Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Discontinuous Galerkin applied to a generic two-phase flow model in a porous medium

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

MASTER OF SCIENCE
in
APPLIED MATHEMATICS

by

Jeroen Wille

Delft, the Netherlands
July 2009

MSc THESIS APPLIED MATHEMATICS

Discontinuous Galerkin applied to a generic two-phase flow model in a porous medium

Jeroen Wille

Delft University of Technology

July 2009

Delft, the Netherlands

Abstract

In this Master thesis we will construct a model for a general two-phase flow in a porous medium. The area of application of the model constructed in this Master thesis is rather large. Two examples are:

1. Geotechnical engineers use this model for pumping oil out of the ground where water is used as a wetting fluid. Adding the water enables the oil to be pumped more easily. In this case the two-phases are mainly oil and water.

2. In a growing number of buildings engineers try to store energy by pumping water into a large basin beneath the building. In these basins there is some fluid, usually gas, present in the spaces where no water is stored. The applications in this thesis are based upon this model.

The model for a two-phase flow in a porous medium consists of two coupled layers, which we use two different numerical methods, one for each layer. Specifically, we use the discontinuous Galerkin method to treat the transport layer and the finite element method to treat the pressure layer. We then demonstrate the effectiveness of our model and methods applied to a simplified heating/cooling problem.

Groundwater extraction may not “fulfill the primary goal of returning ground water to its beneficial uses.” - EPA [6].

Preface

This thesis is the result of my Master of Science research project at the Delft University of Technology, The Netherlands. This project has been carried out at the faculty of Electrical Engineering, Mathematics and Computer Science, abbreviated as EEMCS, at the chair of Numerical Analysis.

I would like to thank my supervisors Fred Vermolen and Jennifer Ryan for their continuous support and good advice, Hans Bruining for his advice on constructing the two-phase flow model, Dennis den Ouden, my roommate at the faculty, for his help writing my report and his knowledge of LaTeX, Lieke de Jong for her helping me with my English writing and all my family and friends for their emotional support.

I hope this Master thesis will be as informative for you as doing the project has been for me.

Delft, July 2009 Jeroen Wille

Contents

Chapter 1

Introduction

In this Master thesis we will construct a model for a general two-phase flow in a porous medium. The area of application of the model constructed in this Master thesis is rather large. Two examples are:

1. Geotechnical engineers use this model for pumping oil out of the ground where water is used as a wetting fluid. Adding the water enables the oil to be pumped more easily. In this case the two-phases are mainly oil and water.

2. In a growing number of buildings engineers try to store energy by pumping water into a large basin beneath the building. In these basins there is some fluid, usually gas, present in the spaces where no water is stored. The applications in this thesis are based upon this model.

The model for a two-phase flow in a porous medium consists of two coupled layers, which we use two different numerical methods, one for each layer.

Investigating the two-phase flow model reveals an interface between the two fluids. Using numerical analysis to solve this model requires us to approximate only one of the two phases. Approximating only one flow requires the ability to handle discontinuities. Using a standard Galerkin, finite difference or finite volume approach will lead to wiggles and spurious oscillations. For finite difference and finite volume methods, limiters and upwind techniques have been constructed see [8]. For the finite element method, Petrov methods, such as streamline upwind Petrov Galerkin, are present, see for example [9]. These correction methods all have a drawback for the implemented numerical method. Discontinuous Galerkin is a new numerical method which is ideal for dealing with discontinuities, although a limiter is still required. The advantage of the discontinuous Galerkin method is that the approximation is reconstructed only per element. For the second layer we will use the finite element method, as this layer has a smooth solution.

We give a brief outline of the type of equations we will use in our two-phase flow model, which is a simplified model. Further details on the model will be given in Chapter 4. The top layer of this model will consist of a linear transport equation of the form

$$u_t + vu_x = 0.$$

For this equation we will use discontinuous Galerkin. As we already mentioned in the paragraph above, in this equation we need to deal with a discontinuity. Other numerical methods are also able to handel discontinuities, but it comes with extra computational cost or a strong decrease of the order of the solution. Discontinuous Galerkin is able to deal with the discontinuities using a high order approximation without deteriorating the order of the solution as heavily near shocks. An other advantage of discontinuous Galerkin is the adaptivity of the mesh and order of the approximation. The second layer will be of the form

$$\nabla \cdot (c\nabla u) = 0.$$

This equation is elliptic, and since current construction of finite element methods are more suited to elliptic problems, we will apply the finite element method for this layer.

The outline of this thesis is as follows: In Chapter 2 a derivation of the Discontinuous Galerkin method in one dimension will be given. Although the two-phase flow model will be constructed in two dimensions, this derivation is presented as an introduction to the discontinuous Galerkin method in two dimensions, which is given in Chapter 3. In Chapter 4 the generic two-phase flow in a porous medium model will be constructed and finally in Chapter 5 an application of the model is presented. In Chapter 6 we summarize our results and present some conclusions and further questions for further research.

Chapter 2

One dimensional discontinuous Galerkin for advection equations

In this section we take a look at the discontinuous Galerkin method, abbreviated by DG, for the scalar conservation law in one dimension. For more information we refer to [4].

2.1 The discontinuous Galerkin discretization

The idea behind the discontinuous Galerkin method is to approximate each cell not only by one unknown but to approximate the value within each cell by some linear combination of piecewise polynomials of degree at most k for some $k \in \mathbb{N}$. Let us examine the following simple model

$$\begin{aligned} u_t + f(u)_x &= 0, \qquad x \in [0,1], \quad t \in (0,T], \qquad (2.1.1)\\ u(x,0) &= u_0(x), \quad \forall x \in [0,1], \qquad (2.1.2)\end{aligned}$$

with periodic boundary conditions.

To derive the weak formulation, we first partition the interval $(0,1)$ by $\{x_{j+1/2}\}_{j=0}^{N}$. Next we define $I_j = \left(x_{j-1/2}, x_{j+1/2}\right)$ and $\Delta_j = x_{j+1/2} - x_{j-1/2}$ for $j = 1, \ldots, N$. Denote by Δx the maximum element size, $\max_{1 \le j \le N} \Delta_j$. Define V_h to be the following finite dimensional space

$$V_h := V_h^k \equiv \left\{ v \in L^1(0,1) : v|_{I_j} \in P^k(I_j), j = 1, \ldots, N \right\}, \qquad (2.1.3)$$

such that V_h is the space of all functions v being piecewise polynomials on an interval I_j, of degree at most k. For each time $t \in (0,T)$ we want our approximation of u to be in V_h. In order to determine this approximation we first multiply (2.1.1) and (2.1.2) by some arbitrary piecewise continuous function v and integrate over the interval I_j to obtain

$$\begin{aligned} \int_{I_j} \frac{\partial}{\partial t} u(x,t)\, v(x)\, dx + \int_{I_j} \frac{\partial}{\partial x} f(u(x,t))\, v(x)\, dx &= 0, \qquad (2.1.4)\\ \int_{I_j} u(x,0)\, v(x)\, dx &= \int_{I_j} u_0(x)\, v(x)\, dx. \qquad (2.1.5)\end{aligned}$$

We apply integration by parts on the second integral in (2.1.4) to remove the spatial derivative of the function $f(u(x,t))$

$$\begin{aligned}\int_{I_j}\frac{\partial}{\partial t}u(x,t)\,v(x)\,dx-\int_{I_j}f(u(x,t))\,\frac{\partial}{\partial x}v(x)\,dx &\quad + \\ f\left(u\left(x_{j+1/2},t\right)\right)v\left(x_{j+1/2}^{-}\right)-f\left(u\left(x_{j-1/2},t\right)\right)v\left(x_{j-1/2}^{+}\right) &= 0, \qquad (2.1.6)\\ \int_{I_j}u(x,0)\,v(x)\,dx &= \int_{I_j}u_0(x)\,v(x)\,dx.(2.1.7)\end{aligned}$$

The functions v are only defined within each interval I_j and therefore we use $x_{j+1/2}^{-}$ to indicate the point $x_{j+1/2}$ approached from the left, and $x_{j-1/2}^{+}$ to indicate the point $x_{j-1/2}$ approached from the right. In Figure 2.1 these points are illustrated.

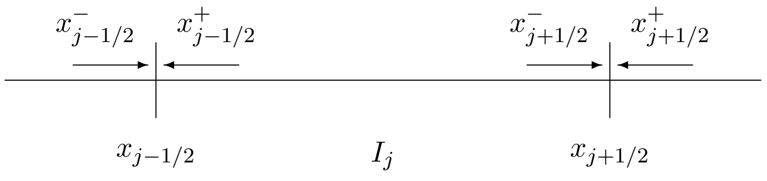

Figure 2.1: Illustration of cell I_j and location of $x_{j-1/2}^{+}$ and $x_{j+1/2}^{-}$.

Next we replace our smooth functions v by test functions $v_h \in V_h$ and replace the exact solution u by the approximation u_h. The function u_h now is discontinuous at the points $x_{j+1/2}$ for $j = 0, \ldots, N$. Therefore, we need to replace the function $f\left(u\left(x_{j+1/2},t\right)\right)$ with a numerical analogue that depends on both $x_{j+1/2}^{-}$ and $x_{j+1/2}^{+}$. For this we define the function h by

$$h(u)_{j+1/2}(t) := h\left(u\left(x_{j+1/2}^{-},t\right),u\left(x_{j+1/2}^{+},t\right)\right), \qquad (2.1.8)$$

where the function h is chosen by the user. The function has to be such that it converges to the entropy solution. We consider two possible choices for h

- The Godunov flux:

$$h^{G}(a,b) := \begin{cases}\min_{a\le u\le b} f(u), & \text{if } a \le b,\\ \max_{a\ge u\ge b} f(u), & \text{if } a > b.\end{cases}$$

- Upwind flux:

$$h^{UW}(a,b) := f(a).$$

For the basis functions we take the Legendre polynomials P_l so that we can exploit their L^2-orthogonality in order to get a diagonal mass matrix

$$\int_{-1}^{1}P_l(x)\,P_m(x)\,dx = \left(\frac{2}{2l+1}\right)\delta_{lm}, \qquad (2.1.9)$$

where δ_{lm} denotes the Kronecker delta function. We also note that we have the equalities $P_l(1) = 1$ and $P_l(-1) = (-1)^l$. In Figure 2.2 we show the first six Legendre polynomials on the interval [-1,1].

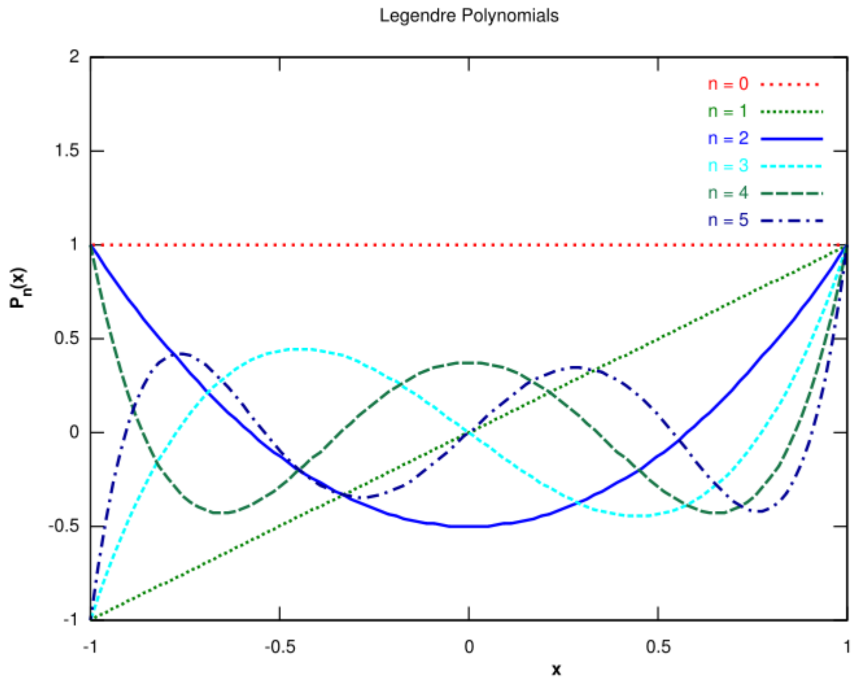

Figure 2.2: The first six Legendre polynomials on [-1,1], where n denotes the order of the polynomial.

Taking $v_h(x) = \varphi_j^m(x)$ and $u_h(x,t) = \sum_{l=0}^{k} u_j^l \varphi_j^l$ with $\varphi_j^l = P_l\left(\frac{2(x-x_j)}{\Delta_j}\right)$. We substitute these basis functions and the equalities $P_l(1) = 1$ and $P_l(-1) = (-1)^l$ in (2.1.6) and (2.1.7) to obtain

$$\begin{aligned}\left(\frac{1}{2m+1}\right)\frac{\partial}{\partial t}u_j^m(t) - \frac{1}{\Delta_j}\int_{I_j} f\left(u_h(x,t)\right)\frac{\partial}{\partial x}\varphi_j^m(x)dx & \\ +\frac{1}{\Delta_j}\left\{h\left(u_h(x_{j+1/2})\right)(t) - (-1)^m h\left(u_h(x_{j-1/2})\right)(t)\right\} &= 0, \qquad (2.1.10)\\ \frac{2m+1}{\Delta_j}\int_{I_j} u_0(x)\varphi_j^m(x)dx &= u_j^m(0), \qquad (2.1.11)\\ \forall j \in \{1,\ldots,N\},\ m \in \{0,\ldots,k\}. &\end{aligned}$$

This gives a system of equations which has to be solved for the variables u_j^m, $\forall j \in \{1,\ldots,N\}$, $\forall m \in \{0,\ldots,k\}$. How this is done will be shown with an example in the next section.

2.2 Application

In order to demonstrate the functionality of the discontinuous Galerkin method, we first apply the given discontinuous Galerkin to a linear conservation equation

$$
\begin{aligned}
u_t + u_x &= 0, && x \in [0,1],\ \ t \in (0,T], && (2.2.1)\\
u(0,t) &= u(1,t), && \forall t \in (0,T], && (2.2.2)\\
u(x,0) &= u_0(x), && \forall x \in [0,1]. && (2.2.3)
\end{aligned}
$$

For the function h we use the upwind flux to weakly enforce continuity. We use a piecewise linear approximation which gives us $k = 1$. Furthermore, we partition the interval $(0,1)$ in $N \in \mathbb{N}$ equally spaced elements and thus we have $(0,1) = (x_{j-1/2}, x_{j+1/2})_{j=1}^{N}$.[1] Since each interval Δ_j has the same length, we have $\Delta_j = \frac{1}{N} = \Delta x,\ \forall j$.
Equations (2.1.10) and (2.1.11) now simplify to

$$
\begin{aligned}
&\left(\frac{1}{2m+1}\right)\frac{\partial}{\partial t}u_j^m(t) - \frac{1}{\Delta x}\sum_{l=0}^{k} u_j^l \int_{I_j} \varphi_j^l \frac{\partial}{\partial x}\varphi_j^m(x)dx \\
&+\frac{1}{\Delta x}\left\{\sum_{l=0}^{k} u_j^l \varphi_j^l(x_{j+1/2}) - (-1)^m \sum_{l=0}^{k} u_{j-1}^l \varphi_{j-1}^l(x_{j-1/2})\right\} = 0, \qquad (2.2.4)\\
&\frac{2m+1}{\Delta x}\int_{I_j} u_0(x)\varphi_j^m(x)dx = u_j^m(0), \qquad (2.2.5)\\
&\forall j \in \{1,\dots,N\},\ m \in \{0,\dots,k\}.
\end{aligned}
$$

We note that we do not have to use the notation $x^+_{j-1/2}$ etc. for the boundary points since the Legendre polynomials are defined on the boundary.

