KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Darcy and non-Darcy fluid flow simulation through well-structured porous medium

Adriaan Hugo E. De Bolle

Thesis submitted for the degree of
Master of Science in
Chemical Engineering, option
Chemical and biochemical process
engineering

Thesis supervisor:
Prof. dr. ir. Simon Kuhn

Academic year 2015 – 2016

Master of Science in Chemical Engineering

KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Darcy and non-Darcy fluid flow simulation through well-structured porous medium

Adriaan Hugo E. De Bolle

Thesis submitted for the degree of
Master of Science in
Chemical Engineering, option
Chemical and biochemical process
engineering

Thesis supervisor:
Prof. dr. ir. S. Kuhn

Assessors:
Prof. dr. ir. G. Stefanidis
Ir. M. Van den Eynde

Mentor:
Ir. M. Mottaghi

Academic year 2015 – 2016

Foreword

Firstly, I would like to thank my coordinators who helped me to learn, investigate, discuss problems and solve some bottlenecks and always gave advise when needed to make this dissertation a great success. Prof. S. Kuhn always wanted to be involved and helped with high expertise. Ir. M. Mottaghi gave me daily supervision and together we followed a journey towards a better understanding. Not every solution is clear and straightforward, but M. Mottaghi guided me often to an alternative way of thinking to tackle problems.

Secondly, I need to give my sincere and honest gratitude to people who helped me "survive" five years of though engineering studies. For this, I look mainly to Jirka Delaere who pushed me to achieve every time an acceptable level or more for each course. Discussions led us to a better understanding. Enduring my studies to become Chemical Engineer was eased by Peter Van Puyvelde, Program Director. He gave me extraordinarily opportunities (internationally) to broaden my view surpassing the engineering domain.

Lastly, I thank my parents, brother, family and friends to shape me as a loving person that looks further than engineering. Life is made greater from experiences work related, leisure and pleasure.

Adriaan Hugo Elise De Bolle

Abstract

Porous structures can be found anywhere. They are widely used in the industry to change process conditions e.g. catalysis and absorption. Natural structures like soil and a system of leaves in a forest are also types of porous structures. Their irregular shape - that alters flow paths of a fluid drastically - makes it very challenging to describe flow passing through a porous medium.

Obviously one can characterise each structure by experiments, but it is more interesting to obtain a mathematical description to predict flow behaviour. In the past, several scientists (Kozeny-Karman, Darcy, Forchheimer, …) proposed equations to describe this behaviour based on porosity and geometrical structure or based on permeability parameters. This dissertation looks to several aspects of this diversity of options and simulates different cases for an explicit porous medium which is a packed bed of spheres and compares results with actual experiments.

A packed bed of spherical particles inside a tube is in reality randomly distributed and touch each other. Building such geometries is quite challenging computationally, because touching spheres gives very small length scales, which gives tolerance issues in traditional CAD (Computer Aided Design) tools. Firstly, a packed bed is made with the framework *Yade*, a DEM (Discrete Element Method) package where randomization of spheres is achieved by dynamical time-based simulations. Randomized spheres are compressed to wanted porosity by moving walls. Because this is time based it seemed tricky to get a completely correct compression distance. This altered the porosity of the final solutions. Alternatively, a script is built based on random positioning in *Python*, but this method is very time-consuming for highly dense packings. Therefore, simulations are made based on centroid co-ordinates found by randomization by *Yade*.

Defining the fluid domain is done by *Salome.* Because of the limitation that touching spheres are difficult to simulate gaps between and overlaps of spheres are induced to limit problems constructing the geometry. Nevertheless, after automated geometry building problems needed to be addressed manually which made geometry construction very time-consuming. The meshes used for the simulations are unstructured and built with the *Netgen 3D-2D-1D* algorithm. The use of a fully unstructured mesh for the whole tube is validated and has shown it gives reasonable results.

Within *OpenFOAM*, a CFD (Computational Fluid Dynamics) software package, results are obtained for different flow regimes with an explicit porous structure. Final simulation results showed analogy with experimental values. Within Darcy's regime the ratio of pressure and velocity should be constant. This is not found in the simulation results. This could be due to a slight difference in porosity or the fact that only a part of the whole porous zone is simulated which gives different importance to the entrance length of the porous zone.

For simulations in *OpenFOAM* it can be interesting to use turbulency models. Turbulency can become important when velocity in pores is high due limited paths and reducing pore size. Secondly, flow gets pulsed out from the porous zone and creates a turbulency zone in the outlet section. In a 2-dimensional case this effect is shown. So this can be clearly seen with explicit methods when the explicit porous structure is simulated. Darcy's law averages flow and cannot predict where this expulsing happens. For Comparison between simulations and experimental values the setup how the pressure is measured is important. For the simulation results pressure drop over the porous zone gave good results, but when experiments would include partly the outlet zone, pressure measurement could be effected by the turbulency zone.

Final results had a slight larger porosity. When using the apparent permeability interpolation permeability and Forchheimer coefficient could be calculated and showed analogy with reality when taking into account the difference in porosity. Concluded can be that explicit simulations can help to characterise thresholds and permeability parameters.

Table of Contents

List of Figures and Tables

List of Figures

List of Tables

List of abbreviations and symbols

Parameters are listed in order of appearance.

Fluid property parameters

Shortcut	Full name
u	Velocity (locally), m/s
u	Velocity vector, $[m/s, m/s, m/s]^T$
g	Gravitational acceleration vector, m/s^2
q	Volumetric flow rate, m^3/s
ρ or ρ_f	Mass density (of fluid), kg/m^3
μ	Dynamic viscosity, Pa.s
K [1]	Permeability scalar, m^2
ν	Cinematic viscosity, m^2/s
K [1]	Permeability 3 x 3 tensor, m^2
F [2]	Forchheimer coefficient, -
β [2]	Non-Darcy flow coefficient, m^{-1}
Re [3]	Reynolds number, -
Re_p	Reynolds based on particle size, -
Re_K	Reynolds based on permeability, -
Re^*	Reynolds based on Ergun, -
K_{app}	Apparent permeability, m
Δp/L	Normalised pressure drop, Pa/m
Δp/Lu	Reduced pressure drop, $Pa.s/m^2$
$x_{fd,H}$	Distance to fully developed, m
U [4]	Superficial velocity, m/s
u_{max}	Maximal velocity, m/s
h_f	Friction loss, m
k	Turbulent energy, m^2/s^2
ϵ	Turbulent dissipation rate, m^2/s^3
ω	Specific turbulent dissipation rate, s^{-1}
d	Darcy resistance vector, m^{-2}
f	Forchheimer resistance vector, m^{-1}
τ [5]	Stress, Pa
τ_w	Wall stress, Pa
τ^R	Reynolds-averaged stress, Pa
u'	Velocity relative to average velocity, m/s

[1] Permeability depends on orientation and flow regime. A tensor gives difference by orientation. K_D is valid within Darcy's regime and K_F within non-Darcy or Forchheimer regime.

[2] Forchheimer is important when inertia effects play a role (increased Re) and corrects Darcy's low.

[3] Reynolds number (Re) is the ratio of inertial and viscous forces.

[4] Velocity is the superficial velocity or volumetric flow divided by cross section. Within a porous media this velocity is not the absolute velocity of a particle.

[5] Stress τ, a fluid parameter, may not be confused with tortuosity τ, a geometrical parameter

μ_T	Turbulent viscosity, Pa.s
u_τ	Friction velocity, m/s
C_f [6]	Skin friction coefficient, -
Co	Courant number, -
δt	Time interval, s

Dimensional parameters (geometry)

Shortcut	Full name
A	Surface, m^2
D	Tube diameter, m
ε	Porosity or void space, -
L_i	Length of inlet zone, m
L_p	Length of porous zone, m
L_o	Length of outlet zone, m
R_c	Radius of cylinder/tube, m
R_s / D_s	Radius / diameter of (spherical) particle, m
τ [7]	Tortuosity, -
x_{SV}	Sauter mean diameter
y^+	yPlus or dimensionless wall distance, -

[6] Not to confuse with Darcy friction coefficient, f_D, and Fanning friction coefficient, f.

[7] Tortuosity is the ratio between the actual path following a flow particle and the ideal (straight) pathway.

Chapter I: Introduction

Computational Fluid Dynamics (CFD) is a broad area. This dissertation looks to simulations of flow through porous media. Porous structures are widely used and even in unexpected applications. This section gives the goal of this dissertation, technical details how to build up the simulations made, why the need for computational approach and overview of applications in different domains.

Goal

The main objective of this thesis is to obtain conclusions whether Darcy's law can be used in simulations with a simpler "black box" method instead of using explicit porous topology and Navier-Stokes equations. Firstly, understanding of simulations, spherical packings, analytical control, computational models, theoretical background of Darcy's law and experimental data is necessary to acquire the main objective.

Technical details

Geometries are built in *Salome* V7_6_0, open source CAD (computer-aided design) software. Discrete Element Method (DEM) within framework Yade is used to create packings in geometry modules prior to simulation within *OpenFOAM*. Therefore, it was not necessary to couple CFD and DEM to generate solutions. For the simulations, within OpenFOAM-2.4.0, the finite volume method is used. This method is more understandable, because physical conservation equations (balance equations) are valid for each control volume. *OpenFOAM* is an open source software package used in Ubuntu – Linux (operational system).

Highly Recommended

All files to build up the simulation in all software mentioned above are located in a github repository: https://github.com/AdriaanDB/Thesis-2016-KULeuven

Why simulation?

Simulations want to achieve an accurate, precise representation of reality. It is interesting to build a simulation for creating new applications or scaling up existing applications. Real process variables have certain disturbances depending on the system. Therefore, it is also important that simulations give a stable solution that is not sensitive towards small changes. Errors mainly slip in simulations because of incorrect set of modelling equations, discretization schemes and iterative procedures that don't converge linked to the tolerance level used.

Utilisation of simulations is limited by computational power. The field of computational simulation started in the '70s. From the '90s personal computers were powerful enough to simulate simple models. Still today, there are limitations. It takes several days to complete a simulation of microalgae using computers at a university. Maybe not expected, but simulating microalgae and the biological model of photosynthesis needs also knowledge of the streamlines (CFD) in a pond where the algae grow. (Chachuat, 2015) Simulation of the human brain during 1s

with the fourth most powerful supercomputer in the world needs 40min of computation time. (Sparkes, 2014)
By experience, making the fluid domain around spherical packing takes at least a day on a personal computer. Meshing of this structure takes several hours and solving as well. But even more time-consuming is manual corrections and iterative procedures, but this will be more clear throughout this dissertation.
Gradually with further increasing computational power the use of simulations will intensify without doubt.

Intermezzo
General procedure to make simulations can be read in Appendix A.

Why porous media?

Porous media are about porosity, permeability and Darcy's law (and corrections). Study and knowledge of porous media is important in many different industry branches. Applications are found in oil recovery, nuclear safety, chemical engineering and others. (Soulaine, 2015) Following some examples:

- **Catalysts**

Catalysts are widely used to improve reaction kinetics mostly in multitubular reactors. Generally, catalysis is a factor in 90% of all chemical processes, as such, a vital part in the chemical industry. (National Academy of Sciences, 2000) Catalyst is divided generally into homogeneous and heterogeneous catalysts. For the latter, solid particles are added to the reaction medium. Specific example is the reaction of ethylene to ethylene oxide where selectivity for this reaction is favoured against side reactions, because of the use of a silver catalyst with a porous, cylindrical structure. (US Brevet n° US 4368144 A, 1980) So catalyst particles can have a porous structure and as well form all together a porous medium inside a tube.

- **Mixing**

A random packing has an amorphous structure and gives irregular flow pathways, which increase the tortuosity τ. Therefore, better mixing of different phases is (temporarily) possible, which can increase overall interphase surface and faster transition to chemical products. Mixing reduces also the (bio)film thickness around solid particles, which give a smaller resistance to mass transfer from bulk to solid surface. (Taherzadeh, Picioreanu, & Horn, 2012)

- **Membrane Technology**

Membranes are in fact thin cuts of a porous structure. Therefore, also for pressure driven membranes Darcy's law and specifically for spherical particles and laminar flow Kozeny-Carman's law can be used. Membranes are used for separating a fluid from its solutes. The solutes are retained and the fluid permeates through the membrane. Depending on pore size one distinguishes micro-, ultra- and nanofiltration. For very narrow pores (<1nm) when filtering salt solution, osmotic pressure has to be taken into account (reversed osmosis). (Van Der Bruggen, Vandecasteele, Van Gestel, Doyen, & Leysen, 2003)

- **Filtration**

-Surface filtration: Particles are retained at the surface of a filter media. These particles form a *cake layer* that increases resistance to flow and increases the pressure drop. Correct estimation of this build-up and the pressure drop over this irregular packed bed gives better scheduling for removal of the cake layer. (Harvard Corporation, 2016)
-Depth filtration: Particles are retained within a packed bed e.g. sand filtration or trickling filters to purify water. Here large particles can also build a cake layer or biofilm that retains unwanted compounds in water at the particle surface. Smaller particles are blocked depending on the different pathways. (Harvard Corporation, 2016)

- **Packed beds**

Packed beds are widely used in the (bio)chemical industry. For biological systems it is very important to measure temperature gradients and pressure gradients. Temperature is important for optimal growth of microorganisms. A traditional, fixed packed bed can be found in solid state fermentation. For these applications the porosity and humidification are additionally important parameters. Air with water droplets and an inlet temperature flows through the bed of solid substrates and can cause movement of the particles and even fluidization of the bed. This can cause porosity changes through the bed. (Pandey, 2003)

- **Oil & Gas**

Fracking of Shale gas is also an example of porous media. Shale gas is found inside porous reservoirs and important is the viscosity of the gas to flow out of these pores. Therefore, viscosity changes are imposed by adding chemicals and breaking up of pores by high-pressured liquid (fracking) to allow more gas to flow out. As such extraction of shale gas requires investigation of the porous characteristics of the well. (King, 2005-2016)

It is clear that porous media and its extents are widely used in different industry branches (physical and chemical). For the further progress of this thesis the bed will be fixed without porosity changes, without chemical reactions and without heat transfer.

