NTNU – Trondheim
Norwegian University of
Science and Technology

Numerical modelling of Bingham fluid flow and particle transport in a rough-walled fracture

Alexander Rikstad Hanssen

Petroleum Geoscience and Engineering
Submission date: May 2013
Supervisor: Rune Martin Holt, IPT
Co-supervisor: Alexandre Lavrov, SINTEF Petroleum Research

Norwegian University of Science and Technology
Department of Petroleum Engineering and Applied Geophysics

ABSTRACT

Drilling operations may encounter mud loss which becomes especially aggravated in naturally fractured formations. The dynamics of fluid loss is a complex process, affected by both the rheology and the composition of the fluid as well as by the fracture morphology. A deep understanding of the underlying physics, in particular non-Newtonian fluid flow and particle transport in rough-walled fractures, is required to combat fluid loss by use of e.g. LCM.

Previous studies have shown increased aperture channelization and particle transport in the direction perpendicular to the shear displacement with Newtonian fluids. The motivation for and goal of this thesis is to investigate how non-Newtonian fluid flow and particle transport behave in rough-walled fractures subject to shear.

One rough-walled fracture with average aperture equal to 0.5 mm and Hurst exponent equal to 0.75 was created by using the recursive subdivision technique. The fracture surfaces were shifted 0.01 m in either coordinate direction. A test matrix of 108 simulations were completed for the two shearing scenarios with flow applied in x- and y-direction.

Conclusions drawn from the study include increased fluid channelization for large values of the ratio between yield stress value and applied pressure difference. The fluid's ability to carry particles was greater in the direction perpendicular to the shear displacement. In addition, the distributions of average cross sectional aperture perpendicular to fluid flow was more rough when flow was applied along the direction of shear displacement, leading to yield dominant flow.

SAMMENDRAG

I boreoperasjoner kan slamtapsituasjoner oppstå, og særlig i naturlig frakturerte formasjoner. Fysikken bak slamtap er en kompleks prosess som påvirkes både av reologien og komposisjonen til fluidet, samt av sprekkens morfologi. For å motvirke slamtap, for eksempel ved hjelp av metoder som LCM, er det nødvendig med en dypere forståelse av den underliggende fysikken, spesielt ikke-Newtonsk strømning og partikkeltransport i ruveggede sprekker.

Tidligere studier har vist økende kanalisering av sprekkåpning og partikkeltransport i retning normalt til forskyvning med Newtonsk fluid. Motivasjonen og målet med oppgaven er å undersøke hvordan ikke-Newtonwsk strømning og partikkeltransport oppfører seg i ruveggede sprekker dannet ved forskyvning.

En sprekk med gjennomsnittlig åpning på 0.5 mm og med Hurst eksponent lik 0.75 ble laget ved hjelp av rekursiv inndelingsteknikk. Sprekkoverflatene ble forskjøvet med 0.01 m i begge koordinatretninger. En testmatrise på 108 simuleringer ble gjennomført for de to forskjellige forskyvningsscenarioene med pålagt strømning i x- og y-retning.

Konklusjoner fra studien inkluderer økende kanalisering av fluidstrømning for høye verdier av forholdet mellom flytpunkt og pålagt trykkforskjell. Fluidets evne til å bære partikler var høyere i retning normalt til forskyvning. Distribusjonen av gjennomsnittlig tverrsnittssprekkåpning normalt til strømningsretninger var mer ru som fører til dominerende flytstyrkestrømning.

TABLE OF CONTENTS

1 Overview 8
2 Introduction 9
2.1 Shear-induced flow anisotropy 9
2.2 Solute transport in sheared fractures 11
2.3 Lubrication theory approximation 12
2.4 Rheological effects on mud losses 14
3 Theory 16
3.1 Fluid flow governing equations 16
3.2 Particle transport governing equation 19
3.3 Recursive subdivision technique 20
3.4 Numerical fracture properties 21
3.4.1 Scaling factor 22
3.4.2 C-factor 23
3.4.3 Hurst exponent 24
3.5 Shear displacement 25
3.6 Estimation of Bingham plastic flow pressure criteria 27
4 Numerical modeling 29
4.1 Modeling 29
4.2 Test matrix 32
4.3 Post processing of data 35
4.3.1 Pressure plots 35
4.3.2 Velocity fields 35
5 Results 37
5.1 Particle transport 38
5.2 Pressure distribution 40
5.3 Velocities 42
5.4 Viscosity 45
5.5 Shear displacement and fluid properties 47
5.6 Breakthrough curves 49

6 Discussion 50
6.1 Analytical 50
6.2 Channelization of fluid flow 53
6.3 Pressure distributions 55
6.4 Evaluation of fractures 55
6.5 Particle transport 57
6.6 Challenges 57
7 Future work 59
8 Conclusion 60
9 Acknowledgements 61
10 References 62

LIST OF FIGURES

Fig. 2.1.1 - Aperture evolution due to shearing - Larger apertures appear more frequently for greater shear displacement. After Koyama et al. (2006). 9

Fig. 2.1.2 - Hurst exponent as function of shear displacement with sample sizes A and B, where A < B. The exponent increases for increased shear displacement for both sizes and reaches a stationary level at larger shear displacement. After Koyama et al. (2006). 10

Fig. 2.2.1 - Schematic view on breakthrough curves. A refers to the reference case, B to flow parallel to the shear displacement and C to flow perpendicular to the shear displacement. After Vilarrasa et al. (2011). 11

Fig. 2.3.1 - $C1(\delta)$ plotted for different values of roughness. 14

Fig. 2.4.1 - (a) The effect of yield stress on ultimate volume of losses (b) The effect of yield stress on cumulative volume of losses as a function of time. After Majidi et. al (2010). The ultimate lost volume is reduced for greater values of yield stress (a). The yield stress also affects the cumulative volume curves, where stationary values are reached faster for high yield stress values. 15

Fig. 3.1.1 - Illustration of fluid flow between two parallel plates. 16

Fig. 3.3.1 - The number of level represents the number of times each cell is subdivided into four new cells. Different levels are represented in (a-d). 20

Fig. 3.3.2 - The recursive subdivision technique first generates the nodes on the four sides of the grid. Each node is calculated by using the sourrounding two nodes. The inner points are created by using the surface elevation values at the the sourrounding four nodes, as shown in (a-c)...21

Fig. 3.4.1 - The scaling factor affects the amplitude of the fracture (Hanssen, 2012).............22

Fig. 3.4.2 - Increasing c-factor leads to greater merging of local vertexes and creates an even terrain. (Hanssen, 2012) 23

Fig. 3.4.3 - Cross section of fracture aperture with varying Hurst exponent. The solid line represent the completely uncorrelated series ($H = 0.5$). When comparing the dashed lines to the uncorrelated series you can see the switching between high and low values for $H = 0.3$ and the long-range positive auto-correlation for $H = 0.8$...24

Fig. 3.5.1 - Schematic of the numerical method used for shear displacement. 25

Fig. 3.5.2 - Aperture distribution for different shear displacement in x-direction. The figure illustrates the effect of increasing shear displacement (a-f). Flow channels become more pronounced with increasing shear displacement. The values of shear displacement is (a) 0.01 m, (b) 0.02 m, (c) 0.03 m, (d) 0.04 m, (e) 0.05 m and (f) 0.06 m. Aperture can be read from the color bar...26

Fig. 3.6.1 - The minimum average aperture path perpendicular to flow direction...................27

Fig. 4.1.1- 3D image of the fracture aperture. The lower fracture surface is shifted one node in the x-direction. Note the channelization perpendicular to shifting (dashed line). 30

Fig. 4.1.2 - 3D image of the fracture aperture. The lower fracture surface is shifted one node in the y-direction. Note the channelization perpendicular to shifting (dashed line) 31

Fig. 4.2.1 - Schematic view of different simulation scenarios. The icons on the right side of the different scenario refers to the different cases further on in this study. 33

Fig. 5.1.1 - Particle concentration for increasing T values: (a) 0.00 (b) 0.19 (c) 0.38 and (d) 0.53 for left column and 0.50 the right column. Shear displacement along x-axis. Concentration values can be read from the color bar on the right-side. 38

Fig. 5.1.2 - Particle concentration for increasing T values: (a) 0.00 (b) 0.18 (c) 0.37 and (d) 0.53. Shear displacement along y-axis. Concentration values can be read from the color bar on the right-side..39

Fig. 5.2.1 - Pressure distributions of Bingham plastic compared to Newtonian fluid flow (a) for increasing T (b-d) values as follows: (b) 0.19 (c) 0.38 and (d) 0.53 for left column and 0.50 the right column. Shear displacement along x-axis. Normalized pressure values can be read from the color bar on the right-side. Dark blue dots represent closed local aperture. 40

Fig. 5.2.2 - Pressure distributions of Bingham Plastics compared to Newtonian fluid flow (a) for increasing T (b-d) values as follows: (b) 0.18 (c) 0.37 and (d) 0.53. Shear displacement along y-axis. Normalized pressure values can be read from the color bar on the right-side. Dark blue dots represent closed local aperture. 41

Fig. 5.3.1 - Velocity distribution for increasing T values: (a) 0.00 (b) 0.19 (c) 0.38 and (d) 0.53 for left column and 0.50 the right column. Shear displacement along x-axis. 42

Fig. 5.3.2 - Velocity distribution for increasing (a-d) T values: (a) 0.00 (b) 0.19 (c) 0.38 and (d) 0.53. Shear displacement along y-axis. 43

Fig. 5.3.3 - Comparison of average velocity in the coordinate directions perpendicular and parallel to shear displacement. Both fluid velocities are in the direction parallel to applied pressure gradient. Data points for T = 0.50 is excluded for Fracture 1 because different T values of the two scenarios were used. 44

Fig. 5.4.1 - Particle concentration plots for varying plastic viscosities (a) 0.001 Pa s (b) Pa s 0.01 (c) 0.03 Pa s. 45

Fig. 5.4.2 - Average velocity component in flow-direction for varying viscosities. Shear displacement applied in x-direction. Flow (a) parallel and (b) perpendicular to shear displacement direction. Unidirectional fluid velocity is higher for (b). The initial fracture surfaces were shifted one node in y-direction...46

Fig. 5.4.3- Average velocity component in flow-direction for varying viscosities. Shear displacement applied in y-direction. Flow (a) parallel and (b) perpendicular to shear displacement direction. Unidirectional fluid velocity is higher for (b). The initial fracture surfaces were shifted one node in x-direction...46

Fig. 5.5.1 - The Bingham number plotted against the non-dimensional yield stress. δ expresses the shifting and the term in the brackets represents the direction of shear displacement. 47

Fig. 5.5.2 - The ratio of velocity parallel and perpendicular to the applied pressure-gradient plotted against non-dimensional yield stress. 48

Fig. 5.6.1 - Breakthrough curves of particle transportation at node lines perpendicular to flow at Nx = 3, 10, 20, 30, 40, 50 and 60. Fluid flow is in parallel to shear displacement in (a) and perpendicular to shear displacement in (b). The two cases are both Newtonian with $\mu = 0.01\ Pa \cdot s$ and $\Delta P = 500\ 000\ Pa$. 49

Fig. 6.2.1 - Cross sectional average aperture perpendicular to flow - (a) Flow along shear displacement (b) Flow normal to shear displacement..53

Fig. 6.2.2 - (a) Newtonian fluid (b) Bingham plastic with T = 0.53. Channelization is more pronounced for the Bingham fluid, e.g. at x = 0.28...54

Fig. 6.4.1 - The Frequency plots of the two fractures with shear displacement of 10 mm.......55

Fig. 6.4.2 - Aperture distribution of fracture 2. Color bar on the right shows the aperture values in metres. Shear displacement in the y-direction..56

Fig. 6.5.1 - Particle concentration plot for (a) Newtonian fluid with $T = 0$ and (b) Bingham Fluid with T = 0.50. The aperture distribution can be seen in (c). Shear displacement applied in x-direction. A denotes flow in x-direction and B denotes flow in y-direction. The particle transport is enhanced in case B and the flow becomes more channelized with increasing T. 57

LIST OF TABLES

Tab. 4.1.1 - Fracture surface input...29

Tab. 4.1.2 - Calculation looping conditions..31

Tab. 4.2.1 - Test Matrix - The case number labels (a-d) refers to the different simulation scenarios (a-d) from Fig. 4.2.1. The pressure calculations in simulations 79-81 did not converge at the initial T ratio. The overpressure was increased to 270 000 Pa to achieve convergence. 34

1 OVERVIEW

Drillers may encounter mud losses which become especially aggravated in naturally fractured formations. The dynamics of fluid loss is a complex process, affected by both the rheology and composition of the fluid as well as by the fracture morphology. A deep understanding of the underlying physics, in particular non-Newtonian fluid flow and particle transport in rough-walled fractures, is required to combat fluid loss by use of e.g. LCM.

The motivation for and goal of this study is to achieve a greater understanding of how fluid properties and fracture morphology affect flow in fractures subject to shear. The recursive subdivision technique was used to create rough-walled fractures satisfying the lubrication theory approximation. In addition, an algorithm was developed to introduce shifting in the coordinate directions in the XY-plane. PROPANICA, a research code developed by Alexandre Lavrov was used to simulate non-Newtonian fluid flow and particle transport. The author has developed numerous scripts written in C++, python, windows batch and UNIX shell code in order to increase simulation capacity and to visualize and present the results.

Two fractures with applied shear displacement were created and flow of non-Newtonian drilling fluid was simulated with flow either normal or parallel to shifting. The effects of yield stress, plastic viscosity, applied pressure difference and direction of flow were investigated in a test matrix consisting of 108 individual simulations. One particle source node was located near the inlet boundary of the model for all simulations. The fractures consist of 64x64 cells with grid-spacing equal 0.01 m in the coordinate directions. The initial Hurst exponent was set equal to 0.75 prior to shearing.

In particular the ratio between the yield stress and the applied pressure difference affected the channelization of fluid flow. Channelization of fluid flow and particle transport become more pronounced for increasing ratio of yield stress and applied pressure difference, both for flow parallel and perpendicular to shearing. The fluid's ability to carry particles was greater for the latter case. When comparing pressure distributions for Newtonian and non-Newtonian fluids, the local pressure varies in regions with closed apertures. Plastic viscosity did not affect fluid channelization or particle transport.

A literature study on the fundamentals of flow in shear-induced fractures with Newtonian fluid is included to provide the reader insight in previous findings. This thesis aims to reflect previous short-comings in the literature.

2 INTRODUCTION

2.1 Shear-induced flow anisotropy

The drilling fluid dynamics are affected by properties of the fluid, by the fracture aperture and morphology. The latter plays an important role in controlling fluid losses. It is difficult to obtain the characteristics of a fracture through laboratory samples due to its limited size. Tortuosity, contact areas and original channeling may not be representative in small-scale samples, and thus the sample's hydro-mechanical properties cannot be transferred to real life fracture fields.

Koyama et al. (2006) highlighted previous shortcomings in the literature by studying the effect of aperture evolution due to shear displacement for different sample sizes. These findings highly impact the understanding of coupled hydro-mechanical analysis in fractured rocks. Fractured rocks were investigated with different shear displacements up to 0.02 m with different sample sizes. Two surfaces were digitally scanned and numerically manipulated. The fracture aperture distribution was created by defining three contact points between the two surfaces. One of the surfaces was of smaller size in order to enable large shear displacement. The evolution of the aperture field revealed that larger apertures appeared more frequently in the case of increased shear displacement as shown in Fig. 2.1.1. The effect was more pronounced for the case with the largest sample.

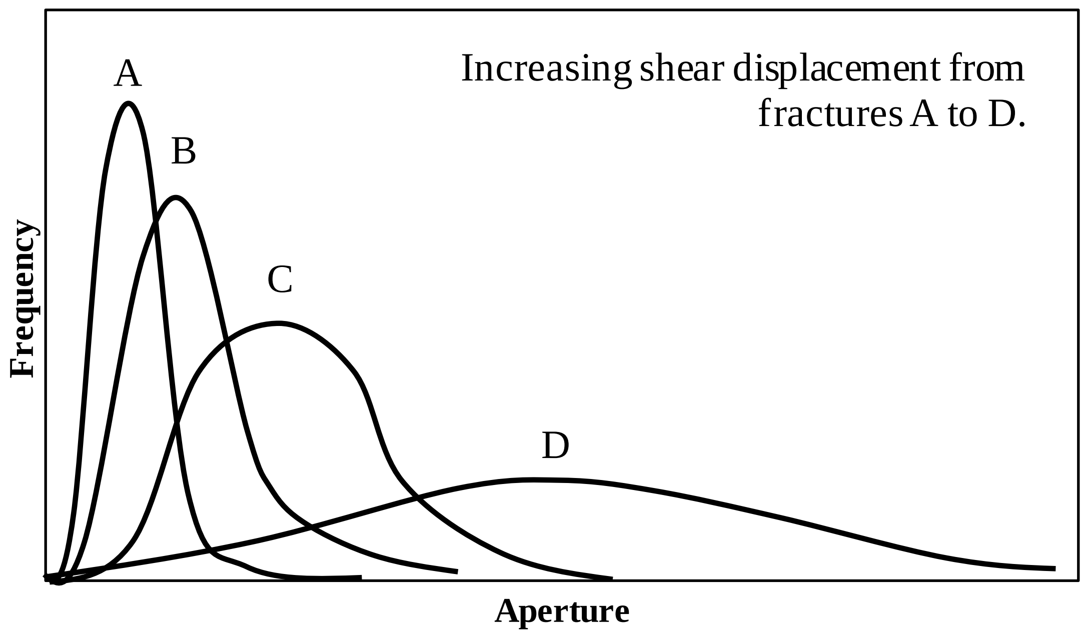


Fig. 2.1.1 - Aperture evolution due to shearing - Larger apertures appear more frequently for greater shear displacement. After Koyama et al. (2006).

Fractures with large shear displacement and smaller sample size show greater aperture correlation and reaches a stationary level of the Hurst exponent. Incompressible fluid flow following the cubic law was numerically simulated to study the shear-induced flow anisotropy. The transmissivity and aperture distribution showed that the fracture aperture increases anisotropically both parallel and perpendicular to the shear displacement direction. The effect was more pronounced in the perpendicular direction and for greater shear displacements (Koyama, et al., 2006). The unidirectional flow showed significant fluid channelization due to shearing (Koyama, et al., 2006).

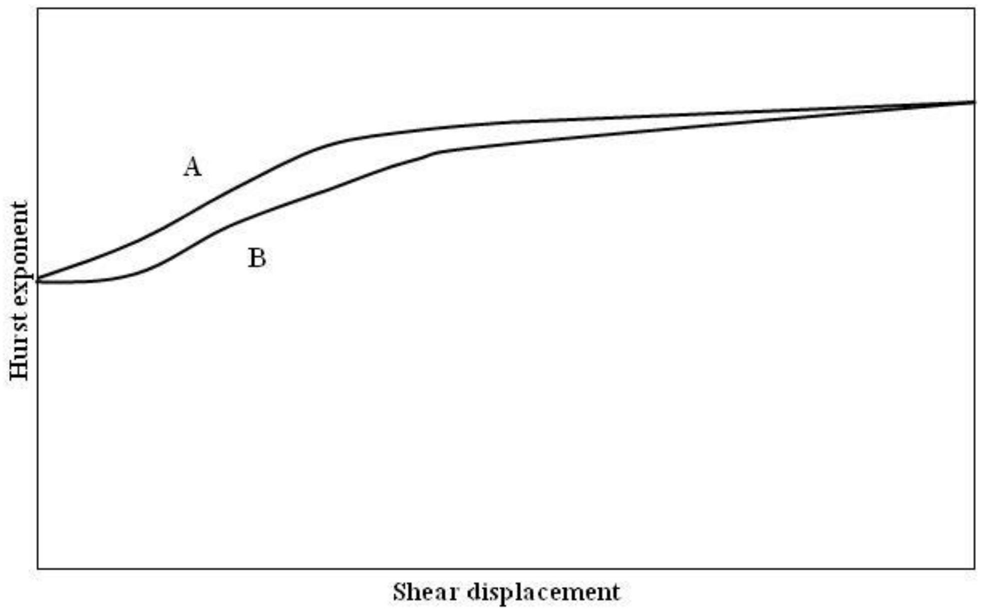


Fig. 2.1.2 - Hurst exponent as function of shear displacement with sample sizes A and B, where size of B is larger than A. The exponent increases for increased shear displacement for both sizes and reaches a stationary level at larger shear displacement. After Koyama et al. (2006).

The Hurst exponent increased for increasing shear displacement for all sample sizes tested by Koyama et al. (2006). The different samples reached a stationary level at large shear displacement, as illustrated in Fig. 2.1.2.

2.2 Solute transport in sheared fractures

Natural fractures can be characterized by their typical anisotropy in aperture fields before shearing (Vilarrasa, et al., 2011). Further fracture evolution may develop due to shear displacement, creating continuous channels adding additional anisotropy to the fracture aperture distribution.

How does the shearing anisotropy affect solute transport in fractures? Vilarrasa et al. (2011) studied the effect of fluid flow and solute transport in shear-induced fractures through numerical modeling. In similarity to Koyama et al. (2006) two fracture surfaces were digitally scanned and numerically manipulated to simulate shear displacement from 1 mm up to 20 mm with 1 mm steps. The transmissivity was calculated for each element, and the fluid was assumed to follow the cubic law.

The Reynolds equation was used to describe the fluid flow in the fractures. Constant pressure boundary conditions were applied on the boundaries parallel to the global flow direction. No-flow boundary conditions were applied on two of the boundaries. Entrance and exit pressure were held constant. The fluid flow was simulated both parallel and perpendicular direction at different shear displacements.

When flowing perpendicular to the shear displacement, the unidirectional flow was significantly higher compared to flow parallel to the shear. The results indicate channelization perpendicular to direction of shear displacement. Clustered particle paths were more dominant with increased shear displacement (Vilarrasa, et al., 2011).

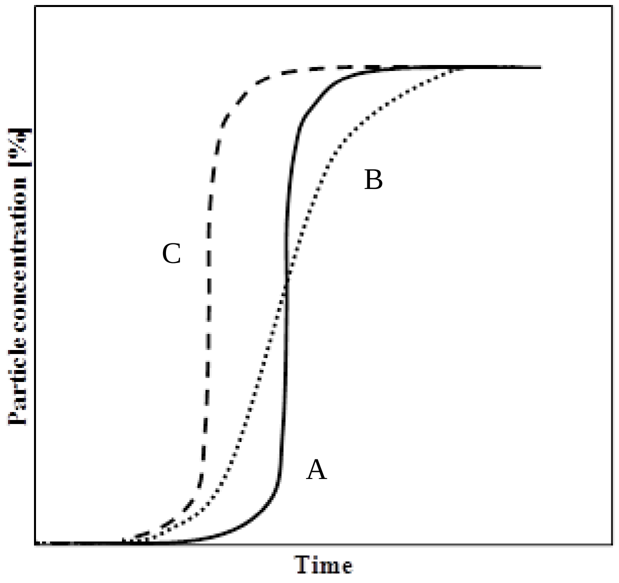


Fig. 2.2.1 - Schematic view on breakthrough curves. A refers to the reference case, B to flow parallel to the shear displacement and C to flow perpendicular to the shear displacement. After Vilarrasa et al. (2011).

Breakthrough curves from the study illustrated in Fig. 2.2.1 show that particle travel velocity increases for increasing shear displacement. The mean particle travel time is smaller in the direction perpendicular to the shear displacement.

In reality, the flow rate increase due to shearing is more pronounced in the early stages of shifting up to 7-8 mm. Larger displacements can reduce the permeability of the fracture leading to less increase of flow rate due to gouge formation. This means that the velocities at high displacements may become overestimated (Vilarrasa, et al., 2011).

2.3 Lubrication theory approximation

In this section the lubrication theory is introduced and an expression for the root mean square of the aperture distribution is derived in order to obtain geometric conditions for the validity of the lubrication theory approximation for the fractures used in this study.

The lubrication theory assumes no inertie and that pressure does not vary along the z-axis. Zimmerman et al. (1991) performed an analysis of the lubrication theory on a rough-walled fracture following a sinusoidal variation. The Reynolds equation was studied under the simplification of 1-D, as follows:

$$\frac{d}{dx}\left(w^3(x)\frac{dp}{dx}\right) = 0 \tag{2.1}$$

where p is pressure, x is the coordinate in the direction of flow and $w(x)$ is the aperture along the x-axis. The rough-walled fracture is assumed to follow a sinusoidal perturbation:

$$w(x) = w_m\left(1 + \delta \sin\left(\frac{2\pi x}{\lambda}\right)\right) \tag{2.2}$$

where δ is the non dimensional amplitude of the roughness variation and λ the wavelength. Integrating Eq. 2.2 by using the definition of the hydraulic aperture given by:

$$w_h = \left(\frac{1}{L}\int_{x_1}^{x_2}\frac{dx}{w^3(x)}\right)^{-\frac{1}{3}} \tag{2.3}$$

the following expression is obtained:

$$w_h^3 = w_m^3\frac{(1-\delta^2)^{\frac{5}{2}}}{1+\left(\frac{\delta^2}{2}\right)} \tag{2.4}$$

where h denotes the hydraulic aperture and m the mean aperture. The expression in Eq 2.4 is a first approximation of the solution to the full Navier-Stokes equation. By comparing Eq 2.4 to the higher order lubrication theory (Zimmerman, et al., 1991) given by:

$$w_h^3 = w_m^3 \frac{(1-\delta^2)^{\frac{5}{2}}}{1+\left(\frac{\delta^2}{2}\right)} \left[1 - \frac{6\pi^2(1-\delta^2)\delta^4}{10\left[1+\left(\frac{\delta^2}{2}\right)\right]} \left(\frac{w_m}{\lambda}\right)^2\right] \tag{2.5}$$

an equation containing the correction due to non-zero values of the ratio between the standard deviation, σ, and the wavelength can be obtained (Zimmerman, et al., 1991). The standard deviation can be expressed, as follows:

$$\sigma = \frac{w_m \delta}{\sqrt{2}} \tag{2.6}$$

Substituting Eq 2.6 into the second term in brackets in Eq. 2.5, yields:

$$C_1(\delta) \cdot \left(\frac{\sigma}{\lambda}\right)^2 \tag{2.7}$$

where

$$C_1(\delta) = \frac{6\pi^2}{5} \frac{(1-\delta^2)\delta^2}{1+\left(\frac{\delta^2}{2}\right)} \tag{2.8}$$

By definition the magnitude of the roughness is within the range:

$$0 < \delta < 1 \tag{2.9}$$

The correction term, C_1, is at its maximum when the wavelength is at its largest. The maximum value of the C_1 is equal to 2.39 at $\delta \approx 0.67$, seen in Fig. 2.3.1.

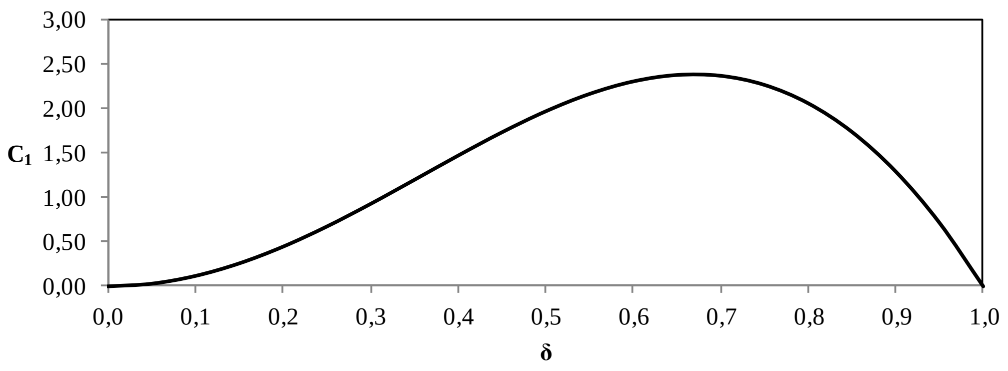


Fig. 2.3.1 - $C_1(\delta)$ plotted for different values of roughness.

By assuming that the relative magnitude of the correction term is less or equal to 0.1 of the value predicted by the lubrication theory (Zimmerman, et al., 1991), Eq 2.7 becomes:

$$2.39\left(\frac{\sigma}{\lambda}\right)^2 < 0.10 \tag{2.10}$$

Rearranging yields

$$\frac{\sigma}{\lambda} < \sqrt{\frac{0,10}{2,39}} \approx 0.2 \tag{2.11}$$

The standard deviation, or root mean square (RMS) on a grid with M cells in both x-direction and y-direction can be calculated, as follows:

$$RMS = \sqrt{\frac{1}{(M+1)^2 - 1}\sum_{i=0}^{i=M}\sum_{j=0}^{j=M}(w(x,y) - w_m)^2} \tag{2.12}$$

The RMS and the grid spacing in both x and y-direction of a square fracture determines whether or not the numerical fracture is within the lubrication theory approximation, as follows:

$$\frac{RMS}{\Delta x} < 0.2 \tag{2.13}$$

$$\frac{RMS}{\Delta y} < 0.2 \tag{2.14}$$

2.4 Rheological effects on mud losses

Drilling in naturally fractured formations can lead to significant fluid loss and in some cases cause severe drilling problems. The industry uses various methods for preventing mud loss such as proper fluid selection and by deployment of Lost Circulation Materials (LCM). The latter method is a fluid system consisting of particles of varying size designed to bridge the fracture aperture, essentially reducing or curing the lost circulation. The particles added to the fluid system typically range from one micrometer to the required bridging width of the fracture.

Various designer mud systems have been developed with different areas of application with different results of curing the mud losses. E.g LCM can be added to the fluid system prior to entering the vulnerable formation as pre-treatment or it can be added when the incident occurs. The amount necessary to avoid and cure lost circulation varies extensively and it is difficult to calculate the probability of lost circulation incidents. In addition, use of heavy LCM systems can permanently damage the productivity of the well in the fracture areas.

