Comparison Between Mimetic and Two-Point Flux-Approximation Schemes on PEBI-grids

Masteroppgave

Thomas Lunde

2. mai 2007
This thesis is submitted as a part of the requirements for a master degree in Scientific Computation at the Department of Informatics, University of Oslo. My supervisors have been Knut-Andreas Lie, Department of Applied Mathematics at SINTEF and Center of Mathematics for Applications, and Jørg E. Aarnes, Department of Applied Mathematics at SINTEF.

I would like to thank Knut-Andreas Lie and Jørg E. Aarnes for their guidance and expertise. They have helped me through all the stages of my research. I also wish to thank my comrades at Colenco Power Engineering in Switzerland: Rabah Namar (the master), Carl-Phillipp Enssle, Olivier Jaquet, Nicolas Hubschwelen, Pascal Siegel and Stephen Shulist. Thank you all for the good times, and for all I learned from you. I would also like to thank my fellow students at SINTEF, for their help and company. In particular I would like to thank Lars Moastuen and Martin Lilleeng Sætra. Finally, I would like to thank my family and friends for being the good people they are.
# Contents

1 Introduction  
1.1 Objectives of This Thesis ........................................... 3

2 Model Equations  
2.1 Rock and Fluid Properties ........................................... 5  
2.2 Flow Equation ............................................................ 7  
2.2.1 Mass-Balance Equation ........................................... 7  
2.2.2 Darcy’s Law ............................................................... 9  
2.2.3 Incompressible Single-Phase Flow ................................. 10  
2.3 Saturation Equation ...................................................... 11

3 Grid Geometry  
3.1 Grids in Reservoir Simulation ................................. 13  
3.1.1 Orthogonality ........................................................... 15  
3.1.2 Structured Grids ....................................................... 16  
3.1.3 Unstructured Grids .................................................. 17  
3.2 Delaunay Triangulations and Voronoi Diagrams ............ 18  
3.2.1 Delaunay Triangulations ........................................... 19  
3.2.2 Voronoi Diagrams .................................................... 21  
3.2.3 2-Dimensional Grids ................................................. 24

4 PEBI-grid Construction  
4.1 Overview ................................................................. 28
4.1.1 Data Structures ........................................... 28
4.2 Delaunay triangulation and Voronoi grid .................. 30
  4.2.1 Binding the Voronoi Diagram .......................... 31
4.3 Extension of 2D-grid into 3D .............................. 32
4.4 Geometric Calculations .................................... 34
  4.4.1 Calculating Area and Normal of Curved Faces ....... 34
  4.4.2 Calculating Cell Volumes ............................... 36

5 Two-Point Flux-Approximation Discretization .............. 41
  5.1 Discretization ............................................. 41
  5.2 Implementation ............................................ 44

6 Mimetic Finite-Differences Discretization ................. 45
  6.1 Mixed-Hybrid Finite-Elements ............................ 45
  6.2 Mimetic Finite-Differences Discretization ............ 51
  6.3 Implementation ............................................ 54

7 Calculation by hand example ............................... 57
  7.1 Problem ................................................... 57
  7.2 Two-point Flux-Approximation Scheme .................. 58
  7.3 Mimetic Finite-Differences Scheme ........................ 60

8 Numerical Experiments ...................................... 65
  8.1 Non-orthogonal Layer ..................................... 66
  8.2 2D PEBI-grid .............................................. 68

9 Conclusion .................................................. 71

Bibliography .................................................. 73
Simulation of physical processes helps us to gain understanding through analysis of mathematical models implemented on computers. But as the computational power increase, the demand for more accuracy from simulations increase. This is the situation we face in scientific computation today, and this situation is not likely to change in the nearest future. Complex problems have to be simplified to be solved on computers, and how we approach this simplification is absolutely critical with respect to computation time and the accuracy of the solution.

A reservoir is a subsurface structure of porous rock, trapping fluids such as oil, gas and water. The reservoir consists of rock that is sufficiently porous and has sufficient permeability to store fluids that have accumulated over millions of years by migration from source rocks. The porous rock is a network of microscopic pores and channels which can transmit fluids.

Reservoirs can be very complex structures. They are formed by sedimentary rocks at the surface of the Earth, through deposition of sediments of weathered rocks and organic matter. This rock may span kilometers horizontally, and the characteristics of the rock can vary to a great extent throughout the reservoir.

Simulation of flow through reservoirs began around sixty years ago, and has been a major subject of research since then. The problem is especially problematic with respect to the credibility of results and computation time. A reservoir is a very complex structure, and uncertainties already appear in the data provided by geologists, unable to expose critical reservoir features before drilling and production.
The problem we will address in this thesis is of another nature: The challenge of simulating such a complex problem on computers and get accurate results. The resolution which geologists measure reservoir data does not allow full pore-scale models. Even so, the resolution of geological data is higher than what the computers today are able to deal with. To account for this, we model the geometry of the reservoir using grids. A grid is an abstraction of the medium it represents, and today's reservoir grids are much coarser than pore-scale models. To simulate flow of fluids through a reservoir, we need to make the continuous problem given by physical laws and relationships possible to calculate on a computer. We do this by discretizing the continuous problem.

Numerous approaches for discretization have been proposed during the almost sixty years of research devoted to this field. This thesis is in particular directed towards two of them: the two-point flux-approximation finite-volume discretization, and the mimetic finite-differences discretization. In addition, we address a particular technique to model the geometry of the reservoir: the PEBI-grid.
1.1 Objectives of This Thesis

The main objectives of this thesis are to:

- Investigate the theoretical aspects and construction of PEBI-grids,
- Compare the two-point flux-approximation and mimetic finite-differences discretization for calculating flow on PEBI-grids.

We start in Chapter 2 to deal with the physical laws and relationships that govern flow through porous media. Chapter 3 is devoted to the geometric foundations of the PEBI-grid. In particular, we review the theory of Delaunay triangulations and Voronoi diagrams. In Chapter 4 we look at how to employ the theory from Chapter 3 in practice in order to construct a PEBI-grid. In Chapter 5 we discretize the flow equation from Chapter 2 using a two-point flux-approximation scheme. In Chapter 6 we discretize the flow equation using a mimetic finite-differences scheme. In Chapter 7, we calculate step-by-step the solution of a simple problem using both a two-point flux-approximation scheme, and a mimetic finite-differences scheme. Chapter 8 is devoted to comparing the two-point flux-approximation discretization and a mimetic finite-differences discretization through numerical experiments. Finally, we draw conclusions and indicate some further extensions and possibilities in Chapter 9.
Like in most engineering problems, flow in porous media is governed by physical laws and relationships. This chapter is devoted to the physics and mechanics behind reservoir simulation. We start by defining the properties of the porous rock and fluids, and continue out the chapter with the mathematical formulations of the physical laws and relationships governing the flow. The equations derived in Section 2.2.3 and Section 2.3 will be the mathematical model subject for later chapters.

For a comprehensive introduction to the dynamics behind flow in porous media, we refer to [7, 5].

## 2.1 Rock and Fluid Properties

**Porosity** is the relative volume of pores in the rock, denoted by $\phi$. To describe a partition of a rock, we need to know how much of the space is occupied by impermeable rock, and how much space is occupied by pores in which a fluid can flow through. A porosity equal 0 means that no pores are present, and that no fluid can flow through. A porosity of 1 will on the other hand mean that the volume is void. Porosity is related to rock compressibility $c_r$ through the following formula:

$$c_r = \frac{1}{\phi} \frac{d\phi}{dp},$$

where $p$ is the pressure. We will however assume that $\phi$ does not depend on the pressure, and neglect rock compressibility.
**Absolute Permeability** is the ability of the rock to transmit a fluid at certain conditions. We say that rock with large, well-connected pores is permeable, while impermeable rock has fewer, or less interconnected pores. Generally, permeability is a full tensor, ranging over many orders of magnitude throughout the reservoir. The tensor represents the ability to transmit the fluid in different directions, and one direction depends on other directions. While the permeability tensor is full, it can often be diagonalized to avoid certain difficulties. The permeability tensor is denoted as $K$, and is in 3D on the general form

$$K = \begin{pmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{pmatrix}.$$  

**Pressure** is the force distributed over a surface, and is denoted by $p$.

**Saturation** is the amount of a volume that is filled up with one or more phases. A volume is fully saturated when phases fill up the entire void in the pores. A phase is a liquid or gas, and we usually limit the phases to water ($w$), oil ($o$) and gas ($g$). We denote the saturation of each phase as $S_{\alpha}$, $\alpha = w, o, g$, and for a fully saturated volume, we can write

$$\sum_{\alpha=w,o,g} S_{\alpha} = 1.$$  

**Velocity** also referred to as flux, is the rate that a phase travel through a medium. We denote the velocity by $v$. Sometimes we refer to the velocity of a single phase $\alpha$ as $v_{\alpha}$, and the total velocity of all phases $v = \sum_{\alpha} v_{\alpha}$.

**Mass Fraction of a Phase Component** is denoted by $c_{\omega\alpha}$ for component $\omega$ of phase $\alpha$, and is the mass fraction a component occupies in a phase. For example, gas and oil can be the two components of the hydrocarbon phase.

**Density** is denoted by $\rho$. The density $\rho_{\alpha}$ of a phase $\alpha$ is the mass per unit of volume.

**Viscosity** of a phase $\alpha$ is denoted by $\mu_{\alpha}$. A liquid with high viscosity is more resistant to flow than a liquid with low viscosity; the more viscous a fluid is, the slower it flows. Water has lower viscosity than light oil, which
again has lower viscosity than heavy oil. Heavy oil is also referred to as viscous oil.

**Phase Compressibility** of a phase is a measure of the relative volume change of a phase as a reaction to a change in pressure. Compressibility is most important for gas, and is often neglected for water. We will for the rest of this thesis assume that phase compressibility can be neglected for all phases.

**Relative Permeability** is a measure of the effective permeability of a phase, and is usually a function of the phase saturations only. We denote the relative permeability of a phase $\alpha$ by $k_{r\alpha}$. Relative permeability curves are often determined through experiments.

**Capillary Pressure** $P_{c,\alpha_1,\alpha_2}$ is the pressure between two interfacing phases $\alpha_1$ and $\alpha_2$. Capillary pressure is often determined through experiments. We will for the rest of this thesis assume that effects from capillary pressure can be neglected.

**Mobility** is denoted by $\Lambda_\alpha$ for a phase $\alpha$, and is the ratio of permeability to viscosity:

$$\Lambda_\alpha = \frac{k_{r\alpha}}{\mu_\alpha}.$$  

### 2.2 Flow Equation

For the flow model, pressure, velocity and saturation are the primary unknowns. In this chapter, we will present the equations for modelling flow in porous media, and derive a set of equations with pressure and velocity as primary unknowns. The saturation equation is derived in the next chapter.

#### 2.2.1 Mass-Balance Equation

The law of mass conservation is fundamental for the mass-balance equation, which gives the basic equation for the flow model. The law states that mass can neither be created nor destroyed; mass can only change its form. For a closed system where mass is neither flowing in nor out, the mass inside the system is constant, regardless of the processes acting inside.
In most practical applications of flow simulation, we do not have closed systems. For a reservoir model, we might have flow coming in and out through boundaries, or wells that inject or produce mass. To account for material entering or leaving the system, we use the principle of mass-balance. The mass-balance principle states that mass entering the system must either leave the system, or be accumulated within it. This is a direct extension of the law of mass conservation. For mass-balance in an arbitrary domain $\Omega$, we require the following to hold:

$$\{\text{accumulation}\} + \{\text{outflow}\} = \{\text{inflow}\}.$$ 

In integral form, the mass-balance equation reads

$$\frac{\partial}{\partial t} \int_{\Omega} \phi \sum_{\alpha} (\bar{c}_{\omega \alpha} \rho_{\alpha} S_{\alpha}) \, dx + \int_{\partial \Omega} \sum_{\alpha} (\bar{c}_{\omega \alpha} \rho_{\alpha} v_{\alpha}) \cdot n \, ds = \int_{\Omega} \sum_{\alpha} (\bar{c}_{\omega \alpha} q_{\alpha}) \, dx, \quad (2.1)$$

for a domain $\Omega$ with boundary $\partial \Omega$ with normal vector $n$ as depicted in Figure 2.1. The subscript $\alpha$ refers to a phase in which a component $c$ accumulates and flows.

![Fig. 2.1: A domain $\Omega$ with a boundary $\partial \Omega$ with normal vector $n$.](image)

The change of a fluid within $\Omega$ is determined by the flux $v$ through $\partial \Omega$.
and the amount of the matter created within $\Omega$, denoted by $q$. A positive $q$ implies that we have a source, and a negative $q$ implies a sink.

