Faculty of Physics and Astronomy

University of Heidelberg

**Diploma thesis**

**in Physics**

**submitted by**

**Jorrit Fahlke**

**born in Witzwort**

**2008**

# Pore Scale Simulation of Transport in Porous Media

**This diploma thesis has been carried out by Jorrit Fahlke at the**

**Institute of Environmental Physics**

**under the supervision of**

**Prof. Kurt Roth**

**and**

**Prof. Peter Bastian**

**Institute of Parallel and Distributed Systems, University of Stuttgart**

**Simulation von Transport in porösen Medien:** Bei der Durchführung von Stofftransportprozessen in porösen Medien beobachtet man oft eine asymmetrische Durchbruchskurve mit einem sehr langsamen Absinken der Konzentration. Dieses Phänomen tritt selbst bei nicht-sorbierenden Stoffen auf und wird als Tailing bezeichnet. Es gibt mehrere Hypothesen zur Erklärung des Phänomens, eine Modellierung erfolgt oft mit dem Mobile-Immobile-Modell (MIM). Dabei nimmt man an, das ein Teil des Wassers im Boden unbeweglich ist. Stoffe können in diese stagnierenden Zonen diffundieren, was zu dem beobachteten Tailing führt.

In dieser Arbeit wird überprüft, ob Zonen langsamen Flusses, wie sie z. B. durch sackgassenartige oder quer zur Fließrichtung liegende Poren entstehen können, tatsächlich als Ursache des Tailings in Frage kommen. Es wurde ein Programm entwickelt, um Transportprozesse auf der Porenskala zu simulieren, welches anschließend anhand verschiedener Testprobleme verifiziert wurde. Bei einer Beispielrechnung, die mit einem einfachen zufällig erzeugten porösen Medium durchgeführt wurde, ließ sich tatsächlich signifikantes Tailing beobachten.

**Simulation of Transport in porous Media:** When performing solute transport in porous media one often observes an asymmetric break-through curve with a very slow decline of the concentration. This phenomenon even appears with non-sorbing solutes and is known as tailing. There are several hypotheses to explain this phenomenon. The modelling is often done using the mobile-immobile model (MIM), which assumes that parts of the solvent are not moving along with the general flow. The solutes can move into these stagnant zones by diffusion which leads to the observed tailing.

In this thesis it is checked whether tailing can indeed be explained by stagnant zones, which may result e.g. from dead-end pores or pores perpendicular to the direction of the flow. A program to simulate transport in porous media was developed and verified using several test-problems. An example calculation with a randomly generated porous medium was performed. The resulting break-through curves showed significant tailing.

# Contents

# 1 Introduction

## 1.1 Motivation

Transport in porous media is important for many applications. Examples are the distribution nutrients in soil, the pollution of soil by toxic substances or the processes taking place in certain chemical or biological reactors.

On the pore scale the processes in porous media are understood pretty well from first principles. The same is not always true on the continuum scale however — there are several phenomena which cannot easily be explained from pore scale processes.

One such phenomenon is known as tailing. Consider a pulse of some solute moving with the water flow through soil. The displacement of the pulse will happen according to the macroscopic flow field, but the pulse will also be spread by diffusion. After some time, the shape of the pulse should be essentially Gaussian, independent of its original shape. Experiments show that the shape of the pulse is indeed mainly Gaussian, but they also observe a long tail trailing behind, making the shape rather asymmetrical.

To explain this tail the mobile-immobile model was introduced. On the pore scale, it splits the water phase into a moving and an unmoving part, the mobile and the immobile phase respectively. In uniform porous media, the mobile phase is identified with the flow channels, while the immobile phase is identified with stagnant regions of minimal velocity, which might occur in dead-end pores and the like. Exchange of solute between the two phases will happen by diffusion exclusively.

On the continuum scale, in addition to transport and dispersion in the mobile phase there is now an additional mechanism which has to be taken into account. Solute can now move into the immobile phase, essentially building a reservoir. When the main pulse has passed, the solute will move slowly back into the mobile phase, leading to the observed tail.

## 1.2 This work

In this work we examine whether tailing can indeed be explained by the mobile-immobile model. We simulate transport in a sample porous medium numerically on the pore scale. As fundamental processes we use only convection and diffusion.

If we indeed observe a Gaussian pulse with a tail we know that convection and diffusion alone are enough to explain tailing. It also tells us that stagnant regions are indeed a significant part the immobile phase. Of course this would not preclude the contribution of additional fundamental processes and neither would it preclude additional contributions to the immobile phase. These would require separate examinations.

Simulation on the pore scale should allow us to identify stagnant regions. The mobile-immobile model gives no criterion where exactly to make the distinction between mobile and immobile part. If we find a region however, which contains significant concentration long after the main pulse has passed it, we can safely assume we have found a part of the immobile region.

## 1.3 Overview

Chapter 2 gives a short summary of the transport processes encountered in porous media. The transition from the pore to the continuum scale is discussed and the mobile-immobile model is presented. Finally the transport equation is discussed.

Chapter 3 deals with the discretization of the transport equation. The theory of partial differential equation is touched and the discontinuous Galerkin space discretization method is presented. This is then applied to the transport equation. For the time discretization, several schemes are presented.

In chapter 4 the issues of implementing the discretization derived in chapter 3 are discussed. The implementation is then applied to several test problems to check its correct operation.

Chapter 5 shows the application of the implementation to an artificial porous medium, confirming the mobile-immobile model. Similar real-world experiments are discussed.

Finally, in chapter 6 the accomplishments are summarized and future applications and improvements are discussed.

# 2 Transport in Porous Media

Transport in porous media describes the displacement of a substance, the *solute*, which is dissolved in another substance, the *solvent*. In most cases the solvent is water. The concentration $C$ of the solute may be given as the amount of the solute per amount of the solvent, or as the amount of the solute per total amount of solute and solvent. The amount of solute and solvent may be measured either as mass (kg), substance (mol) or volume ($\mathrm{m}^3$). Solute and solvent may even be measured in different units. Since we consider only incompressible flow, it does not matter which of those units we take, they are all proportional. Furthermore, only small concentrations are considered, as non-linear effects might occur if the amount of the solute approaches that of the solvent. Therefore it does not matter whether we use the total amount or only the amount of the solvent in the denominator. Thus we will usually give $C$ in some arbitrary units. The flux $\mathbf{j}$ denotes concentration per area and time.

## 2.1 Fundamental Transport Mechanisms

On the pore scale, transport is described by two processes: convection, and molecular diffusion.

### Convection

Convection is the movement of solute due to the flow of the solvent. Its contribution to the flow is

$$\mathbf{j}_c = \mathbf{u}C, \tag{2.1}$$

where $\mathbf{u}$ is the velocity of the solvent.

### Molecular Diffusion

Molecular diffusion occurs due to the thermodynamic motion of molecules and small particles (Brownian motion). In the thermodynamic limit it leads to a flux following the gradient of the concentration

$$\mathbf{j}_m = -D_m \nabla C, \tag{2.2}$$

where $D_m$ is the coefficient of molecular diffusion. This is known as *Fick's flux law* or *Fick's first law*.

$D_m$ is the diffusion coefficient in an unbounded space. In porous media however there are lots of boundaries which the solute cannot easily pass. Thus at the continuum scale the effective diffusion coefficient $D_{\mathrm{eff}}^{\mathrm{diff}}$ is lower (see Roth (2007), section 6.1.3 for details).

## 2.2 Derived Transport Mechanisms

### Representative Elementary Volume

The representative elementary volume is used for the transition from the pore scale to the continuum scale. To show its use we define the indicator function for the pores

$$\chi_p(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is in a pore,} \\ 0 & \text{if } \mathbf{x} \text{ is in the solid phase.} \end{cases} \tag{2.3}$$

On the continuum scale a description of each individual pore is not needed. Still, one needs to know how much of the overall volume is pore volume. This is described by the macroscopic parameter porosity $\Phi$, which can be obtained from the indicator function by averaging using a suitable weight function $\kappa(\mathbf{x})$ with support $\Omega_\kappa$:

$$\Phi(\mathbf{x}) := \langle\, \chi_p \,\rangle(\mathbf{x}) := \int_{\Omega_\kappa} \chi_p(\mathbf{x}' - \mathbf{x})\kappa(\mathbf{x}')d\mathbf{x}' \tag{2.4}$$

Naturally it is required that $\kappa(\mathbf{x}) \geq 0\ \forall \mathbf{x}$ and $\int_{\Omega_\kappa} \kappa(\mathbf{x})d\mathbf{x} = 1$.

The macroscopic version $\alpha_m$ of some other microscopic property $\alpha_\mu$ can be obtained using

$$\alpha_m(\mathbf{x}) := \langle\, \alpha_\mu \chi_p \,\rangle(\mathbf{x}) := \int_{\Omega_\kappa} \alpha_\mu(\mathbf{x}' - \mathbf{x})\chi_p(\mathbf{x}' - \mathbf{x})\kappa(\mathbf{x}')d\mathbf{x}'. \tag{2.5}$$

One choices for $\kappa$ include a constant within a sphere of radius $\sigma$ centered at the origin and zero everywhere else. For very small $\sigma$ the spatial distribution of the porosity $\Phi$ will vary wildly between 0 and 1. If we enlarge $\sigma$, the variation will decrease. Eventually, all remaining variations in the distribution of $\Phi$ can be attributed to inhomogeneities at the macro scale and $\Omega_\kappa$ is large enough to contain a representative sample of the micro scale. This is called the representative elementary volume or REV.

Other choices for $\kappa$ are possible. For further details on the REV see Roth (2007).

### Dispersion

For the transport mechanisms this means that on the macro scale pore scale variations cancel out and an averaged flux field remains. However these variations still affect the macro scale. Together with molecular diffusion they lead to new derived transport mechanisms, collectively referred to as dispersion.

Dispersion is similar to molecular diffusion. Diffusion is described by the coefficient of molecular diffusion $D_m$, whereas dispersion is described by $D(\mathbf{u})$, an anisotropic tensor which depends on the velocity. The contribution to the flux is

$$\mathbf{j}_{\text{disp}} = -D(\mathbf{u})\nabla C \tag{2.6}$$

and includes the contribution of molecular diffusion.

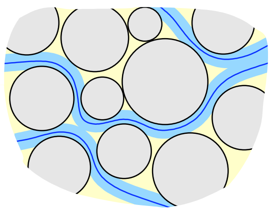


**Figure 2.1:** Sketch of immobile water in saturated, uniform, granular media. Water flow will occur along a network of flow paths (dark blue lines). The region within $l_\mathrm{m} = D_m/v$ of a flow path may be considered as mobile (light blue) whereas those beyond act as immobile (light yellow). Figure taken from Roth (2007).

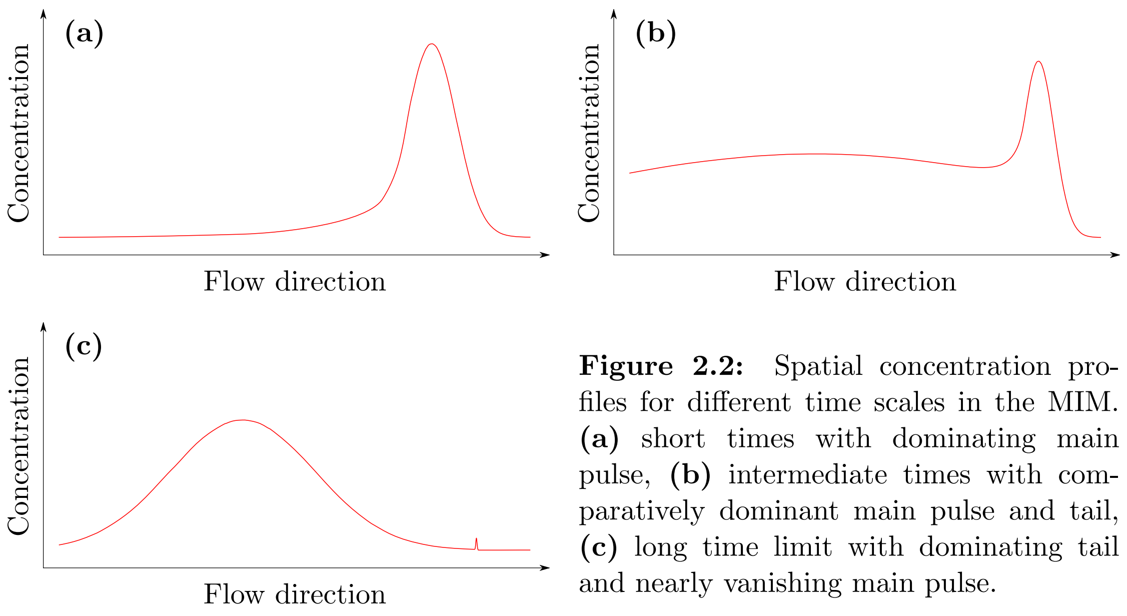


**Figure 2.2:** Spatial concentration profiles for different time scales in the MIM. **(a)** short times with dominating main pulse, **(b)** intermediate times with comparatively dominant main pulse and tail, **(c)** long time limit with dominating tail and nearly vanishing main pulse.

### 2.2.1 The Mobile-Immobile Model

The mobile-immobile model (MIM, Coats and Smith (1964); van Genuchten and Wierenga (1976), cited by Roth (2007)) is a simple model which nevertheless roughly explains tailing. It assumes that the solvent volume can be separated in a mobile part with larger velocities and an immobile part with no velocity. This could be for example porous pebbles in a sand matrix where the water in the sand is mobile while the water in the pebbles is immobile. But if we look at any porous medium at the pore scale we find preferred flow channels and regions with almost no flow as well (figure 2.1). The water in the flow channels forms the mobile phase whereas the water in the regions without flow forms the immobile phase.

MIM assumes that the convective transport is happening in the mobile region, and that the immobile region does not contribute at all. Exchange between the two regions happens mostly by diffusion. The density of each region is assumed to be macroscopically uniform.

Now imagine a solute pulse passing through such a medium. Most of the concentration will stay in the mobile region and just move undisturbed. If the initial pulse was Gaussian or if dispersion is strong enough to make it Gaussian, we will see it as a Gaussian shaped pulse in concentration profile. Some of the solute will diffuse into the immobile region however and stay there for some time, until diffusion moves it back into the mobile region, where it will move with the flow again. In the concentration profile, this will show as a long tail to the main pulse (figure 2.2a).

For longer time scales, the main pulse in the concentration profile will get smaller, while the tail will get bigger and will eventually evolve into a Gaussian in its own right (figure 2.2b). In the limit the main pulse has vanished completely, and most of the solute is concentrated in the immobile region (figure 2.2c).

## 2.3 The Transport Equation

Transport is described by a linear partial differential equation (PDE). We will refer to it as the "transport equation," but it is often also called convection-diffusion equation, or convection-diffusion-reaction equation if it includes reaction. In the simplest case the transport equation is

$$\partial_t C + \nabla\{\mathbf{u}C - D_m \nabla C\} = 0. \tag{2.7}$$

$\partial_t C$ is the time development term, $\nabla(\mathbf{u}C)$ the convection term and $-D_m \Delta C$ the diffusion term.

While (2.7) describes convection and diffusion to simulate on the pore scale we will start from the transport equation for a porous medium on the continuum scale, where additional processes must be taken into account:

$$R\Phi(\partial_t C + \lambda C) + \nabla \mathbf{j} = q(C) \qquad \text{in } \Omega \tag{2.8a}$$

$$\mathbf{j} = \mathbf{u}C - D(\mathbf{u})\nabla C \tag{2.8b}$$

$R$ is the retardation factor. It describes a slowdown of the solute compared to the solvent because of adsorption on the pore surface. $\Phi$ is the porosity as explained in (2.4). $R\Phi\lambda C$ is the decay term with $\lambda$ being the decay or reaction rate of the solute. It describes the decay of the solute due to radioactivity or chemical reactions, if the products have no further influence. $D(\mathbf{u})$ describes the dispersion. Unlike the scalar $D_m$, $D(\mathbf{u})$ is an anisotropic tensor depending on the velocity, thus the dispersion term is $-\nabla\{D(\mathbf{u})\nabla C\}$. $q(C)$ describes constant sources and sinks of concentration, and non-linear reactions (linear reactions are described by the decay term).

Equation (2.7) is a special case of (2.8). In free space, the domain is homogeneously filled with solvent, i.e. porosity $\Phi = 1$. Furthermore adsorption can only take place at the domain boundary, so the retardation factor is $R = 1$ within the whole domain. Due to the lack of pores dispersion does not take place and only molecular diffusion is left, which leads to $D(\mathbf{u}) = D_m$. If we further assume no decay or other reaction and no sources or sinks, we get $\lambda = 0$ which make the decay term vanish and $q(C) = 0$, so the right hand side vanishes as well. Using all these simplifications makes (2.8) equivalent to (2.7).

Although (2.7) is sufficient for our application, using (2.8) makes it easier to expand on this work later.[1] We will not fully implement (2.8) however, but we will delay the

[1] An example is the simulation of pebbles in a sand matrix at a scale where the pebbles a resolved, but the sand is viewed as a continuous porous medium.

application of simplification as long as possible (see 4.1.3).

