Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Martin Alexander Toresen

Simulation of Clay Swelling Behavior and Mass Transport Utilizing a Discrete Element Method

Master's thesis in Applied Physics and Mathematics
Supervisor: Srutarshi Pradhan
June 2021

NTNU
Norwegian University of
Science and Technology

Martin Alexander Toresen

Simulation of Clay Swelling Behavior and Mass Transport Utilizing a Discrete Element Method

Master's thesis in Applied Physics and Mathematics
Supervisor: Srutarshi Pradhan
June 2021

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

NTNU
Norwegian University of
Science and Technology

Preface

This master thesis has been written during the spring semester of 2021, and is a continuation of my Specialization Project and project report written during the autumn of 2020. This thesis concludes my M.Sc.-degree in Applied Physics and Mathematics. I would very much like to thank my supervisor Dr. Srutarshi Pradhan. Dr. Pradhan works as a Researcher at the PoreLab, Physics Department, Norwegian University of Science and Technology. PoreLab[1], is a Norwegian Center of Excellence which is co-hosted by the Norwegian University of Science and Technology (NTNU) in Trondheim, and the University of Oslo (UiO). Flow in porous media is the primary research field, and both theoretical, computational and experimental work are being conducted by researcher from different scientific backgrounds (PoreLab n.d.). I am grateful for Dr. Pradhan's insights and guidance which have helped shape this work. Part of the work done for this master thesis was presented as a conference poster at the Interpore2021 conference.

The numerical experiments have been conducted using the Julia programming language. If anyone would like to test out the models or utilize it in other projects I will very happily share the code with them. The code will be provided upon request to my e-mail: *martiato@stud.ntnu.no*.

[1] Link to the PoreLab website: `https://porelab.no/`

Abstract

This master thesis will focus on creating a simple numerical model to explore clay swelling in shale. The secondary objectives will be to benchmark the behaviour of the model and numerical results with experimental work on swelling in shale. The advantage of creating a numerical model, over experimental studies, is that one is able to much more quickly try new ideas. As an example, some swelling experiments may take several weeks in the laboratory, while the model will finish in a few minutes. This though will require one to be able to calibrate the model with experiments for it to return sensible results.

Shale is one of the most common rocks on the planet. As such it is commonly encountered as part of petroleum extraction. One of the main components of shale is clay, which has the characteristic property that under the right circumstances it can swell and expand greatly in size. For certain clay types it may swell and grow several times its original size. This property can be very problematic at a drill site, where it can severely affect the output of extraction. This can be very costly. On the other hand, swelling of clay, if better understood, could be taken advantage of to for example aid in the fastening of the casing, which in turn might provide cost-saving opportunities. For this reason it is crucial to understand more about how the swelling happens.

A *Discrete Element Method* was chosen for the model, after looking at experimental results and some previous work done in the area. It is computationally tractable, intuitive and easily explained. It was also believed that the important drivers of the swelling could be captured in the model. The model was developed in Julia, a relatively new programming language designed to be both fast and easy to use, and a modern successor to Fortran and C for numerical simulations. A wellbore annulus was chosen for the problem geometry, with the option to either model it in two or three dimensions. This was, as mentioned, motivated by its similarity to the situation in a borehole. Particles are placed on a regular grid, either as clay, pores or inert quartz particles. Two processes, swelling of clay particles and mass transport, were identified as important and have been implemented. Several parameters such as temperature and stress have also been included to affect the simulations.

An analysis of the cluster structures that appear in the annulus was compared to known results from percolation theory, such as critical percolation threshold and cluster distributions. The apparent clusters and channels that manifest themselves are important to understand the bulk properties of the entire sample. Simulations of both swelling and mass transport were run for different sets of parameters, to see if they capture some of the behaviour that has been observed experimentally. To end with both were included to look at total swelling pressure on the inner ring, which can be compared to the pressure felt by a casing in a wellbore. The results of the model simulations are in line with experimental results and we recognize some of the same behaviours.

Sammendrag

Denne masteroppgaven har som hovedmål å lage en enkel modell for å utforske svelling av leirepartikler i skifer. Modellens oppførsel og numeriske resultater vil også sammenlignes med eksperimentelt arbeid utført på nettopp svelling av leire. Fordelen med numeriske beregninger fremfor eksperimentelle studier på området, er at numeriske simuleringer kan utføres betraktelig raskere. Eksempelvis kan eksperimentelle leiresvelling-studier ta flere uker på laboratorium, mens man kan oppnå tilsvarende resultater på bare noen minutters modellkjøring. Likevel er det viktig å påpeke at datamodeller kun oppnår gode resultater dersom de er kalibrert hensiktsmessig, samt at modellparameterne har realistiske verdier.

Skifer er en av de vanligste bergartene på jorda. Av den grunn så er skifer ofte å finne i forbindelse med oljeutvinning. En av de viktigste komponentene i skifer er leire, som har den karakteristiske egenskapen at den under de riktige omstendighetene kan svelle og øke mye i størrelse. For visse leiretyper så kan økningen være på opptil flere ganger original størrelse. Denne egenskapen kan være veldig problematisk i et oljefelt, hvor det kan det ha en stor innvirkning på utvinningsgraden til oljefeltet. Dette kan være veldig kostbart. Dersom det oppnås bedre forståelse av leiresvelling, kan det tenkes at dette for eksempel kan utnyttes for å sikre *casing*'en i et borehull, som igjen potensielt kan være kostnadsreduserende. Av denne grunn er det essensielt å få en bedre forståelse av svelling av leirepartikler i skifer.

En *discrete element method* ble valgt etter undersøkelse av eksperimentelle resultater og tidligere arbeid gjort innenfor feltet. Denne formen for modellering er lett å forstå, samtidig som man kan holde modellens kjøretid på et akseptabelt nivå. Samtidig antok man at viktige svelling-drivere kunne inkluderes i denne type modell. Modellen har blitt utviklet i språket Julia, et relativt nytt programmeringsspråk. Julia vil for mange fremstå som en slags moderne arvtager til Fortran og C for numeriske simuleringer, ved å både være raskt og lettanvendelig.

En *sirkelring*, lignende et borehull, har blitt brukt som geometrien i modellen, og har blitt implementert både for to og tre dimensjoner. Formen har blitt valgt nettopp for å etterstrebe likhet med en brønn i et oljefelt. Partiklene blir plassert på et grid, og vil være enten leire, tomrom eller inert kvarts. To hovedprosesser har blitt identifisert som nøkkelprosesser, svelling av leirepartikler og masseflyt, og begge disse prosessene har derfor blitt implementert i modellen. Flere parametere, blant annet temperatur og trykk er med på å påvirke simuleringene.

Analyser av klyngestrukturer som oppstår i sirkelringen blir sammenlignet med teoretiske verdier fra perkolasjonsteori, spesielt klyngefordeling og kritiske terskelverdier. Klyngestrukturene og kanalene som oppstår er vesentlige for de sammensatte egenskapene til prøven med tanke på svelling. Simuleringer av svelling og av masseflyt har blitt kjørt for forskjellige parameterverdier, samt for å undersøke om modellkjøringene ville klare å fange noen av egenskapene som har blitt funnet eksperimentelt. Til slutt ble svelling- og masseflyt-modulene koblet sammen og det totale svelletrykket inn mot den indre ringen ble undersøkt. Hensikten var her å simulere tilsvarende trykk mot casing'en som man vil oppleve å finne i en oljebrønn. Modellkjøringsresultatene ga mange av de samme resultatene som ble observert eksperimentelt.

Contents

List of Figures

List of Tables

1 Introduction

The work conducted in this master thesis is a continuation of my Specialization Project and the project report that I wrote during the autumn of 2020. The main objective during last autumn's work was to get familiarized with modelling of clay swelling, and to start building the basis for a model which could be used to explore a wider range of aspects later in the master thesis. On the other hand, this master thesis has been written with the purpose of being self-contained. The model that is built is based on the principles of the discrete element method and the motivation for this, along with detailed descriptions of how it has been implemented, will be presented in the proceeding chapters.

The remainder of this introductory chapter will serve a two-fold purpose. Firstly, the introduction will provide a foundation about clay in Section 1.1. The focus will be to describe clay and fields where such a model framework might be used to gain better understanding of areas where clay is present. The petroleum industry, more specifically the part concerning well drilling and well plugging, is considered relevant for clay swelling understanding and modelling. More about this will be presented in Section 1.2.

The other purpose of this introduction will be to outline the motivation behind creating this model and to give a background for the choices made later on. This will primarily by provided in Section 1.3, where relevant literature and some experimental findings will be provided. These will create a foundation for the work with this master thesis work.

1.1 Clay

As a key component of clay and soil, clay minerals are one of the most prevalent naturally occurring minerals (Nandi 2018). They consist of 1 nm thick crystalline phyllosilicate layers (Hansen et al. 2012). The clay minerals are divided into groups two groups based on the atomic structure of these layers, the 1:1 and the 2:1 groups. These layers are either tetrahedral silicate sheets or octahedral hydroxide sheets. 1:1 clay minerals consists of one of each, while a 2:1 clay mineral will have one octahedral sheet in between two tetrahedral sheets. Depending on the structure and the molecular content, clay minerals will exhibit very different properties. When it pertains to swelling, particles with a high swelling capacity are found in the 2:1 group. As an example, montmorillonite in the 2:1 group can swell significantly due to water entering between the layers. Hansen et al. (2012) further describe how temperature impacts swelling, more specifically how clay can transition from a passive to an active swelling stage. There are several other factors that influence the swelling properties, such as pressure, presence of water and ions, as well as salinity (Balaban et al. 2015).

Their ability to both swell and contract significantly is one distinguishing feature of clay minerals. This property of clay can be used productively in certain processes, but it can in other cases have disastrous consequences if the clay structure collapses suddenly and uncontrollably. This can be catastrophic for buildings and infrastructure, and may cause fatal accidents. One example of the latter is the very sudden decomposition of quick clay, which can be caused by very minor load changes, but which in turn can cause large areas of clay to transition into fully floating material. Quick clay can form when clay minerals are deposited in contact with salts in seawater which causes the formation of electrostatic bindings between particles. These bindings can be broken at later stages in the presence of freshwater. This can for instance occur if land rises, as it has done in Scandinavia after the melting of the heavy ice cap after the Ice Age (Norwegian Geotechnical Institute n.d.). One large quick clay landslide was filmed in Rissa, Norway, back in 1978[2]. The film shows how important it is to conduct proper geological and chemical investigations of clay properties before permitting construction of infrastructure or buildings in clay-rich soils, especially in areas susceptible of having quick clay. Unfortunately, another fatal quick clay slide occurred on December 30$^{\text{th}}$

[2]Film retrieved from `https://www.nrk.no/video/rissaraset-29-april-1978_104908` [Accessed 13-Jan 2021]

2020 in Gjerdrum, Norway (The Norwegian Water Resources and Energy Directorate 2021). Ten people sadly lost their lives and the quick clay slide also caused massive infrastructure damages to the area[3]. In my opinion, this should further motivate towards research in this area with the hope that better understanding can lead to better prevention mechanism which can help to secure lives and infrastructure. Global warming and accompanied changes in temperature and precipitation patterns might indirectly increase this hazard as heavy rainfall can cause sudden, large increases in erosion in small streams or rivers, causing smaller landslides which again can set of larger quick clay slides (Solberg & Hansen 2021)

Clay sediments can through diagenesis form sedimentary rocks. The diagenesis process takes place when the temperature and pressure in the sediment increase (Burdige 2018). This is naturally occurring as new sediments are deposited on top of already deposited material. Such sediment burials take place over long time periods, and the pressure and temperature will therefore increase slowly, at least if compared to human lifetime. If biomass are buried in the absence of oxygen, hydrocarbons might form in the process as well. Which chemical compositions these hydrocarbons will take depends on the temperature and pressure gradients that the hydrocarbons are exposed to. Oil and gas has a lower densities than water and will migrate upwards from the sedimentary parent rock if the porosity and permeability allow it. A trap formation, consisting of an impermeable cap a top of a porous rock type might enclose enough petroleum for it to become considered an economically viable petroleum field. A schematic example of such petroleum trap, with petroleum extraction through a well, can be seen in Figure 1. Understanding how porosity and permeability varies in rock have been considered a very important research question within geophysics and petroleum engineering, and there are numerous publications on the matter. Some recent publications include Wood (2020) which investigates how porosity and permeability can be predicted utilizing data from multiple well-logs and machine-learning. Clay minerals are also a component of shale which is a fine-grained

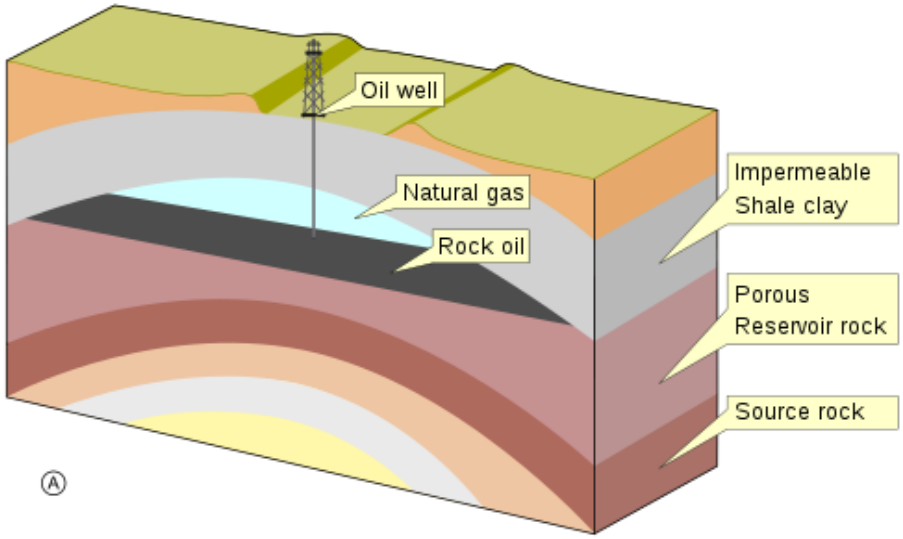


Figure 1: Illustration of oil and gas reservoir with an impermeable rock cap by MagentaGreen (2014), distributed under CC BY-SA 3.0.

sedimentary rock. 55% of all sedimentary rocks on the planet, are shale (Zou 2017). As such, it is the most common sedimentary rock on the planet. Across the globe there are large deposits of oil and natural gas in shale-rich environments. Getting the hydrocarbons of out the shale can, on the other hand, be very difficult. There are many techniques that are used, among them hydraulic fracturing (Eberhardt & Amini 2018), also

[3] https://www.nrk.no/nyheter/leirskredet-i-gjerdrum-1.15307406 [Accessed 23-Jun 2021]

known as fracking, which is quite controversial due to its potential adverse environmental consequences.

Clay has also played a fundamental role throughout human history as a widely used construction material and in ceramics. Clay bricks is known to have been utilized as early as in the Mesopotamian period according to Fernandes et al. (2010), which means that the building material has been in use for around 6000 years. Akinshipe & Kornelius (2017) highlights clay bricks' cost effectiveness and flexibility in construction as some of the material's major advantages. Some of the chemical and thermodynamic processes that clay undergoes in the different firing stages in brick manufacturing process are also explained in Akinshipe & Kornelius (2017). In more modern times, there have been many new applications of clay in different industrial processes, including paper-making and chemical filtering of water. For instance, the clay mineral kaolinite is added to the cellulose fiber slurry as a *filler* and it is furthermore used also in for paper coating according Hubbe & Gill (2016), and the usage of kaolinite does improve the paper's appearance (Bundy & Ishley 1991).

As an aside, a body of rock consists of enormous amounts of particles. Let us take a rock cube measuring $1m \cdot 1m \cdot 1m$ as an example. If we further assume that the particles are perfect spheres and that everyone has a diameter of 63 μm. A diameter of this size is in the lower range of what is considered to be *very fine sand* and *silt* in accordance with the much used Udden-Wentworth grain-size scale, where an upgraded version can be found in Blair & McPherson (1999). By further assuming that the sand/silt particles are packed together with a *simple cubic* structure and touch each other, such hypothetical rock cube of $1m^3$ will consist of a total of $\approx 4.0 \cdot 10^{12}$ sand/silt particles, calculated as shown in Equation 1. A numerical simulation will never be able to approach such a large number, but hopefully a particle based model with of a more modest size will still provide some interesting results and insights.

$$\textit{Number of sand/silt particles in cube} = \left(\frac{1m}{63\,\mu m}\right)^3 = 4.0 \cdot 10^{12} \tag{1}$$

1.2 Petroleum industry

The petroleum sector is another large field where understanding of clay behavior is considered to be important. Since the late 1960's the Norwegian petroleum industry has contributed radically towards the economic development of Norway. Over the 50-year time span the petroleum industry has generated astonishing export revenues, as seen in Figure 2, for the Norwegian state and population through taxes and partially or fully state-owned companies such as Equinor and Petoro. These incomes have allowed the state to build up one of the largest sovereign wealth funds in the world and is meant to serve as a economic buffer for the days after the *petroleum era*. Maybe even more importantly, the petroleum era has led Norwegian companies to achieve world-leading expertise within some offshore technology fields and this knowledge can hopefully be re-focused towards new and more sustainable areas in the coming decades. One of the more imminent operational challenges for the Norwegian petroleum sector is that many offshore oil wells are approaching their end of life. This means that they must be securely plugged and abandoned. It is estimated that 3000 wells must be plugged at the Norwegian continental shelf, and that the total costs of the plugging and abandonment (P&A) processes can amount to staggering 900 billion Norwegian kroner (SINTEF 2018). When the well is plugged and abandoned it is important that the field is securely and permanently sealed off, to ensure that leakages and environmental harm will not occur.

As a result of growing concern of climate change, the petroleum industry is facing an uncertain future both with respect to demand and sales prices. But there are also interesting opportunities that lay ahead, and old offshore oil fields might serve an important part of a greener future where industrial carbon dioxide emissions are captured and transported to a designated long term place for storage. The geological formations that once contained large amount of petroleum or gas in porous rocks underneath a natural seal of an impermeable

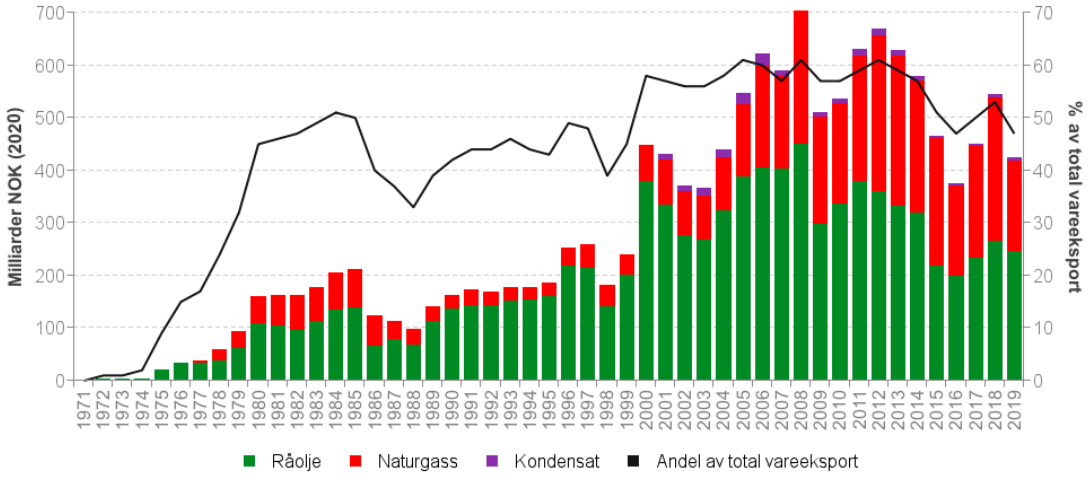


Figure 2: Export value of Norwegian petroleum products in 1971-2019: oil in green, natural gas in red, condensate in purple, in the unit Billion Norwegian Krone [10^9 NOK] on the left axis. On the right axis the black curve shows how large the export percentage that petroleum products have constituted. The numbers are corrected for inflation in accordance with the consumer price index. Source: Norsk Petroleum (2020)

rock layer, can be used to store CO_2, and thus play an important role in carbon capture and storage (CCS) processes. Injecting CO_2 into oil reservoirs is not an new idea, and Equinor has done this at the Sleipner field since 1996 (Equinor 2019). Using CO_2 injections to extract larger shares of the available oil in a field is called CO_2-EOR (CO_2-enhanced oil recovery). Here CO_2 injections can help extract petroleum that might otherwise have been impossible to extract from the field. Núñez-López & Moskal (2019) provide a description of CO_2-EOR, and how this can lead to lower emissions associated with the oil extraction. Additionally it is worth to mention that increased environmental focus has led petroleum companies to change from synthetic drilling fluids to water-based drilling fluids. While water-based drilling fluids are more environmentally-friendly, water can have a large effect on clay swelling. This can cause difficulties such as delays and cost increases. To identify swell-inhibiting measures can therefore be valuable to petroleum companies. Better understanding of the clay swelling process is therefore presumed to also be of interest.

CO_2-capturing from industrial processes is currently not being exploited without significant subsidies, e.g. as exemplified by the newly ratified state-funded *Longship*-project (The Norwegian Government 2020) where CO_2 from the Norcem's cement plant in Porsgrunn will be captured. As the cement industry is a very large global CO_2-emitting sectors, it is important to find emission reduction measures for this sector according to International Energy Agency (2020). Production of blue hydrogen, where natural gas is reformed to hydrogen and the CO_2 is captured and stored, is another opportunity for the petroleum industry to create low-emission products, will also require storage of large quantities of CO_2 in the bedrock. Long-term storage of CO_2 and plugging of old petroleum fields share the necessity of creating lasting, safe and economical plugging and abandonment of the wells. Utilizing clay to further enforce the sealing might become an option for such purposes, and swelling clays was already back in 1982 suggested as a measure to reduce the permeability of the sealing, hence also strengthening the sealing, of nuclear waste deposits (Moore et al. 1982).

As part of the process to understand the P&A and CCS processes better it will be beneficial to establish numerical methods that can be used to conduct numerical experiments. From a cost perspective, compu-

tational studies can be very cheap in comparison to exploring multiple physical well samples. Although numerical experiments cannot substitute the exploration of actual rock samples from the wells, it can facilitate faster learning and investigation of a larger variety of parameter combination than what would be feasible using only physical samples.

1.3 Motivation and relevant literature

Swelling of clay has been of interest in the petroleum sciences for many years, especially within the drilling phase where swelling of clay may cause problems. An illustration of a wellbore can be seen in Figure 3, with a cross-sectional perspective to the left and partially transparent 3D annulus structure to the right.

Wellbore instability and formation damage are two large concerns during drilling, according to Kmieć et al. (2018). After drilling and before cementing, the drilling fluids (especially water-based) may activate and swell the clay into the wellbore. This may lead to problems in the casing or other structural deficiencies in the wellbore. Formation damage is when the properties of the formation, the rock structure encasing the borehole, changes. For instance, the permeability of the rock matrix may increase, leading to less oil recovery (Kmieć et al. 2018). A potential application of clay swelling is in the cementation phase of a wellbore. If it becomes possible to induce a stress or inward pressure, one may possibly utilize clay swellling to fasten the casing in place, potentially saving costs at the same time. As previously mentioned, clay

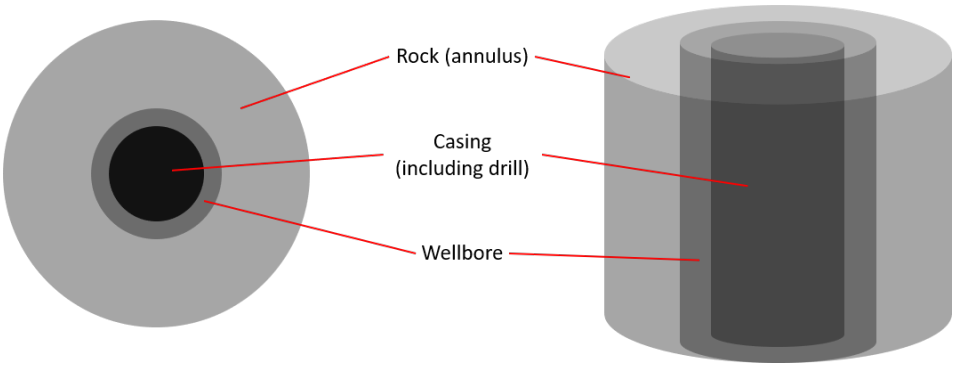


Figure 3: Wellbore overview

is a significant component in shale, one of the most common rocks on earth. There has been conducted experimental work and research on shale to look at swelling behaviour and creep. There is also significant work done on the swelling of clay particles, where one study why and how the particles swell. However, to connect these two regimes is difficult, especially with respect to how one can connect the swelling of clay particles at a micro-level to the observed bulk swelling of large shale samples. Clay swelling is one of many different modes of deformation, strain or swelling that occurs in shale samples. In this thesis, the main purpose is to create a simple model of how swelling of clay particles impacts bulk properties of a sample. Therefore, relevant literature has been reviewed in order to understand the most important drivers behind the swelling process.