We will solve this system by determining the initial coefficients first. In order to do so we determine the integrals $\int_{I_j} u_0(x)\varphi_j^m(x)dx$ exactly. Having determined the initial values we can derive the mass and stiffness matrix. For the mass matrix we simply have

$$
M_{ml} = \frac{1}{2m+1}\delta_{ml}. \qquad (2.2.6)
$$

Due to the spatial integral we obtain the following stiffness matrix

$$
\begin{aligned}
S_{ml} &= \frac{1}{\Delta x}\int_{I_j} \varphi_j^l \frac{\partial}{\partial x}\varphi_j^m dx,\\
&\Rightarrow\\
S &= \begin{bmatrix} 0 & 0 \\ \frac{2}{\Delta x} & 0 \end{bmatrix}. \qquad (2.2.7)
\end{aligned}
$$

Next we have two matrices, A and B, corresponding to the flux of the current cell and the previous cell. We have $A\mathbf{u}_j + B\mathbf{u}_{j-1}$:

$$
\frac{1}{\Delta x}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\mathbf{u}_j + \frac{1}{\Delta x}\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}\mathbf{u}_{j-1},\quad \mathbf{u}_j = \begin{pmatrix} u_j^0 \\ u_j^1 \end{pmatrix}. \qquad (2.2.8)
$$

[1] This is the same as writing $(0,1) = \bigcup_{j=1}^{N}(x_{j-1/2}, x_{j+1/2})$.

Finally we apply a simple Euler forward scheme for the time derivative to obtain

$$M\mathbf{u}_j(t_{new}) = (M - \Delta tA + \Delta tS)\,\mathbf{u}_j(t_{old}) - \Delta tB\mathbf{u}_{j-1}(t_{old}). \tag{2.2.9}$$

This system has the size of the order of the approximation. The mass matrix is the same for every element. Therefore one defines the inverse immediately, rather than the inverse mass matrix per element and per time step. This will give us

$$M^{-1} = (2m+1)\delta_{ml}, \tag{2.2.10}$$

and we solve per element and per time step

$$\mathbf{u}_j(t_{new}) = M^{-1}(M - \Delta tA + \Delta tS)\,\mathbf{u}_j(t_{old}) - M^{-1}\Delta tB\mathbf{u}_{j-1}(t_{old}). \tag{2.2.11}$$

2.3 Comparison to standard Galerkin

In this section we will make a small comparison of the discontinuous Galerkin method and the finite element method, abbreviated by FEM, in one dimension.

The most obvious difference between DG and FEM is the location of the unknowns and the reconstruction of the solution using the approximated values. Since DG is discontinuous over the elements the solution is represented as piecewise polynomials although the initial condition is continuous. This is shown in Figure 2.3. In this figure we show what happens when DG and the FEM determine the initial values for the coefficients of the solution. As initial condition we used

$$u_0(x) = 5 + \sin(2\pi x). \tag{2.3.1}$$

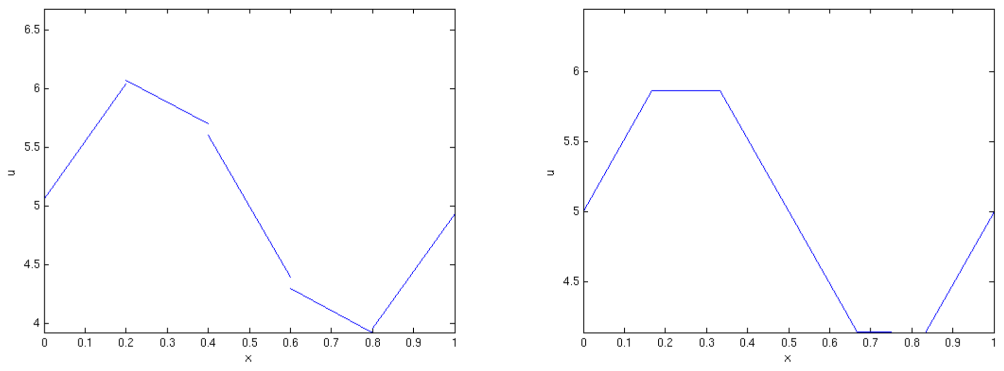

Figure 2.3: Initial values for coefficients with initial condition (2.3.1) for discontinuous Galerkin (left) and the finite element method (right) using linear approximation and five elements.

One can clearly see that the initial solution is not continuous anymore when applying discontinuous Galerkin. We have only taken five elements on purpose, since this effect will not be noticeable anymore when using a large number of elements. The advantage of using discontinuous Galerkin is shown in Figure 2.4. Here we used the discontinuous initial condition

$$u_0(x) = \begin{cases} 5, & \text{if } x \leq 0.5; \\ 0, & \text{otherwise.} \end{cases} \tag{2.3.2}$$

Furthermore we used six elements to have to the discontinuity exactly between two elements. In this figure we see that using discontinuous Galerkin will result in being able to reconstruct

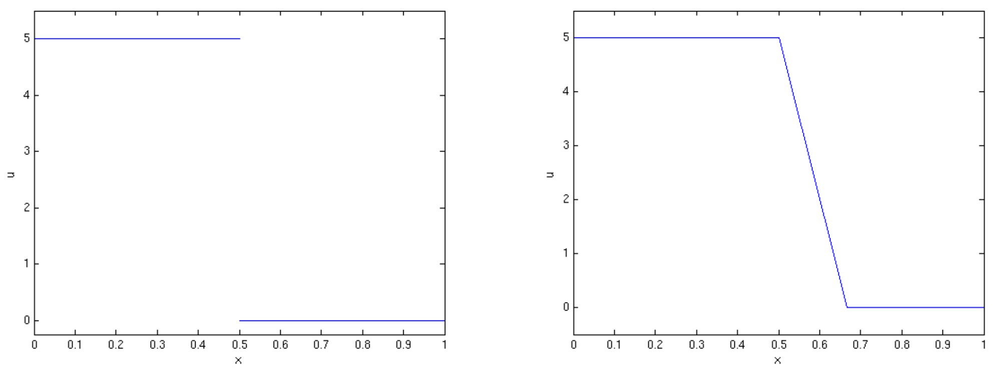

Figure 2.4: Initial values for coefficients with initial condition (2.3.2) for discontinuous Galerkin (left) and the finite element method (right) with linear elements.

the initial condition perfectly, whereas the finite element method has a large slope on the interval $[0.5, 0.66]$, depending on the element size.

We will also note some important differences between discontinuous Galerkin and the finite element method with respect to the discretization. First of all there is a difference in the spaces for the test- and basis function we use. In finite elements the space of test- and basis functions does not need to be the same, whereas this is the case in discontinuous Galerkin. Furthermore, there is a difference in the basis functions. In DG we use piecewise polynomial basis functions per element, whereas in finite elements the basis functions are defined as piecewise polynomial on the entire region, although they will be equal to zero for the majority of the domain. Finally, we note that with the FEM, when doing the discretization, we obtain a sum of the solution in the endpoints of the region where we have to apply boundary conditions, whereas with DG we will have this sum for each element where we use some flux function.

Finishing this section we will make some remarks on computational cost. Since this is highly dependent on the programmer and the language used, we will only note the differences but we will not draw any conclusions. We have seen that the mass-matrix in DG is independent of the size of the element and the solution. Therefore we have to determine the inverse of this matrix only once because this matrix will be of size $(k+1) \times (k+1)$, this will not have a very high computational cost. Using finite elements the matrix will be much larger, having a size equal to the number of nodes used. However, this matrix is independent of the solution as well, and therefore needs to be inverted only once as well, if one uses a explicit time integration method. One can also use a iterative method. However, this needs to be applied

at every time step.
Using the discontinuous Galerkin method, the solution at the new time step of each element is independent of the solution at the new time step of the other elements. Therefore one can solve system (2.2.11) more easily in parallel. However, since we always need an explicit time integration method for DG, we will always have to satisfy stability criteria.

2.4 Shock Detection

Upon taking a discontinuous initial condition when using the discontinuous Galerkin method applied to a linear first order hyperbolic equation we will have wiggles near discontinuities that develop over time. In practice this may lead to nonlinear instabilities and to nonphysical solutions like negative concentrations or pressures, see for example Figure 2.5. Since limiting reduces the quality of the solution in smooth regions, we try to determine only these regions where limiting is needed. We will construct this shock detector based on [3].

The derivation of the shock detector does not depend on the number of dimensions. Therefore we will derive the procedure for multi dimensions and indicate cells by Ω instead of I, the latter being convenient for the one dimensional case only. We also use Q for the approximation. Let us look at a given problem on a certain cell Ω_j. We partition the boundary $\partial\Omega_j$ of this cell into two parts. The first part is, denoted by $\partial\Omega_j^-$, where we have inflow and therefore $(v \cdot n) < 0$. The other part is where we have outflow, $(v \cdot n) > 0$, which we denote by $\partial\Omega_j^+$. According to [3] smooth solutions of hyperbolic conservation laws show strong superconvergence phenomena at outflow boundaries such that

$$\frac{1}{|\partial\Omega_j^+|} \int_{\partial\Omega_j^+} (Q_j - q) \ d\Gamma = \mathcal{O}(h^{2k+1}). \tag{2.4.1}$$

Here, $|\partial\Omega_j^+|$ denotes the length/area of $\partial\Omega_j^+$. Further, k is again the order of the approximation and h is the size of an element. Note that q is the exact solution of the equation and Q_j is the discontinuous Galerkin value of q on Ω_j. We will use this information to detect locations of shocks. For example consider a jump in Q_j across $\partial\Omega_j^-$. We split up this integral into an integral of the inflow on the element and an integral of the outflow of the neighboring element. We do so since the order of both contributions is known. Hence, we examine

$$\mathbf{I}_j = \int_{\partial\Omega_j^-} (Q_j - Q_{nbj}) \ d\Gamma = \int_{\partial\Omega_j^-} (Q_j - q) \ d\Gamma + \int_{\partial\Omega_{nbj}^+} (q - Q_{nbj}) \ d\Gamma. \tag{2.4.2}$$

In this equation Q_{nbj} stands for the value of Q in a neighboring element of Ω_j with common boundary $\partial\Omega_{j,nbj}$. We know from equation (2.4.1) that the second integral in this equation is $\mathcal{O}(h^{2k+2})$. Furthermore we know that the first integral across the inflow boundary is $\mathcal{O}(h^{k+2})$ so $\mathbf{I}_j$ is $\mathcal{O}(h^{k+2})$ for smooth solutions on $\partial\Omega_j^-$ as shown in [1]. However, if q is discontinuous near $\partial\Omega_j$ then $q - Q_j$ and/or $q - Q_{nbj}$ will be $\mathcal{O}(1)$. Using this information we construct a discontinuity detector by normalizing $\mathbf{I}_j$ to some convergence rate and the solution Ω_j:

$$\mathcal{I}_j = \frac{\left| \int_{\partial\Omega_j^-} (Q_j - Q_{nbj}) \ d\Gamma \right|}{h^{(k+1)/2} |\partial\Omega_j^-| \|Q_j\|}. \tag{2.4.3}$$

We know that in smooth regions $\mathbf{I}_j \to 0$, and so $\mathcal{I}_j \to 0$ as well, if $h \to 0$ or $k \to \infty$. However near a discontinuity we have $\mathcal{I}_j \to \infty$. The discontinuity detection scheme we use is then

$$\begin{cases} \mathcal{I}_j > 1 & \Rightarrow \quad q \text{ is discontinuous,} \\ \mathcal{I}_j < 1 & \Rightarrow \quad q \text{ is smooth.} \end{cases} \tag{2.4.4}$$

2.5 Application of the shock detector to a scalar conservation equation

We will apply this shock detector to equation (2.2.4) to test the performance of this strategy. We note the following:

- The boundary $\partial\Omega_j^-$ consists of one point, $x_{j-1/2}$, hence, the term $|\partial\Omega_j^-|$ drops out;
- For the norm, we take the element average, as shown in equation (2.5.1);
- The integral is an integral over one point, so the integral reduces to $Q_j(x_{j-1/2}^-) - Q_{j-1}(x_{j-1/2}^+)$;

For the element average we have the following

$$\begin{aligned} \|Q_j\| &= \frac{1}{\Delta_j}\int_{x_{j-1/2}}^{x_{j+1/2}} Q_j(x)dx, \\ &= \frac{1}{\Delta_j}\int_{x_{j-1/2}}^{x_{j+1/2}} u_j^0 + u_j^1\frac{2(x-x_j)}{\Delta_j}dx, \\ &= \frac{1}{\Delta_j}\int_{x_{j-1/2}}^{x_{j+1/2}} u_j^0 dx, \\ &= u_j^0. \end{aligned} \tag{2.5.1}$$

So we see the norm reduces to u_j^0, when taking only linear piecewise polynomials.[2]

As a first test we take the following initial condition:

$$u(x,0) = \begin{cases} 5, & \text{if } x < 0.5, \\ 1, & \text{otherwise.} \end{cases} \tag{2.5.2}$$

After 100 time steps we obtain the results in Figure 2.5. The red + signs in this figure indicate where the shock detection method indicates a discontinuity. This figure clearly shows us that taking a discontinuous initial condition leads to wiggles, and furthermore it shows us that the shock detection method quite accurately indicates the positions where limiting is needed, for simple examples. However this shock detection method also has a drawback. When we take an initial condition with some large, but smooth slopes, the detector may indicate some slopes as discontinuities. The limiter method may adjust the coefficients where it is not needed. Figure 2.6 is a nice example of this case. In this example we have chosen the following initial condition:

$$u(x,0) = 5 + \sin(12\pi x). \tag{2.5.3}$$

[2]This shows us that using only constant polynomials will in fact give us the finite volume method.

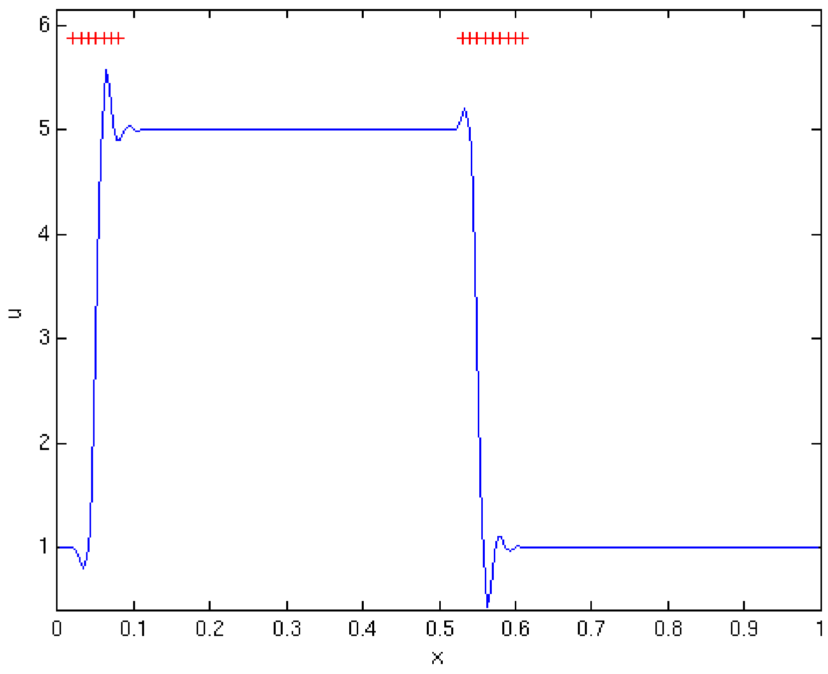

Figure 2.5: Solution of (2.2.4) with initial condition (2.5.2) after 100 time steps.

