TECHNISCHE
UNIVERSITÄT
DRESDEN

Fakultät Umweltwissenschaften,
Fachrichtung Hydrowissenschaften,
Institut für Grundwasserwirtschaft

Computational Fluid Dynamics in Unconsolidated Sediments: Model Generation and Discrete Flow Simulations

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium (Dr. rer. nat.)

vorgelegt
der Fakultät Umweltwissenschaften
der Technischen Universität Dresden

von Dipl.-Math. Dmitri Naumov
geboren am 13. November 1981 in Irkutsk/Rußland

Dresden, 1. August 2014

Eingereicht am: 1. August 2014
Tag der Verteidigung: 30. März 2015

Gutachter:
Prof. Dr. Olaf Kolditz: Technische Universität Dresden
Prof. Dr. Natalya Kizilova: V.N. Karazin Kharkiv National University
Prof. Dr. Haibing Shao: Technische Universität Bergakademie Freiberg

Erklärung des Promovenden

Die Übereinstimmung dieses Exemplars mit dem Original der Dissertation zum Thema:

Computational Fluid Dynamics in Unconsolidated Sediments:
Model Generation and Discrete Flow Simulations

wird hiermit bestätitgt.

Ort, Datum Unterschrift (Vorname, Name)

Contents

1. *Introduction* 1
  1. Abstract 1
  2. Thesis structure 1
  3. Results 2

I. Theory

2. *Equations of fluid flow in pore-space of porous media* 6
  1. The Navier-Stokes Equations 6

3. *Fluid flow equations in homogenized porous media* 8

II. Numerics

4. *Finite Volume Method* 12
  1. Domain discretization 12
  2. Spatial discretization 12
  3. Time discretization 12

5. *Numerical solution of the Navier-Stokes Equations* 13
  1. PISO algorithm 14
  2. OpenFOAM 16

6. *Finite Element Method* 19

III. Simulations

7. *Porous media* 22
  1. General properties 22
  2. Examples of artificial porous media 24
  3. Summary 28

8. *Flow-through simulation setup* 28
  1. Flow-through laboratory experiments 29
  2. Summary 32

9. *Row of spheres in square duct* 32
  1. Setup description 33
  2. Meshing 33

3. Evaluation 35
4. Summary 41

10. *Face-centred sphere packing* 41
1. Setup of the face-centred sphere packing problem 42
2. Influence of the outlet length 43
3. Discretization error analysis 45
4. Boundary layer thickness 49
5. Summary 50

11. *Artificial sediments in a pipe.* 51
1. Setup description 52
2. Mesh preparation 52
3. Tetrahedral mesh generation from polyhedrons 56
4. An a-posteriori error driven mesh refinement 57
5. Evaluation 57
6. Summary 59

12. *Structure and development of OGS6 FEM code.* 59
1. Shortcomings of ogs5 60
2. Localization of local assembler data 60
3. Comparison of elliptic pde global matrix assembly routines 60
4. Benchmarking matrix vector operations for FEM 62
5. Summary 64

13. *Coupling to OpenFOAM* 64

IV. Visualization

14. *Overview* 72

15. *Visualization techniques* 72

16. *3D virtual reality laboratory* 74

17. *Two-sided material rendering technique* 75
1. Abstract 76
2. Introduction 76
3. Software and data processing 77
4. Two-sided material rendering 78
5. Photorealistic rendering 80
6. Conclusions 81
7. Future work 83

V. Outlook

18. *Porous media generation* 88

19. *Sphere row in square duct* 88

20. *Face-centred sphere packing* 88

21. *Artificial sediments in a pipe* 89

22. *OpenGeoSys and OpenFOAM coupling* 89

23. *Visualization* 90

VI. Appendix

References 92

Theses

The main achievements of this work are gathered in the following eight points:

- An artificial (*in-silico*) unconsolidated sediment generator was implemented.
- Numerical solutions of the Navier-Stokes Equations using DNS method with OpenFOAM software were performed.
- The validity of the numerical solutions was analysed for different mesh generation methods and mesh resolutions.
- A reduction of the computational efforts by removing the inlet and outlet parts of a duct replacing them with appropriate boundary conditions. The validity of the solutions was also analysed.
- We analysed the pressure drop and velocity distributions in simple cubic and face-centred cubic sphere packings.
- A coupling scheme of OpenFOAM simulations to the OpenGeoSys FEM code is described.
- An analysis of the current version 5 of OpenGeoSys was analysed for performance issues. An efficient implementation of the critical FEM's part—the local assembler—was written using a data locality concept.
- For the visualisation of the fluid flow results on pore-scale a two-sided material visualization technique was applied in interactive environment and still-image rendering.

Thesen

Die wesentlichen Resultate dieser Arbeit sind in folgenden acht Punkten zusammengefasst:

- Ein Programm zur Erstellung künstlicher (*in-silico*), nicht konsolidierter Sedimente wurde implementiert.
- Numerische Lösungen der Navier-Stokes Gleichungen mit der OpenFOAM Programm wurden ausgeführt.
- Die numerischen Lösungen wurden auf Validität für unterschiedliche Gittererstellungsmethoden und -auflösungen untersucht.
- Der Rechenaufwand wurde durch eine Verkleinerung des Rechengebietes (Reduktion des Ein- und Auslasses) und Modifikation der entsprechenden Randbedingungen reduziert. Diese Lösungen wurden ebenso auf Validität untersucht.
- Der Druckabfall und die Geschwindigkeitsverteilungen in einfachen kubischen und dichtesten Kugelpackungen wurden untersucht.
- Eine Kopplungsschema zwischen den OpenFOAM Lösungen und dem OpenGeoSys FEM Programm wurde beschrieben.
- Eine Performanzanalyse der aktuallen Version 5 des OpenGeoSys Programms wurde durchgeführt und eine effiziente Implementierung der zeitkritischen FEM Programmteile, unter anderem des lokalen Assemblers, wurde programmiert.
- Für Visualisierungen in einer interaktiven Umgebung der Fluidflusse auf der Porenskala wurde eine Visualisierungstechnik angewandt, bei welcher Materialien von beiden Seiten unterschiedlich dargestellt werden.

Zusammenfassung

Numerische Lösung der Navier-Stokes Gleichungen sind in letzten Dekaden mit immer zugänglicheren und leistungsstärkeren Komputerresourcen populärer geworden. Simulationen in rekonstruierten oder künstlich-erzeugten Porenraumgeometrien werden oft ausgeführt um Einsichten in mikroskopische Fluidflußstrukturen oder um Eigenschaften homogenisierter Medien, wie hydraulische Leitfähigkeit, zu erhalten. Eine physikalisch adequate Darstellung der Porenskalenflüsse ist erst mit Analyse grösserer Gebiete möglich.

Wir lösen die inkomperssiblen Navier-Stokes Gleichungen in künstlichen geordneten und zufälligen Porenraumstrukturen. Die einfach kubischen und kubisch dichtesten Kugelpackungen in einem rechteckigen Kanal werden analysiert. Für Fluidflusssimulationen in zufälligen porösen Medien werden Kugel-, Ikosahedra- und Würfelpackungen, welche nicht konsolidierte Sediemente nachbilden, mit Festkörper-Physik Simulationssoftware erzeugt. Die, sogenannte Direct Numerical Simulation (DNS) Methode, imlementiert in der quelloffenen (Computational Fluid Dynamics) CFD-Software OpenFOAM, wird für die Lösung der Navier-Stokes Gleichungen benutzt.

Der Zusammenhang zwischen der Anzahl der Kugel in den geordenten Packungen, dem Gittertyp und der Gitterauflösung wird für Strömungen bis zur Reynoldszahl 100 basierend auf den Kugeldurchmessern untersucht. Die Gittergenerierungsmethode für zufällige Medien basiert auf einer Oberflächennährungsmethode. Die resultierenden tetrahedra Gitter werden anschliessend für stationäre Strömungssimulationen verwendet; Eine Gitterverfeinerung basierend auf einem a-posteriori Fehlerschätzer wird verwendet.

Die Resultate der Strömungssimulationen werden zweirlei weiter benutzt: 1) Die daraus abgeleiteten hydro-mechanischen Eigenschaften der analysierten Medien für die grösseren meso und makro Grundwasser Simulationen. Ein Konzept für einseitige Anbindung für Simulationen auf der grossen Skala ist vorgestellt. 2) Visualisierung: Eine Renderingtechnik wurde in einer interaktiven 3D-Umgebung und auch zum Erstellen von Bildern angewandt. Mit diesem Verfahren ist es möglich einen besseren Überblick über die lokalen Strömungsstrukturen zu erhalten.

Der OpenGeoSys FEM Code für Lösung der Grundwassersimulationen auf der grossen Skala wurde auf deren Effizienz untersucht. Die Resultate dieser Analyse bildeten eine Basis für die Implementierung der neuen Version des Programms—ogs6. Die Verbesserungen bestehen unter anderem in einem Vergleich der Linearen Algebra numerischen Bibliotheken und einer Speicherzugriffseffizienten Implementierung des FEM lokalen Assembler Teils.

Abstract

Numerical solutions of the Navier-Stokes Equations became more popular in recent decades with increasingly accessible and powerful computational resources. Simulations in reconstructed or artificial pore geometries are often performed to gain insight into microscopic fluid flow structures or are used for upscaling quantities of interest, like hydraulic conductivity. A physically adequate representation of pore-scale flow fields requires analysis of large domains.

We solve the incompressible Navier-Stokes Equations in artificial ordered and random pore-space structures. A simple cubic and face-centred packings of spheres placed in a square duct are analysed. For the fluid flow simulations of random media, packings of spheres, icosahedra, and cubes forming unconsolidated sediments are generated using a rigid body simulation software. The Direct Numerical Simulation method is used for the solution of the Navier-Stokes Equations implemented in the open-source computational fluid dynamics software OpenFOAM.

The influence of the number of spheres in ordered packings, the mesh type, and the mesh resolution is investigated for fluid flow up to Reynolds numbers of 100 based on the spheres' diameter. The random media mesh generation method relies on approximate surface reconstruction. The resulting tetrahedral meshes are then used for steady-state simulations and refined based on an a-posteriori error estimator.

The fluid flow simulation results can further be used twofold: 1) They provide homogenized hydro-mechanical properties of the analysed medium for the larger meso and macro groundwater flow simulations. A concept of one-way binding for large-scale simulations is presented. 2) Visualisation: A post-processing image rendering technique was employed in interactive and still image visualisation environments allowing better overview over local fluid flow structures.

The OpenGeoSys FEM code for the solution of large-scale groundwater processes was inspected for computational efficiency. The conclusions drawn from this analysis formed the basis for the implementation of the new version of the code—ogs6. The improvements include comparison of linear algebra software realisations and an implementation of optimized memory access patterns in FEM-local assembler part.

"If a cat were to disappear in Pasadena and at the same time appear in Erice, that would be an example of global conservation of cats. This is not the way cats are conserved. Cats or charge or baryons are conserved in a much more continuous way."—Feynman at summer school in Erice in Italy, 1964.

1. Introduction

1.1 Abstract

Numerical solutions of the Navier-Stokes Equations became more popular in recent decades with increasingly accessible and powerful computational resources. Simulations in reconstructed or artificial pore geometries are often performed to gain insight into microscopic fluid flow structures or are used for upscaling quantities of interest, like hydraulic conductivity. A physically adequate representation of pore-scale flow fields requires analysis of large domains.

We solve the incompressible Navier-Stokes Equations in artificial ordered and random pore-space structures. A simple cubic and face-centred packings of spheres placed in a square duct are analysed. For the fluid flow simulations of random media, packings of spheres, icosahedra, and cubes forming unconsolidated sediments are generated using a rigid body simulation software. The Direct Numerical Simulation method is used for the solution of the Navier-Stokes Equations implemented in the open-source computational fluid dynamics software OpenFOAM.

The influence of the number of spheres in ordered packings, the mesh type, and the mesh resolution is investigated for fluid flow up to Reynolds numbers of 100 based on the spheres' diameter. The random media mesh generation method relies on approximate surface reconstruction. The resulting tetrahedral meshes are then used for steady-state simulations and refined based on an a-posteriori error estimator.

The fluid flow simulation results can further be used twofold: 1) They provide homogenized hydro-mechanical properties of the analysed medium for the larger meso and macro groundwater flow simulations. A concept of one-way binding for large-scale simulations is presented. 2) Visualisation: A post-processing image rendering technique was employed in interactive and still image visualisation environments allowing better overview over local fluid flow structures.

The OpenGeoSys FEM code for the solution of large-scale groundwater processes was inspected for computational efficiency. The conclusions drawn from this analysis formed the basis for the implementation of the new version of the code—ogs6. The improvements include comparison of linear algebra software realisations and an implementation of optimized memory access patterns in FEM-local assembler part.

1.2 Thesis structure

Main objective of this work is to develop reliable framework for analysis of hydraulical properties of porous media using computer simulations on pore-scale. The main objective can be divided into four independent parts, which reflect the structure of the thesis.

Generation of ordered and random porous media Generation of artificial porous media in computer is discribed in the first part. Ordered simple cubic and face-centred sphere packing are generated. For random structures a simulator for generation of unconsolidated sediments using rigid body dynamics is developed. The unconsolidated porous media are then quantified in terms of their statistical properties.

Direct Numerical Simulations in porespace Numerical solutions of the Navier-Stokes Equations in small domains are possible without any further assumptions for a turbulence model. We run Direct Numerical Simulations (DNS) using existing open-source computational fluid dynamics (CFD) software and verify the results for ordered sphere packings placed in a square duct. The simulations are then extended to simple artificial porespace geometries representing unconsolidated sediments consisting of spheres, or icosahedra, or cubes. The mesh generation part is analysed with respect to different mesh types and resolutions, as well as adaptive tetrahedral mesh refinement based on an *a-posteriori* error estimator.

FEM method implementation and coupling The results of the fluid flow simulations on pore-scale providing hydraulical properties of the homogenized medium are used in large scale groundwater simulations. The coupling scheme is described.

The software used for the solution of elliptic problem is based on the OpenGeoSys5 code. This code was analysed for performance and maintinance problems and an improved concept was implemented in the next version of the software—ogs6. Analysis of memory-access patterns together with data locality concept, and a comparison of linear algebra implementations contributed to higher efficiency of the FEM code.

Visualisation technique For the visualisation of fluid flows on pore-scale an image rendering technique is presented; It is used for both the interactive environment and still image rendering. A detailed description of possible visualisations is discussed. The presented two-sided material rendering technique allows for good overview of the local fluid flow structures while maintaining spatial reference to the surrounding sediment's grains.

1.3 Results

The main achievements of this work are gathered in the following eight points:

- An artificial (*in-silico*) unconsolidated sediment generator was implemented.
- Numerical solutions of the Navier-Stokes Equations using DNS method with OpenFOAM software were performed.
- The validity of the numerical solutions was analysed for different mesh generation methods and mesh resolutions.

- A reduction of the computational efforts by removing the inlet and outlet parts of a duct replacing them with appropriate boundary conditions. The validity of the solutions was also analysed.
- We analysed the pressure drop and velocity distributions in simple cubic and face-centred cubic sphere packings.
- A coupling scheme of OpenFOAM simulations to the OpenGeoSys FEM code is described.
- An analysis of the current version 5 of OpenGeoSys was analysed for performance issues. An efficient implementation of the critical FEM's part—the local assembler—was written using a data locality concept.
- For the visualisation of the fluid flow results on pore-scale a two-sided material visualization technique was applied in interactive environment and still-image rendering.

Part one

Theory

2. Equations of fluid flow in pore-space of porous media

2.1 The Navier-Stokes Equations

Equations in this chapter are based on the book “A Mathematical Introduction to Fluid Mechanics” by Alexandre J. Chorin and Jerrold E. Marsden [CM00].

For a general introduction to fluid dynamics see for example [Bat99, LLSR66].

The groundwater flow in porous media can be described by the Navier-Stokes Equations for incompressible, Newtonian fluids. The incompressibility condition is not restricting the applicability of the simulations because the fluid velocities in groundwater flows are low and there are any shocks or other density-variation dependent effects. Density variations could be expected solely in the immediate bore hole region or any place with very large pressure gradients, or when dealing with less viscous fluids, for example gases.

The incompressible Navier-Stokes Equations The Navier-Stokes Equations in their general form are given by the continuity equation

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

and the momentum equation

$$\rho\left(\frac{\partial u}{\partial t} + u \cdot \nabla u\right) = -\nabla p + \nabla \cdot \mathbb{T} + f,$$

where ρ is fluid’s density, u is fluid’s velocity, p—pressure, $\mathbb{T}$ is the deviatoric stress tensor, and f are the body forces.

Assuming the fluid to be incompressible, the continuity equation becomes

$$\nabla \cdot u = 0,$$

which already greatly simplifies the general form for the numerical analysis. We further assume the fluid being an isotropic, Newtonian fluid. The total stress tensor $\sigma := -\mathbb{I}p + \mathbb{T}$ simplifies to

$$\sigma = -\mathbb{I}p + \mu\frac{1}{2}\big(\nabla u + (\nabla u)^t\big),$$

where μ is (constant) dynamic viscosity. The momentum equation becomes:

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = \nabla \cdot \sigma = -\frac{\nabla p}{\rho_0} + \frac{\mu}{\rho_0}\Delta u + f, \tag{1}$$

where ρ_0 is constant fluid’s density.

Non-dimensionalization and scaling Introducing characteristic quantities: U—velocity, L—length-scale, and T—time, equal to L/U, we get partially non-dimensionalized momentum equation from Navier-Stokes Equations (1) with starred quantities and all differential operators being dimensionless:

$$\frac{\partial u^\star}{\partial t^\star} + (u^\star \cdot \nabla)u^\star = -\Big(\frac{1}{U^2\rho_0}\Big)\nabla p + \Big(\frac{\mu}{U\rho_0 L}\Big)\Delta u^\star + \Big(\frac{L}{U^2}\Big)f.$$

In the context of groundwater flow application, the body forces f can be set to the gravitational force g, and the term $(L/U^2)f$ becomes $1/\mathrm{Fr}$, where $\mathrm{Fr} := U^2/(gL)$ is the Froude number. The expression $\mu/(U\rho_0 L)$ equals to $1/\mathrm{Re}$, Re being the Reynolds number. The remaining quantity requiring non-dimensionalization is the pressure p.

In the following equations all variables are in the dimensionless form, and we drop the * symbol everywhere.

There are two possibilities for the pressure: one is applicable to high velocity flows and introduces characteristic pressure $P := \rho_0 U^2$—essentially the specific kinetic energy, or the dynamic pressure, yielding

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\nabla p + \Big(\frac{1}{\mathrm{Re}}\Big)\Delta u + \Big(\frac{1}{\mathrm{Fr}}\Big)f;$$

But then the Reynolds number is also large, and we can drop the $1/\mathrm{Re}\ \Delta u$ term, and also the gravitational forces for the same reason. The resulting equation is the momentum equation in the Euler equations for an inviscid flow.

For the fluid flow in porous media another characteristic pressure (also known as the viscous pressure scale) is more suitable:

$$P := U\mu/L.$$

The fully dimensionless momentum equation reads then

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = \Big(\frac{1}{\mathrm{Re}}\Big)(-\nabla p + \Delta u) + \Big(\frac{1}{\mathrm{Fr}}\Big)f.$$

In the absence of body forces (which we assume in the sequel because the main interest lies in dynamic and not static quantities) the Navier-Stokes Equations for an incompressible, Newtonian fluid reads

$$\begin{aligned} &\nabla\cdot u = 0, \\ &\frac{\partial u}{\partial t} + (u \cdot \nabla)u = \frac{1}{\mathrm{Re}}(-\nabla p + \Delta u). \end{aligned} \tag{2}$$

In the above we silently assumed the viscosity being a constant, which holds only for isothermal processes. Given low Reynolds numbers of the processes being studied, heat generation due to friction can be safely ignored. No further equations are required but for boundary conditions.

Boundary conditions Setting boundary conditions for the pressure and the velocity is necessary. The simplest boundary condition is called “no-slip” boundary condition and is applied to static surfaces *e.g.* duct's walls or grain's surfaces. It forces the velocity to be zero at the surface, and the pressure gradient normal to that surface:

$$u = 0 \quad \text{and} \quad \frac{\partial p}{\partial \mathbf{n}} = 0.$$

Other two boundary conditions widely used are the inlet and the outlet boundary conditions. At the inlet a velocity vector is given, and the pressure gradient normal to the inlet is again zero:

$$u = u_{\text{inlet}} \quad \text{and} \quad \frac{\partial p}{\partial \mathbf{n}} = 0.$$

At the outlet a pressure is set to a given value (usually zero, when only relative pressure is of interest), and derivatives of the velocity vector components normal to the outlet are forced to be zero.

$$p = p_{\text{outlet}} \quad \text{and} \quad \frac{\partial u_i}{\partial \mathbf{n}} = 0 \quad \text{for} \quad i \in x, y, z.$$

Initial conditions Good choice of the initial conditions for velocity (and pressure[a]) can greatly improve convergence of the numerical system to a valid state. When nothing is known about them for a particular geometry initialization of the velocity field from solution of the potential flow provides a good starting point. Reusing solutions on same geometry but for different Reynolds number is also a good option. Another possibility is to reuse a solution computed on a coarser mesh, provided there is such solution.

The Stokes equation

Applying scale analysis to the incompressible Navier-Stokes Equations for isothermal, Newtonian fluid, equation (2), yields the Stokes equation under the assumption of low Reynolds number. In the momentum equation the transport term and the inertial forces can be neglected leaving only the right-hand-side of the second equation in (2):

$$\begin{aligned} \nabla \cdot u &= 0, \\ \nabla p &= \Delta u. \end{aligned} \tag{3}$$

---

[a] Given a velocity field the pressure field can be recovered as in PISO numerical method in Section PISO algorithm on page 14.

3. Fluid flow equations in homogenized porous media

The fluid flow in the pore space is described by the Navier-Stokes Equations in its most general form. For low Reynolds numbers the Navier-Stokes Equations are reduced to the Stokes Equations. But for large scale simulations (deep geothermal reservoir [BZM+10], CO_2-storage [KKC+10], flow in Thuringian basin [RFSK13]) with scales of interest ranging from meter to hundreds of kilometers, simulations on pore scale resolution are not longer feasible.

For such large scale simulations a spatially averaged fluid flow description was experimentally determined by Darcy [Dar56] and derived later from the Navier-Stokes or the Stokes equations by a homogenization process described for example in [Tar80], [Mas02], and recent lecture notes by Ian Tice [Tic14] and citations therein. An overview of different derivations of Darcy's law are given in [Bea88, pp. 161–176] including capillary tube models (bundle of tubes or network models), fissure models (a system of fractures of given width), hydraulic radius models (Poisseuille's equation, leading to Kozeny-Carman equations [Car56]), resistance to flow models (Stokes' equation for drag of spherical particle), statistical models (tracer particle movement in disordered medium), and the averaging of the Navier-Stokes Equations.

Darcy's law

"In 1856, Henry Darcy investigated the flow of water in vertical homogeneous sand filters in connection with the fountains of the city of Dijon, France. From his experiments, Darcy concluded that the rate of flow (volume per unit time) Q is (a) proportional to the constant cross-sectional area A, (b) proportional to ΔH and (c) inversely to the length L."[Bea88, p. 119 f.] This observations are combined into the well known Darcy's law [Dar56]:

$$Q = KA\Delta H/L,$$

where K is the hydraulic conductivity coefficient measured in units of speed, $[K] = m/s$. A schematics of an experiment are shown in the following figure.

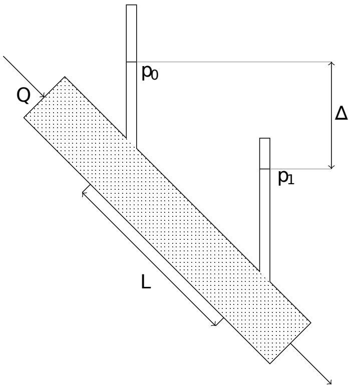


Figure 3.1 Seepage through an inclined sand filter. (Figure adopted from [Bea88, p. 120].)

The hydraulic conductivity coefficient K depends on both, the fluid properties and the geometrical properties of the porous medium. When deriving the Darcy's law by averaging of the Navier-Stokes Equations one can express the hydraulical conductivity coefficient as $K = k\rho g/\mu$, where k is the permeability (or intrinsic permeability), ρ—fluid's density, g—gravity acceleration and μ—the fluid's dynamic viscosity. The Darcy's law takes then the following form [Bea88, p. 133]:

$$q_i = -(k\rho g/\mu)\frac{\partial(p/\rho g + z)}{\partial x_i}, \quad i = 1, 2, 3, \tag{4}$$

where $\mathbf{q} = Q/A$, p is the hydrostatic pressure, and z is the direction of the gravitational force. In this form, the permeability k is a coefficient depending on porous media structure only and not on the fluid's properties.

Extensions to Darcy's law Higher permeabilities (of unconsolidated sands for example) or higher velocities (near bore wells) invalidate creeping flow regime assumption used in the derivation of Darcy's law from the Navier-Stokes Equations. Some of the fluid's energy is used to overcome resistance of eddies formed in the transition to turbulent flow regime.

One of the extensions to Darcy's law is the Forchheimer term added to compensate for the non-linear behaviour when inertial terms in the Navier-Stokes Equations become significant. For theoretical development see for example [Whi96].

Another extension is the Brinkman term—a "Stokes-type" term—added to the Darcy's law equation. [Bri49] "The term is for matching flow velocity boundary conditions at a free liquid:porous medium boundary" [MBG94] used, for example, in simulation with multiple porosities.

Part two

Numerics

4. Finite Volume Method

4.1 Domain discretization

Discretization of a domain—the pore space—into finite set of control volumes, which forms a computational domain, is done by subdivision of the input domain. The control volumes must fulfil some requirements on their shape. All of the control volumes are filling the computational domain and do not overlap. The flat faces are shared between two control volumes at most—these are the internal faces; Those faces not shared by two control volumes are the boundary faces.

Meshes for simple input geometries can be described by a structured, cartesian mesh. This type of meshes is simple to generate and can be the first step in the construction of more complex, unstructured meshes. The OpenFOAM's utility `blockMesh`, for example, provides such functionality.

Not simply connected geometries including holes need either a tetrahedral mesh generator (*e.g.* TetGen [Si13]) or modification of a coarse structured mesh (*e.g.* OpenFOAM's approach with `snappyHexMesh` tool [OFb]) generated before. A mixed element meshes or meshes consisting of arbitrary polyhedra conforming to the above restrictions can also be used.

4.2 Spatial discretization

Given a general conservation law in differential form

$$\frac{d}{dt}q(x,t) - \nabla\cdot f(q(x,t)) = 0, \tag{5}$$

the integral over a cell C_i (a finite volume) is

$$\frac{d}{dt}\int_{C_i} q(x,t)\;dx - \int_{C_i} \nabla\cdot f(q(x,t))\;dx = 0.$$

This equation is required to be fulfilled by the FVM for all cells of the domain.

The right-hand-side can be transformed into a sum of integrals over its surface (the faces of the cell):

$$\begin{aligned}\int_{C_i} \nabla\cdot f(q(x,t))\;dx &= \oint_{\partial C_i} f(q(x,t))\cdot dS \\ &= \sum_{f\in\text{faces}(C_i)} \int_{S_f} f(q(x,t))\cdot dS,\end{aligned}$$

The FVM is constructed in such way, that the numerical scheme is conservative, *i.e.* the solution will change only when the boundary conditions change. The total mass within the discretized domain is preserved.

The above equation (5) may include a convection or diffusion terms where $\nabla\cdot f(q(x,t)) = \nabla\cdot(u(x,t)q(x,t))$ with u being a velocity or $\nabla\cdot f(q(x,t)) = \nabla\cdot(\alpha(x,t)\nabla q(x,t))$ with α being a diffusion coefficient. Source terms can be directly incorporated onto the right-hand-side and also need an integration over every cell.

4.3 Time discretization

After the spatial discretization of the equations by FVM we have an ODE in the following general form:

$$\frac{dQ_i(t)}{dt} = F(Q_i, x_i, t, \frac{\partial Q_i(t)}{dx}, \ldots),$$

where $Q_i(t)$ is the value in node i at position x_i.

The simple time discretization methods include the forward Euler, the backward Euler, and the Crank-Nicolson methods, where the first two are of first order, and the latter of second order (for blending coefficient $1/2$). Of the three methods, the forward Euler discretization is an explicit method; The next timestep is given in terms of current time step only:

$$Q_i^{n+1} = Q_i^n + \Delta t\, F^n(Q_i, x_i, t, \frac{\partial Q_i(t)}{dx}, \ldots),$$

and $F^n()$ denotes an approximation of $F()$ for time $t = t_n$.

The other two methods are implicit and involve solution of possibly non-linear system of equations. The backward Euler method is given by:

$$Q_i^{n+1} - \Delta t\, F^{n+1}(Q_i, x_i, t, \frac{\partial Q_i(t)}{dx}, \ldots) = Q_i^n.$$

Both Euler methods are of first order, while the Crank-Nicolson method (for blending coefficient $\beta = 1/2$) is of second order. It is given by:

$$\begin{aligned} Q_i^{n+1} - \Delta t\Big[\beta\, F^{n+1}(Q_i, x_i, t, \ldots)\Big] &= \\ &= Q_i^n + \Delta t\Big[(1-\beta)\, F^n(Q_i, x_i, t, \ldots)\Big]. \end{aligned}$$

The time discretization choice depends not only on the accuracy of the method, but also on spatial discretization (the mesh size, leading to the Courant number, see p. 15) and the properties of the whole discretized system (dealing with stability of the results).

5. Numerical solution of the Navier-Stokes Equations

For the numerical solution of the Navier-Stokes Equations we use the OpenFOAM CFD software, detailed description of which is given in the Section OpenFOAM on p. 16. The Navier-Stokes Equations used in the discretization procedure are in the form:

$$\nabla \cdot u = 0,$$
$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\nabla p + \nu \Delta u,$$

where the pressure p is the kinematic pressure,[a] and ν is the kinematic viscosity. [Jas96, p. 68f] The Reynolds number is given by $\mathrm{Re} = UL/\nu$, where U and L are characteristic velocity and length scales.

5.1 PISO algorithm

Pressure Implicit with Splitting of Operators (PISO) method is employed for the solution of time dependent Navier-Stokes Equations. A derivation of this method developed originally by [Iss86] is given for example in [FP02, Jas96]. We repeat the main steps here.

