ON EFFICIENT ALGORITHMS FOR FILTRATION RELATED MULTISCALE PROBLEMS

Zahra Lakdawala

Vom Fachbereich Mathematik der Technische Universität Kaiserslautern zur Verleihung des akademischen Grades *Doktor der Naturwissenschaften (Doctor rerum naturalium, Dr. rer. nat.)* genehmigte Dissertation.

1. Gutachter: Prof. Dr. Oleg Iliev
2. Gutachter: Prof. Patrick Jenny

Datum der Disputation: 26.02.2010

D386

Acknowledgements

It is always difficult to express gratitude that would do justice to the contributions that numerous individuals have made to one's academic career. Nonetheless, I will attempt to do so with the realization that words alone are insufficient to convey my appreciation to all the individuals mentioned below.

First and foremost, I would like to express my sincere and profound gratitude to my supervisor, Professor Oleg Iliev. This thesis would not have been possible without his constant support and patient mentoring. I am indebted to him for his invaluable encouragement during the entire course of my PhD.

Moreover, I would also like to acknowledge the contributions of Professor Patrick Jenny, Dr. Guiseppe Bonfigili, Professor Raimondas Čiegis, Dr. Vadimas Starikovicius and Dr. Peter Popov. This work would be incomplete without the many fruitful discussions and collaborative exercises that I had with them.

Additionally, I would like to thank the Department of Flow and Material Simulations at Fraunhofer ITWM, Department of Mathematics at the Technical University of Kaiserslautern and the Institute of Fluid Dynamics at ETH Zurich. In this respect, I am indebted to Dr. Andreas Wiegmann and Dr. Stefan Rief for their support and the enriching discussions that I had with them. Thanks are also due to Sebastian for livening up our shared office with his sense of humor and creating an environment where I was able to work productively.

Furthermore, I would also like to express my appreciation to the Graduiertenkolleg and Fraunhofer ITWM for their financial support. My work at ITWM would not have been possible without their generous assistance.

Lastly, I would like to acknowledge the support of my family. Thanks are due to my mother and my two brothers for their constant support. Also, my work would not have been possible without the patient and comforting encouragement of my husband, Ali Raza. Important too is the memory of my late father. Though he may not be here, his memory lives on and sustains us all, and it is to him that this thesis is dedicated.

Contents

Preface

Industry is increasingly relying on computer aided engineering for the development of novel technologies. This is true for different applications such as the design of oil filters in the automotive industry, soot filtration in diesel filters, air filtration etc. The complex procedure of designing, testing and optimizing new processes and products usually has to be carried out in an iterative and an experimental fashion by means of time consuming trial and error steps with expensive prototypes. In order to reduce the design time and production costs, computer simulation thus gains a growing importance. Nowadays, simulation is not able to replace the experimental process entirely, but is able to assist in such a way that principal design decisions can be made faster and less prototypes are needed. Figure 1 shows a particular example of the extent to which simulation codes and numerical algorithms can benefit the industry.

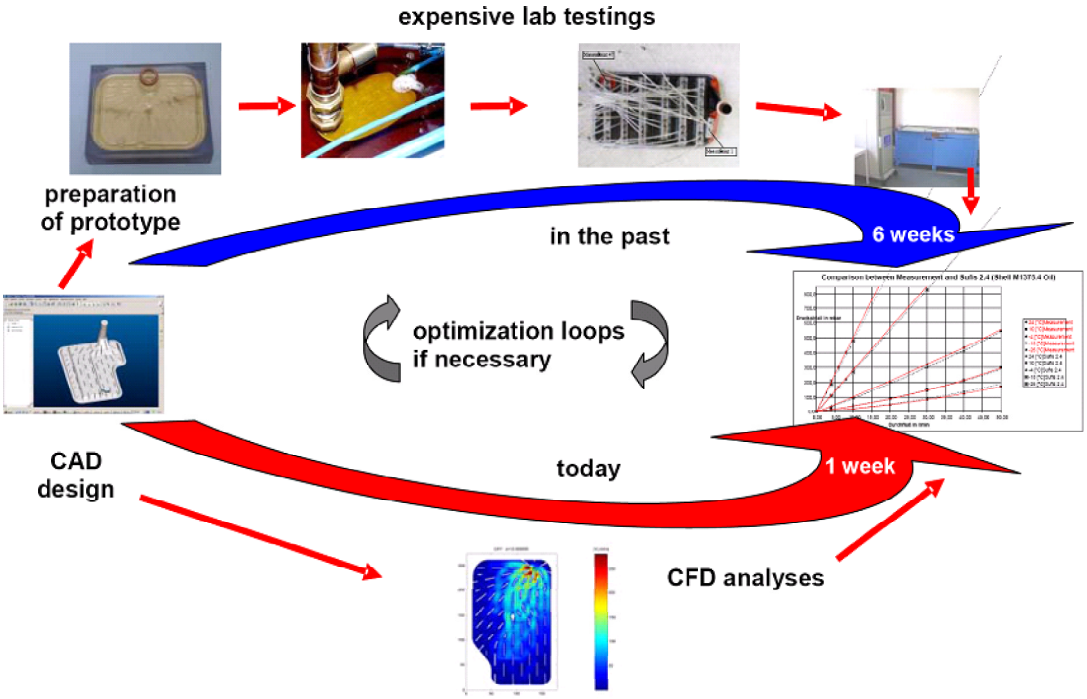


Figure 1: Design process of oil filters. Courtesy: IBS Filtran

This thesis deals with the numerical study of multiscale problems arising in the model-

ing of processes of the flow of fluid in plain and porous media. On the pore level, the porous media is represented as a connected domain of pore space filled with the fluid. The fluid flows through the network of pores and channels, as shown in Figure 2. The most commonly used porous media, such as soil, sand, ceramics, foam, rubber etc. have very complicated solid matrix structures, for which the exact configuration of the micro structure is not exactly known. Due to the heterogeneity of porous material at the pore and particle level, it is very expensive to predict/observe the flow at that scale. In most cases, this is not needed as well. The properties of interest exist at a much coarser scale but are influenced by the processes at the pore level scale. Therefore, even though the equations are principally described at the microscopic level, the description and solution of the transport problem becomes computationally very expensive. Additionally, superflous information is often needed for smoothing and obtaining useful large scale averaged data which can be verified at the measurement/experimental level. Hence, the porous media description is sought at a coarser scale by the means of effective material properties. These properties can be measured and problems can be solved on the macroscopic scale (cf. [88], [2]). The macroscopic formulation has a cost attached to it, such that it hides some small heterogeneities which are important for the microscopic formulation. Nonetheless, due to solvability issues, description at the macroscopic level merits further examination.

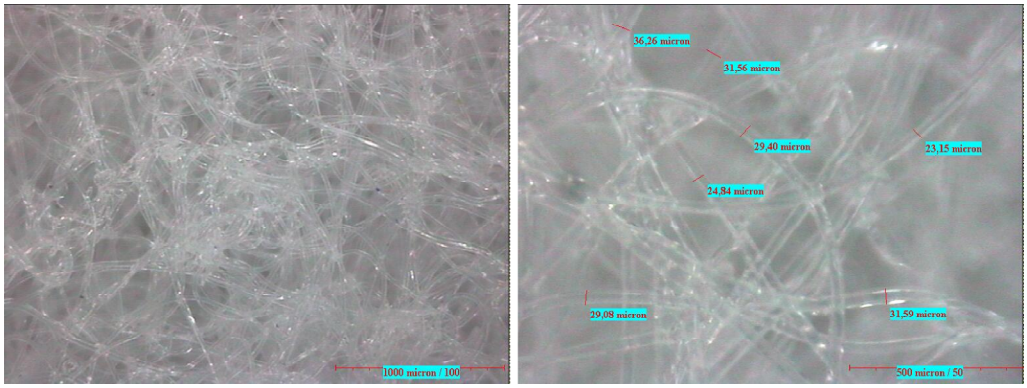


Figure 2: Porous Structure: Interconnected network of pores. Courtesy IBS Filtran

On the macroscopic level, one of the ways to describe porous materials is to homogenize and average over the micro structure and obtain a macro model in the framework of continuum mechanics. This is only true when the scales are distinctly separated. Averaging techniques, such as local volume averaging or homogenization, aims to exclude all the micro variables to have independent, closed macro systems containing the aver-

aged unknown variables and material parameters, such as porosity[1] and permeability[2] A comprehensive understanding of homogenization and averaging can be found in [63] and [48]. For the scope of this thesis, we limit our discussions of porous media, mainly to the macroscopic formulation where averaged/upscaled parameters are directly employed without derivations. However, in Chapter 4, we discuss the coupling of micro and macro scales models.

The processes for flow in plain and porous media are mostly governed by nonlinear partial differential equations (PDEs). Solutions are typically sought through some iteration process (cf. [80], [41], [38]) where large matrix equations need to be solved. The numerical solution of such systems require very large computational efforts. The speed with which the fully resolved matrix equations are solved and the amount of memory required to solve them pose serious bottlenecks in performing realistic large scale simulations. Even modern computers are unable to meet the industrial demands of large scale simulations which limits the problem size to the availibility of computer resources, raising a concern on the issue of solvability and accuracy. We take into consideration that accuracy is a fundamental concern, in terms of the restricted problem sizes.

When dealing with flow in plain and porous media simultaneously, the Fictitious Regions Method (FRM) permits using one system of Navier-Stokes-Brinkmann eqations to treat the fluid and porous regions. Justification is given in [3]. In such a case, the coefficients of the equations vary so that the single system is reduced to the Navier-Stokes equations in the liquid zone and to a Brinkmann-like model in the porous media. The varying coefficients increase the difficulty in solving the discrete problem. Special averaging techniques (cf. [78], [23]) and the use of efficient preconditioners (cf. [80], [23]) are a cure to such ill conditioned problems.

To add to the challenge, the process of filtration is mainly governed by two physical processes, i.e. the flow of fluid and the transport of impurities/dirt particles that are suspended in the fluid (cf. [13], [18]). The transport and capturing behaviour of particles within the filter[3] has an important influence on the efficiency and lifetime of the filter. For example, the oil filters are one of the basic components for the automotive industry. The life cycle pressure drop of the oil filters significantly impacts the corrosive damage of the engines. Generally, the pressure drop is a direct function of the filtration efficiency, such that high-efficiency filtration results in a high pressure drop. This pressure drop depends on the filter media, the filter housing and the flow. When a

[1] *Porosity*, ϕ, is a measure of the void spaces in a material, and is measured as a fraction between 0-1, $\phi = \frac{v_v}{v_t}$, where v_v is the volume of void-space (occupied by fluids) and v_t is the total volume of material, including the solid and void components.

[2] *Permeability* relates to the ability of the fluid to flow through the porous medium and is most relevant to flow computations.

[3] A filter can be described shortly as a filter box (which could be of complicated shape) with inlet/s for dirty oil and outlet/s for filtrated oil. The inlet/s and outlet/s are separated by a filtering medium, which is usually a single layer or a multi layer porous media.

filter is nearly clogged, the pressure drop is higher than in a new, clean filter. The significant parameters that affect filtration are the flow velocity, pressure drop, porosity and the increasing blockage effect due to the deposited particles during the filtration period. The flow transport, particle transport and capturing are different physical processes. The governing equations are different for different processes and they happen on several scales. Such a multi physics problem, where several processes interact at different scales, is a challenge in itself. The computation of the particles transport normally takes place in two steps. First the fluid flow profile needs to be determined and then depending on the geometrical, dynamical and material properties of the fluid flow, a suitable model determines the particle transport. For such a case, the choice of a correct model and the development of efficient algorithms becomes a crucial point of debate. In the course of this thesis, we discuss models and numerical algorithms which are verified by comparing the simulation results with measurements and analytical solutions, wherever possible.

The overall task of modeling and simulating the filtration-related multiscale processes becomes interdisciplinary as it employs physics, mathematics and computer programming to reach its aim. Keeping the challenges in mind, we outline the goals of this thesis followed by its structure.

Contents and Contributions of this thesis

This section will outline the main goals and contributions of this thesis. The main focus is to overcome the limitations of accuracy, speed and memory and to develop novel efficient numerical algorithms which could, in part or whole, be utilized by those working in the field of porous media. This work has essentially four parts, each of which is distributed into four separate chapters.

Parallelization

Chapter 1 discusses the single grid basic algorithm and a corresponding parallel algorithm to solve the macroscopic Navier-Stokes-Brinkmann model (cf. [23, 83]). Section 1.3 includes a short description of the grid generator in the form of a computational domain, the finite volume based discretization in space and time of the equations and the modified Chorin-type fractional time stepping algorithm. Further on, the chapter addresses the parallelization of the single grid version and improvements in terms of memory requirement and speed for solving the problem. It aims to devise, analyze and numerically test the different variants of parallel numerical algorithms for the Navier-Stokes-Brinkmann equations. We discuss two types of parallelizations. The main aim is to discuss the domain or data parallelization paradigm in Section 1.4 which is used to build a parallel algorithm which employs the MPI library to implement the data commu-

nication among processors. The numerical complexity of the algorithm is studied and compared with the sequential algorithm developed by [55], [56]. Noting that the linear solver is the most expensive part of the algorithm, we compare the domain parallelization approach with another variant where only the linear solver is parallelized using OpenMP. Performance on different hardware architectures/specifications is compared. Numerical experiments are always done in comparison with the single grid algorithm. As the algorithms are new, we provide the necessary formulation and report new results.

Subgrid method

As a second goal, we derive and numerically test an upscaling subgrid algorithm for solving the Navier-Stokes-Brinkmann equations [53] in Chapter 2. A one-scale model is shortly described in Section 2.2. In many complicated filters, the filter medium or the filter element geometry are too fine to be resolved by a feasible computational grid. The subgrid approach describes how the fine grid details are incorporated by solving auxiliary problems in appropriately chosen grid cells on a relatively coarse computational grid. This is done via a systematic and a careful procedure of modifying and updating the coefficients of the Navier-Stokes-Brinkmann system in the chosen cells. The key ingredient to the proposed algorithm, besides employing the coarse and the fine grid, is the usage of correct parameters, which becomes the critical part of this algorithm. Section 2.3 introduces the concept of *quasi-porous* coarse cells and describes the subgrid algorithm. In Section 2.4, the results are presented in such a way, that the efficiency and accuracy of the subgrid method is compared with the results obtained by solving the same problem on the coarse and fine computational grid. Results from the numerical simulation of industrial filters are also presented in this Section. Since it is a new algorithm, specific to filtration related flow models, new findings and results follow a detailed formulation of the numerical algorithm.

Multiscale Finite Volume Method

Moving a step further in the line of multiscale methods, an iterative Mutliscale Finite Volume (iMSFV) method is developed for the Stokes-Darcy system [66] in Chapter 3. The standard iMSFV method, developed by Hadi et. al. (cf. [45]) is extended for solving the momentum and pressure correction equations in a fractional time stepping algorithm framework. Section 3.1 includes a short description of the coupled Stokes-Darcy problem and a small note on the interface conditions used. Section 3.2 presents the building blocks of our algorithm. These are the single grid algorithms for the Stokes and the Darcy problems discretized on a staggered grid described in Subections 3.2.2 and 3.2.3. The MSFV and iMSFV method for the Darcy problem is described in Subection 3.2.4. Section 3.3 presents the multiscale finite volume algorithm for the coupled Stokes-Darcy system. Similar to the subgrid algorithm, this method allows to simulate

flow and transport in porous media with geometries that can be too complicated to be resolved by a feasible computational grid. iMSFV method bears the advantage of reconstructing the fine scale solution with a good precision at any desired time step. The coarse scale ensures global coupling and the fine scale ensures appropriate resolution. We discuss how this method performs and its potential development for further scientific and industrial goals. In Section 3.4, results from the numerical simulations are presented to validate the approach and to study its performance. Finally, some conclusions are drawn.

Coupled flow and transport solvers on macro-meso-micro scales

Chapter 4 deals with ways to incorporate changes occurring at different (meso) scale level. The flow equations are coupled with the Convection-Diffusion-Reaction (CDR) equation, which models the transport and capturing of particle concentrations. The two system equations are coupled and we try to understand and identify some of the unknown parameters appearing in the model equations. In Section 4.3, we first formulate the problem and discuss the different models, followed by its analytical solutions. Models for the Transmission Filter Effectiveness Method (TFEM) and Multipass efficiency test (cf. [33]) and analytical solutions for the models are discussed. Section 4.4 contains various ways to determine the unknown parameters in our model. A larger part of the discussion includes parameter identification methods, which is done via analytical solutions, measurement data, and micro simulations. The thesis opens new horizons to incorporate the two-scale processes occurring at different scales, which is essential for depicting the real loading of the porous media. By employing the numerical method for the coupled flow and transport problem, we understand the interplay between the flow velocity and filtration. Moreover, for filtrations problems, recognition of the importance of accounting for variations in permeability is a challenge altogether. Permeability is closely affected by the capturing of dirt particles on one scale which further affects the flow field on another scale. However, this is still work in progress and the results are not included in the content of this thesis. The last section includes validation of the parameter identification methods and results of the complete simulations on filter elements, which are compared with measurements and with analytical solutions whenever possible.

Finally in Chapter 5, we summarize the work of this thesis.

Chapter 1

A basic single grid algorithm for the Navier-Stokes-Brinkmann equations and its parallelization

1.1 Introduction and Goals

The numerical solution of PDEs, such as the porous media models or the Navier-Stokes equations, require very large computational efforts due to the size stiffness. A significant step in reducing the CPU time and/or increasing the accuracy of the simulations is the usage of parallel computers and clusters of workstations. The power of modern personal computers is increasing constantly, but not enough to fulfill all scientific and engineering computational demands. In such cases, parallel computing may be the answer. Parallel computing not only gives access to increasing computational resources but it also reduces computational time. This is mainly because clusters of workstations can be used as local dedicated computational nodes or as a parallel computer, according to momental needs of a department. A good review on the state of the art in numerical solution of PDEs on parallel computers is given in [20]. This book surveys the major topics that are essential to high-performance simulation on parallel computers, including programming models, load balancing, mesh generation, efficient numerical solvers, and scientific software.

This chapter aims to discuss parallelization of an existing numerical algorithm for the Navier-Stokes-Brinkmann (NSB) system of equations (cf. [55, 56, 69]), which is extensively used for the simulation of fluid flows in industrial filters. Even though the thesis specifically discusses and tests the algorithm for the NSB system, it can be generally applied to any system for incompressible Newtonian flows based on Cartesian grids with a finite volume discretization. The domain or data parallelization paradigm is used to build a parallel algorithm. The MPI library is used to implement the data communication among processors.

The chapter is organized as follows. In Section 1.2, we first formulate the problem. Section 1.3 describes the finite volume methodology used for solving the flow problem, space discretization and fractional time step discretization, respectively. A parallel algorithm based on the parallel domain (data) decomposition method is described in Section 1.4. The systems of linear equations are solved by a parallel version of the preconditioned BiConjugate Gradient Stabilized algorithm as shown in Subsection 1.4.1. A theoretical model, which estimates the complexity of the parallel algorithm is proposed. The results of computational experiments corresponding to the SP5[1] computer and a cluster of workstations (the specific hardware architecture is mentioned later along with the results) are presented and the efficiency of some popular parallel preconditioners is investigated. During computations, the diagonal and the incomplete LU (ILU) factorization preconditioners are considered. A parallel version of ILU is obtained by doing the factorization of a local part of the matrix at each processor. It is well known that such strategy reduces the convergence rate of the iterative algorithm, however, the parallelism of the obtained preconditioner is the same as the one obtained for the diagonal one. In Section 1.5, the computational results of experiments are presented, where some industrial filters are simulated and flows in such filters are investigated. Some final conclusions are given in Section 1.6.

1.2 Governing equations

The Brinkmann equations (cf. [14, 17]) are introduced as an extension of Darcy model for flow in porous media for the case of highly porous media (note, that porosity of the nonowen filtering media, which is our primary interest, is often more than 0.9). Concerning the hierarchy of the models for the porous media flow, we refer to the recent review in [76]. The Brinkmann model describing the flow in porous media, Ω_P, and the Navier-Stokes equations (cf. [38, 41]) describing the flow in the pure fluid region, Ω_F, together with the interface conditions for the continuity of the velocity and the continuity of the normal component of the stress tensor, are reformulated such that a single system of partial differential equations governs the flow in the pure liquid and in the porous media. This is done using the fictitious regions method (FRM). Note that the FRM allows the use of one system of equations in order to treat the fluid, porous and solid regions simultaneously (cf. [3, 81, 86] and the references therein). The coefficients of the equations vary in a way such that the single system is reduced to the Navier-Stokes equations in the liquid zone, and to the Brinkmann-like model in the porous media. This approach is only relevant when the interface conditions are chosen as mentioned above. Moreover, some details on modeling and simulation of flow through oil filters using

[1] Service Packs are a collection of software enhancements and fixes that is applied to an installed version of an operating system. The fifth service pack particularly introduced to upgrade a major release of software.

Navier-Stokes-Brinkman system can be found in [23, 55, 69].

The Navier-Stokes-Brinkmann system of equations describing laminar, incompressible and isothermal flow in the whole domain reads

$$\underbrace{\rho\frac{\partial\vec{u}}{\partial t}-\nabla\cdot(\mu\nabla\vec{u})+(\rho\vec{u}\cdot\nabla)\vec{u}}_{Navier-Stokes}+\overbrace{\mu\tilde{\mathbf{K}}^{-1}\vec{u}+\underbrace{\nabla p=\tilde{\vec{f}}}}^{Darcy} \tag{1.1}$$

$$\nabla\cdot\vec{u}=0, \tag{1.2}$$

where

$$\tilde{\vec{f}}=\begin{cases}\vec{f}_{NS} & \text{in } \Omega_F\\ \vec{f}_B & \text{in } \Omega_P\end{cases}\qquad \tilde{\mathbf{K}}^{-1}=\begin{cases}0 & \text{in } \Omega_F\\ \mathbf{K}^{-1} & \text{in } \Omega_P\end{cases}$$

Here $\vec{u}, p$ stand for velocity vector and pressure respectively, and ρ, μ denote the density, viscosity and $\mathbf{K}$ is the permeability tensor of the porous medium.
The equations are equipped with the following boundary conditions relevant to filtration processes. At the inlet, dirichlet inflow velocity $\vec{u}_{in}$ is prescribed. Dirichlet pressure $p = p_{out}$ and Neumann velocity $\frac{\partial\tilde{\mathbf{u}}}{\partial\mathbf{n}} = \mathbf{0}$ is prescribed is at the outlet. Elsewhere, on the solid walls, no-slip condition $\vec{u} = 0, p = 0$ is specified.

1.3 A single grid algortihm for solving Navier-Stokes-Brinkmann equations

The numerical algorithm can be decomposed into the following steps:
1. Grid initialization
2. Space discretization
3. Time stepping and numerical algorithm to compute $\vec{u}, p$.

In the following sections, we discuss each of these steps in detail.

1.3.1 Grid Initialization and Notations

In this chapter and the following ones, we will use some standard notations. Consider a set of non-overlapping finite control volumes (CVs) $E = \{E_n | n = 1, ..., N\}$ which spans over the 3D computational domain Ω, where N denotes the number of finite volumes. We have

$$\Omega = \cup_{n=1,N} E_n.$$

$E_n := E_n(\mathbf{x})$ denotes a finite volume on an orthogonal gird, with its centre node denoted by $\mathbf{x}$ with index (i, j, k). The neighbouring CVs in a certain space direction are denoted

by indices

$$i^{\pm} := (i \pm 1, j, k),$$
$$j^{\pm} := (i, j \pm 1, k),$$
$$k^{\pm} := (i, j, k \pm 1),$$

corresponding to the East/West, North/South, Top/Bottom neighbours. The set of finite volumes containing E_n's neighbours is denoted by

$$\mathcal{N}_n = \{E_n(\mathbf{x}_{i^-}), E_n(\mathbf{x}_{i^+}), E_n(\mathbf{x}_{j^-}), E_n(\mathbf{x}_{j^+}), E_n(\mathbf{x}_{k^-}), E_n(\mathbf{x}_{k^+})\}.$$

The size of each finite volume is denoted by

$$h_n = \{h_n^x, h_n^y, h_n^z\}.$$

Finally, the volume of the finite volume is denoted by v_n.

Implementation details: The geometrical information about the computational domain is usually provided in a CAD format, for example, the *.stl* format. A pre-processor based on the Level Set Method (cf. [64]), is used in order to process the given CAD data for attaining the assembly of a filter housing. The output of the pre-processor is a computational domain (i.e., the internal volume of the filter housing), along with a generated grid. An example of a computational domain can be seen in Figure 1.1.

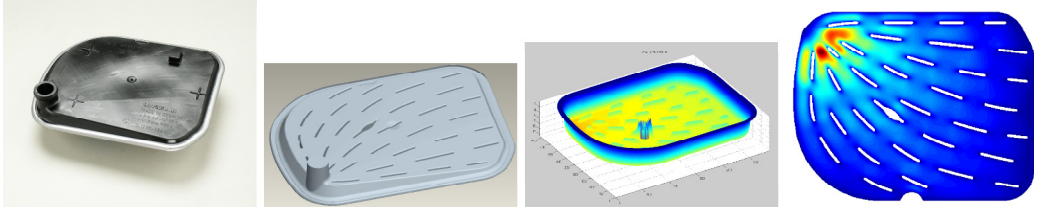


Figure 1.1: Computational domain of a filter housing and a snapshots of the visualization tool

1.3.2 Space discretization

The collocated arrangement is used where all unknowns are related to the centres of the control volumes/blocks. For the unknown ϕ, being the velocity component or pressure, the centre value for finite volume E_n is denoted by

$$\phi_n := \phi(\mathbf{x}_n).$$

The terms in equations (1.1) and (1.2) have common operators, namely the divergence and the gradient operators. The Navier-Stokes-Brinkmann system can be considered as a general transport equation excluding the Brinkmann term (discretization of which is

fairly straightforward), therefore discretization of a standard transport equation is considered. The discretization is considered to be local for each finite volume. After applying the Gauss divergence theorem (cf. [37]), discretization for the general transport equation is obtained.

$$\rho \frac{\partial \phi}{\partial t} v_n + \tag{1.3}$$

$$\{[(\rho u^x \phi - \gamma_\phi \frac{\partial \phi}{\partial x}) h_n^y h_n^z]_{i^{+0.5}} - [(\rho u^x \phi - \gamma_\phi \frac{\partial \phi}{\partial x}) h_n^y h_n^z]_{i^{-0.5}}\} + \tag{1.4}$$

$$\{[(\rho u^y \phi - \gamma_\phi \frac{\partial \phi}{\partial y}) h_n^x h_n^z]_{j^{+0.5}} - [(\rho u^y \phi - \gamma_\phi \frac{\partial \phi}{\partial y}) h_n^x h_n^z]_{j^{-0.5}}\} + \tag{1.5}$$

$$\{[(\rho u^z \phi - \gamma_\phi \frac{\partial \phi}{\partial z}) h_n^x h_n^y]_{k^{+0.5}} - [(\rho u^z \phi - \gamma_\phi \frac{\partial \phi}{\partial z}) h_n^x h_n^y]_{k^{-0.5}}\} + \tag{1.6}$$

$$= F_\phi v_n \tag{1.7}$$

In the case of the momentum equations, the general transport variable ϕ is one of the velocity components $\vec{u} = (u^x, u^y, u^z)$, $\gamma_\phi = \mu$ (dynamical viscosity) and $F_\phi = -\frac{\partial p}{\partial d} + f^d$, $d = (x, y, z)$. The equation also shows the way in which the velocity and pressure is decoupled. The pressure field is considered to be known when the momentum equations are calculated. The linearization of the momentum equation is of a fixed point type: the convective terms are written in an equivalent non divergent form and the velocity components are considered to be known. In each momentum equation one uses velocity approximations of Gauss Seidel type, where the already computed velocity components, denoted by $\bar{\bar{u}}$, are used within the scope of the same iteration when the next momentum equations are linearized.

On each of the finite volumes E_n, a seven point (for 3D) discretization stencil is used and the equation has the following form

$$\sigma_n \phi_n + \sum_{\mathcal{N}_n} \sigma \phi := \sigma_n \phi_n + \sigma_{i^\pm} \phi_{i^\pm} + \sigma_{j^\pm} \phi_{j^\pm} + \sigma_{k^\pm} \phi_{k^\pm} = F_n, \tag{1.8}$$

where variable ϕ are the unknowns and σ are the discretization coefficients corresponding to the centre of the finite volume E_n and its neighbours. The exact form of σ's will be discussed shortly. We simplify the notation, for example $\sigma_{i^+} := \sigma(x_{i^+})$. F_n is the contribution of the source/sink term for the variable ϕ located at the centres. We see how the values at each internal face are obtained through a linear interpolation of the neighbour node values. The discretization of the convective, diffusive terms are discussed for the case of an orthogonal grid.

Convective term discretization: For the convective term, two types of discretization schemes are considered, namely a second order central difference scheme and an upwind difference scheme. Let us consider the convective term associated with the east face of E_n, i.e. $\rho \phi h_n^y h_n^z u_{i^{+0.5}}^x$. If the east neighbour exists and the wall is internal, the convective term contributes to σ_n and σ_{i^+}. For the central difference scheme, the east

wall value is linearly interpolated through the nodal values

$$\phi_{i^{+0.5}} = (1-\theta)\phi_n + \theta\phi_{i^+}, \quad \text{where } \theta = \frac{h_n^x}{h_n^x + h_{i^+}^x} \tag{1.9}$$

Therefore, for the east wall, the convective term contributes to the following discretization coefficients

$$\begin{aligned} \sigma_n &\leftarrow (1-\theta)\rho h_n^y h_n^z u_{i^{+0.5}}^x, \\ \sigma_{i^+} &\leftarrow \theta\rho h_n^y h_n^z u_{i^{+0.5}}^x. \end{aligned}$$

For the case of the upwind difference scheme, the contribution depends on the direction of the flow velocity

$$\phi_{i^{+0.5}} = \phi_n \text{ if } u_{i^{+0.5}}^x > 0 \tag{1.10}$$

$$\phi_{i^{+0.5}} = \phi_{i^+} \text{ if } u_{i^{+0.5}}^x < 0. \tag{1.11}$$

Therefore for case (1.10), we have

$$\begin{aligned} \sigma_n &\leftarrow \rho h_n^y h_n^z u_{i^{+0.5}}^x \\ \sigma_{i^+} &\leftarrow 0. \end{aligned}$$

Similarly for case (1.11), we have

$$\begin{aligned} \sigma_n &\leftarrow 0 \\ \sigma_{i^+} &\leftarrow \rho h_n^y h_n^z u_{i^{+0.5}}^x. \end{aligned}$$

Diffusive term discretization: Let us consider the diffusive term $\gamma_\phi(\frac{\partial\phi}{\partial x})_{i^{+0.5}} h_n^y h_n^z$. A second order central difference scheme is used for the derivatives of the type $(\frac{\partial\phi}{\partial x})_{i^{+0.5}}$. In case of an internal wall the gradient is approximated by

$$(\frac{\partial\phi}{\partial x})_{i^{+0.5}} \approx \frac{\phi_{i^+} - \phi_n}{\bar{h}_{i^{+0.5}}} \quad \text{where } \bar{h}_{i^{+0.5}} = \frac{h_{i^+}^x + h_n^x}{2} \tag{1.12}$$

The contribution of the diffusive flux is as follows

$$\begin{aligned} \sigma_n &\leftarrow -\frac{1}{\bar{h}_{i^{+0.5}}}\gamma_\phi h_n^y h_n^z, \\ \sigma_{i^+} &\leftarrow \frac{1}{\bar{h}_{i^{+0.5}}}\gamma_\phi h_n^y h_n^z. \end{aligned}$$

1.3.3 Time Stepping and numerical algorithm

Here we formulate the numerical algortihm (cf. [55, 56]) for the macroscopic flow equations (1.1) and (1.2) in a general framework for a collocated arrangement of discrete variables. This algorithm serves as the basic algorithm for some of the subsequent chapters and also as a building block for the parallel algorithm/s described in the next

sections too. We first introduce some notations. The operators corresponding to the discretized convective and diffusive term are denoted by $C(\vec{u})\vec{u}$ and $D\vec{u}$ respectively. The particular form of these operators depend on the discretization, as discussed in the Section 1.3.2. The discretization of the gradient and the divergence operator is denoted by G and G^T. $B\vec{u}$ denotes the Darcy operator in the momentum equations. The superscript $^{k+1}$ and ^{k} denotes the new time level and the old time level. τ denotes the time step $\tau = t^{k+1} - t^k$.