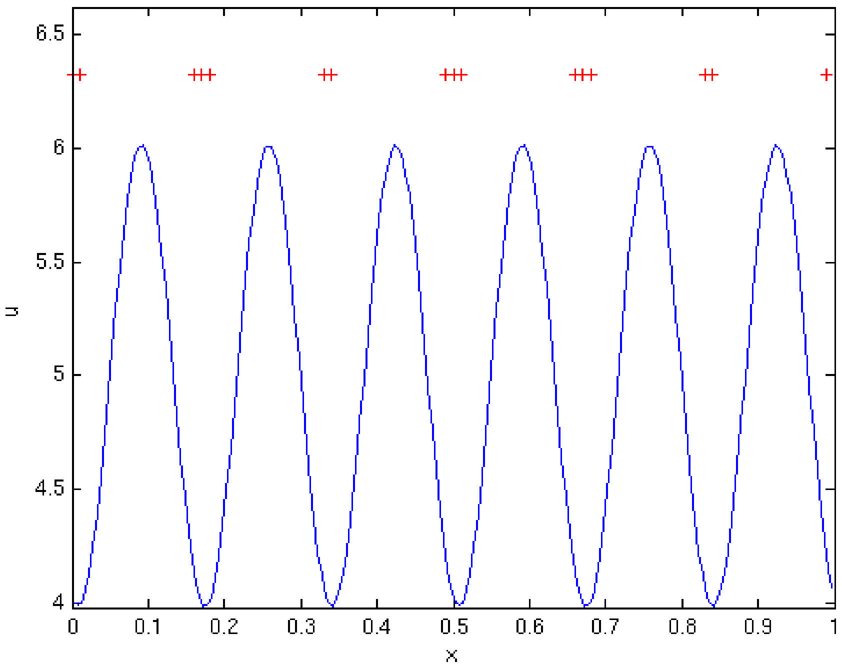

Figure 2.6: Solution of (2.2.4) with initial condition (2.5.3) after 100 time steps.

It is quite clear that the large slopes are indicated as shocks. At first sight however, it seems rather strange that only the bottom peaks are indicated as discontinuities, and not the upper peaks as well. This is due to the definition of the indicator $\mathcal{I}_j$. In this definition we divide by the norm of Q_j and in one dimension this is the same as the absolute value of the zeroth coefficient. At the top peaks this value is 1.5 times as large as at the bottom peaks, so the indicator value is here 1.5 times lower than the bottom peaks. This factor leads to not indicating the top peaks as shocks.

2.6 Limiter

Now that we have a method to determine shocks, we need a method to limit near these shocks. We implement the limiter discussed in [7]. The idea behind this limiter is to start limiting, in each cell where it is needed, at the highest coefficient. The limiter stops if the limited value is the same as the coefficient. We determine the limited value $\tilde{u}_j^l$ by

$$\tilde{u}_j^l = \operatorname{minmod}\left(u_j^l, u_{j+1}^{l-1} - u_j^{l-1}, u_j^{l-1} - u_{j-1}^{l-1}\right), \tag{2.6.1}$$

where the minmod function is defined by

$$\operatorname{minmod}(a,b,c) := \begin{cases} \operatorname{sgn}(a)\min(|a|,|b|,|c|), & \text{if } \operatorname{sgn}(a) = \operatorname{sgn}(b) = \operatorname{sgn}(c), \\ 0, & \text{otherwise,} \end{cases} \tag{2.6.2}$$

We compare the value u_j^l to a forward and backward difference of the values of one order lower. We do not need to divide these differences by the distance, since this is already done in the derivation of the discontinuous Galerkin method. We start in each cell at $l = k$ and stop the limiting procedure if either the limited value is the same as the original coefficient, $\tilde{u}_j^l = u_j^l$, or we have arrived at the lowest order, so $l = 0$.

2.7 Overview

We give a brief overview of how the different methods in this chapter are used. We present this in a pseudo-code algorithm as in Algorithm 2.7.1. In this algorithm the solution u is stored as a matrix. The columns indicate the values for the cells. The rows indicate the coefficients of the different orders. By $u(i,:)$ we mean all coefficients of element i. This algorithm is implemented at every time step. As one can see this algorithm is applicable to every order.

We start by putting all the coefficients into a new matrix called u_{new}. The limited values will be based on u itself and will be stored in u_{new}. In this way, each cell will only use the data from applying the limiter. Next we set the coefficients of the zeroth element to the same coefficients of the last element. The zeroth element does not exist, but we need to set this, in order to be able to apply the limiter on the first element. Similarly we set the $(n+1)$-th coefficients equal to the first coefficients, because we have periodic boundary conditions.

Then we start an iteration over the elements to determine the coefficients on the new time step. On line five and six we set the approximated value of the cell and its left neighbor equal to zero. We calculate the approximation at the inflow boundary. Using the properties

Algorithm 2.7.1 Pseudo-code algorithm for one dimensional shock detection and limiting.

```
$u_{new} = u$
$u(0,:) = u(n,:)$
$u(n+1,:) = u(1,:)$
**for** $j = 1$ to $n$ **do**
  $u_j = 0;$
  $u_{j-1} = 0;$
  **for** $l = 0$ to order **do**
    $u_j = u_j + (-1)^l u(j,l)$
    $u_{j-1} = u_{j-1} + u(j-1,l)$
  **end for**
  $normu_j =$ norm$(u(j,:))$
  $radius = (h/2)^{((order+1)/2)}$
  $I_j =$ abs$(u_j - u_{j-1})/(radius \cdot normu_j)$
  **if** $I_j > 1$ **then**
    **for** $l =$ order to 1 **do**
      $repl =$ minmod$(u(j,l), u(j+1,l-1) - u(j,l-1), u(j,l-1) - u(j-1,l-1))$
      **if** $repl \neq u(j,l)$ **then**
        $u_{new}(j,l) = repl$
      **else**
        break
      **end if**
    **end for**
  **end if**
**end for**
$u = u_{new}$
```

$P_l(1) = 1$ and $P_l(-1) = (-1)^l$ of Legendre polynomials.

Next we determine the norm by some function. This is done to keep the algorithm applicable for higher orders. We also determine the factor radius, where h is just the element length. With this data we can determine the indicator $\mathcal{I}_j$. On line 14 we check whether limiting is needed. If so, we iterate over the coefficients starting at the highest. We apply the minmod function and check whether this results is different. If so we save the new value in u_{new} and continue. Otherwise we stop and go to the next element. When all limiting is done we restore the matrix u with all limited values.

2.8 Results

We apply this method to the simple scalar conservation law as in equation (2.2.4) and examine the two separate cases of Section 2.5. For the discontinuous initial condition we see the results in Figure 2.7. We see that the limiter works quite well. It is expected that with this limiter there will still be some smearing, but the wiggles are completely gone and the smearing is within the boundaries. Figure 2.8 nicely illustrates the drawback of this method. In the right figure we clearly see that the bottom peaks are limited and therefore these peaks look somewhat flattened out.

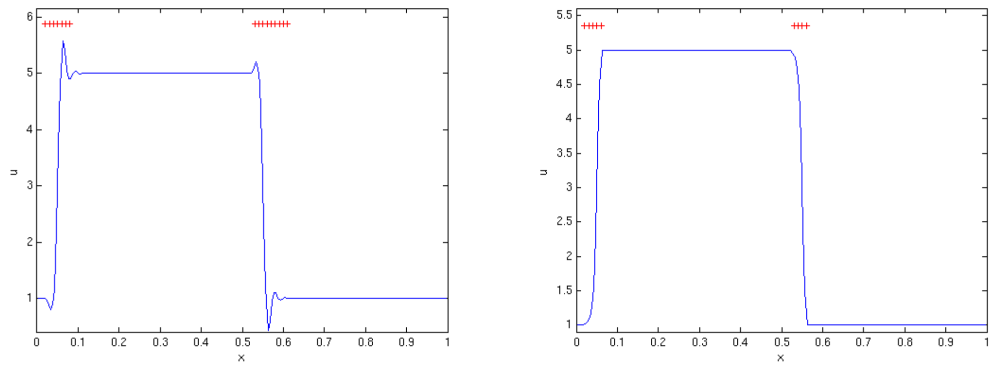

Figure 2.7: Solution of (2.2.4) with initial condition (2.5.2) after 100 time steps without(left) and with(right) limiting.

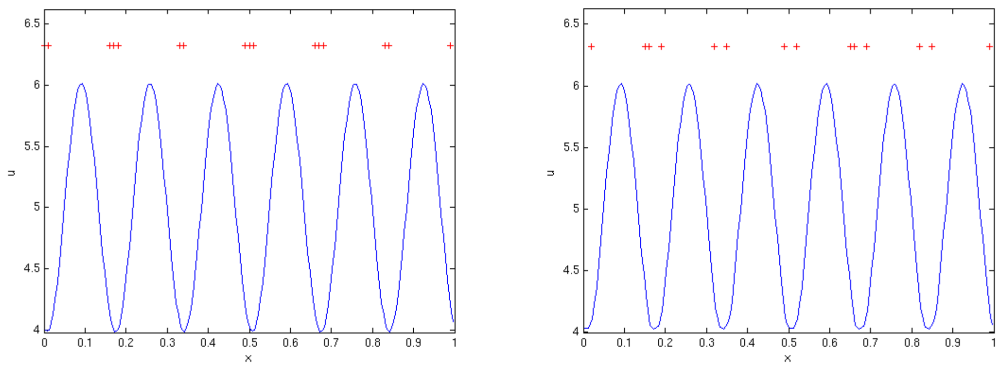

Figure 2.8: Solution of (2.2.4) with initial condition (2.5.3) after 100 time steps without(left) and with(right) limiting.

Chapter 3

Two dimensional discontinuous Galerkin for advection equations

In this chapter we will investigate the Discontinuous Galerkin method applied to the advection equation in two dimensions.

3.1 The discontinuous Galerkin discretization

We examine the following system on the unit square:

$$u_t + \nabla \cdot (\mathbf{v}u) = 0, \qquad \mathbf{x} \in (0,1) \times (0,1),\ t \in (0,T], \tag{3.1.1a}$$

$$u(x,0,t) = u(x,1,t), \quad t \in (0,T] \tag{3.1.1b}$$

$$u(0,y,t) = u(1,y,t), \quad t \in (0,T] \tag{3.1.1c}$$

$$u(x,y,0) = u_0(x,y), \quad x,y \in (0,1) \tag{3.1.1d}$$

with $\mathbf{v} = (v_1, v_2)^T$ some constant velocity vector. This is a linear version of $u_t + \nabla \cdot \mathbf{f}(u) = 0$. For a derivation of this general equation we refer to [4]. To approximate the solution we first determine a triangulation of the unit square. Then, we multiply equation (3.1.1a) by some test function v and we integrate over an element Ω_j to obtain

$$\frac{d}{dt}\int_{\Omega_j} uv d\Omega + \int_{\Omega_j} (\nabla \cdot (\mathbf{v}u))v d\Omega = 0. \tag{3.1.2}$$

We denote by V_h the finite dimensional space of all piecewise linear functions on Ω_j. We substitute $v = v_h \in V_h$ and $u = u_h \in V_h$ and we apply *Green's theorem* to obtain

$$\frac{d}{dt}\int_{\Omega_j} u_h v_h d\Omega - \int_{\Omega_j} \mathbf{v}u_h \cdot \nabla v_h d\Omega + \int_{\partial\Omega_j} v_h u_h \mathbf{v} \cdot \mathbf{n} d\Gamma = 0, \quad \forall v_h \in V_h. \tag{3.1.3}$$

We note that we can replace the integral over the boundary $\partial\Omega_j$ by a sum of integrals over the edges

$$\int_{\partial\Omega_j} v_h u_h \mathbf{v} \cdot \mathbf{n}\ d\Gamma = \sum_{e \in \partial\Omega_j} \int_e v_h u_h \mathbf{v} \cdot \mathbf{n}_e\ d\Gamma. \tag{3.1.4}$$

In this equation $\mathbf{n}_e$ denotes the outward normal vector on edge e. We remark that u_h is discontinuous at the element boundaries and therefore equation (3.1.4) is not well defined. Therefore we need to replace the term $u_h\mathbf{v}\cdot\mathbf{n}_e$ in this integral by the numerical flux h_e

$$h_e = h_e\left(u_h\left(t, x^{int(\Omega_j)}, y^{int(\Omega_j)}\right), u_h\left(t, x^{ext(\Omega_j)}, y^{ext(\Omega_j)}\right)\right), \tag{3.1.5}$$

which we will define in equation (3.1.10). We now have

$$\frac{d}{dt}\int_{\Omega_j} u_h v_h d\Omega - \int_{\Omega_j} \mathbf{v} u_h \nabla v_h d\Omega + \sum_{e\in\partial\Omega_j}\int_e v_h h_e \, d\Gamma = 0, \quad \forall v_h \in V_h. \tag{3.1.6}$$

Within element Ω_j, we substitute $u_h(x,y,t) = \sum_{k=1}^3 u_{j,k}(t)\varphi_k(x,y)$ and $v_h(x,y) = \varphi_i(x,y)$, $i = 1,\ldots,3$ where $\varphi_i(x,y)$ is the linear function which takes the value one at the midpoint m_i of edge i and the value zero at midpoints of the other two edges.

We still need to replace the integrals by quadrature rules. For the boundary integral we apply the following Gaussian quadrature rule:

$$\int_{-1}^1 g(x)dx \approx g\left(\frac{-1}{\sqrt{3}}\right) + g\left(\frac{1}{\sqrt{3}}\right). \tag{3.1.7}$$

We parameterize edge $e = (e_{\min}, e_{\max})$ by $e = \frac{e_{\min}+e_{\max}}{2} + s\left(\frac{e_{\max}-e_{\min}}{2}\right)$ for $s \in [-1,1]$. Using this parametrization and the quadrature (3.1.7) we obtain for a general edge $e = (e_{\min}, e_{\max})$

$$\begin{aligned}\int_e g(\mathbf{x})d\mathbf{x} &\approx \frac{|e|}{2} g\left(\frac{e_{\min}+e_{\max}}{2} - \frac{1}{\sqrt{3}}\left(\frac{e_{\max}-e_{\min}}{2}\right)\right)\\ &+ \frac{|e|}{2} g\left(\frac{e_{\min}+e_{\max}}{2} + \frac{1}{\sqrt{3}}\left(\frac{e_{\max}-e_{\min}}{2}\right)\right).\end{aligned} \tag{3.1.8}$$

For the integral over the interior of an element we use the three midpoint rule

$$\int_{\Omega_j} g(x,y)d\Omega \approx \frac{|\Omega_j|}{3}\sum_{i=1}^3 g(m_i), \tag{3.1.9}$$

where $|\Omega_j|$ denotes the area of element Ω_j. We now define the numerical flux function h_e. We use the simple Lax-Friedrichs flux

$$h_e(a,b) = \frac{1}{2}[a\ \mathbf{v}\cdot\mathbf{n}_e + b\ \mathbf{v}\cdot\mathbf{n}_e - \alpha_e\cdot(b-a)]. \tag{3.1.10}$$

In this flux function we take for a the approximated value of u at some position x_{ref} on the edge. For b we take the approximated value in the neighboring element at the same position x_{ref}. The constant α_e is the numerical viscosity constant which should be an estimate of the largest eigenvalue of the Jacobian $\frac{\partial}{\partial u}u_h(t,x,y)\mathbf{v}\cdot\mathbf{n}_e$ for (t,x,y) in a neighborhood of edge e. Since we have $\frac{\partial}{\partial u}u_h(t,x,y)\mathbf{v} = \mathbf{v}$, we take α_e to be the largest eigenvalue of $\mathbf{v}\times\mathbf{n}_e$ in absolute value.

