NTNU
Norwegian University of
Science and Technology

Correlating Recovery Factors with Measures of Reservoir Heterogeneity

Asgeir Nyvoll

Petroleum Geoscience and Engineering
Submission date: June 2018
Supervisor: Carl Fredrik Berg, IGP
Co-supervisor: Stein Krogstad, SINTEF Digital

Norwegian University of Science and Technology
Department of Geoscience and Petroleum

Summary

Field scale reservoir simulations are computationally intensive, and therefore also time consuming. This thesis focuses on the development and evaluation of computationally easy measures of reservoir heterogeneity, by comparing their correlation with recovery factors after waterflooding and tertiary polymer injection. Expected recovery is related to heterogeneity, as heterogeneities increase the chance of bypassed oil during a waterflood. While field scale simulations include multiple fluid phases and many time steps, heterogeneity measures are often based on steady-state solutions of single-phase flow, which require much less computations. Heterogeneity measures can therefore be used for rapid screening of reservoir models, including positioning of wells and other potential recovery strategies. This allows for the evaluation of more options than what is feasible with only field scale simulations. After the initial screening, the best options can then be used for more thorough considerations.

The work has been focused on new heterogeneity measures based on a model of permeability description by the use of streamlines, and heterogeneity measures from a parameter called the diffusive time of flight. The permeability description divides the averaged permeability over a reservoir model into hydraulic conductance, tortuosity, constriction and effective bulk volume. New measures have further been compared with existing dynamic measures of reservoir heterogeneity. In addition, the required code to calculate both new and existing heterogeneity measures has been implemented as an extension to the Matlab Reservoir Simulation Toolbox (MRST), which is an open-source tool for reservoir simulation and evaluation developed by SINTEF. The code will be shared with the MRST team at SINTEF, and can be included in coming releases of the software.

The results show that several heterogeneity measures are closely correlated to expected recovery. Measures from diffusive time of flight were more promising than measures based on the permeability model, though especially tortuosity also shows correlation with recovery. However, the best correlations were found among some of the existing heterogeneity measures, especially sweep efficiency at 1 and 2 pore volumes injected and the dynamic

Lorenz coefficient.

The best existing measures gave a Pearson correlation coefficient above 0.9, which is very significant. The close correlations are interesting, as they indicate that the flow fields from single-phase steady-state solutions, which the heterogeneity measures are based on, has resemblance to the average flow fields during multiphase simulations.

Sammendrag

Feltskala reservoarsimuleringer er regnetunge, og derfor også tidkrevende. Denne oppgaven fokuserer på utvikling og evaluering av mål på reservoarheterogenitet som ikke er regnetunge, ved å sammenligne deres korrelasjon med utvinningsgrad etter vannflømming og tertiær polymerinjeksjon. Forventet utvinningsgrad er relatert til heterogenitet, ettersom heterogeniteter øker sannsynligheten for forbigått olje under en vannflømming. Mens simuleringer på feltskala inkluderer flere fluidfaser og mange tidssteg, så er heterogenitetsmål ofte basert på løsninger av enfasestrømning ved stabile forhold, som behøver mye færre utregninger. Heterogenitetsmål kan derfor brukes for rask utsortering av reservoarmodeller, inkludert plassering av brønner og andre potensielle utvinningsstrategier. Dette muliggjør evaluering av flere alternativ enn hva som er gjennomførbart med bare fulle simuleringer på feltskala. Etter utsorteringen kan de beste alternativene videre bli brukt for grundigere vurderinger.

Arbeidet har vært fokusert på heterogenitetsmål basert på en modell for permeabilitetsbeskrivelse ved bruk av strømlinjer, og heterogenitetsmål fra en parameter kalt diffusiv flukttid (”diffusive time of flight”). Permeabilitetsbeskrivelsen deler gjennomsnittspermeabiliteten over en reservoarmodell opp i hydraulisk konduktans, tortuositet, ”innsnevring” og effektivt bulkvolum. Nye mål har videre blitt sammenlignet med eksisterende dynamiske heterogenitetsmål. I tillegg har den nødvendige koden for å beregne nye og eksisterende heterogenitetsmål blitt implementert som en utvidelse til Matlab Reservoir Simulation Toolbox (MRST), som er et verktøy for reservoarsimulering og evaluering med åpen kildekode, som er utviklet av SINTEF. Koden vil bli delt med MRST-gruppen hos SINTEF, og kan bli inkludert i kommende utgivelser av programvaren.

Resultatene viser at flere heterogenitetsmål er nært korrelert med forventet utvinning. Mål fra diffusiv flukttid var mer lovende enn mål basert på permeabilitetsmodellen, selv om spesielt tortuositet også viser korrelasjon med utvinning. De beste korrelasjonen ble derimot funnet blant noen av de eksiterende heterogenitetsmålene, spesielt sveipe-effektivitet ved ett og to porevolum injisert og den dynamiske Lorenz-koeffisienten.

De beste eksisterende målene ga en Pearson-korrelasjonskoeffisient på over 0.9, noe som er meget signifikant. De nære korrelasjonene er interessante, ettersom de indikerer at strømningsfeltene fra løsninger av enfasestrømning under stabile forhold, som heterogenitetsmålene er basert på, har likheter med de gjennomsnittlige strømningsfeltene under flerfasesimuleringer.

Preface

This thesis has been written during the spring of 2018, as the final work of a Master of Science in Petroleum Engineering with specialization in Reservoir Engineering and Petrophysics at the Norwegian University of Science and Technology (NTNU).

I would like to express my gratitude to my supervisor Associate Professor Carl Fredrik Berg (NTNU). He has introduced me to an interesting topic, and followed up with guidance and comments during the entire project period. Hopefully he has enjoyed our many discussions as much as I have. I would also like to thank the team at SINTEF that develops the open-source software MRST, and especially my co-supervisor Stein Krogstad (SINTEF Digital), who has given feedback and help whenever I have asked for it.

Finally, I would like to thank friends, fellow students and family that have motivated me, challenged me and cared for me, over the last 24 years.

Table of Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

DTOF	Diffusive time of flight
FHI	Flow heterogeneity index
LHS	Left-hand side
MRST	Matlab Reservoir Simulation Toolbox
PVI	Pore volumes injected
RF	Recovery factor
SPE	Society of Petroleum Engineers

Subscripts

α	Fluid phase
e	Effective
i, j, k	Counters, usually in the x, y and z directions
src	Source or sink term
x, y, z	Directions in a Cartesian coordinate system.
b	Breakthrough
e	Effective
inj	Injected

inv	Investigation
p	Pore

Symbols

β	Magnitude function for streamline Darcy velocity
ϵ	Relative error
$\hat{P}$	Fourier transformed pressure function
$\hat{Q}$	Fluid discharge through a streamtube cross-section
κ	Streamline permeability factor
λ, χ	Stream surfaces (also known as stream functions)
$\mathbb{K}$	Permeability tensor
$\mathbb{S}$	Set of streamlines
$\mathcal{S}$	Streamline
μ	Viscosity
ω	Various uses: 1) Frequency 2) Vorticity
Φ	Storage capacity
ϕ	Porosity
σ	Standard deviation
τ	Tortuosity
τ_d	Diffusive time of flight
τ_t	Time of flight of a neutral particle
$\vec{l}$	Unit tangent vector
$\vec{n}$	Normal vector
$\vec{Q}$	Volumetric rate vector
$\vec{q}$	Darcy velocity vector
$\vec{v}$	Effective velocity of a neutral particle
$\vec{x}$	Position vector $[x, y, z]$
A	Cross-sectional area

A_k, A_0	Amplitude-related terms
B	Constriction factor
B	Hydraulic conductance
c_t	Total compressibility
C_V	Coefficient of variation
$dQ_{\mathcal{S}}$	Rate of infinitesimal streamtube
E_V	Displacement efficiency
E_V	Volumetric sweep efficiency
F	Flow capacity
h	Various uses: 1) Thickness 2) Horizontal
H_K	Koval factor
H_s	Shear heterogeneity index
H_V	Vorticity index
J	Determinant of Jacobian
k	Permeability
k_D	Layer permeability from Darcy's law over full model
L_c	Lorenz coefficient
L_n	Estimated length of streamline with using n substeps
p	Pressure
Q	Volumetric flow rate
q	Darcy velocity
r	Various uses: 1) Radius 2) Ratio
S	Streamtube cross-section with boundary δS
s	Distance
t	Time
t_D	Dimensionless time
V	Volume

v_i	Velocity in direction i
V_{DP}	Dykstra-Parsons coefficient

Units

cP	Centipoise
m	Meter
mD	Millidarcy
Pa	Pascal
s	Second
kg	Kilogram

Chapter 1

Introduction

The ultimate goal of reservoir engineering is the optimal exploitation of hydrocarbon resources. However, due to the limited data and tremendous complexity of hydrocarbon reservoirs, the optimal solution is not realistically achievable. Ever since the famous experiments by Darcy (1856), 3 years before what has become known as the beginning of the modern petroleum industry with the discovery of oil in 1859 in Titusville (Pennsylvania), engineers have known that flow in porous media is governed by Darcy's law. Darcy's law can be expressed as

$$\vec{q} = \frac{\vec{Q}}{A} = -\frac{\mathbb{K}}{\mu}\nabla p \quad , \tag{1.1}$$

where the *Darcy velocity* $\vec{q}$ is the volumetric flow rate $\vec{Q}$ per cross-sectional area A of the porous medium; $\mathbb{K}$ is the permeability tensor; μ is the fluid viscosity, and ∇p is the pressure gradient. The permeability tensor $\mathbb{K}$ is defined as

$$\mathbb{K} = \begin{bmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{bmatrix} \quad , \tag{1.2}$$

which can be reduced to a scalar k under the assumption of an isotropic permeability field. While the permeability is a rock parameter, and Darcy's law only holds for single-phase, the concept of relative permeability has later been introduced to extend the use of Equation (1.1) to multiphase systems (Wyckoff and Botset, 1936). The relative permeability of a phase α is defined as

$$k_{r_\alpha} = \frac{k_\alpha}{k} \quad . \tag{1.3}$$

Lack of data and computational power limited the use of Darcy's law to mostly analytic solutions for a long time. The introduction of computers then revolutionized the field of reservoir engineering, where reservoir simulations based on numerical models now play a major role. Reservoir models are created by incorporating knowledge of e.g. fluid models, reservoir geology, fluid dynamics, petrophysical data and PVT properties, and then history matched against available production data. The simulation models are applied to forecast future production, and responses of various injection and production strategies. Depending on the size, complexity and timespan of the model, each simulation can take anything from seconds, to days or weeks. One may also create an ensemble of equiprobable geological realizations to include geological uncertainty, which will further increase the computational expense. It is therefore not feasible to test every possible well location, injection strategy or set of well controls with full field simulations, leaving the engineer to choose a smaller set of cases for further testing.

The recovery factor is related to the goal of optimal exploitation of hydrocarbon resources, as a higher recovery factor usually means more income. The exception is if the increased recovery factor comes with a much longer production period, for which the discounted income in a net present value calculation could decrease. The recovery factor is defined as

$$RF = \frac{\text{hydrocarbon produced}}{\text{original hydrocarbon in place}} = E_D E_V \quad , \tag{1.4}$$

where E_D and E_V are the displacement and volumetric sweep efficiencies, respectively. The displacement efficiency is a factor that describes the fraction of the original hydrocarbon in place that is displaced in the swept zone. The volumetric sweep efficiency is the fraction of the volume that is swept at any given time (e.g. during a waterflood), and it is sometimes divided into a product of areal sweep efficiency and vertical sweep efficiency (Lake, 1989). The examples used in this thesis are for two-dimensional horizontal flow, so the volumetric sweep efficiency will equal the areal sweep efficiency (the fraction of the area that is swept).

The sweep will be uniform in a homogeneous reservoir, and hence you would expect that the volumetric sweep efficiency is large. On the other hand, in a heterogeneous reservoir the fluids will follow the paths of least resistance. The volumetric sweep efficiency can therefore be drastically decreased, which will reduce the expected recovery factor.

Hence, it is clear that there is a connection between heterogeneity and expected recovery. However, a major challenge lies in the fact that there is no universally accepted quantification of reservoir heterogeneity. Several potential measures of heterogeneity have already been proposed (see e.g. Koval, 1963; Lake, 1989; Jensen et al., 2000; Shook and Mitchell, 2009; Rashid et al., 2012), many of which are based on streamline calculations from simple single-phase and incompressible flow experiments. Additionally, as part of the author's *Specialization Project* at NTNU during the fall of 2017, a streamline based permeability description at the pore scale (Berg, 2014) was extended to the Darcy scale (Nyvoll, 2017). The goal of this thesis has been to improve the implementation of this model, in addition to developing heterogeneity measures based on the model and other methodologies. The measures have further been compared to existing heterogeneity measures, by correlating the measures with oil recovery after waterflooding and recovery increase due to polymer injection.

1.1 Structure of the Thesis

Chapter 2 is written to give the reader the necessary background for the later chapters. The chapter covers theory on topics such as streamline and streamtube simulation, time of flight, diffusive time of flight, existing heterogeneity measures, and it also presents the reservoir model used for comparison of heterogeneity measures. Chapter 3 covers algorithms for streamline simulation and diffusive time of flight, the model of permeability description from Nyvoll (2017), new potential heterogeneity measures that have been tested, and the oil recovery simulations that have been used to benchmark the heterogeneity measures. Chapter 4 covers the main results of the work, including results from an uncertainty analysis of the implementations. Discussions of the implementations and results then follows in Chapter 5. Chapter 6 covers the main conclusions, while recommendations for further work are included in Chapter 7.

Chapter 2

Background

The following chapter will cover the essentials of the field of streamline simulation, and the closely related time-of-flight. In addition, the chapter covers the theory of diffusive time of flight, and the SPE10 reservoir model that is used for the examples in this work. The methodologies applied later in this thesis are based mainly on the theory presented in this chapter, and the understanding of this theory is therefore essential to the understanding of the methodologies, results and discussions that follows in subsequent chapters. The descriptions and derivations should be sufficiently comprehensive for the reader to follow this thesis without specific knowledge of streamline simulation.

2.1 Streamlines and Streamtubes in Porous Media

2.1.1 Streamlines

A *streamline* is a line which is continuously tangential to the instantaneous velocity field of fluid flow. It is related, but not necessarily equal, to pathlines and streaklines. Streamlines are lines tangential to the current velocity field, and therefore not dependent of the velocity field at previous times, while a pathline is a line that follows the trajectory of a neutral particle in space over time. Lastly, a streakline is a line that connects all neutral particles that have passed through a specific point in space (see e.g. Bear, 1972; Datta-Gupta and King, 2007; Kreyszig, 2011). It follows that streamlines, pathlines and streaklines all will coincide in steady-state flow.

Following the derivations in Bear (1972) and Nelson (1963) we have that a pathline is described by the solution of

$$\frac{dx}{v_x(\vec{x},t)} = \frac{dy}{v_y(\vec{x},t)}) = \frac{dz}{v_z(\vec{x},t)} = dt \quad , \tag{2.1}$$

where $\vec{x} = [x, y, z]$ is a position in space. Equation (2.1) can be parameterized to

$$\lambda = \lambda(\vec{x},t) \quad , \quad \chi = \chi(\vec{x},t) \quad , \quad s = s(\vec{x},t) \quad , \tag{2.2}$$

where λ=constant , χ=constant and s=constant represent surfaces which at their intersection describe the position of a neutral particle in space at time t. As streamlines coincide with pathlines in steady-state flow, we can parameterize streamlines similarly as in Equation (2.2) by removing the time term t. Streamlines are usually described as the intersection between two such surfaces, λ=constant and χ=constant, while the last coordinate, s, is the distance along their intersection (which is the streamline). It is required that λ, χ and s are independent, which means that the Jacobian of the coordinate transformation must be non-zero (Bear, 1972):

$$J = \left\| \frac{\partial(\lambda, \chi, s)}{\partial\,(x, y, z)} \right\| = \nabla s \cdot \nabla\lambda \times \nabla\chi \neq 0 \quad . \tag{2.3}$$

Since the intersections of the surfaces described by λ and χ for various constants define streamlines, they are necessarily continuously tangential to the velocity field. Hence, there is no flow passing through them. Nelson (1963) therefore calls the functions λ and χ the *stream functions of three-dimensional flow*, but they are also commonly known as *stream surfaces* and *bistreamfunctions* (see e.g. Bear, 1972; Datta-Gupta and King, 2007). Due to the fact that λ and χ are functions describing no-flow surfaces, we have that (Nelson, 1963; Bear, 1972)

$$\nabla\lambda \cdot \vec{q} = \nabla\chi \cdot \vec{q} = 0 \quad , \tag{2.4}$$

where the Darcy velocity $\vec{q} = [q_x, q_y, q_z]$ is related to the effective particle velocity by a scaling with the local porosity ϕ, as the Darcy velocity is related to the cross-sectional area of the porous medium, while the actual cross-sectional area for fluid flow is the cross-sectional area of the pore space. Hence,

$$\vec{q} = \phi\vec{v} \quad . \tag{2.5}$$

The no-flow conditions in Equation (2.4) can be written out to a set of two independent equations:

$$q_x \frac{\partial \lambda}{\partial x} + q_y \frac{\partial \lambda}{\partial y} + q_z \frac{\partial \lambda}{\partial z} = 0 \quad , \tag{2.6}$$

and

$$q_x \frac{\partial \chi}{\partial x} + q_y \frac{\partial \chi}{\partial y} + q_z \frac{\partial \chi}{\partial z} = 0 \quad . \tag{2.7}$$

Nelson (1963) and Bear (1972) then create a new set of equations by first eliminating q_x, q_y and q_z, one at a time. From Equation (2.7) we can find

$$q_z = -q_x \frac{\left(\frac{\partial \chi}{\partial x}\right)}{\left(\frac{\partial \chi}{\partial z}\right)} - q_y \frac{\left(\frac{\partial \chi}{\partial y}\right)}{\left(\frac{\partial \chi}{\partial z}\right)} \quad , \tag{2.8}$$

and then insert Equation (2.8) into Equation (2.6) to find

$$q_x \left(\frac{\partial \lambda}{\partial x} - \frac{\left(\frac{\partial \chi}{\partial x}\right)}{\left(\frac{\partial \chi}{\partial z}\right)} \frac{\partial \lambda}{\partial z} \right) = q_y \left(\frac{\left(\frac{\partial \chi}{\partial y}\right)}{\left(\frac{\partial \chi}{\partial z}\right)} \frac{\partial \lambda}{\partial z} - \frac{\partial \lambda}{\partial y} \right) \quad , \tag{2.9}$$

which can be rewritten as

$$\frac{q_x}{\left(\frac{\partial \lambda}{\partial y}\frac{\partial \chi}{\partial z} - \frac{\partial \lambda}{\partial z}\frac{\partial \chi}{\partial y} \right)} = \frac{q_y}{\left(\frac{\partial \lambda}{\partial z}\frac{\partial \chi}{\partial x} - \frac{\partial \lambda}{\partial x}\frac{\partial \chi}{\partial z} \right)} \quad . \tag{2.10}$$

Solving for q_x and q_y in a similar fashion as in Equations (2.8)-(2.10), we finally get the relation

$$\frac{q_x}{\left(\frac{\partial \lambda}{\partial y}\frac{\partial \chi}{\partial z} - \frac{\partial \lambda}{\partial z}\frac{\partial \chi}{\partial y} \right)} = \frac{q_y}{\left(\frac{\partial \lambda}{\partial z}\frac{\partial \chi}{\partial x} - \frac{\partial \lambda}{\partial x}\frac{\partial \chi}{\partial z} \right)} = \frac{q_z}{\left(\frac{\partial \lambda}{\partial x}\frac{\partial \chi}{\partial y} - \frac{\partial \lambda}{\partial y}\frac{\partial \chi}{\partial x} \right)} = \beta \quad , \tag{2.11}$$

or simply

$$\vec{q} = \beta \nabla \lambda \times \nabla \chi \quad , \tag{2.12}$$

where $\beta = \beta(x, y, x)$ is a function that accounts for the magnitude (Nelson, 1963). The function β has to be a function of only λ and χ for incompressible fluids as $\nabla \cdot \vec{q} = 0$. Nelson (1963) justifies the choice of $\beta = 1$ for incompressible fluids, while Bear (1972) says that it can be shown that in general $\beta = {}^{1}/_{\rho_e}$, where ρ_e is the effective density which is equal to 1 for incompressible flow (Datta-Gupta and King, 2007). Thus, we now have an expression for the Darcy velocity in a streamline for incompressible flow, expressed by the stream functions λ and χ:

$$\vec{q} = \nabla\lambda \times \nabla\chi \quad . \tag{2.13}$$

The equations can also easily be reduced to a simpler expression for two-dimensional flow by using $z =$ constant as the stream surface λ, hence we get

$$\vec{q}_{\mathrm{2D}} = [q_x, q_y] = \nabla z \times \nabla\chi = [0, 0, 1] \times \left[\frac{\partial\chi}{\partial x}, \frac{\partial\chi}{\partial y}\frac{\partial\chi}{\partial z}\right] = \left[-\frac{\partial\chi}{\partial y}, \frac{\partial\chi}{\partial x}\right] \quad . \tag{2.14}$$

Streamlines are of great use in the evaluation of flow patterns, and can e.g. be used to visualize well pairs, drainage regions and stagnation points (Lie, 2016; Datta-Gupta and King, 2007). They are an important part of the group of techniques called *flow diagnostics*, which by (Lie, 2016, pg. 360) is defined as "simple and controlled numerical flow experiments that are run to probe a reservoir model, establish connections and basic volume estimates, and quickly provide a qualitative picture of the flow patterns in the reservoir". However, we will see through the next sections that the opportunities of streamline-based techniques reach far further than just visualization of flow patterns.

2.1.2 Streamtubes

Streamlines can be used to visualize flow paths in a reservoir, but they have infinitesimal thickness and are therefore not suitable for volumetric calculations in a discretized reservoir model. *Streamtubes* in three dimensions are constructed for this. Say that we use two pairs of stream surfaces: $\lambda_1 = \lambda$, $\lambda_2 = \lambda + \Delta\lambda$ and $\chi_1 = \chi$, $\chi_2 = \lambda + \Delta\chi$. Returning to the Jacobian in Equation (2.3), and inserting the new expression for incompressible Darcy velocity $\vec{q}$ from Equation (2.13), Berg (2014) gets

$$J = \left|\left|\frac{\partial(\lambda, \chi, s)}{\partial\,(x, y, z)}\right|\right| = \nabla s \cdot (\nabla\lambda \times \nabla\chi) = \frac{\vec{q}}{q} \cdot \vec{q} = q \quad , \tag{2.15}$$

where Berg (2014) uses that ∇s is the unit vector in the direction of s, which also the normalized Darcy velocity vector $\vec{q}/q$ is. Berg (2014) then uses Stokes' theorem to show that the fluid discharge $\hat{Q}$ through a streamtube cross-section S with boundary δS is

$$\hat{Q} = \int\int_S \vec{q} \cdot \vec{n} dS = \int_{\delta S} \lambda \nabla \chi \cdot \vec{l} ds = (\lambda_2 - \lambda_1)(\chi_2 - \chi_1) = \Delta\lambda\Delta\chi \quad , \tag{2.16}$$

where $\vec{l}$ is the unit tangent to δS.

As the stream surfaces that defines the streamtube are all constants, the flow rate through a streamtube also has to be constant. Hence, we can now do one-dimensional calculations in streamtubes to simplify multi-dimensional problems (Datta-Gupta and King, 2007).

2.2 Time of Flight

2.2.1 Particle Time of Flight

The *time of flight* is the time it takes a neutral particle to reach a point in the reservoir. In Sections 2.1.1 and 2.1.2 we have used a coordinate system for streamlines and streamtubes described by the intersection of two stream functions λ and χ, together with a distance s along the streamline. The distance s is though a function of velocity and time, and as the velocity field can be calculated with a regular finite-difference method, we can exchange the distance coordinate s with a time coordinate τ_t (Datta-Gupta and King, 2007). In general, the time it takes to travel the distance s through a velocity field in space is the integral of the reciprocal of the velocity over the path, which can be expressed as

$$\tau_t = \int_0^s \frac{1}{|\vec{v}|} ds = \int_0^s \frac{\phi}{|\vec{q}|} ds \quad , \tag{2.17}$$

in the second equality above the relation from Equation (2.5) is used, as the particle velocity is the effective velocity through the pore space, and not the Darcy velocity. Equation (2.17) can also be written on the form (Datta-Gupta and King, 2007)

$$\vec{q} \cdot \nabla \tau_t = \phi \quad . \tag{2.18}$$

If we now return to the Jacobian of the transformation from Cartesian coordinates, this time using τ_t as the last streamline coordinate, the same result as in Equation (2.18) can be found (Datta-Gupta and King, 2007)

$$J = \left|\left|\frac{\partial(\lambda,\chi,\tau_t)}{\partial\left(x,y,z\right)}\right|\right| = \nabla\tau_t\cdot\left(\nabla\lambda\times\nabla\chi\right) = \nabla\tau_t\cdot\vec{q} = \phi \quad . \tag{2.19}$$

We will see that the time of flight is a very useful measurement. It can for example be used to track the front of an injected fluid in time, which can be used in the evaluation of time of breakthrough. As Equation (2.19) shows that there is a relation between the coordinate transformation and the porosity ϕ, and hence also the pore volume, the time of flight can also be used to find the displaced volume as a function of time (Datta-Gupta and King, 2007).

2.2.2 Diffusive Time of Flight

The neutral particle time of flight introduced in Section 2.2.1 is useful for many purposes, but when matching it with field data, e.g. through a tracer test, it can take a long time for the tracer to reach a producer. In addition, at the time the tracer reaches the well we will necessarily have breakthrough of injection fluid, and it might already be too late to initiate countermeasures. A pressure front will though arrive at a much earlier time, which is a major reason for why transient well testing is used (Kamal, 2009; Datta-Gupta and King, 2007). The *diffusive time of flight*, which is the defined as the time it takes the peak of a pressure response from an impulse source/sink (Dirac-function) to reach a point in the reservoir, has therefore been introduced (Vasco et al., 2000). Similar approaches had already been used for diffusive electromagnetic imaging (Virieux et al., 1994), the scalar wave equation (Fatemi et al., 1995) and diffusive tracer transport (Vasco and Datta-Gupta, 1999).

The diffusivity equation is the general equation for the pressure propagation in a reservoir, and can on a general form be written as (Datta-Gupta and King, 2007)

$$\phi\left(\vec{x}\right)\mu c_t\frac{\partial p\left(\vec{x},t\right)}{\partial t} - \nabla\cdot\left(\mathbb{K}\nabla p\left(\vec{x},t\right)\right) = 0 \quad , \tag{2.20}$$

where c_t is the total compressibility, and the other parameters are defined as before. This equation is used as the basis for both the diffusive time of flight, and regular well testing. In well testing the solution is usually found for a line source/sink, which basically means a well with negligible wellbore radius relative to drainage radius. For a well in radial transient flow, an appropriate approximation of the solution of the diffusivity equation at a radius r and a time t after a change to a constant bottom hole rate Q is (Lee, 1982)

$$p(r,t) = p_i + \frac{Q\mu}{kh} Ei\left(\frac{-\phi\mu c_t r^2}{4kt}\right) \quad , \quad Ei(-x) = -\int_x^\infty \frac{e^-\xi}{\xi} d\xi \quad , \tag{2.21}$$

where p_i is the initial pressure, and h is the thickness of the formation. Boundaries and other heterogeneities will induce discrepancies in the wellbore flowing pressure, and can therefore be used for reservoir evaluation. It is though important to have a quantification of the depth/radius at which such heterogeneity lies. This is what is commonly termed as the *radius/depth of investigation*. As the pressure propagates as a wave, and not a sharp front, there are several definitions of the radius of investigation (See e.g. Kuchuk, 2009). The definition as used in Lee (1982) is though common, which is the radius of the maximum pressure change. This radius can be found be setting the second derivative of the pressure in Equation (2.21) equal to zero. By first finding the first derivative

$$\frac{dp(r,t)}{dt} = \frac{Q\mu}{kh}\left(-\frac{e^{\frac{-\phi\mu c_t r^2}{4kt}}}{t}\right) \quad , \tag{2.22}$$

and further the second derivative which is equal to 0

$$\frac{d^2p(r,t)}{dt^2} = \frac{Q\mu}{kh}\left(\frac{e^{-\frac{\phi\mu c_t r^2}{4kt}}\left(t - \frac{\phi\mu c_t r^2}{4k}\right)}{t^3}\right) = 0 \quad , \tag{2.23}$$

we get the radius of investigation

$$r_{inv} = \sqrt{\frac{4kt}{\phi\mu c_t}} \quad . \tag{2.24}$$

Equivalently the time of maximum pressure change at a radius r is given as

$$t_{max} = \frac{\phi\mu c_t r^2}{4k} \quad . \tag{2.25}$$

The definition of diffusive time of flight by Vasco et al. (2000) is in fact analogous to the definition of radius of investigation by Lee (1982), recognizing that a sudden rate change ΔQ in the definition of radius of investigation in Lee (1982) in fact can be regarded as a Heaviside step-function, for which the derivative is the Dirac-function. The derivation of the diffusive time of flight was introduced for isotropic media with an asymptotic approach in Vasco et al. (2000). Similar approaches had already been used for diffusive electromagnetic propagation (Virieux et al., 1994) and tracer transport (Vasco and Datta-Gupta, 1999). Vasco et al. (2000) define the location of the pressure front as the location of the peak pressure response from an impulse source or sink, commonly known as the peak of the Dirac-function. This is analogous to the definition of radius of investigation in Lee (1982), recognizing that a sudden rate change can be regarded as a Heaviside step-function, for which the derivative is the Dirac-function.

The derivation showed here will be a general form for a full tensor permeability $\mathbb{K}$, and therefore differs slightly from e.g. Vasco et al. (2000) and Kulkarni et al. (2001), though it follows the same principles. Vasco et al. (2000) begin the derivation of the diffusive time of flight by transforming equation (2.20) to the frequency domain by applying the Fourier transform

$$\hat{P}\left(\vec{x},\omega\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p\left(\vec{x},t\right) e^{-i\omega t} dt \quad , \tag{2.26}$$

where ω is the frequency of the pressure wave. We can then write (2.20) as

$$\nabla \cdot \left(\mathbb{K}\nabla\hat{P}\left(\vec{x},\omega\right)\right) = \phi\left(\vec{x}\right)\mu c_t\left(i\omega\right)\hat{P}\left(\vec{x},\omega\right) \tag{2.27}$$

As we consider the pressure front as a wave that propagates through the porous medium, an asymptotic solution of $\hat{P}\left(\vec{x},\omega\right)$ can be found on the form (Fatemi et al., 1995; Vasco et al., 2000)

$$\hat{P}\left(\vec{x},\omega\right) = e^{-\sqrt{-i\omega}\tau_d\left(\vec{x}\right)} \sum_{k=0}^{\infty} \frac{A_k\left(\vec{x}\right)}{\left(\sqrt{i\omega}\right)^k} \quad , \tag{2.28}$$

where τ_d is a pseudo phase function of the wave, which is called the *diffusive time of flight*, and A_k is a term related to the amplitude. A sharp front solution will be reduced to the terms governed by rapid variation, thus the terms with the highest order of the frequency ω. Reducing (2.28) to the single most frequency sensitive term leaves us with the *zeroth-order approximation* (Virieux et al., 1994), also called the *geometric ray approximation* (Kulkarni et al., 2001)

$$\hat{P}\left(\vec{x},\omega\right) = A_0\left(\vec{x}\right) e^{-\sqrt{-i\omega}\tau_d\left(\vec{x}\right)} \quad . \tag{2.29}$$

Inserting Equation (2.29) into Equation (2.27) we get

$$\nabla \cdot \left(\mathbb{K}\nabla\left(A_0(\vec{x})\, e^{-\sqrt{-i\omega}\tau_d(\vec{x})}\right)\right) = \phi(\vec{x})\,\mu c_t\,(-i\omega)\,A_0(\vec{x})\,e^{-\sqrt{-i\omega}\tau_d(\vec{x})} \quad . \tag{2.30}$$

We will leave the right hand side of equation (2.30) for a while, and focus on the left hand side of the equation. By writing out the inner gradient of the left hand side, we get

$$\begin{aligned} LHS = \nabla \cdot \Big(\mathbb{K}\Big(&\nabla A_0(\vec{x})\, e^{-\sqrt{-i\omega}\tau_d(\vec{x})} \\ &+ \nabla\tau_d(\vec{x})\,A_0(\vec{x})\left(-\sqrt{-i\omega}\right) e^{-\sqrt{-i\omega}\tau_d(\vec{x})}\Big)\Big) \quad . \end{aligned} \tag{2.31}$$

The ∇A_0-term of (2.31) is neglected as it will be of a lower order of $\sqrt{-i\omega}$, and hence the second term will dominate Equation (2.31) for a sharp front. By doing this we further get

$$\begin{aligned} LHS = A_0(\vec{x})&\left(-\sqrt{-i\omega}\right) e^{-\sqrt{-i\omega}\tau_d(\vec{x})}\left(\nabla\cdot\mathbb{K}\nabla\tau_d(\vec{x})\right) \\ &+ \left(\left(-\sqrt{-i\omega}\right)\nabla\left(A_0(\vec{x})\,e^{-\sqrt{-i\omega}\tau_d(\vec{x})}\right)\right)\cdot\mathbb{K}\nabla\tau_d(\vec{x}) \quad , \end{aligned} \tag{2.32}$$

where we can observe that the second term after the equality sign is equal to $-\sqrt{-i\omega}$ times the inner gradient at the left hand side of Equation (2.30). This gradient has already been written out in Equation (2.31), where we also observed that one of the terms became of higher order of $\sqrt{-i\omega}$. Hence, we can in the same way neglect the first term of Equation (2.32), and we end up with

$$LHS = (-i\omega)\,A_0(\vec{x})\,e^{-\sqrt{-i\omega}\tau_d(\vec{x})}\left(\nabla\tau_d(\vec{x})\cdot\mathbb{K}\nabla\tau_d(\vec{x})\right) \quad . \tag{2.33}$$

When inserting Equation (2.33) into the left hand side of Equation (2.30), and dividing both sides by their common factors, we end up with a form of the Eikonal equation

$$\nabla\tau_d(\vec{x})\cdot\mathbb{K}\nabla\tau_d(\vec{x}) = \phi(\vec{x})\,\mu c_t \quad , \tag{2.34}$$

where all the frequency dependent terms are cancelled out. The Eikonal equation can be efficiently solved with an implementation of the *Fast Marching Method (FMM)* presented by Sethian (1996), which will be presented later. For an isotropic medium Equation (2.34) reduces to

$$|\nabla \tau_d (\vec{x})| = \sqrt{\frac{\phi(\vec{x})\, \mu c_t}{k(\vec{x})}} \quad , \tag{2.35}$$

which is equal to the results found in Vasco et al. (2000) and Kulkarni et al. (2001) for isotropic permeability. Returning to the expression for particle time of flight in Equation (2.19), we have that as $\nabla \tau_t$ and $\vec{q}$ are in the same direction

$$|\nabla \tau_t (\vec{x})| = \frac{\phi}{q} \quad . \tag{2.36}$$

Comparing Equation (2.35) and Equation (2.36) we see that the expression for diffusive time of flight in an isotropic medium has a similar structure to the expression for particle time of flight. By dimensional analysis, it can also be seen that τ_d has the dimensions of $\sqrt{\text{time}}$, and it is therefore not the actual travel time of the pressure front. To connect τ_d to actual time Vasco et al. (2000) and Kulkarni et al. (2001) finds the inverse Fourier transform of the zeroth order approximation in Equation (2.29) in accordance with the solutions presented in Virieux et al. (1994). The limitation of this new expression for pressure is that it is dependent on the flow regime. For 1D flow, the solution is (Virieux et al., 1994)

$$p(\vec{x},t) = A_0 (\vec{x}) \frac{\tau_d(\vec{x})}{2\sqrt{\pi t}} e^{-\frac{\tau_d^2(\vec{x})}{4t}} \quad , \tag{2.37}$$

while for 2D radial flow the solution is (Virieux et al., 1994; Kulkarni et al., 2001)

$$p(\vec{x},t) = A_0 (\vec{x}) \frac{\tau_d(\vec{x})}{2\sqrt{\pi t}} e^{-\frac{\tau_d^2(\vec{x})}{4t}} \quad , \tag{2.38}$$

and lastly for 3D spherical flow is (Virieux et al., 1994; Kulkarni et al., 2001)

$$p(\vec{x},t) = A_0 (\vec{x}) \frac{\tau_d(\vec{x})}{2\sqrt{\pi} t^{3/2}} e^{-\frac{\tau_d^2(\vec{x})}{4t}} \quad . \tag{2.39}$$

As discussed previously in this section, the pressure solutions in Equations (2.37)-(2.39) are results of an impulse source/sink, which is the derivative of a step function, and the time of their maximum is therefore equivalent to the time of maximum pressure change in Equation (2.25). The maximum of e.g. the case of 2D radial flow in Equation (2.38) is (Kulkarni et al., 2001)

$$\frac{\partial p(\vec{x},t)}{\partial t} = A_0(\vec{x}) \frac{\tau_d(\vec{x})}{2\sqrt{\pi}} e^{-\frac{\tau_d^2(\vec{x})}{4t}} \left(-\frac{1}{t^2} + \frac{\tau_d^2(\vec{x})}{4t^3} \right) = 0 \quad , \tag{2.40}$$

and thus

$$t_{\max_{2\mathrm{D}}} = \frac{\tau_d^2(\vec{x})}{4} \quad . \tag{2.41}$$

Equivalent solutions for linear 1D flow and spherical 3D flow are $t_{\max_{1\mathrm{D}}} = \tau_d^2(\vec{x})/2$ and $t_{\max_{3\mathrm{D}}} = \tau_d^2(\vec{x})/6$ (Kulkarni et al., 2001; Virieux et al., 1994). For more complex flow regimes, as the numerical examples we will look at in this thesis, the transformation from τ_d to actual time is not as easy, but τ_d can still be used as a tool to evaluate the structure of the pressure propagation.