Rybacki et al. (2017) performed creep experiments on samples of Posidonia shale. Figures 1a-f in Rybacki et al. (2017), shown in Figure 4, show the evolution of strain of shale samples over time by applying differential stress, confining pressure and temperature. Rybacki et al. (2017) found that both high temperature and higher differential stress enhances the deformation rate of the sample. It is worth to note that some of their samples experienced creep and sample failure, both of which are properties that is outside of the scope of this thesis.

Deriszadeh et al. (2014) used an electrical potential gradient to a set of clay and shale samples to facilitate the process of swelling. In Figure 5, which is a page from Deriszadeh et al. (2014)'s paper, is included to show how two different clay types samples react when swelling. The upper plot contains kaolinite clay and lowest plot contains bentonite clay, and the plots show how they react when subjected to an applied electrical potential. It is readily apparent that the two different clay types react differently to the applied potential, where the latter seems to behave more linearly than the first. On its own, without being subjected to an outside force of some sort, the swelling of clay may take a very long time to reach its maximum size. However, under electrical potential gradient conditions, an accelerated swelling was observed and this show that it is possible to trigger the swelling of clay by applying an electrical field. Deriszadeh et al. (2014) hypothesize that the electrical field help drive mass transport through the sample and enhance fluid flow. This is an interesting finding, and mass flow will therefore be implemented in the numerical model built during the work with this thesis.

The aim of this project to contribute with a model framework that can be utilized to study clay swelling behavior further. I will utilize numerical methods to study how bulk clay swelling can occur. Most of the cases that will be studied are created to simulate the characteristics of the immediate surroundings of a petroleum borehole, thus all of the modelled samples are shaped as annuli either in 2D or 3D. When constructing oil wells, it is very important to ensure that the wellbores and casings are securely fastened to the adjoining rock on the side to prevent dangerous and environmentally harmful blowouts. Since wells can be multiple kilometers deep, it might cut across several different rock types, and the fastening process must therefore account for the different rocks properties when deciding on the chemical and mechanical layout of casing. The wellbore is typically anchored to the adjoining rock by filling the void with cement. One suggested hypothesis is that clay, which might already be present in the adjoining rocks, can be triggered to swell to seal the gap between the casing and rocks in the borehole. This can potentially reduce the need of cement, and can thus provide cost-savings and help to reduce the emission intensity. Cement production, and especially the calcination step is a very large CO_2 emitter.

Temperature, stress gradient and different particle densities and distributions are identified as the most fundamental parameters that such model must contain, and these parameters will therefore be included in the modelling framework. By building a computational model, one could simulate different external forces' impact on different rock compositions, more easily than by drilling wells and investigating the samples in a laboratory. As an example, some of the experiments by Deriszadeh et al. (2014) took several weeks to complete. A computational model should take both shorter time to run, and also provide a much more cost-effective option. It should however not be underestimated how important it is with experimental work to calibrate computational models correctly. Hence, the proposed model in this thesis cannot provide realistic results without the insights and results provided by experimental studies.

To conclude, the aim of this thesis is to create a framework and methods that can be applied efficiently to seek for better understanding of clay, and in particular how clay swelling occurs. The goal of this master thesis will be to create such a model. To achieve that goal we will start by creating a simplified computational model and gradually increase complexity.

The remaining parts of this master thesis is structured as follows: Chapter 2 will provide a theoretical foundation about aspects used directly or indirectly when building the clay modelling framework. This includes some examples from literature, description of some interesting modelling methods and software, as well as some details about using Julia as programming language for scientific programming work. Chapter 3 presents the model framework that has been created to study clay swelling, clustering and mass transport. The results of the computational experiments will be presented in Chapter 4. The concluding remarks are found in Chapter 5. Lastly, suggested future work will be described in Chapter 6.

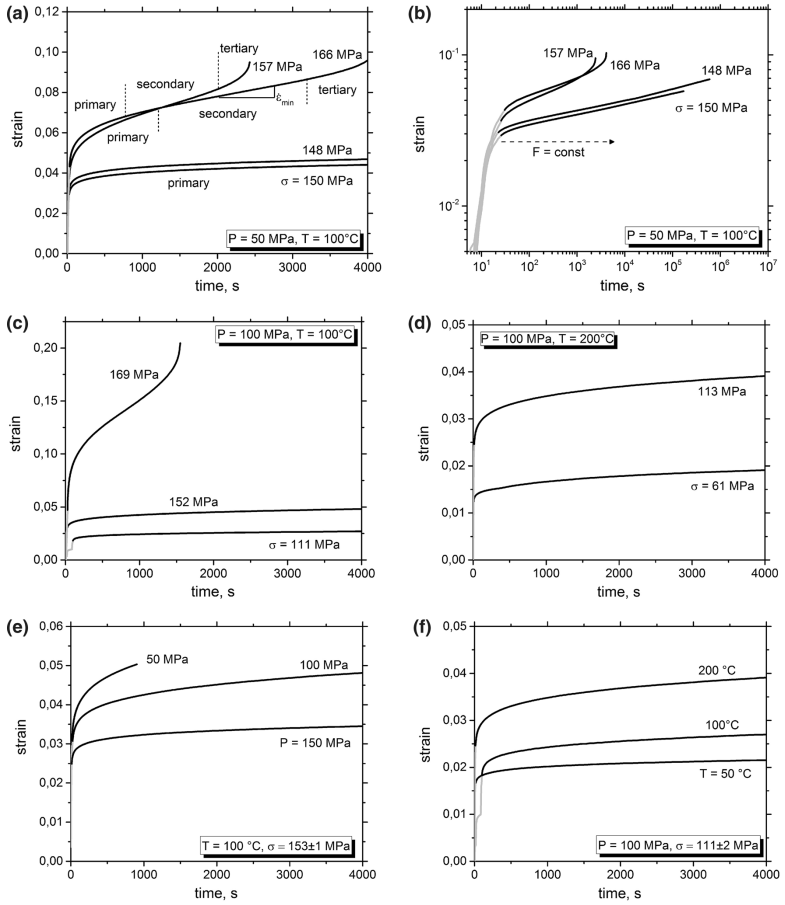


Fig. 1 Creep curves of Dotternhausen Posidonia shale in relation to applied differential stress σ (**a**–**d**), confining pressure P (**e**) and temperature T (**f**). At high stress, creep curves show not only primary (decelerating), but also secondary (quasi-steady state) and tertiary (accelerating) creep leading to final failure. Increasing pressure reduces the creep rate (**e**), whereas increasing temperature enhances the creep rate (**f**). For comparison, all curves are cut off at 4000 s except in (**b**), where complete curves are shown in log–log scale. During the first few seconds, the applied force F was increased up to the desired level that stabilized within 10–20 s and subsequently held constant until manual test termination or sample failure (**b**)

d). Samples subjected to low differential stress showed primary creep behavior, where the strain increment per unit time step continuously decreases with increasing time (primary or decelerating creep phase). In contrast, samples loaded at high differential stress showed subsequently secondary creep with a linear increase in strain with increasing time (i.e., constant creep rate) and finally tertiary (accelerating) creep until failure. The transition from primary to secondary creep occurred in a narrow stress range, corresponding to about 84–90% of the compressive strength obtained in constant strain rate tests of 5×10^{-4} s^{-1} under similar P–T conditions (Rybacki et al.

Figure 4: Page 4 from Rybacki et al. (2017) has been provided here for the purpose of highlighting the experimental work conducted by the mentioned authors. Figure credits: Rybacki et al. (2017).

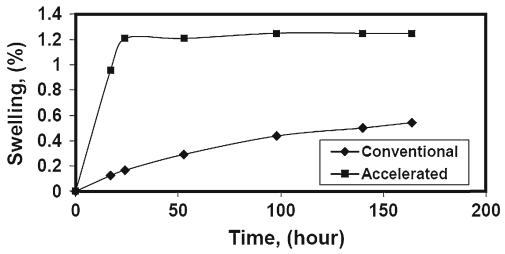


Fig. 4 Swelling behavior of reconstituted kaolinite clay sample in conventional and electrically induced free swell tests

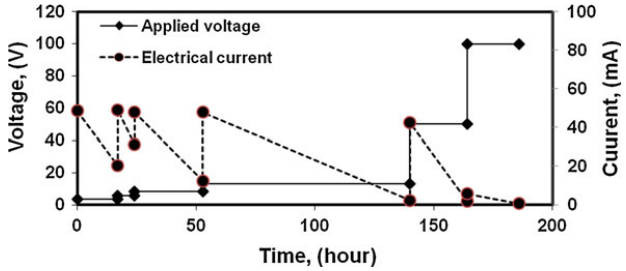


Fig. 5 Applied electrical potential and recorded electrical current across the sample in the electrically induced free swell test on reconstituted kaolinite clay sample

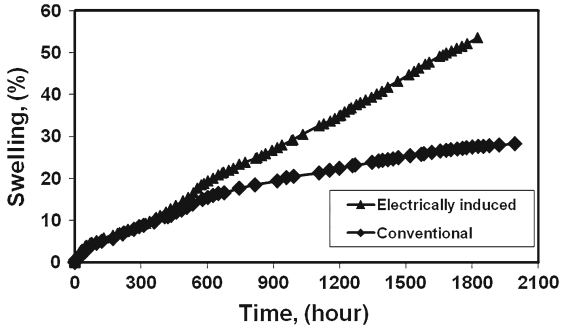


Fig. 6 Swelling behavior of reconstituted bentonite clay sample in conventional and electrically induced free swell tests

the experiment (Fig. 5), the applied electrical potential was required to be raised to maintain a small electric current. At 164 h, the electrical potential was raised to 100 V and the recorded electrical current was about 6 mA and decreased to 2 mA in 24 h.

The experimental data of the conventional and electrically induced free swell tests on reconstituted bentonite clay samples (Table 1) are shown in Figs. 6 and 7, respectively. From

Figure 5: Page 4 from Deriszadeh et al. (2014) has been provided here for the purpose of highlighting the experimental work conducted by the mentioned authors. Figure credits: Deriszadeh et al. (2014).

2 Theoretical Background

A theoretical background about the aspects regarded as relevant for modelling and simulation of clay swelling will be provided in this chapter. A basic foundation of the driving forces relevant for such model will be provided in Section 2.1. A short review of how mass transportation can be modelled when solving different types of problems will be presented in Section 2.2. Section 2.3 introduces percolation theory and provides expected behavior for cluster sizes for neighboring particles in a grid. Sample randomness and particle distributions will also be briefly presented. Lastly, in Section 2.4, details about Julia as programming language and motivation for utilizing Julia for this thesis will be presented.

2.1 Forces and particle interactions of significance in clay swelling

Shale rock samples consists of a complex combination of minerals, clays and other organic materials. There may also be liquids and gasses. As a whole it will be subjected to different temperatures, pressure and stresses. There will be chemical processes happening and charged ions will be interacting with each other. The microstructure of the shale rock itself will impact the physical properties of the sample. In porous structures, where the surface area of the surrounding solids are large in comparison to the free-flow areas, capillary forces and surface modelling plays a major role in determining fluid flow. All in all, shale rocks provide a difficult environment to recreate fully in a numerical model. In this section some of these underlying mechanisms will be discussed in order to lay the groundwork for how the model framework is built up and why it was created in the chosen way.

Firstly, swelling of clay will be discussed. Clay swelling require that the particles is in contact with water. This water will then penetrate the layers of clay, causing the material to increase in size. The exact chemistry and mechanics are complicated but can be partially explained by interlayer forces between clay layers and repulsive double-layer forces between clay particles at short distances. More details about this can be found in Anandarajah et al. (2012) and Shang et al. (2018). At longer range there will be attractive *van der Waals* forces, but they are neutralized by the stronger double-layer forces at shorter ranges which will ensure that the particles are kept separated. (Li et al. 2021) As water is generally regarded as a main driver of clay swelling, it should be assumed that water will have to be present in the rock sample to create clay swelling. This water-presence assumption will be used throughout the entire thesis.

In a wellbore, and the surrounding formation, there will definitely be stresses and pressures in the rock. Gravity may also be relevant, but is believed to be of less importance than stress and pressure gradients in the rock structures. No gravitational forces will therefore be added to the programming modules handling 3D rock structures. Another simplification which have had to be undertaken, is the neglection of pore deformation due to changing surroundings. This might be very relevant in some petroleum based applications, because the change of fluid content in porous rock formations might change the pore structure. This can for instance happen in petroleum fields when oil and gas are withdrawn from the reservoirs. This will often cause rock compaction. Despite deemed potentially quite relevant for the drilling applications and enhanced oil recovery, changes of mechanical properties and pore deformation will not be considered in this thesis. The reason for this is that there already exist dedicated models for this use, such as COMSOL's *Biot Poroelasticity - Application ID: 483* (n.d.). Aspects such as applied stress and pore pressure is present in the Biot theory, and how additional presence of osmotic effects can influence deformation in shales is explained in more detail in Sherwood (1993).

2.2 Modelling of mass transport through porous structures

Mass transport is of major interest in multiple engineering fields, ranging from vast hydropower dams to microscale modelling of creeping flow through porous rock formations. For this thesis, where clay swelling around the wellbore is of special interest, it is therefore of importance to find a suitable way of accounting for water and clay particle flow. Since clay swelling only can occur under circumstances where there is water present, as well as empty void to expand into, it can be challenging to find a model or method which accounts for all relevant aspects simultaneously. Therefore, a short review of relevant mass transport modelling aspects will be provided in this section. This theory and knowledge base helped make appropriate assumptions for the mass transport aspects included in my model. It also helped to make suitable simplifications and to ensure computational tractability. The model implementation that was made during this work will be presented in Chapter 3.

Fluid flow modelling can primarily be approached in to different ways, namely through a *Lagrangian* or *Eulerian* (Çengel & Cimbala 2010) approach. These two approaches differ in the sense of how fluid motion is described. The Lagrangian is based around tracking each individual particle, as well as how it moves and interacts with its surroundings. In the Lagrangian description such a particle is called a *fluid parcel*. In the Eulerian description one looks at the flow through a region of space, referred to as a *Control Volume* without focusing on any specific particles. One instead looks at aggregate measures through the region such as average flow velocity. Although these two variants of flow modelling have not become a major part of the chosen modelling framework developed during my work with this thesis, some aspects from the *Lagrangian* approach do share some resemblance with the particle handling in the chosen modelling format. Specifically so with respect to how individual particles can be controlled on a grid structure. More details about this will be provided in the proceeding chapter.

These specifications are reflected in computational fluid dynamics, where Eulerian simulations predominately use a fixed mesh-structure and control volumes to keep track of the action. On the other side, Lagrangian simulations will instead often be based around defining particles as nodes, and then by applying a coordinate system to be able to track position, velocity or acceleration of the particle as it is subjected to influence from external force. These forces are then often quantified as vector fields to ease calculations.

Flow in porous media is a major research area within the petroleum industry, especially in relation to understanding how the oil, gas and water flows within a reservoir affect the pressure and production rates from the petroleum field. Understanding these concepts through detailed flow modelling is also fundamental for enhanced oil recovery. An overview of important aspects along with an introduction to flow simulation within petroleum reservoirs is given in Lie (2017). Lie (2017) points out that there are several size scales which must be considered in such modelling, ranging from a micrometer-scale where pores and pore channels are the main parameters, to large rock formations which stretch across entire reservoirs. This can be seen on page 9 in the presentation provided by Lie (2017). Since my thesis is mainly concerned with a particle representation, one can categorize the clay swelling and mass transport modelling as more on the *microscale*-side. Furthermore, both single-phase and multi-phase flow is described within a reservoir modelling setting. For modelling of well and flow of petroleum into the well from the surrounding rock, an *analytic subscale model* is presented on page 113-123 by Lie (2017). Although such modelling is not considered directly within this thesis' scope, there are some similarities and basic understanding of flow behaviour around a well which has been implemented in the mass transport model, e.g., that particles and liquid will be subjected to a pressure gradient and lead them towards the well.

Zolotukhin & Ursin (2000) also provide introductory level insights about petroleum reservoirs and provide the definition for several physical properties and equations, such as the definition of absolute porosity and effective porosity of a rock formation. The first is the share of all pore volume divided by the total volume

of the rock structure. The latter only accounts for the volume of interconnected pores, and the definitions of porosity is listed by Equations (2) and (3), respectively. Permeability, a measure of how easily a fluid flows through a porous medium, can be derived from Darcy's Law which is provided by Equation (4). ∇p is here the pressure gradient across the rock section in Pa. μ is the dynamic viscosity of the fluid in the unit: $Pa \cdot s$. Lastly, the k denotes the permeability of the rock and have the same dimension as surface area, i.e., m^2, and normally k is referred to as *darcy*.

$$\phi_{abs} = \frac{V_{pores}}{V_{total}} \tag{2}$$

$$\phi_{eff} = \frac{V_{eff}}{V_{total}} \tag{3}$$

$$q = -\frac{k}{\mu}\nabla p \tag{4}$$

Finite Difference Method (FDM) and Finite Element Method (FEM) are two methods, used to solve differential equations on a grid, often referred to as a mesh. Finite Volume Method (FVM) is a method based on dividing a region into cells and applying conservation laws between each cell. Conservation of mass, energy and momentum are typically utilized. FVM is very well suited for flow problems and is extensively used in Computational Fluid Dynamics (CFD). There exists several methods that can be used to computationally model and assess such behavior of mass transport, and as computers have gotten faster and faster over years, computational fluid dynamics (CFD) has become an important engineering field.

There exists several frameworks and tools to do low-level Computational Fluid Dynamics, abbreviated CFD, like the CFD module from COMSOL which can be combined with their multiphysics platform. This can be used to explore a very wide range of engineering problems as exemplified in COMSOL Inc. (2021*a*) and COMSOL Inc. (2021*b*). COMSOL is one of several commercial softwares within this field, but can be expensive to obtain for usage outside of academia and education. OpenFOAM on the other hand is free and open-source and have many of the same possibilities for conducting CFD analyses (OpenFOAM 2021). For this thesis' case, a simple model that could be used to illustrate the bulk swelling behaviour observed in real life shale environments, was sought. The hypothesis was that a simpler discrete element model could reconstruct observed bulk properties without having to detail down to clay particle scale. CFD was therefore considered to be less suitable for clay swelling and clay mass transportation at this stage.

2.3 Sample randomness and percolation theory

In Section 2.1, an elaboration of which parameters is believed to influence clay swelling were presented. As mentioned there, the availability of space to swell into is believed to be a influencing factor. Therefore, an attempt to understand how both total porosity and effective porosity will influence the swelling results have been undertaken.

From numerous samples from oil wells that have been collected, x-rayed and studied over the years, a relatively good understanding of the Norwegian continental shelf and its rock compositions have been obtained. Likewise, extensive seismic activity and geological surveys have been undertaken, which also has given good understanding of larger formations and possible oil and gas fields, yielding an overall good understanding of the geology. When building a computational model, it should be attempted to use the geological knowledge to create realistic, although randomly generated test samples that are usable for computational simulation studies. Based on known bedrock composition, including ratios of different particle types, one can randomly draw out particle by particle according to the chosen particle distribution. These particles will then create

a rock sample either in 2D or 3D. Since each sample will be composed of a very large number of particles, the ratio of each particle type will approximate the exact model input ratio as the number of particles increases, in accordance with the *law of large numbers*. Additionally, one will see that particle of the same type will form clusters where neighboring particles of the same type can be considered as *interconnected*. Since these clusters consisting of interconnected particles also will be randomly created, they will also be of varying sizes. However, how many and how large such clusters will be is subject to larger variance between different samples, as there is relatively speaking fewer large clusters compared to total number of particles. Through known percolation theory results, it is possible to get the expected size of such clusters for different geometries. Often it is difficult to calculate an analytical expected cluster size, but there have been done many numerical studies on the matter. The reason why the size of clusters are of interests for this thesis, is that inter-connectivity between clay particles is believed to influence clay swelling and that bulk behavior is connected to group swelling inside cluster. Since the main aim of this thesis is to contribute with enhanced knowledge about clay swelling and modelling of such, percolation theory can therefore come in handy. Correspondingly, percolation theory can be used to estimate the expected clay cluster size around a well, as this can be further used to predict and assess the potential of applying clay swelling techniques as part of the wellbore anchoring process.

By choosing a random starting point in the grid and picking the particle in this position as the starting particle, one could try calculate the expected size of the cluster that this starting particle is part of. However, one would soon realize that is quite difficult to do analytically. This complicates the process, and no known closed form expressions of the expected size of a cluster in infinite grid exists in two or three dimensions, at least not that the writer of this thesis is aware of. Regardless of the lack of analytical expressions, cluster analysis is an area of interest within several science and engineering fields, and an abundant number of numerical experiments have been studied. This field of study is called *Percolation theory*, and for instance the compendium by Malthe-Sørenssen (2020) provides and introduction to the topic. On the more theoretical side, Kesten (1982) provides a solid mathematical foundation of the topic. One fundamental concept is that of a *percolating cluster*. As the density of particular particle type increases, so will the largest cluster size increase as well. At a certain point, this cluster size will increase greatly and the single percolating cluster will span the entire grid. In the case of an infinite grid, this will be a discrete jump at a value known as the *critical percolation threshold*. However, for a finite grid, this sudden increase will be much smoother than a discrete jump.

A practical use case for percolation theory with this thesis' model, is that it would be possible to find out when a clay cluster spans the entire sample. Another point would be when there is a channel[4] that spans the entire sample, allowing a clay particle to eventually move through the entire sample instead of being isolated in one finite sub-region of the sample.

Based on the material mentioned above, and in connection with the aim of this thesis, the following parameters and their implications for clay swelling is perceived to be of interest:

- Expected size of the largest clay cluster
- Critical Percolation Threshold and how this corresponds to the clay particle ratio in a specific sample

2.4 About the Julia programming language

For this thesis, I have chosen to use the Julia programming language. Julia is perhaps not as famous as other programming languages often used in computational physics, e.g., such as C/C++, Python or Fortran.

[4] How *channel* is defined for this thesis will be explained in Section 3.3.1.

Julia has the advantage of allowing both competitive computational performance while being a high level and modern language. More about Julia and its possibilities are outlined in this Section.

Julia is a relatively new, free and open source programming language designed for numerical computing (Bezanson et al. 2017). Julia's goals have been to be both a high-performance language, while still being high level and readable.

The creators of Julia talk about the two-language problem. Performance critical code has be to written in low level languages like C, C++ and Fortran, while data munging, data analysis and visualizations are easier done in languages like Python or R. While there exists packages and methods for Python that makes the language quite competitive in speed when used correctly, like numpy and cython, it comes at a significant reduction in flexibility and simplicity. Additionally, these packages are themselves often written in a low level language, C in numpy's case (Harris et al. 2020). For many cases that will not be a problem, but if one finds himself in a position where the specific package is not sufficient to achieved wanted programming, it can be very difficult to extend the functionality or to create custom code.

Julia on the other hand, has a syntax that almost feels like a scripting language, but has the performance to rival C and Fortran. It is written purely in Julia and has great introspection capabilities, which makes it easy to see how things work *under the hood*, and this makes it much easier to change things and to extend the native capabilities.

Julia is not an object-oriented language as one would know it from languages like C++ or Python. In Julia one cannot define functions or methods inside classes. Instead Julia's key paradigm is *multiple dispatch*. At run time, the dispatch process will select which potentially overloaded method to call based on the types of all inputs of the method. This, combined with a robust type system, facilitates writing both code that is generic and modular, but that is also highly performant. As an example, in Julia one can have a **function f**, seen in the code-block below, which takes two numbers as inputs and then does something with. Let us assume that in the special case where both numbers are integers, there exist an optimized solution that significantly improves performance.

```
function f(x::Number, y::Number)
    #Does something
end

function f(x::Integer, y::Integer)
    #Does something faster
end
```

In Julia one can then just define both of these function with the same name but with different input types. At run-time one can simply call the **function f** and the dispatcher will call the appropriate method based on the actual types of x and y. The user of the function would not need to be concerned about this, since the dispatcher will do the "correct job" without needing explicit definitions or any extra information. In practice, it is not necessary to create overloaded methods for all input types. The compiler already knows the types of function arguments and local variables, and will on its own build specialized and high-performant function for every input type combinations. As Julia is not a compiled language, a small downside is that the first time Julia encounters a specific combination of types for a function, it will take some time to compile that version. In longer running simulations or in tight-loops where the same functions are called over and over again this is completely negligible, but the compilation time can be noticed when running shorter script or just starting up a new session.