The boundary conditions which may be applied are Dirichlet, Neumann (or flux) and outflow:

$$C = C_0 \qquad \text{on } \Gamma_C \text{ (Dirichlet BC)} \tag{2.9a}$$

$$\mathbf{jn} = J \qquad \text{on } \Gamma_J \text{ (Neumann BC)} \tag{2.9b}$$

$$\mathbf{jn} = (\mathbf{u}C - D(\mathbf{u})\nabla C)\mathbf{n} \qquad \text{on } \Gamma_O \text{ (outflow BC)} \tag{2.9c}$$

$\mathbf{n}$ denotes the outer normal to $\Omega$. $\Gamma_C$, $\Gamma_J$ and $\Gamma_O$ are the Dirichlet, Neumann and outflow boundaries, respectively. The special case $J = 0$ of the Neumann boundary condition is known as "noflux." On the outflow boundary it is required that

$$\mathbf{u} \cdot \mathbf{n} > 0 \qquad \text{on } \Gamma_O. \tag{2.10}$$

When using computed velocities it may happen that for small parts of the boundary intended for the outflow BC this condition is not fulfilled. For those parts the outflow BC can be replaced by a noflux BC, which will have the intended effect.

# 3 Numerical Treatment

Simulation of physical processes is a vast area. Many methods exist to simulate linear partial differential equations (PDE), and the applicable methods depend very much on the problem at hand.

**Definition 3.1** (Linear PDE)**:**
The general form of an $n$-th order linear partial differential equation is

$$\sum_{|\alpha|\le n} a_\alpha(\mathbf{x})\partial_\alpha u(\mathbf{x}) = f(\mathbf{x}) \qquad \forall \mathbf{x} \in \Omega \subset \mathbb{R}^d, \alpha \in \mathbb{N}^d. \tag{3.1}$$

It is linear in the unknown function $u(\mathbf{x})$. The order of the highest derivative determines the order of the PDE.
The transport equation is a second order linear PDE in general, and a first order linear PDE in the case without diffusion or dispersion.

### Classification of second order PDEs

The general form of a second order linear PDE is (see Hackbusch (2005))

$$\sum_{i,j=1}^{d} a_{ij}(\mathbf{x})\partial_i\partial_j u(\mathbf{x}) + \sum_{i=1}^{d} b_i(\mathbf{x})\partial_i u(\mathbf{x}) + c(\mathbf{x})u(\mathbf{x}) = f(\mathbf{x}). \tag{3.2}$$

We will write $\mathbf{b} = (b_i)$ and $A = (a_{ij})$. Since $\partial_i$ and $\partial_j$ may be exchanged, we require that $A$ be symmetric.

**Definition 3.2** (Elliptic second order PDE)**:**
We call (3.2) *elliptic* if all eigenvalues of $A$ have the same sign.

**Definition 3.3** (Hyperbolic second order PDE)**:**
We call (3.2) *hyperbolic* if one eigenvalue of $A$ has the opposite sign of all other eigenvalues.

**Definition 3.4** (Parabolic second order PDE)**:**
We call (3.2) *hyperbolic* if one eigenvalue of $A$ vanishes and $\operatorname{rank}(A, \mathbf{b}) = d$.
There are in principle more types of second order linear PDEs, but those are far less common.

### Classification of first order PDEs

First order linear PDEs are always *hyperbolic*. We get other types only with systems of first order linear PDEs.

The type of a linear PDE is very important since different types require very different ways of solving. Also, the initial and boundary conditions which can be specified are different for each type.

Our transport equation is a *linear second order parabolic PDE*, as long as $D \neq 0$. For $D = 0$ it is a *linear first order hyperbolic PDE*.

### Order of Convergence

The order of convergence is a measure of how much a scheme benefits from a finer grained discretization. To define it we need $h$, the typical size of the elements for space discretization schemes, or the typical size of the time step for time discretization schemes. We also need an error estimate

$$e(h) = \|u_h - u\|, \tag{3.3}$$

where $\|\cdot\|$ is commonly the $L^2$ or $H^1$ norm.

**Definition 3.5** (Order of Convergence)**:**
A methods order of convergence is $n$ (equivalently, the method is $n$-th order accurate), if and only if

$$e(h) = O(h^n) \qquad \text{for } h \to 0. \tag{3.4}$$

Different norms will result in different orders of convergence. We will use the $L^2$ norm exclusively.

Obviously one wants to use a scheme with a high order of convergence, unless other reasons prohibit it, since that allows to use coarser grids and longer time steps and thus may consume less memory and fewer CPU cycles. When combining several methods (like space and time discretization), the method with the lowest convergence order determines the overall convergence order.

Often one wants to know whether a program actually yields the expected order of convergence, since a lower than expected order of convergence is a good indicator that there are bugs left in the code. One can run the program twice, once with a coarse $h_1$ and once with a fine $h_2$ and compare the resulting errors $e(h_1)$ and $e(h_2)$.

**Definition 3.6** (Experimental Order of Convergence)**:**
We define the experimental order of convergence to be

$$n_{\text{exp}} := \frac{\log\left(\dfrac{e(h_1)}{e(h_2)}\right)}{\log\left(\dfrac{h_1}{h_2}\right)}. \tag{3.5}$$

Of course one will usually run the program not twice, but many times for successively smaller $h$, since $n_{\text{exp}}$ will approach $n$ only in the limit of $h \to 0$. If no exact solution $u$ is available, it is usually approximated by calculating a reference solution with a very small $h$.

## 3.1 Space Discretization

For the space discretization, a discontinuous Galerkin (DG) scheme is used. It is the foundation for the unfitted DG method developed by Engwer and Bastian (2005, 2008), which we will use to handle the complex geometry of the pore scale, see section 3.1.3.

### 3.1.1 Discontinuous Galerkin

DG is a finite elements method (FEM) with special ansatz functions which are non-zero only on one element and zero everywhere else. As the ansatz functions are discontinuous at the element boundaries, the resulting solution will be discontinuous as well. On the elements, the shape functions are usually polynomials. If the highest shape function order is $n$ then the order of convergence of the method is $n + 1$ for sufficiently regular solutions (see Bramble and Hilbert (1970); Brenner and Scott (2008)).

DG starts from the weak formulation of the PDE. It then obtains coupling between elements by transforming divergence integrals into integrals over the boundaries of the elements. Given a linear PDE

$$Lu = f \qquad \text{on } \Omega, \tag{3.6}$$

with the linear differential operator $L$ the weak formulation is obtained by multiplying with a test function $v$, integrating over $\Omega$ and stating that this new equation must be fulfilled for each test function $v \in V$:

$$\int_\Omega Lu \cdot v \, dx = \int_\Omega f \cdot v \, dx \qquad \forall v \in V. \tag{3.7}$$

$V$ is the space of test functions. If $\{v_i\}$ is a base[1] of $V$ with respect to the scalar product

$$(a, b) := \int_\Omega a \cdot b \, dx \qquad a, b \in V, \tag{3.8}$$

then (3.7) is equivalent to

$$\int_\Omega Lu \cdot v \, dx = \int_\Omega f \cdot v \, dx \qquad \forall v \in \{v_i\}. \tag{3.9}$$

Then the divergence terms in $L$ are transformed into integrals over the elements' border using integration by parts and Gauss' theorem. This is shown in detail later in section 3.1.2 for the application to the transport equation. This term now lives on the

[1] not necessarily an orthogonal base

intersections and will give us the coupling between elements, together with some other terms.

We use a DG method with a good order of convergence and local mass conservation. It is described in Bastian (2003). It includes penalty terms introduced by Oden, Babuška and Baumann (Oden *et al.* (1998)) and the interior penalty according to Wheeler (1978) and Rivière (2000), which were originally developed for elliptic PDEs. These artificial penalty terms make sure that the jumps in the concentration between elements stay small while still maintaining a good approximation of the solution. While the interior penalty is implemented in the code and thus presented here, it is not actually used in any calculations, since it has adjustable parameters which are hard to get right.

To guarantee positive concentration in all elements we apply upwinding. Consider a flow over an interface from an element $e$ with concentration 0 into an element $f$ with concentration 1. If we would take $\langle \mathbf{u}C \rangle \cdot \mathbf{n}_e$ as the flux, we would take concentration out of an empty element which would result in a negative concentration. But this is just the most prominent example, it generally makes more sense to let the upwind element control the flux coming out of it. So we will use the upwind flux $\langle \mathbf{u} \rangle C^* \cdot \mathbf{n}_e$, where $C^*$ is the upwind concentration.

Partial integration and Gauss' theorem, the penalty terms and upwinding together yield a new equation which may be written as

$$a(u,v) = l(v) \qquad \forall v \in V, \tag{3.10}$$

with $a(\,\cdot\,,\,\cdot\,)$ a bilinear form and $l(\,\cdot\,)$ a linear form. Now a concrete function space must be chosen for $V$ with a finite number $m$ of base functions $\{\phi_\mu\}$. Since any function $v \in V$ can be represented by a linear combination of the base functions $\{\phi_\mu\}$, (3.10) can be written as $m$ equations

$$a(u,\phi_\mu) = l(\phi_\mu) \qquad \forall \mu \in \mathbb{N} : 0 \leq \mu < n. \tag{3.11}$$

As customary for Galerkin methods the solution $u$ is discretized by using the same base $\{\phi_\nu\}$ with $0 \leq \nu < n$ to represent $u$:

$$u \approx \sum_{0 \leq \nu < n} u_\nu \phi_\nu. \tag{3.12}$$

Inserting this into (3.11) and moving the coefficients $u_\nu$ in front of the bilinear form $a$ one gets

$$\sum_{0 \leq \nu < n} u_\nu a(\phi_\nu,\phi_\mu) = l(\phi_\mu) \qquad \forall \mu \in \mathbb{N} : 0 \leq \mu < n. \tag{3.13}$$

With the matrix $(A)_{\mu\nu} := a(\phi_\nu,\phi_\mu)$ and the two vectors $(x)_\nu := u_\nu$ and $(b)_\mu := l(\phi_\mu)$, this is just a system of linear equations

$$Ax = b. \tag{3.14}$$

### 3.1.2 Discontinuous Galerkin Method for the Transport Equation

We will now discuss a concrete example and apply DG to the transport equation. In the following the concentration $C$ is the unknown function, which was previously known as $u$. The right hand side $f$ describes the source/sink/reaction term $q$.

The transport equation (2.8) can be written as

$$\underbrace{R\Phi\partial_t C}_{\text{Term A}} + \underbrace{R\Phi\lambda C}_{\text{Term B}} + \underbrace{\nabla\mathbf{j}}_{\text{Term C}} = \underbrace{q(C)}_{\text{Term D}} . \tag{3.15}$$

The time derivative in term A does not really fit into the scheme of (3.6). We shall handle it separately in section 3.2. Terms B and C correspond to the left hand side $Lu$ of (3.6), while term D corresponds to the right hand side $f$. Term C is the divergence term and will require most of our attention.

Before we start, we need some definitions which we will use in the following.

**Definition 3.7** (Partition of $\Omega$)**:**
We partition the domain $\Omega$ into disjoint elements and call this subdivision $E_h = \{e_1, \ldots, e_{n_h}\}$. We call the sub-domain of element $e \in E_h$ $\Omega_e$ and the outward unit normal to $e$ $\mathbf{n}_e$.

**Definition 3.8** (Internal and External Skeleton)**:**
We call

$$\Gamma_{\text{int}} := \{\gamma_{ef} = \partial\Omega_e \cap \partial\Omega_f | e, f \in E_h \wedge \gamma_{ef} \neq \emptyset\} \tag{3.16}$$

the internal skeleton and

$$\Gamma_{\text{ext}} := \{\gamma_e = \partial\Omega_e \cap \partial\Omega | e \in E_h \wedge \gamma_e \neq \emptyset\} \tag{3.17}$$

the external skeleton.

**Definition 3.9** (Jump and Average)**:**
Since there may be discontinuities at element boundaries we sometimes need to make clear which value of a discontinuous function to take. We call the continuous extension of $g$ from $\Omega_e$ to $\partial\Omega_e$ $g|_e$:

$$g|_e(\mathbf{x}) := \lim_{\mathbf{x}' \to \mathbf{x}} g(\mathbf{x}') \qquad \forall \mathbf{x}' \in \Omega_e. \tag{3.18}$$

We call $[\, g\,]_{ef} := g|_e - g|_f$ the jump of $g$ on $\gamma_{ef}$ and $\langle\, g\,\rangle_{ef} := \frac{1}{2}(g|_e - g|_f)$ the average. Since $\langle\, g\,\rangle_{ef} = \langle\, g\,\rangle_{fe}$ we will omit the index to the average most of the time. We note the property

$$[\, gh\,]_{ef} = [\, g\,]_{ef}\langle\, h\,\rangle + \langle\, g\,\rangle[\, h\,]_{ef}. \tag{3.19}$$

#### Weak Formulation

The weak formulation of the problem (2.8) is

$$\int_\Omega R\Phi\partial_t C \cdot v\, dx + \int_\Omega R\Phi\lambda C \cdot v\, dx + \int_\Omega \nabla\mathbf{j} \cdot v\, dx = \int_\Omega q(C) \cdot v\, dx \qquad \forall v \in V. \tag{3.20}$$

### Boundary Integral

First (3.20) is expressed as a sum over all elements:

$$\sum_{e\in E_h}\int_{\Omega_e} R\Phi\partial_t C\cdot v\,dx+\sum_{e\in E_h}\int_{\Omega_e} R\Phi\lambda C\cdot v\,dx+\sum_{e\in E_h}\int_{\Omega_e}\nabla\mathbf{j}\cdot v\,dx \\ =\sum_{e\in E_h}\int_{\Omega_e} q(C)\cdot v\,dx\qquad\forall v\in V. \tag{3.21}$$

Now the integral over the divergence is transformed into an integral over the elements' borders:

$$\begin{aligned}\sum_{e\in E_h}\int_{\Omega_e}\nabla\mathbf{j}\cdot v\,dx &=\sum_{e\in E_h}\left\{-\int_{\Omega_e}\mathbf{j}\cdot\nabla v\,dx+\int_{\Omega_e}\nabla(\mathbf{j}v)\,dx\right\} && \text{(integration by parts)}\\ &=\sum_{e\in E_h}\left\{-\int_{\Omega_e}\mathbf{j}\cdot\nabla v\,dx+\int_{\partial\Omega_e}v\mathbf{j}\cdot\mathbf{n}_e\,ds\right\} && \text{(Gauss' theorem)}\\ &=-\sum_{e\in E_h}\int_{\Omega_e}\mathbf{j}\cdot\nabla v\,dx+\sum_{\gamma_{ef}\in\Gamma_{\text{int}}}\int_{\gamma_{ef}}[\,v\mathbf{j}\,]_{ef}\cdot\mathbf{n}_e\,ds\\ &\quad+\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e}v\mathbf{j}\cdot\mathbf{n}_e\,ds.\end{aligned} \tag{3.22}$$

We use the property (3.19) and demand conservation of mass $[\,\mathbf{j}\,]=0$ so the sum over the internal skeleton becomes

$$\sum_{\gamma_{ef}\in\Gamma_{\text{int}}}\int_{\gamma_{ef}}[\,v\mathbf{j}\,]_{ef}\cdot\mathbf{n}_e\,ds=\sum_{\gamma_{ef}\in\Gamma_{\text{int}}}\int_{\gamma_{ef}}[\,v\,]_{ef}\langle\,\mathbf{j}\,\rangle\cdot\mathbf{n}_e\,ds. \tag{3.23}$$

We split the sum over the external skeleton into three sums over the external skeleton restricted to the Dirichlet, the Neumann and the outflow boundaries respectively and insert the Neumann and the outflow boundary conditions (2.9b, 2.9c):

$$\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e}v\mathbf{j}\cdot\mathbf{n}_e\,ds=\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_C}v\mathbf{j}\cdot\mathbf{n}_e\,ds+\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_J}vJ\,ds \\ +\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_O}v(\mathbf{u}C-D\nabla C)\cdot\mathbf{n}_e\,ds. \tag{3.24}$$

Inserting the flow (2.8b) we now get for all of term C

$$\sum_{e\in E_h}\int_{\Omega_e}\nabla\mathbf{j}\cdot v\,dx = -\sum_{e\in E_h}\int_{\Omega_e}(\mathbf{u}C - D\nabla C)\cdot\nabla v\,dx \tag{3.25a}$$

$$+\sum_{\gamma_{ef}\in\Gamma_{\text{int}}}\int_{\gamma_{ef}}[\,v\,]_{ef}\langle\,\mathbf{u}C - D\nabla C\,\rangle\cdot\mathbf{n}_e\,ds \tag{3.25b}$$

$$+\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_C} v(\mathbf{u}C - D\nabla C)\cdot\mathbf{n}_e\,ds \tag{3.25c}$$

$$+\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_J} vJ\,ds \tag{3.25d}$$

$$+\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_O} v(\mathbf{u}C - D\nabla C)\cdot\mathbf{n}_e\,ds. \tag{3.25e}$$

### Penalty Terms and Upwinding

We now need to insert the OBB penalty term

$$\sum_{\gamma_{ef}\in\Gamma_{\text{int}}}\int_{\gamma_{ef}}[\,C\,]_{ef}\langle\,D\nabla v\,\rangle\cdot\mathbf{n}_e\,ds. \tag{3.26}$$

This term is symmetric to the right part of (3.25b) and it tends toward zero with the size of the elements, since then $[\,C\,]$ tends toward zero. This term penalizes large jumps at the element boundaries.

The interior penalty on the internal skeleton has the form

$$\sum_{\gamma_{ef}\in\Gamma_{\text{int}}}\frac{\sigma}{|\gamma_{ef}|^{\beta}}\int_{\gamma_{ef}}[\,C\,]_{ef}[\,v\,]_{ef}\,ds, \tag{3.27}$$

for suitable parameters $\sigma$ and $\beta$. As this term is implemented in the code, it is also discussed here. However, it is not used in the results presented in sections 4.2 and 5 since it does not necessarily tend toward zero with the element size tending toward zero.