We recall now the divergence theorem.

**Theorem 2.1.** Let $\Omega$ be a region in space with boundary $\partial \Omega$. Then the volume integral of the divergence $\nabla \cdot F$ of $F$ over $\Omega$ and the surface integral of $F$ over the boundary $\partial \Omega$ of $\Omega$ are related by

$$\int_{\Omega} \nabla \cdot F \, dx = \int_{\partial \Omega} F \cdot n \, ds$$

The divergence theorem applied to (2.1) gives

$$\frac{\partial}{\partial t} \int_{\Omega} \phi \sum_{\alpha} (\tau_{\omega \alpha} \rho_{\alpha} S_{\alpha}) \, dx + \int_{\Omega} \nabla \cdot \left( \sum_{\alpha} (\tau_{\omega \alpha} \rho_{\alpha} v_{\alpha}) \right) \, dx = \int_{\Omega} \sum_{\alpha} (\tau_{\omega \alpha} q_{\alpha}) \, dx.$$

As this holds for an arbitrary domain $\Omega$, the integrands have to be equal. We drop the integral signs and get

$$\frac{\partial}{\partial t} \phi \sum_{\alpha} (\tau_{\omega \alpha} \rho_{\alpha} S_{\alpha}) + \nabla \cdot \left( \sum_{\alpha} (\tau_{\omega \alpha} \rho_{\alpha} v_{\alpha}) \right) = \sum_{\alpha} (\tau_{\omega \alpha} q_{\alpha}). \quad (2.2)$$

### 2.2.2 Darcy’s Law

To model the actual fluid flow through the porous rock, we use the empirical relation called Darcy’s law. This relation is only valid for slow, viscous flow; flow in subsurface reservoirs falls into this category. Darcy’s law states that the velocity of the fluid is proportional to a combination of the gradient of the fluid pressure and the effects due to gravity:

$$v_{\alpha} = -\frac{k_{\tau \alpha}}{\mu_{\alpha}} K (\nabla p_{\alpha} - \rho_{\alpha} G). \quad (2.3)$$

Here $G$ denotes the gravitational forces and $p$ the pressure. Assuming that we suppress gravitational forces, i.e., $G = 0$, we can make the following two observations: no flow will occur if there is no pressure gradient; if there is a pressure gradient, the fluid will flow from high pressure towards low pressure.
2.2.3 Incompressible Single-Phase Flow

When considering only a single phase consisting of one component, (2.2) and (2.3) reduce to
\[
\frac{\partial}{\partial t} \phi \rho + \nabla \cdot \rho v = q, \tag{2.4}
\]
\[
v = -\frac{1}{\mu} K (\nabla p - \rho G). \tag{2.5}
\]

Incompressible flow is flow where we have a constant phase density \( \rho \). We assume incompressible flow and a constant porosity \( \phi = 1 \) such that the time derivative in (2.4) vanishes:
\[
\nabla \cdot \rho v = q. \tag{2.6}
\]

The assumptions that now follow are performed to simplify the flow model for further use in this thesis. Since we do not impact the essential characteristics of the flow equations, the discretizations presented in later chapters are fully valid for (2.2) and (2.3).

We assume that \( \rho = 1 \) such that (2.6) can be written as
\[
\nabla \cdot v = q. \tag{2.7}
\]

We further suppress gravitational forces, i.e., \( G = 0 \). We can then write (2.5) as
\[
v = -\lambda \nabla p, \tag{2.8}
\]
where
\[
\lambda = \frac{k_r}{\mu} K.
\]

The parabolic mass-balance equation for multi-phase flow (2.2) and Darcy’s law (2.3) have through the simplification been transformed into (2.7) and (2.8). We can also write the combination of the two as
\[
-\nabla \cdot \lambda \nabla p = q. \tag{2.9}
\]

(2.9) is called the pressure equation.

In order to close the model, we need to specify boundary conditions. We will assume that no flow occurs across the boundary, i.e.,
\[
v \cdot n = 0, \quad x \in \partial \Omega. \tag{2.10}
\]
We will henceforth refer to (2.10) as a no-flow boundary condition.

2.3 Saturation Equation

For a complete model of flow in porous media, we need equations for the phase saturations. We will in this section consider flow of two phases. We will refer to the two phases as the wetting-phase denoted by \( w \), and the non-wetting phase denoted by \( n \), i.e., \( \alpha = w, n \). We will derive the saturation equation under the assumptions made in Section 2.2.3, and further assume that the porous media is fully saturated. For a fully saturated porous media, the two phases occupy the void space completely:

\[
S_w + S_n = 1.
\]

We have that \( \nabla p_w = \nabla p_n \) since we neglect effects from capillary pressure. We can assume that

\[
p_w = p_n = p.
\]

Let \( v = v_w + v_n \), \( q = q_w + q_n \) and \( \Lambda = \Lambda_w + \Lambda_n \). We also assume that we have a constant porosity \( \phi = 1 \). From (2.2) we then get that

\[
\frac{\partial S_w}{\partial t} + \nabla \cdot f(S_w)v = q_w,
\]

(2.11)

where

\[
f(S_w) = \frac{\Lambda_w(S_w)}{\Lambda_w(S_w) + \Lambda_n(1 - S_w)},
\]

where \( \Lambda_w \) denotes the mobility of the wetting phase. We call (2.11) the *saturation equation*. We only need to know the velocity of the wetting-phase \( v_w \) to be able to calculate the wetting phase saturation \( S_w \) and the non-wetting phase saturation \( S_n = 1 - S_w \).

See [7, 1] for a more detailed derivation of (2.11).
In this chapter, we motivate the use of, and review the theory, of PEBI-grids. In order to understand the properties and construction of our grid of interest, two computational geometric concepts, presented in Section 3.2, are essential for this chapter:

- Delaunay triangulations.
- Voronoi diagrams.

In Section 3.2.3 we introduce the very motivation for the main object of this thesis: Comparing the two-point flux-approximation scheme and mimetic finite-differences scheme on PEBI-grids. The orthogonality property in $2\frac{1}{2}$-dimensional PEBI-grids with curved layers will not be preserved, and an error will be introduced if orthogonality is assumed.

The discussion in this chapter will be extended into Chapter 4, focusing more on implementational issues regarding PEBI-grid construction.

### 3.1 Grids in Reservoir Simulation

Simulation of flow through a porous medium requires that the physical space is represented by a finite geometry. We call this geometry a *grid* (see Figure 3.1). A grid is a collection of *grid-cells* that are uniform blocks representing different parts of the physical space. Rock properties like porosity and permeability are assigned to each grid-cell, and are assumed to be uniform inside the grid-cells. The aim is then to calculate the flow between grid-cells.
A grid is a set of non-overlapping, connected grid-cells. Grid-cells are non-overlapping when each point in the domain is represented inside one grid-cell only. Grid-cells are connected in the sense that they are neighboring or interfacing each other. We call the surface area where two grid-cells are interfacing each other an interface. Two interfacing grid-cells are said to have a connection, and the total set of connections in a grid is called the connectivity of the grid.

A grid-cell is generally a closed polygon in 2D, and a closed polyhedron in 3D. A grid-cell has the following properties of interest:

- volume,
- cell-center coordinate (also called grid-point or generating-point),
- corner-points/vertices.
We sometimes refer to grid-cells as elements or control volumes, depending on the context. This depends on how we interpret the grid-cell when discretizing the model equations. A vertex is simply a coordinate in space, see Figure 3.2 c). Some grids have grid-cells defined by the vertices. PEBI-grids are defined by their cell-centers, and vertices define the structure of the geometry only.

The sides at the boundary of the grid-cell are called faces (Figure 3.2 b)). A face has the following properties of interest:

- area,
- normal,
- face-center coordinate.

### 3.1.1 Orthogonality

Before turning to specific grids, we will discuss orthogonality of grids. Let us consider an arbitrary grid-cell $E_i$ with cell-center $x_i$. Let $\gamma_{ij}$, $j = 1, ..., N_{\gamma}^i$ be the interfaces of $E_i$ to neighboring grid-cells $E_j$, $j = 1, ..., N_{\gamma}^i$, where $N_{\gamma}^i$ denotes the number of interfaces of $E_i$. Further, let $l_{ij}$ be the line drawn between $x_i$ and $x_j$. Then $E_i$ is locally orthogonal if

$$ l_{ij} \cdot \gamma_{ij} = 0, \quad j = 1, ..., N_{\gamma}^i. $$

(3.1)

If (3.1) holds for all elements, i.e., $i = 1, ..., N_E$, then we say that the grid is orthogonal (Figure 3.3)$^1$.

$K$-orthogonality also respects the permeability tensor $\mathbf{K}$. A grid is $K$-orthogonal if

$$ l_{ij} \cdot \mathbf{K} \gamma_{ij} = 0, \quad j = 1, ..., N_{\gamma}^i $$

(3.2)

holds for all $i = 1, ..., N_E$.

Grids having interfaces aligned with the coordinate axes are $K$-orthogonal for diagonal tensors.

$^1$In some literature, orthogonal grids is a label for grids with interfaces aligned with the Cartesian coordinate axes. As this thesis focuses on non-Cartesian grids, we will use a broader definition for orthogonal grids.
3.1.2 Structured Grids

A structured grid is structured in the sense that grid-points and vertices are arranged regularly. Structured grids have long traditions in the field of reservoir simulation due to their simplicity. The most typical example on structured grids is the Cartesian grid. Cartesian grids have interfaces aligned with the Cartesian x-, y- and z-axes.

Cartesian grids can have problems representing a reservoir accurately. Having a set of rules that defines the geometry of all grid-cells does not account for the complex structure of reservoirs. Flow patterns are likely to occur in other directions than across cell-faces, and the solution might suffer from the infamous grid-orientation effect. It is infeasible to align a Cartesian grid-geometry along geological features such as faults and sedimentary layers.

Regular hexagonal grids (see Figure 3.4) are used to account for grid-orientation effects and reduce the effect from the lack of K-orthogonality, but they do not offer the flexibility of altering the geometry according to the reservoir. Radial grids give a good representation of the flow patterns around wells, but are impractical for representing a full reservoir. A solution to account for the advantages and disadvantages of the different structured grids is to combine them into what is called a hybrid-grid. Hybrid-grids use different structured sub-grids in different locations of the domain, such that geological features are represented by the most suitable grids. This allows a better representation of the reservoir, and many pitfalls from using structured grids are avoided. The implementation and use of hybrid-grids can at the same time be cumbersome, since different parts of the reservoir
must be treated differently.

While structured grids can be orthogonal, the grid cannot be adjusted to be K-orthogonal throughout a complex, heterogeneous reservoir with full permeability tensors. If the permeability tensors are diagonalized, a Cartesian grid with faces aligned along the Cartesian axes will be K-orthogonal.

3.1.3 Unstructured Grids

Unstructured grids benefit from being flexible when defining the geometry. They do not have a set of rules that applies for the entire grid, and vertices and faces can be aligned according to the reservoir structure. The industry-standard in reservoir simulation today is the corner-point grid (Figure 3.5), generated from a structured Cartesian grid. The cells in the corner-point grid can be altered, and corner-point grids are therefore more flexible than Cartesian grids.

There is a category of unstructured grids that is more flexible than altered structured grids, namely triangulation-based grids. Generally, triangulation-based grids do not have limitations at all. However, in order to attain good results from simulations, numerical schemes require some care when constructing such grids. In the rest of this chapter we will discuss triangulation-based grids, and in particular the orthogonal PEBI-grids.
3.2 Delaunay Triangulations and Voronoi Diagrams

Triangulations prove to be very powerful when applied to grid-generation. In general, a triangulation is a subdivision of a convex or non-convex domain into non-overlapping triangles. For most applications we typically require that the union of the triangles completely fills the convex hull of the set of points.

**Definition 3.1.** Let $\mathcal{P} = \{x_i\}, i = 1, \ldots, n$ be a set of points in the real vector space $V$. Then the convex hull $C$ of $\mathcal{P}$ is the minimal convex set containing $\mathcal{P}$:

$$C = \left\{ \sum_{i=1}^{n} \lambda_i x_i : \lambda_i \geq 0, i = 1, \ldots, n, \sum_{i=1}^{n} \lambda_i = 1 \right\}.$$  

This set of points, called generating points, is either given or selected with a suitable procedure. There are several ways to triangulate a set of points. One approach may differ significantly from another in terms of triangle quality.