To draw the most performance out of Julia there are some guidelines to follow. The most important is to have functions and data structures that locally have defined types. For example having a function that will always return 64-bit float and never a null or anything else will allow the Julia compiler to take advantage of many optimizations. Mainçon (2021) explains more about these aspects, and about Julia's type system, which is important to consider to be able to write both stable and efficient Julia code and programs.

Julia also has a very convenient and simple way to multi-thread or parallelize code. The pattern below has been used extensively in cases where a large number of samples have been studied, and where the results of the individual runs are average to create an aggregated result. By just adding a small macro in front of a for-loop, Julia will automatically spread the loop over different threads, which will greatly reduce the time needed. This is especially valuable for long-running tasks.

```
Threads.@threads for j in 1:number_of_samples
    #Some independent long-running calculation
end
```

While still relatively new, Julia has a fast-growing community and continuously improving ecosystem of tooling and packages. Some packages like *DifferentialEquations.jl* (Rackauckas & Nie 2017), for solving of differential equations and *JuMP* (Dunning et al. 2017), which is a solver-agnostic domain-specific language for mathematical optimization, are already just as good as equivalent packages in other languages. Julia might also be one of the languages to best support automatic differentiation which implies it might become on of the core languages for doing machine learning. That being said, as expected from a new language there are still missing or immature packages and Julia obviously cannot compete in all domains with more established languages.

3 Model

To investigate clay swelling in this thesis a model has been built based on the principles of the discrete element method. The reason behind choosing such a model for this work was provided in the previous chapter, in Chapter 2, along with a theoretical background about the aspects deemed relevant in the context of modelling and simulation of clay swelling. This chapter will present the different aspects of the clay swelling model that have been made during the work with this thesis. The basic configurations and system geometry used in the model will be described in Section 3.1. More advanced features are detailed in subsequent sections, Section 3.2 and Section 3.3. Section 3.2 describe swelling, grid interactions, and how these are modeled. Section 3.3 details how simulation of mass transport has been implemented. Lastly, a quick overview of the code structure will be provided in Section 3.4.

3.1 Model setup and sample geometries

To model this problem a discrete element method has been used. The geometry that has been chosen is an annulus. Computational simulations can be conducted both in two dimensions for a cross-section on an annulus, or in three dimension to capture the wellbore geometry. The 2 dimensional simulation will naturally have lower computational demands. Other shapes could also easily be investigated by tweaking the grid. For this study this shape was specifically chosen because of its close resemblance to the rock formation surrounding a wellbore.

3.1.1 Grid and annulus

The first step when initializing a sample geometry is to choose the grid size (in x-,y- and z-direction), and the dimensions of the annulus given by an outer radius R_{outer} and inner radius R_{inner}. The height of the annulus will normally be chosen to be equal to the z-dimension, and the x- and y-dimensions are normally equal due to the symmetric shape of the annulus. The particles are thus placed within the outer radius R_{outer} and inner radius R_{inner} of the annulus. This is illustrated by Figure 6.

Within the annulus each grid point will either be assigned as empty, to simulate porosity, with a probability of ρ_p or as quartz with a probability of ρ_q. The rest of the particles will be clay with a probability ρ_c. Since all points must be either empty, quartz or clay, it means that the sum of the particles ratios ρ_p, ρ_q and ρ_c will equal 1. Figure 7 shows a zoomed-in illustration of the grid structure with three different types of particles placed on the grid intersections. As this illustration only shows a small part of the full structure, the annulus layout is not visible. Only the part of the grid that falls within the annulus shaped sample can be populated by particles. Both the inner and outer radii are treated as walls, and the areas outside these walls can neither contain particles, nor can there be particles swelling into these areas. For the computational study the grid will be much larger. For a comprehensive illustration of the annulus grid for a more realistically sized grid, the reader is referred to Figure 19 in the *Computational Results*-chapter, under Section 4.1.

Every particle, quartz and clay, will be treated like discs and will be initialized with a radius r_i. To begin with that will be drawn from a uniform distribution from 0.1 to 0.9 for both. The cross-section area of the annulus is given by Equation (5).

$$A_{\text{ann}} = \pi(R_{\text{outer}}^2 - R_{\text{inner}}^2) \tag{5}$$

The volume of the annulus is given by Equation (6).

$$V_{\text{ann}} = \pi H(R_{\text{outer}}^2 - R_{\text{inner}}^2) \tag{6}$$

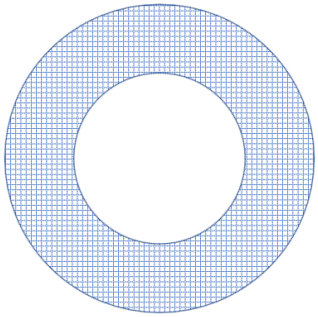


Figure 6: Illustration of an annulus shaped grid structure that will be used in this work.

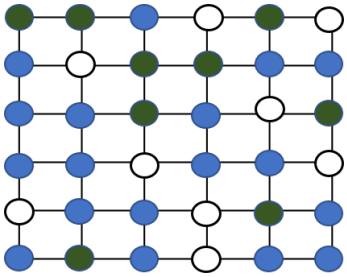


Figure 7: Illustration of the grid and different particle types.

A radius scaling factor is introduced to keep the radii of the particles and thus the swelling probability independent of the choice of grid size N. The factor is given by Equation (7).

$$\gamma = \frac{R_{\text{outer}}}{N}. \tag{7}$$

The total area of the particles inside the annulus will be then be as given by Equation (8).

$$A_{\text{particles}} = \pi \sum_i (\gamma r_i)^2 \tag{8}$$

3.1.2 Cluster definitions

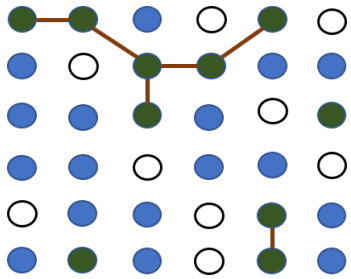


Figure 8: Illustration of clay clusters where the clay particles are marked in green and where neighbor clay particles are connected with a brown line. The lattice also contains another particle type (blue markers) as well as empty spaces (white markers).

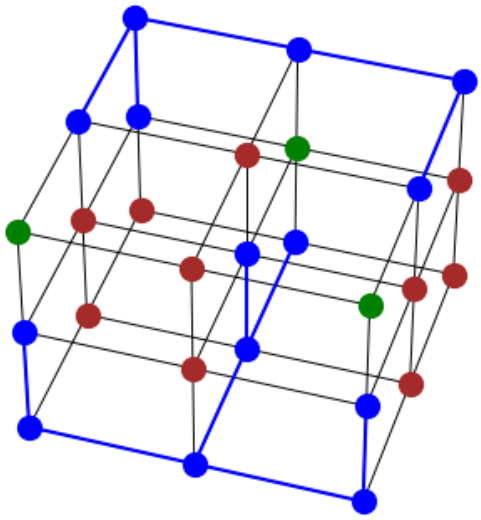
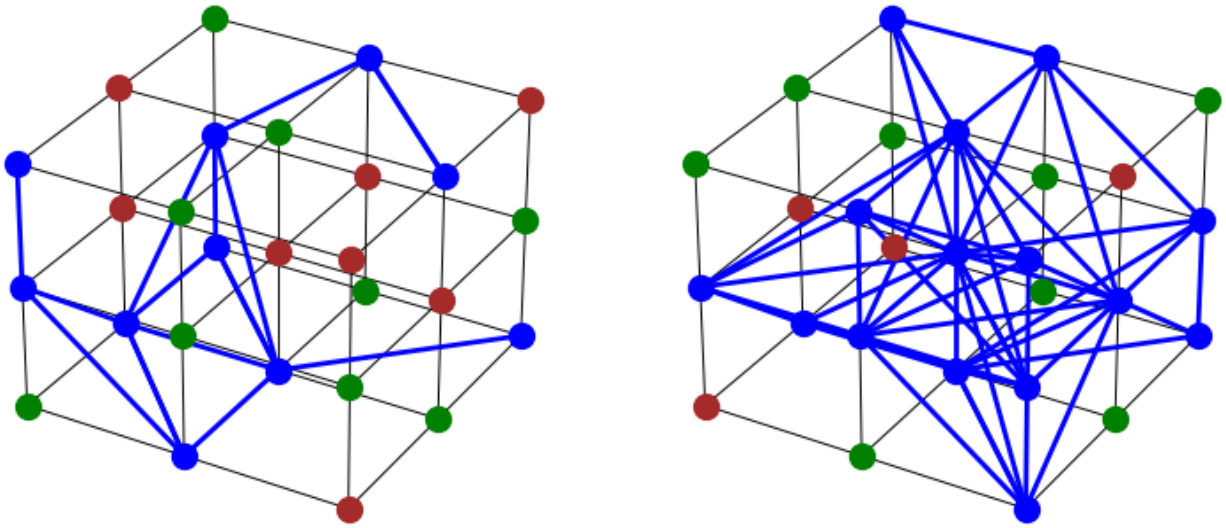


Figure 9: Illustration of 3D clay clusters where only the most immediate neighbors are considered to be connected in a cluster, i.e., that a particle at max can have 6 connections.

For this exercise we will consider the grid as a network. A network consists of nodes and edges between the nodes. In our case the grid points will be the nodes. Only neighboring clay particles will have edges between them, while quartz particles and empty grid points will have no edges. An illustration of this can be seen in Figure 8. We have considered the 8 nearest particles as neighbors of a given particle. This is known as a Moore Neighborhood (Malarz & Galam 2005). A simpler setup on the square grid is the von Neumann's neighborhood, where you only consider the points north, south, west and east of the particle. Both of these cases have been extensively studied, and we will compare our results in both cases. The equivalent neighborhoods in 3D will be the 6-neighbor case where one also looks directly above and directly below, or the 26-neighbor case where one looks at all adjacent particles as neighbors. Figure 9 shows the 6-neighbor case in 3d for a 3x3x3 grid, while 10 shows the 26-neighbor case. Notice that in the 26-neighbor case, the number grows dramatically as the number of clay particles increases.

3.1.3 Cluster labeling algorithm

A common algorithm to label clusters on a grid is a 2-pass algorithm known as the Hoshen-Kopelman algorithm (Hoshen & Kopelman 1976), though there exists quite a few others due to the many use cases.



(a) (b)

Figure 10: Examples of 3x3x3 grids where all 26 surrounding neighboring particles are considered to be possible connection points. A relatively low number particles in *(a)* gives comparatively fewer connections than in *(b)*

The Hoshen-Kopelman algorithm is simple to implement and fast, thus it was the one that was chosen to be implemented.

It consists of two passes over the grid, row-wise top to bottom. In the first pass for each clay particle we look at its previous neighbors to see if they have been assigned a label yet. In the 4-neighbor case that will be the particle above and to the left. We know the other particles do not have labels since they have not been processed yet. If none of the neighbors have a label, i.e. are not clay particles, we assign the current particle a new label. If only one of the neighbors have a label we assign the same label to the current particle since they will be part of the same cluster. If more than one neighbors have labels and they are different we will need to connect the clusters. First we choose the lowest label and assign it to the current particle and then we note that it and and the other neighbors are part of the same cluster. On the second pass of the we rename the labels of the particles in such a way that particles of the same cluster have the same label.

Technically one can say that the labels that are part of the same cluster, are in the same equivalence class. Finding the label on the second pass will then be to find representative member of that equivalence class. To efficiently create these equivalence classes and to find the representative member one has the disjoint-set data structure, also known as a union-find structure (Leeuwen 1990), which implements these two operations, union and find. To implement this the *IntDisjointSets* from the package *DataStructures.jl*[5]

[5]https://github.com/JuliaCollections/DataStructures.jl [Online; accessed 17-Dec 2020]

3.2 Swelling dynamics

One main part of the modelling framework is the the inclusion of swelling behavior. Clay particles have a propensity to swell many times their size when in contact with water. In the subsequent section the swelling effects implementation will be explained in more detail, and in Section 3.2.2 additional aspects such as neighborhood interactions will be presented.

3.2.1 Swelling effect

Between the outer and inner radius of the annulus there is a stress-difference σ and a temperature T. Both of these parameters are believed to have an impact on how the swelling process of clay occurs, and has been motivated by Rybacki et al. (2017). Although Rybacki et al. (2017) mainly considers the creep of shale, it seems likely that both the stress-difference σ and a temperature T also would influence a clay particle's likelihood to swell. Additionally it seems reasonable to assume that a particle cannot swell towards infinity, meaning that at a certain point the swelling should stop. Based on this we will test of a swelling probability where a clay particle i will swell with a probability P_i^S, as stated by Equation (9). The probability of swelling will thus dependent on both the particle's radius r_i, the temperature T and the stress-difference σ. Increasing temperature and pressure will increase the probability to swell, and for very large values of the mentioned parameters the swelling probability will approximate 1. At the same, it is worth mentioning that larger clay particles will have a decreasing probability to swell further.

$$P_i^S = \exp\left(\frac{-r_i^2}{\sigma T}\right) \tag{9}$$

There will be an upper limit on how many times each clay particle can swell, denoted as $\eta_{\max}$. The amount of times a particle i has swelled will be denoted with η_i. Every time a particle swells the radius increases by a factor α. Both α and $\eta_{\max}$ are parameters and are kept constant for a given simulation. Different types of clay have different swelling potential, and these parameters are included to account for that. The iteration is continued for a given number of iterations unless it is stopped early. Reasons for that may be because $A_{\text{particles}} \geq A_{\text{ann}}$ or all clay particles have swelled the maximum number of times. Neither the empty grid points nor the quartz particles will swell. Quartz particles are inherently relatively stable particles, and will for this entire work be considered to be fully inert. When running the simulation clay and empty grid points will therefore merely be skipped. Realistically quantifying all the swelling parameters will require real-life data and experimental research. Presence of water is a prerequisite for clay swelling. In this thesis, water is presumed to be present and the lack of water will therefore not be impacting the swelling probability in this modelling study.

3.2.2 Grid interactions

From clay theory, it is known that clay particles will be influenced by nearby particles. This influence can be caused by multiple sources. One of the more apparent sources of grid interaction is coming from the distribution of particles itself. For a particular particle, the composition and type of neighboring particles is of especially high importance when it comes to investigating grid interactions. For instance, swellable particles such as clay may be heavily influenced by the space occupation of neighboring particles, as this will dictate how much they themselves can occupy. How much empty spaces and how they are distributed and connected, referred to as porosity and permeability, will have large effects on water or other liquids. At the moment this is not in our model, but it is of interest to set up a framework where these parameters can be modelled. Just as large particles in the neighborhood may hinder swelling, emptiness may facilitate it. Some fundamental forces, such as electrostatic forces can also influence clay formation and swelling. As

one sometimes observes clay masses consisting of charged layers and water formation between the layers, it can be interesting to consider the electrostatic forces occurs between the charged particles, and try to investigate these forces effects on the clay formation and swelling.

As a first step towards creating a comprehensive model, I have started by implementing some nearest-neighbor interactions. For this thesis, I will mostly consider only the space related interactions.

In this section, we will consider a two dimensional grid with a particle's nearest-neighbors to be the eight adjoining grid spaces around the considered particle. This is illustrated in Figure 11 where the red particle has eight nearest neighbors, seven of whom are particles marked in pink. On the eighth position, on the bottom left, there is empty space which here is illustrated with a white marker.

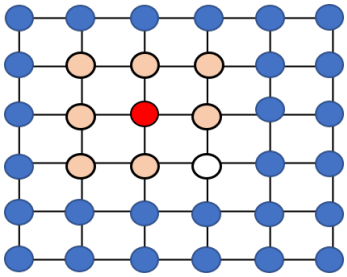


Figure 11: Illustration of the eight nearest-neighbors that a particle will have. The red particle have seven pink neighbor particles and one empty space as a neighbor, where the empty space is illustrated with the white marker.

As the neighboring particles swell, particle i will have less space available to do it, and this should be accounted for in the model. In this model, one is looking at the average amount of swelling compared to the max swell amount $\eta_{\max}$ for the neighboring points. As that increases the chance to swell will decrease. This is done by adjusting the swelling probability by a factor β_{nb} as indicated in Equation (10). n_{nb} is here the number of neighboring grid points.

$$\beta_{\text{nb}} = 1 - \frac{1}{\text{n}_{\text{nb}}} \sum_{i \in \text{nb}} \frac{\eta_i}{\eta_{\max}} \tag{10}$$

On the other hand, if there are empty grid points close to a particle, the probability will increase by the factor β_{por}, as given by Equation (11). Since empty spaces often are referred to as *pores* in geological terminology, I have denoted this adjustment factor with the subscript por. n_{nb} is still the number of neighboring particles, and n_{pr} is the number of neighboring spaces which are empty.

$$\beta_{\text{por}} = 1 + \frac{\text{n}_{\text{pr}}}{\text{n}_{\text{nb}}} \tag{11}$$

The final swelling probability $P_i^{S,\text{final}}$ is then given by the product of the terms given by Equations (9), (10) and 11, as shown in Equation (12).

$$P_i^{S,\text{final}} = \beta_{\text{nb}} \cdot \beta_{\text{por}} \cdot P_i^S \tag{12}$$

3.3 Mass transport simulation

A borehole is not a static system, and it is composed of different materials, some of which are liquid and some that are solid. All solid rock bodies are subjected to pressure, stresses and instabilities. A full reconstruction of borehole systems, with forces and constituents, is out of this thesis' scope. To begin modeling of a dynamic system, the clay particles will now be freed from their static grid positions. The clay particles will then be able to move through the porous rock structure of the annulus, instead of merely swelling in their static positions. The mass transport part of the model will treat both clay swelling and movement simultaneously.

3.3.1 Permeable channels

The set of quartz particles will as a whole define the immobile rock matrix of the annulus. The clay particles will be able to move to adjacent empty grid point, occupying it. The grid point it leaves behind will then become empty. These empty grid points will be referred to as *pores*. One can consider all the pores or the grid points that contain a clay particle as the *permeable part* of the annulus. Any given clay particle will be limited to move within a finite part of the grid due to blockage from the immobile quartz particles. Such an enclosed region of connected clay particles and pores will be referred to as a *channel*. As can be seen in Figure 12, a channel consists of interconnected clay and pores, respectively shown by the green and white circles. These channels are enclosed by immobile and impermeable quartz particles. In this example, diagonal points are not considered to be interconnected. The two largest channels in the illustration have been marked in red in Figure 12.

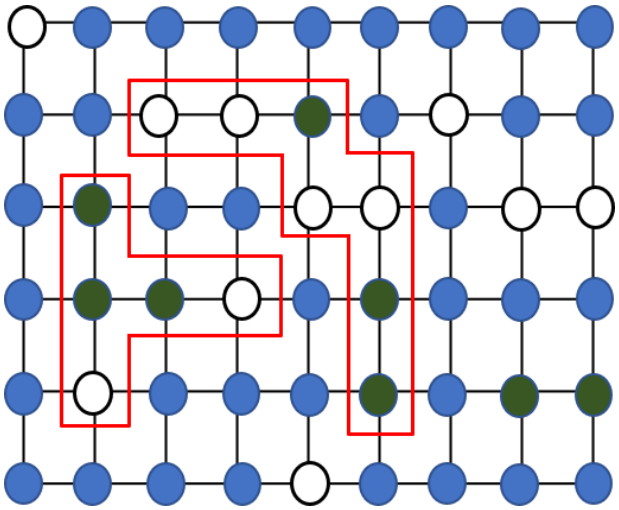


Figure 12: Illustrative example of the definition of *channels*

For certain density distributions, especially for cases with low quartz densities, a single channel may become very large. Such large channels can extend themselves throughout the entire annulus. Density distribution and expected cluster sizes were discussed in Section 3.1.2, and the expected channel size will similarly be given by percolation theory, with the small exception that the aggregate density of the clay particles and pores must be used.

For this thesis, the quartz particles are treated to be fully inert and immobile. This has the implication that the clay particles will never be able to escape the channel they were initially placed inside, because their impermeable outer boundary of quartz will remain fully stationary. To treat quarts as fully inert and

impermeable may not be accurate in all settings, and easing this restriction is something that could be of interest in a later projects. Another aspect that could be of interest is to investigate how clay and other particles might fall from the inner radius of the annulus and into the drilling hole.

3.3.2 Random and biased random walks

Movement will be simulated in discrete time-steps in this model. At each time-step, a clay particle will have a certain probability to move to an adjacent pore. The properties of the grid points will then be swapped. Thus, the clay particle will have moved into another empty spot in the channel, leaving its previous position empty. This newly empty pore will then become accessible for other migrating clay particles in the next time-step. It should be noted that two clay particles may not occupy the same grid point at the same time.

Figure 13a shows the five possible moves that can be undertaken by the four clay particles that have adjacent empty grid points. In the next time-step, some of the particles from the first time-step have moved, and swapped places with the previously empty grid points as can be seen in Figure 13b. In the latter figure, the new possible movements have also been indicated. One interesting aspect, observed in Figure 13b, is that three different clay particles have the possibility to move into the same empty grid point. As mentioned, only one particle can occupy a grid point at once, hence when sequentially running processing the swaps, this grid point might become unavailable before the second and third clay particle get their movement drawn out. The model will therefore always check to see which adjacent grid points are still available before deciding if it is possible to move there.

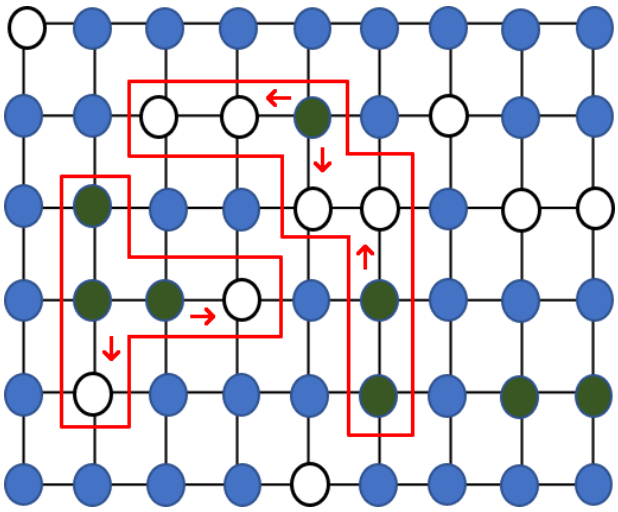
As illustrated in Figure 13, the mass transport modelling is behaving like a *random walk* amongst many obstacles on a grid. Figure 14 has been included to ease the explanation of the simple random walk procedure that has been implemented in the code. The clay particle in the middle, denoted by i, has 4 von Neumann neighbors and 8 Moore neighbors. The clay particle i can either remain in position i, or move into a neighbor grid point that is empty, i.e., j, k, m or n when using the Moore neighborhood definition, and only k or m for the von Neumann setting. Equation (13) show the set of empty neighbors utilizing Moore definition. Equation (14) contains the discrete probability distribution, denoted as Ω^M, and where $Pr(i \to j)$ is the probability of moving from i to j. It should be noted that the discrete choices do not need to have the same probability. Similarly, Equations (15) and (16) show the same for the von Neumann setting.

$$\mathcal{N}_i^M \in \{j, k, m, n\} \tag{13}$$

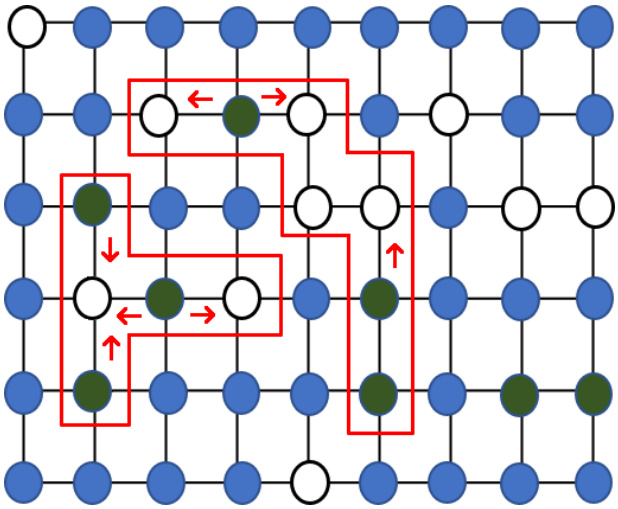
$$\Omega^M \in \{Pr(i \to i),\ Pr(i \to j),\ Pr(i \to k),\ Pr(i \to m),\ Pr(i \to n)\} \tag{14}$$

$$\mathcal{N}_i^{VN} \in \{k, m\} \tag{15}$$

$$\Omega^{VN} \in \{Pr(i \to i),\ Pr(i \to k),\ Pr(i \to m)\} \tag{16}$$



(a) In time-step number 1, four clay particles can undertake in total five moves



(b) In time-step number 2, five clay particles can undertake in total seven moves, but there are only five empty points to move into.