Now we apply upwinding to (3.25b), this means we replace $C$ in the left part with the upwind concentration

$$C^*(\mathbf{x}) := \begin{cases} C|_e(\mathbf{x}) & \text{if } \langle\,\mathbf{u}(\mathbf{x})\,\rangle\cdot\mathbf{n}_e(\mathbf{x}) > 0 \\ C|_f(\mathbf{x}) & \text{else.}\end{cases} \tag{3.28}$$

With OBB penalty and upwinding (3.25b) now becomes

$$
\begin{aligned}
\sum_{\gamma_{ef}\in\Gamma_{\text{int}}} \int_{\gamma_{ef}} [\, v\,]_{ef} \langle\, \mathbf{u}C - D\nabla C\,\rangle \cdot \mathbf{n}_e \, ds \to \\
&\sum_{\gamma_{ef}\in\Gamma_{\text{int}}} \int_{\gamma_{ef}} [\, v\,]_{ef} C^* \langle\, \mathbf{u}\,\rangle \cdot \mathbf{n}_e \, ds \\
&+ \sum_{\gamma_{ef}\in\Gamma_{\text{int}}} \frac{\sigma}{|\gamma_{ef}|^\beta} \int_{\gamma_{ef}} [\, C\,]_{ef} [\, v\,]_{ef} \, ds \\
&+ \sum_{\gamma_{ef}\in\Gamma_{\text{int}}} \int_{\gamma_{ef}} \big\{ [\, C\,]_{ef} \langle\, D\nabla v\,\rangle \cdot \mathbf{n}_e - [\, v\,]_{ef} \langle\, D\nabla C\,\rangle \cdot \mathbf{n}_e \big\} \, ds.
\end{aligned}
\tag{3.29}
$$

**Dirichlet Boundary Conditions**

Similarly to the OBB penalty, we impose the Dirichlet boundary conditions (2.9a) weakly on (3.25c). This adds the term

$$\sum_{\gamma_e\in\Gamma_{\text{ext}}} \int_{\gamma_e\cap\Gamma_C} (C - C_0) D\nabla v \cdot \mathbf{n}_e \, ds. \tag{3.30}$$

There is also a term coming from the interior penalty:

$$\sum_{\gamma_{ef}\in\Gamma_{\text{ext}}} \frac{\sigma}{|\gamma_{ef}|^\beta} \int_{\gamma_{ef}\cap\Gamma_C} (C - C_0) v \, ds. \tag{3.31}$$

Please note that the test function $v$ evaluated outside the whole domain is zero, so this term is very similar to its counterpart on the internal skeleton.

Again upwinding is applied. On the boundary that means choose either the element's concentration $C$ or the Dirichlet boundary value $C_0$ depending on whether the velocity $\mathbf{u}$ points into or out of the domain, respectively. Since $C_0$ is known and $C$ is unknown it is not convenient to define the upwind concentration $C^*$ for the boundary. Instead we shall define the Dirichlet inflow and outflow boundaries.

**Definition 3.10** (Dirichlet Inflow and Outflow Boundary)**:**
We divide the Dirichlet boundary $\Gamma_C$ into the Dirichlet outflow Boundary

$$\Gamma_C^{\text{out}} := \{\mathbf{x} \in \Gamma_C | \mathbf{u}(\mathbf{x}) \cdot \mathbf{n} > 0\} \tag{3.32}$$

and the Dirichlet inflow Boundary

$$\Gamma_C^{\text{in}} := \{\mathbf{x} \in \Gamma_C | \mathbf{u}(\mathbf{x}) \cdot \mathbf{n} \leq 0\}, \tag{3.33}$$

where $\mathbf{n}$ is the unit normal vector pointing out of the domain.

Thus with upwinding we get

$$\begin{aligned}\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_C} v\mathbf{u}C\cdot\mathbf{n}_e\,ds \to \quad & \sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_C^{\text{out}}} v\mathbf{u}C\cdot\mathbf{n}_e\,ds \\ & + \sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_C^{\text{in}}} v\mathbf{u}C_0\cdot\mathbf{n}_e\,ds.\end{aligned} \tag{3.34}$$

Altogether the Dirichlet boundary term now becomes

$$\begin{aligned}\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_C} v(\mathbf{u}C - D\nabla C)\cdot\mathbf{n}_e\,ds \to & - \sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_C} vD\nabla C\cdot\mathbf{n}_e\,ds \\ & + \sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_C} (C-C_0)D\nabla v\cdot\mathbf{n}_e\,ds \\ & + \sum_{\gamma_e\in\Gamma_{\text{ext}}}\frac{\sigma}{|\gamma_e|^\beta}\int_{\gamma_e\cap\Gamma_C} (C-C_0)v\,ds \\ & + \sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_C^{\text{out}}} v\mathbf{u}C\cdot\mathbf{n}_e\,ds \\ & + \sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_C^{\text{in}}} v\mathbf{u}C_0\cdot\mathbf{n}_e\,ds.\end{aligned} \tag{3.35}$$

Putting it all together and moving the completely known term to the right hand side:

$$\sum_{e\in E_h}\int_{\Omega_e} R\Phi\partial_t C\cdot v\,dx \tag{3.36a}$$

$$+\sum_{e\in E_h}\int_{\Omega_e} R\Phi\lambda C\cdot v\,dx \tag{3.36b}$$

$$-\sum_{e\in E_h}\int_{\Omega_e} (\mathbf{u}C - D\nabla C)\cdot\nabla v\,dx \tag{3.36c}$$

$$+\sum_{\gamma_{ef}\in\Gamma_{\text{int}}}\int_{\gamma_{ef}} [\,v\,]_{ef} C^*\langle\,\mathbf{u}\,\rangle\cdot\mathbf{n}_e\,ds \tag{3.36d}$$

$$+\sum_{\gamma_{ef}\in\Gamma_{\text{int}}}\frac{\sigma}{|\gamma_{ef}|^\beta}\int_{\gamma_{ef}} [\,C\,]_{ef}[\,v\,]_{ef}\,ds \tag{3.36e}$$

$$+\sum_{\gamma_{ef}\in\Gamma_{\text{int}}}\int_{\gamma_{ef}} \big\{[\,C\,]_{ef}\langle\,D\nabla v\,\rangle\cdot\mathbf{n}_e - [\,v\,]_{ef}\langle\,D\nabla C\,\rangle\cdot\mathbf{n}_e\big\}\,ds \tag{3.36f}$$

$$+ \sum_{\gamma_e \in \Gamma_{\text{ext}}} \int_{\gamma_e \cap \Gamma_C} \left\{ CD\nabla v \cdot \mathbf{n}_e - vD\nabla C \cdot \mathbf{n}_e \right\} ds \tag{3.36g}$$

$$+ \sum_{\gamma_e \in \Gamma_{\text{ext}}} \frac{\sigma}{|\gamma_e|^\beta} \int_{\gamma_e \cap \Gamma_C} Cv\, ds \tag{3.36h}$$

$$+ \sum_{\gamma_e \in \Gamma_{\text{ext}}} \int_{\gamma_e \cap \Gamma_C^{\text{out}}} v\mathbf{u}C \cdot \mathbf{n}_e\, ds \tag{3.36i}$$

$$+ \sum_{\gamma_e \in \Gamma_{\text{ext}}} \int_{\gamma_e \cap \Gamma_O} v(\mathbf{u}C - D\nabla C) \cdot \mathbf{n}_e\, ds \tag{3.36j}$$

$$= \sum_{e \in E_h} \int_{\Omega_e} q(C) \cdot v\, dx \tag{3.36k}$$

$$- \sum_{\gamma_e \in \Gamma_{\text{ext}}} \int_{\gamma_e \cap \Gamma_J} vJ\, ds \tag{3.36l}$$

$$- \sum_{\gamma_e \in \Gamma_{\text{ext}}} \int_{\gamma_e \cap \Gamma_C^{\text{in}}} v\mathbf{u}C_0 \cdot \mathbf{n}_e\, ds \tag{3.36m}$$

$$+ \sum_{\gamma_e \in \Gamma_{\text{ext}}} \int_{\gamma_e \cap \Gamma_C} C_0 D\nabla v \cdot \mathbf{n}_e\, ds \tag{3.36n}$$

$$+ \sum_{\gamma_e \in \Gamma_{\text{ext}}} \frac{\sigma}{|\gamma_e|^\beta} \int_{\gamma_e \cap \Gamma_C} C_0 v\, ds \qquad \forall v \in V. \tag{3.36o}$$

Note the similarity to (3.10). Even though we could match (3.10) and (3.36) it is better to give the time derivative term (3.36a) special treatment. We write (3.36) similar to (3.10):

$$m(\partial_t C, v) + a(C, v) = l(v) \qquad \forall v \in V. \tag{3.37}$$

$m(\,\cdot\,,\,\cdot\,)$ is another bilinear form denoting time derivative term (3.36a). $a(\,\cdot\,,\,\cdot\,)$ denotes the rest of the left hand side of (3.36), while $l(\,\cdot\,)$ denotes the right hand side. Using a base $\{\phi_\mu\}$ for $v$ and $\{\psi_\nu\}$ for $C$ we get

$$\sum_\nu \partial_t C_\nu \cdot m(\psi_\nu, \phi_\mu) + \sum_\nu C_\nu \cdot a(\psi_\nu, \phi_\mu) = l(\phi_\mu) \qquad \forall \mu. \tag{3.38}$$

To get to a system of linear equations, we define $(M)_{\mu\nu} := m(\psi_\nu, \phi_\mu)$ and $(x)_\nu := C_\nu$. $A$ and $b$ are defined as previously for (3.14):

$$M\partial_t x + Ax = b \tag{3.39}$$

We now require that $\phi_\mu = \psi_\nu$. This allows us to invert $M$. On a computer this may be very expensive in terms of CPU time and memory use. If $M$ is sparse with $n$ nonzero entries, $M^{-1}$ may still be dense with $O(n^2)$ entries. We have already used that all ansatz functions $\phi_\mu$ are nonzero only on one element $e$. This means that if we have one ansatz function $\phi_\mu$ which is nonzero on element $e$ and one ansatz function $\phi_\nu$ which is nonzero

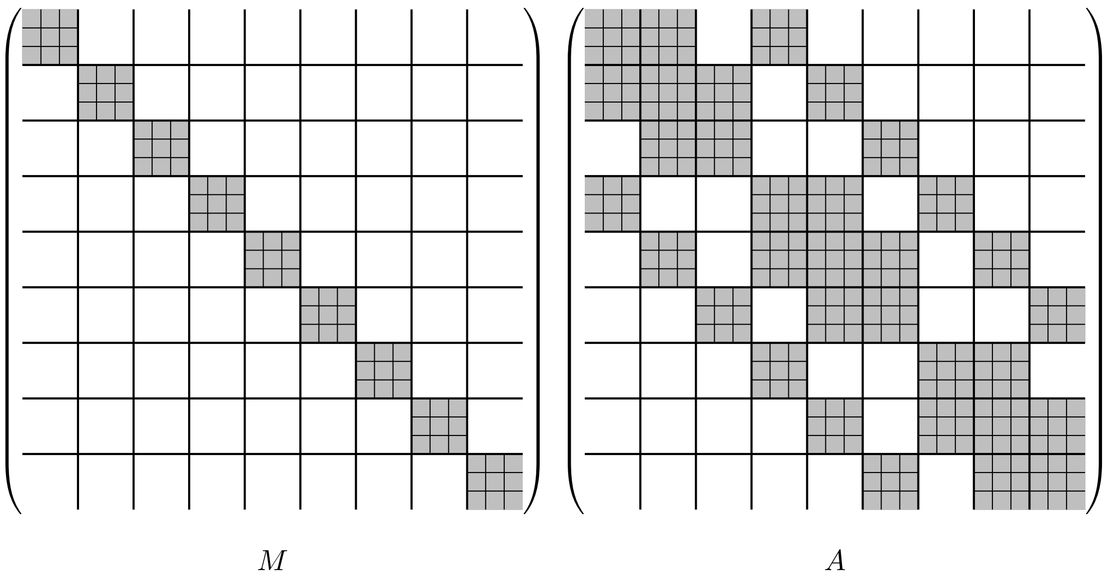


**Figure 3.1:** Block structure of the matrices $M$ and $A$. This example is for a domain partitioned into $3 \times 3$ elements with linear shape functions (constant, $x$ and $y$). The gray squares ▦ represent dense blocks in the matrix. Each hollow square □ represents one block with all entries zero. Each block corresponds to one combination of elements $(e,\ f)$. The dense blocks on the diagonal represent the interaction of one element with itself. Those not on the diagonal represent interactions with neighboring elements. Each sub-square within one dense block corresponds to a pair of shape functions $(i,\ j)$.

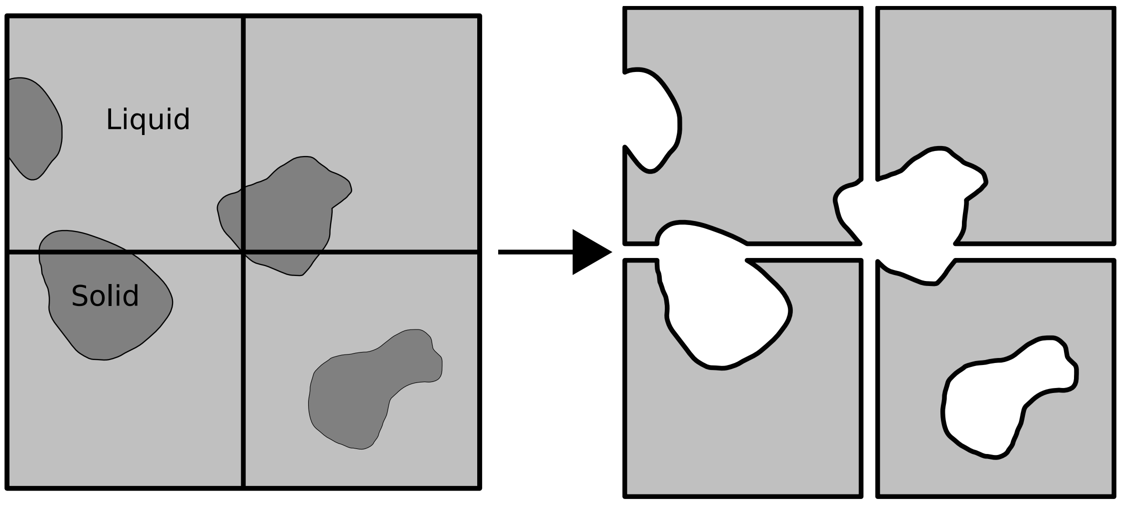


**Figure 3.2:** UDG constrains the support of the shape functions to the liquid phase.

on element $f$ and $m(\phi_\nu, \phi_\mu) \neq 0$ it follows that $e$ and $f$ are the same element. In other words, if we group the indices $\mu$ in such a way that all base functions $\phi_\mu$ which are nonzero on the same element $e$ get adjacent indices, then $M$ will be block-wise diagonal with each diagonal block corresponding to one element (figure 3.1). We can simply invert each block on the diagonal of $M$ to get $M^{-1}$. This does not require much CPU time as long as the number of shape functions per element are small, and $M^{-1}$ will have the same sparsity pattern as $M$. Thus the end result of this section is

$$\partial_t x + M^{-1}Ax = M^{-1}b. \tag{3.40}$$

We shall see how to deal with the time derivative in section 3.2.

### 3.1.3 Unfitted DG

Up to now we talked about a very general partitioning $E_h$ of our domain $\Omega$ into elements. In practice this partitioning is done by a grid, which may be structured or unstructured. Unstructured grids seem better suited for our purpose since unlike structured grids they can readily represent the pore structure. However the process of generating an unstructured grid which fits some structure is often complicated and may require a lot of fine tuning and manual post-processing.

Christian Engwer is working on a method called unfitted discontinuous Galerkin (UDG, Engwer and Bastian (2005, 2008)), which allows to use complex structures even with regular grids. It works by restricting the support of the shape functions to the geometry, that means to the liquid phase in our case. This may be used on top of more or less arbitrary grids, especially structured ones. When used with structured grids it will even result in smaller matrices compared to unstructured grids without UDG.

## 3.2 Time Discretization

Now we need to deal with the time derivative in (3.40). We keep only the time derivative on the left hand side and explicitly state time dependencies:

$$\frac{dx(t)}{dt} = M^{-1}b(t) - M^{-1}Ax(t) \tag{3.41}$$

The time derivative is now a total derivative since time is the only variable left.

The left hand side is usually discretized as

$$\frac{dx(t)}{dt} \to \frac{1}{\Delta t^n}(x^n - x^{n-1}), \tag{3.42}$$

with $x^n := x(t^n)$ the vector of unknowns after step $n$, $\Delta t^n := t^n - t^{n-1}$ the duration of step $n$ and $t^n$ the end time of step $n$. $x^0 := x(t^0)$ is the vector representing the initial conditions, and $t^0$ is the starting time of the simulation.