2.3 Heterogeneity Measures

The following section will present some existing heterogeneity measures. There are two main classes of heterogeneity measures (Jensen et al., 2000): *static* and *dynamic*. As the terms imply, the static measures use the static parameters of the formation to estimate the heterogeneity, while dynamic measures are based upon simple flow experiments. The dynamic measures are therefore measures of the heterogeneity effect on flow, and hence they are more likely to give better correlations with oil recovery. The dynamic measures presented here will later be correlated with simulated recovery data, and compared with potential new heterogeneity measures that are presented in Section 3.4.

2.3.1 Static Heterogeneity Measures

The following static models are heterogeneity models that are based on a vertical cross-section, and assumes that the reservoir properties at the cross-section continues infinitely away from the cross-section. A typical example is a vertical well passing through several horizontal layers, for which the properties are assumed constant. Two parameters will have to be introduced, the static *Flow Capacity* - F and the static *Storage Capacity* - Φ. For a reservoir of n layers, each of thickness h_i, permeability k_i and porosity ϕ_i we have that the relative velocity of a neutral particle in the different layers, is related to the ratio $r_i = k_i/\phi_i$ (Lake, 1989). If the layers are first sorted by decreasing r, the cumulative flow capacity up to layer j is

$$F_j = \frac{\sum_{i=1}^{j} Q_i}{\sum_{i=1}^{n} Q_i} = \frac{\sum_{i=1}^{j} k_i h_i}{\sum_{i=1}^{n} k_i h_i} \quad , \tag{2.42}$$

hence F_j is the fraction of the total flow that flows faster or at the same speed as layer j (Lake, 1989). The cumulative storage capacity up to layer j is defined as

$$\Phi_j = \frac{\sum_{i=1}^{j} V_{p_i}}{\sum_{i=1}^{n} V_{p_i}} = \frac{\sum_{i=1}^{j} \phi_i h_i}{\sum_{i=1}^{n} \phi_i h_i} \quad , \tag{2.43}$$

which is the fraction of the pore volume that flows faster or at the same speed as layer j.

2.3.1.1 Lorenz Coefficient

The static Lorenz coefficient can be obtained by plotting F as a function of Φ, called a F-Φ curve (see e.g. Lake, 1989; Shook and Mitchell, 2009). The curve will be a straight line from 0 to 1 for a homogeneous reservoir, and twice the area between the actual curve and the line between 0 and 1 is used as a heterogeneity measure. The Lorenz coefficient is thus defined as

$$L_c = 2\left(\int_0^1 F d\Phi - 0.5\right) \quad . \tag{2.44}$$

Figure 2.1 shows a F-Φ diagram with a Lorenz coefficient L_c=0.67. As seen from both Figure 2.1 and Equation (2.44), the Lorenz coefficient will always be in the range 0 (homogeneous) to 1 (infinitely heterogeneous).

2.3.1.2 Dykstra-Parsons Coefficient

The Dykstra-Parsons coefficient is in its simplest form defined as (see e.g. Jensen et al., 2000)

$$V_{DP} = \frac{k_{50} - k_{84.1}}{k_{50}} \quad , \tag{2.45}$$

where the permeabilities are sorted in descending order and k_{50} is the median permeability and $k_{84.1}$ is the permeability value for which 84.1% of the permeabilities are either equal or higher. As 84.1% is one standard deviation, the coefficient is also called the coefficient

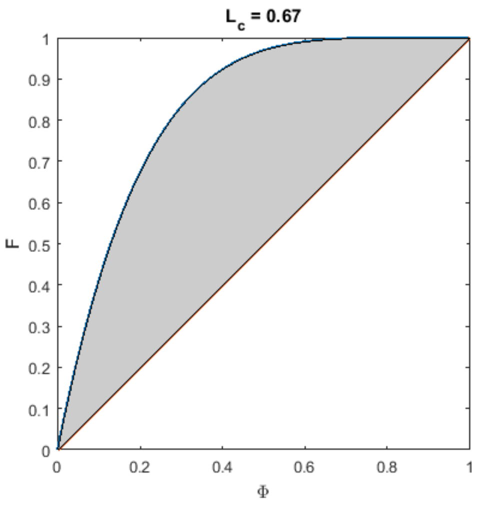


Figure 2.1: F-Φ curve for a model with Lorenz coefficient = 0.67.

of permeability variance (Jensen et al., 2000). There is also a modified variant of the Dykstra-Parsons coefficient for which the porosity heterogeneity is taken into account. If we again use the ratio $r = {}^{k}\!/\!_{\phi}$, and define the mean of all r_i as $\bar{r}$, then the modified Dykstra-Parsons coefficient is (Lake, 1989)

$$\begin{aligned} V_{DP} = \frac{\left(\frac{dF}{d\Phi}\right)_{\Phi=0.50} - \left(\frac{dF}{d\Phi}\right)_{\Phi=0.841}}{\left(\frac{dF}{d\Phi}\right)_{\Phi=0.50}} &= \frac{\frac{r(\Phi=0.50)}{\bar{r}} - \frac{r(\Phi=0.841)}{\bar{r}}}{\frac{r(\Phi=0.50)}{\bar{r}}} \\ &= \frac{r(\Phi = 0.841) - r(\Phi = 0.841)}{r(\Phi = 0.50)} \quad , \end{aligned} \tag{2.46}$$

where $r(\Phi = 0.50)$ and $r(\Phi = 0.841)$ are the ratios ${}^{k}\!/\!_{\phi}$ of the latest layer to break through when Φ reaches 0.5 and 0.841, respectively. Hence, the modified Dykstra-Parsons coefficient is related to the $F-\Phi$ curve, but while the Lorenz coefficient uses the area under the full curve, the Dykstra-Parsons coefficient only uses the relative difference between the median of the derivative and 1 standard deviation above.

2.3.2 Dynamic Heterogeneity Measures

The dynamic heterogeneity measures are as mentioned based on flow experiments, and we will see that streamline simulations are often useful to estimate them with limited computational cost.

2.3.2.1 Lorenz Coefficient

The dynamic Lorenz coefficient is based on the same principle as the static which was presented in Section 2.3.1.1, but it as applicable to more general reservoirs than the horizontally layered model with equal layer size. Shook and Mitchell (2009) defines the cumulative flow capacity for layer j, F_j, as

$$F_j = \frac{\sum_{i=1}^{j} Q_i}{\sum_{i=1}^{n} Q_i} \quad , \tag{2.47}$$

where Q_i is the volumetric flow rate of the i^{th} streamtube to reach breakthrough. The cumulative storage capacity is then

$$\Phi_j = \frac{\sum_{i=1}^{j} V_{p_i}}{\sum_{i=1}^{n} V_{p_i}} = \frac{\sum_{i=1}^{j} Q_i \tau_{t_i}}{\sum_{i=1}^{n} Q_i \tau_{t_i}} \quad , \tag{2.48}$$

where τ_{t_i} is the time of flight for the i^{th} streamline/streamtube at breakthrough. The time of flight from inlet to outlet is also known as the *residence time*. Equation (2.48) represents a complete piston like displacement in each streamtube, e.g. a single-phase or unit mobility flow with constant rate. In such a case, the pore volume of the streamtube has to be the product of rate and time. The dynamic Lorenz coefficient is then defined equally as the static through Equation (2.44).

2.3.2.2 Flow Heterogeneity Index

Shook and Mitchell (2009) also briefly discusses a *flow heterogeneity index* (FHI) that can be obtained from the F-Φ curve, referring to private communication with a C. Harrison in Chevron. The FHI is defined as

$$FHI = \frac{F}{\Phi}\Big|_{\frac{\partial F}{\partial \Phi}=1} \quad . \tag{2.49}$$

We observe that FHI is 1 for a homogeneous reservoir, and increasing with increasing heterogeneity. In theory there is no upper limit, and a L_c=1 is equivalent to FHI=∞. Wu et al. (2008) showed that the derivative of the F-Φ-curve also can be expressed as

$$\frac{\partial F}{\partial \Phi} = \frac{\bar{\tau}_t}{\tau_{t_i}} \quad , \tag{2.50}$$

which means that the cumulative ${}^{F}/_{\Phi}$ ratio for the streamtube which has time of flight equivalent to the average time of flight $\bar{\tau}_t$ can be used as the FHI measure, without finding the derivative of the full F-Φ curve. For the case in Figure 2.1 the unit slope occurs at Φ=0.33 and F=0.865, hence FHI=2.62. Shook and Mitchell (2009) conclude that FHI is a good measure of heterogeneity, though the dynamic Lorenz coefficient is better.

2.3.2.3 Koval Factor

Koval (1963) originally introduced the Koval heterogeneity factor H_K, which is defined as Jensen et al. (2000)

$$H_K = \frac{1}{t_{D_b}} \quad , \tag{2.51}$$

where t_{D_b} is the dimensionless breakthrough time for a single phase / unit mobility ratio system equal to

$$t_{D_b} = \frac{V_{\text{inj}_b}}{V_p} = \frac{t_b \sum_{i=1}^{n} Q_i}{\sum_{i=1}^{n} Q_i \tau_{t_i}} = E_{V_b} \quad . \tag{2.52}$$

The last equality of Equation (2.52) holds for a set of n streamtubes, each with a rate Q_i and breakthrough-time τ_{t_i} where we have a complete displacement, and where the shortest time of breakthrough for the set of streamtubes is t_b. We can see that the dimensionless breakthrough time actually is the volumetric sweep efficiency E_V at time of breakthrough. Idrobo et al. (2000) have used sweep efficiency at breakthrough as a heterogeneity measure with good results, in addition to sweep efficiency at a later time. Shook and Mitchell (2009) suggest that sweep efficiency at 1 pore volume injected (PVI) is a better choice, as the breakthrough data in their results gave multiple curves for various permeability correlation lengths, while 1 PVI gave a single trend. Equivalently to FHI in Section 2.3.2.2, the Koval factor has a lower limit of 1 for heterogeneous models, and no upper limit. Volumetric sweep efficiency at time of breakthrough is on the other hand limited by 0 for an infinitely heterogeneous reservoir, and 1 for a homogeneous reservoir.

2.3.2.4 Vorticity Factor

There is a relation between permeability variation and rotation of the flow field. The curl of the Darcy velocity, called the *vorticity*, is related to the gradient of the logarithm of the permeability field. Under the assumption of incompressible fluid, constant viscosity and no gravity effects, Heller (1963) showed that the vorticity can be expressed as

$$\omega = \nabla \times \vec{q} = \nabla \ln k \times \vec{q} \quad . \tag{2.53}$$

The vorticity is a local measure. In the case where the flow direction and the gradient of the logarithm of permeability are aligned, the cross product will be zero, and the flow is locally vorticity free. The same happens if the permeability is locally homogeneous and isotropic. It is obvious from Equation (2.53) that vorticity is related to heterogeneity in the permeability field, but to use it as a heterogeneity measure, the local vorticities have to be merged to an overall parameter. Rashid et al. (2010) introduces a parameter called the shear homogeneity index H_s, which was further developed to a vorticity index H_v in Rashid et al. (2012). The vorticity index is defined as

$$H_V = C_V(|\omega|) = \frac{\sigma_{|\omega|}}{\overline{|\omega|}} \quad , \tag{2.54}$$

where $C_V(|\omega|)$ is the so called *coefficient of variation* of the (absolute value of) vorticity, which is the ratio of standard deviation $\left(\sigma_{|\omega|}\right)$ and the mean $\left(\overline{|\omega|}\right)$. It can be noticed that Rashid et al. (2010) and Rashid et al. (2012) defines the coefficient of variation as the inverse of how it is defined here, and thus $H_V = {}^{1}/{C_V}$. The definition of C_V in Equation (2.54) is though more commonly used (see e.g. Jensen et al., 2000). H_V will generally decrease with increasing heterogeneity, as the mean usually increases faster than the standard deviation. For a reservoir model of an infinite number of cells the limits are 0 and ∞, but as more thoroughly discussed in (Rashid et al., 2010) and (Rashid et al., 2012), a limited number of cells will introduce lower bounds > 0 and upper bounds $< \infty$ due to the definition of variance of a limited set. We can also see that as long as the mean is larger than the variance, H_V will be below 1. Jensen et al. (2000) also covers the general properties of the coefficient of variation in even more detail (see e.g. Chapter 6-2.1 in Jensen et al., 2000).

2.3.2.5 Coefficient of Variance for Time of Flight

Shook and Mitchell (2009) also suggests to use the coefficient of variance for streamline time of flight as a heterogeneity measure, but they observed the same effect as for breakthrough sweep efficiency that was briefly discussed in Section 2.3.2.3: Multiple curves for various permeability correlation lengths were observed when comparing with oil recovery, and they would therefore not recommend the coefficient of time of flight variance. Shook and Mitchell (2009) define this measure as

$$C_V(\tau_t) = \frac{\sigma_{\tau_t}}{\bar{\tau}_t} \quad , \tag{2.55}$$

where σ_{τ_t} is the rate weighted standard deviation of the residence time distribution, and $\bar{\tau}_t$ is the rate weighted mean residence time (time from inlet to outlet).

2.4 Geological Model: SPE10, Model 2

The Society of Petroleum Engineers (SPE) released two synthetic models for the 10th SPE Comparative Solution Project on Upscaling, a project which is summarized by Christie and Blunt (2001). The second model has later been used been used in several studies within the fields of upscaling and heterogeneity measures (see e.g.: Rasaei and Sahimi, 2007; Rashid et al., 2012; Krogstad et al., 2017), which is the main reason why this model has been chosen for this work. The model will later be referred to simply as the *SPE10 model*. The model is already available in MRST's *spe10* module, though the raw data can also be downloaded directly from the project homepage (Society of Petroleum Engineers, 2000).

The model consists of 1,122,000 equally sized grid cells, where each cell is 20ft $\times$ 20ft $\times$ 2ft. The grid cells are distributed into 85 horizontal layers with 60 $\times$ 220 grid cells. Each layer will in this thesis be used as a separate geological realization, such that all examples are for two-dimensional horizontal flow in a single layer. The model is meant to be representable for the upper part of the Ness formation and the Tarbert formation in the North Sea Brent group. Many of the major oil fields in the North Sea have reservoirs in the Brent group, such as the British Brent field and the Norwegian Statfjord and Gullfaks fields. The Tarbert formation was deposited as a marginal marine environment (shoreface), while the underlying Ness formation is thought to be a delta plain/fluvial environment (Norwegian Petroleum Directorate, 2014).

The porosity and permeability fields are shown in Figure 2.2, where the Tarbert formation is lifted to let us see a horizontal layer of the Upper Ness formation underneath. We observe that there is a close correlation between high porosity and high permeability regions. The Tarbert layers have less sharp contrasts, but there is still a large variation of permeability and porosity values. The fluvial channels in the Upper Ness formation (lower 50 layers) are easily observed as high permeability and porosity regions with sharp contrast to the backdrop. The probability density functions of porosity and permeability for each formation are shown in Figure 2.3. We observe that the Tarbert formation has a single normal distribution of porosity, while the Upper Ness formation is a composite of two normal distributions, one for the channels and one for the backdrop. The permeability is for both formations log-normal distributed, and again we observe that the distribution for Upper Ness is a composite of two distributions.

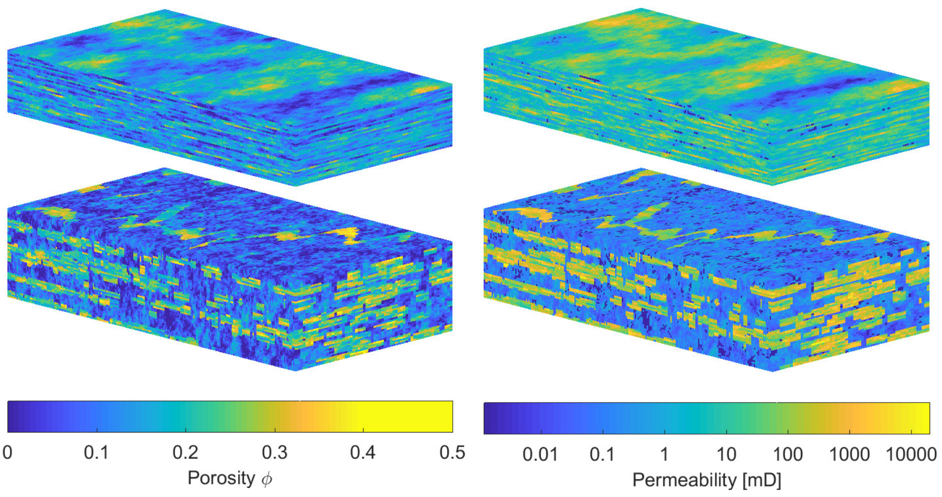


Figure 2.2: SPE10 model 2. The 35 layers of the Tarbert formation are lifted to visualize a horizontal section of the Upper Ness formation underneath. Vertical direction exaggerated by 5 times compared to horizontal directions.

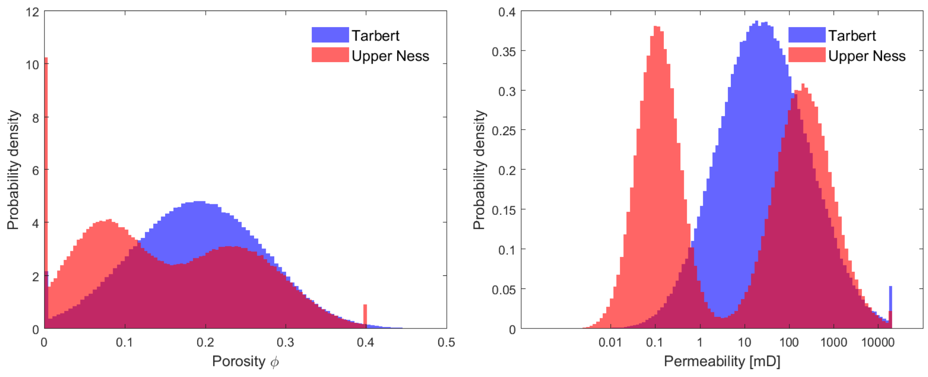


Figure 2.3: SPE10 model 2. Porosity and permeability distributions (free after Lie, 2016).

Chapter 3

Methodology

3.1 Streamline Tracking and Time of Flight Pollock's Algorithm

3.1.1 Algorithm

To be able to use streamline and streamtube based methodologies in reservoir modelling, we need to have a way track them in space. For two-dimensional flow it is possible to calculate a stream function by a corner-point discretization, from which we can trace the streamlines that enclose a streamtube. When expanded to three dimensions this is though not feasible, as calculating the geometry of the streamtube to obtain its cross-sectional area at all points becomes a tedious task. A solution to this problem is to instead track streamlines at the center of streamtubes, and then allocate a constant rate to them based on their rate at the inlet, assuming that the streamline path is sufficiently representable for the streamtube throughout the reservoir (Datta-Gupta and King, 2007). The most commonly used methodology for streamline tracking in finite-difference models is the algorithm presented by Pollock (1988). The algorithm is based on linear interpolation of velocity internally in each cell to calculate cell-by-cell streamline paths. The algorithm provides mass balance, but can result in unrealistic velocity discontinuities at cell intersections. Pollock's method can be described by taking a cell of dimensions Δx, Δy and Δz in a Cartesian coordinate system, as shown in Figure 3.1. The average flow velocity over each cell face can be calculated by dividing the rate over the cell face by the cross-sectional area of the pore volume at the cell face. As an example, the velocity v_{x_i} in the x-direction will be used, but the velocities in all directions can be calculated similarly:

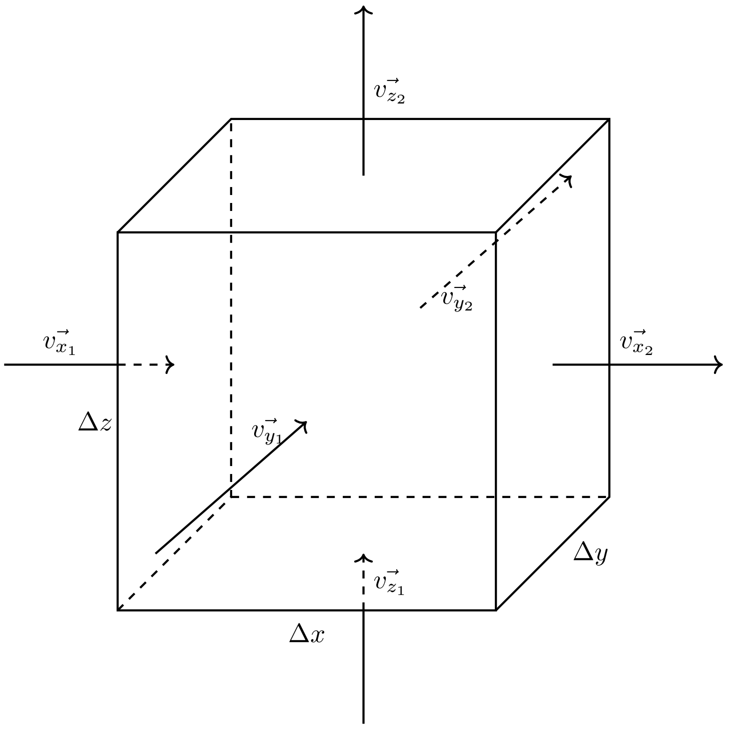


Figure 3.1: A cell with average velocities over each surface marked (free after Pollock, 1988).

$$v_{x_i} = \frac{q_{x_i}}{\phi} = \frac{Q_{x_i}}{\phi A} = \frac{Q_{x_i}}{\phi \Delta y \Delta z} \quad , \tag{3.1}$$

where Q_{x_i} is the flow rate over the cell face; A is the area of the cell face and; ϕ is the porosity of the cell and v_{x_i} is the average flow velocity in the x-direction over the cell face. The mass balance of the cell is then conserved through the following equation, with Q_{src} as the production/injection rate in the cell

$$\phi \left(\frac{v_{x_2} - v_{x_1}}{\Delta x} + \frac{v_{y_2} - v_{y_1}}{\Delta y} + \frac{v_{z_2} - v_{z_1}}{\Delta z} \right) = \frac{Q_{src}}{\Delta x \Delta y \Delta z} \quad , \tag{3.2}$$

which is a discretization of

$$\nabla \cdot q = Q_{src} \quad . \tag{3.3}$$

The velocity of a neutral particle can then be calculated by linear interpolation independently in each direction as

$$\vec{v}(\vec{x}) = \begin{bmatrix} v_x(x) \\ v_y(y) \\ v_z(z) \end{bmatrix} = \begin{bmatrix} (x-x_1)\frac{v_{x2}-v_{x1}}{\Delta x} + v_{x_1} \\ (y-y_1)\frac{v_{y2}-v_{y1}}{\Delta y} + v_{y_1} \\ (z-z_1)\frac{v_{z2}-v_{z1}}{\Delta y} + v_{z_1} \end{bmatrix} \quad . \tag{3.4}$$

Pollock (1988) then calculates the time of flight to each exit cell face, and uses the lowest value as the correct time of flight for the cell. From Equation (2.1) we have that

$$d\tau_t = \frac{dx}{v_x(x)} = \frac{dy}{v_y(y)} = \frac{dz}{v_z(z)} \quad . \tag{3.5}$$

Let us denote the point of entry as $a = [x_a, y_a, z_a]$ and the point of exit $b = [x_b, y_b, z_b]$. Each cell face are described by a constant x, y or z value, and the time of flight to reach the cell face can therefore be found by integrating Equation (3.5) from point a to the cell face for the relevant coordinate direction. The following example is given for time of flight to the cell face described by $x = x_1$, but the same methodology can be applied in all directions

$$\Delta\tau_{t_{x_1}} = \int_{x_a}^{x_1} \frac{dx}{(x-x_1)\frac{v_{x2}-v_{x1}}{\Delta x} + v_{x_1}} = \frac{\Delta x}{v_{x_2} - v_{x_1}} \ln\left(\frac{v_{x_1}}{v_x\left(x_a\right)}\right) \quad . \tag{3.6}$$

A couple of special considerations need to be made. If $v_x\left(x_a\right) = 0$ in Equation (3.6) above, there will not be a time of flight in the x-direction. Also, if $v_{x_1} = v_{x_2}$ the integral in Equation (3.6) will become undefined, while it in reality means a constant v_x. The correct equation for the time of flight to the surface $x = x_1$ when $v_{x_1} = v_{x_2} = v_x$ is therefore

$$\Delta\tau_{t_{x_1}}(v_x = constant) = \int_{x_a}^{x_1} \frac{dx}{v_x} = \frac{(x_1 - x_a)}{v_x} \quad . \tag{3.7}$$

The actual time of flight for the cell ($\Delta\tau_t$) is the lowest positive time of flight of the individual exit faces. After finding $\Delta\tau_t$ for the cell, the exit point $b = [x_b, y_b, z_b]$ can be found by inserting the known $\Delta\tau_t$ into (3.6). For the x-coordinate we get

$$\Delta\tau_t = \int_{x_a}^{x_b} \frac{dx}{(x-x_1)\frac{v_{x2}-v_{x1}}{\Delta x} + v_{x_1}} \quad , \tag{3.8}$$

which results in

$$x_b = x_1 + \frac{\Delta x}{v_{x_2} - v_{x_1}} \left(v_x(x_a) * e^{\left(\frac{v_{x_2} - v_{x_1}}{\Delta x} \Delta \tau_t\right)} - v_{x_1} \right) \quad . \tag{3.9}$$

The approaches for the other two directions y and z are equal. In the special case $v_x(x_a) = 0$ will $x_b = x_a$, and if $v_x = v_{x_1} = v_{x_2}$ then $x_b = x_a + v_x \Delta \tau_t$. After completing the calculations for the current cell, the algorithm repeats the process in the cell that shares cell face through which the streamline exits.

Pollock's algorithm estimates both coordinates and time of flight for a streamline, and hence we have a way to solve Equation (2.17). Increased coordinate density can be achieved by taking substeps in time internally in the cell when calculating coordinates.

3.1.2 Expansion of MRST Implementation

An implementation of the Pollock algorithm is already available in the current version of MRST (2017b), but some modifications to the current implementation of the algorithm has been done during the work for this thesis. More information about the MRST software is available in Lie (2016). The existing implementation returns up to three parameters for each step:

- Streamline coordinates
- Current grid cell
- Time of Flight

In addition to these three, the modified version also returns:

- Velocity vector
- Streamline length
- Pressure gradient

In addition to new available outputs, support for periodic grids has also been added. Periodic grids can e.g. be useful in upscaling when the assumption of no flow boundaries is not necessarily realistic. All the added functionalities will be described in the following sections. The codes for the recommended new variation of the Pollock algorithm implementation is found in Appendix A.3.

3.1.2.1 Velocity Vector

The existing MRST implementation solves the particle movement in each grid cell by transforming the coordinate system to a unit cell. Returning the velocity vector is therefore simply done by linear interpolation similar to Equation (3.4). The velocity vector can be used in the calculation of streamline lengths, as we will see in Section 3.1.2.2.

3.1.2.2 Streamline Lengths

Two options for streamline length calculations are added to the existing implementation. The first option is a simple straight line approximation between each coordinate, visualized in Figure 3.2. As it is a method of shortest path it will necessarily always be an underestimation of the length, but it is guaranteed to converge towards the right solution when increasing the number of substeps. Figure 3.2 shows an example of this straight line approximation for a streamline passing through two cells with two substeps in each grid cell. The distance s from point $\vec{x}_i$ to $\vec{x}_{i+1}$ is simply

$$s = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2} \quad . \tag{3.10}$$

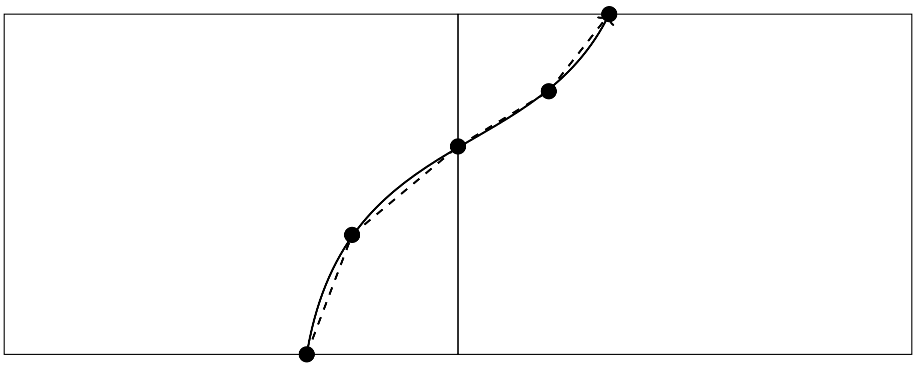


Figure 3.2: Streamline using two substeps. Solid line: Actual streamline. Dashed line: Straight line approximation. Black dots: Evaluation points in Pollock algorithm.

As the Pollock algorithm can return the velocity vector in each evaluation point, it is also possible to use these as a basis for length calculation. Velocity is the first derivative of distance, hence the path length s from point a to point p can be expressed as

$$s = \int_{\tau_{t_a}}^{\tau_{t_p}} |v(\vec{x}(t))| dt = \int_{\tau_{t_a}}^{\tau_{t_p}} \sqrt{v_x^2(t) + v_y^2(t) + v_z^2(t)} dt \quad , \tag{3.11}$$

where τ_{t_a} and τ_{t_p} are the respective time of flights at point a and point p. The velocity components of a cell as functions of time can be found by a similar approach as Equation (3.6), rearranging it for the unknown velocity at a time of flight τ_t and assuming that point a is the entry point in the cell. For the x-direction the velocity expression becomes

$$v_x\left(\tau_t\right) = v_x\left(x_a\right) e^{\frac{v_{x2}-v_{x1}}{\Delta x}\left(\tau_t - \tau_{t_a}\right)} \quad . \tag{3.12}$$

Inserting the componentwise velocities from Equation (3.12) into Equation (3.11) will result in an integral with no simple analytic solution. The integral can be approximated by using the mean velocity for each substep, leading to a trapezoidal approximation of the velocity integral between the evaluation points. The length s between point $\vec{x}_i$ and point $\vec{x}_{i+1}$ is then approximated as

$$s \approx \frac{v(\vec{x}_i) + v(\vec{x}_{i+1})}{2}\left(\tau_{t_{i+1}} - \tau_{t_i}\right) \tag{3.13}$$

Increasing the number of substeps is therefore expected to improve this length approximation. An advantage of this method compared to the straight line method is that it is not dependent on the Cartesian coordinates, which is useful for a periodic grid where the streamline can exit on one side and immediately reenter on the opposite side of the model.

3.1.2.3 Pressure Gradients

According to Darcy's law, the flow velocity will be in the negative pressure gradient direction, hence the pressure should be continuously decreasing along a streamline. Pressure values from a regular finite difference solver are cell-averaged pressures, which is a challenge when trying to create accurate gradients following arbitrary streamlines. The methods tested during the work for this thesis can be separated into 3 main groups

- Linear pressure interpolation between two cells
- Trilinear pressure interpolation between all surrounding cells
- 1D Darcy flow calculation

Linear Interpolation of Pressure
As the permeability and porosity is assumed constant within each cell, it is a relatively fair to assume that it is sufficient to find the pressure drop between each cell face. The pressure drop can then be divided by the step length between each cell, which was discussed in

Section 3.1.2.2. Figure 3.3 is an illustration of a streamline $\mathcal{S}$ through cell i and cell $i+1$. The proposed approximation of the pressure $p_{\mathcal{S}}$ at the streamline intersection is

$$p_{\mathcal{S}} \approx p_i + \left(\frac{p_j - p_i}{\frac{l_i}{k_i} + \frac{l_j}{k_j}} \right) \left(\frac{l_i}{k_i} \right) \quad , \tag{3.14}$$

where p_i and p_j are the average pressures of the previous and next cells, k_i and k_j are the permeabilities of the previous and next cells and l_i and l_j are the lengths of the directional vectors $\vec{l_i}$ and $\vec{l_j}$. In the case of permeability anisotropy, the permeability value used is the magnitude of the product of the permeability tensor $\mathbb{K}$ and the unit vectors $\vec{l_i}/l_i$ and $\vec{l_j}/l_j$. Equation (3.14) can then be rewritten as

$$p_{\mathcal{S}} \approx p_i + \left(\frac{p_j - p_i}{\frac{l_i}{|\mathbb{K}_i \frac{\vec{l_i}}{l_i}|} + \frac{l_j}{|\mathbb{K}_j \frac{\vec{l_j}}{l_j}|}} \right) \left(\frac{l_i}{|\mathbb{K}_i \frac{\vec{l_i}}{l_i}|} \right) = p_i + \left(\frac{p_j - p_i}{\frac{l_i^2}{|\mathbb{K}_i \vec{l_i}|} + \frac{l_j^2}{|\mathbb{K}_j \vec{l_j}|}} \right) \left(\frac{l_i^2}{|\mathbb{K}_i \vec{l_i}|} \right) \quad . \tag{3.15}$$

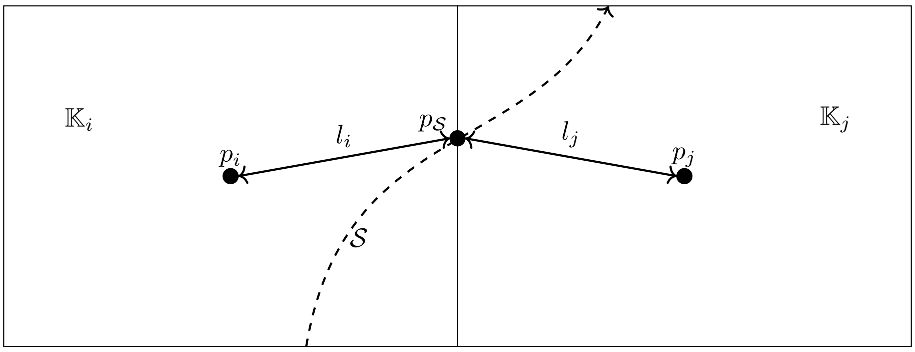


Figure 3.3: Approximation of streamline pressure at cell surface. The streamline follows the dashed line.

The permeability weighting is used as the pressure drop in Darcy's law has a linear relation to permeability, assuming that everything else is constant. Hence, the pressure is likely to be closer to the pressure in the cell with the highest permeability. The main advantage of this method is that it ensures continuously decreasing pressures, as flow over an intersection only occurs from a cell with higher pressure. As the pressures are interpolated between neighboring cells, the pressure estimations will never be very far off, as long as the finite volume solver that was used to generate the cell pressures is relatively accurate. The main disadvantage is that the resolution is low, with only one gradient per cell. In addition, the method does not use information from other cells. In e.g. the example of Figure 3.3 the pressure is likely to be higher at the bottom of the figure than at the top, as the main direction of the streamline is upwards. This effect is not caught by this method, as it would give the same output for a streamline crossing the cell face close to the top of the figure as a streamline crossing close to the bottom. The trilinear interpolation method presented next tries to mitigate this issue.

Trilinear Interpolation of Pressure

A trilinear interpolation of the surrounding cell pressures can potentially increase the pressure gradient resolution. The method proposed here contains of two main steps. First an estimation of pressures in all vertices (cell corners), and then a trilinear interpolation of pressure between the 8 vertices of each cell, assuming hexahedral and rectilinear grid cells. The pressure estimation of each vertex is done similarly as the recently presented linear interpolation method, but instead of using a weighted average of just two cells, we use all cells that borders to each vertex. In 3D this means that up to 8 cells can influence the pressure of each vertex. The pressure at any point in a cell can then be estimated by trilinear interpolation in space between the pressures in the cells vertices. Figure 3.4 shows a unit cell with pressures defined in each vertex. The trilinear pressure interpolation scheme for a pressure $p(\vec{x}) = p(x, y, z)$ in this unit cell becomes

$$p_{0,0} = p_{0,0,0}(1-x) + p_{1,0,0}x \tag{3.16}$$

$$p_{0,1} = p_{0,0,1}(1-x) + p_{1,0,1}x \tag{3.17}$$

$$p_{1,0} = p_{0,1,0}(1-x) + p_{1,1,0}x \tag{3.18}$$

$$p_{1,1} = p_{0,1,1}(1-x) + p_{1,1,1}x \tag{3.19}$$

$$p_0 = p_{0,0}(1-y) + p_{1,0}y \tag{3.20}$$

$$p_1 = p_{0,1}(1-y) + p_{1,1}y \tag{3.21}$$

$$p(x,y,z) = p_0(1-z) + p_1 z \quad . \tag{3.22}$$

As seen from Equation (3.22) we are first interpolating in the x-direction, then the y-direction, and at last the z-direction. There is no need for permeability weighting even in the case of permeability anisotropy, assuming that x, y and z are also the main directions of permeability anisotropy, since the x-, y- and z-directions are handled independently in the interpolation. A useful property of trilinear interpolation is that at an outer surface the pressure will only be a function of the vertices of the same surface, hence the calculated pressure at a point on a surface between two cells will be the same no matter which cell you use as a basis for the calculation. The pressure gradient can now be estimated by dividing the pressure drop between two points by the step length discussed in Section 3.1.2.2.