The coupling between the velocity components and the pressure gradient is given in the momentum equation. The continuity equation does not contain a pressure term to calculate the pressure, so usually both equations are combined by taking the divergence of the momentum equation, and simplify it by means of the continuity equation:

$$\nabla \cdot (\nabla p) = -\nabla \cdot ((u \cdot \nabla)u),$$

yielding the pressure equation. Because the numerical approximations of the divergence and the gradients must be the same as in the discretization of the momentum equation, it is better to start the derivation of the discretized pressure equation from the already discretized momentum and continuity equations.

Discretized Navier-Stokes Equations Let $a_P^u u_P$ be a discretization of velocity at node P, and $\sum_N a_N^u u_N$ the contributions from the neighbor nodes N. The superscript u indicates dependency of the operators a_i^u on the velocity field. We write the discretized momentum equation keeping the pressure gradient in its original form as:

$$a_P^u u_P + \sum_N a_N^u u_N = \frac{u^{\mathrm{old}}}{\Delta t} + q - \nabla p.$$

For simplicity we introduce an operator

$$H(u) = -\sum_N a_N^u u_N + \frac{u^{\mathrm{old}}}{\Delta t} + q,$$

---

[a] "Note that the absolute pressure is of no significance in an incompressible flow; only the gradient of the pressure (pressure difference) affects the flow." [FP02, p. 167]

containing all source terms, the contributions from the neighbor nodes, and the transport part, but not the pressure gradient:

$$a_P^u u_P = H(u) - \nabla p \tag{6}$$

Formally we write the solution of the above equation as

$$u_P = \frac{H(u)}{a_P^u} - \frac{\nabla p}{a_P^u}, \tag{7}$$

noting that u_P does not satisfy the continuity equation for now. The next step is to correct the pressure field to fulfil the continuity equation. Inserting the solution (7) into discretized continuity equation yields

$$\nabla\cdot\left(\frac{1}{a_P^u}\nabla p\right) = \nabla\cdot\left(\frac{H(u)}{a_P^u}\right). \tag{8}$$

The equations (6) and (8) form the discretized Navier-Stokes Equations for incompressible fluid, now with all differential operators in consistent, discretized form (denoted by the ˜ symbol):

$$a_P^u u_P = H(u) - \tilde{\nabla} p \qquad \text{—momentum predictor,} \tag{9}$$

$$\tilde{\nabla}\cdot\left(\frac{1}{a_P^u}\tilde{\nabla} p\right) = \tilde{\nabla}\cdot\left(\frac{H(u)}{a_P^u}\right) \qquad \text{—pressure equation.} \tag{10}$$

Pressure-velocity coupling The pressure-velocity coupling can be done in two ways: A simultaneous solution of both equations, which require additional memory space for the block-coupling matrices, or in a sequence, which leads to the family of PISO algorithms. There are several variations on the PISO algorithm proposed by [Iss86], which include the SIMPLE algorithm proposed by [Pat80] for steady-state equations.

The PISO algorithm can be outlined in three main steps:

- Using some guess on the pressure field (preferentially from the previous time step) solve the momentum equation (9). This gives an approximation to the new velocity field.
- Formulate the pressure equation (10) using the predicted velocity field, and solve to obtain a corrected pressure field.
- With the new pressure field correct the velocity field using equation (7).

After the third step the pressure equation must be solved again. This procedure is repeated “until a velocity field which satisfies both the momentum and continuity equations is obtained.” [FP02, p. 175]

Timestepping To achieve accuracy required for direct numerical simulation the Courant-Friedrichs-Lewy (CFL) condition [CLF28] must be satisfied: $\Delta x/\Delta t \leq 1$. And to reduce the numerical errors in time stepping the CFL condition is set to $1/5$.

We are using second order spatial central differencing scheme and explicit Euler time-stepping schemes.

Although most of the performed simulations are steady-state, a transient solution is computed. Solutions of the Navier-Stokes

Equations in complex geometries can develop unsteady behaviour although in another similar geometry with the same Reynolds number the solution would converge to a steady-state. Transient behaviour can be easily recognized when observing initial residuals of the momentum predictor and pressure correction systems of linear equations.

In the PISO inner loop 7 correction steps were used; No non-orthogonal correction is performed for the structured cartesian meshes which are orthogonal—the mesh lines intersect at right angles. In the case of non-orthogonal, tetrahedral meshes (generated with TetGen [Si13]) there are, usually up to four non-orthogonal corrections steps depending on the mesh quality.

5.2 OpenFOAM

"OpenFOAM (Field Operation And Manipulation) is a free, open source CFD software package developed by OpenCFD Ltd at ESI Group and distributed by the OpenFOAM Foundation" [OFb].

Currently used version is 2.1.0 and publicly available source code allow changes to the solvers giving great flexibility over the solution algorithm.

OpenFOAM cases Every simulation—an OpenFOAM case—is organized in simple directory structure. For single threaded simulations the structure is the following one:

```
case
├─ 0 .................................. Time step with initial conditions or solutions.
│  ├─ U
│  └─ p
├─ {0.1, 0.2, ...} .................... Further time steps.
│  ├─ U
│  └─ p
├─ constant ........................... Constant simulation properties like turbulence
│  ├─ transportProperties                model properties or the mesh.
│  ├─ turbulenceProperties
│  ├─ RASProperties
│  ├─ dynamicMeshDict
│  └─ polyMesh ........................ The mesh consists of the boundary descrip-
│     ├─ blockMeshDict                   tion, the cells (and their connections), and
│     ├─ boundary                        the points.
│     ├─ faces
│     ├─ neighbour
│     ├─ owner
│     └─ points
└─ system ............................. Configuration of the simulation, time-stepping,
   ├─ controlDict                        IO, linear solvers, discretization schemes.
   ├─ fvSchemes
   └─ fvSolution
```

The subsequent simulation results are stored in directories corresponding to the performed time step like `0.1`, `0.2` *etc.* in the case's root directory.

The mesh files are placed into the `constant` directory but solvers which are modifying a mesh will write a copy into the current time-step directory.[b] In this work we run simulations only on static meshes.

All configuration files are read into dictionaries—a possibly nested key-value structures. Given such representation of a simulation setup allows simple access of a special configuration setting within the solver's code.

Running simulation After a case has been prepared, starting the simulation for example using `icoFoam` solver—a transient solver for incompressible Newtonian fluid flow—is done by calling `icoFoam -case path-to-case`, or when running in parallel `mpirun -np number-processors icoFoam -case path-to-case -parallel`.

The simulation progress is written to the standard output and contains information about current time step, and especially the output of linear equation system solvers:

```
> icoFoam -case path-to-case
...
Time = 14.8504
Courant Number mean: 0.07833 max: 0.19738 deltaT = 0.00086
DILUPBiCG:  Solving for Ux, Initial residual = 0.00155, Final residual = 9.96858e-07,
                                                                                No Iterations 2

DILUPBiCG:  Solving for Uy, Initial residual = 0.00049, Final residual = 2.12377e-07,
                                                                                No Iterations 2

DILUPBiCG:  Solving for Uz, Initial residual = 0.00165, Final residual = 9.47528e-07,
                                                                                No Iterations 2

GAMG:  Solving for p, Initial residual = 0.00267, Final residual = 1.91465e-06,
                                                                               No Iterations 13

time step continuity errors : sum local = 2.47687e-10, global = 8.50541e-15,
                                                                       cumulative = 1.78686e-09

GAMG:  Solving for p, Initial residual = 0.00092, Final residual = 6.63471e-07,
                                                                               No Iterations 16

time step continuity errors : sum local = 8.58776e-11, global = 2.82134e-14,
                                                                       cumulative = 1.78689e-09
...
GAMG:  Solving for p, Initial residual = 8.60021e-08, Final residual = 6.99740e-09,
                                                                                No Iterations 6

time step continuity errors : sum local = 9.05251e-13, global = -1.65606e-15,
                                                                       cumulative = 1.78686e-09

ExecutionTime = 86287.95 s  ClockTime = 86379 s
```

---

[b] Solvers changing the mesh are used for fluid-structure interaction, for example.

Output of linear equation solvers for velocity components and seven (four are not shown above) for pressure correction loop can be parsed for monitoring of running simulation and graphical representation.

The `foamLog` utility provides parsing of such log-files such that these can be processed numerically or graphically. An example in the Figure 5.1 shows convergence of the initial residuals norms of the linear solvers in momentum predictor and (first iteration) pressure correction loop during a steady-state simulation. In the following Figure 5.2 plots of a transient simulation clearly differ from the steady state simulation.

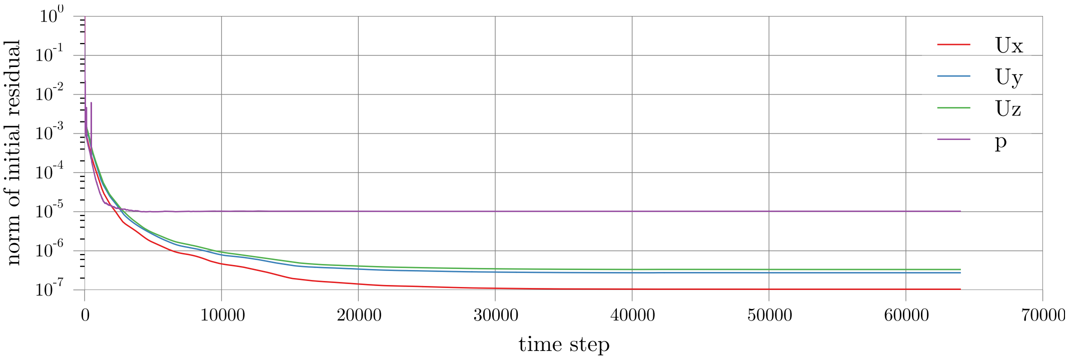


Figure 5.1 Linear solver initial residuals plot of a steady-state OpenFOAM simulation. Residuals for the three velocity components and the first iteration of pressure correction loop are shown.

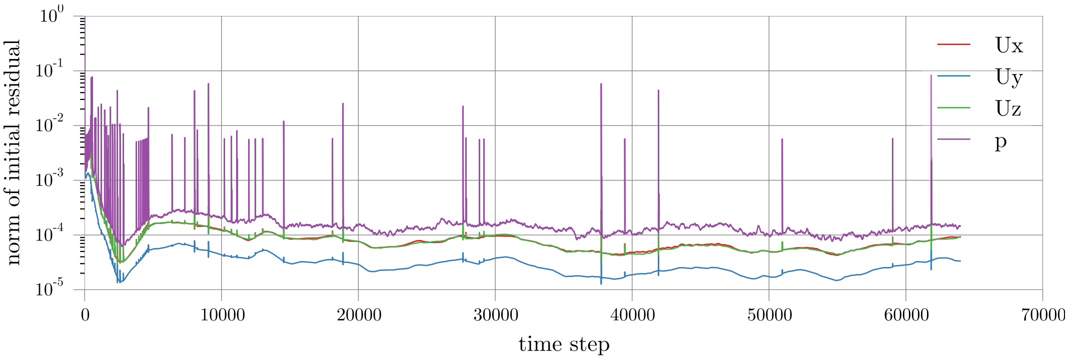


Figure 5.2 Linear solver initial residuals plot of a transient OpenFOAM simulation. Residuals for the three velocity components and the first iteration of pressure correction are shown. Visible is an initial stabilization phase up to time step 20000, and statistically stable solution until the end.

Post-processing After successful simulation the results are stored into time-step directories. There are many utilities provided by OpenFOAM for analysis. For example to determine the average inlet pressure there is a program `patchAverage`, which when called as `patchAverage -case case p inlet`, prints the average of p over the inlet patch for all time steps:

```
> patchAverage p inlet

Time = 0.4

    Reading volScalarField p
    Average of volScalarField over patch inlet[0] = 0.8752209307916339

Time = 0.5
    Reading volScalarField p
    Average of volScalarField over patch inlet[0] = 0.9334450356082965
```

Another option is to export the data into another file format or to import it directly into your favourite post-processing program; The Visualization Toolkit (VTK) [SML03] is a software system for 3D computer graphics, image processing and visualization. It has become very popular in the last decade and is available on many platforms. To export the data into the VTK format one can use an OpenFOAM utility `foamToVTK`, which exports the meshes and data as unstructured meshes, or resample the data into a structured mesh or a 3D image. This tool is freely available on github `https://github.com/endJunction/foamToVTI` repository. We try to use as many as possible of OpenFOAM's post-processing utilities, but switch to VTK formats for special post-processing.

Some aspects of the further post-processing based on VTK are discussed in the Section Visualization on p. 72.

6. Finite Element Method

The Finite Element Method is based on the variational formulation of a boundary (and initial) value partial differential equation. Consider a second order elliptic boundary value problem

$$\begin{aligned} Lu &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{11}$$

with homogeneous Dirichlet boundary conditions.

The pde (11) is multiplied with a test-function and integrated over the domain. By integration by parts the boundary conditions are included. The so-called weak formulation is then: Find $u \in V$ such that

$$a(u,v) = b(v) \quad \text{for all } v \in V,$$

where $a(.,.)$ is a symmetric positive definite, bilinear form and $b(.)$ is a linear functional given by:

$$\begin{aligned} &a : V \times V \to \mathbb{R}, && a(u,v) = \int_\Omega Lu\, v \, d\Omega \\ &b : V \to \mathbb{R}, && b(u) = \int_\Omega f\, v \, d\Omega \end{aligned}$$

The computational domain Ω is discretized onto finite number of elements where a finite dimensional space $V_h \subset V$ is defined.

The pde is transformed into a finite set of algebraic equations (or ode's in case of time-dependent problem), which are then solved. The actual solution process may include solution of the ode (time marching), Newton/Picard iterations for non-linear problem, and finally solution of (sparse) linear system of equations.

Part three

Simulations

7. Porous media

7.1 General properties

Porous media are classified by their statistical properties like porosity or grain size distribution, and depending on application their hydraulical, mechanical or electrical properties like permeability, bulk modulus or electric conductivity.

The statistical porous media properties can be quantified either by visual inspection (*e.g.* statistical analysis of two dimensional slices under microscope) or derived from measurement of hydraulical, mechanical or electrical properties assuming there is a valid relationship between the quantified property like porosity and measured property like electric conductivity.

Porosity of a porous medium strongly depends on how well the medium is sorted and consolidated. In this work we analyse only unconsolidated sediments.

For the unconsolidated sediments the porosity vary usually around 50% and strongly depends on how well the sediment's particles are sorted and compacted.

Statistical and hydraulical properties Let $V_0 \subset \mathbf{R^3}$ be the volume containing a porous medium and $V_{\text{void}} \subset V_0$ be the pore space volume. Define the microscopic porosity function $Z(x)$ for $x \in V_0$ as

$$Z(x) := \begin{cases} 1 & \text{if } x \in V_{\text{void}}, \\ 0 & \text{otherwise.} \end{cases}$$

Definition of the function $Z(x)$ allows mathematically rigorous definition of statistical properties of a porous medium.

Representative Elementary Volume In analysis of porous media we differentiate between two scales: micro-scale, which is of the same order as the grain or pore sizes, and macro-scale, where properties of porous medium are averaged over some volume, and it is assumed, that the chosen volume is representative.

A representative volume has same properties as any larger volume. The properties might be geometrical, hydraulic or mechanical or any other nature. For a volume to be a representative volume in other than geometrical properties, it must be at least geometrically representative.

Consider the average of some quantity ξ over a volume V_p at point p (*e.g.* a sphere centred at p). When V_p is very small, we expect large variation of the averaged quantity $\bar{\xi} = \frac{1}{|V_p|} \int_{V_p} \xi \, dV$ and, as the volume covers larger and larger parts of the porous medium, the variations should become smaller.

Porosity A basic property of a porous medium is porosity ϕ. The bulk porosity ϕ_{bulk} of a porous medium of volume V_0 is defined by the ratio between void volume or the pore space volume V_{void} and V_0:

$$\phi_{\text{bulk}} = \frac{V_{\text{void}}}{V_0}. \tag{12}$$

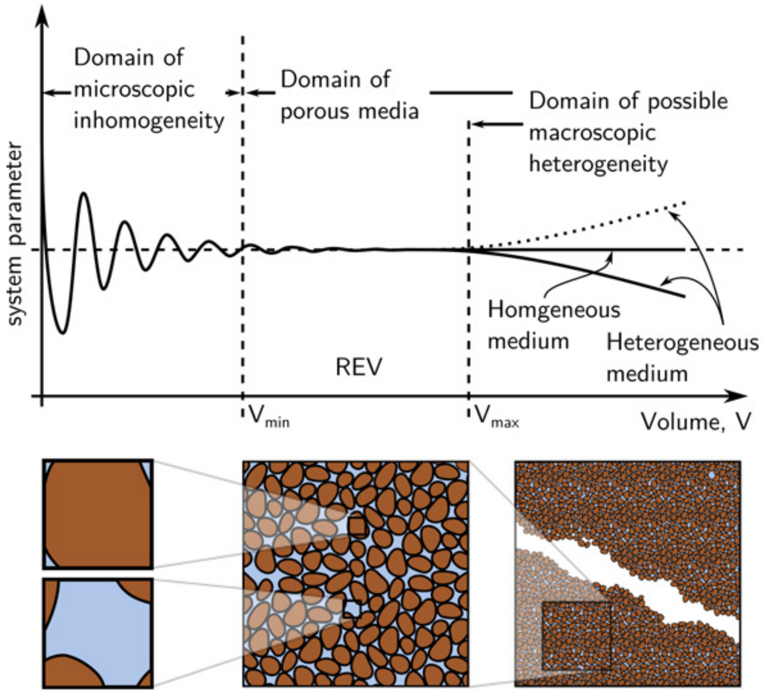


Figure 7.1 Possible definition of a REV based on bulk porosity. Variation of a system parameter and corresponding magnification levels. (Figure adopted from [Bea88, p. 30] by Norbert Böttcher with kind permission. [Böt14])

Taking in account that possibly not all of the pores will be accessible in a hydraulic experiment, effective porosity ϕ_{eff} is defined similarly to the above by using the volume of pores which contributes to the fluid flow instead of the total void volume V_{void}:

$$\phi_{\text{eff}} = \frac{V_{\text{eff}}}{V_0}. \tag{13}$$

This definition is not exact in general and depends on the porous medium and the fluid being analysed [EP88].

By averaging the microscopic porosity $Z(x)$, as described above, we can also define the bulk (or volumetric) porosity as

$$\phi_{\text{bulk}} = \frac{1}{|V_p|} \int_{V_p} Z(x)\, dV.$$

The bulk porosity is a dimensionless quantity often given in per-cent. (In the microscopic porosity function V_{void} can be replaced by V_{eff} yielding the effective porosity ϕ_{eff}.)

Following Bear[Bea88, p. 19] “we choose to define the [representative elementary volume] REV through the concept of porosity, which seems to be the basic porous matrix property.” In the following Figure 7.1 the volumetric average of the microscopic porosity is shown as function of the volume size.

We are interested to choose the REV as small as possible in order to reduce the use of computational resources in numerical simulations. Of course the minimum REV is dependent on the grain size of the underlying porous medium.

Autocorrelation Description of a porous medium using only its porosity is not sufficient for further analysis. “Information about the shape of the void region may be obtained by averaging the product $Z(r)Z(r+\rho)$ with respect to r keeping ρ fixed”[Pra61] (also cited in [Bea88, p. 43]). Although the two-point average “contains most of the parameters commonly used to characterize a porous medium, it does not of course constitute a complete description, even in a statistical sense. A fuller, but still incomplete, description is given by the three-point average” [Pra61]. Higher (than the third) order averages are usually not used.

The two-point average function is know to provide further insights in the structure of a porous medium. General introduction and in-depth analysis is given for example in [Tor01] and [Vog02].

Autocorrelation of a random process X between times s and t—the two-point average—is defined as

$$R(s,t) = \langle (X_s - \mu_s)(X_t - \mu_t) \rangle, \tag{14}$$

where μ_s, μ_t are the expected values at times s and t, respectively. Naturally for the autocorrelation function to be well defined the expected values and standard deviations must be defined too.

For a process for which the expected value μ and standard deviation σ are time-independent, *i.e.* a second order stationary process, and if the autocorrelation depends only on time difference between t and s, the autocorrelation can be written in terms of time difference $\tau = s - t$ as:

$$R(\tau) = \langle (X_{t+\tau} - \mu)(X_t - \mu) \rangle.$$

Translating this to a three-dimensional signal $Z(r)$, $r = (r_0, r_1, r_2)^t$, and define $\Delta r = (\Delta r_0, \Delta r_1, \Delta r_2)^t$ the autocorrelation function becomes

$$R(\Delta r) = \langle (Z(r+\Delta r) - \mu)(Z(r) - \mu) \rangle.$$

for anisothropic porous medium. If the medium is isotropic, the autocorrelation function depends on distance $\rho \geq 0$ only:

$$R(\rho) = \langle (Z(r + (\rho,\rho,\rho)^t) - \mu)(Z(r) - \mu) \rangle.$$

The autocorrelation function decreases from μ at $\rho = 0$ to μ^2 for $\rho \to \infty$. The negative value of the derivative of $R(\rho)$ at $\rho = 0$ corresponds to $1/4$ of the specific pore surface area; The integral $\int_0^\infty (R(\rho) - \mu^2)\, \mathrm{d}\rho$ is a characteristic length scale.

7.2 Examples of artificial porous media

Spherical packing In spherical packing the void volume is connected and the bulk porosity is equal to effective porosity; For this section define $\phi := \phi_{\text{bulk}} = \phi_{\text{eff}}$.

Simple cubic The most simple spherical packing is simple cubic packing. The defining function $Z(x)$ for spheres $B_3(r)$ with radius r in volume $[-r, r]^3$ is

$$Z(x) := \begin{cases} 1 & x \in [-r, r]^3 \setminus B_3(r), \\ 0 & x \in B_3(r). \end{cases}$$

The porosity of this packing is given by

$$\begin{aligned} \phi = \langle Z(x) \rangle &= \frac{1}{8r^3} \int_{[-r,r]^3} Z(x)\, dx \\ &= \frac{1}{8r^3} \int_{[-r,r]^3 \setminus B_3(r)} 1\, dx = \frac{(8r^3 - \frac{4\pi}{3} r^3)}{8r^3} \\ &= 1 - \frac{\pi}{6} \approx 0.476401, \end{aligned}$$

and the standard deviation by

$$\begin{aligned} \sigma^2 &= \left\langle (Z(x) - \phi)^2 \right\rangle \\ &= \frac{1}{8r^3} \int_{[-r,r]^3} (Z(x) - \phi)^2\, dx \\ &= \frac{1}{8r^3} \left(\int_{B_3(r)} \phi^2\, dx + \int_{[-r,r]^3 \setminus B_3(r)} (1 - \phi)^2\, dx \right) \\ &= \frac{\pi}{6} \phi^2 + (1 - \frac{\pi}{6})(1 - \phi)^2 \\ &= \frac{1}{108} (\pi - 9)(\pi - 6)\pi \\ &\approx 0.487113. \end{aligned}$$

Face-centred cubic More dense sherical packings are face-centred cubic and body-centred cubic packings with equal porosities of

$$\phi = 1 - \frac{\pi}{3\sqrt{2}}.$$

"In face-centred cubic packing, each layer has spheres placed diagonally next to each other. The next layer of spheres is placed in the crevices between spheres on the bottom layer. Every third layer directly overlies each other." [Gro00, GLR03]. For the generation of the packing following vectors are used to determine spheres' centres:

$$(1, 0, 0), \quad (-1/2, \sqrt{3}/2, 0), \quad (0, -1/\sqrt{3}, \sqrt{2/3}).$$

In [Gro00] a calculation of the volume occupied by the spheres is given.

Artificial porous media For generation of artificial unconsolidated sediments a software "settleDyn"—short for Settle Dynamics—is used, which is based on a simple ballistic method coupled with a rigid body dynamics.

> The "settleDyn" software was originally developed by Guido Blöcher[BZ08] with simplified physical model under the name "Settle3D". It allowed translations and rotations of a rigid body but did not take in account the velocity or linear and angular momenta. To create physically more accurate sedimentation process the new implementation is based on the "Bullet Physics Engine" [Bul]. The simulation driver, which is controlling the simulation process, is implemented in "Haskell" [Has]. "Haskell" is a pure functional programming language, which suites well for high-level abstraction of a simulation driver. The command line interface—the front-end—is described in the supplied documentation. All varieties of artificial sediments used in this thesis are generated using the "settleDyn" software.[a]

The purpose of the sedimentation simulation is to generate an unconsolidated sediment resembling a natural one with the same statistical properties like grain-size distribution, porosity, autocorrelation. A comparison can be made by analysing thin sections or the three dimensional volume (using a CT-scanner for example).

We assume the grains to be rigid. Deformations can be neglected because of the low weight and impact velocities, altogether resulting in low moments of inertia. The dynamics of body motion is described by Newton's Second and Third Laws. Besides the gravity there are drag forces acting on a falling grain—these are ignored in the simulation.[b] There are further factors involved in real processes: A very difficult one to estimate properly is the friction. Friction may influence the compactness of the resulting sediment. In the simulation there is friction between grains, but it is set to some arbitrary value.

The *grain's shape* is another parameter in the simulation, which can be controlled to a certain degree of approximation. The surface is consisting of flat triangles and this is the only restriction; Convex or concave shapes can be used.

The shapes used in this study are sphere, icosahedron, and cube. This choice is made to restrict the space of possible sediments while preserving the variation of grain-roundness. Other variations, elongation or surface roughness, for example, are not considered, but there is no limitation by the settleDyn software.

Grain size distribution is one of the most important parameters of a sediment. The grain size distribution is obtained by sieving the sediment and weighting the mass of each sieve residues. The distribution is given in mass-per-cent for a specific mesh size.

In the simulation the grain's size is defined as the second shortest edge length of the grain's axis-aligned bounding box. Another possibilities include radius of the circumsphere or the second radius

---

[a] The "settleDyn"–source code is available under the GNU Public License v.3 or later [Naub].

[b] In current setup a falling grain will always accelerate and never reach terminal velocity, but we constrain the maximum height in the simulation, such that no arbitrary big velocities occur.

of the circumscribed ellipsoid, which are a little more difficult to implement while giving no advantage over the current choice.

The discrete grain size distribution (*e.g.* from sieving) can be approximated by the Weibull distribution [Wei51]. The advantage of using this particular continuous distribution is that it is given by only two parameters and describes a distribution of a sediment obtained by a crushing process very well.[c]

The probability and cumulative density functions of the Weibull distribution for a random variable $x \geq 0$ are given by

$$\mathrm{pdf}(x;k,\lambda) = \left(\frac{k}{\lambda}\right)\left(\frac{x}{\lambda}\right)^{k-1} \exp\left(-\left(\frac{x}{\lambda}\right)^{k}\right)$$

and

$$\mathrm{cdf}(x;k,\lambda) = 1 - \exp\left(-\left(\frac{x}{\lambda}\right)^{k}\right), \tag{15}$$

where k—the shape and λ—the scale parameters are both non-negative. The mean (μ) and variance (σ^2) of a Weibull-distributed variable are:

$$\mu = \lambda\Gamma(1+1/k), \qquad \sigma^2 = \lambda^2\Gamma(1+2/k) - \mu^2. \tag{16}$$

In the examples of artificial sediments we shall use three grain size distributions: an univariate with grain size 1, a bivariate with grain sizes $1/2$ and 1, and Weibull-distributed. The last distributions is generated using shifted Weibull distribution, where x is substituted for $x-m$ in order to avoid very small grains:

$$\mathrm{cdf}(x;k,\lambda,m) = 1 - \exp\left(-\left(\frac{x-m}{\lambda}\right)^{k}\right), \tag{17}$$

with parameters $k = 6$, $\lambda = 2/3$ and $m = 1/3$. Their mean[d] and variance are $\mu \approx 0.9518$, $\sigma^2 \approx 0.01436$. The probability and cumulative density functions are shown in the following figure:

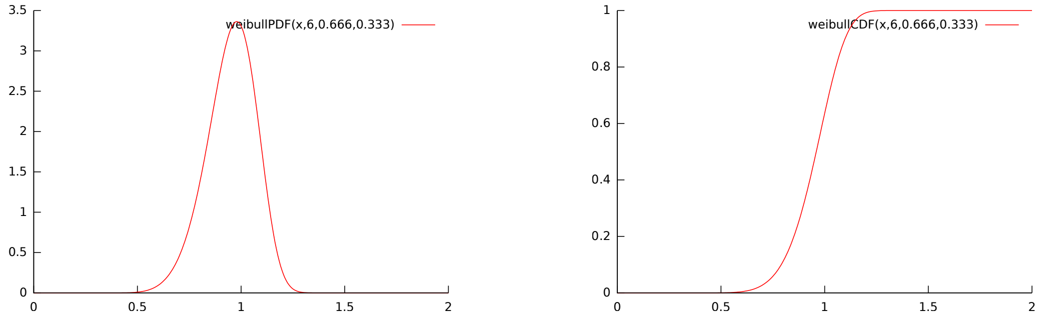


Figure 7.2 Weibull probability (left) and cumulative density functions (right) used in examples of artificial sediments.

---

[c] The Weibull distribution is also know as Rosin-Rammler distribution [RR33] and is used in powder technologies for particle size description. See also [BW95].

[d] The mean of a shifted Weibull-distributed variable is also shifted by m.

Weibull distributed numbers can be generated from the inverse cumulative density function given a uniform distribution of random numbers in $[0, 1]$:

$$y = \mathrm{cdf}(x; k, \lambda, m) \Leftrightarrow \mathrm{icdf}(y; k, \lambda, m) = x.$$

$$\mathrm{icdf}(y; k, \lambda, m) = \begin{cases} m & \text{for } y \leq 0 \\ m + \lambda \left| \ln y \right|^{1/k} & \text{else} \end{cases} .$$

Examples of artificial sediments

- *Spherical grains.* In vast majority of scientific research on artificial sediments the grains are spheres of different sizes. We study three different grain size distributions mentioned above: univariate, bivariate, and a Weibull distributed. We refer to this setups with corresponding abbreviations: SG1, SG2, and SG3.