When seeking a good triangulation, we want to avoid poorly shaped triangles in terms of their angles. The Delaunay triangulation is an optimal triangulation in this regard, where a triangle is selected as the triangle with
the largest minimal angle. This triangulation is often applied directly for discretization and mesh generation of a domain. Alternatively, the Delaunay triangulation can be used as a base for so-called triangulation-based grids, or flexible grids. An example in flow simulation is the common corner-point grid, which can be constructed by merging triangles into quadrilaterals. Arbitrary location of grid-points prove to be the biggest advantage of basing mesh generation on triangulation.

Our interest from here on concerns the Voronoi diagram, derived directly from the Delaunay triangulation as its dual. The Voronoi diagram is the partitioning of a domain into convex polygons such that each polygon contains exactly one generating point, and each point inside a polygon is closer to its generating point than to any other. A triangulation with significant "triangle quality" gives the Voronoi diagram a particular property of interest: the line between two generating points intersects the interface between them orthogonally. The Voronoi diagram is then orthogonal.

The Delaunay triangulation and the Voronoi diagram can be extended into \( N \) dimensions, but in this thesis we only consider the two-dimensional case. From two to three dimensions, the edges of the Voronoi diagram become hyperplanes\(^2\). With respect to grid-generation, this increase of complexity shows the higher-dimensional Voronoi diagram to be rather unsuitable for practical applications.

Before continuing, we want to clarify some terms. We often use the term Voronoi grid for the Voronoi diagram in an implementational context. An edge is the same as a face in 2D; a Voronoi edge is a face in a Voronoi diagram. In the same way, Delaunay edges are the faces which makes up triangles in a Delaunay triangulation. A Voronoi region is the same as a grid-cell, but in a mathematical context.

### 3.2.1 Delaunay Triangulations

Delaunay triangulation is considered the main choice of triangulations when concerned with triangle quality. Existing triangulation algorithms are relatively cheap computationally, and there exists an extensive literature on both theoretical aspects and implementational issues.

There are several ways to define Delaunay triangulation. We will now give two commonly used definitions, meant to give an intuitive understanding

\(^2\)A hyperplane in three dimensions is a plane dividing the space in two half-spaces.
of the properties of the triangulation. The first definition is based on the MaxMin angle criterion. With the MaxMin angle criterion, we prefer triangles with largest minimal angle, see Figure 3.6.

![Fig. 3.6: Two triangulations of the same set of points. The triangulation to the right is optimized in the sense of the MaxMin angle criterion.](image)

**Definition 3.2.** A Delaunay triangulation of a convex set of points $\mathcal{P}$ is the triangulation $T'(\mathcal{P})$ optimized in the sense of the MaxMin angle criterion.

Consider now the unique circle passing through the three vertices of a triangle. This is called the *circumcircle* of the triangle (see Figure 3.7). The center of the circumcircle is called the *circumcenter* of the triangle.

![Fig. 3.7: The circumcircles of two triangles.](image)
An alternative definition of Delaunay triangulation follows, which is equivalent to Definition 3.2.

**Definition 3.3.** The Delaunay triangulation of a convex set of points \( P \) is the triangulation \( T(P) \) such that no point in \( P \) is inside the circumcircle of any triangle in \( T(P) \).

Among all triangulations of a set of vertices, the Delaunay triangulation maximizes the minimum angle in the triangulation, minimizes the largest circumcircle, and minimizes the largest min-containment circle, where the min-containment circle of a triangle is the smallest circle that contains it. This has been proven in [20]. Additionally, the closest two generating points are connected by an edge of the Delaunay triangulation. This is called the closest pair property, and such two neighboring points are often referred to as natural neighbors.

### 3.2.2 Voronoi Diagrams

Perpendicular Bisector (PEBI)-grids is a term used to describe grids where the line drawn between two neighboring grid-points intersects the interface at the midpoint between them orthogonally. The interface is then the perpendicular bisector of that line. As we will show in this section, Voronoi diagrams are orthogonal in this respect. When the circumcenters of the underlying triangulation are located inside their triangles, the grid is orthogonal. In the field of reservoir simulation PEBI-grids are for this reason directly associated with Voronoi diagrams.

For a survey of other applications of the Voronoi diagram, see [4].

**Definition 3.4.** The Voronoi diagram of a set of \( n \) points \( P = \{x_i\} \) in the plane, is the partitioning of the plane into \( n \) convex polygons such that each polygon contains exactly one generating point, and every point inside a given polygon is closer to its generating point than any other.

The convex polygons are called Voronoi regions. If we let \( V(x_i) \) denote the Voronoi region of generating point \( x_i \) in \( P \), we can express the Voronoi region as

\[
V(x_i) = \{x \mid d(x, x_i) < d(x, x_j), \ j = 1, \ldots, n\}. \tag{3.3}
\]
A Voronoi region is not closed, that is, a point that is equally close to two generating points does not belong to a region defined by (3.3). We include these points by the defining the closure of \( V(x_i) \) as

\[
\overline{V(x_i)} = \{ x | d(x, x_i) \leq d(x, x_j), j = 1, ..., n \}.
\]

The regions of the generating points in \( \mathcal{P} \) lying at the convex hull are unbounded. The other regions are bounded. This means that the Voronoi regions of the points lying at the convex hull are edges that stretch into infinity. We will discuss how to define a boundary in Section 4.2.1.

The Voronoi diagram of a set of points can be directly derived from the Delaunay triangulation as its dual. By the duality principle there exists for every triangle, a polyhedron in which faces and vertices occupy complementary locations. The duality can be summarized as follows:

- The circumcenter of a Delaunay triangle corresponds to a Voronoi vertex.

- A Delaunay edge corresponds to a Voronoi edge as they intersect each other orthogonally.

- A Delaunay vertex corresponds to a Voronoi region; a vertex in the triangulation is the center of a Voronoi region.

The line, or Delaunay edge, between the center of two locally orthogonal neighboring regions, intersects the interface between them orthogonally. In addition, the interface is conveniently located in the middle of the two centers. The interface is the perpendicular bisector of the Delaunay edge. An edge is a perpendicular bisector to an interface if the edge is perpendicular to the interface, and passes through the midpoint of the interface.

Not all Voronoi diagrams are locally orthogonal. If the circumcenter of a triangle happens to be outside the triangle in the Delaunay triangulation, the problem depicted in Figure 3.9 arises: The line drawn between the two points does not intersect the interface. As a consequence, the regions can not be labeled locally orthogonal. In the context of grid construction, some ways to compensate for this are smoothing algorithms for minimizing the number of neighborhood relationships, or constraining the triangulation [20]
Fig. 3.8: a) A set of generating points, b) the Delaunay triangulation of the points, c) the Voronoi diagram of the points
by inserting additional points where the constraint is not met. Alternatively, one simply disregards the non-orthogonality. Constrained Delaunay triangulations are used to guarantee high quality grid-cells, and can guarantee an orthogonal Voronoi diagram at the cost of additional points. It does not introduce errors as in the case with smoothing procedures.

Fig. 3.9: The circumcenter is outside its triangle.

3.2.3 $2\frac{1}{2}$-Dimensional Grids

The Voronoi diagram and Delaunay triangulation can be generalized into three dimensions [22]. Using full 3D Voronoi grids is, however, unpractical due to the difficulties of grid-construction, constructing well-behaving grid-cells, and visualization.

Rather than extending the gridding concepts into three dimensions to account for variety in the vertical direction, we will present another approach commonly used in reservoir simulation: extending the 2D grid into 3D. The idea is to

- Generate the 2D grid, representing a horizontal cross section of the reservoir.
- Extend the plane into layers in the vertical direction to account for vertical heterogeneities in the reservoir.
If the 2D grid consists on \( n \) cells, \( n \) grid-cells are generated for each layer. A layer is one extension of the 2D grid into the vertical direction. This approach is suited for the typical structure of reservoirs, as layers can be aligned with the sedimentary layers. We will sometimes refer to 2\( \frac{1}{2} \)D grids as layered grids.

![Diagram of 2D and 2\( \frac{1}{2} \)D grids](image)

Fig. 3.10: a) A 2D grid, b) 2\( \frac{1}{2} \)D grid with one layer, c) 2\( \frac{1}{2} \)D grid with ten layers.

If the layers are planar as in Figure 3.10, the orthogonality property of the grid will be preserved. However, if the layers are curved or not parallel to the horizontal plane (see Figure 3.11), the grid will lose its orthogonality. Geological layers are in most cases curved, faulted\(^4\), and even tend to pinch out and collapse into other layers.

The two-point flux approximation scheme, described in Chapter 5, uses only two grid-points to calculate the flux between grid-cells. If the interface between two grid-cells is non-orthogonal, there will appear an error. In practice, if the simulation gives results that are ”good enough”, ignoring this error can be justified by the fact that there are so many uncertainties in the original problem, so that the non-orthogonality only represents a fraction of these. Often, the outcome of such simulations are accompanied by an error.

\(^3\)The term curved is here used for non-planar surfaces.

\(^4\)Faults are breaks or large cracks in the rock that cause a displacement in the sedimentary layers, see [7].
estimate taking the non-orthogonality into account. The mimetic finite-differences method, presented in Chapter 6, is a method designed to give more accurate solutions for non-(K-)orthogonal grids.

This concludes the main motivation for comparing the two-point flux-approximation scheme with mimetic finite-differences: Will the mimetic finite-differences approach produce a significant improvement in accuracy under the effects of curved layers in PEBI-grids.
The aim of this chapter is to employ the theoretical concepts from Chapter 3 in order to construct PEBI-grids. The presentation should be independent of implementation platforms. We want to emphasize that the following content does not cover the entire scope of PEBI-grid construction and its applications. This chapter suggest one approach to construct layered PEBI-grids, and can be viewed as an overview of, and extension, to previous work [22, 18]. In particular, we present formulas for calculating certain geometric properties in Section 4.4, extending the scope of earlier work.

We commence in Section 4.1 by giving an overview of the construction process, and proposing a set of data structures to store geometry. We continue in Section 4.2 by discussing the implementation of Delaunay triangulation and the Voronoi diagram. Section 4.3 addresses the extension of the 2D grid into 3D. Some non-trivial geometric calculations are handled in Section 4.4.

Note that we will not give the algorithms for calculating the Delaunay triangulation and Voronoi diagrams. Derivation of effective algorithms has been the subject of research for decades, and a wide selection of implementations are available. This will be further commented in Section 4.2.

Distributing generating points is the initial step in PEBI-grid construction. Choosing the coordinates of the points according to a geological model is the “right way” to interpose the grid structure. As shown in Chapter 3, the generating points will act as cell-centers (in 2D). Distribution of generating points and assigning physical properties to grid-cells has been extensively discussed in [22, 18], and will be left out in this work.
4.1 Overview

The main steps in constructing a layered PEBI-grid involves:

1. Distribution of generating points.
2. Delaunay triangulation to obtain the connectivity between generating points.
3. Construction of 2D Voronoi diagram to obtain the 2D PEBI-grid.
4. Extending the 2D grid into 3D (to a layered PEBI-grid).

The geometric properties of the grid that need to be calculated are:

- Cell-volumes.
- Face-area.
- Face-centers.
- Face-normals.

4.1.1 Data Structures

In this section we suggest a set of data structures to store grid-geometry. In order to apply for the majority of the programming platforms, the data structure of choice is the list.

Several lists contain ids only. An id is an integer reference to an element in another list, and we denote a reference to a list item with square brackets. For example, a vertex id $V[i]$ refers to item no. $i$ in the list of vertices $V$.

Generating points and vertices can be kept in lists of coordinates:

\[ P = \{ x_1, \ldots, x_{N_E} \} , \]
\[ V = \{ v_1, \ldots, v_{N_V} \} . \]

Here $x_i$ and $v_i$ are spatial coordinates.