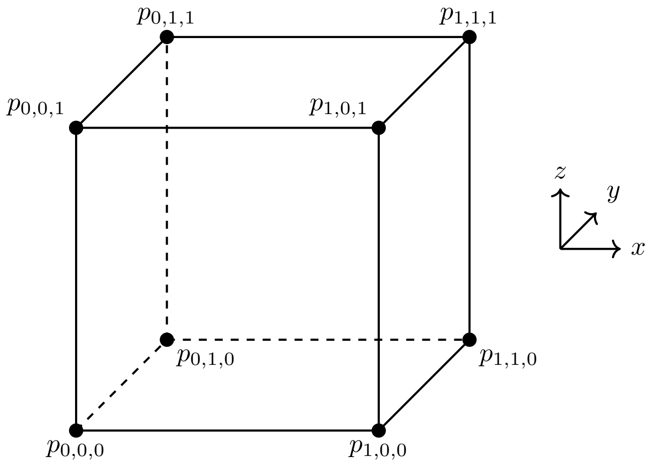
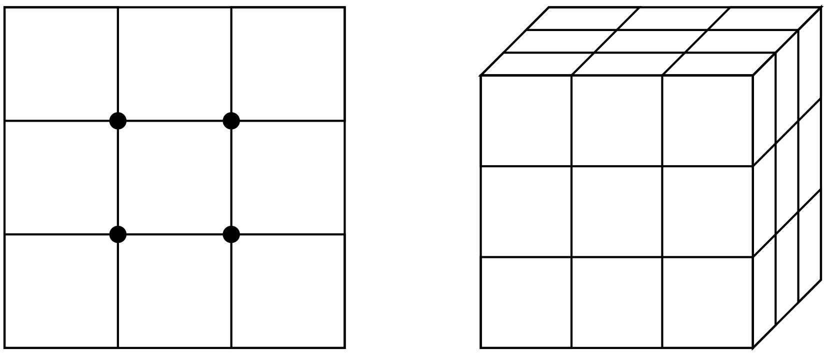


Figure 3.4: A cell with pressures defined in each vertex that can be used with trilinear interpolation

The main disadvantage of the trilinear interpolation method is that it does not guarantee that the pressures are continuously decreasing. This is a result of the fact that pressures of up to 27 cells can affect the resulting value. For the examples in this thesis which are all for single layer horizontal flow the maximum number of cells is reduced to 9, but it is still enough to introduce the possibility of pressure oscillations.



(a) The 9 cells affecting the pressures of the 4 vertices (black dots) of the central cell in 2D

(b) The 3x3x3=27 cells affecting the pressures of the 8 vertices of the central cell (not seen) in 3D

Figure 3.5: The surrounding cells that affect the pressure interpolation in the central cell

1D Darcy Pressure Gradient

From 1D Darcy's law we have that the pressure gradient is

$$\nabla p = -\frac{\vec{q}\mu}{k} = -\frac{\phi\vec{v}\mu}{k} \quad , \tag{3.23}$$

where $\vec{v}$ is the velocity vector from the Pollock algorithm. Since the path of the streamline is known we also know the porosity ϕ and the permeability k, which in the case of anisotropy is $k = |\mathbb{K}^{\vec{v}}/_{v}|$. Using the mean velocity, porosity and permeability for each substep, we find that the pressure gradient between $\vec{x}_i$ and $\vec{x}_{i+1}$ can be approximated as

$$\nabla p == -\frac{(\phi(\vec{x}_i) + \phi(\vec{x}_{i+1}))(\vec{v}(\vec{x}_i) + \vec{v}(\vec{x}_{i+1}))\mu}{2(k_i + k_{i+1})} = \quad , \tag{3.24}$$

where $k_i = |\mathbb{K}(\vec{x}_i)\frac{\vec{v}(\vec{x}_i))}{v(\vec{x}_i))}|$ in the case of anisotropy.

3.1.2.4 Periodic Boundary Conditions

Constant pressure and/or no flow boundary conditions are not always good assumptions, e.g. in an upscaling case where the boundaries are the boundary of a course grid cell, and we want to do streamline simulations on the fine grid within the course cell. For the SPE10 model, the layers are all rectangular cuboids, and it is likely that the reservoir would actually continue outside each layer. A periodic boundary condition can be used if we assume that the best guess for what lies outside each model is a copy of the model itself. In practice, it basically means that we allow flow to exit through one boundary and immediately let the same amount reenter through the same point at the opposite boundary as illustrated with a streamline in Figure 3.6, with a constant pressure drop between the two boundaries.

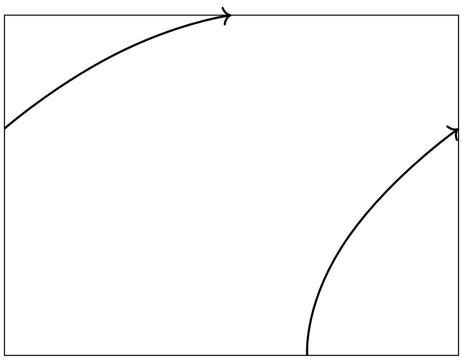


Figure 3.6: Streamline reentering at opposite side in a periodic grid

3.2 Diffusive Time of Flight Using Fast Marching Method

3.2.1 Algorithm

Eikonal equations such as Equation (2.34) can be numerically solved by a numerical method originally presented in Sethian (1996) and Sethian (1999) called the *Fast Marching Method*, or simply FMM. FMM provides the means to track a continuously advancing front, as the pressure front we are looking at for the diffusive time of flight. The implementation that is used in this thesis is a variety of FMM for anisotropic permeabilities presented in Zhang et al. (2013). The main idea of FMM is that the advancing front can be solved by an iterative solution with an upwind discretization, as the downwind points are not yet affected by the front and hence they cannot contribute to the solution.

FMM is a stepwise iterative algorithm. Each cell can have three possible states: *Accepted*, which means that the cells current value is final; *neighbor* which means that the cell borders at least one *accepted* cell; and *far*, which means the cell does not border to an *accepted* cell. The general form of the algorithm can be described by the following procedure (Sethian, 1999; Zhang et al., 2013), which is also presented as a flow chart in Figure 3.7:

1. Initialization: Give a set of start cells their initial value, which usually for diffusive time of flight is $\tau_d = 0$, and then mark their state as *accepted*.

2. Identification: Identify unaccepted neighbors of accepted cells.

   If no unaccepted neighbors: Quit.

3. Evaluation: Evaluate value for unaccepted neighbor based on upwind discretization from their accepted neighbors. Mark as *neighbors*.

4. Acceptance: Mark the *neighbor* with the lowest value as *accepted*. Return to step 2.

To find the diffusive time of flight using FMM, we need to discretize Equation (2.34) to be able to do the evaluation step. The derivation for a general hexahedral corner point grid can be found in Zhang et al. (2013), but a Cartesian grid with principal permeability directions in the x, y and z directions is assumed here, as it is sufficient for the examples later. We can then rewrite Equation (2.34) as

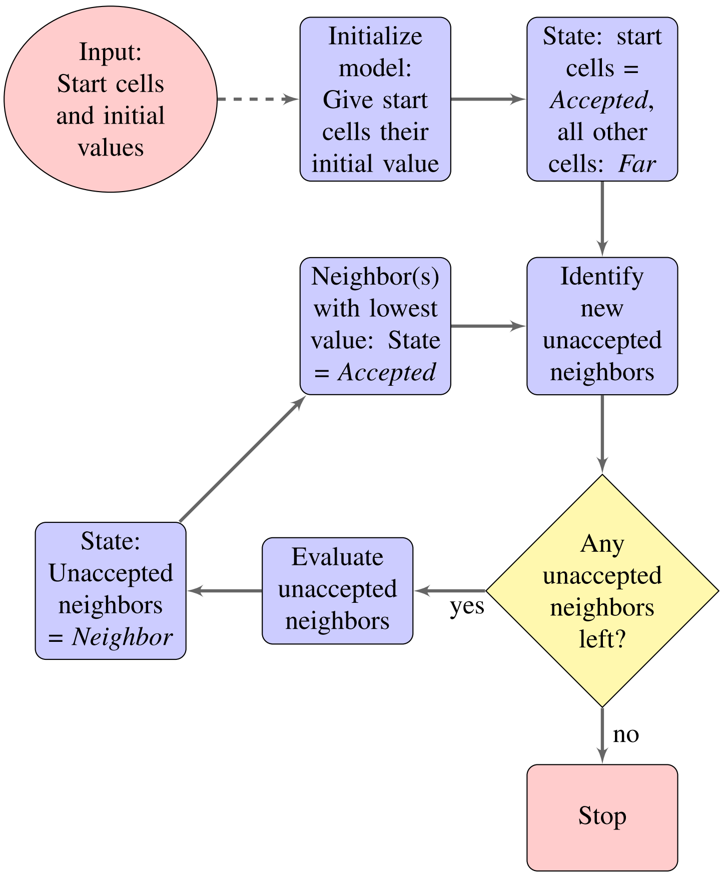

Input: Start cells and initial values
Initialize model: Give start cells their initial value
State: start cells = *Accepted*, all other cells: *Far*
Identify new unaccepted neighbors
Any unaccepted neighbors left?
yes
no
Stop
Evaluate unaccepted neighbors
State: Unaccepted neighbors = *Neighbor*
Neighbor(s) with lowest value: State = *Accepted*

Figure 3.7: Flow chart for Fast Marching Method.

$$\begin{aligned}\nabla\tau_d(\vec{x})\cdot\mathbb{K}\nabla\tau_d(\vec{x}) &= \begin{bmatrix}\frac{\partial\tau_d}{\partial x}\\ \frac{\partial\tau_d}{\partial y}\\ \frac{\partial\tau_d}{\partial z}\end{bmatrix}\cdot\begin{bmatrix}k_{xx} & 0 & 0\\ 0 & k_{yy} & 0\\ 0 & 0 & k_{zz}\end{bmatrix}\begin{bmatrix}\frac{\partial\tau_d}{\partial x}\\ \frac{\partial\tau_d}{\partial y}\\ \frac{\partial\tau_d}{\partial z}\end{bmatrix}\\ &= k_{xx}\left(\frac{\partial\tau_d}{\partial x}\right)^2 + k_{yy}\left(\frac{\partial\tau_d}{\partial y}\right)^2 + k_{zz}\left(\frac{\partial\tau_d}{\partial z}\right)^2 = \phi(\vec{x})\,\mu c_t \quad . \qquad (3.25)\end{aligned}$$

As discussed by Zhang et al. (2013), the pressure front can reach the cell center from one of four corners in 2D, or one of eight corners in 3D. Let us take the 3D numerical stencil in Figure 3.8 as an example. The cell we are evaluating is (i, j, k), which can have up to 6 accepted neighbor cells. For e.g. the corner defined by $(i-1, j, k)$, $(i, j, k+1)$ and

$(i, j-1, k)$, a numerical approximation to Equation (3.25) is (Zhang et al., 2013)

$$\frac{\left(\tau_{d_{i,j,k}} - \tau_{d_{i-1,j,k}}\right)^2}{\zeta_{i-1}^2} + \frac{\left(\tau_{d_{i,j,k}} - \tau_{d_{i,j-1,k}}\right)^2}{\zeta_{j-1}^2} + \frac{\left(\tau_{d_{i,j,k}} - \tau_{d_{i,j,k+1}}\right)^2}{\zeta_{k+1}^2} = 1 \quad , \tag{3.26}$$

where it is assumed that cells $(i-1, j, k)$, $(i, j, k+1)$ and $(i, j-1, k)$ are all accepted. Terms for unaccepted cells will be omitted, as they are not yet affected by the pressure front, and can therefore cannot contribute to the fronts propagation towards (i, j, k). The pressure front slowness, which is distance weighted between the cells, is governed by the factor ζ, which for e.g. ζ_{i-1} is defined as (Zhang et al., 2013)

$$\zeta_{i-1} = \frac{l_+(i-1, j, k)}{\sqrt{\frac{k_{xx(i-1,j,k)}}{\phi_{(i-1,j,k)}\mu c_t}}} + \frac{l_-(i, j, k)}{\sqrt{\frac{k_{xx(i,j,k)}}{\phi_{(i,j,k)}\mu c_t}}} \quad , \tag{3.27}$$

where $l_+(i-1, j, k)$ is the distance between the center of $(i-1, j, k)$ and the cell face with (i, j, k), and $l_-(i, j, k)$ is the distance between the center of (i, j, k) and the cell face with $(i-1, j, k)$. Keep in mind that if the total compressibility is assumed to be a function of porosity, then this value will also have to be handled similar to ϕ.

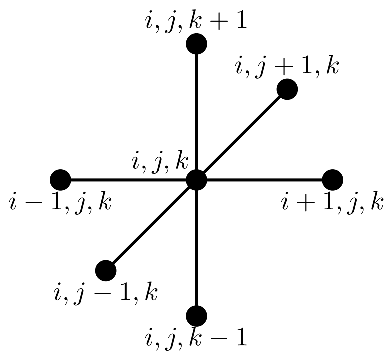


Figure 3.8: Numerical stencil in 3D (free after Sethian, 1999)

After computing all corners with at least one accepted neighbor, the updated value for the cell is the lowest value obtained from the individual corners, as this value will represent the first possible pressure front arrival. The cell can then be compared with the other cells which are defined as *neighbor*, for which the lowest value is considered *accepted* in accordance with the procedure illustrated in Figure 3.7. Note that the considered value for a cell specified as *neighbor* only needs to be recalculated in the next iteration in the case where one of its own neighbors becomes accepted.

3.3 Permeability Description Using Streamlines

3.3.1 Theory

Berg (2012) introduces a description of electric conductance in pore networks by the use of tortuosity and constriction factors of infinitesimal streamtubes in the pore space of a porous medium. The concept was further developed to describe permeability from a pore structure in Berg (2014), which is also partly discussed in Berg and Held (2016). For large scale reservoir modelling it is not possible to work with pore models, both as it is impossible to image the pore structure of a full reservoir, and because the simulations would be computationally impossible. It is therefore desirable to transfer the concept of pore structure permeability description to a larger scale, where the flow is controlled by Darcy's law (herein denoted as the Darcy scale). Section 2.1 presented the theoretical basis of streamlines and streamtubes in porous media at the Darcy scale. Combining this with the methodology in Berg (2014), we will find an analogous model for permeability description by the use of streamlines in Darcy scale reservoir models.

To describe the conversion from pore scale to Darcy scale permeability description, let us consider a rectangular horizontal layer of length Δs, cross-sectional area A, total pressure drop over the layer Δp, total volume $V = A\Delta s$ and total flow rate Q as indicated in Figure 3.9. Similarly to Berg (2014) we now introduce a set of infinitesimal streamtubes $\mathcal{S} \in \mathbb{S}$, where the set $\mathbb{S}$ of streamtubes covers the entire effective bulk volume V_e. The volume V_e is then the total volume of the grid cells that contribute to flow between inlet and outlet. From the divergence theorem we have

$$Q\Delta p = \int_{\partial V_e} p(\vec{q}\cdot\vec{n})dS = \int_{V_e} \nabla\cdot(p\vec{q})dV = \int_{V_e} \nabla p\cdot\vec{q} + p(\nabla\cdot\vec{q})dV = \int_{V_e} \nabla p\cdot\vec{q}dV, \tag{3.28}$$

where $\nabla\cdot\vec{q} = 0$ is used in the last equality. Combining Equation (3.28) with a 1D version of Darcy's law from Equation (1.1), then solving for permeability, we can get

$$k = -\frac{Q\mu\Delta s}{A\Delta p} = -\frac{Q\Delta p\mu\Delta s^2}{V\Delta p^2} = \frac{1}{V}\int_{V_e} -\mu(\nabla p\cdot\vec{q})\left(\frac{\Delta s}{\Delta p}\right)^2 dV \quad . \tag{3.29}$$

The streamline coordinates can now be introduced to Equation (3.29). From Section 2.1.2

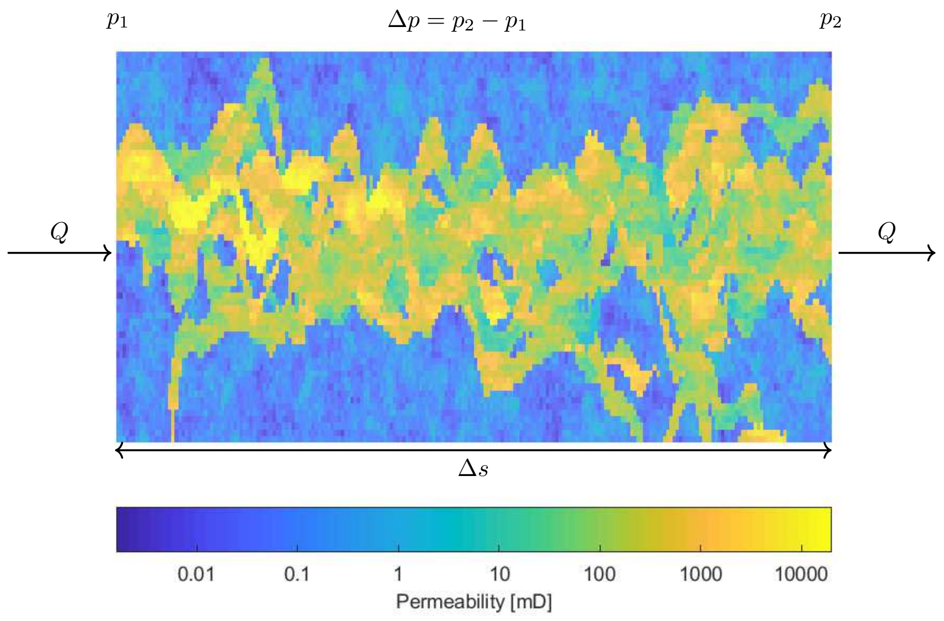


Figure 3.9: Reservoir model. Permeability field from Upper Ness formation (SPE10 model layer 80).

we have that the flow rate of a streamtube has to be constant. We can therefore use Equation (2.16) and allocate an infinitesimal flow rate $dQ_{\mathcal{S}} = d\lambda d\chi$ to the infinitesimal streamtube $\mathcal{S}$, so that $\int_{\mathbb{S}} dQ_{\mathcal{S}} = Q$. We can now write

$$\begin{aligned} k &= \frac{1}{V}\int_{V_e} -\mu(\nabla p \cdot \vec{q}) \left(\frac{\Delta s}{\Delta p}\right)^2 dV = \frac{1}{V}\int_{\mathbb{S}}\int_{\mathcal{S}} -\mu\frac{\nabla p \cdot \vec{q}}{q}\left(\frac{\Delta s}{\Delta p}\right)^2 ds dQ_{\mathcal{S}} \\ &= \frac{1}{V}\int_{\mathbb{S}} -\mu\Delta p \left(\frac{\Delta s}{\Delta p}\right)^2 dQ_{\mathcal{S}} = \frac{1}{V}\int_{\mathbb{S}} \kappa(\mathcal{S}) dQ_{\mathcal{S}} = \frac{Q}{V}\kappa(\mathcal{S}) \quad , \end{aligned} \tag{3.30}$$

where $\frac{1}{q}$ is the Jacobian for the variable change shown in Equation (2.15), and $\kappa(\mathcal{S})$ will be called the streamline permeability factor. Similarly to Berg (2014) we will here separate the streamline permeability factor in Equation (3.30) into streamline tortuosity, hydraulic conductance and constriction factors, resulting in

$$\kappa(\mathcal{S}) = -\frac{\mu\Delta s^2}{\Delta p} = \frac{\left(\frac{\Delta s}{l_{\mathcal{S}}}\right)^2 \int_{\mathcal{S}} -\frac{\mu}{\nabla p} ds}{\frac{1}{l_{\mathcal{S}}^2}\Delta p \int_{\mathcal{S}} \frac{1}{\nabla p} ds} = \frac{\tau^2(\mathcal{S}) B(\mathcal{S})}{C(\mathcal{S})} \quad , \tag{3.31}$$

where the streamline tortuosity $\tau(\mathcal{S})$, hydraulic conductivity $B(\mathcal{S})$ and constriction factor $C(\mathcal{S})$ will be further described in the following sections. Observe from Equations (3.30) and (3.31) that as μ, Δs and Δp are from the model and equal for all streamlines, the streamline permeability factor $\kappa(\mathcal{S})$ has to be a constant for all streamlines in a specific model.

The expression in Equation (3.31) is almost entirely equal to the expression derived in Berg (2014), but the pressure gradient at Darcy scale follows Darcy's law for flow in porous media, instead of Stokes's law as in the pore scale model in Berg (2014). We will therefore also see that the permeability descriptors have slightly different physical meaning at the Darcy scale.

3.3.2 Tortuosity $\tau(\mathcal{S})$

The streamline tortuosity is here defined as

$$\tau(\mathcal{S}) = \frac{\Delta s}{l_{\mathcal{S}}} \quad , \tag{3.32}$$

where $l_{\mathcal{S}}$ is the actual length of streamline $\mathcal{S}$. Observe that $\tau(\mathcal{S})$ is dimensionless. This is the same definition as in Berg (2014) and Bear (1972), and equivalent to the definitions in Berg (2012) and Berg and Held (2016), but the inverse of the tortuosity definition in e.g. Carman (1937). We have that $\tau(\mathcal{S}) \leq 1$, as Δs is the shortest possible path. When looking at flow at the pore scale, tortuosity occurs whenever the pore throats are not aligned in the same direction in the pore network, forcing the fluid to find paths around the grains. This form of tortuosity is though not the tortuosity we are considering at Darcy scale. As mentioned in the presentation of the vorticity heterogeneity index in Section 2.3.2.4, it was proved by Heller (1963) that the curl of the Darcy velocity field, known as the vorticity, is a function of the gradient of the logarithm of the permeability field through the following relation (assuming incompressible fluid with constant viscosity and no gravity effects):

$$\nabla \times \vec{q} = \nabla \ln k \times \vec{q} \quad . \tag{3.33}$$

Equation (3.33) shows that the streamlines, which per definition follows the velocity field, will not deviate from a straight line unless there is a changing permeability field. A line drive in a homogeneous reservoir model will therefore lead to $\tau(\mathcal{S}) = 1$, as expected. Heterogeneous models will have $\tau(\mathcal{S}) < 1$ unless the gradient of the permeability field is in the same direction as the velocity field at all points. The latter would result in $\nabla \times \vec{q} =$

$\nabla \ln k \times \vec{q} = \vec{0}$, and no vorticity.

3.3.3 Hydraulic Conductance $B(\mathcal{S})$

The term for streamline hydraulic conductance in Equation (3.31) will be defined as

$$B(\mathcal{S}) = \int_{\mathcal{S}} -\frac{\mu}{\nabla p} ds \quad . \tag{3.34}$$

The physical meaning of the streamline hydraulic conductance $B(\mathcal{S})$ can be further evaluated. The local pressure gradient of an infinitesimal streamtube $\mathcal{S}$ with constant rate $dQ_{\mathcal{S}}$ can be expressed by Darcy's law as

$$\nabla p = -\frac{dQ_{\mathcal{S}}\mu}{Ak} \quad , \tag{3.35}$$

where k is the local permeability in the direction of the streamtube, and A is the streamtubes instantaneous cross-sectional area. Inserting Equation (3.35) into Equation (3.34) we can then get

$$B(\mathcal{S}) = \int_{\mathcal{S}} -\frac{\mu}{\nabla p} ds = \int_{\mathcal{S}} k\frac{A}{dQ_{\mathcal{S}}} ds = \int_{\mathcal{S}} \frac{k}{q} ds = \int_{\mathcal{S}} \frac{k}{\phi v} ds = \int_{\mathcal{S}} \frac{k}{\phi} |\nabla \tau_t| ds \quad , \tag{3.36}$$

where q and v are the magnitudes of Darcy velocity and effective particle velocity in the direction of the streamline. As we can see from the expressions in Equation (3.36), the streamline hydraulic conductance is related to the local permeability and flow velocity over the length of the streamline, and it has dimensions area $\times$ time.

3.3.4 Constriction $C(\mathcal{S})$

After defining tortuosity and hydraulic conductance, the rest of Equation (3.31) will be called the streamline constriction factor, which hence is defined as

$$\frac{1}{l_{\mathcal{S}}^2} \Delta p \int_{\mathcal{S}} \frac{1}{\nabla p} ds \quad . \tag{3.37}$$

The streamline constriction factor at the pore scale as derived in Berg (2014) was shown to be a function of the variation of cross-sectional area of the streamtubes squared. This was a result of the relation between the pressure gradient and the velocity in the Stokes equation. The pressure gradient at Darcy scale is instead a function of the product of the cross-sectional area A of an infinitesimal streamtube and the permeability k in the direction of the streamtube, as shown for the hydraulic conductance $B(\mathcal{S})$ in the previous section. The Darcy scale constriction factor $C(\mathcal{S})$ is thus a function of the variation of Ak for an infinitesimal streamtube around $\mathcal{S}$, with a minimum value of 1, meaning no variation.

3.3.5 Effective Conductance, Tortuosity and Constriction

Like the methodology for streamline decomposition at the pore scale in Berg (2014), we want to find effective permeability components. These will then be parameters for the full model in question, and not just locally for individual streamtubes. They are therefore more likely to be representative for the general reservoir behavior, and might be potential measurements of e.g. heterogeneity as discussed later in Section 3.4.

The permeability described by pore scale permeability descriptors in Berg (2014) was a direct function of the ratio of the volume that contributes to flow, namely the effective porosity ϕ_e. The permeability at Darcy scale will also be dependent on the ratio of the volume that contributes to flow, though the effective volume is no longer the effective pore space, but instead the effective bulk volume V_e. Hence, we seek an expression for the permeability of the model of the form:

$$k = \frac{V_e B_e \tau_e^2}{V C_e} \quad . \tag{3.38}$$

Bear and Bachmat (1967) shows that the effective hydraulic conductance is the volume-weighted average of the local hydraulic conductance. Where the volume in question is the total effective bulk volume V_e. In a reservoir model, the total effective bulk volume will simply be the total volume of all cells that contribute to flow between inlet and outlet. We then get by averaging Equation (3.34)

$$B_e = \frac{1}{V_e} \int_{\mathbb{S}} B(\mathcal{S}) dQ_{\mathcal{S}} \quad . \tag{3.39}$$

An interesting observation can be done by reformulating the expression, inserting Equation 3.34 into Equation 3.39. Remember that ds, $-\nabla p$ and $\vec{q}$ all have the same direction. The variables in the following derivation can therefore be regarded as scalars.

$$B_e = \frac{1}{V_e}\int_{\mathbb{S}} B(\mathcal{S})dQ_{\mathcal{S}} = \frac{1}{V_e}\int_{\mathbb{S}}\int_{\mathcal{S}} -\frac{\mu}{\nabla p} ds dQ_{\mathcal{S}}$$

$$= \frac{1}{V_e}\int_{\mathbb{S}}\int_{\mathcal{S}} -\frac{q\mu}{q\nabla p} ds dQ_{\mathcal{S}} = \frac{1}{V_e}\int_{\mathbb{S}}\int_{\mathcal{S}} -\frac{k}{q} ds dQ_{\mathcal{S}} = \frac{1}{V_e}\int_{V_e} k dV = \bar{k} \quad . \tag{3.40}$$

The effective hydraulic conductance is thus in fact the volumetric average of the permeability for the effective bulk volume V_e. Especially interesting is it to note that B_e is in fact the volumetric average of permeability in the direction of flow at any point. True volumetric averaging can therefore now be performed even with permeability anisotropy for steady-state flow fields, resulting in a single scalar permeability value. This value will of course be dependent on direction of applied pressure gradient on the model, and it can therefore not be regarded as a pure rock parameter.

To find the tortuosity factor τ_e^2 we will again use the special constriction free situation with $C_e = C(\mathcal{S}) = 1$, which as discussed in Section 3.3.4 means no variation in the Ak product for an infinitesimal streamtube along $\mathcal{S}$. This leads to an expression which resembles the expression for the effective tortuosity factor in Berg (2014):

$$\tau_e^2 = \frac{Vk}{V_e B_e} = \frac{V\frac{1}{V}\int_{\mathbb{S}}\kappa(\mathcal{S})dQ_{\mathcal{S}}}{V_e\frac{1}{V_e}\int_{\mathbb{S}}B(\mathcal{S})dQ_{\mathcal{S}}} = \frac{\int_{\mathbb{S}}B(\mathcal{S})\tau^2(\mathcal{S})dQ_{\mathcal{S}}}{\int_{\mathbb{S}}B(\mathcal{S})dQ_{\mathcal{S}}} \quad . \tag{3.41}$$

The effective tortuosity is thus weighted against the streamline conductance $B(S)$, with more weight on tortuosity in high conductance streamlines. Finally, solving Equation (3.38) for C_e and inserting the expressions from Equations (3.31), (3.39) and (3.41):

$$C_e = \frac{V_e B_e \tau_e^2}{Vk} = \frac{\int_{\mathbb{S}}B(\mathcal{S})\tau^2(\mathcal{S})dQ_{\mathcal{S}}}{\int_{\mathbb{S}}\kappa(\mathcal{S})dQ_{\mathcal{S}}} = \frac{\kappa(\mathcal{S})\int_{\mathbb{S}}C(\mathcal{S})dQ_{\mathcal{S}}}{\kappa(\mathcal{S})\int_{\mathbb{S}}dQ_{\mathcal{S}}} = \frac{1}{Q}\int_{\mathbb{S}}C(\mathcal{S})dQ_{\mathcal{S}} \quad . \tag{3.42}$$

With this rate-weighted constriction factor; a possible decomposition of the permeability for the full model to an effective hydraulic conductance factor, an effective tortuosity factor, an effective constriction factor and the ratio of effective to total bulk volume, is hence shown to exist at Darcy scale for a porous medium with a streamline-based approach. The relation is of the form we sought in Equation (3.38), and therefore equal to

$$k = \frac{V_e B_e \tau_e^2}{V C_e} \quad . \tag{3.43}$$

The methodology is seen as analogous to the pore scale model in Berg (2014).

3.4 New Heterogeneity Measures

This section will cover new proposed heterogeneity measures. They will be compared with existing heterogeneity measures in Chapter 4, using simulated recovery factors at 90% water cut in a two-phase oil/water system, and increased recovery due to polymer injection as benchmarks. The setup for the examples are further described in Section 3.5.

3.4.1 Tortuosity and Constriction Based Heterogeneity Measures

The effective tortuosity factor τ_e^2 from Equation (3.41), is a rate and hydraulic conductance weighted mean tortuosity of the streamlines, and it is therefore a measure of the average deviation of a straight path between inlet and outlet. Increased tortuosity is likely to increase the probability of bypassed/unswept regions, and can therefore be a potential heterogeneity measure in itself.

The coefficient of variation of the set of streamline tortuosities $\tau^2(\mathcal{S})$ can also have potential as a heterogeneity measure. The advantage over the effective τ_e^2 is that it says more about the disordinance of the flow field. If e.g. close to all the flow occurs in a channel, as will be seen in some of the examples, the tortuosity of the individual streamlines can be close to equal, as they follow the same channel. This will lead to a lower coefficient of variation, even though the flow might be very tortuous. The mean tortuosity used will be the weighted τ_e^2 from Equation (3.41), and the weighted variance is then

$$\sigma_{\tau^2}^2 = \frac{\int_{\mathbb{S}} B(\mathcal{S})\left(\tau^2(\mathcal{S}) - \tau_e^2\right)^2 dQ_{\mathcal{S}}}{\int_{\mathbb{S}} B(\mathcal{S}) dQ_{\mathcal{S}}} \quad . \tag{3.44}$$

The standard deviation is the square root of variance, hence the coefficient of variation becomes

$$C_V(\tau^2) = \frac{\sigma_{\tau^2}}{\tau_e^2} \quad . \tag{3.45}$$

The other parameters of the permeability description in Equation (3.38) are not likely to be good measures for heterogeneity, at least when compared with recovery potential. As shown in Section 3.3.5, the hydraulic conductance B_e is simply a volumetric average of the permeability tangential to flow. In addition, we have seen that $B(\mathcal{S})$ and B_e does not have the same dimensions (area$\times$time and area, respectively), and they can therefore not be compared directly in e.g. a variance similar to Equation (3.44). The ratio $^{V_e}/_{V}$ of the bulk volume that contributes to flow is related to heterogeneity, but it will not necessarily have any effect on recovery, as the inactive parts often will be low porosity zones. The last parameter, the constriction factor C_e, is definitely also related to heterogeneity, as it is a measure of the variation of the pressure gradient along the flow path. On the other hand, permeability variations perpendicular to the flow direction will not decrease the areal sweep efficiency, as the flow direction will be the same even though it will increase the constriction factor. Hence, the correlation with recovery is not necessarily strong. However, correlations with a rate weighted variance and coefficient of variance of the inverse of the constriction factor have been tested, defining the variance as

$$\sigma^2_{\frac{1}{C}} = \frac{1}{Q}\int_{\mathbb{S}} \left(\frac{1}{C(\mathcal{S})} - \frac{1}{C_e} \right)^2 dQ_{\mathcal{S}} \quad . \tag{3.46}$$

It should be noted that the inverse of the constriction factor is used instead of the constriction factor alone, as it is the inverse that is the contribution in the permeability in Equation (3.43). Still, the rate weighting is left the same way, as the inverse of the constriction will have larger effect in streamtubes of higher rate. The coefficient of variation has then been defined as

$$C_V\left(\frac{1}{C}\right) = \frac{\sigma_{\frac{1}{C}}}{\frac{1}{C_e}} = \sigma_{\frac{1}{C}} C_e \quad . \tag{3.47}$$

3.4.2 Sweep Efficiency From Diffusive Time of Flight

Section 2.3.2.3 presented heterogeneity measures based on streamline time of flight, including the Koval factor, and the volumetric sweep efficiency at breakthrough and different amounts of PVI. Diffusive time of flight can be used instead of particle time of flight to generate similar measures. The fast marching method (FMM) described in Section 3.2 will return the diffusive time of flight to the individual grid cells. Combining this with the porosities and bulk volumes of each cell, it is possible to obtain what will be called the *diffusive sweep efficiency* at any given time. It will be defined as the ratio of the pore volume that the pressure front has reached at any given time. While sweep efficiency from regular time of flight is based on a steady-state flow regime, the diffusive time of flight is a more dynamic parameter, as the pressure front propagates throughout space as a wave, without ”seeing” what is in front of it. The pressure front is therefore also expected to show some sort of diffraction effects when e.g. passing through a narrow high permeability channel.

Sweep efficiency at a number of pore volumes injected is not a relevant parameter for diffusion time of flight, as it is a measure of pressure propagation, and not mass flux. Potential measures to investigate are the diffusive sweep efficiency at pressure front breakthrough, and at the time of pressure front arrival to the last observer well section/cell. The inverse of sweep efficiency at breakthrough is the equivalent of a diffusive Koval factor, which will be compared to the regular Koval factor.

3.5 Setup for Numerical Results

The numerical examples and results in this thesis have been conducted on the SPE10 model presented in Section 2.4. Each horizontal layer has been handled as an individual geological realization, resulting in 85 two-dimensional models with horizontal flow. The main goal has been to correlate simple single-phase experiments with more computationally demanding two-phase simulations.

3.5.1 Single-Phase Flow Experiments

The single-phase steady-state flow fields were found with the already existing MRST method *incompTPFA* (incompressible two-point flux approximation). The setup is shown in Figure 3.10. It is a line drive with constant pressure boundaries on the two shortest horizontal edges, and a constant pressure drop of 400 bars in the longest horizontal directions. The magnitude of the pressure drop is not important, as changing it will only scale the flow rate according to Darcy's law. The same applies to the fluid viscosity of $1cP$. The heterogeneity measures are all independent of the magnitude of the rate, and they are therefore not affected (as long as there is a pressure drop over the model). Fluid density has been set to $1014 {}^{kg}/_{m^3}$, but also this parameter is irrelevant, as we only consider horizontal flow, and there is therefore no gravity effect. The longest horizontal edges have been set to either no flow or periodic conditions.

Streamlines were then tracked with the extended Pollock algorithm called *pollockMod* (code in Appendix A.3), generating coordinates, time of flights, streamline lengths, velocities and pressure gradients. These are used as a basis for the effective permeability descriptors from Section 3.3 and the flow based heterogeneity measures (except the vorticity factor, which is from Krogstad et al. (2017)).

The same setup was used for diffusive time of flight. No steady-state solution is required for this method, though a total compressibility is needed. Similarly to flow, this is just a scaling factor for the velocity, and does not affect the heterogeneity measures as long as it is constant for the entire model. The method *computeDTOF* (code in Appendix A.4) supports

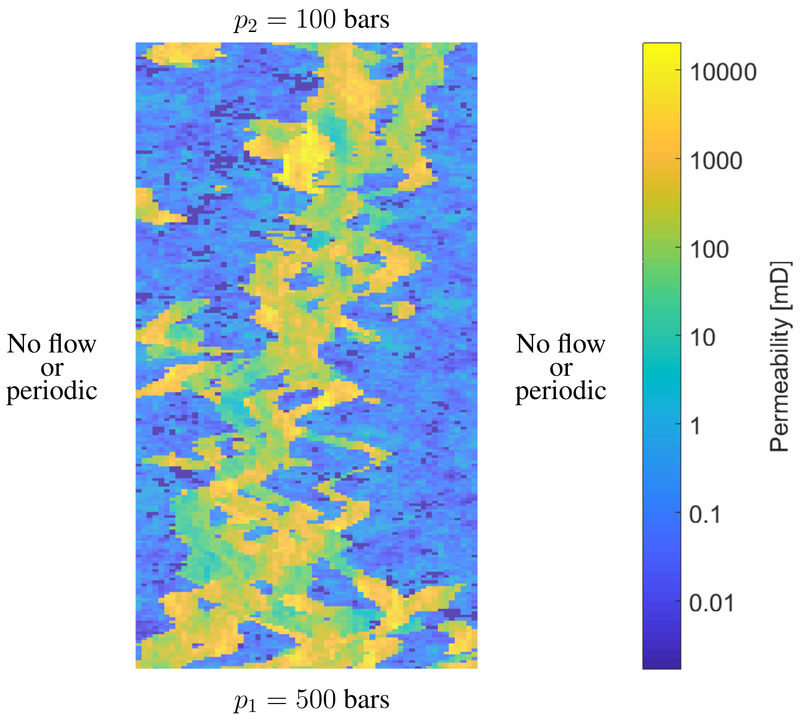


Figure 3.10: Setup for a single layer. The permeability field is from layer 50 (Upper Ness).

a single compressibility value for the whole model, or a vector of compressibilities for each cell. In this work a constant compressibility of $c_t = 4.4 * 10^{-5} bars^{-1}$ has been used.