  The spheres are approximated by triangulated surfaces consisting of 320 triangles and 162 vertices. They are generated by subdivision procedure starting from a icosahedron over two iterations. Subdivision is applied to each triangle, which is split into four triangles by adding new vertices at edge midpoints and pushing out newly created vertices onto the sphere's surface.
- *Icosahedral grains.* We study three different grain size distributions, which are the same as for spherical grains and refer to this setups with corresponding abbreviations: IG1, IG2, and IG3.
- *Cubical grains.* Again, we study three different grain size distributions, as before and refer to this setups with corresponding abbreviations: CG1, CG2, and CG3.

7.3 Summary

In this section most important geometric properties of porous media were mentioned; grain size and shape, porosity, two-point correlation and the derived properties: the surface area and characteristic pore length scale.

Spherical packings—the simple cubic and face-centred cubic—were introduced. These form the most simple porous media studied later and are well suited for benchmarking.

More complicate porous media were created with new "settleDyn" code based on a physics game engine "Bullet". Similar code was used in previous work but lacked the more appropriate and realistic sedimentation dynamics.

For the simulations on more complex porous media the resulting unconsolidated sediments of the "settleDyn" code are used; these include spherical, icosahedral, and cubic shapes. All three shapes are used in generation sediments with univariate, bivariate, and Weibull grain size distributions. This results in nine setups for fluid-flow simulations.

8. Flow-through simulation setup

8.1 Flow-through laboratory experiments

A sample of a porous medium like borehole sandstone is placed in a duct with geometry fitted to the sample's geometry and fixated in there. Porous media consisting of loose particles are clamped from both sides, inlet and outlet, either by permeable membranes or by plugs with smaller inlet and outlet openings than the particles' diameters. Porous media like sandstones do not require such fastening elements. Confining pressure can be applied by using a soft shell around the sample, which is then put under pressure in an oil bath, for example. A more complicated setup is required to control the sample's temperature, for example, or for insertion of other measuring devices. (See for example [MSKM08] for detailed experimental setup.)

At the inlet the fluid's pressure is increased by means of a pump or elevated reservoir connected by a pipe to induce fluid flow. A set flow rate or pressure are maintained during an experiment. Temperature, chemical composition and tracer additives can be controlled at the inlet.

Measurements of the fluid are taken usually at the outlet because of easy access; pressure, velocity distribution, temperature, chemical composition and other physical quantities can be measured.

Geometry and computer simulation setup Computer simulations, to be compared to laboratory experiments, mimic the above experimental setup to some extent. Observation of all quantities is simple in a computer experiment but not the configuration of the inlet properties and copying of the boundary conditions; the latter are model dependent. Before continuing with the setup description we first choose a set of equations describing the fluid flow. Then we will be able to set initial and boundary conditions according to the chosen model.

We solve the Navier-Stokes Equations of an incompressible, Newtonian fluid with constant viscosity. The choice of these constraints is mainly due to computational complexities and physical properties of the problem being studied. The Reynolds number Re is a ratio of inertial forces to viscous forces and can be used to differentiate between laminar and turbulent flow regimes. Usually it is defined by a grain's diameter D as the characteristic length scale, fluid's superficial velocity u_{s}, and kinematic viscosity of the fluid ν;[a]

$$\mathrm{Re} := u_{\mathrm{s}} D \nu^{-1}. \tag{18}$$

---

[a] "Although, by analogy to the Reynolds number for pipes, D should be a length dimension representing the elementary channels of the porous medium, it is customary (probably because of the relative ease of determining it) to employ some representative dimension of the grains for D (in an unconsolidated porous medium)." [Bea88, p. 125]. We shall also add to this that the hydraulic diameter would be a better length-scale. Different definitions of the Reynolds number will lead to differences in conclusions based on it later on.

The computationally intensive transient, possibly non-laminar flows are excluded because of limited computational resources available.

The computational time is increasing with the number of mesh elements N, which is proportional to the Reynolds number approximately as $\mathrm{Re}^{9/4} \approx N^3$. This estimation arises when looking onto Kolmogorov microscales [Pop00, Chapter 6].

The Kolmogorov length scale η is $(\nu^3/\varepsilon)^{1/4}$, where ε is the kinetic energy dissipation rate. It is proportional to u_s^3/D giving the following estimate for η based on the Reynolds number: $D/\eta \approx \mathrm{Re}^{3/4}$. The number of the mesh elements resolving the Kolmogorov length scale η in one dimension is $N = D/h \gtrsim D/\eta = \mathrm{Re}^{3/4}$. Hence the above estimate in three dimensions.

Consider a cubic domain filled with n^3 spheres (in simple cubic packing, $\phi \approx 50\%$ porosity) of diameter D. A sphere discretized at resolution $h = D/128 = D/2^7$ allows direct numerical simulations for Reynolds numbers up to $(D/h)^{4/3} = 2^{9.\bar{3}} \approx 645$. For the given computational resources (largest computer's memory of 256 GiB) the mesh size can not exceed approximately $N_\mathrm{max} = 50 \cdot 10^6$ elements; thus there are at most $\sqrt[3]{(N_\mathrm{max}/\phi)/(2^7)^3} \approx 3.6^3$ spheres in the domain. For a densest packing (fcc) the porosity is around 25%, giving $\approx 4.5^3$ spheres. Both setups are not suitable for spatial averaging process because of their small size.

A reduction of the mesh resolution increases the number of spheres in the domain, but reduces the maximum Reynolds number at the same time; some examples: For $D/h = 64$ mesh resolution, the maximum $\mathrm{Re} = 2^8 = 256$, and $n^3 \approx 7.3^3$ or $\approx 9.1^3$ for a densest packing. For $D/h = 32$ mesh resolution, the maximum $\mathrm{Re} = 2^{6.\bar{6}} \approx 101$, and $n^3 \approx 14.5^3$ or $\approx 18.3^3$ for a densest packing. The latter two resolutions are more suitable for spatial averaging. Another possibility is to use a turbulence model instead of the cost intensive Direct Numerical Simulation.[b]

The low Reynolds number leads to small velocity magnitudes and, consequently, to low Mach number, where we can neglect any compressibility effects. It also means that shock waves and other similar singularities are excluded.[c]At low Reynolds numbers the flow regime is laminar, and for the most cases steady-state.[d] At the upper range of the possible Reynolds numbers starting from 10 to 100 (based on the above definition of the Reynolds number (18)), transient flow patters can occur. We restrict our computer simulations to flows with steady-state behaviour. As extension of this work the transient flows should be studied by using the large eddy simulation turbulence model (or with much larger computational resources).

Observing the fluids' velocities from the experimental point of view, low permeability samples require very high pressure gradients

---

[b] Some simulations using Large Eddy turbulence model were performed showing promising results, but were not followed further.

[c] The Mach number $\mathrm{Ma} := u_\mathrm{s}/a$, where a is the speed of sound. The speed of sound of a fluid is usually larger than 100m/s but for very low temperatures, pressures, or densities, thus resulting in low Mach numbers for the given superficial velocities and incompressible fluid flow.

[d] This depends on the geometry too.

to arrive at non-creeping flow, which are usually not observed far from bore wells (in the context of deep geothermal power plants or gas storage.[e]) The case of very low Reynolds number is covered by Darcy's Law which can be derived from the Stokes equation for creeping flows and is very well studied.

We shall concentrate our simulation efforts in the transitional range of flow regimes from laminar to turbulent. Such regime can occur in loose sediments with gas flow through. [DWK+13] Another possible application field is the upper layer (few centimetres) of a river bed called hyporheic zone. [TSM+13]

---

[e] For semipervious and impervious soils with permeabilities less than $k = 10^{-8}\text{cm}^2$, which corresponds to very fine sand, silt, loess, loam or stratified clay [Bea88, p. 136], the particle diameter from $D = 10^{-3}\text{mm}$ for clay to 10^{-2}mm for silt [Bea88, p. 40], the fluid properties $\rho = 10^3\text{kg/m}^3$ and $\mu = 3 \cdot 10^{-4}\,\text{Pa} \cdot \text{s}$ the negative pressure gradient is $\nabla p = \text{Re}\,\mu^2/(D\rho\,k) \in [45, 450] \cdot 10^6\text{Re} \cdot \text{Pa/m}$, and for $\text{Re} \geq 1$ it is $\geq 450 \cdot \text{MPa/m}$. Such pressure gradients can arise for example while hydraulic fracturing procedures.[LHZ05, WWF+13]

Consider a simple geometry copying an experimental setup; a square duct with a porous media sample placed inside, as shown in the below figure.

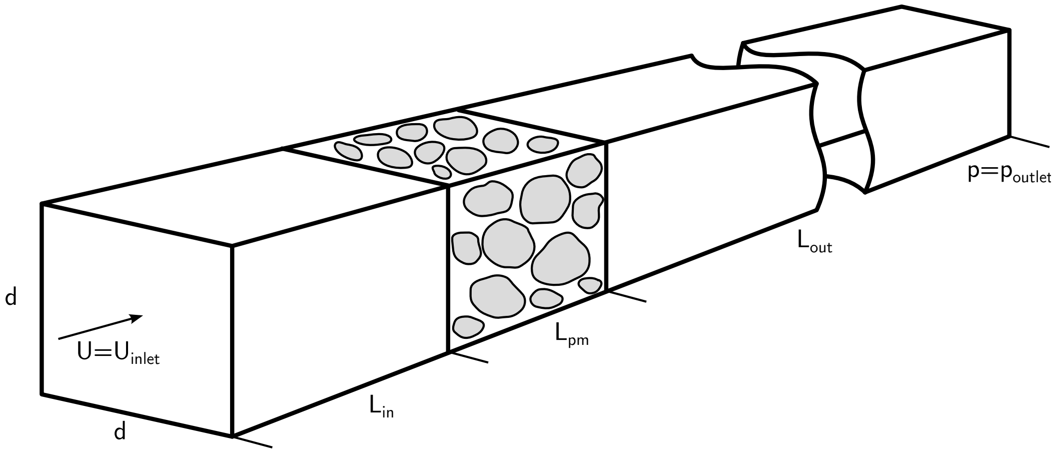


Figure 8.1 Square duct (size $d \times d$) partially filled with porous media sample with long inlet and very long outlet sections. Inlet velocity and pressure outlet boundary conditions indicated. Inlet, porous media, and outlet sections as indicated by their lengths L_{in}, L_{pm}, L_{out}.

The arrows are indicating the flow direction from the inlet, where usually a velocity boundary condition is assigned, to the outlet, where the pressure is set and the flow is perpendicular to the outlet's surface.

Based on this initial configuration we shall analyse the following topics: reduction of the computational domain's size (removing inlet and outlet), grid convergence of the Direct Numerical Simulation using PISO method.

8.2 Summary

From a description of an experimental flow-through experiment we described a general computer setup for fluid flow simulations. A short analysis of the flow regimes combined with computational complexities of the solution of compressible, non-isothermal fluid flows resulted in the choice of incompressible Navier-Stokes Equations for the fluid flow simulation.

Estimation based on the Kolmogorov length scales of the required mesh resolutions for the Direct Numerical Simulation for a given Reynolds number yield the size of the largest computational setup; Maximum available computer memory constraints the number of elements in a mesh resulting in computational domain for 3.6^3 to 4.5^3 spheres (depending on the packing) for the Reynolds number up to 645. Lower Reynolds numbers would allow larger computational domains (more suitable for spatial averaging) but the inertial effects will also decrease. Large Eddy Simulations are also a viable option but lack verification data.

At the given computational resources only steady-state simulations are possible in acceptable time (around 48h per simulation on 480 CPU cores for single simulation).

General simulation setup is shown and objectives to reduce the computational efforts are stated.

9. Row of spheres in square duct

The most simple setup to start validation of the numerical method consists of a square duct filled with a row of spheres of same diameter as the duct's height/width.

9.1 Setup description

Consider a square duct with dimensions $D \times D \times N \cdot D$ filled with row of N spheres with diameter $D = 1$, such that their centres are placed in the duct's centre, as shown in the Figure 9.1.

On the left side an inlet boundary condition is imposed, where the velocity field is mapped from inside the domain as shown in the Figure 9.1. To limit possible numerical oscillations, the velocity field is scaled, such that the volumetric flux is constant. On the right side an outflow boundary condition, and on the duct's walls and spheres the no-slip boundary condition is specified.

The inlet and outlet are essentially not existent here as we want to study the mesh refinement and mesh-type influence at different Reynolds numbers. The inlet/outlet length influences are studied in the Section 10.2 Influence of the outlet length.

Most simulations were run on 4 spheres to reduce computational complexity. To estimate the influence of sphere number we have tested a setup with 8 spheres in the row. Four spheres are sufficient for the setups following.

Simulations with kinematic viscosities $\nu = 0.1$ and 0.01 at mesh resolutions $h = D/32$, $D/64$, $D/128$, and $D/192$ were solved with `icoFoam` OpenFOAM solver for incompressible, Newtonian flows until steady-state.

For finer meshes lower viscosities down to $\nu = 5 \cdot 10^{-4}$ were simulated.

For the evaluation the following variables were used:

- pressure drop Δp,
- average velocities u_{avg} and u_{rms},

Additionally velocity profiles along of three selected lines were compared.

In the following Figure 9.1 a diagonally cut domain with typical velocity and pressure distributions is shown.

9.2 Meshing

All of these setup variations (resolution and kinematic viscosity) were applied to four different meshes:

- *hexahedral*—the domain was discretized with hexahedrons of same size, and the spheres' surfaces kept staggered.
- *hexahedral/refined*—the domain was discretized with hexahedrons as before, but the cells on spheres' surface are subdivided twice, such that the mesh size is quarter of the interiors mesh size. A transition layer is present.
- *snapped*[a]—the domain was discretized with hexahedrons of same size (as in the first case). On the spheres' surfaces "snapping"

---

[a] Or also called "body-fitted" mesh.

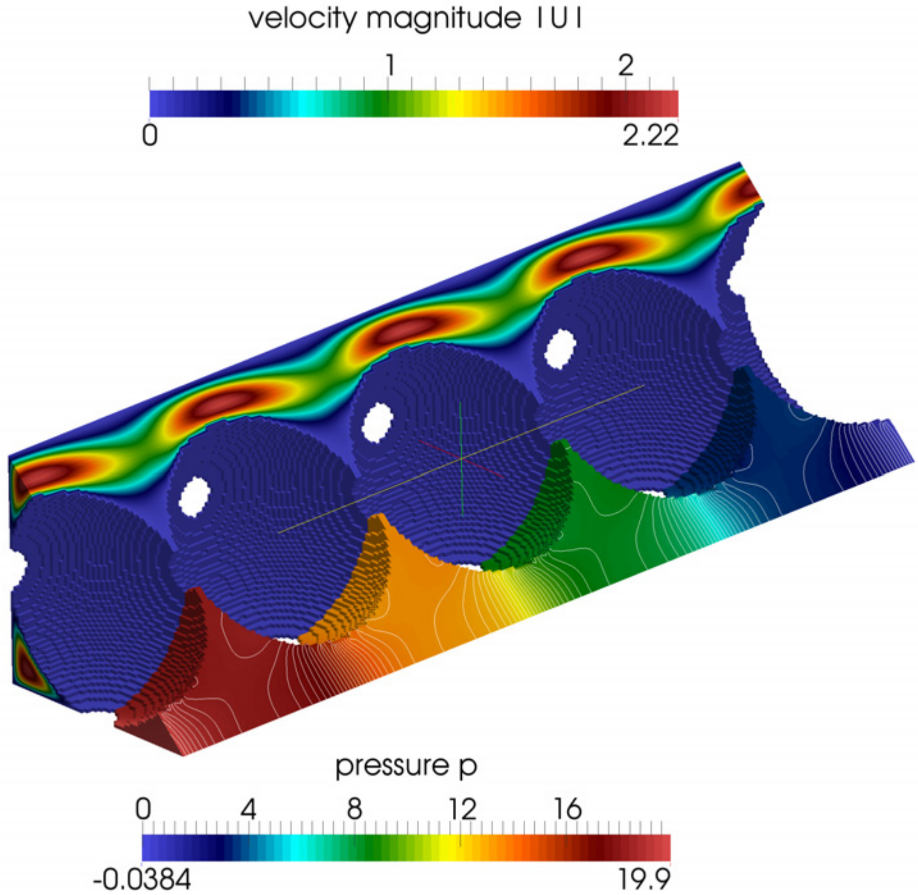


Figure 9.1 Diagonal cut geometry of the sphere row in square duct simulation. Flow from left to right. Typical velocity (upper part) and pressure distribution (lower part) are shown. The solution shown corresponds to the hexahedral mesh with mesh resolution $h = D/64$, and kinematic viscosity $\nu = 0.01$. The average inlet velocity is scaled to unity.

of the hexahedrons to the boundary was applied resulting in much smoother surface representation on cost of the distorted cell shapes.[b]

- *snapped/refined*—the domain was discretized with hexahedrons and refined on the spheres' surface (as in the second case), and the "snapping" procedure was applied. The already refined boundary can cling to the spheres better, than in the coarse (not refined) case because of more degrees of freedom.

---

[b] This increases the non-orthogonality of the mesh and will require special non-orthogonality correction in the inner iterations of the PISO algorithm [Jas96, pp. 83–86].

Cross-sections of the studied meshes (at the same initial resolution) are shown in the Figures 9.2 and 9.3.

Notes

- “Snapped” and refined meshes are present only in two resolutions: $h = D/32$ and $D/64$ because of insufficient memory resources available. There are for the hexahedral mesh: *ca.* 62k cells at $D/32$ mesh resolution, 500k for $D/64$, 3992k for $D/128$, 13487k for $D/192$, and 62717k for $D/256$[c]; and for the refined hexahedral meshes there are: 482k for $D/32$, and 3374k for $D/64$. The “snapped” meshes have similar number of cells.
- “Snapped” meshes are not available for resolutions higher than $h = D/128$ because the OpenFOAM’s meshing code `snappyHexMesh` did not converge in the mesh quality improvement loop. The meshes were not usable for numerical simulations because elements of insufficient quality were present.

---

[c] Simulations for this resolution exceeded the available computational resources and are not included in the results.

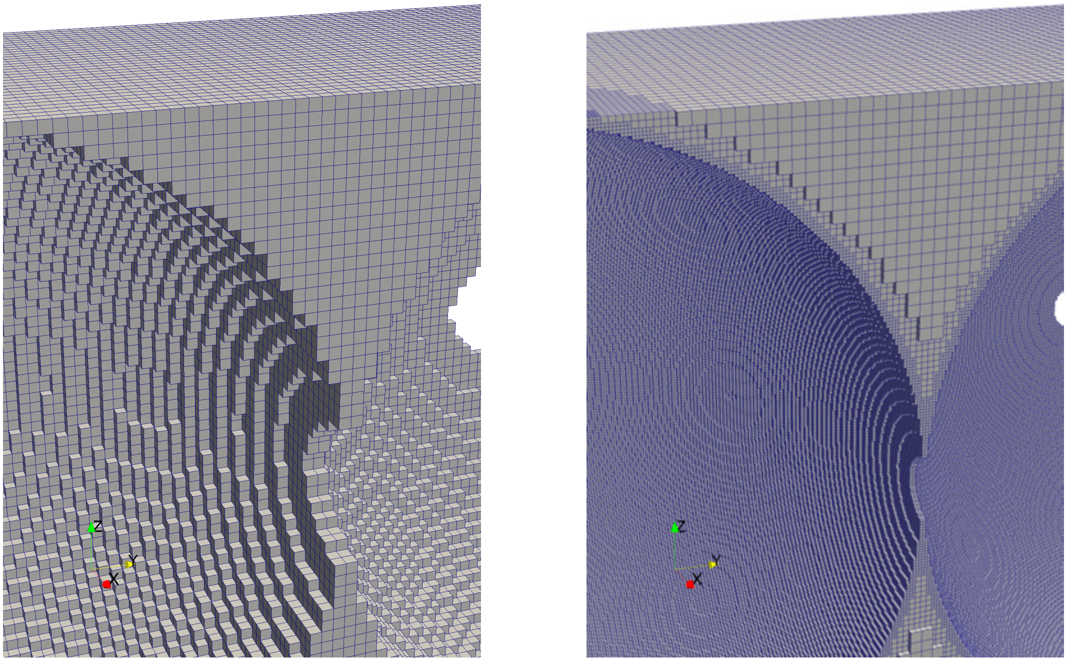


Figure 9.2 Hexahedral meshes. Cross-cut of a row of spheres in a square duct.

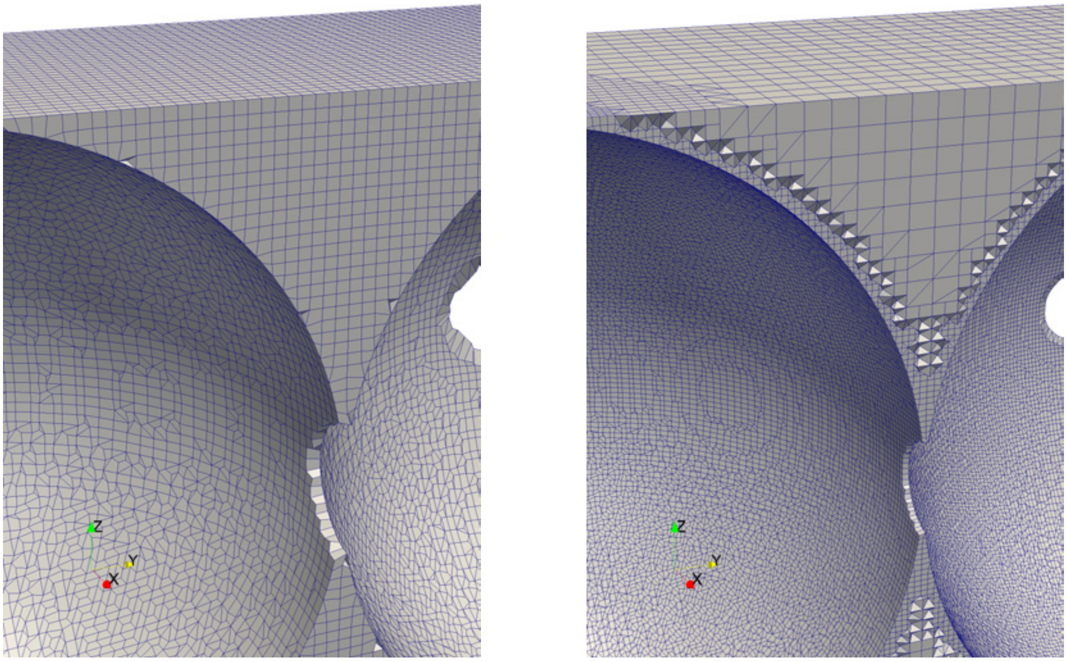


Figure 9.3 Almost hexahedral meshes with “snapped” boundaries. Cross-cut of a row of spheres in a square duct.

9.3 Evaluation

Pressure drop

Plots of the pressure drop between outlet and inlet versus mesh resolution and kinematic viscosity are shown in the following Figures 9.4 and 9.5.

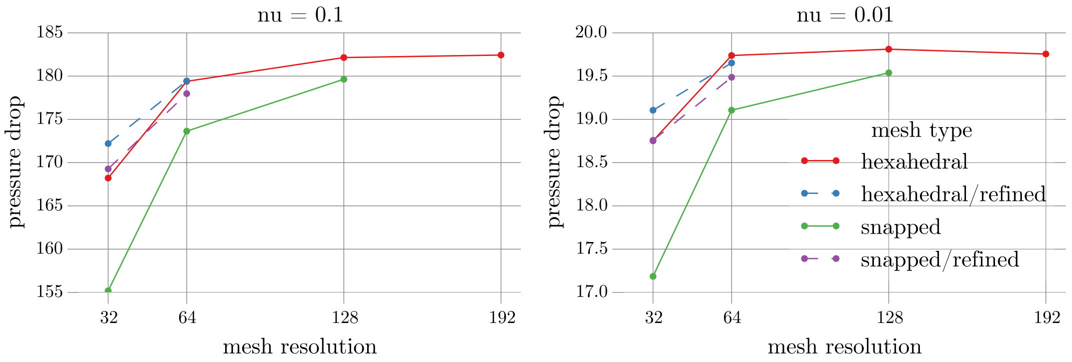


Figure 9.4 Pressure drop at two selected kinematic viscosities $\nu = 0.1$ and 0.01 for different mesh resolutions.

With increasing mesh resolution convergence of the pressure drop is observed. There is only a small difference between mesh types but for the snapped, unrefined mesh. A reason for this is not clear, but probably has to do with the mesh's non-orthogonality and slower numerical convergence.

For the highest resolution $h = D/192$ simulations with kinematic viscosities down to $5 \cdot 10^{-4}$ were performed. The pressure drop and the difference to lower resolutions is shown in Figure 9.5:

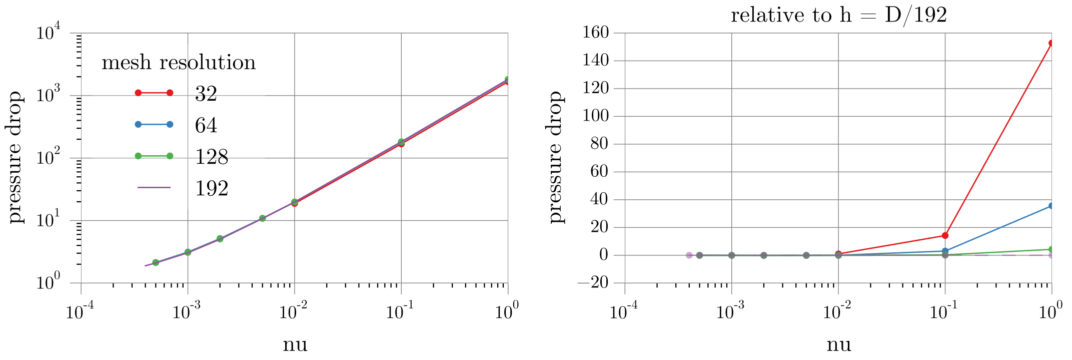


Figure 9.5 Pressure drop versus kinematic viscosity. The graph on the right shows the difference to the pressure drop at mesh resolution $h = D/192$.

Linear regression of the pressure drop at mesh resolution $h = D/192$ (with most data points and most expected accuracy of the solution, as shown in the previous Figure 9.4) through three points at highest kinematic viscosities $\nu = 1$, 0.1, and 0.01 gives a line

$$\Delta p(\nu) = 1.8185092 \cdot 10^3 \nu + 1.10405251$$

with the total residual from least squares interpolation being ≈ 0.48.[d]

---

[d] The total residual is the squared Euclidean 2-norm of the least squares approximation of the linear equation system for the polynomial fit.

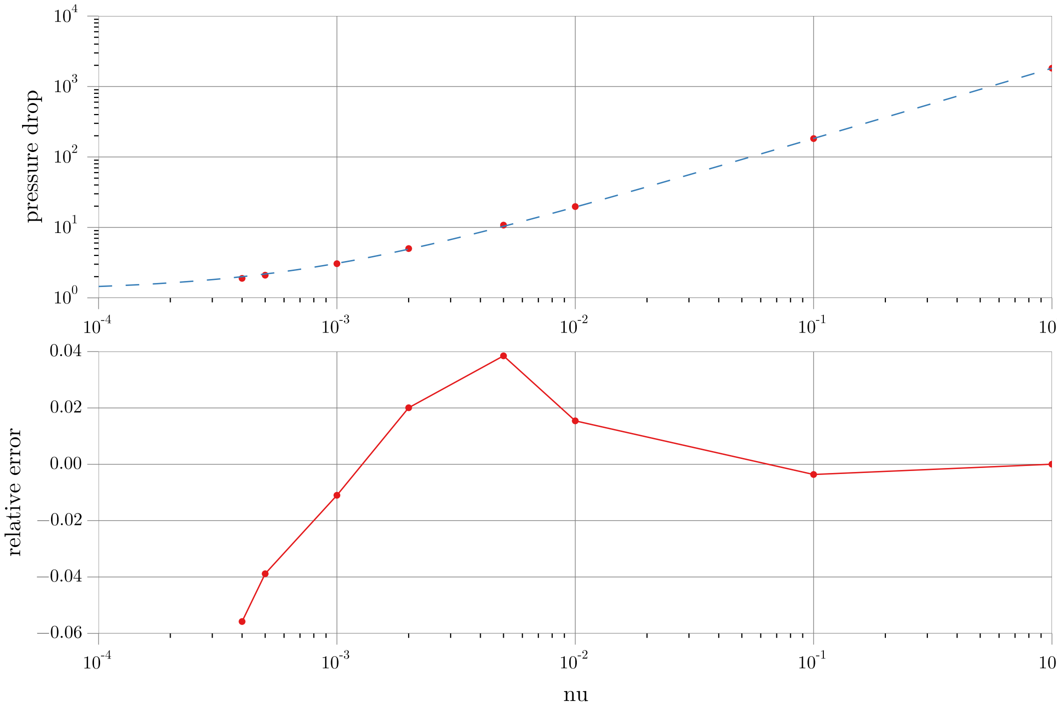


Figure 9.6 Linear fit and the relative error of pressure drop for mesh resolution $h = D/192$.

Using the full data set, the linear fit is:

$$\Delta p(\nu) = 1.81827260 \cdot 10^3 \nu + 1.26826977$$

with the total residual ≈ 0.73. The log-log plot and the relative error plot are shown in the following Figure 9.6:

Both lines are almost undistinguishable graphically, by the linear fit's slope, and only slightly in the offset.[e]

---

[e] The current offset is very small and could be set to zero as $\nu \to 0$, if the inertial terms were not present. The presence of inertial forces might explain the slightly higher offset for the full data set.

Velocity profiles We have sampled velocity profiles at three selected lines at different mesh resolutions for a constant kinematic viscosity $\nu = 0.01$. The lines are defined by the following end-points (for $D = 1$):

- line A: $(1.5, 0, 0)$—$(1.5, \sqrt{3/2}, \sqrt{3/8})$,
- line B: $(1/2, \sqrt{1/2}, 1/2)$—$(7/2, \sqrt{1/2}, 1/2)$,
- line C: $(1/2, \sqrt{1/2}, 1.075)$—$(7/2, \sqrt{1/2}, 1.075)$.

The distance between sampling points is $D/256$. The results for $\nu = 0.01$ are shown in the three following figures; Scales of the velocity components are different in each plot!