The number of vertices can vary from face to face in PEBI-grids. A face should therefore be a list of vertex ids on the form

\[ F_i = \{ V_{[i]}, \ldots, V_{[N_{V[i]}]} \} . \]
and the list of faces on the form
\[ \mathcal{F} = \{F_1, \ldots, F_{N_F}\}. \]

A grid-cell is defined by a set of faces:
\[ E_i = \left\{ F_{[1]}, \ldots, F_{[N_F]} \right\}, \]
\[ E = \{ E_1, \ldots, E_{N_E} \}. \]

For the list of face normals, one should take into account that faces shared by two cells have one normal for each of the two cells. We therefore give the normals cell-wise:
\[ N_i = \left\{ n_{i_1}, \ldots, n_{i_{N_F}} \right\}, \]
\[ N = \{ N_1, \ldots, N_{N_E} \}. \]

Face areas are put straight forward into a list:
\[ A = \{|F_1|, \ldots, |F_{N_F}|\}, \]
as well as cell-volumes:
\[ V = \{|E_1|, \ldots, |E_{N_E}|\}. \]

We finally need a list to look up the interfaces. We represent one interface as
\[ \Gamma_k = \left\{ E_i, E_j, F_{[m]}, n_{n(i)}^{(i)} \right\}. \]

Interface \( \Gamma_k \) is the surface area between two neighboring cells \( E_i \) and \( E_j \), where \( n_n^{(i)} \) is the unit normal pointing outwards from cell \( E_i \). With this information, we can derive all necessary information about the interface. The set of all interfaces is kept in a list:
\[ \mathbb{I} = \{ \Gamma_1, \ldots, \Gamma_{N_i} \}. \]

We will from now on distinguish between 2D coordinates and 3D coordinates, as we operate with both dimensions throughout the grid construction. We will add a superscript 2D to denote a lists in a 2D context. For example, \( \mathcal{P}^{2D} \) denotes the list of generating points in 2D, while \( \mathcal{P} \) denotes the list of cell-centers in 3D.
4.2 Delaunay triangulation and Voronoi grid

An algorithm should take advantage of the fact that the Delaunay triangulation and the Voronoi diagram can be derived from each other. If the Delaunay diagram is constructed, Voronoi edges can be calculated as the line perpendicular to a Delaunay edge, limited by other intersecting Voronoi edges. We will however not go into detail of how to calculate the Delaunay triangulation nor how to calculate the Voronoi diagram. Instead we refer to [11, 13, 6, 20] for efficient Delaunay triangulation algorithms, and to [12, 21, 14, 19, 2] for algorithms for calculating the Voronoi diagram.

The reader should also be aware of matured open source software such as QHull\(^1\) and CGAL\(^2\).

The Delaunay triangulation defines the connectivity (interfaces) between grid-cells, and can together with the generation of the Voronoi diagram build \(\mathbb{I}^{2D}\). The output from the generation of the Voronoi diagram goes as follows: Voronoi vertices go into \(\mathbb{V}^{2D}\), Voronoi edges go into \(\mathbb{F}^{2D}\), and Voronoi cells into \(\mathbb{V}^{2D}\).

One should be careful to identify the Voronoi edge between two neighboring generating points in the Voronoi diagram generation algorithm. Identifying interfaces at a later point can be a very expensive process. If one calculates the Voronoi edges as the perpendicular bisectors of Delaunay edges directly, this information is available. The interfaces list is then on the form

\[
\Gamma_k = \{E_i, E_j, F[m], n^{(i)}_n\},
\]

\[
\mathbb{I}^{2D} = \{\Gamma_1, \ldots, \Gamma_N\}.
\]

Note that we have chosen to include the normal \(n^{(i)}_n\) already for the 2D grid. If one set the z-component of \(n^{(i)}_n\) to 0, this still holds in 3D. Faces derived from the Voronoi diagram will be strictly vertical and non-curved. When the 2D grid is extended into 3D, only the top- and bottom-faces will be irregularly shaped.

---

\(^1\)QHull is an easy-to-use triangulation software, and is used for triangulation in Mat-\lab. QHull supports Voronoi diagram generation and constrained Delaunay triangulation, but does not offer the extensive functionality of large computational geometry libraries as for example CGAL. For more, see [6] or http://www.qhull.org.

\(^2\)Computational Geometry Algorithm Library (CGAL) is a comprehensive library that offer efficient functionality to all aspects of PEBI-grid construction. For more, see http://www.cgal.org.
4.2.1 Binding the Voronoi Diagram

We have up until now not discussed how to define the boundary of the Voronoi-grid. Remember that the edges (faces) of the Voronoi diagram lying at the convex hull are of infinite length. We therefore need to specifically define the boundary by making these edges finite. This can be done in several ways, and we will here suggest to let the boundary be defined by the convex hull of the generating points. This will preserve orthogonality of the Voronoi diagram.

We see how the edges lying at the convex hull stretch towards infinity in Figure 4.1. We limit these edges by adding a vertex at the convex hull (see Figure 4.2). Depending on the geometric information available, the coordinate of the new vertex can be calculated in two equivalent ways:

- Calculate the midpoint between two neighboring generating points lying at the convex hull.
- Calculate the point of intersection where the Voronoi edge intersects the Delaunay edge.

There will be added two new faces for each grid-cell at the convex hull. Each new face is defined by the generating-point of the grid-cell, and the new vertex. Each new vertex is shared by two neighboring grid-cells.
There also exists algorithms for generating bounded Voronoi diagrams [2].

4.3 Extension of 2D-grid into 3D

To extend the 2D grid into 3D, we need a set of functions defining the z-coordinates of the grid-vertices. We let \( Z_l(x, y), l = 1, ..., N_L + 1 \) be the functions defining the z-coordinates of the vertices in a grid made up by \( N_L \) layers. We assume that

\[
Z_{l-1} < Z_l < Z_{l+1}, \quad l = 2, ..., N_L, \quad \forall x, y,
\]

i.e., the functions does not intersect each other at any point.

Vertices

There will be \( N_V^{2D} \cdot (N_L + 1) \) vertices in 3D, where \( N_V^{2D} \) denotes the number of vertices in 2D. Let \( E \) be a 2D cell with vertices \( v_1, ..., v_{N_V} \). This cell will contain the following vertices in layer \( l \) in 3D:

\[
\left( [v_1]_x, [v_1]_y, Z_l(v_1) \right)^T, ..., \left( [v_{N_V}]_x, [v_{N_V}]_y, Z_l(v_{N_V}) \right)^T,
\]

\[
\left( [v_1]_x, [v_1]_y, Z_{l+1}(v_1) \right)^T, ..., \left( [v_{N_V}]_x, [v_{N_V}]_y, Z_{l+1}(v_{N_V}) \right)^T,
\]

where \([v_k]_x\) and \([v_k]_y\) denote the x- and y-component of \( v_k \). The vertices should be ordered by their z-coordinates in the global 3D vertex list:

\[
V = \{ \nu_1, ..., \nu_l, ..., \nu_{N_L+1} \}.
\]

Here \( \nu_l \) denotes the list of vertices with z-coordinates given by \( Z_l \).
4.3 Extension of 2D-grid into 3D

Cell-centers

There will be $N_{E}^{2D} \cdot N_{L}$ cell-centers in 3D, where $N_{E}^{2D}$ denotes the number of generating points in 2D. The cell-center $x_i$ of $E_i$ in layer $l$ has the coordinate

$$\left( [x_{i,x}], [x_{i,y}], (Z_{l+1}(x_i) + Z_l(x_i)) / 2 \right)^T.$$

We have here assumed that $Z_l(x_i)$ can be evaluated. If $Z_l(x_i)$ can not be evaluated, the z-coordinate of $x_i$ can be calculated as the average z-coordinate of the vertices defining $E_i$ in 3D.

Faces

There will be defined two new faces for each grid-cell. Let $E$ be a 3D grid-cell with z-coordinates defined by $Z_l$ and $Z_{l+1}$, and with vertices $v_1, \ldots, v_{N_V}, v_{N_V+1}, \ldots, v_{2N_V}$. The two additional faces will be defined by

$$F_{\text{top}} = \left\{ v_{1}, \ldots, v_{N_V} \right\},$$

$$F_{\text{bot}} = \left\{ v_{N_V+1}, \ldots, v_{2N_V} \right\}.$$

In 2D, faces are defined by two vertices. Face $F$ defined by $v_k$ and $v_{k+1}$ in 2D has the following 3D-coordinates:

$$F = \left\{ v_k, v_{k+1}, v_{k+N_V}, v_{k+N_V+1} \right\}.$$

Here $N_V$ denotes the number of vertices in one layer (number of vertices in 2D). The faces of a 3D grid-cell will be defined by the vertices

$$v_k, v_{k+1}, \ldots, v_{N_V}, v_{k+N_V^{2D}}, v_{k+N_V^{2D}+1}, \ldots, v_{k+N_V^{2D}+N_V},$$

where $N_V$ here denotes the number of vertices of the 2D grid-cell, and $N_V^{2D}$ denotes the number of vertices in one layer.

Interfaces

There will be a connectivity between grid-cells in the vertical direction. New vertical interfaces can be defined during the construction of the top- and bottom-faces of 3D grid-cells. There will be $N_{E}^{2D} \cdot (N_{L} - 1)$ new interfaces. The interface between two vertically neighboring grid-cells $E_i$ and $E_{i+N_{E}^{2D}}$,
where \( N_{2D}^E \) denotes the number of generating points in 2D, is defined by

\[
\Gamma = \left\{ E_i, E_{i+N_{2D}^E}, \mathbb{F}_{[m]}, n_f \right\}.
\]

Here \( \mathbb{F}_{[m]} \) is a reference to the face shared by \( E_i \) and \( E_{i+N_{2D}^E} \), and \( n_f \) is the normal to the shared face \( F_m \) from \( E_i \) to \( E_{i+N_{2D}^E} \).

A layered PEBI-grid with varying number of grid-cells in the layers can be generated by disabling grid-cells after extending the 2D Voronoi grid into 3D. One disable a cell by treating all interfaces of that cell as the boundary of the domain.

### 4.4 Geometric Calculations

In this section we calculate the area, normals and volumes of the grid-cells. We do this by doing a triangulation, and take advantage of the properties of the \( 2\frac{1}{2}D \) PEBI-grid.

#### 4.4.1 Calculating Area and Normal of Curved Faces

Calculating the area of a curved face of \( n \) vertices is non-trivial. Our approach is to subdivide the face into simpler geometric structures that have an area easier to calculate. We could either let each face generated by the subdivision be defined as a new face in the grid-cell, or let the information from the subdivision tell something about the whole subdivided face. Defining new faces can give a better representation of the geometry, but the grid will have increased complexity. We rather triangulate the curved face and express the area of the face as the sum of the areas of the triangles.

We let \( F \) be a curved face. Then the area of \( F \) can be written as

\[
|F| = \sum_{l=1}^{N_T} |T_l|,
\]

where \( T_l, l = 1, ..., N_T \) are the triangles from a triangulation of \( F \).

The question is how we choose to triangulate \( F \). If the curve of the face is given as a function (see Section 4.3), introducing additional vertices inside the face can give a more accurate representation when triangulating. In
many cases subdivisions complicate the implementation, have a negative impact on the efficiency, and the gained accuracy might be insignificant. After all, the grid is a subdivision of the domain already.

For layered PEBI-grids, curved faces only occur at the top and bottom of a grid-cell. We therefore suggest to only subdivide the top and bottom faces, as depicted in Figure 4.3 b). For each grid-cell, two vertices are introduced. The x- and y-coordinates of the two vertices will be the x- and y-coordinates of the cell-center. The z-coordinate of the new vertex will be the z-coordinate of the layer function that defines the curved face. In particular, if $Z(x)$ is a layer function (see Section 4.3) defining the z-coordinates of $F_k$, the new vertex has the coordinate $Z(x_{[i]})$ where $x_{[i]}$ denotes the 2D coordinates of the cell-center of $E_i$.

The order in which the vertices are given is important in an implementation. We let $v_l$, $l = 1, ..., n$ be the vertices of face $F$, and $v_{\text{new}}$ be the vertex added in the middle of the face. The triangulation we suggest will result in $n$ triangles. The vertices should be given such that they are ordered by angle around $v_c$. If we assume that the list of vertices is circular, or more specifically that $v_{n+1} \equiv v_1$, we can express the area of a curved face $F$ as

$$|F| = \sum_{l=1}^{n} \sqrt{s(s-v_{\text{new}})(s-v_l)(s-v_{l+1})},$$

where

$$s = \frac{1}{2} (v_{\text{new}} + v_l + v_{l+1}).$$
Faces that are not top- or bottom-faces are not curved, and an additional vertex does not have to be introduced. They consist of four vertices, and the area can be expressed as the sum of two triangles:

\[ |F| = \sqrt{s_1(s_1 - v_1)(s_1 - v_2)(s_1 - v_3)} + \sqrt{s_2(s_2 - v_1)(s_2 - v_3)(s_2 - v_4)}, \]

where

\[ s_1 = \frac{1}{2} (v_1 + v_2 + v_3), \quad s_2 = \frac{1}{2} (v_1 + v_3 + v_4). \]

The normal of a curved face can be given as

\[ n = \frac{\sum_{l=1}^{n} |T_l| n_l}{\sum_{l=1}^{n} |T_l| |n_l|}, \tag{4.1} \]

where \( |T_l| \) denotes the area of triangle \( T_l \), and the normal of each triangle can be expressed as

\[ n_l = (v_{\text{new}} - v_{l+1}) \times (v_{\text{new}} - v_{l}). \tag{4.2} \]

The normal of non-curved faces can be calculated by

\[ n = (v_2 - v_1) \times (v_3 - v_1). \tag{4.3} \]

Note that (4.1) and (4.3) depend on which order the vertices are given. The given formulas are valid if the vertices are ordered clockwise. If the vertices are ordered counter-clockwise, (4.1) and (4.3) express the negative normal.