Chapter II: Literature

Before addressing simulations, it is always recommended to get a theoretical background. This section gives therefore a literature overview concerning various aspects regarding porous media: difference in modelling, representative equations and experiments done in the past by third parties.

Flow through porous media

Usually flow is rather slow in porous media. This is called creeping flow. In this case the assumption of incompressible flow is accurate and flow is independent of flow rate and the type of the fluid. These conditions are governed by the Stokes equations (Re < 1), a subset of the Navier-Stokers equations. (Soulaine, 2015) Care should be addressed when putting thresholds such as Re<1, because this is not always correctly bounding different flow regimes. As well the term "creeping" can be misleading, because velocity can rise drastically in small pores. Direct modelling uses the Stokes equations and boundary conditions. This will lead to option A in the geometry section. Continuum or Darcy modelling uses the Darcy equations based on average velocity and average pressure within Representative Elementary Volumes (REV) without explicit pore topology. This will lead to option B. Difference between A and B can be seen in Figure 1. Figure 2 gives a wide range of porous media at different length scales. For example, to simulate a (burning) forest, it is very computational power consuming to represent all branches and leaves. When considering the forest as a porous continuum medium Darcy's law can be used, which decreases needed power. (Soulaine, 2015)

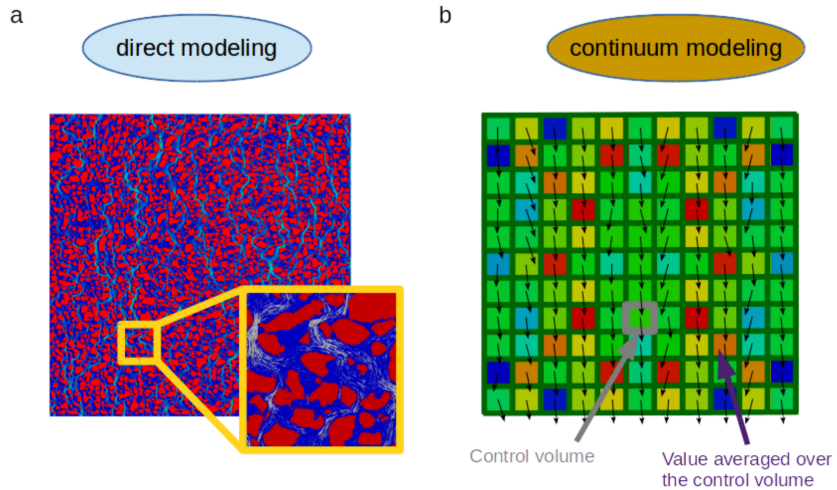


Figure 1 : Difference between simulation with and without explicit pore topology (Soulaine, 2015)

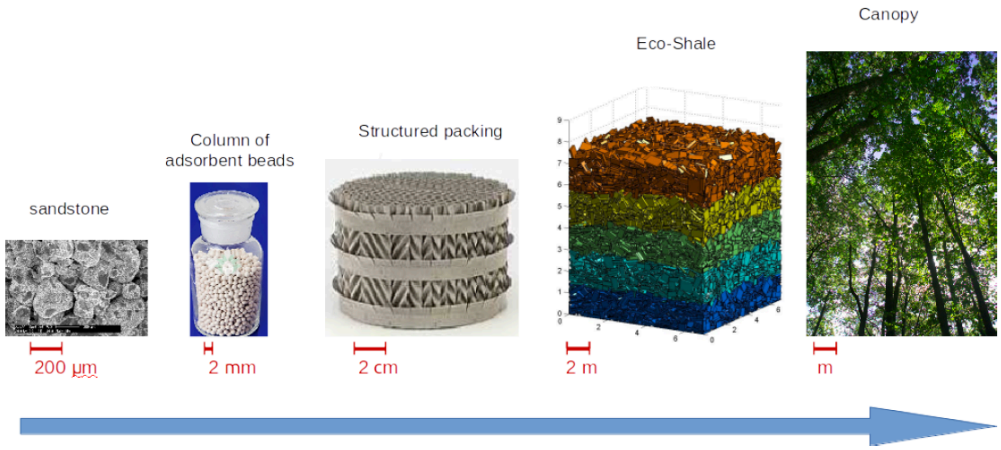


Figure 2 : Examples of porous media for a wide range of length scales (Soulaine, 2015)

Darcy's law

Vector form of Darcy's law (and conservation of mass), where β stands for liquid phase filling the void space:

$$\begin{cases} \nabla < \boldsymbol{u}_\beta > = 0 \\ < \boldsymbol{u}_\beta > = -\dfrac{\boldsymbol{K}}{\mu_\beta}(\nabla < p_\beta >^\beta - \rho_\beta \boldsymbol{g}) \end{cases}$$

Analytical form of Darcy's equation for one dimension, where U is the superficial velocity or flow volumetric flux:

$$U = \frac{q}{A} = \frac{K_D}{\mu}\frac{dp}{dx}$$
$$\frac{dp}{dx} = \frac{\mu}{K_D}U$$

Forchheimer proposed a quadratic correction term (1D) seen underneath. When viscous effects are dominant this correction term is zero. When inertial effects start to play a role, at higher velocity, the term becomes non-zero. The effect is shown in Figure 5.

$$\frac{dp}{dx} = \frac{\mu}{K_F}U + \beta\rho U^2$$
$$\beta = \frac{F}{\sqrt{K_F}}$$

Both parameters K and F strongly depend on the internal structure of the porous medium. Permeability stands for openness of the porous medium to fluid flow. (Dukhan, Bağcı, & Özdemir, 2014) This includes not only porosity or void space, but also the amount of possible paths and narrowness. Important to notice is that these parameters are not truly constant since they depend on the flow regime

(Darcy, transitional, Forchheimer or turbulent). (Dukhan, Bağcı, & Özdemir, 2014) Therefore, one needs to distinguish different permeability parameters, K_D and K_F, in different flow regimes.

Reynolds as onset for non-Darcy

Traditionally the Reynolds number (Re) characterises the laminar, transitional and turbulent flow regime with each one a different set of equations. For Hagen-Poiseuille tube flow (without any obstacles and tortuosity τ equals one) Re is given by:

$$Re = \frac{\rho D U}{\mu}$$

For porous media several Re are found in literature depending on the context:

$$Re_p = \frac{\rho D_p U}{\mu} \quad (1)$$

$$Re_K = \frac{\rho \sqrt{K} U}{\mu} \quad (2)$$

$$Re^* = \frac{R_p U \rho_f}{\mu(1-\varepsilon)} \quad (3)$$

Equation 1 is used when the geometry of the porous medium is known, because particle diameter or a characteristic length determines the value. Equation 2 uses permeability, found by experiments, so the explicit geometry details don't have to be known. Last equation (3) is used for turbulent regimes (Ergun) and correlations of friction factor. To define Darcy and non-Darcy flow and the transition between these regimes mostly equation one and two are used as threshold between these regimes. This is important, because different equations are valid within these regimes and when trespassing non-Darcy threshold, the Forchheimer correction term, with different permeability coefficient, is needed.

Flow characteristics through an explicit packed bed

This section is based on explicit knowledge of the porous structure and gives a summary of important equations. More detailed analysis can be found in Transport Phenomena or Powder Technology. Recommended book: *Introduction to Particle Technology – Second Edition: Fluid Flow Through a Packed Bed of Particles* (Rhodes, 2008)

Blake-Kozeny relationship, valid for laminar flow, spherical packing and ε < 0.50:

$$U = \frac{q}{A} = \frac{\Delta p\, \varepsilon^3}{72\gamma^2 \mu L_p (1-\varepsilon)^2} {D_s}^2$$

Generally, for all Re-regimes the pressure drop over the bed is given by Ergun, where the first term is for laminar and both for turbulent:

$$\frac{-\Delta p}{L_p} = 150 \frac{\mu U}{{x_{SV}}^2} \frac{(1-\varepsilon)^2}{\varepsilon^3} + 1.75 \frac{\rho_f U^2}{x_{SV}} \frac{(1-\varepsilon)}{\varepsilon^3}$$

Defining an alternative Reynolds number based on the Ergun equation for laminar and turbulent flow:

$$Re^* = \frac{R_s U \rho_f}{\mu(1-\varepsilon)}$$

one can get a correlation for the friction factor to calculate pressure drop:

$$f^* = \frac{150}{Re^*} + 1{,}75$$

and calculate the pressure drop:

$$f^* = \frac{-\Delta p}{L_p} \frac{\varepsilon^3}{\rho_f U^2 (1-\varepsilon)^2}$$

Experimental flow

Simulations are based on real experimental data to make comparison easier. For the well-structured porous medium, a packed bed of spheres of 3mm in diameter is chosen as investigated geometry for this dissertation. This traditional packed bed is experimentally studied by Dukhan et al. in following paper: *Experimental flow in various porous media and reconciliation of Forchheimer and Ergun relations* (Dukhan, Bağcı, & Özdemir, 2014).

Three different porous structures are used for experiments of water flow: 1-mm spherical packing, 3-mm spherical packing and a metal foam with each a porosity of respectively 35%, 35,5% and 87,6%. Dukhan et al. concluded discrepancies between reported permeability and form drag coefficient, because of incorrect detection of different flow regimes. Within this paper different pressure drop

correlations, Forchheimer and Ergun, are discussed for the turbulent or post-Darcy regime. Experimental setup investigated is chosen as such all different regimes have experimental results and can be simulated. The setup is divided in an inlet-, porous and outlet section. Geometry specifications are bundled in Table 1 and fluid properties of water in Table 2.

L_i (m)	L_p (m)	L_o (m)	D_c (m)	D_s (m)	ε (%)
0,2	0,304	0,2	0,0514	0,003	35,5

Table 1: Specifications of the tube dimensions (Dukhan, Bağcı, & Özdemir, 2014)

ρ (kg/m3)	μ (Pa s)	ν (m^2/s)
1000	0,001	1e-6

Table 2: Fluid properties (water)

The setup is built to withhold very low and very high flow rates, respectively Darcy and post-Darcy conditions. Pressure was measured by *Validyne pressure differential sensors*, model *DP15* and *DP45* for high and low range respectively. How pressure drop is measured is important to compare correctly experiments with simulations. Used sensors are transducers consisting of a diaphragm. (Validyne Engineering) Therefore, best comparison is made with simulated pressure drop between two cross sections. Details about this device is seen in Figure 3 and the operation manual. (Validyne Engineering) Accuracy of the low range sensor is ± 0,25% and of high range ± 0,5%. Including uncertainty of length and velocity gives for the Darcy region a maximum uncertainty of 1,56% for the pressure drop. For all other flow regimes, the uncertainty in the measured pressure drop is maximum 1,01% concerning the experiments of Dukhan et al.

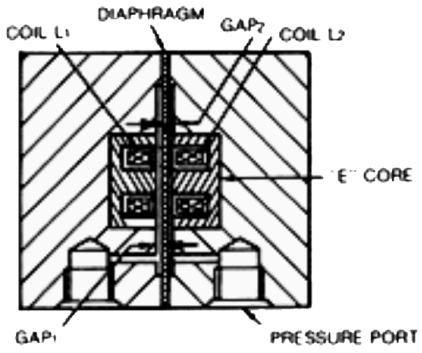


Figure 3: Typical transducer cross section for pressure drop sensors

Measured pressure drop is plotted by Dukhan et al. in Figure 4 and Figure 5. For a 3-mm spherical packing experimental data stretch from Darcy to turbulent regime and investigation of all regimes and transition(s) is possible. Reported flow regimes are summarized in Table 3. Each of these regimes hold different permeability and Forchheimer coefficient as seen in Table 4. Based on these values different cases to simulate are withdrawn as interesting and case choices are given in Table 5. Respectively by case number the case should simulate Darcy's

regime at low pressure drop, Darcy at its limit close to transition, transition regime, Forchheimer where inertia gets important and fully turbulent.

	Darcy	Transition	Forchheimer	Turbulent
Velocity Range (m/s)	[0; 0,007]	[0,007; 0,02]	[0,02; 0,09]	[0,09; …]

Table 3: Velocity range of different flow regimes (Dukhan, Bağcı, & Özdemir, 2014)

	K (*10^9 m^2)	F (m)
Darcy	6,18	NA
Forchheimer	7,34	0,43
Turbulent	6,61	0,39

Table 4: Permeability and Forchheimer coefficient in different flow regimes (Dukhan, Bağcı, & Özdemir, 2014)

	Case 1	Case 2	Case 3	Case 4	Case 5
Velocity (m/s)	0,001	0,006	0,0135	0,021	0,012

Table 5: Different cases that this dissertation wants to simulate

Dukhan et al. show with their experiments that also within the turbulent regimes an analogous Darcy's law with correction can be stated as:

$$\frac{dp}{dx} = \frac{\mu}{K_t} U + \frac{\rho F}{\sqrt{K_t}} U^2$$

As such the Forchheimer correction factor is also not truly constant, but constant in a specific flow regime: Forchheimer or turbulent.