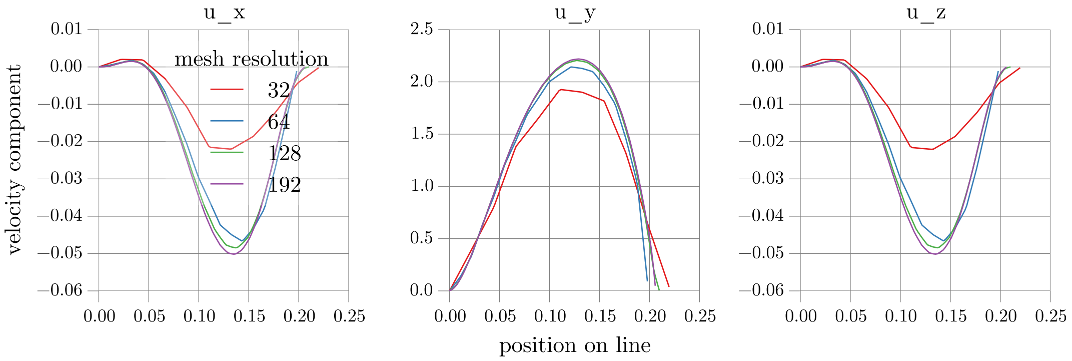


Figure 9.7 Velocity profiles along the first sampling line A for $\nu = 0.01$.

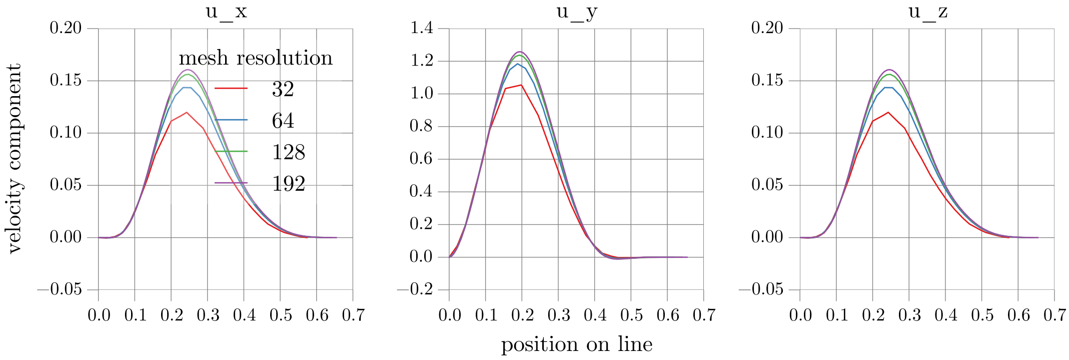


Figure 9.8 Velocity profiles along the second sampling line B for $\nu = 0.01$.

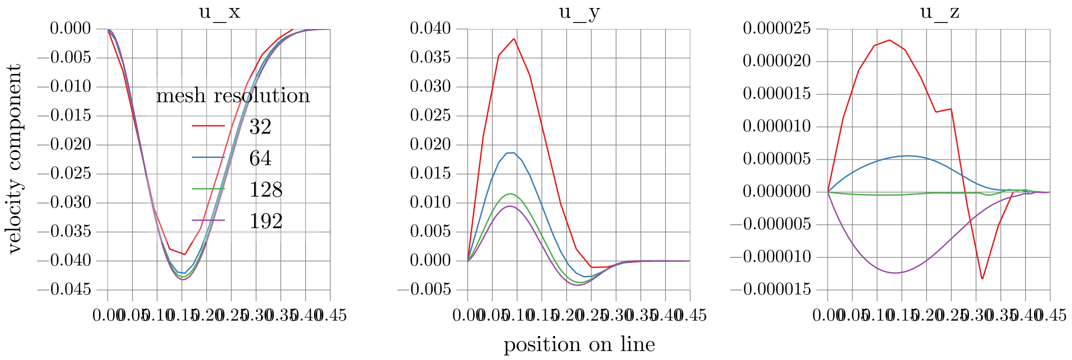


Figure 9.9 Velocity profiles along the third sampling line C for $\nu = 0.01$.

In all but the last plot clear convergence of the velocity profiles with higher mesh resolution is observable. (In the last plot the z-velocity component is numerically insignificant.) Aside the coarseness of the velocity profiles at lower mesh resolutions, starting with $h = D/64$ towards finer meshes, the solutions are almost as good as at the highest mesh resolution. This was also observed in the pressure drop evaluation in Figure 9.5.

For the highest mesh resolution the velocity profiles w.r.t. the kinematic viscosity have strong dependency. Especially on the second and the third lines back-flow patterns are different. The results are shown in the following three figures:

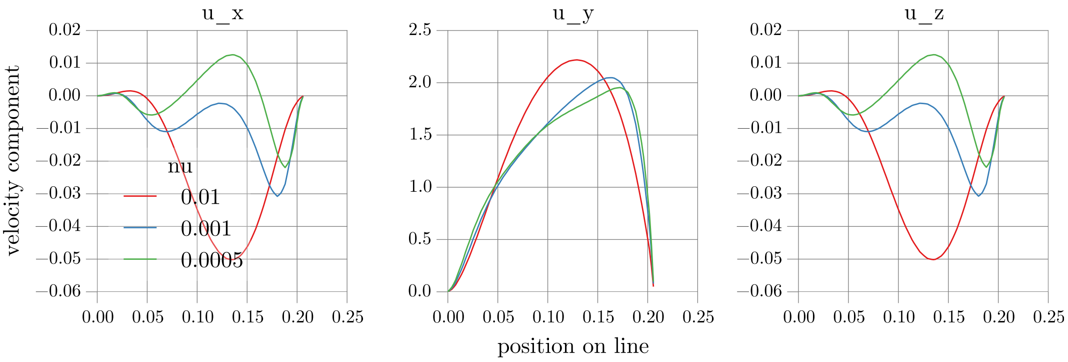


Figure 9.10 Velocity profiles along the first sampling line A at mesh resolution $h = D/192$.

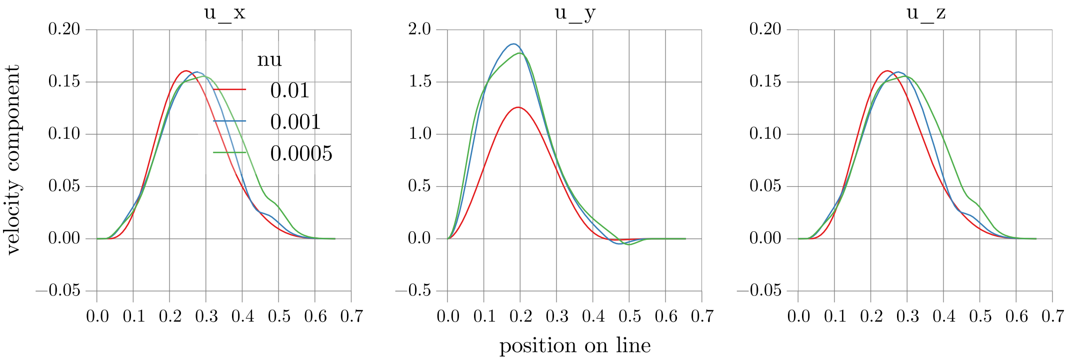


Figure 9.11 Velocity profiles along the second sampling line B at mesh resolution $h = D/192$.

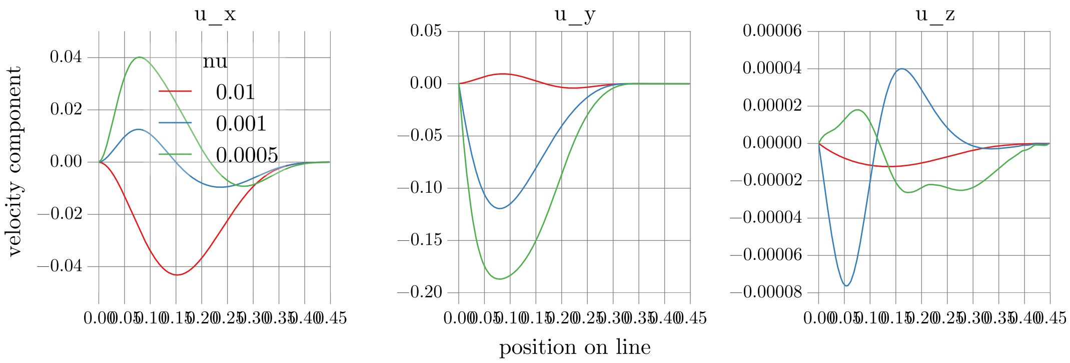


Figure 9.12 Velocity profiles along the third sampling line C at mesh resolution $h = D/192$.

Along the first sampling line a clear skewing of almost parabolic velocity profile at high viscosity towards sphere's surface is visible. Also notice the change of direction for the x- and z-velocity components.

Along the second line a small back-flow pattern develops, visible in the slight dip of the y-component to negative values. Together with the back-flow the x- and z-components change at the same location (around $^1/_2$).

In the last figure strong back-flow is observable.

9.4 Summary

Simple setup for validation of the fluid flow solutions was introduced in this section. Relation of the pressure drop and the fluid's velocity to four setup variables as been studied:

- number of spheres (more than 4 spheres have no influence on the pressure drop),
- kinematic viscosity,
- mesh resolution, and
- mesh type (hexahedral/snapped, and single-size/boundary refined),

Meshing difficulties of the smoothed/snapped meshes did not allow us to study this type of meshes at all resolutions. The meshs' strong non-orthogonality has bad impact on the convergence of the numerical method (and computational time). This is on of the main reasons why only hexahedral (or almost hexahedral) meshes are studied in the following setups.

Convergence of the pressure drop and velocity profiles with increasing resolution was observed. Lower viscosities lead to back-flow pattern development.

It was expected that the back-flow pattern at lower viscosities will cause stronger deviation of the pressure drop from a straight line. This is barely visible (almost insignificant) in the linear fit for the pressure drop above (Figure 9.6). One explanation for this behaviour could be that the low back-flow velocities are almost indistinguishable from the no-slip boundary condition at nearest walls when compared to the magnitudes observed along the preferential flow path.

10. Face-centred sphere packing

10.1 Setup of the face-centred sphere packing problem

In a square duct of length $L_{\text{pm}} + L_{\text{out}}$ several rows of spheres with diameter $D = 1$ are placed in face-centred arrangement. The spheres occupy full duct's volume from its entrance at $x = 0$ to $x = N$, where $N = 4$ is the number of spheres in single dimension. (Only spheres with their centres lying in the given x-range are included.) The square duct's width and height are $(N-1)D\sqrt{3}/2$ and $(N-1)D\sqrt{2/3}$, such that there are two full spheres in the ducts centre and a half of a sphere adjacent to duct's walls. Overall there are $4 \times 3 \times 3 = 36$ full spheres—depending on counting method—included in the simulation.

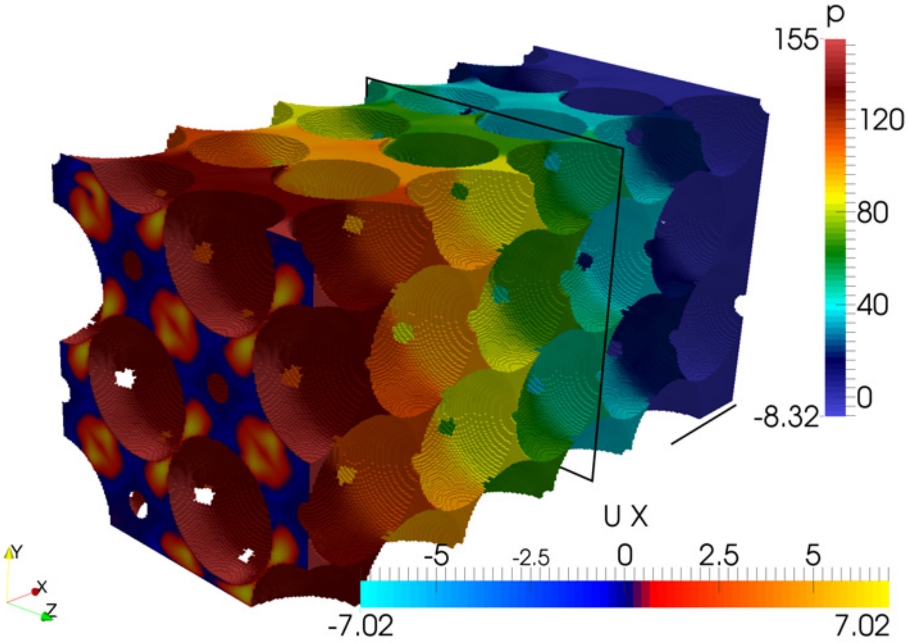


Figure 10.1 Geometry of a face-centred sphere packing in a square duct simulation. Flow from left to right. The outlet length is $L_{\text{out}} = 1$ indicated by short line segment. Typical inlet velocity profile (mapped from the domain's interior, indicated by black frame) and pressure drop along the duct is shown.

We solve the incompressible Navier-Stokes Equations with following boundary conditions: No-slip boundary condition ($U = 0$ and $\partial p/\partial n = 0$) is set on the walls and on the sphere surfaces.

The inlet velocity is mapped from domain's interior and its average is constrained to 1.

Such a mapping allows removing of the inlet portion of the square duct (not filled with the spheres). The mapping from the last sphere's row (at $x = L_{\mathrm{pm}}$) is also possible, but the boundary conditions imposed there in case of outlet length $L_{\mathrm{out}} = 0$ would probably influence the velocity field.

The homogeneous Neumann boundary condition is applied to the pressure at the inlet. At the outlet the homogeneous Neumann boundary condition is applied to the velocity, and pressure is set to zero.

The fluid's kinematic viscosity ν is chosen to be $^1\!/_{100}$. For this value of ν the flow remains stationary while small eddies appear in some regions. At lower viscosities $\nu < {}^1\!/_{200}$ the flow becomes transient.

Several scenarios are simulated; Their outcomes are discussed and compared below. Simulation cases for varying outlet length L_{out} at fixed mesh resolution, and different mesh resolution at fixed outlet length $L_{\mathrm{out}} = 0$ are created. Refinement of the boundary layer and its influence on the discretization errors is also done for $L_{\mathrm{out}} = 0$.

All here presented simulation converged to expected steady-state observed both, in the initial residuals and convergence of physical properties.

10.2 Influence of the outlet length

The inlet must be far enough away from the sample such that its boundary condition has no influence on the flow inside the sample. Similarly, the outlet's boundary conditions must not change the flow too. The inlet's length can be estimated from a lengthwise laminar flow development theories. This, so called entrance length for laminar flow in square ducts of width d is about $L_{\mathrm{e}}/d = 0.072\,\mathrm{Re}_{\mathrm{d}}$ to $0.090\,\mathrm{Re}_{\mathrm{d}}$, as reported in [Joh98, pp. 28-73].

The required length of the outlet is much greater. For transient flows it can be up to 50 times the grain's diameter and increasing with the Reynolds number.[a] For steady-state flows in our case, my experience says, that an outlet length approximately of 15 grain's diameter is sufficient for the flow to become normal to the outlet's surface.[b]

Main disadvantage of this is much larger domain, whereas only the flow inside the sample being studied is of interest. Consider a porous medium with porosity around 50% consisting of spherical grains with diameter D placed in a square duct in simple cubic packing. Let the size of the sample be 10 grains' diameters. Then the volume occupied by the inlet of length $2D$ plus the outlet of

---

[a] In [WS97] free boundary condition were studied for backward-facing step internal flow problem for lengths up to 66 step heights, where the flow has reached fully developed Hagen-Poiseuille profile.

[b] The grains' diameter is not a good length scale in this case. The size of largest eddies or the size of the pores would be more appropriate but more difficult to estimate/measure.

length $15D$ is 3.4 times larger than the effective volume of porous medium: $17 \cdot (10D)^2$ versus $0.5 \cdot 10 \cdot (10D)^2$. Using this setup large parts of the computational domain are necessary for correct simulation, but of no interest.

It is possible to remove the inlet part almost without causing any disturbances to the flow by mapping velocity from inside of the domain, but it requires symmetry. The idea of velocity mapping from domain's inside was used in Eugene de Villiers PhD thesis [dV06, p. 136] to simulate proper turbulent inlet (and although this is a different topic it is ideas origin).

Given a domain with translational symmetry in flow direction at some point downstream. The velocity values at the symmetry plane (or in general a surface) can be mapped one-to-one to the inlet. In order to avoid possible oscilations the velocity is scaled such that the total mass flow corresponds to a given value.

In OpenFOAM this boundary condition is known as "mappedPatch" and requires geometrical data in boundary description file:

```
case/constant/polyMesh/boundary:

inlet
{
    type            mappedPatch;
    nFaces          5020;
    startFace       1268431;
    sampleMode      nearestCell;
    sampleRegion    region0;
    samplePatch     none;
    offsetMode      normal;
    distance        -3;
}
```

and specification for the values (in this case of the velocity field) in the `case/0/U` file:

```
boundaryField
{
    inlet
    {
        type                mapped;
        interpolationScheme cell;
        setAverage          true;
        average             (1 0 0);

        value               uniform (0 $inletArea 0);
    }
    ...
}
```

We study the influence of the short (and finally removed) outlet part on seven cases for outlet lengths $L_{\text{out}} = 0, 1, 2, 4, 8, 16$, and 32 at mesh resolution $h = D/32$. (Various other resolution were tested ($h = D/20$ to $D/192$ and are presented in the Table 10.1.)

Difference of the $L_{\text{out}} = 0$ to the longest outlet with $L_{\text{out}} = 32$ is negligible for global variables pressure drop, and average and root-mean-square velocities, as well as for local pressure and velocity.

The results are summarized in the following table on page p. 46 and visualized in the Figure 10.2

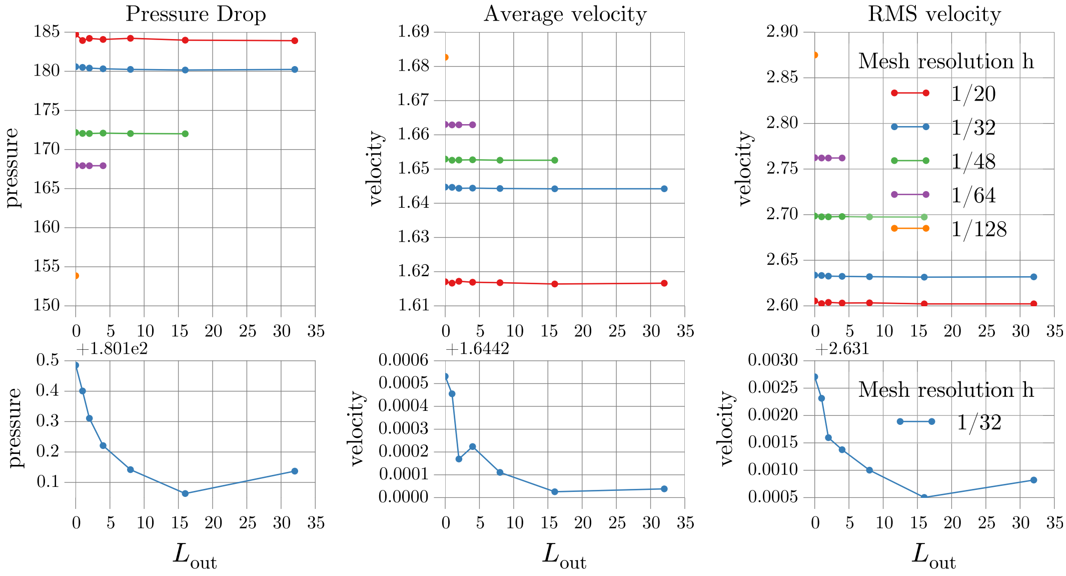


Figure 10.2 Dependency of pressure drop and average velocity from outlet length L_{out} and mesh resolution h of the face-centred sphere packing in square duct simulations. Lower part shows only one resolution ($h = D/32$) for more details not visible in the upper part of the figure.

This observation of the negligible outlet length influence on the solution allow to use shortest possible outlet ($L_{\text{out}} = 0$) for all other simulations.

10.3 Discretization error analysis

For the outlet length $L_{\text{out}} = 0$ convergence of the pressure drop, and average velocities w.r.t. to the mesh resolution is studied. The incompressible Navier-Stokes Equations with kinematic viscosity $\nu = {}^1\!/_{100}$ are solved using transient solver. All solutions are converging to a steady-state, indicated by vanishing initial residuals and converging physical properties.

The simulation results are summarized in the following Table 10.2, and Figures 10.3 and 10.4. Unfortunately the expected convergence with finer mesh resolutions is not achieved.

Using Richarson extrapolation the estimated observed order of convergence is much lower than the order of the numerical algorithm. The Richardson extrapolation, the observed order of convergence, and the Roache's grid convergence index are described in detail in Chapter 8 “Discretization error” in [OR10, pp. 286–342].

Taking last three mesh resolutions ($h = {}^1\!/_{144}$, ${}^1\!/_{160}$, and ${}^1\!/_{192}$) the observed order of convergence for the pressure drop Δp is 0.6539 resulting in Richardson extrapolated value of 125.93. The grid convergence index is 64.59 resulting in 43.8% uncertainity of the pressure drop on the finest mesh. This means, that the pressure drop on

Mesh resolution	Outlet length	Pressure drop	Average velocities		Surface area	porosity
h	L_{out}	Δp	u_{rms}	u_{avg}	A	ϕ
1/20	0	184.741	2.60543	1.61705	116.456	0.261438
1/20	1	183.935	2.60248	1.61665	116.456	0.261438
1/20	2	184.207	2.60383	1.6172	116.456	0.261438
1/20	4	184.072	2.60313	1.61691	116.456	0.261438
1/20	8	184.219	2.60332	1.61678	116.456	0.261438
1/20	16	183.98	2.60218	1.6164	116.456	0.261438
1/20	32	183.917	2.60223	1.61663	116.456	0.261438
1/32	0	180.585	2.63371	1.64473	122.782	0.258225
1/32	1	180.501	2.63331	1.64466	122.782	0.258225
1/32	2	180.411	2.6326	1.64437	122.782	0.258225
1/32	4	180.321	2.63238	1.64442	122.782	0.258225
1/32	8	180.242	2.632	1.64431	122.782	0.258225
1/32	16	180.163	2.6315	1.64423	122.782	0.258225
1/32	32	180.237	2.63182	1.64424	122.782	0.258225
1/48	0	172.172	2.69854	1.65292	125.255	0.259236
1/48	1	172.055	2.69766	1.65257	125.255	0.259236
1/48	2	172.04	2.69762	1.65264	125.255	0.259236
1/48	4	172.096	2.69792	1.6527	125.255	0.259236
1/48	8	172.04	2.69751	1.65258	125.255	0.259236
1/48	16	172.011	2.69737	1.65258	125.255	0.259236
1/64	0	167.974	2.76243	1.66304	127.097	0.259393
1/64	1	167.926	2.76211	1.66291	127.097	0.259393
1/64	2	167.923	2.76207	1.66293	127.097	0.259393
1/64	4	167.926	2.76212	1.66294	127.097	0.259393
1/80	0	163.458	2.80568	1.66951	128.011	0.25913
1/96	0	160.251	2.84178	1.67782	128.577	0.259234
1/112	0	156.306	2.85845	1.67999	129.176	0.25948
1/128	0	153.853	2.87497	1.68268	129.52	0.259497
1/144	0	151.916	2.8899	1.68595	129.819	0.259467
1/160	0	150.186	2.90055	1.68802	130.064	0.259489
1/192	0	147.459	2.91705	1.69173	130.43	0.259509

Table 10.1 Results of the face-centred sphere packing in square duct simulations for different mesh resolutions h and outlet lengths L_{out}.

Mesh resolution	Pressure drop	Average velocities		Surface area	porosity
h	Δp	u_{rms}	u_{avg}	A	ϕ
1/20	184.741	2.60543	1.61705	116.456	0.261438
1/32	180.585	2.63371	1.64473	122.782	0.258225
1/48	172.172	2.69854	1.65292	125.255	0.259236
1/64	167.974	2.76243	1.66304	127.097	0.259393
1/80	163.458	2.80568	1.66951	128.011	0.25913
1/96	160.251	2.84178	1.67782	128.577	0.259234
1/112	156.306	2.85845	1.67999	129.176	0.25948
1/128	153.853	2.87497	1.68268	129.52	0.259497
1/144	151.916	2.8899	1.68595	129.819	0.259467
1/160	150.186	2.90055	1.68802	130.064	0.259489
1/192	147.459	2.91705	1.69173	130.43	0.259509

Table 10.2 Results of the face-centred sphere packing in square duct simulations for different mesh resolutions h for outlet lengths $L_{\mathrm{out}} = 0$.

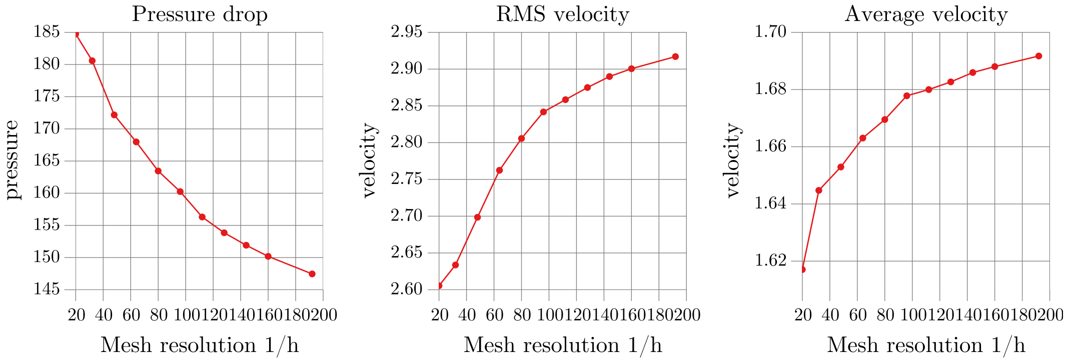


Figure 10.3

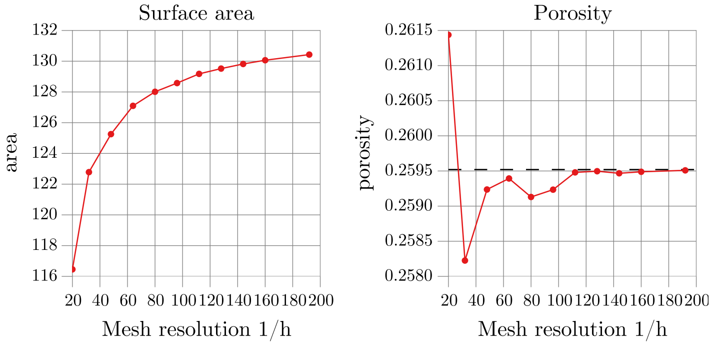


Figure 10.4

the finest mesh $\Delta p = 147.459$ lies in the 95% uncertainity band[c], which is as wide as 147.459 ± 64.59. This is, of course, unacceptable. The numerical algorithm is second-order accurate, and the observed order is expected to be not far away from this value: It can be as low as 1 for highly non-orthogonal meshes or values measured on the domain's boundary, where the one-sided numerical derivatives are of order 1, but not acceptable for such "integral" values as pressure drop along the full length of the simulation domain.

For the rms velocity average u_{rms} the observed order of convergence is 0.7697, the extrapolated value is 3.027, the grid convergence index is 0.3285, and the uncertainity for the last value is 11.26%. Although this is slightly better as for the pressure drop, but not sufficient.

The results for the average velocity u_{avg} are similar to those of u_{rms}.

The reasons for this behavoir is most probably the increasing surface area (Figure 10.4), therefore breaking the necessary precondition of systematic mesh refinement: the geometry of the computational domain changes with mesh resolution.

Surface area

The surface area of a discretized sphere of radius r is not equal to $A = 4\pi r^2$. Due to hexahedral mesh the surface is staggered and the surface area is larger than that of a smooth sphere.

Consider a one quater of a circle (in $\mathbb{R}^2$) with radius r. Because of the discretization by quads (in $\mathbb{R}^2$) the so called taxicab metric is applicable: For the two points $(r, 0)$ and $(0, r)$ the distance between them is always $2r$ and is independent of the actual path as long as there is no "going back".

For a slab of thickness h of a cylinder of radius r its surface area (without the caps) is $A^{\text{cyl}}_{r,h} = 8rh$.

Through the discretization procedure the radius of each a slab i is $\hat{r}_i := j \cdot h$, $i, j \in \mathbb{Z}$, such that $(i \cdot h)^2 + (j \cdot h)^2 = 1/2^2$.[d] Because j is an integer, some rounding is required, *e.g.* $j = \lfloor\sqrt{1/(4h^2) - i^2}\rfloor$. The rounding method used is not important; we use rounding down, but rounding to the nearest integer or rounding up are also possible and do not change the limit of the following sum for h towards zero.

For the slab i of the unit sphere, its surface area (without the caps) is then $A^{\text{cyl}}_{\hat{r}_i,h} = 8h^2\lfloor\sqrt{1/(4h^2) - i^2}\rfloor$, and summing over all slabs from $i = 0$ to $1/2 \cdot 1/\mathrm{h}$, the surface area of all slabs cut in one direction is

$$2 \cdot 8h^2 \sum_{i=0}^{1/(2h)} \lfloor\sqrt{1/(4h^2) - i^2}\rfloor.$$

This accounts not for the whole surface: The surfaces orthogonal to the cutting direction are not included. Following the same procedure for an orthogonal cutting direction, and not counting the surfaces already included into the first sum, one half of them, the total surface

---

[c] The Roache's grid convergence index is designed such that to achieve 95% certainity.

[d] Sphere's diameter is 1.

area of the discretized unit sphere is:

$$A_{1,h}^{\text{sphere}} = 3 \cdot 8h^2 \sum_{i=0}^{1/(2h)} \lfloor \sqrt{1/(4h^2) - i^2} \rfloor .$$

This series is slow convergent albeit not to the true surface area, but overestimating it by $^3/_2$.

The surface areas for the mesh resolutions $D/h = 32$, 64, 128, 192, 256, and 288 are 4.7964, 4.7558, 4.7346, 4.7274, 4.7237, and 4.7224 respectively.

This fits the results in the Figure 10.4 very well assuming the spheres' surface is of 27 spheres: for the mesh resolutions $D/h = 32$, 64, 128, and 192 the simulation surface areas are 4.547478, 4.707289, 4.797033, and 4.830725 respectively. The small difference is accounted for the duct's wall surface (excluding the inlet and outlet).

10.4 Boundary layer thickness

Studying the outlet length influence, revealed strong solution dependency on the mesh resolution. Two major effects are taking place here: One is the changing mesh geometry resolving finer and finer structures of the input geometry, and the other is the discretization error changing with higher mesh resolution. To separate this two issues the following series of simulations was prepared.