Majidi et al. (2010) showed that the rheological properties of the fluid system greatly affect mud losses in fractured formations and that the fluid system should be designed according to the rheological parameters in conjunction with LCM. A mathematical model was developed to investigate the effect of yield stress and shear-thinning on mud losses in fractured formations. A constant overpressure was assumed and different cases with different rheological parameters were tested. Majidi et al. (2010) showed that the ultimate mud losses were essentially controlled by the yield stress. The ultimate mud losses significantly decrease with increasing yield stress as shown in Fig. 2.4.1a.

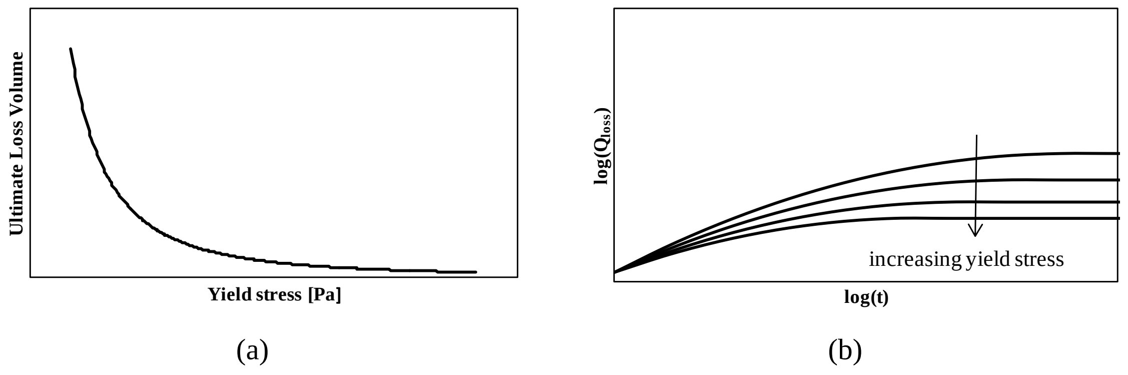


Fig. 2.4.1 - (a) The effect of yield stress on ultimate volume of losses (b) The effect of yield stress on cumulative volume of losses as a function of time. After Majidi et. al (2010). The ultimate lost volume is reduced for greater values of yield stress (a). The yield stress also affects the cumulative volume curves, where stationary values are reached faster for high yield stress values.

Mud losses stabilize faster for fluids with large yield stress values (Fig. 2.4.1b). It takes a long period of time before fluids with low yield stress value stabilize at the ultimate lost volume. Ultimate fluid losses were unaffected by change of plastic viscosity.

3 THEORY

In this section the governing fluid flow and particle transport equations are derived to give the reader insight in the mathematical model of the simulations. In addition, an algorithm for generating self-affine surfaces used in this study is introduced.

3.1 Fluid flow governing equations

Flow of visco-plastic fluids in complex geometries is encountered when drilling fluid escapes into natural fractures. In some cases plug regions are predicted theoretically (Lipscomb & Denn, 1984), however studies have shown that such yield surfaces cannot exist in such complex flow fields, and thus the flow and deformation must occur at all interior points (Frigaard & Ryan, 2004).

Fluid flow equations can be found by solving the conversavtion of mass and momentum conservation given by:

$$\nabla \cdot \boldsymbol{v} = 0 \tag{3.1}$$

$$\rho \dot{\boldsymbol{v}} = -\nabla p + \nabla \cdot \boldsymbol{\tau} \tag{3.2}$$

respectively, where $\boldsymbol{v}$ is the velocity vector, ρ is density, p is pressure and $\boldsymbol{\tau}$ is the shear stress vector. The lubrication theory approximation is assumed valid, thus flow is locally fully-developed (Lipscomb & Denn, 1984). The velocitity functions can be expressed as $v_x = v_x(z),\ v_z = 0$, where v_x is the velocity in the direction of flow and v_z is the velocity perpendicular to flow (Fig 3.1.1).

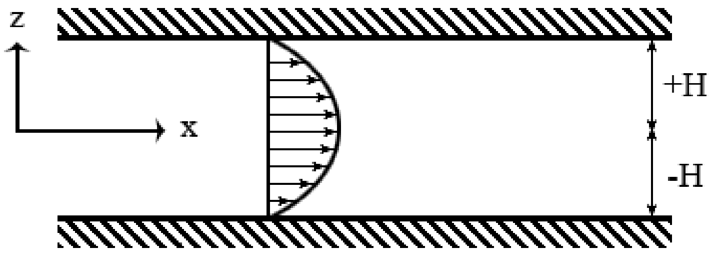


Fig. 3.1.1 - Illustration of fluid flow between two parallel plates.

Assuming incompressible flow and that the conservation of mass is satisfied, the following conclusions can be drawn:

$$\tau_{xx} = \tau_{zz} = 0 \tag{3.3}$$

$$\tau_{xz} = \tau_{xz}(z) \tag{3.4}$$

$$p = p(x) \tag{3.5}$$

where τ is the shear stress. The x-component of the momentum equation can be written, as follows:

$$\frac{d\tau_{xz}}{dz} = -\frac{\Delta p}{\Delta x} \tag{3.6}$$

Integrating on both sides yields:

$$\tau_{xz} = -\frac{\Delta p}{\Delta x}\int dz = \frac{\Delta p}{\Delta x}\, z + C_1 \tag{3.7}$$

Due to assumed symmetry around the x-axis, $C_1 = 0$. The shear stress, τ_{xz}, exceeds the yield stress, τ_y, for values where:

$$|z| \geq z_0 = \frac{\tau_y}{\left(\frac{\Delta p}{\Delta x}\right)} \tag{3.8}$$

At the walls, $z = \pm H$, there is no flow because non-slip boundary condition is assumed at the fracture walls, thus $v_x = 0$. By integrating Eq. 3.7 an expression for velocity in x-direction can be obtained (Lipscomb & Denn, 1984), as follows:

$$v_x = -\left(\frac{z^2 - H^2}{2\mu}\right)\left(\frac{\Delta p}{\Delta x}\right) + \frac{\tau_y}{\mu}(|z| - H), \qquad z_0 \leq |z| \leq H \tag{3.9}$$

substituting Eq. 3.8 ($z = \pm z_0$) into the Eq. 3.9 the following expression is obtained for the velocity of the plug in the central region:

$$v_x = -\left(\frac{z_0^2 - H^2}{2\mu}\right)\left(\frac{\Delta p}{\Delta x}\right) + \frac{\tau_y}{\mu}(z_0 - H), \qquad |z| \leq z_0 \tag{3.10}$$

We then consider the case of pressure-driven flow between rough surfaces. Assuming that the angles of the planes are small, the lubrication theory can be assumed valid, essentially meaning that the flow is locally fully-developed in a conduit of varying aperture:

$$\frac{\delta\tau_{xy}}{\delta z} = \left|\frac{dp}{dx}\right| \tag{3.11}$$

Integrating Eq. 3.11 yields:

$$\tau_{xz} = \left|\frac{dp}{dx}\right| z \tag{3.12}$$

A yield surface exists at:

$$z_0 = -\frac{\tau_y}{\left|\frac{dp}{dx}\right|} \tag{3.13}$$

Substituting the expression into the x-component of the momentum equation:

$$v_x = -\left(\frac{z_0^2 - H(x)^2}{2\mu}\right)\left(\frac{dp}{dx}\right) + \frac{\tau_y}{\mu}\left(z_0 - H(x)\right), \qquad |z| \le z_0 \tag{3.14}$$

The flow rate per unit width is equal to:

$$q_x = \frac{2\,H^2(x)\,\tau_y}{3\mu}\left[-\frac{H(x)}{\tau_y}\frac{dp}{dx}\right] \cdot \left(1 - \frac{3}{2}\left[-\frac{H(x)}{\tau_y}\frac{dp}{dx}\right]^{-1} + \frac{1}{2}\left[-\frac{H(x)}{\tau_y}\frac{dp}{dx}\right]^{-3}\right), z_0 < H \tag{3.15}$$

where

$$H(x) = \frac{w(x)}{2} \tag{3.16}$$

and w(x) is the aperture distribution along the x-axis. A similar procedure can be used to obtain the flow rate in y-direction for a flow in a three-dimensional fracture. Substituing Eq. 3.17 into Eq. 3.16 and rearranging yield:

$$q_x = \begin{cases} 0 \qquad , & \left|\frac{\partial p}{\partial x}\right| < \frac{2\tau_y}{w} \\ -\frac{w^3}{12\mu}\frac{\partial p}{\partial x} - \frac{1}{3\mu}\frac{\tau_y^3}{|\partial p/dx|^3}\frac{\partial p}{\partial x} + \frac{1}{4\mu}\frac{w^2\tau_y}{|\partial p/\partial x|}\frac{\partial p}{\partial x}\,, & \left|\frac{\partial p}{\partial x}\right| \ge \frac{2\tau_y}{w} \end{cases} \tag{3.17}$$

$$q_y = \begin{cases} 0 \qquad , & \left|\frac{\partial p}{\partial y}\right| < \frac{2\tau_y}{w} \\ -\frac{w^3}{12\mu}\frac{\partial p}{\partial y} - \frac{1}{3\mu}\frac{\tau_y^3}{|\partial p/dy|^3}\frac{\partial p}{\partial y} + \frac{1}{4\mu}\frac{w^2\tau_y}{|\partial p/\partial y|}\frac{\partial p}{\partial y}\,, & \left|\frac{\partial p}{\partial y}\right| \ge \frac{2\tau_y}{w} \end{cases} \tag{3.18}$$

3.2 Particle transport governing equation

The solute transport in rock fractures is valid under the assumption that the lubrication theory is valid and that the flow is incompressible. The concentration equation (Lavrov, 2011) can be written as follows:

$$\frac{\delta C}{\delta t} + \boldsymbol{v}\,\boldsymbol{\nabla C} = 0 \tag{3.19}$$

where $C = f(x, y, z, t)$ is the solute concentration and $\boldsymbol{v}$ is the fluid velocity vector. The governing equation in scalar form is:

$$\frac{\delta C}{\delta t} + v_x \frac{\delta C}{\delta x} + v_y \frac{\delta C}{\delta y} = 0 \tag{3.20}$$

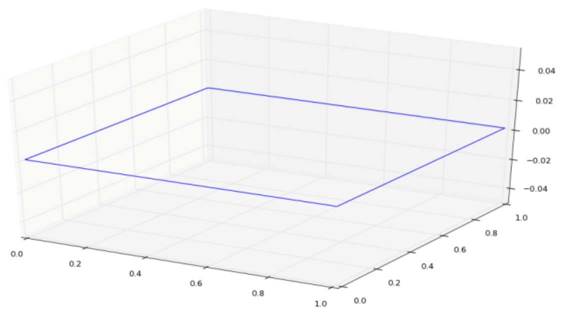
v_x and v_y are the velocities in x- and y-direction averaged across the fracture aperture, as follows:

$$v_x = \frac{1}{w} \int_{-w/2}^{w/2} v_x(z)\,dz \tag{3.21}$$

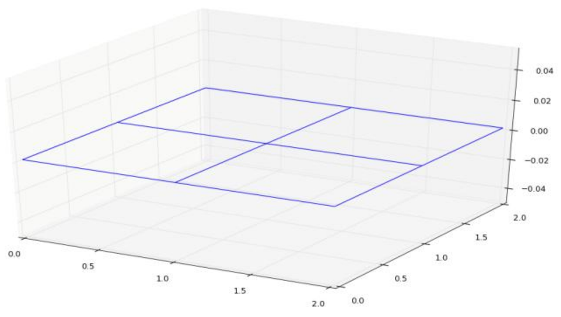
$$v_y = \frac{1}{w} \int_{-w/2}^{w/2} v_y(z)\,dz \tag{3.22}$$

3.3 Recursive subdivision technique

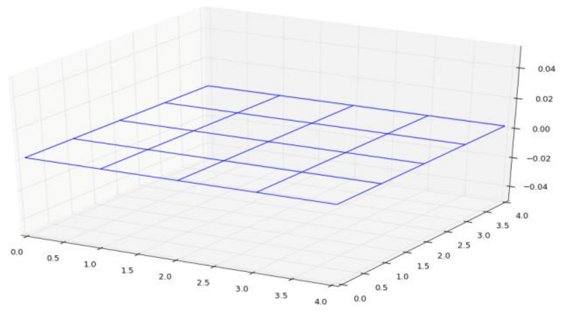
The recursive subdivision technique is an algorithm for generating a self-affine surface. Each existing cell is subdivided recursively, meaning that a new midpoint is created and the number of cells are expanded. The number of levels, denoted N, is the variable that determines the depth of recursion.The algorithm has both *internal* and *external* consistency, meaning that the object is continous in any scale and that the 3D-orientation is unaltered when rotated or resized (Fournier, et al., 1982). The steps of recursion are illustrated in Fig 3.3.1.



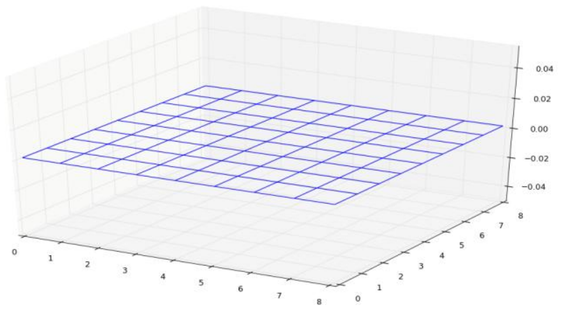
(a) N = 0



(b) N = 1



(c) N = 2



(c) N = 3

Fig. 3.3.1 - The number of level represents the number of times each cell is subdivided into four new cells. Different levels are represented in (a-d).

Each fracture cell consists of four nodes, thus the total number of nodes is equal to:

$$(2^N + 1)(2^N + 1) \tag{3.23}$$

Elevations at corner nodes are set equal to zero. The boundary nodes are recursively subdivided level by level. The elevation at boundary nodes placed at $i = 0, j \in [0,2^N]$ and $i = 2^N, j \in [0,2^N]$ is calculated by adding the elevation values from the two sourrounding nodes vertically. The elevation at boundary nodes placed at $j = 0, i \in [0,2^N]$ and

$j = 2^N, i \in [0,2^N]$ is calculated by adding values of the two sourrounding nodes horizontally. In addition, a fractional brownian motion term, F^*, is added to each data point in order to obtain a realistic fracture surface expressed as follows:

$$F^* = f(0,1) \cdot f_s \cdot C^{-H \cdot n} \tag{3.24}$$

where $f(1,0)$ is the standard normal distribution, f_s is the scaling factor, C is the c-value, n is the level of recursion and H is the Hurst exponent.

The elevation at the inner points of the grid is computed using diagonal elements, meaning that the surrounding values of surface height at the four nodes and the fBm-term are added (Fournier, et al., 1982).

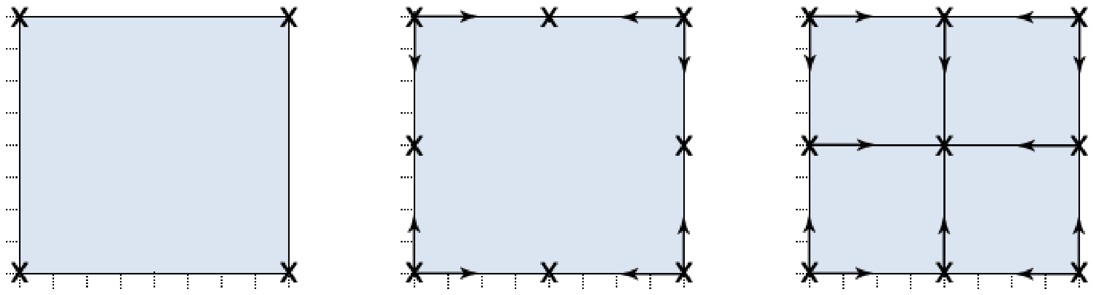


Fig. 3.3.2 - The recursive subdivision technique first generates the nodes on the four sides of the grid. Each node is calculated by using the sourrounding two nodes. The inner points are created by using the surface elevation values at the the sourrounding four nodes, as shown in (a-c).

3.4 Numerical fracture properties

Two fracture surfaces are created by using the algorithm described in section 3.3. Apertures at all node points are calculated as follows:

$$w_{i,j} = H_1(N_{i,j}) - H_2(N_{i,j}) + \alpha, \qquad i,j = 0,1,..2^N \tag{3.25}$$

where $w_{i,j}$ is the local aperture, α is the separation distance and $H_1(N_{i,j})$ and $H_2(N_{i,j})$ are the elevation height of the two surfaces. When the two surfaces overlap, the aperture value is set equal to zero. The separation between the two surfaces affects the aperture and influences the number of the nodes that are fully closed ($w_{i,j} = 0$).

The fracture aperture and topography are also determined by three other parameters, namely the Hurst exponent, c-factor and scaling factor.

3.4.1 Scaling factor

The scaling factor affects the local amplitude of the fracture surface. Low values will even out aperture towards the mean height and thus reduce the standard deviation of the apertures.

$$\lim_{\mathrm{f_s} \to 0} F^* = 0 \tag{3.26}$$

In the extreme case where the scaling factor approaches zero, the fracture aperture will approach the separation δ for all local apertures. The effects of the scaling factor are illustrated in Fig. 3.4.1.

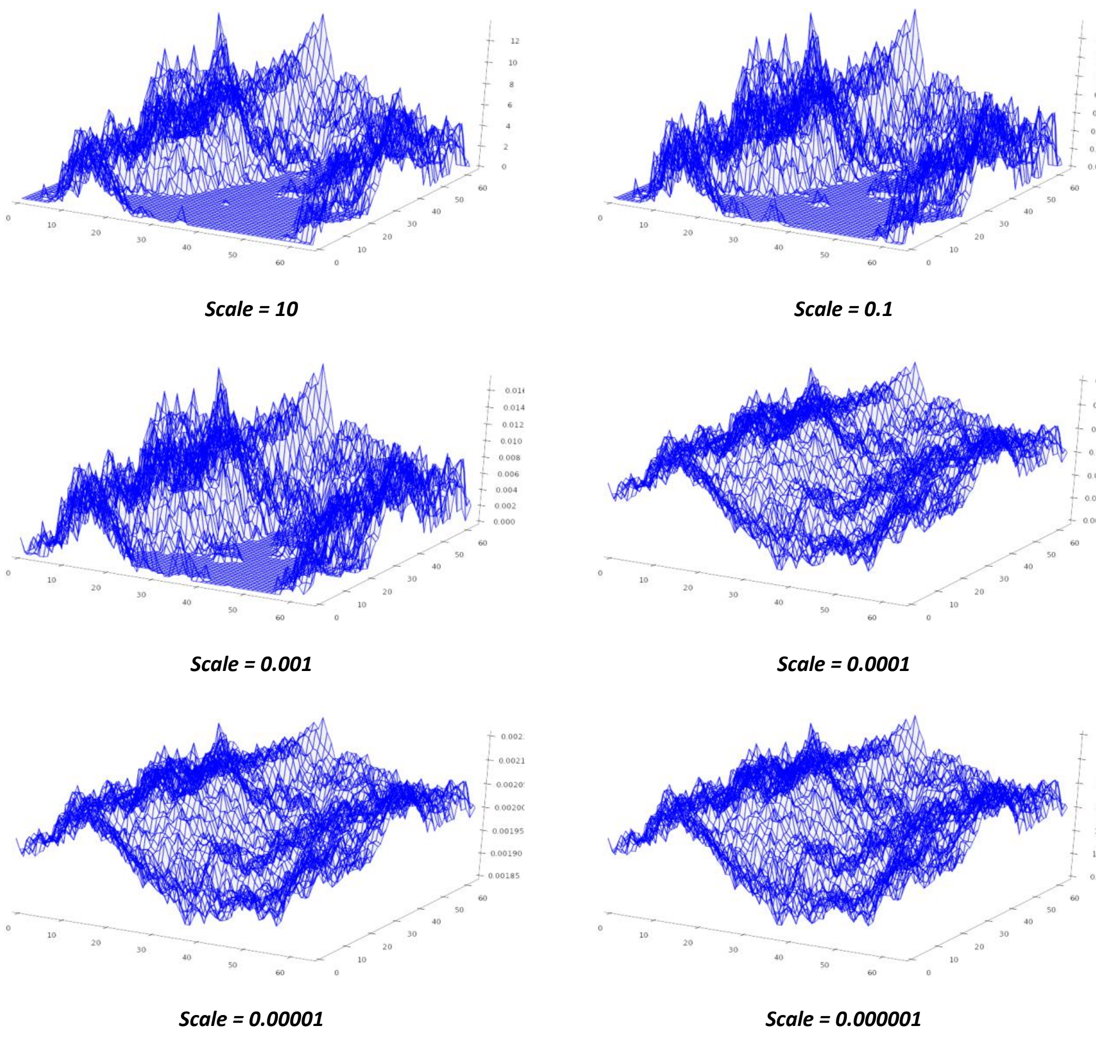


Fig. 3.4.1 - The scaling factor affects the amplitude of the fracture (Hanssen, 2012).

3.4.2 C-factor

The c-factor affects the smoothness of the terrain by merging local vertexes. For higher values the terrain becomes less rough. The effects are illustrated in Fig. 3.4.2.

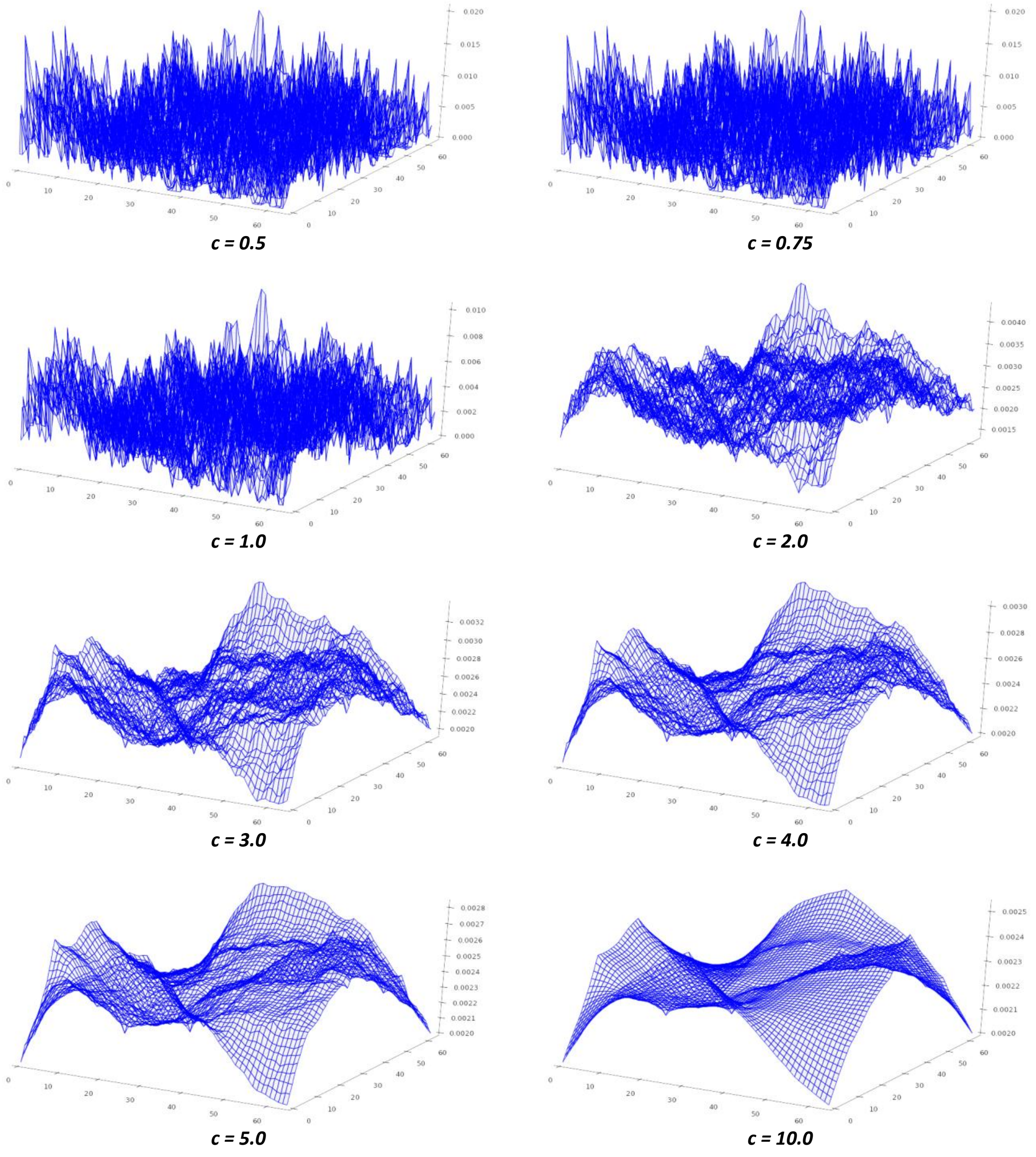


Fig. 3.4.2 - Increasing c-factor leads to greater merging of local vertexes and creates an even terrain. (Hanssen, 2012)

3.4.3 Hurst exponent

The fractal auto-correlation is determined by the Hurst exponent. By assuming that the exponent ranges from 0 to 1 and that $B(t)$ is an ordinary Brownian motion, the moving average of $dB(t)$ can be characterized as the fractional Brownian motion of the Hurst exponent (Mandelbrot & Van Ness, 1968).

For values $H > 0.5$ the series tends to have a long-range positive auto-correlation. High values are likely to be followed by high values and low values are likely to be followed by low values. For values $H < 0.5$ the series have a long-range switching between high and low values. The fractional Brownian motion is completely uncorrelated for $H = 0.5$. Fig. 3.4.3 illustrates cross-sections of computer generated fractures with varying Hurst exponent.

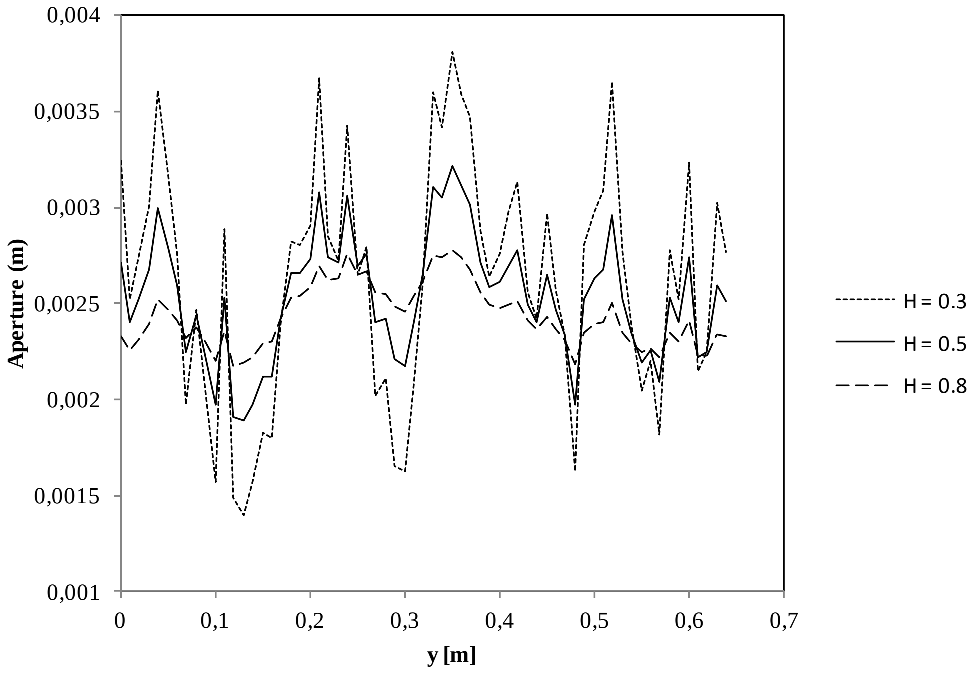


Fig. 3.4.3 - Cross section of fracture aperture with varying Hurst exponent. The solid line represent the completely uncorrelated series ($H = 0.5$). When comparing the dashed lines to the uncorrelated series you can see the switching between high and low values for $H = 0.3$ and the long-range positive auto-correlation for $H = 0.8$.

Cracks created by brittle fractures generally have a Hurst exponent close to 0.8. This is supported by a wide range of experimental data (Talon, et al., 2010).

3.5 Shear displacement

Two identical surfaces are created with the number of levels equal to:

$$N = N_{initial} + 1 \tag{3.27}$$

where $N_{initial}$ is the number of levels that the fracture will be reduced to after applied shear displacement. The upper surface is shifted by δ_x and/or δ_y nodes in the coordinate direction. The shifted area is then reduced to the initial number of levels $N_{initial}$. The area highlighted yellow in Fig. 3.5.1 represents this area.

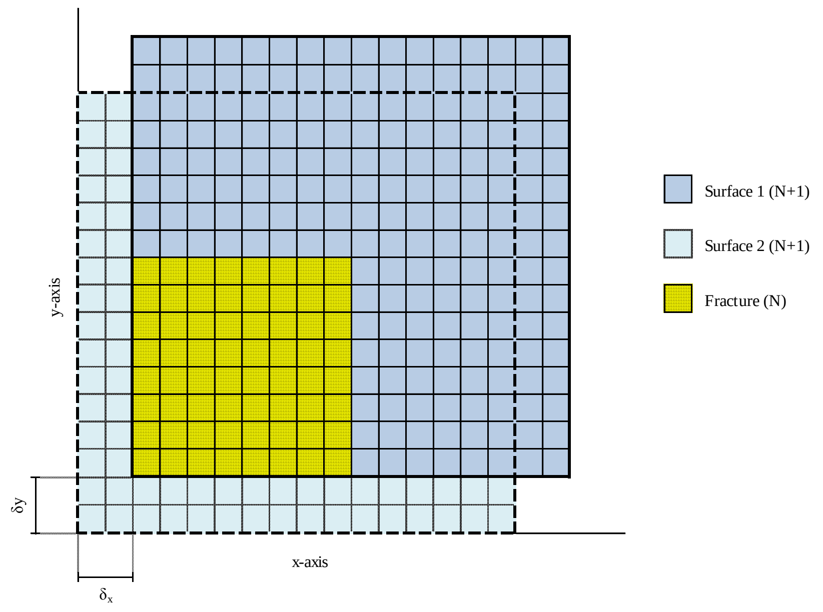


Fig. 3.5.1 - Schematic of the numerical method used for shear displacement.