An example setup that iterates through the individual layers and does the required calculations for the heterogeneity measures is shown in Appendix A.1.

3.5.2 Waterflooding and Tertiary Polymer Injection

The two-phase recovery data used to benchmark the heterogeneity measures are from Krogstad et al. (2017). They presented a proxy for polymer flooding, which in addition to recovery after waterflooding, and recovery increase due to polymer injection, also includes an estimation of the recovery increase due to improved macroscopic (volumetric) sweep efficiency from polymer flooding. Simulations have been conducted in Eclipse to verify that the results from Krogstad et al. (2017) for recovery at 90% water cut, and for recovery increase due to polymer injection, are reasonable. The main principles of the methodology will be briefly presented here to describe the physical meaning of the macro sweep recovery increase data used for comparison with heterogeneity measures in Section 5.4, but for the full description of the model one may read Krogstad et al. (2017).

Krogstad et al. (2017) uses an immiscible two-phase model which includes three fluid components: oil, water and polymer. The polymer can be dissolved in the water phase to increase its viscosity, and thereby improve the mobility ratio to oil. At first, only water is injected with a constant injection rate until the water cut reaches 90%. The recovery factor at this point is what will be used as recovery at 90% water cut later. From this point on, polymer is injected with the water at a constant concentration and rate, until a fixed amount of polymer has been injected. As the layers of the SPE10 model can have different pore volumes, the injected volume of polymer solution does not equal a fixed number of pore volumes injected, but according to Krogstad et al. (2017) the fixed amount of polymer solution corresponds to about 0.8 pore volumes for most layers. In addition, a second polymer injection is computed from the same starting point (water cut 90%), where the flow field has been locked to its state at the start of polymer injection. Thirdly, a regular waterflood is continued from the point where the water cut reaches 90%, until the same amount of water has been injected as for the two cases of polymer injection. The idea is then that the difference between the polymer injections with locked flow fields and the regular waterfloods will give the recovery increase due to improved sweep efficiency at the microscopic level. The recovery difference between the polymer injections with unlocked and locked flow fields should then represent the recovery increase due to increased macroscopic sweep efficiency, which they show is related to heterogeneity. Lastly, the total recovery increase due to polymer injection is the difference between the polymer injections with unlocked flow fields, and the regular waterfloods.

The setup used by Krogstad et al. (2017) is similar to Figure 3.10, only that they use no flow boundaries for all boundaries, and instead have a horizontal injector at one edge, and a horizontal producer at the other. The differences of the flow fields between using wells or constant pressure boundaries at the two shortest horizontal edges are minimal. The reason why constant pressure boundaries have been used for the heterogeneity measures, is that the current Pollock algorithm does not include handling of wells. This will also remove any potential near-well effects.

Chapter 4

Results

4.1 Streamlines Tracked With Pollock Algorithm

The extended Pollock algorithm in MRST has been used to create sets of streamlines for single-phase steady-state solutions of the horizontal layers of the SPE10 model. These streamlines have later been used for calculations of permeability parameters and heterogeneity measures. A few example layers are shown here to have an idea of the structure of the flow fields of both the Tarbert and Upper Ness formations, and the difference between no flow boundaries and periodic boundary conditions. These conditions are only applied to the longest horizontal direction, while the boundaries in the shortest horizontal direction are defined as constant pressure boundaries, creating a line drive from the bottom of each figure to the top. Only one streamline is used per inlet cell to make sure that the permeability field behind is visible. Each streamline is just a visualization of the direction of the flow field, and not the flow velocity. However, regions with short spacing between the streamlines are likely to be regions with higher flux than their surroundings. Figure 4.1 show the streamlines for a Tarbert layer with no flow boundaries to the left and periodic boundaries to the right. We can observe that for the no flow model the streamlines near the left and right side of the model straightens out, while they are more tortuous in the periodic model where they are allowed to cross the boundary. The same effect can be observed in layer 68 from the Upper Ness formation in Figure 4.2. We can also observe, especially from the channelized Upper Ness model in Figure 4.2, that the streamlines gather even more in the high permeability regions in the periodic model than in the no flow case. Comparing Figure 4.1 and Figure 4.2 it is also clear that the streamline spacing is more uniform for the Tarbert layer in Figure 4.1.

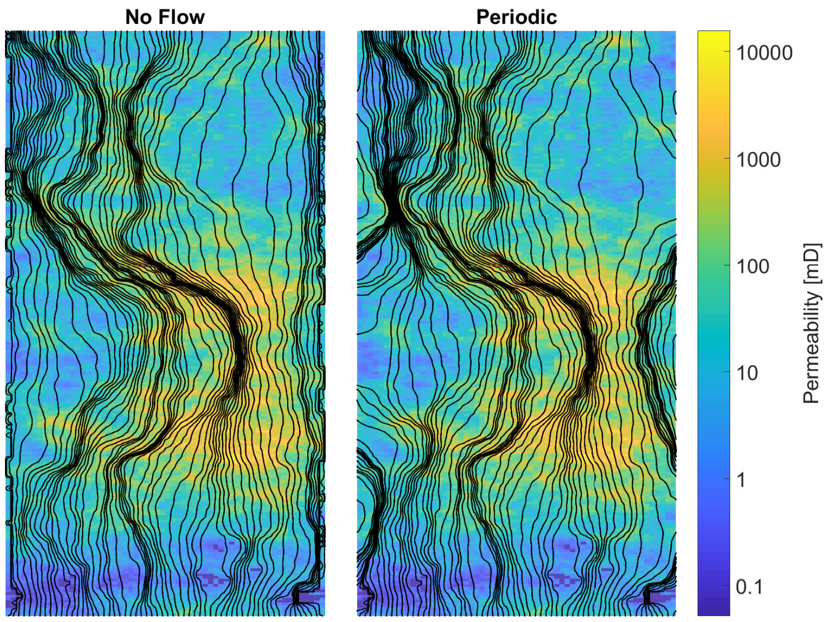


Figure 4.1: Streamlines in a Tarbert layer (layer 21) for no flow and periodic boundaries.

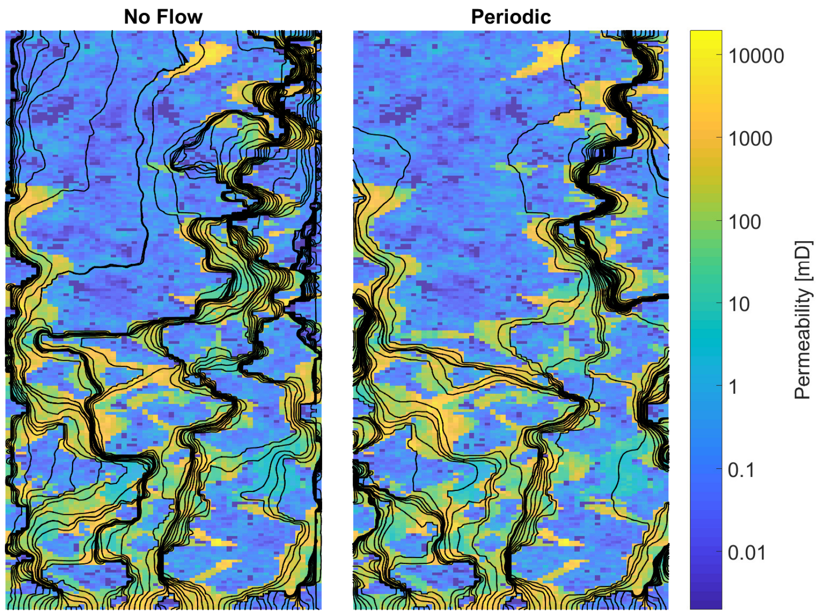


Figure 4.2: Streamlines in an Upper Ness layer (layer 68) for no flow and periodic boundaries.

4.2 Particle and Diffusive Time of Flight

Figure 4.3 compares the "regular" flow based particle time of flight (τ_t) with the diffusive time of flight (τ_d) for layer 21 in the SPE10 model, the same as in Figure 4.1. The colorbar is defined from "low" to "high" time of flight / diffusive time of flight instead of numerical values, as the diffusive time of flight for a heterogeneous medium cannot generally be converted to actual time, as discussed in Section 2.2.2. In the same section we also saw that the diffusive time of flight has the dimensions $\sqrt{\text{time}}$, while the particle time of flight has dimensions of time. The diffusive time of flight has therefore been squared before plotting to improve the comparison. The particle time of flight has been calculated using the existing *computeTimeOfFlight* method in MRST, which is an upwind finite-volume discretization of Equation (2.18) (See Lie, 2016, Section 5.3). It is therefore not based on the streamline time of flight from the Pollock algorithm. The advantage of the finite-volume method is that it gives a value for all cells, while a large number of streamlines could be necessary to ensure that all cells are reached with the Pollock algorithm. The *computeTimeOfFlight* method is therefore useful for visualization purposes. The diffusive time of flight is found with the new method *computeDTOF*, which is an implementation of the fast marching method presented in Section 3.2. Both examples are line drive bottom to top with no flow boundaries at the longest sides. The particle time of flight is cut off at 5 times the mean time of flight to improve the visualization. It can be observed that the low permeability region at the left side of the model as seen in Figure 4.1 has a greater influence on the particle time of flight, compared to the diffusive time of flight. Additionally the far right corner has a stalling point due to inactive cells which we can see is quickly reached by the pressure front, while the flow rate is infinitesimal, and the particle time of flight is therefore high.

4.3 Uncertainty Analysis

Before the solvers of streamline parameters and diffusive time of flight can be applied to generate heterogeneity measures, it is important to quantify to which extent the solvers are reliable. This section will therefore cover the results from an uncertainty analysis of the implementations.

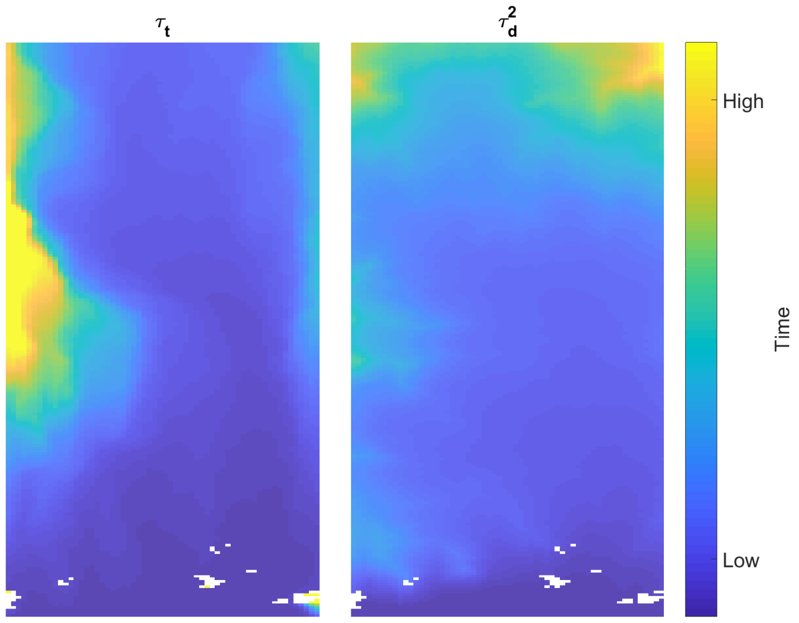


Figure 4.3: Comparison of particle time of flight (τ_t) to the left, and diffusive time of flight squared (τ_d^2) to the right. Particle time of flight cutoff at 5 times mean time of flight.

4.3.1 Expanded Pollock Algorithm

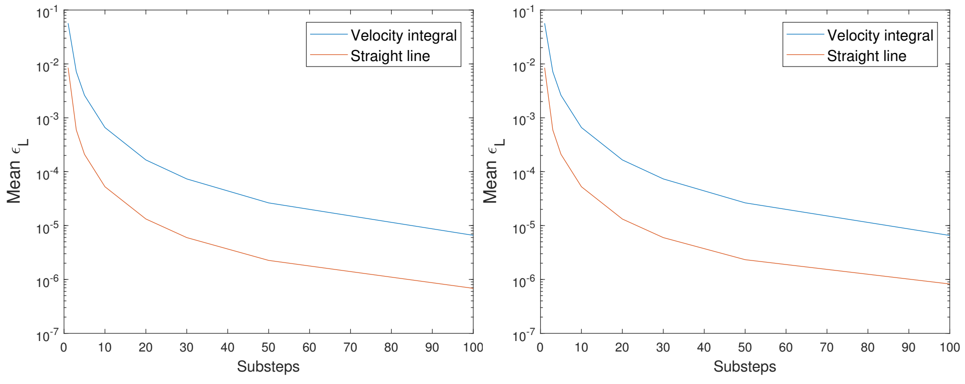
4.3.1.1 Streamline Length

The new MRST implementation of the Pollock algorithm in Section 3.1.2 includes two options for streamline length estimation, the shortest path between each coordinate and a trapezoidal numerical integration of velocity. In addition to these two options, the length estimation is also dependent on the number of substeps in each cell. Increasing the number of substeps will improve the length estimation, but we want to keep the number of substeps as low as possible to decrease computational cost. To compare the methods, and the effect of substeps, a test of 1000 substeps will be used as a proxy for the true length, with 1 streamline starting at each inlet cell. The length error estimate ϵ_L for n substeps is therefore defined as

$$\epsilon_L = \frac{|L_n - L_{1000}|}{L_{1000}} \quad , \tag{4.1}$$

where L_n is the length of a streamline using n substeps and L_{1000} is the length of the same streamline using 1000 substeps.

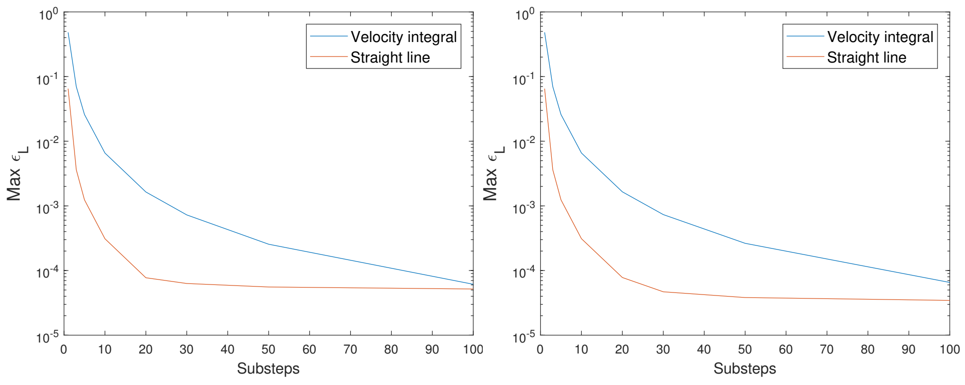
Figure 4.4 shows the mean relative error $\overline{\epsilon_L}$. In case 4.4a the straight line length with 1000 substeps is used as L_{1000}, while the velocity integral length is used in case 4.4b. We can observe that the straight line method is superior with both reference points, but both methods converge towards the right solution. The mean error is less than 1% already at 1 substep for the straight line method, and 3 substeps for the velocity integral method.



(a) Reference: Straight line **(b)** Reference: Velocity integral

Figure 4.4: Mean relative error $\overline{\epsilon_L}$ compared to L_{1000}

In addition to the mean, it is also interesting to look at the worst case scenario. The two subfigures in Figure 4.5 show the maximum relative error $\epsilon_{L,\text{max}}$. We can observe that both methods continuously improve when increasing the number of substeps, but the velocity integral method is again inferior to the straight line.



(a) Reference: Straight line **(b)** Reference: Velocity integral

Figure 4.5: Max relative error $\epsilon_{L,\text{max}}$ compared to L_{1000}

4.3.1.2 Pressure Gradient Method

Section 3.1.2.3 presents three methods of pressure gradient estimation:

- Permeability weighted linear interpolation of pressure between two neighboring cells
- Permeability weighted trilinear interpolation of pressure between all cell centers surrounding a point
- 1D Darcy's law along streamlines

As we saw from Equation (3.40), the effective hydraulic conductance B_e should equal the volumetric average of the permeability tangential to the flow field. B_e is dependent on the precision of the pressure gradient since $B(\mathcal{S})$ is, as seen from Equation (3.34). As the SPE10 layers are horizontally isotropic, and all examples in this thesis are for single layer flow, the error estimate used to quantify the influence of various parameters will be

$$\epsilon_{B_e} = \frac{|B_e - \overline{k_h}|}{\overline{k_h}} \quad , \tag{4.2}$$

where $\overline{k_h}$ is the volumetric average of permeability in the volume contributing to flow. This value is found by a setting a lower limit of flux through each cell which is positive and slightly larger than the machine error. Cells that in theory should have no flow, but due to the errors introduced through the numerical flow solver get very small rates, are thus excluded. From the definition of $B(\mathcal{S})$ and B_e in Equations (3.34) and (3.39) we can see that the effective hydraulic conductivity is a function of multiple variables, including the streamtube pressure gradient, flow rate, path and length, the active bulk volume V_e and the fluid viscosity which is assumed constant. The error estimate in Equation (4.2) is therefore a measure of the total error of all these contributions, some of which may work in different directions. It will be assumed that the relative change of ϵ_{B_e} when changing pressure gradient method will be an adequate measure to compare the methods. 5 substeps per cell and 5 streamlines per inlet cell are used in all cases, with the straight line method used for streamline step lengths. The linear interpolation method does not take the position at the exit face into account, only the distance from the cell center. It is therefore prone to inaccuracies if the travel distance internally in a cell is short. A minimum step length parameter is therefore included, and the effect of this parameter can be seen in Figure 4.6. As the lowest mean error is for minimum step length = 1.5 m (approximately half of the shortest horizontal grid dimension), this value is used for the comparison with the other methods.

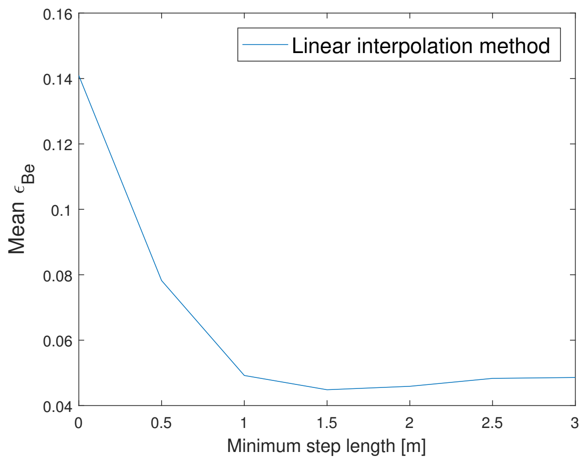


Figure 4.6: Mean relative error $\overline{\epsilon_{B_e}}$ of effective hydraulic conductance as a function of minimum step length, for linear interpolation method.

The three different pressure gradient methods are compared in Figure 4.7. We can observe that especially for the Tarbert layers (layer 1 to 35) in the SPE10 model, the 1D Darcy method is significantly better than the other two. The trilinear pressure interpolation gives the least accurate results, especially in the channelized Upper Ness formation (layer 36 to 85). Key numbers are summarized in Table 4.1. Based on these results, the 1D Darcy method is used for the rest of this thesis.

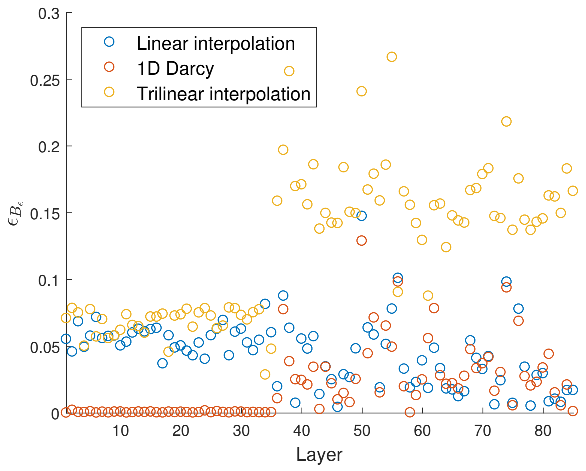


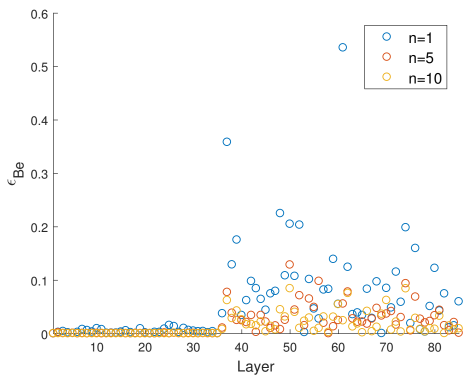
Figure 4.7: Layer ϵ_{B_e} for each pressure gradient method.

Table 4.1: Key Data for Pressure Method Comparison

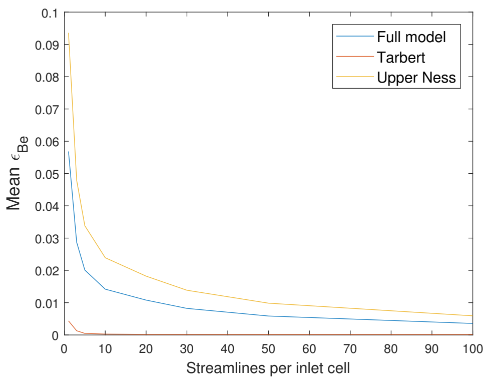
	Linear Interpolation		1D Darcy		Trilinear Interpolation	
Formation	$\overline{\epsilon_{B_e}}$	$\max \epsilon_{B_e}$	$\overline{\epsilon_{B_e}}$	$\max \epsilon_{B_e}$	$\overline{\epsilon_{B_e}}$	$\max \epsilon_{B_e}$
Tarbert	0.056	0.081	4.8e-4	0.002	0.067	0.079
Upper Ness	0.037	0.147	0.034	0.129	0.162	0.267
Full model	0.045	0.147	0.020	0.129	0.123	0.267

4.3.1.3 Number of Streamlines

Increasing the streamline density will decrease the streamtube cross-sectional area, and therefore get us closer to the infinitesimal streamtube in the analytic solution. Hence, increasing streamline density should improve results, but also increase the computational expense. It is therefore of interest to investigate the achievable improvement from increasing the number of streamlines. The error of effective hydraulic conductance from Equation (4.2) will again be used as an error estimator, as B_e is a rate weighted value of $B(\mathcal{S})$ for the set of streamlines. Figure 4.8a shows ϵ_{B_e} for each layer of the SPE10 model for $n = 1$, $n = 5$ and $n = 10$ streamlines started in each inlet cell. Figure 4.8b shows the mean error $\overline{\epsilon_{B_e}}$ as a function of streamlines per inlet cell for the full SPE10 model, for the Tarbert formation alone (layer 1-35), and for the Upper Ness formation alone (layer 36-85). 5 substeps per cell have been used for both figures. We can observe from both Figure 4.8a and Figure 4.8b that the error is decreasing with increased streamline density as expected, with the most rapid decline for few streamlines. The observed error of the Tarbert layers are significantly less than for the channelized Upper Ness layers, and for practical purposes negligible.



(a) Layer ϵ_{B_e} for $n = 1$, $n = 5$ and $n = 10$ streamlines per inlet cell



(b) Mean error $\overline{\epsilon_{B_e}}$ as a function of the number of streamlines per inlet cell

Figure 4.8: Relative error of hydraulic conductance B_e as a function of the number of streamlines.

Some of the heterogeneity measures presented in Section 2.3 and Section 3.4 are based on streamline time of flight and the flow rate allocated to each streamtube, where a key assumption is that the streamline time of flight is representable for the whole streamtube. Reducing the number of streamlines means that each streamline time of flight has to be representable for a larger flow rate, and the sensitivity of the heterogeneity measures should therefore also be considered before choosing a specific streamline density. The streamline time of flight is independent of the number of substeps in each cell in the MRST implementation, and the number of substeps will therefore not contribute to variations in the heterogeneity measures. The dynamic Lorenz coefficient L_C presented in Section 2.3.2.1 is only a function of the time of flight of each streamline and the allocated rates to the streamtubes surrounding each streamline. One would therefore expect that L_C is a reliable parameter to use for this sensitivity study. Assuming that L_C for $n = 100$ streamlines per inlet cell is close to the true L_C we have a proxy for the relative error for n streamlines per inlet cell given as

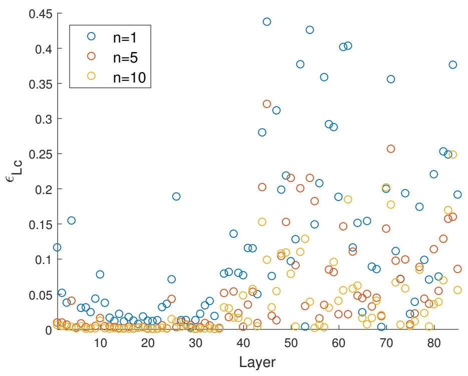
$$\epsilon_{L_C} = \frac{|L_C(n) - L_C(n = 100)|}{L_C(n = 100)} \quad . \tag{4.3}$$

Figure 4.9a shows the relative error ϵ_{L_C} for each layer of the SPE10 model for $n = 1$, $n = 5$ and $n = 10$ streamlines per inlet cell using n=100 as the ”true” value. Similar to ϵ_{B_e} in Figure 4.8a we can observe that the errors are generally higher in the Upper Ness formation (layer 36-85), and that they decrease with the number of streamlines. Figure 4.9b shows the mean error for the full model and the two formations individually, and again we see the trend from Figure 4.8b where the Upper Ness formation is the main contributor to errors.

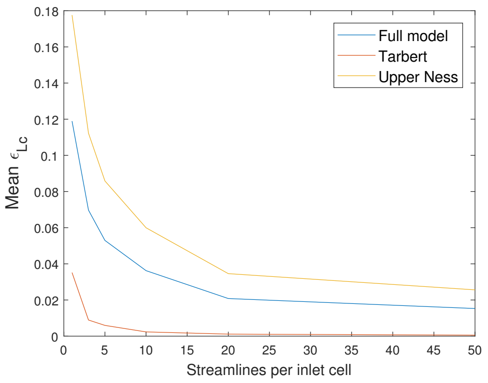
4.3.2 Fast Marching Method for Diffusive Time of Flight

As discussed in Section 2.2.2 there are is an analytic solution to the relation between diffusive time of flight in transient flow for a homogeneous and isotropic medium. This is not the case for the SPE10 model, so a simpler model will be used to verify the results. Using a horizontal homogeneous and isotropic layer, we can compare the results from the analytic solution in Equation (2.25) with the diffusive time of flight from the FMM, converted to actual time of flight with the relation in Equation 2.41. The relative error is thus

$$\epsilon_{\text{DTOF}} = \frac{|t_{\text{DTOF}} - t_{\text{Analytic}}|}{t_{\text{Analytic}}} \quad , \tag{4.4}$$



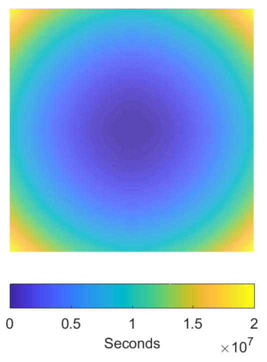
(a) Layer ϵ_{L_C} for $n = 1$, $n = 5$ and $n = 10$ streamlines per inlet cell.



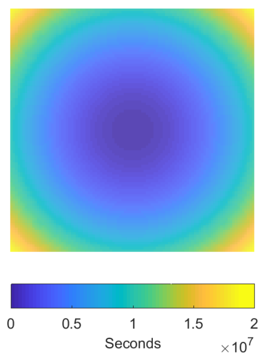
(b) Mean error $\overline{\epsilon_{L_C}}$ as a function of the number of streamlines per inlet cell.

Figure 4.9: Relative error of Lorenz coefficient L_C as a function of the number of streamlines, relative to L_C when using 100 streamlines per inlet cell.

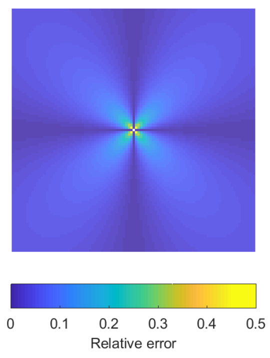
where t_{DTOF} and t_{Analytic} are the times from Equation (2.41) and 2.25, respectively. Figure 4.10 shows a comparison of t_{DTOF} and t_{Analytic} for a radial pressure front in a homogeneous and isotropic model. The model is 101m x 101m x 1m with 10201 uniform cells of 1m x 1m x 1m. All cells have a permeability of 1 Darcy, 30% porosity, 1cP fluid viscosity and a total compressibility of 5e-5 Pa^{-1}. From the relative errors ϵ_{DTOF} in Subfigure 4.10c we can observe that the relative errors are largest on the diagonals, and decrease with distance from the origin. The mean relative error ($\overline{\epsilon_{\text{DTOF}}}$) is 0.044.



(a) t_{DTOF}



(b) t_{Analytic}



(c) ϵ_{DTOF}

Figure 4.10: Comparison of radial diffusive time of flight and analytic solution in a homogeneous and isotropic model.

4.4 Permeability description Using Streamlines

The methodology from Section 3.3 for permeability description by the use of streamlines has been applied on the 85 layers of the SPE10 model for both no flow and periodic boundary conditions. The results are generated using the Pollock algorithm with 20 streamlines per inlet cell and 10 substeps per cell. Table 4.2 contains the final results for both straight line and velocity integrated streamline lengths for the no flow boundaries, and for the velocity integrated lengths with periodic boundary conditions. 1D Darcy approximation along streamlines are used for the pressure gradients, as it has proven to be the most of precise of the evaluated methods. The steady-state flow field is found with the MRST method *incompTPFA*. The labels of each column in Table 4.2 are defined as:

- k_D: Layer permeability calculated with 1D Darcy's law over the full model, as in Equation (3.29). (Units: m^2)
- k_S: Layer permeability calculated from the streamline permeability descriptors as in Equation (3.38). (Units: m^2)
- B_e: Effective hydraulic conductance. Equal to the volumetric permeability average tangential to the flow field. (Units: m^2)
- C_e: Inverse of effective constriction factor C_e. (Unitless)
- τ_e^2: Effective tortuosity factor. (Unitless)

We can observe that the differences between straight and integral line length methods for no flow boundaries are less than what can be expressed with two significant figures, as the tables are equal. The largest relative difference between these two sub-tables in the original data set was 0.003. The difference between no flow boundaries and periodic boundaries are though more significant.

4.5 Heterogeneity Measures

Various existing and potential heterogeneity measures were presented in Section 2.3 and Section 3.4. Table 4.3 shows summarizes results for several heterogeneity measures using the no flow boundary condition and 1D darcy approximation for pressure gradients. This means that they correspond to column 2 to 6 in Table 4.2. The measures are separated into three groups: one with existing measures, a second with potential new measures based on the proposed streamline based permeability descriptors, and a third with new potential measures based on diffusive time of flight. The labels of each column in Table 4.3 are defined as:

- H_V: Vorticity factor. These results are from Krogstad et al. (2017)
- L_C: Dynamic Lorenz coefficient
- FHI: Flow heterogeneity index
- H_K: Koval factor (Both from particle and diffusive time of flight)
- $E_{V,b}$: Volumetric sweep efficiency at breakthrough (Both from particle and diffusive time of flight)
- $C_V(\tau_t)$: Coefficient of variation of streamline time of flight (rate weighted)
- $E_{V,1PVI}$: Volumetric sweep efficiency at 1 pore volume injected
- $E_{V,2PVI}$: Volumetric sweep efficiency at 2 pore volumes injected
- $\sigma^2_{\tau^2}$: Variance of tortuosity factor
- $C_V(\tau^2)$: Coefficient of variation for the tortuosity factor
- $\sigma^2_{1/C}$: Variance of the inverse of the constriction factor
- $C_V(1/C)$: Coefficient of variation of the inverse of the constriction factor

All the measures are unitless. Some layers are missing the vorticity factors from Krogstad et al. (2017), as the two-phase simulations they used to correlate the heterogeneity measure with did not converge for all layers, and they have therefore not reported the vorticity factor for these layers.