Figure 13: Illustration of clay particle movement during two time-steps

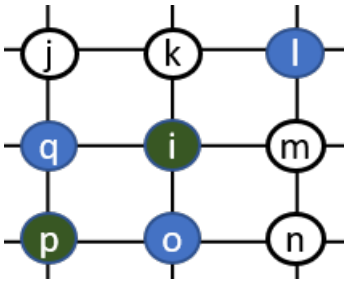


Figure 14: In the simple random walk case, the clay particle at position i has an equal probability to move into any of its adjacent empty grid points j, k, m and n.

The $Pr(i \to j)$ can be defined in different ways depending on the chosen application one would like to explore. In this thesis, I have included a basic case, where all moves are considered to have equal probability, and a case which includes a radially inward force towards the center of the annulus. The latter one therefore includes some form of bias in the movement probability distribution.

The probability to stay in the same spot, $\Pr(i \to i)$, will in the model be independent of number of neighbors and will be used as a simulation parameter and denoted as λ.

$$\left(\sum_{x \in \mathcal{N}_i} Pr(i \to x) \right) + Pr(i \to i) = 1 \tag{17}$$

$$Pr(i \to i) = \lambda \tag{18}$$

For the case where all movements have the same probability, the probability to move to one of them will solely depend on the number of neighboring empty spots $|\mathcal{N}_i|$.

$$Pr(i \to x) = \frac{1 - \lambda}{|\mathcal{N}_i|} \tag{19}$$

In the case with the radially inward force, the probability of moving in certain directions, those aligned with the force, will be increased at the expense of movements in the opposite directions of the force. This will instead create a *biased random walk*. The biased random walk should follow certain properties.

Clearly the probability of moving in the direction of an arbitrary point x should depend on the angle between the movement $\vec{v}_{ix}$ and the force $\vec{F}$. Figure 15 show an example of this, where particle i is subjected to the inward force $\vec{F}$, and has the direction vector $\vec{v}_{ik}$ to the empty grid point k. The probability of moving towards the empty grid point k will should depend on the dot product of $\vec{F}$ and $\vec{v}_{ik}$. Moving in the direction of the force should be significantly more probable than against it.

The magnitude of the force should influence how much more likely the particle is to follow the direction of the force. Additionally, by having a non-zero probability of moving in the opposite direction of the force, one would reduce the chance of having particles get stuck in dead-ends. The latter could easily be the case if a probability of zero was given.

The previously defined σ, the stress-difference felt by the particles, will be used as the magnitude of the force, while $(-\vec{r})$ is the unit vector from the particle towards the center of the annulus. All particles are assumed to experience this force and the magnitude is constant throughout the annulus.

Two different schemes have been implemented. λ, the probability to stay, can be kept constant for regardless of how strong the force acting on the particle or it can be reduced under larger forces. The first case is the simplest as λ is kept constant.

To begin with the potential directions are weighted according to their alignment with the force and the magnitude of the force. An exponential function, as seen in Equation (20), was used so that the factors are equal without external forces and that the directions along the force are weighted disproportionately more at higher forces.

$$Pr^*(i \to x) = e^{\sigma \cdot (-\vec{r} \cdot \vec{v}_{ix})} \tag{20}$$

To create a probability vector, these factors should be scaled so that they plus the probability to stay in place sums to 1. The scaling factor γ is found by summing the factors and dividing by $1 - \lambda$, as shown i

Equation (21). Thus for the simple case, Equation (22) gives the probability for the clay particle to move from position i to a position x.

$$\gamma_i = \frac{1}{1-\lambda} \sum_{x \in \mathcal{N}_i} Pr^*(i \to k) \tag{21}$$

$$Pr(i \to x) = \frac{Pr^*(i \to x)}{\gamma_i} \tag{22}$$

One other approach, which potentially is more realistic, is that the probability to stay in the same grid point, i.e., not moving, is reduced for higher force levels. For this setting, a modified λ denoted as λ^*, is used in the calculation. Equation (23) shows how the λ from the first setting is changed proportionally with $e^{-\sigma}$ and Equation (24) show how that is use to calculate the modified scaling factor γ_i^*. It should be highlighted that this scaling could have been undertaken in multiple ways, and that proper calibration should be performed if this model should be used to study real-life cases or applications. Such calibration must then probably be based on lab test on rock samples and experience and datalogs from real-life well drilling. This results in the overall probability of moving from position i to x as given in Equation (25).

$$\lambda^* = \lambda e^{-\sigma} \tag{23}$$

$$\gamma_i^* = \frac{1}{1-\lambda^*} \sum_{x \in \mathcal{N}_i} Pr^*(i \to k) \tag{24}$$

$$Pr(i \to x) = \frac{Pr^*(i \to x)}{\gamma_i^*} \tag{25}$$

Lastly, it should be noted that not only do the two different settings both yield the same probability distributions when the external force is set to zero, but they become identical to the random walk. The behaviour can be seen in Figure 17. At higher force levels the directions along the force will dominate, but even at higher forces there is a small chance to go in another direction as well.

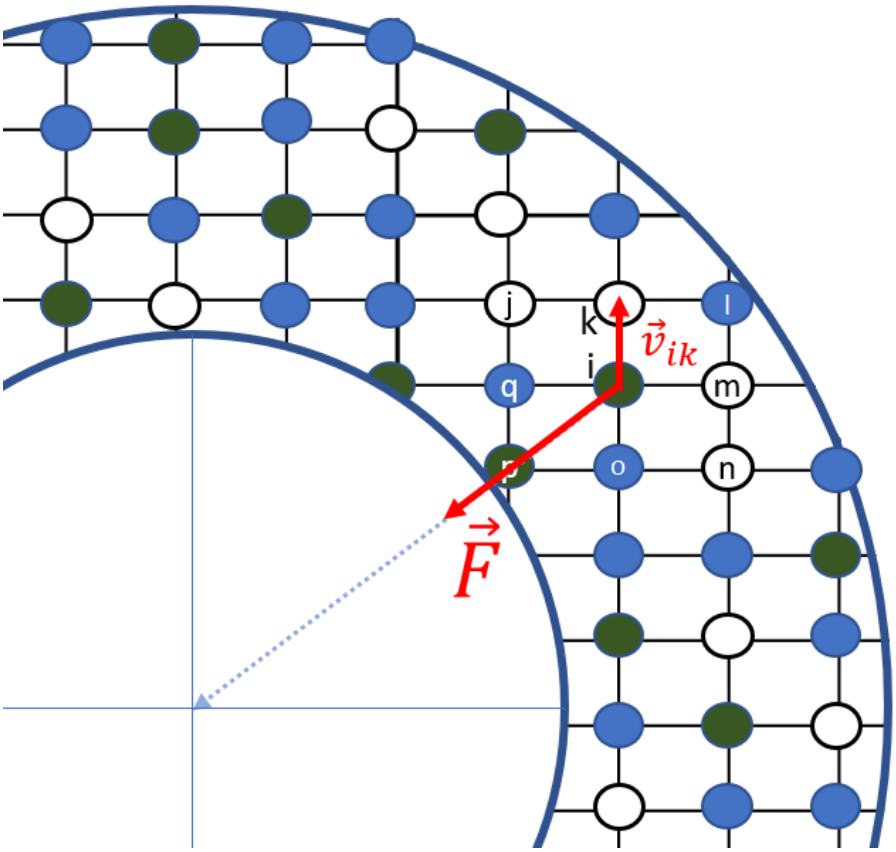


Figure 15: The direction of the force acting on the clay particle in position i will influence how likely it is for the particle to move into position k. The angle between the movement vector $\vec{v}_{ik}$ and the force $\vec{F}$ will be proportional to the likelihood of moving there.

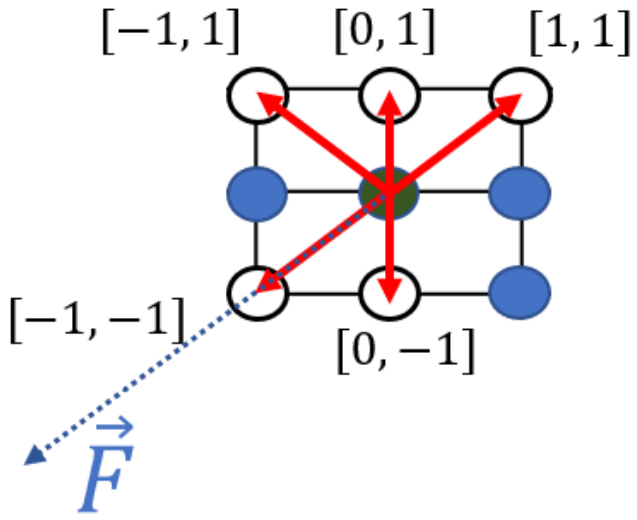


Figure 16: The particles are subjected to a force $\vec{F}$, the center clay particle can move along the five red direction vectors. This illustration is used as a starting point to illustrate the implemented biased random walk model.

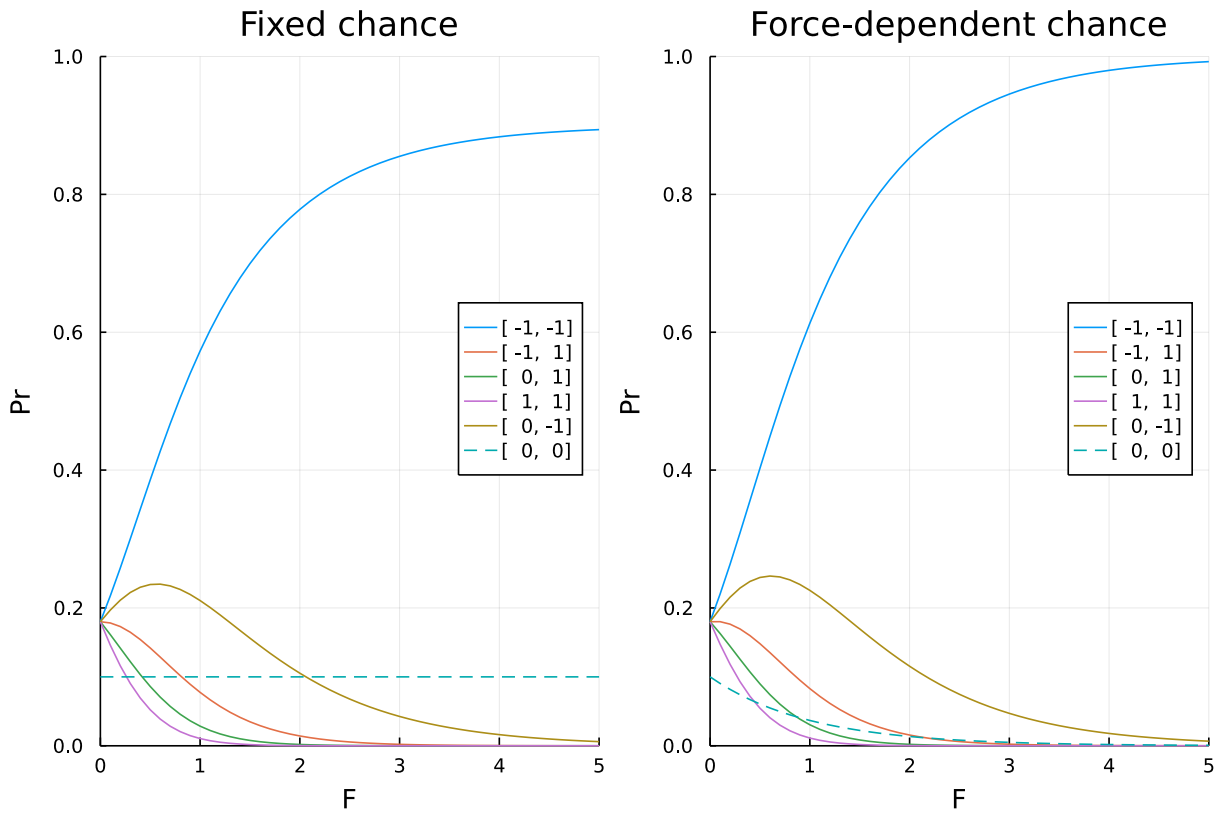


Figure 17: Example of the discrete probability distribution for biased random walk as a function of the magnitude of the force $\vec{F}$. The probabilities are given for the example in Figure 16

.

3.4 Code overview

A brief overview of the code structure is shown in Figure 18. The code is structured in a Julia *module*, which is the way to package code in separate units in Julia. That is defined in the *Swelling.jl*-file which also contains some basic definitions for particle types and simulation setup. The geometry of the problem is defined in the *Annulus.jl*-file, where the annulus type is defined, which contains the sample and its data. It leverages the grid structure from *Grid.jl*. The code for the grids was partially taken from the package Agents.jl (Datseris et al. 2021), due to their effective way of handling neighbors of a gridpoint. The file *HoshenKopelman.jl* includes the implementation of the Hoshen-Kopelman algorithm for our use case. Lastly, the files *Simulate.jl* and *Movement.jl* respectively handle the swelling of clay in the annulus and the mass transport in the annulus. As the code is structured as a module, it can be used as a package, and there are a series of files building on the Swelling-package to run the simulations and create the figures made for this thesis. All the files contained in the module are fully included in Appendix C for the interested reader.

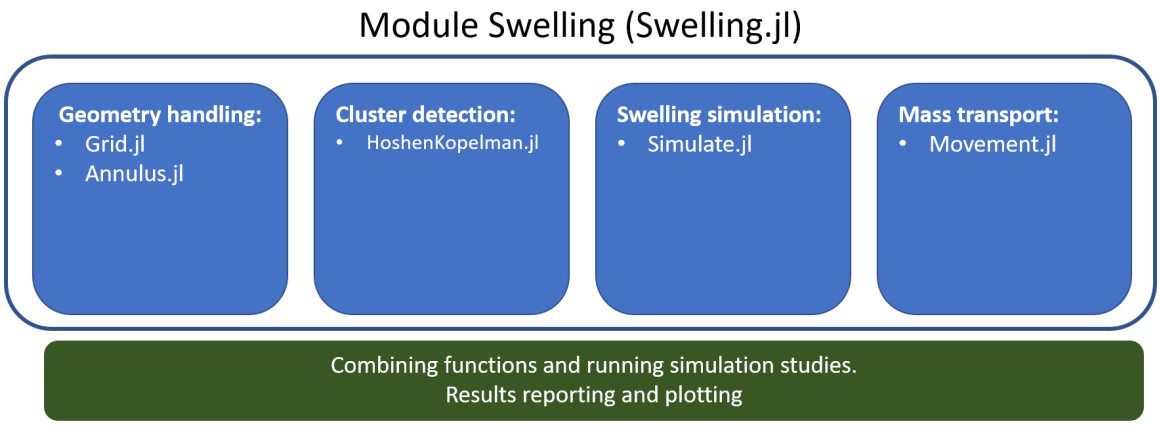


Figure 18: Overview of the model built in a Julia module

4 Computational Results

In this chapter the results from the computational study will be presented. First, in Section 4.1, I will be outlining the results of the sample generation procedure and show some figures of the samples. Afterwards, in Section 4.2 we will analyze the cluster formation and compare to known results. For both these sections I analyse randomly drawn samples, and study its static properties. In Section 4.3 the samples will be subjected to swelling simulations, where between each of the iterations all clay particles in the sample has a certain probability of swelling. In Section 2.2 the results of mass transport simulations are presented, where the clay particles will have a probability to migrate. In the last section of this computational study, Section 4.5 all systems are used to simulate how the swelling processes in a shale sample will affect it and specifically how swelling of clay in the matrix around a wellbore may be felt.

4.1 Sample generation

An annulus is initialized with the parameters given in Table 1, where ρ_p, ρ_q and ρ_c are the respective density ratios for empty spaces (pores), quartz and clay particles. N is the grid width, and the inner and outer radius values are given by R_{outer} and R_{inner}. The parameter sizes are chosen to be similar to the composition found in real-life shale, and are in accordance with Dayal & Mani (2017). The setup is then shown in Figure 19. The quartz particles are green, while the clay particles are gray. The particles types are chosen at random, and the particle radii are drawn from the uniform distribution from 0.1 to 0.9. As the particles are chosen at random no particular structure is apparent, although one can observe clusters spread around in the sample. A three-dimensional sample was also generated with the following properties; $\rho_p = 0.32$, $\rho_q = 0.40$ and $\rho_c = 0.28$, and can be seen in Figure 20. This particular annulus has relatively small dimensions with an outer radius of 20 particles, an inner radius of 5 particles and the height is 40 particles. This had to be chosen in order to create a plot without too much chaos. Additionally, only clay clusters above a minimum size of 70 particles and where at least one particle is connected to the inner annulus are shown in the plot to further reduce the visual complexity. There will be some small variations in the clay percentage, porosity and the total clay area/volume for different samples and we will expect slightly different behaviours for each run. However, since the number of particles are quite large, it is not expected to see any large differences between different samples on an aggregate level, although there might be some local variation on parts of the annulus, especially with respect to inter-connectivity between the clay particles. To show the small differences ten different two-dimensional samples where generated and information about clay content, porosity and clay area is shown in Table 2.

Table 1: The values of the parameters N, ρ_p, ρ_q, ρ_c, R_{outer} and R_{inner} for the sample generation

N	ρ_p	ρ_q	ρ_c	R_{outer}	R_{inner}
151	0.1	0.5	0.4	20	5

4.2 Cluster analysis

In this section the work done on cluster analysis will be presented. The grid size N will be increased to 301 in this section to reduce finite size effects and we will vary the clay density ρ_c. The porosity and quartz density are considered to be of less importance here, since only the static case is analysed and it is only looked at connected clay clusters. The inner and outer radii are the same as was given in Table 1 in the previous section. For every sample a label will be given to each cluster, by applying the Hoshen-Kopelman algorithm as outlined in Section 3.1.3.

As mentioned in previously, clay clusters above a certain minimum size that are connected to the inner circle

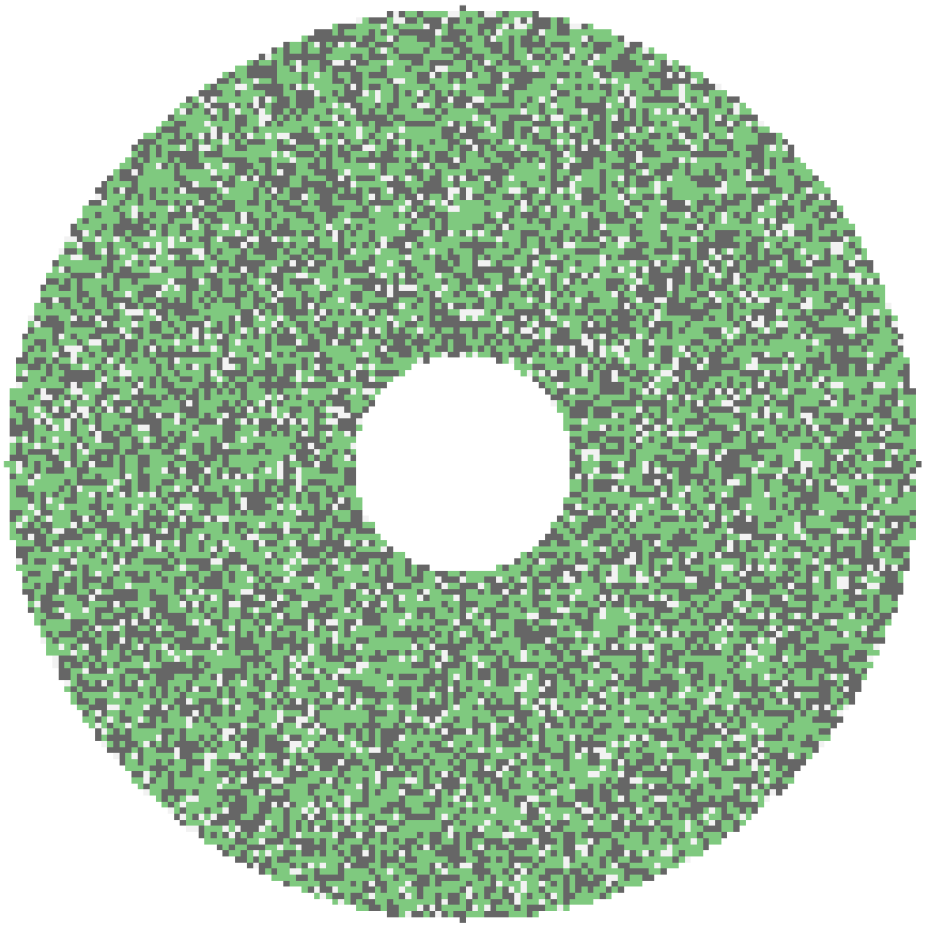


Figure 19: A generated sample with $\rho_p = 0.1$, $\rho_q = 0.5$ and $\rho_c = 0.4$. The green particles represent quartz while the grey ones represent clay. The empty grid points are left white.

of the annulus is considered to be of special interest. The clay particles connected to the inner part will push into the inner annulus as they swell and if their neighbors are swelling as well we will have a cascading effect. This hypothesis has motivated the inclusion analysis of clay clusters which are in contact with the inner circle of the annulus. Figure 21 displays all clusters connected to the inner circle of the annulus and which have a cluster size of more than 10 interconnected particles. The von Neumann neighborhood, where each particle has four neighbors, was used here. At lower densities of clay there are a few clusters, but the ones that are there do not propagate far into the sample. As the density increases the numbers of clusters and cluster sizes grow steadily up to a certain point, at which few or a single large cluster appears. This can then be considered as a percolating cluster, which means that we have the passed a critical clay density. Malarz & Galam (2005) gives the percolation threshold for the von Neumann case as $p_c = 0.592$. The percolation threshold is defined in the limit towards infinite grid size. Our grid is quite limited, as such it is natural that we do not see one single percolating cluster in our sample at $\rho_c = 0.60$ even though it has then passed the threshold. At the slightly higher density of $\rho_c = 0.62$, the percolating cluster is however very clear.

Figure 22 shows the size of the largest clay cluster divided by the total number of clay particles for densities of clay around the percolation thresholds. This is shown both in two dimensions and three dimensions

Table 2: The clay percentage, porosity and the total area of the clay particles for 10 different samples.

Clay percentage [%]	Porosity [%]	Total clay area
39.47	9.68	109.07
39.95	10.11	112.17
40.02	9.78	110.66
40.22	10.08	111.74
39.93	9.48	110.64
39.77	10.17	111.66
40.15	10.14	111.7
40.62	9.93	114.73
40.21	9.72	113.33
39.45	10.44	110.3

with the defined neighbor configurations. The numbers displayed in the plots are the average values for the largest cluster size computed based on 70 different samples. Around the critical value, i.e. around the percolation threshold, there is a significant increase in the largest cluster size. For the largest neighbor case, each particle has access to a much larger neighborhood, and it is thus expected that the cluster sizes will be larger for comparable clay densities. With more neighbors per particle, the percolation threshold must therefore necessarily be lower for the 8-neighbor case. The percolation threshold is in two dimensions given by Malarz & Galam (2005) as $p_c = 0.407$ for the 8-neighbor case and for the 4-neighbor case this value is $p_c = 0.592$. In 3 dimensions the 6-neighbor threshold is $p_c = 0.312$ and the 26 neighbor is $p_c = 0.098$, both given by Kurzawski & Malarz (2012). For larger and more continuous grid geometries, the rapid increase around the percolation threshold would have been even more prominent. At an infinite grid the plots shape should be a step function. For the annulus shaped grid with the grid sizes used here, the maximum cluster sizes increase gradually at first, as seen in Figure 22. This gradual increase continues until there is a rapid increase in cluster size at a density sightly higher than the theoretical percolation values. This can be explained by the limited grid size and possibly also the middle hole in the annulus shape.

Some additional samples were generated randomly and used to make the plots seen in Figure 23. Here the average number of clusters are shown on the y-axes and the cluster size on the x-axis, thus showing the cluster size distribution. This has been done for the three clay densities (0.38, 0.40 and 0.42) around the percolation threshold. Each subplot is based on the average of 750 different samples. A line was fitted on the first half of the x-axis to show the exponential decay around the threshold. There is quite a lot of noise around the upper end of the cluster size. To remedy that one would have to run many more iterations but that would have taken a long time.