### 4.4.2 Calculating Cell Volumes

As calculating the volume of a general polyhedron is non-trivial, we follow the same approach as we did for calculating area of faces: subdividing the volume into simpler volumes. While we subdivide surfaces into triangles, we subdivide volumes into tetrahedra.

We assume we have introduced a new vertex in the middle of the top- and bottom-faces of the grid-cells. We first subdivide the polyhedra of \( n \) vertices into \( n \) polyhedra of six vertices each as in Figure 4.4.

We then do a second subdivision of each sub-volume into three tetrahedra each as depicted in Figure 4.5.
Fig. 4.4: A PEBI-grid cell of $n$ vertices is first subdivided into $n$ sub-volumes.

Fig. 4.5: Each sub-volume is triangulated into three tetrahedra.
The volume of polyhedron $E$ is now given by

$$|E| = \sum_{i=1}^{n} \sum_{m=1}^{3} |T_{mi}|,$$

where

$$|T_{mi}| = \frac{(a \times b) \cdot c}{6}$$

is the volume of the $m$'th triangle of sub-volume $T_i$, and $a$, $b$ and $c$ are the vectors defining the tetrahedra as shown in Figure 4.6.
Fig. 4.7: $2\frac{1}{2}$D PEBI-grid
To solve the model equations from Chapter 2 numerically, we need to discretize the problem. In this chapter, we discretize the flow equation using a two-point flux-approximation (TPFA) finite-volume scheme. As the name suggests, the TPFA scheme uses two points to approximate the flux. The TPFA scheme generally converges for orthogonal grids only.

### 5.1 Discretization

We want to solve the equation given by the conservation law

\[- \nabla \cdot \lambda \nabla p = q, \quad x \in \Omega\]  

(5.1)

for pressure $p$. The velocity is given by Darcy’s law:

\[v = -\lambda \nabla p, \quad x \in \Omega.\]  

(5.2)

We assume no-flow on the boundary, i.e.,

\[v \cdot n = 0, \quad x \in \partial \Omega.\]

We let the physical domain $\Omega$ be partitioned into a set of finite control volumes $E_i$:

\[\Omega_h = \bigcup_{i=1}^{N_E} E_i,\]
where quantities inside $E_i$ represent the average of the physical quantities inside this control volume.

We derive a set of mass-balance equations from (5.1) by integrating over an arbitrary control volume $E_i$:

$$
\int_{E_i} -\nabla \cdot \lambda \nabla p \, dx = \int_{E_i} q \, dx. \tag{5.3}
$$

We assume that $v = \lambda \nabla p$ is sufficiently smooth for (5.3) to hold. To this end, let $\gamma_{ij}$ be the interface between $E_i$ and a neighboring control volume $E_j$

$$
\gamma_{ij} = \partial E_i \cap \partial E_j,
$$

with area $A_{ij}$ and normal $n_{ij}$. Invoking the divergence theorem transforms (5.3) to

$$
\int_{\gamma_{ij}} -\lambda \nabla p \cdot n_{ij} \, ds = \int_{E_i} q \, dx.
$$

The two-point flux approximation uses two points to calculate the flux across an interface. The flux between $E_i$ and $E_j$ across $\gamma_{ij}$ can simply be expressed as

$$
v_{ij} = -\int_{\gamma_{ij}} \lambda \nabla p \cdot n_{ij} \, ds, \tag{5.4}
$$

where the gradient of pressure is calculated using a central finite difference stencil:

$$
\nabla p \approx \frac{2(p_j - p_i)}{d_i + d_j} \quad \text{on} \quad \gamma_{ij}. \tag{5.5}
$$

Here $p_i$ and $p_j$ denote the averaged pressure in $E_i$ and $E_j$, respectively, and $d_i$ and $d_j$ the distance between $\gamma_{ij}$ and the respective cell centers.

Inserting the approximation of $\nabla p$ in (5.5) into (5.4) gives

$$
v_{ij} = -\frac{2(p_j - p_i)}{d_i + d_j} \int_{\gamma_{ij}} \lambda \, ds. \tag{5.6}
$$

To estimate the directional cell mobilities $\lambda$ across the interface $\gamma_{ij}$ we apply a distance-weighted harmonic mean. Let $\lambda_{i,ij} = n_{ij} \cdot \lambda_i n_{ij}$ and $\lambda_{j,ij} = n_{ij} \cdot \lambda_j n_{ij}$ be the directional cell permeabilities. Then distance-weighted harmonic mean of $\lambda$ across $\gamma_{ij}$ is then given by

$$
\lambda_{ij} = (d_i + d_j) \left( \frac{d_i}{\lambda_{i,ij}} + \frac{d_j}{\lambda_{j,ij}} \right)^{-1}. \tag{5.7}
$$
Inserting (5.7) into (5.6) gives
\[ v_{ij} = -2 \left( \frac{p_j - p_i}{d_i + d_j} \right) \int_{\gamma_{ij}} (d_i + d_j) \left( \frac{d_i}{\lambda_{i,ij}} + \frac{d_j}{\lambda_{j,ij}} \right)^{-1} n_{ij} ds \]
\[ = 2 |\gamma_{ij}| (p_i - p_j) \left( \frac{d_i}{\lambda_{i,ij}} + \frac{d_j}{\lambda_{j,ij}} \right)^{-1}. \tag{5.8} \]

We now define the transmissibilities as
\[ [T]_{ij} = 2 |\gamma_{ij}| \left( \frac{d_i}{\lambda_{i,ij}} + \frac{d_j}{\lambda_{j,ij}} \right)^{-1}, \tag{5.9} \]
and we can write
\[ \sum_j [T]_{ij} (p_i - p_j) = \int_{E_i} q dx. \tag{5.10} \]
Let \( \bar{q} = \{ \bar{q}_i \} \) be a vector defined by
\[ \bar{q} = \int_{E_i} q dx. \]
Then (5.10) is a symmetric, linear system on the form
\[ Ap = \bar{q}, \tag{5.11} \]
as we can write
\[ \sum_j [T]_{ij} (p_i - p_j) = [T]_{i1} (p_i - p_1) + \ldots + [T]_{ij} (p_i - p_j) + \ldots + [T]_{ii} (p_i - p_i) + \ldots + [T]_{iN_E} (p_i - p_{N_E}) = \bar{q}_i. \]
We can from this see that
\[ [A]_{ij} = \begin{cases} \sum_k [T]_{ik} & \text{if } j = i \\ - [T]_{ij} & \text{if } E_i \cap E_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases}. \]
To make (5.11) positive-definite, we force \( p_1 = 0 \) by adding a positive constant to the first diagonal element of \( A \).

After solving the linear system (5.11) for \( p \), we can calculate the fluxes across each interface from (5.8). The velocity across \( \gamma_{ij} \) is given by (5.8).
5.2 Implementation

One of the strengths of the TPFA scheme is its easy implementation. The following algorithm is generalized for grid-cells with an arbitrary number of faces. Let \( m(I[k]) \rightarrow (E_{[i]}, E_{[j]}) \) be a mapping taking an interface reference (id) as input, and gives the element reference (id) of the two neighboring elements of this interface as output.

**COMPUTE TRANSMISSIBILITIES:**

set \([T]_{ij} = 0\) for \(i = 1, ..., N_E, \ k = 1, ..., N_E\) 

for \(k = 1, ..., N_\gamma\)

\((i, j) \leftarrow m(k)\)

\(\lambda_{i,ij} = n_{ij} \cdot \lambda_{n_{ij}}\)

\(\lambda_{j,ij} = n_{ij} \cdot \lambda_{n_{ij}}\)

\([T]_{ij} = [T]_{ji} = 2 \ |\gamma_{ij}| \left( \frac{d_i}{\lambda_{n_{ij}}} + \frac{d_j}{\lambda_{n_{ij}}} \right)^{-1}\)

**CONSTRUCT MATRIX A:**

\(A = -T\) 

for \(i = 1, ..., N_F\)

\([A]_{ii} = \sum_{j=1}^{N_F} [T]_{ij}\)

\([A]_{11} = [A]_{11} + \text{trace}(\lambda_1)\)

**OBTAIN SOLUTION:**

solve \(Ap = q\) for \(p\)

**CALCULATE VELOCITIES ACROSS INTERFACES:**

for \(k = 1, .., N_\gamma\)

\((i, j) \leftarrow m(k)\)

\(v_{ij} = 2 |\gamma_{ij}| (p_i - p_j) \left( \frac{d_i}{\lambda_{n_{ij}}} + \frac{d_j}{\lambda_{n_{ij}}} \right)^{-1}\)

**Algorithm 5.1:** Solving \(-\nabla \cdot \lambda \nabla p = f\) using a TPFA scheme.
In this chapter we present the mimetic finite-differences scheme. The mimetic finite-differences method (mimetic FDM) is based on the same approach as the mixed finite-elements method (MFEM), where the difference is what constraints we put for the approximation of the velocity. The key concept in the MFEM is to approximate both the pressure $p$ and the velocity field $v$ simultaneously. With a finite-volume method (for example TPFA), we compute the velocity field with a numerical differentiation; using the mixed formulation we treat the velocity field as an unknown.

The TPFA scheme generally converges for orthogonal grids.

### 6.1 Mixed-Hybrid Finite-Elements

MFEM employs a mixed formulation and approximates a velocity field that is continuous on each element, and has a continuous normal component over interfaces between elements.

Unfortunately, a MFEM scheme results in a symmetric, positive semi-definite linear system, which is obviously a drawback. Semi-definite systems are hard and time consuming to solve, being an important reason to why finite-volume methods were preferred until the introduction of the mixed-hybrid formulation ([3]). With the mixed-hybrid formulation we introduce an extra unknown: the pressure at element faces.

Rather than combining the mass-balance equation and Darcy’s law as we did for the TPFA, we write the pressure equation as a system of two first-
order equations:
\[ \nabla \cdot v = q, \quad x \in \Omega, \quad (6.1) \]
\[ v = -\lambda \nabla p, \quad x \in \Omega. \quad (6.2) \]
As before, we assume no-flow boundary conditions:
\[ v \cdot n = 0, \quad x \in \partial \Omega. \]
For compatibility, we must require that
\[ \int_{\Omega} q \, dx = 0. \]
To close the system, we require that
\[ \int_{\Omega} p \, dx = 0; \]
that is, \( p \) is defined up to an arbitrary constant.
We can multiply (6.1) with a test function \( l \) and integrate over \( \Omega \):
\[ \int_{\Omega} l \nabla \cdot v \, dx = \int_{\Omega} q l \, dx. \]
Since the permeability tensor is non-singular, \( \lambda \) is non-singular. We can rewrite (6.2) as
\[ \lambda^{-1} v = -\nabla p. \]
Multiplying with a test function \( u \) and integrating over \( \Omega \) gives
\[ \int_{\Omega} u \cdot \lambda^{-1} v \, dx = -\int_{\Omega} u \cdot \nabla p \, dx \]
\[ = -\int_{\partial \Omega} u p \cdot n \, dx + \int_{\Omega} p \cdot \nabla u \, dx. \quad (6.3) \]
Let \( L^2 \) be the space consisting of functions that are twice integrable:
\[ L^2(\Omega) = \left\{ v \in \Omega : \int_{\Omega} v^2 \, dx < \infty \right\}. \]
We can now define the following Sobolev space
\[ H^{1,\text{div}}(\Omega) = \left\{ v \in L^2(\Omega)^d : \nabla \cdot v \in L^2(\Omega) \right\}, \]
consisting of functions that are twice integrable, and having a divergence
being twice integrable. We define the closure of $H^{1, \text{div}}(\Omega)$ as

$$H^{1, \text{div}}_0(\Omega) = \left\{ v \in H^{1, \text{div}}(\Omega) : v \cdot n = 0, \; x \in \partial \Omega \right\}. \tag{6.1}$$

We let the domain $\Omega$ be partitioned into a set of finite control volumes $\Omega_h$:

$$\Omega_h = \bigcup_{i=1}^{N_E} E_i. \tag{6.2}$$

We now introduce the following bilinear forms:

$$b(\cdot, \cdot) : H^{1, \text{div}}_0(\Omega_h) \times H^{1, \text{div}}_0(\Omega_h) \rightarrow \mathbb{R}, \tag{6.4}$$

$$c(\cdot, \cdot) : H^{1, \text{div}}_0(\Omega_h) \times L^2(\Omega) \rightarrow \mathbb{R}, \tag{6.5}$$

where

$$b(u, v) = \sum_{E \in \Omega_h} \int_E u \cdot \chi^{-1} v \; dx, \tag{6.6}$$

$$c(v, p) = \sum_{E \in \Omega_h} \int_E p \nabla \cdot v \; dx. \tag{6.7}$$

We let $(\cdot, \cdot)$ denote the $L^2$-inner-product,

$$(\cdot, \cdot) : L^2(\Omega) \times L^2(\Omega) \rightarrow \mathbb{R},$$

$$(p, l) = \int_{\Omega} pl \; dx.$$ The variational form of (6.1)-(6.2) reads as follows: find $(p, v) \in L^2(\Omega) \times H^{1, \text{div}}_0(\Omega)$ such that

$$b(u, v) - c(u, p) = 0, \tag{6.8}$$

$$c(v, l) = (q, l), \tag{6.9}$$

holds for all $u \in H^{1, \text{div}}_0(\Omega)$, and $l \in L^2(\Omega)$.