For an estimation of the permeability K_D within Darcy's regime Kozeny's theory can be used. Estimation for spherical packing is given by:

$$K = \frac{\varepsilon^2 {D_p}^2}{36 * \kappa\ (1-\varepsilon)^2}$$

which gives for $\varepsilon = 0{,}335$, $D_p = 0{,}003m$ and $\kappa = 5{,}24$:

$$K = \frac{0{,}355^2 0{,}003^2}{36 * 2{,}24(1-0{,}335)^2}$$
$$K_D = 3{,}18e-8$$

This value is one order too high. This estimation takes into account the porosity, but also the structure is very important for permeability, because the structure determines the tortuosity and number of paths as such alters permeability.

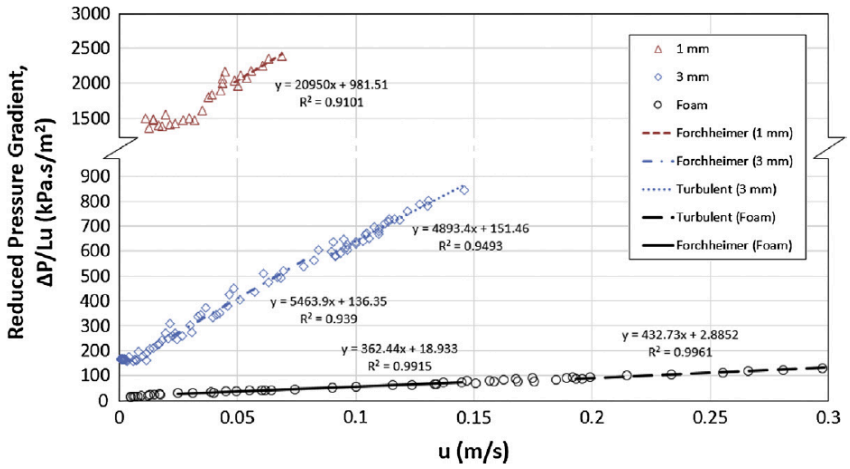


Figure 4: Reduced pressure in function of velocity for three different porous structures. (all data plotted)

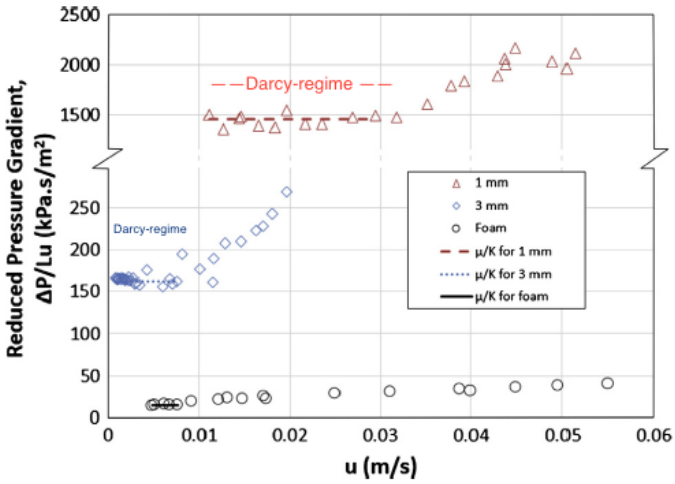


Figure 5: Reduced pressure in function of velocity for three different porous structures within Darcy, transitional and Forchheimer regime. (low range velocity data)

Apparent permeability

Another interesting article with additional information is: *The impact of porous media heterogeneity on non-Darcy flow behaviour from pore-scale simulation.* (Muljadi, Blunt, Raeini, & Bijeljic, 2015) This paragraph is based on this article.

Sometimes a flow with Re_p below one is said to be creeping flow, but the onset of non-Darcy flow can vary significantly depending on the porous structure. Simulation showed even a difference of three orders of magnitude for critical Re_K. Depending on the porous structure eddies can arise and increase tortuosity, which influences parameters. Determination of the porosity parameters can be done by experimental data, but also by X-ray imaging and direct pore-scale simulations. This paper defines the apparent permeability to highlight the transition from Darcy to non-Darcy flow. Equalling Darcy's law and Forchheimer correction defines the apparent permeability as follows:

$$\frac{\nabla p}{L\mu U} = \frac{1}{K_{app}} = \frac{1}{K} + \beta \frac{\rho U}{\mu}$$

and when plotting the experimental data one can determine the intersect and slope to get a value for K and β respectively like seen, as example, in Figure 6. This approach can be used for Forchheimer and turbulent regime.

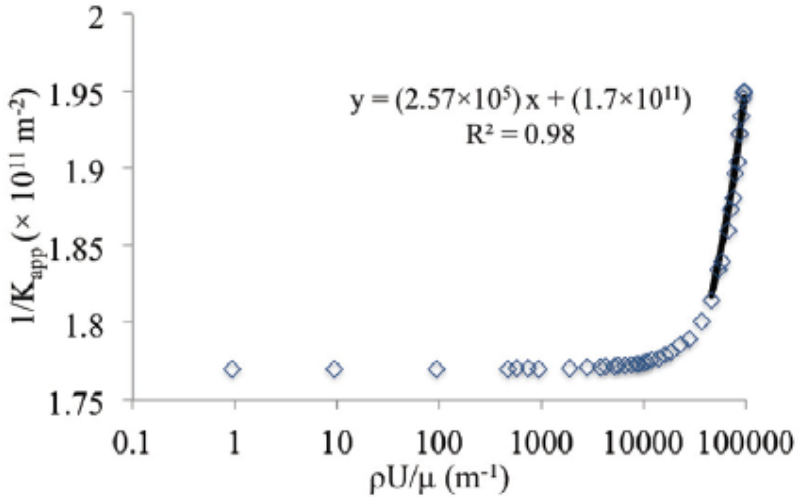


Figure 6: Plot of $1/K_{app}$ vs. ρU/ μ to get K_F or K_T and β

Chapter III: Geometry and mesh generation

The first step of a simulation is making the geometry and generating the mesh. This section gives different steps to take and gives specific information to build or treat porous structures.

Prerequisite information

This thesis looks to a flow that enters an inlet zone, where the velocity profile gets fully developed, goes through a porous region and exits passing an outlet zone. Building a cylindrical geometry with porous zone has more flexibility within Salome than *blockMesh*, an *OpenFOAM* utility.

There are two options to verify flow characteristics of a porous structure:

A. Building specific **porous structure**: Ideally the mesh gets divided into a structured mesh for zones without porous structure and unstructured mesh for the porous structure. Flow characteristics are obtained by using simpleFoam or icoFoam (with steadyState ddt scheme) as solver. Porosity parameters can be calculated during post-processing steps.

B. Starting from **porosity parameters**: A structured mesh can be generated for the whole tube where a porous, volumetric group (cellZone) is defined. Based on permeability and Forchheimer parameters flow characteristics can be solved by porousSimpleFoam or porousInterFoam for multiphase systems.

For the well-structured porous medium a packed bed of spheres of 3mm in diameter is chosen and an article (Dukhan, Bağcı, & Özdemir, 2014) as experimental data reference, which was summarized in section Experimental flow page 17. Geometry specifications were bundled in Table 1 and fluid properties in Table 2. Building such geometry is heavy for a normal computer when the amount of spheres increases. By experience it seems Salome needs more time to make spheres than (extra) looping in a *Python* script. Therefore, it's easier to make a *Python* script that efficiently creates automatically the whole geometry and groups boundary faces. Firstly, structured packings were generated as a simplification: simple cubic and cubic body centred (BCC). These scripts can be found in Appendix B. The first holds 48,4% of porosity ε or void space when adding $0{,}01 * R_s$ spacing between spheres. Manipulation of the latter to $0{,}034 * R_s$ spacing holds 35% like the experimental data of the article. Detailed calculation of porosity can be found in Appendix C. In reality spheres are randomly distributed inside a tube with fixed volume. For structured packings tortuosity is too low and unreal. Randomization gives a higher tortuosity. Next section explains how this randomization is achieved.

Randomization of spherical packings

Possibilities concerning the randomization of a spherical packing depends on the porosity. With problems like start-up of clogging or crystallization of a compound, where a low amount of spheres is inside a larger volume, randomizing coordinates is quite easy for not-moving particles. Decreasing porosity gives less space between the spheres and makes it tougher to get extra random spheres that don't overlap spheres already added. A script can be found in Appendix D where sphere coordinates are obtained with randomization of centroid coordinates when controlling the overlap or gap between spheres. This approach is limited and is interesting with high porous structures (porosity > 50%). Because of this limitation a DEM (Discrete Element Method) tool is used within the framework *Yade*. Within Yade creating a packing is automated and has two general possibilities:

1. Using **pre-build algorithms** like a high porous cloud (spherePack.makeCloud) and dense packing (spherePack.randomDensePack). But it is not possible to control and fix volume and porosity at the same time with these commands. Based on trial and error it is possible to reach a solution, but creating a close packing without overlaps by tension is an issue.
2. Using a **dynamical simulation** with forces that are applied on faces or volumes. At the end it is difficult to get a stress-free solution so overlaps of spheres are possible, because of tension. A script is made which can be found in Appendix E. This script generates a structured BCC packing inside a compact volume, uses mechanical engines to add harmonic motions to randomize packing by "shaking" the volume, compresses to a wanted volume and exports a text file with the co-ordinates of the randomly distributed spheres, which can be read in *Salome* and should first filter spheres too close to each other. Following Figure 7 gives the start and end from BCC to random packing, underneath a video where the whole mechanical evolution is simulated.

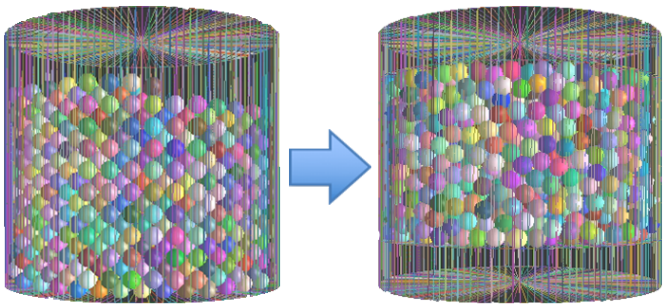
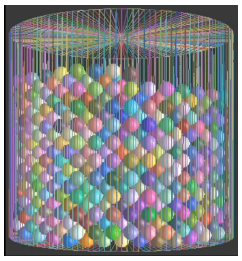


Figure 7: Randomization of BCC packing



Building geometry and meshing

All geometries in this section are created with *Python* scripts loaded into *Salome*. The flexibility of a *Python* script allows the user to create the geometry and group faces automatically for different sphere radii of the packing. Scripts can be run from a closable (*disown*) terminal window without graphical interface (*-t*) in background (*&*) with command "*salome –t name.py >log & disown*". If spheres are too close to each other geometrical and mesh problems can follow, therefore sphere radii are adapted so an overlap ($+x\%\ R_s$) or gap ($-x\%\ R_s$) is created. In Appendix F the filtering script can be found. There are different options to build a geometry depending on meshing choice. Following are the options ordered by decreasing complexity:

A. Structured/unstructured

For a structured mesh it is necessary to use *Divided Disk* to build a cylinder. This is a cylinder divided into sub-shapes, wherein it is possible to create a hexagonal mesh. To define the fluid domain spheres are cut. In this case one sphere needs to be cut out of several sub shapes. Internal faces and different cylinder zones can be seen in Figure 8, where a spherical packing of 15mm in diameter is taken. Decreasing the diameter or increasing amount of spheres in a fixed volume leads to problems of the cutting operation. Script can be found in Appendix B. Information about structured, unstructured meshing and building a script for a packed bed was found in topics on *cfd-online*. (Forum Salome-platform, 2015) Combining a structured, hexagonal mesh and unstructured, tetrahedral mesh needs a small trick. At the face between meshes there are pyramids that can't be treated by an UNV-file. Therefore, it is necessary to export the created mesh in *Salome* to a MED-file, import in gmsh, another external CAD software, and use the *OpenFOAM* command *gmshToFoam* instead of *ideasUnvToFoam* to add into an *OpenFOAM* case (Forum Salome-platform, 2009)

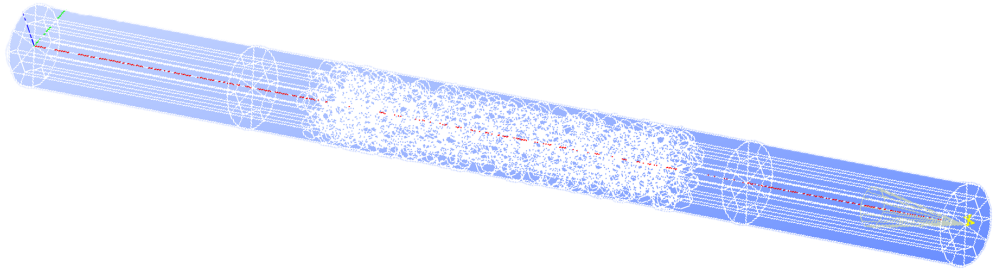


Figure 8: Geometry based on divided disk with 15mm packing

B. Unstructured with viscous layer

For an unstructured mesh there are no sub-shapes, but one cylindrical volume. This gives less internal faces so mesh algorithms have less difficulties. To add viscous layers (denser mesh closer to a boundary layer) it is necessary to have a transition zone between the inlet- and outletzone. Meshing needs to be separated into the algorithms *Tetrahedron (Netgen)* and *Netgen 1D-2D* to define viscous layers. This separation lead to problems.