For the mass matrix we have

$$M_{ij} = \int_{\Omega_j}\varphi_k\varphi_i d\Omega = \frac{|\Omega_j|}{3}\delta_{ik}. \tag{3.1.11}$$

For the stiffness matrix we obtain

$$S_{ij} = \int_{\Omega_j} \mathbf{v}\varphi_k \nabla\varphi_i d\Omega = \frac{|\Omega_j|}{3}\mathbf{v}\cdot\nabla\varphi_i \sum_{l=1}^{3} \varphi_k(m_l) = \frac{|\Omega_j|}{3}\mathbf{v}\cdot\nabla\varphi_i. \tag{3.1.12}$$

For the right hand side we obtain

$$\begin{aligned} F_i &= -\sum_e \frac{|e|}{4}\left[u_h\left(t, x_{e_1}^{int(\Omega_j)}\right)\mathbf{v}\cdot\mathbf{n}_e + u_h\left(t, x_{e_1}^{ext(\Omega_j)}\right)\mathbf{v}\cdot\mathbf{n}_e\right]\varphi_i(x_{e_1}) \\ &- \frac{|e|}{4}\alpha_e\cdot\left(u_h\left(t, x_{e_1}^{ext(\Omega_j)}\right) - u_h\left(t, x_{e_1}^{int(\Omega_j)}\right)\right)\varphi_i(x_{e_1}) \\ &+ \frac{|e|}{4}\left[u_h\left(t, x_{e_2}^{int(\Omega_j)}\right)\mathbf{v}\cdot\mathbf{n}_e + u_h\left(t, x_{e_2}^{ext(\Omega_j)}\right)\mathbf{v}\cdot\mathbf{n}_e\right]\varphi_i(x_{e_2}) \\ &- \frac{|e|}{4}\alpha_e\cdot\left(u_h\left(t, x_{e_2}^{ext(\Omega_j)}\right) - u_h\left(t, x_{e_2}^{int(\Omega_j)}\right)\right)\varphi_i(x_{e_2}), \end{aligned} \tag{3.1.13}$$

with for edge $e = (e_{\min}, e_{\max})$

$$e_1 = \frac{e_{\min}+e_{\max}}{2} - \frac{1}{\sqrt{3}}\left(\frac{e_{\max}-e_{\min}}{2}\right), \tag{3.1.14}$$

$$e_2 = \frac{e_{\min}+e_{\max}}{2} + \frac{1}{\sqrt{3}}\left(\frac{e_{\max}-e_{\min}}{2}\right). \tag{3.1.15}$$

In Figure 3.1 we show the location of e_1 and e_2 based on $e_{\min}$ and $e_{\max}$ for a general triangle.

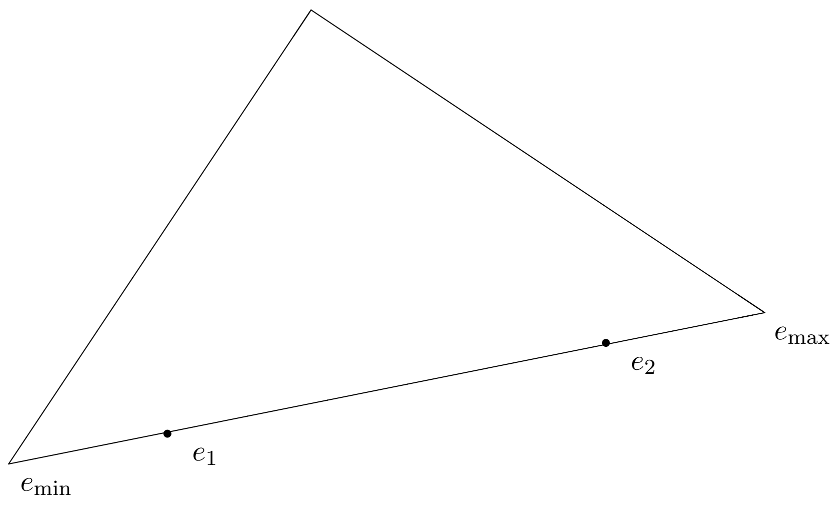

Figure 3.1: Location of e_1 and e_2 for a general edge $e = (e_{\min}, e_{\max})$.

Using Euler forward we need to solve the system

$$M\frac{\underline{u}^{k+1} - \underline{u}^k}{\Delta t} = S\underline{u}^k + F^k, \tag{3.1.16}$$

for each element in our mesh. If one takes a higher order approximation, then the time integration method needs to be of a higher order as well.

3.2 Comparison to standard Galerkin

Similar to Section 2.3, where we made a comparison between discontinuous Galerkin and the finite element method in one dimension, we will make a comparison in two dimensions in this section.

Again we start by stating the difference in the location of the unknowns. With the FEM the unknowns are located at the element vertices, consisting of linear basis functions, whereas with DG the unknowns are located at the midpoints of the edges. For a quadratic approximation we need to add more unknowns to each element. With DG these unknowns are added at the vertices of the elements, whereas with the finite element method we add these unknowns at the midpoints. So for a quadratic approach the locations of the unknowns are identical for DG and FEM.

Similar to one dimension we also have a difference in the basis- and test functions. For finite elements these basis functions are again defined on the entire region and do not have to be from the same space as the test functions. Petrov-Galerkin methods are all about using different spaces for the basis- and test functions. The Streamline Upwind Petrov Galerkin method may even be the most common method. For discontinuous Galerkin, the two spaces are the same and the functions are defined only on the element itself. Again we have discontinuity over the elements for the DG method. This can be seen for instance in Figure 3.2 in the left picture. This picture also illustrates a drawback of DG when reconstructing the solution in order to make a plot. We have determined the solution at the vertices and plotted the solution per element. Having two midpoints with the value zero and one midpoint with the value one, the vertices will have values $1, 1, -1$. Having just one midpoint with the value zero, and two midpoints have the value one, the vertices will have values $2, 0, 0$. So the reconstructed solution appears to give us physically incorrect values, whereas the discontinuous Galerkin method gives us values between zero and one.

Also in two dimension we use piecewise polynomial basis functions per element with DG, whereas in finite elements the basis functions are defined as piecewise polynomial on the entire region, although they will be equal to zero for the majority of the domain. Furthermore we have a integral over the boundary of the region where we apply boundary conditions versus a sum over the edges where we use some flux function.

Making a comparison based on computational cost we will also have the same differences in global, with some minor adjustments. The mass matrix does depend on the size of an element in two dimensions. However, this can be ignored by dividing equation (3.1.16) by Ω_j. Using this the mass matrix is no longer dependent on the size of the element and can be inverted very cheaply which needs to be done only once. Again the system needs to be solved for each element, which can be done in parallel.

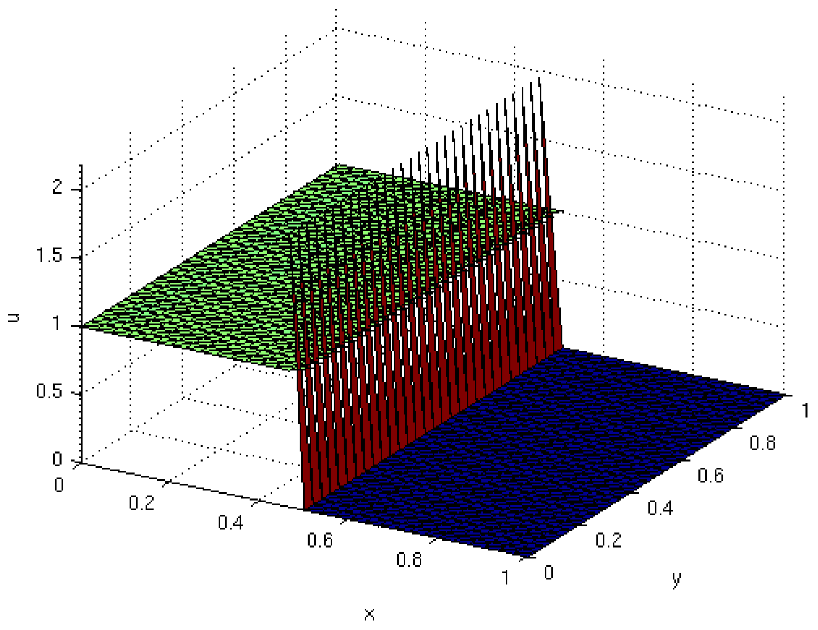

Figure 3.2: Illustration of discontinuous initial condition with discontinuous Galerkin.

3.3 Shock detection

As in one dimension, we want a method that is able to detect the location of shocks such that we can apply our limiter only in these regions. In Section 2.4 we discussed a shock detector for a general dimension. Therefore we use the indicator in equation (2.4.3) given by

$$\mathcal{I}_j = \frac{\left|\int_{\partial\Omega_j^-} (u_j - u_{nbj}) \ d\Gamma\right|}{h^{(k+1)/2}|\partial\Omega_j^-|\|u_j\|}. \tag{3.3.1}$$

For h we take the radius of the circumscribed circle of element Ω_j. For the norm we take the maximum norm of the local solution maxima at the integration points. These integration points are the midpoints of the boundaries. We know that at midpoint m_i we have $u_j(x_i, y_i, t) = \sum_{k=1}^{3} u_{j,k}(t)\varphi_k(x_i, y_i) = u_{j,i}(t)$. This reduces the norm to

$$\|u_j\| = \max_{1\leq i\leq 3} |u_{j,i}|. \tag{3.3.2}$$

The integral in the numerator in (3.3.1) can be calculated exactly upon using approximations for the variables u_j and u_{nbj}. For an edge e_i, where we have inflow, we determine the integral as follows:

$$\begin{aligned}\int_{e_i} (u_j(x,y,t) - u_{nbj}(x,y,t)) \ d\Gamma &= \int_{e_i} u_j(x,y,t) \ d\Gamma - \int_{e_i} u_{nbj}(x,y,t) \ d\Gamma, \\ &= \int_{e_i} \sum_{k=1}^{3} u_{j,k}(t)\varphi_k(x,y) \ d\Gamma - \int_{e_i} \sum_{k=1}^{3} u_{nbj,k}(t)\varphi_k(x,y) \ d\Gamma.\end{aligned}$$

Two of these basis functions have the value zero at m_i, and since they are linear, the integral over the edge will be equal to zero. The third basis function, $\varphi_i(x,y)$, will have the value one at m_i and hence the integral will be equal to $u_{j,i}|e_i|$. Hence, finally we have

$$\int_{e_i} (u_j(x,y,t) - u_{nbj}(x,y,t)) \; d\Gamma = |e_i|(u_{j,i}(t) - u_{nbj,i}(t)). \tag{3.3.3}$$

3.4 Limiter

Using the shock detector in the previous section allows us to limit the approximations of elements only where limiting is needed. We will derive this limiter using [4]. This limiter is based on the approximated solution instead of purely on the coefficients. If we were to construct a limiter based on the coefficients like in one dimension, we need a higher order approximation.

We start by examining Figure 3.3[1], in which we deal with element Ω_0 and its neighbors Ω_1, Ω_2 and Ω_3. In this figure, c_i denotes the element center of Ω_i.

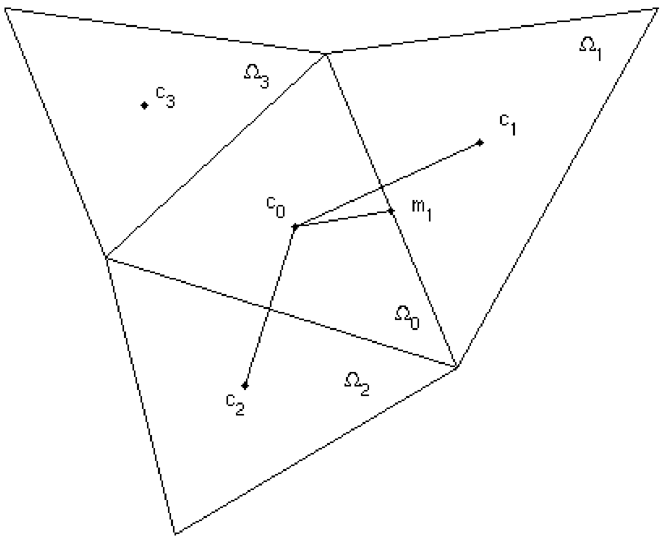

Figure 3.3: Illustration of limiting for element Ω_0

We have

$$m_1 - c_0 = \alpha_1(c_1 - c_0) + \alpha_2(c_2 - c_0), \tag{3.4.1}$$

for some coefficients α_1 and α_2 which only depend on the midpoint m_1 for a given geometry. Since u_h is linear per element we have

$$u_h(m_1) - u_h(c_0) = \alpha_1(u_h(c_1) - u_h(c_0)) + \alpha_2(u_h(c_2) - u_h(c_0)). \tag{3.4.2}$$

[1]This figure has been constructed after a similar figure in [4].

We define the cell averages $\bar{u}_{\Omega_j}$ by

$$\bar{u}_{\Omega_j} = \frac{1}{|\Omega_j|}\int_{\Omega_j} u_h d\Omega = u_h(c_j), \quad j = 0, 1, 2, 3. \tag{3.4.3}$$

We now have

$$\begin{aligned} \tilde{u}_h(m_1, \Omega_0) &= u_h(m_1) - \bar{u}_{\Omega_0}, &(3.4.4)\\ \Delta\bar{u}(m_1, \Omega_0) &= \alpha_1(\bar{u}_{\Omega_1} - \bar{u}_{\Omega_0}) + \alpha_2(\bar{u}_{\Omega_2} - \bar{u}_{\Omega_0}). &(3.4.5)\end{aligned}$$

We see that $\bar{u}_h = u_h - \bar{u}_{\Omega_0}$ is the approximated value minus the cell average. Since we use a linear approximation this should consist of the slopes of the solution.

For $(x, y) \in \Omega_0$ we can now write

$$u_h(x, y) = \sum_{i=1}^{3} u_h(m_i)\varphi_i(x, y) = \bar{u}_{\Omega_0} + \sum_{i=1}^{3} \tilde{u}_h(m_i, \Omega_0)\varphi_i(x, y). \tag{3.4.6}$$

This we will use in the actual reconstruction of the limited value in equations (3.4.10) and (3.4.16).

For continuous initial conditions we should have $\tilde{u}_h(m_1, \Omega_0) = \Delta\bar{u}(m_1, \Omega_0)$. However for discontinuous initial conditions we might have $\tilde{u}_h(m_1, \Omega_0) \neq \Delta\bar{u}(m_1, \Omega_0)$.

To approximate the second term in the right-hand side of equation (3.4.6), we define the TVB modified minmod function by

$$\text{minmodTVB}(a_1, a_2, \ldots, a_m) := \text{minmod}\left(a_1, a_2 + \text{sgn}(a_2)Mh^2, \ldots, a_m + \text{sgn}(a_m)Mh^2\right), \tag{3.4.7}$$

where the function minmod is defined as

$$\text{minmod}(a_1, \ldots, a_m) := \begin{cases} \text{sgn}(a_1)\min(|a_1|, \ldots, |a_m|), & \text{if } \text{sgn}(a_1) = \ldots = \text{sgn}(a_m),\\ 0, & \text{otherwise.}\end{cases} \tag{3.4.8}$$

The minmod function is a very common limiter in finite volume methods. For more information on this limiter and limiting with finite volumes we refer to [8].
In the TVB modified minmod function again we use the radius of the circumscribed circle of element Ω_j for h. M is a constant that should be an upper bound for the second derivative of local solution extrema. We note that for small values of M we have order reduction whereas high values of M may allow spurious oscillations in the solution.

Using the TVB modified minmod function we determine the quantities Δ_i

$$\Delta_i = \text{minmodTVB}\left(\tilde{u}_h(m_i, \Omega_0), \nu\Delta\bar{u}(m_i, \Omega_0)\right). \tag{3.4.9}$$

In this function ν should be larger than one. Based on [4] we take $\nu = 1.5$. We now examine two separate cases. For $\sum_{i=1}^{3} \Delta_i = 0$ we simply set

$$u_h(x, y) = \bar{u}_{\Omega_0} + \sum_{i=1}^{3} \Delta_i\varphi_i(x, y), \tag{3.4.10}$$

or even more simply $u_{0,i} = \bar{u}_{\Omega_0} + \Delta_i$ for $i = 1, \dots, 3$.