The integral on $\partial \Omega$ in (6.3) vanishes if we require that $u \cdot n = 0$.

To discretize, we substitute the continuous spaces $L^2(\Omega)$ and $H^{1, \text{div}}_0(\Omega)$ with finite spaces $U \subset L^2(\Omega)$ and $V \subset H^{1, \text{div}}_0(\Omega_h)$. We let $U$ be the space of constant functions:

$$U = \text{span} \{ \chi_i : E_i \in \Omega_h \}, \quad \chi_i(x) = \begin{cases} 1, & \text{if } x \in E_i, \\ 0, & \text{otherwise.} \end{cases} \tag{6.10}$$
When substituting $L^2(\Omega)$ with $U$, (6.7) becomes

$$c(v,p) = \sum_i p_i \int_{E_i} \nabla \cdot v \, dx.$$ 

The MFEM differs from mimetic FDM by how we choose the finite space $V$. In a MFEM, we seek the velocity field in the space

$$V = \left\{ v \in H^{1, \text{div}}_0(\Omega) : v|_{E_i} \text{ have linear components } \forall E_i \in \Omega_h, \right. $n\left. (v \cdot n_{ij})|_{\gamma_{ij}} \text{ is constant } \forall \gamma_{ij} \in \Omega_h, v \cdot n_{ij} \text{ is continuous across } \gamma_{ij} \right\}. \quad (6.11)$$

Unfortunately the linear system arising from (6.8)-(6.9) is only symmetric positive semi-definite. To get a positive-definite system, we employ a mixed-hybrid formulation. A mixed-hybrid formulation does not influence the velocity solution at all, but enables recovery of a more accurate solution for the pressure.

We first remove the constraint that the normal velocity must be continuous across interfaces in $V$. Continuity of the normal component is reintroduced by adding an extra unknown representing the pressure at interfaces. We denote this unknown as $\pi$. By imposing constraints on the element boundaries by means of Lagrangian multipliers from the multiplier space

$$M(\Omega; \partial \Omega_h \setminus \partial \Omega) = \left\{ \pi \in L^2(\partial \Omega_h \setminus \partial \Omega) : \pi \in P_0(F) \text{ on } F \in \partial \Omega_h \setminus \partial \Omega \right\},$$

$$P_0 = \text{span}\{1\},$$

we apply the hybridization technique to obtain a symmetric, positive-definite linear system of equations. As the pressure now is defined on the boundary in (6.3) by introducing $\pi$, we introduce the bilinear form

$$d(\cdot, \cdot) : L^2(\Omega)^d \times L^2(\partial \Omega_h) \to \mathbb{R},$$

$$d(v, \pi) = \sum_{E \in \Omega_h} \int_{\partial E} \pi v \cdot n_E \, dx.$$ 

The mixed-hybrid formulation of (6.1)-(6.2) reads: find $(v, p, \pi) \in H^{1, \text{div}}_0(\Omega) \times$
\[ \begin{align*}
L^2(\Omega) \times L^2(\partial \Omega_h \setminus \partial \Omega) \text{ such that} \\
b(u, v) - c(u, p) + d(u, \pi) &= 0, \\
c(v, l) &= (q, l), \\
d(v, \mu) &= 0,
\end{align*} \]

holds for all \((u, l, \mu) \in H^1_0(\Omega) \times L^2(\Omega) \times L^2(\partial \Omega_h \setminus \partial \Omega).\)

For the discretization of (6.12)-(6.14) we seek the discrete solutions in the finite product-spaces \(V \times U \times \Pi \in H^1(\Omega) \times L^2(\Omega) \times L^2(\partial \Omega_h \setminus \partial \Omega)\) such that (6.12)-(6.14) hold for all \((v, u, \mu) \in V \times U \times \Pi.\) We let the approximation space \(U\) be defined as in (6.10), and \(\Pi\) consist of functions that are constant on element faces:

\[ \Pi = \text{span} \{ \pi_\gamma : \gamma \in \partial \Omega_h \setminus \partial \Omega, |\gamma| > 0 \}. \]

The linear system arising from (6.12)-(6.14) can be written in block form as

\[ \begin{bmatrix}
B - C^T \Pi^T \\
C \\
\Pi
\end{bmatrix}
\begin{bmatrix}
v \\
p \\
\pi
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
q \\
0
\end{bmatrix}, \]

(6.15)

where

\[ B = \begin{bmatrix}
b(\psi^{(i)}_k, \psi^{(j)}_l)
\end{bmatrix}, \]

\[ C = \begin{bmatrix}
c(\psi^{(i)}_k, \chi_j)
\end{bmatrix}, \]

\[ \Pi = \begin{bmatrix}
d(\psi^{(i)}_k, \mu_\gamma)
\end{bmatrix}. \]

Faces can be ordered in such a way that \(B\) is a block-diagonal matrix with each block corresponding to one element. More specifically, the faces of element \(E_i\) are numbered such that they are represented locally inside the \(i^{th}\) block of \(B.\) The matrix \(B\) is then on the form

\[ B = \begin{bmatrix}
B^{(1)} \\
\vdots \\
B^{(i)} \\
\vdots \\
B^{(N)}
\end{bmatrix}. \]

The test functions for the velocity \(\psi^{(i)}_k, k = 1, ..., N_F^{(i)}\) are supported inside
We assume that each block \( B^{(i)} \), \( i = 1, \ldots, N_E \) is invertible, implying that \( B \) is invertible. We can then perform a Schur-complement reduction with respect to \( B \) ([17]). We can from (6.15) write

\[
Bv - C^Tp + \Pi^T\pi = 0, \quad (6.16)
\]
\[
Cv = q, \quad (6.17)
\]
\[
\Pi v = 0. \quad (6.18)
\]

Multiplying (6.16) by \( B^{-1} \) gives

\[
v = B^{-1}(C^Tp - \Pi^T\pi). \quad (6.19)
\]

Inserting this into (6.17) and (6.18) gives

\[
Cv = CB^{-1}C^Tp - CB^{-1}\Pi^T\pi = q, \quad (6.20)
\]
\[
\Pi v = \Pi B^{-1}C^Tp - \Pi B^{-1}\Pi^T\pi = 0. \quad (6.21)
\]

We now define

\[
D = CB^{-1}C^T
\]

and

\[
F = \Pi B^{-1}C^T.
\]

We can then write (6.20) and (6.21) as

\[
Dp - F^T\pi = q, \quad (6.22)
\]
\[
Fp - \Pi B^{-1}\Pi^T\pi = 0. \quad (6.23)
\]

We now observe that

\[
c(\psi^{(i)}_k, \chi_j) = \sum_m \chi_j|_m \int_{E_m} \nabla \cdot \psi^{(i)}_k \, dx \neq 0 \Rightarrow i = j,
\]

and therefore \( D \) must be a diagonal matrix. \( D \) is then easily invertible, and
we can from (6.22) write
\[ p = D^{-1}F^T \pi + D^{-1}q. \]  
(6.24)

Inserting this into (6.23) gives
\[ \left( \Pi B^{-1} \Pi^T - FD^{-1}F^T \right) \pi = FD^{-1}q. \]  
(6.25)

To solve (6.15), we first solve (6.25) for \( \pi \). To obtain the solution of \( p \), we insert the solution of \( \pi \) into (6.24) to obtain \( p \). We finally insert the two solutions into (6.19) to obtain the solution of the velocity \( v \).

6.2 Mimetic Finite-Differences Discretization

To calculate \( b(\cdot, \cdot) \) in (6.6), we need the test function to be defined inside the entire element. This calculation is non-trivial for general polyhedra. The basic idea of the mimetic FDM is to seek the solution in a less constrained but more flexible space than \( V \). If we can find an inner-product \( m(\cdot, \cdot) \) defined on \( \partial \Omega_h \) that mimics the inner-product \( b(\cdot, \cdot) \), we do not need to have the velocity defined explicitly inside the grid-cells. We seek the solution in the space

\[ M = \text{span} \{ \psi^{(i)}_k \}, \psi^{(i)}_k(x) = \begin{cases} 1, & \text{if } x \text{ belongs to face } F^{(i)}_k, \\ 0, & \text{otherwise}. \end{cases} \]  
(6.26)

While \( V \) is the finite counterpart of \( H^{1,\text{div}}(\Omega) \), \( M \) represents a broader set of solutions. We often use the lowest-order Raviart-Thomas functions as test functions for the MFEM. For the mimetic FDM, we choose the test functions as in (6.26).

The introduction of \( M \) implies a substitution of \( b(\cdot, \cdot) \) with an inner-product \( m(\cdot, \cdot) \). Let \( \mathbf{M}^{(i)} \) be the block-diagonal matrix of \( \mathbf{M} \) associated with element \( E_i \) with \( N^{(i)}_F \) faces. We define the inner-product as
\[ m(\cdot, \cdot) : L^2(\partial \Omega_h) \times L^2(\partial \Omega_h) \rightarrow \mathbb{R}, \]
\[ m(u, v) = \sum_{E_i \in \Omega_h} m(\psi^{(i)}_k, \psi^{(i)}_l) = \sum_{E_i \in \Omega_h} (u^{(i)})^T \mathbf{M}^{(i)} v^{(i)}, \quad u^{(i)}, v^{(i)} \in \mathbb{R}^{N^{(i)}_F}. \]
Further, we assume that \( v_k^{(i)} \) represents the total flux across face \( F_k^{(i)} \):

\[
v_k^{(i)} = \int_{F_k^{(i)}} v \cdot n_k^{(i)} \, ds, \quad F_k^{(i)} \in \partial \Omega_h.
\]

(6.27)

**Fig. 6.1:** Mimetic FDM approximates velocity along the normal components of faces. Positive velocity means that the flux goes out of the element along the normal, while negative velocity corresponds to a flux into the element along the normal.

Since \( M \) is a less constrained space than \( V \), we want the matrices \( M^{(i)} \) to be defined such a way that the discrete counterpart of \( m(\cdot, \cdot) \) imitates or mimics the discrete counterpart of \( b(\cdot, \cdot) \) (hence the name mimetic). It was shown in [8] that to obtain a family of accurate inner-products, two conditions are required. Let \( m_h^{(i)}(\cdot, \cdot) \) be the discrete inner-product associated with \( m(\psi_k^{(i)}, \psi_l^{(i)}) \). The following two conditions ensure both convergence and stability to the solutions of the mimetic FDM.

**CONDITION 1** There exists two positive constants \( s^* \) and \( S^* \) such that

\[
|E_i| v^T v \leq m_h^{(i)}(v, v) \leq S^* |E_i| v^T v
\]

(6.28)

for all \( E_i \subset \Omega_h \) and \( v \in \mathbb{R}^{N_F^{(i)}} \).

This assumption states that the diagonal block matrix \( M^{(i)} \) is spectrally equivalent to the matrix \( |E_i| I \) where \( I \) is the \( N_F^{(i)} \times N_F^{(i)} \) identity matrix. The two positive constants \( s^* \) and \( S^* \) depend on the skewness of \( E_i \) and the tensor \( \lambda \).

**CONDITION 2** For every element \( E_i \subset \Omega_h \), every \( v_i \) obtained by inserting \( v = \lambda \nabla p \) into (6.27) for a linear function \( p \) on \( E_i \), and every \( u_i \in \mathbb{R}^{N_F^{(i)}} \) we
This assumption states that the inner-products $m_h(i)(\cdot, \cdot)$ should obey the Gauss-Green formula

$$
\int_\Omega v \cdot \nabla p \, dx + \int_\Omega p \nabla \cdot v \, dv = \int_{\partial \Omega} pv \cdot n \, ds.
$$

for linear pressure, i.e., the solution of the velocity must be exact for linear pressure.