C. Unstructured without viscous layer

The least complex approach has one cylindrical volume without transition region. Therefore, one meshing algorithm, *Netgen 1D-2D-3D*, can be used and reduces meshing errors. The script can be found in Appendix F. The whole geometry is built by periodic repetition of one tenth of the porous zone. This gives unreal interfaces. This geometry can be seen in Figure 9. Meshing was not achieved, because of narrow regions between the fused periodic parts. Later it will be seen that this part will be validated and afterwards it is interesting to try and build the whole geometry. Simulation results will use this approach with a quite large transition region (10% L_i). Adding a transition gives the possibility to define difference wall patches for the inlet, porous and outlet zone separately.

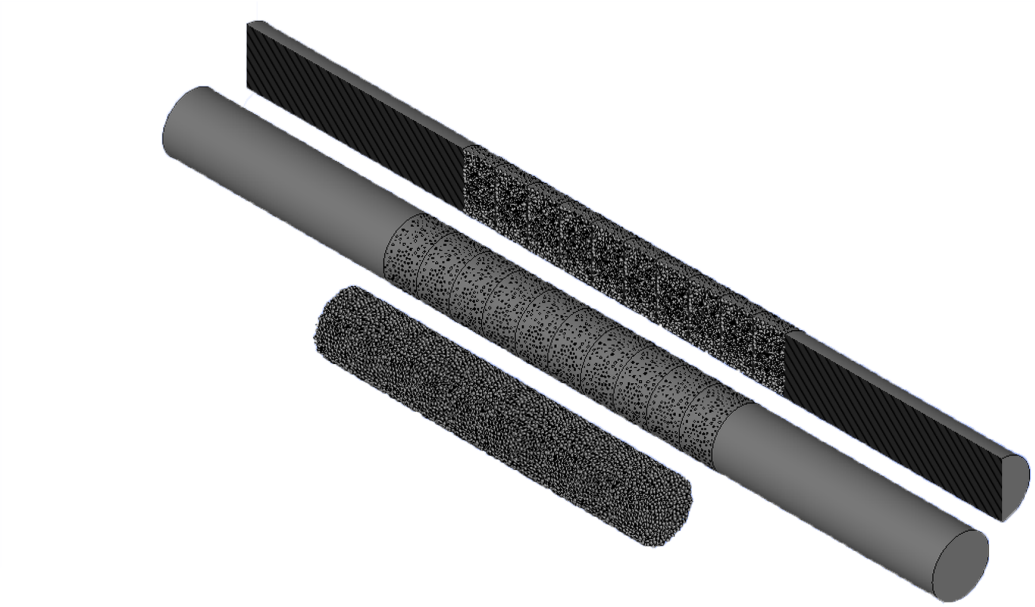


Figure 9: Geometry based on one cylindrical volume with 3mm packing

All the options are validated in next section Analytical control and validation of mesh in order to see if the error with respect to Hagen-Poiseuille tube flow increases (too much) with decreasing complexity. This analysis concludes that an unstructured mesh gives a rather good solution and will be used to calculate flow characteristics for the five cases suggested in section Experimental flow.

Analytical control and validation of mesh

Validation is always necessary to make a simulation more reliable. Comparison of analytical parameters and velocity profile needs to be done for a simple case (here: 3-dimensional tube flow). Choose of inlet velocity is based on the experimental data of Figure 5 from previous chapter. For the porous structure with 3-mm mono-sized spherical packing velocity 0,001 m/s lies in the Darcy-regime, where the reduced pressure is constant, because of the proportionality between pressure drop and velocity. (Dukhan, Bağcı, & Özdemir, 2014)

Analytical parameters

Calculation of analytical parameters is automated for different configurations for a simple Hagen-Poiseuille tube flow in an excel sheet. Specific calculations of flow parameters are given below and summarized for all wanted cases in Table 6. Case choices could be found in Table 5 section Experimental flow.

$$Re = \frac{\rho\, d\, |U|}{\mu} = \frac{d\, |U|}{\nu} = \frac{0{,}0514m\ 0{,}001\frac{m}{s}}{1e-6\frac{m^2}{s}} = 51{,}4$$

with Hagen-Poiseuille (only for laminar flow) :

$$x_{fd,H} = 0.05\, D\, Re = 0{,}05 * 0{,}0514m * 51{,}4 = 0{,}132m = 66\%\ of\ inlet\ zone$$

$$\begin{cases} u_{max} = \frac{1}{4\mu}\left(-\frac{dp}{dx}\right)R^2 \\ q = UA = -\frac{\pi}{8\mu}\left(\frac{\Delta p}{L}\right)R^4 \end{cases} => u_{max} = \frac{2q}{\pi R^2} = \frac{2q}{A} = 2 * U$$

$$q = UA = 0{,}001\frac{m}{s} * \pi\frac{0.0514^2}{4} = 2{,}075 * 10^{-6}\frac{m^3}{s} = -\frac{\pi}{8\mu}\left(\frac{\Delta P}{L}\right)R^4$$

$$=> \frac{\Delta p}{L} = 0{,}012 * \frac{Pa}{m} \quad \text{and } u_{max} = 0{,}002\ m/s$$

with friction losses (for laminar flow) over inlet zone:

$$h_f = f\frac{L}{D}\frac{U^2}{2g}$$

$$f = \frac{64}{Re} = \frac{64}{51{,}4} = 1{,}245$$

$$\frac{h_f}{L} = 1{,}245\frac{1}{0{,}0514m}\frac{\left(0{,}001\frac{m}{s}\right)^2}{2 * 9{,}81\frac{m}{s^2}} = 1{,}24 * 10^{-6};$$

$$\Delta p = h_f \rho g$$

$$Verification: \frac{\Delta p}{L} = 1{,}24 * 10^{-6} * 1000\frac{kg}{m^3}9{,}81\frac{m}{s^2} = 0{,}012\ \frac{Pa}{m}$$

Re	$x_{fd,H}$ (%)	u_{max} (m/s)	$\Delta p/L$ (Pa/m)	f (-)	h_f/L (-)
51,4	66	0,002	0,012	1,25	1,24e-6
308,4	396	0,012	0,073	0,208	7,41e-06
693,9	892	0,027	0,164	0,092	1,67e-05
1079,4	1387	0,042	0,497	0,059	2,59e-05
6168	257	0,24	1,453	0,036	1,48e-04

Table 6: Summary analytical parameters of laminar fluid flow for inlet zone

After this analytical control one can make several conclusions:

- $x_{fd,H}$

The length towards fully developed flow is given by a percentage of the inlet zone length. In nearly all cases the flow won't achieve full development before entering the porous zone. If it is necessary to have a fully developed flow before the porous zone the inlet length should be simulated longer as 0,2m or a coded inlet boundary layer should be used. But it should be pointed out, when looking to whole experimental setup, that the flow comes from a narrow tube to the inlet zone. As such the setup won't achieve full development either.

- $\Delta p/L$

The pressure drop for the non-porous zones are at least 1000 magnitudes smaller compared to the pressure drop over the porous zone.

Intermezzo

It is interesting to note that for duct simulations the maximum velocity is 1,5 times inlet velocity. Making a 2-dimensional simulation (z-direction has 1 cell and top and button faces are empty) between two walls gives the results for a duct and not for tube flow. So simulations should use 3D mesh.

Validation mesh cell density and meshing algorithms

Using the sampleDict it is possible to export Ux, velocity in the flow direction, data points over a specific line. Then comparison of these data points to analytical flow profile gives information of whether the mesh grading needs to be finer in the flow direction or close to boundary layers. Several tetrahedral meshes are investigated and summarized in the following Table 7.

maxSize	0,0703764	0,00703	0,0035		0,00175
minSize	0,0161183	0,00161	0,0007		0,00035
viscousLayers	no	no	yes		yes
quadrangles	no	no	no		no
non-orthogonality	73	49,8	60,5		61,8
u_{max}	[0,00116; 0,00139]	[0,00173; 0.0019]	[0,00188; 0,00193		[0,00192; 0,001935]

Table 7: Tetrahedral meshes with its characteristics and velocity solution

Looking to non-orthogonality of tetrahedral meshes choice of schemes is very important. Depending in the mesh type and geometry schemes should been chosen differently. Unstructured, tetrahedral meshes can take time to converge.

Therefore, it is important to use the "*checkMesh*" utility. This gives information about orthogonality, skewness and controls the mesh for errors. Depending on non-orthogonality different schemes with *limiters*[8] and *non-orthogonalCorrectors* should be taken. (Guerrero, Introductory OpenFOAM® Course: Tips and tricks, 2015) Incorrect choice can lead to more instable calculation and slow convergence. The utility "*renumberMesh*" further decreases the amount of iteration needed.

Tetrahedral mesh is compared to octahedral mesh for tube flow and summarized in Table 8 and meshes are shown in Figure 1.

	U_{in} (m/s)	u_{max} (m/s)	$x_{fd,H}$ (m)	$\Delta p/L$ (Pa/m)
Analytical	0,001	0,002	0,132	0,012
Octahedral	0,001	0,001985	0,13 – 0,15	± 0,0127
Tetrahedral	0,001	[0,00192; 0,001935]	0,13 – 0,15	± 0,0132

Table 8: Comparison between analytical parameters and simulation obtained with octahedral and tetrahedral meshes

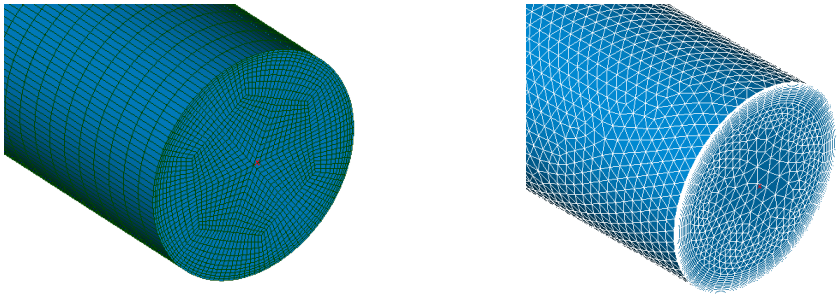


Figure 10: Validated mesh cell density for structured, unstructured without viscous layers

[8] When non-orthogonality is higher than 80 (checkMesh), facelimiters are added to gradSchemes and «blending factor» 0,333. More orthogonal meshes use celllimiters and blending factor up to 1 (orthogonal correction) or don't need correction. Within fvSolution non-orthogonalCorrectors can be changed as well, from 1 to 3 for respectively orthogonal and non-orthogonal mesh. (Guerrero, 2015)

Conclusion

After this part of making the geometry and meshing, several obstacles are tackled and some unsolved. The following two paragraphs try to explain some difficulties and the last paragraph concludes on how to go further.

Why is it a complex geometry?

Next summation gives several difficulties and reasons why a dense spherical packing is not that easy to handle:

- Spheres may not approach each other and the wall too closely. This is related to tolerance and which distances can be treated by computational calculation. Information about *Salome* development can be found in *Developer's documentation*. (Salome Platform, 2007-2015)
- Boolean operations like *Cut* are difficult operations and depend on tolerance, interference and intersection. (Salome Platform, 2001-2010) Therefore, the cut tool can sometimes give trouble when cutting out a sphere to define the fluid domain. In *Python* this is tackled by using try-except blocks.
- Packing is random and not structured therefore all possible outcomes need to be implemented.
- *Salome* can sometimes crash. Therefore, it is interesting to make brep.file backups during the calculation. These backups can be used as starting point when changing sphere radii manually. (see meshing problems)

How to solve mesh problems manually?

Mesh problems can always arise, mostly intersecting triangles or multiple edges. These (geometrical) errors seem random and the reason for this is not found. Therefore, these errors are solved manually by using "*bad mesh to group*" and "*bounding box*" or "*Modification – Add Node*" so the error can be located and assigned to a specific sphere. Then the following method is taken:

- Getting "*Centre of mass*" of a specific sphere. If only a close by point is found the script "*findPoint.py*" can be used to get the closest centroid. Make a slightly bigger sphere and cut again. If the *Cut* operation succeeds, go to step 3 immediately.
- Localize the sphere and rerun script (starting from a brep.file backup) to build geometry and make a specific sphere smaller so this sphere has no intersections with other boundaries.
- Rerun mesh algorithm "Netgen 3D-2D-1D".

How to go further?

To start it is easier to simulate partly the porous zone without fusing parts together and normalise fluid properties. Tetrahedral meshes can give also good solutions and for the porous zone unstructured mesh is the only possibility. One should remember that the spherical packing is not totally comparable to reality, because of induced overlaps and gaps. All script to build the geometry can be found in Appendix F. Next chapters will try to achieve good simulation results for the five cases.

Chapter III: Solving

When the mesh is generated the solving stage can start. This chapter gives information about the solvers used in *OpenFOAM*, how a solution is controlled by convergence criteria and schemes and takes a look at turbulency models.