Time discretization

Unsteady term discretization: The contribution of the unsteady term $\rho \frac{\partial \phi}{\partial t} v_n$ is

$$\begin{aligned} \sigma_n &\leftarrow \frac{(\rho v_n)}{\tau}, \\ F_n &\leftarrow \frac{(\rho v_n)}{\tau} \phi_n^k. \end{aligned}$$

The fractional time step discretization scheme can be written as

$$(\rho \vec{u}^* - \rho \vec{u}^k) + \tau(C(\vec{u}) - D + B)\vec{u}^* = -\tau G p^k \tag{1.13}$$

$$\left(\rho \vec{u}^{k+1} - \rho \vec{u}^*\right) + \tau(B\vec{u}^{k+1} - B\vec{u}^*) = -\tau(Gp^{k+1} - Gp^k) \tag{1.14}$$

$$G^T \rho \vec{u}^{k+1} = 0. \tag{1.15}$$

The numerical algorithm is based on the popular idea of splitting the operators and unknowns into different equations to divide the problem into easier substeps. Operator splitting methods, projection methods, fractional time stepping methods are different names associated with a similar concept (cf. [21, 41, 38, 37]). All these schemes solve a pressure equation after the momentum equations. Therefore, in essence, our algorithm can be viewed as a modification of the well known Chorin method for the Navier Stokes equations. Summing up Equations (1.13) and (1.14) results in an implicit discretization of the momentum equations. Equation (1.13) is solved with respect to the velocities using the old value of the pressure gradient, thus obtaining a prediction for the velocity. To solve the second equation for pressure correction one takes the divergence from it and uses the continuity equation. The result is a Poisson type equation for the pressure correction which will be discussed shortly.

Special treatment of the Darcy term

Brinkmann term discretization: The Brinkmann term $\gamma_\phi K^{-1} \phi$ in Equation (1.1) contributes to the discretization of the momentum equation as follows

$$\begin{aligned} \sigma_n &\leftarrow \gamma_\phi K_{11}^{-1}, \\ F_n &\leftarrow \gamma_\phi K_{12}^{-1} \bar{u}^y + \gamma_\phi K_{13}^{-1} \bar{u}^z. \end{aligned}$$

The Darcy term needs special treatment in the Navier-Stokes-Brinkmann case. The term is taken into account for both equations (1.13) and (1.14). The pressure equation should be carefully derived in this case. A naive application of the Chorin method would give

$$G^T(\rho\vec{u}^{k+1} - \rho\vec{u}^*) + G^T\tau(B\vec{u}^{k+1} - B\vec{u}^*) = -G^T\tau(p^{k+1} - p^k).$$

Denoting q for the pressure correction, where $q = p^{k+1} - p^k$, one can derive the following from Equation (1.14)

$$-G^T\tau Gq = G^T(\rho\vec{u}^{k+1} - \rho\vec{u}^*) + G^T\tau(B\vec{u}^{k+1} - B\vec{u}^*). \tag{1.16}$$

Furthermore, using the continuity equation $G^T\rho\vec{u}^{k+1} = 0$ and assuming that $G^TB\vec{u^{k+1}} \approx 0$, the above equation is reduced to

$$-G^T\tau Gq = -G^T(\rho\vec{u}^* + \tau B\vec{u}^*). \tag{1.17}$$

A drawback of the pressure correction equation is that its operator does not 'account for' the porous media and that the continuity equation is approximately satisfied in the porous media. For a constant time step, its operator is equivalent to the Poisson equation with constant coefficients. In order to achieve better convergence, lets consider another approach in forming a pressure correction equation. Let us denote I as the identity matrix 3×3 matrix. Rewriting Equation (1.14) gives

$$(I + \frac{\tau}{\rho}B)\rho\vec{u}^{k+1} - (I + \frac{\tau}{\rho}B)\rho\vec{u}^* = -\tau Gq. \tag{1.18}$$

Keeping in mind that $(I + \frac{\tau}{\rho}B)^{-1}$ always exists because B is positive definite, the above equation can be transformed into

$$\rho\vec{u}^{k+1} - \rho\vec{u}^* = -(I + \frac{\tau}{\rho}B)^{-1}\tau(Gp^{k+1} - Gp^k). \tag{1.19}$$

Now applying the divergence operator G^T to this equation and using the continuity equation, the following **pressure correction equation** is obtained

$$G^T(I + \frac{\tau}{\rho}B)^{-1}\tau Gq = G^T\rho\vec{u}^*. \tag{1.20}$$

It is easy to see that the pressure correction equation in the pure fluid region, where $B = 0$ reduces to the standard Chorin scheme. The same equation for pure fluids is obtained within the SIMPLEC approach (cf. [38]) or within the Schur complement approach (cf. [84]). It is more important to see that equation 1.19 collapses to the well known Darcy equation in the porous medium if I is much less than B. Thus, the equation describes both, the pure and the porous zones equally well. The choice of the time discretization influences the accuracy of the numerical solution and the stability of the algorithm. The approximation for the velocity after the solution of the momentum equations does not satisfy the continuity equation. Therefore using the continuity equation, one searches

for corrections of the velocities. After the pressure correction equation is solved, the pressure is updated

$$p^{k+1} = p^k + q \tag{1.21}$$

and the new velocity is calculated based on equation (1.19)

$$\rho\vec{u}^{k+1} = \rho\vec{u}^* + (I + \frac{\tau}{\rho}B)^{-1}\tau Gq. \tag{1.22}$$

It should be noted that one time step of the algorithm requires three linear system solves for the velocities in three directions and a system solve for the pressure correction equation. BiCGSTAB method is used to solve the linear system. Details are given later in Section 1.3.5. A sketch of the complete algorithm is given in Algorithm 1. A_{u_j} and A_q denote the discretization matrix for velocities and pressure correction.

Algorithm 1: Sequential numerical algorithm for Navier-Stokes-Brinkmann system

1. Initiate computational domain
2. ok = true; k = 0;
3. **while**(ok) **do**
4. k = k+1; $u^k = u^{k-1}; p^k = p^{k-1}$;
5. **for** (i=0; i < MaxNonLinearIter; i++) **do**
6. Compute velocities from momentum equations
   $A_{u_j}u_j^* = f_j - Gp^k, \;\; j = 1,2,3;$
7. Solve equation for the pressure correction
   $A_q q = r;$
8. Correct the velocities
   $u_j^k = u_j^* + \zeta_u(\mathscr{P}^{-1}Gq)_j, j = 1,2,3;$
9. Correct the pressure
   $p^k = p^k + \zeta_p q;$
   end do
10. **if** (final time step) ok = false;
    end do

So far we have discussed the finite volume discretization (cf. [38]) on the generated Cartesian grid. Cell centered grid with collocated arrangement of the velocity and pressure is used. The Rhie Chow interpolation (cf. [78]) is used to avoid spiral oscillations, which could appear due to the collocated arrangement of the unknowns. Special attention is paid to the discretizations near the interfaces between the fluid and the porous medium. To get an accurate approximation for the velocity and for the pressure on the interface, a special modification of the discretizations near the interface is used. First of all, it should be noted that the pressure gradient in the momentum equation is discretized in each cell separately. To do this, the pressure values from the cell centers are interpolated to the cell faces. In the pure fluid region, this is done by linear interpolation

where the pressure gradient is discretized via central differences. The more crucial part is the interpolation of the pressure on the interface between the pure fluid and the porous regions. Problem dependent interpolation is employed, using the operator from 1.20 together with the mass conservation assumption (i.e., continuity of the normal component of the flux across the interface). For a detailed illustration of this approach, suppose that p_L is a pressure value in a pure fluid cell located left from the interface, and p_R is a pressure value in a porous cell located right from the interface. The pressure at the interface is assumed to be continuous, and the value there, p_w, in the case of isotropic media (i.e. scalar permeability) is calculated from 1.20 as follows:

$$p_w = \frac{(1+\frac{\tau}{\rho}b)_L^{-1} p_L + (1+\frac{\tau}{\rho}b)_R^{-1} p_R}{(1+\frac{\tau}{\rho}b)_L^{-1} + (1+\frac{\tau}{\rho}b)_R^{-1}} \tag{1.23}$$

where b is the component of B in the isotropic case.

1.3.4 Special implementation feature

In practice, a series of computations with different velocities and different viscosities need to be performed for a fixed geometry. Particularly, in the case of oil filter simulations, the performance of a filter is usually evaluated at different flow rates and different temperatures (the last results in different viscosities and different densities). Also, recall that each single computation is performed for incompressible nonisothermal fluid. In order to reduce the computational efforts, a start from previous procedure is briefly discussed. This routine takes advantage of the fact that there exists a good initial guess for all simulated cases except for the first one. The cases to be simulated are ordered such that the parameter

$$\gamma_l = \frac{\mu_l}{Q_l} \tag{1.24}$$

is increasing. Here Q stands for the prescribed flow rate at the inlet, and l stands for the current number of the set of input parameters. After the first case corresponding to γ_l is computed, the computations corresponding to γ_{l+1} start with reading the l-th steady state solution, and thereby rescaling the pressure in accordance with the formula

$$G_{p_{l+1}} = \frac{\gamma_l}{\gamma_{l+1}} G p_l. \tag{1.25}$$

1.3.5 Computational memory requirements

During each iteration at steps (6) and (7) of Algorithm 1, four systems of linear equations are solved. The non-symmetric linear systems of the form

$$\mathbf{A}\phi = f$$

BiCGSTAB (Vec x, Vec f, Matrix A, double ε)
begin
(1) Compute the precondinioner $\mathscr{P}$;
(2) normf = $\|f\|$;
(3) **if** (normf $< \varepsilon$) normf = 1;
(4) $r = f - Ax$;
(5) $\tilde{r} = r$; ok = true; i = 0;
(6) **while** (ok) **do**
(7) i++; $\rho_1 = (\tilde{r}, r)$;
(8) **if** (i == 1) **then**
(9) p = r;
else
(10) $\beta = (\rho_1 \alpha)/(\rho_2 \omega)$;
(11) $p = r + \beta\,(p - \omega v)$;
end if
(12) $\mathscr{P}\hat{p} = p$;
(13) $v = A\hat{p}$;
(14) $\alpha = \rho_1/(\tilde{r}, v)$;
(15) $s = r - \alpha v$;
(16) **if** ($\|s\|$ / normf $< \varepsilon$) **then**
(17) $x+ = \alpha\hat{p}$;
(18) ok = false;
else
(19) $\mathscr{P}\hat{s} = s$;
(20) $t = A\hat{s}$;
(21) $\omega = (t, s)/(t, t)$;
(22) $x+ = \alpha\hat{p} + \omega\hat{s}$; $r = s - \omega t$;
(23) $\rho_2 = \rho_1$;
(24) **if** ($\|r\|$ / normf $< \varepsilon$) ok = false;
end if
end do
end BiCGSTAB

Figure 1.2: Serial BiCGSTAB algorithm

are solved using the preconditioned BiConjugate Gradient Stabilized method [12]. A short description of the BiCGSTAB method is given in Figure 1.2.

Solving such systems typically requires 70 to 90 % of computational time. The power of modern computers is increasing constantly, but is still unable to fulfill all scientific and engineering computational demands. Finer resolution implies greater computational complexities, not only in terms of computational time but also memory that is needed to solve the system. Parallel computing provides an answer to such computational bottlenecks. A detailed review on the parallel version of the discussed algorithm is given in the following section.

1.4 Parallel algorithm

The objective is to have an efficient solution on parallel machines, using finite volume methods on problems with complex geometries which are approximated on very fine grids. After discretization, to solve the obtained linear system is extremely time and memory consuming. Development of effective parallel algorithms is a challenging task for such a problem.

One of the possibilities is to parallelize only the linear system solver, for example by using the well known parallel software package PETSc. Typically, the linear solver works with the spatial discretization coefficients which are distributed among the number of parallel processes in a parallel matrix. The linear system is then solved using parallel linear algebra solvers from PETSc on each process, and the solutions are then collected from the various processes. Another way for parallelizing the linear solver is using OpenMP parallelization. This is discussed in Subsection 1.5. We should keep in mind that parallelization of the linear solver has the advantage of gaining in terms of computational time.

Alternatively, in many cases the main goal of parallel computations is not only to solve the problem faster, but also to increase the size of the simulated problems. Therefore the distribution of the discretization part of the algorithm to scale the problem size according to the increased number of processors becomes a crucial point to consider. Also, when only the linear solver is parallelized, the discretization is done only on one master processor while the linear systems of equations are solved in parallel. The drawback of this practice becomes apparent with solving larger sized problems, when the additional costs, pertaining to the distribution of the matrix and the right hand side vector among processors and assembling the solution on the master processor are considered.

In effect, we are left with the following challenges for an optimal parallel algorithm:

1. Load balancing.
2. The costs incurred by the exchange of information, where we should define optimal mapping to minimize overlapping regions.

3. Parallelizing the linear solver.

Keeping in mind the above considerations, the essential details of the developed parallel algorithms are described in the following sections. A theoretical model estimating the complexity of the parallel problem is also proposed.

1.4.1 Domain decomposition Approach

Domain decomposition methods have demanded much attention over the past years, and its success owes to the fact that they provide a high concurrency level and come with easy implementation on complex modern parallel computers. The goal of the domain decomposition approach is to split the original domain into smaller simple subdomains, compute local solutions and use iterative solvers to appropriately interface the solutions. Defining an optimal criteria to divide the domain into subdomains is not a trivial problem as it involves both numerical constraints and parallel implementation constraints. The former arise from the complicated imbalanced shape of the domains, whereas the parallel implementation constraints boil down to minimizing the information exchange or communication between processors. We refer to this constraint as the load balancing poblem, with the objective to partition the problem into well balanced subdomains and minimal interaction with a good aspect ratio.

To understand the load balancing problem, we start with defining a suitable mapping

$$V = V_1 \cup V_2 \cup \ldots V_P$$

of all finite volumes onto the set of P processors. Here, V_j defines the elements mapped onto the $j-th$ processor. The load balancing problem should be solved during implementation of this step. First, it is essential that each processor has about the same number of elements since this number will determine the computational complexity of Algorithm 1. Depending on the stencil of discretization, the computational domains of processors can overlap. The information belonging to the overlapped regions should be exchanged between the processors. For distributed memory computers, the MPI library is used to send explicit messages between the processors, contributing to the additional costs of the parallel algorithm. Therefore the underlying goal of defining the optimal data mapping is to minimize the overlapping regions.

Load Balancing

The p-way graph partitioning problem is N-P complete i.e. no polynomial time algorithm is likely to be found to solve this problem. Therefore, heuristic algorithms have been developed to find a good solution in a reasonable time. The multilevel partitioning method is one of the most efficient partitioning methods with linear complexity. The state of the art implementation of a family of multilevel partitioning methods for parti-

tioning unstructured graphs and hypergraphs is available as a METIS software library (cf. [62]).

A simpler domain decomposition approach has also been used for this particular case. It takes into account that the orthogonal 3D structured grid is used as a reference grid for the definition of the computational grid. Therefore, standard 3D decomposition of the processors $P_1 \times P_2 \times P_3$ can be used. Such a strategy simplifies the implementation of the data exchange algorithms since it is very easy to define the neighbours of each processor and the overlapping elements.

In order to solve the load balancing problem for a given number of processors, all combination of 3D processors topologies is generated. A topology with the best load balancing that minimizes the number of overlapping elements is chosen.

In a series of computational experiments, the quality of the obtained partitioning was tested and compared to the partitioning computed by METIS. Table 1.1 shows the values of the load disbalance parameter d_P and the number of overlapping elements w_P (or edges cutting the partitioned subsets of elements in the case of METIS partitioning) are presented for the METIS and orthogonal 3D partitioning. The grid was generated for a real industrial application, the graph of this grid has 596094 nodes and 1507732 edges, the auxiliary grid has 5428000 nodes.

It is seen that the simple grid partitioning algorithm gives mappings with good load balancing and the number of overlapped elements is also close to the number of similar elements in the partitioning that was generated by METIS.

P	$d_{P,Metis}$	$2w_{P,Metis}$	$d_{P,3D}$	$w_{P,3D}$
2	1.0	6090	1.05	5874
4	1.0	12164	1.19	16884
8	1.0	24162	1.21	34450
12	1.0	32836	1.22	60270

Table 1.1: Experimental investigation of the quality of partitioning algorithms

Data initialization

In this section, we estimate the costs of data initialization. The master processor reads the information on the grid from a file and broadcasts it to other processors. The complexity of the global broadcast operation strongly depends on the architecture of the parallel computer (cf. [25, 44, 47]). The cost of broadcasting s data items between P processors is estimated as

$$B(s,P) = R(P)(\alpha_b + \beta_b s)$$

where $R(P)$ depends on the algorithm used to implement the broadcast operation and the architecture of the computer. For the simplest algorithm, $R(P) = P$. Taking into account the time $O(n)$ required to read data from the file and assuming that $\alpha_b \ll \beta_b s$, a bound on the costs of grid initialization is found as

$$W_{P,init} = (c_0 + \beta_b P)s. \tag{1.26}$$

Note that this part of the computations does not depend on the number of non linear iterations and the number of time steps. Therefore, the initialization costs can be neglected for problems where a long transition time is simulated.

Parallel discretization

The sequential algorithm is decomposed into local computations supplemented with corresponding communication operations. The matrices and right-hand side vectors are assembled element by element. This can be done locally by each processor, if all ghost values[2] (see Figure 1.3) of the vectors belonging to overlapping regions are exchanged among processors. The data communication is implemented by an *odd - even* type algorithm and can be done in parallel between different pairs of processors. Thus, we can estimate the costs of data exchange operation as

$$W_{exch} = \alpha_e + \beta_e m,$$

where m is the number of items sent between two processors, α is the message startup time and β is the time required to send one element of data.

The time required to calculate all coefficients of the discrete problem is given by

$$W_{P,coeff} = c_1 d_P \frac{s}{P},$$

where d_P is a load disbalance parameter.

Parallel BiCGSTAB algorithm

The sequential BiCGSTAB algorithm is modified in a way such that its convergence properties are not changed during the parallelization process. The only exception is due to implementation of the preconditioner $\mathscr{P}$. If $\mathscr{P}$ is a diagonal part of the matrix, i.e. for Jacobi smoothing a parallel realization of the preconditioner is exactly the same as for sequential one. In the case of ILU preconditioner, parallelization can reduce the convergence rate of the parallel BiCGSTAB algorithm. These questions will be addressed in the following section. Here, it is assumed that the number of iterations required to solve the systems of linear equations, at steps (6) and (7) in Algorithm 1 are the same for the sequential and parallel versions of the BiCGSTAB algorithm.

[2]Ghost cells are an extra layer of cells added to local domain in the processor. These are the cells which are contained in other processors but whose information is needed by the process for local discretization.

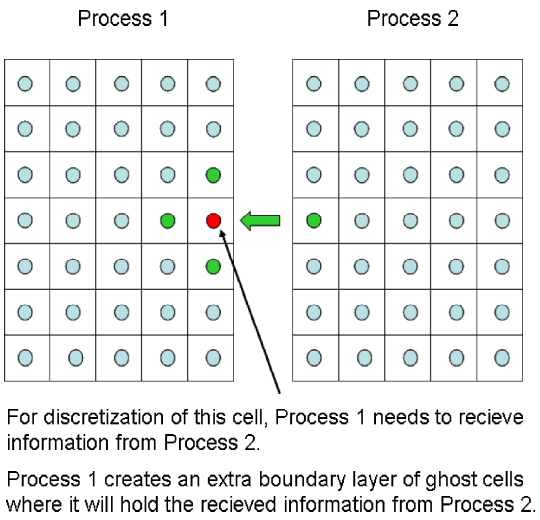


Figure 1.3: An illustration of ghost cells and the information that needs to be exchanged among processes.

Four different operations of the BiCGSTAB algorithm require different data communications between processors:

1. Vector *saxpy* operations, at steps (11), (15) and (22) in Algorithm 1.2, can be computed in parallel, when parameters α, β, ω are given. No communication between processors is needed, since all required data is locally available on each processor. The complexity of all vector *saxpy* operations calculated during one iteration is
$$W_{P,saxpy} = c_2 d_P \frac{s}{P}.$$

2. Implementation of the matrix – vector multiplication, at steps (4), (13) and (20) in Algorithm 1.2, requires additional information when boundary nodes of the local part of the vector x are updated (note, that these nodes are inner nodes in the global grid). Such information is obtained by exchanging data with neighbour processors in the specified topology of processors. The amount of data depends on the grid stencil, which is used to discretize the PDE model, i.e. on the overlap of local subgrids. The communication step can be done in parallel. After exchange of the ghost elements the multiplication Ax is performed locally on each processor. Taking into account that matrices are sparse, the complexity of two matrix-vector multiplications during one iteration is estimated by
$$W_{P,mv} = c_3 d_P \frac{s}{P} + 2(\alpha_e + \beta_e m).$$

3. The computation of inner products of two vectors at steps (2), (7), (14), (21) and (24) in Algorithm 1.2 require a global communication of all processors: first all

processors compute inner products of local parts of vectors and then these local products are summed up. Different algorithms can be used to implement the global reduction step. In MPI, there exists a special function `MPI_ALLREDUCE`, which computes a sum and distributes it to all processors. It is assumed that MPI library is optimized for each type of super–computer, taking into account specific details of the computer network. The complexity of computation of all inner products and norms during one iteration is estimated as

$$W_{P,dot} = c_4\, d_P \frac{s}{P} + 5R(P)(\alpha_r + \beta_r).$$

For a simple implementation of `MPI_ALLREDUCE` function, when all processors send their local values to the master processor, which accumulates results and broadcasts the sum to all processors, $R(P) = cP$.

4. The computation of the preconditioner $\mathscr{P}$ is done locally by each processor without any communication operation. The complexity of this step is given by

$$W_{P,\mathscr{P}} = c_5\, d_P \frac{s}{P}.$$

The solution of linear systems $\mathscr{P}x = b$ also requires only local computations. Thus the complexity of steps (12),(19) of Algorithm 1.2 is given by

$$W_{P,\mathscr{P}^{-1}} = c_6\, d_P \frac{s}{P}.$$

After summing up all the estimates, the theoretical model of the complexity of the parallel algorithm is achieved

$$\begin{aligned} W_P = (c_0 + \beta_b P)s + K\big((c_1 + c_5)d_P \frac{n}{P} + c_7(\alpha_e + \beta_e m(P))\big) \\ + N\big((c_2 + c_6 + c_{dot})d_P \frac{s}{P} + c_8 R(P)(\alpha_r + \beta_r) + c_9(\alpha_e + \beta_e m(P))\big), \end{aligned} \tag{1.27}$$

where K is the number of steps in the outer loop of Algorithm 1, and N is a total number of BiCGSTAB iterations. Note that the initialization costs (i.e. the first term of the total costs) do not depend on the number of time steps K and they can be neglected in the case when K is a large number.

1.5 Numerical Results and Conclusions

The accuracy of the theoretical complexity model developed above was tested experimentally. Computations were performed on IBM SP5 computer at CINECA, Bologna and on Virgo cluster of computers at ITWM, Kaiserslautern. Results of simulations are presented in Figure 1.4.
The same industrial application, as in Section 1.4.1 for grid partitionaing, is used to test the prediction accuracy of the theoretical model.

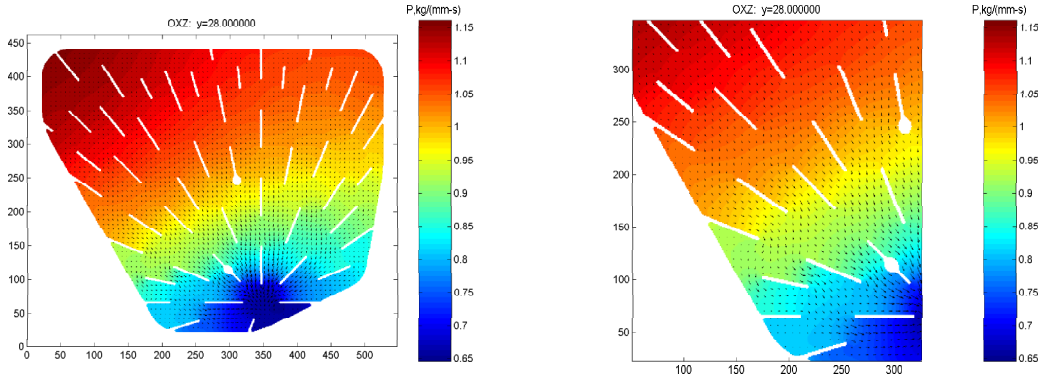


Figure 1.4: Results of simulations: velocity (arrows) and pressure (colour) in a cross–section of the filter

P	T_{init}	T_1	T_2	T_{dot}	T_{exch}	T_{total}	S_P	E_P
1	1.66	106	834	117.3	0.0	1059	1	1
2	3.24 3.24	54.5 55.7	435 438	63.0 63.4	4.7 4.7	560.5 565.0	1.89	0.95
4	4.23 4.80	31.0 31.5	252.5 248	36.1 37.0	6.6 6.1	330.7 327.7	3.20	0.80
8	5.12 6.50	15.6 16.0	126.2 126.1	20.7 21.0	7.3 6.4	175.0 176.0	6.05	0.76
12	7.39 9.21	10.7 10.8	84.9 84.8	15.3 15.8	5.1 7.3	123.3 126.0	8.59	0.71

Table 1.2: Experimental results and theoretical predictions of CPU times for Virgo cluster

In the following tables, experimental times and theoretical predictions of CPU time

$$T_{init} = (c_0 + \beta_b P)s, \quad T_1 = K(c_1 + c_5)d_P\frac{s}{P}, \quad T_2 = N(c_2 + c_6)d_P\frac{s}{P},$$

$$T_{exch} = (Kc_7 + Nc_8)(\alpha_e + \beta_e m), \quad T_{dot} = c_4 d_P\frac{s}{P} + c_8 R(P)(\alpha_r + \beta_r)$$

are presented. For each number of processors, the first line gives experimental values of CPU and the second lines present theoretical predictions. In Table 1.2, the results of computations are presented for the Virgo cluster of computers.

The presented results show that theoretical complexity model gives accurate predictions of different parts of Algorithm 1. The efficiency of the parallel algorithm is also good. We note that cluster Virgo uses a Myrinet communication network, therefore communication costs do not reduce seriously the efficiency of the algorithm for a given number

of processors. Results of calculations done on SP5 computer are presented in Table 1.3.

P	T_{init}	T_1	T_2	T_{dot}	T_{exch}	T_{total}	S_P	E_P
1	1.83	82.1	735	70.3	0.0	889	1	1
2	3.12	44.1	381	35.7	0.58	464	1.92	0.96
	3.12	43.1	386	37.1	0.58	470		
4	3.58	23.3	203	19.1	3.20	252	3.52	0.88
	5.16	24.4	218.7	21.4	1.2	271		
8	4.54	11.8	93.2	10.2	3.06	122.7	7.25	0.91
	6.66	12.4	111	11.6	1.9	143.6		
12	6.65	8.4	63.0	6.61	3.1	87.8	10.1	0.84
	8.78	8.2	73.8	8.1	2.7	101.5		

Table 1.3: Experimental results and theoretical predictions of CPU times for SP5

It can be seen that the theoretical complexity model overestimates the CPU time. The accuracy of the model can be increased if the well-known fact that efficiency of vector operations increases is taken into account. Also, a superlinear speedup of the parallel algorithm is obtained for larger numbers of processors due to the better cache memory utilization in SP5 processors. A simple test was implemented, where matrix operations $A := A + B, C := C - \mathscr{P}$ were performed many times. The dimension of matrices were taken to be $4 \cdot 10^6$. The following results were obtained:

$$T_1 = 35.3, \;\; T_2 = 15.3, \;\; T_4 = 7.18, \;\; T_8 = 2.83, \;\; T_{16} = 1.29.$$

Data distribution using METIS library

In previous computations, a 3D data decomposition among processors was used. Since the geometry of a computational region is quite complicated, such a decomposition leads to an imbalance of the work-load between processors (up to 1.20 times). Additionally, a general grid distribution algorithm was implemented, which was based on graph distribution algorithms implemented in METIS library (cf. [62]). In Table 1.4, the results of computations are presented and the two strategies of data distribution are compared. The computations were performed on Virgo cluster of computers, but in this case a Gigabit Ethernet network is used. The code was compiled with the full optimization option $O3$ in order to make the ration between computation and communication speeds more challenging. In order to get more realistic estimates of the speed-up and the efficiency coefficients, the initialization time T_{init} was excluded from the computation time T_P, since this time could be neglected for real simulations. During the computational experiments, the solution was computed only for six time steps.

P	$T_{P,3D}$	$S_{P,3D}$	$E_{P,3D}$		$T_{P,M}$	$S_{P,M}$	$E_{P,M}$
1	885.7	1.00	1.00		893.0	1.00	1.00
2	480.4	1.84	0.92		459.7	1.94	0.97
4	270.6	3.27	0.82		234.3	3.81	0.95
8	150.8	5.87	0.73		127.0	7.03	0.88
12	104.4	8.48	0.70		90.2	9.92	0.83

Table 1.4: Experimental results for 3D and METIS data distributions

Parallel Preconditioners

There have been many studies of the use of various ordering techniques to overcome the trade–off between parallelism and convergence in ILU factorization. Some new multicolor orderings are proposed by D'Azevedo *et al.* [32], Doi and Washio [27], Monga–Made and Van der Vorst [73], Čiegis [24]. The comparison of parallel preconditioners for non-symmetric sparse linear systems is done by Ma [72].