In order to quantify the discretization error, the domain's geometry should be fixed. Introducing a boundary layer of thickness γ with finer or same resolution as in the domain's interior, fixes the domain's geometry: the boundary of the geometry is independent of choosen mesh resolution. With this setup the discretization error in the domain's interior can be studied in dependence of the boundary layer thickness γ.

The lowest mesh resolution is chosen to be $h_0 = D/32$. It allows for two—mainly due to memory requirements for meshing—subsequent refinement steps. Starting from h_0, the mesh geometry is refined two times in the boundary layer region, giving mesh resolution $h = D/128$ in that region. In the mesh's interior the resolution is kept at the initial value excluding a thin transition layer from larger to smaller hexahedron elements.

The simulation results are collected in the Table 10.3.[e]

The first Figure 10.5 shows the convergence of pressure drop and velocity averages towards the solution at highest mesh resolution $h = {}^{D}/_{128}$ with increasing boundary layer thickness γ. From the graphics one can conclude that the boundary layer of thickness $\gamma = {}^1/_{10}$ is almost as good as the solution at highest mesh resolution, while saving slightly over 6% of the mesh elements compared to the highest mesh resolution.

The second Figure 10.6 shows the behaviour of the pressure drop and average velocities depending on the mesh resolution for different boundary layer thicknesses. Keeping the boundary (and the boundary

---

[e] The number of elements for the boundary layer thickness $\gamma = 0$ is larger than for $\gamma = {}^1/_{100}$ because the default transition layer thickness of the `snappyHexMesh` is larger than the explicitly specified one.

Mesh resolution	Number of elements	Pressure drop	Average velocities		BL thickness
h	#	Δp	u_{rms}	u_{avg}	γ
32	3664644	156.920	2.76876	1.68968	0
32	2705322	160.174	2.73540	1.69395	$^{1}/_{100}$
32	4377804	155.070	2.77172	1.68936	$^{2.5}/_{100}$
32	7181766	153.892	2.83417	1.68556	$^{5}/_{100}$
32	9680220	153.763	2.86776	1.68312	$^{10}/_{100}$
64	3974940	156.519	2.79984	1.68641	0
64	3220452	159.195	2.79227	1.68778	$^{1}/_{100}$
64	4755804	154.655	2.81124	1.68523	$^{2.5}/_{100}$
64	7351908	153.779	2.85064	1.68366	$^{5}/_{100}$
64	9712728	153.845	2.87109	1.68285	$^{10}/_{100}$
128	10354890	153.853	2.87497	1.68268	∞

Table 10.3 Results of the face-centred sphere packing in square duct simulations for different boundary layer thicknesses γ and three mesh resolutions $D/h = 32$, 64, and 128. The boundary is discretized at mesh resolution $D/h = 128$, therefore the boundary layer thickness at this mesh resolution is infinite.

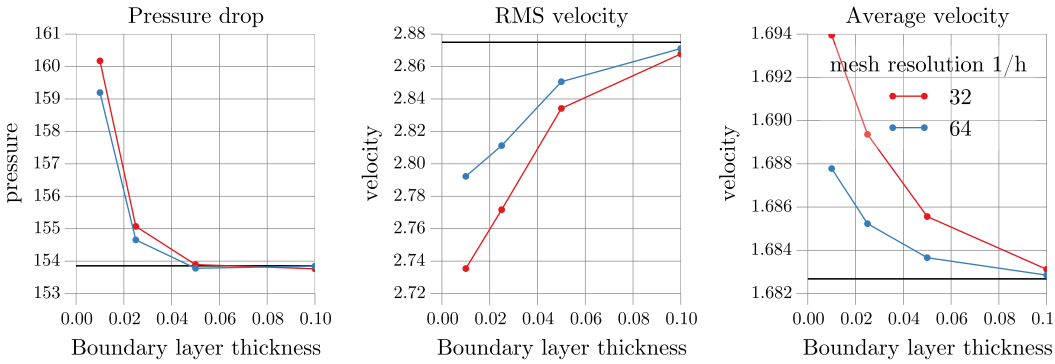


Figure 10.5 Convergence of pressure drop and velocity averages with increasing boundary layer thickness. The horizontal black line represents the solution at highest mesh resolution $h = {}^{\mathrm{D}}/_{128}$.

elements not further away than the boundary layer thickness γ) unchanged, is a more systematic way of mesh refinement as required for the Richardson extrapolation discussed in the Section Discretization error analysis (p. 45). For the boundary layer thicknesses larger than $\gamma = {}^{2.5}/_{100}$ can be observed in all inspected variables.

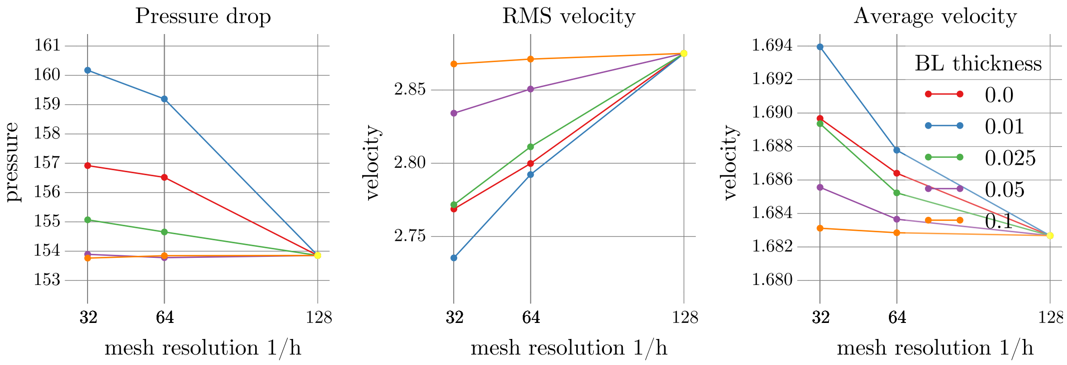


Figure 10.6 Convergence of pressure drop and velocity averages for different mesh resolutions and different boundary layer thicknesses γ.

10.5 Summary

We have described and set up a face-centred sphere packing fluid flow simulation. Special inlet boundary condition (recycling velocity from domain's inside), converting the problem into one with "periodic" boundaries (in flow direction), was applied.

A major point in reduction of the computational domain's size involves the outlet part of the geometry, as mentioned in Section Simulation setup on p. 32. A reduction of at least 3.4 fold was achieved by the reduction of the inlet and the outlet parts. The outlet length influence was studied in detail. For the pressure drop it results in relative difference to the maximum outlet length of approximately 0.45% for the lowest mesh resolution $h = {}^1/_{20}$ and approximately 0.23% for $h = {}^1/_{32}$, and decreasing with higher resolution. For the average velocities the picture is identical to the pressure drop dependency on the outlet length.

In the second part two simulation setups were created for discretization error estimation. For both the outlet length $L_{\text{out}} = 0$ was used. The naïve approach of mesh refinement applied to the whole geometry led to unsystematic refinement required by the applied Richardson extrapolation. This was also visible in the Figures 10.3 and 10.4.

More systematic mesh refinement was done defining a boundary layer thickness. The same dependencies as before were analysed. This time the discretization error vanish with higher mesh resolution for a boundary layer of a minimum thickness $\gamma = {}^{2.5}/_{100}$.

11. Artificial sediments in a pipe.

11.1 Setup description

Consider a pipe with diameter 5 and length 10 filled with a random sediment generated by the previously described "settleDyn" software (Section Examples of artificial porous media on p. 25). To the filled pipe segment an inlet and an outlet pipe segments of the same diameter are attached.

On the left side inlet boundary condition is imposed with constant parabolic velocity profile and volumetric flux equal to unity. On the right side is outflow, and on the duct's walls and grains the no-slip boundary condition is specified.

In the Figure 11.1 a longitudinally cut domain with velocity magnitude distribution for spherical grain setups SG1, SG2, and SG3 is shown on the next page; Same setups for icosahedral and cubical grains are presented on the following pages (Figures 11.2, 11.3).

11.2 Mesh preparation

Mesh generation from the "settleDyn" output to the final tetrahedral mesh requires several steps. The sedimentation software outputs a single STL-file for every simulated grain, which contains the triangulated surface description.[a] From this set of files those are selected, which lie in a specified bounding box completely; The reduced set is used for the mesh generation.

It was already observed (see [GVK97], cited in [MKDMS14, Section 2.2.2]), that fluid flow through a column of packed particles is different in the column's centre than along the walls. This happens because of looser packing along the walls. The bounding box defining the volume of interest is chosen such that the boundary effects along the vertical walls and layering on the bottom (from the sedimentation simulation) remain outside of the volume of interest (VOI); The packed bed for the fluid flow simulation is, so to say, stencilled out of the core without destroying the packing on the new boundary. An example is rendered in the Figure 11.4.

It is not expected, that this precaution will avoid all of the boundary effects because the flow is restricted only to one half of the space by the wall, but it should remove the effects associated with different packing *i.e.* the geometrical influences.

Meshing software There are several open-source (or at least free for academic use) mesh generation software packages available. We have restricted this work to three options:

- snappyHexMesh: The most straight forward approach is the `snappyHexMesh` utility from the OpenFOAM meshing tools.

---

[a] One could also save a single STL-file for every variation of the grain, and then for each simulated grain a reference to the original and a 4×4 transformation matrix. This would save considerable amount of space used but it is also more difficult to use for further processing, which is the main disadvantage.

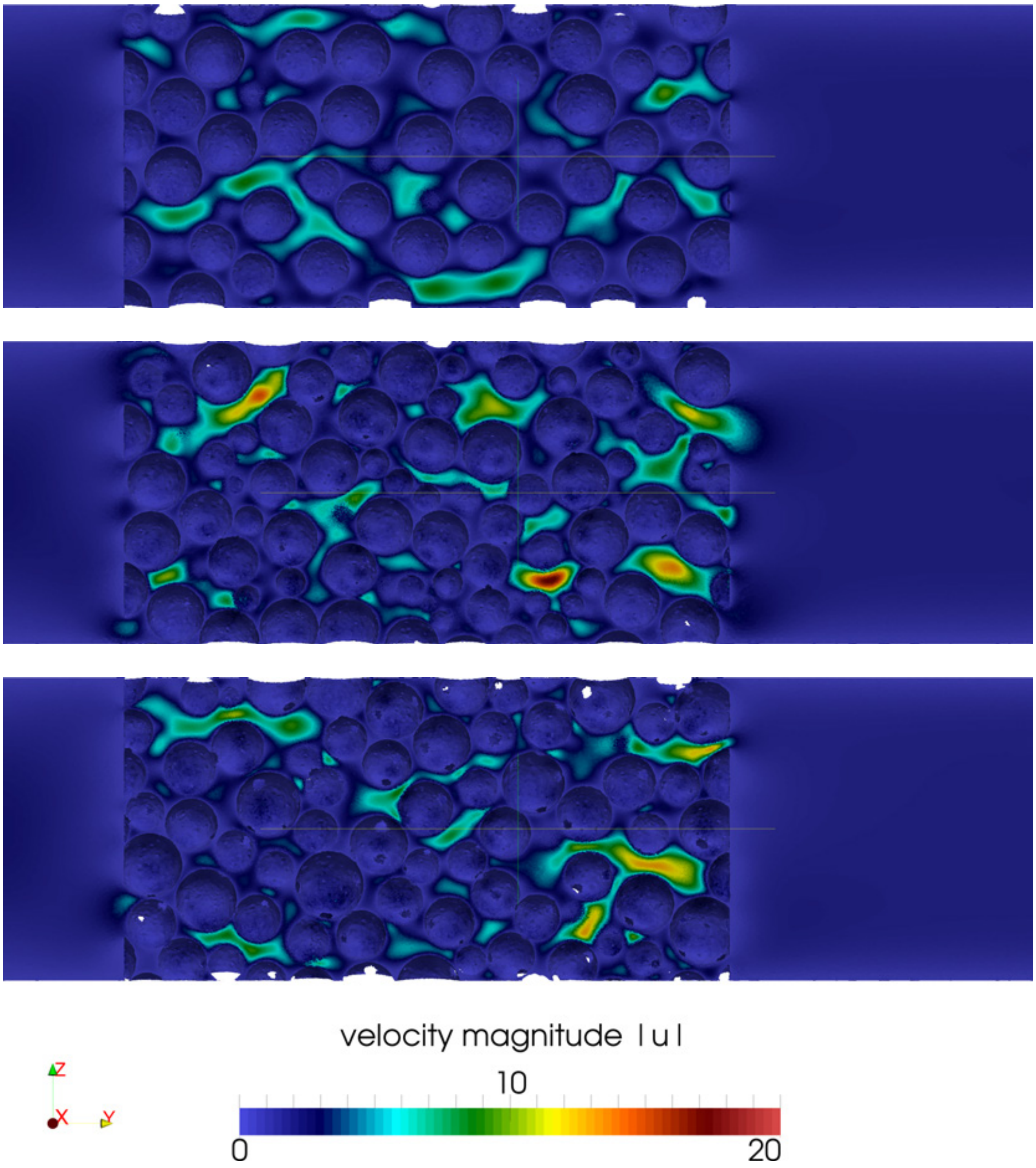


Figure 11.1 Longitudinal cut through a pipe filled with spherical grains. The three setups corresponds to the SG1, SG2, and SG3 setups. Fluid flow is from left to right. The velocity magnitude is shown with same colour scale for all setups, whereas the maximum velocity magnitudes are 10.9748, 18.0944, and 17.5923.

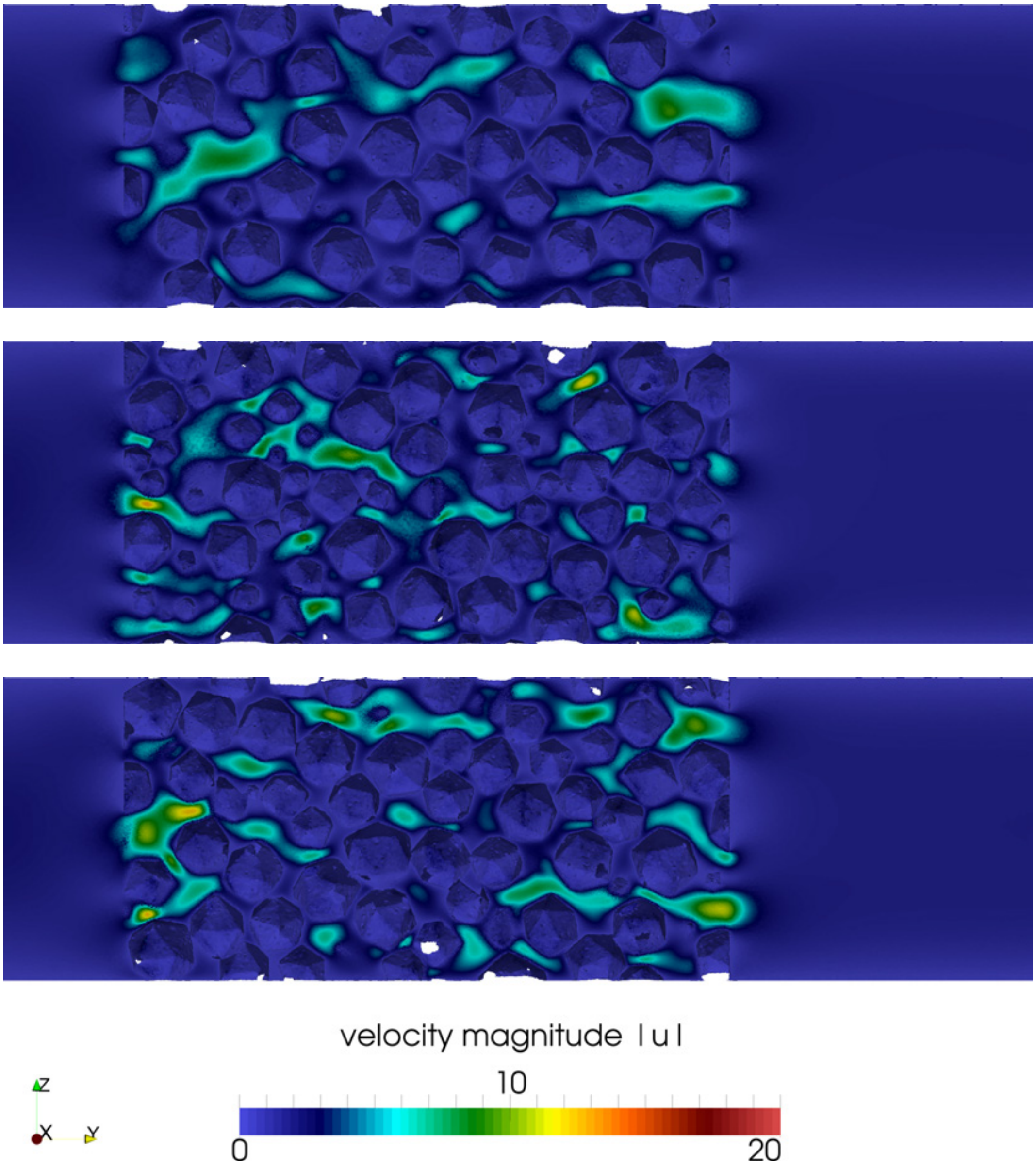


Figure 11.2 Longitudinal cut through a pipe filled with spherical grains. The three setups corresponds to the IG1, IG2, and IG3 setups. Fluid flow is from left to right. The velocity magnitude is shown with same colour scale for all setups, whereas the maximum velocity magnitudes are 10.7398, 13.9347, and 15.3164.

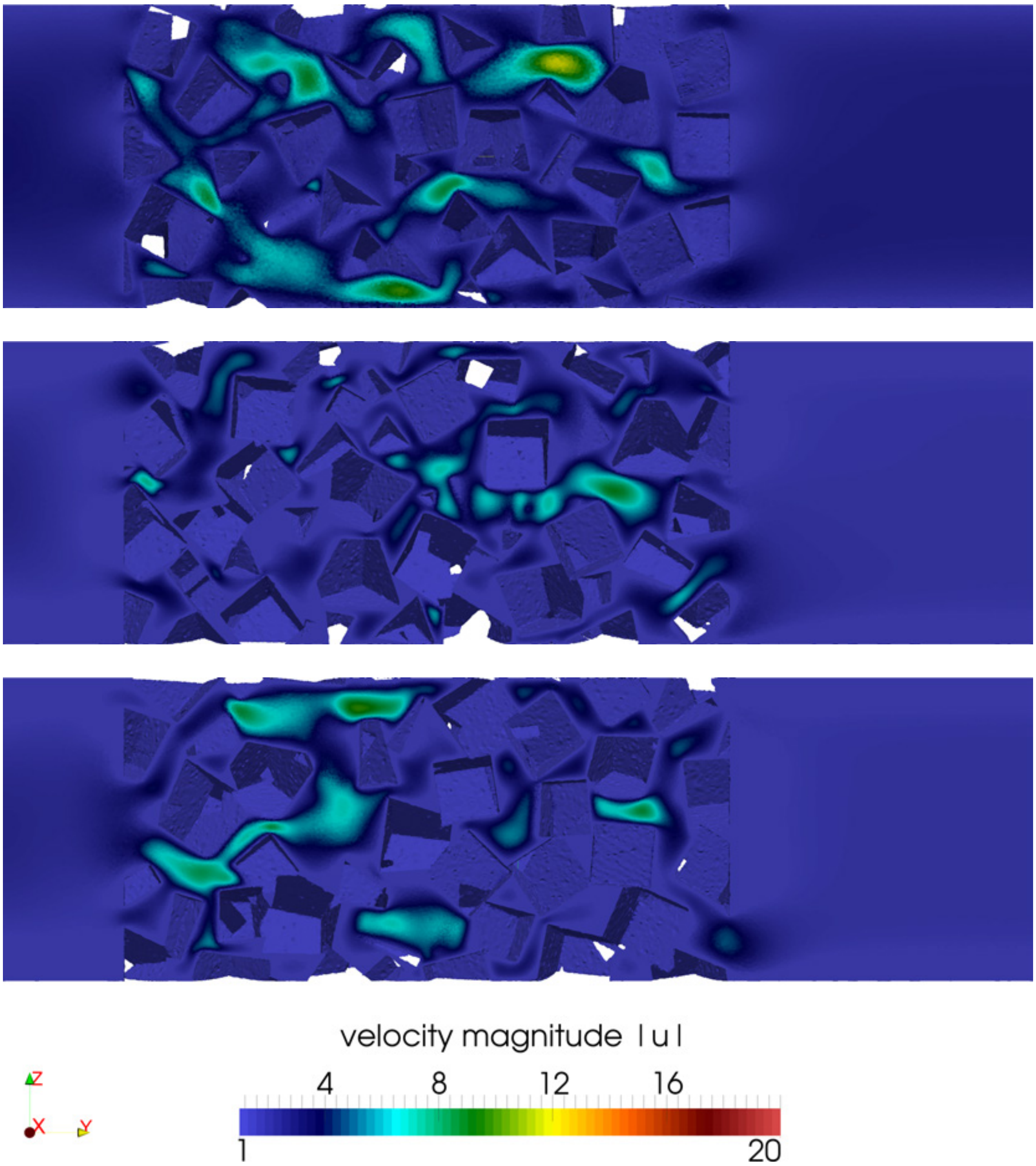


Figure 11.3 Longitudinal cut through a pipe filled with spherical grains. The three setups corresponds to the CG1, CG2, and CG3 setups. Fluid flow is from left to right. The velocity magnitude is shown with same colour scale for all setups, whereas the maximum velocity magnitudes are 11.8876, 11.7083, and 12.0186.

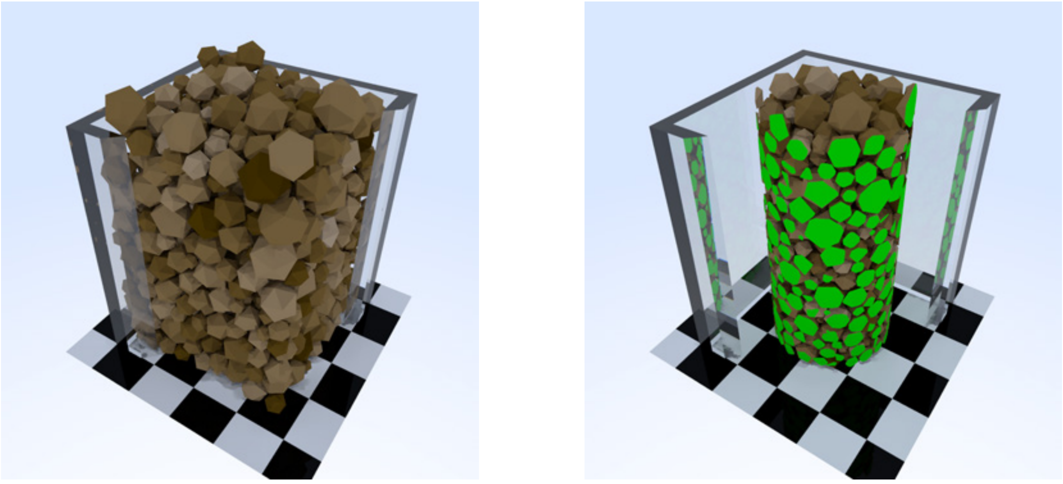


Figure 11.4 Example of a settlyDyn generated sediment consisting of icosahedrons, IG3 case, placed in a 5 × 5 box. There are 3576 grains at the simulation's end. For the fluid flow simulation a cylindrical stencil from the box's middle is taken.

- TetGen: "TetGen is a program to generate tetrahedral meshes of any 3D polyhedral domains."[Si13]
- CGAL "The Computational Geometry Algorithms Library, offers data structures and algorithms like triangulations (2D constrained triangulations, and Delaunay triangulations and periodic triangulations in 2D and 3D) . . . "[CGA]

11.3 Tetrahedral mesh generation from polyhedrons

Due to numerical inaccuracies (and the penalty algorithm used in the Bullet Physics Engine[Bul]) the sedimented grains are not only touching each other in some points (or lines, or planes) but are penetrating each other. While this is not significant for visualizations (or in games, for which the Bullet Physics Engine was developed), those penetrations lead to uncertainties in decisions whether a point is inside or outside of a grain.

A 3D Delaunay triangulation algorithm, which is able to work with multiple subdomains, with surface reconstruction implemented in the 3D triangulation library of CGAL[CGA] was used therefore. The detailed description of the algorithm is outside of the current scope, but the required interface is easy to understand and to implement.

In the CGAL's language an oracle, telling to which domain a given point belongs, is required. We implemented such an oracle which is assigning one of three possible subdomains to any point in space:

- A point may belong to the *outside* subdomain if it is outside of the domain boundary, which is a cylinder in the above simulations.
- The point belongs to the *grains* subdomain if it is inside any of the grains and not outside the domain boundary.
- Otherwise this point belongs to the *inside* subdomain.

There are also three boundaries to be reconstructed from the point information: outside-inside, inside-grain, and outside-grain.

The inside/outside test relies on number of triangle/random segment intersections, where the segment starts in a point to be tested and ends at a random point outside the objects' bounding box.

The result of the triangulation is written in CGAL's native format and contains both types of the subdomains, the *inside* and the *grains*. For the OpenFOAM simulation without fluid-structure interaction, only the *inside* domain is needed. An appropriate conversion tool was implemented.

11.4 An a-posteriori error driven mesh refinement

The tetrahedral mesh created with CGAL has relatively good mesh quality criteria.[b] For the fluid flow we want to apply mesh refinement in locations indicated by an *a-posteriori* error estimator to reduce the discretization error.

An *a-posteriori* error estimator uses current solution to estimate the discretization error. For the transient and steady-state fluid flows Hrvoje Jasak has presented several *a-posteriori* error estimators in the Chapter 4 of his PhD-thesis [Jas96, p. 153] in the context of OpenFOAM. We use the momentum error estimator for subsequent mesh refinement of those elements showing large discretization errors.

OpenFOAM does not support mesh refinement for tetrahedra natively. With help of the TetGen—a tetrahedral mesh generator [Si13]—it is possible to refine an existing tetrahedral mesh (generated by CGAL in the first place) providing the desired element size for every of the elements (in the coarse mesh). Such coupling was implemented as sketched in the following algorithm:

1) generation of initial tetrahedral mesh with CGAL including all important geometric features[c]
2) solution of the Navier-Stokes Equations and an *a-posteriori* error estimation
3) mesh conversion from OpenFOAM into a suitable input for TetGen[d]
4) mesh refinement with TetGen using maximum volume constraints provided by the error estimation
5) conversion of the refined mesh into OpenFOAM format (using OpenFOAM's tool `tetgenToFoam`)
6) mapping of the previous solution fields onto the new mesh
7) repeat from step 2) for steady-state simulations

---

[b] Due to highly irregular domain, especially in vicinity of contact points, a good (high quality of elements) 3D Delaunay triangulation is difficult to achieve, despite the surface smoothing in the surface reconstruction algorithm. Additionally the following mesh quality optimizers can be run: A Lloyd optimizer, an ODT optimizer, an perturber, and an exuder (in this order). (See CGAL's documentation for more details on mesh quality optimizers [CGA].)

[c] Further refinements do not change the geometry.

[d] This tool—called in the spirit of OpenFOAM naming convention `foamToTetgen`—is available as open-source code under the GNU Public License v.3 or later at [Naua].

Velocity magnitude distribution The range of the velocity magnitudes (measured in the pipe section filled with porous media), the mean of the velocity magnitude, and the variance are collected in the table to the right and the following Figure 11.5.

	Velocity magnitude		
	Maximum	Mean	Variance
CG1	11.8876	2.18767	3.09181
CG2	11.7083	2.51699	3.90339
CG3	12.0186	2.59228	4.44521
IG1	10.7398	2.18000	3.02657
IG2	13.9347	2.67052	4.53971
IG3	15.3164	2.73128	4.54077
SG1	10.9748	2.48718	3.16167
SG2	18.0944	3.05317	6.06938
SG3	17.5923	3.12331	6.71795

There, the dashed lines and numbers indicates the maximum values. The horizontal red lines represent the mean value, and the blue boxes are showing the standard deviation range around the mean value.

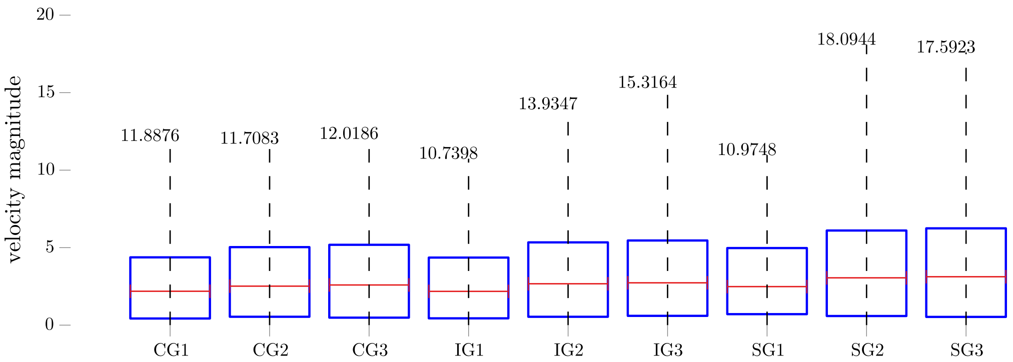


Figure 11.5 Statistics of velocity magnitude distribution in different setups. (See above text for description.)