Prior to running solve algorithms

After the geometry is build and a mesh is created without errors within *Salome* solving can still be aborted, because of choice of schemes, correctors and "checkMesh" failures. It is very important to run "*checkMesh*" prior to solving. This utility makes sets of incorrect cells, like *zeroVolumeCells*. The cells can be deleted using "*setSet*"[9] to define subsets and using "*subsetMesh*"[10] to delete them. When using *Netgen 3D-2D-1D* it is possible to play with minSize/maxSize and remake mesh. A bad mesh will always give troubles so normally remaking is better. For example, in case of adding turbulence models mesh manipulation is more sensitive and can give solving abortion and mesh remake is necessary anyway. Alternatively, one can try how the solution quality differs when leaving checkMesh failures or deleting "incorrect" cells. For example, 4 too skew faces on a million cells won't alter solution quality drastically (normally).

Solvers

Depending on geometry there are two solvers used. For explicit porous structures (spherical packing) the solver simpleFoam is used. Using Darcy's law and correction the solver porousSimpleFoam should be used and a "black box" geometry. More detailed description follows here:

A. SimpleFoam

Using simpleFoam solvers it is possible to have laminar Navier-Stokes or turbulent Navier-Stokes. Following equation is Navier Stokes and could have an extra term to describe turbulency effects, which will be discussed in section Laminar versus Turbulency models page 33.

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) = \frac{-\nabla p}{\rho} + \nu \nabla^2 \boldsymbol{u} + turbulency$$

B. PorousSimpleFoam

To use this solver there has to be a *cellZone* (*porousZone*) defined. It is even possible to define different *cellZones* with each a different set of porous parameters. PorousSimpleFoam can use two methods to obtain a source term for the momentum equation. Firstly, there is *DarcyForchheimer* that can define a local co-ordinate system to define permeability multidirectional for anisotropic structures. (Greenshield, 2014) The built-in solver can only define one viscous resistance and one inertial resistance term for each zone (CFD Support) Therefore,

---

[9] SetSet is a command-line utility. Following commands make a subset: 1) "*cellSet <name> new cellToCell zeroVolumeCells*" any 2) "*cellSet <name>" invert*" 3) "*cellSet <name> subset*". Now subsetMesh can be run. (OpenFOAM wiki)

[10] SubsetMesh puts a subset in an empty patch. Example: "*subsetMesh <name> -overwrite*". (OpenFOAM wiki) It can be necessary to remove this extra patch manually.

it would be interesting to include a mechanism for thresholds and changing of the constants. For this mechanism the solver will need to be adapted. Following equations show how the resistance vectors, $\boldsymbol{d}$ and $\boldsymbol{f}$, are defined, with a and b coefficients of the Darcy polynomial:

$$dp = \frac{\mu}{K}u + \frac{F}{\sqrt{K}}\rho u^2$$
$$dp = au^2 + bu$$
$$\boldsymbol{d} = \frac{b}{\mu}\,\boldsymbol{e_d} = \frac{1}{K}\,\boldsymbol{e_d}$$
$$\boldsymbol{f} = \frac{2a}{\rho}\,\boldsymbol{e_f} = \frac{2F}{\sqrt{K}}\,\boldsymbol{e_f} = 2\beta\,\boldsymbol{e_f}$$

and give the source term in three dimensions (CFD Support):

$$S_i = -(\mu d_i + \frac{1}{2}\rho|u|f_i)u_i$$

Secondly, the *powerLaw* method can be used, which is isotropic and uses the following source term with two empirical constants, C_0 and C_1. (CFD Support):

$$\mathrm{S} = -\rho C_0 |u_i|^{\frac{C_1-1}{2}}$$

Convergence control/criteria

Each iteration (or time step) *OpenFOAM* will give a "solution". Depending on what tolerance the user gives *OpenFOAM* will stop its calculation. To check whether this solution is converged and is a good solution following control mechanisms are used for this thesis' simulations:

- **Residuals**

With SIMPLE solvers it is possible to add *ResidualControl* to stop when specific residuals are reached for wanted variables. Only this won't give a certainty on a good solution. Normally for steady solution residuals will decrease over iterations, but for unsteady solutions due to turbulency, changing boundaries, … residuals can increase as well. (Guerrero, 2015) During calculation one can add "*>log*" to save iteration steps into a text file. Then *Residual Plots* can be drawn using *gnuplot*. (tutorial Heydlauff, 2009) In Figure 11 a residual plot is shown. Steady solution's residuals only decrease, but if *runTimeModifiable* is activated, schemes can be changed during the calculation to more accurate (higher order) ones. Here grad(U) scheme was changed from *Gauss Linear* to *leastSquares* as example, but becomes rapidly divergent.
It is sometimes possible to reduce initial instability and oscillations by using initial p and U files from another case, for example when stepping from laminar to a turbulency model.

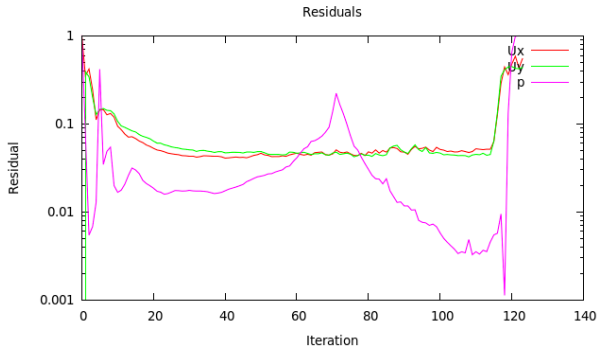


Figure 11: Ongoing residual plot of a simpleFoam simulation that diverges after scheme change

- **FunctionObjects**

With function objects in the *controlDict* it is possible to "postprocess" during run time. Tracking of variable values is possible. This can be done in iteration steps with specific functions. Locally, variables could have unreal values. Therefore, it is interesting to use *FieldMinMax* to get minimum and maximum values of velocity and pressure. (OpenFOAM docu) Evolution of maximum values could also give information about convergence. Even better are *faceSource* functions. With such functions it is possible to define new areas that don't have to be patches or cellZones and get variable values, averages or integrations. (OpenFOAM github, 2012) Immediately during run time Δp over the porous zone can be tracked without the need of waiting for full convergence and post processing steps. Adding following lines to the *controlDict* does the trick:

```
functions
{
  outletPressurePlane
  {
        type              faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled           true;
        outputControl     outputTime;
        log               true;
        valueOutput true;
        source            sampledSurface;
        sourceName        bla;
        surfaceFormat     vtk;
        sampledSurfaceDict
        {
              type              plane;      // always
triangulated
              basePoint       (0.233 0 0);
              normalVector    (1 0 0);
        }
        operation       areaAverage;
        fields
        (
              p
        );
  }
}
```

It seems, by experience, that for highly non-orthogonal meshes for irregular randomly shaped structures the Residuals don't all evolve to low values (order: 1e-4), but nevertheless property values can be stable and give the "correct" (simulation) value. As such high Residuals don't mean necessarily a bad solution. (Guerrero, Tips and tricks, 2015)

Laminar versus Turbulency models

Traditionally flow characteristics are divided between laminar and turbulent, because laminar Navies Stokes Equations (NSE) have an analytical solution, e.g. Poiseuille flow. This in contrast to turbulent flows who are always based on empiricism. That's why different turbulency models, within *OpenFOAM*, exist similar to all different correlations to characterize friction factor for example. Using *simpleFoam* gives the possibility of adding such turbulency models. There are various turbulency models that are advisable depending on the application. Different models are summed up and it is explained why they are (not) used for this dissertation:

- **RAS / RANS**

Generally, a stress term is added to the NSE to include turbulency effects:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) = \frac{-\nabla p}{\rho} + \nu \nabla^2 \boldsymbol{u} - \frac{1}{\rho}\nabla \tau$$

and stress is generally expressed as two terms, where the turbulent term takes into account the variation with respect to the average velocity and defines μ_T, the turbulent eddy viscosity, which is not a real physical property:

$$\tau = \underbrace{\mu \frac{\partial \boldsymbol{u}}{\partial y}}_{laminar} - \underbrace{\rho \overline{u'v'}}_{turbulent} = \mu \frac{\partial \boldsymbol{u}}{\partial y} - \mu_T \frac{\partial \boldsymbol{u}}{\partial y}$$

RAS simulations estimate the stress τ based on Reynolds averaging (Boussinesq Approximation) and gives τ^R the Reynolds-averaged stress (RAS) and Reynolds-Averaged Naviers Stokes (RANS):

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) = \frac{-\nabla p}{\rho} + \nu \nabla^2 \boldsymbol{u} - \frac{1}{\rho}\nabla \tau^R$$

Within the RAS group there are several methods such as k-Epsilon and k-Omega. Each one adds two transport equations to describe μ_T and τ^R (τ). The k-Epsilon model adds two variables allowing to calculate the diffusion and convection effect of turbulence energy. Respectively the transported variables named dissipation of energy, ε and turbulent kinetic energy, k. (CFD Online) The k-Epsilon is normally for full turbulent flows, but because it is easy to implement it is still widely used. (CFD Online, 2010) Nevertheless, this last statement is not generally applicable and in some cases incorrect. k-Omega models introduce the variable ω, which is turbulency frequency or vorticity and works well for rotational flows. This model has more accuracy for near wall treatment and is better for low-Reynolds flow. (CFD Online)

Again: all these extra variables and equations serve to measure μ_T and τ^R (τ).

Practically it is necessary to find appropriate initial and boundary conditions. Equations underneath help to get initial values (Guerrero, 2015), with I the turbulence intensity and viscosity ratio μ_T on μ:

$$\mu_T = \frac{\rho k}{\omega}$$
$$k = \frac{3}{2}(uI)^2$$
$$\omega = \frac{\rho k}{\mu}\left(\frac{\mu_T}{\mu}\right)^{-1}$$

For boundary conditions there are different options depending on the coarseness of the mesh: wall function models (*nutWallFunction*, *nutRoughWallFunction*, *epsilonWallFunction*, …), scalable wall functions or no wall function. (Greenshields, 2015) These functions are independent from the simulation, but influenced by the mesh. Defining a dimensionless wall distance:

$$y^+ = \frac{u_\tau\, y}{\nu}$$

with u_τ friction velocity and y the distance to the first cell centre normal to the wall. The friction velocity can be calculated and cinematic viscosity is (normally) fixed so yPlus (y^+) can only be influenced by the mesh size. Following equations help to get an initial estimation of yPlus and needed coarseness of the mesh.

$$Re = \frac{\rho D U}{\mu} \; or \; Re_p = \frac{\rho D_p U}{\mu} or \; Re_K = \frac{\rho\sqrt{K}U}{\mu}$$
$$u_\tau = \sqrt{\frac{\tau_w}{\rho}}$$
$$\tau_w = \frac{1}{2} C_f \rho U$$
$$C_f = \frac{1.328}{\sqrt{Re}} \; (valid\ for\ laminar\ flow)$$
$$C_f = \frac{3\,\varepsilon^{0.5}}{Re_K} \; (valid\ for\ porous\ zone)$$ (Ingham & Pop, 1998)

The difference between the simulation and wall function (black box) is shown in Figure 12. Wall functions should not be used when $y^+ < 30$.

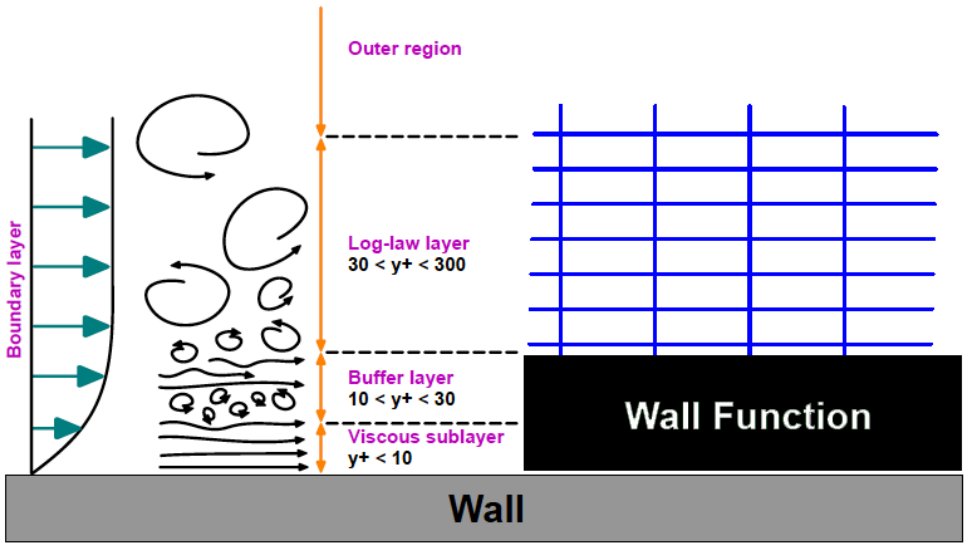


Figure 12: Wall function are black boxes for near wall treatment based on a model and not the simulation and span at least viscous sub layer and buffer layer (Guerrero 2015)

More information about boundary conditions for turbulency models can be found on CFD-Online (CFD-Online, 2014) and tools exist that help estimating all parameters. (CFD-Online) Using turbulency models it is necessary to start with upwind schemes for the *divSchemes* of turbulent parameters (k, ϵ, ω) otherwise it is much more likely to get divergence. During the simulation it can be tried to set higher order schemes active.

- **SST** (Shear Stress Transport)

This method combines different turbulence models. It formulates the k-ε behaviour for free flow and switches to k-ω for the inner parts of boundary layers. (Menter, 1994) It introduces around 10 extra coefficients and auxiliary relations. (CFD Online, s.d.) Therefore, the necessary computer power increases which slows down calculation time.