A simple parallel version of ILU preconditioner was implemented, with each processor computing the required factorization by using only a local part of the matrix A. Such a Jacobi type ILU preconditioner is fully parallel, but the convergence rate of the obtained iterations is decreased (cf. [11, 24]). It is very difficult to estimate the convergence rate of the BiCGSTAB algorithm with the Jacobi ILU preconditoner even for problems obtained after discretization of the Laplace equation on uniform grids. The efficiency of preconditioners also depends strongly on the given problem coefficients and the properties of the grid. Therefore mainly experimental investigations are used in the analysis of simplified preconditioners. In Table 1.5, the performance of BiCGSTAB iterative algorithm with the Jacobi ILU preconditioner is given. Here, N_P is the total number of BiCGSTAB iterations calculated in solving all systems of linear equations by using P processors, and S_P and E_P are the speed-up and efficiency coefficients, respectively, of the parallel algorithm. Note that the number of iterations was exactly the same for any number of processors in the previous experiments, i.e. the iterative process was always stopped after computing the maximal number of iterations. The computations were done on the ITWM cluster Virgo.

It can be seen that the number of iterations for the BiCGSTAB algorithm with the Jacobi ILU preconditioner increase when compared with the global ILU preconditioner. Therefore, the efficiency of parallel algorithm is decreased (compare the new values of E_P with the values given in Table 1.2). However, the quality of the Jacobi ILU preconditioner is still quite satisfactory.

The calculations of the ILU factorization and the solution of problems $\mathscr{P}x = f$ are quite costly. The efficiency of the sequential ILU preconditioner with the Jacobi diagonal pre-

P	N_P	T_P	S_P	E_P
1	3304	1246	1.00	1.000
2	3741	742	1.68	0.840
4	4070	465.5	2.68	0.670
8	4137	248.6	5.01	0.627
12	4181	175.6	7.10	0.591

Table 1.5: Iteration numbers N_P, CPUtime T_P, speed-up S_P and efficiency E_P coefficients for the Jacobi ILU preconditioner

conditioner was compared. The same problem was solved by using $N = 9849$ iterations and $T_1 = 2315$ CPU time. Thus, the number of iterations increased 2.98 times, but the CPU time increased by 1.85 times only.

Dependence on hardware of parallel computers

In this section, we present results of the developed parallel algorithm with new data structures, that allow to significantly reduce memory requirements of the solver. The auxiliary 3D reference grid that was used until now is removed, and METIS is used for graph partioning. For the test problem, the same industrial filter is used as in previous sections. The maximum number of BiCGSTAB iterations is taken to be 600. Although it is not sufficient for the full convergence of the pressure correction equation, it suffices as we seek to find only a stationary solution. This imposes a hard bound restriction that the parallel linear solver for all tests perform the same number of iterations despite possible differences in the overall convergence. Results are presented for the first three timesteps of Algorithm 1.

Distributed memory vs Shared memory systems

Firstly, the performance of the DD algorithm is tested on a distributed memory parallel machine -Vilkus cluster at VGTU. It consists of Pentium 4 processors (3.2 GHz, level 1 cache 16KB, level 2 cache 1MB, 800MHz FSB) interconnected via Gigabit Smart Switch. Obtained performance results are prescribed in Table 1.6. Here, for each number of processes P, the wall clock time T_P, the values of the algorithmic speed-up coefficients $S_P = \frac{T_1}{T_P}$, and the efficiency $E_P = \frac{S_P}{P}$ are presented.

As we can see, the scalability of the DD parallel algorithm is robust. According to the theoretical model of its complexity (1.27), it scales better for the systems with better interconnect network: with smaller α, β, $R(P)$. This can be seen from the results in Table 1.7, which were obtained on ITWM Hercules cluster: dual nodes (PowerEdge 1950) with dual core Intel Xeon (Woodcrest) processors (2.3 GHz, L1 32+32 KB, L2 4 MB, 1333 MHz FSB) interconnected with Infiniband DDR. The superlinear speedup is

	$P=1$	$P=2$	$P=4$	$P=8$	$P=12$	$P=16$
T_P	456	234	130	76.9	53.8	41.8
S_P	1.0	1.95	3.51	5.93	8.48	10.9
E_P	1.0	0.97	0.88	0.74	0.71	0.68

Table 1.6: Performance results of DD parallel algorithm on VGTU Vilkas Cluster

explained also by the growing number of cache hits with increasing p.

	$P=1$	$P=2$	$P=4$	$P=8$	$P=12$	$P=16$
T_P	335.3	164.0	79.1	37.4	23.9	17.6
S_P	1.0	2.04	4.24	8.96	14.3	19.05
E_P	1.0	1.02	1.06	1.12	1.17	1.19

Table 1.7: Performance results of DD parallel algorithm on ITWM Hercules cluster (one process per node)

Next, we performed computations with the same parallel algorithm on a shared parallel machine - multicore computer. The system has an Intel(R) Core(TM)2 Quad processor Q6600. Four processing cores are running at 2.4 GHz each and are sharing a 8 MB of L2 cache and a 1066 MHz Front Side Bus. Each of the four cores can complete up to four full instructions simultaneously. The performance results are presented in Table 1.8.

	$P=1$	$P=2$	$P=3$	$P=4$
T_P	281.9	186.3	183.6	176.3
S_P	1.0	1.51	1.54	1.60
E_P	1.0	0.76	0.51	0.40

Table 1.8: Performance results of DD algorithm on the multicore computer: Intel Core 2 Quad processor Q6600 @ 2.4GHz.

The results of the test-runs show that the speed-up of 1.51 is obtained for two processes and it saturates for a larger number of processes: algorithm is not efficient even for 3 or 4 processes. In order to get more information on the run time behaviour of the parallel MPI code we have used a popular parallel profiling tool Intel(R) Trace Analyzer and Collector. It showed that for $P=2$ all MPI functions took 3.05 seconds and for $P=4$ processes 8.93 seconds(1.90, 2.50, 1.17, 3.36s). Thus the communication part of the algorithm is implemented very efficiently. It scales/grows according to the predictions of our theoretical model, but not linearly. Since the load balance of the data is also equal to one, the bottleneck of the algorithm arises due to the conflicts in memory

access between different processors.

The same tests on a better shared memory architecture -single node of ITWM Hercules cluster (2×2 cores) gave us slightly better, but qualitatively the same results. From Table 1.9, we can see that the use of all 4 cores on the nodes is not efficient. The run-time is almost the same as using two separate nodes with 2 processes (Table 1.7).

	$P=1$	$P=2$	$P=4$
T_P	335.3	185.9	153.2
S_P	1.0	1.80	2.19
E_P	1.0	0.90	0.55

Table 1.9: Performance results of DD parallel algorithm on the multicore computer: ITWM Heclules cluster's single node (PowerEdge 1950)

OpenMP parallel algorithm

The second approach is to use OpenMP application program interface to see if our analysis in 1.5 is correct or we can get something better with special programming tools for shared memory computers. Since for shared memory computers we expect a more favorable ratio between computation and data exchange times, such a decrease of the parallel algorithm efficiency is explained by the conflicts in memory access between different processors. The main part of the CPU time is spent in the realization of BiCGSTAB iterations, and for sparse matrices frequent misses of L2 cache accesses are expected. Therefore there is a possibility to parallelize only the linear system solver, i.e. the BiCGStab routine. Parallel algorithm is obtained quite easily by putting special directives in saxpy, dot product, matrix-vector multiplication and preconditioner operations. In this way, the discretization is done sequentially on the master thread only, while the linear systems of equations are solved in parallel.

The first important result following from computational experiments is that a direct application of OpenMP version of the parallel algorithm is impossible since the asynchronous block version of the Gauss-Seidel preconditioner does not give converging iterations. Therefore, we have used a diagonal preconditioner for all computational experiments presented in this subsection. In Table 1.10, we compare the results obtained by both algorithms, namely the DD parallel algorithm with MPI and the OpenMP parallel algorithm. As can be seen, the same conclusion can be drawn for both cases: we obtain a reasonable speed-up only for 2 processors. The use of 3 and 4 processors is inefficient on the multicore computer used in our tests.

Next, the information collected by the profiling of the OpenMP parallel algorithm is presented. Our goal was to see and compare the performance of parallelization of dif-

		$P=1$	$P=2$	$P=3$	$P=4$
DD MPI	T_P	198.0	139.9	138.4	139.1
	S_P	1.0	1.42	1.43	1.42
	E_P	1.0	0.71	0.72	0.71
OpenMP	T_P	202.3	155.0	155.3	151.9
	S_P	1.0	1.31	1.31	1.33
	E_P	1.0	0.65	0.65	0.66

Table 1.10: Performance results of DD and OpenMP parallel algorithms with the diagonal preconditioner on the multicore computer: Intel Core 2 Quad processor Q6600 @ 2.4 GHz

ferent sections of the linear solver. In Table 1.11, the summary of execution times of four constituent parts of the algorithm are given, here *saxpy* denotes for the CPU time of all saxpy type operations, *dot* is the overall time of all dot products and norm computations, *mult* denotes the CPU time of all matrix – vector multiplications, and *precond* is the time spent in the implementation of the diagonal preconditioner.

section	$P=1$	$P=2$	$P=3$	$P=4$
saxpy	38.6	31.3	31.8	32.1
dot	17.8	13.6	13.3	12.9
mult	80.0	48.1	46.4	43.8
precond	47.1	41.8	43.2	42.9

Table 1.11: Profiling results of the OpenMP parallel algorithm on the multicore computer: Intel Core 2 Quad processor Q6600 @ 2.4 GHz

It follows from the results in Table 1.11 that the considerable speed-up is achieved only for the matrix–vector multiplication part, but even here the speed-up is significant only for two processors. Again, this can be explained by the memory access bottleneck. The ratio of computations to data movement is better in matrix–vector multiplication, therefore we get better parallelization for that operation.

1.6 Summary

A parallel algorithm is described for the Navier-Stokes-Brinkmann equations based on Algorithm 1. The first parallelization approach is based on the data decomposition method. The data is distributed among processors by using two approaches. A structured reference grid is distributed using the optimal decomposition topology. In the second one, the general mesh is decomposed using the Metis library. A theoretical

model is proposed for the estimation of complexity of the given parallel algorithm. The theoretical and experimental results obtained are in very good agreement, and thus it can be predicted that the proposed parallel algorithm scales well, and it can be used efficiently for simulation of oil filters with complicated 3D geometries.

The results of the performance tests show that fine-tuned MPI implementation of the domain decomposition parallel algorithm for the sequential Algorithm 1 performs very well not only on distributed memory systems but also on shared memory systems. MPI communications inside the shared memory are well optimized in the current implementations of MPI libraries.

The problems with scalability on some of the shared memory systems (e.g. multicore computers) arise due to the hardware architecture and not due to the use of MPI. Unfortunately, the use of simple shared programming tools cannot overcome this problem of hardware architecture.

Despite the gains in parallel computing, it still requires expensive systems like clusters, multi processor machines and complicated hardware. In the next chapter, we aim to discuss the subgrid algorithm which benefits from solving problems on a coarse scale with fine scale accuracy with significantly less computational resources.

Chapter 2

On a numerical subgrid upscaling algorithm for Navier Stokes Brinkmann equations

2.1 Introduction

The discussion in this Chapter is motivated by the fact that the demands from industry regularly pose new challenging problems to applied mathematics. In many cases the existing algorithms do not work, and new specialized algorithms for classes of industrial problems are needed. In line with Chapter 1, we still focus our discussion on developing algorithms for a class of filtration problems (filtering solid particles out of liquid). As it was observed, numerical simulations allow significant reduction in time and costs for design of new filter elements with proper flow rate - pressure drop ratio. Additionally, it is observed that the CFD simulations assisting this design are characterized by two peculiarities:

- High accuracy for the flow velocity within a filter element is not required, as long as the pressure drop over the complete filter element is properly computed.
- 30-36 simulations with different flow rates and different viscosities have to be performed for each geometry to evaluate the performance of the filter element at different flow conditions.

For this chapter, our aim is to develop efficient algorithms for this particular class of problems and to account for the peculiarities mentioned above.
In the case of liquid filtration, the flow is usually laminar, and it is described by (Navier-)Stokes-Brinkman system of equations (1.1) discussed in Chapter 1.

Most of the filters are characterized by the complicated shape of the filtering media (e.g. pleats and/or perforated porous layer/s), and/or by the complicated shape of the filter element housing (e.g. ribs, perforated inner cylinder, etc.). Some examples are illustrated in Figure 2.1.

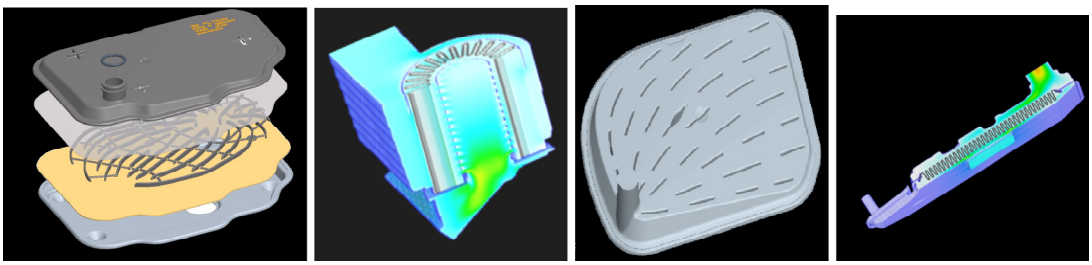


Figure 2.1: Some examples of the complex geometries and complicated shapes of the filtering media.

The existing commercial (e.g. Gambit, http://www.fluent.de) and academic (e.g. Netgen, http://www.hpfem.jku.at/netgen/) grid generators, are often unable to generate grids in such complicated domains, even when an experienced researcher is the user. In many cases, the generated grid is either of very bad quality, or the grid cannot be generated at all. Most engineers working on the design of filter elements are not experienced in grid generation techniques, and an alternative approach is needed in this case. The grid generation procedure works robustly if one restricts to voxel or brick elements, and therefore we chose this approach. A pre-processor based on the level set method (cf. [64]) is used to process the given CAD data for attaining the assembly of a filter housing in the form of a computational domain (union of voxel or brick control volumes). Different conventions can be used to characterize a control volume by a certain material type. Most commonly, the material type of its center point is assigned to the whole CV, or the dominating material type in the control volume is assigned to it. This grid generation approach is stable, and a fluid flow solver using the aforementioned grid generator is successfully used for a variety of industrial applications (cf. [31, 56, 57]). However, in general, such types of grids contain more elements: in certain cases, a very fine grid has to be used to accurately resolve all geometrical features, which in turn results in a large number of elements.

For cases where geometrical features are at different scales, adopting approaches for multiscale problems can provide an increase in efficiency of flow algorithms. Recall that Darcy equation can be rigorously derived as a macroscopic model for flow in porous media in the case of periodic or statistically homogeneous porous media, starting from Stokes system of equations at pore scale (cf. [48]). Depending on the the porosity, Allaire [2] homogenizes Stokes problem to Darcy or to Brinkman system. Clearly, the homogenization approach works under very strong restrictions, i.e. periodic or statistically homogeneous porous media, Stokes (and not Navier-Stokes) system at pore scale. The homogenization theory works in the case of scale separation, and it allows for a drastic reduction of the computational costs: for example, only one auxiliary problem is solved in a periodicity cell on the fine scale, its solution is post-processed to compute the coefficients of the macroscopic equation, and further the macroscopic equation is

solved at the coarse scale. The coefficients of the macroscopic equation in this case are called upscaled, homogenized, or effective coefficients. In the cases when the homogenization theory does not work, its ideas are still often used within numerical upscaling approaches, such as Multiscale Finite Element Method [49], Mixed Multiscale Finite Element Method, Mixed MsFEM, [43], Multiscale Finite Volume Method [16, 45, 59], Subgrid Method [5, 6, 54, 87]. Another approach which serves as a building block for numerical upscaling procedures is the volume averaging approach combined with Representative Elementary Volume, REV, concept (cf. [63]). Unlike the homogenization theory, this approach is not based on asymptotic expansions, but on volumetric averaging of functions and their derivatives. For example, if a block of porous medium is large enough (i.e. representative), its Darcy permeability is defined from the requirement that the macroscopic pressure drop is equal to the microscopic pressure drop computed by solving Stokes problem at pore level.

In the case of saturated flow in porous media, homogenization is studied either in connection with the mesoscopic and macroscopic Darcy models (upscaling elliptic equation with oscillating coefficients to macroscopic elliptic equation, i.e. upscaling Darcy to Darcy (cf. [60]), or in connection with impermeable porous matrix and fluid flow in the (connected) pore space, i.e. upscaling Stokes to Darcy (cf. [48]). To the best of our knowledge, upscaling of Stokes-Brinkman system at mesoscale to some macroscale system of equations is not studied in the literature. In this respect, it is worthwhile to note that for the case of industrial filters under consideration, the appearing multiscale problems contain fluid, solid, and porous regions.

In this chapter, we discuss a subgrid upscaling algorithm for the flow equations. Essentially we are working with slow flows, namely Stokes-Brinkman regimes at mesoscale, but also Navier-Stokes regimes are carefully computed in some cases. The subgrid approach was recently used in some simpler applications, namely the Darcy problem (single phase flow in porous medium) (cf. [9], [26], [87]). In the subgrid approach one solves a problem on a coarser grid, but accounts for the unresolved geometrical features by solving local auxiliary problems on finer grid in all, or in some of the coarse cells. This chapter deals with applying a similar approach in solving the incompressible Stokes-Brinkman equations in highly complex domains. The numerical solution for such systems is computationally expensive in terms of memory usage and computational time, and the subgrid approach is developed to compute a reliable pressure drop at reasonable computational costs.

The remainder of the chapter is organized as follows. A one-scale model (it can be also called single grid model), i.e. the Stokes-Brinkman system of equations, is described in Section 2.2. It includes a short description of the Finite Volume discretization, and the Chorin projection method employed for solving the model numerically. Section 2.3 is devoted to introducing the concept of *quasi-porous* coarse cells and to the description of the subgrid algorithm for the Stokes-Brinkman system. Due to lacking theoretical

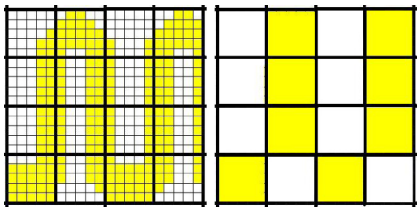


Figure 2.2: Illustration of an example of quasi porous cell/s where the fine scale resolves the geometrical details and the coarse scale does not.

results for the upscaling of the Stokes-Brinkman system, we perform a numerical study of this approach. In Section 2.4, we present validation results for the developed subgrid upscaling procedure. Results from numerical simulation of industrial filters are also presented in this Section. Finally, some conclusions are drawn.

2.2 Governing equations and single grid algorithm

The fine scale system includes the Stokes-Brinkman system of equations, as described in detail in Chapter 1, in a complicated 3D domain Ω, consisting of porous, solid and fluid subdomains, i.e. $\Omega = \Omega_P \cup \Omega_S \cup \Omega_F$.

$$\underbrace{\rho\frac{\partial \vec{u}}{\partial t} - \nabla\cdot(\mu\nabla\vec{u})\vec{u}}_{Stokes} + \overbrace{\mu\tilde{\mathbf{K}}^{-1}\vec{u} + \underbrace{\nabla p = \vec{\tilde{f}}}}^{Darcy} \tag{2.1}$$
$$\nabla\cdot\vec{u} = 0.$$

The notations follow from Chapter 1. The flow domains considered in the current work are of different geometric characteristics, as shown in Figure 2.1. The governing equations are solved subject to the following boundary conditions which are typical in filtration problems. At the inlet of the free flow region, a velocity profile is specified. At the outlet, zero pressure is imposed. No slip boundary conditions are imposed elsewhere on walls.

The coarse scale system also employs the Stokes-Brinkman system of equations but with upscaled permeability for a specified coarse grid, such that the Equation (2.1) is replaced with

$$\rho\frac{\partial \vec{u}_0}{\partial t} - \nabla\cdot(\mu\nabla\vec{u}_0) + \mu\tilde{\mathbf{K}}_{eff}^{-1}\vec{u}_0 + \nabla p_0 = \vec{\tilde{f}}_0 \tag{2.2}$$

where $\vec{u}_0$, p_0 denote the coarse scale velocity and pressure respectively. Here $\tilde{\mathbf{K}}_{eff}$ stands for the effective upscaled permeability, and the details of its computations are given in Section 2.3.

Single grid algorithm

Lets recall from Chapter 1 that the computational domain is a connected union of control volumes (CVs), where each CV is a brick. The equations are discretized by a finite volume method. Collocated arrangement of the unknowns $\vec{u}$ and p is used, i.e. the unknowns are assigned to the centres of CVs. Chorin method [21] for (Navier-)Stokes equations [38, 41], with a proper modification for (Navier-)Stokes-Brinkman case [23], is used as a projection method decoupling velocity and continuity equations. Momentum equations are solved to obtain an approximation for the velocities. Pressure correction system is then solved as a projection step after which further corrections are found to fulfill the continuity equation. To accelerate the computations, the algorithm is also parallelized, as discussed in Chapter 1. A detailed description of the sequential and parallel numerical algorithm was given in Chapter 1.

2.3 Subgrid Algorithm

In this section, a subgrid algorithm for the Stokes-Brinkman system is discussed. The developed algorithm is also (carefully) used, in some cases, for the Navier-Stokes-Brinkman system of equations. This is done in the sense of iterative upscaling or local-global iterations (cf. [30] for the iterative upscaling for the Darcy problem). The goal is to develop an algorithm which computes not only the pressure drop across a filter element on a relatively coarse grid, but also preserves the pressure drop accuracy corresponding to a finer grid. The idea is to account for a subgrid resolution on the coarse grid by solving proper local auxiliary problems on the fine grid. The solution is further processed to calculate the permeability of the quasi-porous coarse cells.

For a given computational domain, Ω, we consider a fine grid and a coarse grid. The fine grid is assumed to contain all important geometrical details, but solving the Stokes-Brinkman system of equations on this grid is memory and CPU intensive, or even impossible. As discussed in the Introduction, a material type (in our case fluid, solid, or porous) is assigned to each fine grid cell during the pre-processing stage. The coarse grid has a size, such that the problem is solvable at acceptable computational costs. Each coarse cell is a union of fine grid cells. Each coarse cell is considered, and the coarse grid cells containing mixture of solid & liquid, or liquid & porous, or solid & porous, or solid & liquid & porous, fine grid cells, are defined as *quasi-porous* cells, for which effective (upscaled) permeability tensors have to be computed. It is assumed that the Stokes-Brinkman equations describe the flow on the fine grid and the coarse grid. In the latter case, the coarse scale permeability is the effective (upscaled) permeability tensor $\tilde{\mathbf{K}}_{eff}$. Currently, the diagonal permeability tensor i.e. $\tilde{\mathbf{K}}_{eff} = \{K_{11}, K_{22}, K_{33}\}$ is considered, and the extension to the full tensor is ongoing work.

The computation of upscaled coefficients requires solving the local fine scale problem

in some coarse grid cells. Two approaches are considered:

- The upscaled (effective) permeability is computed for each individual *quasi-porous* coarse grid cell. Similar to the block-permeability upscaling procedure for single phase flow in porous media (cf. [88]), the localization boundary conditions are specified on its boundary (see discussion below for more details; possible over-sampling is also discussed below);

- Alternatively, certain union of coarse cells is considered as a single block for which auxiliary problem is solved on the underlying fine grid. Thereafter, the upscaled (effective) permeability is assigned to each of the coarse cells forming the block. This approach is motivated by the Representative Elementary Volume concept, where a reasonably large heterogeneous (at a fine scale) volume has to be considered, in order to assign effective (upscaled) coarse scale properties to it.

The upscaling approach considered here is similar to the one used by Durlofsky et al.[30] for the elliptic pressure equation. The Darcy law is used there to compute the effective (upscaled) permeability of a coarse cell from the averaged fine grid pressure and velocity. The different types of (localization) boundary conditions can be specified for the auxiliary problems. Periodic boundary conditions suit very well for periodic geometries. The pressure drop in one of the directions, and no flow (or symmetry) in the remaining directions, are often used. In the homogenization approach considered in [2], the constant velocity at infinity is prescribed as the boundary condition for the auxiliary problem. It is wellknown that the choice of the localization boundary conditions plays an important role in numerical upscaling methods, and we are currently working on comparing results for different (localization) boundary conditions. This work, however, will be reported elsewhere. Currently, the following local boundary conditions are considered for the auxiliary problems for the Stokes-Brinkman system:

- inflow velocity U_{in} at the inlet face (for the current direction);

- outflow b.c. at the outlet face: $p = P_{out}$, $\frac{\partial \mathbf{u}}{\partial \mathbf{n}} = \mathbf{0}$;

- symmetry b.c. elsewhere.

The auxiliary problems are solved for each direction d with the numerical algorithm 1 in Chapter 1. Next, the average inlet pressure P_{in} is computed from the local fine scale solution and is used to obtain the effective permeability K_{dd} of the quasi-porous cell in direction d from the Darcy law [63]:

$$K_{dd} = \frac{\mu U_{in} L_d}{P_{in} - P_{out}}, \tag{2.3}$$

where L_d is the distance between the inlet and outlet faces. The use of this approach is motivated by the fact that we are interested in computing the accurate pressure drop

of the complete filter element. By calculating the effective permeability using Equation (2.3), we account for the resistance of the geometrical features which were otherwise unresolved on the coarse computational grid.

Implementation of the subgrid method

Here, we shortly discuss the Subgrid method. The algorithm consists of four main components:

1. Selection of the quasi-porous coarse cells
2. Solution of the auxiliary problems with a fine grid resolution on selected coarse cells using Algorithm 1 with boundary conditions, as specified above
3. Computation of the effective permeabilities for the quasi-porous cells
4. Solution of the full coarse scale problem with Algorithm 1, using the calculated effective permeabilities.

Step (1) includes a mapping of each coarse cell onto the underlying union of fine grid cells. If the composition of fine grid cells contains a mixture of fluid and/or porous and/or solid, it is marked with a 'quasi-porous' flag. Alternatively, at this stage, a block of coarse cell can be specified as quasi-porous.

Step (2) defines the local auxiliary problem on the quasi-porous coarse cells. The same space discretization, i.e. the finite-volume method, is considered on the fine scale and a solution is computed. Note, that the additional layers can be added to a quasi-porous cell. In upscaling techniques, this is known as *oversampling*. Mostly, the domain is extended by a 'border ring' with fine grid cells. We consider additional fine scale bordering layers of type 'fluid' on a specified boundary. The number of additional layers is denoted by l in the next section. If $l = 0$, it is a purely local auxiliary solve. If $l = 1$, it means that the auxiliary problem is solved in coarse cell extended with the boundary layers in the specified direction, as shown on Figure 2.5(d).

Step (3) assigns the effective permeability tensors to the coarse cells, postprocessing the auxiliary fine grid solution computed from step (2). Steps (2) and (3) are performed for every quasi-porous cell selected in step (1). Note, that in the case when the coarse cells/blocks are periodic (e.g., this is the case when each coarse quasi-porous block contains one pleat), the auxiliary problem needs to be solved once for each type of quasi-porous coarse cells. Furthermore, the once computed permeabilities are reused in the computations with the same geometry, but with a different flow rate or viscosity (recall that 30-36 simulations have to be performed for each filter element geometry).

Step (4) includes the implementation of the standard algorithm (1) which solves the Stokes-Brinkman system of equations on the coarse grid.

Remark 1: (1) The same solver can be used for solving global coarse scale problem and auxiliary problems on the fine grid. (2) Different stopping criteria can be specified for the local and global problems.

Remark 2: If the number of quasi-porous blocks are too large, Steps (2)-(3) become the most time consuming. Implementation is modular-based and current work in progress includes parallelizing the subgrid method, such that the auxiliary problems are distributed to different processes for faster computations.

2.4 Results and validation

The results from numerical simulations are presented in this section. The performed simulations can be divided into three groups:

- Validation for the upscaled permeability computation and for thin porous layer (Subsections 2.4.1 and 2.4.2). Stokes flow around a spherical obstacle is considered in the Subsection 2.4.1 to check the influence of the used voxel grid as well;

- Numerical study of Brinkman to Brinkman upscaling. As mentioned earlier, the upscaling approaches for the Stokes and Darcy flows are theoretically considered in the literature, and due to the lack of theoretical results for the upscaling of Brinkman equation, we perform a numerical study here;

- Simulations for the industrial filters with complicated geometry of the porous media (perforated layer in one case, pleats in the other case), and complicated geometry (solid mesh supporting the perforated layer in one case, and complicated filter element housing in the other case).

Whenever possible, the results from the simulations with the upscaled equations are compared with the results obtained with the fine grid problem for the same geometry. This is the methodology employed for the validation of the developed subgrid algorithm. It should be noted that the single grid simulations have already been validated against measurements (cf. [69]).
The computations and CPU time measurements were performed on dedicated node of Fraunhofer ITWM cluster 'Hercules' with dual Intel Xeon 5148LV ('Woodcrest') processor (2.33 GHz).

2.4.1 Permeability for a periodicity cell: flow around a sphere

For better understanding of the subgrid algorithm, an example of an auxiliary problem for a quasi-porous coarse cell is considered. Consider a cube occupied by the fluid with a sphere-shaped solid obstacle inside, as shown in Figure 2.3. Suppose that we consider

a domain which is a periodic arrangement of such cells. Permeability for such a geometry is computed many times via homogenization and other upscaling approaches, and it is also measured. Our goal for solving the Stokes flow in this geometry is twofold. i.) From one side, we want to check if the boundary conditions for the auxiliary cell problems and REV type approach for calculating the upscaled permeability, as described above, gives good results; ii) Additionally, we want to check the influence of the voxel-based discretization grid that is used to discretize the sphere.

The auxiliary problem is solved on the fine grid resolution scale of 0.25mm with the following input data: cube size - $L = 12\,\text{mm}$, with different sized spheres with radius r; inflow velocity $U_{in} = 1.1574\,\text{mm/s}$, $P_{out} = 0\,\text{kg}/(\text{mm}\,\text{s}^2)$, density$(\rho) = 1.0 \cdot 10^{-7}\,\text{kg/mm}^3$, viscosity$(\mu) = 1.0 \cdot 10^{-4}\,\text{kg}/(\text{mm}\,\text{s})$. Table 2.4.1 shows the results obtained by our algorithm with permeability computed using Darcy law, denoted by $\mathbf{K}_{Darcy}$. The computed values are compared with the results on unit cell problem reported by Griebel and Klitz [42] and Sangani and Acrivos [82], denoted by $\mathbf{K}_{Cell}$ and $\mathbf{K}_{Experiment}$ respectively. It is observed that the computed results are in good comparison with the cited results. A systematic study on grid resolution was performed, and it concluded that the error reduces with finer grid resolutions where the voxel-based disretization better approximates the solid sphere.

