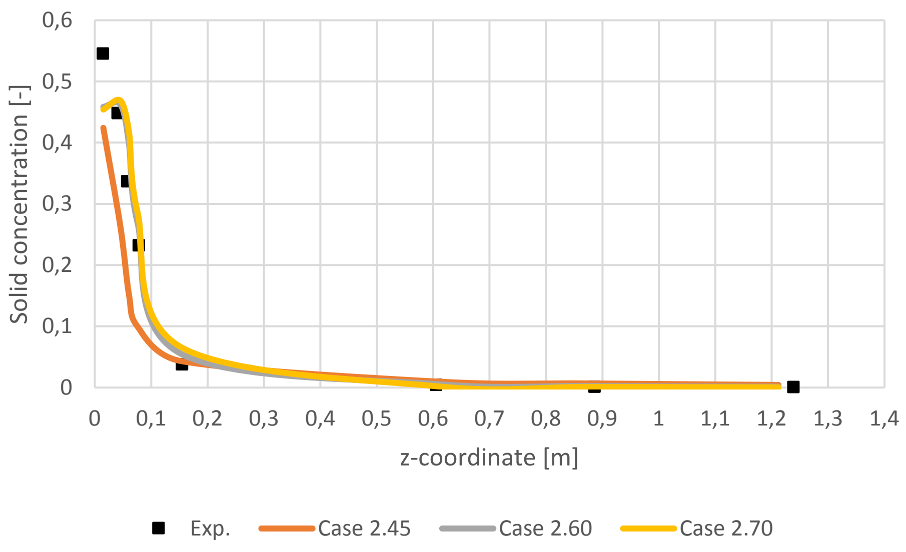


Figure 31 Average solid volume concentration along the height of the riser, group B

The numerical simulation is able to predict a dense bottom bed in the bottom of the riser. Figure 31 shows that all the cases manage to predict a bottom bed of solid density in the range of 40-50%. This is a great improvement from the previous results in this thesis and previous work done on this case experiment, which were between 10-15% solid concentration.[1]

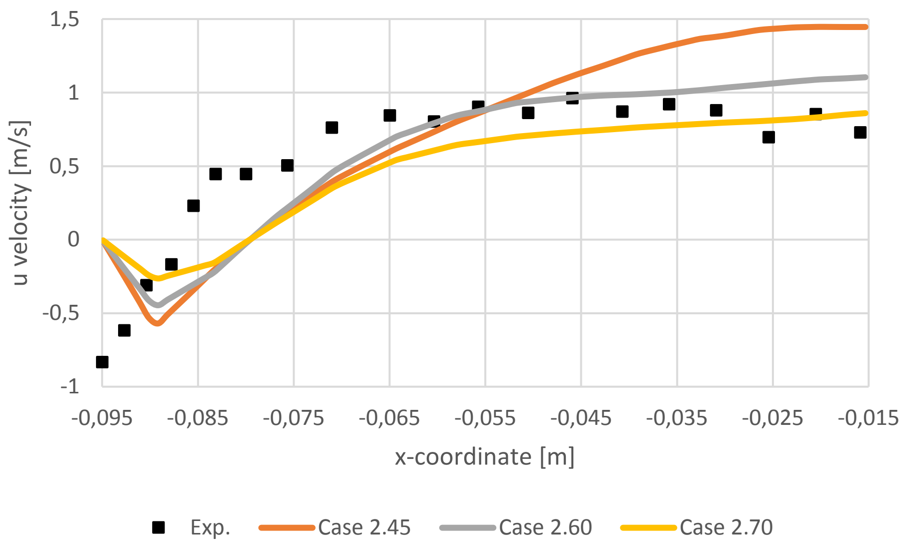


Figure 32 Axial particle velocity along profile 1, group B

Figure 32 shows the axial velocity along profile 1. Due to the fact that the simulation is run with no-slip boundary conditions for the walls, the negative velocity of falling particles are not predicted close to the wall. Case 2.45 and Case 2.60 overestimates the velocity of the particles in the center of the riser. Case 2.70 compares relatively well with the experimental data towards the center of the riser.

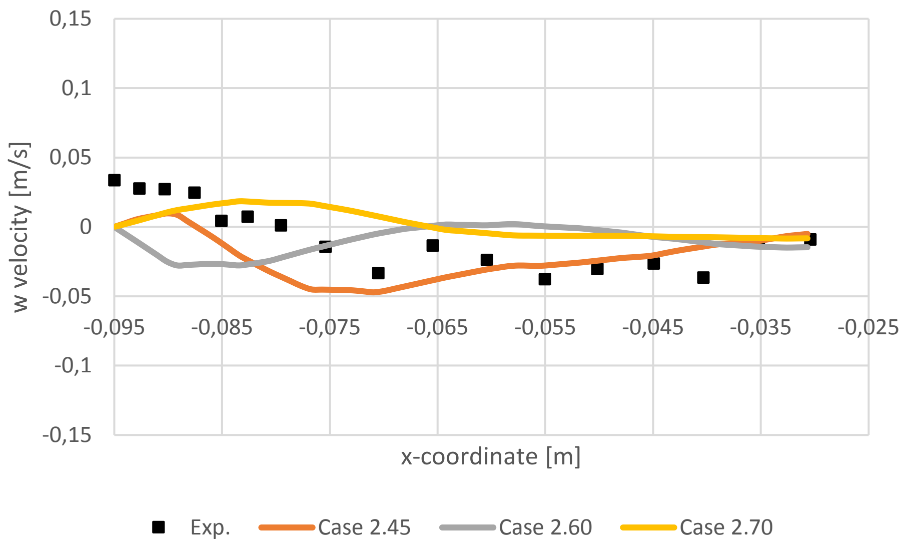


Figure 33 Spanwise particle velocity along profile 1, group B

Figure 33 shows the spanwise velocity along profile 1. The magnitude of the spanwise velocity compares well with the experimental data. The diameter size does not seem to impact the magnitude of the spanwise velocity.

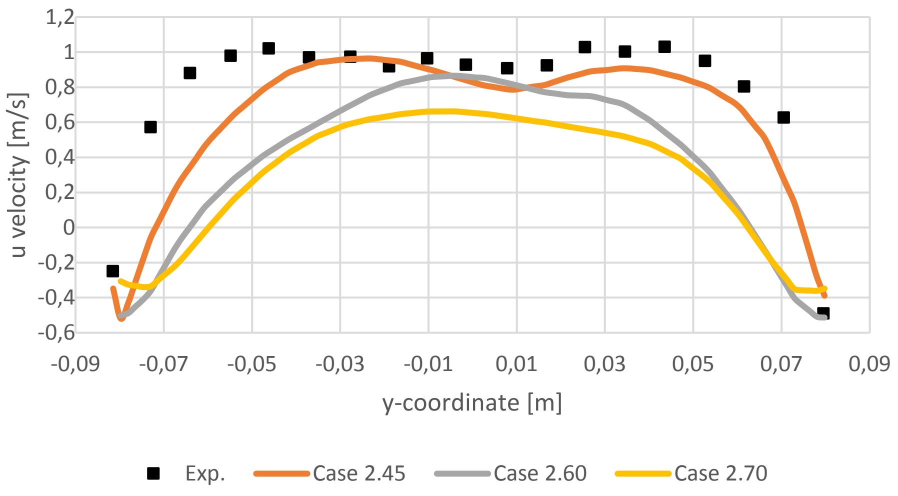


Figure 34 Axial particle velocity along profile 2, group B

Figure 34 shows the axial velocity along profile 2. Case 2.45 compares well with the experimental data. Case 2.60 and Case 2.70 underestimates the velocity. The velocity profile is not symmetric along the center, which can indicates that the simulations should be run for longer averaging time.