- **LES** (Large Eddy Simulation)

For very non-orthogonal meshes it is not advisable to use LES (Guerrero, Introductory OpenFOAM® Course: Tips and tricks, 2015). Using tetrahedral meshes is already non-orthogonal and introducing a complex porous structure will increase this non-orthogonality even further.

Chapter V: Results and discussion

Building the whole 3-dimensional geometry of 28779 spheres (3 mm) inside a millitube is not easy as earlier discussed. This is in fact only a matter of computer capacity. Therefore, it is preferred to make partly the porous zone and then normalise by length. After inlet boundaries there is always an entrance section where the flow is developing and a fully developed section. The flow collapses against the spheres and "searches" its new way through it. Simulating only partly the whole structure will give a different fraction of importance of entrance section and fully developed section. Furthermore, after a porous zone the flow expulses into the outletzone and creates vorticity (unsteadiness). Firstly, it is interesting to "see" these phenomena in a 2D slice including the whole porous length and keeping these effects in mind for the 3D case. How all these cases are build up can be found in Appendix F.

2-dimensional

In 2D the flow cannot move up and down, but only in a plane. The upper and lower parts are empty and therefore do not hold any physical condition. The result is that the flow feels less wall restrictions. Therefore, the flow acts like moving through two parallel walls. The number of possible paths gets drastically decreased and the flow needs to find its way in a very dense packed plane. So when maintaining the porosity, it is not possible to permeate through the whole length. Therefore, sphere radius is decreased from 1,5 mm average to 1,4 mm average to build simulations for this part.

In Figure 13 and Figure 14 simulation for an inlet velocity of 0,001 m/s is shown, which gives 0,0015 m/s as maximum velocity between parallel plates. (Caplow) The turbulent zone after the porous region is still changing even tough residuals are steady. Pressure drop and maximum velocity magnitude are also steady. Only velocity components are responsible for the shown changes. Δp over the porous zone equals to 0,214 kPa, which gives a reduced pressure drop of 7379 kPa s / m^2. This is a lot more than the 3D experiments.

It can be concluded that the zone after the porous zone is already quite turbulent at low velocity (Darcy). Normally this part is of interest, because it alters the pressure measurement and turbulency models could help to simulate this part. Besides the possibility of high velocities within pores this is a second reason why turbulency models could help simulations of porous structures.

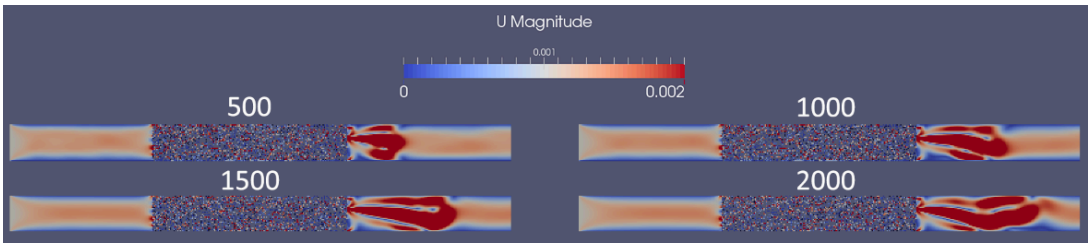


Figure 13: Velocity magnitude profile at iteration step 500, 1000, 1500 and 2000 for a parallel 2D porous plane

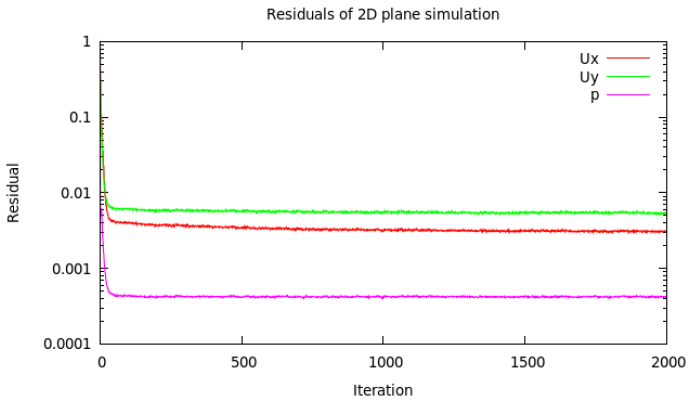


Figure 14: Residual plot for 2D porous plane

Trial 1

Simulation are mostly an iterative process. After some results, one can find some necessities to improve the geometry or mesh. The conclusion of the first trial is that the geometry doesn't represent enough the reality of the experiments. The mesh is important for the turbulency models (y^+) and showed for high-velocity cases better results. Next is normalisation discussed and the cases set in Experimental flow page 17.

Normalisation

Like discussed in the introduction, normalisation won't represent reality. The more spheres (longer simulated porous part) the better fraction entrance – fully developed represents reality. Following Table 9 gives the normalised pressure drop and Darcy permeability.

# Spheres	Δp/L	Simulation	Experiments
479 spheres	0,204 kPa/m	K_D = 4,9e-9 m²	K_D = 6,18e-9 m² (21% error)
2877 spheres	0,181 kPa/m	K_D = 5,5e-9 m²	K_D = 6,18e-9 m² (11% error)

Table 9: Normalised pressure drop over a porous zone (1st case scenario: u=0,001m/s)

Calculation of reduced pressure in Darcy's regime as experiments presented gives:

$$\frac{\Delta p}{Lu} = \frac{\mu}{K_D} = \frac{0{,}001\ Pa\ s}{6{,}18 * 10^{-9}\ m^2} = 161{,}8\ \frac{kPa\ m^2}{s} = constant$$

Experiments gave a normalised pressure drop of 0,1618 kPa/m (case 1). The trend of decreasing normalised pressure gives a good perspective, but is still too high.

Cases

Here simulation results are obtained for the different cases. Table 10 gives the results of the first trial where different *RASModels* are tried at different mesh sizes. The reduced pressure drop, between two cross sections, is always compared to the reduced pressure drop found by Dukhan et al. in their experiments.

U (m/s)	RASModel	ε (%)	Mesh size	Reduced Δp (Sim) kPa.m²/s	Reduced Δp (Exp) kPa.m²/s
0,001	Laminar	30,5 [11]	Coarse	181	161,8
	Laminar	30,5	Medium	186	161,8
	Laminar	30,5	Fine	193	161,8
0,006	Laminar	30,5	Coarse	258	161,8
0,0135	Laminar	30,5	Coarse	360	210
	kEpsilon	30,5	Medium	360	210
	kEpsilon	50 [12]	Fine	148	-
	kOmega	50	Fine	74	-
0,021	Laminar	30,5	Coarse	437	251
0,12	Laminar	30,5	Coarse	1368	739
	kOmega	30,5	Medium	1369	739
	kEpsilon	50	Fine	340	-

Table 10: Simulation results for different cases (Trial 1)

[11] Compression of packing was too much and was not adapted to meet porosity of Dukhan et al. experiments.

[12] Induced only gaps and not overlaps to get non-touching spheres

Following points give conclusions after investigation of the results:

- **Geometry and porosity**

For randomization of the bed (Randomization of spherical packings) a dynamical simulation was made. To achieve to wanted porosity the bed got compressed at the end. It seems *Yade* compressed slightly too much even though script did not contain errors. The amount of spheres is too much. To oppose this slight over-compression 2797 spheres will be taken in trail 2 instead of 2877. To motivate the normalisation also $1/5^{th}$ will be tried.
Because of tolerance, there are induced gaps and overlaps. It is possible that more overlaps were created as such lowering the possible paths, which increases pressure drop for the simulation. The filtering script is revisited and a good threshold would be maximum ± 5% R_s interval of touching to get a good interchange between gap and overlap. When inducing only gaps when spheres touch the porosity becomes much larger. This is done during simulations of Trial 1 and gave $\epsilon = 50\%$. This provided a lower limit.
Looking to Blake-Kozeny and Ergun equations the pressure is highly dependent on the porosity (power 2 -or 3). Results are very sensitive for incorrect porosity and therefore geometry is very important and should represent reality correctly to get good results.
Because of limitations only parts were simulated. It is possible to break up the geometry into several parts and construct different parts that can be fused. But the interface between parts always gives meshing troubles (tolerance). Suggested could be that parts (1/10th or 1/5th) get periodically repeated with a transition zone sufficiently large enough to eliminate tolerance issues.

- **Mesh**

Up to Forchheimer regime it seems laminar models are sufficient to give appropriate results. At high velocity range introducing turbulency models does not give different values. The boundary walls of the mesh are divided in inletWalls, porousWalls, sphereSurfaces and outletWalls. In this way fineness of the mesh could be manipulated close to each patch. The yPlus value was not always correct, because it should be between 30 and 300 to use wall function. For Trial 2 cases will be limited to Laminar models, because difference is not large. Note that the turbulency zone after the porous medium is not included and can still form a turbulency zone.
Different mesh sizes are always used (coarse – medium - fine) to validate mesh independency of the simulation results. When using laminar models for the next trial immediately finer meshes will be taken.

- **Packing**

Another possibility of incorrect representation could be the packing. The method taken is static and uncoupled from DEM. Spheres could be still flexible and move slightly which decreases the pressure needed. But for such a dense packing ($\epsilon =$ 35,5 %) this movement could be considered rather small. If needed it would be possible to make CFDEM simulations to couple fluid flow and particles.

- **Measurement**

As seen the turbulency after the porous zone can alter the measurement. It is important to exactly know how the measurement are taken and if this turbulency effect is included in the measurement. Experimental values clearly showed a constant reduced pressure in the Darcy region, but this is not seen for the simulation results. This can be due to different thresholds, because the porosity is not the same as experiments.

Trial 2

After the conclusions made in Trial 1 a better geometry is now made. The used meshes are immediately finer to get faster a mesh independency. The simulation results are bundles in Table 11. Results are better and closer to the experimental values. The reduced pressure in the Darcy region is still not constant. This could also be due to the entrance effects that the flow is not yet smoothened and developed. In the turbulent regime results are more accurate. More cases are added to be able to get at least two points within the same regime to construct apparent permeability plots. Also an additional point in the Darcy regime is taken to check whether reduced pressure drop is constant.

U (m/s)	RASModel	ε (%)	Mesh size	Reduced Δp (Sim) kPa.m²/s	Reduced Δp (Exp) kPa.m²/s
0,001	Laminar	38 [13]	Medium	109,8	161,8
		38	Fine	110,5	161,8
0,002	Laminar	38	Medium	118,2	161,8
0,006	Laminar	38	Medium	151,5	161,8
0,0135	Laminar	38	Medium	198,7	210
0,021	Laminar	38	Medium	239,5	251
0,08	Laminar	38	Medium	506,8	574
0,12	Laminar	38	Medium	732,0	739
	kEpsilon	38	Coarse	715,1 [14]	739
0,15	Laminar	38	Medium	873,9	885,5

Table 11: Simulation results for different cases (Trial 2)

[13] Porosity is slightly higher, because of unpredictable *Yade*'s compression error, but mostly because some spheres are left out to remove cutting issues.
[14] Value seemed stable, but kEpsilon diverges suddenly (bounded k and epsilon)

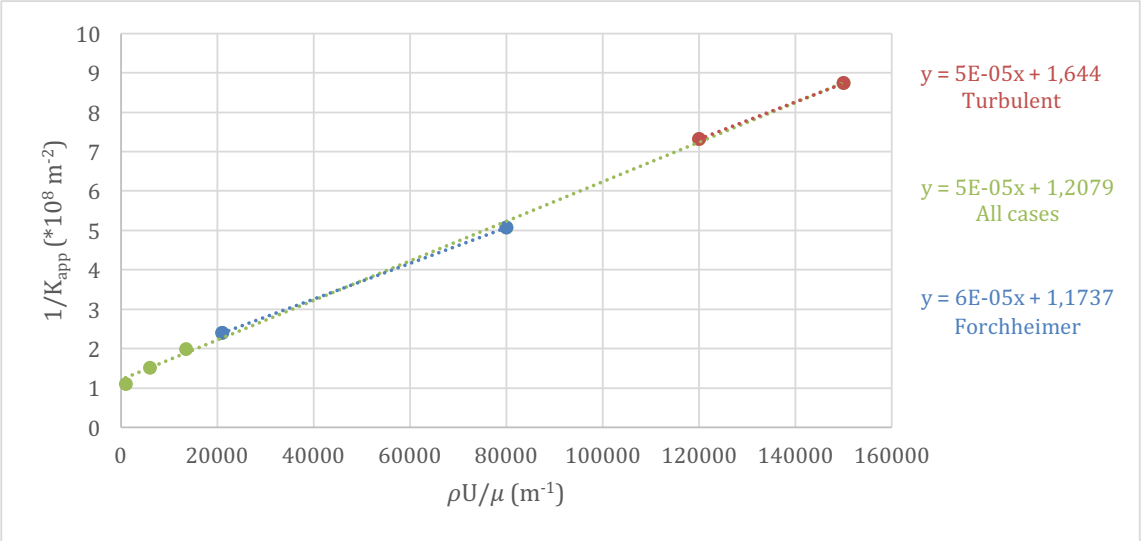


Figure 15: Plot apparent permeability interpolation of simulation results

To obtain the porous parameters, permeability and Forchheimer, the simulations results are interpolated with following equation:

$$\frac{\nabla p}{L\mu U} = \frac{1}{K_{app}} = \frac{1}{K} + \beta\frac{\rho U}{\mu} = \frac{1}{K} + \frac{F}{\sqrt{K}}\frac{\rho U}{\mu}$$

Using the slope and intercept gives the parameter values which are compared with the experimental values in Table 12. It is normal that the porous parameters are higher, because porosity is higher as such less resistance to flow. It is remarkable that within the turbulent regime parameters are more similar. The pressure drop over the porous zone at low velocity, Darcy region, has never been constant in the obtained simulations, neither with the additional case ($U = 0{,}002 m/s$). It can be concluded that the explicit method doesn't predict perfectly different regimes and knowledge obtained from the experiments by Dukhan et al. where necessary. Looking at Figure 15 it is very difficult to distinguish different regimes. At low velocity there should be a platform for the Darcy regime.