In case we have $\sum_{i=1}^{3} \Delta_i \neq 0$, we compute

$$\begin{aligned} \text{pos} &= \sum_{i=1}^{3} \max(0, \Delta_i), && (3.4.11)\\ \text{neg} &= \sum_{i=1}^{3} \max(0, -\Delta_i), && (3.4.12)\\ \theta^{+} &= \min\left(1, \frac{\text{neg}}{\text{pos}}\right), && (3.4.13)\\ \theta^{-} &= \min\left(1, \frac{\text{pos}}{\text{neg}}\right), && (3.4.14)\\ \hat{\Delta}_i &= \theta^{+} \max(0, \Delta_i) - \theta^{-} \max(0, -\Delta_i). && (3.4.15) \end{aligned}$$

We now set

$$u_h(x, y) = \bar{u}_{\Omega_0} + \sum_{i=1}^{3} \hat{\Delta}_i \varphi_i(x, y), \qquad (3.4.16)$$

or $u_{0,i} = \bar{u}_{\Omega_0} + \hat{\Delta}_i$ for $i = 1, \dots, 3$. For more information on this limiter we refer to [4].

3.5 Overview

Again we give an overview how to apply the shock detector and limiter after a time step integration has been done. We present this overview in a pseudo-code algorithm as in Algorithms 3.5.1 and 3.5.2. In contrast to Section 2.7 it is not convenient any more to apply the slope detector and limiter in the same iteration. Therefore we apply the shock detector separately of the limiter and we store the results of the shock detector in an array called *indicated*.

Again we store our solution u in a matrix. This matrix has dimensions $number\_of\_elements \times 3$. Each row corresponds to an element and the columns correspond to the three midpoints. Furthermore we use a matrix called *elem_neighbor* having the same dimensions. In this matrix we store per element and per edge the element number of the neighboring element on that edge. In Algorithm 3.5.1 we start by iterating over the elements. For each element we determine the norm. Then we loop over the edges in line number 6. For each edge we calculate the length of the edge and the outward normal vector on that edge. Next we determine whether we have inflow in line 10. If we do not have inflow we skip the remainder of this iteration by the **continue** command. If we have inflow we add the length of the edge to *length_gamma_minus* which denotes the total length of all edges where we have inflow. Also we determine the largest edge. Next we determine the neighboring element on the edge and using equation (3.3.3) we determine the value of $\int_{e_j} (u_{ielem} - u_{nbielem})\, d\Gamma$ and add this to *int*, the value of the total integral across all edges where we have inflow. Having done this for all three edges we set the radius of the circumscribed circle. We determine the indicator with all calculated variables in line 18. At last we set in the indicated array a one for a shock and a zero otherwise.

Algorithm 3.5.1 Pseudo-code algorithm for 2D shock detection

```
for $ielem = 1$ to $number\_of\_elements$ do
  $int = 0$
  $length\_gamma\_minus = 0$
  $max\_edge = 0$
  norm $= \max(|u(ielem, 1)|, |u(ielem, 2)|, |u(ielem, 3)|)$
  for $jedge = 1$ to $3$ do
    determine $length\_edge$
    $max\_edge = \max(max\_edge, length\_edge)$
    determine $normal$
    if $v \cdot normal \geq 0$ then
      continue
    end if
    $length\_gamma\_minus = length\_gamma\_minus + length\_edge$
    $neigh\_elem = elem\_neighbor(ielem, jedge)$
    $int = int + length\_edge \cdot (u(ielem, jedge) - u(neigh\_elem, jedge))$
  end for
  $radius = max\_edge/2$
  $Ind = |int|/(radius \cdot length\_gamma\_minus \cdot norm)$
  if $Ind > 1$ then
    $indicated(ielem) = 1$
  else
    $indicated(ielem) = 0$
  end if
end for
```

For the limiter we look at Algorithm 3.5.2. In this algorithm we use a matrix *center_coords* which has dimensions $number\_of\_elements \times 2$. In this matrix for each row the center of the corresponding element is saved.

In Algorithm 3.5.2 we start by setting ν and M which we need to apply equations (3.4.9) and (3.4.7). Next we iterate over the elements. For each element we check whether a shock has been detected and we skip the remainder of the loop if this is not the case. We determine φ_1, φ_2 and φ_3 and, using the coordinates of the center of the element, we calculate the approximated solution at the center in line number 10. We repeat this process for the neighboring elements where we use the $\varphi_i's$ of the neighboring element.

Using the coordinates of the midpoint of the edge and the centers of neighboring elements we construct a matrix and righthand side to determine α_1 and α_2 as in equation (3.4.1). With these values of α_1 and α_2 we determine $\Delta\bar{u}_1$ and we repeat the procedure for $\Delta\bar{u}_2$ and $\Delta\bar{u}_3$ for which we need to determine α_1 and α_2 again as well.

Next we determine $\tilde{u}_1$ and with the minmodTVB function we calculate Δ_1. This process is repeated as well to calculate Δ_2 and Δ_3. Finally we check in line 30 whether we can limit directly or need to adjust the variables Δ_i first. In case we need to adjust these variables we take the sum over the positive values and the sum over the negative values. Then we

Algorithm 3.5.2 Pseudo-code algorithm for 2D limiting

```
$nu = 1.5$
$M = 100$
**for** $ielem = 1$ **to** $number\_of\_elements$ **do**
  **if** $indicated(ielem) == 0$ **then**
    **continue**
  **end if**
  **determine** $phi\_1,\ phi\_2$ **and** $phi\_3$
  $xc = center\_coords(ielem, 1)$
  $yc = center\_coords(ielem, 2)$
  $uk0 = u(ielem, 1) \cdot phi\_1(xc, yc) + u(ielem, 2) \cdot phi\_2(xc, yc) + u(ielem, 3) \cdot phi\_3(xc, yc)$
  **determine** $uk1,\ uk2$ **and** $uk3$ **similarly**

  **determine** $m1x$ **and** $m1y$
  $neigh\_elem1 = elem\_neighbor(ielem, 1)$
  $neigh\_elem2 = elem\_neighbor(ielem, 2)$
  $xn1 = center\_coords(neigh\_elem1, 1)$
  $yn1 = center\_coords(neigh\_elem1, 2)$
  $xn2 = center\_coords(neigh\_elem2, 1)$
  $yn2 = center\_coords(neigh\_elem2, 2)$
  $mat = [[xn1 - xc\ xn2 - xc]; [yn1 - yc\ yn2 - yc]]$
  $RHS = [m1x - xc; m1y - yc]$
  **solve** $mat \cdot alpha = RHS$ **for** $alpha = [alpha1; alpha2]$
  $deltau1 = alpha1 \cdot (uk1 - uk0) + alpha2 \cdot (uk2 - uk0)$
  **determine** $deltau2$ **and** $deltau3$ **similarly**

  $uhtilde1 = u(ielem, 1) - uk0$
  $delta1 = \text{minmodTVB}(uhtilde1, nu \cdot deltau1)$
  **determine** $delta2$ **and** $delta3$ **similarly**
  $sum\_delta = delta1 + delta2 + delta3$
  **if** $sum\_delta \neq 0$ **then**
    $pos = \max(0, delta1) + \max(0, delta2) + \max(0, delta3)$
    $neg = \max(0, -delta1) + \max(0, -delta2) + \max(0, -delta3)$
    $theta\_plus = \min(1, neg/pos)$
    $theta\_min = \min(1, pos/neg)$
    $delta1 = theta\_plus \cdot \max(0, delta1) - theta\_min \cdot \max(0, -delta1)$
    $delta2 = theta\_plus \cdot \max(0, delta2) - theta\_min \cdot \max(0, -delta2)$
    $delta3 = theta\_plus \cdot \max(0, delta3) - theta\_min \cdot \max(0, -delta3)$
  **end if**
  $u(ielem, 1) = delta1 + uk0$
  $u(ielem, 2) = delta2 + uk0$
  $u(ielem, 3) = delta3 + uk0$
**end for**
```

determine θ^+ and θ^- as in equations (3.4.13) and (3.4.14) respectively. Finally we calculate the adjusted values for Δ_i and limit the coefficients of the approximation within this element.

3.6 Results

We start testing Discontinuous Galerkin with a simple continuous initial condition. This way we can see that any imperfections are due to a wrong implementation and not due to the method. As initial condition we take

$$u(x, y, 0) = 5 + \sin(2\pi x)\sin(2\pi y). \tag{3.6.1}$$

Furthermore we take

$$\mathbf{v} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{3.6.2}$$

In Figure 3.4 we see the result of solving (3.1.16) for 100 time steps with the continuous initial condition. The surface plot shows what we would expect. We can now investigate

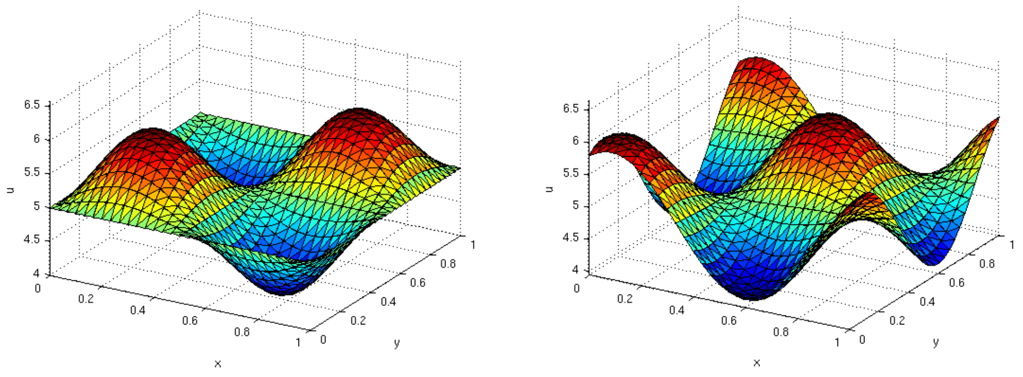

Figure 3.4: Initial condition (3.6.1) (left) and solution of (3.1.16) after 100 time steps.

what happens if we take a discontinuous initial condition. For this, we take

$$u(x, y, 0) = \begin{cases} 5, & \text{if } x + y < 1; \\ 0, & \text{otherwise.} \end{cases} \tag{3.6.3}$$

as our initial condition. In Figure 3.5 we see the discontinuous initial condition on the left. On the right we see a plot of the mesh where the red dots indicate the elements where a shock has been detected. We see that the shock along the line $x+y = 1$ is not detected. However we have not yet applied any time integration in this result. So for every midpoint of an element on this line the corresponding coefficient has exactly the same value as the coefficient in that midpoint of the neighboring element. Therefore the numerator of the indicator will always be zero and thus a shock can not be detected. This is not a disaster since after already one time step integration the shock along the line $x + y = 1$ will be detected. This is shown in Figure 3.6. Finally we look at the approximation after time integration, see the results in Figure 3.7. In this figure we see on the left the result if we don't apply a limiter. This plot clearly shows wiggles. On the right we see the approximation when applying a limiter. Now the wiggles have disappeared.

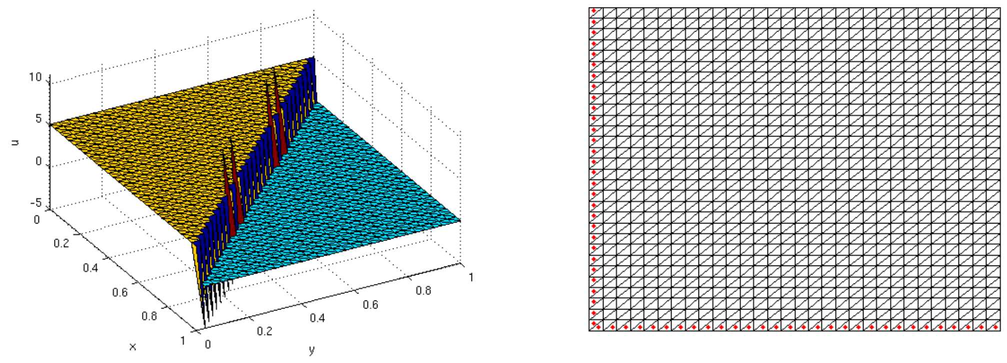

Figure 3.5: Initial condition (3.6.3) (left) and shock detection of initial condition (right).

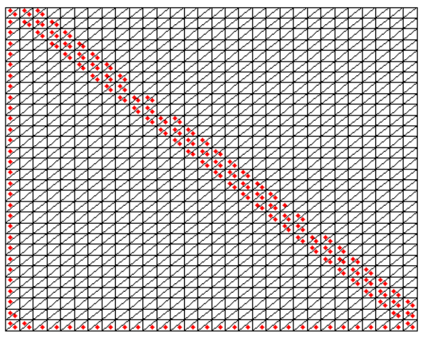

Figure 3.6: Shock detection after time integration.

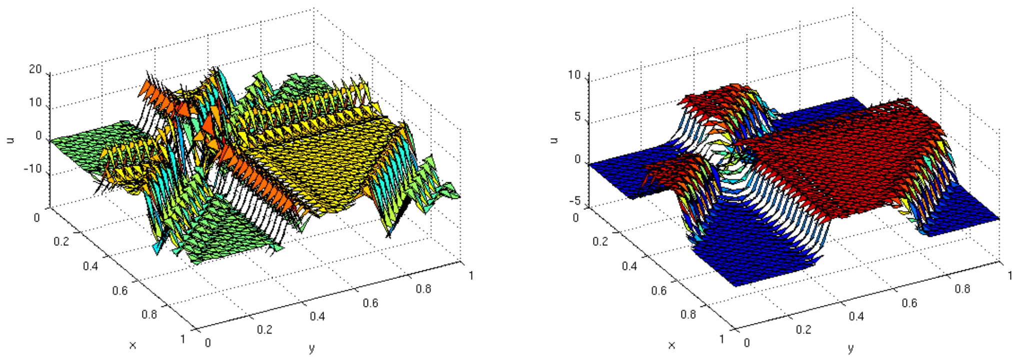

Figure 3.7: Solution of (3.1.16) with initial condition (3.6.3) after 100 time steps without and with limiting.

Chapter 4

Two-phase flow in a porous medium

Having investigated Discontinuous Galerkin in two dimensions we can now construct the model for a general two-phase flow in a porous medium. Using [2] we will make a model for this two-phase flow after which we will apply discontinuous Galerkin.

4.1 Constructing the model

We use a multi-phase flow for the flow of two phases.

We have on some domain Ω and some time interval $(0, T]$

$$\begin{aligned} \varphi\frac{\partial S_1}{\partial t} + \nabla \cdot q_1(S_1) &= 0, && (4.1.1a)\\ \varphi\frac{\partial S_2}{\partial t} + \nabla \cdot q_2(S_2) &= 0, && (4.1.1b)\end{aligned}$$

where $S_1 + S_2 = 1$. Furthermore, we have a boundary and initial condition

$$\begin{aligned} S_1(\mathbf{x}, t) &= f(\mathbf{x}), \quad \forall x \in \Gamma_1,\ \forall t \in (0, T], && (4.1.2a)\\ S_1(\mathbf{x}, 0) &= S_0(\mathbf{x}), \quad \forall x \in \Omega, && (4.1.2b)\end{aligned}$$

with $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$.[1] Since the equation is of first order we only have a boundary condition on the inflow boundary, Γ_1. Furthermore, since we have $S_1 + S_2 = 1$ we do not need any boundary or initial conditions for S_2. In these equations we model the **saturation** of phase 1 and phase 2 by S_1 and S_2 respectively. The constant φ indicates the **porosity** of the porous medium, which is independent of the fluid flowing through the medium. The functions q_1 and q_2 model the **volumetric flow** through the porous medium. For these functions q_1 and q_2 we have Darcy's Law. This law gives us

$$\begin{aligned} q_1 &= -\frac{\kappa_1}{\mu_1}\nabla(p_1 + \rho_1 g z), && (4.1.3)\\ q_2 &= -\frac{\kappa_2}{\mu_2}\nabla(p_2 + \rho_2 g z). && (4.1.4)\end{aligned}$$

Taking a two dimensional area with a fixed depth we will be able to neglect the terms $\rho_1 g z$ and $\rho_2 g z$. Furthermore, we have κ_α, $\alpha = 1, 2$, being the **permeability** of the porous medium

[1]We need the boundaries Γ_2 and Γ_3 later on, so it is convenient to already define them here.

compared to phase α. The **viscosity** of phase α is indicated by μ_α.