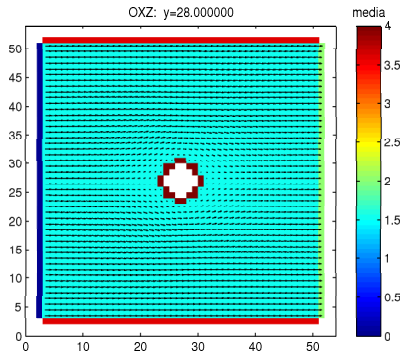


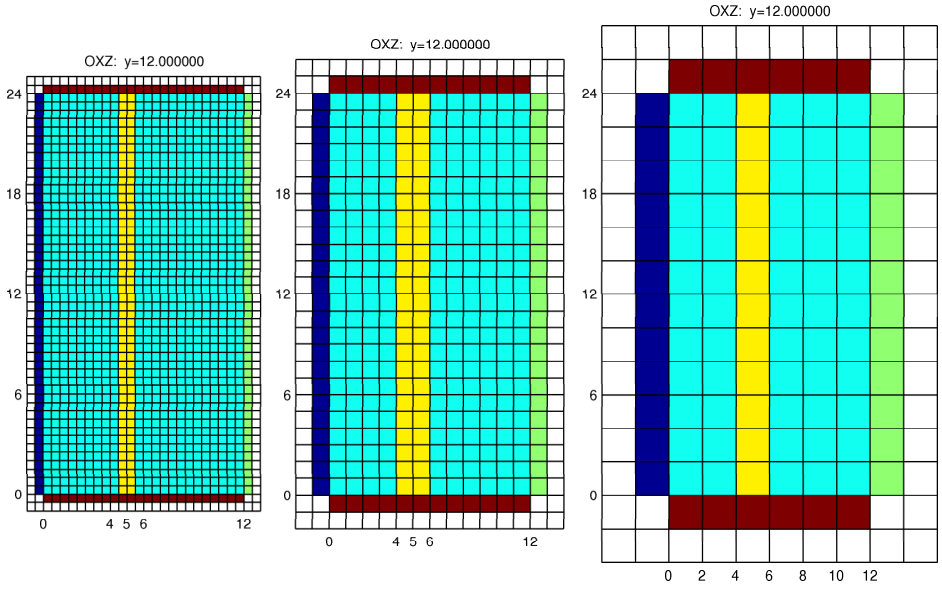
Figure 2.3: Cube with solid sphere inside. Inlet, outlet and symmetry walls are marked with blue, green and red colors respectively.

r	$\mathbf{K}_{Darcy}$	$\mathbf{K}_{Cell}$ (Griebel, Klitz)	$\mathbf{K}_{Experiment}$ (Sangani, Acrivos)
0.2	1.18×10^{-1}	1.23×10^{-1}	1.23×10^{-1}
0.25	7.34×10^{-2}	7.40×10^{-2}	7.46×10^{-2}
0.3	4.35×10^{-2}	4.46×10^{-2}	4.45×10^{-2}
0.35	2.48×10^{-2}	2.52×10^{-2}	2.52×10^{-2}

Table 2.1: Comparison of permeability in a 3D array of spheres (cell problem) for different radii

2.4.2 Channel filter with the single porous layer

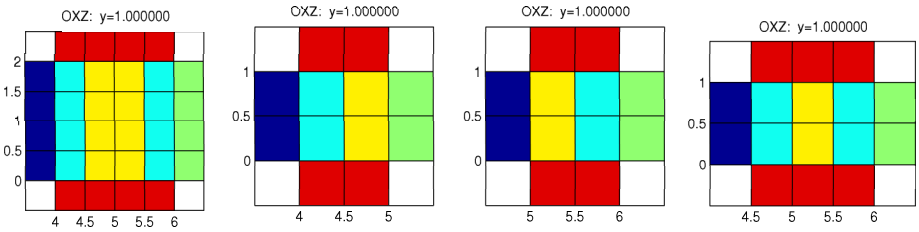
This example is chosen to be a simplified filter element geometry (parallelepiped) with a thin layer of porous media. The computational domain is $\{(x,y,z) : 0 \le x \le 12, 0 \le y \le 24, 0 \le z \le 24 \text{ mm}\}$, with a single porous layer $\{(x,y,z) : 4.5 \le x \le 5.5, 0 \le y \le 24, 0 \le z \le 24 \text{ mm}\}$, as shown in Figure 2.4. On coarse grids, when the step size of the grid is bigger than the thickness of the layer, the latter can not be correctly resolved. The thickness of the filtering medium is correctly resolved on 0.5 mm grid. However, on grid 2 mm, the pre-processor provides a porous layer with 2 mm thickness (recall that the pre-processor assigns to a grid cell the material type of its centre point). Solving problem (2.1) with $U_{in} = 28.935185\,\text{mm/s}$, $\rho = 1.0 \cdot 10^{-7}\,\text{kg/mm}^3$, $\mu = 1.0 \cdot 10^{-4}\,\text{kg/(mm s)}$, and isotropic permeability $K = 1.0 \cdot 10^{-4}\,\text{mm}^2$, we obtain (correct) pressure drop of $28.95\,\text{kg/(mm s}^2)$ on 0.5 mm grid, and a twice bigger (wrong) value, i.e. $57.88\,\text{kg/(mm s}^2)$ on 2 mm grid. Note, that the considered flow is essentially one-dimensional one and one can easily find its solution.



(a) 0.5 mm grid (b) 1 mm grid (c) 2 mm grid

Figure 2.4: Channel filter with the single porous layer.

Application of the subgrid algorithm for 2 mm coarse and 0.5 mm fine grids gives 144 (12×12) quasi-porous cells, which are all identical in this case, as shown in Figure 2.5 (a). Solution of the auxiliary problem in X direction gives the value of permeability $K_{11} = 2.0 \cdot 10^{-4}$ (the other components of the permeability tensor are not important in this case). Using the obtained permeability for the quasi-porous cells, we solve the problem on the 2 mm coarse grid, and get the correct pressure drop, i.e. 28.95.



(a) 2 mm (b) 1 mm *left* (c) 1 mm *right* (d) *right* + 1 layer

Figure 2.5: Auxiliary problems for 2 and 1 mm quasi-porous cells in X direction for channel filter with the single porous layer.

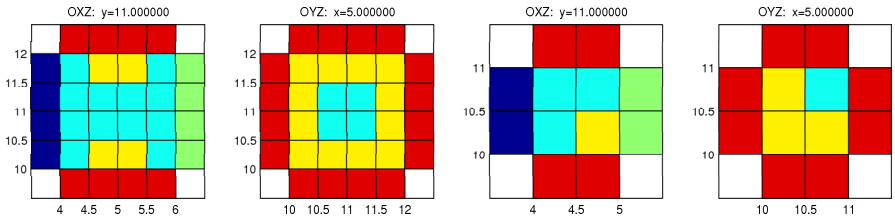
Including additional layers at inlet and outlet

Using 1 mm size for the coarse grid, we get two different geometries of quasi-porous cells in X direction, as illustrated in Figure 2.5 (b) and (c). Solving auxiliary problems for these geometries gives wrong values of the upscaled permeability in X direction: $K_{11} = 1.60 \cdot 10^{-4}$ for *left* cell and $K_{11} = 2.67 \cdot 10^{-4}$ for *right* cell. This is because the fine porous cells are touching the outlet of auxiliary problem in first case, and the inlet in the second, which does not allow full development of the flow. To deal with this issue, we have an option of adding 'border rings', which are a specified number of layers of fine fluid cells before the inlet and/or outlet of the actual domain in any specified direction. Figure 2.5 (d) shows the auxiliary problem for the *right* 1 mm cell with 1 additional fluid layer before the inlet. Solving this auxiliary problem gives correct value of upscaled permeability: $K_{11} = 2.0 \cdot 10^{-4}$. The same value is obtained for the *left* cell with additional fluid layer before the outlet. Finally, solving the global problem on the coarse 1 mm grid with the upscaled permeabilities gives correct value of the global pressure drop, i.e. 28.95.

2.4.3 Channel filter with single porous layer with hole

As a next example, the same filter geometry is considered as in subsection 2.4.2 but with a small hole in the porous layer - $\{(x,y,z) : 4.5 \leq x \leq 5.5,\ 10.5 \leq y \leq 11.5,\ 10.5 \leq z \leq 11.5 \text{ mm}\}$ and with $U_{in} = 2.8e-03$. This hole is correctly resolved on 0.5 mm grid. On 2 mm grid, this $1 \times 1 \times 1$ mm hole is exactly in the center of $2 \times 2 \times 2$ mm coarse cell, as illustrated in Figure 2.6 (a) and (b). Figure 2.6 (a) is a view along the channel, whereas Figure 2.6 (b) is a view across the channel. Therefore, as opposed to the previous case, this cell is fluid on 2 mm grid, and the cross section of the hole as well as the thickness of the porous layer are wrongly depicted. On 1 mm grid, all 8 $1 \times 1 \times 1$ mm cells, covering the hole, have a porous center and therefore remain porous, as shown in Figure 2.6. Hence, 1 mm grid resolution remains the same as shown in Figure 2.4(b) in Section 2.4.2.

In Table 2.2, results for the different grid resolutions using the single grid Algorithm 1



(a) 2 mm (b) 2 mm (c) 1 mm (d) 1 mm

Figure 2.6: Auxiliary problems for 2 and 1 mm quasi-porous cells in X direction for channel filter with the single porous layer with a hole.

are presented. On the 2 mm grid, the hole is too big, therefore we get relatively small pressure drop. The grids with stepsize 0.5 and 0.25 mm resolve the hole exactly (see Figure 2.6 (b)). Clearly it is expensive to compute accurate solution on a single coarse grid, and a subgrid approach is the remedy for efficient simulations.

Grid resolution	Number of CVs	Memory [MB]	CPU time [s]	Pressure drop [kg/(mm s^2)]
2 mm	864	0.80	2	0.00353
0.5 mm	55296	33.73	206	0.01486
0.25 mm	442368	247.1	2489	0.01669

Table 2.2: Simulation results for channel filter with single porous layer with hole (obtained with numerical algorithm 1).

Next, we discuss the results of the subgrid algorithm for this problem. In Table 2.3, we present the results obtained for 2 mm coarse and several (0.5, 0.25 mm) fine scale combinations. Calculations were done with different number of additional fluid layers l used for the solution of the auxiliary problems. We show the computed values of the upscaled permeability K_{11} for the quasi-porous cell with hole and the global pressure drop dP. Also, the CPU time T_{SG} of preprocessing step only is given, because the subsequent solution of system on the coarse 2 mm grid was taking less than 10 seconds for all shown cases. The maximal memory used by the solver in all cases was 1.08 MB (coarse grid solver + upscaled permeability tensors).

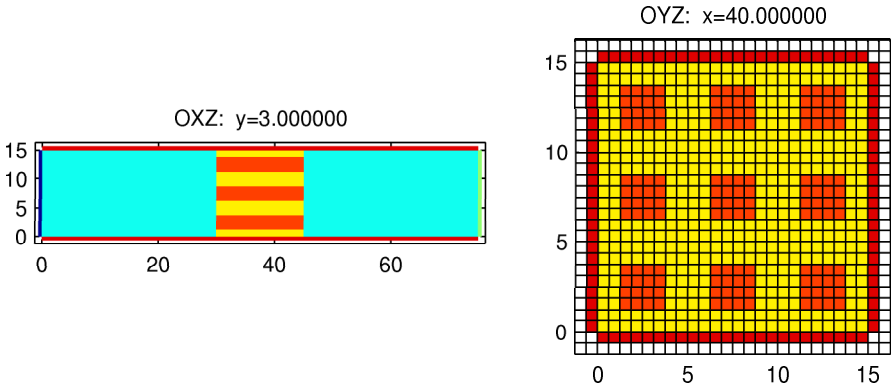
2.4.4 Channel filter with periodic porous layer

The next example is a channel/parallelepiped filter $\{(x,y,z) : 0 \le x \le 75,\ 0 \le y \le 15,\ 0 \le z \le 15 \text{ mm}\}$ with porous layer $(30 \le x \le 45)$ consisting of two porous materials with a periodic structure, as shown in Figure 2.7.

The first porous material(in yellow) in Figure 2.7 has permeability $K = 1.0 \cdot 10^{-4}\,\text{mm}^2$. The second porous material (in red color) is more permeable with permeability $K =$

Scales	l	K_{11}	T_{SG}	dP
2/0.5 mm	0	0.0226	14	0.0168
2/0.5 mm	2	0.0272	140	0.0156
2/0.5 mm	4	0.0273	203	0.0156
2/0.25 mm	0	0.0198	155	0.0177
2/0.25 mm	2	0.0217	345	0.0172
2/0.25 mm	4	0.0218	516	0.0171

Table 2.3: Simulation results for channel filter with single porous layer with hole (obtained with Subgrid algorithm). l is the number of additional fluid layers. K_{11} is the upscaled permeability obtained for quasi-porous cell with hole. T_{SG} is CPU time of preprocessing step. dP is the obtained global pressure drop.



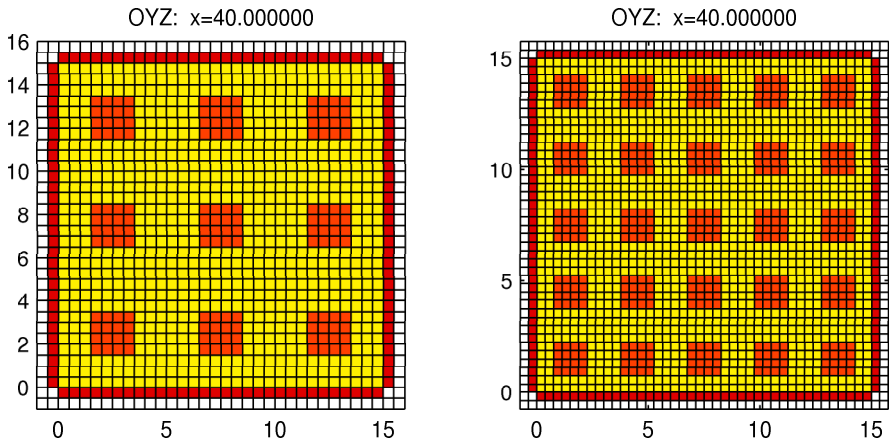
(a) (b) (3×3: 8×8)

Figure 2.7: Channel filter with two peridic porous layers (3×3) represented in yellow and red. (a) and (b) represent two different cross sections of $x-z$ axis and $y-z$ axis of the filter.

$2.0 \cdot 10^{-4}\,\mathrm{mm}^2$. The problem (2.1) is solved with the same parameters as before.

For Figure 2.7, the results from the Algorithm 1 on the fine scale, $24 \times 24 \times 24$, are obtained. The pressure drop $dP = 80.005$ is obtained in 9.42 seconds. The grid is further coarsened 8 times, such that $8 \times 8 \times 8$ fine cells comprises of one coarse cell and the results using the coarse and the fine grid are obtained using the subgrid algorithm. The results obtained are $dP = 80.008$ in 4.55 seconds.

Figure 2.8 (a) shows a geometry with a fine scale discretization containing $30 \times 30 \times 30$ finite volumes. Compared to Figure 2.7(b), there is 1.5% more of the less permeable porous material and we expect a greater pressure drop across the filter element. Using the Algorithm 1, results in $dP = 86.212$ in 21.65 seconds. The grid is then coarsened 3 times, such that $10 \times 10 \times 10$ fine cells constitutes one coarse cell. Using both, fine and coarse scales, the subgrid algorithm results in $dP = 86.215$ in 8.39 secondss.



(a) (3x3: 10x10) (b) (5x5: 8x8)

Figure 2.8: Channel filter with peridic porous layer.

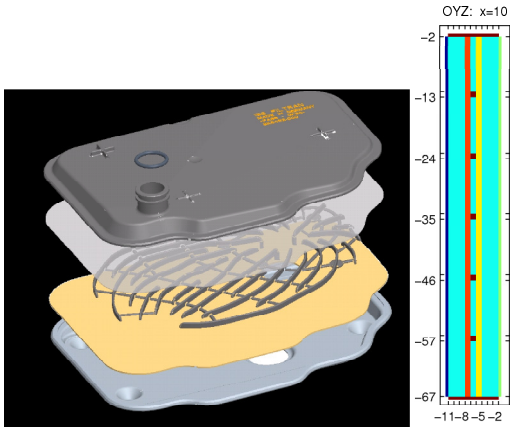
Figure 2.8 (b) illustrates five, instead of three, periods of the second type of the porous material which has a higher permeability. The geometry compares with the geometry shown in Figure 2.7. We see that pressure drop $dP = 80.012$ is obtained in 52.67 seconds using Algorithm 1 using the fine scale discretization of $40 \times 40 \times 40$ cells. The coarse discretization contains $5 \times 5 \times 5$ cells. Subgrid algorithm results in $dP = 80.015$ in 21.82 seconds.

Lastly, instead of three, periods of the second type of the porous material which has a higher permeability for the geometry that compares with the geomtery shown in Figure 2.8. We see that pressure drop $dP = 86.249$ is obtained in 124.96 seconds using Algorithm 1 using the fine scale discretization of $50 \times 50 \times 50$ cells. The coarse discretization contains $5 \times 5 \times 5$ cells. Subgrid algorithm results in $dP = 86.224$ in 41.23 seconds.

2.4.5 Channel filter with the combi layers

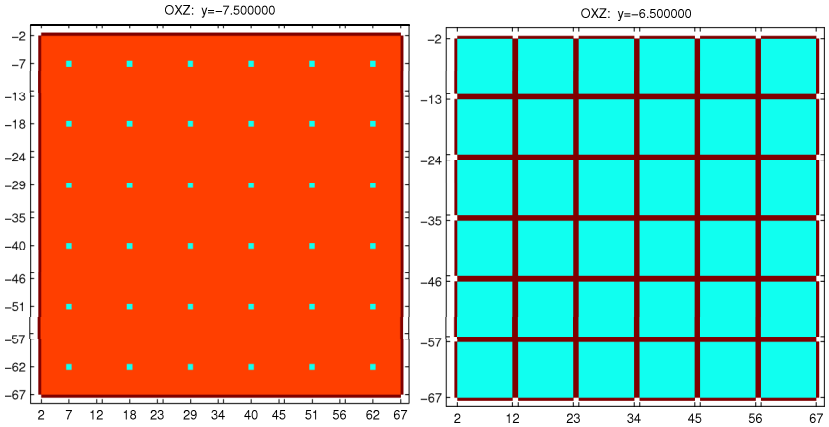
The next example is chosen to serve the needs of the design of new generation of filters as shown in Figure 2.9 (a) . It should be noted that oil filters are often equipped with a so called bypass option. The bypass prevents the oil to filter through the porous media, especially at cold temperatures (i.e. high viscosity). The idea behind such design was to prevent the destruction of the filter due to the high pressure in such cases. Instead, the oil bypasses the filtering medium via some small pipe. Obviously, the bypassed oil is not filtered in this situation. Alternatively, the new league of filters are designed without the bypass option, but with an additional layer of a fine perforated filter layer that allows the oil to flow through the holes at cold temperatures. Additionally, a coarse filter layer in the form of a solid mesh is added to filter out the large particles. With the support of the numerical simulations, the size of the holes and the distance between the two porous layers are designed such that most oil flows through the porous part of the first layer

at high temperature, and not through the holes. It is common understanding that the porous layers are the primary cause of the pressure drop across the filter. This is true only when the filter element is designed in a way that there is enough space between the inlet(bottom) and the porous layer, and between the porous layer and the outlet(top). Therefore, we study this filter in a simplified geometry, i.e. a simple channel geometry as shown in Figure 2.9 (b). The considered filter element is parallelepiped $\{(x,y,z) : 2 \leq x \leq 67, -11 \leq y \leq -2, -67 \leq z \leq -2 \text{ mm}\}$ with two filtering porous layers and a supporting solid mesh between them, as shown in Figure 2.9 (b). Additionally, the first porous layer ($-8 \leq y \leq -7$) has a set of 6×6 holes, as shown in Figure 2.10 (a), where each hole is a cube: $1 \times 1 \times 1$ mm, as illustrated in Figure 2.11 (a).



(a) (b)

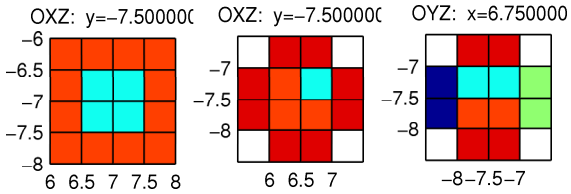
Figure 2.9: (a) A real industrial filter with multiple porous media layers. (b) Channel filter with the multiple porous layers (a simplified variant of the industrial filter to understand real processes).



(a) (b)

Figure 2.10: Cross sectional view of the complicated structure of the multiple porous layers in the channel filter.

When using the 2 mm resolution, the mesh layer is not captured at all and both porous layers have the wrong, doubled thickness, on this grid: $-8 \leq y \leq -6$ and $-6 \leq y \leq -4$, respectively. The 1 mm grid represents the channel and all three layers (including the solid mesh) correctly except for the holes in the first porous layer. As shown in Figure 2.11, all 4 $1 \times 1 \times 1$ mm cells, covering the hole, have a porous center and therefore are represented as porous in 1 mm grid. The 0.5 and 0.25 mm grids resolve the whole geometry of the filter exactly.



(a) (b) (c)

Figure 2.11: Resolution of one of the holes in first porous layer

System 2.1 is solved with the following flow parameters: $U_{in} = 3.944773\,\mathrm{mm/s}$, $\rho = 1.0 \cdot 10^{-7}\,\mathrm{kg/mm^3}$, $\mu = 1.0 \cdot 10^{-4}\,\mathrm{kg/(mm\,s)}$, isotropic permeability of first porous layer, $K = 4.2 \cdot 10^{-5}\,\mathrm{mm^2}$, and isotropic permeability of second porous layer, $K = 7.5 \cdot 10^{-4}\,\mathrm{mm^2}$. Note that in this test case the main flow is in Y direction.

In Table 2.4, the results from the numerical algorithm 1 for different grid resolutions are presented. It is due to the aforementioned errors in the geometry resolution for 2 to 1 mm grids, that the resulting pressure drop is over estimated. As one goes to finer resolutions, i.e. 0.5 to 0.25 mm, the geometry is resolved more accurately resulting in a more accurate pressure drop. However, computation on the finer grids come with the cost of extensive CPU time consumption and memory usage.

Grid resolution	Number of CVs	Memory [MB]	CPU time [s]	Pressure drop [kg/(mm s^2)]
2 mm	5445	4.42	63.54	16.762
1 mm	37400	26.49	465.02	11.627
0.5 mm	299200	182.70	28271.47	1.579
0.25 mm	2393600	1338.37	518425.0	1.762

Table 2.4: Simulation results for channel filter with multiple porous layers (obtained with Algorithm 1).

Next, we present the results from the subgrid algorithm for this problem. 2 mm resolution is too coarse to be used even as the coarse scale, where the 1 mm scale seems to be an appropriate choice. Alternatively, a bigger block of coarse cells can be used as an auxiliary problem for computing permeability. As discussed above, for 1 mm grid the

only cells, which will be detected as quasi-porous, are the cells covering the holes in first porous layer (see Figure 2.11 (a)). It means that the total of $4 \times 6 \times 6 = 144$ quasi-porous cells will be detected. As can be seen from Figure 2.11 (c) and (b), solving the auxiliary problem for the main component of upscaled permeability tensor - K_{22}, the flow will be almost aligned with the Y axis and we can expect good results even without the use of full permeability tensor. It is also clear that the additional fluid layers are needed for the auxiliary problems.

Scales	l	K_{22}	T_{SG}	T_{total}	dP
$1/0.5\,\text{mm}$	1	0.01092	54.8	781.5	1.6919
$1/0.5\,\text{mm}$	2	0.01177	93.4	838.3	1.6289
$1/0.25\,\text{mm}$	1	0.00588	883.3	1502.9	2.3862
$1/0.25\,\text{mm}$	2	0.00863	787.7	1464.8	1.9173
$1/0.25\,\text{mm}$	3	0.00960	993.0	1692.1	1.8101
$1/0.25\,\text{mm}$	4	0.00972	1126.3	1829.4	1.7984

Table 2.5: Simulation results for channel filter with combi layers (obtained with Subgrid algorithm). l is the number of additional fluid layers used for auxiliary problems. K_{22} is the upscaled permeability in flow direction obtained for one of the quasi-porous cells. T_{SG} is CPU time of preprocessing step. T_{total} is the total CPU time. dP is obtained global pressure drop.

In Table 2.5 we present the results obtained for two coarse/fine grids combination: $1/0.5$ and $1/0.25\,\text{mm}$. Calculations were done with different number of additional fluid layers l, Reynolds number Re, and accuracy ε used for solution of auxiliary problems. We show the obtained values of upscaled permeability K_{22} for one of the quasi-porous cells, CPU time T_{SG} of preprocessing step and the total solver time T_{total} (i.e. including the CPU time of subsequent solution on the coarse $1\,\text{mm}$ grid), and finally the global pressure drop dP. The maximal memory used by the solver in all cases was $26.68\,\text{MB}$ (coarse grid solver + upscaled permeability tensors).

The results show that the additional number of layers (oversampling) is an essential parameter in our case. Also, the choice of the coarse grid is important for the selection of quasi-porous cells.

2.4.6 Simulation of real industrial Filters - Pleated filter

Lastly, another example of a real filter with complicated filter media is considered. Figure 2.12 shows a pleated filter geometry on coarse and fine grids. The coarse grid does not resolve the shape of the filtering medium properly. Our aim is to solve the problem using Equation (2.2) on the coarse grid, and employ the subgrid algorithm described in Section 2.3 to compute the $\tilde{K}_{eff}$ for selected quasi-porous cells. Figure 2.12 (c) also illustrates the quasi-porous block on which fine scale auxiliary problem is solved. It

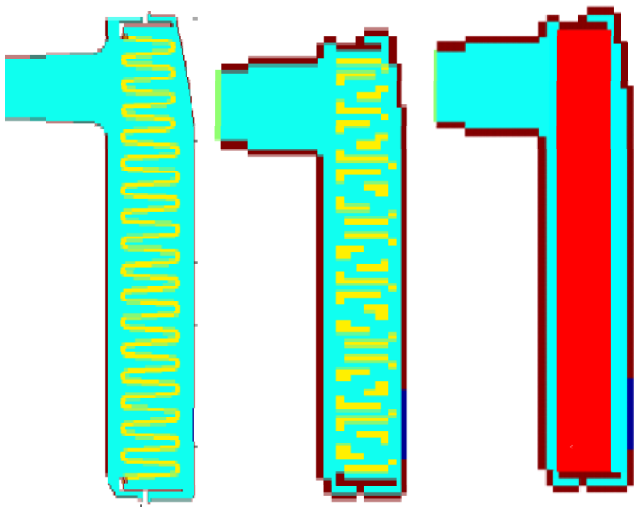


Figure 2.12: Pleats in the filter on different grids. Fine grid resolves the geometrical shape of the filter and the pleats, where as the coarse grid does not. Subgrid treats all pleats as a block of filtering medium with upscaled permeability computed using fine scale.

should be noted that the number of cells in this block are fairly less compared to solving the fine scale problem on the complete filter element. In fact, computing permeability from simulation in a significantly smaller block gives the same results. If all the pleats are of the same size, and the distances between them are equal, it is possible to consider only one small block containing one pleat. The permeability is computed for this block and then assigned to each coarse cell in the quasi-porous block from Figure 2.12 which contains all the pleats.

In Table 2.6, we present the results of solving this problem with the numerical algorithm 1 for two different grid resolutions. The coarse scale is of 1mm size and fine scale resolution is $0.5mm$. The results illustrate the extent to which the computational time for solving a complete problem could increase with finer grid resolutions. Almost 100 times more computational time is required for a problem that is twice finer in grid resolution. Note that different resolutions result in different geometries. For the coarse grid, not only are the shapes of the pleats not correctly accounted for, but also the inlet and outlet are not separated by the porous medium. In fact, the solution is sought for different geometries and for obvious reasons, the pressure drop on the coarse grid is an over estimation.

Note that the subgrid algorithm is employed as a preprocessing step for Algorithm 1. Moreover, in industry, the subgrid method benefits from the fact that for each fixed geometry, many simulations are done using different physical parameters, such as density, viscosity, inflow velocities etc. for assessing the filter performance. In this case, the subgrid method can be used only once and precomputed upscaled permeabilities for the

Grid	$dP\ (mbar)$	T_{SG}	$T_{total}\ (s)$
1mm	2700	-	1421
0.5mm	1874	-	149102
subgrid-1mm-0.5mm	1760	63580	1420

Table 2.6: Results for geometries described in Figure 2.12.

quasi-porous cells/blocks can be further used in subsequent simulations.

2.5 Summary

A subgrid method is introduced and further developed as a computational method for achieving acuurate solutions for our problem. It is mainly employed for cases where the coarse computational grid is unable to accurately account for the complicated geometry and filter media. Algorithm 1 discussed in Chapter 1 is used as the building block for solving auxiliary problems on the quasi porous cells and for solving the global coarse scale problem, with special emphasis based on reusability of solvers. The method emphasizes on the determination of upscaled quantities for use in subsequent coarse scale global simulations. Fine scale solution is sought only on some quasi porous cell/s or collectively on a block (a collection of coarse cells). The results were presented for computer simulation experiments using three dimensional models of oil filters. It is observed that the cell/block permeability is strongly influenced by the resolving grid. CPU time and memory usage is reduced significantly using the subgrid method with the resulting desired accuracy of the pressure drop, which characterizes the flow through filters. In the next chapter, we propose another multiscale finite volume method for solving flow in porous media using the Stokes-Darcy system of equations. As opposed to upscaling of coarse scale operators, fine scale influences are captured via basis functions. It also addresses the limitation of the subgrid method related to its inability to resconstruct fine scale solutions.

Chapter 3

On a multiscale finite volume method for the Stokes-Darcy equations

3.1 Introduction

This chapter concerns another algorithm for the multiscale problems, namely the Multiscale Finite Volume (MSFV) method, which provides not only an accurate solution at the coarse scale, but also allows to reconstruct good approximations to the fine scale velocity. A number of industrial and environmental problems are characterized by fine scale heterogeneities and/or fine scale features of the geometry. The classical simulation techniques lack the potential to account for the fine scale heterogenities, and it is almost impossible or very expensive to solve such problems on feasible grids. To deal with this resolution gap, a variety of numerical multiscale approaches were developed in the last two decades. In the field of flows in heterogeneous porous media, these include Multiscale Finite Volume (MSFV) Method [59], Multiscale Finite Element Method (MsFEM) [49] and the references therein, Mixed MsFEM [1, 43], Heterogeneous Multiscale Method (HMM) [35], Subgrid approach [5, 6, 54, 87], etc. Iterative versions of some of these methods are suggested, aiming at achieving better accuracy, e.g., iterative MSFV (iMSFV) [16, 45], local-global iterations [30], etc. In fact, some of these iterative approaches can be considered as variants of domain decomposition method with special choice of the coarse space (cf. [74]). On the other hand, in the last decade, significant attention was devoted for developing efficient algorithms for coupled flows in plain and porous media, mainly in the case when the flow in the pure fluid (plain) region is described by the Stokes equations, and the flow in the porous media is described by Darcy equations. (cf. [28, 29, 79]). Most of these papers deal with flow tangential to the porous media, when the coupling is incorporated via the so called Beavers-Josef interface condition (cf. [58, 63]. The coupled Stokes-Brinkman system of equations were earlier considered in [55, 69]. The paper by Ehrhardt et. al. [36] also presents a

short summary on available models and interface conditions for coupled flows in plain and porous media.