The most straightforward way to discretize the right hand side is

$$M^{-1}b(t) - M^{-1}Ax(t) \to M^{-1}b(t^{n-1}) - M^{-1}Ax^{n-1} \tag{3.43}$$

("take the old values"), which gives us after some reordering

$$x^n = \Delta t^n M^{-1}b(t^{n-1}) + \{1 - \Delta t^n M^{-1}A\}x^{n-1}. \tag{3.44}$$

This is known as *explicit Euler*. The new value of $x$ can be calculated explicitly, there is no need to solve a linear equation system. The drawback is that unlike the implicit methods presented later, this method is strongly restricted by the CFL condition. This is unfortunate, since in the UDG method the smalled grid element may be *much* smaller than the typical element.

### 3.2.1 Implicit Euler

For the *implicit Euler* the right hand side is discretized as

$$M^{-1}b(t) - M^{-1}Ax(t) \to M^{-1}b(t^n) - M^{-1}Ax^n \tag{3.45}$$

("take the new values"). The discretized equation is

$$\{1 + \Delta t^n M^{-1}A\}x^n = \Delta t^n M^{-1}b(t^n) + x^{n-1}. \tag{3.46}$$

This is a system of linear equations which needs to be solved in each time step. This method is first order accurate and since it is implicit it is not necessary to fulfill the CFL condition strictly. However, since our space discretization allows for a higher order of convergence with the appropriate shape functions this is still not optimal.

### 3.2.2 One Step $\theta$, Crank-Nicholson

One way to achieve a higher order accuracy is the *one step* $\theta$ scheme, see Bastian and Lang (2004). This is essentially a mix of the explicit and the implicit Euler schemes. The right hand side is discretized as:

$$\begin{aligned} M^{-1}b(t) - M^{-1}Ax(t) \to (1-\theta) \cdot \{M^{-1}b(t^{n-1}) - M^{-1}Ax^{n-1}\} \\ +\theta \cdot \{M^{-1}b(t^n) - M^{-1}Ax^n\}, \end{aligned} \tag{3.47}$$

and the full equation reads:

$$\begin{aligned} \{1 + \Delta t^n \theta M^{-1}A\}x^n = & \Delta t^n M^{-1}\{(1-\theta)b(t^{n-1}) + \theta b(t^n)\} \\ & + \{1 - \Delta t^n(1-\theta)M^{-1}A\}x^{n-1} \end{aligned} \tag{3.48}$$

For $\theta = 0$ this is the explicit Euler scheme, for $\theta = 1$ it is identical to the implicit Euler scheme.

The scheme with $\theta = 1/2$ is known as *Crank-Nicholson* and is second order accurate.

### 3.2.3 Fractional Step $\theta$

The *fractional step* $\theta$ scheme (see also Bastian and Lang (2004)) consists of three steps of the one step $\theta$ scheme. The computational cost per time step is three times the cost of one Crank-Nicholson step, but $\Delta t$ can be chosen three times as large as for Crank-Nicholson. $\theta$ and the step size for the sub steps are chosen in a special way:

$$\theta_1 = \sqrt{2} - 1, \qquad \Delta t_1 = \Delta t \cdot \theta_1, \tag{3.49a}$$

$$\theta_2 = 2 - \sqrt{2}, \qquad \Delta t_2 = \Delta t \cdot \theta_2/2, \tag{3.49b}$$

$$\theta_3 = 2 - \sqrt{2}, \qquad \Delta t_3 = \Delta t \cdot \theta_3/2. \tag{3.49c}$$

This scheme is second order accurate and has improved stability properties over Crank-Nicholson. However, our implementation has unsolved problems (see section 4.2.1), so we will not use it in production.

# 4 Implementation and Testing

## 4.1 Implementation Details

As we have seen in section 3 we need to build our matrices $M$ and $A$ and the right hand side vector $b$. This process is called *assembling*.

### 4.1.1 DUNE

The implementation uses the DUNE framework (Distributed and Unified Numerics Environment, see Bastian *et al.* (n.d.b) and Bastian *et al.* (n.d.a)). According to its homepage, "DUNE [...] is a modular toolbox for solving partial differential equations (PDEs) with grid-based methods." It currently offers a unified interface to several existing grid managers (UG, ALBERTA, ALUgrid, ...) and solvers (SuperLU, PARDISO, ...) as well as some DUNE-native grid managers (YaspGrid[1], OneDGrid, ...) and solvers (BiCGstab, CG, ...). DUNE is written in C++ and makes heavy use of templates and compile time polymorphism, which allows for good optimization by the compiler and especially reduces the cost at runtime for wrapping third party libraries, while still keeping the code generic. The uniform interface allows to quickly try different implementations of grids or solvers or other things.

An UDG implementation for DUNE is developed by C. Engwer, which is used for this work.

### 4.1.2 Numerical Quadrature

In the assembling stage we need to evaluate several integrals, which we will do by using Gaussian quadrature rules. Gaussian quadrature works by representing the integral as a weighted sum of the function evaluated at special *quadrature points* $x_i$:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{4.1}$$

If the integrand $f(x)$ is a polynomial of order $2n - 1$ or less then (4.1) becomes exact. Based on this one-dimensional rule, quadrature rules for higher dimensions and different element shapes may be constructed. These are all available in DUNE.

[1] **Y**et **a**nother **s**tructured **p**arallel **grid**

### 4.1.3 Simplifications

We now apply several simplifications:

**Pore scale only $R = \Phi = 1$:** Since we simulate on the pore scale, the retardation factor $R$ and the porosity $\Phi$ are both 1. These only matter at scales where the pores are not resolved.

**No decay $\lambda = 0$:** For our simple problems decay does not matter, so term (3.36b) vanishes.

**No sources $q(C) = 0$:** No concentration sources or sinks, no reactions which produce or consume solute. This makes term (3.36k) vanish.

**Stationary flow field $\partial_t \mathbf{u} = 0$:** This avoids the need to reassemble $A$ and $b$ in each time step.

**Time independence of $\Gamma_C$, $\Gamma_J$ and $\Gamma_O$:** As with the stationary flow field, this avoids the need to reassemble $A$ and $b$ in each time step.

**Time independent boundary values $\partial_t C_0 = \partial_t J = 0$:** This avoids the need to reassemble $b$ in each time step. Although this may seem very limiting, one can actually get quite far without time dependent boundary conditions, though sometimes at the expense of a bigger simulation domain.

**$\Gamma_C$, $\Gamma_J$ and $\Gamma_O$ match element boundaries:**

$$\gamma_e \cap \Gamma_J \neq \emptyset \Leftrightarrow \gamma_e \subseteq \Gamma_J \quad \forall \gamma_e \in \Gamma_{\text{ext}} \tag{4.2}$$

Similarly for $\Gamma_C$ and $\Gamma_O$. This just means that one boundary intersection $\gamma_e$ cannot contain parts of two different boundary condition types.

### 4.1.4 Assembling

Assembling is done by iterating over the grid. For each element, we first handle the integrals over $\Omega_E$ (the *volume term*), then the integrals over the element's intersections with other elements ($\Gamma_{\text{int}}$, the *face term*) and at last the integrals over the element's intersection with the domain boundary ($\Gamma_{\text{ext}}$, the *boundary term*). The integral over the internal intersections $\Gamma_{\text{int}}$ is done twice, once for each element on either side of the intersection.

I'll now give three examples of how to compute the integrals.

#### Volume Term

As an example we will use the term (3.36c):

$$-\sum_{e \in E_h} \int_{\Omega_e} (\mathbf{u}C - D\nabla C) \cdot \nabla v \, dx \tag{4.3}$$

We split the index $\mu$ of our shape functions $\{\phi_\mu\}$ into the elements index $e \in E_h$ and the shape functions index within the element $i$, using an appropriate mapping $m$:

$$\mu = m(e,i), \qquad \phi_\mu = \phi_i^e \tag{4.4}$$

A shape function has element index $e$ if and only if it is nonzero within element $e$. We required earlier that it is nonzero only in exactly one element, so this is unambiguous. Likewise, we split the indices of our matrices and our vector:

$$M_{\mu\nu} = M_{ij}^{ef} \qquad A_{\mu\nu} = A_{ij}^{ef} \qquad b_\mu = b_i^e \qquad \nu = m(f,j) \tag{4.5}$$

Now we can insert the base functions $\phi_i^e$ for $v$ and $\phi_j^e$ for $C$. This gives us the contribution of this term to the matrix element $A_{ij}^{ee}$:

$$-\sum_{e\in E_h} \int_{\Omega_e} (\mathbf{u}\phi_j^e - D\nabla\phi_j^e)\cdot\nabla\phi_i^e \, dx \tag{4.6}$$

## Face Term

We will use (3.36d) as an example:

$$+\sum_{\gamma_{ef}\in\Gamma_{\text{int}}} \int_{\gamma_{ef}} [\,v\,]_{ef} C^* \langle\,\mathbf{u}\,\rangle \cdot \mathbf{n}_e \, ds \tag{4.7}$$

This may be split into four smaller terms

$$\sum_{\gamma_{ef}\in\Gamma_{\text{int}}} \int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_e)\cdot v|_e C|_e \langle\,\mathbf{u}\,\rangle\mathbf{n}_e \, ds \tag{4.8a}$$

$$+\sum_{\gamma_{ef}\in\Gamma_{\text{int}}} \int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_e)\cdot v|_f C|_e \langle\,\mathbf{u}\,\rangle\mathbf{n}_f \, ds \tag{4.8b}$$

$$+\sum_{\gamma_{ef}\in\Gamma_{\text{int}}} \int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_f)\cdot v|_e C|_f \langle\,\mathbf{u}\,\rangle\mathbf{n}_e \, ds \tag{4.8c}$$

$$+\sum_{\gamma_{ef}\in\Gamma_{\text{int}}} \int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_f)\cdot v|_f C|_f \langle\,\mathbf{u}\,\rangle\mathbf{n}_f \, ds \tag{4.8d}$$

with $\Theta$ being the Heaviside step function

$$\Theta(x) = \begin{cases} 0 & x < 0, \\ \frac{1}{2} & x = 0, \\ 1 & x > 0. \end{cases} \tag{4.9}$$

For the matrix entry $A_{ij}^{ee}$ with the shape functions $\phi_i^e$ for $v$ and $\phi_j^e$ for $C$ only term (4.8a) contributes:

$$\begin{aligned}
&\int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_e)\cdot\phi_i^e|_e\,\phi_j^e|_e\langle\,\mathbf{u}\,\rangle\mathbf{n}_e\,ds \\
&+\int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_e)\cdot\underbrace{\phi_i^e|_f}_{0}\,\phi_j^e|_e\langle\,\mathbf{u}\,\rangle\mathbf{n}_f\,ds \\
&+\int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_f)\cdot\phi_i^e|_e\,\underbrace{\phi_j^e|_f}_{0}\langle\,\mathbf{u}\,\rangle\mathbf{n}_e\,ds \\
&+\int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_f)\cdot\underbrace{\phi_i^e|_f}_{0}\,\underbrace{\phi_j^e|_f}_{0}\langle\,\mathbf{u}\,\rangle\mathbf{n}_f\,ds \\
=\;&\int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_e)\cdot\phi_i^e|_e\,\phi_j^e|_e\langle\,\mathbf{u}\,\rangle\mathbf{n}_e\,ds
\end{aligned}\tag{4.10}$$

Likewise, for the matrix entry $A_{ij}^{fe}$ with the shape functions $\phi_i^e$ for $v$ and $\phi_j^f$ for $C$ the only contribution comes from term (4.8c):

$$\begin{aligned}
&\int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_e)\cdot\phi_i^e|_e\,\underbrace{\phi_j^f|_e}_{0}\langle\,\mathbf{u}\,\rangle\mathbf{n}_e\,ds \\
&+\int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_e)\cdot\underbrace{\phi_i^e|_f}_{0}\,\underbrace{\phi_j^f|_e}_{0}\langle\,\mathbf{u}\,\rangle\mathbf{n}_f\,ds \\
&+\int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_f)\cdot\phi_i^e|_e\,\phi_j^f|_f\langle\,\mathbf{u}\,\rangle\mathbf{n}_e\,ds \\
&+\int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_f)\cdot\underbrace{\phi_i^e|_f}_{0}\,\phi_j^f|_f\langle\,\mathbf{u}\,\rangle\mathbf{n}_f\,ds \\
=\;&\int_{\gamma_{ef}} \Theta(\langle\,\mathbf{u}\,\rangle\mathbf{n}_f)\cdot\phi_i^e|_e\,\phi_j^f|_f\langle\,\mathbf{u}\,\rangle\mathbf{n}_e\,ds
\end{aligned}\tag{4.11}$$

Term (4.8b) transforms into (4.8c) and term (4.8d) transforms into (4.8a) if we exchange $e$ and $f$. Since each internal intersection is visited twice, the second time with $e$ and $f$ exchanged, it is enough to only implement (4.8a) and (4.8c).

The Heaviside step function $\Theta$ presents a problem since it is not polynomial and thus cannot be integrated exactly using Gaussian quadrature. We have to do another approximation: we use the integrand without $\Theta$ to determine the quadrature order, and simply don't evaluate the integrand if $\Theta = 0$.

### Boundary Term

Here we will use term (3.36l) as example:

$$-\sum_{\gamma_e\in\Gamma_{\text{ext}}}\int_{\gamma_e\cap\Gamma_J} vJ\,ds \tag{4.12}$$

Since this term does not contain the unknown $C$ it does not contribute to the matrix $A$ but to the vector $b$. Inserting $\phi_i^e$ for $v$ and using constraint (4.2) we get the contribution to the vector entry $b_i^e$:

$$-\int_{\gamma_e} \phi_i^e J \, ds \qquad \forall \gamma_e \subseteq \Gamma_J \tag{4.13}$$

### 4.1.5 Parameterizing UDG

There are several ways how UDG has to be parameterized currently. Most important is the choice whether or not to do *normalizing*. UDG currently works by evaluating a level-set function on the nodes of a fine grid, which we call the *geometry grid*. If one of the nodes of the geometric grid is outside the domain but very near the domain boundary, the process of normalizing will move the boundary onto that node. This is done to avoid very small elements.

Of course, this is not always feasible. In the circular channel setup (Figure 4.9) the velocity is perfectly aligned with the boundary. If we now move the boundary, the flow lines will cross it. However, because of the noflux boundary condition, the flux of concentration across the boundary is zero, hence concentration cannot move across that boundary and will start to accumulate. So in the case of analytic velocities normalizing is generally a bad idea.

If we compute the velocity using Stokes equation normalizing does not hurt, since the velocity computation takes the adjusted geometry into account. In fact in the Stokes implementation, very small elements lead to a badly conditioned matrix, so normalizing becomes mandatory.

## 4.2 Testing the Code

There are several ways how the code can be tested.

#### Convergence Test

This is the most important test. After the final time step we calculate the relative error in the $L^2$ norm

$$\frac{1}{|\Omega|} \sqrt{\int_\Omega (C - C_{\text{ref}})^2 \, dx} \tag{4.14}$$

and use it to determine the experimental order of convergence. The current code is limited to analytic reference solutions $C_{\text{ref}}$, so it is currently not possible to take a solution on a very fine grid as a reference.

#### Breakthrough Test

In each time step $n$, we calculate the relative flux leaving our domain as

$$f_n = \frac{\int_{\Gamma_O} C \mathbf{u} \cdot \mathbf{n} \, ds}{\int_{\Gamma_O} \mathbf{u} \cdot \mathbf{n} \, ds}, \tag{4.15}$$

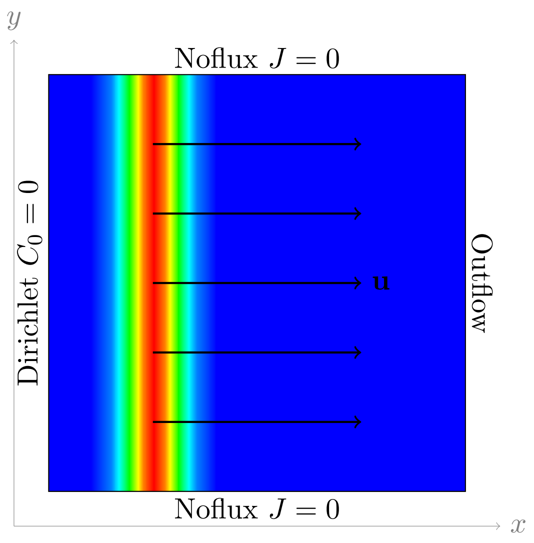


**Figure 4.1:** The *linear channel* setup in 2 dimensions. The domain is "empty," that means it has no internal structure, and its extension is 1 in each direction. The velocity is homogeneous and points in direction of the $x$ axis. Boundary conditions are noflux on the upper and lower boundaries, $C_0 = 0$ to the left and outflow to the right. Initial condition is a gauss pulse at $x = 0.25$.

where "relative" means relative to the outflow of solvent. If we do this for successively finer $h$ we will get a series of breakthrough curves which should eventually converge. Again we can determine the experimental order of convergence in the $L^2$ norm. This time however we can use a calculated solution as the reference.

### Mass Balance Test

This is a test of the velocity function $\mathbf{u}(\mathbf{x})$. It tests the local mass balance

$$\int_{\partial\Omega_e} \mathbf{u} \cdot \mathbf{n}_e \, ds \qquad \forall e \in E_h \tag{4.16}$$

and the global mass balance

$$\int_{\partial\Omega} \mathbf{u} \cdot \mathbf{n} \, ds. \tag{4.17}$$

A nonzero value means there are sources or sinks in the element $e$ or the domain $\Omega$. For the local mass balance, values in the order of $10^{-16}$ relative to the size of the cells are usually OK for IEEE doubles.[2]

## 4.2.1 Gaussian Pulse in Linear Channel

The linear channel setup is shown in figure 4.1. It contains no internal structure and the flow is strictly in $x$-direction. The velocity is homogeneous, has magnitude 1, and points in direction of the $x$ axis. Boundary conditions are noflux (Neumann with $J = 0$) on the upper and lower boundaries, Dirichlet $C_0 = 0$ to the left and outflow to the right. Initial condition is a Gaussian pulse of height $\alpha_0 = 1$ and width $\sigma_0 = 0.05$ at $x = 0.25$. It is homogeneous in $y$- and $z$-direction.