The numerical method used for calculating the aperture after shear displacement is described as follows[1]:

```
for $0 < int(x) < 2^{N_{initial}}$
   for $0 < int(y) < 2^{N_{initial}}$
      Aperture[x,y] = Surface1[x,y] - Surface2[x+ $\delta_x$, y+ $\delta_y$] + $\alpha$
```

Comparing the algorithm in this study to the method used in (Koyama, et al., 2006), similar results are obtained, as shown in Fig. 3.5.2. Shear displacement leads to flow channels normal to the shearing direction and they become more pronounced for increasing shear displacement. The results from both (Koyama, et al., 2006) and (Vilarrasa, et al., 2011) are based on a technique where the aperture field is defined when the two surfaces have three contact points. In reality, a brittle fracture would have several contact areas either from direct contact between the two surfaces or from bridging material removed from the formation due to shearing. The algorithm described above is based on the distance between the surfaces, and areas where the two surfaces overlap are

[1] The full algorithm can be found in Appendix A.3.1.
[2] The pressure plots without normalization can be found in Appendix A1.

set equal to zero. Thus, greater contact areas are achieved when reducing the distance between the two surfaces. This method is suggested as an alternative method for creating shear-induced natural fractures.

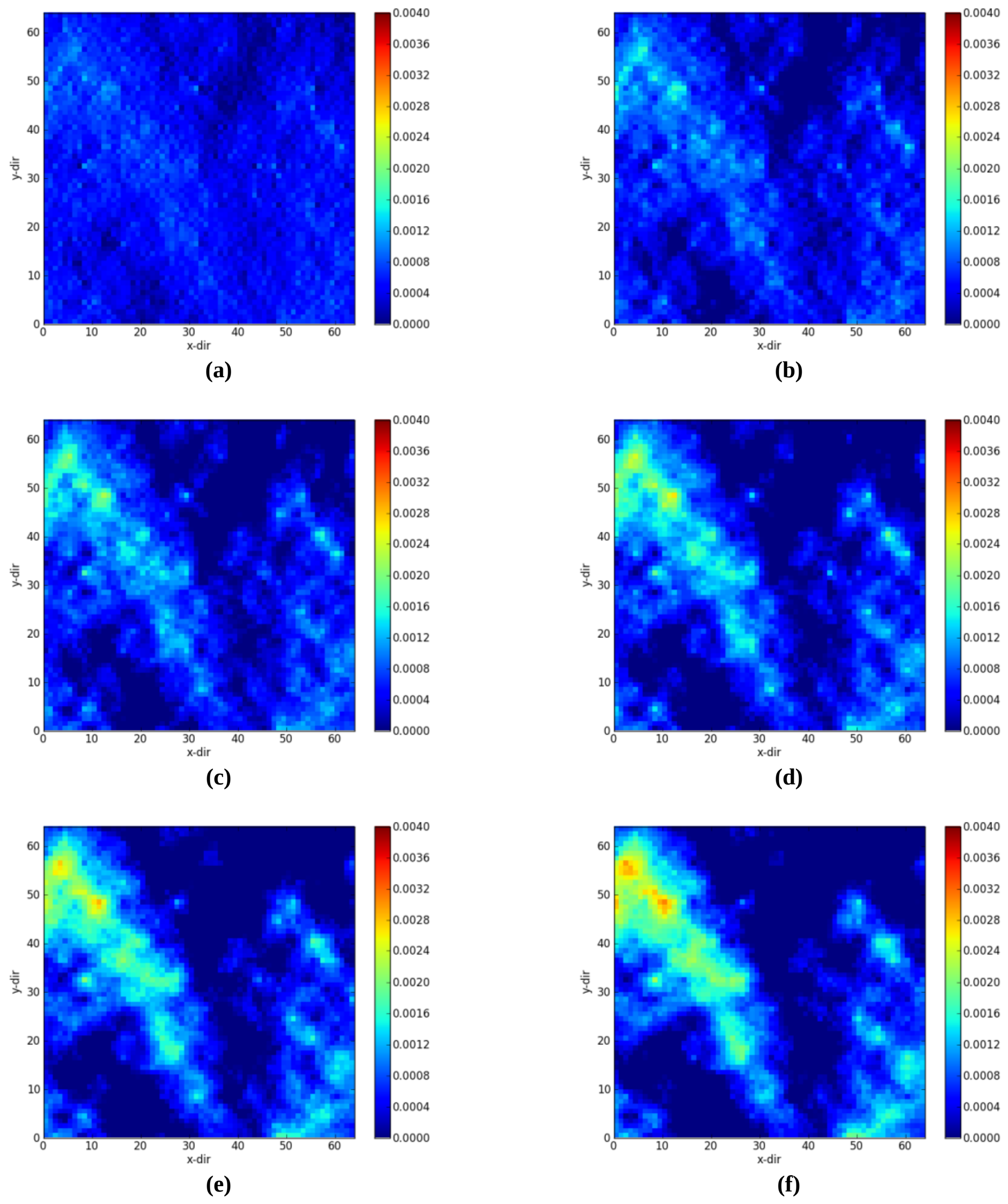


Fig. 3.5.2 - Aperture distribution for different shear displacement in x-direction. The figure illustrates the effect of increasing shear displacement (a-f). Flow channels become more pronounced with increasing shear displacement. The values of shear displacement is (a) 0.01 m, (b) 0.02 m, (c) 0.03 m, (d) 0.04 m, (e) 0.05 m and (f) 0.06 m. Aperture can be read from the color bar.

3.6 Estimation of Bingham plastic flow pressure criteria

Calculating the required pressure to avoid plugging of non-Newtonian fluid flow in rough-walled fractures is a complex task. However, good estimates can serve as approximations to the applied pressure requirements. In this section, an estimate for the required pressure difference is introduced. The estimate is used in the simulations of this study.

In the case of fluid flow between two parallel plates with constant aperture, the required pressure gradient can be calculated as follows:

$$\left|\frac{dp}{dx}\right| \geq \frac{2\,\tau}{w} \tag{3.28}$$

For complex fractures the limit cannot be calculated that easily. The aperture distribution is varying in both coordinate directions such that w in Eq. 3.28 is undefined. If the fractures do not have any overlapping areas the minimum aperture node of the distribution can serve as an over-estimated input to the criteria in Eq. 3.28. If there is an obstacle or a small area of low aperture one would over-estimate the required pressure, mainly because the fluid can simply flow around low transmissivity area. Alternatively, the arithmetic average of the aperture distribution can be used, requiring less pressure difference compared to the minimum aperture method. The problem with this method occurs when you have an aperture distribution with cross-sections of very low aperture. The average would then not be representative for these regions and thus could lead to under-estimation of the required overpressure.

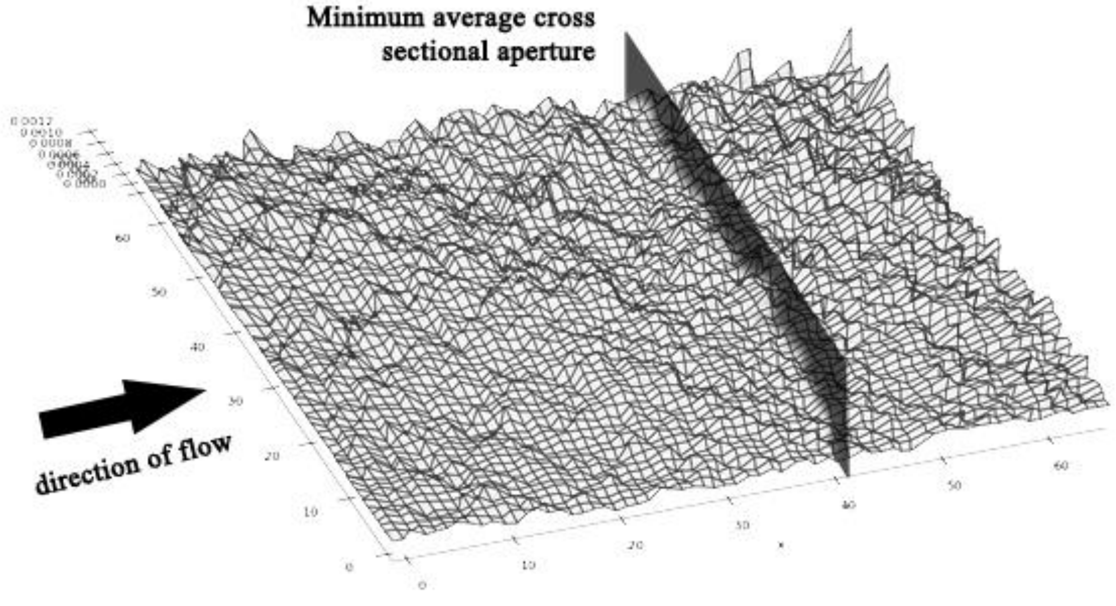


Fig. 3.6.1 - The minimum average aperture path perpendicular to flow direction.

Instead, one can calculate the average aperture of the cross-section with the minimum average aperture (Fig. 3.6.1), similar to the method used in (Talon, et al., 2010), as follows:

$$\bar{w}_{\perp,min} = \min\left\{\frac{\sum_{j=0}^{N} w_{0,j}}{N+1}, \frac{\sum_{j=0}^{N} w_{1,j}}{N+1}, \ldots, \frac{\sum_{j=0}^{N} w_{N,j}}{N+1}\right\} \tag{3.29}$$

where $\bar{w}_{\perp,min}$ is the minimum average cross-sectional aperture. The minimum applied pressure difference to obtain fluid flow for Bingham fluid is given by:

$$\Delta P = \frac{2\,\tau_y\,L_y}{\bar{w}_{\perp,min}} \tag{3.30}$$

where L_y is the length of the fracture.

4 NUMERICAL MODELING

Numerous fluid flow and particle transport models have been developed and laboratory experiments have been carried out the last decades. This is still an on-going research. ((Novotny, 1977), (Daneshy, 1978), (Amadei & Illangasekare, 1994), (Chen, et al., 2005) (Adachi, et al., 2007) and (Auradou, et al., 2010)). This chapter describes the approach used in this study.

4.1 Modeling

In this study the software *PROPANICA*, a research code developed by Alexandre Lavrov, is used to perform fluid flow and particle transport simulations. The fractures are created by a software based on the recursive subdivision technique named *PROPANICAGRIDDER*, also developed by Alexandre Lavrov. *PROPANICAGRIDDER* is modified by the author to include shifting in both x-direction and y-direction and to rotate the fracture mesh 90 degrees clock-wise in order to simulate flow perpendicular to the initial set-up.

The computer generated fracture used in this study has the following variables:

- Number of levels
- Separation between the surfaces
- Grid spacing in x- direction (Δx)
- Grid spacing in y-direction (Δy)
- Hurst exponent
- Scaling factor
- C-factor
- Randomizer

The unique and randomly created feature of each fracture is assigned a value such that the fracture can be regenerated if one wants to alter the other variables. This value is called the randomizer and is denoted R.

One rough-walled fracture with average aperture of 0.005 m was created by using the recursive subdivision technique. Roughly one percent of the local fracture apertures were closed. Input values can be found in Tab. 4.1.1:

N	**α**	**Δx**	**Δy**	**H**	**F_s**	**C**	**R**
(-)	**(m)**	**(m)**	**(m)**	**(-)**	**(-)**	**(-)**	**(-)**
6	0.02	0.01	0.01	0.75	0.02	3.00	2230

Tab. 4.1.1- Fracture surface input.

The initial fracture had a dimension of 128 x 128 cells with 0.01 m grid spacing in both x- and y-direction. One of the two fracture surfaces was shifted one node in the x-direction for the first scenario (Fig. 4.1.1) and one node in y-direction for the second scenario (Fig. 4.1.2). The shifted fractures were then reduced to a lower number of levels, equal to 64 x 64 cells. Fluid flow was performed in both the perpendicular (denoted $\boldsymbol{\delta} \perp \boldsymbol{U}$) and parallel direction (denoted $\boldsymbol{\delta} \mid\mid \boldsymbol{U}$) to shear displacement.

A total number of 108 cases with varying fluid properties of the non-Newtonian fluid and fracture specifications were completed. The fluid properties of particular interest were yield point value and plastic viscosity. In addition, fluid properties were tested for different pressure differences (i.e. entrance pressure minus exit pressure).

The applied pressure difference was kept constant at the inlet and outlet. The model implemented in the *PROPANICA* code is valid under the lubrication theory approximation and assumes fluid incompressibility and that particle dispersion is negligible. Fracture walls are assumed perfectly impermeable, meaning no fluid leak-off through the two surfaces. The cases were simulated with one particle source node placed at $i = 3$ and $j = 32$ for flow in x-direction and $i = 32$ and $j = 62$ in y-direction.

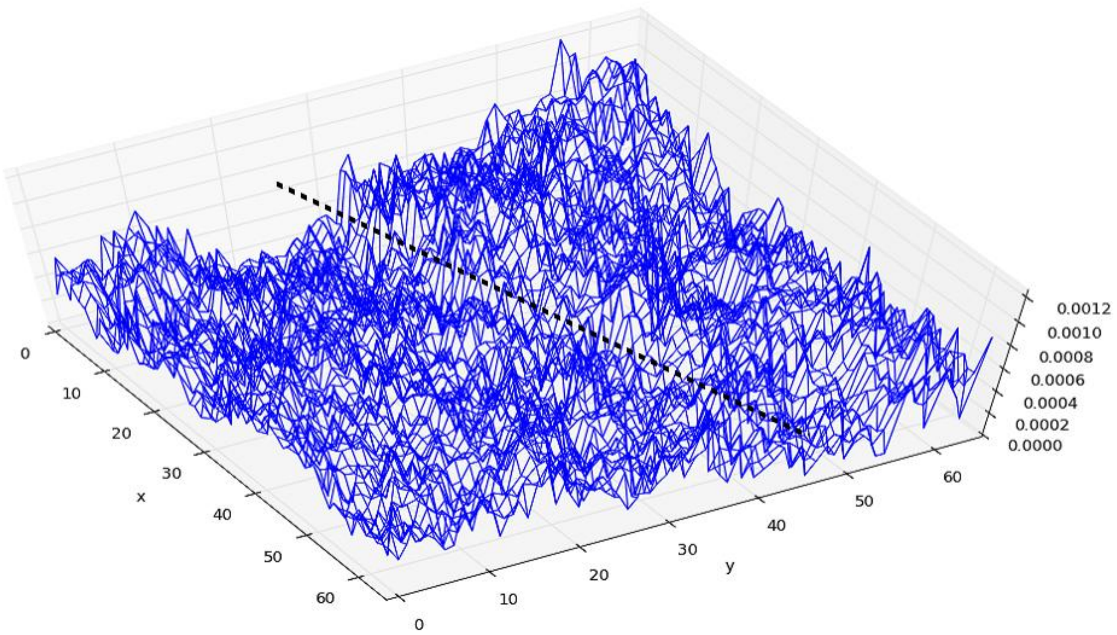


Fig. 4.1.1- 3D image of the fracture aperture. The lower fracture surface is shifted one node in the x-direction. Note the channelization perpendicular to shifting (dashed line).

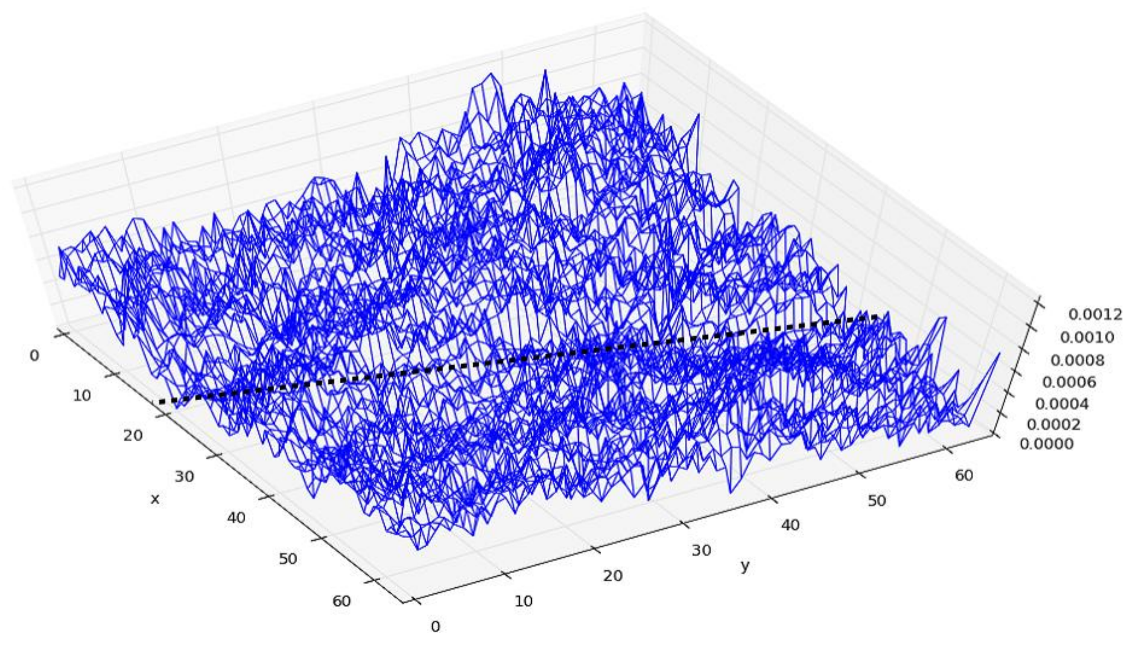


Fig. 4.1.2 - 3D image of the fracture aperture. The lower fracture surface is shifted one node in the y-direction. Note the channelization perpendicular to shifting (dashed line)

Simulations were run in decoupled mode meaning that velocity and pressure distribution are calculated before the particle transport and are not affected by particle concentration. *PROPANICA* uses an iterative solution method described in (Patankar, 1980). To achieve accurate results the iterations are looped until the relative variation in the pressure field drops below a given exit tolerance and the error in the x-flow rate between column nodes drops below a given flow rate exit tolerance. The exit tolerance values are given in Tab. 4.1.2:

Exit tolerence flow x-dir	**Exit tolerence pressure**
(-)	**(-)**
1.0e-7	1.0e-3

Tab. 4.1.2 - Calculation looping conditions.

The target time for the particle transport computation was as follows:

$$t_{target} = C_m \frac{N_x \, \Delta x}{\overline{v_x}} \tag{4.1}$$

where C_m is a user specified constant set equal to 2 for all simulations, N_x is number of nodes in either direction and $\overline{v_x}$ is the average velocity in the global flow direction. 30 time steps are created for each case ranging from 0 to t_{target}. The Courant number given by:

$$\Delta t = C \min\left\{\frac{\Delta x}{|v_{x,max}|}, \frac{\Delta y}{|v_{y,max}|}\right\} \tag{4.2}$$

which controls the time step of the particle transport and was set equal to 0.5 for all cases. The semi-Lagrangian method is used to compute the particle transport. Based on this computation, particle concentration on the grid is obtained at each time step, and breakthrough curves are plotted at locations specified by the user. The semi-Lagrangian method is unconditionally stable for Courant numbers less than unity (Staniforth & Côté, 1991).

4.2 Test matrix

The test matrix (Tab. 4.2.1) consists of a comprehensive sensitivity analysis of each of the four scenarios illustrated in Fig. 4.2.1. A sensitivity analysis on the fluid rheology was completed. In addition, fluid channelization due to shear displacement and varying non-dimensional yield stress was studied. The non-dimensional yield stress is given by:

$$T = \left(\frac{2\,\tau_y\,L_x}{w_{av}\,\Delta P}\right) \tag{4.3}$$

The ratio of the average velocities in the coordinate direction can be written as follows:

$$\frac{V_{||P,avg}}{V_{\perp P,avg}} \tag{4.4}$$

where $V_{||P,avg}$ represent the velocity in along direction of flow and is $V_{\perp P,avg}$the velocity perpendicular to flow. All cases were post-processed individually with pressure fields, velocity distribution and breakthrough curves calculated by the flow equations found in section 3.1. The breakthrough curves calculate the particle concentration at node lines perpendicular to flow as a function of time, as follows:

$$\sum_{j=1}^{N_y} c_{ij} w_{ij} \Delta y \tag{4.5}$$

where c is concentration.

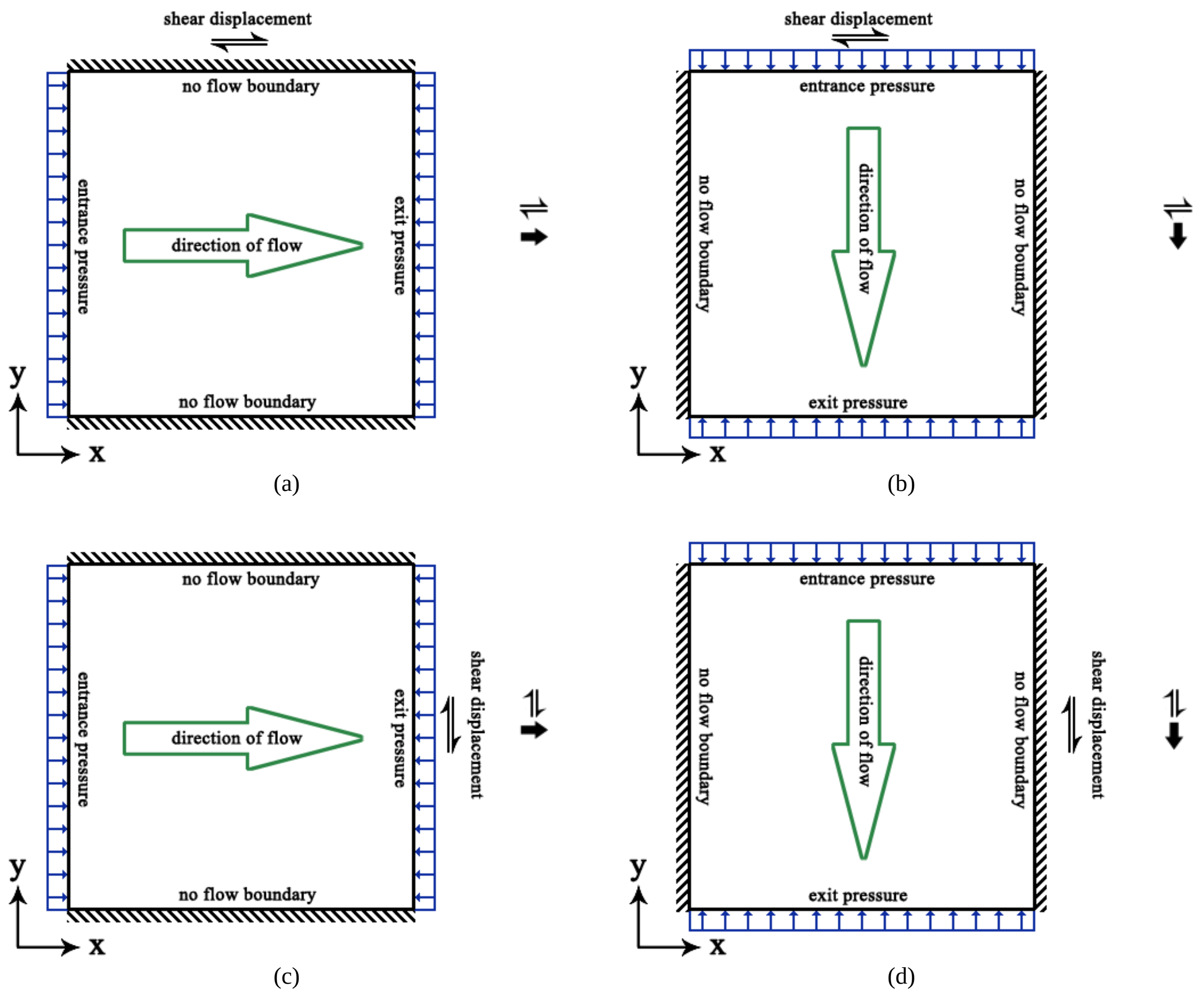


Fig. 4.2.1 - Schematic view of different simulation scenarios. The icons on the right side of the different scenario refers to the different cases further on in this study.

Case number				ΔP	τ_y	μ_{pl}	H	Lx = Ly	w_{avg}	T
a	**b**	**c**	**d**	**(Pa)**	**(Pa)**	**(Pas)**	**(-)**	**(m)**	**(m)**	**(-)**
1	28	55	82	500000	0	0.001	0.75	0.64	4.8e-4	0.0000
1	29	56	83	500000	0	0.010	0.75	0.64	4.8e-4	0.0000
3	30	57	84	500000	0	0.030	0.75	0.64	4.8e-4	0.0000
4	31	58	85	500000	25	0.001	0.75	0.64	4.8e-4	0.1333
5	32	59	86	500000	25	0.010	0.75	0.64	4.8e-4	0.1333
6	33	60	87	500000	25	0.030	0.75	0.64	4.8e-4	0.1333
7	34	61	88	500000	50	0.001	0.75	0.64	4.8e-4	0.2665
8	35	62	89	500000	50	0.010	0.75	0.64	4.8e-4	0.2665
9	36	63	90	500000	50	0.030	0.75	0.64	4.8e-4	0.2665
10	37	64	91	350000	0	0.001	0.75	0.64	4.8e-4	0.0000
11	38	65	92	350000	0	0.010	0.75	0.64	4.8e-4	0.0000
12	39	66	93	350000	0	0.030	0.75	0.64	4.8e-4	0.0000
13	40	67	94	350000	25	0.001	0.75	0.64	4.8e-4	0.1904
14	41	68	95	350000	25	0.010	0.75	0.64	4.8e-4	0.1904
15	42	69	96	350000	25	0.030	0.75	0.64	4.8e-4	0.1904
16	43	70	97	350000	50	0.001	0.75	0.64	4.8e-4	0.3808
17	44	71	98	350000	50	0.010	0.75	0.64	4.8e-4	0.3808
18	45	72	99	350000	50	0.030	0.75	0.64	4.8e-4	0.3808
19	46	73	100	200000	0	0.001	0.75	0.64	4.8e-4	0.0000
20	47	74	101	200000	0	0.010	0.75	0.64	4.8e-4	0.0000
21	48	75	102	200000	0	0.030	0.75	0.64	4.8e-4	0.0000
22	49	76	103	200000	25	0.001	0.75	0.64	4.8e-4	0.3332
23	50	77	104	200000	25	0.010	0.75	0.64	4.8e-4	0.3332
24	51	78	105	200000	25	0.030	0.75	0.64	4.8e-4	0.3332
25	52	79*	106	250000	50	0.001	0.75	0.64	4.8e-4	0.5330
26	53	80*	107	250000	50	0.010	0.75	0.64	4.8e-4	0.5330
27	54	81*	108	250000	50	0.030	0.75	0.64	4.8e-4	0.5330

Tab. 4.2.1 - Test Matrix - The case number labels (a-d) refers to the different simulation scenarios (a-d) from Fig. 4.2.1. The pressure calculations in simulations 79-81 did not converge at the initial T. The overpressure was increased to 270 000 Pa to achieve convergence.

4.3 Post processing of data

This section outlines some of the methods used to present the results in this study. A detailed flow chart of the simulation procedure is provided in Appendix A2.

4.3.1 Pressure plots

The pressure plots in this thesis had to be normalized in order to be able to compare the different distributions due to varying fluid properties[2]. Very little difference would be visible otherwise. The following procedure was used to create normalized pressure plots:

The pressure scaling value is given by:

$$F_{scale} = \frac{\max\{p[x,y]_{ref}\}}{\max\{p[x,y]_{BH}\}} \tag{4.6}$$

where ref denotes the Newtonian reference case, and BH denotes the different Bingham cases with varying T. The Newtonian case has the largest applied pressure gradient and thus:

$$\max\{p[x,y]_{ref}\} \geq \max\{p[x,y]_{BH}\} \tag{4.7}$$

The normalized values of the fluid pressure, p^*, used for comparing the different cases with the reference case are calculated as follows:

$$p^*[x,y] = \frac{F_{scale}\, p[x,y]_{BH} - p[x,y]_{ref}}{p[x,y]_{ref}} \tag{4.8}$$

where

$$-1 \leq p^*[x,y] \leq 1 \tag{4.9}$$

This method was developed by the author to emphasize the difference in pressure distribution due to varying fluid properties.

4.3.2 Velocity fields

The original velocity plots were difficult to compare due to the initial resolution. Instead the velocity plots presented display every fourth vector in the mesh in order to make it

[2] The pressure plots without normalization can be found in Appendix A1.

easier to the different cases. The method is valid because the fluid flow follows streamlines in the fracture grid. The original velocity fields can be found in Appendix A1.

5 RESULTS

This section presents the results from the simulations. Raw velocity data are included in Appendix A.6.1-4 for future references. The results are later discussed in Chapter 6.

Particle transport become more clustered and channelized for increasing non-dimensional yield stress. In addition, the fluid's ability to carry particles is enhanced for greater values of T, as seen in Fig 5.1.1-2.

For increasing non-dimensional yield stress, the normalized pressure increases upstream of regions with contact areas between the two surfaces. The effect is enhanced for increasing non-dimensional yield stress, particularly when the applied pressure gradient is parallel to the shear displacement as seen in Fig 5.2.1-2.

The velocity distributions become more channelized for increasing non-dimensional yield stress, meaning that the ratio of the velocity in the global flow direction to the velocity perpendicular to global flow direction increases.