	Simulated K (*10⁹ m²)	Simulated F (m)	Experiment K (*10⁹ m²)	Experiment F (m)
Darcy	9,10 / 6,58	NA	6,18	NA
Forchheimer	8,52	0,55	7,34	0,43
Turbulent	6,08	0,39	6,61	0,39
All	8,28	0,46		

Table 12: Comparison between parameters (permeability and Forchheimer) from simulation results and experiments by Dukhan et al.

Conclusion

This section gives more conclusions about the results, comparison with other methods and what should go next. For the conclusions about building the geometry it is better to go back to the section Conclusion in Chapter III: Geometry and mesh generation.

Results and comparison

The explicit method is very sensitive and should be exactly representing reality to get correct results, because of the high dependency on porosity (ε). Building the geometry was quite time-consuming, but getting a steady station solution with the solver *simpleFoam* goes quite rapid. It is shown possible to get permeability and Forchheimer parameters that converge to the experimental values. These parameters can be used for implicit methods. Comparing analysed experimental values based on apparent permeability and Darcy's law with Forchheimer's correction with simulation results have shown there is a clear link between explicit methods using Navier-Stokes and black-box equations like Darcy and its extension. Simulations with the solver *porousSimpleFoam* did not give presentable results. This a disappointing fact. The porous cellZone defined in *OpenFOAM* didn't show any resistance to the flow. Therefore, simulation results cannot be compared.

After analysis of how implicit methods work, it can be said that there are some advantages and disadvantages for sure. Having the porous parameters could ease simulations a lot when using implicit simulation methods. But because this method is black box, it won't be possible to investigate turbulency zone and vorticity after the porous structure neither an explicit degree of mixing obtained by the porous structure.

What's next?

Researching never ends. This dissertation's analysis opens other doors for further investigating. Following paragraphs give some possibilities how to tackle better flow analysis of porous structures.

The most used solver was *simpleFoam* which is a steady state solver. Obtaining results was quite fast compared to building geometries. *OpenFOAM* has a built-in utility *decomposePar* to fasten simulation, but *Salome* did not have this. Manually decomposing in separate parts is not possible due to tolerance issues of fused parts. Therefore, it could be interesting to make such utility for *Salome*. This would speed up the process for explicit methods.

The final obtained results are showing convergence towards the experimental values. These results were good in the higher velocity region, but less good at the lower velocity region. At this moment only $1/10^{th}$ of the whole structure was simulated. Further validation would be interesting. Building $1/5^{th}$ was 5 times more time-consuming for Salome which gives 5 days of "cutting" out spheres. Currently this geometry is edited manually to remove meshing errors. Alternatively, blocks of $1/10^{th}$ are fused together with enough spacing ($2*R_s$) and would give a possibility to simulate the whole porous region. Without the ability to decompose *Salome* over different processors, the meshing of this geometry is

slow and has the same computational speed on a personal computer as a *workstation* with additional processors. More information and unfinished cases can be found in Appendix G: What's next.

Implicit method and *porousSimpleFoam* should be further investigated. Adapting the solver to include different flow regimes with each a different set of permeability and Forchheimer coefficient could make it universally applicable as black-box approach. Further, it should be investigated how the explicit and implicit method couple with other models like reaction or adsorption application. Maybe in some cases like absorption explicit knowledge of surface area should be known and implicit method won't be possible. Within *OpenFoam* there is also a solver porousInterFoam. It could be interesting to investigate how well this can show mixing results compared to explicit (*InterFoam*) modelling.

General conclusion

Simulating flow through a complex porous medium can be time-consuming. Using explicit methods to simulate the flow can help to predict and validate experiments for correct characterization of flow characteristics in different flow regimes and calculating permeability parameters. These could be used for analytical analysis and black-box simulations. This is a good perspective for predicting and simulating flow for large industrial processes when the correctness of structural and flow parameters are validated.

Bibliography

Brown, T. L., H. Eugene LeMay, J., Bursten, B. E., Murphy, C. J., & Woodward, a. P. (2012). Sample Exercise 12.1 Calculating Packing Efficiency. In *Chemistry, The Central Science, 12th Edition.* Pearson.

Caplow, T. (n.d.). *Poiseuille Flow Between Parallel Plates (Laminar).* Retrieved from http://www.columbia.edu/: http://www.columbia.edu/cu/gsapp/BT/RESEARCH/Arch-atmos/plates.html

CFD Online. (2010, April 28). *Use of k-epsilon and k-omega Models.* Retrieved from CFD Online forum: http://www.cfd-online.com/Forums/main/75554-use-k-epsilon-k-omega-models.html

CFD Online. (n.d.). *cfd-online.com*. Retrieved May 11, 2016, from SST k-omega model: http://www.cfd-online.com/Wiki/SST_k-omega_model

CFD Online. (n.d.). *K-epsilon models.* Retrieved March 30, 2016, from CFD Online: http://www.cfd-online.com/Wiki/K-epsilon_models

CFD Support. (n.d.). *Defining porosity using Darcy law.* Retrieved May 6, 2016, from cfdsupport.com: http://www.cfdsupport.com/OpenFOAM-Training-by-CFD-Support/node223.html

CFD Support. (n.d.). *Defining porosity using Power law.* Retrieved May 6, 2016, from cfdsupport.com: http://www.cfdsupport.com/OpenFOAM-Training-by-CFD-Support/node224.html

CFD-Online. (2014, January). *Turbulence free-stream boundary conditions.* Retrieved from CFD Online Wiki: http://www.cfd-online.com/Wiki/Turbulence_free-stream_boundary_conditions

CFD-Online. (n.d.). *Turbulence Properties, Conversions & Boundary Estimations.* Retrieved from CFD Online Tools: http://www.cfd-online.com/Tools/turbulence.php

Chachuat, B. (2015, December 10). Multi-scale Modeling of Light-limited Growth in Microalgae Production Systems. Leuven.

Dukhan, N., Bağcı, Ö., & Özdemir, Ö. (2014, June 14). Experimental flow in various porous media and reconciliation of Forchheimer and Ergun relations. *Elsevier Science Direct*, 9.

Forum Salome-platform. (2009, August). *Openfoam conversion*. Retrieved from Salome-platform: http://www.salome-platform.org/forum/forum_10/thread_3005

Forum Salome-platform. (2015, March). *How to mesh a cylindrical tube packed randomly with contacting spheres*. Retrieved from salome-platform: http://salome-platform.org/forum/forum_10/950427366

Greenshield, C. (2014, February 17). *OpenFOAM 2.3.0: Physical Modelling; Co-ordinate System Specification.* Retrieved May 6, 2016, from The OpenFOAM Foundation: http://openfoam.org/release/2-3-0/physical-modelling/

Greenshields, C. (2015, March 1). *OpenFOAM User Guide: 7.2 Turbulence models.* Retrieved April 30, 2016, from CFD Direct: http://cfd.direct/openfoam/user-guide/turbulence/

Guerrero, J. (2015, July 13-17). *Introduction Course OpenFOAM: Steady and Unsteady*. Retrieved from University of Genoa, DICCA:

http://www.dicat.unige.it/guerrero/oftraining/6_4fvm_transient_simulations.pdf
Guerrero, J. (2015, July 13-17). *Introductory OpenFOAM® Course: Tips and tricks.* Retrieved February 10, 2016, from University of Genoa, DICCA: http://www.dicat.unige.it/guerrero/oftraining/9tipsandtricks.pdf
Guerrero, J. (2015, July 13-17). *Introductory OpenFOAM® Course: turbulence modeling*. Retrieved from University of Genoa, DICCA: http://www.dicat.unige.it/guerrero/oftraining/8_1OF_advanced_modeling_turbulence.pdf
Harvard Corporation. (2016). *Surface & Depth Filter Types.* Retrieved May 25, 2016, from Harvard Corporation: http://www.harvardcorp.com/filtration-process/surface-depth-filters.html
Heydlauff, W. (2009, April 30). *Tutorial of how to plot residuals !* Retrieved from cfd-online.com: http://www.cfd-online.com/Forums/openfoam-solving/64146-tutorial-how-plot-residuals.html
Ingham, D., & Pop, I. (1998). In D. B. Ingham, & I. Pop, *Transport Phenomena in Porous Media.* Oxford: Elsevier.
King, H. (2005-2016). *Hydraulic Fracturing of Oil & Gas Wells Drilled in Shale.* Retrieved from Geoscience News and Information: http://geology.com/articles/hydraulic-fracturing/
Masashi, M., Fumio, W., & Toshihiko, K. (1980). *US Patent No. US 4368144 A.*
Menter, F. (1994). In *Two Equation Eddy-Viscosity Turbulence Models for Engineering Applications.* California: AIAA Journal. Retrieved from http://www.cfd-online.com/Wiki/SST_k-omega_model
Muljadi, B. P., Blunt, M. J., Raeini, Q. A., & Bijeljic, B. (2015, June 13). The impact of porous media heterogeneity on non-Darcy flow behaviour from pore-scale simulation. *ScienceDirect*, 12.
National Academy of Sciences. (2000). *Catalytic Process Technology.* Retrieved from NAP OpenBook: http://www.nap.edu/read/10038/chapter/4
OpenFoam. (2015, May 21). *User Guide Version 2.4.0.* Retrieved from CFD Direct: the architects of OpenFOAM: http://cfd.direct/openfoam/user-guide/
OpenFOAM docu. (2011-2016). *fieldMinMax Class Reference Field function objects*. Retrieved from OpenFOAM documentation: http://openfoam.com/documentation/cpp-guide/html/a00802.html
OpenFOAM github. (2012, March). *faceSource: add area-normal integration or averaging*. Retrieved from github.com/OpenFOAM: https://github.com/OpenFOAM/OpenFOAM-2.1.x/blob/master/src/postProcessing/functionObjects/field/fieldValues/faceSource/faceSource.H
OpenFOAM wiki. (n.d.). *SetSet.* Retrieved May 6, 2016, from Unofficial OpenFOAM wiki: https://openfoamwiki.net/index.php/SetSet
Pandey, A. (2003, March). Solid-state fermentation. *Elsevier*, 81-84.
Rhodes, M. (2008). Fluid Flow Through a Packed Bed of Particles. In M. Rhodes, *Introduction to Particle Technology – Second Edition* (Vol. 6, p. 466). Monash University, Australia: John Wiley & Sons Ltd.
Salome Platform. (2001-2010). *General Fuse Algorithm, Partition Algorithm, Boolean Operations Algorithm Backgrounds.* Retrieved February 9, 2016, from SALOME : The Open Source Integration Platform for Numerical

Simulation: http://docs.salome-platform.org/7/gui/GEOM/SALOME_BOA_PA.pdf
Salome Platform. (2007-2015). *Developer's Documentation.* Retrieved February 9, 2016, from Salome Platform Documentation: http://docs.salome-platform.org/7/
Smets, I. (2015, November). Solid State Fermentation. *Microbial Process Technology*. Leuven, Belgium.
Soulaine, C. (2015). On the origin of Darcy's law. In C. Soulaine. Stanford: Stanford.edu.
Sparkes, M. (2014, Jan 13). Supercomputer models one second of human brain activity. *The Telegraph*, 1.
Taherzadeh, D., Picioreanu, C., & Horn, H. (2012, April 4). *Mass Transfer Enhancement in Moving Biofilm Structures.* Retrieved from National Center for Biotechnology Information: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3318122/
Validyne Engineering. (n.d.). *General Operating Instructions of Validyne Variable Reluctance Pressure Transducer.* Retrieved from http://validyne.com/query/DP15_Variable_Reluctance_Pressure_Sensor_Capable_of_Range_Changes
Van Der Bruggen, B., Vandecasteele, C., Van Gestel, T., Doyen, W., & Leysen, R. (2003). A review of pressure-driven membrane processes in wastewater treatment and drinking water production. *Amer inst chemical engineers*, 46. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/ep.670220116/abstract
Versteeg, H. K., & Malalasekera, W. (2007). *An Introduction to Computational Fluid Dynamics: The Finite Volume Method 2nd edition.* Harlow, Essex, England: Pearson.

Appendices

A. General computational procedure to characterise property distributions

Computational Fluid Dynamics (CFD) characterises fluid flows. This section gives a general procedure how property distributions are evaluated (e.g. velocity and pressure distributions). The procedure follows three steps: pre-processing, solving and post-processing. A complete introduction about simulation and its steps can be found in the book: *An Introduction to Computational Fluid Dynamics: The Finite Volume Method 2nd edition* (Versteeg & Malalasekera, 2007)

Pre-processor

Pre-processing consists of making the geometry and mesh generation. This can be done by an *OpenFOAM* build-in mesh generator *blockMesh* or by external software like *Salome*, version 7.6, where the possibility is to make a mesh using the *Salome* modules or loading a *Python* script. For all geometries made *Salome* is used. Meshing divides the full domain into discrete control volumes or mesh cells.