[2]If the velocity function was obtained by taking the derivative of another function halve of the precision is lost, so one can only expect $10^{-8}$.

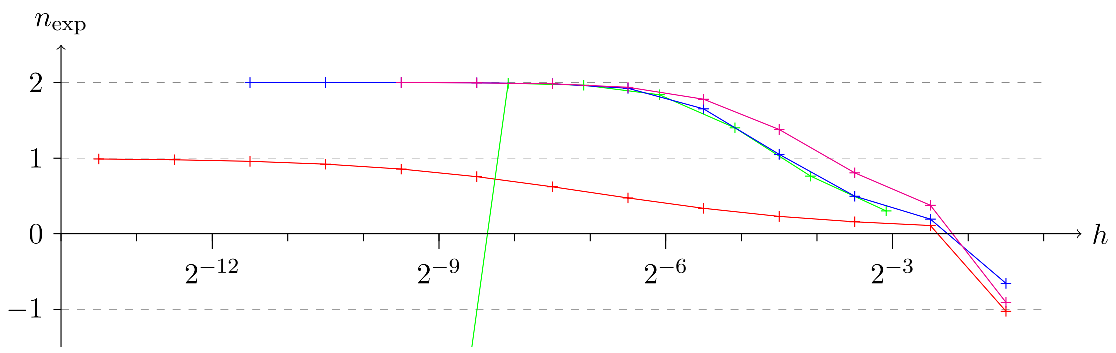


**Figure 4.2:** Order of convergence for the pseudo 1D problem without diffusion. Setup is the linear channel (figure 4.1) with second order shape functions. —+— Implicit Euler. —+— Implicit Euler (double refinement). —+— Crank-Nicholson. —+— Fractional Step.

The duration of the simulation is 0.5, which means that the initial pulse will be transported to $x = 0.75$. If there is diffusion (with the diffusion constant $D$) the final width of the pulse will be $\sigma_{\mathrm{f}} = \sqrt{\sigma_0^2 + D}$ and its final height will be $\alpha_{\mathrm{f}} = \alpha_0/\sqrt{D/\sigma_0^2 + 1}$. We will use $D = 10^{-3}$ exclusively which gives us $\sigma_{\mathrm{f}} = 0.059$ and $\alpha_{\mathrm{f}} = 0.845$. With pure convection the pulse will be undisturbed apart from the translation in $x$-direction.

For the initial refinement the number of time steps is 1 and the number grid elements in each direction is chosen according to the CFL condition[3]

$$\frac{u \cdot \Delta t}{h} < c, \tag{4.18}$$

where $c$ is a problem dependent constant (usually 1). This gives us 2 steps for the implicit Euler and Crank-Nicholson time schemes, and 6 steps for the fractional step time scheme, since each time step may be three times as large as for the other schemes.

### Pseudo 1D

Since this setup is basically 1D, it is sufficient to refine the 2D grid only in $x$-direction and to have only one element in $y$-direction.

Figure 4.2 shows the test results without diffusion and figure 4.3 those with diffusion $D = 10^{-3}$.

The implicit Euler scheme approaches the expected order of convergence of 1 both with and without diffusion. The approach is very slow however compared to the other schemes. The implicit Euler scheme is not very interesting for us due to being only first order accurate. It is included for comparison only and we will not use it later on.

Also for comparison we have included a variant of the implicit Euler scheme, where we refine the time domain twice each time we refine the space domain. This is denoted as "implicit Euler (double refinement)". Here we expect second order accuracy, and indeed

[3]R. Courant, K. Friedrichs and H. Lewy

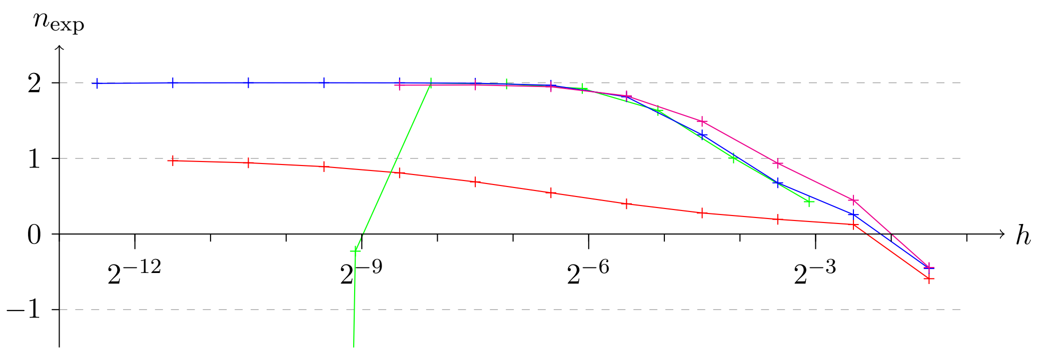


**Figure 4.3:** Order of convergence for the pseudo 1D problem with diffusion $D = 10^{-3}$. Setup is the linear channel (figure 4.1) with second order shape functions. 🟥 Implicit Euler. 🟪 Implicit Euler (double refinement). 🟦 Crank-Nicholson. 🟩 Fractional Step.

it is observed in both tests. We could only test this scheme to a much coarser refinement compared to the others due to the quadratically increasing number of time steps. This limits the usefulness of this scheme; it was only used to test the space discretization while the code was still written and a second order accurate time discretization was not yet available. We will not consider this scheme any further.

The Crank-Nicholson scheme behaves very well. It quickly approaches the expected order of 2.

With the three schemes discussed so far the error increases in the first refinement step. This is probably due to the method with which the error is calculated. Since the exact solution is not a polynomial but a Gaussian pulse, Gaussian quadrature is no longer exact but an approximation. This approximation improves with increasing number of quadrature points and with decreasing $h$. We used quadrature rules with four quadrature points in each dimension, but in the coarsest refinement level this was probably not enough, resulting in the Gaussian being missed and thus in a much too small approximated error.

We have also tested with our implementation of the fractional step scheme. As can be seen from these tests, this implementation of the fractional step has severe problems. Down to $h = \frac{1}{3} \cdot 2^{-7}$ the order of convergence approaches the expected 2. Then it suddenly drops and the error gets worse instead of better. This is a typical sign of an error in the implementation. However, we have been unable to locate it in the available time. Thus we will not use the fractional step scheme any further in this work.

### 2D and 3D

We also tested this setup with the Crank-Nicholson scheme and second order shape function in 2 and 3 dimensions. The results are in figures 4.4 and 4.5. These tests were run with diffusion $D = 10^{-3}$. In 2D we can see how the order converges nicely to the expected 2. In 3D we are limited by computation time and thus cannot reach

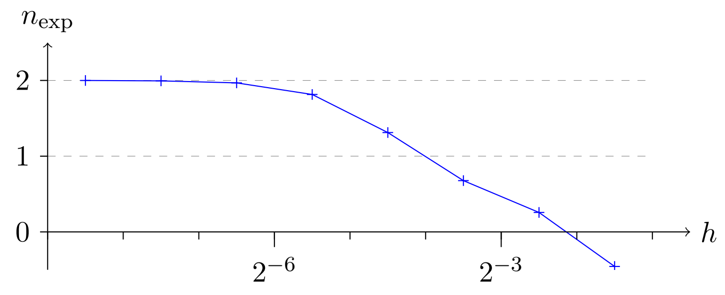


**Figure 4.4:** Order of convergence for the 2D problem with diffusion $D = 10^{-3}$. Setup is the linear channel (figure 4.1) with second order shape functions and Crank-Nicholson.

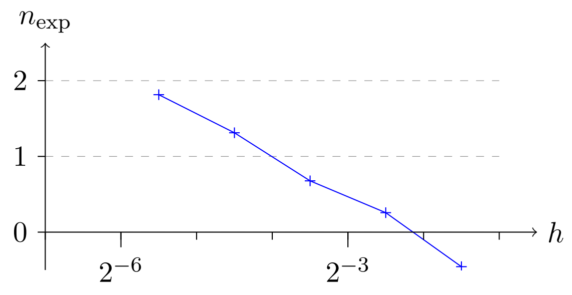


**Figure 4.5:** Order of convergence for the 3D problem with diffusion $D = 10^{-3}$. Setup is the linear channel (figure 4.1) with second order shape functions and Crank-Nicholson.

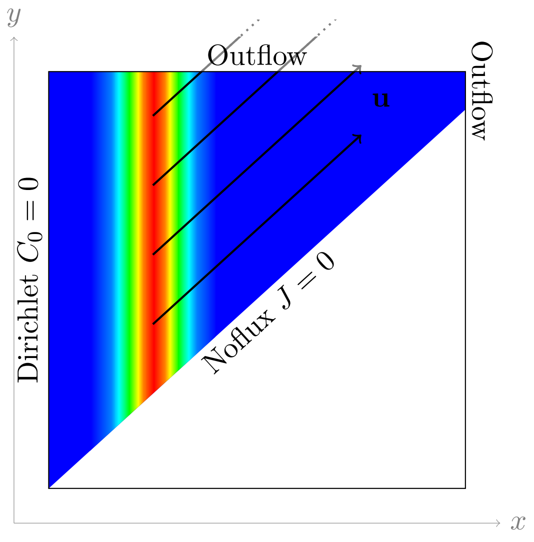


**Figure 4.6:** The *tilted channel* setup in 2D. Like the linear channel setup (figure 4.1), except that the lower right of the domain as marked by a straight line through the origin is removed. The velocity is no longer parallel to the $x$ axis but parallel to the clipping line. The boundary conditions on the top border changed from noflux to outflow. In 3D the domain is simply extended in the $z$-direction.

the limit. On the other hand, up to those refinement levels that could be tested we found no unexpected behavior. In both cases we note again the problem with the bad approximation when calculating the error in the coarsest level, as we have seen it before in the 1D case.

### 4.2.2 Gaussian Pulse in Tilted Channel

The tilted channel setup is shown in figure 4.6. It is basically the unit square (or unit cube in 3D). However, the lower part is clipped away along the line (or plane) given by $y = ax$. In the tests we use $a = 1/1.1$, such that the clipping line intersects the extension of the upper boundary at $x = 1.1$. The velocity is homogeneous, with $u_x = 1$, $u_y = a$ and in 3D $u_z = 0$. That means it is parallel to the clipping line. Boundary conditions are noflux (Neumann with $J = 0$) on the lower boundary, Dirichlet $C_0 = 0$ to the left and outflow on the upper and right boundaries. The initial condition is identical to the linear channel setup: a Gaussian pulse of height $\alpha_0 = 1$ and width $\sigma_0 = 0.05$ at $x = 0.25$.

As before, the duration of the simulation is 0.5, which means that the initial pulse will be transported to $x = 0.75$. It will also be transported by $a/2$ in $y$-direction, but since it is homogeneous in $y$ and the parts that are outside the domain do not matter we can ignore that. Unfortunately we do not know an exact solution for the case with diffusion, since the pulse is no longer perpendicular to the boundary. With pure convection the pulse will be undisturbed apart from the translation in $x$-direction.

The experimental order of convergence for 2D and 3D is shown in figures 4.7 and 4.8. As with the linear channel, the 2D test converges nicely to the expected order of 2. Again, in 3D we cannot really reach the limit due to limited computation time, but we still see nothing contradicting second order accuracy.

The negative order of convergence in the first refinement step is not as bad as before. This is consistent with our previous explanation. Since in this setup the area of the

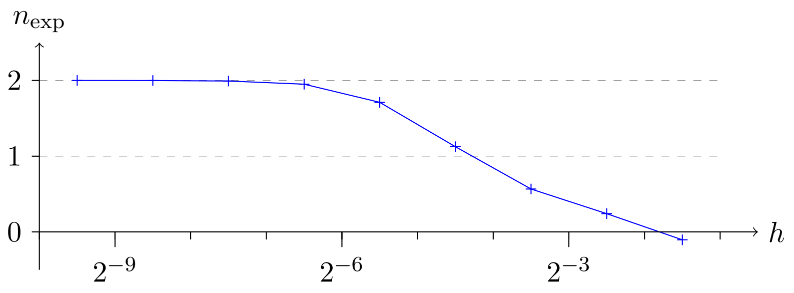


**Figure 4.7:** Order of convergence for the 2D problem without diffusion. Setup is the tilted channel (figure 4.6) with second order shape functions and Crank Nicholson.

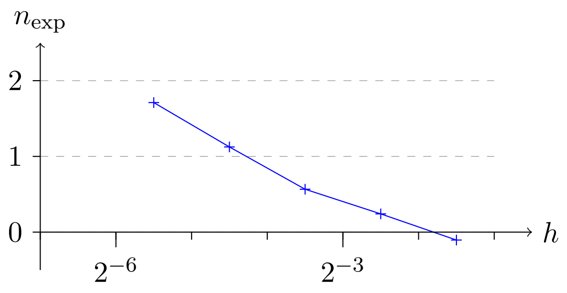


**Figure 4.8:** Order of convergence for the 3D problem without diffusion. Setup is the tilted channel (figure 4.6) with second order shape functions and Crank-Nicholson.

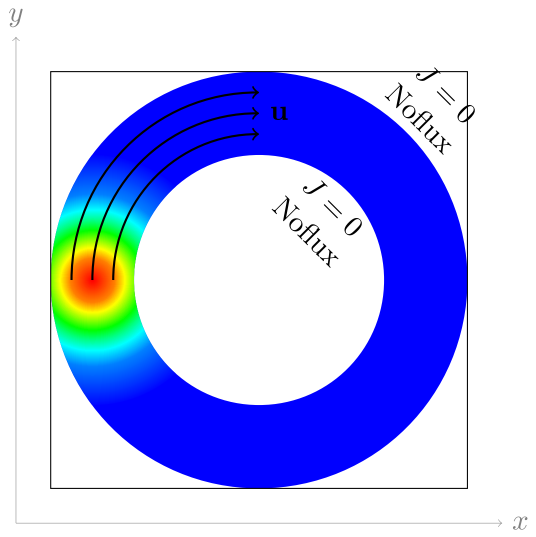


**Figure 4.9:** The *circular channel* setup in 2D. A circular channel with variable inner diameter. The outer diameter is the same as the extension of the grid. The velocity is clockwise along the channel and its magnitude is proportional to the distance from the center (the angular velocity is constant). Boundary conditions are noflux on all boundaries, initial condition is a Gauss peak in the left side of the channel.

Gaussian pulse in the final solution is much smaller than in the linear channel setup, a much smaller error will result overall.

### 4.2.3 Gaussian Peak in Circular Channel

This is the setup in figure 4.9. The domain is the space between two circles in 2D and two cylinders in 3D. The outer diameter is 1 and the inner is 0.6. The flow is clockwise with a constant angular velocity, i.e. the magnitude of the velocity grows linearly with the distance from the center and is 1 at the outer border. This is the first example with a spatially inhomogeneous velocity field. Boundary conditions are noflux everywhere. The initial condition is a Gaussian peak with a maximum of $\alpha_0 = 1$ sitting left of the center in the middle of the channel. Its width is $\sigma_0 = 0.05$.

The duration of the simulation is $\pi/4$ and is chosen such that the pulse will be transported exactly above the center. Unlike before, this is not the limit of what is permitted by the Courant-Friedrichs-Lewy condition. Again, the exact solution for the case with diffusion is unknown, so we will only consider the case without diffusion.

The test results are in figures 4.10 and 4.11. In 2D, the order of convergence approaches 2. For $h < 10^{-7}$ it becomes smaller again, although it still stays very close to 2. For the 3D case, we see the order of convergence actually approaching 2, although barely. The order of convergence is better than in the earlier test, probably because we do not fully exploit the limit permitted by the CFL condition.

The negative order of convergence in the first refinement step is much worse than before. Indeed the reason is different: the grid on which the geometry reconstruction for $h = 2^0$ is done is too coarse to approximate the geometry properly. The domain is broken into four disconnected parts and the peak of the solution is completely outside of the domain.

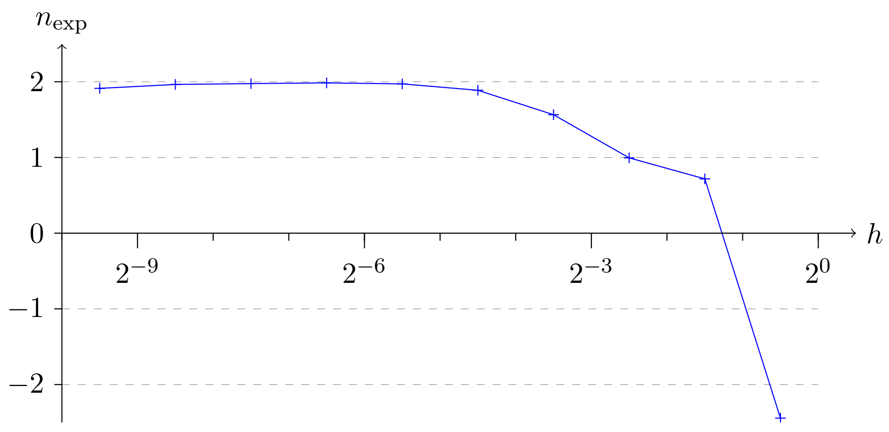


**Figure 4.10:** Order of convergence for the 2D problem without diffusion. Setup is the circular channel (figure 4.9) with second order shape functions and Crank-Nicholson.