5.1 Particle transport

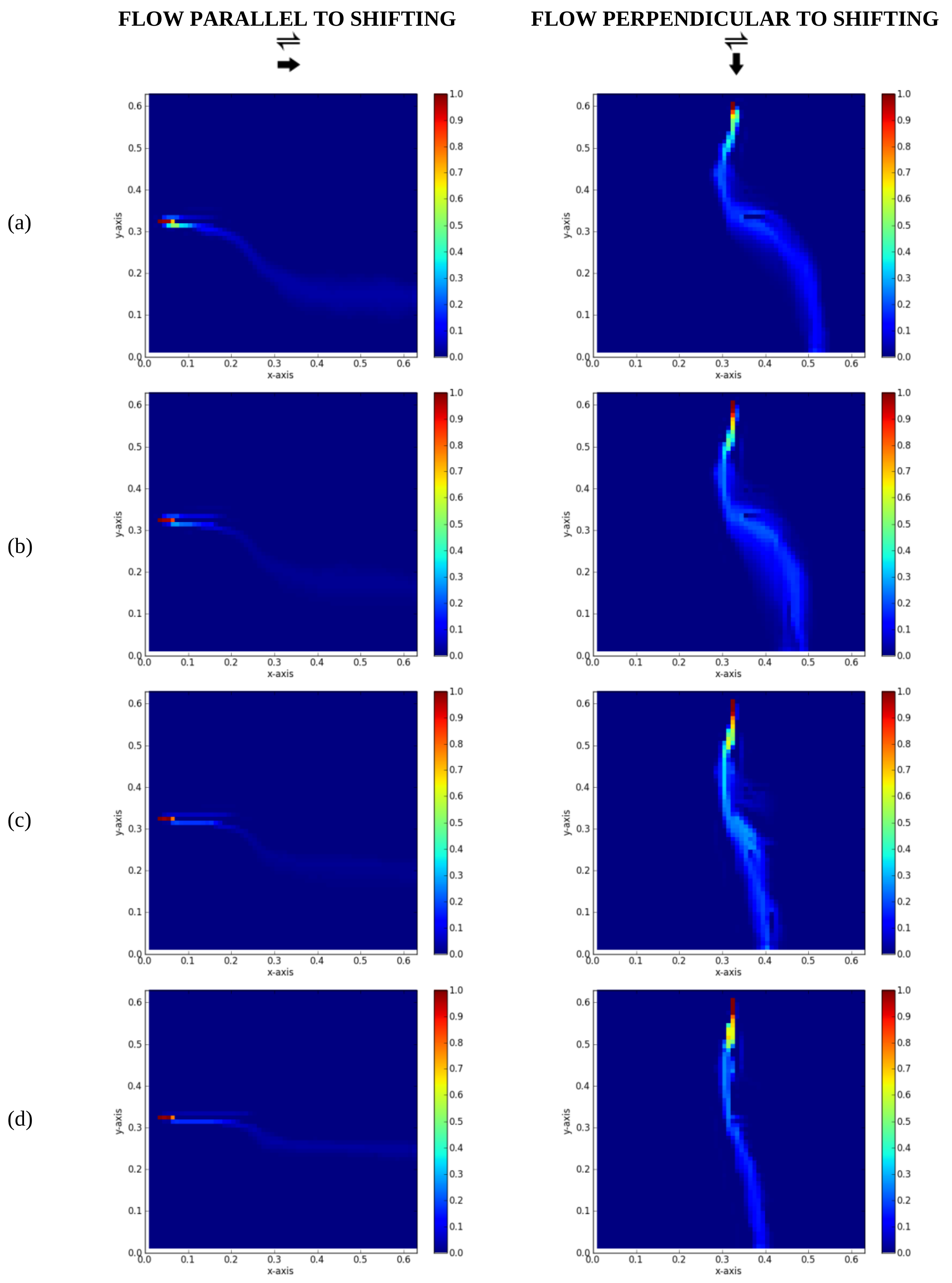


Fig. 5.1.1 - Particle concentration for increasing T values: (a) 0.00 (b) 0.19 (c) 0.38 and (d) 0.53 for left column and 0.50 the right column. Shear displacement along x-axis. Concentration values can be read from the color bar on the right-side.

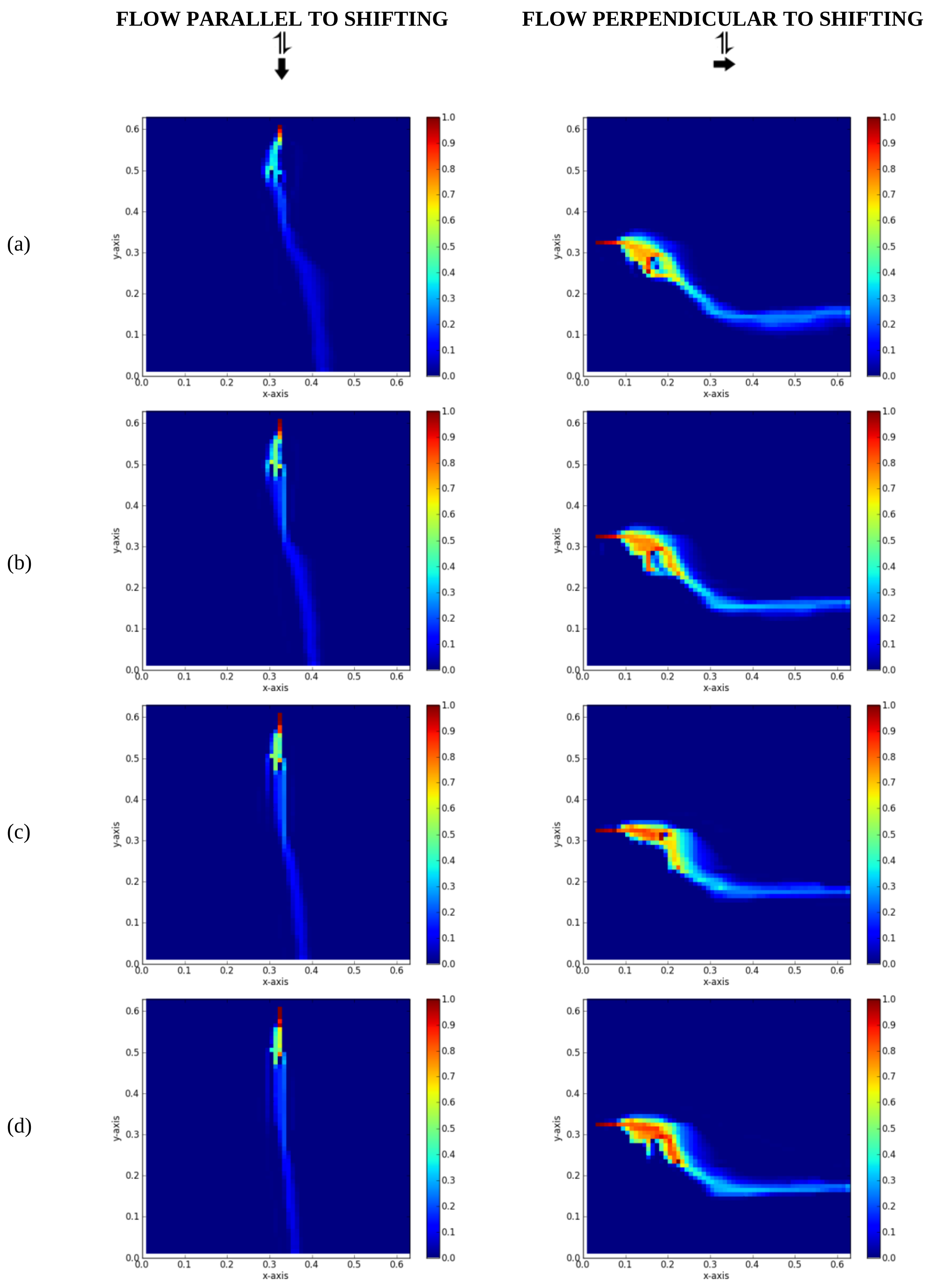


Fig. 5.1.2 - Particle concentration for increasing T values: (a) 0.00 (b) 0.18 (c) 0.37 and (d) 0.53. Shear displacement along y-axis. Concentration values can be read from the color bar on the right-side.

5.2 Pressure distribution

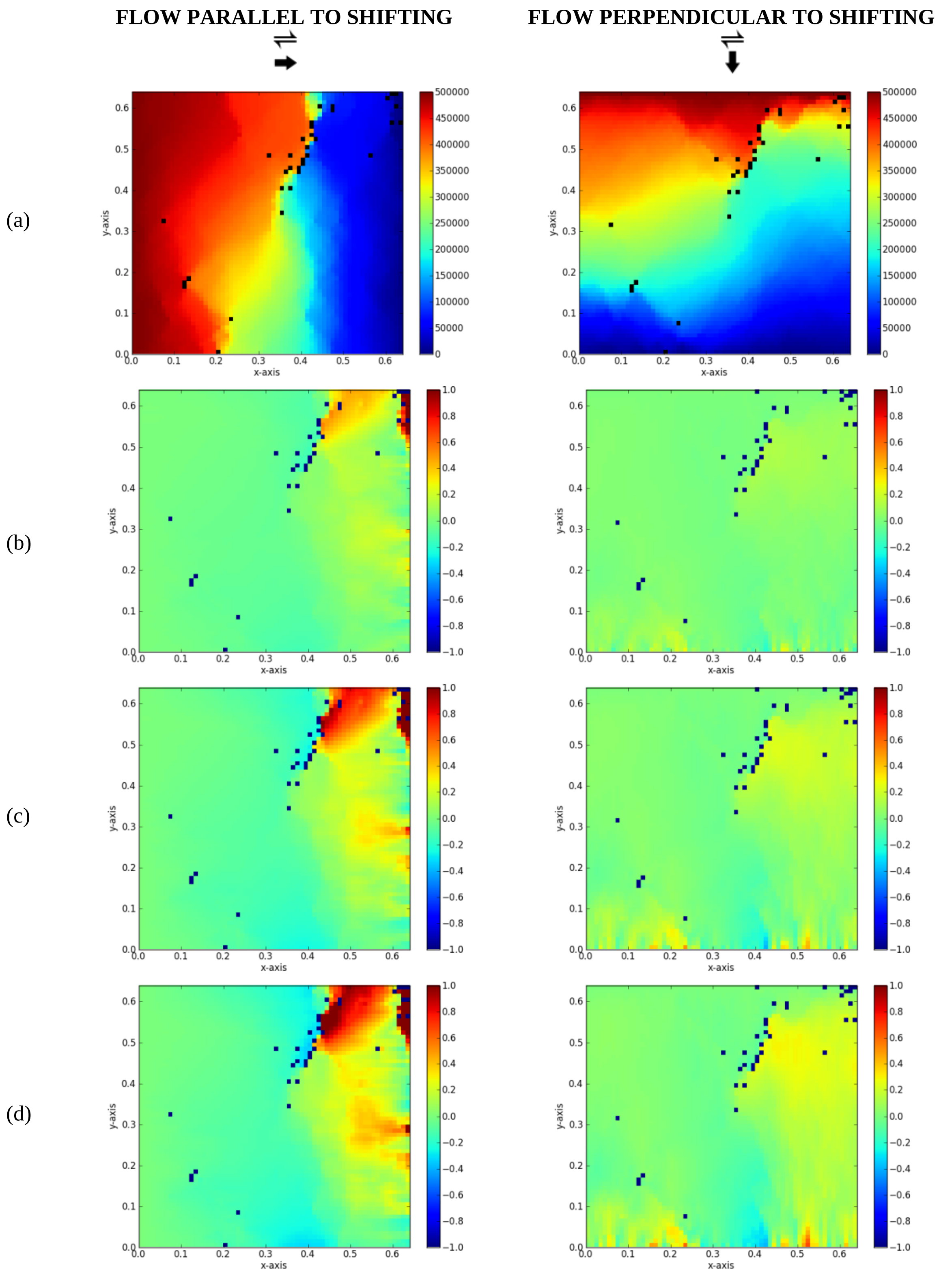


Fig. 5.2.1 - Pressure distributions of Bingham plastic compared to Newtonian fluid flow (a) for increasing T (b-d) values as follows: (b) 0.19 (c) 0.38 and (d) 0.53 for left column and 0.50 the right column. Shear displacement along x-axis. Normalized pressure values can be read from the color bar on the right-side. Dark blue dots represent closed local aperture.

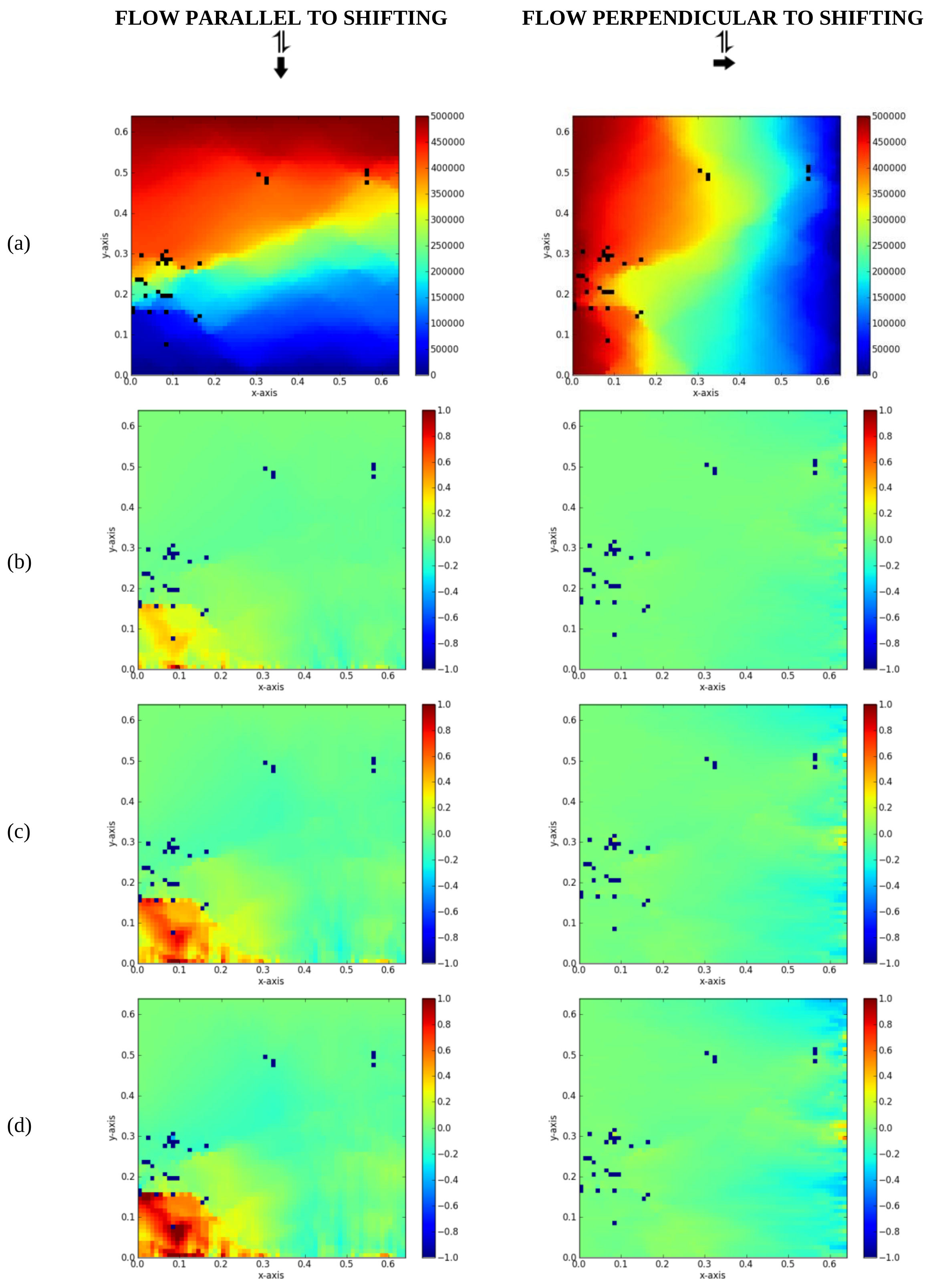


Fig. 5.2.2 - Pressure distributions of Bingham Plastics compared to Newtonian fluid flow (a) for increasing T (b-d) values as follows: (b) 0.18 (c) 0.37 and (d) 0.53. Shear displacement along y-axis. Normalized pressure values can be read from the color bar on the right-side Dark blue dots represent closed local aperture.

5.3 Velocities

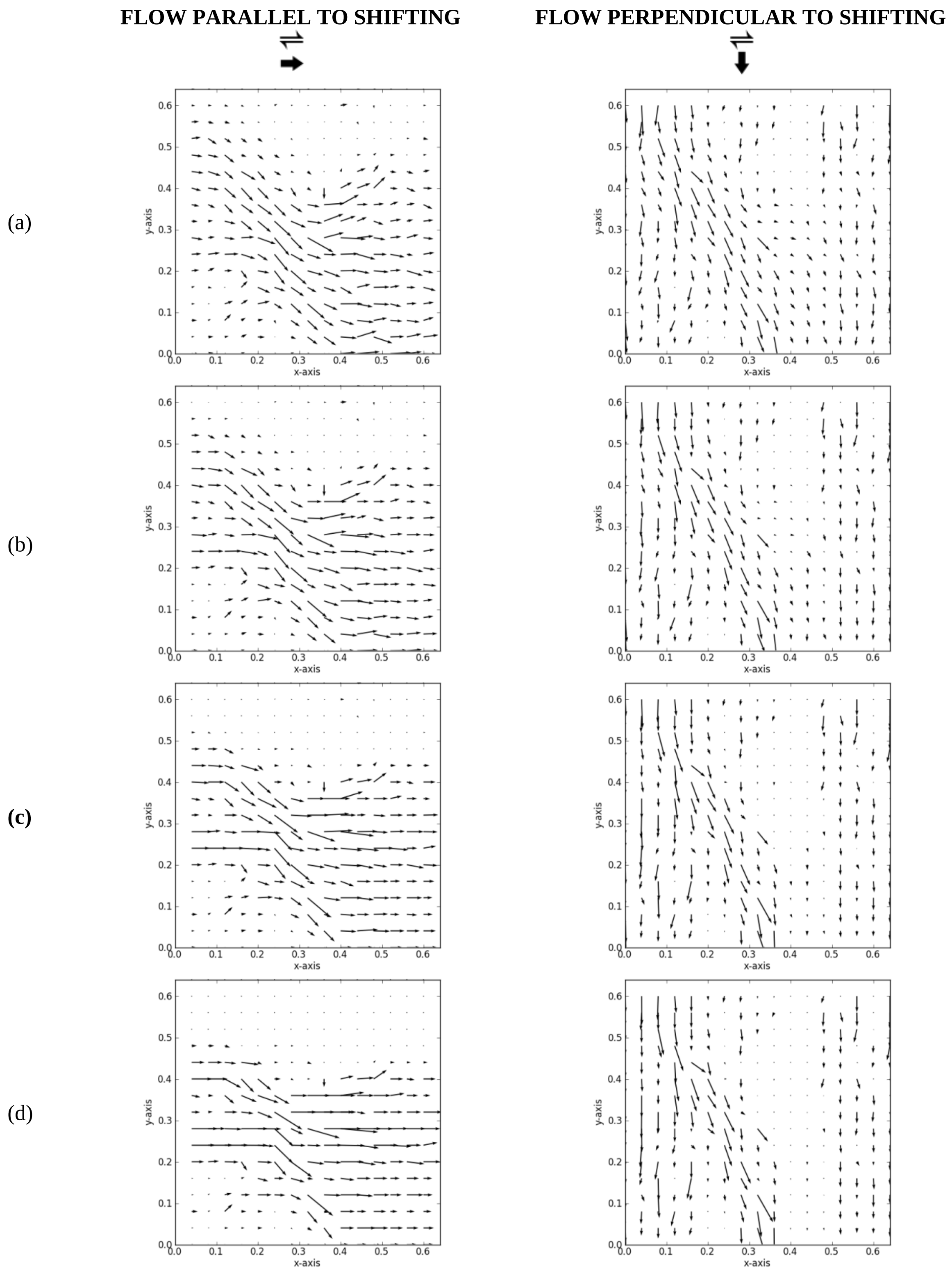
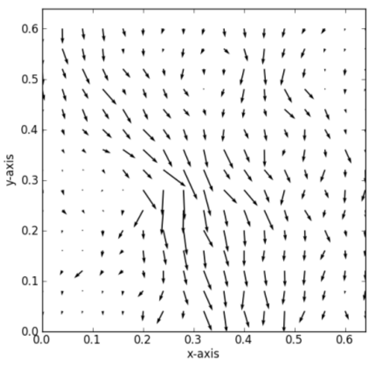
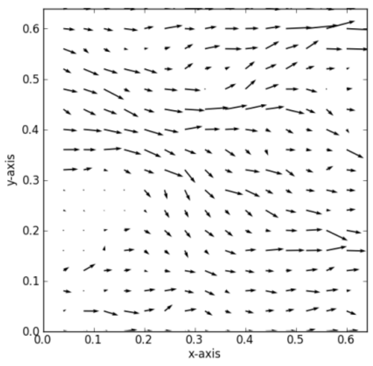


Fig. 5.3.1 - Velocity distribution for increasing T values: (a) 0.00 (b) 0.19 (c) 0.38 and (d) 0.53 for left column and 0.50 the right column. Shear displacement along x-axis.

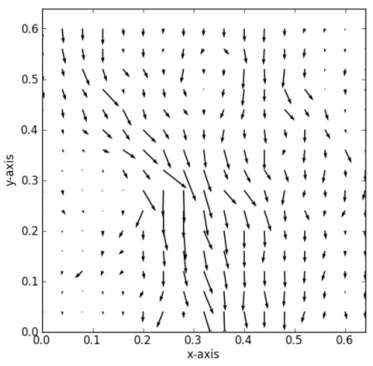
FLOW PARALLEL TO SHIFTING **FLOW PERPENDICULAR TO SHIFTING**

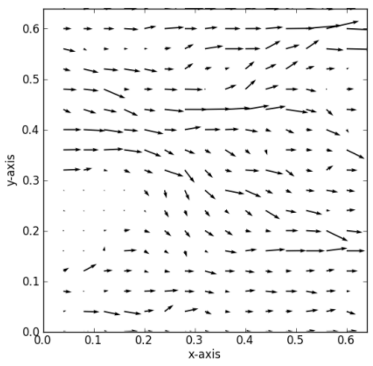
(a)



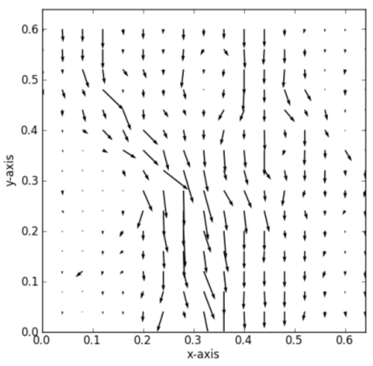


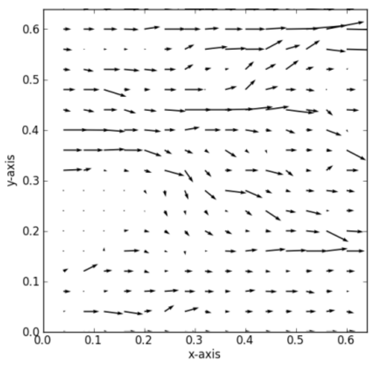
(b)



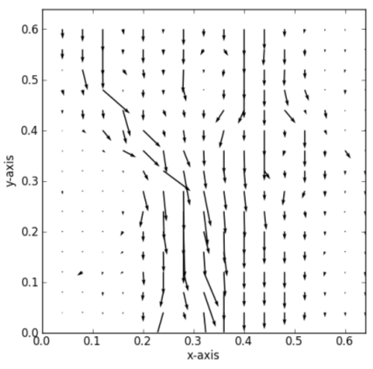


(c)





(d)



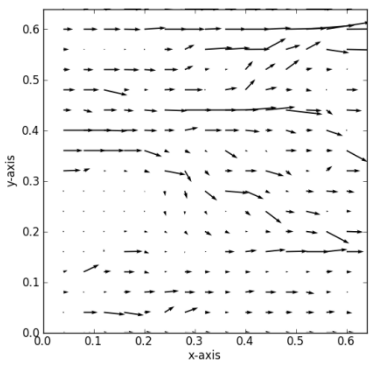


Fig. 5.3.2 - Velocity distribution for increasing (a-d) T values: (a) 0.00 (b) 0.19 (c) 0.38 and (d) 0.53. Shear displacement along y-axis.

Fig 5.3.3 shows that the average velocity in the principal flow direction is significantly higher for flow perpendicular to the shear displacement. The effect becomes more pronounced with increasing non-dimensional yield stress. This behavior occurs for both fractures.

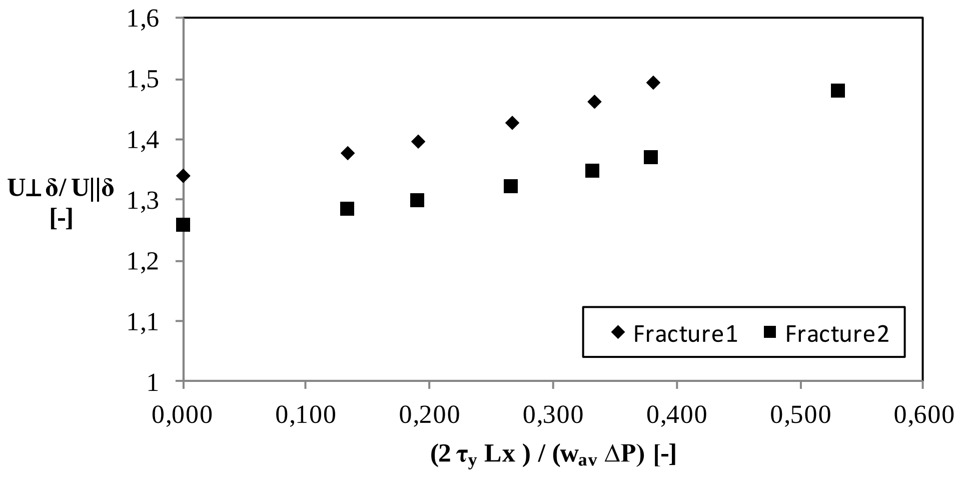


Fig. 5.3.3 - Comparison of average velocity in the coordinate directions perpendicular and parallel to shear displacement. Both fluid velocities are in the direction parallel to applied pressure gradient. Data points for T = 0.50 is excluded for Fracture 1 because different T values of the two scenarios were used.

5.4 Viscosity

Fig. 5.4.1 shows that particle transport is unaffected by plastic viscosity. The average velocity component is greater for unidirectional flow perpendicular to shear displacement compared to flow parallel to shear displacement as shown in Fig. 5.4.2-3. The average velocity component increases for decreasing viscosity.

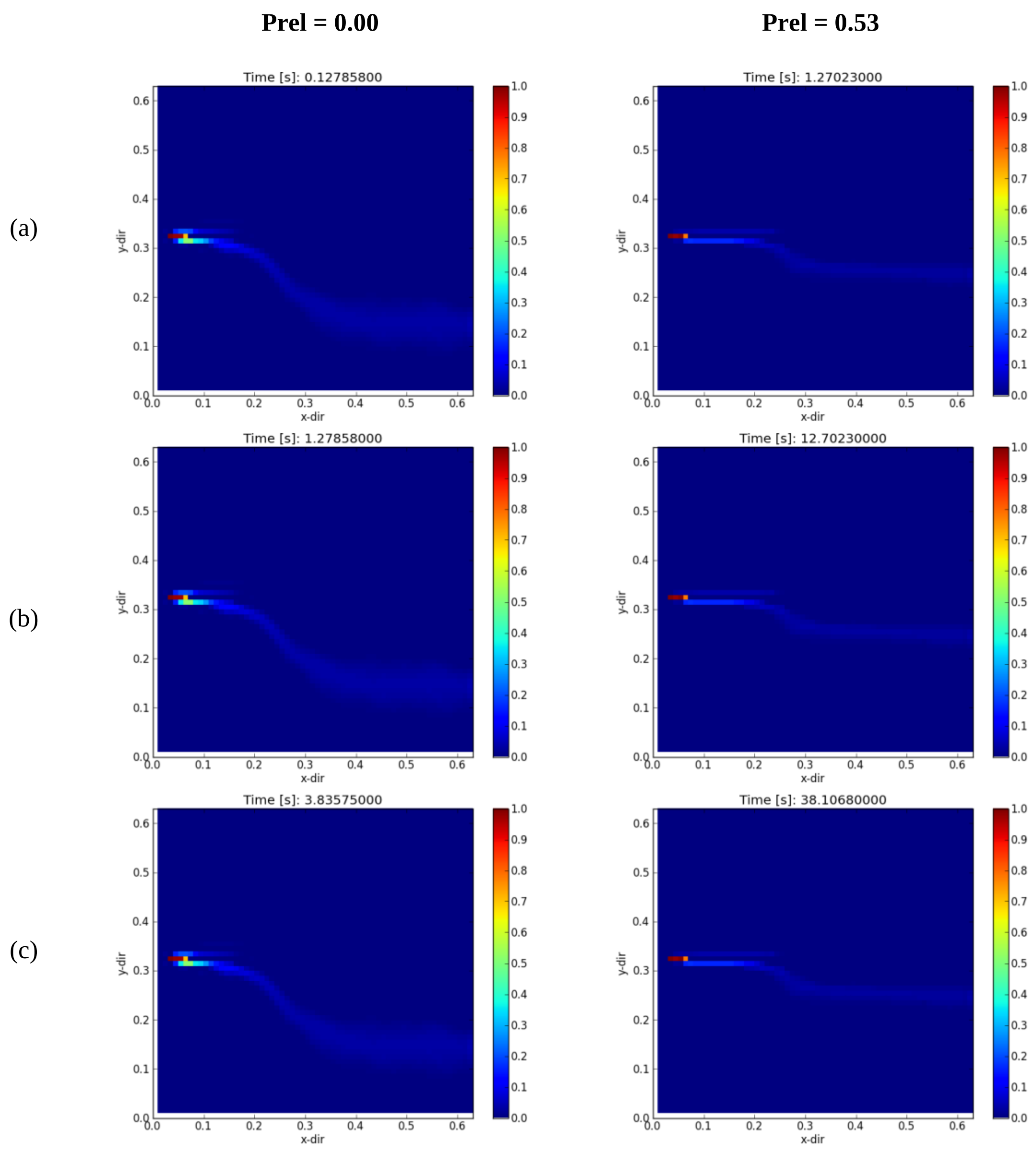


Fig. 5.4.1 - Particle concentration plots for varying plastic viscosities (a) 0.001 Pa s (b) Pa s 0.01 (c) 0.03 Pa s.