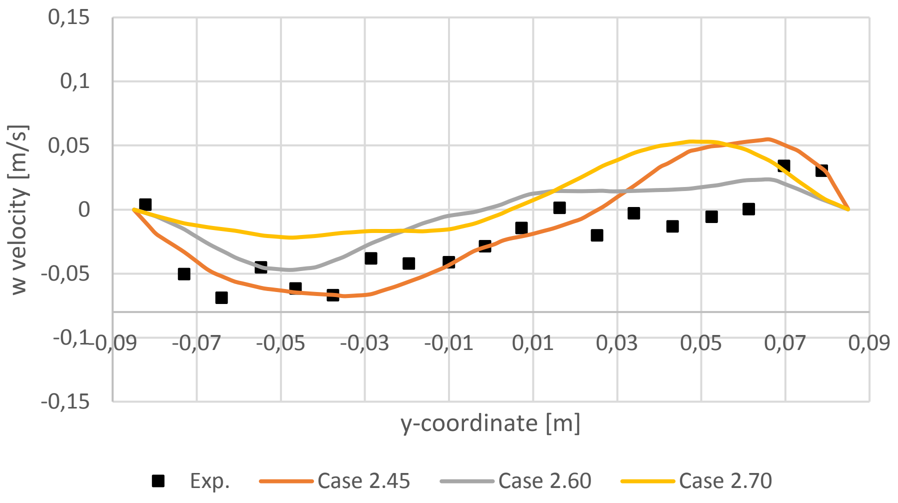


Figure 35 Spanwise particle velocity along profile 2, group B

Figure 35 shows that spanwise particle velocity along profile 2. The same swirling motion that occurred for previous cases can be seen. This fits the trends of the experimental data. The numerical simulations seems to predict more swirling motion then the experimental data suggests.

7.2.2 Particle RMS velocity

The total solid particle RMS velocity is calculated by Eq. 78. Figure 36 and Figure 37 shows the total RMS velocity for axial and spanwise velocity along profile 1, respectively.

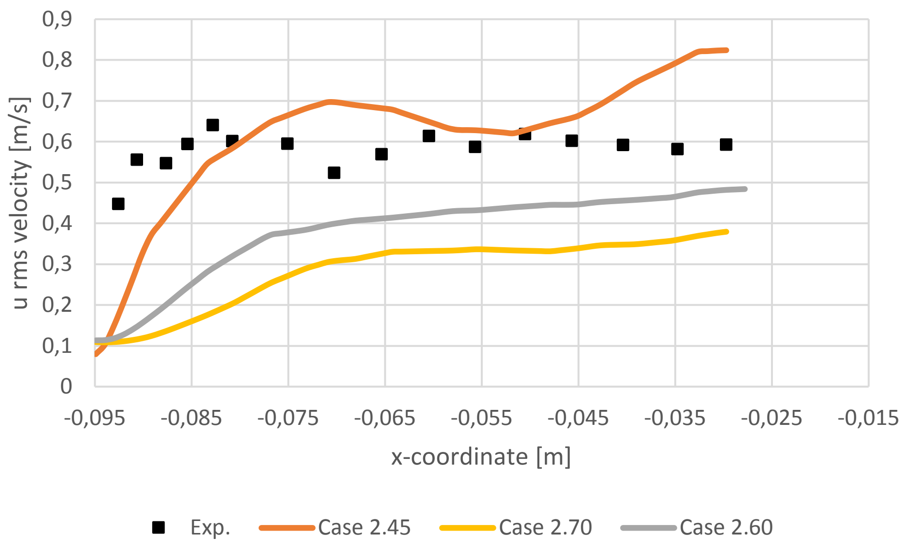


Figure 36 Axial RMS velocity along profile 1, group B

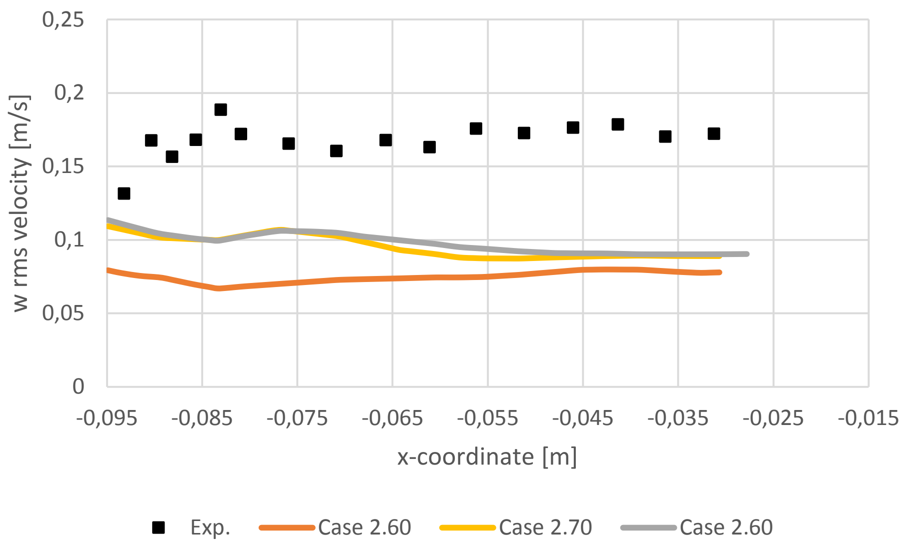


Figure 37 Spanwise RMS velocity along profile 1, group B

The numerical prediction is able to predict the trends of the axial and spanwise fluctuation along profile 1. The magnitude of the fluctuation is underestimated for each case, except for the axial fluctuation of case 2.45. It is expected that the small particles will fluctuate the most.

Figure 38 and Figure 39 shows the total RMS velocity for axial and spanwise velocity along profile 2, respectively.