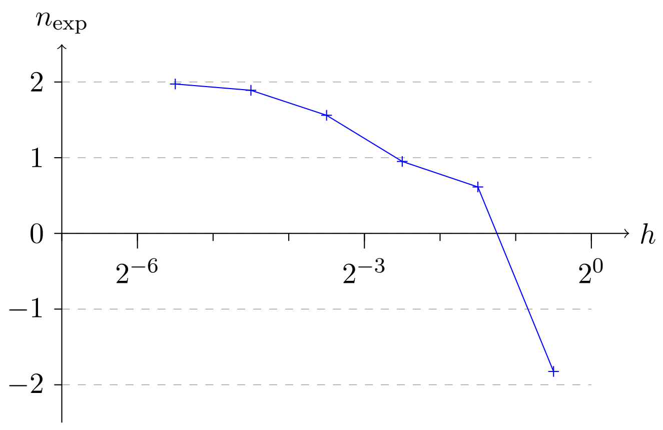


**Figure 4.11:** Order of convergence for the 3D problem without diffusion. Setup is the circular channel (figure 4.9) with second order shape functions and Crank-Nicholson.

# 5 Results

## 5.1 Application: Tailing in Porous Media

As an example application we simulate tailing in porous media. To calculate the velocity field we use a Stokes implementation by S. Pulloor Kuttanikkad. It also uses Dune and UDG and thus integrates nicely with our code. It supports several sets of boundary conditions, of which we will use the "pressuredrop" set: $p = p_1$ on the inflow boundary to the left, $p = p_2 < p_1$ on the outflow boundary to the right, and *noslip* $\mathbf{u} = 0$ everywhere else. The code also offers several discretization schemes, we will use the OBB scheme and no interior penalty. See Pulloor Kuttanikkad (2008) and Engwer *et al.* (2008).

Unfortunately the Stokes discretization implementation currently has one issue, which S. Pulloor Kuttanikkad is still working on at the time of this writing. The local mass balance $\int_{\partial\Omega_e} \langle \mathbf{u} \rangle \cdot \mathbf{n}_e = 0$ (with $e \in E_h$) is only badly fulfilled, and thus the global mass balance $\int_{\partial\Omega} \mathbf{u} \cdot \mathbf{n} = 0$ is problematic as well. The worst deviations in the local mass balance are in the order of $|\mathbf{u}| \cdot |e| \cdot 10^{-3}$. We will ignore this problem for this sample application, since it does not lead to complications later on.

While it is convenient to denote one side of the domain as "the outflow boundary," this may conflict with the condition $\mathbf{u} \cdot \mathbf{n} > 0$ (2.10) for the outflow boundary. Unlike the analytic velocities in section 4.2, which are completely known when denoting the boundary conditions, computed velocity fields may exhibit small flows into the domain at certain points on the intended outflow boundary. This is a result of the discretization, which allows small undershoots if they make the overall result better. We have observed these inflows when a sphere intersects the intended outflow boundary. The correct solution is to dynamically restrict the outflow boundary to the parts where (2.10) is fulfilled, and to use noflux boundary conditions on the other parts. This is done by checking at each quadrature point whether (2.10) holds and selecting outflow or noflux BC accordingly. This is a similar approach to the one we took in 4.1.4 with the Heaviside step function $\Theta$.

We used the setup shown in figure 5.1. It is 2D and has a height of 1 and a width of 1.5. The domain is a random packing of spheres with radius 0.1 and a minimum distance of 0.05 in the region $x > 0.5$. The left part $x \lesssim 0.5$ is free of internal structure. The velocity indicated by the flow lines is calculated using the Stokes equation with pressure drop boundary conditions ($p = p_1$ to the left, $p = p_2$ to the right, and noslip everywhere else). The flow (see also figure 5.2) is from left to right and the velocity field is scaled such that the approximate maximum velocity magnitude

$$u_{\max} = \max_{\mathbf{x}_q} |\mathbf{u}(\mathbf{x}_q)|, \qquad \mathbf{x}_q \in \{\text{quadrature points}\} \tag{5.1}$$

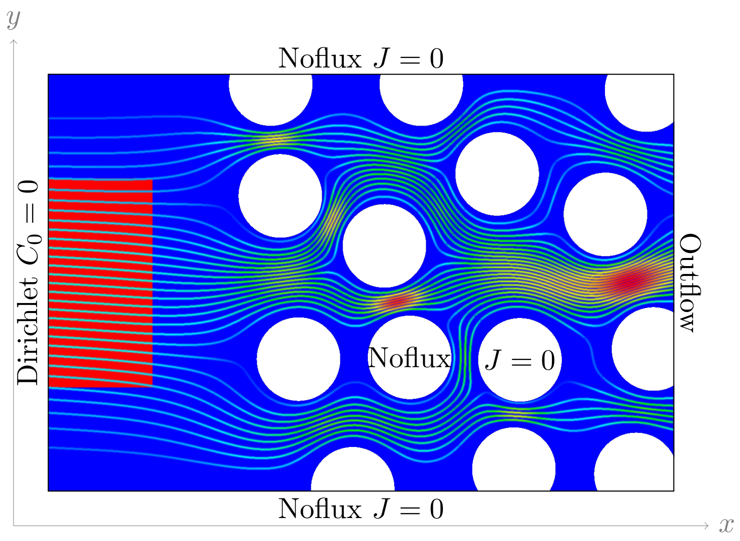


**Figure 5.1:** The setup to simulate tailing. It is 2D and has a height of 1 and a width of 1.5. The domain is a random packing of spheres with radius 0.1 and a minimum distance of 0.05 in the region $x > 0.5$. That region has a porosity of $\Phi = 0.64$. The left part $x \lesssim 0.5$ is free of internal structure. The velocity indicated by the flow lines is calculated with the Stokes equation with pressure drop boundary conditions ($p = p_1$ to the left, $p = p_2$ to the right, and noslip everywhere else) with the flow going from left to right. The boundary conditions for the transport are specified in the illustration. Initial condition is a rectangular pulse of concentration 1 at the left border (as shown by the red box) and zero concentration everywhere else. This illustration was created with $h = 2^{-6}$.

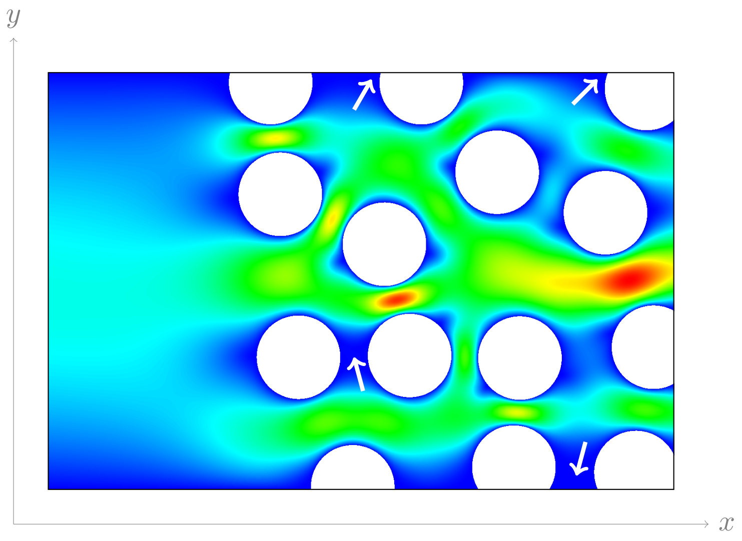


**Figure 5.2:** The magnitude of the velocity field used in the computation. Red denotes $|\mathbf{u}| = 1$ while blue denotes $|\mathbf{u}| = 0$. The arrows denote stagnant zones. This illustration was created with $h = 2^{-6}$.

is 1. For the transport, boundary conditions are Dirichlet $C_0 = 0$ to the left, outflow to the right,[1] and noflux $J = 0$ on the upper, lower and internal boundaries. Initial condition is a rectangular pulse of concentration 1 in the region $0 < x < 0.25$ and $0.25 < y < 0.75$ and zero concentration everywhere else. For the calculation we used diffusion with $D = 10^{-3}$.

The duration of the simulation was chosen to be 16, with the number of time steps in the initial refinement being 32. After $\Delta t = 0.5$ a snapshot of the concentration distribution was taken and saved for later analysis.

The development of the concentration for $h = 10^{-6}$ is shown in figure 5.3. We can see that after about $t = 12.5$ the main pulse has passed and we can identify several stagnant zones still containing concentration. These were marked in figure 5.2.

We determined the breakthrough curves for different $h$ (figure 5.4). These show clear evidence for tailing. The pulse has a steep front and a much gentler slope in the back.

From the breakthrough curves we estimated the order of convergence using the solution for $h = 2^{-8}$ as the reference. First we note the very low order between $h = 2^{-6}$ and $h = 2^{-7}$. This could be because $h = 2^{-7}$ is too close to the reference solution already. However, in this case there is a much stronger effect: the outflow values were written to an ASCII log file using the standard precision, and then read back to produce these plots. This reduces the precision to 6 decimal digits. At $h = 2^{-6}$ the absolute $L^2$ error is already smaller than $10^{-6}$, so the error due to the lack in precision becomes dominant.

The experimental order of convergence is quite jumpy, something we are not used to from the earlier plots in chapter 4.2. However, those were all ideal problems, so this is

[1]As permitted by (2.10); noflux otherwise.

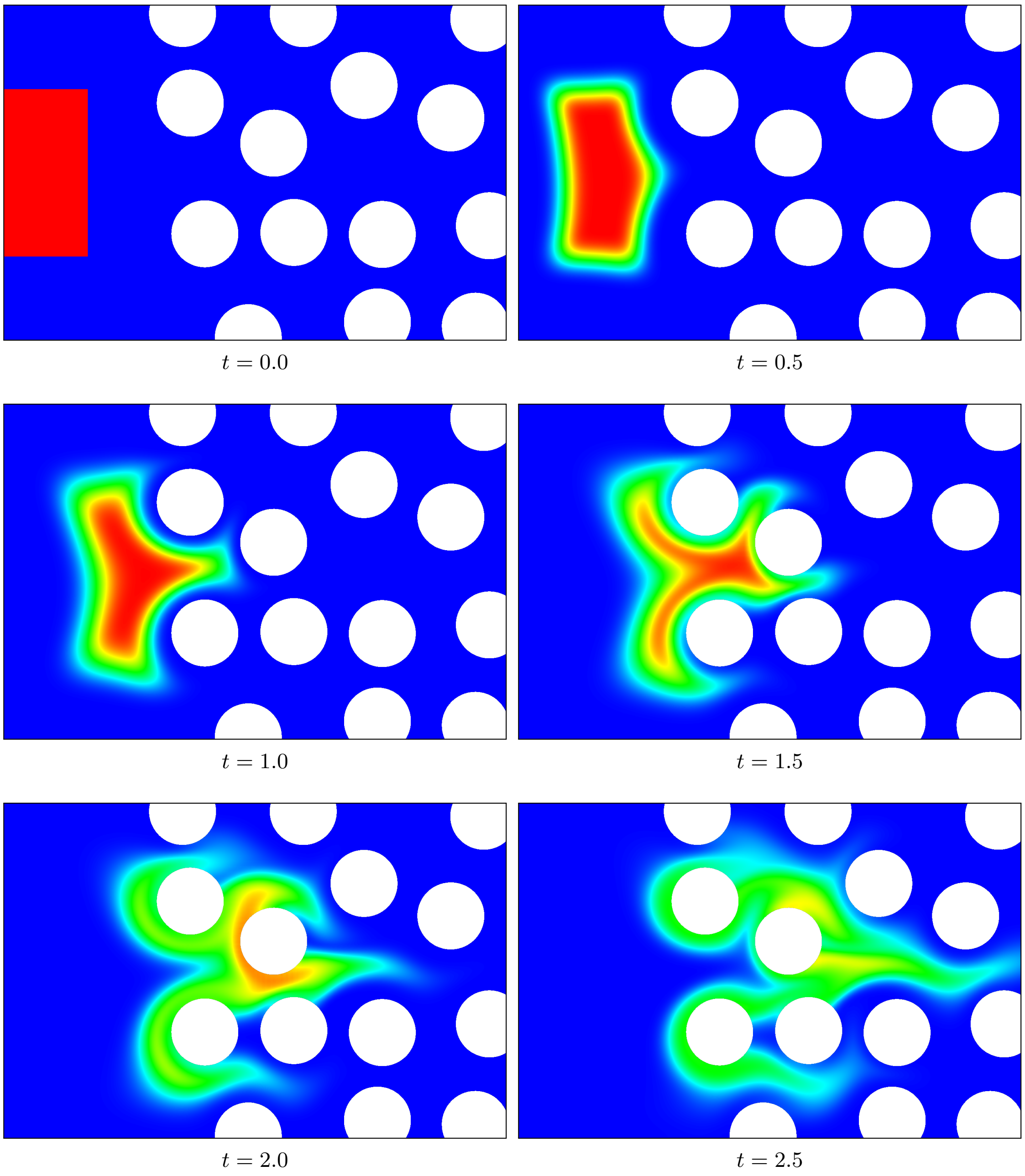


**Figure 5.3:** The development of the concentration given the setup in figure 5.1. This calculation was done with $h = 2^{-6}$ and the situation after each 32nd time step is shown. All pages of this figure overlap by one frame with the previous and next page. Each page has its own color scale; on this page blue denotes $C = 0$ and red denotes $C = 1$. *(Continued on the next page.)*

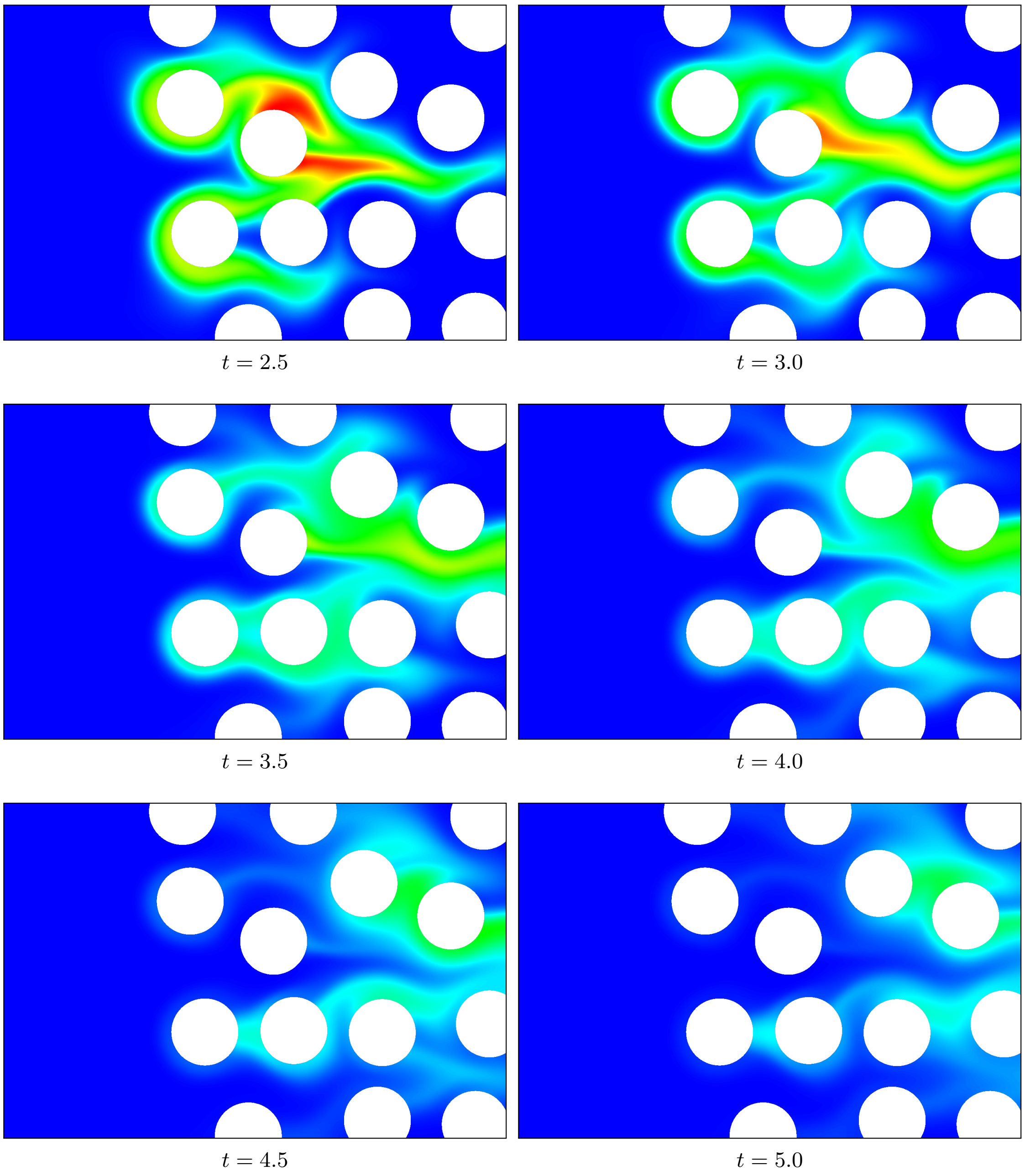


**Figure 5.3 (cont.):** The development of the concentration given the setup in figure 5.1. This calculation was done with $h = 2^{-6}$ and the situation after each 32nd time step is shown. All pages of this figure overlap by one frame with the previous and next page. Each page has its own color scale; on this page blue denotes $C = 0$ and red denotes $C = 0.75$. *(Continued on the next page.)*