In this chapter, the coupled steady state Stokes-Darcy system is considered. Note that the approach can easily be extended to the unsteady problems and to the Stokes-Brinkman model. However, this extension is not part of this thesis.

The coupled Stokes-Darcy equations:
Consider a computational domain Ω, which is occupied by the pure fluid region, denoted by Ω_F, and the porous region, denoted by Ω_P, such that $\Omega = \Omega_F \cup \Omega_P$. The boundary of Ω is denoted with ∂D, so that $\bar{\Omega} = \Omega \cup \partial\Omega$. The porous-plain interface is denoted by Γ_{FP}, where $\Gamma_{FP} = \bar{\Omega}_P \cap \bar{\Omega}_F$.

The flow in the porous region is governed by the Darcy equation and the continuity equation:

$$\vec{u} = -K\nabla p, \text{ in } \Omega_P \tag{3.1}$$

$$\nabla \cdot \vec{u} = 0, \text{ in } \Omega_P. \tag{3.2}$$

The steady state Stokes equations, together with the continuity equations, describe the flow in the pure fluid region Ω_F:

$$-\nabla \cdot (\mu \nabla \vec{u}) = f - \nabla p \text{ in } \Omega_F \tag{3.3}$$

$$\nabla \cdot \vec{u} = 0, \text{ in } \Omega_F. \tag{3.4}$$

Here $\vec{u} = (u, v)$ and p stand for the velocity vector and the pressure, respectively. Addtionally, μ and $\mathbf{K}$ denote the viscosity and the permeability tensor of the porous medium respectively.

In this chapter, we consider the case where the velocities in the fluid and porous regions have comparable magnitudes. The pressure gradient in the porous region is much larger than in the free fluid part. Most of the flow through the porous medium is perpendicular to the interface between the pure fluid region and the porous medium. In this case, according to [70], the following interface conditions apply:

$$\{\vec{u}_F \cdot \mathbf{n}\}|_{\Gamma_{FP}} - \{\vec{u}_P \cdot \mathbf{n}\}|_{\Gamma_{FP}} = 0, \tag{3.5}$$

$$\vec{u}_F \cdot \mathbf{t}|_{\Gamma_{FP}} = 0, \tag{3.6}$$

$$p_P|_{\Gamma_{FP}(porous)} = C, \tag{3.7}$$

where the indices $_F$ and $_P$ denote the quantities on the fluid and porous side of the interface Γ_{FP}. Moreover, C is an unknown constant, whereas $\mathbf{t}$ and $\mathbf{n}$ are the unit tangential and normal vectors respectively. To have a comparable velocity field in Ω_P and Ω_F, the pressure gradient in Ω_P should be much larger than in Ω_F. Although $p|_{\Gamma_{FP}}$ is not a constant and depends on the flow in Ω_F, its variation is negligible in comparison to the

pressure variation in Ω_P. In our case, C is determined from the pressure in the fluid part of the domain.

The boundary conditions on different parts of the boundary domain $\partial\Omega = (\partial\Omega_F \cup \partial\Omega_P) \setminus \Gamma_{FP}$ are chosen in the way that the problem is well posed. Specific choice of boundary conditions related to simulation of filtration problems will be discussed shortly in Section 3.4.

Targeted applications: As discussed in the previous chapters as well, our main focus is to discuss the filtration related processes. The developed algorithm targets mainly the industrial filtration problems, when the fine scale features of the geometry impose a challenge to the numerical methods. As an example, we revisit the new generation filter element as illustrated in Figure 3.1. The filter is composed of three layers of filtering media: the first one is a perforated highly efficient filtering medium, the second layer is a coarse supporting mesh, and the third layer is a lower efficiency/higher permeability filtering medium.

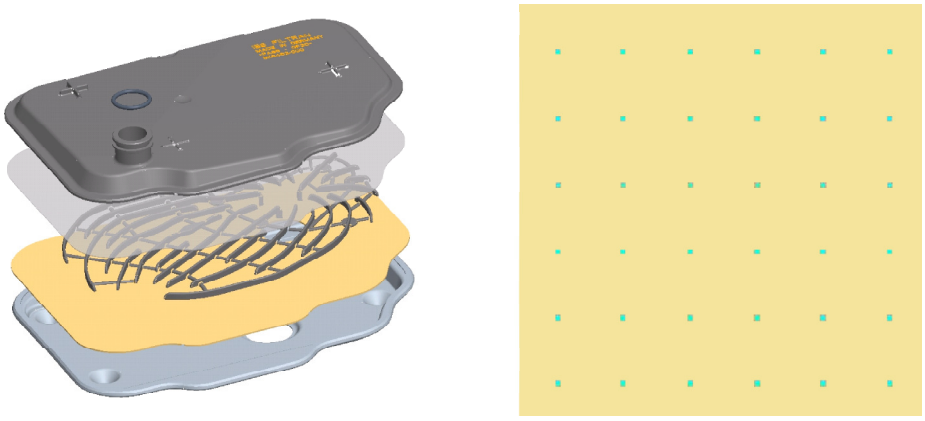


Figure 3.1: Left: A real industrial filter with multiple porous media layers. Right: An example of a perforated filter medium.

It is worthwhile to note that the developed algorithm is not restricted to the application of industrial filters only. Generally, the algorithm can be applied to any problem where

- the porous media is highly heterogeneous on the fine scale;
- the geometry is very complex on the fine scale

Goals and structure: The original MSFV method was developed for the simulation of flows in porous media. The main goal of this chapter is to extend the (iterative)MSFV approach for solving the coupled Stokes-Darcy system for the plain and porous media. The proposed algorithm can also be considered as a domain decomposition approach, which unlike most papers in this area, is applied to the momentum equations only. The

pressure equation, on the other hand, is solved in the full domain. More precisely, the domain is decomposed into the fluid/plain and porous domains. The momentum equations are solved independently for the Stokes and Darcy system on the decomposed domains, while the pressure equation is solved in the full domain. This approach is particularly suitable for problems where the flow is perpendicular to the porous media and for cases where it is better to consider the pressure changes in the full domain.

The remainder of the chapter is organized as follows. The next section shortly presents the building blocks of our algorithm. These are the single grid algorithms for the Stokes and the Darcy problems discretized on a staggered grid described in Subections 3.2.2 and 3.2.3. The MSFV and iMSFV methods for the Darcy problem are described in Subection 3.2.4. Section 3.3 presents the multiscale finite volume algorithm for the coupled Stokes-Darcy system. In Section 3.4, results from the numerical simulations are presented to validate the approach and to study its performance. Finally, some conclusions are drawn.

3.2 Single grid discretizations and algorithms for the independent Stokes and Darcy problem

3.2.1 Grid and grid notations

Consider a rectangular domain

$$\bar{\Omega} = \{(x,y)|0 \leq x \leq L_1; 0 \leq y \leq L_2\}, \;\; \bar{\Omega} = \Omega \cup \partial\Omega.$$

A uniform staggered [38] Cartesian grid of size $h = \frac{L_1}{N_1-1} = \frac{L_2}{N_2-1}$ is introduced in Ω

$$\bar{\omega}^p = \{\mathbf{x}_{i,j}|\mathbf{x}_{i,j} = (x_i, y_j),\; i = 0,1,...N_1; j = 0,1,...N_2\}$$

where

$$x_0 = 0, x_1 = \frac{h}{2},\; x_{i+1} = x_i + h,\; i = 1,2,\ldots,N_1-1, x_{N_1} = L_1;$$

$$y_0 = 0, y_1 = \frac{h}{2},\; y_{j+1} = y_j + h,\; j = 1,2,\ldots,N_2-1, y_{N_2} = L_2$$

Furthermore,

$$\bar{\omega}^u = \{\mathbf{x}_{i+\frac{1}{2},j}|\mathbf{x}_{i+\frac{1}{2},j} = (x_i + \frac{h}{2}, y_j),\; i = -1,0,\ldots,N_1-1;\; j = -1,0,\ldots,N_2-1\},$$

$$\bar{\omega}^v = \{\mathbf{x}_{i,j+\frac{1}{2}}|\mathbf{x}_{i,j+\frac{1}{2}} = (x_i, y_j + \frac{h}{2}),\; i = -1,0,\ldots,N_1-1;\; j = -1,0,\ldots,N_2-1\},$$

where

$$x_{0.5} = 0,\;\; x_{i+\frac{1}{2}} = x_i + 0.5h, i = 0,1,\ldots,N_1-1; x_{N_1} = L_1;$$

$$y_{0.5} = 0,\;\; y_{j+\frac{1}{2}} = y_j + 0.5h,\;\; j = 0,1,\ldots,N_2; y_{N_2} = L_2.$$

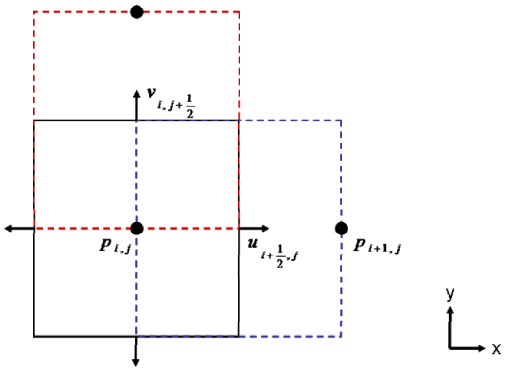


Figure 3.2: 2D blocks for pressure and velocities

The generalization to 3D is straightforward, and so is the usage of a non-uniform Cartesian grid. The discrete pressure is assigned to the centre of each cell, whereas the discrete velocities are specified on the faces, as illustrated in Figure 3.2 (cf. [38] for more details). The control volumes (CVs) assigned to the pressure (black line on Figure 3.2) will be called pressure CVs. It is assumed that the domain Ω is a union of pressure CVs, i.e., $L_1 = (N_1 - 1)h$, $L_2 = (N_2 - 1)h$. With some modifications, the extension of the presented algorithm to cell centered grid is also possible.
Furthermore, assume that a part of the domain Ω is occupied by the porous media, such that

$$\Omega = \Omega_F \cup \Omega_P.$$

It is assumed that Ω_F and Ω_P, are unions of pressure CVs. Next, we denote the pressure and velocity components of the grid in the fluid domain by

$$\bar{\omega}_F^p = \bar{\Omega}_F \cap \bar{\omega}^p, \quad \bar{\omega}_F^u = \bar{\Omega}_F \cap \bar{\omega}^u, \quad \bar{\omega}_F^v = \bar{\Omega}_F \cap \bar{\omega}^v.$$

Analogously, we denote the respective grids in the porous domain by

$$\bar{\omega}_P^p = \bar{\Omega}_P \cap \bar{\omega}^p, \quad \bar{\omega}_P^u = \bar{\Omega}_P \cap \bar{\omega}^u, \quad \bar{\omega}_P^v = \bar{\Omega}_P \cap \bar{\omega}^v.$$

Since the fluid and the porous domains are resolved with pressure CVs, $\omega_P^p \cap \omega_F^p = 0$. However, there are some nodes from the velocity grids which belong to both, the fluid and the porous velocity grids. These are the nodes residing on the interface between the plain and the porous media: $\bar{\omega}_F^u \cap \bar{\omega}_P^u \in \Gamma_{FP}$, $\bar{\omega}_F^v \cap \bar{\omega}_P^v \in \Gamma_{FP}$.

3.2.2 The Stokes problem

The 2D steady state Stokes equations (3.3) and (3.4) are discretized on Ω_F by the finite volume method (cf. [38]). The discretization is similar to the discretization discussed in Chapter 1 for the Navier-Stokes-Brinkmann equations. The only difference is that the pressure correction equation (3.13) is discretized on ω_F^p, while the momentum equations

(3.10) are discretized on ω_F^u and ω_F^v. Since the discretization directly follows from Chapter 1 and [38], we skip the discretization details, and write the discretized Stokes system in the form:

$$D\vec{u} + Gp = f, \text{ on } \bar{\omega}_F^u \times \bar{\omega}_F^v, \tag{3.8}$$

$$Div\vec{u} = 0 \text{ on } \omega_F^p. \tag{3.9}$$

Here, $\vec{u} = (u_{i+\frac{1}{2},j}, v_{i,j+\frac{1}{2}}), i = -1,0,...,N_1, j = -1,0,...,N_2$. Recall that the velocity component $u_{i+\frac{1}{2},j}$ is defined on $\bar{\omega}_F^u$, while the velocity component $v_{i,j+\frac{1}{2}}$ is defined on $\bar{\omega}_F^v$. The discrete pressure is defined on $\bar{\omega}_h^p$. The same notations for the discrete and continuous pressure and velocities are used here. The components of the discrete pressure gradient are defined on $\bar{\omega}_F^u$ and $\bar{\omega}_F^v$, respectively. The discretization of the gradient operator is denoted by G. The operator corresponding to the discretized viscous term in the momentum equation is denoted by D. Div denotes the discrete divergence operator. The boundary condition diescretization is included in the above operators.

There exists a variety of velocity-pressure decoupling approaches for the iterative solution of the Stokes system of equations (cf. [15, 34, 84] and the references therein). Here, we consider the SIMPLE algorithm [38, 75], which is one of the many decoupling methods. We will present it in the form given by [37, 75], as this formulation can later be easily coupled with the Darcy equation in the porous region.

Suppose that the pressure, p^k, is known after $k-$th iteration, then the velocity can be predicted using this pressure:

$$D\vec{u}^* = (f - Gp^k), \text{ on } \Omega_F. \tag{3.10}$$

In fact, it is desired that the momentum equation is exactly satisfied, i.e.

$$D\vec{u}^{k+1} = (f - Gp^{k+1}) \text{ on } \Omega_F. \tag{3.11}$$

Subtracting Equation (3.10) from Equation (3.11), we have

$$\delta\vec{u} = D^{-1}G\delta p \text{ on } \Omega_F, \tag{3.12}$$

where $\delta\vec{u} = \vec{u}^{k+1} - \vec{u}^*, \delta p = p^{k+1} - p^k$. By taking the divergence of Equation (3.12) and employing the continuity equation for $\vec{u}^{k+1}$, we end up with a Poisson-type equation for the pressure correction

$$Div\lambda G\delta p \quad = \quad -Div\vec{u}^* \tag{3.13}$$

In the original SIMPLE algorithm, $\lambda = (diag\{D\})^{-1}$. After the Poisson equation for the pressure correction is solved, the pressure and the velocity components are updated:

$$p^{k+1} = p^k + \delta p \qquad \text{on } \Omega_F \tag{3.14}$$

$$\vec{u}^{k+1} = \vec{u}^{*} + \lambda G \delta p \qquad \text{on } \Omega_F. \tag{3.15}$$

Each iteration of the SIMPLE-type decoupling method includes the following steps

- Compute velocity predictions, u^*, v^* from Equation (3.10);
- Compute δp from Equation (3.13);
- Update pressure and velocities according to Equations (3.14) and (3.15).

Note that iterating over velocity and pressure is typical for such decoupling methods (cf. [38]).

3.2.3 The Darcy problem

The discrete pressure on the porous domain is assigned to the cell centers and the components of the Darcy velocity are assigned to the center of the faces, as it was done in Subsection 3.2.2.
The Darcy equation (3.1) is discretized as follows:

$$\vec{u} = -KGp \text{ on } \bar{\omega}_P^u \times \bar{\omega}_P^v. \tag{3.16}$$

The discretized continuity equation looks as follows:

$$G\vec{u} = f \text{ on } \omega_P^p. \tag{3.17}$$

Substituting the discretised Darcy velocities in Equation (3.16) into the discretised continuity Equation (3.17), we arrive at the standard five point discretisation of the pressure equation on cell centered grid in the porous region:

$$DivKGp = f, \text{ on } \bar{\omega}_P^p. \tag{3.18}$$

Note and recall that the boundary conditions are incorporated in the discrete operators. Usually the Dirichlet boundary conditions (i.e., prescribed pressure), lead to a first order discretization for the velocity on the Dirichlet part of the boundary. For a second order discretization for the velocity, see [50].

3.2.4 An iMSFV method for the Darcy problem

Grids and grid notations

The MSFV (and the iMSFV) method employs two sets of coarse grids, namely a primal and a dual coarse grids, in addition to the underlying fine grid [59], as illustrated in Figure 3.3. As discussed above, a uniform staggered fine grid of size h is used for Ω. For this section, let us assume that the whole domain Ω is porous. The local auxilliary problems are solved separately on each block of the dual coarse grid, where the governing equations are discretized on an underlying fine grid. The solutions of the auxilliary

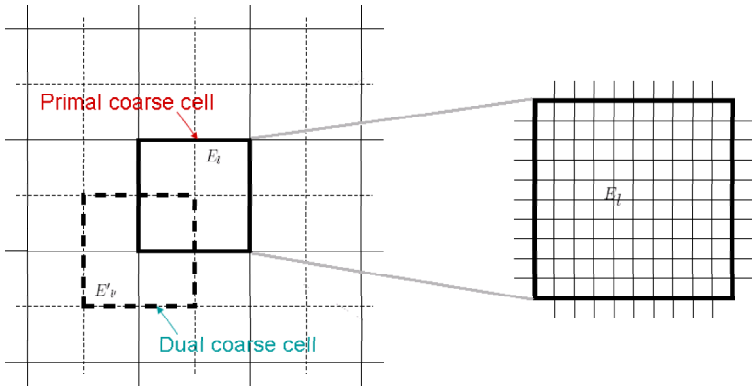


Figure 3.3: Part of the computational domain Ω with coarse(solid lines) and primal grid(dotted lines); bold lines are one coarse block E_l and one dual block $E'_{l'}$

problems are used within the framework of the MSFV method to build a coarse grid discrete sytsem on the primal coarse grid.

Let us consider a set of non-overlapping primal coarse blocks $E = \{E_l, l = 1,\ldots,N_B\}$, which spans over Ω, where N_B denotes the number of the primal (basic) coarse blocks. We have

$$\Omega = \cup_{l=1,N_B} E_l.$$

E_l denotes a primal coarse block. $\mathscr{E}_l$ and $\mathscr{V}_l$ denote all edges and vertices of E_l respectively. The centre node is denoted by x_l . The dual grid is constructed by connecting the centres of primal coarse blocks, resulting in a grid that is staggered to the primal coarse grid. Analogously, the union of the dual coarse blocks $E'_{l'}$ cover the full domain Ω

$$\Omega = \cup_{l'=1,N'_D} E'_{l'}$$

where N'_D is the total number of dual coarse blocks. Likewise, $\mathscr{E}'_{l'}$, $\mathscr{V}'_{l'}$ and $x_{l'}$ denote the edges, vertices and center of $E'_{l'}$.

A MSFV method for the scalar elliptic problem

Here, we shortly describe the MSFV method, earlier developed for the flow in porous media (cf. [59, 71]). Let us consider a scalar elliptic problem in Ω:

$$\nabla \cdot \vec{u} = f \qquad \text{on } \Omega \tag{3.19}$$

$$\vec{u} = -\lambda \nabla p \qquad \text{on } \Omega \tag{3.20}$$

$$\vec{u} \cdot \mathbf{n} = g_1 \qquad \text{on } \partial\Omega_N \tag{3.21}$$

$$\vec{u} = g_2 \qquad \text{on } \partial\Omega_D \tag{3.22}$$

where $\lambda = K$ varies on the fine grid and thus resolves the fine scale heterogeneities. $\partial\Omega_N$ is the Neumann boundary, where as $\partial\Omega_D$ is the Dirichlet boundary of $\partial\Omega$, where

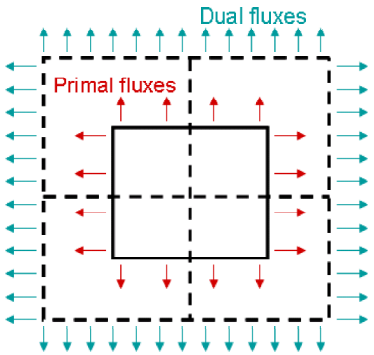


Figure 3.4: The figure shows one primal coarse block (solid line) with its four dual blocks(dashed line). The fine scale fluxes are approximated on the dual block boundaries via the basis function. The superposition of the basis functions, further restricted on the primal coarse block gives an approximation of the fluxes across the coarse block boundaries. The fine scale information is incorporated in the fluxes.

$\partial\Omega = \partial\Omega_N \cup \partial\Omega_D$. The goal is to obtain a solution without having to solve a global fine scale problem. Moreover, it should account for the fine scale heterogeneities of the coefficient K. The basic idea is to decompose the global problem into several local problems. For understanding the algorithm, it is worthwhile to note that the interior of each primal coarse cell includes exactly one corner node belonging to four dual cells in 2D. Additionally, the strict interior and the boundary of a dual block, E'_l, are denoted by $\mathring{E}'$ and $\partial E'$ respectively. The set $\ddot{E}'_l$ consists of the four dual blocks that overlap with the primal block E_l.

The idea of the MSFV method is to capture the fine scale/grid influences on the solution via basis and correction functions. This information is further brought to the coarse scale through the global stifness matrix and the right hand side. The fine scale information is brought to the coarse scale via the computation of the basis functions in the form of fluxes across the boundaries, as illustrated in Figure 3.4.

Four auxilliary basis functions, $\phi_l^{l'}$, are assigned to each center x_l of the primal coarse cell. Here $x_{l'} \in \mathscr{V}_l$ are the centers of the four dual coarse blocks which have x_l as a common vertex. The basis functions are the solutions of the following local problems:

$$\nabla \cdot (\lambda \cdot \nabla \phi_l^{l'}) = 0 \qquad \text{in } \mathring{E}'_{l'} \tag{3.23}$$

$$(\nabla \cdot \mathbf{t})\lambda(\nabla \cdot \mathbf{t})\phi_l^{l'} = 0 \qquad \text{on } \mathscr{E}'_{l'} \backslash (\mathscr{V}'_{l'} \cup \partial\Omega) \tag{3.24}$$

$$\phi_l^{l'}(x_l) = 1 \qquad \text{for } x_l \tag{3.25}$$

$$\phi_l^{l'}(x_k) = 0 \qquad \text{for } x_k \in \mathscr{V}'_{l'} \setminus \{x_l\}, \tag{3.26}$$

$$\lambda \phi_l^{l'} \cdot \mathbf{n} = 0 \qquad \text{on } \partial\Omega_N \tag{3.27}$$

$$\phi_l^{l'} = 0 \qquad \text{on } \partial\Omega_D \tag{3.28}$$

The sum of these four auxilliary basis function is the basis function ϕ_l, after restricting it on the primal coarse cell is

$$\phi_l = \sum_{l' \in \ddot{E}'_l} \phi_l^{l'}.$$

In a similar manner, the four correction functions are computed to incorporate the source term and the boundary conditions.

$$\nabla \cdot (\lambda \cdot \nabla \psi_l^{l'}) = f \qquad \text{in } \mathring{E}'_{l'} \cap \bar{\omega}^p \tag{3.29}$$

$$(\nabla \cdot \mathbf{t}) \lambda (\nabla \cdot \mathbf{t}) \psi_l^{l'} = f \qquad \text{on } \mathscr{E}'_{l'} \backslash (\mathscr{V}'_{l'} \cup \partial\Omega) \tag{3.30}$$

$$\psi_l^{l'} = 0 \qquad \text{on } \mathscr{V}'_{l'}, \tag{3.31}$$

$$\lambda \psi_l^{l'} \cdot \mathbf{n} = g_1 \qquad \text{on } \partial\Omega_N, \tag{3.32}$$

$$\psi_l^{l'} = g_2 \qquad \text{on } \partial\Omega_D. \tag{3.33}$$

The sum of the four correction functions is assigned to the primal coarse cell l: $\psi_l = \sum_{l' \in \ddot{E}'_l} \psi_l^{l'}$. The global fine scale solution p is approximated by p', which is a function of coarse pressure values at the centers of the primal coarse blocks, $\bar{p}_l$

$$p' = \sum_{l=1}^{N_B} (\bar{p}_l \phi_l + \mathscr{R} \psi_l), \tag{3.34}$$

where $\mathscr{R}_{\uparrow}$ denotes the operator that restricts ψ_l to its corresponding primal coarse cell l. A linear system is built to find the coarse pressure values. After substituting equation (3.34) into (3.20), we get

$$-\int_{E_l} \nabla \cdot (\lambda \cdot \nabla p') dV = \int_{E_l} \nabla \cdot (\lambda . \nabla \left(\sum_{l=1}^{N_B} (\bar{p}_l \phi_l + \psi_l) \right) dV. \tag{3.35}$$

Applying the Gauss theorem for each primal coarse block, the following linear system is derived

$$A \bar{p}_l = b_l, \quad l = 1, 2, ..., N_B \tag{3.36}$$

where $A = \{a_{kl}\}, k = 1, \ldots, N_B, l = 1, \ldots, N_B$ is the global stiffness matrix, corresponding to the pressure values in the coarse primal blocks, and having at most nine nonzero entries per row. The entries of the matrix are defined as follows:

$$a_{kl} = \int_{\mathscr{E}_l} (-\lambda \cdot \nabla \phi_k) \cdot \mathbf{n} \, dS. \tag{3.37}$$

Obviously, $a_{kl} \neq 0$ only if $\mathscr{E}_l \cap \text{support}\{\psi_k\} \neq 0$, i.e., for fixed l, $x_k \in \mathscr{V}'_{l'}$, $x_{l'} \in \mathscr{V}_l$. The right hand side is defined in a similar way:

$$b_l = \sum_{k=1}^{N_B} \int_{\mathscr{E}_l} (-\lambda \nabla \psi_k) \cdot \mathbf{n} \, dS + \int_{E_l} f \, dV \tag{3.38}$$

Further details for the MSFV method, particularly for elliptic problems, can be found in [59, 71].

iMSFV method for the scalar elliptic problem

In the above described MSFV algorithm, the approximate solution $p^{'}$ satisfies the same equation as the exact discrete solution p in the interior of the coarse cells, i.e. on $\mathring{E}' \cap \tilde{\omega}^p$. However, on the interfaces between dual coarse cells, where the localization conditions are impossed, $p^{'}$ satisfies $(\nabla \cdot \mathbf{t})\lambda(\nabla \cdot \mathbf{t})p' = f$, while p satisfies $\nabla \cdot (\lambda \nabla p) = f$. The discrepancy occurs due to the ommited normal derivatives in the localization of problems for $\phi_l^{l'}$ and $\psi_l^{l'}$. In order to improve the accuracy of the computed approximate solution p', an iterative version of the MSFV method, namely the iMSFV was proposed (cf. [45]). For the iMSFV method, the basis functions are computed in the same way as above, however the correction functions are updated at every ($n-$th) iteration of the iMSFV method. This implies that after every $n-$th iteration, the localization condition in (3.30) is replaced with another localisation condition, which contains information about the current fine grid iterate of the solution:

$$(\nabla \cdot \mathbf{t})\lambda(\nabla \cdot \mathbf{t})\psi_l^{l'} = f - (\nabla \cdot \mathbf{n})\lambda(\nabla \cdot \mathbf{n})p'^n. \tag{3.39}$$

Essentially at every $n-$th iMSFV iteration, the correction functions are recomputed using the localization condition in Equation (3.39), instead of (3.30).

3.3 An iMSFV method for the Stokes-Darcy problem

In this section, we discuss an iterative procedure for the coupled Stokes-Darcy problem for the case when the coefficients and/or the geometrical features vary on the fine scale.

3.3.1 Velocity-pressure decoupling algorithm

As mentioned above, Stokes system is usually solved via some iterative velocity-pressure decoupling procedure [41, 84]. Although the flow in the porous region is governed by a scalar linear elliptic equation, it has to be solved at each outer iteration due to the coupling conditions on the interface with the free fluid region. The starting point of the algorithm is the SIMPLE-based decoupling algorithm for the Stokes system and a special form of the Darcy problem in the porous medium. For convenience, let us recall the decoupling procedure for the Stokes equations from Subsection 3.2.2.

Let us assume that the pressure p^k is known after the $k-$th iteration. The velocity in the fluid can then be predicted by using p^k. This in only possible if the velocity on the interface between plain and porous media is also taken from the $k-th$ iteration:

$$D\vec{u}^* = (f - Gp^k), \text{ on } \Omega_F. \tag{3.40}$$

In fact, we want the momentum equation to be exactly satisfied:

$$D\vec{u}^{k+1} = (f - Gp^{k+1}) \text{ on } \Omega_F. \tag{3.41}$$

Subtracting equation (3.40) from equation (3.41), we have the velocity correction

$$\delta\vec{u} = \lambda G\delta p \text{ on } \Omega_F, \tag{3.42}$$

where $\lambda = D^{-1}$, $\delta\vec{u} = \vec{u}^{k+1} - \vec{u}^*$ and $\delta p = p^{k+1} - p^k$.

Similarly, we write the equations in the porous region. Considering the Darcy equation, we have

$$\vec{u}^* = -\lambda G p^k \text{ on } \Omega_P, \tag{3.43}$$

$$\vec{u}^{k+1} = -\lambda G p^{k+1} \text{ on } \Omega_P \tag{3.44}$$

where $\lambda = K$. By subtractiing equation (3.44) from equation (3.43), we get $\delta\vec{u} = \lambda G\delta p$ on Ω_P. Recall that the the problem is divergence free in both, the plain and the porous media, domains. This means that $Div\vec{u}^{k+1} \equiv 0$. After applying the divergence operator to the momentum equations in the fluid, as well as in the porous regions, we get the following pressure correction equation on Ω:

$$Div\lambda G\delta p = Div\vec{u}^*, \quad x \in \omega^p \tag{3.45}$$

At each iteration of our decoupling algorithm, we have to solve the momentum equations (3.40) in the fluid region, a trivial form of momentum equations (3.43) in the porous region, and a global pressure correction equation (3.45) in the whole domain. The pressure correction equation (3.45) is solved by the direct application of the iMSFV, as described above. In order to apply iMSFV to Equation (3.40), the computations for the basis functions is modified. They have to be recomputed at each decoupling iteration due to the constantly changing velocity on the plain-porous interface.

For the Stokes-Darcy problem, $\lambda = D^{-1}$ in Ω_F and $\lambda = K = (k_{11}, k_{22})^T$ in Ω_P.

3.3.2 Computation of the basis and correction functions

The algorithm discussed in Subsection 3.3.1 employs the iMSFV method to compute the intermediate velocities $\vec{u}^*$ and the pressure correction δp for each time step. The computation of the basis and correction functions for δp follows from Subsection 3.2.4. However, the computation of basis and correction function differs for the velocities.

For the Stokes-Darcy problem, the coarse blocks E_l and $E'_{l'}$ can be characterized in three ways. The dual blocks can be purely fluid or purely porous or a mixture of fluid and porous fine grid cells. In Section 3.2.4, the computation of the basis and correction function for dual blocks that are either fully in the fluid or porous region was decribed.