Pressure drop and porosity The total pressure drop from the inlet to the outlet was measured. A small pressure drop in the empty inlet and outlet segments of the pipe is ignored. Since the pressure drop is proportional to the solid fraction (from Darcy's Law), both are presented in the following figure:

	Pressure drop	Porosity
CG1	27536.2799	0.5357
CG2	42987.9705	0.5103
CG3	35616.8278	0.5133
IG1	36934.1297	0.4965
IG2	84794.4983	0.4252
IG3	74529.7698	0.4033
SG1	45907.1389	0.5261
SG2	111233.3785	0.4670
SG3	110046.7641	0.4537

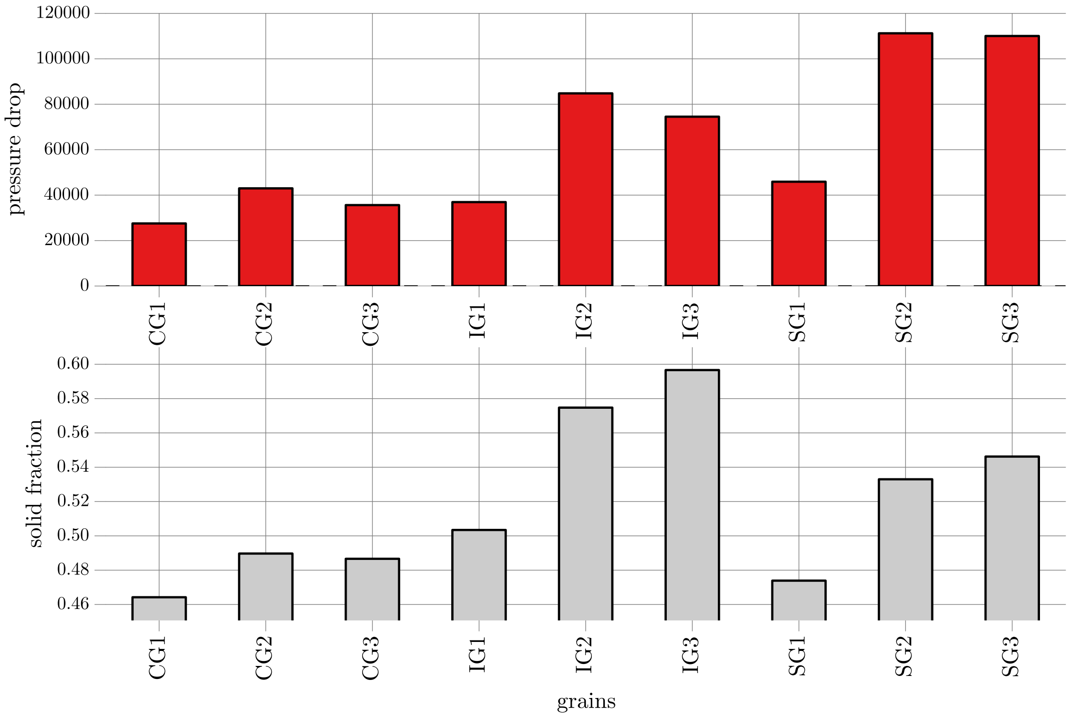


Figure 11.6 Pressure drop for one realization of different sediments: cubic, icosahedral, and spherical packing. 1—univariate grain size distribution (size 1), 2—bivariate distribution (sizes 1 and $^1/_2$), 3—Weibull distribution as described in Equation (15).

11.6 Summary

The most complex part of this simulation series is the mesh generation. From a given set of geometric objects (polyhedrons)—the unconsolidated sediment's grains—generated by settleDyn, an approximate surface reconstruction algorithm was used to create a tetrahedral mesh. The resulting mesh was then converted into the native OpenFOAM format.

For different settleDyn sediments the generated meshes were of different quality and most of them were numerically inadequate for fluid flow simulations; here we used those producing stable numerical results—not a good decision for statistical evaluation of sediments' properties.

An *a-posteriori* driven mesh refinement using TetGen was used showing good results and should be used in all further simulations.

Velocity magnitude and pressure drop statistics were presented. These do not allow a generalization to Kozeny-Carman or Ergun like equations because only one sample was analyzed.

12. Structure and development of OGS6 FEM code.

Development of the OpenGeoSys6 Finite Element Method code (ogs6) started in early 2012 as a successor of the OpenGeoSys5 version. Ogs5 has been successfully applied in various fields. An overview of the ogs5 applications is given in [KBB+12, KGSW12].

12.1 Shortcomings of ogs5

The currently implemented algorithms allow ogs5 to solve manifold numerical problems and these are mainly restricted by the available computational resources.

In some cases, however, application of a well established numerical algorithm requires some changes of the existing code. In case of small changes, like simulation specific equation of state formulation, the changes are localized and can be easily done. But when it comes to larger changes, reordering of algorithm's parts, implementation of an additional problem specific variable, or changes to the way of coupling between different subproblems, the required modifications to the code are scattered around many places. Some of the above changes might break the above mentioned benchmarks partially, and fulfilling both: the correct solution of the current problem, and the existing test cases is sometimes not possible.[a] [KGSW12]

The main problems with extension of the ogs5 source code are caused by its insufficient modularity.[b] This could be avoided, at least partially, by using existing algorithms from libraries like the Standard Template Library (STL) [ISO11], and writing more specialized, but reusable algorithms. The concerns of many about slower program execution are usually unsupported (in my experience), and should be checked with common profiling tools.

12.2 Localization of local assembler data

Another potential problem is the non-locality of data. Large data structures are often allocated peace-by-peace on the heap, increasing the memory fragmentation.

Optimization of data locality helps to improve CPU's cache usage (by reducing cache misses) [GSL+04], [Dre07]. Same observation about performance improvement with high data locality was made by Arnd Meyer in development of an adaptive FEM implementation [Mey14]. For the same reasons are arbitrary dimensional data arrays represented by contiguous arrays in the Visualization Toolkit [SML03][c]

---

[a] This leads to so called "branching" of the code base, where different developer groups have slightly different sources and cannot contribute to the main trunk.

[b] Lack of modularization leads to code replication with slight changes to that code instead of an algorithm's generalization. Quoting Bjarne Stroustrup on this: "Prefer algorithms to unstructured code" [Str12, p. 40]

[c] See the "Visualization Handbook" [HJ11], section 30.2.7 Data Representation, or the "Visualization Toolkit Handbook" [SML03].

12.3 Comparison of elliptic pde global matrix assembly routines

A comparison between three implementations for solving a Laplace equation on a 2D mesh consisting of 10^6 quad elements is performed.

Model The Laplace equation $\Delta u = 0$ is solved on quadratic domain $\Omega = [1000]^2 \in \mathbb{R}^2$. On left and right sides of the domain Dirichlet boundary conditions $u = -1$, $u \in \partial\Omega_{\text{left}}$, and $u = 1$, $u \in \partial\Omega_{\text{right}}$ are applied. On the remaining boundary, a homogeneous Neumann boundary condition is given.

Comparison The measurement is split in following parts: initialization and reading of the mesh, global assembly, and solution of the linear system of equations solver[d] Lis: a Library of Iterative Solvers for Linear Systems [LIS].

The first implementation for the comparison is ogs5 (version 5.4, SVN revision r12810). This version is modified to be comparable to the current ogs6 implementation. The modifications include: Removal of the gravity assembly part, the right-hand-side assembly for liquid flow processes, velocity term calculation, and multi-component flow-related computations.

The second and the third implementations are based on the current ogs6 implementation. Their difference is only in memory allocation located in the local assembler routine. One version is using dynamic size local assembler matrices and vectors; the memory is allocated at the run-time. The other is using fixed (or static) size matrices and vectors, whose sizes can be inferred using template meta-programming techniques [VJ03], thus resulting in memory allocation on the stack instead of on the heap.

For the local matrices and vectors an implementation from the Eigen3 library [GJ$^+$10] is used. This is one of the libraries which provides both types of memory allocation models with the same interface. For other libraries, see the following Section 12.4 Benchmarking matrix vector operations for FEM.

Results Using Google profiling tools [gpe], sampling data of each of the three implementations was collected. The results are given in the following Figure 12.1.

The largest difference between considered implementations is the global assembler part: for non-linear or transient simulations this routine is executed at every iteration/time step (besides of the solution of the linear system of equations). Therefore, decreasing its execution time was one of the main objectives for the code design.

An important difference to ogs5 is the creation of the degrees-of-freedom (dof) table, associating variables on meshes at integration points with global matrix positions. The positions in global matrix are calculated during each time step in ogs5, although it is not always necessary. For this reason, the dof-table is created only once in the new version, resulting in a further decrease of the global assembly's time.

The third result is the difference in runtime between ogs6 with dynamic matrices (used in local assembly) *vs.* fixed size matrices. Using dynamic size matrices results in memory allocation (`malloc`)

---

[d] Solution of the linear system of equations is done for sake of completeness and is not relevant here because we do not compare the performance of the particular solver.

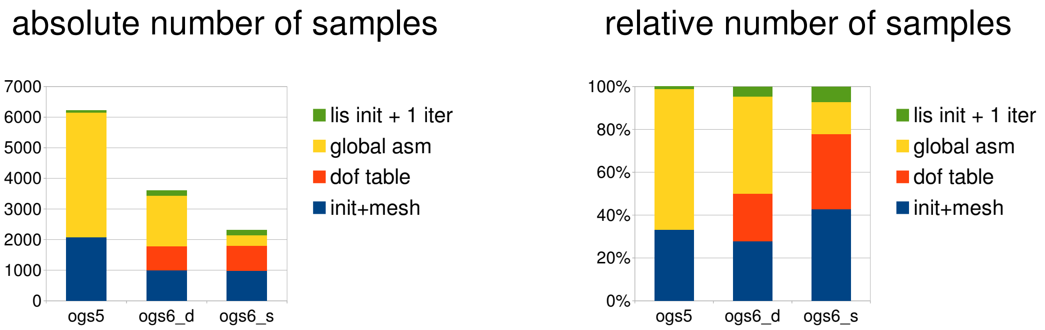


Figure 12.1 Absolute and relative number of samples split into comparable parts: Program initialization and reading of the mesh, degrees-of-freedom table construction (only in ogs6), global assembly routine, and solution of the linear system of equations. The `ogs6_d` and `ogs6_s` abbreviations stand for dynamic and fixed (static) matrices/vectors implementation in the ogs6.

calls every time the local assembler is called (*i.e.* for every element). When using fixed size matrices, the memory space is already allocated on stack, avoiding the extra memory allocation calls.

Notes It can be observed that the number of samples counted by Google profiling tools within linear system of equation solver call (Lis) is lower for `ogs5` than for `ogs6_d` and `ogs6_s`. This is not due to faster solver, but because of parallel execution of the solver using two processor cores in ogs5 implementation and one in ogs6 implementation. Using the same configuration for Lis would result in identical number of samples in all three test cases.

12.4 Benchmarking matrix vector operations for FEM

Solutions of systems of linear equations are ubiquitous in the Finite Element Methods. Two different matrix types are distinguished—dense, and sparse matrices, both of them require different solution algorithms. The sparse matrices are characterized by allocation of memory for non-zero elements only, contrary to dense matrices, where memory space for all matrix entries is allocated. There are two stages in the FEM where first dense matrices are used to compute elementwise contributions to a global matrix, requiring matrix/matrix and matrix/vector operations. And a second step, where the local contributions are added to the global matrix, which is usually of sparse type, and the right-hand-side, both forming a system of linear equations to be solved.

In this section we shall look into performance of different implementations (libraries) of linear algebra operations on vectors and matrices, especially for small size dense matrices required in global assembly of a linearized system.

Libraries One of the most widely spread class of libraries are known as BLAS—Basic Linear Algebra Subprograms—providing routines for basic matrix/vector operations (see [LHKK79] and references therein).

We are going to analyse two BLAS implementations ATLAS—Automatically Tuned Linear Algebra Software [hfd, WD98]—and Intel MKL—Math Kernel Library [Int]. These libraries provide efficient implementations of the matrix/vector operations divided into three levels known as BLAS Level 1, Level 2, and Level 3 for dynamic size matrices and vectors. Level 1 provides vector-vector operations, level 2—matrix-vector and level 3—matrix-matrix operations.

While the above libraries provide C and Fortran language interfaces, there are several libraries building on the aforementioned BLAS interface for the C++ language. One of such libraries is uBLAS [Ubl] included as part of the boost C++ libraries [Boo]. It is using expression template technique [VJ03] to map mathematical notation onto efficient implementations of matrix/vector operations, while trying to remove temporary objects usually arising when performing any chained operations. uBlas provides very stable interface and is part of every boost library release since last decade, but it also was not further developed since *ca.* year 2008 (see the FAQ section [Ubl] for details).

One of the newer alternatives to the uBLAS implementation is the blaze library [Bla]. It is using the uBLAS boost library and to achieve maximum performance an arbitrary library implementing the BLAS interface *e.g.* ATLAS. In the two papers [IHTR12b, IHTR12a] the authors describe the success of expression template as well as the limitations of the template metaprogramming.

Yet another library building on expression templates is Eigen, current version of which being 3 [GJ+10]. It supports dense, and sparse matrices with dynamic, and static size.

Performance measurement Here we select few types of operations relevant in FEM and compare libraries' performances with respect to this operations.

For performance measurements we use the Bench Template Library (BTL)[e] adopted in the Eigen3 framework [Eig11]. This performance measurement framework is updated to the needs of ogs6 development in two aspects. First we include new library interfaces, for example uBLAS or blaze, and second include new matrix/vector operations which corresponds better to the requirements of standard implementation of a FEM, especially in ogs6.

Common matrix/vector operations in FEM By default the BTL already includes benchmarking of the following operations relevant in FEM context; Let A and B be dense matrices, and u and v vectors.[f] Analyze following products:

$$AA, \qquad AA^t, \qquad Av \quad \text{and } A^t v.$$

Local matrix assembly uses further matrix/vector products:

$$v^t v, \quad v^t A v, \qquad A^t A, \quad A^t B A, \qquad v^t A, \quad v^t u^t A.$$

---

[e] originally developed by Laurent Plagne and distributed under the GPLv2 license. A copy was obtained from `https://bitbucket.org/spiros/btl`; accessed on 14 Jul. 2014

[f] Although there are sparse vector implementations, in this context only dense vectors make sense.

The above operations are written in a different way in the programming languages because they lack proper mathematical notations. The abbreviations used in the following figures are:

- `aat`: $A\,A^t$ – matrix with transposed matrix product.
- `atv`: $A^t\,v$ – transposed matrix with vector product.
- `axpby`: $A\,x + b\,y$ – matrix-vector product plus scalar times vector.
- `axpy`: $A\,x + y$ – matrix-vector product plus vector.
- `matrix_matrix`: $A\,A$ – matrix-matrix product.
- `matrix_vector`: $A\,v$ – matrix-vector product.

12.5 Summary

We have plotted the results for two different architectures are given graphically on the pages 65–68 at the end of this section. All graphs are showing performance measured in MFlops versus matrix/vector size.

Main trend of all benchmarks is showing some peak performance upon which the performance reaches a plateau or drops. All FEM relevant benchmarks involving vector multiplication are showing the latter behaviour. This is usually accounted on reaching the cache size limit.

Detailed evaluation of FEM relevant operations for small size ($\leq$ 100) matrices and vectors, shown in the Figure 12.5, splits the libraries into two main groups based on their overall performance. First and more performant group consists of ATLAS, Intel MKL and eigen3. Another is blaze and ublas, both of them building on boost libraries and linked BLAS implementation. Non-optimal compilation of boost and the linked BLAS library is probable cause for the worse performance.

None of the libraries in the first group is performing better on all architectures and for all benchmarks.

Benchmarks' result visualization The four figures on the following pages are visualizing benchmarking results.

- In the Figure 12.2 and Figure 12.3 all performed benchmarks for two different architectures are showing each library's performances in MFlops versus matrix/vector size.
- Selected benchmarks in Figure 12.4
- Small size benchmarks in Figure 12.5

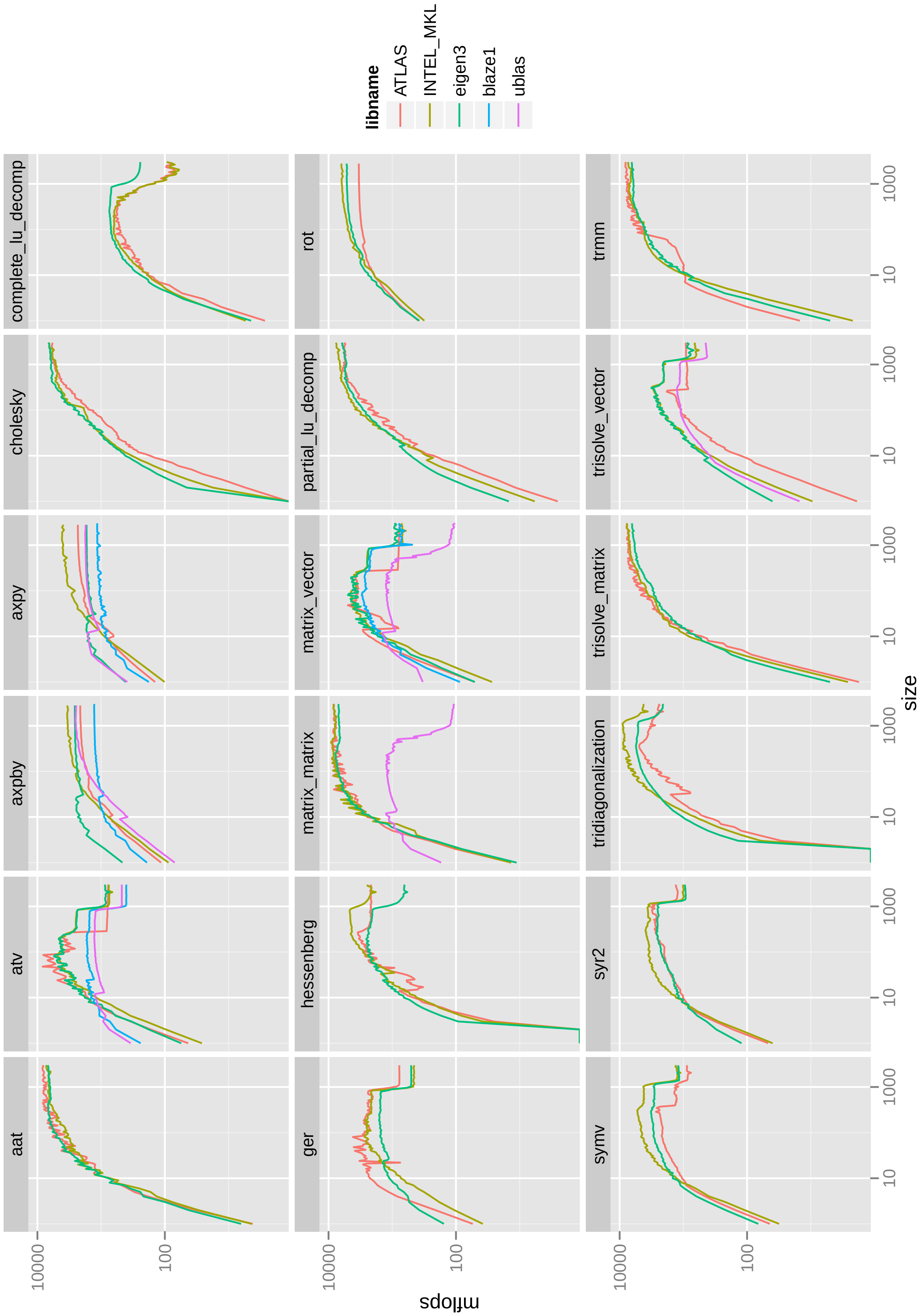


Figure 12.2 Performance of linear algebra libraries on an AMD system (Quad-Core AMD Opteron Processor 8384 @ 800 MHz).

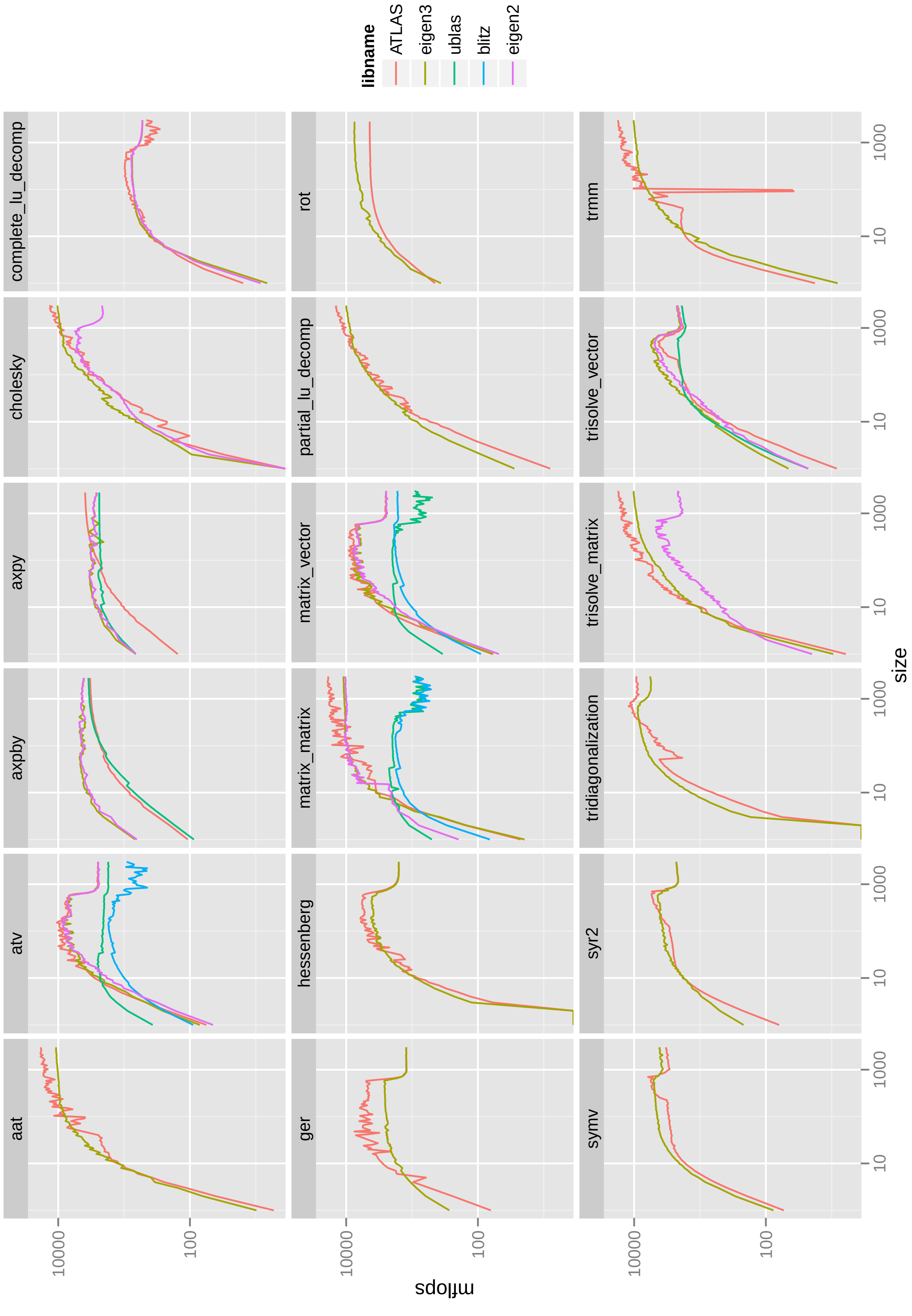


Figure 12.3 Performance of linear algebra libraries on an Intel system (Intel Core i5-2430M CPU @ 2.40 GHz).

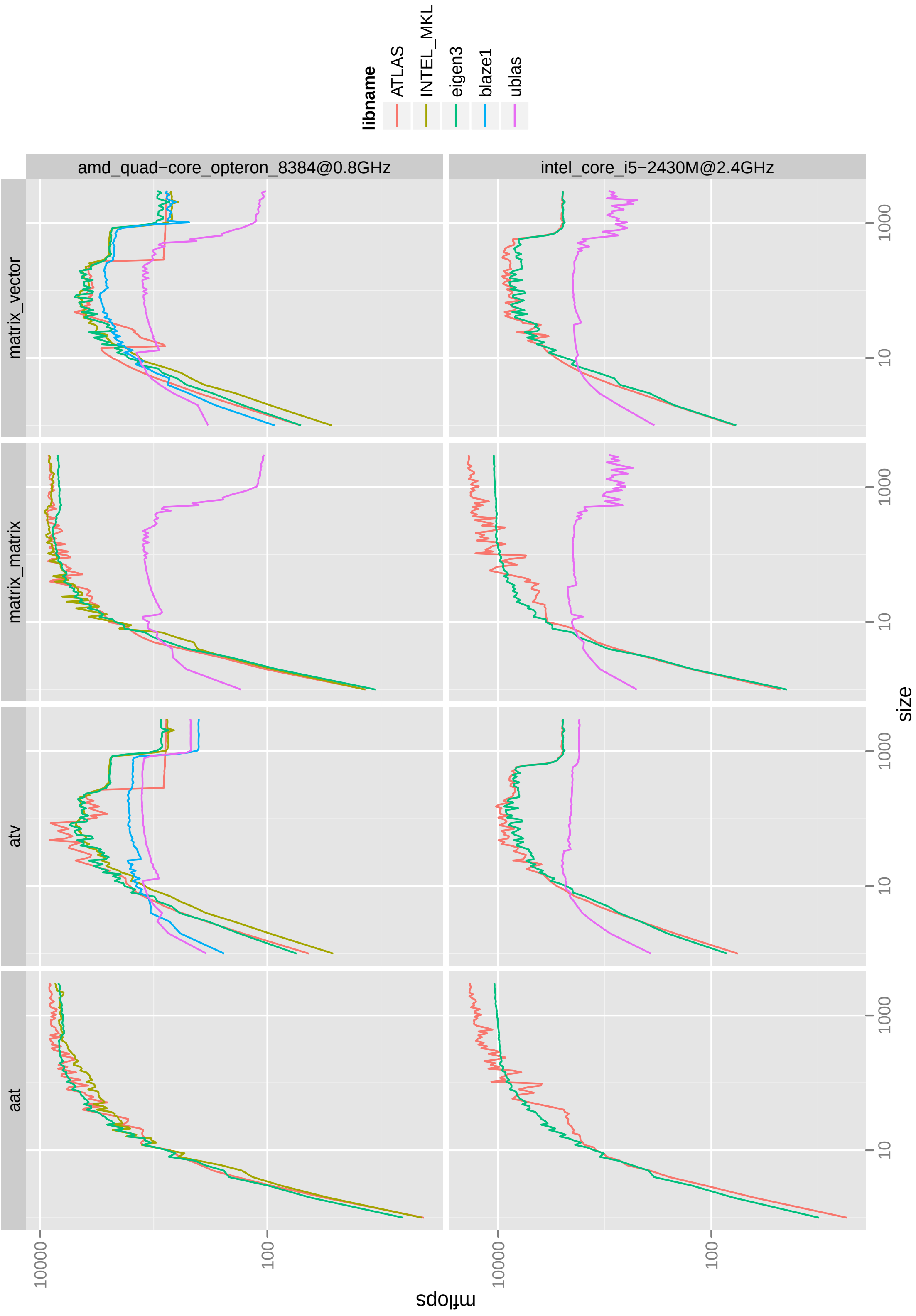


Figure 12.4 Selected benchmarks comparison of linear algebra libraries on an AMD and an Intel systems (Quad-Core AMD Opteron Processor 8384 @ 800 MHz and Intel Core i5-2430M CPU @ 2.40 GHz). Logarithmic scale; Matrix/vector size up to 1000.

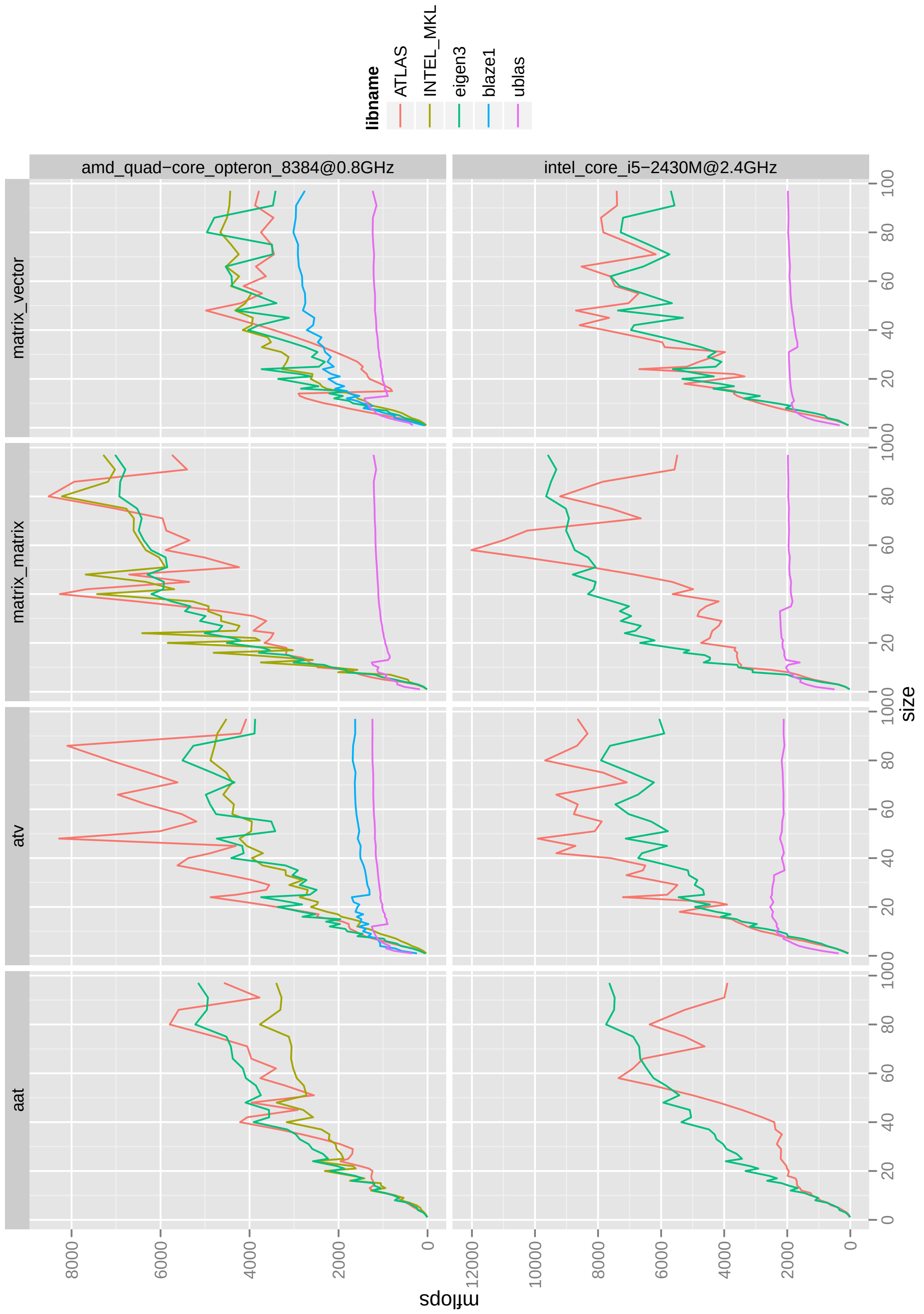


Figure 12.5 Selected benchmarks comparison of linear algebra libraries on an AMD and an Intel systems (Quad-Core AMD Opteron Processor 8384 @ 800 MHz and Intel Core i5-2430M CPU @ 2.40 GHz). Linear scale; Matrix/vector size up to 100.