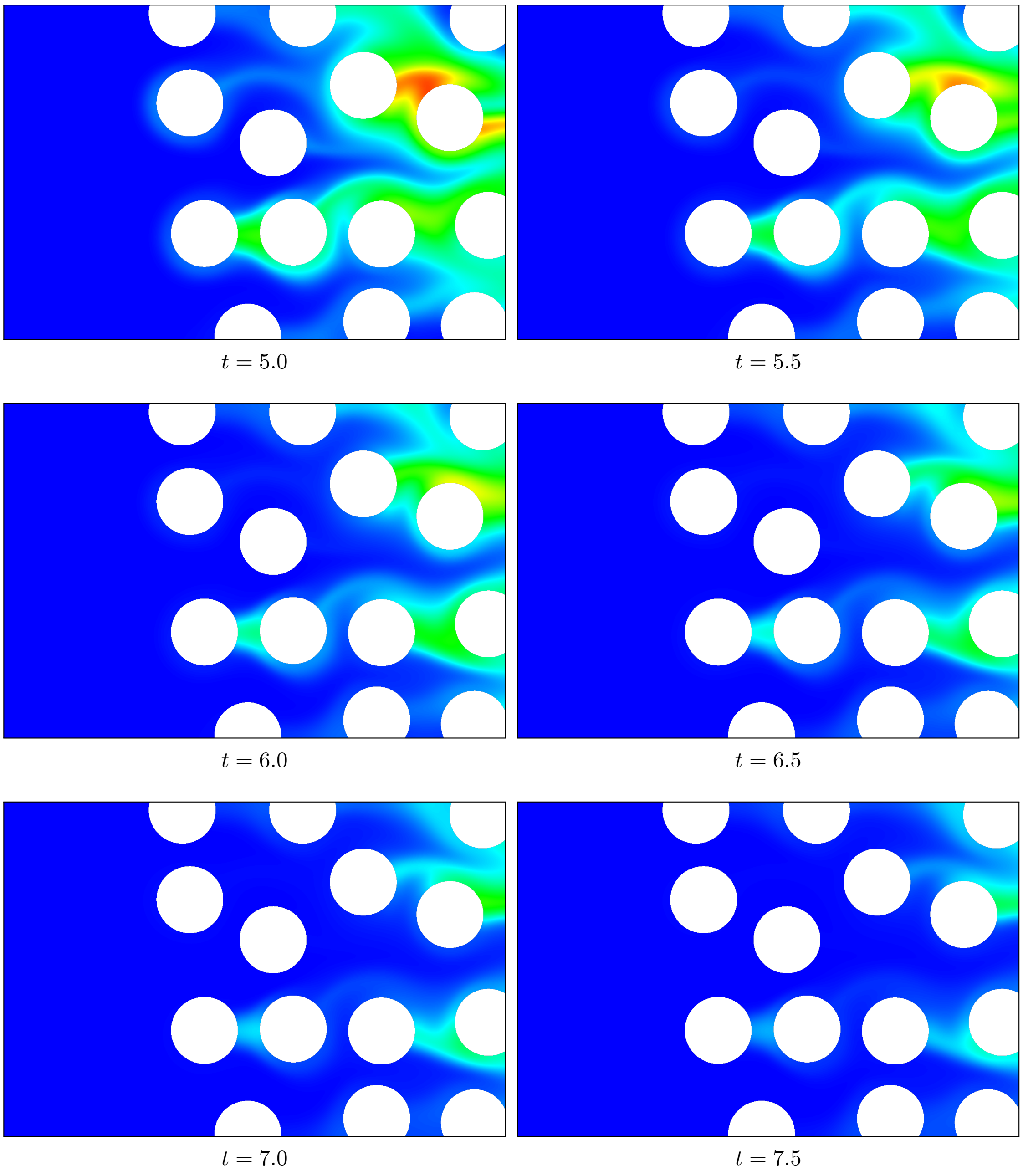


**Figure 5.3 (cont.):** The development of the concentration given the setup in figure 5.1. This calculation was done with $h = 2^{-6}$ and the situation after each 32nd time step is shown. All pages of this figure overlap by one frame with the previous and next page. Each page has its own color scale; on this page blue denotes $C = 0$ and red denotes $C = 0.35$. *(Continued on the next page.)*

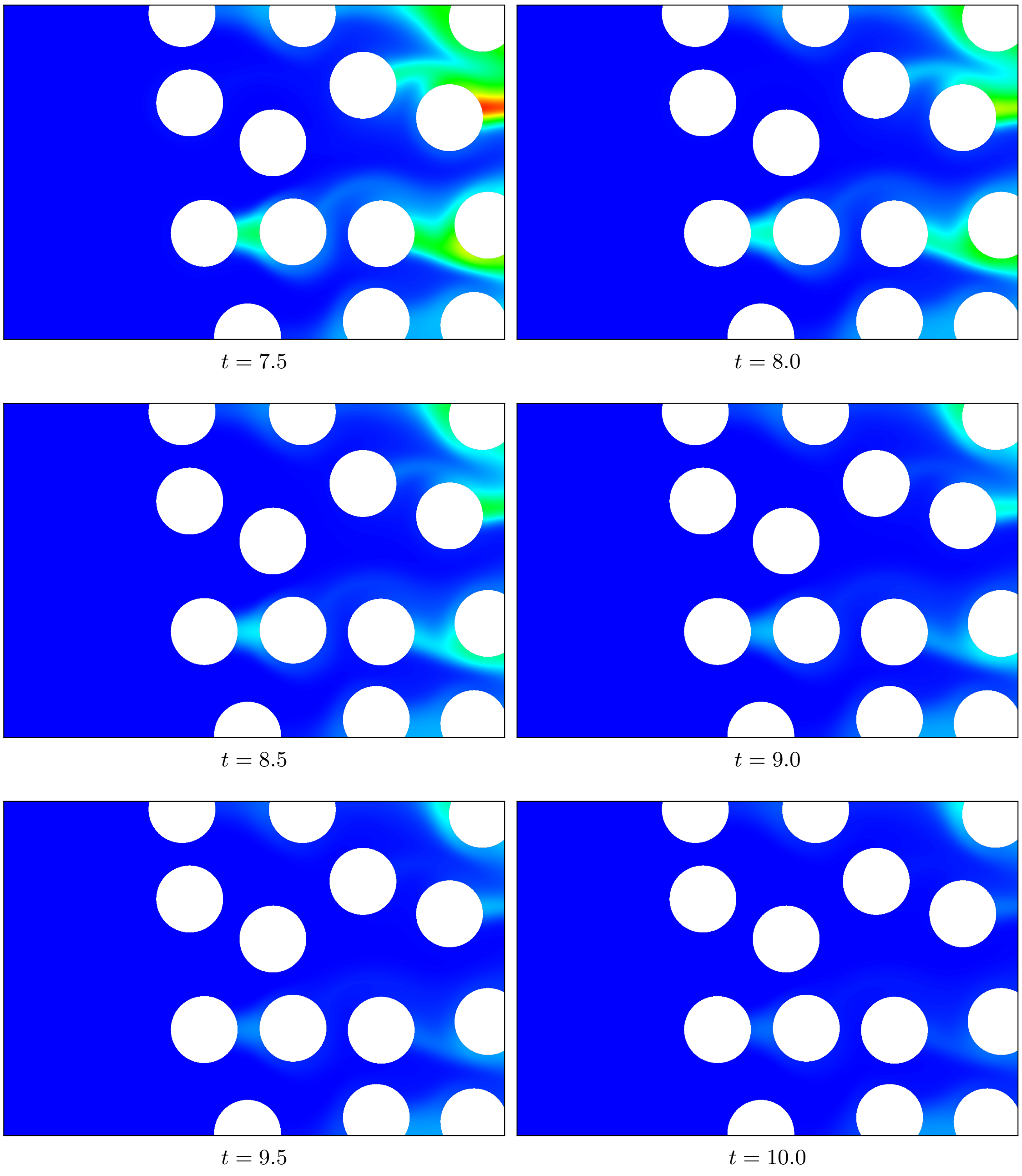


**Figure 5.3 (cont.):** The development of the concentration given the setup in figure 5.1. This calculation was done with $h = 2^{-6}$ and the situation after each 32nd time step is shown. All pages of this figure overlap by one frame with the previous and next page. Each page has its own color scale; on this page blue denotes $C = 0$ and red denotes $C = 0.15$. *(Continued on the next page.)*

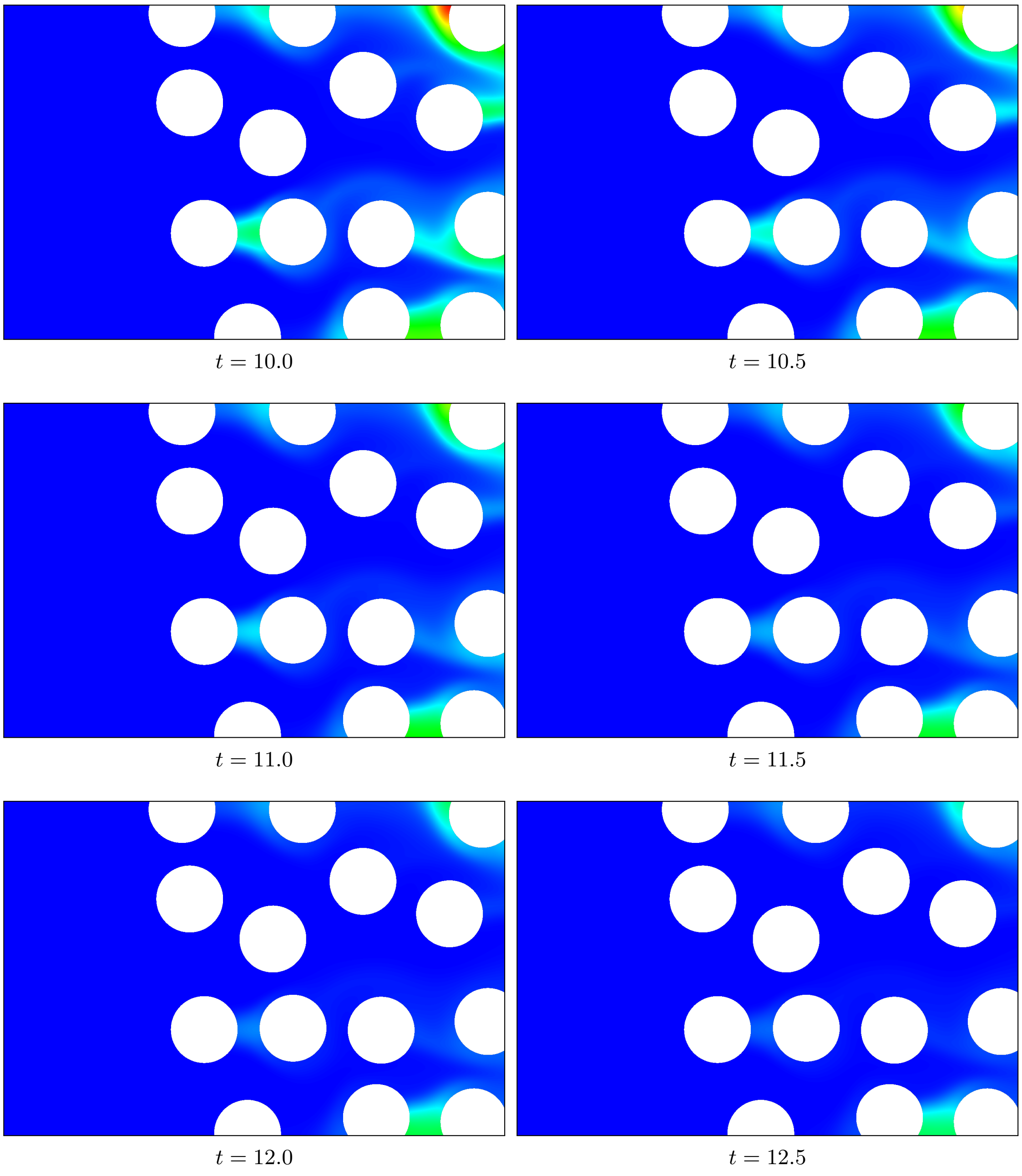


**Figure 5.3 (cont.):** The development of the concentration given the setup in figure 5.1. This calculation was done with $h = 2^{-6}$ and the situation after each 32nd time step is shown. All pages of this figure overlap by one frame with the previous and next page. Each page has its own color scale; on this page blue denotes $C = 0$ and red denotes $C = 0.04$. *(Continued on the next page.)*

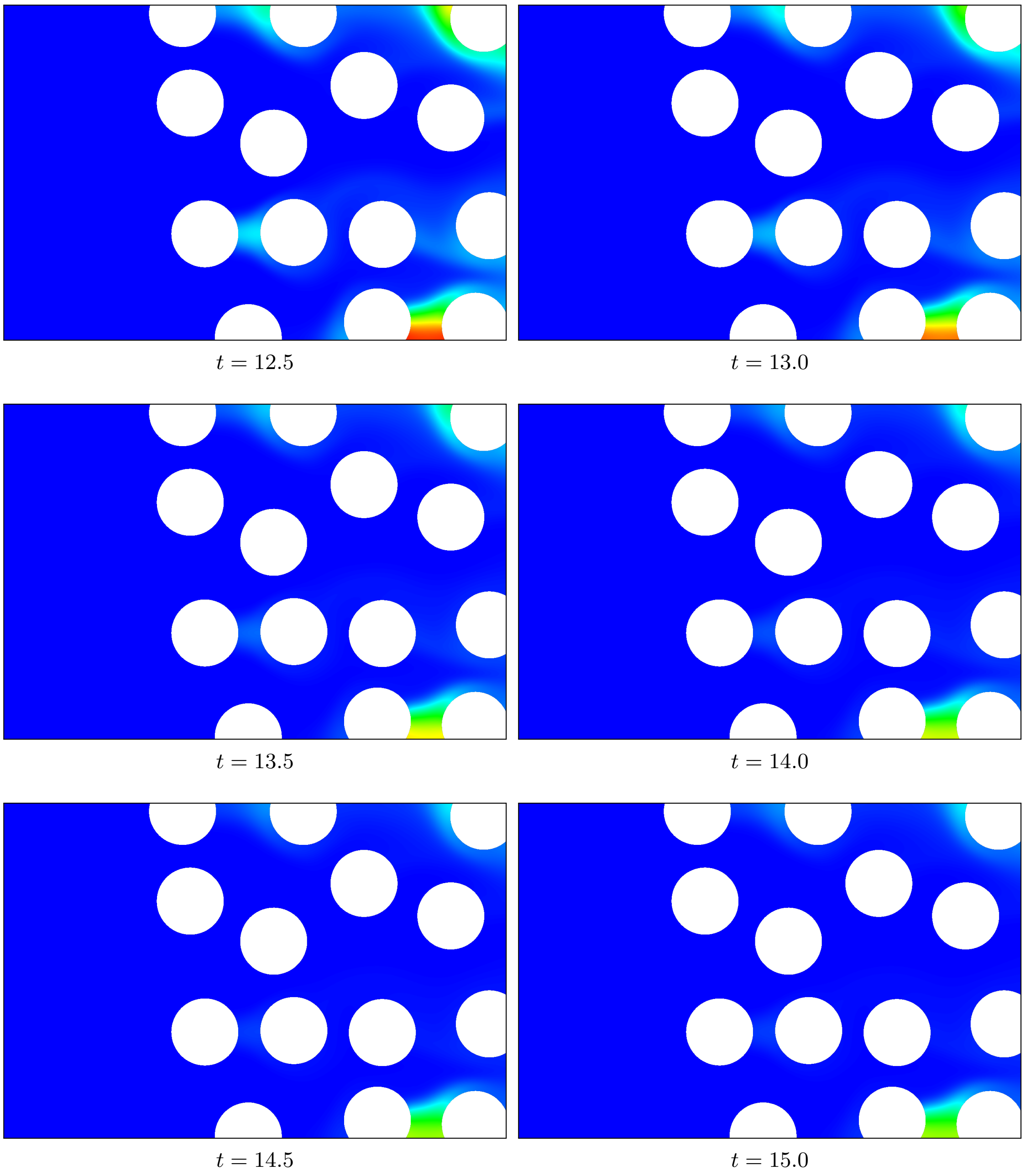
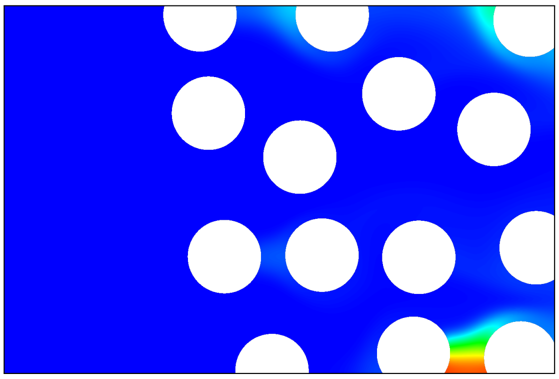
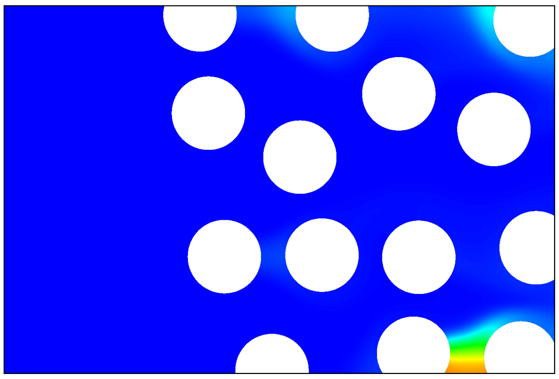


**Figure 5.3 (cont.):** The development of the concentration given the setup in figure 5.1. This calculation was done with $h = 2^{-6}$ and the situation after each 32nd time step is shown. All pages of this figure overlap by one frame with the previous and next page. Each page has its own color scale; on this page blue denotes $C = 0$ and red denotes $C = 0.017$. *(Continued on the next page.)*


$t = 15.0$


$t = 15.5$

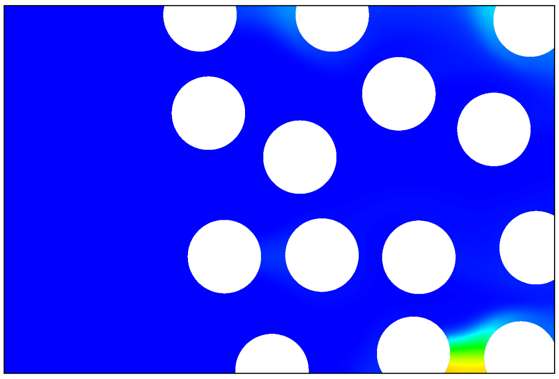

$t = 16.0$

**Figure 5.3 (cont.):** The development of the concentration given the setup in figure 5.1. This calculation was done with $h = 2^{-6}$ and the situation after each 32nd time step is shown. All pages of this figure overlap by one frame with the previous and next page. Each page has its own color scale; on this page blue denotes $C = 0$ and red denotes $C = 0.012$.