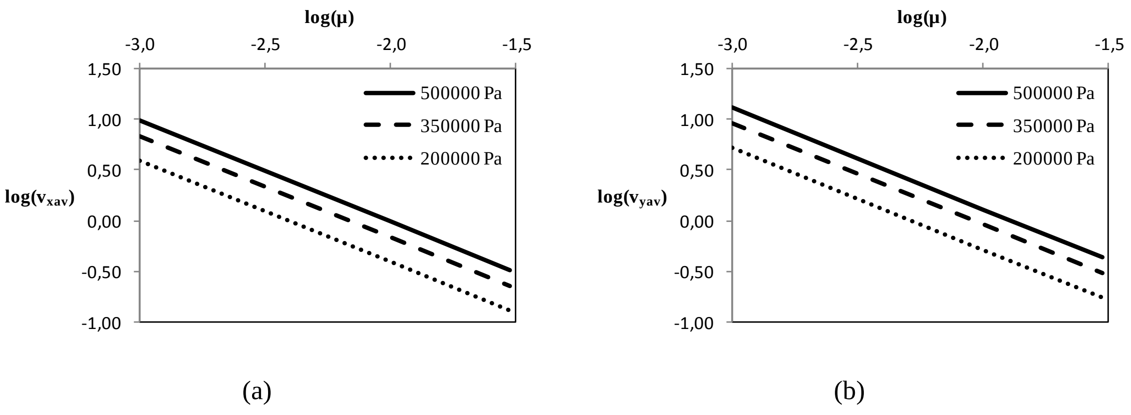


Fig. 5.4.2 - Average velocity component in flow-direction for varying viscosities. Shear displacement applied in x-direction. Flow (a) parallel and (b) perpendicular to shear displacement direction. Unidirectional fluid velocity is higher for (b). The initial fracture surfaces were shifted one node in y-direction.

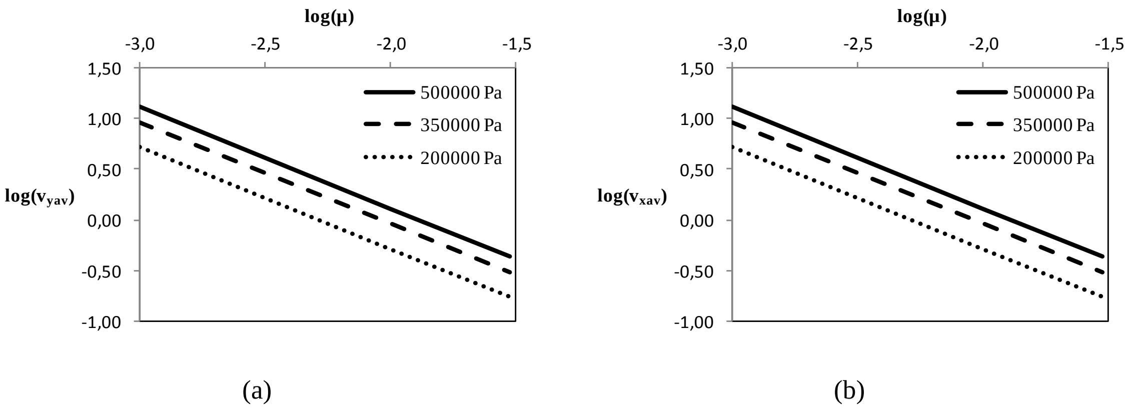


Fig. 5.4.3- Average velocity component in flow-direction for varying viscosities. Shear displacement applied in y-direction. Flow (a) parallel and (b) perpendicular to shear displacement direction. Unidirectional fluid velocity is higher for (b). The initial fracture surfaces were shifted one node in x-direction.

5.5 Shear displacement and fluid properties

The yield stress dominates the fluid when the global flow direction is parallel to the shear displacement (Fig. 5.5.1). The effect becomes more pronounced for increasing non-dimensional yield stress. The Bingham number, B, is given by:

$$B = \frac{\tau_y \, w_{avg}}{2 \, \mu_{pl} \, \bar{v}} \tag{5.1}$$

where $\bar{v}$ is the average velocity in the global flow direction. It physically represents the ratio between the yield stress and viscous forces (Frigaard & Ryan, 2004).

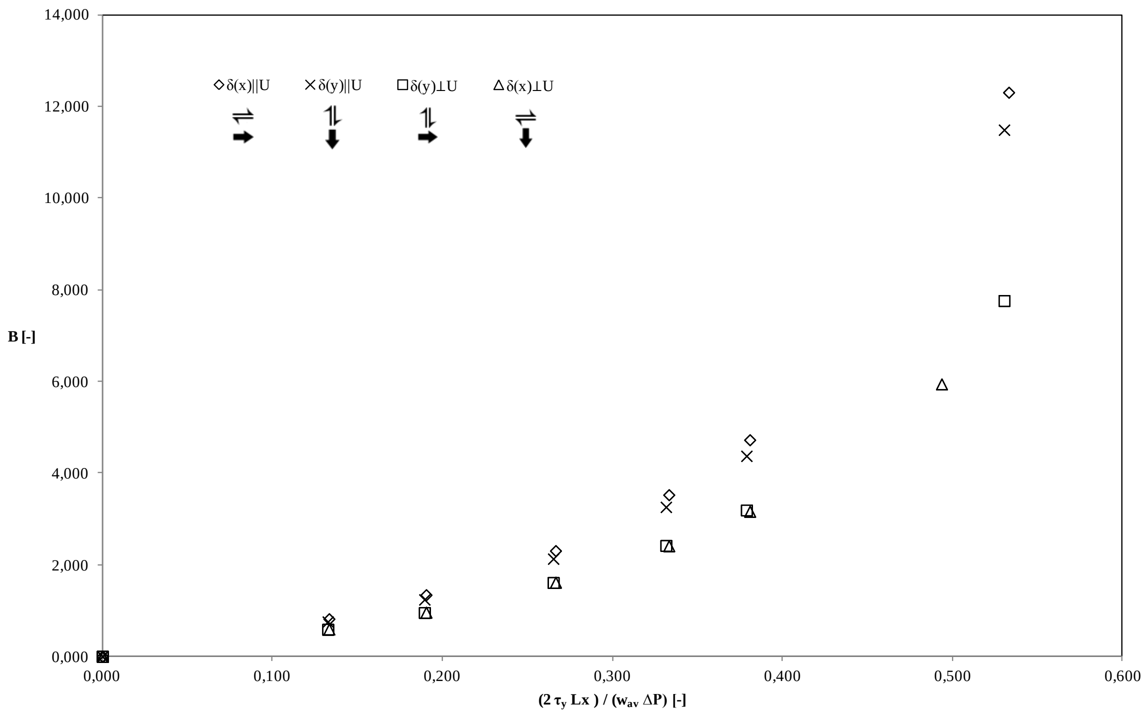


Fig. 5.5.1 - The Bingham number plotted against the non-dimensional yield stress. δ expresses the shifting and the term in the brackets represents the direction of shear displacement.

In the cases where the fluid flow direction is perpendicular to the direction of shear displacement the cases follow a linear trend as seen in Fig 5.5.2. While for cases where the fluid flow direction is parallel to the direction of shear displacement, the behavior is no longer linear.

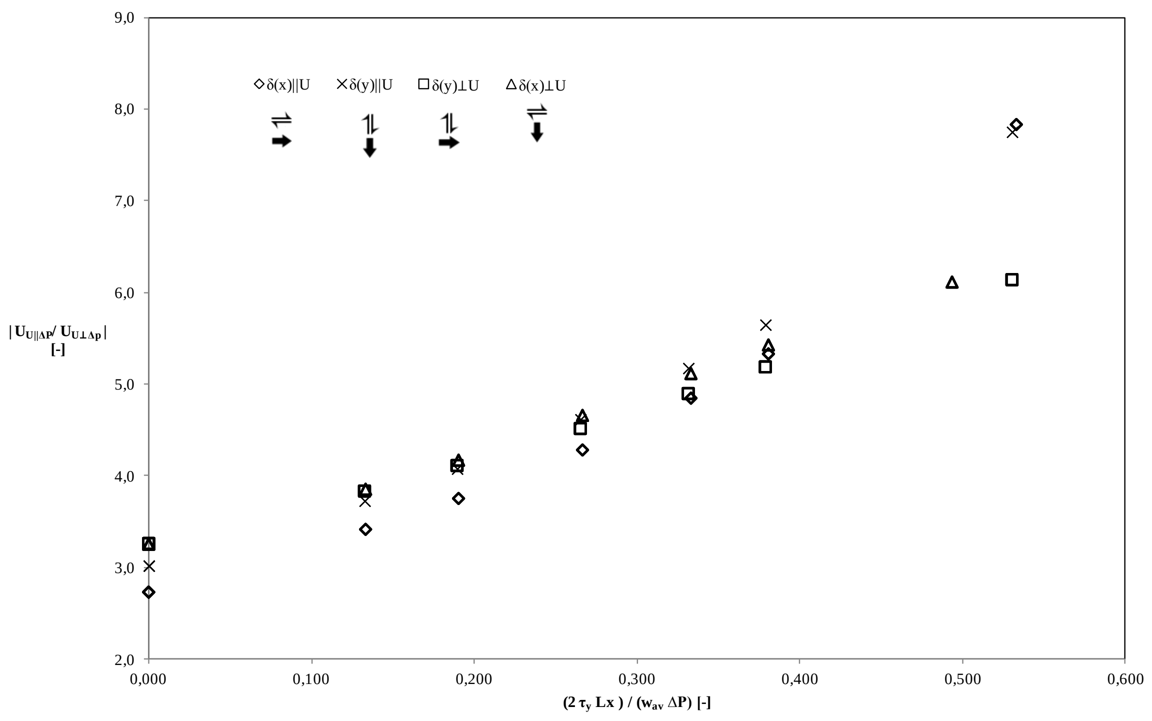
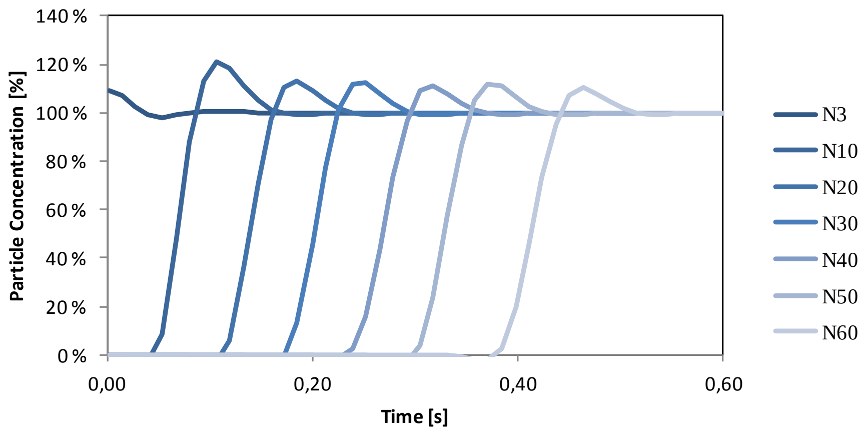


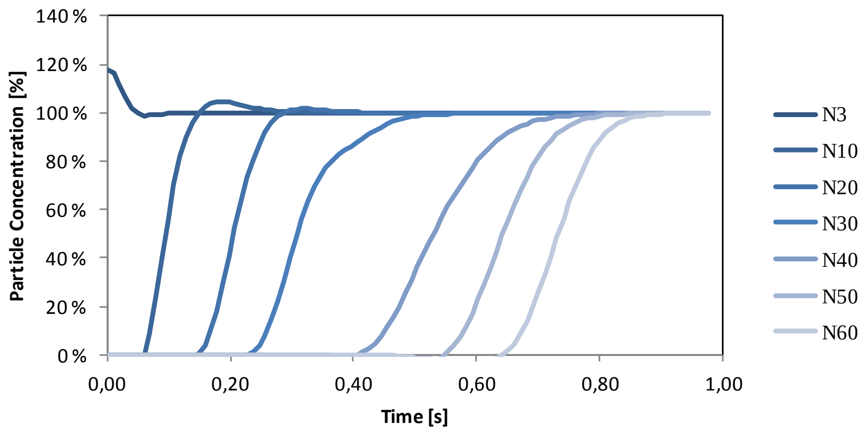
Fig. 5.5.2 - The ratio of velocity parallel and perpendicular to the applied pressure-gradient plotted against non-dimensional yield stress.

5.6 Breakthrough curves

Due to numerical dispersion, breakthrough curves were inaccurate and thus omitted in this study. Breakthrough curves are illustrated in Fig 5.6.1.



(a)



(b)

Fig. 5.6.1 - Breakthrough curves of particle transportation at node lines perpendicular to flow at Nx = 3, 10, 20, 30, 40, 50 and 60. Fluid flow is in parallel to shear displacement in (a) and perpendicular to shear displacement in (b). The two cases are both Newtonian with $\mu = 0.01\ Pa \cdot s$ and $\Delta P = 500\ 000\ Pa$.

6 DISCUSSION

6.1 Analytical

To maintain fluid flow for a Bingham plastic, the applied pressure gradient has to exceed a certain value dependant on the yield stress of the fluid and the local aperture. When the condition is not satisfied, the plug occupies the entire cross-section. The complexity of the fracture makes it difficult to draw conclusions directly from Eqs. 3.17-18. However, some conclusions may be drawn.

The criteria for fluid flow yields:

$$\left|\frac{\delta p}{\delta x}\right| \geq \frac{2\,\tau_y}{w} \tag{6.1}$$

$$\left|\frac{\delta p}{\delta y}\right| \geq \frac{2\,\tau_y}{w} \tag{6.2}$$

When the yield stress value of the fluid approaches zero, fluid flow will occur regardless of aperture, and if obstacles such as bridging material or overlap between the two fracture surfaces are present, the fluid will simply flow around the obstacle. In both cases, Eqs. 3.17-18 would simplify to the well-known Cubic law for Newtonian fluids:

$$q_x = -\frac{w^3}{12\mu}\frac{\partial p}{\partial x} \tag{6.3}$$

$$q_y = -\frac{w^3}{12\mu}\frac{\partial p}{\partial y} \tag{6.4}$$

One way to describe the influence of non-Newtonian properties is to compare the flow rate for Newtonian fluid with Bingham plastic, as follows:

$$\left(\frac{q_{x,BH}}{q_{x,N}}\right) = \frac{-\frac{w^3}{12\mu}\frac{\partial p}{\partial x} - \frac{1}{3\mu}\frac{\tau_y^3}{|\partial p/dx|^3}\frac{\partial p}{\partial x} + \frac{1}{4\mu}\frac{w^2\tau_y}{|\partial p/\partial x|}\frac{\partial p}{\partial x}}{-\frac{w^3}{12\mu}\frac{\partial p}{\partial x}} \tag{6.5}$$

$$\left(\frac{q_{y,BH}}{q_{y,N}}\right) = \frac{-\frac{w^3}{12\mu}\frac{\partial p}{\partial y} - \frac{1}{3\mu}\frac{\tau_y^3}{|\partial p/dy|^3}\frac{\partial p}{\partial y} + \frac{1}{4\mu}\frac{w^2\tau_y}{|\partial p/\partial y|}\frac{\partial p}{\partial y}}{-\frac{w^3}{12\mu}\frac{\partial p}{\partial y}} \tag{6.6}$$

where BH and N denote Bingham fluid and Newtonian fluid flow respectively. The viscosity of the Newtonian fluid is assumed to be equal to the plastic viscosity of the Bingham fluid. Rearranging Eq. 6.5-6 yields:

$$\left(\frac{q_{x,BH}}{q_{x,N}}\right) = 1 + \frac{1}{2}\left(\frac{2\,\tau_y}{w|\partial p/\partial x|}\right)\left[\left(\frac{2\tau_y}{w|\partial p/\partial x|}\right)^2 - 3\right] \tag{6.7}$$

$$\left(\frac{q_{y,BH}}{q_{y,N}}\right) = 1 + \frac{1}{2}\left(\frac{2\,\tau_y}{w|\partial p/\partial y|}\right)\left[\left(\frac{2\tau_y}{w|\partial p/\partial y|}\right)^2 - 3\right] \tag{6.8}$$

The fluid flow conditions in x- and y-direction from Eqs. 6.1-2 can be written, as follows:

$$\frac{2\,\tau_y}{w\,|\delta p/\delta x|} \leq 1 \tag{6.9}$$

$$\frac{2\,\tau_y}{w\,|\delta p/\delta y|} \leq 1 \tag{6.10}$$

Flow rates for Bingham plastics can be expressed by Newtonian flow, as follows:

$$q_{x,BH} = q_{x,N}\left(1 + \frac{1}{2}\left(\frac{2\,\tau_y}{w|\partial p/\partial x|}\right)\left[\left(\frac{2\tau_y}{w|\partial p/\partial x|}\right)^2 - 3\right]\right) \tag{6.11}$$

$$q_{y,BH} = q_{y,N}\left(1 + \frac{1}{2}\left(\frac{2\,\tau_y}{w|\partial p/\partial y|}\right)\left[\left(\frac{2\tau_y}{w|\partial p/\partial y|}\right)^2 - 3\right]\right) \tag{6.12}$$

Define the variables T_x and T_y, as follows:

$$T_x = \frac{2\,\tau_y}{w\,|\delta p/\delta x|} \tag{6.13}$$

$$T_y = \frac{2\,\tau_y}{w\,|\delta p/\delta y|} \tag{6.14}$$

The fluid flow equations for Bingham plastics yield:

$$q_x = \begin{cases} 0 & , \quad P_x > 1 \\ -\frac{w^3}{12\mu}\frac{\partial p}{\partial x}\left(1 + \frac{1}{2}T_x\,(T_x^2 - 3)\right) & , \quad P_x \leq 1 \end{cases} \tag{6.15}$$

$$q_y = \begin{cases} 0 & , \quad P_y > 1 \\ -\frac{w^3}{12\mu}\frac{\partial p}{\partial y}\left(1 + \frac{1}{2}T_y\,(T_y^2 - 3)\right) & , \quad P_y \le 1 \end{cases} \tag{6.16}$$

When the pressure gradient is applied in the x-direction we can see that the flow rate is equal to zero when $T_x > 1$, thus:

$$-1 \le \frac{1}{2}T_x\,(T_x^2 - 3) \le 0 \tag{6.17}$$

In addition, we know that all simulations were run with grid spacing equal to 0.01 m and the fracture with the largest maximum aperture had a maximum aperture close to 0.001 m. The minimum aperture is equal to zero, since all fractures have areas with closed aperture. The ratio between grid spacing and aperture then has to be within the range:

$$10 \le \frac{\Delta x}{w(x,y)} < \infty \tag{6.18}$$

When the aperture approaches zero, the ratio between the grid spacing and aperture will increase such that T_x exceeds its maximum value, and thus plugging will occur. Additionally, the ratio between the yield stress value and pressure difference in the fracture will affect whether or not flow occurs:

$$0 \le \frac{\tau_y}{\Delta p} < \infty \tag{6.19}$$

The entrance pressure of the fracture would depend on the mud pump rates and the environment which the drilling fluid invades. For high pressure fluid flow is likely to occur. Further into the formation the invasion of mud would stop because the overpressure no longer exceeds the yield stress of the drilling fluid leading to plugging. The applied pressure difference in Eq. 6.19 would approach zero and T would exceed its maximum value.

Additionally when comparing pressure difference in the fracture, the applied pressure is likely to vary more extensively in the direction of flow, thus the pressure difference in y-direction is smaller when compared to the pressure difference in x-direction. For greater values of T the flow will behave more channelized in the direction of flow. Flow with low values of T is likely to behave more like Newtonian and less channelized as the conditions for flow are satisfied in both x- and y-direction.

The plastic viscosity is not included in any of the conditions and will not affect the channelization of fluid flow or affect particle transport paths, seen from Eq. 3.20. It is inversely proportional to the flow rate, as follows:

$$q \propto \frac{1}{\mu} \tag{6.20}$$

6.2 Channelization of fluid flow

The fluid flow and particle transport are affected by the fractures' natural anisotropy and by the aperture channels appearing normal to the direction of shifting due to shear displacement. When comparing the cases with applied pressure gradient perpendicular to shear with parallel to shear, the flow velocities along the applied pressure gradient are greater for all cases (Fig. 5.3.3). The velocity distributions (Fig. 5.3.1-2a) for the Newtonian cases also reveal that the fluid flows around the closed apertures leading to greater channelization in areas with high transmissivity. These results comply with the findings in (Koyama, et al., 2006), (Vilarrasa, et al., 2011) and (Yasuhara, et al., 2006).

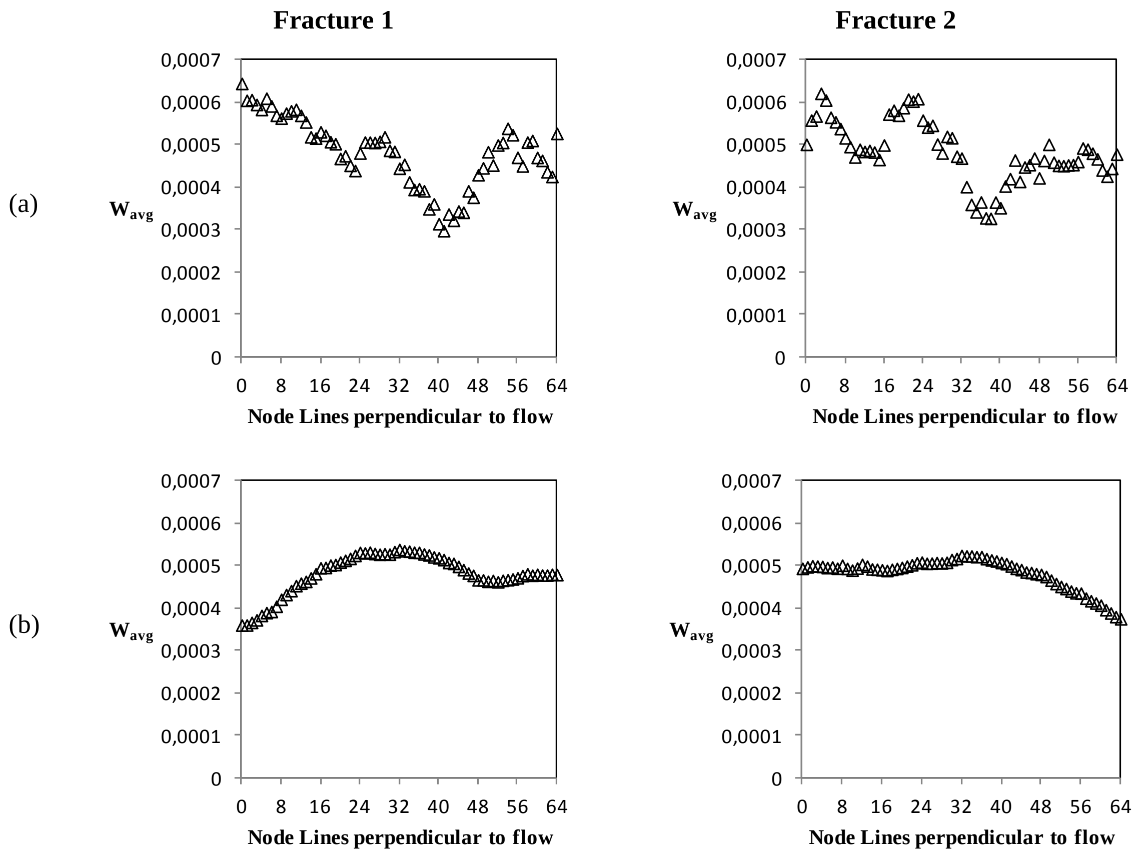
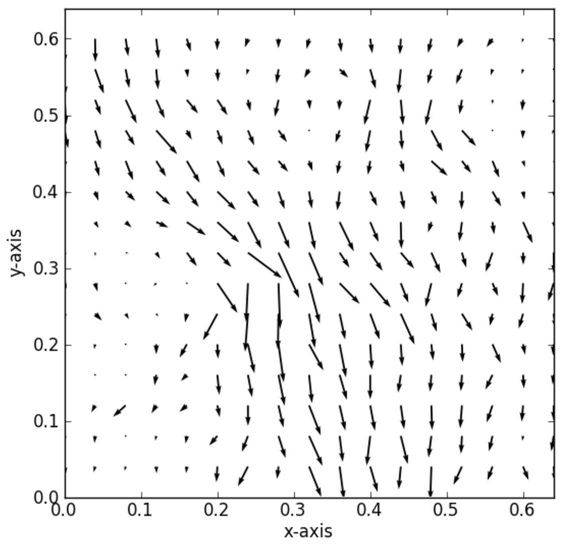


Fig. 6.2.1 - Cross sectional average aperture perpendicular to flow. (a) Flow along shear displacement (b) Flow normal to shear displacement.

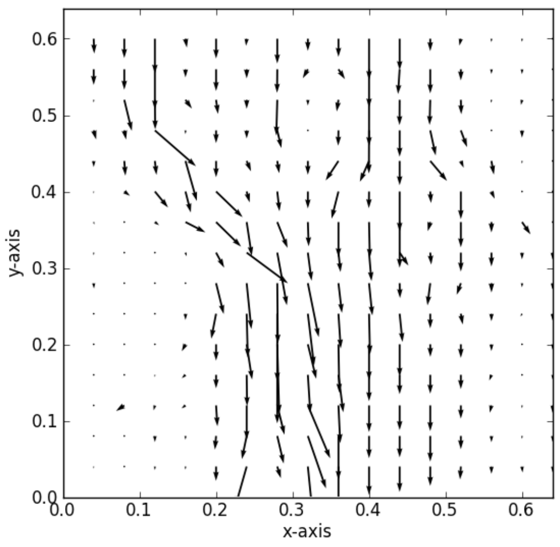
How do the non-Newtonian fluid properties affect channelization of fluid? As discussed in section 6.1 the fluid velocities in both x- and y-direction will differ from the Newtonian cases when T increases. In addition, the applied pressure gradient has to surpass the yield stress of the fluid to avoid plugging, hence greater channelization of fluid flow is expected when the yield stress increases compared to the applied pressure gradient (Eq. 6.1-2).

Fluid flow while the applied pressure gradient is applied perpendicular to shear displacement exhibits a linear ratio between the velocity-ratio and T (Fig 5.5.2). For higher values of T and in the case of applied pressure gradient parallel to shifting, the behavior deviates from the linear relationship. This may be explained by looking at the average cross sectional apertures of the fractures. The terrain is smooth when flowing perpendicular to shear displacement and rough when flowing parallel (Fig 6.2.1). In addition, the minimum cross sectional average aperture for both fractures are smaller parallel to shifting compared to perpendicular. When the same fluid properties and pressure gradient are applied, the cases with flow direction parallel to shifting are more likely to exhibit yield dominant behavior due to lower average aperture. Fig 5.5.1 shows that these cases have higher Bingham numbers meaning that the yield forces dominate the flow, leading to increased channelization.

In general, fluid channelization is more pronounced for higher values of T regardless of fluid flow direction and fracture in this study. Fig 6.2.2 compares Newtonian fluid flow with a viscous Bingham plastic on the same fracture and with the same direction of flow. By comparing (a) and (d) in Fig. 5.3.1-2 the same behavior is found for all four flow scenarios.



(a)



(b)

Fig. 6.2.2 - (a) Newtonian fluid (b) Bingham plastic with T = 0.53. Channelization is more pronounced for the Bingham fluid, e.g. at x = 0.28.

6.3 Pressure distributions

The anisotropy of the fracture affects the Newtonian fluid pressure distributions as seen in Fig 5.2.1a and Fig 5.2.2a. Pressure is maintained in regions with low aperture and in areas with closed aperture, while the opposite behavior occur in regions with higher aperture. But how do the non-Newtonian fluid properties and the direction of shear affect the pressure distributions? The relative change in local pressure becomes more pronounced for increasing T, both for fluid flow parallel and perpendicular to the shear displacement (Fig. 5.2.1-2). The change is more pronounced for flow parallel to shifting. This may be explained by looking at the velocity distributions in Fig. 5.3.1-2. Fluid flows around areas with closed aperture, leading to an increase of pressure upstream of the obstacle. As discussed in section 6.2, the fluid flow becomes more channelized for increasing T.

6.4 Evaluation of fractures

The aperture frequency plots of the fractures used in this study (Fig. 6.4.1) is similar to the fractures used in (Koyama, et al., 2006). The fractures have a slowly varying aperture which satisfies the lubrication theory approximation according to Eq. 2.13-14.

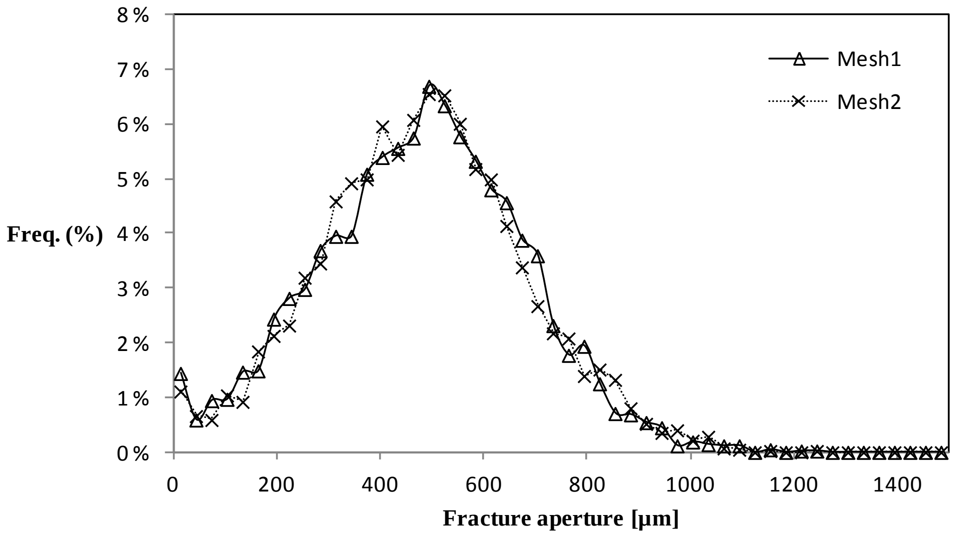


Fig. 6.4.1 - The Frequency plots of the two fractures with shear displacement of 10 mm.