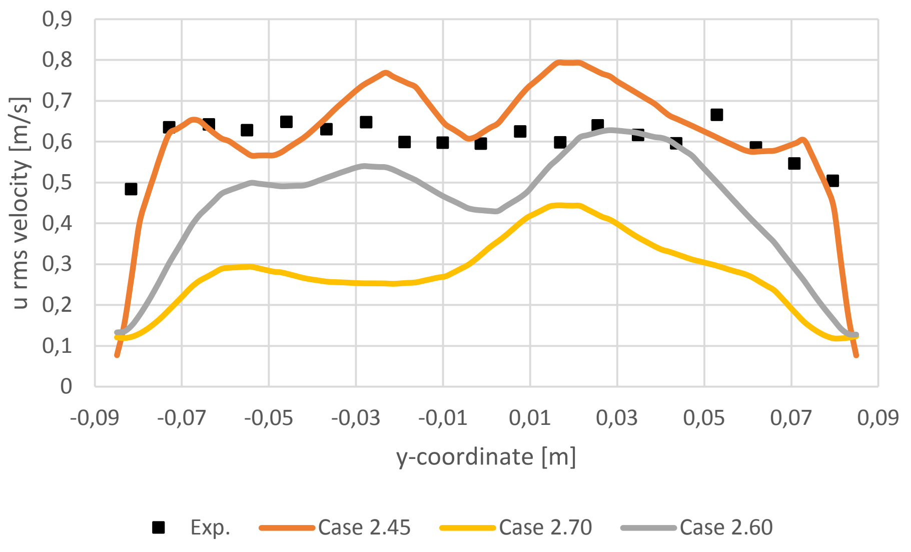


Figure 38 Axial RMS velocity along profile 2, group B

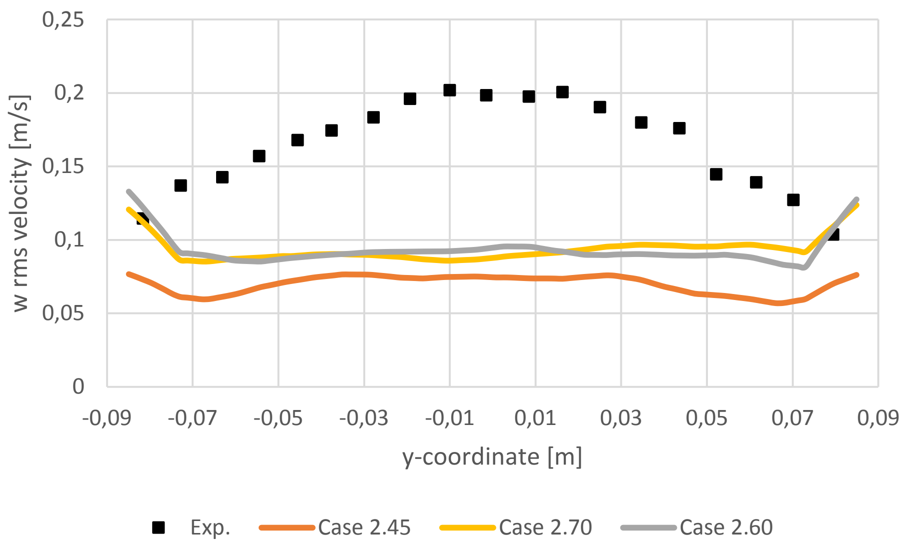


Figure 39 Spanwise RMS velocity along profile 2, group B

The axial fluctuation along profile 1 is underestimated for case 2.70. Case 2.45 slightly overestimated, and case 2.60 slightly underestimates the fluctuation. The fluctuation is not symmetric along the center of the riser. This again may indicate that the simulation for group B needs a longer averaging time than group A. For the spanwise fluctuation the simulations predict an almost constant value toward the center of the riser, while the experimental data suggest a more parabolic behavior. The spanwise fluctuation is underestimated.

The particle kinetic shear stress term $< u'_p v_p' >$ is presented in Figure 40 and Figure 41 for profile 1 and 2, respectively.

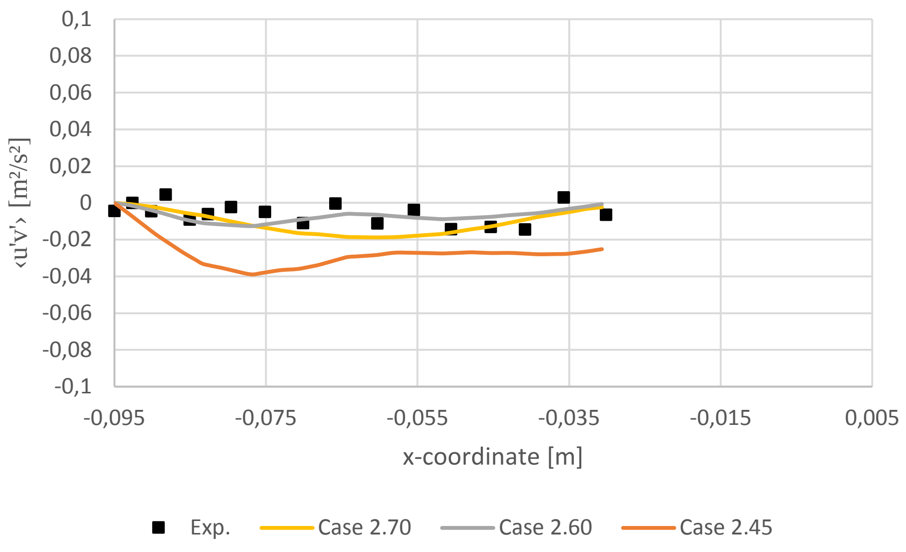


Figure 40 Particle kinetic shear stress along profile 1

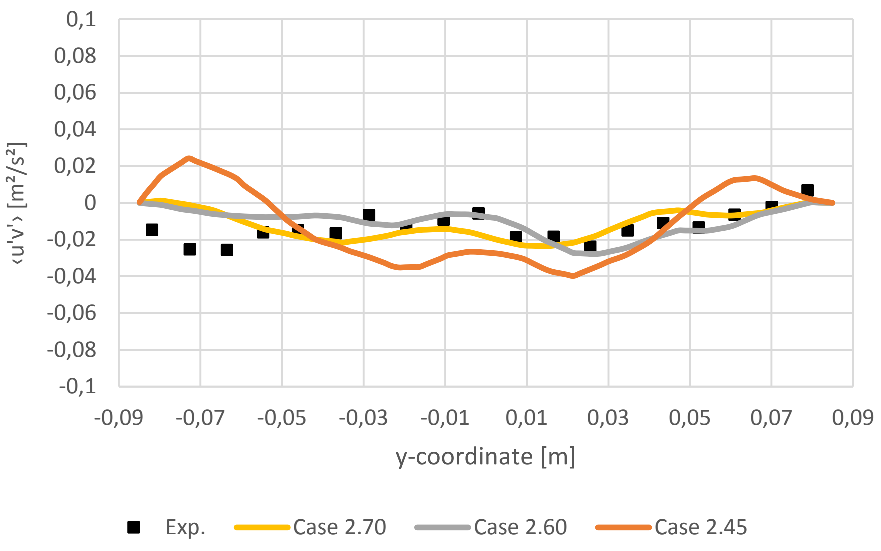


Figure 41 Particle kinetic shear stress along profile 2

Case 2.60 and 2.70 is able to predict the shear stress for both profile 1 and 2 well compared to experimental data. Case 2.45 overestimated the magnitude of the shear stress in for both profiles. The shear stress for profile 1 is very low, indicating that there is almost no flux across the profile. The negative shear stress along profile 2 indicates that some flux is present across the profile. This is expected since as the particles seems to experience a swirling motion.

7.2.3 Residuals

The residual for cases in group A and group B are similar. The residual for Case 2.70 are plotted in Figure 42. Oscillation occur for the residuals, this is mostly due to the constantly changing flow that occur in the CFB. It is obvious that the residuals are sufficiently small, as the final residual for both solid concentration, alpha.particles, and dynamic pressure, p_rgh, are 10^{-8} ,or even less. This low final residual is expected since it is specified in the limits for the solution control, described in chapter 4.7.2

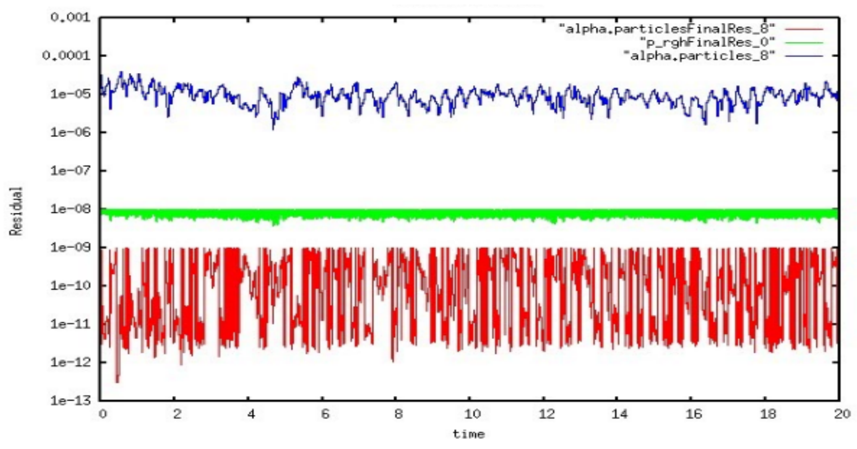


Figure 42 Residual plot for case 2.70

7.2.4 Summary discussion

The simulation does not go directly over to a turbulent regime as for cases in group A. The simulation is able to predict the dense bottom bed that occur in the experimental results. The superficial gas creates bubbles in the bottom bed that erupts and accelerate the particles up the riser. Due to the prediction of the dense bed the pressure drop along the riser height is steep at the bottom of the riser, this fits well with the experimental data. The pressure drop and the solid flux is very well predicted with case 2.70, that has a 70 µm particle diameter representation. The velocity profile with particle size 45µm seems to overestimate the axial velocity. Case 2.70 is able to predict the right axial velocity in the center of the riser, while being underestimated towards the wall.