It turns out that the matrix $M$ is not unique; (6.28) and (6.29) define a family of mimetic finite-difference methods, depending on how we choose to construct $M$. It also turns out that we can find an explicit formula for computing $M^{-1}$ without having to invert $M$ directly (see [10]). This is beneficial with respect to computational efficiency and from an implementational point of view. We also avoid inverting $M$-matrices that are almost singular. Almost singular $M$-matrices can arise when elements have faces with almost zero area.

We refer to [10, 9, 16, 15] for the derivation of the formula for computing $M^{-1}$.

The following formula for computing $M^{-1}$ was derived in [15]. For each element $E_i$ there is an associated matrix $(M(i))^{-1} \equiv W(i)$ on the form

$$
W(i) = \frac{1}{|E_i|} N(i) \lambda \left( N(i) \right)^T + \frac{2 \text{trace}(\lambda)}{|E_i|} \left( I - Z(i) \left( Z(i) \right)^T \right),
$$

where

$$
\begin{align*}
\left[ N(i) \right]_k &= \frac{1}{|E_i|} \int_{F_k(i)} \left( n_k(i) \right)^T \, ds, \\
\left[ C(i) \right]_k &= \frac{1}{|E_i|} \int_{F_k(i)} (x - x_i) \, ds, \\
\left[ D(i) \right]_{kl} &= \begin{cases} \\
|F_k(i)|, & \text{if } k = l, \\
0, & \text{otherwise}. \\
\end{cases}
\end{align*}
$$

and the columns of $Z(i)$ forms an orthonormal basis for the column space of the product $D(i)C(i)$. $x_i$ denotes the center of $E_i$. 

\[6.2 \text{ Mimetic Finite-Differences Discretization}\]
6.3 Implementation

We assume that the faces are ordered in such a way that the faces of $E_i$ start at id

$$
k = \sum_{j=1}^{i-1} N_F^{(j)},
$$

in an element-by-element manner. We present an algorithm for constructing $W$ given by (6.30) in Algorithm (6.1). We have there suggested to use the Gram-Schmidt process for constructing $Z^{(i)}$, even though any orthonormalization method can be utilized.

\[
\begin{align*}
\text{set } [W]_{kl} &= 0 \text{ for } k, l = 1, ..., N_F \\
\text{for } i = 1, ..., N_E &
\begin{align*}
\text{set } [N^{(i)}]_{kl} &= 0 \text{ for } k = 1, ..., N_F^{(i)}, l = 1, ..., N_{\text{DIM}} \\
\text{set } [C^{(i)}]_{kl} &= 0 \text{ for } k = 1, ..., N_F^{(i)}, l = 1, ..., N_{\text{DIM}} \\
\text{set } [D^{(i)}]_{kl} &= 0 \text{ for } k, l = 1, ..., N_F^{(i)} \\
\text{for } k = 1, ..., N_F^{(i)} &
\begin{align*}
[N^{(i)}]_k &= \left(n_k^{(i)}\right)^T \\
[C^{(i)}]_k &= x_{F_k^{(i)}} - x_{E_i} \\
[D^{(i)}]_{kk} &= |F_k^{(i)}| \\
Z^{(i)} &= \text{GramSchmidt} \left( D^{(i)}C^{(i)} \right) \\
W^{(i)} &= \left[ N^{(i)} \lambda \left( N^{(i)} \right)^T + 2 \text{trace}(\lambda) \left( I - Z^{(i)} \left( Z^{(i)} \right)^T \right) \right] \cdot |E_i|^{-1}
\end{align*}
\end{align*}
\]

Algorithm 6.1: Constructing matrix $W$

To construct $C$, we place for each element, the area of the element faces at their designated face-id, see Algorithm (6.2).

Each row in $\Pi$ corresponds with an interface. For each interface, the area of the two faces belonging to that interface are placed at the corresponding ids of the faces.

It is however important to note that all the faces at $\partial \Omega_h$ must be represented in $\Pi$. To honor the no-flow boundary condition $v \cdot n = 0$, $x \in \partial \Omega$, we define for each face on the boundary $F \in \partial \Omega_h$ an interface where we put the area
Algorithm 6.2: Constructing matrix $C$

of $F$ at its designated id. An algorithm to construct $\Pi$ is given in Algorithm (6.3).

Algorithm 6.3: Constructing matrix $\Pi$

We finally solve the system for $\pi$, $p$ and $v$, respectively, in Algorithm (6.4). Note that $D^{-1}$ can be inverted by inverting the diagonal elements only.
\[ D^{-1} = \left( CWC^T \right)^{-1} \]
\[ F = \Pi W C^T \]
\[ S = \Pi W \Pi^T - F D^{-1} F^T \]
\[ r = F D^{-1} q \]
solve \( S \pi = r \) for \( \pi \)
\[ p = D^{-1} \left( F^T \pi + q \right) \]
\[ v = W \left( C^T p - \Pi^T \pi \right) \]

**Algorithm 6.4:** Solving \(-\nabla \cdot \lambda \nabla p = f\) using a mimetic FDM.
7.1 Problem

We will now do a calculation by hand for a simple example to demonstrate the TPFA finite-volume method. We consider a domain $\Omega = [0, 2] \times [0, 2]$ discretized as a grid $\Omega_h$ containing four elements:

$$\Omega_h = E_1 \cup E_2 \cup E_3 \cup E_4.$$  

We assume that physical quantities are assigned appropriately, resulting in the following parameters in the respective elements: $\lambda_1 = 1$, $\lambda_2 = 10$, $\lambda_3 = 10$, $\lambda_4 = 100$, and $q_1 = 1$, $q_2 = 0$, $q_3 = 0$, $q_4 = -1$. We have a source in $E_1$ and a sink in $E_4$. We expect the solution to be symmetric along the diagonal through the domain from (0,0) to (2,2). Further, we assume no flow on the boundary. We have the interfaces $\gamma_{12}$, $\gamma_{13}$, $\gamma_{24}$, $\gamma_{34}$. 
7.2 Two-point Flux-Approximation Scheme

As $\lambda$ is defined inside elements only, we need to approximate $\lambda$ at the interfaces:

\begin{align*}
\lambda_{1,12} &= (1,0) \cdot 1 \cdot (1,0)^T = 1, \\
\lambda_{2,12} &= (1,0) \cdot 10 \cdot (1,0)^T = 10, \\
\lambda_{1,13} &= (0,1) \cdot 1 \cdot (0,1)^T = 1, \\
\lambda_{3,13} &= (0,1) \cdot 1 \cdot (0,1)^T = 10, \\
\lambda_{2,24} &= (0,1) \cdot 10 \cdot (0,1)^T = 10, \\
\lambda_{4,24} &= (0,1) \cdot 100 \cdot (0,1)^T = 100, \\
\lambda_{3,34} &= (1,0) \cdot 10 \cdot (1,0)^T = 10, \\
\lambda_{4,34} &= (1,0) \cdot 100 \cdot (1,0)^T = 100.
\end{align*}
Calculating the transmissibilities given by (5.9) gives

\[ T_{12} = 2 \cdot 0.5 \cdot \left( \frac{0.25}{1} + \frac{0.25}{10} \right)^{-1} = \frac{40}{11}, \]
\[ T_{13} = 2 \cdot 0.5 \cdot \left( \frac{0.25}{1} + \frac{0.25}{10} \right)^{-1} = \frac{40}{11}, \]
\[ T_{24} = 2 \cdot 0.5 \cdot \left( \frac{0.25}{10} + \frac{0.25}{100} \right)^{-1} = \frac{400}{11}, \]
\[ T_{34} = 2 \cdot 0.5 \cdot \left( \frac{0.25}{10} + \frac{0.25}{100} \right)^{-1} = \frac{400}{11}. \]

The matrix \( A \) then becomes

\[
A = \begin{bmatrix}
\frac{80}{11} + 1 & -\frac{40}{11} & -\frac{40}{11} & 0 \\
-\frac{40}{11} & 40 & 0 & -\frac{400}{11} \\
-\frac{40}{11} & 0 & 40 & -\frac{400}{11} \\
0 & -\frac{400}{11} & -\frac{400}{11} & \frac{800}{11}
\end{bmatrix}.
\]

The source term \( q \) is given by

\[ q = [1 \ 0 \ 0 \ -1]^T. \]

Solving the system \( A p = q \) for \( p \) gives

\[ p = [0 \ -\frac{11}{80} \ -\frac{11}{80} \ -\frac{121}{800}]^T. \]

The velocity across the interfaces is calculated from (5.8):

\[ v_{12} = 2 \cdot 1 \cdot \left( 0 + \frac{11}{80} \right) \cdot \left( \frac{\frac{1}{2}}{1} + \frac{\frac{1}{2}}{10} \right)^{-1} = \frac{1}{2}, \]
\[ v_{13} = 2 \cdot 1 \cdot \left( 0 + \frac{11}{80} \right) \cdot \left( \frac{\frac{1}{2}}{1} + \frac{\frac{1}{2}}{10} \right)^{-1} = \frac{1}{2}, \]
\[ v_{24} = 2 \cdot 1 \cdot \left( -\frac{11}{80} + \frac{121}{800} \right) \cdot \left( \frac{\frac{1}{2}}{10} + \frac{\frac{1}{2}}{100} \right)^{-1} = \frac{1}{2}, \]
\[ v_{34} = 2 \cdot 1 \cdot \left( -\frac{11}{80} + \frac{121}{800} \right) \cdot \left( \frac{\frac{1}{2}}{10} + \frac{\frac{1}{2}}{100} \right)^{-1} = \frac{1}{2}. \]

Inside each cell we get the velocities using

\[ \sum_{\gamma_{ij} \subseteq E_i} v_{ij} \cdot n_{\gamma_{ij}}, \]
which gives us

\[ v_1 = v_2 = v_3 = v_4 = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \]

### 7.3 Mimetic Finite-Differences Scheme

We need to construct the matrices \( W, C \) and \( \Pi \), and then perform the multiplications presented in Algorithm 6.4. We assume that the faces are indexed in a clockwise order, beginning at the left face in each element. In this way, the left face in \( E_1 \) has the id 1, while the left face in \( E_2 \) has the id 5.

To construct \( W \), we first need to calculate \( Z^{(i)}, i = 1, 2, 3, 4 \), which form an orthonormal basis for the column space of \( D^{(i)}C^{(i)} \). Since the elements have identical geometry, \( C^{(i)} \) and \( D^{(i)} \) will be identical for all elements. Then \( Z^{(i)} \) must be identical for all elements. The same applies for \( N^{(i)} \). From (6.32) we have that

\[
C^{(i)} = \begin{bmatrix}
-\frac{1}{2} & 0 \\
0 & \frac{1}{2} \\
\frac{1}{2} & 0 \\
0 & -\frac{1}{2}
\end{bmatrix}, \quad i = 1, 2, 3, 4.
\]

\( D^{(i)} \) is a diagonal matrix with face areas as the diagonal elements given by (6.33):

\[
D^{(i)} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}, \quad i = 1, 2, 3, 4.
\]

It follows that

\[
D^{(i)}C^{(i)} = \begin{bmatrix}
-\frac{1}{2} & 0 \\
0 & \frac{1}{2} \\
\frac{1}{2} & 0 \\
0 & -\frac{1}{2}
\end{bmatrix}, \quad i = 1, 2, 3, 4.
\]

The orthonormal basis for the column space of \( D^{(i)}C^{(i)} \) can be found using the Gram-Schmidt process. For a set of vectors \( S = \{v_1, \ldots, v_k, \ldots, v_n\} \), the
k'th column of the orthogonal basis is given by
\[ u_k = v_k - \sum_{j=1}^{k-1} \langle v_k, u_j \rangle u_j, \]
where \( \langle \cdot, \cdot \rangle \) denotes the inner-product of the two vectors. The columns are normalized to form an orthonormal basis:
\[ e_k = \frac{u_k}{\|u_k\|}. \]

By this process, we get that
\[ Z^{(i)} = \begin{bmatrix} -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 0 \\ 0 & -\frac{\sqrt{2}}{2} \end{bmatrix}, \quad i = 1, 2, 3, 4. \]

We have from (6.31) that
\[ N^{(i)} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad i = 1, 2, 3, 4. \]