The Hurst exponent was set equal to 0.75 for all simulations before shear displacement, but as shown in (Koyama, et al., 2006) the Hurst exponent increases when shear displacement is applied. From Fig. 2.1.2, one can see that the Hurst exponent reaches its stationary level at large shear displacements, and thus one can expect the fractures to have a Hurst exponent within the range 0.75 and 1.00. In addition, brittle fractures due to shear displacement generally have a Hurst exponent close to 0.8 (Hansen, et al., 2000), such that one could expect the fractures used in this study to have a Hurst exponent closer to 0.8.

Fig. 6.4.2 illustrates a phenomenon that occurs when creating computer-generated fractures with the algorithm described in Section 3.5. Cross sectional areas of the fracture perpendicular to shearing have regions with great contrast between high and low values of local aperture. The main cause of this phenomenon is that aperture in overlapping regions between the two surfaces, are set equal to zero.

There are still some features that are not considered in this study. The fractures assume a constant morphology throughout the simulation meaning that the local aperture does not vary due to effects of fluid flow and solute transport. The phenomena of removal of bridging materials and erosion of fracture walls are not considered at all.

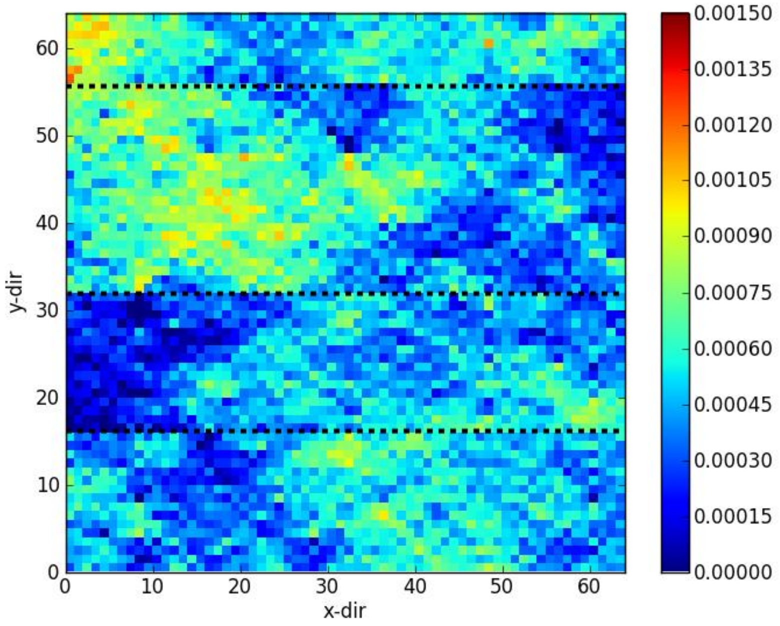


Fig. 6.4.2 - Aperture distribution of fracture 2. Color bar on the right shows the aperture values in metres. Shear displacement in the y-direction.

In reality, temperature changes of the mud and change in fracture morphology will influence the fluid flow (Yasuhara, et al., 2006). Upon start of fluid flow into the fracture, the removal of minerals working as bridging asperities can lead to a changes in average aperture (Yasuhara, et al., 2006). This phenomenon would highly impact the solute transport and fluid behavior of the fracture. The pressure difference within the fracture would have to surpass a much greater yield stress to aperture ratio. E.g. removal of bridging material leading to half the average aperture of the initial state would require two times the pressure gradient for the Bingham fluid to flow.

In addition when fluid flow continuously enters the fracture for a long duration of time, it is likely that the flow would start eroding the fracture walls leading to an increase of average aperture (Yasuhara, et al., 2006) from the reduced state, leading to less channelized flow. One can expect different flow regimes when drilling in naturally fractured formations affecting the fluid flow and particle transport.

6.5 Particle transport

The particle transport is mainly affected by the anisotropy of the aperture distribution. When comparing flow parallel and normal to shear displacement, the fluid's ability to carry particles is greater for the latter case as seen in Fig 6.5.1. Continuous channels in the fracture created by shear displacement or from initial anisotropy thus increase particle transport.

Similar to velocity distributions, the particle transport becomes more channelized for increasing T as seen in Fig. 5.1.1-2.

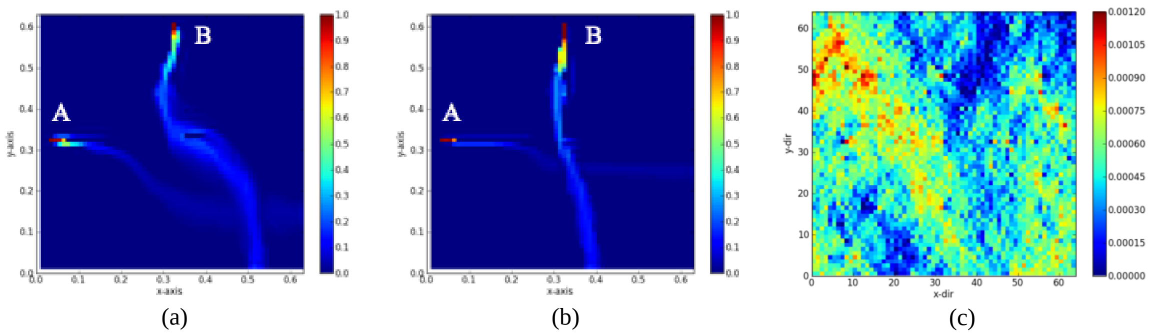


Fig. 6.5.1 - Particle concentration plot for (a) Newtonian fluid with $T = 0$ and (b) Bingham Fluid with T = 0.50. The aperture distribution can be seen in (c). Shear displacement applied in x-direction. A denotes flow in x-direction and B denotes flow in y-direction. The particle transport is enhanced in case B and the flow becomes more channelized with increasing T.

6.6 Challenges

Settling of particles from the drilling fluid is neglected in this study as it is run in de-coupled mode, meaning that velocities and pressure distributions are calculated before transport of particles. The particles settling from the fluid or removed from the formation would affect local velocities, pressure distribution and particle transport. In addition, the fracture aperture field is constant throughout all simulations, not considering the removal of bridging material or degradation of formation rock which would also highly affect the simulations.

The model also considers the fracture walls to be perfectly impermeable, meaning no leak-off to the surrounding formation. In reality the filter cake can be deposited and is dependent on a number of variables including the properties of the mud, particle size range and formation permeability. The influence of leak-off is important to understand when dealing with ultimate volume losses of fluids, but the essentials of non-Newtonian flow in rough-walled fractures can be understood by studying the steady state solution without leak-off.

Numerical dispersion is also a difficulty when simulations are run on fractures with areas of closed local apertures. The breakthrough curves from the simulations in this study exhibited too much dispersion and were therefore included and not interpreted in this study. It is possible to adjust the plots by zeroing out the negative values of particle concentration and adjusting concentrations above 100%. However, the results would no longer be accurate.

Frigaard & Ryan (2004) pointed out the existence of a pseudo plug region for non-Newtonian fluids flowing in a slowly varying aperture. The pseudo plug is a region between the true plug and the shear flow, and its existence would be a game changer of how we think of yield-stress fluid flow in fractures. The model implemented in *PROPANICA* does not account for the existence of the pseudo plug.

7 FUTURE WORK

The author encourages future studies to include fractures with varying shear displacement in the analysis to study how fluid flow and solute transport are affected by increasing shear displacement. Additionally, it is suggested to develop a numerical model that includes different regimes for time-dependant change of aperture due to removal of bridging material and degradation of the fracture walls. The author also encourages future studies to develop a better model to reduce the numerical dispersion for cases with greater yield stress.

8 CONCLUSION

Bingham fluid properties have been studied and tested on two fractures shifted in either coordinate direction. Fluid flow was performed both perpendicular and parallel to the direction of shear displacement. The following conclusions were drawn from the results:

- Shear displacement creates flow channels perpendicular to shifting direction. The greater the shear displacement, the more pronounced the effect.

- The ratio of yield stresses to viscous forces has higher values for flow parallel to shear displacement compared to perpendicular. The fracture terrain causes this behavior, because the aperture behaves less rough perpendicular to shear displacement.

- The plastic viscosity affects the velocities. It does not affect the particle transport or channelization of fluid flow. In general, unidirectional fluid flow was greater in the direction perpendicular to shearing. The effect becomes more pronounced for increasing value of non-dimensional yield stress.

- Channelization of fluid flow and particle transport is enhanced for increasing value of non-dimensional yield stress. This occurs for flow both parallel and perpendicular to the shear displacement.

- The fluid's ability to carry particles increases when flow is applied perpendicular to shear displacement. The effect is more pronounced for fluids with high value of non-dimensional yield stress.

9 ACKNOWLEDGEMENTS

The author wishes to thank Prof. Rune Martin Holt (NTNU) for providing the opportunity to write this master's thesis at SINTEF Petroleum Research. In addition, the useful discussions and input from Prof. Alex Hansen (NTNU) are highly appreciated.

Further the author wishes to thank Alexandre Lavrov (SINTEF Petroleum Research) for supervising during this project and providing background material, in particular the *PROPANICA* code. The encouragement to learn new programming languages was an important factor for carrying out this unique study.

10 REFERENCES

Adachi, J., Siebrits, E., Peirce, A. & Desroches, J., 2007. Computer simulation of hydraulic fractures. *International Journal of Rock Mechanics & Mining Sciences,* Volume 44, pp. 739-757.

Amadei, B. & Illangasekare, T., 1994. A mathematical Model for Flow and Solute Transport in Non-homogeneous Rock Fractures. *Int. J. Rock Mech. Min. Sci. & Geomech. Abstr.,* Volume 31, pp. 719-731.

Auradou, H. et al., 2010. Miscible transfer of solute in different model fractures: From random to multiscale wall roughness. *C.R. Geosciences 342,* Volume 342, pp. 644-652.

Chen, M., Rossen, W. & Yortsos, Y. C., 2005. The flow and displacement in porous media of fluids with yield stress. *Chemical Engineering Science,* Volume 60, pp. 4183-4202.

Daneshy, A., 1978. Numerical solution of sand transport in hydraulic fracturing. *Journal of Petroleum Technology,* Volume 30, pp. 132-140.

Fournier, A., Fussell, D. & Carpenter, L., 1982. Computer rendering of stochastic models. *Communications of the ACM,* Volume 25, pp. 371-384.

Frigaard, I. & Ryan, D., 2004. Flow of a visco-plastic fluid in a channel of slowly varying width. *J. Non-Newtonian Fluid Mech,* Volume 123, pp. 67-83.

Hansen, A., Schmittbuhl, J., Batrouni, G. G. & de Oliveira, F. A., 2000. Normal Stress Distribution of Rough Surfaces in Contact. *Geophysical Research Letters,* Volume 29, pp. 3639-3642.

Hanssen, A. R., 2012. *Numerical Modeling of Drilling Fluid Flow in Natural Fractures,* Trondheim: s.n.

Koyama, T., Fardin, N., Jing, L. & Stephansson, O., 2006. Numerical simulation of shear-induced flow anisotropy and scale-dependent aperture and transmissivity evoluation of rock fracture replicas. *International Journal of Rock Mechanics & Mining Sciences,* Volume 43, pp. 89-106.

Lavrov, A., 2011. *Models for proppant transport and deposition in hydraulic fracture simulation: A review of the state of the art,* Trondheim: SINTEF Petroleum Research Report.

Lipscomb, G. & Denn, M., 1984. Flow of bingham fluids in complex geometries. *Journal of Non-Newtonian Fluid Mechanics,* Volume 14, pp. 337-346.

Majidi, R. et al., 2010. Quantitative Analysis of Mud Losses in Naturally Fractures Reservoirs: The Effect of Rheology. *SPE Drilling & Completion (114130),* pp. 509-517.

Mandelbrot, B. & Van Ness, J., 1968. Fractional Brownian Motions, Fractional Noise and Applications. *SIAM Review No.4,* Volume 10, pp. 422-437.

Novotny, E., 1977. Proppant transport. *SPE Paper 6813.*

Patankar, S. V., 1980. *Numerical Heat Transfer and Fluid Flow.* New York: McGraw-Hill Book Company.

Staniforth, A. & Côté, J., 1991. Semi-Lagrangian Integration Schemes for Atmospheric Models - A review. *Monthly Weather Review,* Volume 119, pp. 2206-2223.

Talon, L., Auradou, H. & Hansen, A., 2010. Permeability of self-affine aperture fields. *Physical Review,* Volume 82, pp. 1-7.

Vilarrasa, V., Koyama, T., Neretnieks, I. & Jing, L., 2011. Shear-Induced Flow Channels in a Single Rock Fracture and Their Effect on Solute Transport. *Transp Porous Med,* Volume 87, pp. 503-523.

Yasuhara, H. et al., 2006. Evolution of fracture permeability through fluid-rock reaction under hydrothermal conditions. *Earth and Planetary Science Letters,* Volume 244, pp. 186-200.

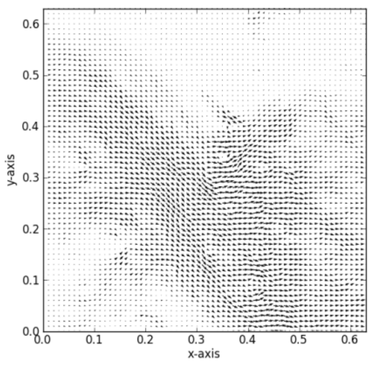
Zimmerman, R., Kumar, S. & Bodvarsson, G., 1991. Lubrication Theory Analysis of the Permeability of Rough-walled Fractures. *Int. J. Rock MEch. Min. Sci.& Geomech. Abstr, Number 4,* Volume 28, pp. 325-331.

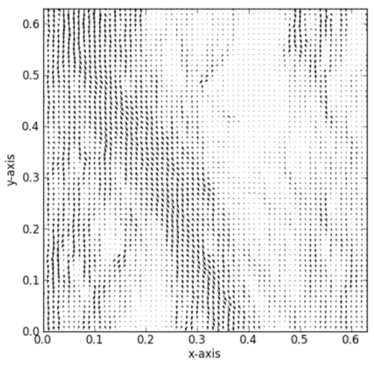
APPENDIX

A.1 ORIGINAL PLOTS

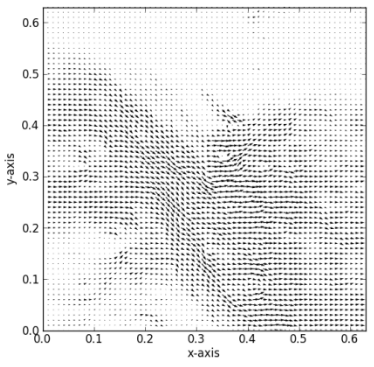
FLOW PARALLEL TO SHIFTING **FLOW PERPENDICULAR TO SHIFTING**

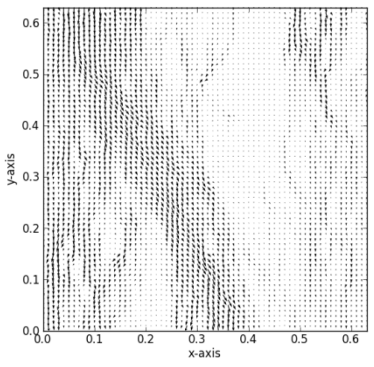
(a)



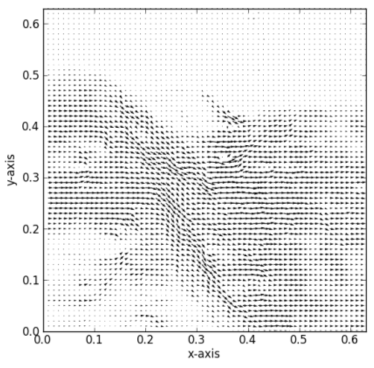


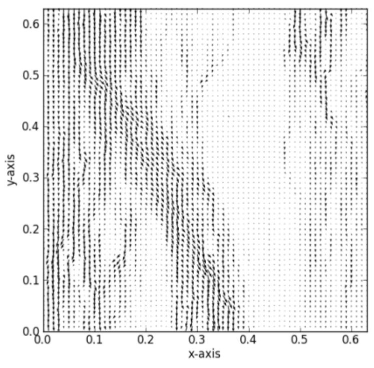
(b)



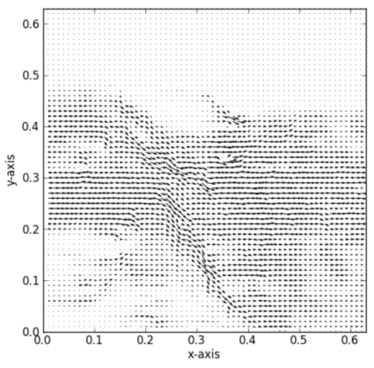


(c)





(d)



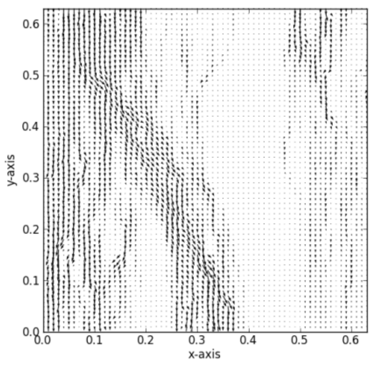


Fig. A.1 - Velocity distribution for increasing T values: (a) 0.00 (b) 0.19 (c) 0.38 and (d) 0.53/0.50. Shear displacement along x-axis.

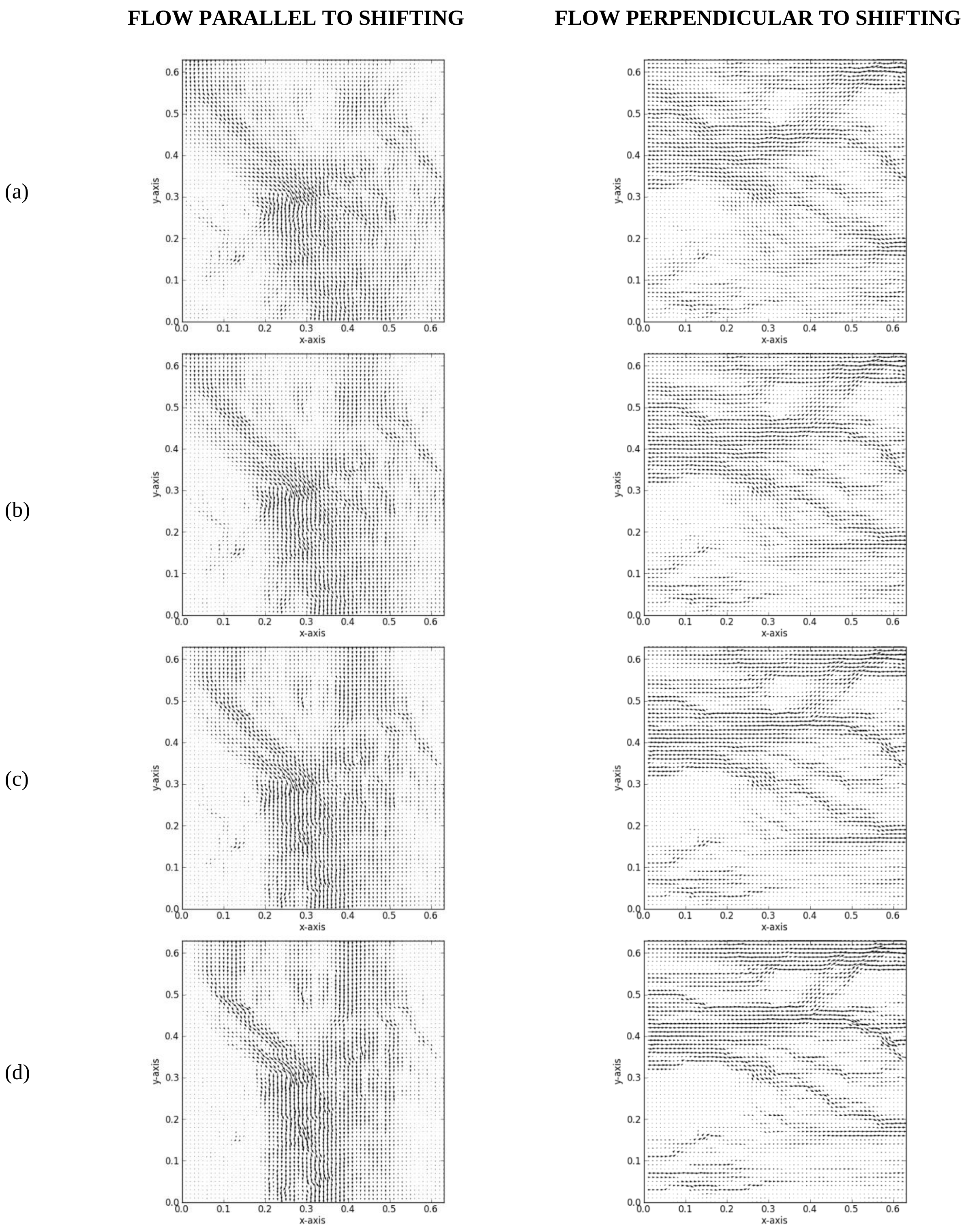


Fig. A.2 - Velocity distribution for increasing T values: (a) 0.00 (b) 0.19 (c) 0.38 and (d) 0.53. Shear displacement along y-axis.

FLOW PARALLEL TO SHIFTING **FLOW PERPENDICULAR TO SHIFTING**

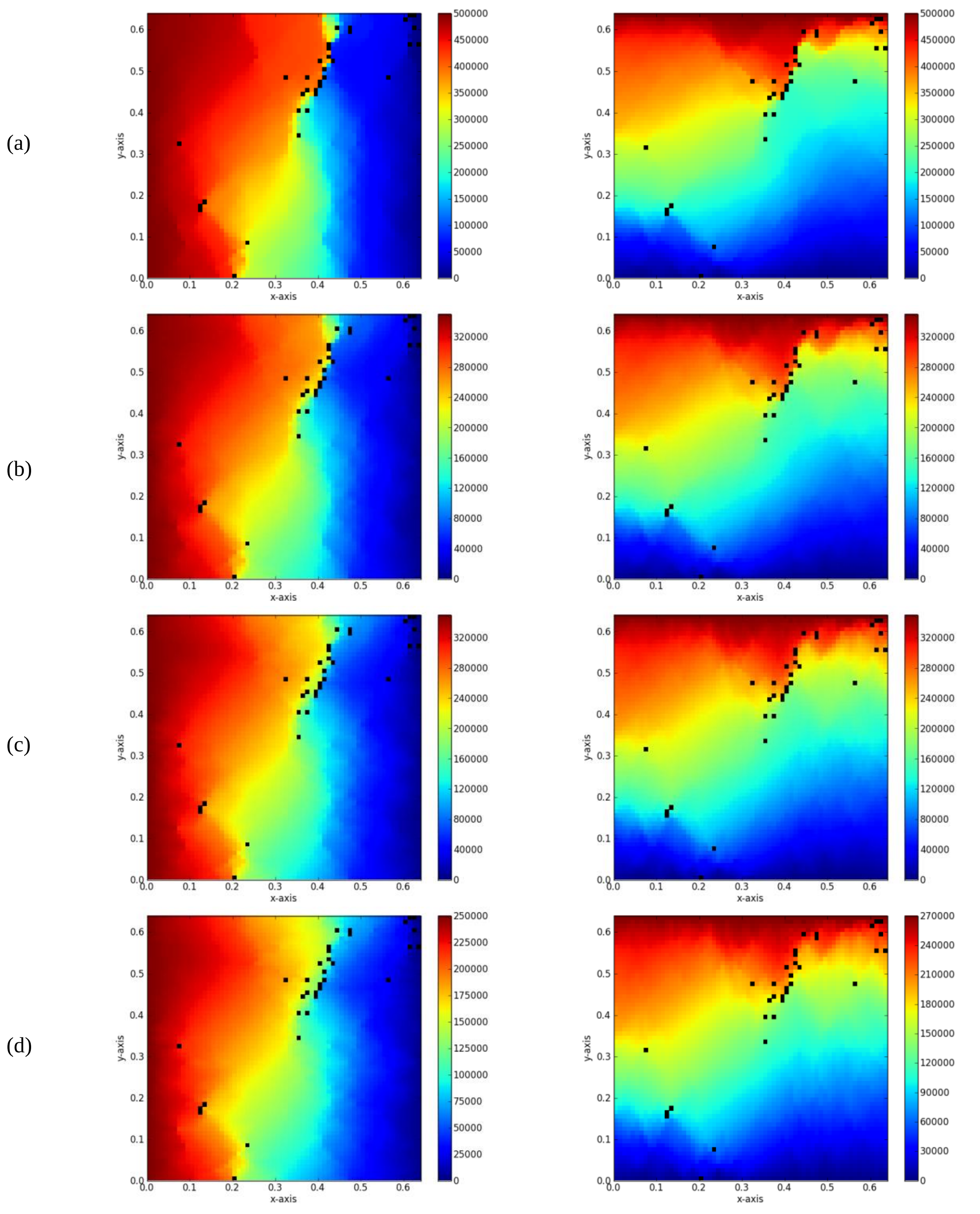
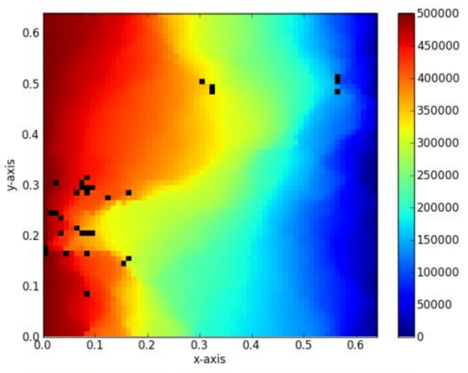
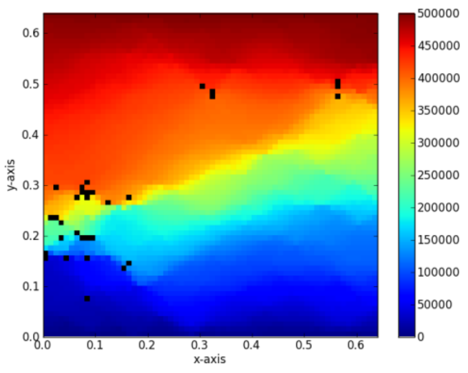


Fig. A.3 - Pressure distribution for increasing T values: (a) 0.00 (b) 0.19 (c) 0.38 and (d) 0.53/0.50. Shear displacement along x-axis. Concentration values can be read from the color bar on the right-side. Black dots represent closed local aperture.