The numerical predictions were found to be sensitive to the representative particle diameter. The result looks to be physically correct when evaluated against the experimental data. Since only one representative particle diameter is represented in the twoPhaseEulerFoam solver it is not expected that every single velocity parameter will fit the experimental values exactly. The experiment is using a wide particle size distribution that will influence the flow in the riser, one particle size will not be able to account for all of it.

8 Conclusion

A multiphase gas-particle flow analysis of a scaled circulating fluidized bed has been investigated, using the open-source CFD code OpenFOAM. The multiphase flow were simulated using the two-phase solver twoPhaseEulerFoam. The study were focused on how different particle diameter sizes would effect the flow.

The simulations were done using two different approaches for how the flow is driven through the CFB system. First the simulations were run with a uniform plug velocity for the superficial gas velocity. This was done to see if OpenFOAM would generate the same type of turbulent regime that has been seen occurring in similar CFD codes created by previous researchers. The second approach was to set the superficial gas velocity to be calculated based on the pressure flux at the inlet boundary.

The results show that both of the simulation approaches is sensitive to the representative particle diameter. The mean-volume-diameter of 45µm representation of the particle size overestimates the solid flux out of the riser. For the simulations to predict the right amount of solid flux the particle diameter representation had to be increased to 70 µm. Simulations run with a uniform plug velocity did not manage to predict the dense bottom bed that occurred for the experimental results. The flow reached a turbulent regime so that the bed never could settle in the riser.

For simulations that ran a superficial gas velocity calculated by the pressure flux the dense bottom bed were predicted. The pressure drop along the riser height compared well with experimental data when using a representative particle size of 70 µm. This result is very good for a two-phase multiphase code. Previously work done on this type of CFB have only been able to predict this type of pressure drop when they simulate the flow with a range of particle sizes simultaneously. The velocity profiles is predicted to follow the same trends as the experimental results, but they are sensitive to the particle size. For particle size of 45 µm the axial velocity is generally overestimated, and for 70 µm the axial velocity is underestimated towards the walls, but fits well in the center of the riser.

To validate the accuracy of the simulation a grid independent test was performed. The number of cells in the system was increased by 100% compared to the mesh used in the simulation. The resulting change in the pressure drop was negligible, the average solid volume concentration along the riser height difference was $\pm 2\%$. This means that the solution is not completely independent of the grid, but good enough to give valid results.

The choice of turbulence model can also influence the accuracy of the results. A Smagorinsky LES turbulence model was used for the gas-phase. Based on the result of velocity fluctuation the choice of the Smagorisnky constant to be 0.19, might dampen the velocity fluctuation.

Based on the validation against experimental data, it is assumed that the obtained results are physically correct for the case simulated with a calculated pressure flux for the superficial gas velocity. Due to the fact that only one mean particle size can be represented by the two-phase solver the velocity profiles will not fit 100% with the experimental results, since the experiment have a wide particle size distribution.

8.1 Further work

Suggestions for further work that can build on the result of this thesis is listed below:

- Run simulations with a different drag model
- Implement the particle interaction and KTGF models to the multiphaseEulerFoam solver, to run the same type of simulations with multiple particle size simultaneously
- Run the simulations with different LES turbulence models

Reason for why some of this suggestions where not done in this thesis were the time restriction, due to the computational time, and the lack of experience in C++ program language.

9 Bibliography

[1] C.H. Ibsen "An Experimental and Computational study of Gas-Particle Flow in circulating Fluidised Reactors" Ph.D, Aalborg University Esbjerg, 2001.

[2] IAE Clean Coal Center, "Developments in circulating fluidized combustion",2013. [Online]. Available: http://www.iea-coal.org.uk/documents/83172/8753/Developments-in-circulating-fluidised-bed-combustion,-CCC/219 [Accessed: 02.05.2016]

[3] K.G. Hansen, et al., "An Experimental and Computational study of Gas-Particle Flow in a scaled circulating fluidized bed", *Chemical Engineering Laboratory, Alborg University Esbjerg*, 2002.

[4] K.G. Hansen, et al. "A computational study of the distribution of particles in a lab-scaled CFB boiler", *The 12th International Conference on Fludization- New Horizon in Fluidisation*, 2007.

[5] C. T. Crow, Red., *Multiphase flow handbook*. New York: CRC Press, 2006.

[6] D. Gidaspow, *Multiphase flow and fluidization*. Academic Press, 1994.

[7] L.-S. Fan and C. Zhu, *Principle of Gas-Solid Flows*. Cambridge: Cambridge University Press, 1998.

[8] C. T. Crow, et al. *Multiphase flows with droplets and particles*. New York:CRC Press, 1998.

[9] C. Venier, et al. "Development of a conservative numerical solver for gas-particle multi-fluid systems uising kinetic theory of granular flow". Mecánica Computacional, Vol XXXIII, pp. 473-497, 2014.

[10] B.G.M van Wachem. "Derivation, Implementation, and validation of Computer Simulation Models for Gas-Solid fluidized beds", Ph.D. Delft University of Technolgy, 2000.

[11] D. Zang et al. "Numerical simulation of the dynamic flow behavior in a bubble column, a study of closure for turbulence and interface forces. *Chemical engineering science*. Vol 61. Issue 23, pp 7593-7608, 2006.

[12] H.K. Versteeg and W. Malalasekera , *An introduction to computational fluid dynamics*. Harlow, England: Pearson Education Ltd., 2007.

[13] CFD Direct, "OpenFOAM User Guide:3.1 Programmin language, OpenFOAM", 2016. [Online]. Available: http://cfd.direct/openfoam/user-guide/programming-language-openfoam/#x9-640003.1 [Accessed 16.05.2016].

[14] CFD Direct, "OpenFOAM User Guide: 4.1 File structures of OpenFOAM cases", 2016. [Online]. Available: http://cfd.direct/openfoam/user-guide/case-file-structure/#x17-930004.1 [Accessed 16.05.2016].

[15] J. Peltola. "Dynamics in Circulating Fluidized bed: Experimental and Numerical Study". Master thesis, Tampere University of Technology, 2009.

[16] H. Rusche. “Computational Fluid Dynamics of Disppersed Two-Phase Flows at High Phase Fraction”, Ph.D. Imperial college of Science, 2002.

[17] G. Holzinger, “OpenFOAM A little User-manual”, 2015. [Online]. Available: https://github.com/OSCCAR-PFM/OSCCAR-DOC-PUBLIC/blob/master/openFoamUserManual_PFM.pdf [Accessed 27.04.2016].

[18] OpenFoam, “OpenFOAM 2.3.0: Multiphase Modelling”, 2014. [Online] Available: http://openfoam.org/release/2-3-0/multiphase / [Accessed 19.05.2016].

[19] CFD Direct, “OpenFOAM User Guide: 4.4 Numerical schemes”, 2016. [Online]. Available: http://cfd.direct/openfoam/user-guide/fvSchemes/#x20-1070004.4 [Accessed: 22.05.2016].