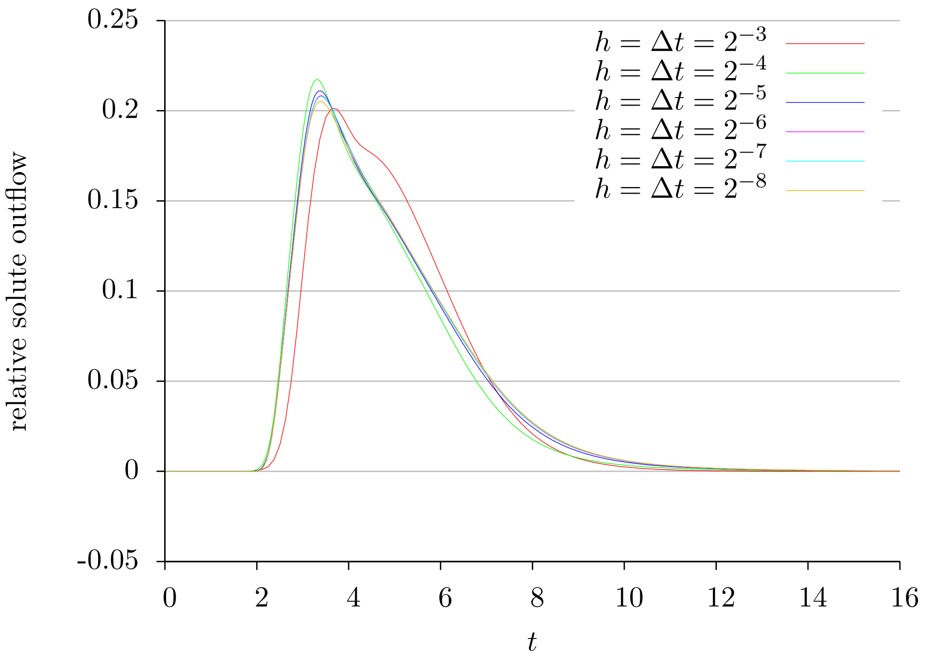


**Figure 5.4:** Breakthrough curves for different refinement levels for the setup in figure 5.1. The curves for $h = 2^{-1}$ and $h = 2^{-2}$ are not shown since the discretization is too coarse to resolve the spheres properly.

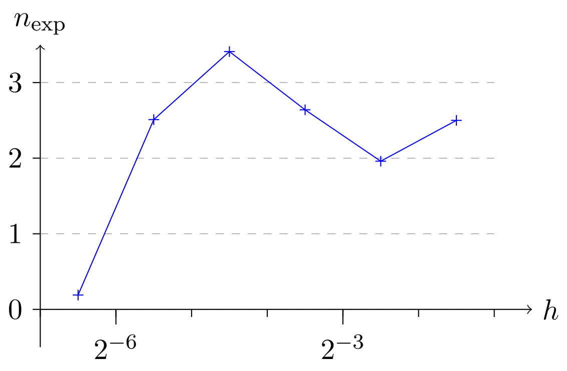


**Figure 5.5:** Order of convergence for the breakthrough curves from figure 5.4. We used the curve with $h = 2^{-8}$ as the reference.

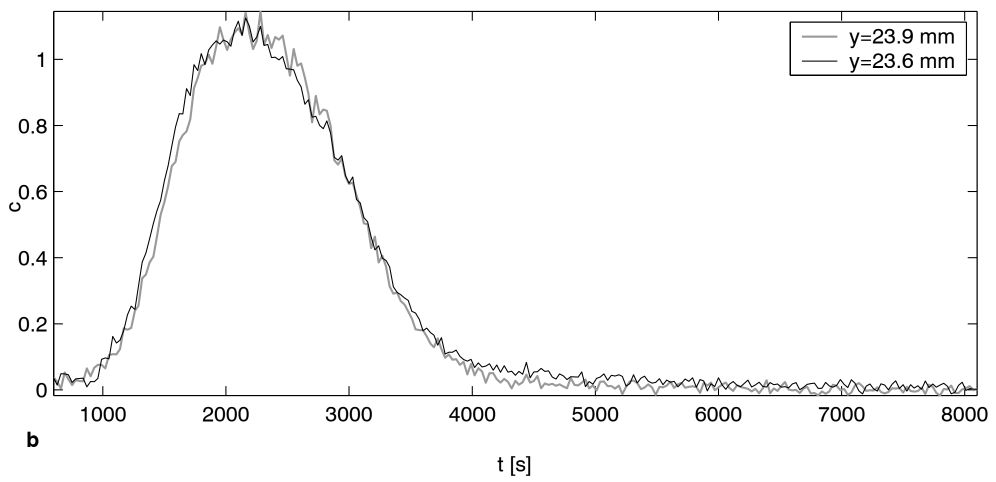


**Figure 5.6:** Tailing as seen in the experiment. The experiment was done in a 80 mm × 40 mm × 40 mm cell with the flow in $x$-direction. These breakthrough curves were taken at $x = 27.9$ mm / $z = 32$ mm and two different $y$ values. This is from figure 8.20 of Stöhr (2003).

not really a surprise.

In the average the order of convergence is higher than the expected 2. This is because we use the maximum velocity in the CFL condition (4.18), while in reality most of the flow is slower. This means the convergence order is initially not limited by the time discretization. For the space discretization we expect third order accuracy due to the second order shape functions.

The numerical results exhibit the features predicted by the MIM. Stagnant zones, as assumed in the MIM, as well as the resulting tailing are observed. This gives qualitative verification of the mobile-immobile model.

## 5.2 Comparison with Experiments

While a confirmation of the MIM was found, a comparison to real-world experiments would be good. Such experiments have been done by M. Stöhr and haven been presented in his dissertation, see Stöhr (2003). He used a cell of $8 \times 4 \times 4$ cm$^3$ packed with a porous medium of plexiglass spheres with a mean diameter of 0.6 mm. For the solvent he used silicone oil with its refractive index matched to the plexiglass. With this setup he was able to measure full three-dimensional concentration profiles using planar laser-induced fluorescence (PLIF) techniques. This way he obtained fully four dimensional datasets with an effective spatial resolution of 70 µm × 70 µm × ca. $0.5 - 1$ mm and a temporal resolution of 30 s.

Figure 5.6 shows a breakthrough curve for one of his experiments. The shape of the pulse is asymmetrical, but the asymmetry is much weaker than in the breakthrough curves obtained through simulation. There are certain differences in the setups used for simulation and real world experiments which make a direct comparison impossible. For example, simulation and experiment were done for differing numbers of dimensions and the size of the domain compared to the size of the pores was much bigger in the experiment.

Especially the differing number of dimensions makes a big difference, since in three dimensions the velocity has more possibilities to flow around an obstacle, thus reducing the probability and size of stagnant zones. These issues can be resolved by making the implementation parallel, thereby allowing the simulation of larger systems (such as 3D).

When these issues are overcome, a quantitative verification of the MIM should be possible.

# 6 Conclusion and Outlook

In this work an Unfitted Discontinuous Galerkin (UDG) method for the convection-diffusion equation was implemented. This allows the direct simulation of transport processes on the pore scale, using state of the art numerical methods. A simulation of an artificial domain was then carried out, resulting in a confirmation of the mobile-immobile model.

For the time discretization second order accurate time schemes were investigated. The space discretization used second order shape functions, resulting in a third order accurate space discretization. The implementation was done using UDG and DUNE as a framework. It was then tested using several test problems with sufficient regularity and analytically given velocity fields. The Crank-Nicholson time scheme was selected to run the simulations.[1] Full convergence rates were observed for two and three space dimensions.

To verify the mobile-immobile model, numerical experiments of pore scale convection dominated transport were carried out in an idealized domain of randomly generated spheres in two space dimension. The flow field was given by the Stokes equation. To solve the Stokes equation an existing implementation for UDG was used.[2]

The simulations concluded in a qualitative confirmation of the mobile-immobile model, a macroscopic empirical law, using only first principles. As predicted by the MIM, the breakthrough curves showed strong tailing, even without adsorption. Stagnant zones were observed, a precondition for the MIM. The simulation showed the expected order of convergence.

This demonstrates that the new model is a powerful numerical tool for pore scale simulations. Computations in two dimensions were shown, and code for three dimensions exists.[3] In fact, the implementation is totally independent on the number of dimensions or the order of the shape functions.

Of course there are many improvements possible which were beyond the scope of this work. One improvement would be to make the input velocity continuous using a BDM projection (Bastian and Rivière (2003)), which would allow a finer grid for the velocity computation than for the transport simulation. Parallelization would permit the simulation much larger systems, which will enable some interesting applications, such as the use of a pore structure obtained from a real-world sample using micro-tomography.

[1]A fractional step $\Theta$ time scheme was also implemented but failed the validation. The programming error has so far not been found. Fractional step $\Theta$ is more robust than Crank-Nicholson and would allow for an even better time discretization.

[2]This implementation of Stokes does not guarantee local mass conservation.

[3]Due to the lack of computing power the simulations were only in two dimensions.

This can then be used for a direct comparison of simulation and experiment and a quantitative verification of the MIM.[4]

Another application is a multi-scale heterogeneous setup like pebbles embedded in a sand matrix, requiring a full velocity dependent dispersion tensor for a simulation at the scale of the pebbles. There are also applications outside the scope of soil physics, like transport in chemical and biological reactors, where a big surface area is often realized using a porous medium.

[4]Of course a mass conservative Stokes discretization would also be required.

# Bibliography

Bastian, P. 2003. Higher order discontinuous galerkin methods for flow and transport in porous media, *In* E. Bänsch (ed.) *Challenges in Scientific Computing – CISC 2002*, number 35 in *LNCSE*, S. 1–22. http://hal.iwr.uni-heidelberg.de/~peter/Papers/Bastian.html#Cisc2002.

Bastian, P. and Lang, S., 2004. Couplex benchmark computations with UG, *Computational Geosciences* **8**:125–147. http://hal.iwr.uni-heidelberg.de/~peter/Papers/Bastian.html#BastianLangCouplex.

Bastian, P. and Rivière, B., 2003. Superconvergence and H(div)-projection for discontinuous Galerkin methods, *Int. J. Numer. Meth. Fluids.* **42**:1043–1057. http://www3.interscience.wiley.com/cgi-bin/jissue/104546832.

Bastian, P., Blatt, M., Dedner, A., Engwer, C., Klöfkorn, R., Kornhuber, R., Ohlberger, M. and Sander, O., n.d.a. A generic grid interface for parallel and adaptive scientific computing. Part II: Implementation and tests in DUNE, Accepted in Computing, http://www.matheon.de/research/show_preprint.asp?action=details&serial=404.

Bastian, P., Blatt, M., Dedner, A., Engwer, C., Klöfkorn, R., Ohlberger, M. and Sander, O., n.d.b. A generic grid interface for parallel and adaptive scientific computing. Part I: Abstract framework, Accepted in Computing, http://www.matheon.de/research/show_preprint.asp?action=details&serial=403.

Bramble, J. H. and Hilbert, S. R., 1970. Estimation of linear functionals on sobolev spaces with application to fourier transforms and spline interpolation, *SIAM Journal on Numerical Analysis* **7**:112–124. http://link.aip.org/link/?SNA/7/112/1.

Brenner, S. C. and Scott, L. R., 2008. *The Mathematical Theory of Finite Element Methods*, Vol. 15 of *Texts in Applied Mathematics*, 3rd edition, Springer-Verlag. http://www.springer.com/math/cse/book/978-0-387-75933-3.

Coats, K. and Smith, B., 1964. Dead-end pore volume and dispersion in porous media, *SPE Journal* **4**:73 – 84. http://www.spe.org/elibrary/servlet/spepreview?id=647-PA.

Engwer, C. and Bastian, P., 2005. A discontinuous galerkin method for simulations in complex domains, *Technical Report 5707*, IWR, Universität Heidelberg. http://www.ub.uni-heidelberg.de/archiv/5707/.

Engwer, C. and Bastian, P., 2008. An unfitted finite element method using discontinuous galerkin, in preparation.

Engwer, C., Bastian, P. and Pulloor Kuttanikkad, S., 2008. An unfitted discontinuous galerkin finite element method for pore scale simulations, *9th International Workshop on State-of-the-Art in Scientific and Parallel Computing*, LNCS, Springer-Verlag. accepted for publication.

Hackbusch, W. 2005. *Theorie und Numerik elliptischer Differentialgleichungen*, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig. http://www.mis.mpg.de/preprints/ln/lecturenote-2805.pdf.

Oden, J. T., Babuška, I. and Baumann, C. E., 1998. A discontinuous *hp* finite element method for diffusion problems, *J. Comput. Phys.* **146**:491–519. http://dx.doi.org/10.1006/jcph.1998.6032.

Pulloor Kuttanikkad, S. 2008. Pore-scale simulation of dispersion using random walk particle tracking, Presentation given at the SgS Oberseminar in Stuttgart, http://www.ipvs.uni-stuttgart.de/abteilungen/sgs/lehre/lehrveranstaltungen/hauptseminare/WS0708/OberseminarSgS_termine/OS13/de.

Rivière, B. 2000. *Discontinuous Galerkin methods for solving the miscible displacement problem in porous media*, PhD thesis, The University of Texas at Austin.

Roth, K. 2007. Soil physics, Lecture notes. http://www.iup.uni-heidelberg.de/institut/forschung/groups/ts/students/sp.

Stöhr, M. 2003. *Analysis of Flow and Transport in Refractive Index Matched Porous Media*, PhD thesis, Institute of Environmental Physics, Department of Physics and Astronomy, University of Heidelberg. http://www.ub.uni-heidelberg.de/archiv/3733/.

van Genuchten, M. T. and Wierenga, P. J., 1976. Mass transfer studies in sorbing porous media I. analytical solutions, *Soil Sci. Soc. Am. J.* **40**:473–480. http://soil.scijournals.org/cgi/content/abstract/soilsci;40/4/473.

Wheeler, M. F. 1978. An elliptic collocation-finite element method with interior penalties, *SIAM Journal on Numerical Analysis* **15**:152–161. http://link.aip.org/link/?SNA/15/152/1.

# Thanks

I would like to thank my parents for supporting me for so long, and for not bothering me every week with requests for updates.
I know that has not been easy, especially for my mother.

I would like to thank Marina Seikel for listening to all the petty chatter I threw at her. That meant a lot to me.

I would like to thank Peter Bastian an Kurt Roth for letting me carry out this diploma thesis and for being so patient.

I would like to thank Olaf Ippisch and Christian Engwer for their guidance and for answering my stupid questions even on weekends or vacations. I would especially like to thank them for stopping me when I went on the wrong track, I know I can get a little stubborn sometimes.

I would like to thank Markus Demleitner and Sven Marnach for their financial support.

I would like to thank Christian Engwer, Olaf Ippisch, Marina Seikel, Hendrik Heinl, Nick Kepper and Markus Demleitner for looking at this thesis again and again and pointing out typos, broke englisch, omissions and worse things.
I'm sure some of you are sick of it by now.

I would like to thank Dan Popović and Enkelejda Tafaj for letting me use their office, and for coping with my moaning. I hope I could return as much.

I would like to thank NIИ for Ghosts I. That was my emotional pacemaker.

I would like to thank Linus Torvalds for Linux and Richard Stallman for GNU even though I imagine their egos are a little inflated.
This work would be unthinkable without their work.

*

Finally there has to be a "no thanks" as well. That goes to the Wohngeldamt Stuttgart for occupying almost a full month of my time by requesting a vast amount of certificates — often of no possible use to them — thereby needlessly prying on my and my supporters privacy, for avoiding giving answers, and for probably screwing me.

Erklärung:

Ich versichere, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Unterschrift