For the dual blocks containing mixed porous and fluid regions, the computation of the basis and correction function differs such that equations (3.23) and (3.29) are replaced with

$$\nabla \cdot (\mu \cdot \nabla \phi_l^{l'}) = 0 \qquad \text{in } \mathring{E}'_{l'} \cap \Omega_F \tag{3.46}$$

$$\nabla \cdot (\mu \cdot \nabla \psi_l^{l'}) = f \qquad \text{in } \mathring{E}'_{l'} \cap \Omega_F \tag{3.47}$$

and the additional condition is posed

$$\phi_l^{l'} = 0 \qquad \text{in } \mathring{E}'_{l'} \cap \Omega_P \tag{3.48}$$

$$\psi_l^{l'} = \vec{u}^k \qquad \text{in } \mathring{E}'_{l'} \cap \Omega_P \tag{3.49}$$

Figure 3.5 illustrates one of the four basis/correction function computed for a dual block with fluid and porous regions.

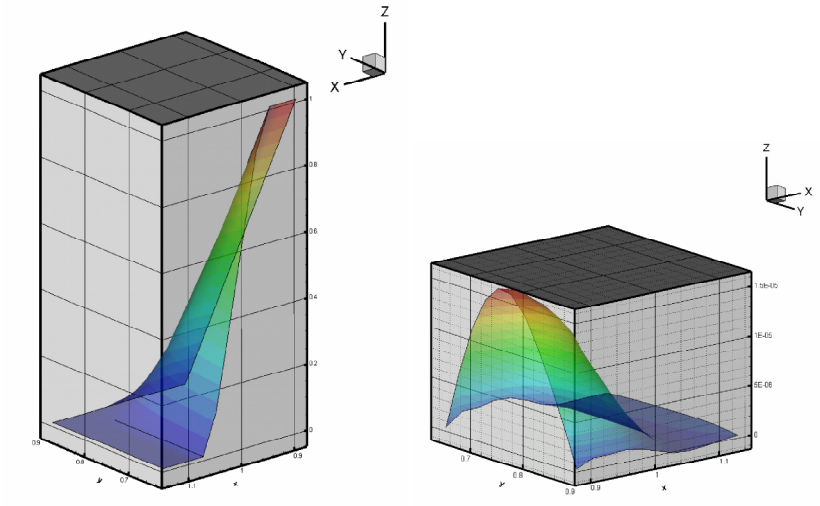


Figure 3.5: Basis and Correction function computed on a dual cell consisting of fluid and porous regions.

3.4 Numerical results and Validation

A 2D rectangular domain $\Omega = \Omega_P \cup \Omega_F$ is considered. The Stokes-Darcy equations are discretized on an underlying fine grid of size 200×200 cells with a coarse grid of 8×8 blocks, corresponding to an upscaling factor of 25×25. The results from the iMSFV method for the Stokes Darcy problem are compared with the fine scale reference solutions.

Boundary Conditions

The flow domains considered in the current work are of different geometric characteristics. The governing equations are solved subject to the following boundary conditions which are mainly employed in filtration problems. At the inlet of the free flow region, a velocity profile is specified. On the impermeable boundaries of the free flow region, no slip wall velocity boundary conditions is imposed. Finally at the outlet of the composite flow domain, zero pressure has been imposed as an exit condition for the flow domain.

3.4.1 Channel Filter

As a first example, we consider a simple geometry of a filter. The filter is parallelepiped $\{(x,y) : -1 \leq x \leq 1, -1 \leq y \leq 1$ with the single porous layer $\{(x,y) : -0.95 \leq x \leq 0.05, -1 \leq y \leq 1$. The following problem parameters are specified: $U_{in} = 1$, $\mu = 0.01$, and isotropic permeability $K = 1.0 \cdot 10^{-4}$. We compare values of the pressure drop between the inlet ($x = -1$) and the outlet ($x = 1$) of the filter. Note that the considered flow is essentially one dimensional. Results obtained from the developed algorithm are compared with the fine scale reference solution, and both result in a pressure drop of 98.863. Figure 3.6 and 3.7 show the pressure and velocity profiles.

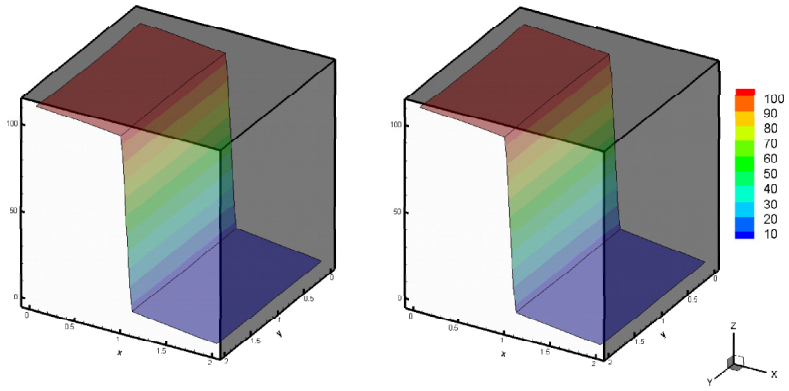


Figure 3.6: Figure shows solution for pressure for Channel Filter. Left: Fine scale reference solution; Right: Computed via Stokes-Darcy iMSFV Method

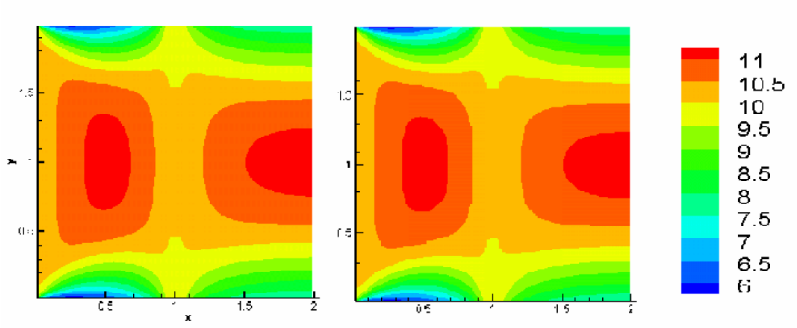
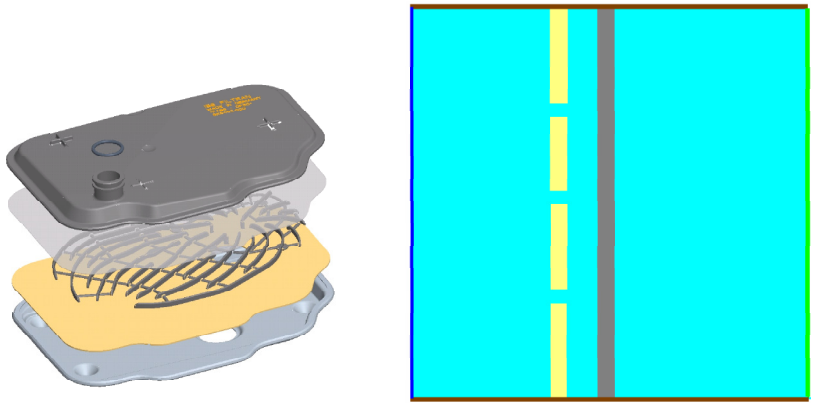


Figure 3.7: Figure shows solution for Velocity for Channel Filter. Left: Fine scale reference solution; Right: Computed via Stokes-Darcy iMSFV Method

3.4.2 Combi Filter

The next example is chosen to be a filter of simple channel geometry as shown in Figure 3.8 (a) but with the more realistic (industrial) combination of filtering layers, illustrated in Figure 3.8 (b). When the filter element is designed in a way that there is

enough space between the inlet(bottom) and the porous layer, and between the porous layer and the outlet(top), the pressure drop is caused practically from the porous layers only. Therefore, we study this filter in a simplified geometry. The filter is parallelepiped $\{(x,y) : 0 \leq x \leq 2, 0 \leq y \leq 2$ with two filtering porous layers, as shown in Figure 3.8 (b). Additionally, the first porous layer ($0.7 \leq x \leq 0.8$, $0 \leq 2$) has 3 holes punctured within it, as shown in Figure 3.8 (b), where each hole is a cube: 1×1 fine grid cell. In Figure 3.9 and 3.10, the pressure and velocity profiles computed from the numerical



(a) (b)

Figure 3.8: (a) A real industrial filter with multiple porous media layers. (b) Channel filter with the multiple porous layers. A simplified channel filter of the industrial filter to understand real processes.

algorithm are presented. The computed results are compared with the fine scale reference solutions, and it can be seen that they both compare really well. Both solutions result in the same pressure drop of 119.335. Moreover, it is worthwhile to note that the reconstructed fine scale velocities through the holes of the porous layer match closely to the fine scale solution too.

3.5 Conclusions

The iMSFV method is extended to solve the coupled Stokes-Darcy problem employing the SIMPLE method for decoupling velocity and pressure. The iMSFV method is used to solve the momentum equations and the pressure correction equation in the SIMPLE algorithm. The basis and correction functions used for the momentum equations are modifed as discussed. The algorithm was validated by using two geometries, a channel filter and a combi channel filter, where the results were match against fine scale solutions. In both examples considered, the pressure and velocity solutions are in excellent agreement with the fine scale reference simulations.

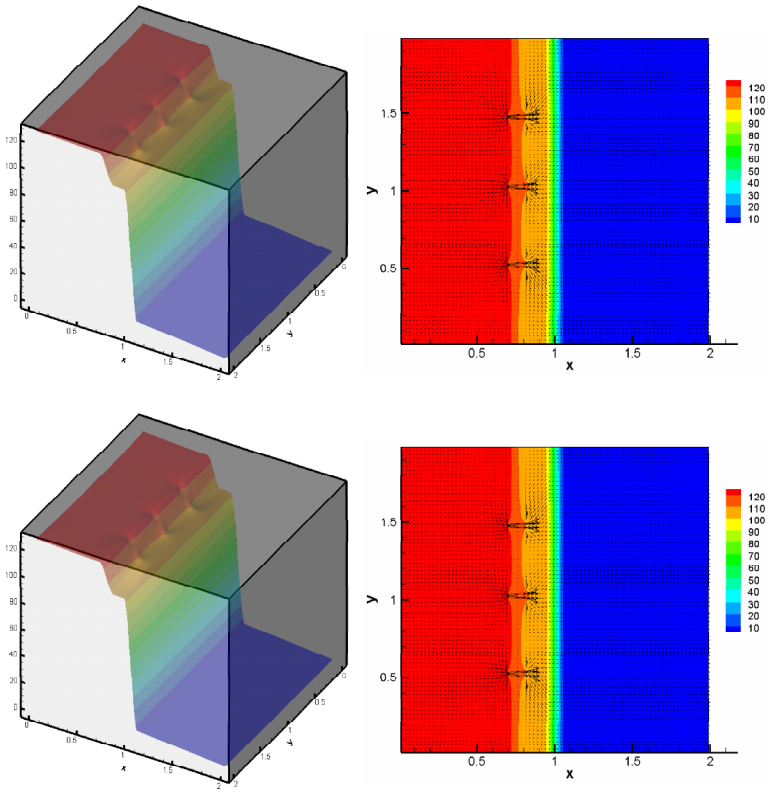


Figure 3.9: Figure shows solution for Pressure for Combi Channel Filter. Top: Fine scale reference solution; Bottom: Computed via Stokes-Darcy iMSFV Method

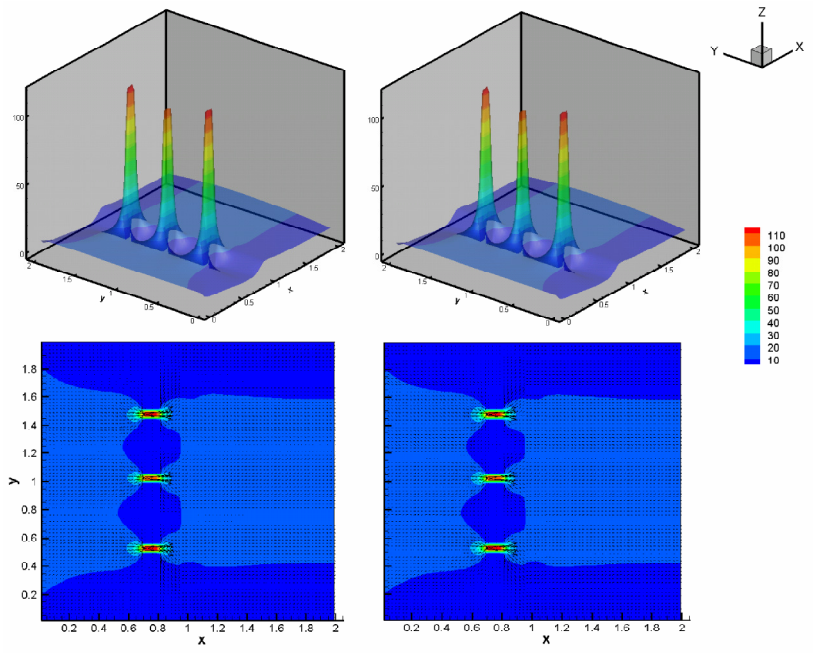


Figure 3.10: Figure shows solution for Velocity for Channel Filter. Left: Fine scale reference solution; Right: Computed via Stokes-Darcy iMSFV Method

Chapter 4

On numerical approaches for filtration efficiency simulation

4.1 Introduction

This chapter concerns the modeling and computer simulation of filtrating solid particles from the fluid. As it is known, the main criteria for determining the performance of a filter element could be listed as the following:

- pressure drop - flow rate ratio;
- dirt storage capacity;
- the size of the biggest penetrating particle;
- increasing blockage effect of captured particles.

The first criterion is closely related to the energy efficiency of the filter element (e.g. the choice of the pump to be used), while the second and the third criteria are more related to the so called *filtration efficiency*, or *filter efficiency*, i.e. the speed with which the particles of a given size are filtered out of the fluid, or the frequency of replacing the filter element (i.e. the lifetime of a filter element). In the previous chapters, the simulation approaches assisting the engineers in the design of filter elements with optimized pressure drop - flow rate ratio was discussed. The current chapter concerns the filter efficiency modelling and simulation.

The filtration efficiency of the manufactured filter elements is evaluated on the basis of certain international standards (see Section 4.2 for details). For this purpose each manufacturer has a certified Lab, where tests with the sample filter elements are performed.

These measurements are expensive, and the industry is constantly on the lookout for simulation tools or methods which could partly replace them. A mathematical model that could be used for computer simulations of filter efficiency is discussed in Section 4.3. In an ideal scenario, a virtual filter element design is desired, where the computer

simulation would assist in atleast recognizing those filter element designs which do not meet the requirements, as specified by the user. This would save the time, efforts and costs of a designing and manufacturing a prototype. Moreover, no Lab tests for efficiency would be required. Within the scope of this chapter, we discuss how modeling and simulation can help to reduce the number of the Lab measurements, or in a broader sense, assist the virtual filter element design.

The selection of the correct filtering medium is essential for filtration efficiency. Other components like the shape/design of filter housing, shape and size of the pleats etc. also severly impact filter efficiency. Performing measurements on test filters with a given flat filering medium can help to evaluate the efficiency of the medium in ideal conditions, but this is not enough to evaluate the performance of real filters with complicated geometries. An approach on how to use measurements on the flat media in order to reduce the number of efficiency tests with real filters, is presented in Section 4.4. The idea is to perform measurements on the flat filtering medium for different flow rates and particle concentrations, and to use the results for creating lookup tables. The tables basically correlate the flow velocity and the upstream concentration of particles to the capturing (deposition) rate of the filtering medium. Next, these tables are used in computer simulations performed for complicated geometries, taking advantage from the fact that numerical simulations provide information for the velocity and the concentrations at any desired location within the filter element.

The filtration process is an intrinsic multiscale process, where the phenomena occuring at microscale (particles and pores) are coupled to the phenomena occuring at macroscale (flow and pressure within the filter element). For certain contaminants, the impact of the filtering medium within the filter housing, along with the coupled micro-meso-macro scale simulations are needed to get a better understanding of the filtration processes. This in turn directly assists in the design of efficient filters. Two multiscale approaches are considered in Section 4.5. One of them suggests an algorithm for coupling macro scale and microscale simulation whereas the second one is a part of the concept for the virtual filter element design. In the latter, microscale simulations (at particles and pores/fibres level) are performed in order to determine the particles capturing rate (needed in the look up tables). This approach provides an option to evaluate the filter efficiency without manufacturing the filter medium and the filtering element.

In the last Section 4.6, results from the numerical simulations for validating some test case are presented. Moreover, examples of real transmission oil filters are considered and validated against measured data.

4.2 International Standards tests for evaluation of filter efficiency

Multipass test: One of the very informative tests is the so called Multipass test, ISO 16899, [89]. As illustrated in Figure 4.1, a filter element is connected to a reservoir, and the liquid from the reservoir is circulated through the filter element with a constant system flow rate. Initially the liquid in the reservoir is clean, but starting at time $t = 0$, it is continuosly contaminated with a certain amount of specified certified dust. The contaminant from the injection system is injected into the reservoir at a specified injection flow rate Q_{dirt}. Circulation of the fluid in the filter test system is done using the system flow rate Q. Usually $Q >> Q_{dirt}$. At the downstream end of the filter element, liquid is removed with flow rate Q_{dirt}, ensuring that the volume in the filter test system is always constant. The measurements are performed until the filter element is (almost completely) clogged, which is monitored by the increasing pressure. This test is widely used in air, water and oil filtration.

In the case of transmission oil filtration (that is our primary ineterst), another test is considered to be more informative for this specific application, namely, *Transmission Filter Effectiveness Method (TFEM).*

TFEM: This is a relatively new approach used for evaluating the filter with respect to fluid cleanliness [33]. This procedure is more appropriate for the low contaminated transmission oil bacause it focuses on the fluid cleanliness as opposed to evaluating the clogging time (as it was done in the Multipass test).

The experimental setup for TFEM comprises of the filter test system but *without* the injection system, as illustrated in Figure 4.1. The filter element in the test system can be connected either to the pressure or to the suction side of a pump. A circulation system is used to provide mixing and pressure to the particle counters. To begin with, the test fluid is cleaned to an appropriate background level with a cleanup filter. When particle counts are sufficiently low, the fluid in the reservoir is contaminated with a specified test dust (e.g. Arizona, ISO medium test dust) to a given concentration C_T^0 and thoroughly mixed during the fluid stabilization period. The test filter is installed and flow is established with a flowrate Q. Particle counter record and document the for a prescribed period of time.

There are different ways in which the particle counts are documented. Following, we present two most commonly used ways in which the particle counts are documented. Variant 1 is mosre useful in the simulation of Multipass tests, whereas Variant 2 is more suitable for the TFEM simulations.

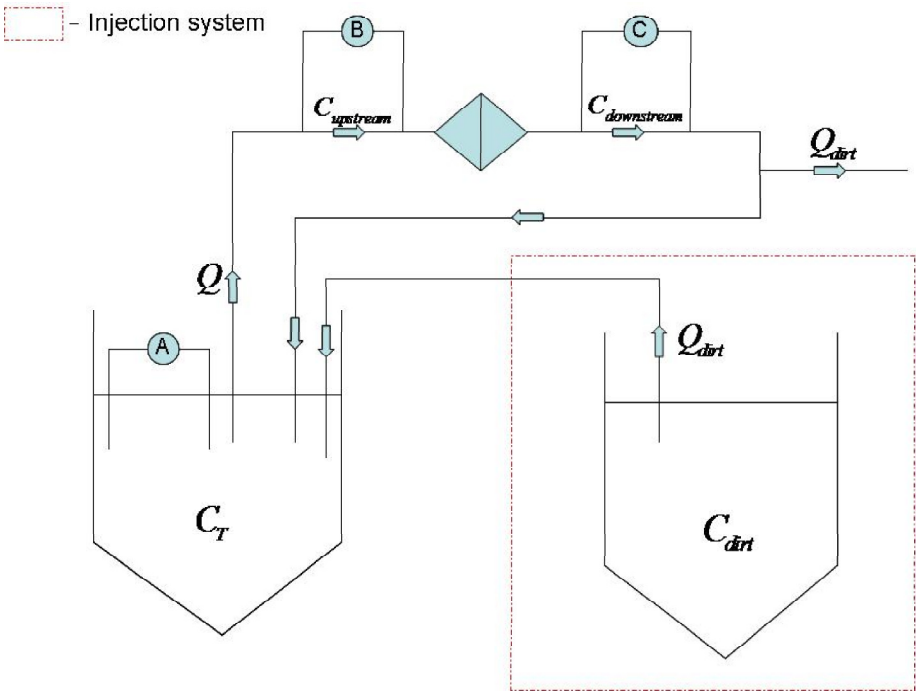


Figure 4.1: Experimental setup for the Transmission Filter Effectivness Method

Variant 1 - particle counters for upstream and downstream concentration

The first variant is when the particle counts within the filter test system are measured in front and behind the filtering medium, i.e. particle counters 'B' and 'C' are used. Counts are graphed after specified time intervals and a trend is evident where the fluid cleanliness begins to improve. It will be seen in the following subsections how this information is interpreted in determing the deposition rate α.

Variant 2 - particle counter inside the tank

The particle concentration ($number/ml$) is a measure of particle counts inside the tank, denoted by C_T. This implies that only the counts obtained from particle counter 'A', as shown in Figure 4.1, is used. Counts are typically graphed every one/ten minutes and a trend is evident where the fluid cleanliness begins to improve. The cleanliness curves for particular particle sizes are derived from these experiments as shown in Figure 4.2. The cleaning curves serve as an indicator for the cleanliness level reached by the filtering medium.

4.3 Model Equations

The Navier-Stokes-Brinkmann system of equations, (4.1) and (4.2), for the fluid transport in plain and porous media, together with the Convection Diffusion-Reaction equa-

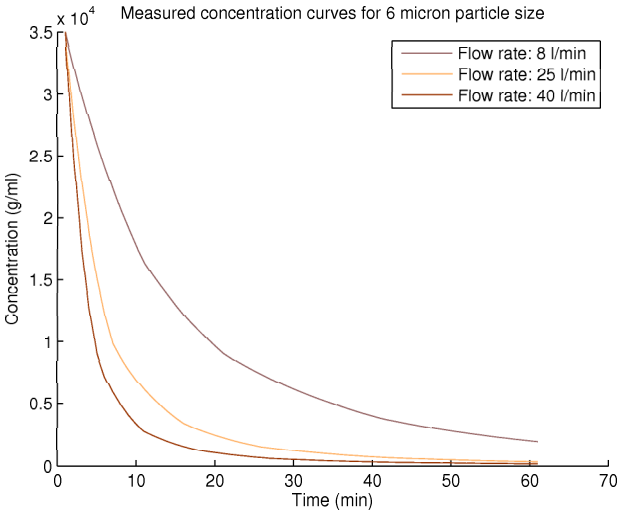


Figure 4.2: Cleanliness curves for different flow rates using Variant 2 for documenting the particle concentrations. *Courtesy IBS Filtran*

tion (4.3) for particle transport are put together into a single system modeling fluid and particle transport in Ω

$$\rho \frac{\partial \vec{u}}{\partial t} - \nabla \cdot (\tilde{\mu} \nabla \vec{u}) + (\rho \vec{u}, \nabla)\vec{u} + \tilde{\mu} \tilde{\mathbf{K}}^{-1} \vec{u} = \vec{f} - \nabla p \tag{4.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{4.2}$$

$$\frac{\partial C^J}{\partial t} + (\vec{u}, \nabla C^J) - D^J \Delta C^J = \frac{\partial M^J}{\partial t}, \qquad J = 1, \cdots, 8 \tag{4.3}$$

The notations for the Navier-Stokes-Brinkmann system are consistent with Chapter 1. Additionally, in Equation (4.3), C stands for the concentration of particles. $C(x,y,z,t)$ is the unknown function in a spatial domain Ω over a time interval $t \in (0,T]$. The initial condition is a given function $C^0(x,y,z)$, i.e. $C(x,y,z,0) = C^0(x,y,z)$. Further on, $\vec{u}$ is the velocity, D is the diffusivity coefficient, and $\frac{\partial M^J}{\partial t}$ the rate of deposition. M^J stands for the captured particles within the filtering medium. Particles are cumalitively grouped into 8 particle sizes, where $J = 1, \ldots, 8$. It is also worthwhile to note that $\frac{\partial M^J}{\partial t} = 0$ for $\Omega \backslash \Omega_P$.

Boundary Conditions

The equations are equipped with the following boundary conditions

- inflow velocity and concentration, $\vec{u}_{in}, C^J_{in}$ at the inlet is prescribed. Also $\frac{\partial p}{\partial n} = 0$;
- outflow b.c. at the outlet: $p = p_{out}$, $\frac{\partial C^J}{\partial \mathbf{n}} = 0$;
- $\vec{u} = 0$, $\frac{\partial C^J}{\partial \mathbf{n}} = 0$ elsewhere.

Assumptions

Due to the complex nature of the dynamical problem, our model and simulations are based on certain assumptions.

1. The system flow rate is constant throughout time (same as the prescribed initial flow rate).
2. In Equation (4.3), convection dominates and the diffusion term is negligible.
3. All particles are considered to be spherical and have the same density.
4. No agglomeration or breaking of particles is considered. This implies that the 8 concentration equations are independent of each other.
5. Gravitational and sedimentation effects are ignored in the model.

Taking into account assumption (1) and (2), that the particles are driven with the flow velocities that are computed by employing the numerical algorithm developed in Chapter 1, we decouple the Navier-Stokes-Brinkmann system and the convection diffusion reaction equations and treat them consecutively, except for cases when the deposited particles change the permeability and recomputing the flow is needed. Assumption (3) and (4) allow us to treat different sizes of contaminant particles in parallel. Currently we are working with eight sizes, what is consistent with the measurement data, which are provided from the Lab measurements.

Summarizing the role of Equation (4.3), it models the mass balance. In this chapter, we pay special attention towards the term describing the deposition of particles, i.e. $\frac{\partial M^J}{\partial t}$. This term, by far, is the most important for filtration processes.

Under the assumption that the amount of the deposited particles is small compared to the pore space of the filtering medium, the amount of the deposited particles is considered to be proportional to the concentration of particles [19]:

$$\frac{\partial M^J}{\partial t} = \alpha^J C^J, \tag{4.4}$$

where α^J (constant of propotionality) is the particles' deposition rate. Combining the two equations, we get:

$$\frac{\partial C^J}{\partial t} + (\vec{u}, \nabla C^J) - D^J \Delta C^J = -\alpha^J C^J \tag{4.5}$$

Such a model fits very well to (almost all) particle sizes and regimes of the TFEM (cf. [33]). TFEM was discussed already in Section 4.2, where the particle deposition compared to the pore space is minimal. This is mostly true for the initial stages of filtration.

Another model used to account for partially loaded filters, or possible clogging, reads as follows [19]:

$$\frac{\partial M^J}{\partial t} = \alpha^J \left(1 + \frac{M}{M_0}\right) C^J, \tag{4.6}$$

In this case, the deposition rate increases with increased deposited mass. This is a typical characteristic of the highly efficienct filtering media. The simulations using this model are not completed and therefore they are not discussed in this thesis. They will be reported elsewhere.

It should be noted that the complete clogging is rear in transmission oil filtration (unlike air filtration). However, in transmission oil filtration, there is another phenomena, known as the *saturation of the efficiency*, which requires for the modification of the model. One speaks about this phenoment in the contect of TFEM, when a certain percentage of the contaminant in the reservoir is cleaned by the filter element to the extent that further circulation of the oil through the filter element does not lead to further cleanliness of the oil, i.e. the process of filtration stagnates. To account for this, the following model is suggested:

$$\frac{\partial M^J}{\partial t} = \alpha^J \left(1 - \frac{M}{M_0}\right) C^J, \tag{4.7}$$

Obviously, at initial stages of filtration with very low loadings of the filter, the parameter M_0 dominates, and the model reduces to Equation (4.4). Moreover, at time $t = 0$, deposited mass $M(0) = 0$. M_0 has some prescribed value M_{max}, where M_{max} is the maximum mass that the filter medium can retain. As time elapses, M approaches M_0 which reduces the right hand side of equation (4.7) to zero. This implies that no more capturing would occur after this time.

For the rest of the discussion in the following subsections, we will drop the superscript ^{J} for the simplicity of notations.

4.3.1 Analytical solution in the case of constant α

This case is the most studied in the literature. Lets consider regimes where diffusion is negligible, and the time variation of the concentration is solely governed by the deposition rate. This case leads to a simplification of equation (4.3) (cf. [19]):

$$u^x \frac{\partial C}{\partial x} = -\frac{\partial M}{\partial t} \tag{4.8}$$

This equation, together with Equation (4.4) allow for analytical solution which is given by:

$$C(x) = C_0 e^{-\frac{\alpha}{u^x} x} \tag{4.9}$$

$$M(x,t) = \alpha t C_0 e^{-\frac{\alpha}{u^x} x} \tag{4.10}$$

Here C_0 is a constant contaminant concentration in front of the filter, and it is assumed that the filter is initially clean, i.e. $M(0) = 0$. Moreover, it is assumed that the flow across the filter is in x direction dominates, and therefore we consider the one dimensional form in equation (4.8).

The model and its analytical solutions include parameter α, which might be an unknown. In the following sections, we will try to discuss parameter identification methods to determine its value.

4.4 Parameter identification methods for α

In this section, several ways for identifying the deposition rate α is discussed. One of the ideas for its identification is via Lab measurements for a simple test filter with flat surface of the filtering medium for a range of velocities for u^x. In the following subsections 4.4.1 and 4.4.2, two different approaches are discussed for each of the documented measurements discussed in 4.2. Alternatively, another idea is proposed in the next section where the expensive Lab measurements are replaced by microscale simulations. Thereafter, the aim is to use the identified parameter $\alpha(u)$ in simulations for arbitrary filter elements.

4.4.1 α computed from measurements in Section 4.2 - Variant 1

This case will be studied under the assumption that the inflow concentration changes relatively slowly, and the steady state versions of the concentration equation (4.8) can be used for each time interval between the measurements. In this case of two measurements, counting the upstream and downstream number of particles (Particle counters B and C in Figure 4.1), one can immediately calculate the β ratio, where

$$\beta(t) = \frac{C_{upstream}(t)}{C_{downstream}(t)}. \tag{4.11}$$

It is assumed that $C_{inflow} = C_{upstream}(t) = C_{up}$ changes slowly in time, in the case when the simplest equation for the deposition rate (4.4) is used. From the analytical solution (4.9), we get

$$C_{downstream}(t) = C_{down} = C_{up} e^{\frac{-\alpha}{u^x} x}. \tag{4.12}$$

Substituting the definition of the beta ratio (4.11) into the equations, we get

$$\alpha = \frac{u^x}{d} ln\beta. \tag{4.13}$$

where d denoted the thickness of the filtering medium.

4.4.2 α computed from measurements in Section 4.2 - Variant 2

Measurements described in Section 4.2 in Variant 2 is the basis of determining α in this subsection. A number of measurements are provided by the IBS Filtran Laboratory on a selected flat filtering media, performed at different flow rates, and with different durations as shown in Figure 4.2. The tests are performed for regimes at which the filter medium is still far from clogging. Based on these measurements, we create the lookup tables, that correlate the fluid velocity and particle concentrations in front of the filtering medium to its deposition rate.