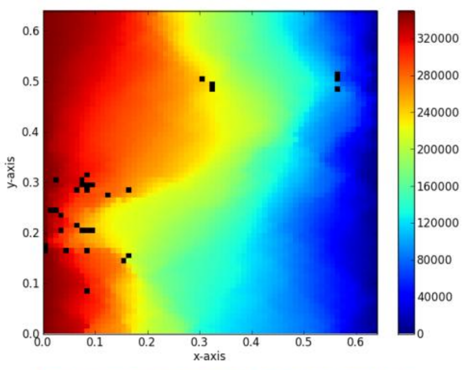
FLOW PARALLEL TO SHIFTING **FLOW PERPENDICULAR TO SHIFTING**

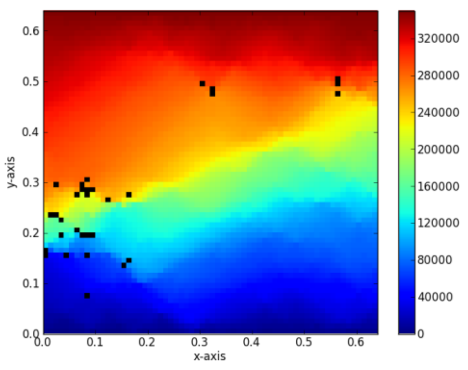
(a)



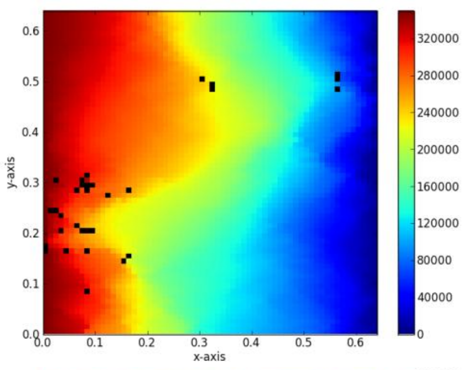


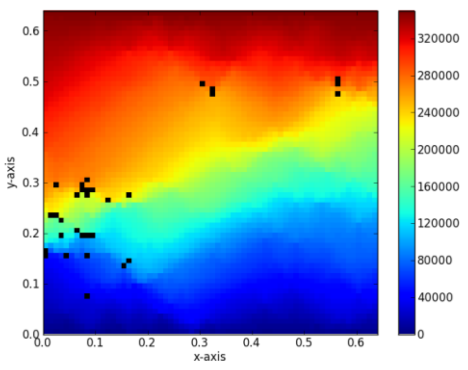
(b)



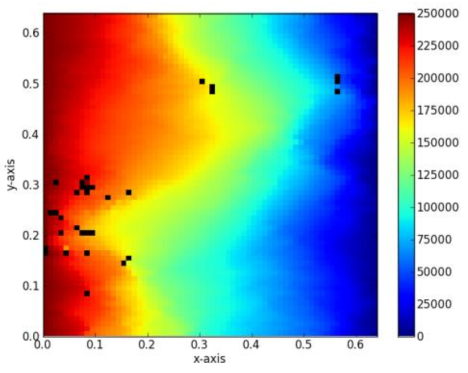


(c)





(d)



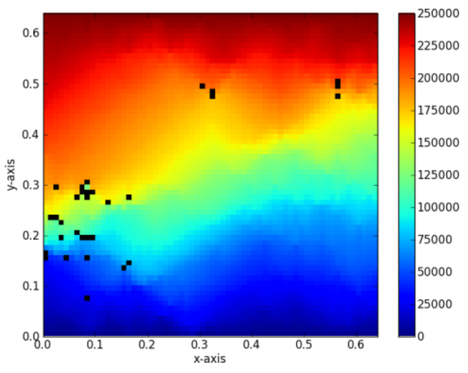
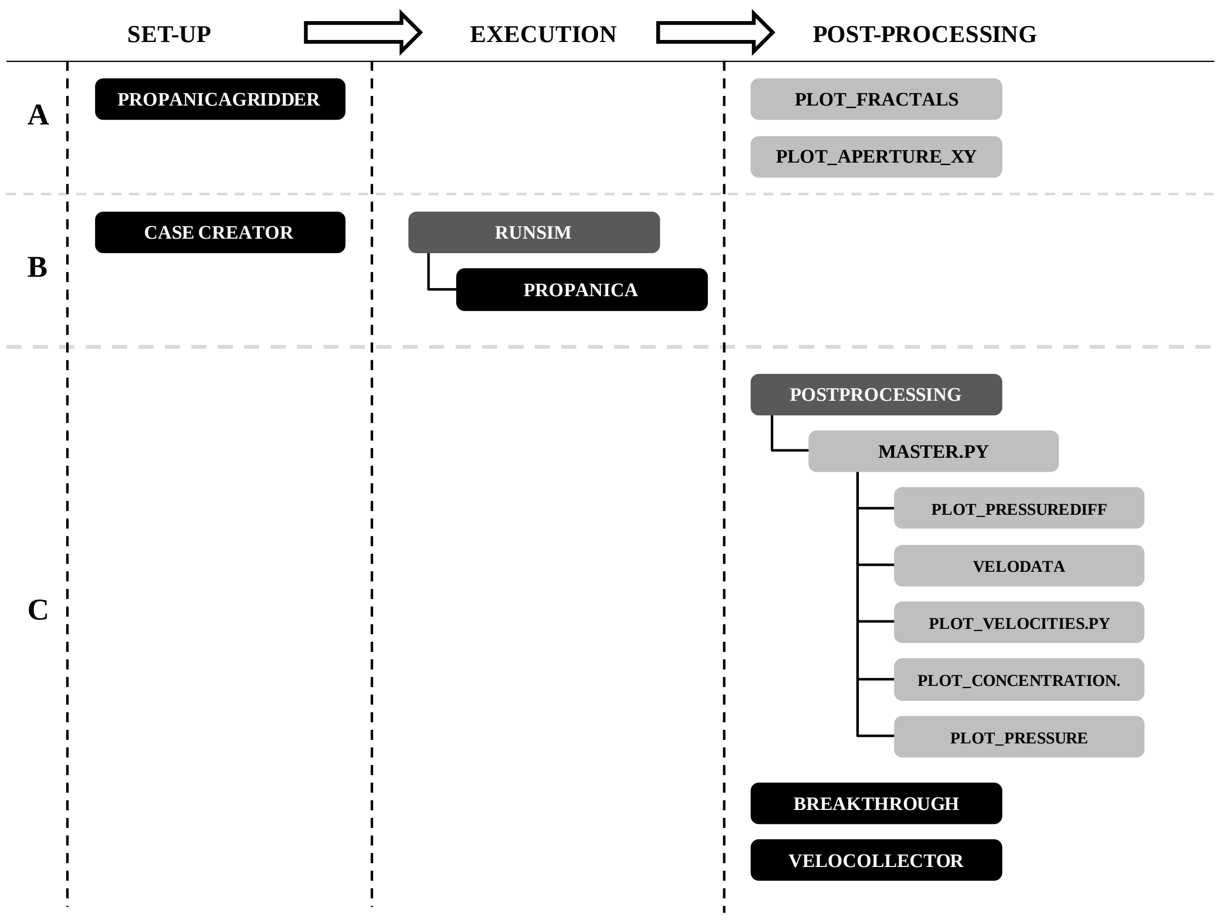


Fig. A.4 - Pressure distribution for increasing T values: (a) 0.00 (b) 0.18 (c) 0.37 and (d) 0.53. Shear displacement along y-axis. Concentration values can be read from the color bar on the right-side. Black dots represent closed local aperture.

A.2 FLOW CHART AND WORKING DIAGRAM



A. PROPANICAGRIDDER is a C++ code that generates rough-walled fractures based on the recursive subdivision technique. The modified version, edited by the author, includes the option to apply shear displacement in x- and/or y-direction. The code also creates a mesh file where the fracture terrain is rotated 90 degrees counter clockwise, so that the user can simulate flow in both x- and y-direction. PLOT_FRACTALS visualizes the fracture in 3D-space while PLOT_APERTURE_XY visualizes the aperture distribution in 2D in the XY-plane.

B. The user locates different fractures in a separate directory specified in CASECREATOR. In addition, the user has to specify fluid properties and other simulation inputs. The script creates individual folders including all data necessary for simulations. The simulations can either be run on a personal computer or on a cluster. The latter method is preferred when comprehensive test matrixes are simulated. If run on cluster, several simulations can be run at the same time, thus RUNSIM.SH, a UNIX script starts simulations to avoid manual execution.

C. When the results from the simulations are ready, the raw-material is post-processed by python visualization scripts. If a comprehensive test-matrix has been simulated, it is preferable to use the Windows batch script POSTPROCESSING.BAT to queue post-processing on your personal computer. The script starts post-processing in several case folders by copying the master python file into every case directory and executes the script. MASTER.PY calls on various post-processing scripts created for visualization of concentration distributions, pressure distributions, velocity fields in the fracture. BREAKTHROUGH and VELOCOLLECTOR are C++ scripts developed to gather velocity information and breakthrough curves into user-friendly SDV-formats that can easily be converted to a spread-sheet format.

This procedure was developed by the author to make it possible to execute 108 cases during a short period of time. All scripts are modified or created by the author and are included in the Appendix (A3-5) for future references.

A.3 C++ CODES

A.3.1 Modified PROPANICAGRIDDER

```
#include <cmath>
#include <iostream>
#include <string>
#include <stdexcept>
#include <exception>
#include <vector>
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <ctime>
#include <windows.h>

using namespace std;

double gauss()
{
	int i;
	double rnorm = 0.0;

	for (i=1; i <= 12; i++ ) rnorm = rnorm + (float) rand()/RAND_MAX;

	return( rnorm - 6.0 );
}

void fractal(std::vector<std::vector<double>>& z, int scale, int n, double
corner1, double corner2, double corner3, double corner4, double fc, double fH,
double fs)
{
	int i, j, level;

	z[0][0]=corner1;
	z[0][n]=corner2;
	z[n][0]=corner3;
	z[n][n]=corner4;

	/* Build side boundaries */
	for (level=1; level<=scale; level++) {
		for (i=n/(int)pow(2.,level); i<=n-n/(int)pow(2.,level);) {
			z[i][0] = 0.5 * (z[i - n/(int)pow(2.,level)][0] + z[i +
n/(int)pow(2.,level)][0]) + gauss() * fs * pow(fc, -fH * (double)level);
			z[i][n] = 0.5 * (z[i - n/(int)pow(2.,level)][n] + z[i +
n/(int)pow(2.,level)][n]) + gauss() * fs * pow(fc, -fH * (double)level);
			i = i + n/((int)pow(2.,level-1));
		}
		for (j=n/(int)pow(2.,level); j<=n-n/(int)pow(2.,level);) {
			z[0][j] = 0.5 * (z[0][j - n/(int)pow(2.,level)] + z[0][j +
n/(int)pow(2.,level)]) + gauss() * fs * pow(fc, -fH * (double)level);
			z[n][j] = 0.5 * (z[n][j - n/(int)pow(2.,level)] + z[n][j +
n/(int)pow(2.,level)]) + gauss() * fs * pow(fc, -fH * (double)level);
			j = j + n/((int)pow(2.,level-1));
		}
	}

	/* Build inner points */
	for (level=1; level<=scale; level++) {
		/* Center points of the level's squares are computed using diagonal
elements */
		for (i=n/(int)pow(2.,level); i<=n-n/(int)pow(2.,level);) {
```

```
                        for (j=n/(int)pow(2.,level); j<=n-n/(int)pow(2.,level);) {
                                z[i][j] = 0.25 *
(z[i+n/(int)pow(2.,level)][j+n/(int)pow(2.,level)] +
z[i+n/(int)pow(2.,level)][j-n/(int)pow(2.,level)] + z[i-
n/(int)pow(2.,level)][j+n/(int)pow(2.,level)] + z[i-n/(int)pow(2.,level)][j-
n/(int)pow(2.,level)]) + gauss() * fs * pow(fc, -fH * (double)level);
                                j = j + n/((int)pow(2.,level-1));
                        }
                        i = i + n/((int)pow(2.,level-1));
                }

                /* Boundary center points of the level's squares are computed using
diagonal elements */
                for (i=n/(int)pow(2.,level); i<=n-n/(int)pow(2.,level);) {
                        for (j=n/(int)pow(2.,level-1); j<=n-n/(int)pow(2.,level-1);)
{
                                z[i][j] = 0.25 * (z[i][j+n/(int)pow(2.,level)] +
z[i][j-n/(int)pow(2.,level)] + z[i-n/(int)pow(2.,level)][j] +
z[i+n/(int)pow(2.,level)][j]) + gauss() * fs * pow(fc, -fH * (double)level);
                                j = j + n/((int)pow(2.,level-1));
                        }
                        i = i + n/((int)pow(2.,level-1));
                }
                for (i=n/(int)pow(2.,level-1); i<=n-n/(int)pow(2.,level-1);) {
                        for (j=n/(int)pow(2.,level); j<=n-n/(int)pow(2.,level);) {
                                z[i][j] = 0.25 * (z[i][j+n/(int)pow(2.,level)] +
z[i][j-n/(int)pow(2.,level)] + z[i-n/(int)pow(2.,level)][j] +
z[i+n/(int)pow(2.,level)][j]) + gauss() * fs * pow(fc, -fH * (double)level);
                                j = j + n/((int)pow(2.,level-1));
                        }
                        i = i + n/((int)pow(2.,level-1));
                }

        } /* Close for level */


}

int main()
{

        std::ofstream surfaces, inputs, rotsurf;
        surfaces.open ("mesh.dat");
        inputs.open ("input.txt");
        rotsurf.open ("mesh_90ccw.dat");
        int max_l;
        double H;
        double scaling;
        double c;
        double dx;
        double dy;
        double mid_distance;
        std::string random_level_zero;
        int randomizer;
        char userin;
        int Sx;
        int Sy;



        // Userinput values
        std::cout << "Enter the number of levels (an integer, e.g. 5): ";
        std::cin >> max_l;
```

```
	std::cout << "Enter the Hurst exponent (a double between 0.0 and 1.0, 
e.g. 0.7): ";
	std::cin >> H;
	std::cout << "Enter the scaling factor (a double, e.g. 0.001): ";
	std::cin >> scaling;
	std::cout << "Enter c (a double, e.g. 2.0): ";
	std::cin >> c;
	std::cout << "Enter the randomizer (an integer, e.g. 5000. Using the same 
randomizer value in different runs of this program will produce identical 
meshes.): ";
	std::cin >> randomizer;
	std::cout << "Enter the separation between the two surfaces (a double, 
e.g. 0.002): ";
	std::cin >> mid_distance;
	std::cout << "Enter grid spacings." << std::endl;
	std::cout << "Enter the grid spacing in the x-direction (a double, e.g. 
0.01): ";
	std::cin >> dx;
	std::cout << "Enter the grid spacing in the y-direction (a double, e.g. 
0.01): ";
	std::cin >> dy;
	std::cout << "Do you wish to include shifting? y/n: ";
	std::cin >> userin;


	//Write input values to inputs.txt

	inputs << "==============================" << std::endl;
	inputs << '\t' << "INPUT" << std::endl;
	inputs << "==============================" << std::endl;
	inputs << std::endl;
	inputs << "Number of levels" << '\t' << static_cast<int>(max_l) << 
std::endl;
	inputs << "Hurst exponent" << '\t' << '\t' << static_cast<double>(H) << 
std::endl;
	inputs << "Scaling factor" << '\t' << '\t' << 
static_cast<double>(scaling) << std::endl;
	inputs << "C factor" << '\t' << '\t' << static_cast<double>(c) << 
std::endl;
	inputs << "Randomizer" << '\t' << '\t' << static_cast<int>(randomizer) << 
std::endl;
	inputs << "Separation" << '\t' << '\t' << 
static_cast<double>(mid_distance) << std::endl;
	inputs << "Grid Spacing x-dir" << '\t' << static_cast<double>(dx) << 
std::endl;
	inputs << "Grid Spacing y-dir" << '\t' << static_cast<double>(dx) << 
std::endl;


	// Determine bool values
	bool usershift = false;
	if ( userin == 'y' || userin == 'Y') {
		usershift = true;
		std::cout << "Enter shifting in x-direction (an integer, e.g. 10): 
";
		std::cin >> Sx;
		std::cout << "Enter shifting in y-direction (an integer, e.g. 10): 
";
		std::cin >> Sy;
		inputs << "Shifting" << '\t' << '\t' << "YES" << std::endl;
		inputs << "Shifting x-dir" << '\t' << '\t' << int(Sx) << std::endl;
		inputs << "Shifting y-dir" << '\t' << '\t' << int(Sy) << std::endl;
		inputs << std::endl;
		}
	else if ( userin == 'n' || userin == 'N' ) {
			usershift = false;
			inputs << std::endl;
			inputs << "Shifting" << '\t' << '\t' << "NO" << std::endl;
			inputs << std::endl;
		}
		else {
			usershift = false;
			inputs << std::endl;
			inputs << "Shifting" << '\t' << '\t' << "NO" << std::endl;
		}

		if (usershift) {

			max_l += 1;

			int N = static_cast<int>(pow(2.0, max_l) );
			for (int i = 0; i <= randomizer; ++i) rand();

			std::vector<std::vector<double>> surf1, aperture;
			for (int i = 0; i <= N; ++i) {
				std::vector<double> row;
				for (int j = 0; j <= N; ++j) {
					row.push_back(0.0);
				}
				surf1.push_back(row);
				aperture.push_back(row);
			}

			// surf1

			fractal (surf1, max_l, N, 0.0, 0.0, 0.0, 0.0, c, H, scaling);

			int M = static_cast<int>(pow(2.0, max_l-1) );

			double average_aperture = 0.0;
			double min_aperture = mid_distance + surf1.at(0).at(0) -
surf1.at(Sx).at(Sy);
			double max_aperture = mid_distance + surf1.at(0).at(0) -
surf1.at(Sx).at(Sy);
			double rms = 0.0;
			int countzero = 0;
			for (int i = 0; i <= M; ++i) {
				for (int j = 0; j <= M; ++j) {
					// double apert = mid_distance + surf1.at(i).at(j) -
surf2.at(i).at(j);
					double apert = mid_distance + surf1.at(i).at(j) -
surf1.at(i+Sx).at(j+Sy);
					int zero_aperture_index = 0;

					if ( apert <= 0.0 ) {
						apert = 0.0;
						zero_aperture_index = 1;
						countzero += 1;
					}
					average_aperture += apert;
					if ( apert > max_aperture ) max_aperture = apert;
					if ( apert < min_aperture ) min_aperture = apert;
					surfaces << i << " " << j << " " <<
static_cast<double>(i) * dx << " " << static_cast<double>(j) * dy << " " <<
apert << " " << zero_aperture_index << std::endl;
					rotsurf << M-j << " " << i << " " <<
static_cast<double>(M-j) * dx << " " << static_cast<double>(i) * dy << " " <<
apert << " " << zero_aperture_index << std::endl;
				}
			}
```

```
            average_aperture /= static_cast<double>( (M + 1) * (M + 1) );
            for (int i = 0; i <= M; ++i) {
                  for (int j = 0; j <= M; ++j) {
                        double apert = mid_distance + surf1.at(i).at(j) -
surf1.at(i+Sx).at(j+Sy);
                        rms += ( apert - average_aperture ) * ( apert -
average_aperture );
                  }
            }
            rms /= static_cast<double>( (M + 1) * (M + 1) - 1 );
            rms = sqrt(rms);



            std::cout << "Mesh file mesh.dat has been written." << std::endl;

            std::cout  << std::endl << "Average aperture = " <<
average_aperture << std::endl;
            std::cout  << "Min aperture = " << min_aperture << std::endl;
            std::cout  << "Max aperture = " << max_aperture << std::endl;
            std::cout  << "Nodes of closed aperture = " << countzero <<
std::endl;
            std::cout  << std::endl;
            std::cout  << "rms = " << rms << std::endl;
            std::cout  << "rms / dx = " << rms / dx << std::endl;
            std::cout  << "rms / dx should normally be between 0.02 and 0.2 for
the liubrication theory approximation to hold." << std::endl;
            std::cout  << "rms / dy = " << rms / dy << std::endl;
            std::cout  << "rms / dy should normally be between 0.02 and 0.2 for
the liubrication theory approximation to hold." << std::endl;

            inputs << "==============================" << std::endl;
            inputs << '\t' << "OUTPUT" << std::endl;
            inputs << "==============================" << std::endl;
            inputs << std::endl;
            inputs << "Avg aperture = " << '\t' << average_aperture <<
std::endl;
            inputs << "Min aperture = " << '\t' << min_aperture << std::endl;
            inputs << "Max aperture = " << '\t' << max_aperture  << std::endl;
            inputs << "RMS = " << '\t' << rms << std::endl;
            inputs << "RMS / dx = " << '\t' << rms / dx << std::endl;
            inputs << "RMS / dy = " << '\t' << rms / dy << std::endl;
            inputs << std::endl;

                  if ( ( rms / dx >= 0.02 ) && ( rms / dx <= 0.2 )) {
                        inputs << "Lubrication theory valid!" << std::endl;
                  }
                  else {
                        inputs << "Lubrication theory not valid. Consider
altering input values" << std::endl;
                  }
            }
      else {

                        int N = static_cast<int>(pow(2.0, max_l) );
            for (int i = 0; i <= randomizer; ++i) rand();

            std::vector<std::vector<double>> surf1, surf2, aperture;
            for (int i = 0; i <= N; ++i) {
                  std::vector<double> row;
                  for (int j = 0; j <= N; ++j) {
                        row.push_back(0.0);
                  }
```

```
                surf1.push_back(row);
                surf2.push_back(row);
                aperture.push_back(row);
        }

        // surf1

        fractal (surf1, max_l, N, 0.0, 0.0, 0.0, 0.0, c, H, scaling);
        fractal (surf2, max_l, N, 0.0, 0.0, 0.0, 0.0, c, H, scaling);


        double average_aperture = 0.0;
        double min_aperture = mid_distance + surf1.at(0).at(0) -
surf2.at(0).at(0);
        double max_aperture = mid_distance + surf1.at(0).at(0) -
surf2.at(0).at(0);
        double rms = 0.0;
        for (int i = 0; i <= N; ++i) {
                for (int j = 0; j <= N; ++j) {
                        // double apert = mid_distance + surf1.at(i).at(j) -
surf2.at(i).at(j);
                        double apert = mid_distance + surf1.at(i).at(j) -
surf2.at(i).at(j);
                        int zero_aperture_index = 0;
                        if ( apert <= 0.0 ) {
                                apert = 0.0;
                                zero_aperture_index = 1;
                        }
                        average_aperture += apert;
                        if ( apert > max_aperture ) max_aperture = apert;
                        if ( apert < min_aperture ) min_aperture = apert;
                        surfaces << i << " " << j << " " <<
static_cast<double>(i) * dx << " " << static_cast<double>(j) * dy << " " <<
apert << " " << zero_aperture_index << std::endl;
                }
        }
        average_aperture /= static_cast<double>( (N + 1) * (N + 1) );
        for (int i = 0; i <= N; ++i) {
                for (int j = 0; j <= N; ++j) {
                        double apert = mid_distance + surf1.at(i).at(j) -
surf2.at(i).at(j);
                        rms += ( apert - average_aperture ) * ( apert -
average_aperture );
                }
        }
        rms /= static_cast<double>( (N + 1) * (N + 1) - 1 );
        rms = sqrt(rms);


        std::cout << "Mesh file mesh.dat has been written." << std::endl;

        std::cout  << std::endl << "Average aperture = " <<
average_aperture << std::endl;
        std::cout  << "Min aperture = " << min_aperture << std::endl;
        std::cout  << "Max aperture = " << max_aperture << std::endl;
        std::cout  << "rms = " << rms << std::endl;
        std::cout  << "rms / dx = " << rms / dx << std::endl;
        std::cout  << "rms / dx should normally be between 0.02 and 0.2 for
the liubrication theory approximation to hold." << std::endl;
        std::cout  << "rms / dy = " << rms / dy << std::endl;
        std::cout  << "rms / dy should normally be between 0.02 and 0.2 for
the liubrication theory approximation to hold." << std::endl;

        inputs << "==============================" << std::endl;
        inputs << '\t' << "OUTPUT" << std::endl;
```

```
            inputs << "=============================" << std::endl;
            inputs << std::endl;
            inputs << "Avg aperture = " << '\t' << average_aperture <<
std::endl;
            inputs << "Min aperture = " << '\t' << min_aperture << std::endl;
            inputs << "Max aperture = " << '\t' << max_aperture  << std::endl;
            inputs << "RMS = " << '\t' << rms << std::endl;
            inputs << "RMS / dx = " << '\t' << rms / dx << std::endl;
            inputs << "RMS / dy = " << '\t' << rms / dy << std::endl;
            inputs << std::endl;

                  if ( ( rms / dx >= 0.02 ) && ( rms / dx <= 0.2 )) {
                        inputs << "Lubrication theory valid!" << std::endl;
                  }
                  else {
                        inputs << "Lubrication theory not valid. Consider
altering input values" << std::endl;
                  }
      }


      inputs.close();
      surfaces.close();
      rotsurf.close();


      return 0;

}
```

A.3.2 CASECREATOR

```
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <direct.h>
#include <exception>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
#include <windows.h>
#include <list>

using namespace std;

void copymesh(int casenum, int meshnum)
{

      //CREATE FOLDER STRINGS
      std::ostringstream oss;
      std::ostringstream dross;
      oss << fixed << showpoint << setprecision(3) << "Mesh" << meshnum;
//    dross << fixed << showpoint << setprecision(3) << "Case_" << casenum;
      dross << fixed << showpoint << "case_" << std::setw(2) <<
std::setfill('0') << casenum;


      //COPY FROM
      string input_dir =
"c://Users/Alexander/Dropbox/School/MasterThesis/Mesh/";
      string input_folder = oss.str();

      //COPY TO
      string output_folder = dross.str();
      string output_dir =
"c://Users/Alexander/Dropbox/School/MasterThesis/Simulations/";

      //FILE NAME
      string file_name = "/mesh.dat";

      //COPY MESH.DAT TO CASE FOLDERS
      CopyFile( (input_dir+input_folder+file_name).c_str() ,
(output_dir+output_folder+file_name).c_str() ,false);
}

void shellcreator(int index)
{
      ostringstream koss;
//    koss << fixed << showpoint << setprecision(3) << "case_" << index;
      koss << fixed << showpoint << "case_" << std::setw(2) <<
std::setfill('0') << index;
      string output_dir =
"c://Users/Alexander/Dropbox/School/MasterThesis/Simulations/";
      string index_string = koss.str();
      string shell_name = "/PROPANICA.sh";

      std::ofstream shellf ( (output_dir+index_string+shell_name).c_str() );

      shellf << "#!/bin/bash" << std::endl;
```

```
	shellf << "#  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 
- - - - - - -" << std::endl;
	shellf << "#PBS -N " << fixed << showpoint << "case_" << std::setw(2) << 
std::setfill('0') << index << std::endl;
	shellf << "#PBS -l nodes=1:ppn=12" << std::endl;
	shellf << "#PBS -l walltime=03:00:00:00" << std::endl;
	shellf << "#PBS -A acc-ipt" << std::endl;
	shellf << "#PBS -q bigmem" << std::endl;
	shellf << "#" << std::endl;
	shellf << "#PBS -m abe" << std::endl;
	shellf << "#PBS -M alexahan@stud.ntnu.no"  << std::endl;
	shellf << "#" << std::endl;
	shellf << "#" << std::endl;
	shellf << "# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 
- - - - - - -" << std::endl;
	shellf << std::endl;
	shellf << "cd $PBS_O_WORKDIR" << std::endl;
	shellf << std::endl;
	shellf << "module load gcc/4.7.0" << std::endl;
	shellf << "/home/alexahan/PROPANICA/case_" << std::setw(2) << 
std::setfill('0') << index << "/PROPANICA.exe > /home/alexahan/PROPANICA/case_" 
<< std::setw(2) << std::setfill('0') << index << "/stdoutputfile" << std::endl;
	shellf << std::endl;
	shellf << "exit 0" << std::endl;
}


void casecreator(double yield, double visco, double pressure, double dx, double 
dy, int index)
{
	// SPECIFY FOLDER NAME
	ostringstream oss;
//	oss << fixed << showpoint << setprecision(3) << "case_" << index;
	oss << fixed << showpoint << "case_" << std::setw(2) << std::setfill('0') 
<< index;
	string output_dir = 
"c://Users/Alexander/Dropbox/School/MasterThesis/Simulations/";
	string index_string = oss.str();
	string file_name = "/case.dat";

	// CREATE FOLDER(S)
	_mkdir( (output_dir+index_string).c_str() );

	std::ofstream casef ( (output_dir+index_string+file_name).c_str() );

	// OPENMP
	casef << "begin  OpenMP" << std::endl;
	casef << "" << std::endl;
	casef << " nthreads = 12" << std::endl;
	casef << "" << std::endl;
	casef << "end" << std::endl;
	casef << "" << std::endl;
	casef << "" << std::endl;

	// FLUID
	casef << "begin fluid" << std::endl;
	casef << "" << std::endl;
	casef << "   rheology = Bingham" << std::endl;
	casef << "   yield_stress = " << yield << std::endl;
	casef << "   plastic_viscosity = " << visco << std::endl;

	casef << "   density = 1000.0" << std::endl;
	casef << "" << std::endl;
	casef << "end" << std::endl;
	casef << "" << std::endl;
```

```
        casef << "" << std::endl;

        // FLUID BOUNDARY CONDITIONS
        casef << "begin fluid_bc" << std::endl;
        casef << "" << std::endl;
        casef << "   source_pressure = " << fixed << setprecision(1) << pressure
<< std::endl;
        casef << "   sink_pressure =  0.0" << std::endl;
        casef << "" << std::endl;
        casef << "end" << std::endl;
        casef << "" << std::endl;

        // MESH
        casef << "begin mesh" << std::endl;
        casef << "" << std::endl;
        casef << "   type = Cartesian" << std::endl;
        casef << "   dx = " << setprecision(3) << dx << std::endl;
        casef << "   dy = " << setprecision(3) << dy << std::endl;
        casef << "" << std::endl;
        casef << "end" << std::endl;
        casef << "" << std::endl;
        casef << "" << std::endl;


        // FLOW
        casef << "begin flow" << std::endl;
        casef << "" << std::endl;
        casef << "   import_velocities = false" << std::endl;
        casef << "   tol_bisection = 1.0e-8" << std::endl;
        casef << "   exit_tolerance_pressure = 1.0e-7" << std::endl;
        casef << "   exit_tolerance_qx = 1.0e-3" << std::endl;
        casef << "   type_of_aperture_average = arithmetic" << std::endl;
        casef << "   velocity_interpolation = linear" << std::endl;
        casef << "   type_of_pressure_initialization = linear" << std::endl;
        casef << "" << std::endl;
        casef << "end" << std::endl;
        casef << "" << std::endl;

        // HISTORY
        casef << "begin history" << std::endl;
        casef << "" << std::endl;
        casef << "   name = hist1" << std::endl;
        casef << "   interpolation = hold" << std::endl;
        casef << "   0.0   1.0" << std::endl;
                     // Maybe add option (?)
        casef << "" << std::endl;
        casef << "end" << std::endl;
        casef << "" << std::endl;
        casef << "" << std::endl;

        // PROPPANT
        casef << "begin proppant" << std::endl;
        casef << "" << std::endl;
        casef << "   assign hist1 node 3 32" << std::endl;

        casef << "" << std::endl;
        casef << "   Coupling = DeCoUpLeD" << std::endl;
        casef << "   target_time_automatic = true" << std::endl;
        casef << "   target_time_multiplier = 2.0" << std::endl;
        casef << "   Courant = 0.5" << std::endl;
        casef << "" << std::endl;
        casef << "   backtrack = iterative" << std::endl;
        casef << "   number_of_iteraions = 1" << std::endl;
        casef << "" << std::endl;
```

```
      casef << "   concentration_interpolation = cubic_lagrange_sequential_xy"
<< std::endl;
      casef << "" << std::endl;
      casef << "   kill_negative_concentrations = false" << std::endl;
      casef << "" << std::endl;
      casef << "   number_of_concentration_dumps = 30" << std::endl;
      casef << "   number_of_monitor_dumps = 100" << std::endl;
      casef << "" << std::endl;

      casef << "   monitor_write_period = 0.010" << std::endl;
      casef << "   breakthrough_line_i = 3" << std::endl;
      casef << "   breakthrough_line_i = 10" << std::endl;
      casef << "   breakthrough_line_i = 20" << std::endl;
      casef << "   breakthrough_line_i = 30" << std::endl;
      casef << "   breakthrough_line_i = 40" << std::endl;;
      casef << "   breakthrough_line_i = 50" << std::endl;
      casef << "   breakthrough_line_i = 60" << std::endl;
      casef << ""<< std::endl;
      casef << "end" << std::endl;

      //CLOSE FILE
      casef.close();
}



int main()
{

      std::vector<double> yield, visco, pressure, hurst;
      std::vector<int> mesh;

      // Yield point values:

      yield.push_back(0.0);
      yield.push_back(25.0);
      yield.push_back(50.0);


//    double s;
//    std::cout << "Enter Yield Point value: " << std::endl;
//    std::cin >> s;
//    yield.push_back(s);

      // Plastic viscosity values:

      visco.push_back(0.001);
      visco.push_back(0.01);
      visco.push_back(0.03);

      // LHS Pressure values:

      pressure.push_back(500000.0);
      pressure.push_back(350000.0);
      pressure.push_back(200000.0);

      // Hurst exponent

      hurst.push_back(0.75);

      // Mesh files

      mesh.push_back(4);

      // Constants
```

```
	double dx=0.01;
	double dy=0.01;

	// DECLARATION OF VALUES

	int numberofcases = mesh.size() * pressure.size() * yield.size() *
visco.size();
	int index = 82;

	cout << std::endl;
	cout << "A total number of " << numberofcases << " cases will be created"
<< std::endl;
	cout << "Number of variables:" << std::endl;
	cout << std::endl;
	cout << "Mesh files:" << '\t' << mesh.size() << std::endl;
	cout << "Yield point:" << '\t' << yield.size() << std::endl;
	cout << "Plastic visco:" << '\t' << visco.size() << std::endl;
	cout << "Pressures:" << '\t' << pressure.size() << std::endl;
	cout << std::endl;

	// CREATE DIRECTORY
	_mkdir("c://Users/Alexander/Dropbox/School/MasterThesis/Simulations/");

	std::ofstream casefile;
	casefile.open
("c://Users/Alexander/Dropbox/School/MasterThesis/Simulations/testmatrix.txt");

	// WRITE HEADER IN TEST MATRIX
	casefile << "Simulation input overview" << std::endl;
	casefile << std::endl;
	casefile << "  NUMERICAL MODELLING OF BINGHAM FLUID FLOW AND" <<
std::endl;
	casefile << "  PARTICLE TRANSPORT IN A ROUGH-WALLED FRACTURE" <<
std::endl;
	casefile << std::endl;
	casefile << "Master Thesis - Drilling Technology " << std::endl;
	casefile << "Alexander Rikstad Hanssen Stud.techn. NTNU" << std::endl;
	casefile << std::endl;
	casefile << "Case" << '\t' << "Mesh" << '\t' << "dP" << '\t' << "YP" <<
'\t' << "PV" << '\t' << "Hurst" << std::endl;
	casefile << "Num" << '\t' << "Num" << '\t' << "(Pa)" << '\t' << "(Pa)" <<
'\t' << "(Pas)" << '\t' << "(-)" << std::endl;
	casefile << "=============================================" << std::endl;

	// LOOP TO CREATE CASE.DAT FILES
//	for (auto it = fox.begin(); it != fox.end(); ++it ) {

	for (std::size_t i=0; i < mesh.size() ;++i) {
		for (std::size_t j=0; j < pressure.size() ;++j) {
			for (std::size_t k=0; k < yield.size() ;++k) {
				for (std::size_t l=0; l < visco.size() ;++l) {

	casecreator(yield[k],visco[l],pressure[j],dx,dy,index);
					shellcreator(index);
					copymesh(index, mesh[i]);
					casefile << index << '\t' << mesh[i] << '\t' <<
pressure[j] << '\t' << yield[k] << '\t' << visco[l] << '\t' << hurst[i] <<
std::endl;
					index = index + 1;
				}
			}
		}
	}
	cout << "Case folders created in directory:" << std::endl;
	cout << "(c://Users/Alexander/Dropbox/School/MasterThesis/Simulations)"
<< std::endl;
	cout << std::endl;
	return 0;
}
```