Table 4.2: Streamline Based Permeability Parameters

	No Flow Boundaries, Straight Line Length					No Flow Boundaries, Integral Line Length					Periodic Boundaries, Integral Line Length				
SPE10 Layer	k_D $[m^2]$	$k_\mathcal{S}$ $[m^2]$	B_e $[m^2]$	$^1/_{C_e}$	τ_e^2	k_D $[m^2]$	$k_\mathcal{S}$ $[m^2]$	B_e $[m^2]$	$^1/_{C_e}$	τ_e^2	k_D $[m^2]$	$k_\mathcal{S}$ $[m^2]$	Be $[m^2]$	$^1/_{C_e}$	τ_e^2
1	1.3E-15	1.3E-15	7.4E-14	2.5E-02	7.1E-01	1.3E-15	1.3E-15	7.4E-14	2.5E-02	7.1E-01	1.4E-15	1.5E-15	7.4E-14	2.7E-02	7.2E-01
2	5.1E-15	5.1E-15	2.0E-13	3.4E-02	7.8E-01	5.1E-15	5.1E-15	2.0E-13	3.4E-02	7.8E-01	5.3E-15	5.4E-15	2.0E-13	3.6E-02	7.7E-01
3	5.9E-15	6.0E-15	3.3E-13	2.7E-02	6.9E-01	5.9E-15	6.0E-15	3.3E-13	2.7E-02	6.9E-01	6.1E-15	6.2E-15	3.3E-13	2.8E-02	7.0E-01
4	1.3E-14	1.4E-14	1.2E-12	1.6E-02	7.1E-01	1.3E-14	1.4E-14	1.2E-12	1.6E-02	7.1E-01	1.4E-14	1.4E-14	1.2E-12	1.6E-02	7.5E-01
5	2.6E-15	2.7E-15	2.6E-13	1.5E-02	7.6E-01	2.6E-15	2.7E-15	2.6E-13	1.5E-02	7.6E-01	3.2E-15	3.3E-15	2.6E-13	1.8E-02	7.4E-01
6	1.0E-14	1.0E-14	2.9E-13	5.0E-02	7.2E-01	1.0E-14	1.0E-14	2.9E-13	5.0E-02	7.2E-01	1.1E-14	1.1E-14	2.9E-13	5.5E-02	7.3E-01
7	1.0E-14	1.0E-14	4.5E-13	2.9E-02	8.2E-01	1.0E-14	1.0E-14	4.5E-13	2.9E-02	8.2E-01	1.0E-14	1.1E-14	4.5E-13	3.0E-02	8.2E-01
8	1.8E-14	1.8E-14	4.1E-13	5.4E-02	7.9E-01	1.8E-14	1.8E-14	4.1E-13	5.4E-02	7.9E-01	1.8E-14	1.8E-14	4.1E-13	5.5E-02	7.9E-01
9	2.3E-14	2.4E-14	8.8E-13	3.3E-02	8.1E-01	2.3E-14	2.4E-14	8.8E-13	3.3E-02	8.1E-01	2.4E-14	2.5E-14	8.8E-13	3.5E-02	8.2E-01
10	9.9E-15	1.0E-14	5.4E-13	2.5E-02	7.5E-01	9.9E-15	1.0E-14	5.4E-13	2.5E-02	7.5E-01	1.1E-14	1.1E-14	5.4E-13	2.8E-02	7.4E-01
11	1.4E-14	1.4E-14	3.2E-13	6.0E-02	7.3E-01	1.4E-14	1.4E-14	3.2E-13	6.0E-02	7.3E-01	1.4E-14	1.5E-14	3.2E-13	6.4E-02	7.2E-01
12	1.2E-14	1.2E-14	3.8E-13	4.2E-02	7.7E-01	1.2E-14	1.2E-14	3.8E-13	4.2E-02	7.7E-01	1.2E-14	1.2E-14	3.8E-13	4.3E-02	7.7E-01
13	3.1E-14	3.2E-14	6.0E-13	7.0E-02	7.5E-01	3.1E-14	3.2E-14	6.0E-13	7.0E-02	7.5E-01	3.3E-14	3.3E-14	6.0E-13	7.3E-02	7.5E-01
14	1.8E-14	1.8E-14	6.4E-13	3.9E-02	7.4E-01	1.8E-14	1.8E-14	6.4E-13	3.9E-02	7.4E-01	2.0E-14	2.0E-14	6.4E-13	4.2E-02	7.4E-01
15	4.4E-14	4.4E-14	3.9E-13	1.6E-01	7.2E-01	4.4E-14	4.4E-14	3.9E-13	1.6E-01	7.2E-01	4.7E-14	4.8E-14	3.9E-13	1.7E-01	7.4E-01
16	2.6E-14	2.7E-14	2.6E-13	1.3E-01	7.7E-01	2.6E-14	2.7E-14	2.6E-13	1.3E-01	7.7E-01	2.7E-14	2.7E-14	2.6E-13	1.4E-01	7.7E-01
17	2.5E-14	2.6E-14	4.1E-13	8.0E-02	7.9E-01	2.5E-14	2.6E-14	4.1E-13	8.0E-02	7.9E-01	2.7E-14	2.7E-14	4.1E-13	8.3E-02	8.1E-01
18	3.1E-14	3.1E-14	9.0E-13	4.7E-02	7.3E-01	3.1E-14	3.1E-14	9.0E-13	4.7E-02	7.3E-01	3.2E-14	3.3E-14	9.0E-13	5.0E-02	7.3E-01
19	2.0E-14	2.0E-14	3.4E-13	8.3E-02	7.2E-01	2.0E-14	2.0E-14	3.4E-13	8.3E-02	7.2E-01	2.3E-14	2.3E-14	3.4E-13	1.1E-01	6.6E-01
20	2.4E-14	2.4E-14	2.3E-13	1.3E-01	8.1E-01	2.4E-14	2.4E-14	2.3E-13	1.3E-01	8.1E-01	2.4E-14	2.4E-14	2.3E-13	1.4E-01	7.9E-01
21	1.2E-14	1.3E-14	2.4E-13	6.9E-02	7.7E-01	1.2E-14	1.3E-14	2.4E-13	6.9E-02	7.7E-01	1.3E-14	1.3E-14	2.4E-13	6.9E-02	7.8E-01
22	1.5E-14	1.5E-14	4.2E-13	4.4E-02	8.1E-01	1.5E-14	1.5E-14	4.2E-13	4.4E-02	8.1E-01	1.6E-14	1.7E-14	4.2E-13	4.8E-02	8.3E-01
23	1.9E-14	1.9E-14	2.3E-13	1.1E-01	8.0E-01	1.9E-14	1.9E-14	2.3E-13	1.1E-01	8.0E-01	2.0E-14	2.0E-14	2.3E-13	1.1E-01	8.0E-01
24	8.7E-15	8.8E-15	9.2E-14	1.1E-01	8.3E-01	8.7E-15	8.8E-15	9.2E-14	1.1E-01	8.3E-01	8.9E-15	9.0E-15	9.2E-14	1.2E-01	8.3E-01
25	5.5E-15	5.5E-15	1.2E-13	6.5E-02	7.5E-01	5.5E-15	5.5E-15	1.2E-13	6.5E-02	7.5E-01	6.6E-15	6.6E-15	1.2E-13	8.1E-02	7.2E-01
26	5.7E-15	5.8E-15	1.9E-13	4.0E-02	7.6E-01	5.7E-15	5.8E-15	1.9E-13	4.0E-02	7.6E-01	6.4E-15	6.5E-15	1.9E-13	4.7E-02	7.4E-01
27	4.1E-15	4.2E-15	2.7E-13	2.1E-02	7.6E-01	4.1E-15	4.2E-15	2.7E-13	2.1E-02	7.6E-01	4.9E-15	5.0E-15	2.7E-13	2.5E-02	7.4E-01
28	1.9E-15	1.9E-15	7.3E-14	3.4E-02	7.9E-01	1.9E-15	1.9E-15	7.3E-14	3.4E-02	7.9E-01	2.2E-15	2.2E-15	7.3E-14	4.0E-02	7.7E-01
29	2.8E-15	2.8E-15	1.0E-13	3.4E-02	8.0E-01	2.8E-15	2.8E-15	1.0E-13	3.4E-02	8.0E-01	3.7E-15	3.8E-15	1.0E-13	4.8E-02	7.6E-01
30	6.0E-15	6.1E-15	8.9E-14	9.7E-02	7.1E-01	6.0E-15	6.1E-15	8.9E-14	9.7E-02	7.1E-01	1.2E-14	1.2E-14	8.9E-14	1.8E-01	7.3E-01
31	1.0E-14	1.1E-14	1.5E-13	9.7E-02	7.1E-01	1.0E-14	1.1E-14	1.5E-13	9.7E-02	7.1E-01	1.1E-14	1.1E-14	1.5E-13	9.9E-02	7.4E-01
32	9.4E-15	9.5E-15	1.1E-13	1.2E-01	7.6E-01	9.4E-15	9.5E-15	1.1E-13	1.2E-01	7.6E-01	1.2E-14	1.2E-14	1.1E-13	1.5E-01	7.9E-01
33	9.2E-15	9.3E-15	1.9E-13	6.6E-02	7.6E-01	9.2E-15	9.3E-15	1.9E-13	6.6E-02	7.6E-01	9.7E-15	9.8E-15	1.9E-13	6.8E-02	7.7E-01
34	1.8E-14	1.8E-14	7.7E-13	2.9E-02	7.9E-01	1.8E-14	1.8E-14	7.7E-13	2.9E-02	7.9E-01	1.9E-14	1.9E-14	7.7E-13	3.0E-02	8.0E-01
35	5.5E-15	5.5E-15	3.3E-13	2.4E-02	7.2E-01	5.5E-15	5.5E-15	3.3E-13	2.4E-02	7.2E-01	7.3E-15	7.4E-15	3.3E-13	3.0E-02	7.5E-01
36	1.7E-16	1.8E-16	5.0E-14	8.5E-03	4.5E-01	1.7E-16	1.8E-16	5.0E-14	8.5E-03	4.5E-01	2.0E-16	2.1E-16	5.1E-14	1.2E-02	3.8E-01
37	3.3E-15	3.5E-15	3.0E-13	3.6E-02	3.4E-01	3.3E-15	3.5E-15	3.0E-13	3.6E-02	3.4E-01	3.3E-15	3.6E-15	2.6E-13	4.5E-02	3.3E-01
38	1.7E-14	1.8E-14	3.7E-13	1.1E-01	4.5E-01	1.7E-14	1.8E-14	3.7E-13	1.1E-01	4.5E-01	1.7E-14	1.8E-14	3.7E-13	1.2E-01	4.5E-01
39	1.7E-14	1.9E-14	2.1E-13	2.0E-01	4.7E-01	1.7E-14	1.9E-14	2.1E-13	2.0E-01	4.6E-01	1.7E-14	1.9E-14	2.1E-13	2.1E-01	4.6E-01
40	2.0E-14	2.2E-14	3.3E-13	1.7E-01	4.1E-01	2.0E-14	2.2E-14	3.3E-13	1.7E-01	4.0E-01	2.0E-14	2.2E-14	3.2E-13	1.8E-01	3.9E-01
41	5.5E-15	6.0E-15	1.9E-13	8.6E-02	4.0E-01	5.5E-15	6.0E-15	1.9E-13	8.6E-02	4.0E-01	5.7E-15	6.2E-15	1.9E-13	8.7E-02	4.0E-01
42	6.0E-16	6.2E-16	2.2E-13	6.4E-03	4.8E-01	6.0E-16	6.2E-16	2.2E-13	6.4E-03	4.8E-01	6.6E-16	6.9E-16	2.2E-13	7.2E-03	4.6E-01
43	1.0E-14	1.1E-14	2.6E-13	8.8E-02	5.2E-01	1.0E-14	1.1E-14	2.6E-13	8.8E-02	5.2E-01	1.1E-14	1.2E-14	2.5E-13	1.2E-01	4.2E-01
44	2.7E-14	2.9E-14	3.8E-13	1.4E-01	5.8E-01	2.7E-14	2.9E-14	3.8E-13	1.4E-01	5.8E-01	3.5E-14	3.7E-14	3.7E-13	1.8E-01	5.7E-01
45	3.8E-14	4.0E-14	3.1E-13	2.2E-01	6.1E-01	3.8E-14	4.0E-14	3.1E-13	2.2E-01	6.1E-01	4.9E-14	5.2E-14	3.1E-13	3.0E-01	5.9E-01
46	1.8E-14	1.9E-14	2.9E-13	1.3E-01	5.3E-01	1.8E-14	1.9E-14	2.9E-13	1.3E-01	5.3E-01	3.0E-14	3.1E-14	2.9E-13	2.1E-01	5.4E-01
47	6.2E-16	6.5E-16	3.0E-13	5.9E-03	3.9E-01	6.2E-16	6.5E-16	3.0E-13	5.9E-03	3.9E-01	1.2E-15	1.2E-15	3.0E-13	1.3E-02	3.3E-01
48	1.0E-14	1.1E-14	2.3E-13	1.3E-01	3.9E-01	1.0E-14	1.1E-14	2.3E-13	1.3E-01	3.9E-01	1.0E-14	1.1E-14	2.0E-13	1.5E-01	3.8E-01
49	1.8E-14	2.0E-14	2.3E-13	1.9E-01	4.8E-01	1.8E-14	2.0E-14	2.3E-13	1.9E-01	4.8E-01	1.8E-14	2.0E-14	2.2E-13	1.9E-01	4.8E-01
50	2.5E-14	2.6E-14	3.5E-13	1.5E-01	5.3E-01	2.5E-14	2.6E-14	3.6E-13	1.5E-01	5.3E-01	2.5E-14	2.6E-14	3.6E-13	1.5E-01	5.0E-01
51	2.9E-14	3.0E-14	4.1E-13	1.6E-01	4.7E-01	2.9E-14	3.0E-14	4.1E-13	1.6E-01	4.7E-01	2.9E-14	3.1E-14	4.2E-13	1.6E-01	4.7E-01
52	2.9E-14	3.1E-14	4.1E-13	1.6E-01	4.9E-01	2.9E-14	3.1E-14	4.1E-13	1.6E-01	4.9E-01	2.9E-14	3.1E-14	3.9E-13	1.7E-01	4.9E-01
53	5.6E-14	5.9E-14	4.3E-13	2.4E-01	5.9E-01	5.6E-14	5.9E-14	4.3E-13	2.4E-01	5.9E-01	5.7E-14	6.0E-14	4.3E-13	2.4E-01	5.9E-01
54	4.6E-14	4.9E-14	4.0E-13	2.0E-01	6.0E-01	4.6E-14	4.9E-14	4.0E-13	2.0E-01	6.0E-01	5.1E-14	5.3E-14	4.0E-13	2.2E-01	6.3E-01
55	5.7E-14	6.0E-14	3.7E-13	2.9E-01	5.7E-01	5.7E-14	6.0E-14	3.7E-13	2.9E-01	5.7E-01	6.4E-14	6.7E-14	3.7E-13	3.1E-01	6.0E-01
56	4.9E-14	5.2E-14	4.1E-13	2.1E-01	6.0E-01	4.9E-14	5.2E-14	4.1E-13	2.1E-01	6.0E-01	5.6E-14	5.9E-14	4.1E-13	2.4E-01	6.1E-01
57	6.3E-14	6.6E-14	3.7E-13	2.9E-01	6.3E-01	6.3E-14	6.6E-14	3.7E-13	2.9E-01	6.3E-01	6.6E-14	6.9E-14	3.8E-13	2.9E-01	6.3E-01
58	6.4E-14	6.8E-14	4.5E-13	2.7E-01	5.7E-01	6.4E-14	6.8E-14	4.5E-13	2.7E-01	5.7E-01	6.7E-14	7.1E-14	4.5E-13	2.8E-01	5.9E-01
59	5.7E-14	6.0E-14	4.0E-13	2.6E-01	5.9E-01	5.7E-14	6.0E-14	4.0E-13	2.6E-01	5.9E-01	5.9E-14	6.2E-14	4.0E-13	2.7E-01	5.7E-01
60	3.8E-14	4.0E-14	4.1E-13	1.9E-01	5.3E-01	3.8E-14	4.0E-14	4.1E-13	1.9E-01	5.3E-01	4.3E-14	4.6E-14	4.1E-13	2.2E-01	5.2E-01
61	3.4E-14	3.6E-14	3.7E-13	2.0E-01	4.9E-01	3.4E-14	3.6E-14	3.7E-13	2.0E-01	4.9E-01	3.7E-14	3.9E-14	3.7E-13	2.2E-01	5.0E-01
62	1.7E-14	1.8E-14	2.9E-13	1.6E-01	4.0E-01	1.7E-14	1.8E-14	2.9E-13	1.6E-01	4.0E-01	1.7E-14	1.9E-14	2.9E-13	1.7E-01	3.9E-01
63	2.2E-14	2.4E-14	2.8E-13	1.9E-01	4.7E-01	2.2E-14	2.4E-14	2.8E-13	1.9E-01	4.7E-01	2.2E-14	2.4E-14	2.8E-13	1.9E-01	4.6E-01
64	4.2E-14	4.4E-14	5.5E-13	1.4E-01	5.9E-01	4.2E-14	4.4E-14	5.5E-13	1.4E-01	5.9E-01	4.2E-14	4.5E-14	5.5E-13	1.5E-01	5.7E-01
65	4.8E-14	5.1E-14	3.6E-13	2.5E-01	5.8E-01	4.8E-14	5.1E-14	3.6E-13	2.5E-01	5.8E-01	4.8E-14	5.1E-14	3.6E-13	2.5E-01	5.8E-01
66	3.7E-14	3.9E-14	5.3E-13	1.4E-01	5.6E-01	3.7E-14	3.9E-14	5.3E-13	1.4E-01	5.5E-01	4.1E-14	4.4E-14	5.3E-13	1.6E-01	5.5E-01
67	4.5E-14	4.8E-14	4.2E-13	2.2E-01	5.4E-01	4.5E-14	4.8E-14	4.3E-13	2.2E-01	5.4E-01	4.8E-14	5.1E-14	4.2E-13	2.3E-01	5.5E-01
68	1.2E-15	1.2E-15	2.6E-13	1.3E-02	3.7E-01	1.2E-15	1.2E-15	2.6E-13	1.3E-02	3.7E-01	1.3E-15	1.3E-15	2.6E-13	1.4E-02	3.8E-01
69	1.8E-14	1.9E-14	2.7E-13	1.6E-01	4.6E-01	1.8E-14	1.9E-14	2.7E-13	1.6E-01	4.6E-01	2.2E-14	2.3E-14	2.7E-13	1.9E-01	4.6E-01
70	4.2E-14	4.5E-14	4.2E-13	2.0E-01	5.4E-01	4.2E-14	4.5E-14	4.2E-13	2.0E-01	5.4E-01	4.2E-14	4.5E-14	4.1E-13	2.0E-01	5.5E-01
71	5.9E-14	6.2E-14	3.8E-13	2.8E-01	6.0E-01	5.9E-14	6.2E-14	3.8E-13	2.8E-01	6.0E-01	5.9E-14	6.2E-14	3.8E-13	2.8E-01	5.9E-01
72	5.0E-14	5.3E-14	4.8E-13	2.0E-01	5.8E-01	5.0E-14	5.3E-14	4.8E-13	2.0E-01	5.8E-01	5.2E-14	5.5E-14	4.8E-13	2.1E-01	5.7E-01
73	2.5E-14	2.7E-14	3.2E-13	1.6E-01	5.4E-01	2.5E-14	2.7E-14	3.2E-13	1.6E-01	5.3E-01	2.6E-14	2.7E-14	3.2E-13	1.6E-01	5.3E-01
74	3.3E-14	3.6E-14	3.7E-13	2.1E-01	4.8E-01	3.3E-14	3.6E-14	3.7E-13	2.1E-01	4.8E-01	3.5E-14	3.7E-14	4.0E-13	2.1E-01	4.7E-01
75	2.2E-14	2.4E-14	3.4E-13	1.6E-01	4.8E-01	2.2E-14	2.4E-14	3.4E-13	1.6E-01	4.8E-01	2.4E-14	2.6E-14	3.4E-13	1.8E-01	4.6E-01
76	1.5E-14	1.6E-14	2.6E-13	1.4E-01	4.7E-01	1.5E-14	1.6E-14	2.6E-13	1.4E-01	4.7E-01	1.5E-14	1.6E-14	2.4E-13	1.5E-01	4.7E-01
77	1.8E-14	1.9E-14	2.0E-13	2.1E-01	4.8E-01	1.8E-14	1.9E-14	2.0E-13	2.1E-01	4.8E-01	1.8E-14	1.9E-14	2.1E-13	2.0E-01	4.8E-01
78	2.6E-14	2.7E-14	4.2E-13	1.3E-01	5.3E-01	2.6E-14	2.7E-14	4.2E-13	1.3E-01	5.3E-01	2.6E-14	2.7E-14	4.2E-13	1.3E-01	5.3E-01
79	5.1E-14	5.4E-14	5.4E-13	2.0E-01	5.2E-01	5.1E-14	5.4E-14	5.4E-13	2.0E-01	5.2E-01	5.1E-14	5.4E-14	5.4E-13	2.0E-01	5.2E-01
80	5.4E-14	5.6E-14	5.3E-13	1.9E-01	5.9E-01	5.4E-14	5.6E-14	5.3E-13	1.9E-01	5.9E-01	5.4E-14	5.6E-14	5.3E-13	1.9E-01	5.9E-01
81	6.0E-14	6.3E-14	4.3E-13	2.6E-01	5.8E-01	6.0E-14	6.3E-14	4.3E-13	2.6E-01	5.8E-01	6.0E-14	6.3E-14	4.1E-13	2.7E-01	5.8E-01
82	4.8E-14	5.0E-14	4.0E-13	2.4E-01	5.4E-01	4.8E-14	5.0E-14	4.0E-13	2.4E-01	5.4E-01	4.8E-14	5.0E-14	4.1E-13	2.4E-01	5.3E-01
83	9.0E-14	9.4E-14	5.3E-13	2.9E-01	6.4E-01	9.0E-14	9.4E-14	5.3E-13	2.9E-01	6.4E-01	9.0E-14	9.4E-14	5.3E-13	2.8E-01	6.5E-01
84	7.9E-14	8.2E-14	5.8E-13	2.2E-01	6.5E-01	7.9E-14	8.2E-14	5.8E-13	2.2E-01	6.5E-01	7.9E-14	8.2E-14	5.8E-13	2.2E-01	6.6E-01
85	9.6E-14	1.0E-13	5.5E-13	2.8E-01	6.5E-01	9.6E-14	1.0E-13	5.5E-13	2.8E-01	6.5E-01	9.6E-14	1.0E-13	5.6E-13	2.8E-01	6.5E-01

Table 4.3: Existing and Potential Heterogeneity Measures for SPE10 Layers

	Existing Flow Based Measures								From Permeability Description				From Diffusive TOF	
SPE10 Layer	H_V	L_C	FHI	H_K	$E_{V,b}$	$C_V(\tau_t)$	$E_{V,1PVI}$	$E_{V,2PVI}$	$\sigma^2_{\tau^2}$	$C_V(\tau^2)$	$\sigma^2_{1/C}$	$C_V(1/C)$	$E_{V,b}$	H_K
1	7.1E-01	2.3E-01	1.5E+00	2.1E+00	4.9E-01	4.2E-01	8.2E-01	1.0E+00	6.5E-03	1.1E-01	7.6E-05	3.4E-01	9.0E-01	1.1E+00
2	5.7E-01	4.2E-01	1.8E+00	2.5E+00	4.0E-01	1.5E+00	6.4E-01	8.0E-01	4.8E-03	8.9E-02	5.7E-05	2.2E-01	9.1E-01	1.1E+00
3	6.3E-01	4.0E-01	1.8E+00	2.4E+00	4.1E-01	9.2E-01	6.8E-01	8.9E-01	1.9E-03	6.4E-02	9.5E-05	3.6E-01	9.5E-01	1.1E+00
4	7.9E-01	2.3E-01	1.3E+00	1.5E+00	6.5E-01	6.5E-01	8.2E-01	9.5E-01	5.8E-03	1.1E-01	2.5E-05	3.0E-01	9.7E-01	1.0E+00
5	7.6E-01	2.8E-01	1.4E+00	2.0E+00	5.1E-01	7.2E-01	7.9E-01	9.2E-01	3.4E-03	7.6E-02	1.2E-05	2.4E-01	9.8E-01	1.0E+00
6	7.3E-01	3.2E-01	1.6E+00	2.0E+00	5.0E-01	8.6E-01	7.6E-01	9.2E-01	4.0E-03	8.8E-02	1.8E-04	2.7E-01	9.7E-01	1.0E+00
7	7.1E-01	3.2E-01	1.5E+00	1.9E+00	5.4E-01	1.1E+00	7.5E-01	8.7E-01	1.7E-03	5.0E-02	2.6E-05	1.8E-01	9.7E-01	1.0E+00
8	7.9E-01	3.2E-01	1.5E+00	1.9E+00	5.3E-01	1.1E+00	7.6E-01	8.8E-01	8.8E-04	3.7E-02	6.1E-04	4.6E-01	9.6E-01	1.0E+00
9	8.2E-01	2.3E-01	1.3E+00	1.6E+00	6.4E-01	7.2E-01	8.2E-01	9.3E-01	8.3E-04	3.6E-02	3.9E-04	6.0E-01	9.6E-01	1.0E+00
10	7.9E-01	2.3E-01	1.3E+00	1.7E+00	5.9E-01	6.0E-01	8.3E-01	9.6E-01	2.7E-03	6.9E-02	1.7E-04	5.2E-01	9.7E-01	1.0E+00
11	7.1E-01	3.0E-01	1.4E+00	2.0E+00	5.0E-01	7.1E-01	7.8E-01	9.1E-01	4.0E-03	8.6E-02	8.1E-04	4.7E-01	9.2E-01	1.1E+00
12	5.5E-01	4.9E-01	2.0E+00	3.3E+00	3.0E-01	1.4E+00	6.4E-01	7.9E-01	1.4E-03	4.9E-02	1.7E-04	3.1E-01	9.0E-01	1.1E+00
13	6.3E-01	4.4E-01	1.9E+00	2.3E+00	4.3E-01	1.4E+00	6.4E-01	8.0E-01	3.3E-03	7.7E-02	2.8E-04	2.4E-01	9.2E-01	1.1E+00
14	8.1E-01	2.4E-01	1.3E+00	1.8E+00	5.6E-01	6.6E-01	8.3E-01	9.3E-01	4.2E-03	8.7E-02	6.6E-05	2.1E-01	9.4E-01	1.1E+00
15	7.4E-01	3.4E-01	1.5E+00	2.0E+00	5.0E-01	1.0E+00	7.4E-01	8.7E-01	3.2E-03	7.8E-02	7.0E-04	1.7E-01	9.3E-01	1.1E+00
16	6.4E-01	3.9E-01	1.7E+00	2.6E+00	3.9E-01	1.0E+00	7.1E-01	8.7E-01	2.8E-03	6.8E-02	4.2E-04	1.5E-01	8.5E-01	1.2E+00
17	8.7E-01	2.4E-01	1.4E+00	1.6E+00	6.2E-01	6.5E-01	8.2E-01	9.3E-01	4.2E-03	8.2E-02	3.3E-04	2.3E-01	9.6E-01	1.0E+00
18	6.2E-01	4.3E-01	1.7E+00	2.5E+00	4.0E-01	1.4E+00	6.7E-01	8.0E-01	5.3E-03	1.0E-01	3.5E-04	4.0E-01	8.8E-01	1.1E+00
19	7.4E-01	3.0E-01	1.5E+00	2.0E+00	4.9E-01	6.3E-01	7.7E-01	9.5E-01	9.0E-03	1.3E-01	1.0E-03	3.9E-01	9.6E-01	1.0E+00
20	7.6E-01	3.1E-01	1.5E+00	2.2E+00	4.6E-01	8.1E-01	7.6E-01	9.1E-01	2.9E-03	6.6E-02	1.3E-03	2.8E-01	9.0E-01	1.1E+00
21	7.0E-01	3.3E-01	1.5E+00	2.1E+00	4.9E-01	9.5E-01	7.5E-01	8.7E-01	3.1E-03	7.2E-02	1.1E-04	1.5E-01	9.3E-01	1.1E+00
22	8.1E-01	2.8E-01	1.5E+00	2.0E+00	5.1E-01	5.6E-01	7.8E-01	9.7E-01	3.1E-03	6.8E-02	1.6E-03	9.1E-01	9.2E-01	1.1E+00
23	7.7E-01	2.9E-01	1.4E+00	2.0E+00	5.0E-01	7.0E-01	8.0E-01	9.3E-01	5.7E-03	9.4E-02	1.4E-03	3.4E-01	8.8E-01	1.1E+00
24	8.5E-01	2.3E-01	1.3E+00	1.7E+00	6.0E-01	5.2E-01	8.3E-01	9.6E-01	1.0E-03	3.9E-02	3.6E-04	1.6E-01	9.5E-01	1.1E+00
25	6.4E-01	2.6E-01	1.4E+00	1.8E+00	5.5E-01	7.5E-01	8.1E-01	9.3E-01	2.8E-03	7.1E-02	4.3E-04	3.2E-01	8.9E-01	1.1E+00
26	7.9E-01	1.9E-01	1.3E+00	1.7E+00	5.9E-01	4.5E-01	8.7E-01	9.7E-01	5.0E-03	9.3E-02	8.7E-04	7.3E-01	9.1E-01	1.1E+00
27	6.9E-01	2.5E-01	1.5E+00	1.8E+00	5.4E-01	5.3E-01	8.1E-01	9.7E-01	4.3E-03	8.6E-02	1.3E-05	1.8E-01	8.1E-01	1.2E+00
28	7.9E-01	2.3E-01	1.3E+00	1.7E+00	5.9E-01	5.5E-01	8.3E-01	9.6E-01	4.8E-03	8.8E-02	1.5E-04	3.6E-01	8.9E-01	1.1E+00
29	8.9E-01	1.6E-01	1.3E+00	1.5E+00	6.5E-01	3.2E-01	8.9E-01	9.9E-01	5.4E-03	9.2E-02	1.4E-03	1.1E+00	9.1E-01	1.1E+00
30	6.9E-01	3.1E-01	1.5E+00	2.2E+00	4.5E-01	6.2E-01	7.7E-01	9.5E-01	8.0E-03	1.2E-01	6.9E-03	8.6E-01	9.2E-01	1.1E+00
31	6.3E-01	3.0E-01	1.5E+00	2.0E+00	4.9E-01	6.3E-01	7.7E-01	9.5E-01	3.7E-03	8.5E-02	2.9E-03	5.6E-01	9.1E-01	1.1E+00
32	8.2E-01	2.5E-01	1.4E+00	1.8E+00	5.4E-01	5.4E-01	8.1E-01	9.7E-01	3.7E-03	8.0E-02	1.4E-02	1.0E+00	9.2E-01	1.1E+00
33	6.5E-01	3.5E-01	1.6E+00	2.5E+00	4.1E-01	7.3E-01	7.4E-01	9.2E-01	8.5E-03	1.2E-01	8.2E-04	4.4E-01	8.6E-01	1.2E+00
34	6.7E-01	4.1E-01	1.8E+00	2.5E+00	4.0E-01	9.8E-01	6.9E-01	8.3E-01	2.5E-03	6.3E-02	3.4E-04	6.3E-01	9.1E-01	1.1E+00
35	7.2E-01	3.4E-01	1.7E+00	2.5E+00	4.1E-01	7.7E-01	7.3E-01	9.1E-01	1.4E-02	1.7E-01	3.3E-04	7.7E-01	8.7E-01	1.1E+00
36	-	4.0E-01	1.6E+00	3.0E+00	3.4E-01	1.2E+00	6.9E-01	8.1E-01	7.6E-03	1.9E-01	3.3E-05	6.8E-01	7.7E-01	1.3E+00
37	-	7.0E-01	3.2E+00	4.6E+00	2.2E-01	3.2E+00	3.9E-01	4.7E-01	7.6E-03	2.5E-01	4.8E-03	1.9E+00	6.5E-01	1.5E+00
38	-	6.4E-01	2.3E+00	4.7E+00	2.1E-01	5.3E+00	5.1E-01	5.8E-01	6.9E-03	1.9E-01	5.3E-03	6.4E-01	6.4E-01	1.6E+00
39	-	5.5E-01	1.9E+00	3.6E+00	2.8E-01	5.3E+00	5.5E-01	6.1E-01	1.3E-02	2.4E-01	1.1E-02	5.1E-01	7.2E-01	1.4E+00
40	-	5.2E-01	1.9E+00	3.4E+00	2.9E-01	5.1E+00	5.8E-01	6.4E-01	7.3E-03	2.1E-01	1.2E-02	6.3E-01	7.5E-01	1.3E+00
41	3.5E-01	6.0E-01	2.3E+00	3.9E+00	2.6E-01	4.4E+00	5.2E-01	6.1E-01	7.0E-03	2.1E-01	5.3E-03	8.5E-01	7.7E-01	1.3E+00
42	-	5.5E-01	2.2E+00	4.3E+00	2.3E-01	1.9E+00	5.7E-01	7.1E-01	7.5E-03	1.8E-01	1.5E-05	6.1E-01	6.7E-01	1.5E+00
43	3.5E-01	7.1E-01	2.7E+00	5.7E+00	1.8E-01	5.3E+00	4.0E-01	4.5E-01	1.0E-02	2.0E-01	2.4E-02	1.7E+00	6.5E-01	1.5E+00
44	4.5E-01	5.1E-01	1.8E+00	3.3E+00	3.0E-01	4.2E+00	6.1E-01	6.9E-01	1.2E-02	1.9E-01	4.5E-03	4.8E-01	7.8E-01	1.3E+00
45	5.1E-01	4.0E-01	1.6E+00	2.7E+00	3.7E-01	4.3E+00	6.6E-01	7.3E-01	7.0E-03	1.4E-01	7.0E-03	3.8E-01	7.9E-01	1.3E+00
46	4.1E-01	6.3E-01	2.4E+00	4.5E+00	2.2E-01	5.3E+00	4.5E-01	5.3E-01	7.3E-03	1.6E-01	1.7E-02	1.0E+00	8.2E-01	1.2E+00
47	-	4.0E-01	1.6E+00	2.5E+00	4.0E-01	2.0E+00	6.8E-01	7.8E-01	5.5E-03	1.9E-01	3.5E-05	1.0E+00	9.0E-01	1.1E+00
48	-	6.8E-01	2.6E+00	4.5E+00	2.2E-01	5.6E+00	4.5E-01	5.1E-01	5.4E-03	1.9E-01	1.6E-02	1.0E+00	6.7E-01	1.5E+00
49	-	6.5E-01	2.4E+00	3.7E+00	2.7E-01	7.9E+00	4.9E-01	5.5E-01	7.8E-03	1.8E-01	3.4E-03	3.1E-01	6.6E-01	1.5E+00
50	3.8E-01	6.0E-01	2.4E+00	3.6E+00	2.8E-01	1.1E+01	4.8E-01	5.4E-01	7.4E-03	1.6E-01	5.4E-03	5.1E-01	6.6E-01	1.5E+00
51	4.5E-01	5.6E-01	2.0E+00	4.3E+00	2.3E-01	4.6E+00	5.7E-01	6.8E-01	1.1E-02	2.2E-01	6.1E-03	4.8E-01	8.3E-01	1.2E+00
52	5.2E-01	4.5E-01	1.7E+00	3.1E+00	3.2E-01	3.3E+00	6.4E-01	7.2E-01	9.1E-03	2.0E-01	8.6E-03	5.8E-01	8.3E-01	1.2E+00
53	6.1E-01	3.7E-01	1.5E+00	3.0E+00	3.4E-01	3.3E+00	7.2E-01	8.2E-01	5.3E-03	1.2E-01	3.7E-03	2.5E-01	8.7E-01	1.2E+00
54	6.1E-01	3.7E-01	1.5E+00	2.2E+00	4.5E-01	3.3E+00	7.1E-01	7.9E-01	1.0E-02	1.7E-01	2.6E-03	2.5E-01	9.0E-01	1.1E+00
55	5.4E-01	5.4E-01	1.9E+00	3.2E+00	3.1E-01	7.0E+00	6.3E-01	6.9E-01	2.0E-02	2.5E-01	1.3E-02	4.0E-01	7.6E-01	1.3E+00
56	-	4.7E-01	1.7E+00	2.8E+00	3.6E-01	3.1E+00	6.6E-01	7.4E-01	1.7E-02	2.1E-01	5.5E-03	3.5E-01	7.6E-01	1.3E+00
57	6.3E-01	3.2E-01	1.4E+00	2.1E+00	4.8E-01	2.1E+00	7.3E-01	8.0E-01	8.7E-03	1.5E-01	4.0E-03	2.2E-01	8.7E-01	1.2E+00
58	5.4E-01	4.4E-01	1.6E+00	2.5E+00	3.9E-01	4.2E+00	7.0E-01	7.7E-01	2.0E-02	2.5E-01	5.8E-03	2.8E-01	8.4E-01	1.2E+00
59	5.0E-01	5.1E-01	1.8E+00	3.2E+00	3.1E-01	4.6E+00	6.0E-01	6.8E-01	2.1E-02	2.4E-01	1.5E-02	4.8E-01	7.5E-01	1.3E+00
60	4.6E-01	4.5E-01	1.7E+00	2.8E+00	3.6E-01	5.7E+00	6.1E-01	6.8E-01	1.3E-02	2.2E-01	4.6E-03	3.6E-01	7.8E-01	1.3E+00
61	5.4E-01	4.1E-01	1.6E+00	2.9E+00	3.4E-01	3.2E+00	6.4E-01	7.1E-01	4.5E-03	1.4E-01	6.3E-03	3.9E-01	8.4E-01	1.2E+00
62	-	5.2E-01	1.9E+00	3.4E+00	2.9E-01	2.9E+00	6.2E-01	7.1E-01	5.1E-03	1.8E-01	4.5E-03	4.2E-01	7.7E-01	1.3E+00
63	4.6E-01	5.0E-01	1.8E+00	3.1E+00	3.2E-01	4.6E+00	6.0E-01	6.7E-01	7.1E-03	1.8E-01	3.1E-03	2.9E-01	7.8E-01	1.3E+00
64	4.6E-01	5.6E-01	2.0E+00	3.9E+00	2.5E-01	7.1E+00	5.7E-01	6.5E-01	7.9E-03	1.5E-01	3.4E-03	4.1E-01	8.1E-01	1.2E+00
65	4.5E-01	4.9E-01	1.8E+00	2.9E+00	3.4E-01	7.1E+00	6.1E-01	6.7E-01	1.1E-02	1.8E-01	9.2E-03	3.8E-01	7.8E-01	1.3E+00
66	-	5.1E-01	1.8E+00	3.1E+00	3.2E-01	6.8E+00	5.7E-01	6.1E-01	8.6E-03	1.7E-01	3.6E-03	4.4E-01	7.6E-01	1.3E+00
67	5.1E-01	5.4E-01	1.8E+00	3.7E+00	2.7E-01	6.7E+00	5.8E-01	6.2E-01	9.2E-03	1.8E-01	6.4E-03	3.7E-01	7.3E-01	1.4E+00
68	3.9E-01	6.2E-01	2.7E+00	5.7E+00	1.7E-01	1.9E+00	5.0E-01	6.6E-01	1.2E-02	3.0E-01	4.0E-04	1.5E+00	8.4E-01	1.2E+00
69	4.3E-01	5.4E-01	1.9E+00	2.9E+00	3.4E-01	6.0E+00	5.6E-01	6.0E-01	6.6E-03	1.8E-01	1.1E-02	6.5E-01	8.3E-01	1.2E+00
70	-	5.4E-01	2.0E+00	3.4E+00	3.0E-01	5.2E+00	5.7E-01	6.5E-01	1.2E-02	2.0E-01	3.6E-03	2.9E-01	7.6E-01	1.3E+00
71	5.4E-01	3.8E-01	1.5E+00	2.3E+00	4.4E-01	5.2E+00	6.9E-01	7.6E-01	8.4E-03	1.5E-01	1.1E-02	3.6E-01	8.1E-01	1.2E+00
72	5.3E-01	3.7E-01	1.5E+00	2.5E+00	4.1E-01	4.3E+00	6.6E-01	7.1E-01	1.1E-02	1.8E-01	4.6E-03	3.4E-01	8.2E-01	1.2E+00
73	-	5.0E-01	1.9E+00	3.0E+00	3.4E-01	4.9E+00	5.9E-01	6.5E-01	1.1E-02	2.0E-01	1.9E-03	2.7E-01	7.6E-01	1.3E+00
74	-	4.9E-01	1.8E+00	2.8E+00	3.6E-01	5.8E+00	5.7E-01	6.3E-01	6.1E-03	1.6E-01	6.0E-03	3.7E-01	7.3E-01	1.4E+00
75	3.7E-01	6.9E-01	2.8E+00	4.4E+00	2.3E-01	7.3E+00	4.1E-01	4.6E-01	1.2E-02	2.3E-01	9.8E-03	6.3E-01	6.6E-01	1.5E+00
76	-	6.9E-01	2.7E+00	4.8E+00	2.1E-01	8.0E+00	4.2E-01	4.8E-01	8.8E-03	2.0E-01	3.9E-03	4.5E-01	6.2E-01	1.6E+00
77	-	4.8E-01	1.8E+00	3.2E+00	3.2E-01	5.7E+00	5.9E-01	6.3E-01	6.5E-03	1.7E-01	2.1E-03	2.2E-01	7.0E-01	1.4E+00
78	-	6.0E-01	2.2E+00	4.0E+00	2.5E-01	9.1E+00	5.0E-01	5.7E-01	1.7E-02	2.5E-01	1.7E-03	3.3E-01	6.9E-01	1.5E+00
79	-	6.1E-01	2.3E+00	4.5E+00	2.2E-01	1.2E+01	5.2E-01	6.0E-01	4.0E-03	1.2E-01	2.3E-03	2.4E-01	7.3E-01	1.4E+00
80	-	5.4E-01	2.0E+00	3.5E+00	2.9E-01	1.4E+01	5.4E-01	6.1E-01	6.9E-03	1.4E-01	1.7E-03	2.2E-01	7.8E-01	1.3E+00
81	4.8E-01	5.2E-01	1.9E+00	3.4E+00	3.0E-01	1.1E+01	5.8E-01	6.4E-01	1.3E-02	2.0E-01	1.1E-02	4.0E-01	7.5E-01	1.3E+00
82	5.1E-01	4.4E-01	1.6E+00	2.7E+00	3.7E-01	6.0E+00	6.6E-01	7.3E-01	1.8E-02	2.5E-01	7.0E-03	3.5E-01	7.9E-01	1.3E+00
83	5.8E-01	4.0E-01	1.5E+00	2.8E+00	3.5E-01	3.5E+00	7.4E-01	8.2E-01	8.4E-03	1.4E-01	9.8E-03	3.5E-01	9.0E-01	1.1E+00
84	6.7E-01	2.9E-01	1.3E+00	2.1E+00	4.9E-01	3.9E+00	8.0E-01	8.6E-01	1.1E-02	1.6E-01	4.1E-03	2.9E-01	9.3E-01	1.1E+00
85	6.0E-01	4.0E-01	1.8E+00	3.0E+00	3.3E-01	1.9E+00	7.1E-01	8.6E-01	1.2E-02	1.7E-01	9.6E-03	3.5E-01	8.7E-01	1.1E+00

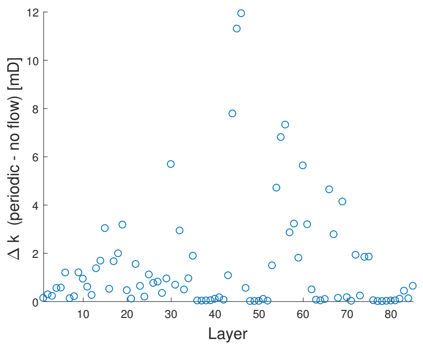
Chapter 5

Discussion

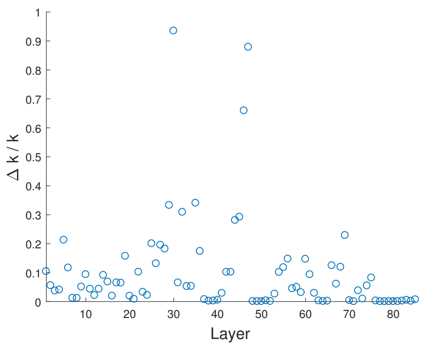
5.1 Effect of Boundary Conditions on Streamlines

Figure 4.1 and Figure 4.2 in Section 4.1 shows streamlines tracked with the Pollock algorithm for no flow boundaries and periodic boundaries on the longest horizontal sides for two layers of the SPE10 model. Regular grids are already supported in MRST's *pollock* method, while the extension to periodic grids has been a part of the work for this thesis. Streamlines will follow the path of least resistance, and hence they will gather in the high permeability regions as observed in both figures. The periodic boundary conditions allow for flow across the sides and the almost straight streamlines at each side which are not natural flow paths unless the reservoir actually is a rectangular cuboid are avoided. It is therefore likely that the periodic grid will be a better approximation in e.g. an upscaling process where the reservoir continues outside the cells that are evaluated for upscaling. As the flow is allowed to find an easier path than in the more conservative no flow case, the upscaled permeability over the whole periodic model has to be less or equal to the upscaled permeability in the no flow model. Figure 5.1a shows the permeability increase for each layer with periodic boundaries. This is the difference between k_D for periodic and no flow boundaries in Table 4.2. The relative increase is shown in Figure 5.1b where we can observe that while most layers have less than 20% increase, there are still a few outliers. The greatest relative increase is in layer 30 with more than 93% increase in the overall permeability.