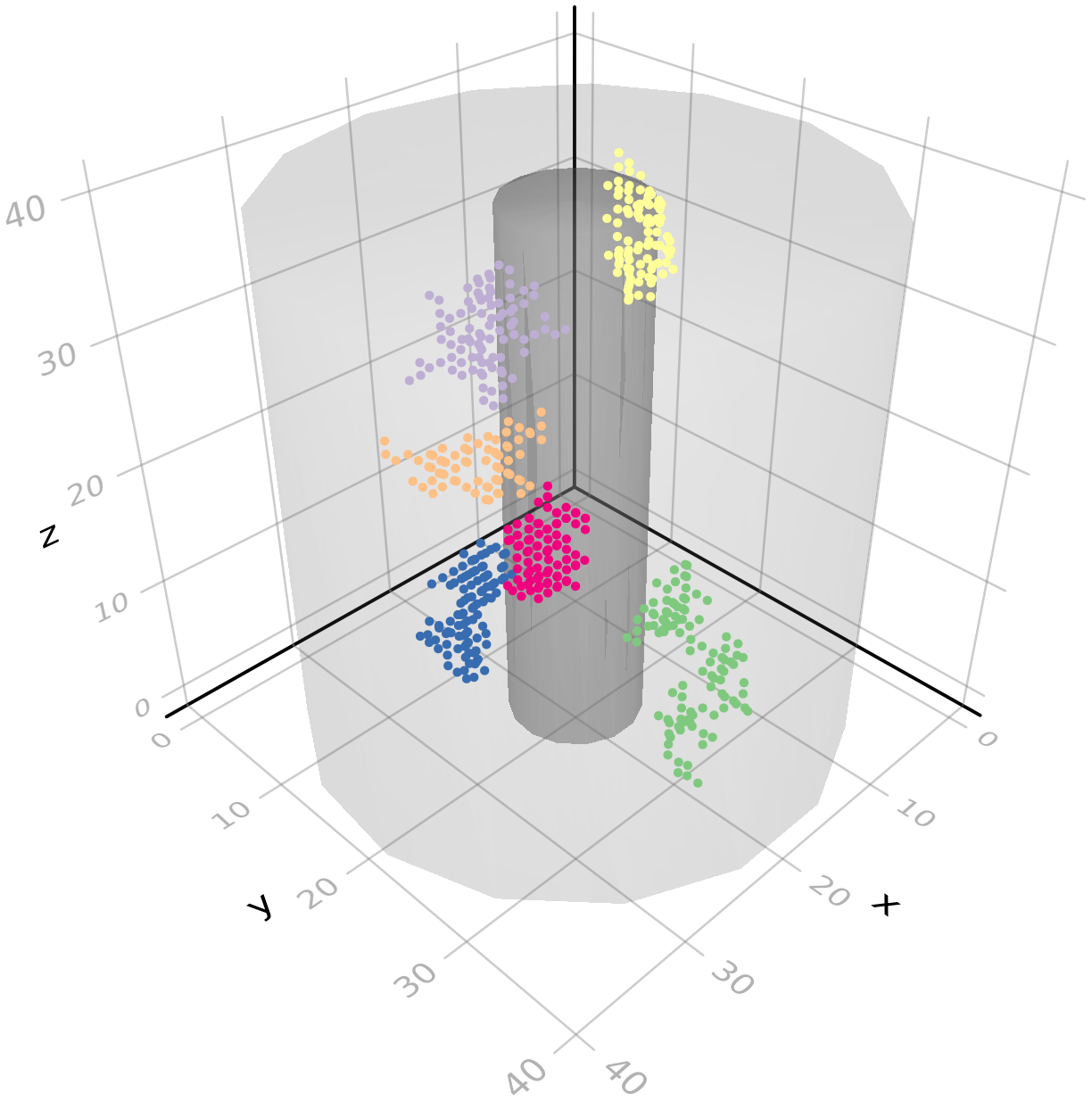


Figure 20: A generated 3D sample with $\rho_p = 0.32$, $\rho_q = 0.40$ and $\rho_c = 0.28$. This annulus has an outer radius of 20 particles, an inner radius of 5 particles and the height is 40 particles. Only clay clusters above a minimum size of 70 particles and where at least one particle is connected to the inner cylinder of the annulus are shown in the plot. All 26 neighbors are considered to be connected.

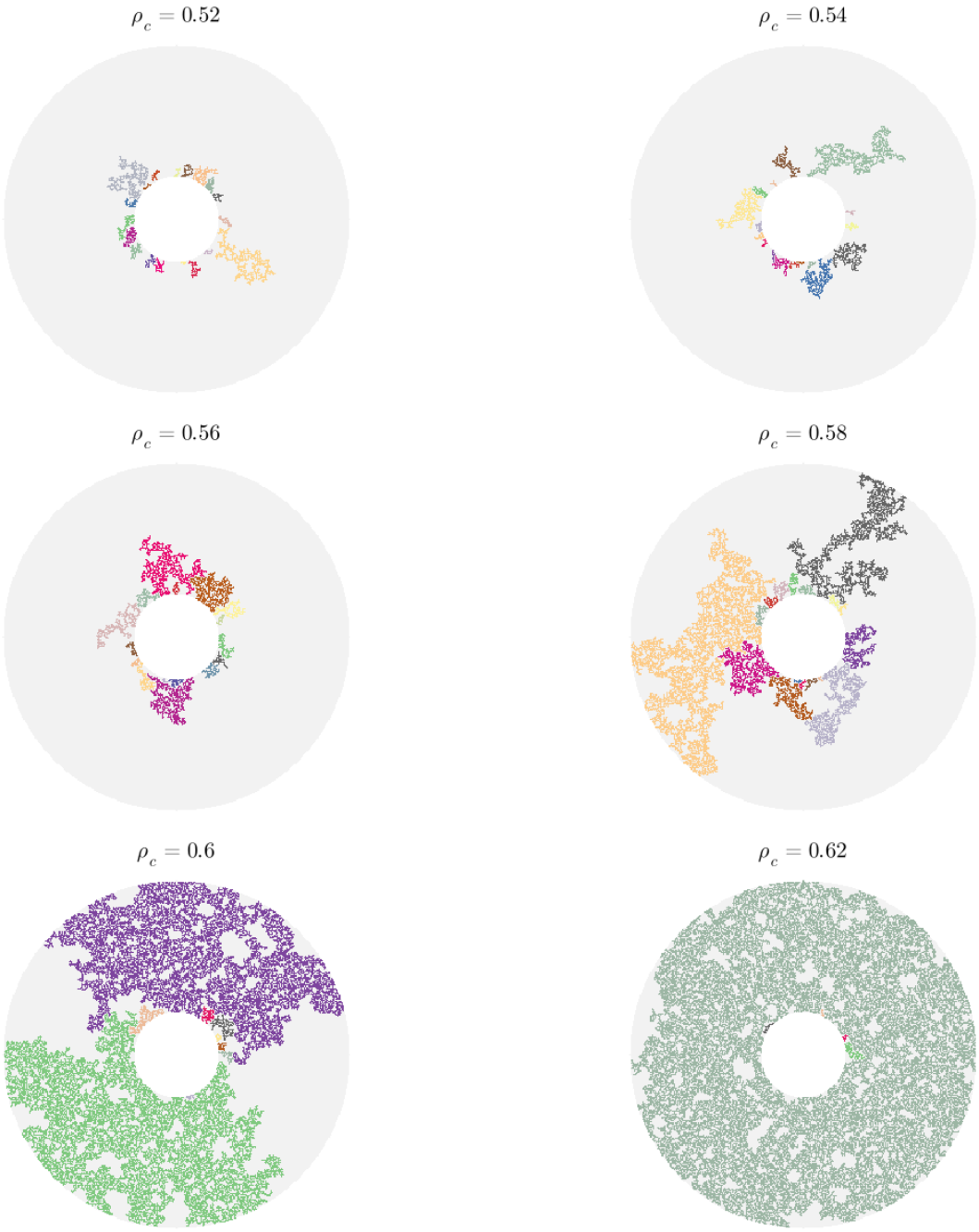


Figure 21: Clusters above a threshold size of 10 connected particles, and where at least one is connected to the inner circle of the annulus. For this computation, the von Neumann neighborhood was used. This is shown for different densities of clay.

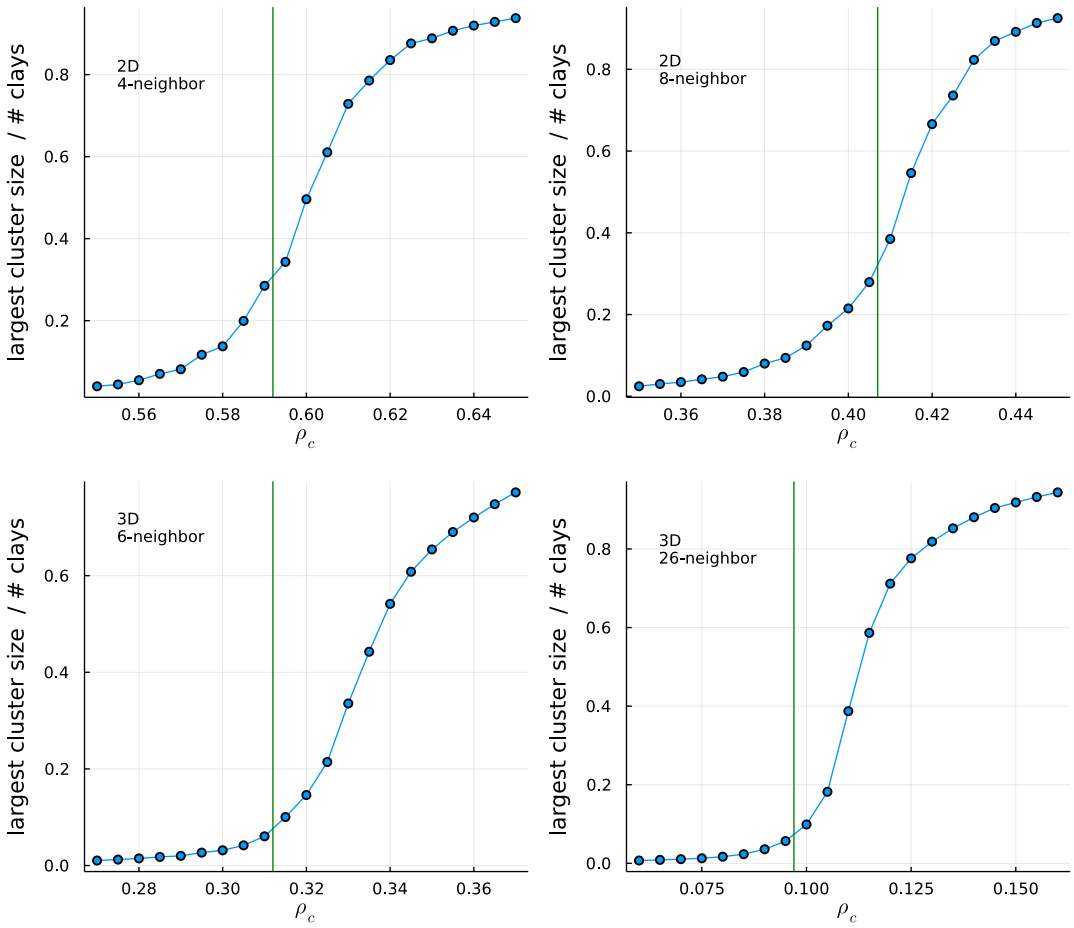


Figure 22: The plots shows the average size of the largest cluster from 70 samples, divided by the total number of clay particles for different clay densities ρ_c. The clay densities are chosen around the percolation thresholds, p_c, for a 4 and 8-neighbor case in 2d , respectively shown in the top left and top right plot, and 6 and 26-neighbor in 3d. The theoretical percolation threshold values are marked with green vertical lines.

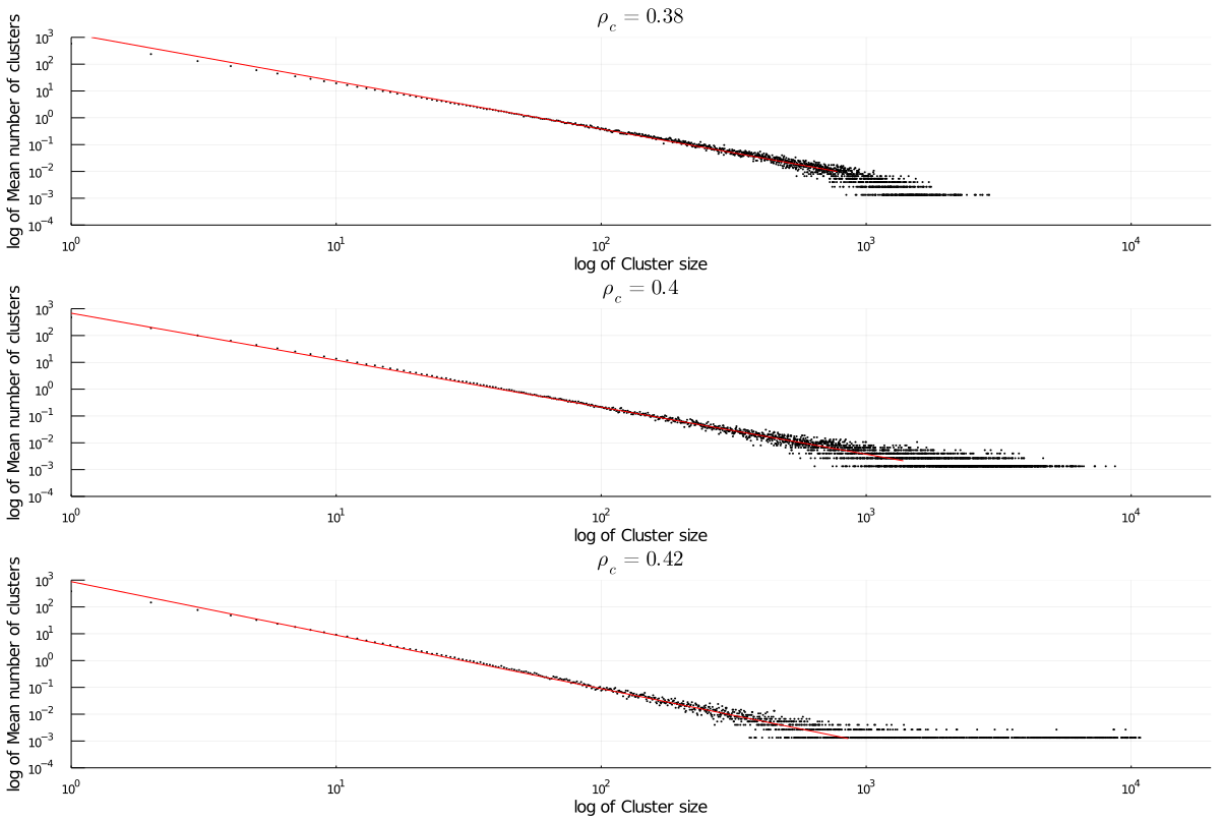


Figure 23: Distribution over cluster sizes for a variety of clay densities around the percolation threshold. Please notice that this is a log-log plot, as larger clay densities will lead to much larger and fewer clusters. A line was fitted on the first half of the x-axis to show the exponential decay around the threshold.

4.3 Swelling simulation

In this section, I will present the swelling simulations and their results for different parameters. Unless otherwise specified the default parameters for this section are given in Table 3. σ is stress difference, T is the temperature, $\eta_{\max}$ is the maximum number of swelling processes that each particle is allowed to undertake and α is the factor that the swelling particle's radius will increase by.

Table 3: The default values of the parameters N, ρ_p, ρ_q, ρ_c, σ, T, R_{outer}, R_{inner}, $\eta_{\max}$ and α for the simulations

N	ρ_p	ρ_q	ρ_c	σ	T	R_{outer}	R_{inner}	$\eta_{\max}$	α
151	0.1	0.5	0.4	10	2	20	5	70	1.01

4.3.1 Approximation for simplified model

To begin with I would like to simplify the problem further to be able to get an overview of the expected behaviour. To achieve that, a two dimensional annulus is chosen and instead of drawing the radii from a uniform distribution between 0.1 and 0.9 for each of the particles, they are all set to have a start radius of 0.5. Furthermore, also to help the understanding of the ongoing simulation, the particles will be treated collectively. The result of this exercise is found in Figure 24. At the beginning, as all clays have the same radius, they will have the same chance to swell and will swell by the same amount. This gives the expected average radius of the next iteration as seen in Equation (26).

$$r_{i+1}^{\text{avg}} = r_i^{\text{avg}}(1 + P^C(r_i^{\text{avg}}) \cdot (\alpha - 1)) \tag{26}$$

P^C is the collective chance to swell given the average radius and α is the swelling size factor, which is constant at 1.01. The basic chance to swell is given by Equation (9) and is still valid, but as the particles gets closer to the swelling cap $\eta_{\max}$ more and more of them will stop swelling.

If we ignore that fact and only consider the basic chance to swell, one gets the blue approximation in Figure 24. The red line is the simulation result. The red and blue curves show the total area of the of clay particles per iteration step. The grey line shows the maximum possible area, which is when all clay particles have swelled $\eta_{\max}$ times. This show that the approximation is valid in the regime before they approach the cap.

It is difficult to model how this system will behave closer to the cap. We notice that at step i, every particle will have been subjected to a set of previous approximate swelling probabilities $\{P_{i-1}^S, P_{i-2}^S,...\}$. This amounts to a set of Bernoulli trials with different chances of success (swell) and to find the expected amount of swells one can use the Poisson-Binomial distribution. A mathematical introduction to the distribution is given in Chen & Liu (1997).

$$P_i^C = P^{PB}\left[n < \eta_{\max}, \{P_{i-1}^S, .., P_1^S\}\right] \cdot P^S(r_i) \tag{27}$$

By using Equation (27) as the swelling probability one gets the green line in Figure 24. This fits well for the early iterations, as the P^{PB}-factor will be very close to 1. Unfortunately, it does not give a better fit closer to the cap as it overestimates the number of particles that has reached $\eta_{\max}$ (the maximum number of swells) at the start of the deviation. The green curve will then underestimate the number of particles that have reached their maximum number of swells for iterations after around 320. The swelling probability is not linear in r, so it may not be possible to treat the problem collectively with an average radius for all particles. For the most part, this approximation method (green curve) fitted quite well with the actual simulation (red curve). It looks like the blue curve could be considered an upper bound of growth.

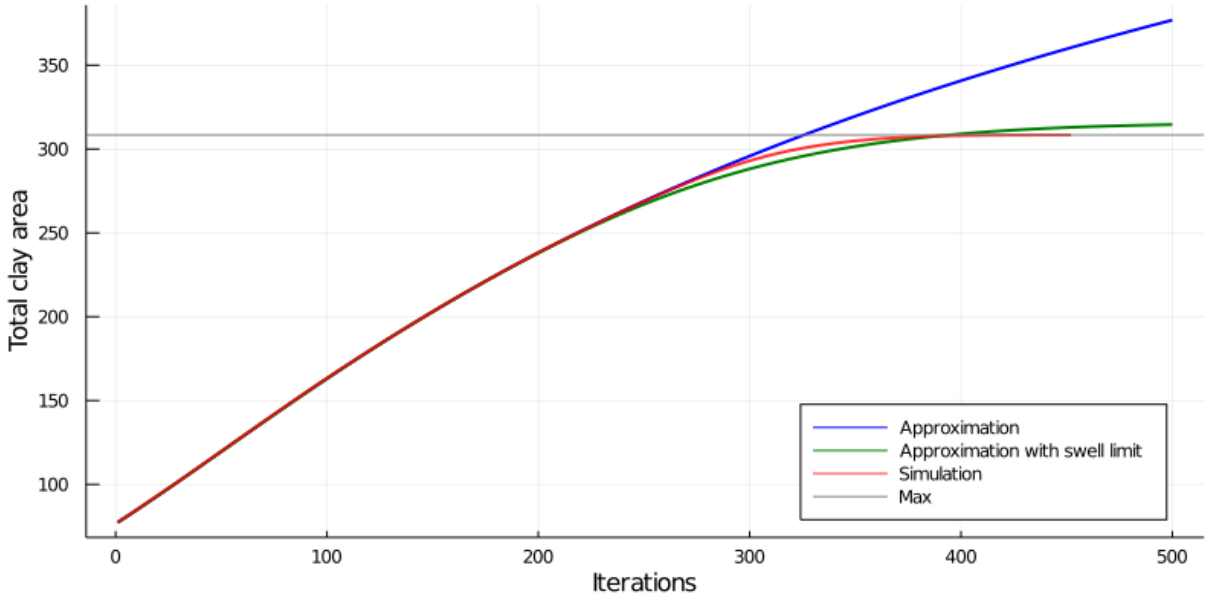


Figure 24: The simulation results of a generated sample with $\rho_p = 0.2$, $\rho_q = 0.4$ and $\rho_c = 0.4$ is shown by the red curve. Additionally, two calculated approximations for the simulations are shown by the blue and green curve. The maximum clay area that is possible for this setting is displayed by the grey line.

4.3.2 Simulation for different parameters

Figure 25 shows the results of the simulation for 10 different samples with the same configuration as stated in Table 3. As mentioned in Section 4.1 there are slight differences in the configuration and which are expected to lead to slightly different outcomes. Deriszadeh et al. (2014) has performed swelling experiments for different clay types. Even though this model is simple, the results obtain in this project are able to reconstruct similar swelling behavior.

Figures 26 and 27, show the evolution of samples with different temperature T and different stress σ, respectively. As known beforehand, both temperature and stress increase the rate of swelling of the samples. As it stands both factors contribute to the swelling in the same way, through the swelling probability given in Equation 9. Naturally, as the model is extended through the master project it would be natural to separate the treatment of these parameters.

4.3.3 Simulation with neighbor effects

In this section, I will present some of the same simulations that were shown in the previous section, but this time the simulations also accounts for neighbor effects through the addition of Equations (10) and (11) in Section 3.2.2 where also the neighbor effects are explained. Figure 28 shows the result of 10 samples with neighbor effects in red and the same 10 samples without the neighbor effects in blue. In the early stages the *porosity effect* allows the red samples to swell slightly faster, but that is quickly overtaken by the other factor, which accounts for neighbors' swelling and subsequent occupation of space. As the clay starts to swell the neighboring particles will slow down, and it takes a long time until it reaches maximum swelling.

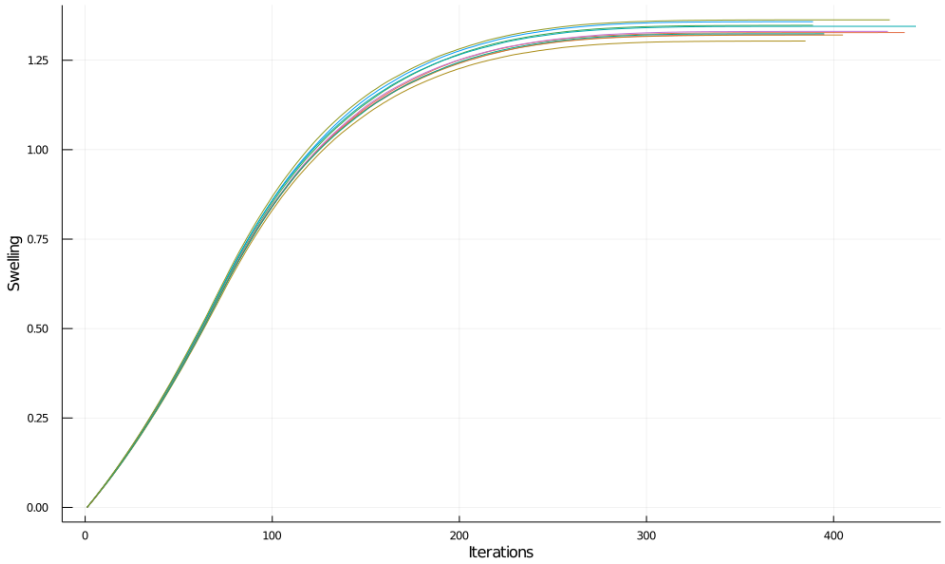


Figure 25: Swelling Simulation for 10 different samples. The y-axis shows the increase in total clay area.