[20] Y. Liu. “Two-Fluid Modeling of Gas-Solid and Gas-Liquid Flows: Solver Development and Application”. Ph.D, Technische Universitat Munchen, 2014.

[21] CFD Direct, “OpenFOAM User Guide: 4.5 Solution and algorithm control”, 2016. [Online] Available: http://cfd.direct/openfoam/user-guide/fvSolution/#x21-1170004.5 [Accessed: 22.05.2016].

[22] B. Hjertager, Lecture notes on OpenFOAM, 1st ed. Stavanger: University of Stavanger, 2009, pp. 1-141.

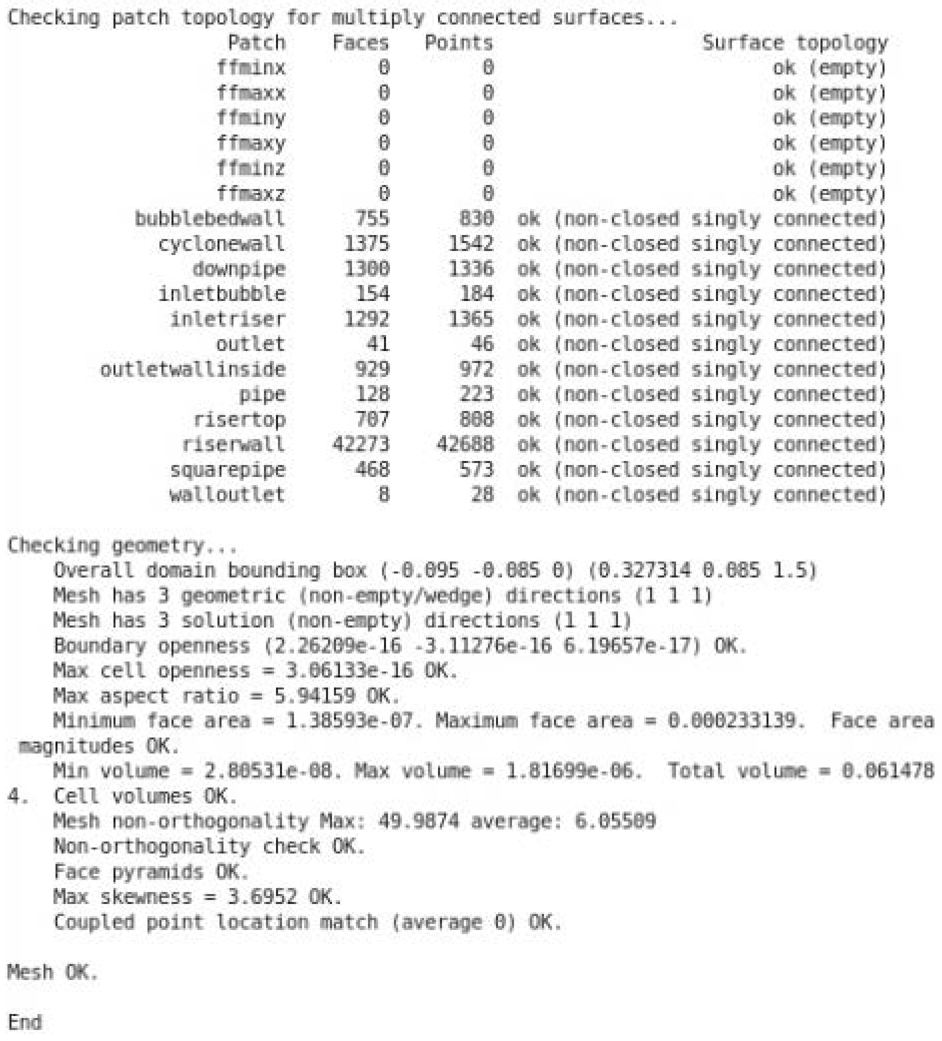
[23] D. Z. Zhang and W. B VanderHeyden, “High-resolution three-dimensional numerical simulation of a circulating fluidized bed”. *Powder technology* 116, pp. 133-141.

[24] Y. Liu and O Hinrichsen, “Numerical Simulation of Tube Erosion in a Bubbling Fluidized bed with a dense Tube Bundle”, *Chemical engineering technology*, Vol 36, No.4, pp.635-644, 2013

[25] V. Mathisen et al. “Prediction of gas/particle flow with an Eularian model including a realistic size distribution. *Powder Technology*. Vol 112, pp 34-45.

Appendix A

A.1 CheckMesh - Group A



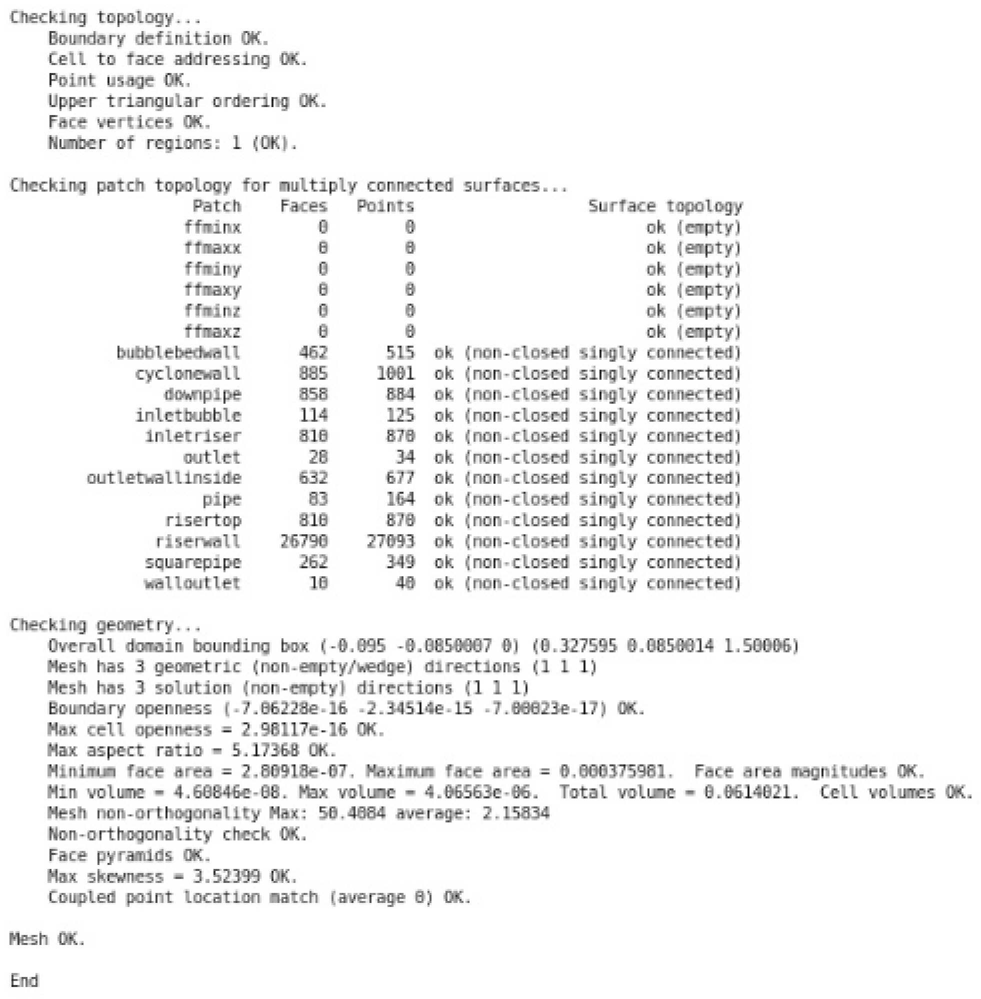
```
Checking patch topology for multiply connected surfaces...
                   Patch    Faces   Points                  Surface topology
                  ffminx        0        0                        ok (empty)
                  ffmaxx        0        0                        ok (empty)
                  ffminy        0        0                        ok (empty)
                  ffmaxy        0        0                        ok (empty)
                  ffminz        0        0                        ok (empty)
                  ffmaxz        0        0                        ok (empty)
           bubblebedwall      755      830  ok (non-closed singly connected)
             cyclonewall     1375     1542  ok (non-closed singly connected)
                downpipe     1300     1336  ok (non-closed singly connected)
             inletbubble      154      184  ok (non-closed singly connected)
              inletriser     1292     1365  ok (non-closed singly connected)
                  outlet       41       46  ok (non-closed singly connected)
        outletwallinside      929      972  ok (non-closed singly connected)
                    pipe      128      223  ok (non-closed singly connected)
                risertop      707      808  ok (non-closed singly connected)
               riserwall    42273    42688  ok (non-closed singly connected)
              squarepipe      468      573  ok (non-closed singly connected)
              walloutlet        8       28  ok (non-closed singly connected)

Checking geometry...
    Overall domain bounding box (-0.095 -0.085 0) (0.327314 0.085 1.5)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Boundary openness (2.26209e-16 -3.11276e-16 6.19657e-17) OK.
    Max cell openness = 3.06133e-16 OK.
    Max aspect ratio = 5.94159 OK.
    Minimum face area = 1.38593e-07. Maximum face area = 0.000233139.  Face area
 magnitudes OK.
    Min volume = 2.80531e-08. Max volume = 1.81699e-06.  Total volume = 0.061478
4.  Cell volumes OK.
    Mesh non-orthogonality Max: 49.9874 average: 6.05509
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 3.6952 OK.
    Coupled point location match (average 0) OK.

Mesh OK.

End
```