Let us now explain how the Lab measurements are used to identify the particles' deposition rate α. A standard concical filter housing is used for this purpose. After the discretization of the flow domain for the conical filter 4.8 (left), only the upstream layer of fluid voxels are considering, i.e. the fluid layer in front of the filtering medium. Let us consider a simple case when the surface area of the filtering medium is $100mm^2$ and each voxel volume is $1mm^3$. This means that 100 such voxels in front of the filtering medium are considered. Our derivations are based on certain simplifications:

- α is different from zero only in the first layer of the filtering medium, accounting only for first layer surface filtration. Our problem is then reduced to determining 100 values of α at each time moment. Note that different values for different cells are considered to account for the possible variation of the velocity and particle concentrations in front of the filter medium.
- The transport processes along the filtering medium are negligible compared to the transport processes across the filtering medium.
- The downstraem cocentration at time t is equal to the upstream concentration at time $(t+\Delta t)$
- There is a perfect mixing in front of the filtering medium.

Under all the above assumptions, 100 one dimensional equations are considered for each of the J particle size. The diffusive term is neglected in these equations, so that they look as follows:

$$\frac{\partial C}{\partial t} + (\vec{u}, \nabla C) + \alpha C = 0$$
$$\frac{C(t+\Delta t) - C(t)}{\Delta t} + \vec{u}\frac{C(t+\Delta t) - C(t)}{h} + \alpha C(t) = 0$$

Here Δt stands for the time interval. Recall that the local velocity in front of the respective porous voxel is considered. Using equation (4.14) along with the measurement results from section 4.2, α can be computed as follows:

$$\alpha(t,y,z) = \frac{(h+\Delta t u^x)(C(t) - C(t+\Delta t))}{h\Delta t C(t)} \tag{4.14}$$

The above equation is used to create look up table, which correlates the upstream velocity u^x, and particle concentration $C(t)$, to the local deposition rate. For simplicity of notations, we assumed the flow to be in the $x-$ direction, and the filtering medium is aligned with the (y,z) plane. The concentration of particles at each time moment is known from the measurements, and can be used to calculate α from the equation above (in the simulations, at a given time moment, $C(t)$ is already calculated, and it is used together with velocity and α to compute $C(t+\Delta t)$). In general, a series of measurements are needed, each performed for a different prescribed inflow velocity. It is only then that we can create comprehensive look up tables accounting for the dependence of α on different velocities.

The above treatment of the local deposition rate is the basis for the TFEM simulations discussed in Section 4.6. For the computation of velocity and particle concentrations, model equations (4.3) and (4.1) are used.

Weighted Interpolation search for varying upstream velocities

In the above discussions, we saw that the correlation tables were created for a specified range of velocities. Let us denote the velocities for which lookuptables are created by u^1_{lut}, u^2_{lut} etc. Equation (4.14) can be written in a simplified abstract form

$$\alpha \quad = \quad \gamma_1(C)u + \gamma_2(C). \tag{4.15}$$

where γ_1 and γ_2 are denoted as the derivative and intercept coefficients for the respective lookup table. The exact form of γ_1 and γ_2 comes from 4.14. Here, we try to the address the case when the lookup table is unavailable for a specified inflow velocity u_{in}. In this respect, there could be cases where u_{in} differs from the ones for which there are available lookup tables. On the other hand, owing to the complex geometry of the filter element or shape of the filtering medium, there could be an uneven distribution of velocity and particle concentrations, as illustrated in Figure 4.3. It is unrealistic to assume that measurements would be available for every velocity to be used. For this purpose, we look for some sort of least square methods, approximation techniques, interpolation/extrapolation techniques etc. In particular, we borrow ideas from interpolation techniques and employ a weighted interpolation and search algorithm, as shown Algorithm in 2 based on upstream velocity u_{up}. Currently this is accounted for by computing local values of α for each voxel(discretization element).

4.5 Multiscale approach for virtual filter element design

In this section, we first make a scale distinction, where the macroscopic equations include the flow and transport equations (4.1) and (4.3). The microscopic equations (cf.

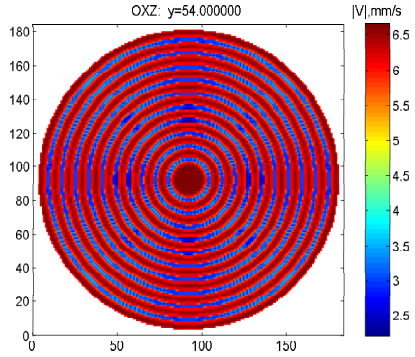


Figure 4.3: Case when there is an uneven distribution of upstream velocities.

Algorithm 2: Interpolation search algorithm for an unprescribed upstream velocity for computing the depostion rate α.

1. Given u_{up}
2. Pick out u^1_{lut}, u^2_{lut} s.t. $u^1_{lut} < u_{up} < u^2_{lut}$
3. Compute weighted coefficient $\varpi = \frac{u^2_{lut} - u_{up}}{u^1_{lut} - u^2_{lut}}$
4. Chose γ^1_1 and γ^1_2 from lookuptable based on u^1_{lut}
4. Chose γ^2_1 and γ^2_2 from lookuptable based on u^2_{lut}
6. Compute $\alpha^1 = u_{up}\gamma^1_1 + \gamma^1_2$, $\alpha^2 = u_{up}\gamma^2_1 + \gamma^2_2$
7. $\alpha = \varpi\alpha^1 + (1-\varpi)\alpha^2$

[68], [77]) include the Stokes equations with periodic conditions for modeling the flow

$$\mu\Delta\vec{u} + \vec{f} = \nabla p \tag{4.16}$$

$$\nabla \cdot \vec{u} = 0 \tag{4.17}$$

and a form of stochastic ordinary differential equations for decribing the motion and deposition of particles

$$d\vec{u_0} = -\gamma \times (\vec{u_0}(\vec{x}) - \vec{u}(\vec{x}))dt + \frac{Q\vec{E}(\vec{x})}{m}dt + \sigma \times d\vec{W}(t) \tag{4.18}$$

$$\frac{d\vec{x}}{dt} = \vec{u_0}. \tag{4.19}$$

Here t, $\vec{x}$, $\vec{u_0}$, m, Q, E, $\vec{u}$, μ denote the time, particle position, particle velocity, particle mass, particle charge, electric field, fluid velocity, and fluid viscosity respectively. Additionally, $d\vec{W}(t)$ is the 3D probability measure, where $\left\langle d\vec{W}_i(t), d\vec{W}_i(t) \right\rangle = \delta_{ij}dt$. $\gamma = 6\pi\rho\mu\frac{R}{m}$, where ρ is the fluid density, R is the particle radius respectively. Lastly, $\sigma^2 = \frac{2k_BT\gamma}{m}$ is the fluctuation dissipiation theorem where k_B is the Boltzmann constant and T is the ambient temperature.

It has already been discussed that the processes at different scales are not entirely independent. The microscale geometry changes due to deposited particles, which further

impacts the macroscale parameters such as permeability and deposition rate. On the other hand, macroscopic velocity influences the micro solution as the ratio between velocity and other forces acting on particles change. Following, we propose two ideas of how the coupling micro and macro models is possible.

4.5.1 Micro-meso-macro scale simulations

In the first case, we discuss the coupling of micro- and macro- scale simulations in a relatively straight forward approach. Firstly, the velocity, pressure and particle concentrations are solved using the fractional time step discretization method within the complete filter element, using Equations (4.1) and (4.3). Note that the time discretization is done at a coarser scale. The macroscopic velocity solution (for some selected voxels) at coarser time moments is then downscaled and extrapolated, which is used as a boundary condition for the microscale cell problem. Such cell problems are solved consecutively only at selected locations of the filtering medium. Within one time step, macro scale and micro scale equations are consecutively solved, with a proper exchange of information in between these semi-steps. On the micro scale, the particles are deposited, the local filter efficiency and permeability is computed. The macro scale parameters, i.e. permeability K and depostion rate α are consecutively upscaled from the subproblems solved on micro scale problems.

The macroscopic solution at each time step are to be downscaled to provide input velocity and particles distribution for the micro scale simulations. The changes in the microstructure have to be monitored in selected locations of the filter media in order to provide proper information for the upscaling procedure.

A sketch of one time step of the coupling procedure is as follows.

1. At the selected locations, as shown in Figure 4.4 of the filtering porous media, local Stokes problems, as well as stochastic ODEs describing the movement and deposition of particles, are solved;

2. Based on a consecutive upscaling procedure, these results are used to update permeability and the absorption rate in the selected locations in a piece of resolved microstructure;

3. A proper interpolation procedure is used to calculate proper permeability and absorption rate in the full porous medium;

4. The updated permeability and absorption rate are used to perform a semi time step with the macroscopic algorithm;

5. The velocities and the concentration of particles are downscaled in order to provide input for the micro scale computations at the next time step.

However, the costs for such an algorithm are assessed in advance. It is noted that there is a need to incorporate two time scales within the model. Moreover, the micro simulations would be possible only on selected locations, which would require some apriori information on the selection of macro scale voxels for which micro problems need to be solver. Additionally, the algorithm comes with a cost of post processing interpolations of upscaled parameters. In the following subsection, we reformulate the approach in another way to reduce the aforementioned difficulties.

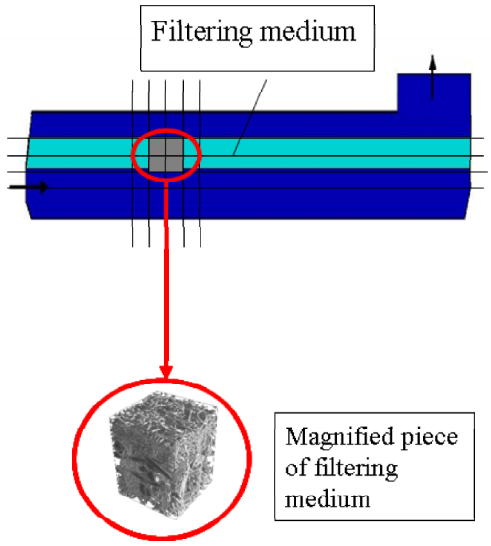


Figure 4.4: Downscaling of local velocity and particle concentrations to be done for selected locations of the filtering medium.

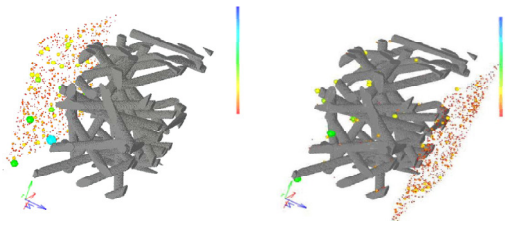


Figure 4.5: Left: Downscaled meso scale velocities and particle concentration to be used as input into micro scale simulations. Right: On the micro scale, particles are deposited, the local filter efficiency and permeability is computed.

4.5.2 Virtual Element Filter Design- Coupled Micro Macro Simulations

In Section 4.4, it was observed that expensive measurements are needed for determining the parameters. In this subsection, we propose an alternate approach where the microscale simulations are used to determine the coefficients for the macroscopic equations. This approach significantly reduces the efforts in acquiring measurements data,

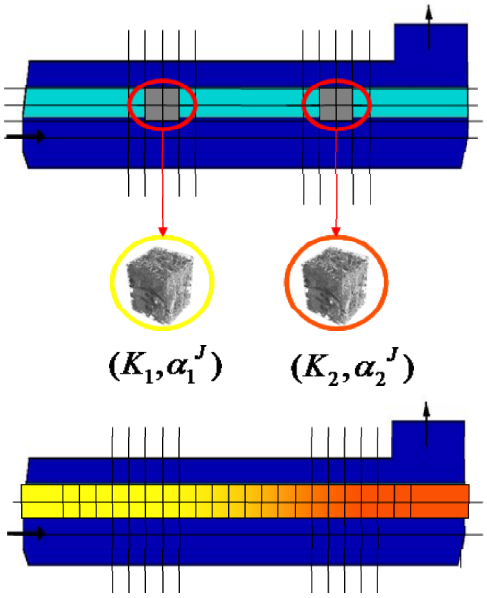


Figure 4.6: The permeability found in selected locations of the porous medium from microscaleis to be inserted back into the macro scale equations.

and in turn significantly reduces the costs and the time for new designs. Moreover, this approach provides an option to evaluate the filter efficiency without manufacturing the filter medium and the filtering element.

In the previous subsection, we saw that at each time moment, the microscale simulation provided detailed information about the amount (concentration) of free particles and of the deposited particles. The deposition rate α is then determined. Instead of exchanging the information at every time moment, it is proposed to save the information provided by the Micro scale simulation apriori before starting the macro scale simulations. This is done in the following way. A virtual filter medium is designed, as illustrated in Figure 4.7(a) using the GeoDict software. Flow is computed using equation (4.16) and the particle transport and deposition is computed using equation (4.18). Lookup table curves, similar to those derived from discussions in Section 4.4 result from the micro scale simulations. The micro algorithm is run several times for different flow rates and information is processed into correlation tables, as illustrated in Figure 4.7(b). The macro scale simulations can use the lookup tables to determine the deposition rate for equation (4.3).

Note that the data provided by the measurements can be replaced by micro simulations for a virtually designed filter medium. The same velocity interpolation approach can be further employed within the algorithm as discussed in subsection 4.4.2.

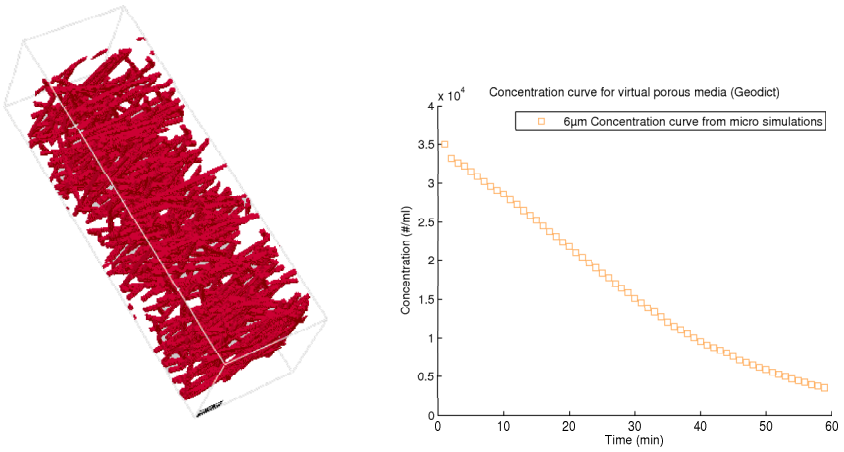


Figure 4.7: (a) Example of a virtual filtering medium generated by Geodict Software. (b) Concentration curve from micro simulations pertaining to the respective filtering medium.

4.6 Numerical Results

In this section, the particular results from experiments and simulations regarding various aspects will be provided. We try to validate the simulation results against analytical solution and experiments.

In the numerical setup, we first describe the physical and numerical parameters used. The inflow velocity, corresponding to different flowrates for different problems are specified at the inlet. For the simple parallelepiped filter, size of the inflow is $50mm^2$. We chose this filter for the particular reason that analysis and validation of the algorithm becomes a relatively simpler task due to its simple geometry where creeping flow along inclined walls etc. are neglected. In the second geometry, we closely match the experimental setup with a conical filter housing as shown in Figure 4.8. Here, the size of inflow is the same size of the filtering medium using in the experimental setup, i.e. $248cm^2$ and the domain is discretized into $185 \times 73 \times 185$ cells.

For all the examples considered in this section, the filtering media has the following properties: the density distribution of fibres is considered to be uniform and an average porosity of 0.932 and the solid volume fraction 0f 0.068 is calculated for the clean filtering medium. Permeability is assumed to be $8.41758e-005mm^2/mm$. Moreover, the density of oil is taken to be $8.386000e-007kg/mm^3$.

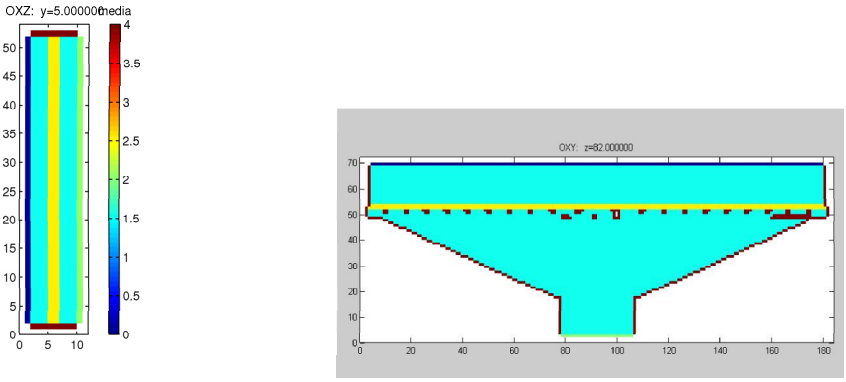


Figure 4.8: Left: Channel Filter Housing. Right: Conical Filter Housing. Inlet, outlet and the filtering medium is marked with colors blue, green and yellow respectively.

4.6.1 Channel Filter using synthetic data

We first test our algorithm using synthetically generated data for the Channel Filter using complete set of measurements, i.e. upstream and downstream particle counts are available as discussed in Section 4.4.1. This includes the balance equation in the tank, i.e. multipass including injection of dirt particles at a specified injection flow rate. Firstly, results in 4.9 are presented for the Channel filter, Figure 4.8 (left). We compare the simulation results with the synthetic data, labeled as 'Analytical solution' in Figure 4.9. In the Figure, we provide plots corresponding to different time steps. Comparing these plots, we see that the solution converges for smaller time steps.

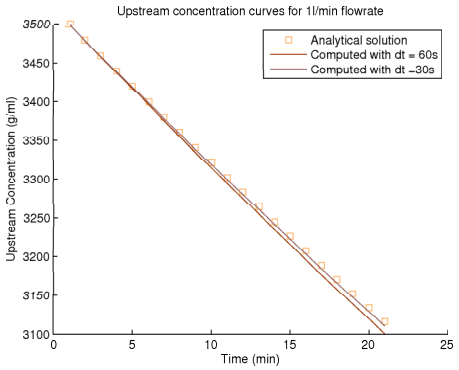


Figure 4.9: Upstream concentration curves for channel filter for flowrate 1 l/min.

4.6.2 Conical Filter using synthetic data

In this section, we present simulation results for the case of conical filter for different flow rates and for different time steps as shown in Figure 4.10.

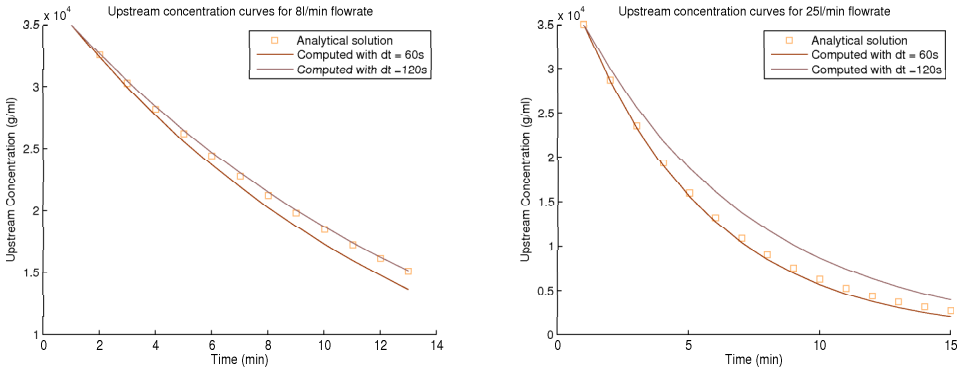


Figure 4.10: Multipass upstream concentration curves for flowrates 8 and 25 l/min

4.6.3 Conical filter using experimental data

Results using α computed from approach in Subsection 4.4.1

The results in Figure 4.11 correspond to the case where constant deposition of the contaminant is considered, as discussed in Section 4.4.1. The deposition rate is computed using Equation (4.13) with model used for initial stages of filtration, i.e using Equation (4.4). We observe that the deposition rate computed from the analytical solution is in accordance with the measurements.

Next, we compare the two models, i.e. Equations (4.4) and (4.7) respectively. When

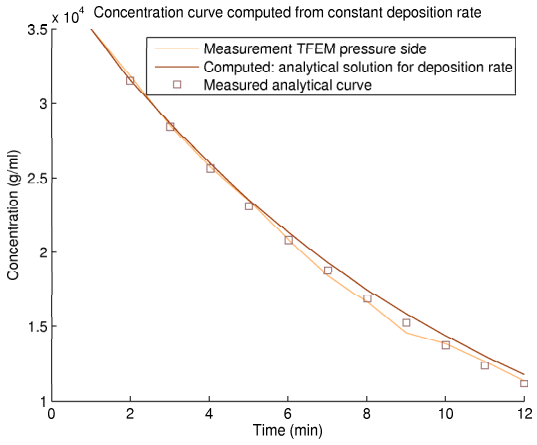


Figure 4.11: Using deposition rate α computed from analytical solution 4.13 for flow rate $8l/min$.

α is computed using Equation 4.4, it is observed that there is complete cleaning, ie. concentration goes to zero. This may be true for the initial stages of filtration for a completely clean filtering medium. In reality, as time elapses, deposition is directly related to the mass of already captured particles and the capacity of the filter itself,

as modeled by Equation (4.7). Moreover, one observes the 'saturation of efficiency' phenomena.In this sense, M_0 can be interpreted as the limiting capacity of the filtering media. The results shown in Figure 4.12 compares the concentration curves vs time using both model equations for deposition rate. Equation (4.7) is tested for two cases, i.e. $M_0 = \ell C_{in}$, where $0 \leq \ell \leq 1$. In the first case, we assume that there is no limiting capacity on the filter, implying that the filtering medium has the capacity to capture all that comes in, i.e. $M_{=}0 = C_{in}$. In the second case $\ell = 0.9$, implying $M_0 = 0.9C_{in}$.

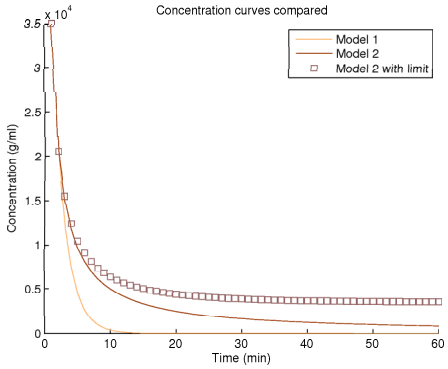


Figure 4.12: Comparing concentration curves for different models for flow rate $25l/min$.

Results using α computed from approach in Subsection 4.4.2

In this section, we also specify the components of the algorithm that are used in the computations while illustrating results. A comparison between the measurement data and simulations is shown in Figure 4.13 using α computed from Equation (4.14) in Section 4.4.2. Note that α is determined for different flow rates. The solid lines represent measurement data on a flat sheet housing that is used to create correlation tables within the algorithm. The measurement curves correspond to a selected filtering medium. Another representation of the results can be seen on the efficiency curves shown in Figure 4.13 (right). The efficiency of the filter is estimated using the particle concentrations described on the curve, which further serves as an indicator towards the performance and lifetime of the filter.

In Figure 4.13, results were compared with measurements for which data was available, i.e. for 8, 25 and 40l/min flowrates. What happens for the cases when the velocity of fluid within the filter is different from the ones for which measurement data is available? For this purpose, we employ the interpolation Algorithm 2. We test the interpolation algorithm for flow rate 15l/min, for which the results are shown in Figure 4.14.

It is observed from Figure 4.14 that velocity interpolations is not only useful when the system flow rate is different from the measurement data, but also when the velocity in

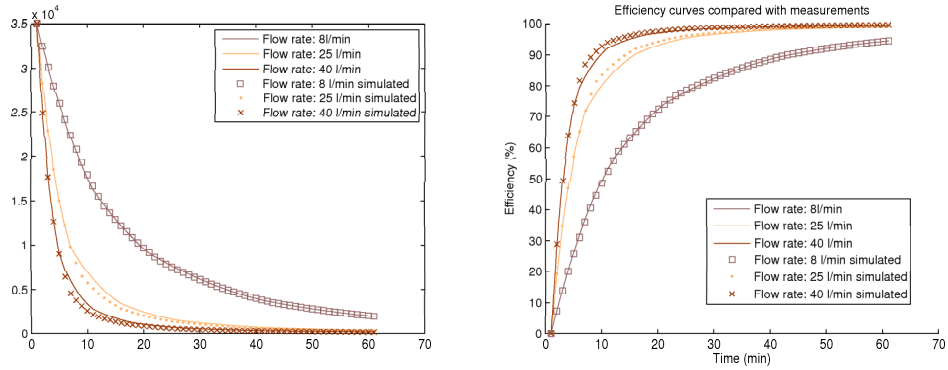


Figure 4.13: Obtained concentration and efficiency curves matched against measurements for flowrates 8, 25 and 40 l/min.

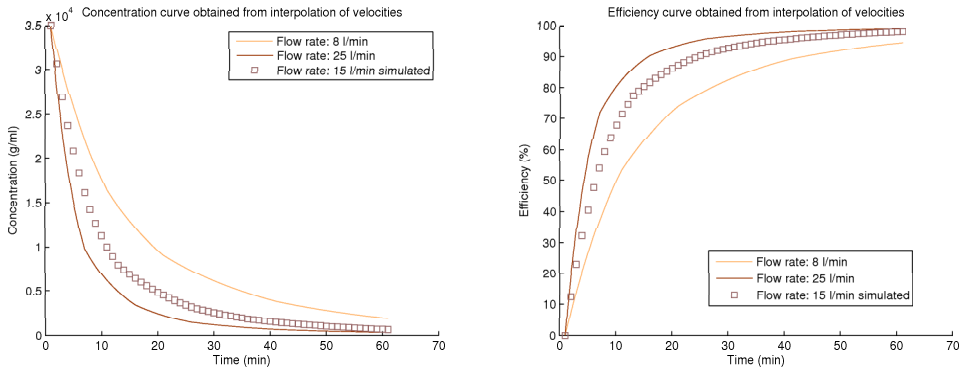


Figure 4.14: Obtained cleanliness and efficiency curve for 15 l/min using velocity interpolation.

front of the filtering media is not uniform. The varying velocities can be accounted for using such a form of velocity interpolation.

Next, we do a simple test for different time steps of the algorithm. For the first case, we take a total time interval of one hour, with a one minute/60 seconds time step. The second case, we again take a total time interval of one hour but with 30 second time stepping. The results obtained are shown in Figure 4.15.

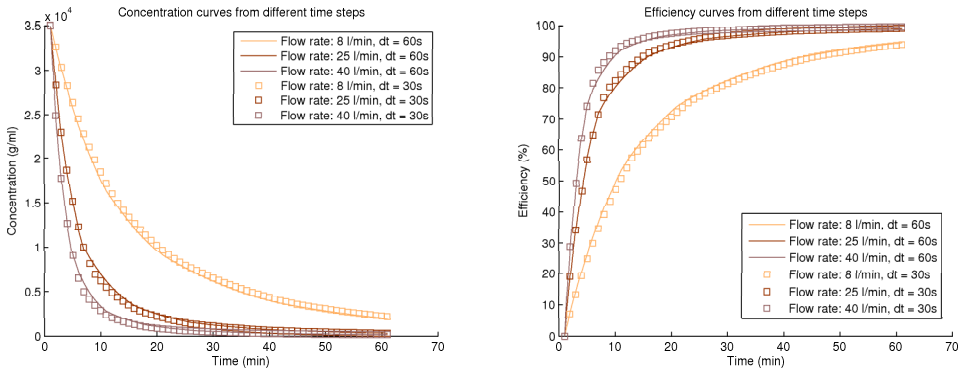


Figure 4.15: Obtained concentration and efficiency curves for different flow rates for different time steps.

4.6.4 A real industrial filter

In this subsection, we compare results with measurements on a real industrial filter, called the FCVT Filter used as an automotive oil filter, as shown in Figure 4.16. For these simulations, correlation tables created for the Flat sheet housing for the Conical Filter showed in the previous section for flowrates 8 l/min, 25 l/min and 40 l/min were employed, to mainly cover for the variation (complete range) of upstream velocities of the filtering medium. The simulation results meet our goal of employ the pre-created lookup tables for flat sheet media and use it for arbitrary filter element simulations.

Algorithm 1 is used to solve the Navier-Stokes-Brinkmann system for the fcvt filter. Figure 4.17 shows the computed velocity profiles. We observe that there is a non uniform distribution of velocity profile and the flow is more concentrated in regions behind the outlet of the filter. Such variations are typical in real filters, depending on the complexity of the filter housing, the complex shapes in which the filtering medium could appear, or due to the spatial location of the inlet/s and outlet/s of the filter itself. Figures 4.18, 4.19, and 4.20 show the raw data in the form of particle counts for different particle sizes for a given system inflow flowrate 8 l/min. Results from simulations are matched against the measurement data and it is observed that the simulations mimick the real process reasonably well.

Figure 4.16: Filter housing of the fcvt filter.

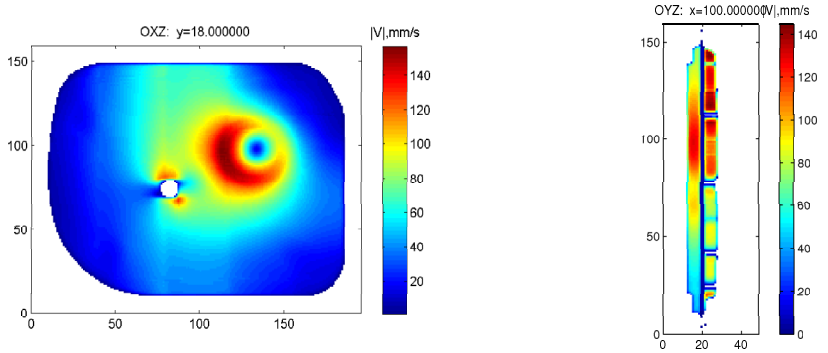


Figure 4.17: Velocity profiles computed using Algorithm 1. Left: Velocity magnitude in the layer in front of the filtering medium. Right: Velocity magnitude in the Y-Z cross section of the filter. It shows the varying velocities with the filter.

Lastly, we perform simulations for different inflow flowrates and observe the behaviour of the filter in terms of filtration efficiency. These results are further compared with measurements and we again see good correlation with measurement results. Table 4.1 compares the measured and simulated overall efficiencies reached for a one hour time interval for the fcvt filter housing geometry. For consistency, this was done for different flowrates. Moreover, Figure 4.21 shows a graphical display of the efficiency profiles for different particle sizes for different flowrates. Simulation results are compared with the measurements.