Figure 5.2 shows the permeability field of layer 30 with tracked streamlines, while Figure 5.3 shows the flow rate through each cell for the same layer, both with no flow and periodic



(a) Increase.



(b) Increase relative to no flow boundaries.

Figure 5.1: Layer permeability increase with periodic boundaries instead of no flow.

boundaries. Comparing the two figures, it is likely that the main reason for the permeability increase of the layer is that the flow can pass directly through the left boundary from the high permeability region in the lower left corner and into the high permeability region right above the center at the right side of the model, without being forced through the low permeability belt between the two regions. From the same two figures, we also get an example of how the streamline density is not necessarily a good indicator of flow rate, as both cases in Figure 5.2 have a concentration of streamlines in the lower right corner, while most of the flow actually occurs in the lower left corner as seen from Figure 5.3.

5.2 Particle and Diffusive Time of Flight

Figure 4.3 shows a comparison of the flow based particle time of flight τ_t and the square of the diffusive time of flight τ_d^2 found by tracking the pressure front. Some trends can be observed from the figure, but many of the details are lost due to relatively equal colors over large areas. A logarithmic color-scale has been tested, but not shown in this thesis, as it did not improve the contrast. Figure 5.4 shows the τ_t and τ_d^2 for layer 30 at the time of breakthrough, resulting in a better contrast. No flow side boundaries are used, and the particle time of flight is from the *computeDTOF* method in MRST, and not streamline based. Similar to Figure 4.3 the colormap is defined as ”low” to ”high”, as there is no direct conversion between the two measures. It can be seen that the low permeability region at the center of the lower half in Figure 5.2 is mainly bypassed by the flow at the time of flow breakthrough, while the pressure front has a more piston like propagation. A key difference between these two methods is that while τ_t is calculated on a steady-state flow

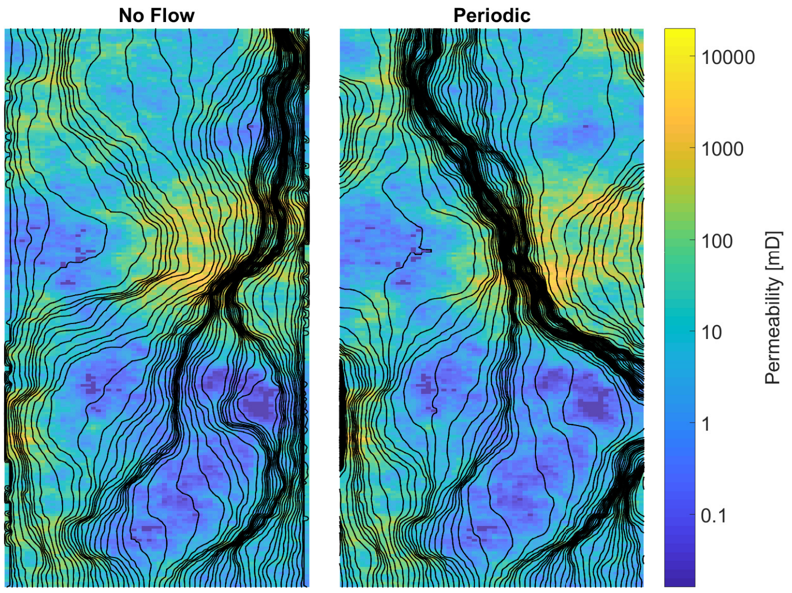


Figure 5.2: Permeability field and streamlines for no flow and periodic boundaries in layer 30.

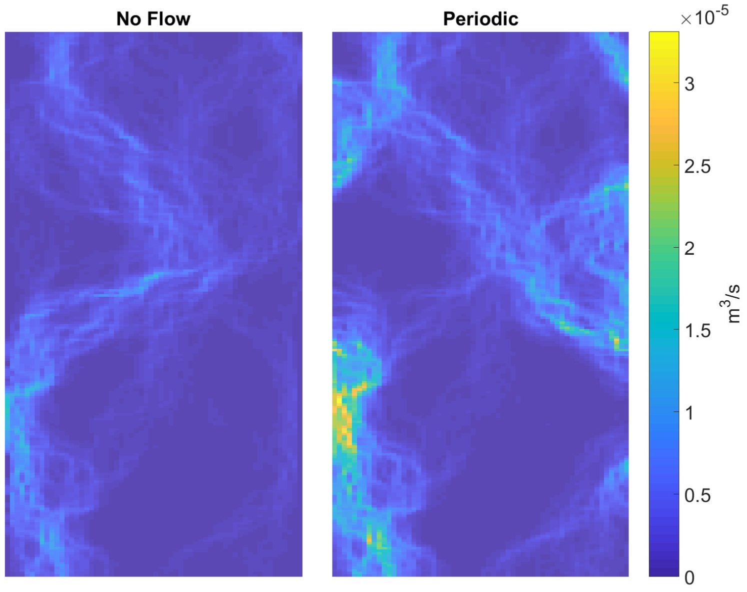


Figure 5.3: Flux through each cell for no flow and periodic boundaries in layer 30.

field, the front propagation in τ_d is a more dynamic method, as the pressure front never knows what lies ahead. A consequence of this is that the flow field passes around low permeability regions, while the pressure front continues until it hits it. This can be seen close to the bottom of figure 5.4 where there is a region of low τ_d^2 near the center, while τ_t is above the cut off value. Additionally, the high permeability region seen at the right side of Figure 5.2 is not reached by the flow for τ_t at breakthrough , while the pressure front reaches it long before breakthrough. The main reason for the increased volumetric sweep of τ_d^2 is probably that the pressure front behaves like a wave which spreads out in all possible directions, while the flow follows the path of least resistance. The different natures of these two methods for time of flight will make an interesting comparison when discussing the use of these methods for heterogeneity measures.

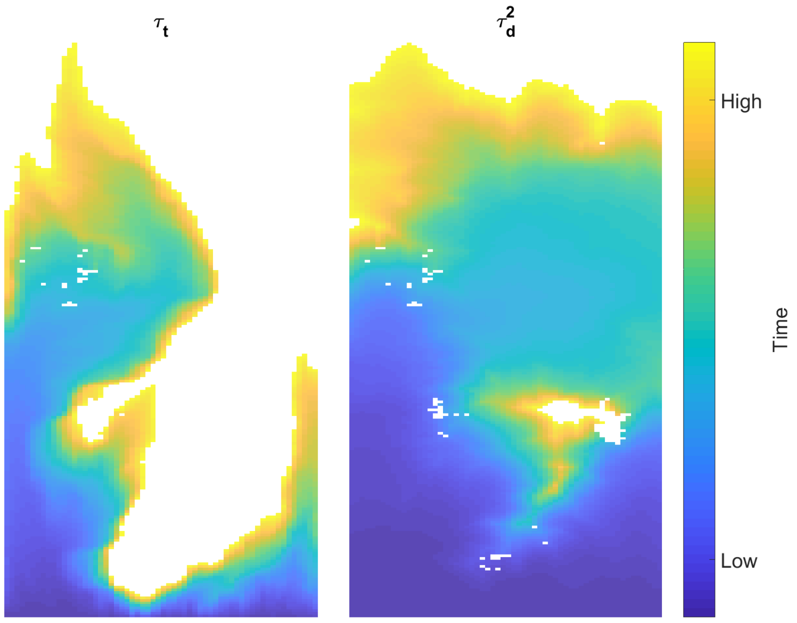


Figure 5.4: Comparison of particle time of flight (τ_t) to the left, and diffusive time of flight squared (τ_d^2) to the right. Cutoff at time of breakthrough for both. Layer 30, no flow boundaries at the sides.

5.3 Uncertainty Analysis

Results from an uncertainty analysis of the streamline and diffusive time of flight algorithms where shown in Section 4.3. These results will be further discussed in this section.

5.3.1 Expanded Pollock Algorithm

5.3.1.1 Streamline Length

The expanded Pollock algorithm allows for two methods of length estimation, a straight line approach for each substep, and a velocity integral using a trapezoidal approximation of the velocity magnitude for each substep. We observed in Figure 4.4 and Figure 4.5 that both the mean relative errors and the maximum relative errors are larger when using the velocity based method. The straight line approach will always be the shortest path between two coordinates, and will therefore always be an underestimation of the streamline length. The velocity integral as implemented here is though only a mean between the velocity magnitude at the beginning and the end of each substep, multiplied with the time of flight. If the velocity is changing significantly over the substep, the mean is not necessarily very representative and this can introduce some errors to the length estimation. An example is in a sharp turn where the flow changes direction, and there can be something close to a stalling point at the middle of the turn. In this case the mean velocity will be overestimated, resulting in an overestimation of the length as well. Still, as we observe from both Figure 4.4 and Figure 4.5, both methods converge towards the correct solution, and the errors are relatively small as long as about 5 or more substeps are used. The heterogeneity measures have been calculated with 10 substeps per grid cell, hence the error contribution from the streamline length estimation is likely to be insignificant. This is also supported from Table 4.2, where there are no differences between the two methods within the two significant digits.

5.3.1.2 Pressure Gradient

Three methods for pressure gradients along streamlines have been presented, and we have seen a comparison of them in Figure 4.7 and Table 4.1. The 1D Darcy calculation gave least errors of the three methods, especially in the Tarbert layers (layer 1-35) where the relative errors were two orders of magnitude less than for both the linear and the trilinear interpolation methods. For the Upper Ness layers (layer 36-85), the relative errors are still lower for the 1D Darcy method, but the linear interpolation is of almost the same accuracy. The reason why we see such an error increase in the tortuous Upper Ness layers for the 1D Darcy calculations is likely to be similar to the example we saw for velocity-integrated streamline lengths in Section 5.3.1.1, where sharp turns could increase errors. This is because the 1D Darcy implementation uses the same velocity approximation as the velocity integral length method. An issue with the 1D Darcy method is that it does not include the pressure grid, while both the linear and the trilinear interpolation methods uses the pressure values for nearby cells. Hence, errors can be accumulated, and the total pressure drop over the model after multiplying gradients with step lengths are not necessarily correct. The result of this is shown in Table 4.2, where the permeabilities k_D from Darcy calculations over the full model are not exactly equal to the permeabilities k_S calculated from streamline parameters. However, as the errors are small, the errors are most likely less than the already existing uncertainty in the geological model, and therefore not very significant.

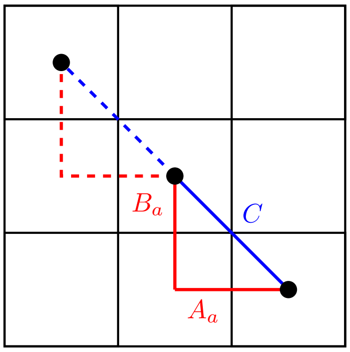
Trilinear interpolation gave the largest errors of the three presented methods. This is most likely a result of the fact that including pressures from up to 9 cells in 2D, and up to 27 cells in 3D, introduce the chance getting positive pressure gradients in the flow direction, which is not physically possible. The integral of the inverse pressure gradient used in the streamline hydraulic conductance $B(\mathcal{S})$ and the streamline constriction factor $C(\mathcal{S})$ will then become somewhat unstable, which is shown in increased errors of the effective hydraulic conductance B_e in Figure 4.7.

5.3.1.3 Number of Streamlines

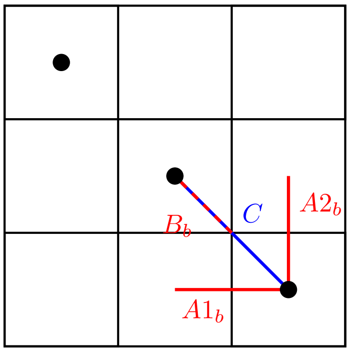
Results of the sensitivity to an increased number of streamlines are shown for two different methods in Section 4.3.1.3. Figure 4.8 shows the error of the effective hydraulic conductance B_e, relative to the volumetric average of k_h, which was shown in Equation (3.40) to be the exact solution (due to isotropic horizontal permeability). We observe from Figure 4.8 that the errors in the Tarbert formation are negligible compared to the errors in the Upper Ness formation with the same number of streamlines. The same difference can be observed in Figure 4.9 where the error of the dynamic Lorenz coefficient is shown. The streamlines are more evenly distributed in the Tarbert formation, while the distance between streamlines in low permeability regions can be quite large, as seen when comparing Tarbert in Figure 4.1 and Upper Ness in Figure 4.2. A streamline in the Upper Ness formation can therefore be representable for a relatively large fraction of the volume, thus reducing the accuracy as B_e is the volumetric average of k_h. When it comes to the Lorenz coefficient in Figure 4.9, the increased errors in the Upper Ness formation can be due to the generally larger variance of time of flights, which is seen from the coefficients of variation in Table 4.3 (column $C_V(\tau_t)$, and the cross-plot in Figure 5.11. When the variation of time of flights is limited, there is also limited room for errors due to too few streamlines. We also observe that the errors are generally larger for the Lorenz coefficient in Figure 4.9 than the hydraulic conductance in 4.8. 20 streamlines per inlet cell have been used in this thesis, which should give results within a few percents of error. A challenge is of course to generalize how many streamlines you would need for an arbitrary reservoir model. If we look at the example $F - \Phi$-diagram in Figure 2.1, from which to Lorenz coefficient is defined, we see that the Lorenz coefficient will be sensitive to the streamtubes with high rates per pore-volume.

5.3.2 Fast Marching Method for Diffusive Time of Flight

Section 4.2 shows results of a comparison between the diffusive time of flight converted to actual time, and the analytic solution from the well-known radius of investigation. The comparison is done in a homogeneous and isotropic medium, because the same comparison with an analytic solution is not possible for a heterogeneous medium, as previously mentioned. Nevertheless, the results should at least show whether the implementation of FMM is working and reliable. Figure 4.10 compares the time from diffusive time of flight and the analytic solution, and especially the relative error between them in Figure 4.10c is interesting. The error is clearly most severe at the diagonals, especially close to the origin of the pressure pulse at the center of the figure. The error at the diagonals is likely to be an effect of a "mild" version of the so-called *Manhattan distance*. With the Manhattan distance, we usually mean the distance following the grid lines instead of the shortest path, similar to the street structure at the island of Manhattan (New York). Figure 5.5 shows compares the regular Manhattan distance, with the milder version that occurs in the FMM implementation. In Figure 5.5a we see the regular Manhattan distance, where the shortest distance between two neighboring points, marked as a solid blue line, is $C = \sqrt{A_a^2 + B_a^2}$. If we assume that the cells are all unit cells (dimensions 1×1), we get $C = \sqrt{1^2 + 1^2} = \sqrt{2}$, while the Manhattan distance marked in red is $A_a + B_a = 1 + 1 = 2$ (we could also step up, and then left, with the same result). The ratio between the Manhattan distance and the shortest distance is thus $\sqrt{2} \approx 1.41$. Another step is marked in dashed lines. If we evaluate the same step with how the diffusive time of flight will be calculated with the FMM implementation, the pulse will first reach the cells connected with $A1_b$ and $A2_b$, which in a unit cell both are of length 1. These are then both weighted into the next step, so that we instead follow the dashed diagonal B_b. The length of B_b is $\sqrt{0.5^2 + 0.5^2} = {}^1/\sqrt{2}$. The "mild" Manhattan length from the origin is then $1 + {}^1/\sqrt{2}$, which is $\approx= 1.21$ times larger than the shortest distance C. The error is therefore reduced compared to a regular Manhattan distance implementation. From Figure 4.10c we observe that the largest relative error is a little less than 0.5. This is a result of the squaring of the diffusive time of flight in the conversion to actual time, as seen in Equation (2.41). The largest expected error is therefore $\approx 1.21^2 - 1 \approx 0.46$. This fits with what we see from Figure 4.10c. The error is thus exactly as expected due to grid effects, and as the relative error is quickly reduced with distance to the origin, the implementation will be assumed to be sufficiently reliable.



(a) Manhattan distance.



(b) ”Mild” form of Manhattan distance.

Figure 5.5: Comparison of the ”true” Manhattan distance, and the ”mild” version we get with the FMM implementation

5.4 Heterogeneity Measures

Various existing and potential heterogeneity measures were presented in Section 2.3 and Section 3.4. Calculated measures for the individual SPE10 layers with no flow boundaries where presented in Table 4.3. To evaluate their potential as heterogeneity measures their correlation with the recovery at 90% water cut, recovery increase by tertiary polymer injection, and recovery increase due to increased macroscopic sweep efficiency by tertiary polymer injection from the study by Krogstad et al. (2017) presented in Section 3.5.2 will be investigated. Some layers are excluded due to convergence issues in the study, but they remaining layers should be relatively representable. The correlation coefficient used is the Pearson correlation coefficient ρ, which is defined as (See e.g. Jensen et al., 2000)

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \quad , \tag{5.1}$$

where X and Y are the two data sets, $Cov(X,Y)$ is the covariance of X and Y and σ_X and σ_Y are the standard deviations of X and Y, respectively. ρ is bounded between -1, meaning perfectly negatively correlated, and 1, meaning perfectly positively correlated. Generally the larger $|\rho|$, the better heterogeneity measure.

The results of existing heterogeneity measures will be discussed first as a benchmark for the new potential measures that will follow afterwards. The same structure is followed for all measures, with three individual plots for recovery at 90% (pure waterflooding), recovery increase due to polymer injection, and macro sweep increase due to polymer injection. In addition, the Tarbert and Upper Ness formations are separated with blue and red color, respectively.

5.4.1 Existing Heterogeneity Measures

Figure 5.6 shows the dynamic Lorenz coefficient for the individual layers. A strong correlation is observed in all three plots, especially for recovery at 90% water cut and macro sweep recovery increase. The vorticity factors in Figure 5.7 also give good correlations with the recovery data, though slightly lower than the Lorenz coefficient, especially for the recovery increase due to improved macro sweep where the data is a bit more spread than for the Lorenz coefficient in Figure 5.6. Notice that the vorticity coefficient is inversely related to the degree of heterogeneity. The flow heterogeneity indices (FHI) in Figure 5.8, the Koval factors in Figure 5.9 and the sweep efficiencies at flow breakthrough in Figure 5.10 all have significant correlation at about the same level, though less than for both Lorenz and vorticity. The coefficients of variation of particle time of flight in Figure 5.11 give a quite high correlation factor for recovery at 90% water cut, but it seems like there is mostly a trend for the less tortuous Tarbert layers, while the data for the tortuous Upper Ness layers are more spread. It is therefore not likely to be a very reliable heterogeneity measure. The best correlations of the existing measures are the volumetric sweep efficiencies at 1 and 2 pore volumes of water injected in Figure 5.12 and Figure 5.13. The generally strong correlations are very interesting, as they are found with a single steady-state and single-phase flow field, and still correlates very well with much more computationally demanding two-phase simulations. It is therefore not unlikely that several of these measures can be used for rapid evaluation of multiple development strategies.

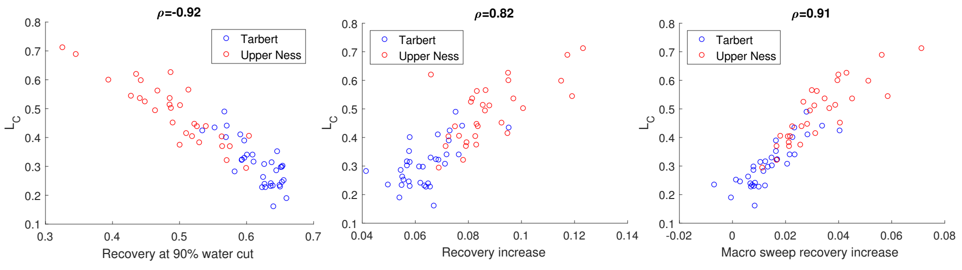


Figure 5.6: Dynamic Lorenz coefficient vs recovery.

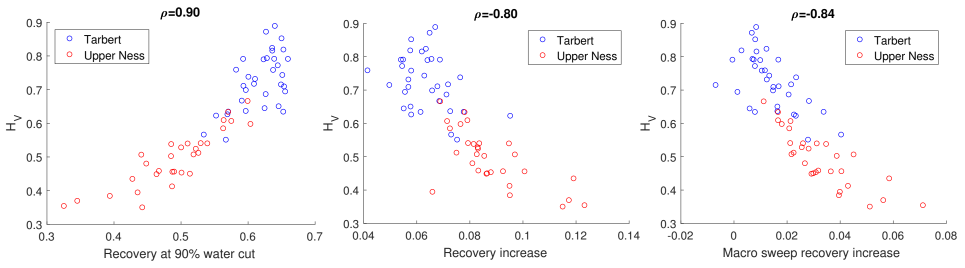


Figure 5.7: Vorticity coefficient vs recovery.

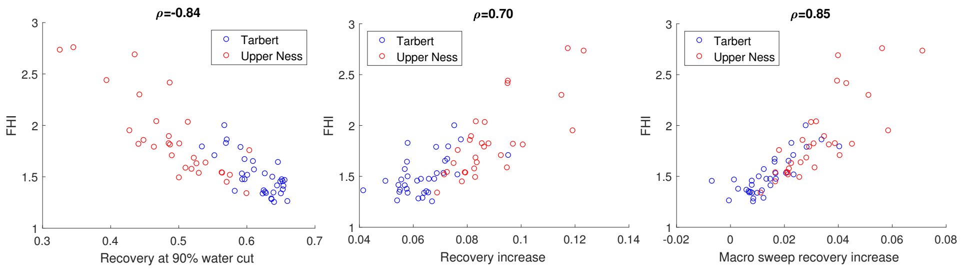


Figure 5.8: Flow heterogeneity index vs recovery.

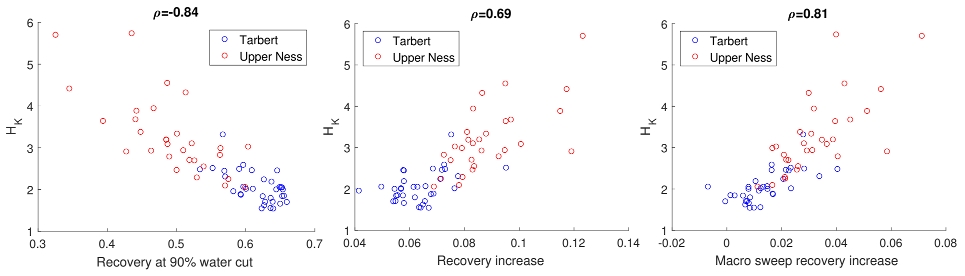


Figure 5.9: Koval factor (from flow) vs recovery.

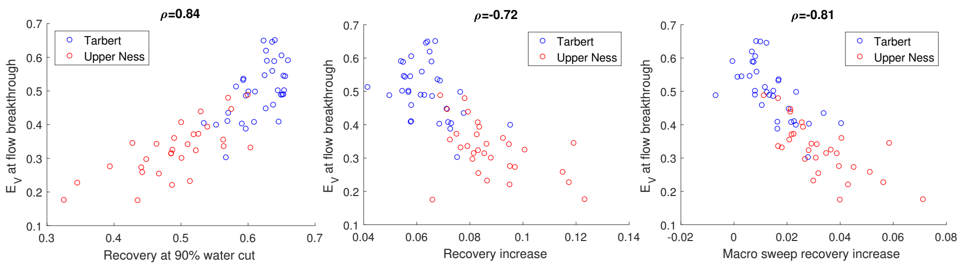


Figure 5.10: Volumetric sweep efficiency at flow breakthrough vs recovery.

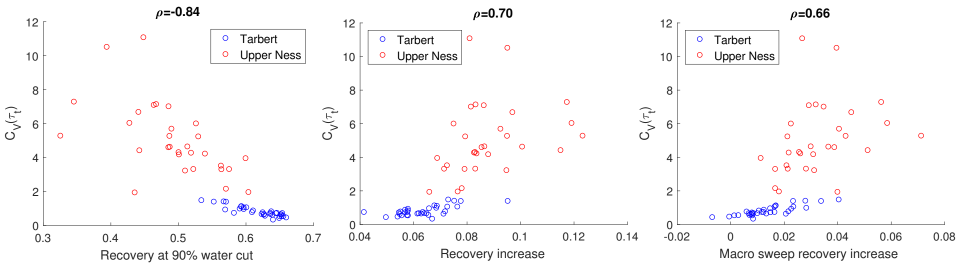


Figure 5.11: Coefficient of variation of time of flight vs recovery.

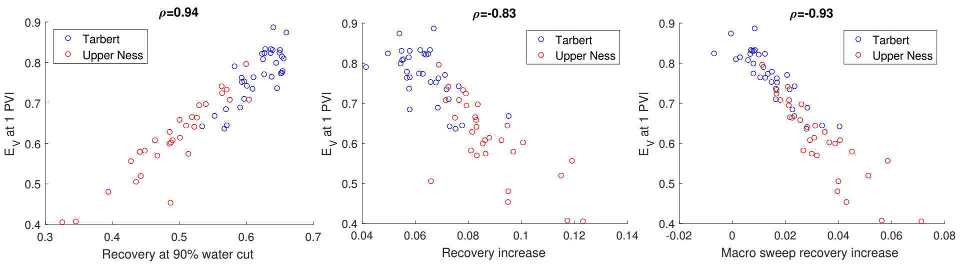


Figure 5.12: Volumetric sweep efficiency at 1 pore volume injected vs recovery.

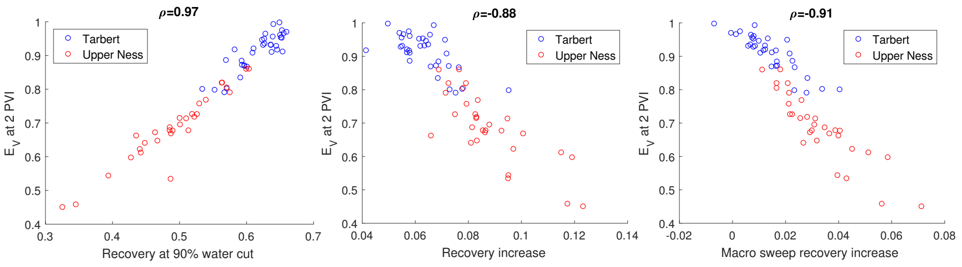


Figure 5.13: Volumetric sweep efficiency at 2 pore volumes injected vs recovery.

5.4.2 New Potential Heterogeneity Measures

Figure 5.14 shows a cross-plot of the inverse of effective constriction factor vs the effective tortuosity factor (column 5 and 6 of Table 4.2). A clear separation between the two formations can be observed. We can also observe that the permeability reduction due to constriction is larger than the reduction due to increased tortuosity for all layers (keep in mind that a lower τ_e^2 means more tortuous). There is no clear correlation between tortuosity and constriction for the Tarbert layers, while there is an observable correlation between tortuosity and constriction for the more tortuous Upper Ness layers.

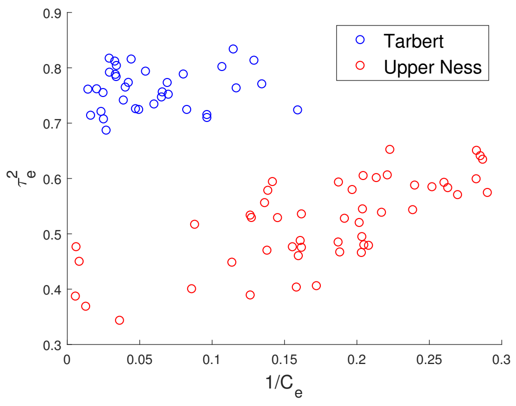


Figure 5.14: Cross-plot of the inverse of the effective constriction factor $^1/_{C_e}$ vs the effective tortuosity factor τ_e^2 for each layer of the SPE10 model.

In Section 3.3.2, the relation between vorticity and tortuosity was discussed. We saw that tortuosity cannot occur if we do not have vorticity. Figure 5.15 therefore shows a cross-plot of the vorticity index H_V and the effective tortuosity τ_e^2. The observed correlation is very strong. Since the vorticity index is already a well-known heterogeneity, and we also see from Figure 5.14 that the effective tortuosity factor and the inverse constriction can be used to separate the two formations of the SPE10 model, it is not unlikely that the permeability descriptors are related to recovery.

Evaluating the permeability descriptors against recovery factors we see in Figure 5.16 that the effective tortuosity factors are reasonably correlated with recovery, and the effective tortuosity can therefore have a potential as a heterogeneity measure. We also find a weak trend in the weighted variance of tortuosity for each layer in Figure 5.17, but the data spread is significantly larger than for the effective tortuosity factors, and hence the correlation is also worse. The coefficient of variation of tortuosity in Figure 5.18 is a better choice than the variance, but still not as good as the effective tortuosity factors alone. An

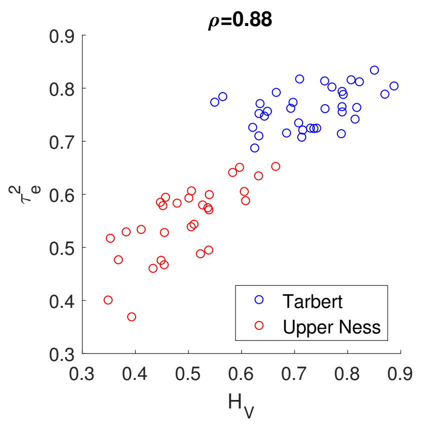


Figure 5.15: Cross-plot of the inverse of the effective constriction factor $^1/C_e$ vs the effective tortuosity factor τ_e^2 for each layer of the SPE10 model.

issue with the tortuosity factor is that for the Tarbert layers marked in blue the correlation is insignificant, while the correlation is better in the channelized Upper Ness layers, which we also saw in Figure 5.14. This is a weakness, as a heterogeneity measure should ideally not be dependent on the reservoir type. The inverse of the effective constriction factor in Figure 5.19 is on the other hand not a reliable measure, which is not very surprising considering that e.g. permeability changes perpendicular on the flow direction can give a considerable constriction increase. An example of this is seen in Figure 4.1 where most streamlines pass almost straight forward through the low permeability region at the bottom, thus keeping the areal sweep. The low permeability region will increase the constriction factor, but not the tortuosity which is governed by the permeability gradient perpendicular on the flow direction (see Equation (3.33)), and hence it is not surprising that the tortuosity correlates better with recovery. The variance of the inverse constriction factor is also not a useful measure as seen in Figure 5.20. Especially the Tarbert layers have almost no variation, which is likely to be related to the fact that the high and low permeability regions in these layers are quite large, which means that a large part of the flow passes through relatively equal conditions. In the Upper Ness layers the permeability channels narrow and changing quickly, as seen in Figure 4.2. It is therefore not surprising that the variance is a lot more significant for these layers for both tortuosity and constriction factors. As neither the inverse constriction factor not its variance shows significant potential as a heterogeneity measure, it is not surprising that the coefficient of variation for the inverse constriction factor in Figure 5.21 also does not correlate well.

The measures based on diffusive time of flight seems to be the best of the new potential measures. The diffusive sweep efficiency at breakthrough of the pressure front in Figure

5.22 has the same correlation coefficient for recovery at 90% water cut as its flow based sibling in Figure 5.10, better correlation with recovery increase due to polymer injection, and slightly worse correlation with recovery increase due to better macro sweep from polymer injection. The diffusive Koval factor in Figure 5.23 is the inverse of the sweep efficiency of breakthrough, and it is therefore not surprising that the same trend is observed here with about the same correlation for 90% water cut, better correlation with recovery increase from polymer injection and slightly worse correlation with recovery increase due to improved macro sweep compared to the flow based Koval factor in Figure 5.9.

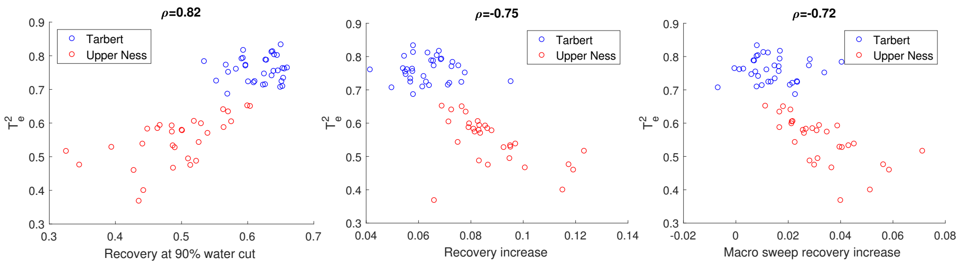


Figure 5.16: Effective tortuosity factor vs recovery.

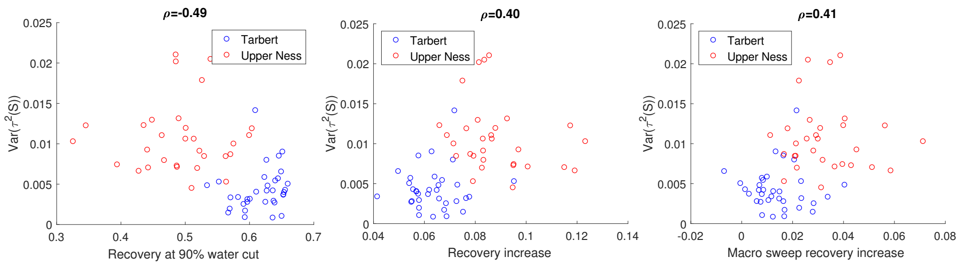


Figure 5.17: Tortuosity variance vs recovery.

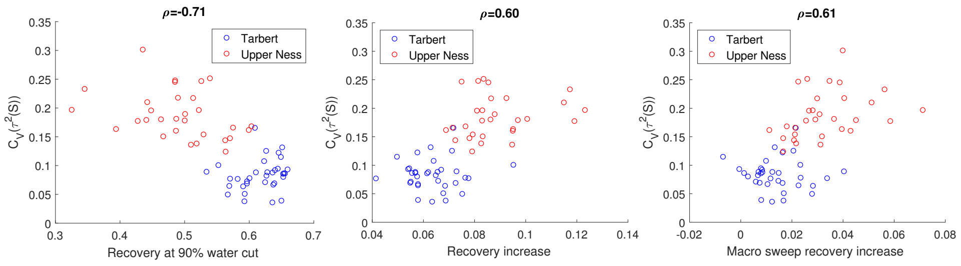


Figure 5.18: Coefficient of variation of tortuosity vs recovery.

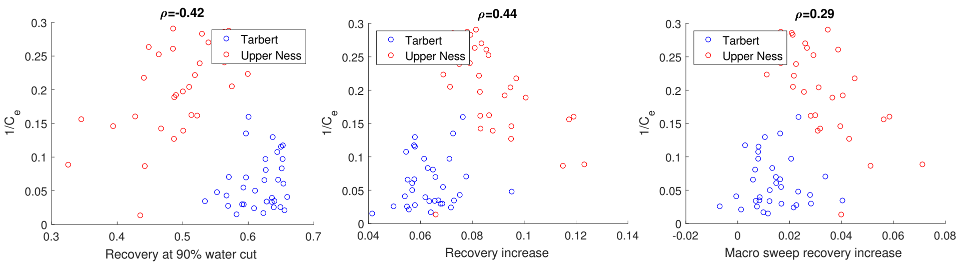


Figure 5.19: Inverse of effective constriction factor vs recovery.

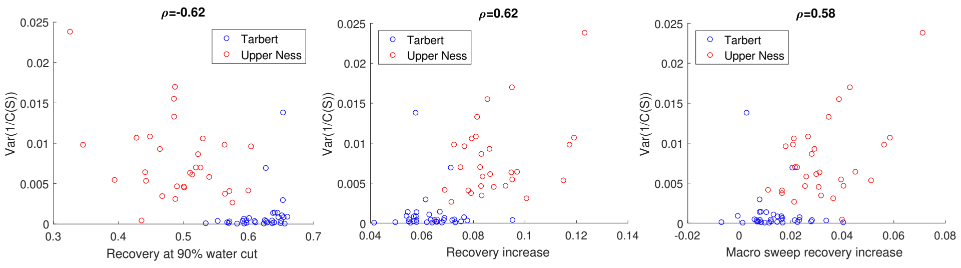


Figure 5.20: Variance of inverse constriction factor vs recovery.