A.2 CheckMesh - Group B



```
Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
                   Patch    Faces   Points                  Surface topology
                  ffminx        0        0                        ok (empty)
                  ffmaxx        0        0                        ok (empty)
                  ffminy        0        0                        ok (empty)
                  ffmaxy        0        0                        ok (empty)
                  ffminz        0        0                        ok (empty)
                  ffmaxz        0        0                        ok (empty)
           bubblebedwall      462      515  ok (non-closed singly connected)
             cyclonewall      885     1001  ok (non-closed singly connected)
                downpipe      858      884  ok (non-closed singly connected)
             inletbubble      114      125  ok (non-closed singly connected)
              inletriser      810      870  ok (non-closed singly connected)
                  outlet       28       34  ok (non-closed singly connected)
        outletwallinside      632      677  ok (non-closed singly connected)
                    pipe       83      164  ok (non-closed singly connected)
                risertop      810      870  ok (non-closed singly connected)
               riserwall    26790    27093  ok (non-closed singly connected)
              squarepipe      262      349  ok (non-closed singly connected)
              walloutlet       10       40  ok (non-closed singly connected)

Checking geometry...
    Overall domain bounding box (-0.095 -0.0850007 0) (0.327595 0.0850014 1.50006)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Boundary openness (-7.06228e-16 -2.34514e-15 -7.00023e-17) OK.
    Max cell openness = 2.98117e-16 OK.
    Max aspect ratio = 5.17368 OK.
    Minimum face area = 2.80918e-07. Maximum face area = 0.000375981.  Face area magnitudes OK.
    Min volume = 4.60846e-08. Max volume = 4.06563e-06.  Total volume = 0.0614021.  Cell volumes OK.
    Mesh non-orthogonality Max: 50.4084 average: 2.15834
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 3.52399 OK.
    Coupled point location match (average 0) OK.

Mesh OK.

End
```

Appendix B

B.1 BlockMeshDict – Group A

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |                                                          |
|    o     o       | HELYX-OS                                                  |
|   o   O   o      | Version: v2.1.1                                           |
|    o     o       | Web:     http://www.engys.com                            |
|       o          |                                                          |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location system;
    object blockMeshDict;
}

    convertToMeters 1;
    vertices
    (

        ( -0.1049999988079071 -0.09500000089406967 -0.01)
        ( 0.3450000011920929 -0.09500000089406967 -0.01)
        ( 0.3450000011920929 0.10499999910593034 -0.01)
        ( -0.1049999988079071 0.10499999910593034 -0.01)
        ( -0.1049999988079071 -0.09500000089406967 1.51)
        ( 0.3450000011920929 -0.09500000089406967 1.51)
        ( 0.3450000011920929 0.10499999910593034 1.51)
        ( -0.1049999988079071 0.10499999910593034 1.51)
    );
    blocks
    (
    hex (0 1 2 3 4 5 6 7) (45 20 152) simpleGrading (1 1 1)
    );
    edges ( );
    patches
    (
     wall ffminx    (        (0 4 7 3)    )  wall ffmaxx    (        (1 2 6 5)    )  wall
ffminy    (        (0 1 5 4)    )  wall ffmaxy    (        (3 7 6 2)    )  wall ffminz    (
(0 3 2 1)    )  wall ffmaxz    (        (4 5 6 7)    )
    );
    mergePatchPairs ( );
    spacing 0.01;
```

B.2 BlockMeshDict – Group B

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |                                                          |
|    o     o       | HELYX-OS                                                  |
|   o   O   o      | Version: v2.1.1                                           |
|    o     o       | Web:     http://www.engys.com                            |
|       o          |                                                          |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location system;
    object blockMeshDict;
}

    convertToMeters 1;
    vertices
    (

        ( -0.1074999988079071 -0.09750000089406967 -0.0125)
        ( 0.3425000011920929 -0.09750000089406967 -0.0125)
        ( 0.3425000011920929 0.10249999910593034 -0.0125)
        ( -0.1074999988079071 0.10249999910593034 -0.0125)
        ( -0.1074999988079071 -0.09750000089406967 1.5125000000000002)
        ( 0.3425000011920929 -0.09750000089406967 1.5125000000000002)
        ( 0.3425000011920929 0.10249999910593034 1.5125000000000002)
        ( -0.1074999988079071 0.10249999910593034 1.5125000000000002)
    );
    blocks
    (
    hex (0 1 2 3 4 5 6 7) (36 16 122) simpleGrading (1 1 1)
    );
    edges ( );
    patches
    (
     wall ffminx    (        (0 4 7 3)    )  wall ffmaxx    (        (1 2 6 5)    )  wall
ffminy    (        (0 1 5 4)    )  wall ffmaxy    (        (3 7 6 2)    )  wall ffminz    (
(0 3 2 1)    )  wall ffmaxz    (        (4 5 6 7)    )
    );
    mergePatchPairs ( );
    spacing 0.0125;
```