13. Coupling to OpenFOAM

One of the motivating ideas for this work is embedding of pore-scale simulation results into the homogenised porous media simulations on the larger meso- and macroscales.

With advances in Computer Tomography availability of 3D scanned porous media samples increased: Scanners with resolutions around $5 \cdot 10^{-6}$m are readily available for small samples. The scans of bore-hole samples, for example, can be directly used for pore-scale simulations with OpenFOAM. This simulation results can be transferred onto the large scale problems solved with OpenGeoSys.

The coupling is one-way, from OpenFOAM to OpenGeoSys:

1) Calculation of porosity of the examined sample.
2) Calculation of (direction dependent) pressure drop Δp with `patchAverage` post-processing tool.[a]
3) Computation of hydraulic conductivity: the flow rate and the fluid's viscosity are known from the simulation input; the porosity and the pressure drop are calculated.[b]
4) The calculated homogenised properties are written into text-based OpenGeoSys input files.
5) If the properties are velocity/temperature dependent, the steps 2) to 4) are repeated. Interpolation is used OpenGeoSys's side.

The results of all OpenFOAM simulations are stored per-material—a unique identifier in OpenGeoSys.

With this the way from scanned or *in-silico* created porous media through determination of their hydro-mechanical properties to large-scale simulations is complete.

---

[a] See Section OpenFOAM, Post-processing on p. 18).

[b] Darcy's law or its extensions can be used.

Part four

Visualization

14. Overview

Visualizations, especially interactive 3D visualizations of CFD results can give further insights into the dynamic of fluid flow. This is in particular interesting in the case of complex geometries like porous media. There have been a lot of research of different visualization techniques, which became more and more computationally expensive recently.

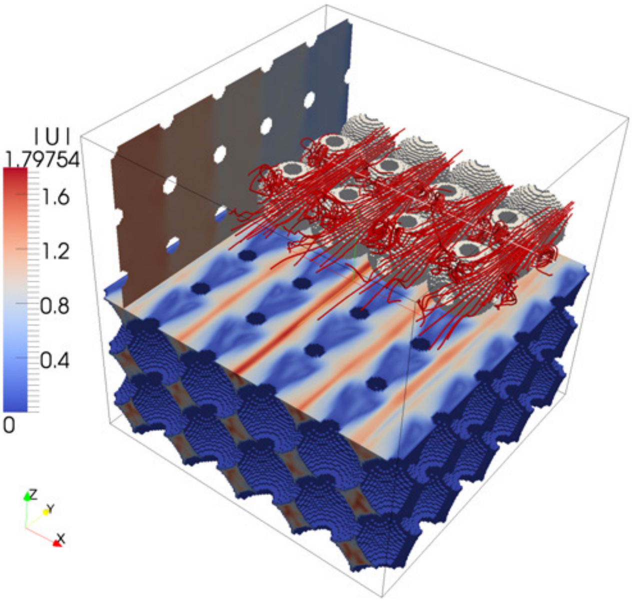


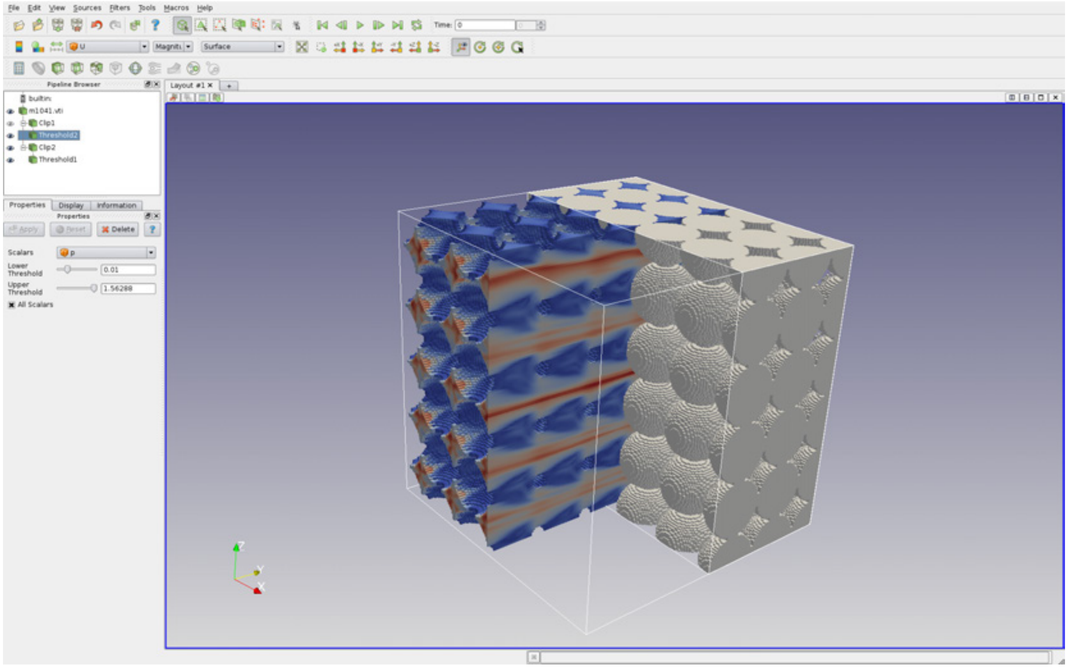
Figure 14.1 Visualization of fluid flow through simple cubic lattice of spheres using 3 different visualization techniques: Surface colouring, slice visualization and streamlines. The setup is described in the simulation part ('chap:sim') and is named `clsSymm_32_6_nu_0.00001`.

15. Visualization techniques

Visualization of fluid flow belongs to the post-processing phase and is usually carried out with specialized 3D visualization program. In this work we use widely accepted Visualization Toolkit (VTK) [SML03] and one of available frontends Paraview [Hen07].

With help of Paraview it is possible to create different kinds of VTK filter pipelines easily, resulting in specific visualization of one or another aspect of the fluid flow. Beside the interactive workflow of Paraview, it provides a reader for OpenFOAM simulations.

An example screenshot of a Paraview window showing an exported simulation visualization is shown in the following figure:



Scalars: p
Lower Threshold: 0.01
Upper Threshold: 1.56288
☒ All Scalars

Figure 15.1 Paraview is a VTK frontend. Main window with simulation snapshot shown: Visualization of turbulent flow in cubical sphere packing. This snapshot is a predcessor of the Figure 14.1.

Beside the Paraview program there is a Python VTK-interface allowing to write scripts in Python for more complex numerically intensive post-processing. There are time dependent and stationary visualization techniques. The time dependent visualizations, or animations, are showing field changes over time, or track some entities depending on time-varying fields, *e.g.* particle tracking and its path line visualization. The time independent techniques include for example streamlines or glyph visualizations.

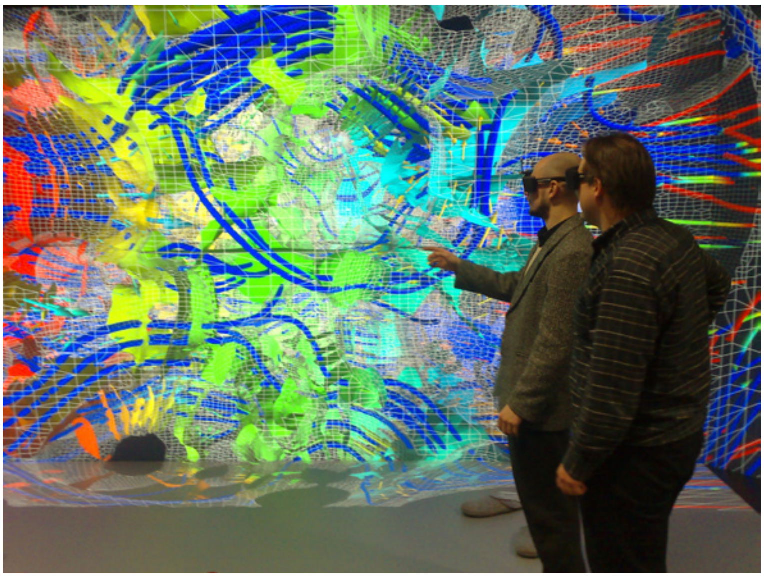


Figure 15.2 Interactive visualization in virtual reality environment.

16. 3D virtual reality laboratory

The Visualization Center at Helmholtz centre for environmental research – UFZ located at Department of Environmental Informatics is a virtual reality environment. It allows 3D and 4D visualizations of complex data sets, which are ubiquitous in geosciences. [http://www.ufz.de/index.php?en=14171].

Technology The "back projection-based stereoscopic visualization environment" is based on 13 SXGA+ projectors connected to a computer cluster consisting of 13 nodes each of them containing a high-end NVidia QuadroFX 5500 graphics adapter. An overview schematic is shown in Figure 'fig:vis:vislab_CMS'.

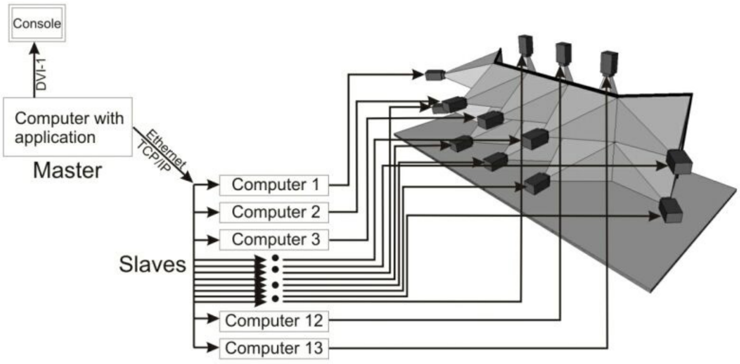


Figure 16.1 Visualization laboratory hardware setup schematic [http://www.ufz.de/index.php?en=14171] showing data flow from frontend computer to the physical screen arrangement.

17. Two-sided material rendering technique

This section reproduces the article “Rendering technique of multi-layered domain boundaries and its application to fluid flow in porous media visualizations” [NBK13].

17.1 Abstract

Current visualization techniques for computational fluid dynamics applications are sophisticated and work well in simple geometries. For complex geometries like pore spaces, multiple domain boundaries obstruct the view and make the studying of fluid flow fields difficult. To overcome these deficiencies, we use two-sided materials to render the domain boundaries.

Using this technique it is possible to place the camera inside the domain and have a non-obstructed view of the surrounding flow field without losing spatial reference to the domain boundaries. As a result, a larger part of fluid flow visualization is visible.

Two-sided material rendering was successfully applied to display still images with Blender Cycles renderer, in a virtual reality environment, and several implementation techniques were explored for using the Visualization Toolkit.

Keywords: 3D Graphics, Virtual reality, Image generation, VTK, OpenGL, VISLab TESSIN.

17.2 Introduction

Visualization of environmental systems is an emerging field in environmental science and technology. Understanding of those complex natural systems for both, applied questions, and basic research, through visualization is an important contribution to this field [TC05], [Dyk05], [BvL06], [KH13]. A key issue for environmental management purposes is data availability and integration from various sources for model set up and validation [RKK12], [RFSK13], [RBK14], as well as quantification and visualization of model uncertainties [WWM+10], [ZWK10], [HBR+14], [WBD+14]. This work is focussing on the latter—an application of a visualization technique for better understanding of flow processes in porous media on pore-scale.

There are many fluid flow or particle cloud visualization techniques giving impressive results when used in simple geometries, *i.e.* when the domain boundaries are not occluding the view (see [MLP+10] and references therein for a recent overview of visualization techniques). In the context of fluid flow simulations in random heterogeneous materials (like sandstones or packed beds), the fluid domain boundaries are not simple and the view inside of such a structure is obstructed. The obvious solution to make all of the boundaries transparent or semi-transparent does not solve the issue because the spatial reference to the domain's surface and its influence on the *local* fluid flow field is lost (in the transparent case) or difficult to imagine.

We apply a rendering technique to fluid flow visualizations in pore space geometries to gain deeper insights into the microscopic behaviour of fluid and particles in pore space. The technique is not limited to pore space geometries and is applicable to other visualizations with multi-layered domain boundaries.

In the following, we apply a two-sided material shading technique to fluid flow visualization in a packed bed domain. We also advocate the use of high-quality rendering software for all images because of improved depth perception through realistic shading and shadow mapping algorithms.

Before describing the two-sided material shading application, we briefly introduce the software used in this work to proceed from data input to final visualization. Next, we explain the idea of two-sided material shading followed by its realization using Blender [Ble13] and as well as in the visualization facility TESSIN VISLab [Zeh12].

17.3 Software and data processing

OpenFOAM for fluid flow simulations We solve the Navier-Stokes Equations for an incompressible Newtonian fluid in a pore space geometry. We use the OpenFOAM computational fluid dynamics solver `icoFoam` to compute a laminar, steady state solution. (OpenFOAM is an Open Source Computational Fluid Dynamics (CFD) Toolbox available at `http://www.openfoam.com/`. We used the current version 2.1.x in this work.) The pore space geometry has been derived from a CT-scan of an experimental setup described in [BLK+12] and [BKL+13] which is a packed bed of spherical pebbles. Their diameter is approximately 0.17 length units in the non-dimensional setup. We set the inlet velocity and the dynamic viscosity to unity.

Particle tracking We use an OpenFOAM solver called `icoUncoupledKinematicParcelFoam`. The particle cloud is injected at the inlet with a constant rate of 10^6 particles per second. The particles are modelled as spheres of a constant size (diameter 10^{-6} length units), not interacting with each other but sticking to the fluid domain's boundaries.

File formats We use the Visualization Toolkit VTK [SML03] as intermediate format for post-processing. OpenFOAM simulation results are converted to native VTK files or imported by Paraview, which is based on the VTK; [Hen07]. Files in the native VTK format are easily manipulated either via a Python (version 2) interface to VTK or by using Paraview. The processed data can then be exported into another 3D data format or an image.

Rendering with Blender Cycles Blender is a computer graphics software to create and postprocess 3D data and is usually used by 3D-artists. We use its powerful scene manipulation tools to set up lights and cameras and its interface to several rendering engines to create final images. The current Blender version is 2.70 with Blender Cycles as a rendering engine (also known as renderer). (Short description is available under `http://www.blender.org/development/release-logs/blender-261/` as " Blender Release Logs Version 2.61" published in December 2012.)

Visualization Toolkit VTK and Paraview Paraview is a GUI to the VTK library allowing creation of fast and interactive visualizations. Still more complex visualization scenarios can be implemented by directly using the VTK library's interface. The VTK's feature using OpenGL shaders greatly extends the range of possible visualizations.

The currently used version of VTK is 5.10, and the version of Paraview is 3.98.

Virtual reality environment To study complex models, we are using an interactive visualization environment shown in the Figure 17.2 at the Visualization Centre at the Helmholtz Centre for Environmental Research – UFZ. It combines two important features: the stereoscopic view and the interaction with the model being studied. Together with a user tracking system, it creates a feeling of being immersed in the visualization [Zeh12].

The hardware set-up of the TESSIN VISLab uses a back projection-based stereoscopic visualization environment with an approximately 6×3 meter large main screen. Alternating images for the left and the right eye are generated, and users wear special glasses which separate these images, resulting in stereoscopic view. An optical tracking system compensates for observer's movements so that correct perspective is maintained. A pointer device allows for interaction with the virtual environment. The rendering is performed on a cluster with 13 workstations (one for each projector).

17.4 Two-sided material rendering

Visualization of fluid flow in complex geometries is especially difficult if the fluid domain is hidden by boundaries. Simple boundaries can be rendered semi-transparent or in wire-frame, for example, without losing spatial reference to the domain boundaries. Domain boundaries appearing in multiple layers one after the other, like in heterogeneous porous media, however, occlude the view field and the main fluid flow visualization. Multiple semi-transparent boundaries are difficult to comprehend even in interactive 3D environments with stereoscopic view.

Showing both the fluid flow visualization and all domain boundaries simultaneously without decreasing clarity of the scene is difficult, but in the case of porous media, we are interested in *local* features of the flow. The local features are usually shown similar to Figure 17.8, where the camera is placed in the void space. In this case, the nearby boundaries hide large parts of the fluid flow visualization (velocity field shown as rendered cone glyphs in this case).

An improved view is shown in the Figure 17.9 in which the camera has been placed near the sphere shown in the middle/bottom of the previous image. Now, the velocity field around all of the nearby spheres is visible. The almost transparent domain boundary has a bluish tint when looking through it (visible on the right-hand side of the image) while the other boundaries are rendered in opaque white.

To create such images, we use a two-sided material, which makes domain boundaries transparent when looking from the inside of the domain (the sphere in the above image) but opaque when looking from the outside.

Realization in Blender and TESSIN VISLab In Blender, two-sided materials are created using the face normal direction $\mathbf{n}$ (together with the camera's pointing vector c). Results of two simple shaders—diffuse and transparent materials—are chosen depending on the sign of the scalar-product between the vectors $\mathbf{n}$ and $\mathbf{c}$ (compare with the Figure 17.1 below): A transparent shader when viewing

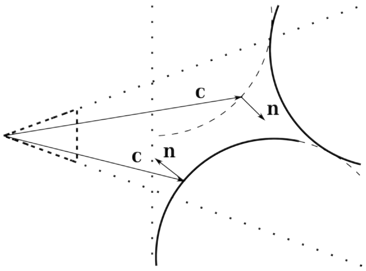


Figure 17.1 Two-sided material based on camera's pointing vector $\mathbb{C}$ and surface's normal vector $\mathbf{n}$. A camera on the left-hand side is shown as triangle with field of view (diagonal dotted lines) and its clipping plane (vertical dotted line). Surfaces (here as circles) are rendered transparent when viewed from inside (dashed line) and opaque when viewed from outside.

from the inside $\langle \mathbf{n}, \mathbf{c} \rangle \geq 0$ and a diffuse shader when viewed from the outside $\langle \mathbf{n}, \mathbf{c} \rangle < 0$.

In the TESSIN VISLab software, we use a feature known as face culling to achieve this effect. In contrast to Blender, the faces viewed from inside are not rendered at all using this software, which can be irritating in an interactive environment. A photograph in the Figure 17.2 shows the same pebble bed dataset. The viewer is looking from inside one of the solid pebbles. Instead of the velocity field, particles rendered as spherical glyphs sticking to the surface are shown.

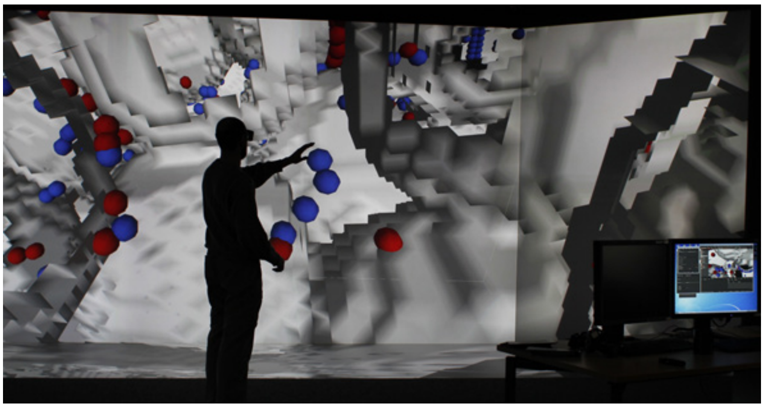


Figure 17.2 Interactive visualization in the Visualization Laboratory of domain boundaries and particles sticking to the surface. Using two-sided materials removes the first layer of the surface when looking from inside and allows studing of local properties.

Realization in VTK A very simple realization using the usual VTK pipeline is equivalent to that used in the TESSIN VISLab software: enabling backface culling in the `vtkActor`'s properties does not render the faces when viewing from the inner space. The advantage of this is very fast rendering of the scene because only one pass is required to display the geometry.

If we choose to indicate whether we are looking from inside or outside, tinting the back faces is a viable option. Extending the previous technique, we render, additionally to the opaque frontfaces, the backfaces only enabling frontface culling. For any camera position there is at most one semi-transparent surface (if the geometry's surface is an oriented 2-manifold); therefore there is no need for depth-sorted geometry.

The depth-sorted geometry (in the camera's view direction) is required as soon as there are more than one semi-transparent surfaces behind each other. One way to fulfil this is to depth-sort the input geometry using VTK's `vtkDepthSortPolyData` filter before rendering. This is a very slow, but (GPU) hardware-independent method. Another limitation is that the complete geometry must be composed into a single `vtkPolyData` object. The corresponding VTK pipeline for this scenario is:

$$\begin{aligned} & \text{vtkPolyData} \\ \rightarrow\ & \text{vtkDepthSortPolyData} \\ \rightarrow\ & \text{vtkPolyDataMapper} \\ \rightarrow\ & \text{vtkActor} \\ \rightarrow\ & \text{vtkRenderer} \end{aligned}$$

Realization in VTK using OpenGL Using of GPU hardware shaders written in the OpenGL Shading Language [SA10] is possible through VTK's API. Given a shader description in an XML file format, which links to the vertex and the fragment shaders, the `vtkActor`'s properties loads and enables the shader via the `LoadMaterial()` and `ShadingOn()` functions, creating a `vtkXMLMaterial` object. To create a two-sided material, it is sufficient to write a vertex and a fragment shaders. The relevant part of the fragment shader—after computing light contributions to the surfaces and possibly other things like shadows—is distinguishing between front and back faces based on the value of `gl_FrontFacing` build-in variable. This boolean variable (available only in the fragment shader) "is set to `TRUE` if the fragment is generated from a front-facing primitive, and `FALSE` otherwise" [SA10, p. 230].

Using the OpenGL Shading Language directly opens shading possibilities which are not (yet) available in VTK [SML03]. A more correct illumination model than Blinn-Phong shading model [Bli77] used by default in OpenGL [Shr10, Chapter "The Mathematics of Lighting", pp. 240-245], for example, could be implemented. Further rendering passes can be added, such as: shadow map pass, ambient occlusion, Gaussian blur, *etc.*, which can improve the scene perception. The advantages of the more realistic scene rendering are same as for Blender and are discussed in the conclusions.

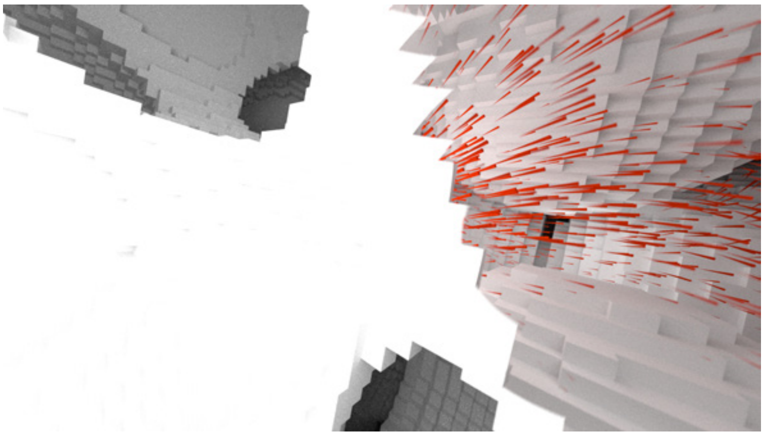


Figure 17.3 Rendering of fluid flow velocity field visualization with the camera placed inside of a grain. Large parts of the velocity field are hidden by the surface of the grain on the left side.

17.5 Photorealistic rendering

The previously described rendering methods are intended for interactive visualizations. Photorealistic rendering is currently only possible for still images only, unless a powerful rendering station is available for real-time processing.

Creation of photorealistic images is done using Blender's rendering engine, Blender Cycles. The precise control over a material's appearance opens many more possibilities for visualization.

In this section, we show the steps leading to the final image in the Figure 17.9, and discuss their improvements.

The camera's view when placed inside of a grain is obscured by an opaque surface, as shown in the left part in the Figure 17.3. Application of the two-sided material technique reveals large parts of a visualization (vector field in our case) but also adds some difficulties related to the "holes" created by the transparent surfaces (Figure 17.4).

Adding a wireframe rendering adds noise above a visualization (shown in the Figure 17.5). This happens when the surfaces consist of a dense mesh. For not so highly detailed surfaces, a wireframe (especially when rendered as tubes generating three-dimensional structure) is a viable option to semi-transparent surfaces creating a window-like view.

Limiting the field of view, when the backfaces of a surface are fully transparent, avoids the "holes" effect. This is achieved by rendering only the nearest surfaces using two-sided material, but those lying further away in the usual opaque style. the Figure 17.6 shows a mix between the opaque and two-sided material images.

For better distinction between the front-facing and back-facing surfaces, those viewed from inside can be additionally coloured (or in general, rendered) differently. This is shown in the Figure 17.7.

The result of the above changes, with additional green tint of the nearest surface viewed from inside, is rendered in the Figure 17.9.

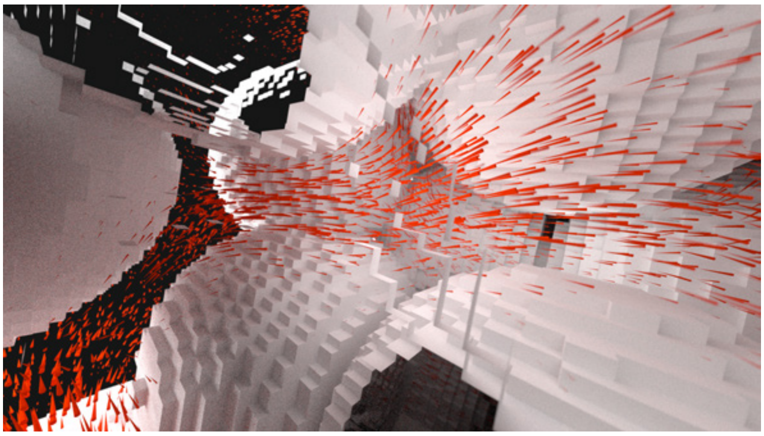


Figure 17.4 Large parts of the velocity field are visible through the transparent surface. Looking through multiple surfaces hinder the 3D perception of the scene.

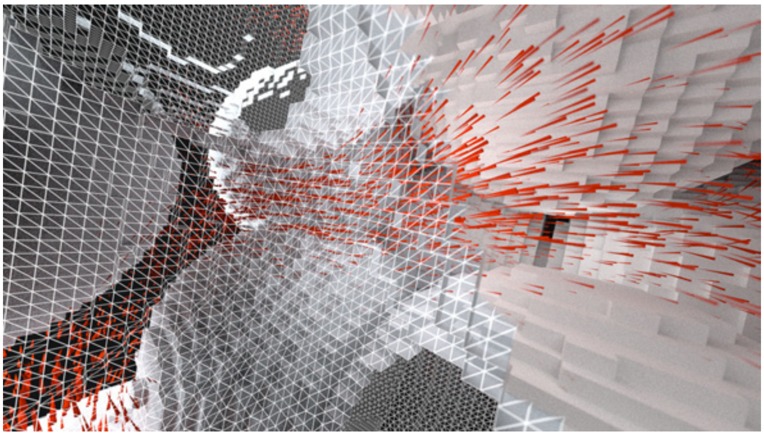


Figure 17.5 Dense surface's mesh rendered as a wireframe mar the vector field visualization, but can be a viable option for less-detailed surfaces.

17.6 Conclusions

In this work, we have presented a technique for visualizations in pore space geometries. Using two-sided materials, it is possible to retain a non-obscured view on common visualizations in complex domains (like streamlines for fluid flow simulations or glyphs for particle clouds). This technique is especially useful in interactive virtual reality environments where a user controls the camera's location and the viewing direction. It was used in the virtual reality TESSIN VISLab environment. Another application of this technique is to create high-resolution still images using the Blender software.

Using a global illumination renderer with physically correct shader and shadow mapping algorithms in Blender, we rendered high-resolution images showing a fluid flow visualization from the

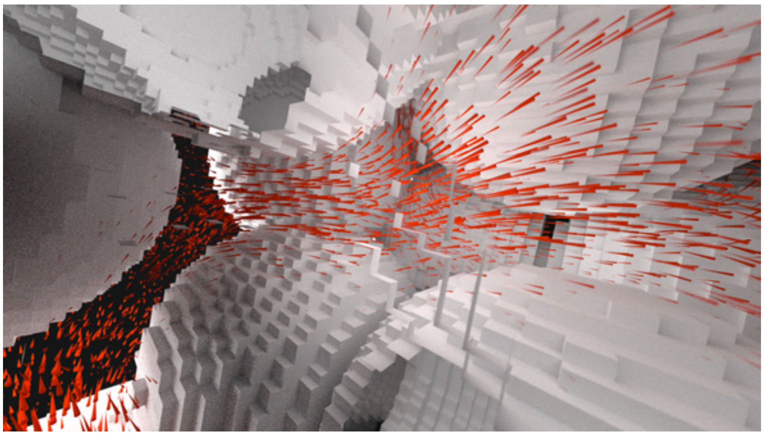


Figure 17.6 Application of the two-sided material is limited by the distance from the camera's position to a surface; Opaque material is used on the far surfaces.

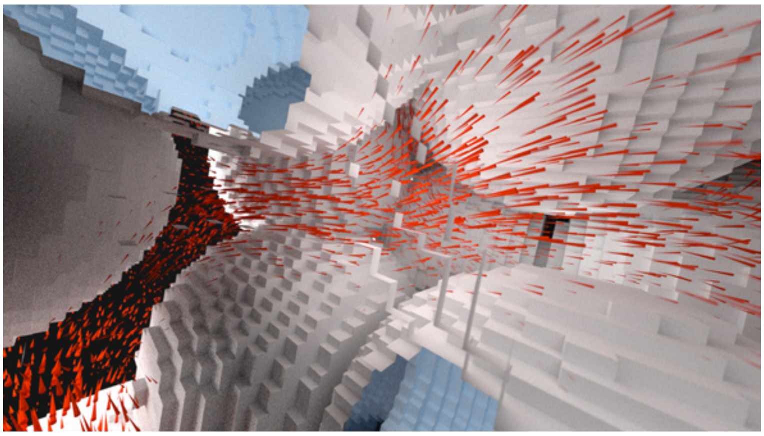


Figure 17.7 Additionally to the distance limited application of the two-sided material, the visible back-facing surfaces are coloured blue.

inside of an otherwise opaque domain boundary. Realistic shadowing and rendering of material surfaces helps to retrieve depth information and create volumetric scenes.