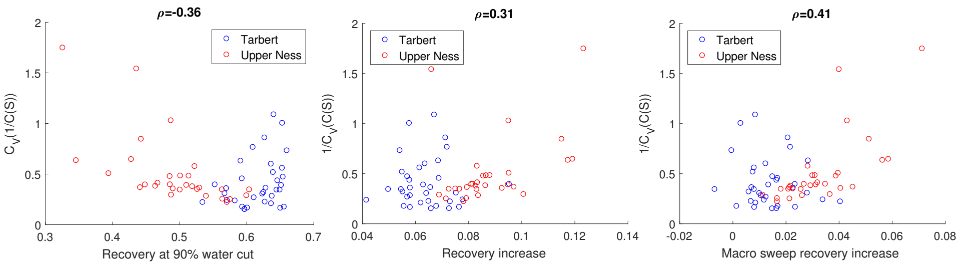


Figure 5.21: Coefficient of variation of inverse of constriction factor vs recovery.

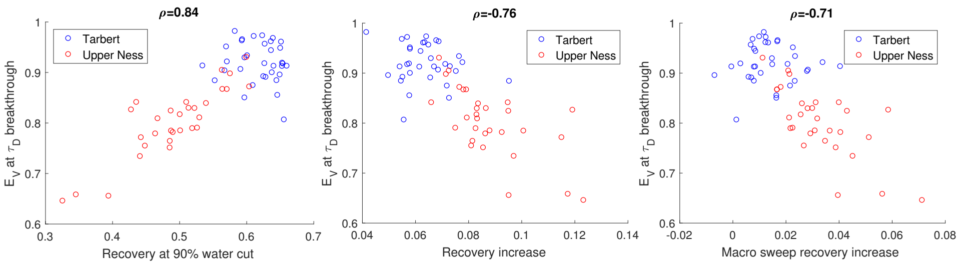


Figure 5.22: Volumetric diffusive sweep efficiency at breakthrough vs recovery.

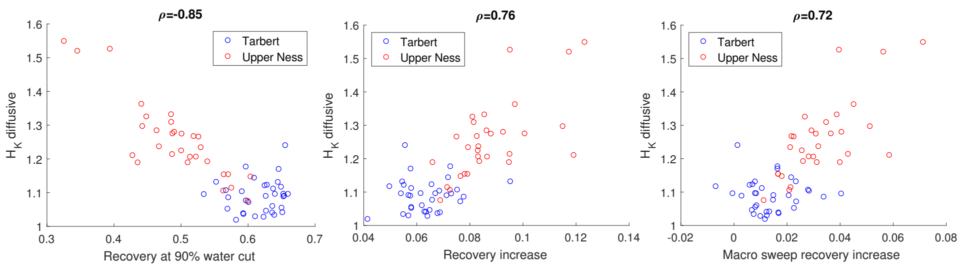


Figure 5.23: Diffusive Koval Factor vs recovery.

Chapter 6

Conclusions

The work of this thesis has been focused on developing new potential measures of heterogeneity, and comparing them with existing measures of heterogeneity. Correlations with recovery factors after waterflooding and tertiary polymer injection have been used as benchmarks for the comparison. An additional aim has been to implement code to generate these measures which is compatible with the open source MRST software, and an improvement of the implementation of a model for permeability description by the use of streamlines.

The main conclusions of the work are:

- Improved implementation of the permeability description model in MRST was achieved, reducing the estimated errors.
- Modifications have been made to the Pollock algorithm in MRST, introducing additional capabilities like pressure gradients, velocity vectors, streamline lengths and the support for periodic grids.
- The Fast Marching Method (FMM) for diffusive time of flight has been implemented, and can be used with existing code in MRST.
- Several heterogeneity measures correlate well with simulated recovery after waterflooding, and with simulated recovery increase due to polymer injection.
- The best correlation between heterogeneity measures and recovery factors were found for volumetric sweep efficiency at 1 and 2 pore volumes injected, while also the Dynamic Lorenz coefficient and vorticity coefficient showed very promising

correlations.

- The diffusive Koval factor was found to be the best heterogeneity measure of the new potential measures that were tested. The correlation was at the same level as the regular Koval factor based on steady-state flow.
- The effective tortuosity factor from the permeability description model showed correlation with recovery, but the correlation was mainly limited to reservoir models with channeling.

Chapter 7

Recommendations for Further Work

Based on the work and findings of this thesis, the following topics can be considered for further work:

- Comparison of heterogeneity measures and recovery factors for other reservoir models, as some of the potential heterogeneity measures showed indications of reservoir dependence (seen as different trends for the Tarbert and Upper Ness formations).
- Use heterogeneity measures in an optimization workflow for reservoir development, where the heterogeneity measures are used in a screening process before more precise simulations.
- The permeability description model, and the extended Pollock algorithm, can potentially be used in upscaling. During upscaling, the permeability of the fine grid is averaged, and the underlying heterogeneity of the fine grid is lost in the upscaled cell. If the permeability descriptors are stored, they could possibly be used on the course grid later. E.g. could the effective tortuosities over the upscaled grid cells be combined to an overall tortuosity factor for a full model.
- Use the FMM implementation of diffusive time of flight in connection with rate and pressure transient analysis, following and extending the works of King et al. (2016) and Li and King (2016). King et al. (2016) uses the diffusive time of flight to connect production data to drainage volume, while Li and King (2016) integrates well test pressure derivative data with geological models.

Bibliography

Bear, J. (1972). *Dynamics of Fluids in Porous Media.* Dover Publications.

Bear, J. and Bachmat, Y. (1967). A generalized theory on hydrodynamic dispersion in porous media. In *IASH Symposium on Artificial Recharge and Management of Aquifers*, volume 72, pages 7–16.

Berg, C. F. (2012). Reexamining archies law conductance description by tortuosity and constriction. *Physical Review E*, 86(4).

Berg, C. F. (2014). Permeability description by characteristic length, tortuosity, constriction and porosity. *Transport in Porous Media*, 103(3):381–400.

Berg, C. F. and Held, R. (2016). Fundamental transport property relations in porous media incorporating detailed pore structure description. *Transport in Porous Media*, 112(2):467–487.

Carman, P. (1937). Fluid flow through granular beds. *Transactions of the Institution of Chemical Engineers*, 15:150–156.

Christie, M. and Blunt, M. (2001). Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Evaluation & Engineering*, 4(04):308–317.

Darcy, H. (1856). *Les Fontaines Publiques de la Ville de Dijon.* Dalmont, Paris, France.

Datta-Gupta, A. and King, M. J. (2007). *Streamline Simulation: Theory and Practice.* Society of Petroleum Engineers, Richardson, Texas, United States of America.

Fatemi, E., Engquist, B., and Osher, S. (1995). Numerical solution of the high frequency asymptotic expansion for the scalar wave equation. *Journal of Computational Physics*, 120(1):145–155.

Heller, J. P. (1963). The interpretation of model experiments for the displacement of fluids through porous media. *AIChE Journal*, 9(4):452–459.

Idrobo, E. A., Choudhary, M. K., and Datta-Gupta, A. (2000). Swept volume calculations and ranking of geostatistical reservoir models using streamline simulation. In *SPE/AAPG Western Regional Meeting*. Society of Petroleum Engineers.

Jensen, J. L., Lake, L. W., Corbett, P. W., and Goggin, D. J. (2000). *Statistics For Petroleum Engineers and Geoscientists*. Elsevier, Amsterdam, Netherlands, 2 edition.

Kamal, M. M. (2009). *Transient Well Testing*. Society of Petroleum Engineers, Richardson, Texas, United States of America.

King, M. J., Wang, Z., and Datta-Gupta, A. (2016). Asymptotic solutions of the diffusivity equation and their applications. In *SPE Europec featured at 78th EAGE Conference and Exhibition*. Society of Petroleum Engineers.

Koval, E. (1963). A method for predicting the performance of unstable miscible displacement in heterogeneous media. *Society of Petroleum Engineers Journal*, 3(02):145–154.

Kreyszig, E. (2011). *Advanced Engineering Mathematics*. John Wiley & Sons, 3 edition.

Krogstad, S., Lie, K.-A., Nilsen, H. M., Berg, C. F., and Kippe, V. (2017). Efficient flow diagnostics proxies for polymer flooding. *Computational Geosciences*, 21(5-6):1203–1218.

Kuchuk, F. J. (2009). Radius of investigation for reserve estimation from pressure transient well tests. In *SPE Middle East Oil and Gas Show and Conference*. Society of Petroleum Engineers.

Kulkarni, K. N., Datta-Gupta, A., and Vasco, D. (2001). A streamline approach for integrating transient pressure data into high-resolution reservoir models. *SPE Journal*, 6(03):273–282.

Lake, L. W. (1989). *Enhanced Oil Recovery*. Prentice Hall, Englewood Cliffs, New Jersey, United States of America.

Lee, W. J. (1982). *Well Testing*. Society of Petroleum Engineers of AIME, Dallas, Texas, United States of America.

Li, C. and King, M. J. (2016). Integration of pressure transient data into reservoir models using the fast marching method. In *SPE Europec featured at 78th EAGE Conference and Exhibition*. Society of Petroleum Engineers.

Lie, K.-A. (2016). *An Introduction to Reservoir Simulation Using MATLAB: User Guide for the Matlab Reservoir Simulation Toolbox (MRST)*. SINTEF ICT, Departement of Applied Mathematics, Oslo, Norway.

Nelson, R. W. (1963). Stream functions for three dimensional flow in heterogeneous porous media. *International Association of Scientific Hydrology*, 64:290–301.

Norwegian Petroleum Directorate (2014). The Brent Group. `http://www.npd.no/no/Publikasjoner/Rapporter/CO2-samleatlas/4-The-Norwegian-North-Sea/41-Geology-of-the-North-Sea/The-Brent-Group/`. Last visited: 02.06.2018.

Nyvoll, A. (2017). Decomposing the effective permeability factor in heterogeneous reservoir models. Spezialization Project.

Pollock, D. W. (1988). Semianalytical computation of path lines for finite-difference models. *Ground Water*, 26(6):743–750.

Rasaei, M. R. and Sahimi, M. (2007). Upscaling and simulation of waterflooding in heterogeneous reservoirs using wavelet transformations: Application to the SPE-10 model. *Transport in Porous Media*, 72(3):311–338.

Rashid, B., Bal, A.-L., Williams, G. J. J., and Muggeridge, A. H. (2012). Using vorticity to quantify the relative importance of heterogeneity, viscosity ratio, gravity and diffusion on oil recovery. *Computational Geosciences*, 16(2):409–422.

Rashid, B., Williams, G., Bal, A.-L., and Muggeridge, A. (2010). Quantifying the impact of permeability heterogeneity on secondary recovery performance. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers.

Sethian, J. A. (1996). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595.

Sethian, J. A. (1999). Fast marching methods. *SIAM review*, 41(2):199–235.

Shook, G. M. and Mitchell, K. M. (2009). A robust measure of heterogeneity for ranking earth models: The f PHI curve and dynamic lorenz coefficient. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers.

Society of Petroleum Engineers (2000). SPE Comparative Solution Project. `http://www.spe.org/web/csp/index.html`. Last visited: 02.06.2018.

Vasco, D. and Datta-Gupta, A. (1999). Asymptotic solutions for solute transport: A formalism for tracer tomography. *Water Resources Research*, 35(1):1–16.

Vasco, D. W., Keers, H., and Karasaki, K. (2000). Estimation of reservoir properties using transient pressure data: An asymptotic approach. *Water Resources Research*, 36(12):3447–3465.

Virieux, J., Flores-Luna, C., and Gibert, D. (1994). Asymptotic theory for diffusive electromagnetic imaging. *Geophysical Journal International*, 119(3):857–868.

Wu, X., Pope, G. A., Shook, G. M., and Srinivasan, S. (2008). Prediction of enthalpy production from fractured geothermal reservoirs using partitioning tracers. *International Journal of Heat and Mass Transfer*, 51(5):1453 – 1466.

Wyckoff, R. D. and Botset, H. G. (1936). The flow of gas-liquid mixtures through unconsolidated sands. *Physics*, 7(9):325–345.

Zhang, Y., Yang, C., King, M. J., and Datta-Gupta, A. (2013). Fast-marching methods for complex grids and anisotropic permeabilities: Application to unconventional reservoirs. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers.

Appendix A

MRST Code

The code given here is tested with Matlab R2017a and MRST 2017b, and is not guaranteed to work on other versions of the softwares, as the code use already existing functions of both Matlab and MRST. MRST can be downloaded from `http://www.sintef.no/projectweb/mrst/downloadable-resources/download/`.

A.1 Heterogeneity Measure Script

Script that generates heterogeneity measures for Table 4.3, and the ”No Flow Boundaries, Straight Line Length”-columns of Table 4.2. Change ”linelength = ’straight’;” to ”linelength = ’integral’;”, and the same code will generate the ”No Flow Boundaries, Integral Line Length”-columns of Table 4.2.

```
%% User-defined variables
%%
%Streamlines
streamlinesPerCell  = 20;
nsubsteps           = 10;
linelength         = 'straight';
% Boundary Conditions
pIn                 = 500*barsa();
pOut                = 100*barsa();

%Fluid
viscosity           = 1*centi*poise;
density             = 1014*kilogram/meter^3;

% rateLimit: since the solver can give very small positive
% rates in cells that are enclosed by no flow cells, but still has positive
% permeability themselves. eps is the floating point relative accuracy
% of Matlab

rateLimit           = 100*eps; % Cutoff for "active cells"


%% Load necessary MRST modules
%%
mrstModule add spe10 incomp heterogeneity diagnostics
% Heterogeneity is the name I have given the module functions from this
% work

%% Pre-allocating memory for various parameters
%%

Be=zeros(85,1);
Ce=Be;
Te2=Be;
keff=Be;
keffs=keff;
kavg=keff;
kavgact=keff;
varTs2=keff;
varCs=keff;
varInvCs=keff;
Lorenz=keff;
FHI=keff;
Hk=keff;
PVI1=keff;
PVI2=keff;
CvTOF=keff;
tofvar=keff;
Evdtof=keff;
Hkdtof=keff;

for count=1:85  %iterating over all layers in SPE10

%% load SPE10 - data, will attempt to download if not available locally
%%
disp(count)  % Current SPE10 layer displayed in console.
[G, W, rock] = getSPE10setup(count); % Loads layer
rock.perm(rock.poro==0,:)=0; % To fix positive perm in deactivated cells.

%% Constant pressure boundaries: Linedrive bottom-to-top
%%
[nx, ny] = deal(G.cartDims(1), G.cartDims(2));
bottomCells = (1:nx)';
topCells    = (1:nx)' + nx*(ny-1);
bc = pside([],G,'YMin',pIn);
bc = pside(bc,G,'YMax',pOut);

%% Computing Transmissibilities, fluid and create initial state
%%
T       = computeTrans(G, rock);
fluid   = initSingleFluid('mu' ,    viscosity, ...
                          'rho', density);
```

```
 initState = initResSol(G, 0.0);
%% Solve incompTPFA to get steady-state pressure solution
%%

 sol = incompTPFA(initState, G, T, fluid, 'bc', bc);

%% Code that validates solution and generates plots of data
%%
sol.flux(isnan(sol.flux))=0;    %Fix NaN
cellNo = rldecode(1:G.cells.num, diff(G.cells.facePos), 2) .';
   cf     = G.cells.faces;
   flux= accumarray([cellNo, cf(:,2)], sol.flux(cf(:,1)));
   clear cf cellNo
   flux(:,2:2:end)=-1*flux(:,2:2:end);
cellFluxSum=sum(flux,2);
cellFlux=sum(flux.*(flux>0),2);


%% Compute start positions for streamlines
startCells = repmat(bottomCells,1,streamlinesPerCell);
startCells = reshape(startCells',[],1);
startPosLoc = [zeros(length(startCells),2), ones(length(startCells),1).*0.5];
startStep = 1/streamlinesPerCell; % Streamline spacing in start cell
                                  % in unit cell coordinates for pollockMod()
first = startStep/2;           % Unit cell coordinate first streamline in cell
last = 1-first;                % Unit cell coordinate first streamline in cell
startPosLoc(:,1)=repmat((first:startStep:last)',length(bottomCells),1);
startPos = [startCells, startPosLoc];

% Finds active start cells
active=cellFlux(startCells)>rateLimit;

[S, T, C,LS,VXYZ,GradP]=pollockMod(G, sol,rock,startPos(active,:), ...
    'substeps', nsubsteps ,'maxsteps', 4e5,'fluid',fluid, 'lineLength',linelength);

% Rate for each streamtube around streamlines:
Qs=ones(size(S)).*flux(startCells(active),3)./streamlinesPerCell;

% Total rate
Q=sum(Qs);


% Effective bulk volume
Ve=sum(G.cells.volumes(cellFlux>rateLimit));

% Pore Volume
OmegaS=sum(sum(poreVolume(G,rock)));

% Find number of streamlines started in active cells:
nstreamlines=length(S);
BS=zeros(nstreamlines,1);
CS=BS;
TauS2=BS;

for i=1:nstreamlines
   % Streamline/streamtube permeability descriptors
   [BS(i), CS(i), TauS2(i)]=streamtubePermDesc(GradP{i}, LS{i}, fluid, 670.56);
   %670.56 is the length of each spe10 layer
end
tof=zeros(length(T),1);
for i = 1 : length(T)
    tof(i,1)=sum(T{i}(:)); % Time of flight /residence time of each streamline
end

%% Heterogeneity Measures
%%
[tofsort,order]=sort(tof);      % Sort residence time for F-Phi curve
pvstrlines=tofsort.*Qs(order);  %pore volume of each streamtube
Phi=cumsum(pvstrlines);
Phi=Phi./Phi(end);              % Normalized storage capacity
F=cumsum(Qs(order));
F=F./F(end);                    % Normalize flow capacity
% Dynamic Lorenz (See Shook and Mitchell (2009), SPE 124625)
Lorenz(count)=computeLorenz(F,Phi);
% End Lorenz

% FHI (See Shook and Mitchell (2009), SPE 124625)
wavgtof=sum(tofsort.*Qs(order))./Q; % Rate weighted tof average
[vl,ix]=min(abs(tofsort-wavgtof)); % Find streamline with tof closest to avg
FHI(count)=F(ix)./Phi(ix); % Calculate FHI
% End FHI

%Hk (see 'Statistics For Petroleum Engineers and Geoscientists' Jensen et
%al. (2000))
tofbreak=min(tof); % Time of breakthrough
Hk(count)=OmegaS./(tofbreak.*Q);
%End Hk

%CvTof (See Shook and Mitchell (2009), SPE 124625)
tofvar(count)=sum(Qs.*(tof-wavgtof).^2)./Q;
CvTOF(count)=sqrt(tofvar(count))./wavgtof;
%end CvTOF

%Ev at 1 and 2 PVI (see Idrobo et al. (2000), SPE 62557 and Shook and
%Mitchell (2009), SPE 124625)
tof1pvi=OmegaS./Q;
PVI1(count)=(sum(tof(tof<tof1pvi).*Qs(tof<tof1pvi))...
    +sum(tof1pvi.*Qs(tof>=tof1pvi)))./OmegaS;

tof2pvi=2*OmegaS./Q;
PVI2(count)=(sum(tof(tof<tof2pvi).*Qs(tof<tof2pvi))...
    +sum(tof2pvi.*Qs(tof>=tof2pvi)))./OmegaS;
%end Ev at 1 and 2 PVI

%DTOF
dtof=computeDTOF(G,rock,fluid,4.4e-10,startCells,[6.096,3.048,0.6096]);
[dtofbreak,ix]=min(dtof(topCells));
pv=poreVolume(G,rock);
Evdtof(count)=sum(pv(dtof<=dtofbreak))/sum(pv); %Ev at breakthrough
Hkdtof(count)=1./Evdtof(count);                 %Diffusive Hk
%End DTOF

V=sum(G.cells.volumes);              % Bulk volume of model
%ks(count)=Q./V.*KS{count}(1);       % ke calculated with Q*K(S)/V
```

```

% Effective permeability descriptors
[Be(count),Ce(count),Te2(count),varTs2(count),varInvCs(count)]= ...
    effPermDesc(BS,CS,TauS2, Qs, Ve);

%ke calculated with Darcy over full layer
keff(count)=-Q*fluid.properties(1)*670.56/...
    (-4e7*sum(G.faces.areas(13421:13480)));
%Average k_h in V (k_x=k_y=k_h)
kavg(count)=mean(rock.perm(:,2));
%Average k_h in Ve (for spe10: Should equal Be)
kavgact(count)=mean(rock.perm(cellFlux>rateLimit,2));
%ke calculated with effective descriptors from streamlines
keffs(count)=Te2(count)*Be(count)*Ve/(Ce(count)*V);
end
%Coefficient of tortuosity variance
CvTs2=sqrt(varTs2)./Te2;
%Coefficient of inverse constriction variance
CvInvCs=sqrt(varInvCs).*Ce;
%Diffusive volumetric sweep efficiency at breakthrough
Evb=1./Hk;
```

A.2 Heterogeneity Measure Script, Periodic Grid

Almost equal to the previous script, but with periodic boundary condition, and 'integral' length setting.

```
%% User-defined variables
%%
tic
%Streamlines
streamlinesPerCell  = 20;
nsubsteps           = 10;
linelength         = 'integral';
% Boundary Conditions
pIn                 = 500*barsa();
pOut                = 100*barsa();

%Fluid
viscosity           = 1*centi*poise;
density             = 1014*kilogram/meter^3;

% rateLimit: since the solver can give very small positive
% rates in cells that are enclosed by no flow cells, but still has positive
% permeability themselves. eps is the floating point relative accuracy
% of Matlab

rateLimit           = 100*eps; % Cutoff for "active cells"

%% Load necessary MRST modules
%%
mrstModule add spe10 incomp heterogeneity diagnostics upscaling
```

```
% Heterogeneity is the name I have given the module functions from this
% work

%% Pre-allocating memory for various parameters
%%

Be=zeros(85,1);
Ce=Be;
Te2=Be;
keff=Be;
keffs=keff;
kavg=keff;
kavgact=keff;
varTs2=keff;
varCs=keff;
varInvCs=keff;
Lorenz=keff;
FHI=keff;
Hk=keff;
PVI1=keff;
PVI2=keff;
CvTOF=keff;
tofvar=keff;
Evdtof=keff;
Hkdtof=keff;

for count=1:85  %iterating over all layers in SPE10

close all % Closing existing figures
%% load SPE10 - data, will attempt to download if not available locally
%%
disp(count)  % Current SPE10 layer
[G, W, rock] = getSPE10setup(count); % Loads layer
rock.perm(rock.poro==0,:)=0; % To fix positive perm in deactivated cells.
%
%% Constant pressure boundaries: Linedrive bottom-to-top
%%
bcr{1}=pside([],G,'RIGHT',0);   bcl{1}=pside([],G,'LEFT',0);

%% Constant pressure boundaries: Linedrive bottom-to-top
%%
[nx, ny] = deal(G.cartDims(1), G.cartDims(2));
bottomCells = (1:nx)';
topCells    = (1:nx)' + nx*(ny-1);
[Gp, bcp]=makePeriodicGridMulti3d(G, bcl, bcr, {0});
bc = pside([],Gp,'YMin',pIn);
bc = pside(bc,Gp,'YMax',pOut);


%% Computing Transmissibilities, fluid and create initial state
%%
T       = computeTransGp(G, Gp, rock);
fluid   = initSingleFluid('mu' ,    viscosity, ...
                           'rho', density);
initState = initResSol(Gp, 0.0);
%% Solve incompTPFA to get steady-state pressure solution
%%
```

```

 sol = incompTPFA(initState, Gp, T, fluid, 'bc', bc,'bcp',bcp);

%% Code that validates solution and generates plots of data
%%
sol.flux(isnan(sol.flux))=0;    %Fix NaN

cellNo = rldecode(1:Gp.cells.num, diff(Gp.cells.facePos), 2) .';
   cf     = Gp.cells.faces;
   flux= accumarray([cellNo, cf(:,2)], sol.flux(cf(:,1)));
   clear cf cellNo
   flux(:,2:2:end)=-1*flux(:,2:2:end);
cellFluxSum=sum(flux,2);
cellFlux=sum(flux.*(flux>0),2);


%% Compute start positions for streamlines
startCells = repmat(bottomCells,1,streamlinesPerCell);
startCells = reshape(startCells',[],1);
startPosLoc = [zeros(length(startCells),2), ones(length(startCells),1).*0.5];
startStep = 1/streamlinesPerCell; % Streamline spacing in start cell
                                  % in unit cell coordinates for pollock()
first = startStep/2;       % Unit cell coordinate first streamline in cell
last = 1-first;            % Unit cell coordinate first streamline in cell
startPosLoc(:,1)=repmat((first:startStep:last)',length(bottomCells),1);
startPos = [startCells, startPosLoc];

% Finds active start cells
active=cellFlux(startCells)>rateLimit;

[S, T, C,LS,VXYZ,GradP]=pollockMod(G, sol,rock,startPos(active,:), ...
    'substeps', nsubsteps ,'maxsteps', 4e5,'fluid',fluid, 'periodic', Gp, 'lineLength
        ',linelength);


% Rate for each streamtube around streamlines:
Qs=ones(size(S)).*flux(startCells(active),3)./streamlinesPerCell;

% Total rate
Q=sum(Qs);


% Effective bulk volume
Ve=sum(G.cells.volumes(cellFlux>rateLimit));

% Pore Volume
OmegaS=sum(sum(poreVolume(G,rock)));

% Find number of streamlines started in active cells:
nstreamlines=length(S);
BS=zeros(nstreamlines,1);
CS=BS;
TauS2=BS;

for i=1:nstreamlines
    % Streamline/streamtube permeability descriptors
   [BS(i), CS(i), TauS2(i)]=streamtubePermDesc(GradP{i}, LS{i}, fluid, 670.56);
   %670.56 is the length of each spe10 layer
end
tof=zeros(length(T),1);
for i = 1 : length(T)
    tof(i,1)=sum(T{i}(:));
end

%% Heterogeneity Measures
%%
[tofsort,order]=sort(tof);      % Sort residence time for F-Phi curve
pvstrlines=tofsort.*Qs(order);  %pore volume of each streamtube
Phi=cumsum(pvstrlines);
Phi=Phi./Phi(end);              % Normalized storage capacity
F=cumsum(Qs(order));
F=F./F(end);                    % Normalize flow capacity
% Dynamic Lorenz (See Shook and Mitchell (2009), SPE 124625)
Lorenz(count)=computeLorenz(F,Phi);
% End Lorenz

% FHI (See Shook and Mitchell (2009), SPE 124625)
wavgtof=sum(tofsort.*Qs(order))./Q; % Rate weighted tof average
[vl,ix]=min(abs(tofsort-wavgtof)); % Find streamline with tof closest to avg
FHI(count)=F(ix)./Phi(ix); % Calculate FHI
% End FHI

%Hk (see 'Statistics For Petroleum Engineers and Geoscientists' Jensen et
%al. (2000))
tofbreak=min(tof); % Time of breakthrough
Hk(count)=OmegaS./(tofbreak.*Q);
%End Hk

%CvTof (See Shook and Mitchell (2009), SPE 124625)
tofvar(count)=sum(Qs.*(tof-wavgtof).^2)./Q;
CvTOF(count)=sqrt(tofvar(count))./wavgtof;
%end CvTOF

%Ev at 1 and 2 PVI (see Idrobo et al. (2000), SPE 62557 and Shook and
%Mitchell (2009), SPE 124625)
tof1pvi=OmegaS./Q;
PVI1(count)=(sum(tof(tof<tof1pvi).*Qs(tof<tof1pvi))...
    +sum(tof1pvi.*Qs(tof>=tof1pvi)))./OmegaS;

tof2pvi=2*OmegaS./Q;
PVI2(count)=(sum(tof(tof<tof2pvi).*Qs(tof<tof2pvi))...
    +sum(tof2pvi.*Qs(tof>=tof2pvi)))./OmegaS;
%end Ev at 1 and 2 PVI

%DTOF
dtof=computeDTOF(G,rock,fluid,4.4e-10,startCells,[6.096,3.048,0.6096]);
[dtofbreak,ix]=min(dtof(topCells));
pv=poreVolume(G,rock);
Evdtof(count)=sum(pv(dtof<=dtofbreak))/sum(pv); %Ev at breakthrough
Hkdtof(count)=1./Evdtof(count);                 %Diffusive Hk
%End DTOF



```

```
V=sum(G.cells.volumes);                  % Bulk volume of model
%ks(count)=Q./V.*KS{count}(1);           % ke calculated with Q*K(S)/V

% Effective permeability descriptors
[Be(count),Ce(count),Te2(count),varTs2(count),varInvCs(count)]= ...
    effPermDesc(BS,CS,TauS2, Qs, Ve);

%ke calculated with Darcy over full layer
keff(count)=-Q*fluid.properties(1)*670.56/...
    (-4e7*sum(G.faces.areas(13421:13480)));
%Average k_h in V (k_x=k_y=k_h)
kavg(count)=mean(rock.perm(:,2));
%Average k_h in Ve (for spe10: Should equal Be)
kavgact(count)=mean(rock.perm(cellFlux>rateLimit,2));
%ke calculated with effective descriptors from streamlines
keffs(count)=Te2(count)*Be(count)*Ve/(Ce(count)*V);
end
%Coefficient of tortuosity variance
CvTs2=sqrt(varTs2)./Te2;
%Coefficient of inverse constriction variance
CvInvCs=sqrt(varInvCs).*Ce;
%Diffusive volumetric sweep efficiency at breakthrough
Evb=1./Hk;
```