We now construct \( W \) cell-by-cell using (6.30):
\[ W^{(i)} = \frac{1}{1} \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \lambda_i \cdot \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} + \frac{2 \text{trace}(\lambda_i)}{1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 0 \\ 0 & -\frac{\sqrt{2}}{2} \end{bmatrix}. \]
We can now easily calculate that

\[
W^{(1)} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix},
\]

\[
W^{(2)} = \begin{pmatrix}
10 & 0 & 0 & 0 \\
0 & 10 & 0 & 0 \\
0 & 0 & 10 & 0 \\
0 & 0 & 0 & 10
\end{pmatrix},
\]

\[
W^{(3)} = \begin{pmatrix}
10 & 0 & 0 & 0 \\
0 & 10 & 0 & 0 \\
0 & 0 & 10 & 0 \\
0 & 0 & 0 & 10
\end{pmatrix},
\]

\[
W^{(4)} = \begin{pmatrix}
100 & 0 & 0 & 0 \\
0 & 100 & 0 & 0 \\
0 & 0 & 100 & 0 \\
0 & 0 & 0 & 100
\end{pmatrix},
\]

forming the block-diagonal matrix

\[
W = \begin{bmatrix}
W^{(1)} & 0 & 0 & 0 \\
0 & W^{(2)} & 0 & 0 \\
0 & 0 & W^{(3)} & 0 \\
0 & 0 & 0 & W^{(4)}
\end{bmatrix}.
\]

To construct \( C \), we place the area of each face at the id of the face:

\[
C = \begin{bmatrix}
11111000000000000000 \\
00001111000000000000 \\
00000000111100000000 \\
00000000001111111111
\end{bmatrix}.
\]

Finally, we assign an id to each interface and to external faces (the faces at the boundary). We let the interfaces be ordered as following:

\[ \{ \gamma_{12}, \gamma_{13}, \gamma_{24}, \gamma_{34} \} ; \]
and the external faces in the order

\[ \{F_1, F_4, F_7, F_8, F_9, F_{10}, F_{14}, F_{15}\}. \]

While we have a no-flow boundary condition, we still need to include them in \( \Pi \). One row in \( \Pi \) represents an interface or external face, and we have that

\[
\Pi = \begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}.
\]

We now follow Algorithm 6.4 to obtain \( p \) and \( v \). As this is straightforward matrix algebra, we will only give the solutions. If we force \( p_1 = 0 \) as we did in Section (?), we get the following solution for pressure:

\[
p = \begin{bmatrix} 0 & -\frac{11}{20} & -\frac{11}{20} & -\frac{131}{20} \end{bmatrix}^T.
\]

We get the velocity along the normal component of each face:

\[
v = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & -\frac{1}{2} \end{bmatrix}^T.
\]

We get the velocities inside each cell by summing the velocities through each face belonging to the element:

\[
\sum_{F_k \subseteq E_i} v_k \cdot n_{k}^{(i)}.
\]
This gives us the cell-centered velocities

\[ v_1 = v_2 = v_3 = v_4 = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \]

As we see, the solution of \( p \) is different for the TPFA discretization and
the mimetic FDM discretization. The velocity field is on the other hand identical.
The numerical experiments performed in this chapter aim towards measuring effects from non-orthogonality. We will compare the quality of the pressure and velocity solutions from the TPFA and mimetic FDM discretizations. The reference solutions are obtained using a TPFA scheme on five times (horizontally) refined grids.

We use the following norms to measure errors. The relative error in the pressure solution is measured with the $L^2$-norm:

$$e_r(p) = \frac{\| p_{\text{ref}} - p \|_{L^2(E)}}{\| p_{\text{ref}} \|_{L^2(E)}}.$$

For analysis of the velocities, we use cell-centered velocities:

$$v_i = \sum_{\gamma_{ij} \subset E_i} v_{ij}.$$

Since the velocity solutions contain few jumps, we assume that the $L^2$-norm is a representative norm for the quality of the solution:

$$e(v) = \| v_{\text{ref}} - v \|_{L^2(E)}.$$

The relative error in the velocity solution is measured as

$$e_r(v) = \frac{\| v_{\text{ref}} - v \|_{L^2(E)}}{\| v_{\text{ref}} \|_{L^2(E)}}.$$
The average error in the velocity field is measured as
\[ \bar{e} = \int_{\Omega} e(v) \, dx. \]

## 8.1 Non-orthogonal Layer

We consider a domain \( \Omega = [-1, 1] \times [-1, 1] \times [-1, 1] \) with a source located inside \([-0.9, -0.8] \times [-0.9, -0.8] \times [-1, 1] \) and a sink inside \([0.8, 0.9] \times [0.8, 0.9] \times [-1, 1] \). We assume a heterogeneous permeability inside the domain, and that no flow occurs through the boundaries.

This is essentially a 2D problem, but to measure the effects from a non-orthogonal layer, we cut the domain in two with a layer function \( Z = \alpha x \) as illustrated in Figure 8.1. We solve the problem for pressure and velocity, and let \( \alpha \) vary in order to increase the non-orthogonality between the layers.

We perform 10 simulations for \( \alpha = 0, \frac{1}{10}, \frac{2}{10}, \frac{3}{10}, \frac{4}{10}, \frac{5}{10}, \frac{6}{10}, \frac{7}{10}, \frac{8}{10}, \frac{9}{10} \). For each value for \( \alpha \) we perform four simulations for different constant directional cell-mobilities. We start by performing the simulation for an isotropic tensor \( \lambda = 1 \). For \( \alpha = 0 \), the grid is K-orthogonal. We then perform three simulations with an anisotropic tensor on the form
\[
\lambda = \begin{pmatrix}
10 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix},
\]
where \( \lambda \) is rotated around the z-axis by \( \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3} \) radians. In other words, \( \lambda \) is a full tensor.

A contour plot of the pressure solution for isotropic tensor and \( \alpha = \frac{9}{10} \) is shown in Figure (8.2). We can see the effect from the non-orthogonality in the TPFA solution, while the mimetic FDM solution is less distorted by the effect.

Table (8.1) shows the relative error in the pressure solution for the reservoir with isotropic tensor, sampled in the top layer in the grid-cell with the source. We see a significant error arising in the TPFA solution as \( \alpha \) increases. As shown in Figure (8.2), the pressure distribution loses symmetry.
Fig. 8.1: Varying layer function $Z = \alpha x$ where a) $\alpha = 0$, b) $\alpha = 0.9$.

Fig. 8.2: Contour plot of five values from the pressure solution with $\alpha = \frac{9}{10}$ for a) TPFA, b) mimetic FDM.
<table>
<thead>
<tr>
<th>α</th>
<th>TPFA</th>
<th>MIMETIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6.61 · 10^{-2}</td>
<td>2.60 · 10^{-2}</td>
</tr>
<tr>
<td>0.1</td>
<td>1.05 · 10^{-1}</td>
<td>1.19 · 10^{-2}</td>
</tr>
<tr>
<td>0.2</td>
<td>1.48 · 10^{-1}</td>
<td>1.40 · 10^{-3}</td>
</tr>
<tr>
<td>0.3</td>
<td>1.94 · 10^{-1}</td>
<td>1.36 · 10^{-2}</td>
</tr>
<tr>
<td>0.4</td>
<td>2.42 · 10^{-1}</td>
<td>2.46 · 10^{-2}</td>
</tr>
<tr>
<td>0.5</td>
<td>2.90 · 10^{-1}</td>
<td>3.41 · 10^{-2}</td>
</tr>
<tr>
<td>0.6</td>
<td>3.39 · 10^{-1}</td>
<td>4.22 · 10^{-2}</td>
</tr>
<tr>
<td>0.7</td>
<td>3.88 · 10^{-1}</td>
<td>4.89 · 10^{-2}</td>
</tr>
<tr>
<td>0.8</td>
<td>4.37 · 10^{-1}</td>
<td>5.44 · 10^{-2}</td>
</tr>
<tr>
<td>0.9</td>
<td>4.86 · 10^{-1}</td>
<td>5.89 · 10^{-2}</td>
</tr>
</tbody>
</table>

Tab. 8.1: Relative error in pressure sampled in the top layer at the source.

The impact of this effect on the velocity is however not critical. The mimetic FDM solution for pressure corresponds good with the reference solution. As the mimetic FDM obeys the Gauss-Green formula, this is expected.

We see in Figure (8.3) plots of the average error of the velocity solution for the four different tensors $\lambda$. The average error is representable for the trend shown in the solution in single grid-cells.

For an isotropic $\lambda$ (see Figure 8.3 a)), we have a K-orthogonal grid for $\alpha = 0$, and we see small differences in the error arising from the TPFA and mimetic FDM discretizations. As the tensor rotates, we see an increased gap between the error curves. In fact, mimetic FDM performs better for full tensors, while the error in the TPFA solution increase with a higher rate.

8.2 $2\frac{1}{2}$D PEBI-grid

We now consider a $2\frac{1}{2}$D PEBI-grid, representing a domain $\Omega = [-1, 1] \times [-1, 1] \times [-1, 1]$. A source is located at $(-0.5, 0.5)$ and a sink at $(0.5, -0.5)$ through all the layers. The source and sink terms are defined inside the cells depicted in Figure , where the grid-cells have a diameter of 0.1.
Fig. 8.3: The average error of the velocity solution for a) isotropic \( \lambda \), b) anisotropic \( \lambda \) rotated by \( \pi \), c) anisotropic \( \lambda \) rotated by \( \frac{4}{3} \pi \), d) anisotropic \( \lambda \) rotated by \( \frac{3}{2} \pi \).
We let $\lambda = 1$, and and measure the errors in the pressure and velocity solution. We sample the errors in the grid-cell with the source in the top layer.

Relative error for pressure is $e_r \approx 2.00 \cdot 10^{-2}$ for mimetic FDM solution, while the error is $e_r \approx 3.43 \cdot 10^{-2}$ for the TPFA solution. At the same location, the relative error in the velocity is $e_r \approx 3.67 \cdot 10^{-1}$ for the mimetic FDM solution, while the error is $e_r \approx 9.15 \cdot 10^{-2}$ for the TPFA solution.

The error in the velocity solution averaged over the domain, is $\overline{e} \approx 9.18 \cdot 10^{-5}$ for the mimetic FDM solution, and $\overline{e} \approx 9.48 \cdot 10^{-5}$ for the TPFA solution.

The errors in the velocities are small for both discretizations, but the mimetic FDM results in a more accurate solution averaged over the domain. This is expected as layers are somewhat curved, and the underlying Voronoi diagram have cells that are not locally orthogonal.
The principal objectives of this research have been to study the theoretical foundations, properties and construction of PEBI-grids, and compare the two-point flux-approximation and mimetic finite-differences discretization on PEBI-grids. We have seen how Delaunay triangulations and Voronoi diagrams relate to the generation of orthogonal PEBI-grids, and how a $2\frac{1}{2}$D PEBI-grid can be used to model a full reservoir. We have further focused on implementational aspects of PEBI-grid construction, and suggested a way to calculate volumes and face areas of curved PEBI-grid cells.

For discretization of the flow equation, the TPFA scheme is frequently favored by the industry due to its simple implementation. We have given a compact and general algorithm for the TPFA scheme. The amount of arithmetic operations performed in a simulator, based on a TPFA scheme, is minimal. We have also given a very general implementation of the mimetic FDM scheme. Compared with the TPFA scheme, mimetic FDM is slightly more complicated to implement. If one compares both schemes with other traditional schemes like the multi-point flux-approximation finite-volume scheme, or the mixed finite elements method, both TPFA and mimetic FDM is very easy to implement. The mimetic FDM requires a somewhat larger amount of arithmetic operations than the TPFA, but converges also for non-orthogonal grids.

We have obtained the following numerical results. We have considered a symmetric 2D problem, but tested the effects from non-orthogonality using layers. We have seen that the mimetic FDM outperforms TPFA as the problem becomes more irregular with respect to the directional cell-
mobilities and non-orthogonality. We have further seen how the velocity solutions obtained from the mimetic FDM are more accurate as directional cell-mobilities rotates. Mimetic FDM does not seem to gain accuracy by K-orthogonality, but rather favor irregular grids with full tensors. The small differences in the error between the TPFA and mimetic FDM solutions is however lower than expected. We have also considered a complex PEBI-grid with non-orthogonality in the plane and curved layers. Also here the mimetic FDM gives better results, but the gained accuracy is again not as high as expected.

If one assume that these results are representative in general, one can question the use of the mimetic FDM scheme on PEBI-grids. It can however be the case that the numerical results presented in this thesis is not generally representative. The non-orthogonality of the grids might not be enough to challenge the TPFA scheme, and increased non-orthogonality could have been introduced. As we saw and would expect, the TPFA scheme was sufficiently accurate for isotropic tensors and K-orthogonality.

As an indication for future research, the mimetic FDM scheme should be employed for large-scale, realistic reservoir scenarios. Further, as the conditions for convergence of the mimetic FDM does not represent a unique inner-product, but a family of inner-products, an optimal inner-product for PEBI-grids can be derived.