B.3 SnappyHexMeshDict – Group A

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |                                                          |
|    o     o       | HELYX-OS                                                  |
|   o   O   o      | Version: v2.1.1                                           |
|    o     o       | Web:     http://www.engys.com                            |
|       o          |                                                          |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location system;
    object snappyHexMeshDict;
}

    castellatedMesh true;
    snap true;
    addLayers true;
```

```
geometry
{
    bubblebedwall.stl
    {
        type triSurfaceMesh;
        name bubblebedwall;
    }

    cyclonewall.stl
    {
        type triSurfaceMesh;
        name cyclonewall;
    }

    downpipe.stl
    {
        type triSurfaceMesh;
        name downpipe;
    }

    inletbubble.stl
    {
        type triSurfaceMesh;
        name inletbubble;
    }

    inletriser.stl
    {
        type triSurfaceMesh;
        name inletriser;
    }

    outlet.stl
    {
        type triSurfaceMesh;
        name outlet;
    }

    outletwallinside.stl
    {
        type triSurfaceMesh;
        name outletwallinside;
    }

    pipe.stl
    {
        type triSurfaceMesh;
        name pipe;
    }

    risertop.stl
    {
        type triSurfaceMesh;
        name risertop;
    }

    riserwall.stl
    {
        type triSurfaceMesh;
        name riserwall;
    }

    squarepipe.stl
    {
        type triSurfaceMesh;
        name squarepipe;
    }

    walloutlet.stl
    {
        type triSurfaceMesh;
        name walloutlet;
    }

}

castellatedMeshControls
{
    features
    (
    );
    refinementSurfaces
    {
        bubblebedwall
        {
            level ( 0 0);
        }

        cyclonewall
        {
            level ( 0 0);
        }

        downpipe
        {
            level ( 0 0);
        }

        inletbubble
        {
            level ( 0 0);
        }

        inletriser
        {
            level ( 1 3 );
        }

        outlet
        {
            level ( 0 0);
        }

        outletwallinside
        {
            level ( 0 0);
        }

        pipe
        {
            level ( 0 0);
        }

        risertop
        {
            level ( 0 0);
        }

        riserwall
        {
            level ( 1 3 );
        }

        squarepipe
        {
            level ( 0 0);
        }

        walloutlet
        {
            level ( 0 0);
        }
    }

    refinementRegions
    {
    }

    locationInMesh ( 0.0 0.0 0.75 );
    maxLocalCells 100000;
    maxGlobalCells 2000000;
    minRefinementCells 0;
    nCellsBetweenLevels 1;
    resolveFeatureAngle 30;
    allowFreeStandingZoneFaces true;
    planarAngle 30;
    maxLoadUnbalance 0.10;
}

snapControls
{
    nSolveIter 30;
    nSmoothPatch 3;
    tolerance 2.0;
    nRelaxIter 5;
    nFeatureSnapIter 10;
    implicitFeatureSnap true;
    explicitFeatureSnap false;
    multiRegionFeatureSnap false;
}

addLayersControls
{
    layers
    {
        inletriser
        {
            nSurfaceLayers 3;
            finalLayerThickness 0.005;
            minThickness 0.001;
        }

        riserwall
        {
            nSurfaceLayers 3;
            finalLayerThickness 0.005;
            minThickness 0.001;
        }

    }

    relativeSizes true;
    expansionRatio 1.0;
    finalLayerThickness 0.3;
    minThickness 0.25;
    nGrow 0;
    featureAngle 130;
    slipFeatureAngle 30;
    nRelaxIter 5;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;
    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 0.3;
    minMedianAxisAngle 90;
    nBufferCellsNoExtrude 0;
    nLayerIter 50;
    nRelaxedIter 20;
    writeVTK false;
    noErrors false;
    layerRecovery 1;
    growZoneLayers false;
```

```
        projectGrownUp 0.0;
    }

    meshQualityControls
    {
        maxNonOrtho 65;
        maxBoundarySkewness 20;
        maxInternalSkewness 4;
        maxConcave 80;
        minFlatness 0.5;
        minVol 1.00E-13;
        minTetQuality 1e-15;
        minArea -1;
        minTwist 0.02;
        minDeterminant 0.001;
        minFaceWeight 0.05;
        minVolRatio 0.01;
        minTriangleTwist -1;
        nSmoothScale 4;
        errorReduction 0.75;
    }

    debug 0;
    mergeTolerance 1E-6;
    autoBlockMesh true;
```

B.4 SnappyHexMeshDict –Group B

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |                                                          |
|    o     o       | HELYX-OS                                                  |
|   o   O   o      | Version: v2.1.1                                           |
|    o     o       | Web:     http://www.engys.com                            |
|       o          |                                                          |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location system;
    object snappyHexMeshDict;
}

    castellatedMesh true;
    snap true;
    addLayers true;
    geometry
    {
        bubblebedwall.stl
        {
            type triSurfaceMesh;
            name bubblebedwall;
        }

        cyclonewall.stl
        {
            type triSurfaceMesh;
            name cyclonewall;
        }

        downpipe.stl
        {
            type triSurfaceMesh;
            name downpipe;
        }

        inletbubble.stl
        {
            type triSurfaceMesh;
            name inletbubble;
        }

        inletriser.stl
        {
            type triSurfaceMesh;
            name inletriser;
        }

        outlet.stl
        {
            type triSurfaceMesh;
            name outlet;
        }

        outletwallinside.stl
        {
            type triSurfaceMesh;
            name outletwallinside;
        }

        pipe.stl
        {
            type triSurfaceMesh;
            name pipe;
        }

        risertop.stl
        {
            type triSurfaceMesh;
            name risertop;
        }

        riserwall.stl
        {
            type triSurfaceMesh;
            name riserwall;
        }

        squarepipe.stl
        {
            type triSurfaceMesh;
            name squarepipe;
        }

        walloutlet.stl
        {
            type triSurfaceMesh;
            name walloutlet;
        }

        box
        {
            type searchableBox;
            min ( -0.095 -0.085 0.0 );
            max ( 0.095 0.085 1.5 );
        }

    }

    castellatedMeshControls
    {
        features
        (
        );
        refinementSurfaces
        {
            bubblebedwall
            {
                level ( 0 0);
            }
            cyclonewall
            {
                level ( 0 0);
            }

            downpipe
            {
                level ( 0 0);
            }

            inletbubble
            {
                level ( 0 0);
            }

            inletriser
            {
                level ( 0 0);
            }

            outlet
            {
                level ( 0 0);
            }

            outletwallinside
            {
                level ( 0 0);
            }

            pipe
            {
                level ( 0 0);
            }

            risertop
            {
                level ( 0 0);
            }

            riserwall
            {
                level ( 0 0);
            }

            squarepipe
            {
                level ( 0 0);
            }

            walloutlet
            {
                level ( 0 0);
            }

        }

        refinementRegions
        {
            box
            {
                mode inside;
                levels (( 1E5 1 ));
            }

        }

        locationInMesh ( 0.0 0.0 0.5 );
        maxLocalCells 100000;
        maxGlobalCells 2000000;
```

```
    minRefinementCells 0;
    nCellsBetweenLevels 1;
    resolveFeatureAngle 30;
    allowFreeStandingZoneFaces true;
    planarAngle 30;
    maxLoadUnbalance 0.10;
}

snapControls
{
    nSolveIter 30;
    nSmoothPatch 3;
    tolerance 2.0;
    nRelaxIter 5;
    nFeatureSnapIter 10;
    implicitFeatureSnap true;
    explicitFeatureSnap false;
    multiRegionFeatureSnap false;
}

addLayersControls
{
    layers
    {
    }

    relativeSizes true;
    expansionRatio 1.0;
    finalLayerThickness 0.3;
    minThickness 0.25;
    nGrow 0;
    featureAngle 130;
    slipFeatureAngle 30;
    nRelaxIter 5;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;
    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 0.3;
    minMedianAxisAngle 90;
    nBufferCellsNoExtrude 0;
    nLayerIter 50;
    nRelaxedIter 20;
}

meshQualityControls
{
    maxNonOrtho 65;
    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
    minFlatness 0.5;
    minVol 1.00E-13;
    minTetQuality 1e-15;
    minArea -1;
    minTwist 0.02;
    minDeterminant 0.001;
    minFaceWeight 0.05;
    minVolRatio 0.01;
    minTriangleTwist -1;
    nSmoothScale 4;
    errorReduction 0.75;
}

debug 0;
mergeTolerance 1E-6;
autoBlockMesh true;
```