A.3 Extended Pollock Algorithm

This is a modified version of the already existing pollock() method in MRST, which includes new capabilities presented in this thesis. This code is for the 1D Darcy pressure gradient approximation presented in Section 3.3, which is the recommended choice of the three methods presented in this work. The less recommended alternatives are shown in Appendix B.

```
function varargout = pollockMod(G,state, rock, varargin)
% Trace streamlines in logically Cartesian grid using Pollock approximation.
% In addition to the regular Pollock approximation, pressure gradients are
% supported.
%
%
% SYNOPSIS:
%   [S,T,C,L,V,GP] = pollockMod(G, state, rock)
%   [S,T,C,L,V,GP] = pollockMod(G, state, rock, startpos)
%   [S,T,C,L,V,GP] = pollockMod(G, state, rock, 'pn', pv, ...)
%   [S,T,C,L,V,GP] = pollockMod(G, state, rock, startpos, 'pn', pv, ...)
%
% PARAMETERS:
%
%   G         - Cartesian or logically Cartesian grid.
%
%   state     - State structure with field 'flux'.
%
%   rock      - Rock structure with the field 'poro'.
```

```
%
% OPTIONAL PARAMETERS
%
%   positions - Matrix of size (N, 1) or (N, d+1), where d is the dimension
%               of the grid, used to indicate where the streamlines should
%               start.
%
%               If the size is (N, 1), positions contains the cell indices
%               in which streamlines should start. Each streamline is
%               started in the the local coordinate (0.5, 0.5, ...). To be
%               precise, this is the mean of the corner points, not the
%               centroid of the cell.
%
%               If the size is (N, d+1), the first column contains cell
%               indices, and the d next columns contain the local
%               coordinates at which to start streamlines.
%
% OPTIONAL PARAMETERS (supplied in 'key'/value pairs ('pn'/pv ...)):
%
%   substeps   - Number of substeps in each cell, to improve visual quality.
%               Default 5.
%
%   maxsteps   - Maximal number of points in a streamline.
%               Default 1000.
%
%   reverse    - Reverse velocity field before tracing.
%               Default false.
%   periodic   - Grid with periodic boundary conditions.
%               Default not periodic boundary conditions.
%
%   fluid      - MRST fluid structure where fluid.properties(1)=viscosity
%               Default viscosity: 1 cP
%
%   lineLength - Method for streamline length calculation. Options:
%                'straight' (straight line between plots) and 'integral'.
%                Default: 'straight'
%
%
% RETURNS:
%
%  S      - Cell array of individual streamlines suitable for calls like
%           streamline(pollock(...)) and streamtube(pollock(...)).
%
%  T      - Time-of-flight of coordinate.
%
%  C      - Cell number of streamline segment, i.e, line segment between
%           two streamline coordinates.
%
%  L      - Streamline lengths
%
%  V      - Velocity vectors
%
%  GP     - Pressure gradients
%
% EXAMPLE:
%
%   [S,T,C,L,V,GP] = pollockMod(G, x, rock,startpos,'fluid',fluid);
```

```
%
%   streamline(S);
% SEE ALSO: pollock()
%
%
%{
ORIGINAL COPYRIGHT FROM MRST:

Copyright 2009-2017 SINTEF ICT, Applied Mathematics.

This file is part of The MATLAB Reservoir Simulation Toolbox (MRST).

MRST is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

MRST is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with MRST.  If not, see <http://www.gnu.org/licenses/>.
%}

% Written by Jostein R. Natvig, SINTEF ICT, 2010.
%
% Modified by Asgeir Nyvoll, MSc student NTNU, 2018

   d = size(G.nodes.coords, 2);

   if mod(length(varargin),2)==0
      positions   = [(1:G.cells.num)', repmat(0.5, [G.cells.num, d])];
   else
      positions = varargin{1};
      if size(positions, 2) ==1
         positions = [positions, repmat(0.5, [size(positions, 1), d])];
      elseif size(positions, 2) ~= 1 + d
         error('Expected array of local positions of width 1 or 1+d.');
      end
      varargin  = varargin(2:end);
   end

   opt = struct('substeps', 5, 'maxsteps', 1000, 'reverse', false, 'periodic',...
       [], 'fluid', [],'lineLength','straight');
   opt = merge_options(opt, varargin{:});

% Check if fluid properties are given for Darcy calculation of pressure gradient
    if isempty(opt.fluid)
        opt.fluid.properties(1)=1*centi*poise; %default value
    end


% Check that streamline length is calculated with integral of inverse
% velocity in case of periodic grid.
        if ~isempty(opt.periodic) && ~isequal(opt.lineLength,'integral')
        warning('Length calculation changed: Periodic grids only support integral')
        opt.lineLength='integral';
        end

   if size(state.flux, 2) > 1
        state.flux = sum(state.flux, 2);
   end


   if opt.reverse
      state.flux = -state.flux;
   end

   [varargout{1:nargout}] = trace(G, state, positions, rock, opt);

end



% ========================================================================
function varargout = trace(G, state, pos, rock, opt)

if isempty(opt.periodic)
        %State grid and coordinate grid are equal when grid is not periodic.
        Gp=G;
else
        Gp=opt.periodic;
end

   d               = size(G.nodes.coords, 2);
   numStreamlines = size(pos,1);
   assert(size(pos, 2) == d+1);

   if ~isfield(G, 'cellNodes')
      cn   = cellNodes(G);
      G.cellNodes = accumarray(cn(:,1:2), cn(:,3));
   end


   % Make array face fluxes for each cell in grid
   cellNo = rldecode(1:Gp.cells.num, diff(Gp.cells.facePos), 2) .';
   cf     = Gp.cells.faces;
   pv     = poreVolume(Gp,rock);
   flux   = accumarray([cellNo, cf(:,2)], state.flux(cf(:,1)));

   %Particle velocity in unit cell
   unitVelo    = flux./pv;
   %Remove NaN cases:
   unitVelo(pv==0,:)=0;

   %Actual velocity
   velo1=state.flux./Gp.faces.areas;
   velo   = accumarray([cellNo, cf(:,2)],velo1(cf(:,1)))./rock.poro;
   velo(pv==0,:)=0;
   clear cf cellNo pv flux velo1

   neighbors  = findNeighbors(Gp);
```

```

magic  = 1000;
XYZ    = nan(numStreamlines, d, magic);
VXYZ   = nan(numStreamlines, d, magic);
T      = nan(numStreamlines, magic);
C      = nan(numStreamlines, magic);
Ls     = nan(numStreamlines, magic);
GradP  = nan(numStreamlines, magic);

active = true(numStreamlines, 1);

% Store initial values
[XYZ(active,:,1)] = globalCoordinate(G, pos(active,1), pos(active, 2:end));
[VXYZ(active,:,1)] = globalVelocity(pos(active,1), pos(active, 2:end), velo);
T(active, 1) = zeros(sum(active), 1);
Ls(active, 1) = zeros(sum(active), 1);
C(active, 1) = pos(active, 1);
GradP(active, 1) = zeros(sum(active), 1);

i = 2;
while any(active)
   % Realloc

   if i+opt.substeps+1 > size(XYZ, 3)
      magic = max(magic, opt.substeps+1);
      XYZ   = cat(3, XYZ,    nan(numStreamlines, d, magic));
      VXYZ  = cat(3, VXYZ,   nan(numStreamlines, d, magic));
      T     = cat(2, T,      nan(numStreamlines, magic));
      C     = cat(2, C,      nan(numStreamlines, magic));
      Ls    = cat(2, Ls,     nan(numStreamlines, magic));
      GradP = cat(2, GradP,  nan(numStreamlines, magic));
   end
   current_cell = pos(active,1);
   C(active, i-1+(1:opt.substeps)) = repmat(pos(active, 1), [1, opt.substeps]);
   % Take another pollock step
   [pos(active,:), t, xyz, VXYZ(active, :, i:i+opt.substeps-1), ...
        Ls(active,i:i+opt.substeps-1), ...
        GradP(active,i:i+opt.substeps-1)] = ...
        step(rock, opt.fluid, pos(active,:), unitVelo, ...
        velo, neighbors, opt.substeps);
   T(active, i-1+(1:opt.substeps)) = repmat(t/opt.substeps, [1, opt.substeps]);

   %Store coordinates
   for k=1:opt.substeps

     %Global coordinate (using coordinate grid G for periodic grid Gp)
     [XYZ(active, :, i+k-1)] = globalCoordinate(G, current_cell, xyz(:,:,k));

     %Streamline step length (various options)
     if isequal(opt.lineLength,'straight')
         %Straight line between neighboring coordinates
         [Ls(active, i+k-1)] = ...
             (sqrt(sum((XYZ(active, :, i+k-1)-XYZ(active, :, i+k-2)).^2,2)));

     elseif isequal(opt.lineLength,'integral')
        continue; %Already done in step method
     else
```

```
            error('Invalid streamline length method. Choose straight or integral');
        end

    end


    % Update active flag
    active(active)   =  pos(active,1) ~= current_cell;

    i = i+opt.substeps;
    %Break if reaching maxsteps. Display warning.
    if i > opt.maxsteps,warning('Maxsteps reached'), break;end
end

%% Pack coordinates in list with streamlines separated by NaN.
p = reshape(permute(XYZ, [3,1,2]), [], d);

i = ~isnan(p(:,1));
j = i|[true;i(1:end-1)];
p = p(j,:);

% Pack streamline coordinates in a cell array suitable for use with
% Matlab streamline, i.e., as in 'streamline(pollock(G, resSol));'
flag = isnan(p(:,1));
ix = find(flag);
dd  = diff([0;ix])-1;
varargout{1} = mat2cell(p(~flag,:), dd, d);
% Pack times of flight.
if nargout > 1
   T = reshape(T', [], 1);
   T = T(j);
   varargout{2} = mat2cell(T(~flag), dd, 1);
end
% Pack cells.
if nargout > 2
   C = reshape(C', [], 1);
   C = C(j);
   varargout{3} = mat2cell(C(~flag), dd, 1);
end

% Pack step lengths.
if nargout > 3
   Ls = reshape(Ls', [], 1);
   Ls = Ls(j);
   varargout{4} = mat2cell(Ls(~flag), dd, 1);
end

% Pack velocities.
if nargout>4
v = reshape(permute(VXYZ, [3,1,2]), [], d);

i = ~isnan(v(:,1));
j = i|[true;i(1:end-1)];
v = v(j,:);

flag = isnan(v(:,1));
ix = find(flag);
```

```
   dd   = diff([0;ix])-1;
   varargout{5} = mat2cell(v(~flag,:), dd, d);
   end

   % Pack pressures gradients.
   if nargout > 5
     GradP = reshape(GradP', [], 1);
     GradP = GradP(j);
     varargout{6} = mat2cell(GradP(~flag), dd, 1);
   end

end



% ======================================================================
function xyz = globalCoordinate(G, c, p)
% Compute global coordinate corresponding to local coorinate p in cells c
% p  - local positions == [xi,eta,zeta] in 3D
% c  -
%
   if numel(c)==1, p = reshape(p, 1, []); end
   % Compute node weight for quadrilateral or hexahedron
   d = size(G.nodes.coords, 2);
   w = ones(size(p,1), 2^d);
   for i=1:d
      mask        = logical(bitget((0:2^d-1)', i));
      w(:, mask)  = w(:, mask).* repmat(  p(:,i), [1, sum( mask)]);
      w(:,~mask)  = w(:,~mask).* repmat(1-p(:,i), [1, sum(~mask)]);
   end

   % Compute weighted average of corner points
   xyz = zeros(size(p,1), d);
   for i=1:d
      xi       = G.nodes.coords(:,i);
      xyz(:,i) = sum( w.*reshape(xi(G.cellNodes(c, :))', 2^d, [])', 2);
   end
end


function vxyz = globalVelocity(c, p, v)
% Return velocity vector at each substep.

d = size(v,2)/2;
w = size(c,1);
vxyz=zeros(w,d);

    for i=1:d
        %Linear interpolation internally in unit cell
        vxyz(:,i) = (v(c,2*i)-v(c,2*i-1)).*p(:,i)+v(c,2*i-1);

    end
end
%% ======================================================================
function [pos, tof, xyz, vxyz, ds, gp] = ...
    step(rock, fluid, pos, unitflux, flux, neighbors, nsubsteps)
% Update pos array by computing new local coordinate and new cell.
% In addition, compute curve within cell, step lengths, time of flight
% and pressure gradients.
%
%
   uf = unitflux(pos(:,1),:);   % velocity unit cell
   f  = flux(pos(:,1),:);       % velocity
   n = neighbors(pos(:,1),:);
   dims = size(pos, 2)-1;
   T    = nan(size(pos,1),dims);
   for i=1:dims
      T(:,i) = computeTime(pos(:,1+i), uf(:,2*i-1:2*i));
   end
   [tof, dir] = min(T, [], 2);
   % Compute positions, velocities, lengths and pressure gradients
   [xyz, d, vxyz, ds, gp] = ...
       computePosVelLenGrad(rock, fluid, pos, uf, tof, f, nsubsteps);
   pos (:,2:end) = xyz(:,:,end);

   % Find direction to look up neighbor cell
   k  = 2*(dir-1)+d(sub2ind([numel(dir), 3], (1:numel(dir))', dir));
   t  = sub2ind(size(n), (1:numel(k))', k);

   % Update cell number if NOT at boundary.
   % IF at boundary, mark dir with NaN to avoid changing local coordinate
   % below.
   ind          = n(t)==0;
   pos(~ind,1) = n(t(~ind));
   dir (ind)    = nan;

   % Change local coordinate when moving to new cell
   k = sub2ind(size(d), (1:size(dir,1))', dir);
   k = k(~isnan(k));
   pos(numel(dir) + k ) = 2-d(k);
end


% ======================================================================
function N = findNeighbors(G)
% Build (n x 2*d) -array of neighbors for each cell in (Cartesian) grid G.
   cellNo = rldecode(1:G.cells.num, diff(G.cells.facePos), 2)';
   col    = 1 +   (cellNo == G.faces.neighbors(G.cells.faces(:,1), 1));
   c      = G.faces.neighbors(double(G.cells.faces(:,1)) + G.faces.num* (col-1));
   N      = accumarray([cellNo, G.cells.faces(:,2)], c);
end


% ======================================================================
function t = computeTime(xi, v)
% Compute time needed to reach xi=0 or xi=1 given velocities v=[v1,v2] at
% xi=0 and xi=1.  The formula is
%
%   t = xi/ui  or t = (1-xi)/ui,    if v1 = v2 = ui, and
%
%   t = 1/(v2-v1)*log(ue/ui),       otherwise
%
% where ui=v2*xi+v1*(1-xi) is the velocity at xi, and ue=v2 if ui>0 or
% ue=v1 if ui<0.
```

```
   tolerance = 100*eps;

   ui          = v(:,1) + xi.*diff(v, 1, 2);%(:,2)-v(:,1));
   ue          = v(:,    2);
   ue (ui<0)   = v(ui<0, 1);
   arg         = ue./ui;
   t           = inf(size(xi));

   % Linear velocity
   ind         = abs(diff(v, 1, 2)) > tolerance*abs(v(:,1));
   t(ind,:)    = 1./diff(v(ind,:), 1, 2).*log(arg(ind,:));

   % Constant velocity
   ds          = -xi;
   ds(ui > 0)  = 1-xi(ui>0);
   t(~ind)     = ds(~ind)./ui(~ind);

   % nan happens for ui=ui=0
   t(arg<0 | isnan(arg))    = inf;
end


% ==========================================================================
function [xyz, d,vxyz,ds,gp] = ...
    computePosVelLenGrad(rock, fluid, pos, uf, tof, f, nsubsteps)
% Compute position at time t given start point xi and velocities v=[v1,v2].
%
%   x = xi + v*t,    if v is constant or
%
%   x = xi + (ui*exp((v2-v1)*t) - ui)/(v2-v1), otherwise
%
   dims = size(pos, 2)-1;
   nel = size(pos, 1);
   xyz = zeros(nel, dims, nsubsteps);
   vxyz= xyz;
   ds  = zeros(nel, nsubsteps);
   gp  = ds;
   d   = zeros(nel, 1);
   du  = zeros(nel,dims);
   dv  = du;
   ui  = du;
   vi  = du;
   dt= tof/nsubsteps;

   for i=1:dims
   du(:,i)= diff(uf(:,2*i-1:2*i), 1, 2);
   dv(:,i)= diff(f(:,2*i-1:2*i), 1, 2);
   ui(:,i)= uf(:,2*i-1) + pos(:,1+i).*du(:,i); %unit cell velocity
   vi(:,i)= f(:,2*i-1) + pos(:,1+i).*dv(:,i);  %velocity
   d(:,i)    = 1 + ~(ui(:,i)<0);
   end

for s=1:nsubsteps
    t = s*dt; %residence time in cell
      for i=1:dims
        tolerance = 100*eps;

        xyz(:,i,s)= inf(size(pos(:,1+i)));

        ind       = abs(du(:,i)) > tolerance*abs(uf(:,2*i-1));

        % linear velocity
        xyz(ind,i,s)    = pos(ind,1+i) + ...
            ( ui(ind,i).*exp(du(ind,i).*t(ind)) - ui(ind,i))./du(ind,i);
        vxyz(ind,i,s)   =(vi(ind,i)).*exp(du(ind,i).*t(ind));


        % Constant velocity
        xyz(~ind,i,s) = pos(~ind,1+i) + uf(~ind,2*i-1).*t(~ind, :);
        xyz(~ind & t==inf,i,s) = pos(~ind & t==inf,1+i);
        vxyz(~ind,i,s)=f(~ind,2*i-1);

    end

    if s==1
        vp=sqrt(sum(vi.^2,2));
        v=sqrt(sum(vxyz(:,:,s).^2,2)); %velocity at end of step
        %mean perm
        k=(sqrt(sum((rock.perm(pos(:,1),:).*vi./vp).^2,2))...
           +sqrt(sum((rock.perm(pos(:,1),:).*vxyz(:,:,s)./v).^2,2)))./2;
    else
        vp=v;
        v=sqrt(sum(vxyz(:,:,s).^2,2)); %velocity at end of step
        %mean perm
        k=(sqrt(sum((rock.perm(pos(:,1),:).*vxyz(:,:,s-1)./vp).^2,2))...
           +sqrt(sum((rock.perm(pos(:,1),:).*vxyz(:,:,s)./v).^2,2)))./2;
    end

        %curve length
        ds(:,s)=dt.*(vp+v)./2;

        %mean pressure gradient
        gp(:,s)= -((v+vp)./2.*rock.poro(pos(:,1))./k).*fluid.properties(1);
end
end
```

A.4 Fast Marching Diffusive Time of Flight

```
function DTOF=computeDTOF(G,rock,fluid,comp,startCells,cellSize)
% Compute diffusive time of flight with Fast Marching Method (FMM). See
% Zhang et al. 2013 (SPE 163637). The diffusive time of flight is related
% to the propagation of a pressure front, similar to the depth of
% investigation.
%
%
% SYNOPSIS:
%   DTOF = computeDTOF(G,rock,fluid,comp,startCells,cellSize)
%
% PARAMETERS:
%
```

```
%   G           - Cartesian or logically Cartesian grid.
%
%   rock        - Rock structure with the fields 'poro' and 'perm'.
%
%   fluid       - MRST fluid structure where fluid.properties(1)=viscosity
%
%   comp        - Total compressibility. Vector of size (1,1) or (N,1)
%                 where N is the number of grid cells.
%
%   startCells  - Cells where DTOF=0;
%
%   cellSize    - Vector of size (1,d) or (N,d) where d is dimension of the
%                 grid and N is the number of cells.
%
% RETURNS:
%
%  DTOF   - Diffusive time of flight
%
%  Written by Asgeir Nyvoll, MSc student NTNU, 2018

nbrs=findNeighbors(G);
DTOF=NaN(G.cells.num,1);
voxelState=zeros(G.cells.num,1); % 0 is far, 1 is evaluated, 2 is accepted
DTOF(startCells)=0;              % initial
voxelState(startCells)=2;        % accept initial
startNbrs=unique(nbrs(voxelState==2,:));
diffu=rock.perm./(rock.poro.*comp.*fluid.properties(1));
slowness=cellSize./sqrt(diffu)./2;
gdim=size(nbrs,2)/2;
if gdim==3
    if all(nbrs(:,5:6)==0)
        gdim=2; %Basically 2D
    end
end
%Preparing first set of neighbors
for n = startNbrs(:).'
    if n>0 && voxelState(n)==0
    voxelState(n)=1;
    DTOF(n)=solveEikonalDTOF(DTOF,voxelState,slowness,nbrs(n,:),n,gdim);
    end
end
% Accepting first neighbor
acceptedValue=min(DTOF(voxelState==1));      % Lowest DTOF is accepted
acceptedNode=find(DTOF==acceptedValue,1);    % Node of accepted DTOF
voxelState(acceptedNode)=2;                  % Update status
unacceptedNbrs=unique(nbrs(acceptedNode,:)); % Neighbors of accepted
unacceptedNbrs=unacceptedNbrs(unacceptedNbrs>0); % Remove boundary
% Remove accepted neighbors
unacceptedNbrs=unacceptedNbrs(voxelState(unacceptedNbrs)~=2);
voxelState(unacceptedNbrs)=1; %update state of unaccepted neighbors

% Iterate until all connected cells are reached
while ~isempty(acceptedNode) && any(voxelState~=2)
    for n=unacceptedNbrs(:).'
        % Update DTOF of unaccepted neighbors of latest accepted node
        DTOF(n)=solveEikonalDTOF(DTOF,voxelState,slowness,nbrs(n,:),n,gdim);
    end
    % Accept next node, find new neighbors, update states
    acceptedValue=min(DTOF(voxelState==1));
    acceptedNode=find(DTOF==acceptedValue);
    voxelState(acceptedNode)=2;
    unacceptedNbrs=unique(nbrs(acceptedNode,:));
    unacceptedNbrs=unacceptedNbrs(unacceptedNbrs>0);
    unacceptedNbrs=unacceptedNbrs(voxelState(unacceptedNbrs)~=2);
    voxelState(unacceptedNbrs)=1;
end
end

function tempDTOF=solveEikonalDTOF(DTOF,voxelState,slowness,nbrs,n,gdim)
%Solves the Eikonal equation with the approximation from Zhang et al.
%
tempDTOF=NaN;

% Evaluate diffusive time of flight from each corner. Smallest value is
% valid
if gdim==3
    for i=1:2
        for j=3:4
            for k=5:6
                tempDTOF=min(singleTestSolver(...
                    DTOF,voxelState,slowness,nbrs,n,[i j k]),tempDTOF);
            end
        end
    end
elseif gdim==2
    for i=1:2
        for j=3:4
                tempDTOF=min(singleTestSolver(...
                DTOF,voxelState,slowness,nbrs,n,[i j]),tempDTOF);
        end
    end
else
    error('Neither 2D nor 3D')
    %Should "never" happen
end
end


function dtof=singleTestSolver(DTOF,voxelState,slowness,nbrs,n,testCase)
%Solves Equations (9) and (10) in Zhang et al. (2013) for one corner
coeff=[0 0 0]; % coeff = [a,b,c] in quadratic equation a*t^2+b*t+c
for i=1:length(testCase)
if (nbrs(testCase(i))>0) %Check that it is not a boundary
    direction=ceil(testCase(i)/2);
    if isnan(slowness(n,direction))
        dtof=NaN;
        return
    end
    if (voxelState(nbrs(testCase(i)))==2 && ...
            ~isnan(slowness(nbrs(testCase(i)),direction)))
        coeff=coeff+1/(slowness(nbrs(testCase(i)),direction)+...
            slowness(n,direction))^2.*...
            [1, -2*DTOF(nbrs(testCase(i))),(DTOF(nbrs(testCase(i))))^2];
    end
```

```
end
end
if all(coeff==0)
    dtof=NaN; return
else
    coeff=coeff-[0 0 1];
    % Solve quadratic equation
    dtof=((-coeff(2)+sqrt(coeff(2)^2-4*coeff(1)*coeff(3)))/(2*coeff(1)));
    return
end
end


function N = findNeighbors(G)
% Build (n x 2*d) -array of neighbors for each cell in (Cartesian) grid G.
   cellNo = rldecode(1:G.cells.num, diff(G.cells.facePos), 2)';
   col    = 1 +   (cellNo == G.faces.neighbors(G.cells.faces(:,1), 1));
   c      = G.faces.neighbors(double(G.cells.faces(:,1)) + G.faces.num* (col-1));
   N      = accumarray([cellNo, G.cells.faces(:,2)], c);
end
```

A.5 Permeability Description Model

A.5.1 Streamline Parameters

```
function varargout=streamtubePermDesc(gradP, dS, fluid, s)
% Calculate streamline permeability descriptors from streamline gradients,
% line lengths, fluid model and shortest path.
%
%
% SYNOPSIS:
% [Be,Ce,Te2,varTs2,varInvCs] = effPermDesc(BS,CS,TauS2, Qs, Ve);
%
%
% PARAMETERS:
%
%   BS        - Streamline/streamtube hydraulic conductance
%
%   CS        - Streamline constriction factor
%
%   TauS2     - Streamline tortuosity factor
%
%   Qs        - Streamtube rate
%
%   Ve        - Effective bulk volume
%

%
% RETURNS:
%
%  Be        - Effective hydraulic conductance
%
```

```
%  Ce         - Effective constriction factor
%
%  Te2        - Effective tortuosity factor
%
%  varTs2     - Weighted variance of tortuosity factors
%
%  varInvCs   - Weighted variance of inverse constriction factors

%
% EXAMPLE:
%
% [Be,Ce,Te2,varTs2,varInvCs] = effPermDesc(BS,CS,TauS2, Qs, Ve);
%
% SEE ALSO: streamtubePermDesc()
% Written by Asgeir Nyvoll, MSc student NTNU, 2018

if length(gradP)~=length(dS)
    error('The length of pressure gradients and streamlines have to be equal')
end


gradInt=sum(1./gradP(gradP~=0).*dS(gradP~=0));

varargout{1}=-gradInt.*fluid.properties(1);

if nargout>1
    Ls=sum(dS);
    pDrop=sum(gradP.*dS);
    varargout{2}=pDrop/(Ls^2)*gradInt;
end

if nargout>2
    varargout{3}=(s/Ls)^2;
end
end
```

A.5.2 Effective Model Parameters

```
function varargout=effPermDesc(BS,CS,TauS2, QS, Ve)
% Calculate effective permeability descriptors from streamline descriptors
% effective bulk volume and rates
%
%
% SYNOPSIS:
% [BS, CS, TauS2]=streamtubePermDesc(GradP, LS, fluid, shortestpath);
%
%
% PARAMETERS:
%
%   GradP        - Pressure gradients for each substep in Pollock
%                  approximation
%
%   LS           - Streamline length for each substep in Pollock approximation
%
```

```
%   fluid        - MRST fluid structure where fluid.properties(1) is viscosity
%
%   shortestpath - Length of streamline if no tortuosity
%
%
% RETURNS:
%
%  BS         - Streamline/streamtube hydraulic conductance
%
%  CS         - Streamline constriction factor
%
%  TauS2      - Streamline tortuosity factor
%
%
% EXAMPLE:
%
% [BS, CS, TauS2]=streamtubePermDesc(GradP, LS, fluid, shortestpath);
%
% SEE ALSO: effPermDesc(), pollockMod()
% Written by Asgeir Nyvoll, MSc student NTNU, 2018

Q=sum(QS);
BSQS=BS.*QS;
sBSQS=sum(BSQS);
  varargout{1}=sBSQS./Ve;

if nargout>1
  varargout{2}=sum(CS.*QS)./Q;
end

if nargout>2
  varargout{3}=sum(TauS2.*BSQS)/sBSQS;
end

if nargout>3
  varargout{4}=sum(BSQS.*(TauS2-varargout{3}(:)).^2)/sBSQS;
end

if nargout>4
  varargout{5}=sum(((1./CS-1./varargout{2}).^2).*QS)./Q;


end
```

Appendix B

MRST Code for Linear and Trilinear Interpolation of Pressure Gradient

MRST code for linear and trilinear pressure interpolations, though they are not recommended to use based on results shown in this thesis. The recommended code to use is found in Appendix A.

The code given here is tested with Matlab R2017a and MRST 2017b, and is not guaranteed to work on other versions of the softwares, as the code use already existing functions of both Matlab and MRST. MRST can be downloaded from `http://www.sintef.no/projectweb/mrst/downloadable-resources/download/`.

B.1 Linear Interpolation

The following function takes the streamline coordinates S and cell numbers C from the Pollock approximation as input, together with the grid G, rock structure, state/sol and inlet and outlet pressures. It returns a vector of pressures in each coordinate that can later be used to calculate the pressure gradient as $(p_2 - p_1)/\Delta s$, where p_1 and p_2 are the pressures at the beginning and end of a step, and Δs is the streamline step length. It is recommended to take steps over a full cell, as using several substeps can result in pressure oscillations (not continuously dropping pressures). This function is from Nyvoll (2017).

```
function P=pIntLoverK(G,rock,sol,S,C,pIn,pOut)
% Linear interpolation of pressure at cell face between two cells scaled
% with permeability

% PARAMETERS:
%
%   G          - Cartesian grid: MRST structure. Has to include
%                G.cells.centroids
%
%   rock       - MRST rock structure witch includes rock.perm
%
%   sol        - MRST structure that includes the field sol.pressure
%                e.g. result from incompTPFA
%
%   S          - Coordinates of streamline at cell faces
%                as found with pollock().
%
%   C          - Cell number for steps in the streamline
%
%   pIn        - Pressure at start of streamline
%
%   pOut       - Pressure at end of streamline
%
% RETURNS:
%
%  P           - Interpolated pressures at cell faces

% Distance from centroid of previous cell
distBef=sqrt(sum((S(2:end-1,:)-...
       G.cells.centroids(C(1:end-1,:))).^2,2));
%Distance from centroid of next cell
distAft=sqrt(sum((S(2:end-1,:)-...
       G.cells.centroids(C(2:end,:))).^2,2));
%Pressure of previous cell centroid
Pbef=sol.pressure(C(1:end-1));
%Pressure of next cell centroid
Paft=sol.pressure(C(2:end));
%Slope of linear interpolation, including perm-scaling:
alpha = (Pbef-Paft)./(distBef./rock.perm(C(1:end-1,:),1)...
       +distAft./rock.perm(C(2:end,:),1));
%Find pressures along streamline:
P=[pIn; Pbef-alpha.*distBef./rock.perm(C(1:end-1,:),1);pOut];
```

B.2 Trilinear Interpolation

Pollock algorithm that includes pressure estimation by the use of trilinear interpolation. Modified version of MRST's existing pollock() function. This function returns coordinates, times of flight, cells and pressures. Again the pressure gradient can be estimated by $(p_2 - p_1)/\Delta s$, where p_1 and p_2 are the pressures at two neighboring coordinates, and Δs is the streamline length between them.

```
function varargout = pollockPressureTriLin(G, state, rock, varargin)
% Trace streamlines in logically Cartesian grid using Pollock approximation.
% In addition to the regular Pollock approximation, pressures are
% supported by the use of trilinear interpolation (not recommended).
%
%
% SYNOPSIS:
%   [S,T,C,P] = pollockMod(G, state, rock)
%   [S,T,C,P] = pollockMod(G, state, rock, startpos)
%   [S,T,C,P] = pollockMod(G, state, rock, 'pn', pv, ...)
%   [S,T,C,P] = pollockMod(G, state, rock, startpos, 'pn', pv, ...)
%
% PARAMETERS:
%
%   G          - Cartesian or logically Cartesian grid.
%
%   state      - State structure with field 'flux'.
%
%   rock       - Rock structure with the field 'poro'.
%
% OPTIONAL PARAMETERS
%
%   positions - Matrix of size (N, 1) or (N, d+1), where d is the dimension
%               of the grid, used to indicate where the streamlines should
%               start.
%
%               If the size is (N, 1), positions contains the cell indices
%               in which streamlines should start. Each streamline is
%               started in the the local coordinate (0.5, 0.5, ...). To be
%               precise, this is the mean of the corner points, not the
%               centroid of the cell.
%
%               If the size is (N, d+1), the first column contains cell
%               indices, and the d next columns contain the local
%               coordinates at which to start streamlines.
%
% OPTIONAL PARAMETERS (supplied in 'key'/value pairs ('pn'/pv ...)):
%
%   substeps    - Number of substeps in each cell, to improve visual quality.
%                Default 5.
%
%   maxsteps    - Maximal number of points in a streamline.
%                Default 1000.
%
%   reverse     - Reverse velocity field before tracing.
%                Default false.
%
% RETURNS:
%
%  S      - Cell array of individual streamlines suitable for calls like
%           streamline(pollock(...)) and streamtube(pollock(...)).
%
%  T      - Time-of-flight of coordinate.
%
%  C      - Cell number of streamline segment, i.e, line segment between
%           two streamline coordinates.
%
%  P      - Interpolated pressure in each coordinate

% EXAMPLE:
%
%   [S,T,C,P] = pollockPressureTriLin(G, state, rock,startpos);
%
%
% SEE ALSO: pollockMod()
%
%
%{
ORIGINAL COPYRIGHT FROM MRST:

Copyright 2009-2017 SINTEF ICT, Applied Mathematics.

This file is part of The MATLAB Reservoir Simulation Toolbox (MRST).

MRST is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

MRST is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with MRST.  If not, see <http://www.gnu.org/licenses/>.
%}

% Written by Jostein R. Natvig, SINTEF ICT, 2010.
%
% Modified by Asgeir Nyvoll, MSc student NTNU, 2018

   d = size(G.nodes.coords, 2);
   if mod(length(varargin),2)==0
      positions   = [(1:G.cells.num)', repmat(0.5, [G.cells.num, d])];
   else
      positions = varargin{1};
      if size(positions, 2) ==1
         positions = [positions, repmat(0.5, [size(positions, 1), d])];
      elseif size(positions, 2) ~= 1 + d
         error('Expected array of local positions of width 1 or 1+d.');
      end
      varargin  = varargin(2:end);
   end
   opt = struct('substeps', 5, 'maxsteps', 1000, 'reverse', false);
   opt = merge_options(opt, varargin{:});

   if opt.reverse
      state.flux = -state.flux;
   end

   [varargout{1:nargout}] = trace(G, state, positions, opt, rock);
```

```

end



% ========================================================================
function varargout = trace(G, state, pos, opt, rock)
   d               = size(G.nodes.coords, 2);
   numStreamlines = size(pos,1);
   assert(size(pos, 2) == d+1);

   if ~isfield(G, 'cellNodes')
      cn  = cellNodes(G);
      G.cellNodes = accumarray(cn(:,1:2), cn(:,3));
   end

   % Make array face fluxes for each cell in grid (Not outer).
   cellNo = rldecode(1:G.cells.num, diff(G.cells.facePos), 2) .';
   cf     = G.cells.faces;
   pv     = poreVolume(G,rock);
   flux   = accumarray([cellNo, cf(:,2)], state.flux(cf(:,1)));
   velo   = flux./pv;
   velo(pv==0,:)=0;
   clear cf cellNo

   neighbors  = findNeighbors(G);

   nodePressure=nodePressures(G,state,rock);
   nodePressure(G.nodes.coords(:,2)==min(G.nodes.coords(:,2)))=5e7;
   nodePressure(G.nodes.coords(:,2)==max(G.nodes.coords(:,2)))=1e7;
   state.cellNodePressure=zeros(size(G.cellNodes));
   state.cellNodePressure(:,:)=nodePressure(G.cellNodes);
   clear nodePressure



   magic  = 1000;
   XYZ    = nan(numStreamlines, d, magic);
   T      = nan(numStreamlines, magic);
   C      = nan(numStreamlines, magic);
   P      = nan(numStreamlines, magic);
   active = true(numStreamlines, 1);


   % Store crossing coordinates of active streamlines
   [XYZ(active,:,1)] = globalCoordinate(G, pos(active,1), pos(active, 2:end));
   T(active, 1) = zeros(sum(active), 1);
   C(active, 1) = pos(active,1);
   P(active, 1) = localPressure(state,    pos(active,1), pos(active, 2:end));

   i = 2;
   while any(active)
      % Realloc
      if i+opt.substeps+1 > size(XYZ, 3)
         magic = max(magic, opt.substeps+1);
         XYZ   = cat(3, XYZ, nan(numStreamlines, d, magic));
         T     = cat(2, T,   nan(numStreamlines, magic));
         C      = cat(2, C,    nan(numStreamlines, magic));
         P      = cat(2, P,    nan(numStreamlines, magic));
      end
      current_cell = pos(active,1);

      % Take another pollock step
      [pos(active, :), t, xyz] = step(pos(active,:), velo, neighbors, opt.substeps);

      % Store crossing coordinates and, optionally, coordinates along curve
      % trajectory in cell of active streamlines
      for k=1:opt.substeps
         [XYZ(active, :, i+k-1)] = globalCoordinate(G, current_cell, xyz(:,:,k));
         P(active, i+k-1) = localPressure(state, current_cell, xyz(:,:,k));
      end
      T(active, i-1+(1:opt.substeps)) = repmat(t/opt.substeps, [1, opt.substeps]);
      C(active, i-1+(1:opt.substeps)) = repmat(pos(active, 1), [1, opt.substeps]);

      % Update active flag
      active(active)    =  pos(active,1) ~= current_cell;

      i = i+opt.substeps;
      if i > opt.maxsteps, break;end
   end

   %% Pack coordinates in list with streamlines separated by NaN.
   p = reshape(permute(XYZ, [3,1,2]), [], d);

   i = ~isnan(p(:,1));
   j = i|[true;i(1:end-1)];
   p = p(j,:);

   % Pack streamline coordinates in a cell array suitable for use with
   % Matlab streamline, i.e., as in 'streamline(pollock(G, resSol));'
   flag = isnan(p(:,1));
   ix = find(flag);
   dd   = diff([0;ix])-1;
   varargout{1} = mat2cell(p(~flag,:), dd, d);
   if nargout > 1
      T = reshape(T', [], 1);
      T = T(j);
      varargout{2} = mat2cell(T(~flag), dd, 1);
   end
   if nargout > 2
      C = reshape(C', [], 1);
      C = C(j);
      varargout{3} = mat2cell(C(~flag), dd, 1);
   end
   if nargout > 3
      P = reshape(P', [], 1);
      P = P(j);
      varargout{4} = mat2cell(P(~flag), dd, 1);
   end
end



% ==========================================================================
function xyz = globalCoordinate(G, c, p)
% Compute global coordinate corresponding to local coorinate p in cells c
% p  - local positions == [xi,eta,zeta] in 3D
% c  -
%
   if numel(c)==1, p = reshape(p, 1, []); end
   % Compute node weight for quadrilateral or hexahedron
   d = size(G.nodes.coords, 2);
   w = ones(size(p,1), 2^d);
   for i=1:d
      mask        = logical(bitget((0:2^d-1)', i));
      w(:, mask)  = w(:, mask).* repmat(  p(:,i), [1, sum( mask)]);
      w(:,~mask)  = w(:,~mask).* repmat(1-p(:,i), [1, sum(~mask)]);
   end

   % Compute weighted average of corner points
   xyz = zeros(size(p,1), d);
   for i=1:d
      xi       = G.nodes.coords(:,i);
      xyz(:,i) = sum( w.*reshape(xi(G.cellNodes(c, :))', 2^d, [])', 2);
   end
end



%% ======================================================================
function [pos, tof, xyz] = step(pos, flux, neighbors, nsubsteps)
% Update pos array by computing new local coordinate and new cell.
% In addition, compute curve within cell.
%
%
%
%
   f = flux(pos(:,1),:);
   n = neighbors(pos(:,1),:);

   dims = size(pos, 2)-1;
   T    = nan(size(pos,1),dims);
   for i=1:dims
      T(:,i) = computeTime(pos(:,1+i), f(:,2*i-1:2*i));
   end
   [tof, dir] = min(T, [], 2);

   xyz = zeros(size(pos,1), dims, nsubsteps);
   d   = zeros(size(pos, 1), 1);
   for s=1:nsubsteps
      for i=1:dims
         t = tof*s/nsubsteps;
         [xyz(:,i,s), d(:,i)] = computePosition(pos(:,1+i), f(:,2*i-1:2*i), t);
      end
   end

   pos (:,2:end) = xyz(:,:,s);

   % Find direction to look up neighbor cell
   k  = 2*(dir-1)+d(sub2ind([numel(dir), 3], (1:numel(dir))', dir));
   t  = sub2ind(size(n), (1:numel(k))', k);

   % Update cell number if NOT at boundary.
   % IF at boundary, mark dir with NaN to avoid changing local coordinate
   % below.
   ind          = n(t)==0;
   pos(~ind,1)  = n(t(~ind));
   dir (ind)    = nan;

   % Change local coordinate when moving to new cell
   k = sub2ind(size(d), (1:size(dir,1))', dir);
   k = k(~isnan(k));
   pos(numel(dir) + k ) = 2-d(k);
end


% ==========================================================================
function N = findNeighbors(G)
% Build (n x 2*d) -array of neighbors for each cell in (Cartesian) grid G.
   cellNo = rldecode(1:G.cells.num, diff(G.cells.facePos), 2)';
   col    = 1 +   (cellNo == G.faces.neighbors(G.cells.faces(:,1), 1));
   c      = G.faces.neighbors(double(G.cells.faces(:,1)) + G.faces.num* (col-1));
   N      = accumarray([cellNo, G.cells.faces(:,2)], c);
end






% ==========================================================================
function t = computeTime(xi, v)
% Compute time needed to reach xi=0 or xi=1 given velocities v=[v1,v2] at
% xi=0 and xi=1.  The formula is
%
%   t = xi/ui  or t = (1-xi)/ui,    if v1 = v2 = ui, and
%
%   t = 1/(v2-v1)*log(ue/ui),       otherwise
%
% where ui=v2*xi+v1*(1-xi) is the velocity at xi, and ue=v2 if ui>0 or
% ue=v1 if ui<0.
   tolerance = 100*eps;

   ui          = v(:,1) + xi.*diff(v, 1, 2);%(:,2)-v(:,1));
   ue          = v(:,    2);
   ue  (ui<0)  = v(ui<0, 1);
   arg         = ue./ui;
   t           = inf(size(xi));

   % Linear velocity
   ind         = abs(diff(v, 1, 2)) > tolerance*abs(v(:,1));
   t(ind,:)    = 1./diff(v(ind,:), 1, 2).*log(arg(ind,:));

   % Constant velocity
   ds          = -xi;
   ds(ui > 0)  = 1-xi(ui>0);
   t(~ind)     = ds(~ind)./ui(~ind);

   % nan happens for ui=ui=0
   t(arg<0 | isnan(arg))    = inf;
end


% ========================================================================
function [x, i] = computePosition(xi, v, t)
% Compute position at time t given start point xi and velocities v=[v1,v2].
%
%   x = xi + v*t,    if v is constant or
%
%   x = xi + (ui*exp((v2-v1)*t) - ui)/(v2-v1), otherwise
%
   tolerance = 100*eps;

   du         = diff(v, 1, 2);
   ui         = v(:,1) + xi.*du;
   i          = 1 + ~(ui<0);
   x          = inf(size(xi));

   ind        = abs(du) > tolerance*abs(v(:,1));

   % linear velocity
   x(ind)     = xi(ind) + ( ui(ind).*exp(du(ind).*t(ind)) - ui(ind))./du(ind);

   % Constant velocity
   x(~ind,:) = xi(~ind,:) + v(~ind, 1).*t(~ind, :);
   x(~ind & t==inf) = xi(~ind & t==inf);
end

function nodeP=nodePressures(G,state,rock)
%Calculates pressure in each node (grid cell corners)
cn=cellNodes(G);
nodeCells=accumarray([cn(:,3), cn(:,2)], cn(:,1));
weights=nodeCells;
cellPressures=weights;
active=nodeCells>0;
wx=weights; wy=weights; wz=weights;
wx(active)=G.cells.centroids(nodeCells(active),1);
wx=wx-G.nodes.coords(:,1);
wx(nodeCells==0)=0;

wy(active)=G.cells.centroids(nodeCells(active),2);
wy=wy-G.nodes.coords(:,2);
wy(nodeCells==0)=0;

wz(active)=G.cells.centroids(nodeCells(active),3);
wz=wz-G.nodes.coords(:,3);
wz(nodeCells==0)=0;


dims=size(rock.perm);
% Length and permeability scaled weighting
if(dims(2)==1)
weights(weights>0)=(abs(wx(active)).*rock.perm(nodeCells(active),1)...
    +abs(wy(active))+abs(wz(active))).*rock.perm(nodeCells(active))...
    ./(wx(active).^2+wy(active).^2+wz(active).^2);
```

```

else
weights(weights>0)=(abs(wx(active)).*rock.perm(nodeCells(active),1)...
    +abs(wy(active)).*rock.perm(nodeCells(active),2)...
    +abs(wz(active)).*rock.perm(nodeCells(active),3))...
    ./(wx(active).^2+wy(active).^2+wz(active).^2);

end
weightsum=sum(weights,2);
cellPressures(active)=state.pressure(nodeCells(active));
nodeP=sum((weights.*cellPressures),2)./weightsum;

end

function streamlinePressure=localPressure(state,cell, pos)
%Trilinear interpolation
c000=state.cellNodePressure(cell,1);
c100=state.cellNodePressure(cell,2);
c010=state.cellNodePressure(cell,3);
c110=state.cellNodePressure(cell,4);
c001=state.cellNodePressure(cell,5);
c101=state.cellNodePressure(cell,6);
c011=state.cellNodePressure(cell,7);
c111=state.cellNodePressure(cell,8);

c00=c000.*(1-pos(:,1))+c100.*pos(:,1);
c01=c001.*(1-pos(:,1))+c101.*pos(:,1);
c10=c010.*(1-pos(:,1))+c110.*pos(:,1);
c11=c011.*(1-pos(:,1))+c111.*pos(:,1);

c0=c00.*(1-pos(:,2))+c10.*pos(:,2);
c1=c01.*(1-pos(:,2))+c11.*pos(:,2);

streamlinePressure=c0.*(1-pos(:,3))+c1.*pos(:,3);
end
```