4.7 Summary

A coupled model for fluid flow and particle transport is introduced. The velocity field of the flow equations based on Algorithm 1 was used to solve the Convection Diffussion

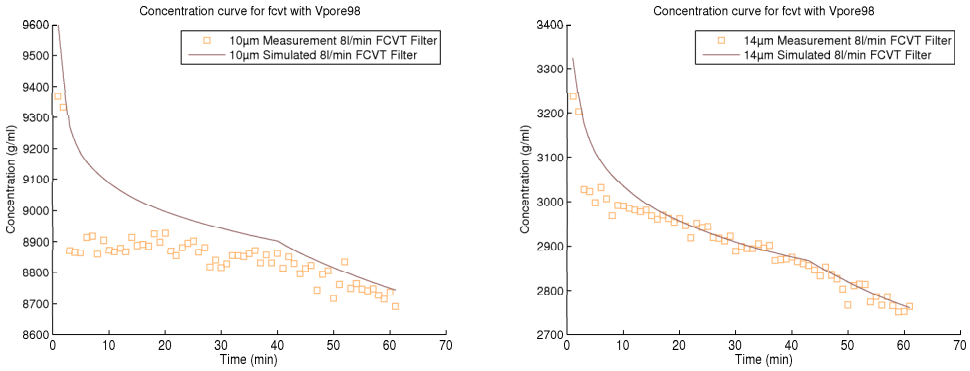


Figure 4.18: Concentration curves matched against raw measurement date for fcvt filter for different particle sizes, with system flow rate 8 l/min

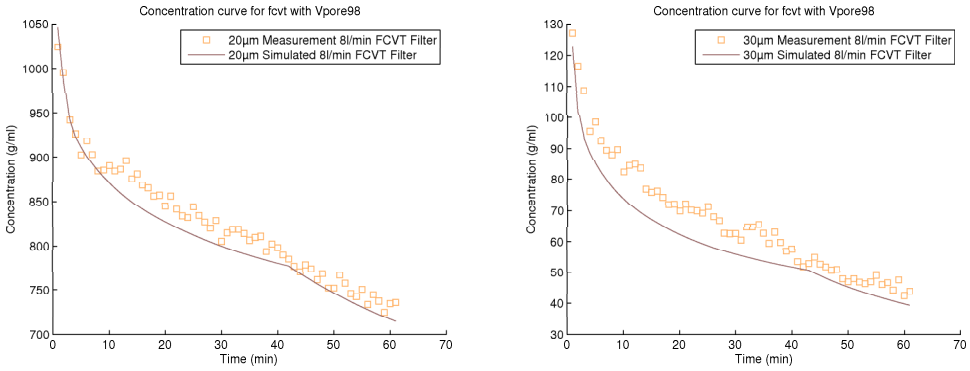


Figure 4.19: Concentration curves matched against raw measurement date for fcvt filter for different particle sizes, with system flow rate 8 l/min

Reaction equation for particle transport. Determination of the deposition rate was done by employing various methods. In the first approach, analytical solution of a one dimensional convection-reaction equation is employed, along with Lab measurements. In the second approach, the deposition rate is solely determined from given set/s of measurements. In both cases, the derived deposition rate was employed in the model equations and results were compared against measurements. It was observed that the methods employed worked well under the prescribed assumptions. An alternative approach for determining the deposition rate is proposed based on micro scale simulations. The results show excellent correlations for the tests using the flat sheet sheet medium within a Conical Filter housing. Results show that using interpolation for different infront velocity and particle concentrations for determining the deposition rate also worked well. Moreover, it is shown that the algorithm can be successfully employed for arbitary filter elements to determine its filtration efficiency.

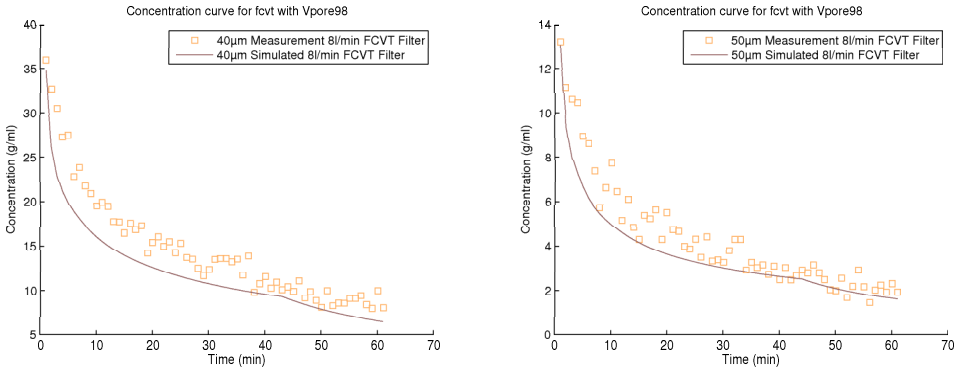


Figure 4.20: Concentration curves matched against raw measurement date for fcvt filter with for different particle sizes, with system flow rate 8 l/min

Table 4.1: Particle reduction efficiencies for different flowrates

Particle Size	10	14	20	30	40	50	60
8 l/min measured (%)	7.40	14.70	28.00	64.8	76.4	84.4	85.00
8 l/min simulated (%)	8.88	16.9	31.6	67.99	81.5	87.57	90.52
25 l/min measured (%)	13	24.9	45.4	81.9	90.8	96.5	99
25 l/min simulated (%)	16.4	29.3	49.6	86.2	93.6	95.6	96.58

Current work in progress

The problem under consideration is highly complex. On the contaminant level, the dust particles suspended in the fluid can be of different quantities, sizes, shapes, densities and are randomly distrubuted within the fluid. From the side of the filtering media, in reality, it is packed with a non uniform distribution of fibres, resulting in nonuniformity in the loading of particles. The complex shape of the filtering media and the filter housing imposes additional complexities, whereby deposition, sedimentation, and gravitational effects start to play role. Additionally, effects like washing away, breaking and agglomeration of particles are prevalent in almost all applications. On the other hand, the problem is constantly changing over time, in the sense that the micro geometry is never constant. With the capturing, deposition and transport of particles, the micro geometry is varying, which has a direct impact of the defining parameters of the constituent equations, such as permeability, deposition rate, porosity etc. Even though the dynamic aspect was not the main topic of our research, it still opened up new aspects of research. Next immediate plans are to look into the effect of changing permeability and the redistribution of flow in correlation with the non uniform deposition of particles within the filtering medium. We expect an overall increase of pressure over time. Several aspects become important in this line of research, e.g. what is the best possible way to recompute permeability for clean/partially loaded/clogged filtering medium? How to

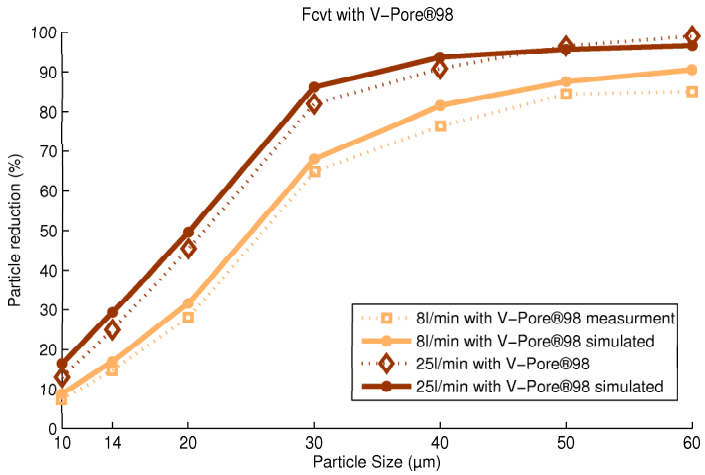


Figure 4.21: Efficiency profiles for different particle sizes for flowrates 8 l/min and 25 l/min (measured and computed).

incorportae the sedimentation and gravitational effects into the model? How to account for the different time scales for the different processes happening on each scale? Our current ongoing work addresses part of these questions but is excluded from this thesis as it is still not completed. However, its worthwhile to note that this work opens new dimensions for future research on this topic.

Chapter 5

Summary

This thesis provided a general framework to overcome the computational limitations and to incorporate multiscale models for filtration related problems. With the aid of the model and algorithms considered, an accurate filter simulation tool was developed, incorporating the several scale models to enable the design of filters and optimize the filter performance.

Chapter 1 discusses a single grid version of the flow algorithm for the Navier-Stokes-Brinkmann equations. We observe that the simulation of such an algorithm imposed serious restrictions on the computational size (and solving time) of the problem. In this regard, a parallel algorithm is developed.

The flow algorithm (single grid and the parallel version) is incorporated in a software simulation tool called Suction Filter Simulation (SuFiS). This is currently in use at IBS Filtran, and is reported to help in the design of oil filters, optimal shape design for the filter housing and achieving optimal pressure drop - flow rate ratio. With continuous use, it is observed that the CFD simulations assisting this design does not always require very high accuracy for the flow velocity within a filter element, as long as the pressure drop over the complete filter element is properly computed. Keeping this in mind, a numerical upscaling subgrid method is developed in Chapter 2 as a computational method for achieving the desired accurate pressure drop for the Navier-Stokes-Brinkmann problem. It is mainly employed for cases where the coarse computational grid is unable to accurately account for the complicated geometry and/or the filter media.

Alternatively, for the cases where the geometrical features are at different scales, various approaches for multiscale problems can be adopted to provide an increase in the efficiency of the flow algorithms. In this respect, the iterative Multiscale Finite Volume (iMSFV) Method, earlier introduced in [59, 45], for solving equations describing flow in heterogeneous porous media, is extended to solve equations describing coupled flow in plain and porous media. This is the work presented in Chapter 3. The Steady state Stokes equations govern the flow in the plain domain, while the Darcy problem is considered for the porous domain. However, the extension to the unsteady case, and/or

to the Brinkman equation in the porous media, is relatively straightforward. As future work, we would like to adopt this algorithm for the Navier-Stokes-Brinkmann equations on non uniform collocated grids.

We notice that the flow algorithms contribute essentially to the design of filters with improved performance. However, prediction of the efficiency and lifetime of the filter require detailed knowledge not only of the flow field transport but also the capturing of impurities through the filter. For this purpose, several other challenging problems are solved to further improve its design: (1) modeling transport and capturing of the dirt particles by the filtering media; (2) parameter identification methods for parameteres in the particle transport model; (3) dependence of flow velocity on particle capturing and filtration efficiency. Within the work of this thesis in Chapter 4, we address each of these issues, with special attention to the usage of correct parameters and to the validation of the algorithm.

The essential contributions of this thesis are as follows:

- A parallel algorithm for the Navier-Stokes-Brinkmann equations, based on a data decomposition approach using MPI implementation is developed.
- Parallelization of the linear solver (BiCGSTAB) using threaded OpenMP approach.
- A numerical upscaling subgrid algorithm for the Navier-Stokes-Brinkmann equations, with a strong emphasis on verifying the accuracy of the upscaled coefficients.
- Extension of the iMSFV method for solving a coupled Stokes-Darcy system.
- A numerical algorithm for the coupled flow and transport problem, namely the coupling of the Navier-Stokes-Brinkmann equations and the Convection-Diffusion-Reaction equation. A larger work in this regard was spent on identifying parameters for the model equations.

5.1 Concluding remarks

In a nutshell, the thesis touched upon essential topics for modeling and solving flow in porous media, ranging from applications such as automotive oil filters to petroleum oil reservoir simulations. All methods developed could be used, with some easy modifications, for many different applications. It should be noted that the developed implementation of the parallel algorithm, subgrid algorithm and micro-macro coupled efficiency algorithm is successfully in use with one of the industrial partners of the Fraunhofer ITWM.

List of notations

Below, some of the used notations are listed. They are subdivided into the two groups –namely, notations, which are common throughout the whole manuscript, and notations specific for every chapter.

Common notations

Ω	computational domain
Ω_P	porous part of the computational domain
Ω_F	fluid part of the computational domain
Ω_S	solid part of the computational domain
$\partial\Omega$	external boundary of Ω
$\bar{\Omega}$	$\Omega \cup \partial\Omega$
p	fluid pressure
f	right hand side of Navier-Stokes-Brinkmann equation
t	time
$\mathbf{K}$	permeability
μ	viscosity of the fluid
^{k}	time level
τ	$t^{k+1} - t^k$
U_{in}	Inlet velcoity
P_{in}	Inlet pressure
P_{out}	Outlet pressure
$\mathbf{I}$	unit tensor
D	discrete diffusion operator
G	discrete gradient operator
G^T	discrete divergence operator

Notations used in Chapter 1,2,4

d	(x,y,z)
C	discrete convective operator
B	discrete Brinkmann operator
N	Number of finite volumes in computational domain, $\{1,...,N\}$
n	index indicating discretization in space
E_n	finite volume, where $n \in 1,...,N$
$\mathscr{N}_n$	set consisting of all neighbouring finite volumes of E_n
(i,j,k)	coordinates of E
h_n	grid size of E_n in each direction, $h = (h^x, h^y, h^z)$
v_n	volume of E_n
$\vec{u}$	(u^x, u^y, u^z) - fluid velocity
$\vec{u}^*$	approximate velocity
q	pressure correction, $p^{k+1} - p^k$
α	message startup time
$\mathscr{P}$	Preconditioner of the discretization matrix under consideration
m	number of data items sent between 2 processors
β	time required to send one element of data
s	cost of broadcasting data items between 2 processors
P	number of processes
$\vec{u}_0$	coarse scale velocity vector
p_0	coarse scale pressure vector

Notations used in Chapters 3

d	(x,y)
$\mathbf{n}$	unit normal vector
$\mathbf{t}$	unit tangential vector
$\vec{u}$	continuous velocity vector, $\vec{u} = (u, v)$
$\bar{\omega}^p$	staggered grid for pressure
$\bar{\omega}^u$	staggered grid for velocity component u
$\bar{\omega}^v$	staggered grid for velocity component v
$\vec{u}^*$	intermediate velocity in the SIMPLE algorithm
$\delta\vec{u}$	velocity correction for the SIMPLE algorithm
δp	pressure correction for the SIMPLE algorithm
p'	fine scale pressure values
$\bar{p}$	coarse scale pressure values
N_B	Number of primal coarse blocks
N_D'	Number of dual coarse blocks
E	A block of the primal coarse grid
E'	A block of the dual coarse grid
$\mathring{E}$	Interior of E
$\mathring{E}'$	Interior of E'
$\mathscr{E}$	Edges of E
$\mathscr{E}'$	Edges of E'
$\mathscr{V}$	Vertices of E
$\mathscr{V}'$	Vertices of E'
ϕ	basis function
ψ	correction function
$\ddot{E}'$	Set of all dual cells belonging to E

List of acronyms

NSB	Navier-Stokes-Brinkmann
MSFV	Muliscale Finite Volume
iMSFV	iterative Multiscale Finite Volume
MsFEM	Multiscale Finite Element Method
FV	Finite Volume
FRM	Fictitious Regions Method
CV	Control Volume
SuFiS	Suction Filter Simulation
TFEM	Transmission Filter Effectiveness Method
REV	Representative Elementary Volume

Bibliography

[1] J. Aarnes, On the use of a mixed multiscale finite element method for greater flexibility and increased speed or improved accuracy in reservoir simulation, SIAM MMS, 2 (2004), pp. 421-439.

[2] G. Allaire, Homogenization of the Navier-Stokes equations in open sets perforated with tiny holes. i: Abstract framework, a volume distribution of holes, Arch. Ration. Mech. Anal. 113 (1991), no. 3, 209-259.

[3] Ph. Angot: Analysis of singular perturbations on the Brinkman problem for fictitious domain models of viscous flows. Math. Meth. Appl. Sci., **22** (1999) 1395–1412.

[4] T. Arbogast, Analysis of the simulation of single phase flow through a naturally fractured reservoir., SIAM J. Numer. Anal. 26 (1989), 12-29.

[5] T. Arbogast, Analysis of a two-scale, locally conservative subgrid upscaling for elliptic problems, SIAM J. Numer. Anal. 42 (2004), no. 2, 576-598.

[6] T. Arbogast, Implementation of a locally conservative numerical subgrid upscaling scheme for two-phase Darcy flow, Computational Geosciences 6, 453-481, 2002.

[7] T. Arbogast, J. Douglas, Jr., and U. Hornung, Derivation of the double porosity model of single phase flow via homogenization theory, SIAM Journal on Mathematical Analysis 21 (1990), no. 4, 823-836

[8] T. Arbogast and K. Boyd, Subgrid upscaling and mixed multiscale finite elements, SIAM J. Numer. Anal. 44 (2006), no. 3, 1150-1171.

[9] T. Arbogast, S. L. Bryant, A two-scale numerical subgrid technique for waterflood simulations, SPE Journal 7 (2002) 446-457.

[10] P. Bastian. *Numerical Computation of Multiphase Flows in Porous Media*. Habilitation Dissertation, Kiel university, 1999.

[11] S. Balay, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L. Curfman McInnes, B.F. Smith, H. Zhang. PETSc Users Manual. ANL-95/11 - Revision 2.3.0, Argonne National Laboratory, 2005.

[12] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, 1994.

[13] J. Bear, A. Verruijt, *Modeling Groundwater Flow and Pollution*, D. Reidel Publishing Company, Dordrecht, Holland, 1987.

[14] J. Bear, Y. Bachmat.: Introduction to Modeling of Transport Phenomena in Porous Media. Kluwer Academic Publishers, Dordrecht, Holland, 1990.

[15] M.Benzi, G.H.Golub, J.Liesen, Numerical Solution of saddle point problems, Acta Numerica 14, 1-137, 2005.

[16] G. Bonfigli, P. Jenny, An efficient multi-scale poisson solver for the incompressible navier-stokes equations with immersed boundaries. J. Comp. Phys., 228(12), 2009.

[17] H.C. Brinkman, A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles, Appli. Sci. Res., t. A1, (1947), 27-34.

[18] R.C. Brown, Air Filtration, An integrated Approach to the Theory and Applications of Fibrous Filters, Pergamon Press, 1993.

[19] R.C. Brown, D. Wake, Loading Filters with Monodisperse Aerosols: Macroscopic Treatment, Journal of Aerosol Science, Vol. 30 (2), pp. 227-234, 1999.

[20] A. Bruaset, A. Tveito. *Numerical Solution of Partial Differential Equations on Parallel Computers*. In: Lecture notes in computer science and engineering. Vol. 51, Springer, 2006.

[21] A. J. Chorin, Numerical Solution of Navier-Stokes equation, Mathematics of Computation 22 (1968), 745-760

[22] A. Churbanov, A. Pavlov, P.Vabishchevich, Operator-splitting methods for the incompressible Navier-Stokes equations on non-staggered grids. I: First-order schemes. *Int. J. Numer. Methods Fluids* **21**, No.8, pp.617-640 (1995).

[23] R. Čiegis, O. Iliev, Z. Lakdawala, On parallel numerical algorithms for simulating industrial filtration problems. Computational Methods in Applied Mathematics 7(2), 118-134(2007).

[24] R. Čiegis. Analysis of Parallel Preconditioned Conjugate Gradient Algorithms. *Informatica*, **16**(3), 317–332 (2005).

[25] R. Čiegis, V.Starikovičius. Realistic performance tool for the parallel block LU factorization algorithm. *Informatica*, **14**(2), 167–180, 2003.

[26] Y. Chen, L.J. Durlofsky, M. Gerritsen, X.H. Wen, A Coupled Local-Global Upscaling Approach for Simulating Flow in Highly Heterogeneous Formations, Advances in Water Resources, 26, 1041-1060 2003.

[27] S. Doi, T. Washio. Ordering strategies and related techniques to overcome the trade–off between parallelism and convergence in incomplete factorizations. *Parallel Computing*, **25**, 1995–2014, 1999.

[28] M. Discacciati, A. Quarteroni: Analysis of a domain decomposition method for the coupling of Stokes and Darcy equations. In: Brezzi, F., et al.(eds.) Numerical Analysis and Advanced Applications– ENUMATH 2001, pp. 3-20. Springer, Berlin (2003).

[29] M. Discacciati, A. Quarteroni, Navier-Stokes/Darcy coupling: modeling, analysis, and numerical approximation. Rev. Mat. Complut. 22(2), 315-426, 2009.

[30] L.J. Durlofsky, Numerical calculation og equivalent grid block permeability tensors for heterogeneous porous media, Water Resources Research 27, 669-708, 1991.

[31] M. Dedering; W. Stausberg ; O. Iliev; Z. Lakdawala; R. Ciegis; V. Starikovicius, On new Challenges for CFD Simulation in Filtration, Proceedings of World Filtration Congress, Leipzig, 2008.

[32] E.F. D'Azevedo, P.A. Forsyth, W.Tang. Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems. *SIAM J. Matrix Anal. Appl.*, **10**(4), 567–584, 1992.

[33] J. G. Eleftherakis, A. Khalil, Advances in Automatic Transmission, SAE Paper 2001-01-0372, Society of Automotive Engineers, Warrendale, PA 2001.

[34] H. Elman, D. Silvester, A. Wathen, Finite Elements and Iterative Solvers with applications in incompressible fluid dynamics, Oxford University Press Inc., New York, 2005.

[35] B. Enquist, Z. Huang, Heterogeneous multiscale method: A general methodology for multiscale modeling, Phys. Rev. B, 67(9), 092101, 2003.

[36] M. Ehrhardt, J. Fuhrmann, E. Holzbecher and A. Linke, Mathematical Modeling of Channel Porous Layer Interfaces in PEM Fuel Cells, in: B. Davat and D. Hissel (ed.), Proceedings of (FDFC2008) Fundamentals and Developments of Fuel Cell Conference 2008, Nancy, France.

[37] J.H.Ferziger, M.Peric, *Computational methods for fluid dynamics* (Springer, 1999).

[38] C.A.J. Fletcher: Computational techniques for fluid dynamics. Springer, Berlin etc., 1991.

[39] Fraunhofer Institute for Industrial Mathematics, Annual Report, 2002. http://www.itwm.fhg.de/en/zentral–berichte/annualreport2002/

[40] Fraunhofer Institute for Industrial Mathematics, Department Flows and Complex Structures, http://www.itwm.fhg.de/en/sks/indexsks/

[41] P. Gresho, R.Sani, *Incompressible flow and the finite element method. Volume 1: Advection-diffusion and isothermal laminar flow*. In collaboration with M. S. Engelman. Chichester: Wiley. 1044 p. (1998).

[42] M. Griebel, M. Klitz. Homogenisation and Numerical Simulation of Flow in Geometries with Textile Microstructures. SIAM MMS, 2009.

[43] A. F. Gulbransen, V.L. Hauge, K.A. Lie, A Multiscale Mixed Finite Element Method for Vuggy and Naturally Fractured Reservoirs, 21st Nordic Seminar on Computational Mechanics, Trondheim, 149-152, 2008.

[44] A. Gupta, V. Kumar, A. Sameh. Performance and scalability of preconditioned conjugate gradient methods on parallel computers. *IEEE Transactions on Parallel and Distributed Systems*, **6** (5), 455–469, 1997.

[45] H. Hajibeygi, G. Bonfigli, M. A. Hesse, and P. Jenny. Iterative multiscale finite-volume method. J. Comp. Phys., 227(19):8604-8621, Oct 2008.

[46] W. Hackbusch, Multi-grid methods and applications, 2nd edition, Springer Series in Computational Mathematics, Springer, Berlin, 2003.

[47] R. Hockney. Performance parameters and benchmarking on supercomputers. *Parallel Computing*, **17**, 1111–1130, 1991.

[48] U. Hornung. *Homogenization and porous media*, Springer Verlag New York Inc., New York, USA, 1996.

[49] T.Y. Hou, Y. Efendiev, Multiscale Finite Element Methods. Surveys and Tutorials in the Applied Mathematical Sciences, Band 4, Springer, 2009.

[50] O. Iliev, On second-order-accurate discretization of 3D interface problems and its fast solution with a pointwise multigrid solver, IMA Journal of Numerical Analysis, 22, 391-406, 2002.

[51] O.Iliev, D.Stoyanov, On a multigrid, local refinement solver for incompressible Navier-Stokes equations, *Mathematical Modelling*, vol.13, No.8, (2001).

[52] O. Iliev, D. Stoyanov: Multigrid adaptive local refinement solver for incompressible flows, Fraunhofer ITWM report No.54, 2003

[53] O. Iliev, Z. Lakdawala, V. Starikovicius, On a numerical subgrid upscaling algorithm for the Navier-Stokes-Brinkmann equations, Preprint submitted to the Journal of Computational and Applied Mathematics.

[54] O. Iliev, R.D. Lazarov, and J. Willems, Fast numerical upscaling of heat equation for fibrous materials, J. Computing and Visualization in Science, 2009.

[55] O. Iliev, V. Laptev: On numerical simulation of flow through oil filters. Comput. Vis. Sci., **6** (2004) 139–146.

[56] O. Iliev, V. Laptev, D. Vassileva : Algorithms and software for computer simulation of flow through oil filters. Proc. FILTECH Europa, 2003, Düsseldorf, pp.327–334.

[57] O. Iliev, Z. Lakdawala, R. Ciegis, V. Starikovicius, M. Dedering, P. Popov, Advanced CFD simulation of filtration processes, Proceedings of Filtech, Wiesbaden, 2009.

[58] W. Jaeger, A. Mikelic, N. Neuss: Asymptotic Analysis of the Laminar Viscous Flow Over a Porous Bed, SIAM J. Sci. Comp. 22(6), pp. 2006-2028,2001

[59] P. Jenny and I. Lunati. Multi-scale finite-volume method for elliptic problems with heterogeneous coefficients and source terms. Proc. Appl. Math. Mech., 6(1):485-486, 2006.

[60] V.V. Jikov, S.M. Kozlov, O.A. Oleinik, Homogenization of Differential Operators and Integral Functionals, Springer-Verlag, Berlin, 1994.

[61] Y. Jung, S. Torquato, Fluid permeabilities of triply periodic minimal surfaces, Physical Review Em 72, 056319, 2005.

[62] G. Karypis, V. Kumar. Parallel multilevel k-way partitioning scheme for irregular graphs. *SIAM Review*, **41**(2), 278–300, 1999.

[63] M. Kaviany, Principles of Heat Transfer in Porous Media, Springer, New York etc., 1991.

[64] A. Kumar, Numerical Estimation of Surface Parameters by Level Set Methods, PhD Thesis

[65] V. Kumar, A. Grama, A. Gupta, G. Karypis. *Introduction to parallel computing: design and analysis of algorithms*. Benjamin/Cummings, Redwood City, 1994.

[66] Z. Lakdawala, O.Iliev, G. Bonfigli, P. Jenny, On a multiscale finite volume method for the coupled Stokes-Darcy equations, In preparation.

[67] H.P. Langtangen. *Computational Partial Differential Equations. Numerical Methods and Diffpack Programming.* Springer, Berlin, 2002.

[68] A. Latz and A. Wiegmann, Simulation of fluid particle separation in realistic three dimensional fiber structures. Filtech Europa, Volume I, pp. I-353 - I-361, October 2003.

[69] V. Laptev, Numerical solution of coupled flow in plain and porous media. PhD thesis, Technical University of Kaiserslautern, 2003.

[70] Th. Levy; E. Snchez-Palencia,Equations and interface conditions for acoustic phenomena in porous media. J. Math. Anal. Appl. 61(3) , 813–834. 1977

[71] I. Lunati and P. Jenny. Multiscale finite-volume method for density-driven flow in porous media. J. Comp. Geosci., 12(3):337-350, Jan 2008.

[72] S. Ma. Comparisons of the parallel preconditioners for large nonsymmetric sparse linear systems on a parallel computer. *International Journal of High Speed Computing*, **12**(1), 55–68, 2004.

[73] M. Monga–Made, H.A. Van der Vorst. A generalized domain decomposition paradigm for parallel incomplete LU factorization preconditionings. *Future Generation Computer Systems*, **17**, 925–932, 2001.

[74] J. M. Nordbotten, P. E. Bjrstad, On the relationship between domain decomposition preconditioners and the multiscale finite volume method, Special issue of Computational Geosciences on Multiscale methods for flow and transport in heterogeneous porous media, 12(3), 367-376, 2008.

[75] S. V. Patankar, Numerical Heat Transfer and Fluid Flow, Hemishphere , 1980.

[76] K.R. Rajagopal, On a hierarchy of approximate models for flows of incompressible fluids through porous solids, Math. Models Meth. Appl. Sci., 17 (2007), 215-252.

[77] S. Rief, D. Kehrwald, K. Schmidt, A. Wiegmann, Simulation of ceramic DPF Media, Soot deposition, Filtration efficiency and pressure drop evolution, World Filtration Congress, Leipzig, 2008.

[78] C.M. Rhie, W.L. Chow, *Numerical study of the turbulent flow past an airfoil with trailing edge separation*, AIAA Journal 21 (1983),1525-1532.

[79] B. Riviere , I. Yotov, Locally Conservative Coupling of Stokes and Darcy Flows, SIAM Journal on Numerical Analysis, 42(5), p.1959-1977, 2004.

[80] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd edition, SIAM, Philadelphia, 2003.

[81] A.A. Samarskii, P. N. Vabischevich, Computional Heat Transfer, Vol. 1-2, John Wiley & Sons Inc., New York, 1995.

[82] A. Sangani, A. Acrivos, Int. J. Multiphase Flow, 8, 343, 1982

[83] V. Starikovicius, R. Ciegis, O. Iliev, Z.Lakdawala, A parallel solver for the 3D Simulation of Flows through Oil Filters, Springer Optimization and its applications, Volume 27, 181-191(2008).

[84] S. Turek, *Efficient solvers for incompressible flow problems. An algorithmic and computational approach.* Lecture Notes in Computational Science and Engineering. 6. Berlin: Springer. 352 p. (1999).

[85] D. Thomas, P. Contal, V. Renaudin, P.Penicot, D. Leclerc, J. Vendel, Modeling Pressure Drop in Hepa Filters During Dynamic Filtration, Journal of Aersol Sciences, Vol. 30 (2), pp. 235-246, 1999.

[86] P. Vabishchevich, *The method of fictitious domains in problems of mathematical physics. (Metod fiktivnykh oblastej v zadachakh matematicheskoj fiziki.)*, Moskva: Izdatel'stvo Moskovskogo Univ. 158 p. ,1991.

[87] J. Willem, Numerical Upscaling for Multiscale Flow Problems, PhD thesis, Technical University of Kaiserslautern, 2009.

[88] X.H. Wu, Y. Efendiev and T.Y. Hou, *Analysis of upscaling absolute permeability.* Discrete Contin. Dyn. Syst., Series B 2, No. 2, 185-204 (2002).

[89] ISO 16889, Hydraulic fluid power filters-Multi-pass method for evaluating filtration performance of a filter element.

CURRICULUM VITAE

31. Juli 1981	Born in Karachi, Pakistan
1986 – 1997	Primary and secondary school: St. Josephs Convent School Karachi, Pakistan
1997 – 1999	Cleared exams for Ordinary Level' Cambridge University Examination Board
1999 – 2000	Cleared exams for Advanced Level' Cambridge University Examination Board
2000 – 2003	Lahore University of Management Sciences, Lahore, Pakistan Major in Computer Science, Minor in Mathematics
2003 – 2005	Technical University in Kaiserslautern, Department of Industrial Mathematics
Dec. 2005	Masters in Mathematics
2006 – 2010	Doctorate studies at the University of Kaiserslautern, Department of Mathematics and Fraunhofer Institute for Industrial Mathematics, Department of Flows and Materials Simulation