B.5 g

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           (0 0 -9.81);


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

B.6 PhaseProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      phaseProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phases (particles air);

particles
{
    residualAlpha   1e-6;

    diameterModel constant;
    constantCoeffs
    {
        d               0.70e-4;
    }

    alphaMax        0.64;
    residualAlpha   1e-6;
}

air
{
    residualAlpha   0;

    diameterModel constant;
    constantCoeffs
    {
        d               1;
    }

    residualAlpha   0;
}

blending
{
    default
    {
        type            none;
        continuousPhase air;
    }
}

sigma
(
    (particles and air) 0
);

aspectRatio
(
);

drag
(
    (particles in air)
    {
        type            GidaspowErgunWenYu;
        residualRe      1e-3;
        swarmCorrection
        {
            type        none;
        }
    }
);

virtualMass
(
);

heatTransfer
(
    (particles in air)
    {
        type            RanzMarshall;
        residualAlpha   1e-3;
    }
);

lift
(
);

wallLubrication
(
);

turbulentDispersion
(
);

// Minimum allowable pressure
pMin            10000;


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

B.7 TurbulenceProperties.particles

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties.particles;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  RAS;

RAS
{
    RASModel kineticTheory;

    turbulence      on;
    printCoeffs     on;

    kineticTheoryCoeffs
    {
        equilibrium             off;

        e                       0.95;
        alphaMax                0.64;
        alphaMinFriction        0.5;
        residualAlpha           1e-4;

        viscosityModel          Gidaspow;
        conductivityModel       Gidaspow;
        granularPressureModel   Lun;
        frictionalStressModel   JohnsonJackson;
        radialModel             SinclairJackson;

        JohnsonJacksonCoeffs
        {
            Fr                      0.05;
            eta                     2;
            p                       5;
            phi                     28.5;
            alphaDeltaMin           0.05;
        }
    }

    phasePressureCoeffs
    {
        preAlphaExp     500;
        expMax          1000;
        alphaMax        0.64;
        g0              1000;

    }
}


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

B.8 ThermoPhysicalProperties.particles

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties.particles;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState rhoConst;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        nMoles      1;
        molWeight   182.256;
    }
    equationOfState
    {
        rho         7800;
    }
    thermodynamics
    {
        Cp          435;
        Hf          0;
    }
    transport
    {
        mu          0;
        Pr          1;
    }
}


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

B.9 TurbulenceProperties.air

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties.air;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  LES;

RAS
{
    RASModel kEpsilon;

    turbulence      off;
    printCoeffs     off;
}

LES
{
    LESModel Smagorinsky;

    turbulence      on;
    printCoeffs     on;

    delta cubeRootVol;

    cubeRootVolCoeffs
    {
        deltaCoeff 1;
    }

}


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

B.10 ThermoPhysicalProperties.air

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties.air;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        nMoles      1;
        molWeight   28.9;
    }
    equationOfState
    {
       rho    1.20;
       }
    thermodynamics
    {
        Cp          1007;
        Hf          0;
    }
    transport
    {
        mu          1.84e-05;
        Pr          0.7;
    }
}


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

Appendix C

C.1 fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.*"
    {
        nAlphaCorr      1;
        nAlphaSubCycles 2;

        smoothLimiter   0.1;

        implicitPhasePressure yes;
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-9;
        relTol          0;
        minIter         1;
    }

    p_rgh
    {
        solver          GAMG;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     1;
        tolerance       1e-8;
        relTol          0;
    }

    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    "U.*"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-5;
        relTol          0;
        minIter         1;
    }

    "(h|e).*"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-6;
        relTol          0;
        minIter         1;
    }

    "Theta.*"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-6;
        relTol          0;
        minIter         1;
    }

    "(k|epsilon).*"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-5;
        relTol          0;
        minIter         1;
    }
}

PIMPLE
{
    nOuterCorrectors 3;
    nCorrectors      1;
    nNonOrthogonalCorrectors 0;
    pRefCell 0;
    pRefValue 0;
}

relaxationFactors
{
    equations
    {
        ".*"            1;
    }
}


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

C.2 fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default     Euler;
}

gradSchemes
{
    default     Gauss linear;
}

divSchemes
{
    default                         none;

    "div\(phi,alpha.*\)"            Gauss vanLeer;
    "div\(phir,alpha.*\)"           Gauss vanLeer;

    "div\(alphaRhoPhi.*,U.*\)"      Gauss limitedLinearV 1;
    "div\(phi.*,U.*\)"              Gauss limitedLinearV 1;

    "div\(alphaRhoPhi.*,(h|e).*\)"  Gauss limitedLinear 1;
    "div\(alphaRhoPhi.*,K.*\)"      Gauss limitedLinear 1;
    "div\(alphaPhi.*,p\)"           Gauss limitedLinear 1;

    div(alphaRhoPhi.particles,Theta.particles) Gauss limitedLinear 1;

    "div\(alphaRhoPhi.*,(k|epsilon).*\)"  Gauss limitedLinear 1;

    div((((alpha.air*thermo:rho.air)*nuEff.air)*dev2(T(grad(U.air))))) Gauss linear;


div((((thermo:rho.particles*nut.particles)*dev2(T(grad(U.particles))))+(((thermo:rho.particles
*lambda.particles)*div(phi.particles))*I)))  Gauss linear;
}

laplacianSchemes
{
    default     Gauss linear uncorrected;
    bounded     Gauss linear uncorrected;
}

interpolationSchemes
{
    default     linear;
}

snGradSchemes
{
    default     uncorrected;
    bounded     uncorrected;
}

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

C.3 ControlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     twoPhaseEulerFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         20;

deltaT          0.0002;

writeControl    runTime;

writeInterval   0.5;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable on;

adjustTimeStep  yes;

maxCo           0.9;

maxDeltaT       1e-2;

functions
{
    fieldAverage1
    {
        type            fieldAverage;
        functionObjectLibs ( "libfieldFunctionObjects.so" );
```

```
enabled        true;
        outputControl   outputTime;
       timeStart             5;
       timeEnd               20;
        fields
        (
            U.particles
            {
                 mean        on;
                 prime2Mean  on;
                 base        time;
            }

            U.air
            {
                 mean        on;
                 prime2Mean  on;
                 base        time;
            }

            alpha.particles
            {
                 mean        on;
                 prime2Mean  on;
                 base        time;
            }

            p
            {
                 mean        on;
                 prime2Mean  on;
                 base        time;
            }

       Theta.particles
       {
               mean    on;
               prime2Mean     on;
               base           time;
       }
        );
    }

    setTest
    {
       type sets;
       functionObjectLibs ( "libfieldFunctionObjects.so" );

       setFormat raw;
       interpolationScheme cellPointFace;

       fields
       (

       p

       );

       sets
       (
               linez
               {
               type uniform;
               axis z;
               start (0 0 0);
               end (0 0 1.5);
               nPoints 100;
               }
               );

               outputControl   outputTime;
```

```
           }
}

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

C.4 DecomposParDict

```
/*--------------------------------*- C++ -*-------------------------
---------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  3.0.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*------------------------------------------------------------------
---------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    note        "mesh decomposition control dictionary";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * //

numberOfSubdomains  10;

method          scotch;

simpleCoeffs
{
    n           (1 1 20);
    delta       0.001;
    order       xyz;
}

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

Appendix D

D.1 Content of Enclosed memory stick

The enclosed memory stick includes a PDF version of the thesis, as well as all the case files. All cases include complete 0, constant and system folders, and a short animation of the bottom bed for case 2.70.