Solver

This part describes how Partial Differential Equations (PDEs) that describe transport phenomena (nature laws) are solved. Internally the solver integrates and discretises the equations to get finally the solution.

Integration

The finite volume method uses the transport equation integrated over a finite control volume as a starting point for computational procedures. (Versteeg & Malalasekera, 2007)

Discretisation

Integration of a specific transport equation at a nodal point yields a discretised equation, wherein property and (proportional) coefficient are evaluated at the nodal point and gradients linearly approximated. For parameters at cell boundary, surrounding the nodal point, the average between the parameter's value at the two adjacent nodal points is taken (central differencing method). All of this gives a discretised, algebraic form. For convection problems the method is different, because it spreads influence in the flow direction and not in all directions like diffusion. For convective fluxes property values are necessary at faces between adjacent nodes. Calculating of the value of transported properties at control volume faces can give troubles. Therefore, there are different discretisation schemes (QUICK, power-law and differencing schemes) to obey conservativeness, boundedness (convergence of iteration) and transportiveness (Peclet Number). (example eq. 4.10 book of:) (Versteeg & Malalasekera, 2007)

Solution

Matrix solution methods are designed to solve the set of discretised, algebraic equations of each node to obtain a distribution of a property. (examples section

4.3 and 4.6 book of:) (Versteeg & Malalasekera, 2007) For transient solvers the Courant number is important. For steady state solvers, e.g. *simpleFoam*, Courant is meaningless. This number gives how many mesh cell a particle has "seen" during a time step. For Co higher than 1 the Lagrangian path of a particle is longer than a mesh cell size. This gives a numerical instability. Following the definition: $Co = \frac{\delta t|U|}{\delta x}$. *OpenFOAM* has different build-in solver. The important solvers used for this dissertation can be found in following Table 13.

Solver	Description
icoFoam	Transient solver for incompressible, laminar flow of Newtonian fluids
simpleFoam	Steady-state solver for incompressible, turbulent flow
porousSimpleFoam	Steady-state solver for incompressible, turbulent flow with implicit or explicit porosity treatment
porousInterFoam	Solver for 2 incompressible, isothermal immiscible fluids using a VOF (volume of fluid) phase-fraction based interface capturing approach, with explicit handling of porous zones

Table 13: Used solvers within this papers that are needed for simulation of porous structures (OpenFoam, 2015)

Post-processor

Validation of the code requires detailed knowledge of boundary conditions, used solver (fvSolution, fvSchemes) and experimental data. Important is to validate firstly the geometry and mesh. A post-processing tool is paraFoam.
OpenFOAM has several interesting commands to simplify some operations. A summary can be found in Table 14.

Command	Procedure Step	Definition
refineMesh	After mesh generation	makes a mesh finer (better flexibility in Salome)
mirrorMesh	After mesh generation	mirrors a mesh when a mirror axis is defined in mirrorMeshDict
transformPoints	After mesh generation	scales each dimension with (different scalar)
checkMesh	After mesh generation	checks mesh and prints warnings and errors
Co	Post-processing	prints Courant number for each written timestep
foamCalc	Post-processing	calculates for example the different components of U
sample	Post-processing	generates matrices of property distributions when a sampleDict is defined

Table 14: Interesting terminal commands within OpenFOAM

B. Structured packing

Scripts to create a simple cubic and bcc packing can be found in github: https://github.com/AdriaanDB/Thesis-2016-KULeuven/tree/master/Structured-Packing

C. Calculation of void space for spherical packing

This appendix looks to the calculation of two arrangements of spherical packing. The first is called *primitive cubic*, which is a repetition of one A-layer, and the second *body-centred cubic,* which repeats alternating A- and B-layers. Normally in these arrangements spheres are touching, but to avoid (meshing) problems it is better to add an additional spacing, Δs, between the spheres. Following the calculation of the porosity ε:

Primitive cubic

$$V_{cell} = (2 * R_s + \Delta s)^3$$
$$V_{sphere} = \frac{4}{3}\pi * {R_s}^3$$

where $\Delta s = 0$:

$$packing\ effiency = \frac{\frac{4}{3}\pi * {R_s}^3}{(2 * R_s)^3} = \frac{\frac{4}{3}\pi}{8} = 0{,}524$$
$$\varepsilon = (1 - packing\ efficiency) * 100\% = 47{,}6\%$$

where $\Delta s = 0{,}01 R_s$:

$$packing\ effiency = \frac{\frac{4}{3}\pi}{8{,}12} = 0{,}516$$
$$\varepsilon = 48{,}4\%$$

Body-centred cubic

Trigonometry (Pythagoras) teaches us that side length of the cube, Δz, is:

$$\Delta z = \frac{4R_s}{\sqrt{3}}$$

then one can calculate the rest:

$$V_{cell} = (\Delta z + \Delta s)^3$$

where $\Delta s = 0$:

$$packing\ efficiency = \frac{\mathbf{2} * \frac{4}{3}\pi * {R_s}^3}{(\Delta z)^3} = \frac{\frac{8}{3}\pi}{\frac{64}{3^{3/2}}} = 0{,}680$$
$$\varepsilon = (1 - packing\ efficiency) * 100\% = 32{,}0\%$$

where Δs is so that porosity equals to 35% (in a box):

$$packing\ effiency = 1 - \varepsilon = 0{,}65$$
$$\Delta s = x\,[\%] * R_s$$
$$packing\ efficiency = \frac{\mathbf{2} * \frac{4}{3}\pi * {R_s}^3}{(\Delta z + \Delta s)^3} = \frac{\frac{8}{3}\pi}{(\frac{4}{\sqrt{3}} + x)^3} = 0{,}65$$

$$x = \sqrt[3]{\frac{\frac{8}{3}\pi}{0{,}65}} - \frac{4}{\sqrt{3}} = 3{,}36\%$$

Body centred cubic is closer to a random spherical packing. The experimental data of (Dukhan, Bağcı, & Özdemir, 2014) have a porosity of 35%. Therefore, the last calculation looks how much additional spacing needs to be added between the spheres to obtain this porosity. It is important to note that not only porosity determines fluid flow properties. Fluid flow characteristics are different when the amount of possible paths and narrowness is different. (Dukhan, Bağcı, & Özdemir, 2014)

More examples and information about calculation of packing efficiency, crystal structures and ionic structures can be found in a typical handbook like *Chemistry, the Central Science*. (Brown, H. Eugene LeMay, Bursten, Murphy, & Woodward, 2012)

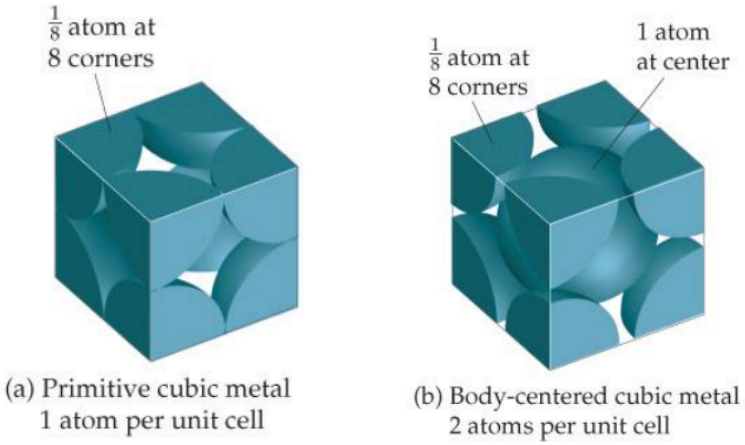


Figure 16: Arrangement spherical packing (Brown, H. Eugene LeMay, Bursten, Murphy, & Woodward, 2012)

D. Randomization based on random coordinates

Script can be found on github:
https://github.com/AdriaanDB/Thesis-2016-KULeuven/tree/master/Random-Packing

E. Randomization based on DEM (Yade)

Script can be found github:
https://github.com/AdriaanDB/Thesis-2016-KULeuven/tree/master/myYade

F. Building geometry for unstructured mesh and solving cases

Scripts can be found on github:
https://github.com/AdriaanDB/Thesis-2016-KULeuven/tree/master/Export

Script's goal:
Building a part from filtered centroids: build.py
Fusing periodic porous zones with inlet- and outletzone: makeWhole.py
Define boundary faces and mesh: makeGroups.py

Solving cases can be found on github:
https://github.com/AdriaanDB/Thesis-2016-KULeuven/tree/master/OpenFOAM/adriaan-2.4.0/run

G. What's next

Several trials are already set up for further investigation.

Building 1/5th

Computation time increased a lot to build 1/5th. Therefore, it was broken down after 5 days with 4500 of 5550 spheres were cut to try to have results and time. Then it was determined that $L_p = 48{,}6\,mm$. With an additional run and if test (x<48,6) wanted porosity is achieved. Currently, the geometry is adapted 6 times after each time a mesh error that is located and "cured" manually. Geometry can be seen in Figure 17.

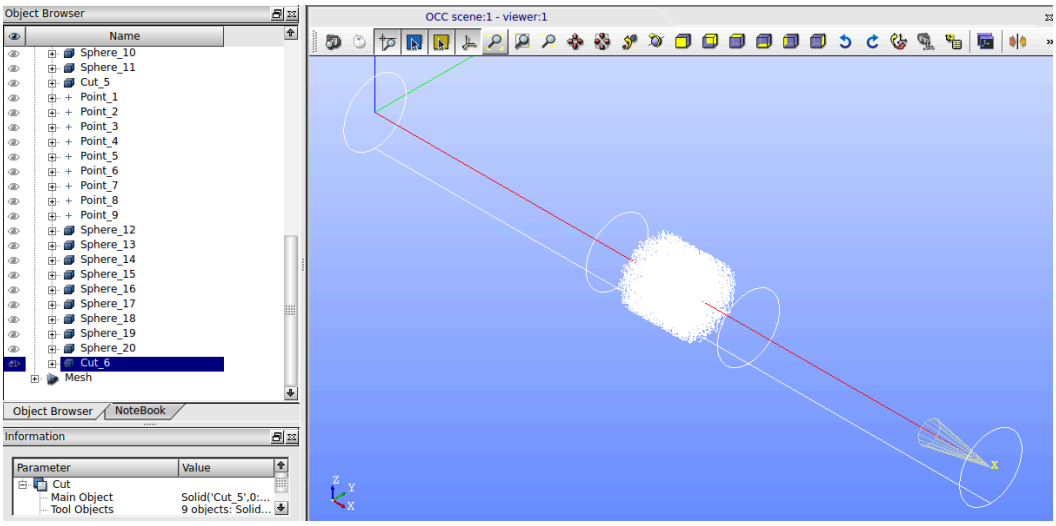


Figure 17: Screenshot Salome : geometry of a tube with nearly 1/5th of the porous zone

Full length of porous zone

Building the whole geometry based on periodic repetition is only useful when the periodic unit is validated to represent reality correctly. Results of Trial 2 were good enough to use for building the whole geometry by repetition. As can be seen in Figure 18 there is "safety" zone between units so tolerance won't be an issue at these induced interfaces.

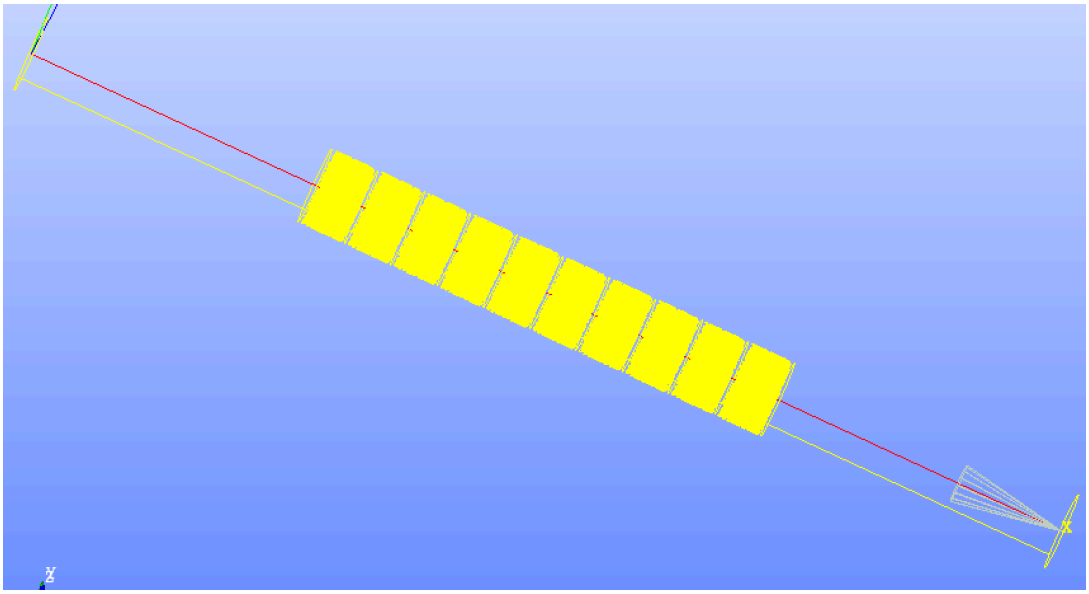


Figure 18: Whole geometry build by periodic repetition of 1/10th

Mixing

Scripts and cases for further investigation can be found on github: https://github.com/AdriaanDB/Thesis-2016-KULeuven/tree/master/What-s-Next

There is case that is analogous to the *damBreak* tutorial to check how water behaves when collapsing against the porous zone when the tube is also partly filled with air. Currently this is more a closed system than an open tube system.