A.3.3 VELOCOLLECTOR

```
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <direct.h>
#include <exception>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
#include <windows.h>
#include <list>

using namespace std;

void main ()
{
      std::ofstream velodata;
      velodata.open
("C://Users/Alexander/Dropbox/School/MasterThesis/Modeling/EXCEL/velodata.txt")
;                 // Opens results file
      velodata << "Vxavg;Vxmax;Vyavg;Vymax" << std::endl;

      int casenum = 94;

      for (int i=94; i <= 95  ;++i) {
            std::ostringstream oss;
            string input_dir =
"C://Users/Alexander/Dropbox/School/MasterThesis/Simulations/Batch4/";
      // Specifies directory
            oss << fixed << showpoint << "case_" << std::setw(2) <<
std::setfill('0') << casenum;              // Specifies Case folder
            string input_folder = oss.str();
            string file_name = "/Results/velodata.txt";
                                                              // Specifies
filename

            string STRING;
            ifstream infile;
            infile.open ((input_dir+input_folder+file_name).c_str());

            getline(infile,STRING);                                        //
Saves the line in STRING.
            velodata << STRING << std::endl;                         // Prints
our STRING.
            infile.close();
            casenum += 1;
      }
      velodata.close();
}
```

A.3.4 BREAKTHROUGH CURVES

```
#include <fstream>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <direct.h>
#include <exception>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <stdexcept>
#include <string>
#include <vector>
#include <windows.h>
#include <list>

using namespace std;

int main()
{
	// Declaration
	int casenumber;
	int batch;
	double time;
	double concentration;
	int points = 100;		// Number_of_monitor_dumps found in case.dat

	// Opens output-file
	std::ofstream output;
	output.open
("C://Users/Alexander/Dropbox/School/MasterThesis/breakthrough/output3.txt");

	// Declares breakthrough matrixes
	std::vector<double> breakthrough, timesteps;

	// Specifies breakthrough node lines (also found in case.dat)
	std::vector<int> nodes;
	nodes.push_back(3);
	nodes.push_back(10);
	nodes.push_back(20);
	nodes.push_back(30);
	nodes.push_back(40);
	nodes.push_back(50);
	nodes.push_back(60);

	// User specifies casenumber. Batch folder automatically chosen.

	std::cout << "Enter a casenumber between 1 and 108 (an integer): ";
	std::cin >> casenumber;

	if ( (casenumber >= 0) && (casenumber <= 27) ) {
		batch = 1;
	}
	else if ( (casenumber >= 28) && (casenumber <= 54) ) {
		batch = 2;
	}
	else if ( (casenumber >= 55) && (casenumber <= 81) ) {
		batch = 3;
	}
	else if ( (casenumber >= 82) && (casenumber <= 108) ) {
```

```
            batch = 4;
        }
        else {
            std::cout << "Please enter case number between 1 and 108" <<
std::endl;
            return 1;
        }

        // Loops through BreakthroughLineAtNodeColumn*.dat files
        // and copies data to breakthrough vector

        for (int i=0;i<=6;++i) {
            std::ostringstream casenum, nodenum, batchstring;
            batchstring << fixed << showpoint <<
"C://Users/Alexander/Dropbox/School/MasterThesis/Simulations/Batch" <<
std::setw(1) << batch << "/";
            casenum << fixed << showpoint << "case_" << std::setw(2) <<
std::setfill('0') << casenumber;
            nodenum << fixed << showpoint << "/BreakthroughLineAtNodeColumn" <<
nodes.at(i) << std::setw(2) << ".dat";
            string output_dir = batchstring.str();
            string index_string = casenum.str();
            string file_name = nodenum.str();
            std::ifstream fin( (output_dir+index_string+file_name).c_str() );

            if (!fin)
            {
                // Exit if file cannot be opened
                cout << "could not open file" << endl;
                return 1;
            }


            // Read and load data into vectors
            while (fin >> time >> concentration) {
                breakthrough.push_back(concentration);
                timesteps.push_back(time);
            }
        }

        // Writes output file in .sdv format
        output << ";N3;N10;N20;N30;N40;N50;N60" << std::endl;
        for (int i=0;i<=99;++i) {
            output << timesteps.at(i) << ";" <<
breakthrough.at(i)/breakthrough.at((points*1)-1) << ";" <<
breakthrough.at(i+points*1)/breakthrough.at((points*2)-1) << ";"
                << breakthrough.at(i+points*2)/breakthrough.at((points*3)-1)
<< ";" << breakthrough.at(i+points*3)/breakthrough.at((points*4)-1) << ";"
                << breakthrough.at(i+points*4)/breakthrough.at((points*5)-1)
<< ";" << breakthrough.at(i+points*5)/breakthrough.at((points*6)-1) << ";"
                << breakthrough.at(i+points*6)/breakthrough.at((points*7)-1)
<< std::endl;
        }

        output.close();
      return 0;
}
```

A.4 Python scripts

A total number of 10 post-processing scripts were developed and used in this thesis to create images and plots of velocity fields, pressure distribution, aperture fields, 3D fracture images, particle transport plots, collecting velocity data and finding the minimum average aperture path perpendicular to the flow direction. Additional scripts were created in order to visualize data where flow was performed perpendicular to original flow direction.

A.4.1 PLOT_VELOCITY.PY

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from numpy import *
from string import split
from pylab import *

mode = 0

lines = file('velocities.dat').readlines()
X     = []
Y     = []
U     = []
V     = []
for line in lines:
    s = line.split()
    X.append( float(s[2]) )
    Y.append( float(s[3]) )
    U.append( float(s[4]) )
    V.append( float(s[5]) )
    mode = mode + 1
mode = int(mode**0.5) + 2

rowX = zeros(mode-2)
rowY = zeros(mode)
rowU = zeros(mode)
rowV = zeros(mode)

Xprepared = zeros((mode-2)*mode)
Yprepared = zeros((mode-2)*mode)
Uprepared = zeros((mode-2)*mode)
Vprepared = zeros((mode-2)*mode)

for i in range(0, (mode-2)*mode, 1):
    Xprepared[i] = X[i]
    Yprepared[i] = Y[i]
    Uprepared[i] = U[i]
    Vprepared[i] = V[i]

Xprepared.shape = (mode-2,mode)
Yprepared.shape = (mode-2,mode)
Uprepared.shape = (mode-2,mode)
Vprepared.shape = (mode-2,mode)

quiver(Xprepared, Yprepared, Uprepared, Vprepared)
# colorbar()
# clim(0,0.2)
#fig = plt.figure()
```

```
#ax = fig.add_subplot(111, projection='3d')
#ax.plot_wireframe(Xprepared, Yprepared, Zprepared, rstride=1, cstride=1)
plt.axes().set_aspect('equal')

B = 2*X[(mode-2)*mode -1] - X[(mode-2)*mode-2]

plt.title('Velocities')
ylim( (0,B))
xlim( (0,B))

savefig('Results/velocities.png',bbox_inches='tight')

plt.close()
```

A.4.2 PLOT_PRESSURE.PY

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from numpy import *
from string import split
from pylab import *

mode = 0

lines = file('pressures.dat').readlines()
X     = []
Y     = []
Z     = []
for line in lines:
    s = line.split()
    X.append( float(s[2]) )
    Y.append( float(s[3]) )
    Z.append( float(s[4]) )
    mode = mode + 1
mode = int(mode**0.5)


rowX = zeros(mode)
rowY = zeros(mode)
rowZ = zeros(mode)

Xprepared = zeros(mode*mode)
Yprepared = zeros(mode*mode)
Zprepared = zeros(mode*mode)

for i in range(0, mode*mode, 1):
    Xprepared[i] = X[i]
    Yprepared[i] = Y[i]
    Zprepared[i] = Z[i]

Xprepared.shape = (mode,mode)
Yprepared.shape = (mode,mode)
Zprepared.shape = (mode,mode)

pcolor(Xprepared, Yprepared, Zprepared)
colorbar()
A = Z[0]
B = 2*X[(mode*mode) -1] - X[(mode*mode)-2]


plt.title('Pressure')
ylim( (0,B))
xlim( (0,B))
```

```
clim(0,A)
#fig = plt.figure()
#ax = fig.add_subplot(111, projection='3d')
#ax.plot_wireframe(Xprepared, Yprepared, Zprepared, rstride=1, cstride=1)

savefig('Results/Pressure.png',bbox_inches='tight')

#show()
plt.close()
```

A.4.3 PLOT_FRACTALS.PY

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from numpy import *
from string import split
from pylab import *

mode = 0

lines = file('mesh.dat').readlines()
X     = []
Y     = []
Z     = []
for line in lines:
    s = line.split()
    X.append( float(s[0]) )
    Y.append( float(s[1]) )
    Z.append( float(s[4]) )
    mode = mode + 1
mode = int(mode**0.5)

rowX = zeros(mode)
rowY = zeros(mode)
rowZ = zeros(mode)

Xprepared = zeros(mode*mode)
Yprepared = zeros(mode*mode)
Zprepared = zeros(mode*mode)

for i in range(0, mode*mode, 1):
    Xprepared[i] = X[i]
    Yprepared[i] = Y[i]
    Zprepared[i] = Z[i]

Xprepared.shape = (mode,mode)
Yprepared.shape = (mode,mode)
Zprepared.shape = (mode,mode)


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(Xprepared, Yprepared, Zprepared, rstride=1, cstride=1)
plt.title('Mesh')

show()
```

A.4.4 PLOT_APERTURE_XY.PY

```
import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D
from numpy import *
from string import split
from pylab import *

mode = 0

lines = file('mesh.dat').readlines()
X     = []
Y     = []
Z     = []
for line in lines:
    s = line.split()
    X.append( float(s[0]) )
    Y.append( float(s[1]) )
    Z.append( float(s[4]) )
    mode = mode + 1
mode = int(mode**0.5)

rowX = zeros(mode)
rowY = zeros(mode)
rowZ = zeros(mode)

Xprepared = zeros(mode*mode)
Yprepared = zeros(mode*mode)
Zprepared = zeros(mode*mode)

for i in range(0, mode*mode, 1):
    Xprepared[i] = X[i]
    Yprepared[i] = Y[i]
    Zprepared[i] = Z[i]

Xprepared.shape = (mode,mode)
Yprepared.shape = (mode,mode)
Zprepared.shape = (mode,mode)

B = 2*X[(mode*mode) -1] - X[(mode*mode)-2]
ylim( (0,B))
xlim( (0,B))
pcolor(Xprepared, Yprepared, Zprepared)
plt.colorbar()
plt.clim(0,0.0015)
ylabel('y-dir')
xlabel('x-dir')
plt.axes().set_aspect('equal')


#fig = plt.figure()
#ax = fig.add_subplot(111, projection='3d')
#ax.plot_wireframe(Xprepared, Yprepared, Zprepared, rstride=1, cstride=1)
#plt.xlabel('x')
#plt.ylabel('y')
#plt.title('')


show()
```

A.4.5 VELODATA.PY

```
import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D
from numpy import *
from string import split
from pylab import *

mode = 0

lines = file('velocities.dat').readlines()
X     = []
Y     = []
U     = []
V     = []
for line in lines:
    s = line.split()
    X.append( float(s[2]) )
    Y.append( float(s[3]) )
    U.append( float(s[4]) )
    V.append( float(s[5]) )
    mode = mode + 1
mode = int(mode**0.5) + 2
U_max = U[0]
V_max = V[0]

U_av = 0
V_av = 0

for i in range(0, (mode-2)*mode, 1):
    U_av = abs(U_av) + abs(U[i])
    V_av = abs(V_av) + abs(V[i])
    if V[i]**2 > V_max**2:
        V_max = V[i]
    if U[i]**2 > U_max**2:
        U_max = U[i]


U_av = U_av / i
V_av = V_av / i

f = open('Results/velodata.txt','w')
f.write('{};{};{};{}\n'.format( U_av, U_max , V_av , V_max ))
f.close
```

A.4.6 FIND_MINAPERTURE.PY

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from numpy import *
from string import split
from pylab import *
import sys
print sys.argv

mode = 0

lines = file('mesh.dat').readlines()
X     = []
Z     = []
L     = []
for line in lines:
    s = line.split()
    X.append( float(s[0]) )
    Z.append( float(s[4]) )
    L.append( float(s[3]) )
    mode = mode + 1
mode = int(mode**0.5)

# Finds the 2^levels coeff.

W = zeros(mode)
W_avg = 0
count = 0

for i in range (0,mode,1):
    for j in range (0,mode*mode,1):
        if ( X[j] == i ):
            W[i] = W[i] + Z[j]

W_min = W[0]/(mode)



for l in range (0,mode*mode,1):
    W_avg = W_avg + ( Z[l] / (mode*mode) )

f = file('NodeLinesAperture.dat','w')

node = 0

for k in range(0,mode,1):
    W[k] = W[k] / mode
    f.write('{} {}\n'.format(k,W[k]))
    if ( W[k] < W_min ) :
        W_min = W[k]
        node = k

f.close()


c = int(W_min * 10**6)
d = int(W_avg * 10**6)

print
print (' W_min = {} [10^-6 m]  @  Node Line {}').format( c,node )
print (' W_avg = {} [10^-6 m]').format( d )
print

#YP  = int(raw_input('Enter a yield point value: '))


YP = 5
Lx = L[mode*mode - 1]

print '  YP \t dP'
print '  (Pa) \t (Pa)'
print ' --------------'
for i in range(1,15,1):
    a = int(YP*i)
    b = int(2*a*Lx/W_min)
    print ('  {}  \t {}').format(a,b)



#dP = (2*YP*Lx/W_min)

print

raw_input("Press Enter to continue...")
```

A.4.7 MASTER.PY

```
import os
import sys
import os.path

foldername = "Results"

if not os.path.exists(foldername):
    os.makedirs(foldername)

sys.path.append("C:\Users\Alexander\Dropbox\School\MasterThesis\PythonScripts")

#####################################################
#                                                   #
#   RESULTS ARE SAVED IN THE SUBFOLDER /RESULTS/    #
#   FOR EACH CASE FILE. CHANGE DIRECTORY BY         #
#   EDITING RUN_POSTPROCESSING.BAT                  #
#                                                   #
#####################################################

# // CREATE PLOT PARTICLE SOLUTE PLOTS (.PNG-IMAGES)
import plot_concentration

# // CREATE BREAKTHROUGH LINES PLOT (.PNG-IMAGE)
import plot_breakthrough

# // CREATE PRESSURE DISTRIBUTION (.PNG-IMAGE)
import plot_pressure

# // CREATE VELOCITY DISTRIBUTION (.PNG-IMAGE)
import plot_velocities

# // COLLECT VELOCITY DATA FOR CASE_OVERVIEW.XLS IN SDV FORMAT
import velodata
```

A.5 Windows Batch and Linux Shell Scripts

A.5.1 POSTPROCESSING.BAT

```
prompt $
cls

@echo off


echo STARTING POST-PROCESSING
echo[

FOR /l %%x IN (1,1,27) DO (
    copy [DIRECTORY]\PythonScripts\master.py
      [DIRECTORY]\Simulations\Batch4\case_%%x /Y > nul
    cd [DIRECTORY]\Simulations\Batch1\case_%%x
    python master.py
    echo     CASE NUM %%x
)


echo[
echo COMPLETE
echo[

PAUSE
```

A.5.2 RUNSIM.BAT

```
for name in $(seq -w 79 81)

do
    cd
    cd "PROPANICA/case_"$name
    qsub PROPANICA.sh
done
```

A.6 SIMULATIONS DATA

A.6.1 CASE 1-27

#	Scenario	dP	YP	PV	w_av	P	Avg. Vx	Avg. Vy	$(Vx/Vy)_{avg}$	B
		(Pa)	(Pa)	(Pas)	(m)	(-)	(m/s)	(m/s)	(-)	(-)
1	1	500000	0	0,001	0,0004802	0,000	9,683	31,011	1,169	0,000
2	1	500000	0	0,010	0,0004802	0,000	0,968	3,101	1,169	0,000
3	1	500000	0	0,030	0,0004802	0,000	0,323	1,034	1,169	0,000
4	1	500000	25	0,001	0,0004802	0,133	7,325	23,952	1,222	0,819
5	1	500000	25	0,010	0,0004802	0,133	0,733	2,395	1,222	0,819
6	1	500000	25	0,030	0,0004802	0,133	0,244	0,798	1,222	0,819
7	1	500000	50	0,001	0,0004802	0,267	5,204	18,358	1,338	2,307
8	1	500000	50	0,010	0,0004802	0,267	0,520	1,836	1,338	2,307
9	1	500000	50	0,030	0,0004802	0,267	0,173	0,612	1,338	2,307
10	1	350000	0	0,001	0,0004802	0,000	6,778	21,708	1,169	0,000
11	1	350000	0	0,010	0,0004802	0,000	0,678	2,171	1,169	0,000
12	1	350000	0	0,030	0,0004802	0,000	0,226	0,724	1,169	0,000
13	1	350000	25	0,001	0,0004802	0,190	4,468	14,930	1,264	1,344
14	1	350000	25	0,010	0,0004802	0,190	0,447	1,493	1,264	1,344
15	1	350000	25	0,030	0,0004802	0,190	0,149	0,498	1,264	1,344
16	1	350000	50	0,001	0,0004802	0,381	2,540	9,973	1,386	4,727
17	1	350000	50	0,010	0,0004802	0,381	0,254	0,997	1,386	4,727
18	1	350000	50	0,030	0,0004802	0,381	0,085	0,332	1,386	4,727
19	1	200000	0	0,001	0,0004802	0,000	3,873	12,404	1,169	0,000
20	1	200000	0	0,010	0,0004802	0,000	0,387	1,240	1,169	0,000
21	1	200000	0	0,030	0,0004802	0,000	0,129	0,413	1,169	0,000
22	1	200000	25	0,001	0,0004802	0,333	1,702	6,375	1,363	3,527
23	1	200000	25	0,010	0,0004802	0,333	0,170	0,638	1,363	3,527
24	1	200000	25	0,030	0,0004802	0,333	0,057	0,213	1,363	3,527
25	1	250000	50	0,001	0,0004802	0,533	0,975	4,527	1,488	12,312
26	1	250000	50	0,010	0,0004802	0,533	0,098	0,453	1,488	12,312
27	1	250000	50	0,030	0,0004802	0,533	0,033	0,151	1,488	12,312

A.6.2 CASE 28-54

#	Scenario	dP	YP	PV	w_av	P	Avg. v||p	Avg. v p	v||p / v p	B
		(Pa)	(Pa)	(Pas)	(m)	(-)	(m/s)	(m/s)	(-)	(-)
28	2	500000	0	0,001	0,0004827	0,000	13,205	47,291	1,863	0,000
29	2	500000	0	0,010	0,0004827	0,000	1,320	4,729	1,863	0,000
30	2	500000	0	0,030	0,0004827	0,000	0,440	1,576	1,863	0,000
31	2	500000	25	0,001	0,0004827	0,133	10,237	39,868	2,044	0,589
32	2	500000	25	0,010	0,0004827	0,133	1,024	3,987	2,044	0,589
33	2	500000	25	0,030	0,0004827	0,133	0,341	1,329	2,044	0,589
34	2	500000	50	0,001	0,0004827	0,265	7,487	32,334	2,243	1,612
35	2	500000	50	0,010	0,0004827	0,265	0,749	3,233	2,243	1,612
36	2	500000	50	0,030	0,0004827	0,265	0,250	1,078	2,243	1,612
37	2	350000	0	0,001	0,0004827	0,000	9,243	33,104	1,863	0,000
38	2	350000	0	0,010	0,0004827	0,000	0,924	3,310	1,863	0,000
39	2	350000	0	0,030	0,0004827	0,000	0,308	1,103	1,863	0,000
40	2	350000	25	0,001	0,0004827	0,189	6,317	25,655	2,139	0,955
41	2	350000	25	0,010	0,0004827	0,189	0,632	2,566	2,139	0,955
42	2	350000	25	0,030	0,0004827	0,189	0,211	0,855	2,139	0,955
43	2	350000	50	0,001	0,0004827	0,379	3,779	18,478	2,325	3,193
44	2	350000	50	0,010	0,0004827	0,379	0,378	1,848	2,325	3,193
45	2	350000	50	0,030	0,0004827	0,379	0,126	0,616	2,325	3,193
46	2	200000	0	0,001	0,0004827	0,000	5,282	18,916	1,863	0,000
47	2	200000	0	0,010	0,0004827	0,000	0,528	1,892	1,863	0,000
48	2	200000	0	0,030	0,0004827	0,000	0,176	0,631	1,863	0,000
49	2	200000	25	0,001	0,0004827	0,331	2,494	11,517	2,286	2,420
50	2	200000	25	0,010	0,0004827	0,331	0,249	1,152	2,286	2,420
51	2	200000	25	0,030	0,0004827	0,331	0,083	0,384	2,286	2,420
52	2	250000	50	0,001	0,0004827	0,530	1,554	9,426	2,442	7,765
53	2	250000	50	0,010	0,0004827	0,530	0,155	0,943	2,442	7,765
54	2	250000	50	0,030	0,0004827	0,530	0,052	0,314	2,442	7,765

A.6.3 CASE 55-81

#	Scenario	dP	YP	PV	w_av	P	Avg. v||p	Avg. v?p	v||p / v?p	B
		(Pa)	(Pa)	(Pas)	(m)	(-)	(m/s)	(m/s)	(-)	(-)
55	3	500000	0	0,001	0,0004802	0,000	12,972	52,021	2,472	0,000
56	3	500000	0	0,010	0,0004802	0,000	1,297	5,202	2,472	0,000
57	3	500000	0	0,030	0,0004802	0,000	0,432	1,734	2,472	0,000
58	3	500000	25	0,001	0,0004802	0,133	10,088	40,456	2,355	0,595
59	3	500000	25	0,010	0,0004802	0,133	1,009	4,046	2,355	0,595
60	3	500000	25	0,030	0,0004802	0,133	0,336	1,349	2,355	0,595
61	3	500000	50	0,001	0,0004802	0,267	7,426	30,354	2,229	1,617
62	3	500000	50	0,010	0,0004802	0,267	0,743	3,035	2,229	1,617
63	3	500000	50	0,030	0,0004802	0,267	0,248	1,012	2,229	1,617
64	3	350000	0	0,001	0,0004802	0,000	9,080	36,414	2,472	0,000
65	3	350000	0	0,010	0,0004802	0,000	0,908	3,641	2,472	0,000
66	3	350000	0	0,030	0,0004802	0,000	0,303	1,214	2,472	0,000
67	3	350000	25	0,001	0,0004802	0,190	6,238	24,992	2,286	0,962
68	3	350000	25	0,010	0,0004802	0,190	0,624	2,499	2,286	0,962
69	3	350000	25	0,030	0,0004802	0,190	0,208	0,833	2,286	0,962
70	3	350000	50	0,001	0,0004802	0,381	3,794	16,851	2,235	3,165
71	3	350000	50	0,010	0,0004802	0,381	0,379	1,685	2,235	3,165
72	3	350000	50	0,030	0,0004802	0,381	0,126	0,562	2,235	3,165
73	3	200000	0	0,001	0,0004802	0,000	5,189	20,808	2,472	0,000
74	3	200000	0	0,010	0,0004802	0,000	0,519	2,081	2,472	0,000
75	3	200000	0	0,030	0,0004802	0,000	0,173	0,694	2,472	0,000
76	3	200000	25	0,001	0,0004802	0,333	2,488	10,501	2,200	2,413
77	3	200000	25	0,010	0,0004802	0,333	0,249	1,050	2,200	2,413
78	3	200000	25	0,030	0,0004802	0,333	0,083	0,350	2,200	2,413
79	3	270000	50	0,001	0,0004802	0,494	2,019	10,293	2,337	5,946
80	3	270000	50	0,010	0,0004802	0,494	0,202	1,029	2,337	5,946
81	3	270000	50	0,030	0,0004802	0,494	0,067	0,343	2,337	5,946

A.6.4 CASE 81-108

#	Scenario	dP	YP	PV	w_av	P	Avg. v||p	Avg. v p	v||p / v p	B
		(Pa)	(Pa)	(Pas)	(m)	(-)	(m/s)	(m/s)	(-)	(-)
82	4	500000	0	0,001	0,0004827	0,000	10,493	41,685	1,793	0,000
83	4	500000	0	0,010	0,0004827	0,000	1,049	4,169	1,793	0,000
84	4	500000	0	0,030	0,0004827	0,000	0,350	1,390	1,793	0,000
85	4	500000	25	0,001	0,0004827	0,133	7,968	32,552	1,885	0,757
86	4	500000	25	0,010	0,0004827	0,133	0,797	3,255	1,885	0,757
87	4	500000	25	0,030	0,0004827	0,133	0,266	1,085	1,885	0,757
88	4	500000	50	0,001	0,0004827	0,265	5,661	24,190	1,953	2,131
89	4	500000	50	0,010	0,0004827	0,265	0,566	2,419	1,953	2,131
90	4	500000	50	0,030	0,0004827	0,265	0,189	0,806	1,953	2,131
91	4	350000	0	0,001	0,0004827	0,000	7,345	29,180	1,793	0,000
92	4	350000	0	0,010	0,0004827	0,000	0,735	2,918	1,793	0,000
93	4	350000	0	0,030	0,0004827	0,000	0,245	0,973	1,793	0,000
94	4	350000	25	0,001	0,0004827	0,189	4,862	20,189	1,915	1,241
95	4	350000	25	0,010	0,0004827	0,189	0,486	2,019	1,915	1,241
96	4	350000	25	0,030	0,0004827	0,189	0,162	0,673	1,915	1,241
97	4	350000	50	0,001	0,0004827	0,379	2,758	12,348	1,971	4,376
98	4	350000	50	0,010	0,0004827	0,379	0,276	1,235	1,971	4,376
99	4	350000	50	0,030	0,0004827	0,379	0,092	0,412	1,971	4,376
100	4	200000	0	0,001	0,0004827	0,000	4,197	16,674	1,793	0,000
101	4	200000	0	0,010	0,0004827	0,000	0,420	1,667	1,793	0,000
102	4	200000	0	0,030	0,0004827	0,000	0,140	0,556	1,793	0,000
103	4	200000	25	0,001	0,0004827	0,331	1,850	8,113	1,979	3,262
104	4	200000	25	0,010	0,0004827	0,331	0,185	0,811	1,979	3,262
105	4	200000	25	0,030	0,0004827	0,331	0,062	0,270	1,979	3,262
106	4	250000	50	0,001	0,0004827	0,530	1,050	5,117	1,878	11,495
107	4	250000	50	0,010	0,0004827	0,530	0,105	0,512	1,878	11,495
108	4	250000	50	0,030	0,0004827	0,530	0,035	0,171	1,878	11,495