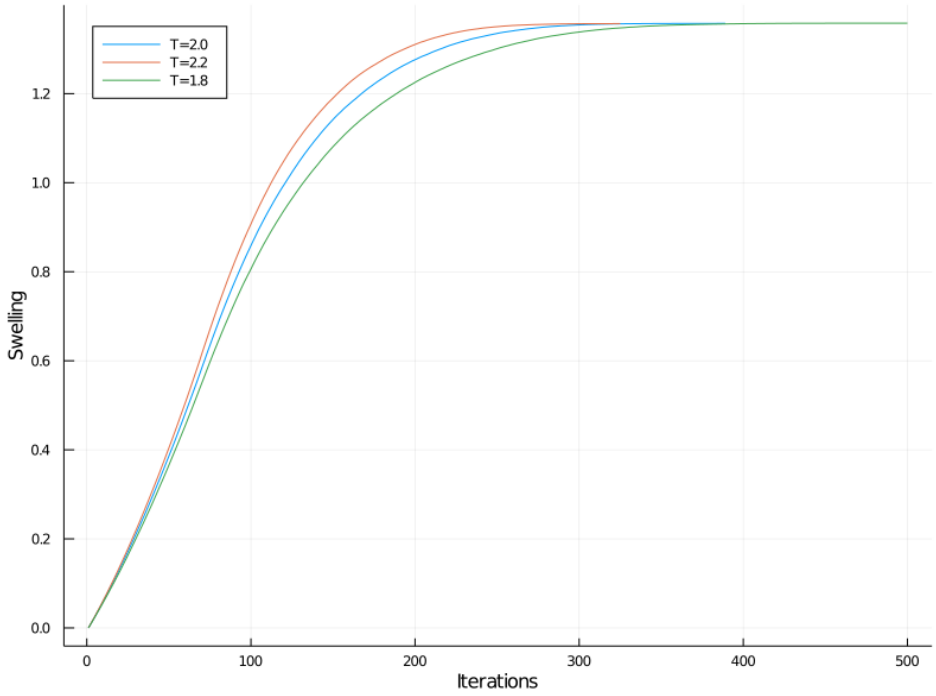


Figure 26: Swelling simulation for the same sample at different temperatures T. The y-axis shows the increase in total clay area.

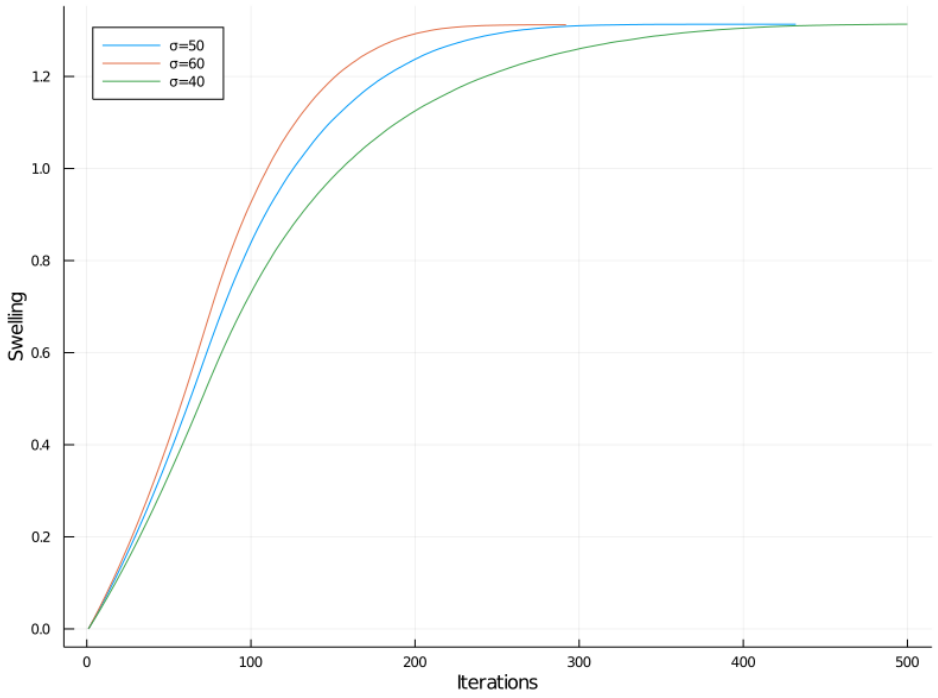


Figure 27: Simulation for the same sample at different stress σ. The y-axis shows the increase in total clay area.

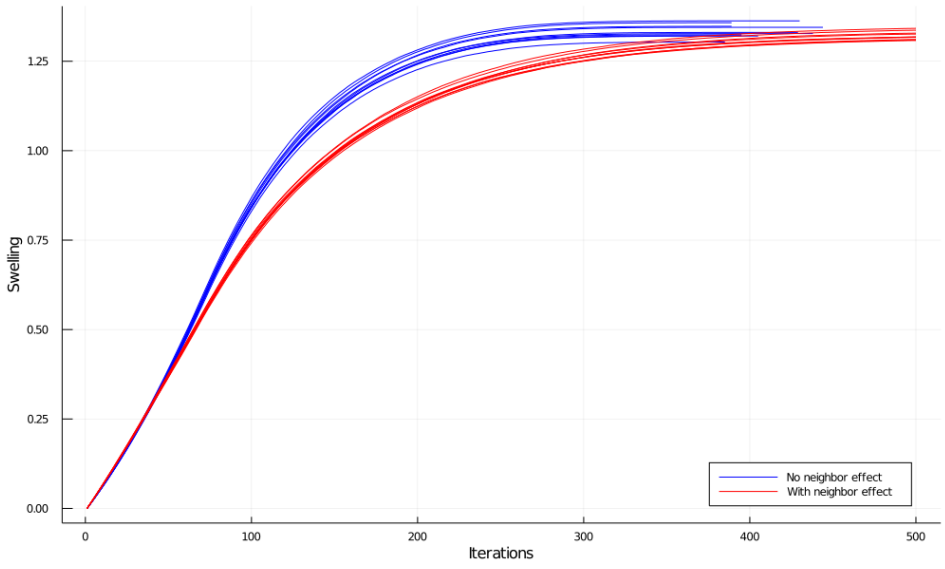


Figure 28: Swelling simulation for 20 different samples, where the 10 samples in blue is simulated without neighbor effects, and the 10 in red have been simulated with neighbor effects.

4.4 Mass transport

Several simulations of mass transport have been conducted. First, I will present the results from the the fully random walk setting, before I continue on to show the results with bias where the sample is subjected to an inward force. Table 4 shows the values that have been used in the mass transport simulations in this sections. The particles, and their initial grid position on the annulus, will be randomly generated each time.

Table 4: The default values of the parameters N, ρ_p, ρ_q, ρ_c, σ (different values for random and biased random walk), R_{outer}, R_{inner}, $\eta_{\max}$ and α for the simulations

N	ρ_p	ρ_q	ρ_c	σ (random walk)	σ (biased random walk)	R_{outer}	R_{inner}
150	0.25	0.5	0.25	0	10	20	5

4.4.1 Necessary model correction

When running the first test of the random walk module, both for the cases with and without the inward force, it became readily apparent that the results looked different than expected. This can be observed in Figure 30 and Figure 31, which show the same annulus sample after 150 time-steps with the random walk and biased random walk, respectively. It becomes evident that something make the particles move more towards the *southwest* direction, and that this trend is clearly visible for both versions of the random walk module.

After ensuring that all different functions worked as expected, it became apparent that special care had to be taken regarding the order the clay particles are processed in. More specifically, when always scanning the grid sequentially in the same order, the particles would eventually accumulate on the side that is processed first. This is due to the fact that only one clay particle can occupy a grid point at a time. The particles will get more easily stuck going in the direction of iteration. When reversing the iteration order, the particles get stuck on the opposite side of the annulus, and sample plots for this can for the interested reader be found in the Appendix B. An illustration of how the direction of iteration impacts the results can be seen in Figure 29. In this illustration, I have showed the most extreme difference, and have set the particles all to move in the same direction of the force to explain the weakness of the sequentially drawing and how much this could impact the results. Although, there is no force present to *guide* or *bias* the particles in one direction for the random walk, the same trend is visible there as well. This is then due to the fact that there are a much larger likelihood of available grid points in the direction against the iteration, and that this on average will cause more particles to move in that direction. To solve this the clay particles are processed in

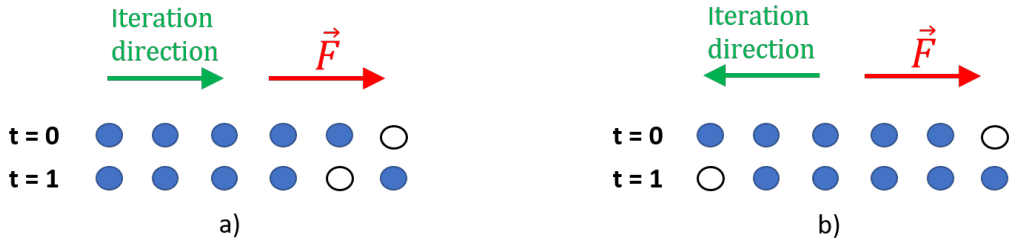


Figure 29: Illustration of how sequential drawing will impact on movement outcome

random order each iteration. No systemic issue is thus apparent. Another option would be to plan all the

move before committing to any one move, and resolve collisions properly. Shuffling the order was deemed a simpler solution and accurate enough for the work in this thesis. Four figures which further show how this sequential drawing process will cause *distorted* movement can be found in Appendix B.

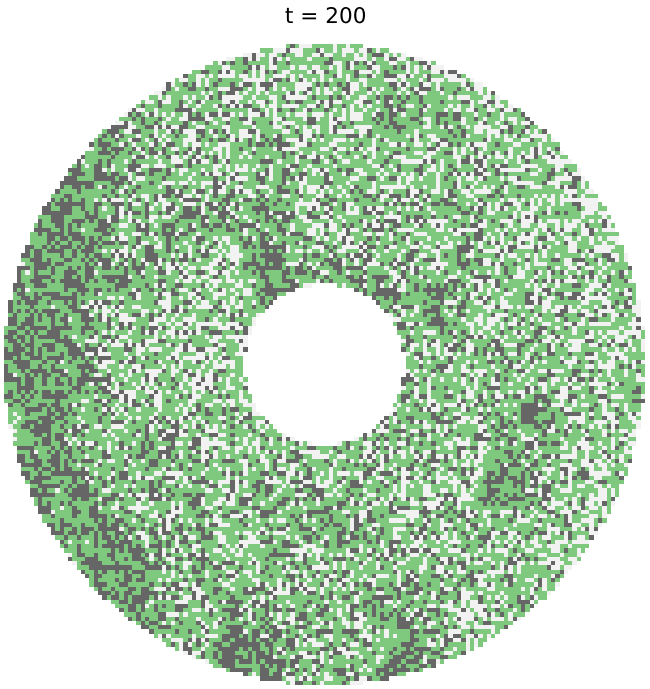


Figure 30: Sequential drawing and impact on movement outcome - example with random walk

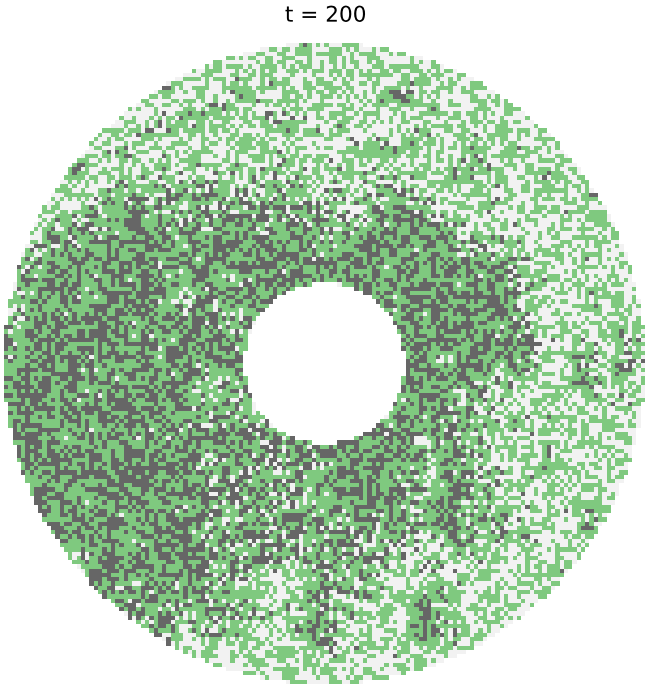


Figure 31: Sequential drawing and impact on movement outcome - example with biased random walk

4.4.2 Random walk and biased random walk

By its very nature, i.e. the aspect of supposedly being fully random, one would not expect significant patterns to appear from a random walk simulation. On the other hand, it will constitute as a baseline to compare against the biased random walk which was also implemented. It is first when the inwards stress is turned on that the inner ring of the annulus will feel the increasing clustering of the clay particles around it, which was attempted implemented for the biased random walk.

Figure 32 show the mass transport of a two dimensional annulus simulated over 150 time-steps. As expected there is no noticeable crowding anywhere on the annuli and after 150 time-steps, the sample looks about the same as when it started.

Figure 33, on the other hand, displays what happens to the samples when the biased random walk version of the mass transport is used. Gradually, the outer particles migrate closer to the center of the sample. This will proceed until they cannot go any further, either because they are in a channel that is fully isolated, or that they are blocked by other clay particles. Applying a constant stress on the sample, working radially inwards, will gradually build up swelling pressure on the internal wall of the annulus. The clay particles move around the internal wall and with the bias from the applied pressure, the particles slowly make their way towards the center through their channels.

Figure 35 and Figure 36 show instead a three dimensional example of mass transport, with Figure 35 showing the sample at time step 0 and Figure 36 shows it at time step 40. Only clay clusters with a size of more than 70 particles and with at least one clay connected to the inner ring is shown in the plot. In three dimensions, each particle has access to more neighbors than the equivalent sample in two dimensions, and one can see that that after only 40 time steps a very large cluster has formed. At the inner ring the felt pressure will be dependent on the amount of swelling that the has undertaken as the clay will then push inwards. To begin with, one can look at the counted number of clay clusters as an aggregate measure for this swelling pressure working inward to the borehole.

Figure 34 shows how the numbers of clay particles connected to inner ring increases with the biased random walk. On the other hand, in the random walk the number is about constant. Due to the geometry there is more space towards the middle and outer portions of the annuli, and the particles will spread out a bit, reducing ever so slightly the cluster count connected to the center in the fully random case. For the biased random walk, the amount of particles grows quickly before slowing down as the particles accumulate towards the center. Many clay particles will never be able to reach the center because the are in channels that are not connected to the inner ring.

4.5 Combined swelling and mass transport

In this section all previous systems will be turned on and combined in a single iteration loop. For each iteration each clay particle will have a possibility to swell one time with same mechanism as in Section 4.3. Furthermore, the particle will also have the possibility to move one step, equally set up as in Section 4.4. The value of primary interest here is the total area of clay particles in clusters where at least one particle is connected to the inner annulus. This is comparable in certain respects to the swelling pressure pushing on the inner ring, and therefore relevant for the use case that we are looking at, i.e., where pressure on the wellbore could reach problematic levels.

Figure 37 show two samples which were simulated for 400 time-steps. Here the 4-neighborhood was used. The blue one was subjected to $\sigma = 50$ while the other to $\sigma = 20$. In the case with the higher stress, the pressure accumulates faster on the inner ring, before it starts to level off a bit. At lower stress, the

sample still swells but the rate is much lower. The figure is quite similar to the figures in Deriszadeh et al. (2014), which are include in Figure 5, specifically the ones showing accelerated versus conventional swelling. Increasing the pressure has the same kind of effect as they observed by adding a electrical field to force mass transport. Figure 38 shows the same configuration but in three dimensions with a 6-neighborhood. The dimensions were scaled down to 50x50x50 because of the computational load. For the same configuration the samples swell faster in three dimension. With more neighbors the particles flows quicker towards the inner annulus as there are more "avenues" through the sample. Additionally, the cross-section is not as large as it had to be scaled down and this makes it plateau faster.

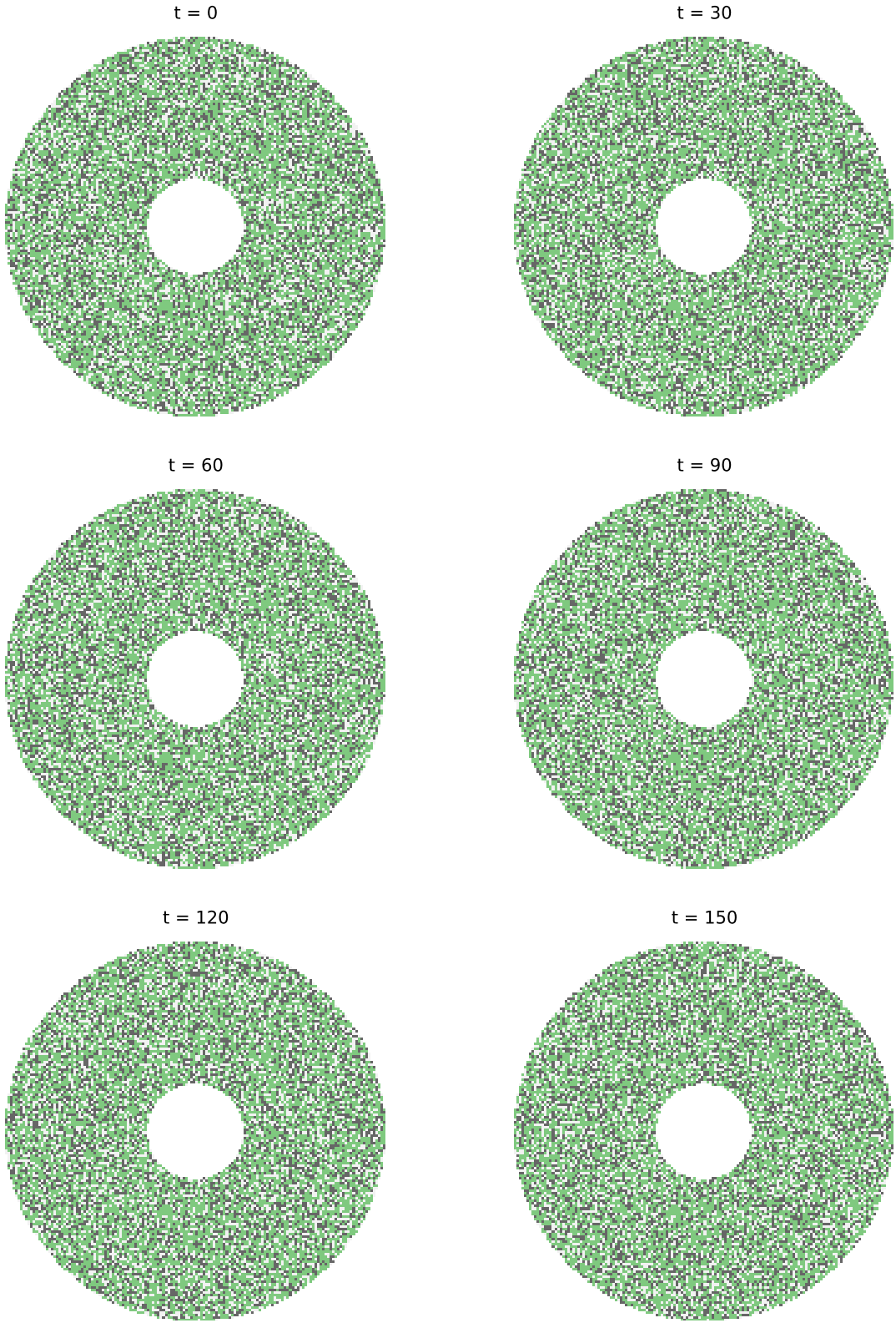


Figure 32: Sample subjected to random walk shown at time-steps 0, 30, 60, 90, 120 and 150. The clay particles are grey, quartz is green and the pores are white. No noticeable change is apparent.

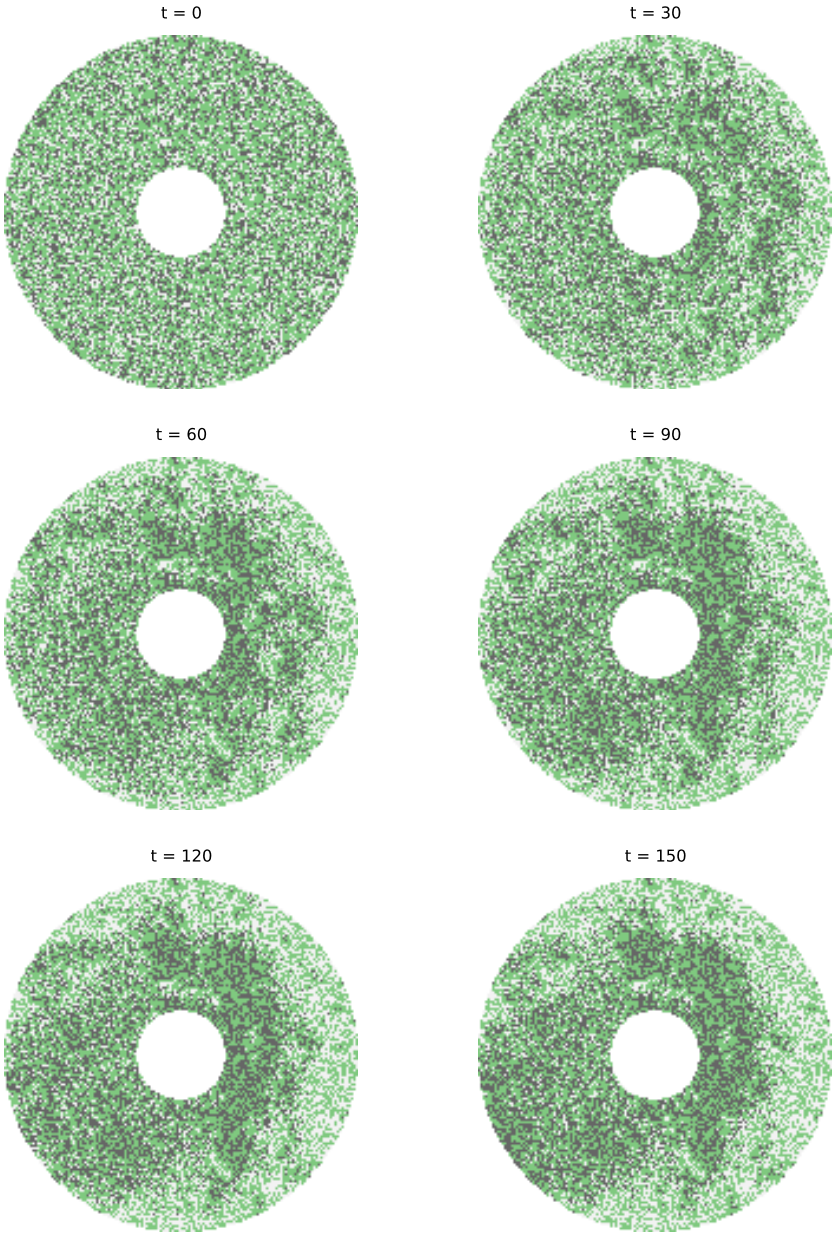


Figure 33: Sample subjected to biased random walk shown at time-steps 0, 30, 60, 90, 120 and 150. The clay particles are grey, quartz is green and the pores are white.

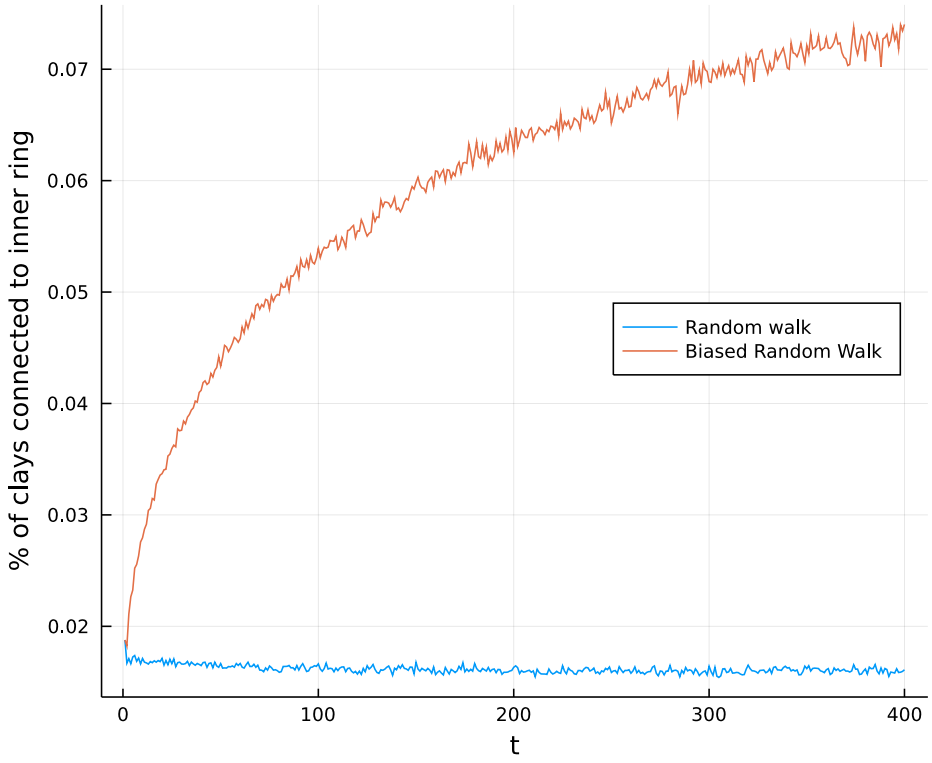


Figure 34: Average percentage of clay particles connected to inner ring for 100 samples subjected to random walk shown compared to 100 samples subjected to biased random walk. Both types were done with a 2D annulus of size 150x150 with $\rho_c = 0.25$, $\rho_q = 0.50$, $\rho_p = 0.25$.