We also have an **irreducible phase saturation** level for the phase flows. This means that below this saturation level, a phase is unable to flow when subjected to a potential p_α. In reality the phase does flow if the saturation is below this level. However, the velocity is so low that it is hardly noticeable and therefore we are able to neglect this velocity.

The phase-permeabilities κ_1 and κ_2 depend on the saturation as stated earlier. We have, for a saturation level of one, that these variables are equal to the one-phase permeability κ. As one can imagine some fluids flow easier through soil than other fluids. Therefore, we introduce the **relative permeability** $\kappa_1 = \kappa\kappa_{r1}$ and $\kappa_2 = \kappa\kappa_{r2}$. For a saturation level smaller than or equal to the irreducible phase saturation, S_{1c} or S_{2c}, we should have $\kappa_{r1} = 0$ respectively $\kappa_{r2} = 0$. For $S_1 > S_{1c}$ or $S_2 > S_{2c}$ the relative permeability κ_{r1} respectively κ_{r2} will increase monotonically. For these functions there are many relations based on the fluids one investigates. However we will use the following simplified relations from [2]:

$$\begin{aligned}
\kappa_{r1} &= \kappa'_{r1} S_{1e}, && (4.1.5a)\\
\kappa_{r2} &= \kappa'_{r2} S_{2e}, && (4.1.5b)\\
S_{1e} &= \frac{S_1 - S_{1c}}{1 - S_{1c} - S_{2c}}, && (4.1.5c)\\
S_{2e} &= \frac{S_2 - S_{2c}}{1 - S_{1c} - S_{2c}}. && (4.1.5d)
\end{aligned}$$

In this equation the variables κ'_{r1} and κ'_{r2} will depend on the fluids and are in general not equal. To satisfy the irreducible phase saturation condition we can rewrite

$$\kappa_{r1'} = \begin{cases} \kappa_{r1'}, & \text{if } S_1 > S_{1c}, \\ 0, & \text{otherwise,} \end{cases} , \qquad (4.1.6)$$

$$\kappa_{r2'} = \begin{cases} \kappa_{r2'}, & \text{if } S_2 > S_{2c}, \\ 0, & \text{otherwise.} \end{cases} \qquad (4.1.7)$$

For more complex relations for the relative permeability we refer to [2].
In total we now have for q_1 and q_2

$$\begin{aligned}
q_1 &= -\frac{\kappa\kappa'_{r1} S_{1e}}{\mu_1} \nabla p_1, && (4.1.8)\\
q_2 &= -\frac{\kappa\kappa'_{r2} S_{2e}}{\mu_2} \nabla p_2. && (4.1.9)
\end{aligned}$$

For different phases one usually has a difference in the pressures. Therefore one defines the capillary pressure function p_{cap} which is defined as the difference in pressure across the interface between the non-wetting and wetting phase for two immiscible fluids. In oil-water systems, water will be the wetting phase whereas in gas-liquid systems the liquid, which might be oil, will be the wetting phase. In our case we will assume that phase one is the wetting fluid. This gives us

$$p_{cap} = p_2 - p_1. \qquad (4.1.10)$$

Having a non-zero capillary pressure will remove the discontinuity in the equation. The capillary pressure is with respect to coordinates only depending on the height and since we take our region at a fixed depth it is justifiable to take the capillary pressure equal to zero. This is also discussed in [5]. So we have

$$p_1 = p_2 =: p. \tag{4.1.11}$$

4.2 Solving the partial differential equations in the model

In order to solve the model and we need to state boundary and initial conditions. We will apply the model to a test region $\Omega = [0,1] \times [0,1]$. Since the model is only of first order, we need just one boundary condition which will be a Dirichlet boundary condition. Since we are investigating a flow problem, we will apply the boundary condition only to a part of the boundary where we have inflow. We denote this part of the boundary by Γ_1. Since we have $S_1 + S_2 = 1$ we only need to solve this model for one phase and we can reconstruct the other phase. Therefore we will solve this model only for the wetting fluid, which is assumed to be phase 1. This gives us

$$\varphi \frac{\partial S_1}{\partial t} + \nabla \cdot q_1(S_1) = 0, \qquad \mathbf{x} \in \Omega,\ t \in (0,T], \tag{4.2.1a}$$

$$\varphi \frac{\partial S_2}{\partial t} + \nabla \cdot q_2(S_2) = 0, \qquad \mathbf{x} \in \Omega,\ t \in (0,T], \tag{4.2.1b}$$

$$S_1(\mathbf{x},t) = f(\mathbf{x}), \quad \forall \mathbf{x} \in \Gamma_1,\ \forall t > 0, \tag{4.2.1c}$$

$$S_1(\mathbf{x},0) = S_0(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega. \tag{4.2.1d}$$

with $S_1 + S_2 = 1$. Adding equation (4.2.1a) to equation (4.2.1b) gives us

$$\varphi \frac{\partial S_1}{\partial t} + \nabla \cdot q_1(S_1) + \varphi \frac{\partial S_2}{\partial t} + \nabla \cdot q_2(S_2) = 0, \tag{4.2.2}$$

which can be rewritten as

$$\varphi \frac{\partial (S_1 + S_2)}{\partial t} + \nabla \cdot (q_1(S_1) + q_2(S_2)) = 0. \tag{4.2.3}$$

However, since $S_1 + S_2 = 1$ this reduces to

$$\nabla \cdot (q_1(S_1) + q_2(S_2)) = 0. \tag{4.2.4}$$

Substituting equation (4.1.8) and equation (4.1.9) into equation (4.2.4) gives

$$-\nabla \cdot \left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} \nabla p_1 + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \nabla p_2 \right) = 0. \tag{4.2.5}$$

Using equation (4.1.11), the actual equation we will need to solve

$$-\nabla \cdot \left(\left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \nabla p \right) = 0, \tag{4.2.6}$$

for p where we have $p_1 = p_2 = p$. We now also need boundary conditions for the pressure. In the same region Ω we solve the following system

$$-\nabla \cdot \left(\left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \nabla p \right) = 0, \qquad \mathbf{x} \in \Omega, t \in (0, T], \tag{4.2.7a}$$

$$p(\mathbf{x}) = p_0(\mathbf{x}), \quad \mathbf{x} \in \Gamma_1, \tag{4.2.7b}$$

$$-\left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \frac{\partial p}{\partial n}(\mathbf{x}) = Q_1, \qquad \mathbf{x} \in \Gamma_2, \tag{4.2.7c}$$

$$-\left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \frac{\partial p}{\partial n}(\mathbf{x}) = 0, \qquad \mathbf{x} \in \Gamma_3. \tag{4.2.7d}$$

Note that in this system we have a fixed pressure at the boundary where we have inflow of the saturation. Furthermore we have outflow at boundary Γ_2. So in general Q_1 will be negative. Boundary Γ_3 will be isolated. Although it may not be clear on first sight, p is depends on time. Equation (4.2.7a) is depending on the saturation S_1 and S_2 which are dependent on time.

4.2.1 Solving the equation for the saturation

Now we have constructed system (4.2.7) to solve for the pressure, we will be able to start solving system (4.1.1) numerically. Therefore, we will apply the two dimensional discontinuous Galerkin method to equation (4.2.1a), which is $\varphi \frac{\partial S_1}{\partial t} + \nabla \cdot q_1(S_1) = 0$, assuming that we know ∇p. We make this derivation similar to Section 3.1. We start by determining a triangulation of the area. Next we multiply this equation by a test function v and integrate over an element Ω_j to obtain

$$\varphi \frac{d}{dt} \int_{\Omega_j} S_1 v d\Omega + \int_{\Omega_j} (\nabla \cdot q_1(S_1)) v d\Omega = 0. \tag{4.2.8}$$

Again we denote by V_h the finite dimensional space of all piecewise linear functions on the element Ω_j. We substitute into equation (4.2.8) $v = v_h \in V_h$ and $S_h \in V_h \approx s_1$, where S_h denotes the numerical approximation for S_1, and we apply *Green's theorem* to obtain

$$\varphi \frac{d}{dt} \int_{\Omega_j} S_h v_h d\Omega - \int_{\Omega_j} q_1(S_h) \nabla \cdot v_h d\Omega + \int_{\partial\Omega_j} q_1(S_h) v_h \cdot \mathbf{n} d\Gamma = 0, \;\; \forall v_h \in V_h. \tag{4.2.9}$$

Again we can replace the integral over the boundary $\partial\Omega_j$ by a sum of integrals over the edges of the element

$$\varphi \frac{d}{dt} \int_{\Omega_j} S_h v_h d\Omega - \int_{\Omega_j} q_1(S_h) \nabla \cdot v_h d\Omega + \sum_{e \in \partial\Omega_j} \int_e q_1(S_h) v_h \cdot \mathbf{n_e} d\Gamma = 0, \quad \forall v_h \in V_h. \tag{4.2.10}$$

The vector $\mathbf{n}_e$ again denotes the outward normal vector on edge e. We remark that S_h is discontinuous at the element boundaries and therefore equation (4.2.10) is not well defined. Therefore we need to replace the term $q_1(S_h) \cdot \mathbf{n}_e$ in this integral by the numerical flux h_e

$$h_e = h_e \left(S_h \left(t, x^{int(\Omega_j)}, y^{int(\Omega_j)} \right), S_h \left(t, x^{ext(\Omega_j)}, y^{ext(\Omega_j)} \right) \right) \tag{4.2.11}$$

For interior element edges we will use the Lax-Friedrichs flux function as in equation (3.1.10)

$$h_e(a,b) = \frac{1}{2}[q_1(a)\cdot\mathbf{n}_e + q_1(b)\cdot\mathbf{n}_e - \alpha_e\cdot(b-a)]. \tag{4.2.12}$$

For edges on the boundary of $[0,1]\times[0,1]$ we will simply use

$$h_e(a) = q_1(a). \tag{4.2.13}$$

In this flux function we take for a the approximated value of S at some position x_{ref} on the edge. For b we take the approximated value in the neighboring element at the same position x_{ref}. The constant α_e is the numerical viscosity constant which should be an estimate of the largest eigenvalue of the Jacobian $\frac{\partial}{\partial S}q_1(S_h)(t,x,y)\cdot\mathbf{n}_e$ for (t,x,y) in a neighborhood of edge e.

Using the derivation as we have done in two dimensions in Chapter 3, we need to solve

$$MS_1^{t_{new}} = MS_1^{t_{old}} + \Delta t\left(S + F^{t_{old}}\right). \tag{4.2.14}$$

We note that the stiffness matrix will in fact be a vector in this derivation. The integral $\int_{\Omega_j} q_1(S_h)\nabla\cdot v_h d\Omega$ gives us

$$\begin{aligned} S_i &= \int_{\Omega_j} q_1\left(\sum_{k=1}^{3} S_{j,k}(t)\varphi_k(x,y)\right)\nabla\cdot\varphi_i(x,y)d\Omega, \\ &= \frac{|\Omega_j|}{3}\nabla\varphi_i\cdot\sum_{l=1}^{3} q_1\left(\sum_{k=1}^{3} S_{j,k}\varphi_k(\mathbf{x}_l)\right), \\ &= \frac{|\Omega_j|}{3}\nabla\varphi_i\cdot\sum_{l=1}^{3} q_1(S_{j,l}). \end{aligned} \tag{4.2.15}$$

4.2.2 Solving for the pressure

In the previous subsection we have solved system (4.2.14) for the saturation of the wetting phase at the new time step, assuming we had already determined the gradient of the pressure p in each element. In this subsection we will solve system (4.2.7) in order to actually know the pressure. For this system we will use the finite element method, since the pressure will always be continuous.

We have the following system for the pressure

$$-\nabla\cdot\left(\left(\frac{\kappa\kappa'_{r1}S_{1e}}{\mu_1}\frac{\kappa\kappa'_{r2}S_{2e}}{\mu_2}\right)\nabla p\right) = 0, \qquad \mathbf{x}\in\Omega \tag{4.2.16a}$$

$$p(\mathbf{x}) = p_0(\mathbf{x}), \qquad \mathbf{x}\in\Gamma_1, \tag{4.2.16b}$$

$$-\left(\frac{\kappa\kappa'_{r1}S_{1e}}{\mu_1}+\frac{\kappa\kappa'_{r2}S_{2e}}{\mu_2}\right)\frac{\partial p}{\partial n}(\mathbf{x}) = Q_1, \qquad \mathbf{x}\in\Gamma_2, \tag{4.2.16c}$$

$$-\left(\frac{\kappa\kappa'_{r1}S_{1e}}{\mu_1}+\frac{\kappa\kappa'_{r2}S_{2e}}{\mu_2}\right)\frac{\partial p}{\partial n}(\mathbf{x}) = 0, \qquad \mathbf{x}\in\Gamma_3. \tag{4.2.16d}$$

We start by multiplying equation (4.2.16a) by a test function v, subject to $v|_{\Gamma_1} = 0$, and we integrate over the entire domain Ω to obtain

$$\int_\Omega -\nabla \cdot \left(\left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \nabla p \right) v d\Omega = 0. \tag{4.2.17}$$

Next we apply *Green's theorem* which gives us

$$\int_\Omega \left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \nabla p \cdot \nabla v d\Omega = \int_\Gamma \left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \nabla p \cdot \mathbf{n} v d\Gamma. \tag{4.2.18}$$

We split the boundary Γ in Γ_1, Γ_2 and Γ_3. On Γ_1 we have $p = p_0$ so $v = 0$. On Γ_3 we have $0 = \left(-\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} - \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \frac{\partial p}{\partial n} = \left(-\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} - \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \nabla p \cdot \mathbf{n}$. So only the integral over the boundary Γ_2 will not vanish. Substituting $\left(-\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} - \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \frac{\partial p}{\partial n} = Q_1$ on Γ_2 will give us

$$\int_\Omega \left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \nabla p \cdot \nabla v d\Omega = \oint_{\Gamma_2} Q_1 v d\Gamma. \tag{4.2.19}$$

We now substitute $p(\mathbf{x}, t) = \sum_{j=1}^{N+N_b} p_j(t) \varphi_j(\mathbf{x})$, with N_b the number of boundary nodes, and $v = \varphi_i(\mathbf{x})$ for $i = 1, \ldots, N$ in equation (4.2.19) to obtain

$$\sum_{j=1}^{N+N_b} p_j \int_\Omega \left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \nabla \varphi_j \cdot \nabla \varphi_i d\Omega = \oint_{\Gamma_2} Q_1 \varphi_i d\Gamma, \qquad i = 1, \ldots, N. \tag{4.2.20}$$

We will write this system as $Sp = F$. For the element stiffness matrix S^{el} we have

$$\begin{aligned} S^{el}_{ij} &= \int_{\Omega_{el}} \left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) \nabla \varphi_j \cdot \nabla \varphi_i d\Omega, \\ &= \nabla \varphi_j \cdot \nabla \varphi_i \frac{|\Omega_j|}{3} \sum_{k=1}^{3} \left(\frac{\kappa \kappa'_{r1} S_{1e}}{\mu_1} + \frac{\kappa \kappa'_{r2} S_{2e}}{\mu_2} \right) (\mathbf{x}_k). \end{aligned} \tag{4.2.21}$$

For the element right hand side vector F^{el} we obtain

$$\begin{aligned} F^{el}_i &= \oint_{\Gamma_{el}} Q_1 \varphi_i d\Gamma, \\ &= \frac{|\Gamma_{el}|}{2} Q_1(\mathbf{x}_k). \end{aligned} \tag{4.2.22}$$

Having solved $Sp = F$ for the pressure we need to determine the gradient of the pressure. We examine the linear basis functions we used

$$\varphi_i(x, y) = \alpha_i + \beta_i x + \gamma_i y. \tag{4.2.23}$$

Using this relation we can determine the gradient of the pressure per element by

$$\nabla p_{el_j} = \sum_{k=1}^{3} \begin{pmatrix} \beta_k \\ \gamma_k \end{pmatrix} p_k. \tag{4.2.24}$$

4.3 Overview

As we have seen in the previous section we have examined two systems. Both systems depend on the solution of the other system. Therefore we need to solve both systems at every time step. Since an initial condition is given for the saturation, we start in each time step by solving $Sp = F$ and determining the gradient of the pressure from equation (4.2.24). Hereafter, we can solve (4.2.14) for one time step. If necessary, we can now apply the shock detector and the limiter for the saturation. Then the process is repeated at the next time step. In Algorithm 4.3.1 we give a simplistic overview of the coupling of these systems. We store the saturation in a matrix S_1 with three values per row for the three unknowns. The gradient of the pressure is stored in a vector *grad_pressure* of length *number_of_elements*.