The rendering algorithms in Blender Cycles allow creation of almost photorealistic images; correct shading and multiple lights help to position scene elements easier in the 3D space. Near realistic rendering algorithms greatly improve the perceivability of complex scenes at the cost of increased demand for computational resources with higher rendering accuracy.

In attempt to close the gap between photorealistic rendering and real-time virtual environment, we explored several implementations based on VTK using its OpenGL Shading Language availability.

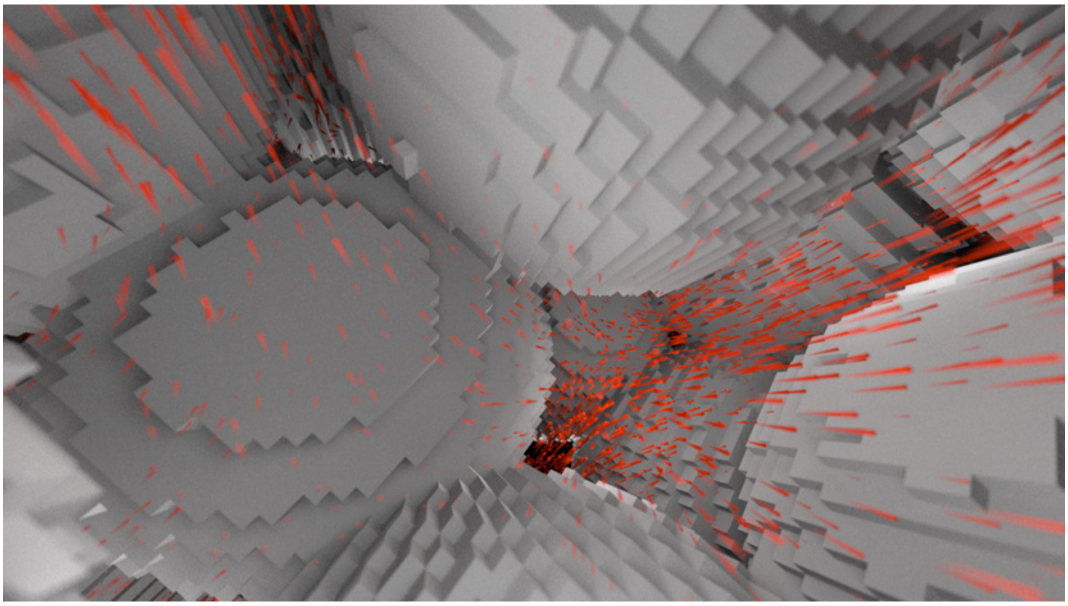


Figure 17.8 Rendering of fluid flow velocity field visualization with the camera placed in the void space. Large parts of the velocity field are hidden by the spheres.

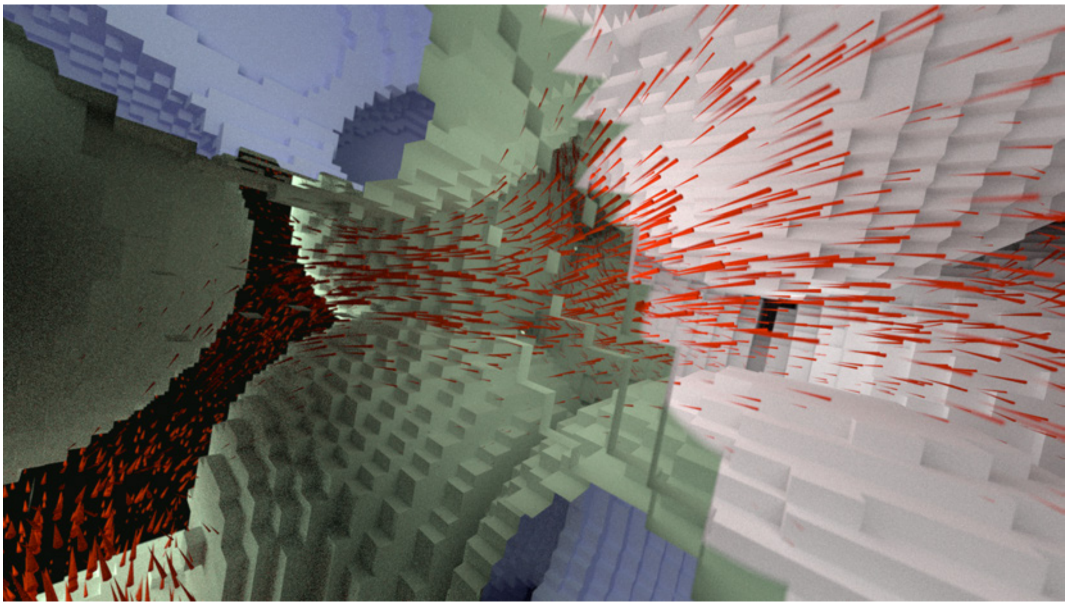


Figure 17.9 Rendering of the fluid flow velocity field visualization with the camera placed *inside* one of the spheres (middle/bottom one shown in the Figure 17.8). Larger parts of the fluid flow visualization are visible. The blue opaque surfaces render the faraway surfaces visible from the inside, where as the nearest surface is rendered transparent with green tint.

17.7 Future work

Implementation of OpenGL shaders opens visually improved real-time scene rendering possibilities. We would like to explore more of these techniques similar to those used in contemporary video game graphics [LJ02], [dSdOF+08], [SBS08], [BSKKY09], [SBS09], [LTDS+13].

Another aspect is the interaction of an actor with its virtual environment; we shall explore constraints given by rigid body simulation of the actor and the solid objects in the scene.

Acknowledgements The authors would like to thank Thomas Nagel, Carolin Helbig, Jens-Olaf Delfs and Norbert Böttcher, who provided valuable ideas to the writing of this paper, and Leslie Jakobs for proofreading.

The presented work has been partly funded by the German Federal Ministry of Education and Research (BMBF) as a part of the GEOTECHNOLOGIEN Program. The conceptual development as well as visualization methodologies of this work are also incorporated into the A-DuR (grant ID: 02E10588), CO2BENCH (grant ID: 03G0797D) and INFLUINS projects (grant ID: 03IS2091D), funded by BMBF.

Part five

Outlook

18. Porous media generation

In the first part of this work we dealt with generation of porous media. Besides the regular sphere packings, random *in-silico* artificial porous media were generated. It would be useful in near feature to analyse the packing densities of different grain size distributions of varying shapes using two-point correlation (or in general any higher order) statistics. Variing packing density in columns' centre to column's boundaries were expected but not analysed. This change in porosity along a cross section has direct influence on the fluid flow simulations and should be quantified.

The shapes of grains in an unconsolidated sediment also (as the grain grain size distribution) have strong influenco on the porosity. It would be interesting to compare real sediments for specific grains' shapes for verification of the generated porous media structures.

19. Sphere row in square duct

General simulation geometry was discussed in the Section Simulation setup p. 29, where we roughly estimated the possible limits for the Reynolds number and the computational domain size. A direct extension of the Direct Numerical Simulation (DNS) method would be an application of a Large-Eddies Simulation (LES) method for simulaiton of larger domains. The validity of the fluid-flow simulations using a turbulence model (like the LES) could be done by comparison to the already performed DNS.

The study of a sphere row in a square duct can be easily extended to a simple-cubic packing of spheres to study the influence of duct's walls, and the difference of the fluid flow in the centre of the filled duct to its boundaries.

The simple and small setup should also be used for comparison of tetrahedral and other meshing procedures.

It was expected that the development of back-flow pattern increase the pressure drop in a non-linear way; This did not happen and we have shown a linear dependence of the pressure drop versus the Reynolds number as predicted by the Darcy's law. This is indeed an interesting question.

20. Face-centred sphere packing

As the geometries become more complex, better mesh generation algorithms would greatly improve also the numerical solution. One could try to get tetrahedral meshes for example with TetGen or CGAL ([Si13, CGA]) with additional mesh optimization algorithms to reduce the non-orthogonality. Another available mesh generation code was recently implemented in OpenFOAM version 2.3.0. It is a "fully parallelised, meshing tool called foamyHexMesh. It is designed to generate hex-dominant meshes" [OFa]. producing hex-dominated meshes.

The effect of larger surface area has large influence on the numerical result of the simulation. Although the pressure drop for hexahedral meshes lies in the same range as for the snapped meshes, the overall effect was not studied; This is an important decision point for further simulations whether using of hexahedral meshes provides sufficient accuracy or the more numerically disadvantegeous "snapped" meshes are necessary.

The analysis of setups including a boundary layer lead to the question of adaptive mesh refinement; We have seen a significant influence of boundary layer discretization on the solution. Mesh refinement in other places may result in the more appropriate element size distribution for accurate simulation.

For further investigation of the back-flow pattern influence until significant deviation of the pressure drop *vs.* kinematic viscosity (as for the sphere row in square duct; Figure 9.5), higher mesh resolutions are required. (This is coupled with higher—proportionally to $\mathrm{Re}^{9/4}$—computational resources.) Alternatively a turbulence model can be used but it will need a verification benchmark.

21. Artificial sediments in a pipe

An analysis of the fluid flow in random media was impeded by the mesh generation procedure. For statistical analysis of random unconsolidated sediments a much more reliable meshing approach is required. Usage of three different meshing codes: CGAL, TetGen, and OpenFOAM converters are not feasible for automatic mesh generation. It is important not only to create a mesh for a given geometry, but a mesh suitable for numerical simulations: Most of the generated meshes lead to numerical instabilities of the solution.

22. OpenGeoSys and OpenFOAM coupling

The future workflow for analysis of hydro-thermo-mechanical properties of porous media can be done in a computer simulation: Starting from a CT-scan of a borehole sample, for example, resolving all of the effective porespace, a fluid flow pore-scale simulation calculates the (anisotropic) hydraulic conductivity of the sample. A couplet HTM simulation can give insights into sample's properties under load or high temperatures. An additianal advantage over laboratory experiments is the repeatability of the simulation, which is not always possible in an experiment in case of sample contamination.

23. Visualization

See the paper's conclusions section on page 82.

Part six

Appendix

References

[Bat99] George Keith Batchelor. *An introduction to fluid dynamics.* Cambridge University Press, Cambridge, U.K.; New York, NY, 1999.

[Bea88] Jacob Bear. *Dynamics of fluids in porous media.* Dover publications, Inc., New York, 1988.

[BKL+13] Thomas Barth, Johannes Kulenkampff, Mario Ludwig, S. Bras, Marion Gründig, K. Franke, Johanna Lippmann-Pipke, and Uwe Hampel. Study of particle deposition and resuspension in pebble beds using positron emission tomography. In *The 15th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH)*, 2013. Unpublished.

[Bla] blaze-lib: A high performance c++ math library. `https://code.google.com/p/blaze-lib`. Accessed: 14 Jul. 2014.

[Ble13] Blender Online Community. *Blender - a 3D modelling and rendering package.* Blender Foundation, Blender Institute, Amsterdam, 2013. Version 2.66.

[Bli77] James F. Blinn. Models of light reflection for computer synthesized pictures. In *Proc. 4th annual conference on computer graphics and interactive techniques*, pages 192–198, 1977.

[BLK+12] Thomas Barth, Mario Ludwig, Johannes Kulenkampff, Marion Gründig, Andr Bieberle, Uwe Hampel, and Johanna Lippmann-Pipke. Pet measurements of liquid aerosol particle deposition in pebble beds. In *6th International Topical Meeting on High Temperature Reactor Technology HTR2012*, 2012.

[Boo] Boost c++ libraries, version 1.55.0. `http://www.boost.org`. Accessed: 16 Jul. 2014.

[Böt14] Norbert Böttcher. *Thermodynamics of porous media: non-linear flow processes.* Doctoral thesis, Technische Universität Dresden, Fakultät Umweltwissenschaften, 2014.

[Bri49] H.C. Brinkman. A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles. *Applied Scientific Research*, 1(1):27–34, 1949.

[BSKKY09] Gloria J Brown-Simmons, Falko Kuester, Christopher JH Knox, and So Yamaoka. Earth and planetary system science game engine. In *Transactions on Edutainment II*, pages 203–218. Springer, 2009.

[Bul] Bullet physics engine. `http://www.bulletphysics.org`.

[BvL06] J Beurden and Ron van Lammeren. Linking land use modelling and 3D visualisation. *Innovations in design & . . .*, pages 1–17, 2006.

[BW95] Wilbur K. Brown and Kenneth H. Wohletz. Derivation of the weibull distribution based on physical principles and its connection to the rosin-rammler and lognormal distributions. *Journal of Applied Physics*, 78(4):2758 – 2763, 1995.

[BZ08] Guido Blöcher and Günter Zimmermann. Settle3D–a numerical generator for artificial porous media. *Computers & Geosciences*, 34(12):1827 – 1842, 2008.

[BZM+10] MG Blöcher, G Zimmermann, I Moeck, W Brandt, A Hassanzadegan, and F Magri. 3d numerical modeling of hydrothermal processes during the lifetime of a deep geothermal reservoir. *Geofluids*, 10(3):406–421, 2010.

[Car56] Philip C. Carman. *Flow of Gases Through Porous Media.* Academic Press, 1956.

[CGA] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[CLF28] Richard Courant, Hans Lewy, and Kurt Friedrichs. über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100:32 – 74, 1928.

[CM00] Alexandre Joel Chorin and Jerrold Eldon Marsden. *A Mathematical Introduction to Fluid Mechanics.* Springer, third edition, 2000.

[Dar56] Henry Darcy. *Les Fontaines Publiques de la Ville de Dijon.* Dalmont, Victor, Paris, 1856.

[Dre07] Ulrich Drepper. What every programmer should know about memory. http://www.akkadia.org/drepper/, 11 2007.

[dSdOF$^+$08] Selan Rodrigues dos Santos, Jauvane Cavalvante de Oliveira, Luciane Machado Fraga, Paulo Roberto Trenhago, and Silvano Maneck Malfatti. Using a rendering engine to support the development of immersive virtual reality applications. In *Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2008. VECIMS 2008. IEEE Conference on*, pages 74–79. IEEE, 2008.

[dV06] Eugene de Villiers. *The Potential of Large Eddy Simulation for the Modeling of Wall Bounded Flows.* PhD thesis, Department of Mechanical Engineering, Imperial College of Science, Technolgy and Medicine, 6 2006.

[DWK$^+$13] Jens-Olaf Delfs, Wenqing Wang, Thomas Kalbacher, AshokKumar Singh, and Olaf Kolditz. A coupled surface/subsurface flow model accounting for air entrapment and air pressure counterflow. *Environmental Earth Sciences*, 69(2):395–414, 2013.

[Dyk05] Jason Dykes. *Exploring Geovisualization International Cartographic Association.* Elsevier, 2005.

[Eig11] Eigen benchmarks. http://eigen.tuxfamily.org/index.php?title=Benchmark, 2011. Accessed: 13 Jul. 2014.

[EP88] Eric Eslinger and David R. Pevear. *Clay minerals for petroleum geologists and engineers.* SEPM, 1988.

[FP02] Joel Henry Ferziger and Milovan Peric. *Computational methods for fluid dynamics.* Springer, Berlin; New York, 2002.

[GJ$^+$10] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. Accessed: 13 Jul. 2014.

[GLR03] Alfredo Guardo, M Angels Larrayoz, and F Recasens. Regular packing types for cfd simulation of scf extraction and reaction equipment. In G. Brunner, I. Kikic, and M. Perrut, editors, *Proceedings of the* 6th *International Symposium on Supercritical Fluids*, pages 643–648. Institut National Polytechnique: Lorraine, France, 2003.

[gpe] gperftools: Fast, multi-threaded malloc() and nifty performance analysis tools. http://code.google.com/p/gperftools/. Accessed: 14 Feb. 2014.

[Gro00] Louis J. Gross. http://www.tiem.utk.edu/~gross/bioed/webmodules/spherepacking.htm, 9 2000. Accessed: 13 Nov. 2013.

[GSL+04] Xiaoyang Gao, S Sahoo, Qingda Lu, Gerald Baumgartner, J Ramanujam, C Lam, and P Sadayappan. Compiler techniques for efficient parallelization of out-of-core tensor contractions. Technical report, Technical Report OSU-CISRC-12/04-TR67, The Ohio State University, Columbus, OH, 2004.

[GVK97] M. Giese, D. Vortmeyer, and A. Krischke. Effektive viskositäten zur modellierung von strömungen in festbetten. *Chemie Ingenieur Technik*, 69(9):1309–1310, 1997.

[Has] The glorious glasgow haskell compilation system user's guide, version 7.8.3. http://www.haskell.org/ghc/docs/latest/html/users_guide/index.html. Accessed: 16 Jul. 2014.

[HBR+14] Carolin Helbig, Hans-Stefan Bauer, Karsten Rink, Volker Wulfmeyer, Michael Frank, and Olaf Kolditz. Concept and workflow for 3d visualization of atmospheric data in a virtual reality environment for analytical approaches. *Environmental Earth Sciences*, pages 1–14, 2014.

[Hen07] Amy Henderson. *ParaView Guide, A Parallel Visualization Application*. Kitware Inc., 2007. Version 3.98.

[hfd] See homepage for details. ATLAS homepage. http://math-atlas.sourceforge.net/. Accessed: 13 Jul. 2014.

[HJ11] C.D. Hansen and C.R. Johnson. *Visualization Handbook*. Elsevier Science, 2011.

[IHTR12a] Klaus Iglberger, Georg Hager, Jan Treibig, and Ulrich Rüde. Expression Templates Revisited: A Performance Analysis of Current Methodologies. *SIAM Journal on Scientific Computing*, 34(2):C42–C69, January 2012.

[IHTR12b] Klaus Iglberger, Georg Hager, Jan Treibig, and Ulrich Rude. High performance smart expression template math libraries. In *2012 International Conference on High Performance Computing & Simulation (HPCS)*, pages 367–373. IEEE, July 2012.

[Int] Intel(r) math kernel library 11.1 update 1 for linux. https://software.intel.com/en-us/intel-mkl. Accessed: 14 Feb. 2014.

[ISO11] ISO. ISO C++ standard. ISO 14882:2011(E), International Organization for Standardization, Geneva, Switzerland, 2011.

[Iss86] Raad I. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40–65, 1986.

[Jas96] Hervoje Jasak. *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. Doctoral thesis, Imperial College of Science, Technology and Medicine, 1996.

[Joh98] R.W. Johnson. *The Handbook of Fluid Dynamics*. A CRC handbook. CRC Press, 1998.

[KBB+12] Olaf Kolditz, Sebastian Bauer, Lars Bilke, Norbert Böttcher, Jens-Olaf Delfs, Thomas Fischer, Uwe J. Görke, Thomas Kalbacher, Georg Kosakowski, C.I. McDermott, C.H. Park, F. Radu, Karsten Rink, Haibing Shao, H.B. Shao, F. Sun, Yuan Yuan Sun, A.K. Singh, J. Taron, Marc Walther, Wenqing Wang, Norihiro Watanabe, Y. Wu, M. Xie, W. Xu, and Björn Zehner. Opengeosys: an open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical (thm/c) processes in porous media. *Environmental Earth Sciences*, 67(2):589–599, 2012.

[KGSW12] Olaf Kolditz, Uwe J. Görke, Haibing Shao, and Wenqing Wang. *Thermo-Hydro-Mechanical-Chemical Processes in Porous Media: Benchmarks and Examples.* Lecture Notes in Computational Science and Engineering. Springer, 2012.

[KH13] Johannes Kehrer and Helwig Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *Visualization and Computer Graphics*, 19(3):495–513, 2013.

[KKC$^+$10] T. Kempka, M. Kühn, H. Class, P. Frykman, A. Kopp, C.M. Nielsen, and P. Probst. Modelling of CO2 arrival time at ketzin part i. *International Journal of Greenhouse Gas Control*, 4(6):1007 – 1015, 2010. CO2 Storage at the EGU General Assembly 2009.

[Kol02] Olaf Kolditz. *Computational methods in environmental fluid mechanics.* Springer Science Publisher, Berlin, 2002.

[LHKK79] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for fortran usage. *ACM Trans. Math. Softw.*, 5(3):308–323, September 1979.

[LHZ05] Björn Alexander Legarth, Ernst Huenges, and Günter Zimmermann. Hydraulic fracturing in a sedimentary geothermal reservoir: Results and implications. *International Journal of Rock Mechanics and Mining Sciences*, 42(78):1028 – 1041, 2005. Rock Physics and Geomechanics Rock Physics and Geomechanics.

[LIS] Lis, a Library of Iterative Solvers for linear systems. http://www.ssisc.org/lis. Accessed: 13 Feb. 2014.

[LJ02] Michael Lewis and Jeffrey Jacobson. Game engines. *Communications of the ACM*, 45(1):27, 2002.

[LLSR66] L. D. Landau, E. M. Lifshitz, J. B. Sykes, and W. H. Reid. *Fluid Mechanics*, volume 6. Pergamon Press, Oxford, third edition, 1966.

[LTDS$^+$13] Zhihan Lv, Alex Tek, Franck Da Silva, Charly Empereur-mot, Matthieu Chavent, and Marc Baaden. Game on, science - how video game technology may help biologists tackle visualization challenges. *PLoS ONE*, 8(3):e57990, 03 2013.

[Mas02] Nader Masmoudi. Homogenization of the compressible navierstokes equations in a porous medium. *ESAIM: Control, Optimisation and Calculus of Variations*, 8:885–906, 1 2002.

[MBG94] Nicos S. Martys, Dale P. Bentz, and Edward J. Garboczi. Computer simulation study of the effective viscosity in brinkmans equation. *Physics of Fluids (1994-present)*, 6(4):1434–1439, 1994.

[Mey14] Arndt Meyer. Programmbeschreibung SPC-PM3-AdH-XX. Technical report, Professuren für Numerische und Angewandte Mathematik an der Fakultät für Mathematik der Technischen Universität Chemnitz, 1 2014. In German.

[MKDMS14] Barbara Mette, Henner Kerskes, Harald Drück, and Hans Müller-Steinhagen. Experimental and numerical investigations on the water vapor adsorption isotherms and kinetics of binderless zeolite 13x. *International Journal of Heat and Mass Transfer*, 71(Complete):555–561, 2014.

[MLP$^+$10] Tony McLoughlin, Robert S Laramee, Ronald Peikert, Frits H Post, and Min Chen. Over two decades of integration-based, geometric flow visualization. In *Computer Graphics Forum*, volume 29, pages 1807–1829. Wiley Online Library, 2010.

[MSKM08] Harald Milsch, Erik Spangenberg, Johannes Kulenkampff, and Steffi Meyhöfer. A new apparatus for long-term petrophysical investigations on geothermal reservoir rocks at simulated in-situ conditions. *Transport in Porous Media*, 74(1):73–85, 2008.

[Naua] Dmitri Naumov. Openfoam to tetgen mesh conversion tool. Available upon request.

[Naub] Dmitri Naumov. Settle dynamics—a sedimentation process simulator. `http://www.naumov.de/settle3D`.

[NBK13] Dmitri Naumov, Lars Bilke, and Olaf Kolditz. Rendering technique of multi-layered domain boundaries and its application to fluid flow in porous media visualizations. In *Workshop on Visualisation in Environmental Sciences (EnvirVis)*, volume 1, pages 43–46. The Eurographics Association, Springer, 2013.

[OFa] foamyHexMesh: a hex-dominated OpenFOAM mesh generator. `http://www.openfoam.org/version2.3.0/foamyHexMesh.php`. Accessed: 16 Jul. 2014.

[OFb] OpenFOAM. `http://www.openfoam.org`.

[OR10] William L. Oberkampf and Christopher J. Roy. *Verification and Validation in Scientific Computing*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[Pat80] Suhas Patankar. *Numerical Heat Transfer and Fluid Flow*. Series in computational methods in mechanics and thermal sciences. Taylor & Francis, 1980.

[Pop00] Stephan B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[Pra61] Stephen Prager. Viscous flow through porous media. *Physics of Fluids (1958-1988)*, 4(12):1477–1482, 12 1961.

[RBK14] Karsten Rink, Lars Bilke, and Olaf Kolditz. Visualisation strategies for environmental modelling data. *Environmental Earth Sciences*, pages 1–12, 2014.

[RFSK13] Karsten Rink, Thomas Fischer, Benny Selle, and Olaf Kolditz. A data exploration framework for validation and setup of hydrological models. *Environmental Earth Sciences*, 69(2):469–477, 2013.

[RKK12] Karsten Rink, Thomas Kalbacher, and Olaf Kolditz. Visual data exploration for hydrological analysis. *Environmental Earth Sciences*, 65(5):1395–1403, 2012.

[RR33] P. Rosin and E. Rammler. The laws governing the fineness of powdered coal. *Journal of the Institute Fuel*, 7(31):29 – 36, 1933.

[SA10] Mark Segal and Kurt Akeley. The OpenGL Graphics System: A Specification (Version 4.0 (Core Profile) - March 11, 2010), 2010.

[SBS08] Ifan DH Shepherd and ID Bleasdale-Shepherd. Towards effective interaction in 3d data visualizations: what can we learn from videogames technology. In *International Conference on Virtual Geographic Worlds, Hong Kong*, pages 7–8, 2008.

[SBS09] Ifan DH Shepherd and Iestyn D Bleasdale-Shepherd. Videogames: the new GIS? 2009.

[Shr10] Dave Shreiner. *OpenGL programming guide : The official guide to learning OpenGL, version 3.0 and 3.1.* Pearson Education, Inc., 7th edition, 2010.

[Si13] Hang Si. TetGen. A quality tetrahedral mesh generator and 3d delaunay triangulator. Technical Report 13, WIAS—Weierstrass Institute for Applied Analysis and Stochastics Leibniz Institute in Forschungsverbund Berlin e.V., 2013. http://www.tetgen.org.

[SML03] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit: An Object-Oriented Approach To 3D Graphics, Third Edition.* Kitware Inc., 2003.

[SP80] Enrique Sánchez-Palencia. *Non-homogeneous media and vibration theory*, volume 127 of *Lecture notes in physics.* Springer-Verlag, 1980.

[SPSK05] S.K. Sahoo, R. Panuganti, P. Sadayappan, and S. Krishnamoorthy. Cache miss characterization and data locality optimization for imperfectly nested loops on shared memory multiprocessors. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 44a–44a, April 2005.

[Str12] Bjarne Stroustrup. Foundations of c++. In *21st European Symposium on Programming (ESOP)*, Tallinn, Estonia, March 2012.

[Tar80] Luc Tartar. *Incompressible Fluid Flow in a Porous Medium. Convergence of the Homogenization Process.*, chapter Appendix, page 368. Volume 127 of *Lecture notes in physics* [SP80], 1980.

[TC05] JJ Thomas and KA Cook. *Illuminating the path: The research and development agenda for visual analytics.* National Visualization and Analytics Ctr, 2005.

[Tic14] Ian Tice. From stokes flow to darcy's law. Lecture notes, 2014.

[Tor01] Salvatore Torquato. *Random heterogeneous materials: microstructure and macroscopic properties.* Springer Verlag, 2001.

[TSM+13] Nico Trauth, Christian Schmidt, Uli Maier, Michael Vieweg, and Jan H. Fleckenstein. Coupled 3-d stream flow and hyporheic flow model under varying stream and ambient groundwater flow conditions in a pool-riffle system. *Water Resources Research*, 49(9):5834–5850, 2013.

[Ubl] uBLAS : Basic linear algebra library, part of [Boo]. http://www.boost.org/doc/libs/1_55_0/libs/numeric/ublas. Accessed: 14 Jul. 2014.

[VJ03] David Vandevoorde and Nicolai M Josuttis. *C++ templates : the complete guide.* Addison-Wesley, Boston [etc.], 2003.

[Vog02] Hans-Jrg Vogel. Topological characterization of porous media. In Klaus Mecke and Dietrich Stoyan, editors, *Morphology of Condensed Matter*, volume 600 of *Lecture Notes in Physics*, pages 75–92. Springer Berlin Heidelberg, 2002.

[WBD+14] Marc Walther, Lars Bilke, Jens-Olaf Delfs, Thomas Graf, Jens Grundmann, Olaf Kolditz, and Rudolf Liedl. Assessing the saltwater remediation potential of a three-dimensional, heterogeneous, coastal aquifer system. *Environmental Earth Sciences*, pages 1–11, 2014.

[WD98] R. Clint Whaley and Jack Dongarra. Automatically Tuned Linear Algebra Software. In *Proceedings of the IEEEACM SC98 Conference*, pages 1–27, 1998.

[Wei51] Waloddi Weibull. A statistical distribution function of wide applicability. *J. Appl. Mech.-Trans. ASME*, 18(3):293 – 297, 1951.

[Whi96] Stephen Whitaker. The forchheimer equation: A theoretical development. *Transport in Porous Media*, 25(1):27–61, 1996.

[WS97] Morten M.T. Wang and Tony W.H. Sheu. Implementation of a free boundary condition to navier-stokes equations. *International Journal of Numerical Methods for Heat & Fluid Flow*, 7(1):95–111, 1997.

[WWF+13] Li Wah Wong, Norihiro Watanabe, Sven Fuchs, Klaus Bauer, Mauro Cacace, Guido Blöcher, Oliver Kastner, and Günter Zimmermann. Sensitivity analysis of impacts of natural internal fault zones and well design on fluid flow and heat transfer in a deep geothermal reservoir. In *EGU General Assembly Conference Abstracts*, volume 15, page 8572, 2013.

[WWM+10] Norihiro Watanabe, Wenqing Wang, Christopher I. McDermott, Takeo Taniguchi, and Olaf Kolditz. Uncertainty analysis of thermo-hydro-mechanical coupled processes in heterogeneous porous media. *Computational Mechanics*, 45(4):263–280, 2010.

[Zeh12] Björn Zehner. Scientific 3d visualization—representing complex data sets in a comprehensive way. In *UFZ-Bericht 6/2012*, page 19, Leipzig, April 2012. Helmholtz-Zentrum für Umweltforschung GmbH - UFZ.

[ZWK10] Björn Zehner, Norihiro Watanabe, and Olaf Kolditz. Visualization of gridded scalar data with uncertainty in geosciences. *Computers & Geosciences*, 36(10):1268 – 1275, 2010.