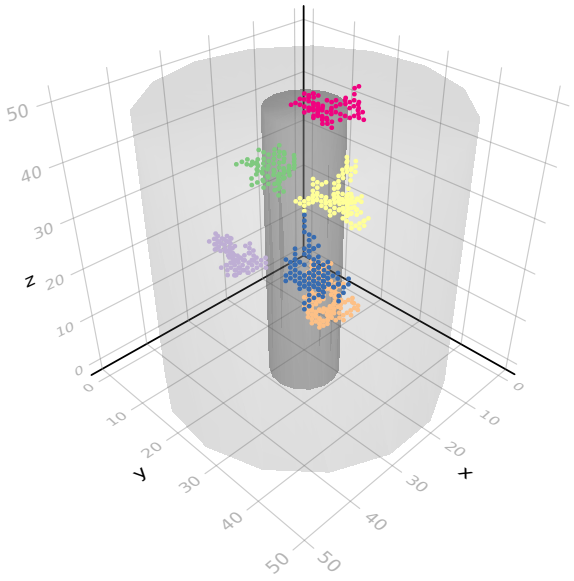


Figure 35: Example of 3D mass transport at time-step 0. Only the six most immediate neighbors are considered to connect clusters and allow for movement. $\rho_p = 0.18$, $\rho_q = 0.55$ and $\rho_c = 0.27$.

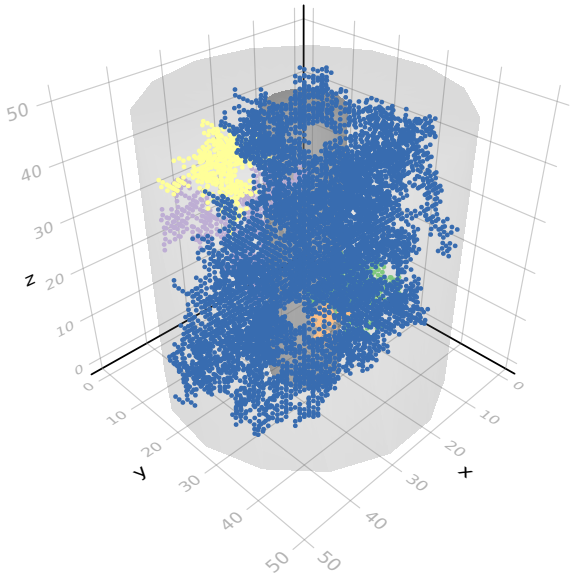


Figure 36: Example of 3D mass transport at time-step 40. Only the six most immediate neighbors are considered to connect clusters and allow for movement. $\rho_p = 0.18$, $\rho_q = 0.55$ and $\rho_c = 0.27$

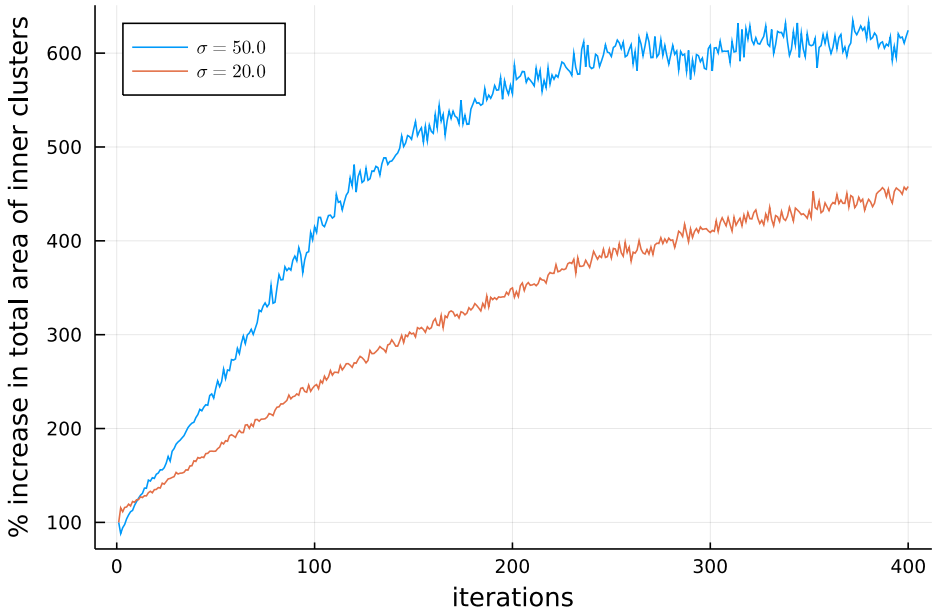


Figure 37: Two samples setups are simulated for 400 time steps and averaged over 75 runs, with both swelling and mass transport. The samples are 2D, with a size of 150x150. The densities are $\rho_p = 0.25$, $\rho_q = 0.50$ and $\rho_c = 0.25$. The blue line has a $\sigma = 50$ while the orange has a $\sigma = 20$.

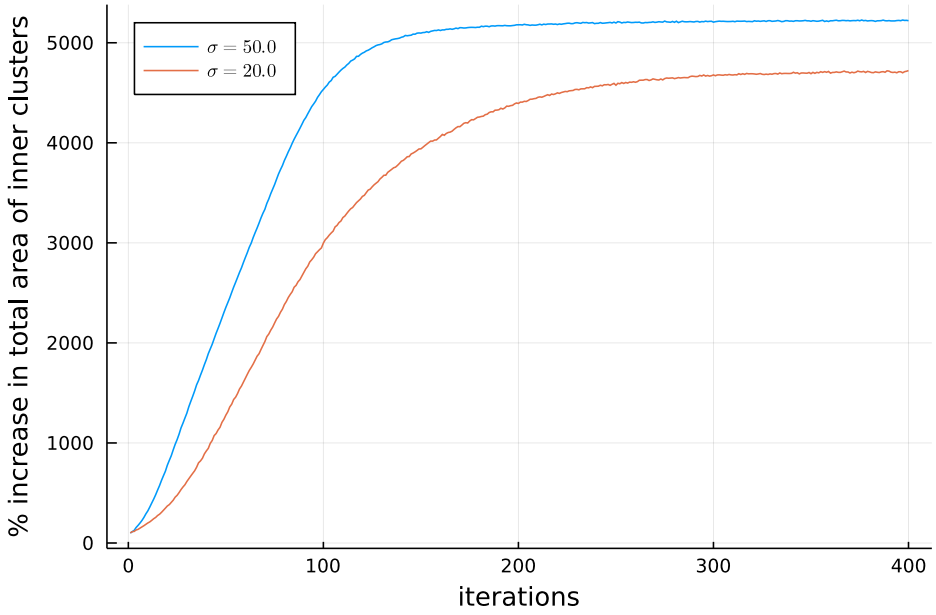


Figure 38: Two samples setups are simulated for 400 time steps and averaged over 75 runs, with both swelling and mass transport. The samples are 3D, with a size of 50x50x50. The densities are $\rho_p = 0.25$, $\rho_q = 0.50$ and $\rho_c = 0.25$. The blue line has a $\sigma = 50$ while the orange has a $\sigma = 20$.

5 Concluding Remarks

The main outcome of this master thesis work has been the creation of a simulation model built on the principles of a discrete element method (DEM). This has been used to model clay swelling and mass transport in shale samples. The modelling framework has been primarily focused around annuli-structures, and both 2D or 3D grids have been implemented. Individual particles are placed on a grid structure and simulated with different discrete time simulations. The model has been implemented using the Julia programming language, which makes it possible to combine high-level programming in a modern language with good tooling, while still getting very competitive performance for numerical simulations. One aim has been to utilize Julia's capabilities to write clean and understandable code, that is fast, with efficient algorithms, such as the 2-pass labelling algorithm Hoshen-Kopelman for both 2D and 3D grids.

Additionally, some work has been conducted to find an appropriate level of detail to include in the model, and also which shale and clay properties to model and which parameters that would be both of interest and possible to include. At this point temperature, stress gradient, different particle densities and distributions are identified as important. There are also parameters directly related to swelling and mass transport, which were two main drivers of sample swelling that were implemented in this thesis. This is a complicated field, and there are certainly other parameters and processes as well that could be included. Some of these will be mentioned in Section 6 where future research ideas are outlined.

Some analyses in the computational study were conducted on static samples to explore aspects of the sample generation, while other samples have been investigated through simulations methods.

Cluster analysis was the main investigative tool used on the static samples. This included to study clusters and maximum cluster sizes, and looking distributions for the number of cluster as a dependent variable of cluster size. As one would expect, the size of the largest clay cluster is highly dependent on the clay density. For the annulus shaped grid, it was tested how the largest cluster size of the samples will correspond to the theoretical percolation thresholds. This has been done for for all grid settings (different neighbor definitions and 2D versus 3D), and as expected when considering a finite grid, the observed critical threshold for clay density will be a bit higher than the theoretical percolation threshold which considers an infinite grid.

In addition to the several different static sample investigations that have been conducted, many simulations have been run with a variety of different parameter configurations. Here the purpose has been to investigate swelling in shale, and this was performed by looking at clay swelling, mass transport and a combination of both. The setup and processes have been explained in Section 3. Specifically, the swelling pressure on the inner circle of the annulus was found when the sample was subjected to both clay swelling and mass transport. The results from the numerical experiments conducted in this modelling framework are in general in agreement with what was expected from literature as explained in Section 1.3. To more concretely be able to connect the output of the model with the physical conditions in a borehole, it would be necessary to calibrate the parameters and simulation results by using experimental values.

Section 6 lists some ideas for future research for this model of clay swelling in shale.

6 Future Research

In this section, I will outline some ideas for further development of the model. While working on the master thesis several ways to improve the model were identified.

The model has so far seemed to capture some aspects of swelling in clay, yet it cannot be used to make any real predictions. The most important follow-up work, would to calibrate the model with real world data. Microstructure information from image analysis of field cores from shale rocks would be an important input for further improvement. This would allow us to accurately choose and set parameters to make the model more realistic.

The presence of water is perceived to be of key importance in the process of swelling and contraction of clay particles. This far, it has been assumed that water is present so that the clay could swell, but the presence and amount of water could also be included in the model. It could both be incorporated as a parameter or simulated with its own process. One might get some ideas from computational fluid dynamics on how to incorporate this.

There are also important chemical reactions and processes happening in shale. For instance, different ions will in a water solution impact the charge and the forces inside the layers of clay particles and between clay particles. This is something one might be able control during e.g., drilling, and this can thus provide real-life applications for the petroleum industry. This could also make it easier model different types of clay, which have different swelling properties. This is something that could be further investigated, and it should therefore be considered for future work.

References

Akinshipe, O. & Kornelius, G. (2017), 'Chemical and thermodynamic processes in clay brick firing technologies and associated atmospheric emissions metrics-a review', *Journal of Pollution Effects & Control* **05**.

Anandarajah, A., Amarasinghe, P. M. & University, J. H. (2012), 'Behavior of Swelling Clays: A Discrete Element Study', p. 10.

Balaban, R. d. C., Vidal, E. L. F. & Borges, M. R. (2015), 'Design of experiments to evaluate clay swelling inhibition by different combinations of organic compounds and inorganic salts for application in water base drilling fluids', *Applied clay science* **105-106**, 124–130.

Bezanson, J., Edelman, A., Karpinski, S. & Shah, V. B. (2017), 'Julia: A Fresh Approach to Numerical Computing', *SIAM Review* **59**(1), 65–98. Publisher: Society for Industrial and Applied Mathematics.

Biot Poroelasticity - Application ID: 483 (n.d.), `https://www.comsol.no/model/biot-poroelasticity-483`. [Online; accessed 10-March 2021].

Blair, T. & McPherson, J. (1999), 'Grain-size and textural classification of coarse sedimentary particles', *Journal of Sedimentary Research* **69**, 6–19.

Bundy, W. M. & Ishley, J. N. (1991), 'Kaolin in paper filling and coating', *Applied Clay Science* **5**(5), 397 – 420.

Burdige, D. J. (2018), Diagenesis, *in* P. T. Bobrowsky & B. Marker, eds, 'Encyclopedia of Engineering Geology', Springer International Publishing, Cham, pp. 229–231.

Chen, S. X. & Liu, J. S. (1997), 'Statistical applications of the poisson-binomial and conditional bernoulli distributions', *Statistica Sinica* **7**(4), 875–892.

COMSOL Inc. (2021*a*), 'Simulate fluid flow applications with the cfd module', `https://www.comsol.no/cfd-module`. [Online; accessed 2-Mar 2021].

COMSOL Inc. (2021*b*), 'Understand, predict, and optimize physics-based designs and processes with comsol multiphysics', `http://www.comsol.com/products/multiphysics/`. [Online; accessed 2-Mar 2021].

Datseris, G., Vahdati, A. R. & DuBois, T. C. (2021), 'Agents.jl: A performant and feature-full agent based modelling software of minimal code complexity'.

Dayal, A. M. & Mani, D. (2017), Chapter 5 - exploration technique, *in* A. M. Dayal & D. Mani, eds, 'Shale Gas', Elsevier, pp. 65 – 93.

Deriszadeh, M., Deriszadeh, M., Wong, R. C. K. & Wong, R. C. K. (2014), 'One-dimensional Swelling Behavior of Clay and Shale Under Electrical Potential Gradient', *Transport in porous media* **101**(1), 35–52. Place: Dordrecht Publisher: Springer Netherlands.

Dunning, I., Huchette, J. & Lubin, M. (2017), 'JuMP: A Modeling Language for Mathematical Optimization', *SIAM Review* **59**(2), 295–320.

Eberhardt, E. & Amini, A. (2018), Hydraulic fracturing, *in* P. T. Bobrowsky & B. Marker, eds, 'Encyclopedia of Engineering Geology', Springer International Publishing, Cham, pp. 489–495.

Equinor (2019), 'Sleipner partnership releases co2 storage data', `https://www.equinor.com/en/news/2019-06-12-sleipner-co2-storage-data.html`. [Online; accessed 12-Dec 2020].

Fernandes, F. M., Lourenço, P. B. & Castro, F. (2010), Ancient clay bricks: Manufacture and properties, *in* M. B. Dan, R. Přikryl & Á. Török, eds, 'Materials, Technologies and Practice in Historic Heritage Structures', Springer Netherlands, Dordrecht, pp. 29–48.

Hansen, E., Hemmen, H., Fonseca, D., Coutant, C., Knudsen, K., Plivelic, T., Bonn, D. & Fossum, J. (2012), 'Swelling transition of a clay induced by heating', *Scientific reports* **2**(1), 618–618. Place: England Publisher: Nature Publishing Group.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. & Oliphant, T. E. (2020), 'Array programming with NumPy', *Nature* **585**(7825), 357–362. Number: 7825 Publisher: Nature Publishing Group.

Hoshen, J. & Kopelman, R. (1976), 'Percolation and cluster distribution. i. cluster multiple labeling technique and critical concentration algorithm', *Phys. Rev. B* **14**, 3438–3445.

Hubbe, M. & Gill, R. (2016), 'Fillers for papermaking: A review of their properties, usage practices, and their mechanistic role', *BioResources* **11**.

International Energy Agency (2020), 'Cement - tracking report', https://www.iea.org/reports/cement. [Online; accessed 12-Dec 2020].

Kesten, H. (1982), 'Percolation theory for mathematicians', https://doi.org/10.1007/978-1-4899-2730-9. [Online; accessed 1-Mar 2021].

Kmieć, M., Karpiński, B., Antoszkiewicz, M. & Szkodo, M. (2018), 'Laboratory research on the influence of swelling clay on the quality of borehole cementing and evaluation of clay-cutting wellbore tool prototype', *Applied clay science* **164**, 13–25.

Kurzawski, L. & Malarz, K. (2012), 'Simple cubic random-site percolation thresholds for complex neighbourhoods', *Reports on mathematical physics* **70**(2), 163–169.

Leeuwen, J. v. (1990), 'Handbook of theoretical computer science : A : Algorithms and complexity'.

Li, Q., Shi, W. & Yang, Q. (2021), 'Method to determine van der waals potential energy of particle interactions for soil clay by dynamic light scattering', *European journal of soil science* .

Lie, K.-A. (2017), 'Introduction to flow simulation as used in reservoir engineering', https://folk.ntnu.no/andreas/hasselt17/introResSim.pdf. [Lecture held at Multiscale Methods Summer School June 26–29, 2017, Hasselt, Belgium. Online; accessed 6-Jun 2021].

MagentaGreen (2014), 'Scheme of a oil trap of the type geological saddle, called in technical term anticline'. [Illustration published online at https://commons.wikimedia.org/wiki/File:Anticline_trap.svg under the licence CC BY-SA 3.0. Further licence information: https://creativecommons.org/licenses/by-sa/3.0, via Wikimedia Commons].

Mainçon, P. (2021), 'Writing type-stable julia code', https://blog.sintef.com/industry-en/writing-type-stable-julia-code/. [Online; accessed 1-May 2021].

Malarz, K. & Galam, S. (2005), 'Square-lattice site percolation at increasing ranges of neighbor bonds', *Phys. Rev. E* **71**, 016125.

Malthe-Sørenssen, A. (2020), 'Percolation theory using python', https://www.uio.no/studier/emner/matnat/fys/FYS4460/v20/notes/book.pdf. [Online; accessed 6-Jun 2021].

Moore, D. E., Morrow, C. A. & Byerlee, J. D. (1982), 'Use of swelling clays to reduce permeability and its potential application to nuclear waste repository sealing', *Geophysical research letters* **9**(9), 1009–1012.

Nandi, A. (2018), Clay, *in* P. T. Bobrowsky & B. Marker, eds, 'Encyclopedia of Engineering Geology', Springer International Publishing, Cham, pp. 133–134.

Norsk Petroleum (2020), 'Eksport av olje og gass', https://www.norskpetroleum.no/produksjon-og-eksport/eksport-av-olje-og-gass/. [Online; accessed 29-Oct 2020].

Norwegian Geotechnical Institute (n.d.), 'Landslides - quick clay', https://www.ngi.no/eng/Services/Technical-expertise/Landslides/Quick-clay#. [Online; accessed 11-Nov 2020].

Núñez-López, V. & Moskal, E. (2019), 'Potential of co2-eor for near-term decarbonization', *Frontiers in Climate* **1**, 5.

OpenFOAM (2021), 'About openfoam', https://www.openfoam.com/. [Online; accessed 2-Mar 2021].

PoreLab (n.d.), 'About porelab', https://porelab.no/about/. [Online; accessed 1-Dec 2020].

Rackauckas, C. & Nie, Q. (2017), 'DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia', *Journal of Open Research Software* **5**, 15.

Rybacki, E., Herrmann, J., Wirth, R. & Dresen, G. (2017), 'Creep of Posidonia Shale at Elevated Pressure and Temperature', *Rock mechanics and rock engineering* **50**(12), 3121–3140. Place: Vienna Publisher: Springer Science and Business Media LLC.

Shang, X., Lu, J., Kuang, L., Yang, C. & Zhou, G. (2018), 'Empirical formulae for electric double-layer repulsion between two arbitrarily inclined clay particles', *Journal of Rock Mechanics and Geotechnical Engineering* **10**(6), 1183–1189.
URL: *https://www.sciencedirect.com/science/article/pii/S167477551830115X*

Sherwood, J. D. (1993), 'Biot poroelasticity of a chemically active shale', *Proceedings of the Royal Society. A, Mathematical and physical sciences* **440**(1909), 365–377.

SINTEF (2018), 'Economic analysis of coordinated plug and abandonment operations', https://www.sintef.no/prosjekter/ecopa/. [Online; accessed 12-Dec 2020].

Solberg, I.-L. & Hansen, L. (2021), 'Kan det gå kvikkleireskred på tomta mi?', https://www.ngu.no/blogg/kan-det-g%C3%A5-kvikkleireskred-p%C3%A5-tomta-mi. [Online; accessed 6-Jun 2021].

The Norwegian Government (2020), 'Press conference on carbon capture and storage', https://www.regjeringen.no/en/aktuelt/press-conference-on-carbon-capture-and-storage/id2765326/. [Online; accessed 5-Oct 2020].

The Norwegian Water Resources and Energy Directorate (2021), 'Kvikkleireskredet i gjerdrum', https://www.nve.no/om-kvikkleire/kvikkleireskredet-i-gjerdrum/. [Online; accessed 6-Jun 2021].

Wood, D. A. (2020), 'Predicting porosity, permeability and water saturation applying an optimized nearest-neighbour, machine-learning and data-mining network of well-log data', *Journal of Petroleum Science and Engineering* **184**, 106587.

Zolotukhin, A. B. & Ursin, J.-R. (2000), Introduction to petroleum reservoir engineering, 2nd edn, Cappelen Damm AS.

Zou, C. (2017), Chapter 10 - Shale Oil and Gas, *in* C. Zou, ed., 'Unconventional Petroleum Geology (Second Edition)', Elsevier, pp. 275–321.

Çengel, Y. A. & Cimbala, J. M. (2010), Fluid mechanics: Fundamentals and applications, 2nd SI units edn, McGraw-Hill Education, pp. 131–133 and 853–867.

A Appendix - Submitted Conference Poster

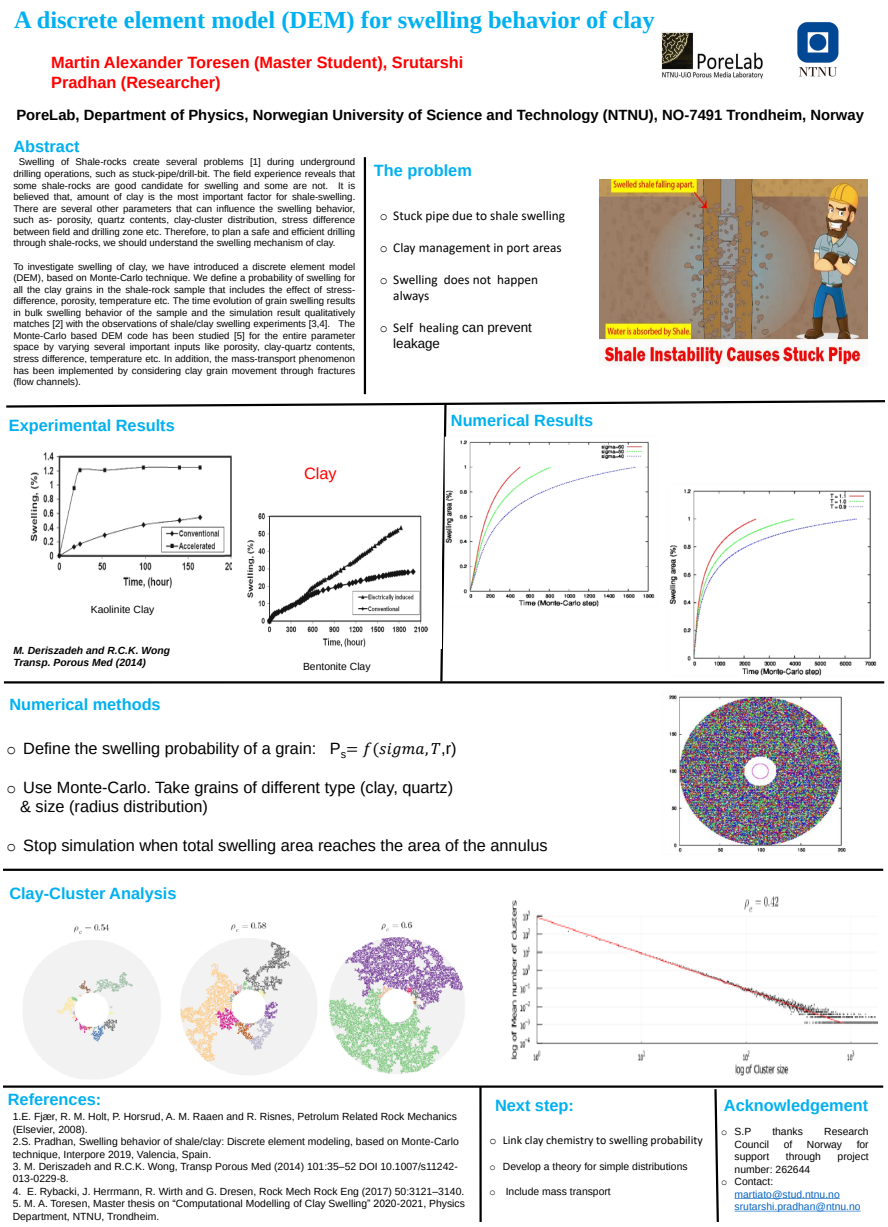


Figure A1: This conference poster was submitted for the Interpore2021 Conference held in May 2021 by Dr. Srutarshi Pradhan and myself.