Algorithm 4.3.1 Pseudo-code algorithm for 2D phase flow in a porous medium

```
determine $S_1$
for $itime = 1$ to $T_{end}$ do
  determine $grad\_pressure$
  determine $S_1$
  apply shock detector
  apply limiter
end for
```

The saturation can also be stored at the end of each time step. If necessary one can reconstruct the solution for the corner points or Gaussian points[2] first.

4.4 Shock detector and limiter

We will use the shock detector as discussed in Section 3.3. However, this shock detector assumes that every element has a neighbor on every edge. So we need to make a slight adjustment for elements on the boundary. We adjust the shock detector by flagging every element with an edge on the boundary as detected.

Furthermore, we will use the limiter from Section 3.4. This limiter is also based on the assumption that every element has a neighbor on every edge. So we need to adjust the limiter as well. We have to use a grid where every element has at most one edge on the boundary. If an element would have two edges on the boundary, we would be unable to use the limiter, since we need at least two neighboring elements to determine the slopes between the solutions on the elements. So we assume that every element has at most one edge on the boundary. We constructed the limiter so that each edge uses the neighboring element, and the next edge and element, when going clockwise over the edges. We need to adjust this to the next non-boundary edge. Although we will not be able to determine a limited value for the unknowns at the edges, we will be able to limit the other two unknowns in those elements. In Algorithm 4.4.1 we give an overview for limiting of elements with an edge on the boundary. In this algorithm we assume that the element number *ielem* is given, and that limiting is needed. Furthermore, we assume edge two to be on the boundary.

[2]Just as in one dimension, these points are located halfway between the midpoint of the edge and the element center.

Algorithm 4.4.1 Pseudo-code algorithm for 2D limiting of an element with edge number two on the boundary

```
    $nu = 1.5$
    $M = 100$

    **determine** $phi\_1$, $phi\_2$ **and** $phi\_3$
 5: $xc = center\_coords(ielem, 1)$
    $yc = center\_coords(ielem, 2)$
    $uk0 = u(ielem, 1) \cdot phi\_1(xc, yc) + u(ielem, 2) \cdot phi\_2(xc, yc) + u(ielem, 3) \cdot phi\_3(xc, yc)$
    **determine** $uk1$ **and** $uk3$ **similarly**

10: **determine** $m1x$ **and** $m1y$
    $neigh\_elem1 = elem\_neighbor(ielem, 1)$
    $neigh\_elem3 = elem\_neighbor(ielem, 3)$
    $xn1 = center\_coords(neigh\_elem1, 1)$
    $yn1 = center\_coords(neigh\_elem1, 2)$
15: $xn3 = center\_coords(neigh\_elem3, 1)$
    $yn3 = center\_coords(neigh\_elem3, 2)$
    $mat = [[xn1 - xc\ xn3 - xc]; [yn1 - yc\ yn3 - yc]]$
    $RHS = [m1x - xc; m1y - yc]$
    **solve** $mat \cdot alpha = RHS$ **for** $alpha = [alpha1; alpha2]$
20: $deltau1 = alpha1 \cdot (uk1 - uk0) + alpha2 \cdot (uk3 - uk0)$
    $mat = [[xn3 - xc\ xn1 - xc]; [yn3 - yc\ yn1 - yc]]$
    **solve** $mat \cdot alpha = RHS$ **for** $alpha = [alpha1; alpha2]$
    $deltau3 = alpha1 \cdot (uk3 - uk0) + alpha2 \cdot (uk1 - uk0)$

25: $uhtilde1 = u(ielem, 1) - uk0$
    $delta1 = \text{ minmodTVB}(uhtilde1, nu \cdot deltau1)$
    $uhtilde3 = u(ielem, 3) - uk0$
    $delta3 = \text{ minmodTVB}(uhtilde3, nu \cdot deltau3)$

30: $u(ielem, 1) = delta1 + uk0$
    $u(ielem, 3) = delta3 + uk0$
```

4.5 Results

In this section we will show a simple result of solving system (4.1.1). In the next chapter we will show some more realistic applications. In order to be able to solve this system we have to define some parameters. Again, in the next chapter, we will clarify the choice of these parameters. We have chosen

$$\begin{aligned}
\varphi &= 0.4, && (4.5.1a)\\
\mu_1 &= 0.0012\ [kg \cdot m^{-1} \cdot s^{-1}], && (4.5.1b)\\
\mu_2 &= 2 \cdot 10^{-5}[kg \cdot m^{-1} \cdot s^{-1}], && (4.5.1c)\\
\kappa &= 10^{-3}\ [m^2], && (4.5.1d)\\
\kappa'_{r1} &= 0.5, && (4.5.1e)\\
\kappa'_{r2} &= 0.01, && (4.5.1f)\\
S_{1c} &= 0.15, && (4.5.1g)\\
S_{2c} &= 0.05. && (4.5.1h)
\end{aligned}$$

Furthermore, we have to define the boundaries and the boundary conditions. We choose for the boundaries $\Gamma_1 = \Gamma(x = 0)$ and $\Gamma_3 = \Gamma(x = 1)$. As initial condition we take

$$S_1(\mathbf{x}, 0) = 0, \quad \forall \mathbf{x} \in \Omega. \qquad (4.5.2)$$

As boundary conditions we use

$$\begin{aligned}
S_1(\mathbf{x}, t) &= 1, && \forall \mathbf{x} \in \Gamma_1, \forall t > 0, && (4.5.3)\\
p(\mathbf{x}) &= 1, && \forall \mathbf{x} \in \Gamma_1, && (4.5.4)\\
\left(-\frac{\kappa\kappa'_{r1}S_{1e}}{\mu_1} - \frac{\kappa\kappa'_{r2}S_{2e}}{\mu_2}\right)\frac{\partial p}{\partial n}(\mathbf{x}) &= -10, && \mathbf{x} \in \Gamma_2. && (4.5.5)
\end{aligned}$$

In Figure 4.1 we show the results of solving the two-phase flow in a porous medium using the above mentioned parameters and boundary- and initial conditions. The solution is not

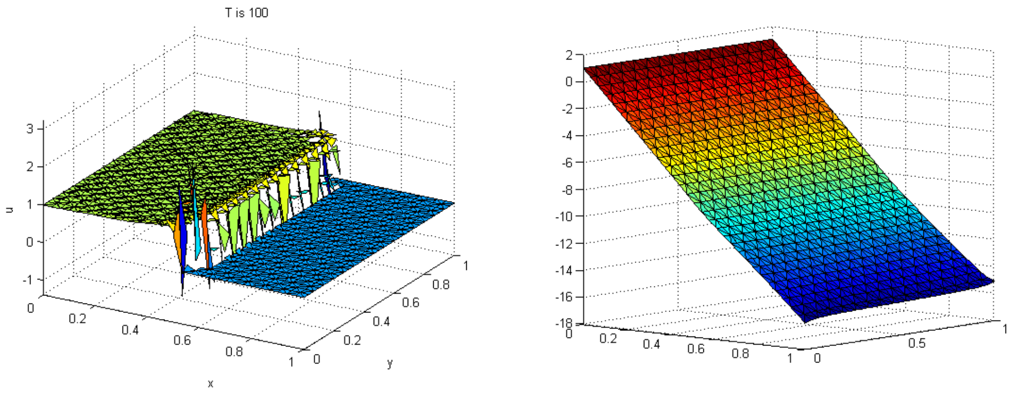

Figure 4.1: Solution of (4.1.1) after 100 time steps left and corresponding pressure right.

entirely smooth and it looks if the limiter is not working properly. However for this problem we can not assume to obtain a fully smooth solution. In Figure 4.2 we show the same result where we did not apply the limiter. So we can conclude that the limiter does the job quite well.

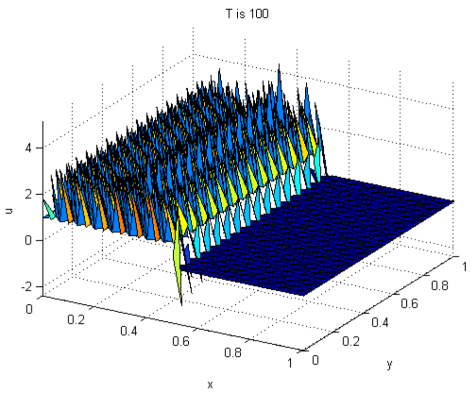

Figure 4.2: Solution of (4.1.1) after 100 time steps without limiting.

Chapter 5

Application of the two-phase flow in a porous medium model

In this chapter we will apply the model for a generic two-phase flow in a porous medium from Chapter 4 to a realistic example.

5.1 Background

In order to store energy, the faculty of EEMCS uses four large pumps to store cold and warm water in the ground. Two of these pumps only work with warm water, and two of them only deal with cold water. Roughly speaking we can sketch the situation during the winter as in Figure 5.1. One can see from the sketch that there are two pumps which pump warm water which are opposite to each other. The same is true for the cold water pumps. The area between the bubbles is air. Furthermore, it is crucial that the bubbles of water never overlap since the water will be mixed and the resulting temperature will be useless. We also know that the edges of the basin are made of clay. The permeability of clay is so low that we can view the basin as completely isolated. One may wonder why the four bubbles of water are not separated by clay as well. We can see the answer from Figure 5.2. This figure illustrates the same basin during the summer. As one can see, there are some areas which are occupied by cold water in the summer and occupied by warm water in the winter. If we were to split the basin into four smaller basins separated by clay, the total area of all basins would need to be significantly larger.

When cold water is needed for air-conditioning, the cold water pumps start retrieving water. The porter uses a tool by which he gives the temperature needed in each room. Furthermore, the tool registers the temperature of the retrieved water and then determines the amount of water that needs to be pumped out each hour to achieve this temperature. This water will be heated by using the air-conditioner and the two warm water pumps will pump this heated water back into the ground. When there is need for warm water the process will be reversed.

5.2 Upscaling

In reality we have a three dimensional basin in which the water is stored. However, it is much easier and faster to solve a two dimensional model. Therefore we can apply upscaling. This

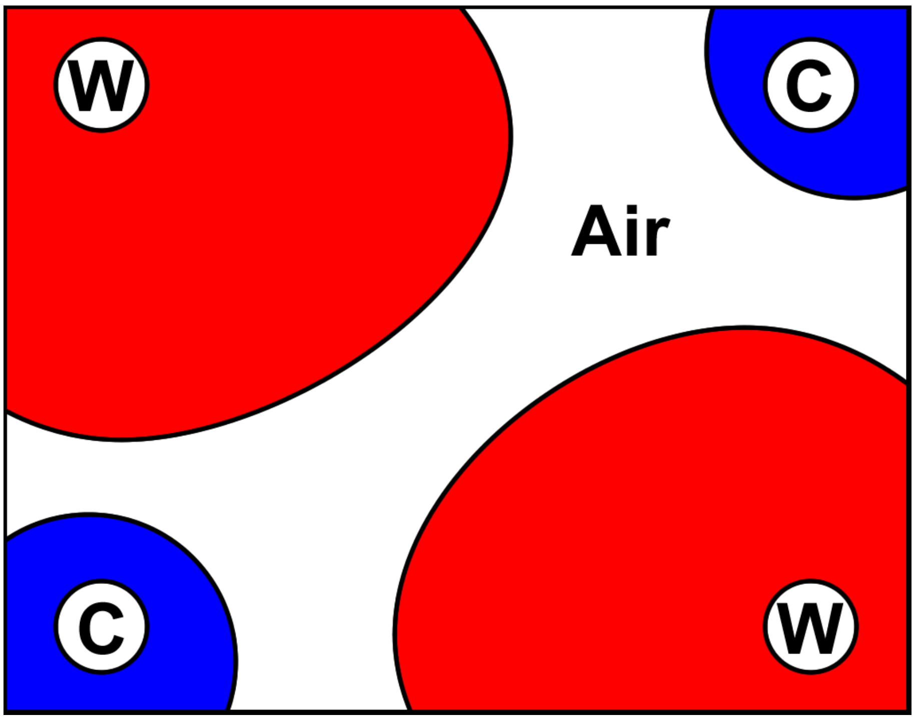

Figure 5.1: Sketch of winter layout.

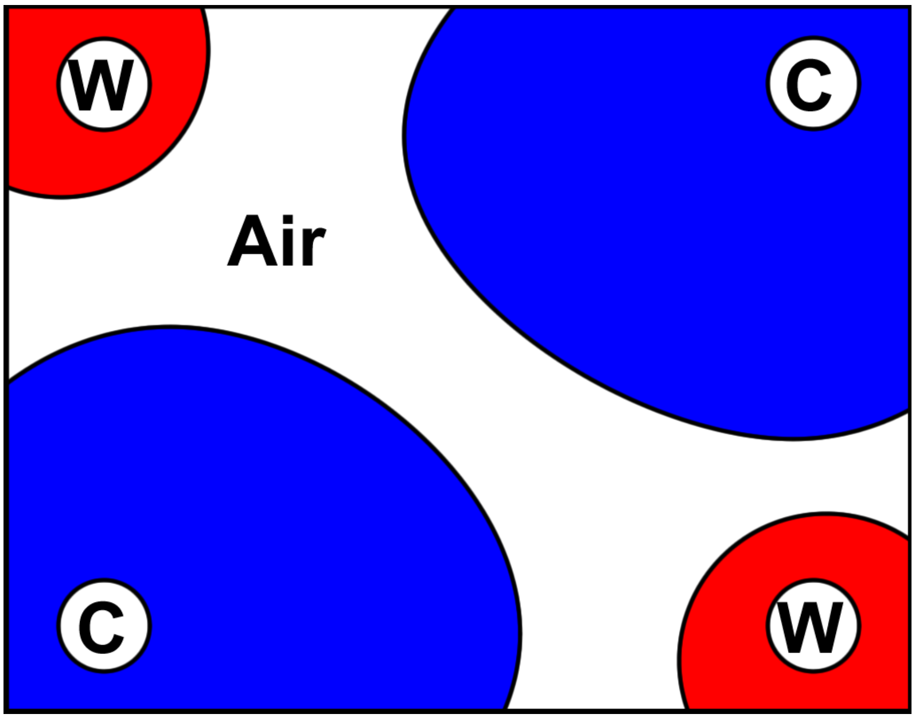

Figure 5.2: Sketch of summer layout.

subject is widely discussed in [5]. We will explain the general idea of this method.

With upscaling one integrates the three dimensional equation over the height and models the saturation by the average saturation, $\bar{S}_1$, over the height. Assuming we have a constant **porosity** and **permeability** we have

$$\bar{S}_1 = \frac{1}{|z_2 - z_1|} \int_{z_1}^{z_2} S_1 dz. \tag{5.2.1}$$

We write the term q_1 as $q_1 = -\lambda(S_1)\nabla p$ for notational convenience. Looking at the x-derivative of the pressure, we now obtain

$$\begin{aligned} \bar{q}_{1,x} &= \frac{1}{|z_2 - z_1|} \int_{z_1}^{z_2} -\lambda(S_1)\frac{\partial p}{\partial x} dz, && (5.2.2) \\ &= \frac{1}{|z_2 - z_1|} \int_{z_1}^{z_2} -\lambda(S_1) dz \frac{\partial p}{\partial x}. && (5.2.3) \end{aligned}$$

Since λ is linear depending on S_1 we can exchange integration and the function λ. This gives us

$$\bar{q}_{1,x} = -\lambda(\bar{S}_1)\frac{\partial p}{\partial x}. \tag{5.2.4}$$

For the y-derivative of we obtain a similar result

$$\bar{q}_{1,y} = -\lambda(\bar{S}_1)\frac{\partial p}{\partial y}. \tag{5.2.5}$$

However, for the z-derivative it appears that the integral will be zero. Using this we can model the three dimensional model as a two dimensional model with an average saturation.

5.3 Choosing the parameters

In Section 4.5 we assumed some values for the parameters in the model. In this section we will clarify how these values were chosen.

We start with the **porosity**, φ. Typical values for soil are between 0.36 and 0.43. For simplicity, we take a value well within these bounds, that is 0.40.

Next we have the **viscosity** of the two flows. For water with a temperature of 20 degrees Celsius this value is equal to $0.001[kg \cdot m^{-1} \cdot s^{-1}]$. For water with a temperature of 6 degrees Celsius the viscosity is $0.0014[kg \cdot m^{-1} \cdot s^{-1}]$. These are the ideal temperatures for the flow of warm and cold water. In reality these temperatures are closer to each other, and so are the viscosities. We model the two flows of water as one flow, and therefore we take $\mu_1 = 0.0012\ [kg \cdot m^{-1} \cdot s^{-1}]$. The flow of air is usually far less viscous. A typical value for the viscosity of air is $2 \cdot 10^{-5}$. So $\mu_2 = 2 \cdot 10^{-5}\ [kg \cdot m^{-1} \cdot s^{-1}]$.

Next we have the **permeability** of soil. This is highly dependent upon the type of soil and usually varies throughout a basin. In Section 5.5 we will investigate what happens when the **permeability** is not constant. For now, we assume that the **permeability** is constant with a value of 10^{-3}, therefore $\kappa = 0.001\ [m^2]$.

For the **relative permeability** we also have the parameters κ'_{r1} and κ'_{r2}. For water this value is equal to 0.5, and therefore $\kappa'_{r1} = 0.5$. For a gaseous fluid, this value is low. We have $\kappa'_{r2} = 0.01$.

This leaves us with the **irreducible phase saturation**. For water a typical value is 0.15, for air we take 0.05.

5.4 Results

In this section we will solve the model from Chapter 4 for a two-phase flow in a porous medium applied to the water storage model. This water storage model actually has four pumps inside the basin instead of on the boundary. Therefore we will make the basin somewhat smaller such that the four pumps will be located at the corners of the region Ω.

For the initial condition we take

$$S_1(x,y,t) = \begin{cases} 1, & \text{if } x \le 0.2 \text{ and } y \le 0.2, \\ 1, & \text{if } x \ge 0.8 \text{ and } y \ge 0.8, \\ 1, & \text{if } x \le 0.4 \text{ and } y \ge 0.6, \\ 1, & \text{if } x \ge 0.6 \text{ and } y \le 0.4, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall x, y \in \Omega, t > 0. \tag{5.4.1}$$

On the boundary Γ_1 we have

$$\begin{aligned} \Gamma_1 &= \{x = 0, 0 \le y \le 0.2\} \\ &\cup \{0 \le x \le 0.2, y = 0\} \\ &\cup \{x = 1, 0.8 \le y \le 1\} \\ &\cup \{0.8 \le x \le 1, y = 1\}. \end{aligned} \tag{5.4.2}$$

On the boundary Γ_2 we have

$$\begin{aligned} \Gamma_2 &= \{x = 1, 0 \le y \le 0.4\} \\ &\cup \{0.6 \le x \le 1, y = 0\} \\ &\cup \{x = 0, 0.6 \le y \le 1\} \\ &\cup \{0 \le x \le 0.4, y = 1\}. \end{aligned} \tag{5.4.3}$$

The boundary conditions for pressure are

$$p(\mathbf{x}) = 1, \qquad \forall \mathbf{x} \in \Gamma_1, \tag{5.4.4}$$

$$\left(-\frac{\kappa\kappa'_{r1}S_{1e}}{\mu_1} - \frac{\kappa\kappa'_{r2}S_{2e}}{\mu_2}\right)\frac{\partial p}{\partial n}(\mathbf{x}) = -10, \quad \forall \mathbf{x} \in \Gamma_2. \tag{5.4.5}$$

In Figure 5.3 we show the initial condition and the boundaries Γ_1 and Γ_2. We divide the region Ω into $N \times N$ squares where each square is made of eight triangular elements as shown

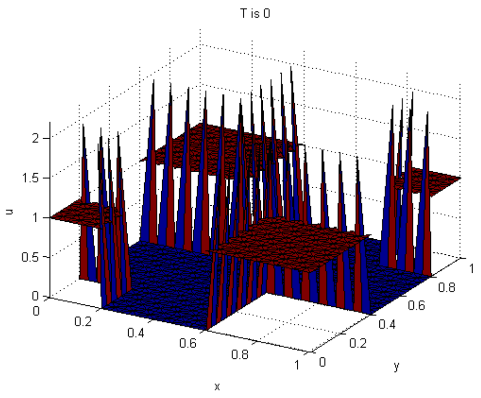

Figure 5.3: Initial condition (5.4.1).

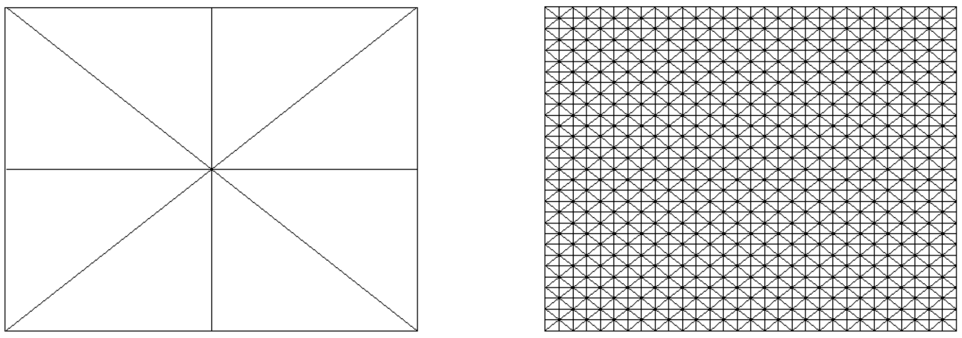

Figure 5.4: Single square (left) and entire mesh (right).

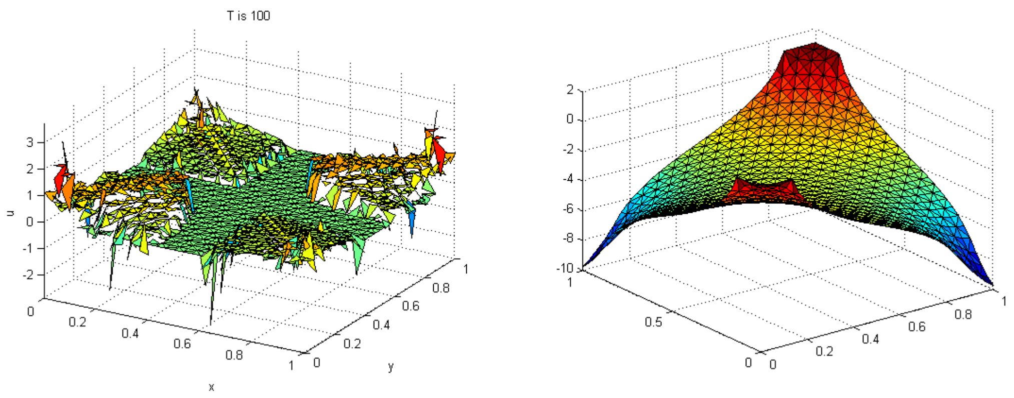

Figure 5.5: Solution of (4.1.1) after 100 time steps left and corresponding pressure right.

in Figure 5.4. In this figure we show a single square on the left and the entire mesh for $N = 15$ on the right. Now solving the model using the given boundary- and initial conditions will give us the result in Figure 5.5. This figure is not that clear on first sight. If we investigate this figure we note that solution on the $x = 0, y = 0$ and $x = 1, y = 1$ corner has expanded. Furthermore, we note that the solution on the remaining two corners has been diminished by the air-phase. However, this solution has flattened out instead of completely withdrawing. There are two explanations for the phenomena. First of all we have to deal with the **irreducible water saturation**. When the saturation level of water gets below the **irreducible water saturation** the flow of water stops moving. So there will be some residual water left because of this treshold. The second reason has to deal with the **viscosity** of the air-phase. Due to the very small viscosity of air, the air-flow does not exercise enough force on the flow of water. Therefore, it partially flows together with the water-flow.

Last we note that the solution is not physically not entirely correct. We have seen in the results section of the previous chapter that this might happen. Actually not using the limiter will blow the solution up to 10^{27} on at least one element. Using different values for ν and M in the limiter might limit the solution better.

5.5 Effect of permeability

In this section we investigate the effect of having a non-constant **permeability**. However, in Section 5.2 we assumed a constant **permeability**. Therefore, we have to be cautious adjusting the **permeability** so that we are able to still use upscaling.

We begin by splitting the region $[0, 1] \times [0, 1]$ into separate domains with different permeabilities. For $y \leq 0.5$ we still have $\kappa = 10^{-3}$ $[m^2]$. For $y > 0.5$ we will now have $\kappa = 5 \cdot 10^{-3}$ $[m^2]$. So the **permeability** will be 5 times as large. By taking the **permeability** equal across the entire height, we will not have issues with upscaling.

All of the other parameters are the same as in Section 5.4. However, we use different boundaries, boundary conditions and initial condition, so we can show the effect of different permeabilities more clearly. As in Section 4.5 we have chosen for the boundaries $\Gamma_1 = \Gamma(x = 0)$ and $\Gamma_3 = \Gamma(x = 1)$. As initial condition we have taken

$$S_1(\mathbf{x}, 0) = 0, \quad \forall \mathbf{x} \in \Omega. \tag{5.5.1}$$

For the boundary conditions we take

$$S_1(\mathbf{x}, t) = 1, \qquad \forall \mathbf{x} \in \Gamma_1, \forall t > 0, \tag{5.5.2}$$

$$p(\mathbf{x}) = 1, \qquad \forall \mathbf{x} \in \Gamma_1, \tag{5.5.3}$$

$$\left(-\frac{\kappa\kappa'_{r1}S_{1e}}{\mu_1} - \frac{\kappa\kappa'_{r2}S_{2e}}{\mu_2}\right)\frac{\partial p}{\partial n}(\mathbf{x}) = -10, \quad \forall \mathbf{x} \in \Gamma_2. \tag{5.5.4}$$

In Figure 5.6 we show the results. This figure clearly shows that part of the solution is flowing faster where we have a higher permeability.

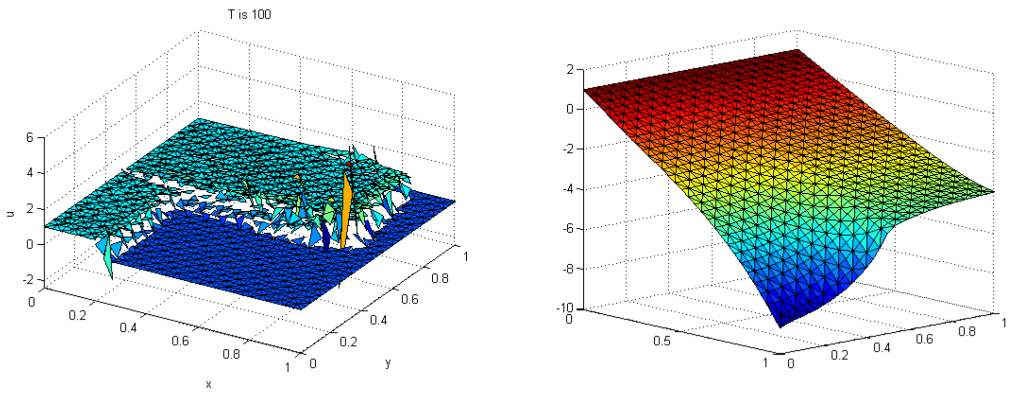

Figure 5.6: Solution of (4.1.1) with a non-constant permeability after 100 time steps left and corresponding pressure right.

Chapter 6

Conclusions

This chapter summarizes the results we have obtained in this thesis.

In this thesis we have constructed a model for a general two-phase flow in a porous medium. In Chapter 5 we have used this model in an application on the water storage beneath the faculty of EEMCS. We have used discontinuous Galerkin in order to make a numerical approximation of the solution of this model. In Chapters 2 and 3 we have derived the discontinuous Galerkin method in one and two dimensions as a basis for the two-phase flow model.

In the application of the two-phase flow model we have demonstrated what happens when we have two pumps, located on two opposite sides, pumping in water and two pumps, located at the other two corners, extracting water. We have seen that the water-phase is diminished by to the two pumps which are extracting water, which is partially due to the force exercised by the air-phase. Furthermore, we have seen that some small amount of water remains in the basin, which is physically sound. This is partially due to the water saturation getting below the **irreducible water saturation** and partially due to the small **viscosity** of air-phase. We have also see that the other two bubbles of water are expanding like we would expect.

We have also investigated the effect of having a non-constant permeability of the soil. We have seen that the fluid flows faster through soil with a higher permeability.

In order to obtain a solution we have used the discontinuous Galerkin method for the saturation. We have seen that discontinuous Galerkin with a limiter is suitable for solving equations with a discontinuity. However, the limiter needs to be adapted for regions with non-periodic boundary conditions. Applying the discontinuous Galerkin method to a two-phase flow model adds an extra difficulty. The pressure will now have to be non-constant. This makes it more difficult for the limiter to obtain limited values for the coefficients. But we still are confident in the result obtained from the discontinuous Galerkin method.

Further research consists in cerating a more realistic model. In this way the two-phase flow model can model more complex phenomena, such as **relative permeability**. Also, the capillary pressure could be included. With respect to the numerical method a higher order approximation can be used. With a higher order approximation the limiter discussed in Chapter 2 can also be expanded for two dimensions, as the limiter only uses the coefficients and not the reconstructed solution. For the limiter from Section 3.4 there can be experimented with values for specific constants like the upper bound for the second derivative. Also, a comparison in computational cost can be made between discontinuous Galerkin and the finite element method.

Bibliography

[1] S. Adjerid, K. Devine, J.E. Flaherty, and L. Krivodonova. A posteriori error estimation for discontinuous galerkin solutions of hyperbolic problems. *Comput. Methods Appl. Mech. Engrg. 191*, 2002.

[2] Hans Bruining. *Multi-Phase Flow in Porous Media.* Faculty of Geotechnical Engineering, Technical University Delft, 1999.

[3] N. Chevaugeon, J.E. Flaherty, L. Krivodonova, and J. Xin. *Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws.* Elsevier B.V., 2003.

[4] Bernardo Cockburn. *An introduction to the discontinuous galerkin method for convection-dominated problems.* School of Mathematics, University of Minnesota, Minneapolis, 1997.

[5] D.B. Das and S.M. Hassanizadeh. *Upscaling Multiphase Flow in Porous Media.* Springer, 2005.

[6] J.F.Keely. Performance evaluation of pump-and-threat remediations. *EPA Ground Water Issue, EPA/450/4-89/005*, 1989.

[7] Lilia Krivodonova. *Limiters for high-order discontinuous Galerkin methods.* Elsevier B.V., 2007.

[8] Randall J. Leveque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge University Press, 2002.

[9] J. van Kan, A. Segal, and F. Vermolen. *Numerical Methods in Scientific Computation.* VSSD, 2008.