B Appendix - Additional figures

B.1 Random walk

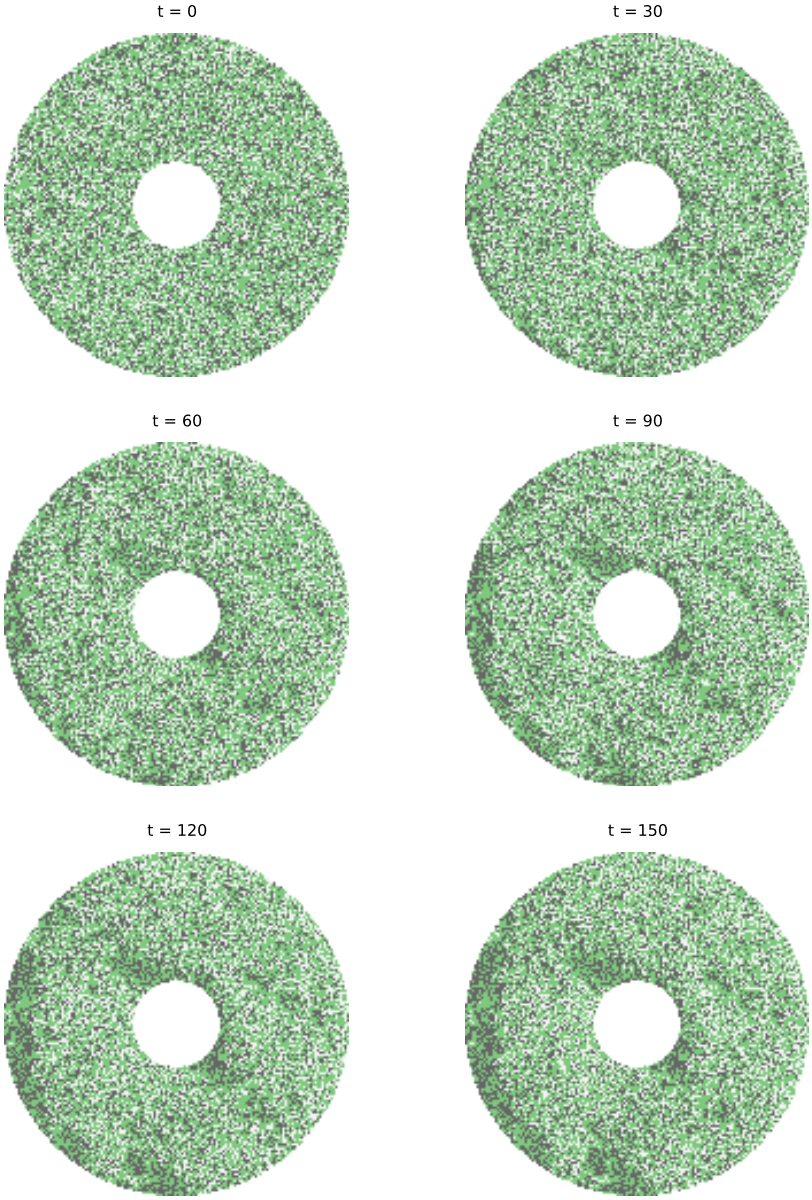


Figure A2: Sample subjected to *distorted* random walk shown at time-steps 0, 30, 60, 90, 120 and 150. The clay particles are grey, quartz is green and the pores are white.

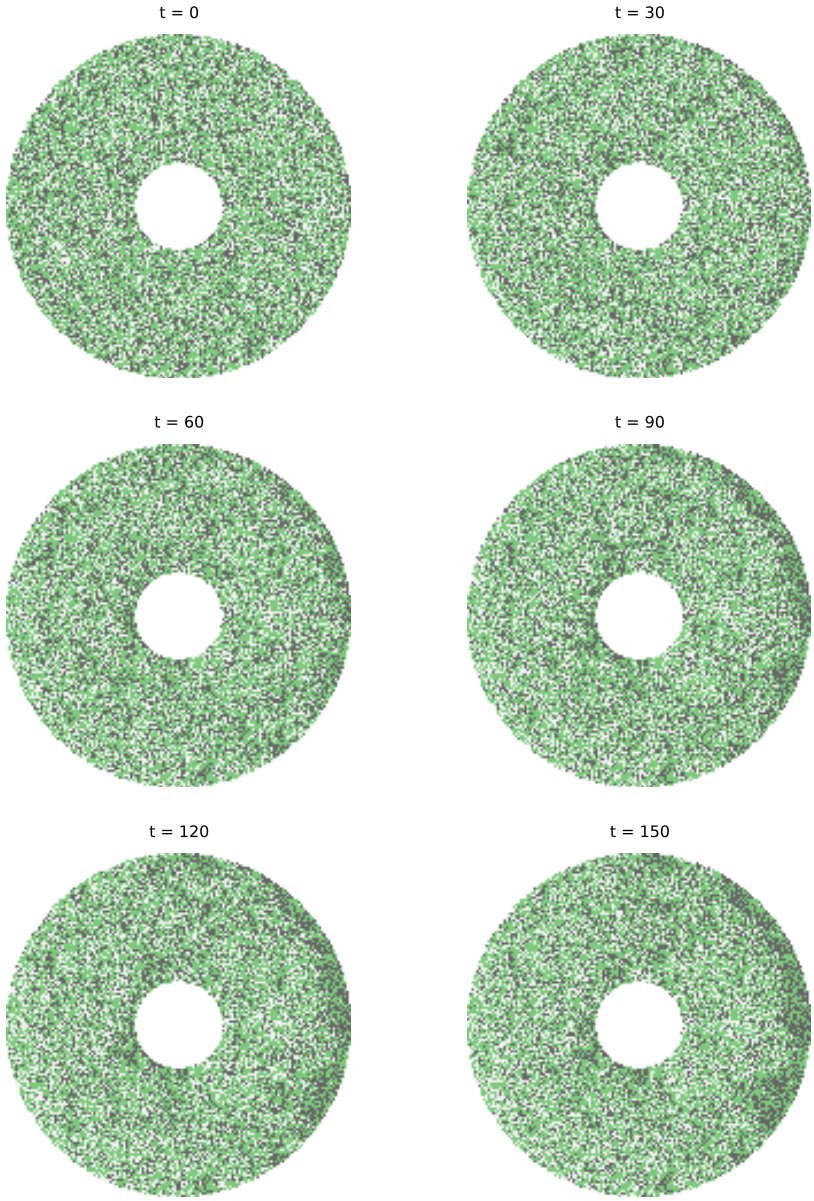


Figure A3: Sample subjected to *distorted* random walk shown at time-steps 0, 30, 60, 90, 120 and 150 and drawn in the opposite order compared to Figure A2. The clay particles are grey, quartz is green and the pores are white.

B.2 Biased random walk

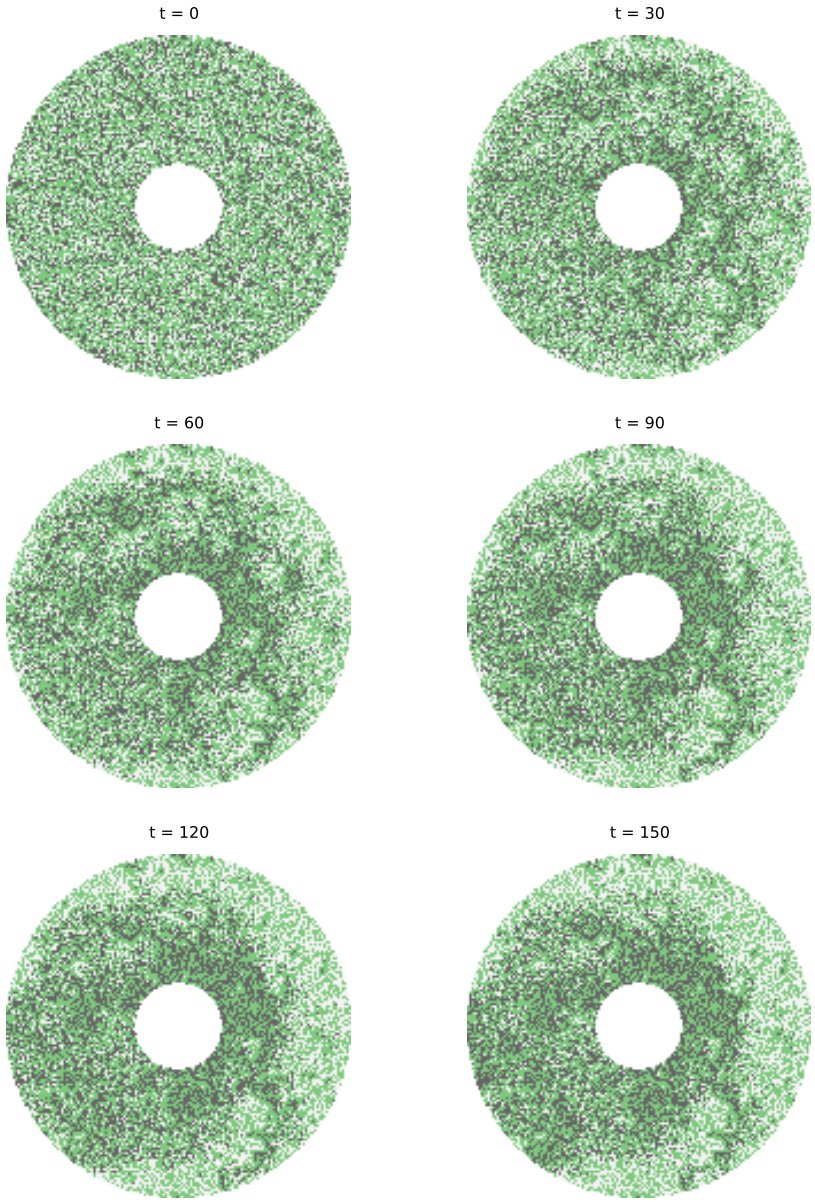


Figure A4: Sample subjected to *distorted* random walk shown at time-steps 0, 30, 60, 90, 120 and 150. The clay particles are grey, quartz is green and the pores are white.

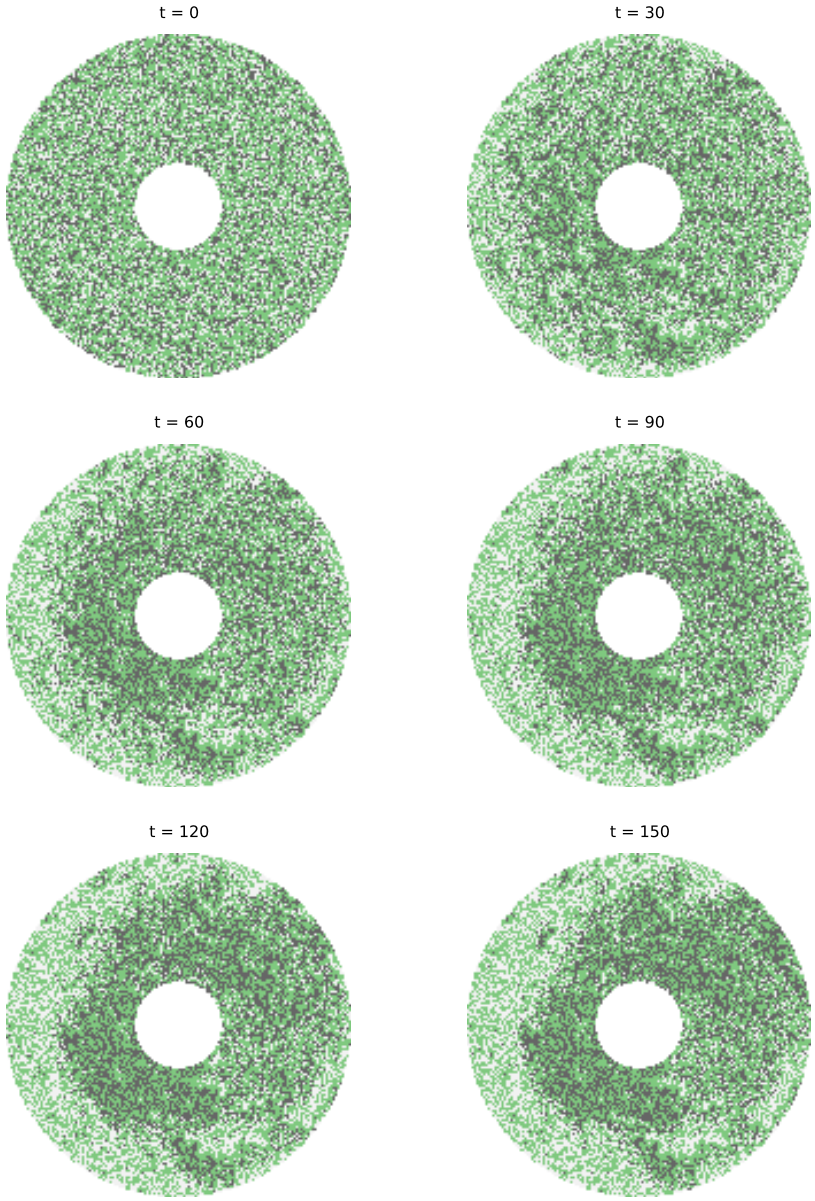


Figure A5: Sample subjected to *distorted* biased random walk shown at time-steps 0, 30, 60, 90, 120 and 150 and drawn in the opposite order compared to Figure A4. The clay particles are grey, quartz is green and the pores are white.

C Appendix - Code

C.1 Swelling.jl

```
module Swelling

using DataStructures
using StatsBase: Weights, sample, shuffle!
using LinearAlgebra: dot, normalize!, normalize, norm

export Annulus, SimulationSetup, simulate, initialize, HoshenKopelman, move_ann
export Particle, void, pore, quartz, clay


export Grid, initialize_neighborhood!, grid_neighborhood

@enum Particle begin
    void = -1
    pore = 0
    quartz = 1
    clay = 2
end

Base.zero(::Type{T}) where T <: Particle = void

mutable struct SimulationSetup
    n
    particle_distribution
    temperature
    stress_gradient
    outer_radius
    inner_radius
    max_swells
    radius_increase_per_swelling
    neighbor_effects
    λ
    fixed_λ
    dim
    metric
    σ
    diag

    function SimulationSetup(
        n::Integer,
        particle_distribution::Array{Float64,1},
        temperature::Float64,
        stress_gradient::Float64,
        outer_radius::Float64,
        inner_radius::Float64,
        max_swells::Integer,
        radius_increase_per_swelling::Float64,
        neighbor_effects::Bool,
        λ::Float64,
        fixed_λ::Bool,
        dim::Int = 2,
        metric::Symbol = :euclidean
        )
        σ = stress_gradient / (n / 2)
        if metric ==:euclidean
            diag = false
        elseif metric == :chebyshev
            diag = true
        else
            error("Unknown metric")
        end
```

```
        new(n, particle_distribution, temperature, stress_gradient,
            outer_radius, inner_radius,
            max_swells, radius_increase_per_swelling, neighbor_effects,
             λ, fixed_λ, dim, metric, σ, diag)
    end
end
function SimulationSetup(n::Integer, particle_distribution::Array{Float64,1})
    SimulationSetup(n, particle_distribution, 2.0, 50.0, 20.0,
                    5.0, 70, 1.01, false, 0.1, true)
end

function SimulationSetup(n::Integer, particle_distribution::Array{Float64,1},
                         dim::Int)
    SimulationSetup(n, particle_distribution, 2.0, 50.0, 20.0,
                    5.0, 70, 1.01, false, 0.1, true, dim)
end


include("Grid.jl")
include("Annulus.jl")
include("AnnulusUtils.jl")
include("HoshenKopelman.jl")
include("Simulate.jl")
include("Movement.jl")

end
```

C.2 Grid.jl

```
struct Region{D}
    mini::NTuple{D, Int}
    maxi::NTuple{D, Int}
end


struct Hood{D}
    whole::Region{D}
    βs::Vector{CartesianIndex{D}}
end

Base.length(r::Region{D}) where {D} = prod(r.maxi .- r.mini .+ 1)

function Base.in(idx, r::Region{D}) where {D}
    @inbounds for ϕ in 1:D
        r.mini[ϕ] <= idx[ϕ] <= r.maxi[ϕ] || return false
    end
    return true
end

struct Grid{D, T}
    s::Array{T ,D}
    metric::Symbol
    hoods::Dict{Float64,Hood{D}}
end

function Grid(
    d::NTuple{D,Int};
    fill::T = 0,
    metric::Symbol = :chebyshev,
) where {D, T}
    s = Array{T ,D}(undef, d)

    for i in eachindex(s)
```

```
    s[i] = zero(T)
    end

    return Grid{D, T}(
        s,
        metric,
        Dict{Float64,Hood{D}}(),
    )
end

Base.getindex(gs::Grid, i...) = gs.s[i...]
Base.setindex!(gs::Grid, x, i...) = gs.s[i...] = x
Base.CartesianIndices(gs::Grid) = CartesianIndices(gs.s)
Base.length(gs::Grid) = length(gs.s)
Base.iterate(gs::Grid, i...) = Base.iterate(gs.s, i...)

function initialize_neighborhood!(space::Grid{D, T}, r::Real) where {D, T}
    d = size(space.s)
    r0 = floor(Int, r)
    if space.metric == :euclidean
        # hypercube of indices
        hypercube = CartesianIndices((repeat([(-r0):r0], D)...,))
        # select subset of hc which is in Hypersphere
        βs = [β for β ∈ hypercube if norm(β.I) ≤ r && β.I != (0, 0) && β.I != (0, 0, 0)]
    elseif space.metric == :chebyshev
        βs = vec([CartesianIndex(a) for a in Iterators.product([(-r0):r0
                                    for ϕ in 1:D]...) if a != (0, 0) && a!= (0, 0, 0)])
    else
        error("Unknown metric type")
    end
    whole = Region(map(one, d), d)
    hood = Hood{D}(whole, βs)
    space.hoods[float(r)] = hood
    return hood
end


function grid_neighborhood(α::CartesianIndex, grid::Grid, r::Real = 1)
    hood = if haskey(grid.hoods, r)
        grid.hoods[r]
    else
        initialize_neighborhood!(grid, r)
    end
    _grid_neighborhood(α, hood)
end


function _grid_neighborhood(
    α::CartesianIndex,
    hood,
)
    return Iterators.filter(x -> x ∈ hood.whole, (Tuple(α + β) for β in hood.βs))
end
```

C.3 Annulus.jl

```
struct Annulus{D, T <: Real}

    particle_types::Grid{D, Particle}
    swelling_counts::Grid{D, Int}
    radiuses::Grid{D, Float64}
    inner_ring::Vector{CartesianIndex}
    n::Integer
```

```
    unit_radius::Float64
    outer_radius::T
    inner_radius::T
    metric::Symbol

    function Annulus(dims::NTuple{D, Int}, outer_radius::T, inner_radius::T,
                     metric::Symbol=:euclidean) where {D, T <: Real}
        particle_types = Grid(dims, fill=void, metric=metric)
        swelling_counts = Grid(dims, fill=0, metric=metric)
        radiuses = Grid(dims, fill=0.0, metric=metric)
        inner_ring = Vector{CartesianIndex}()
        n = dims[1]
        unit_radius = outer_radius / n
        new{D, T}(particle_types, swelling_counts, radiuses, inner_ring,
                  n, unit_radius, outer_radius, inner_radius, metric)
    end
end

function Annulus(n::Int, outer_radius::Real, inner_radius::Real,
                 dim::Int = 2, metric::Symbol=:euclidean)
    Annulus(ntuple(x -> n, dim), promote(outer_radius, inner_radius)..., metric)
end
Annulus(n::Int, dim::Int = 2) = Annulus(n, 20, 5, dim)
Annulus(ss::SimulationSetup) = Annulus(ss.n, ss.outer_radius, ss.inner_radius, ss.dim, ss.metric)


function sample_particle(weights::Weights)
    Particle(sample(weights)-1)
end


sample_radius(sampler::Sampleable) = rand(sampler)
sample_radius(constant_radius::Real) = constant_radius

initialize(ann::Annulus) = initialize(ann, [0.2, 0.4, 0.4])
initialize(ann::Annulus, ss::SimulationSetup) = initialize(ann, ss.particle_distribution)

function initialize(ann::Annulus, particle_distribution::Vector,
                    radius_sampler = Uniform(0.1, 0.9))
    _initialize(ann, particle_distribution, radius_sampler)
    empty!(ann.inner_ring)
    find_inner_ring(ann)
end

function _initialize(ann::Annulus, particle_distribution::Vector, radius_sampler)
    radius = floor(Integer, (ann.n) / 2)
    middle = ((ann.n + 1) / 2, (ann.n + 1) / 2)

    outer_radius_squared = radius^2
    inner_radius_squared = (ann.inner_radius / ann.outer_radius * radius)^2

    weights = Weights(particle_distribution)

    for I in CartesianIndices(ann.particle_types)

        ann.swelling_counts[I] = 0
        ann.radiuses[I] = 0.0

        x, y = Tuple(I)
        radius_squared = (x - middle[1])^2 + (y - middle[2])^2

        if inner_radius_squared < radius_squared <= outer_radius_squared
            particle_type = sample_particle(weights)
            ann.particle_types[I] = particle_type
            if particle_type == quartz || particle_type == clay
                ann.radiuses[I] = sample_radius(radius_sampler)
            end
```

```
        end
    end
end

function _initialize(ann::Annulus{3}, particle_distribution::Vector, radius_sampler)
    radius = floor(Integer, (ann.n) / 2)
    middle = ((ann.n + 1) / 2, (ann.n + 1) / 2)

    outer_radius_squared = radius^2
    inner_radius_squared = (ann.inner_radius / ann.outer_radius * radius)^2

    weights = Weights(particle_distribution)

    for I in CartesianIndices(ann.particle_types)
        ann.swelling_counts[I] = 0
        ann.radiuses[I] = 0.0

        x, y, z = Tuple(I)
        radius_squared = (x - middle[1])^2 + (y - middle[2])^2

        if inner_radius_squared < radius_squared <= outer_radius_squared
            particle_type = sample_particle(weights)
            ann.particle_types[I] = particle_type
            if particle_type == quartz || particle_type == clay
                ann.radiuses[I] = sample_radius(radius_sampler)
            end
        end
    end
end

function find_inner_ring(ann::Annulus{2})
    i = ceil(Integer, ann.n / 2)
    j = i

    indexes_to_check = [(i, j)]
    checked_indexes = Set([])

    while length(indexes_to_check) > 0
        i, j = pop!(indexes_to_check)

        if ann.particle_types[i, j] == void
            for x in i - 1:i + 1
                for y in j - 1:j + 1
                    if (x, y) ∉ checked_indexes
                        push!(indexes_to_check, (x, y))
                        push!(checked_indexes, (x, y))
                    end
                end
            end
        else
            push!(ann.inner_ring, CartesianIndex(i, j))
        end
    end
end

function find_inner_ring(ann::Annulus{3})
    i = ceil(Integer, ann.n / 2)
    j = i

    indexes_to_check = [(i, j)]
    checked_indexes = Set([])

    z = 1

    while length(indexes_to_check) > 0
        i, j = pop!(indexes_to_check)
```

```
        if ann.particle_types[i, j, z] == void
            for x in i - 1:i + 1
                for y in j - 1:j + 1
                    if (x, y) ∉ checked_indexes
                        push!(indexes_to_check, (x, y))
                        push!(checked_indexes, (x, y))
                    end
                end
            end
        else
            push!(ann.inner_ring, CartesianIndex(i, j, z))
        end
    end

    z_inners = Vector{CartesianIndex}()
    for inner in ann.inner_ring
        z_inners = vcat(z_inners, [CartesianIndex(Tuple(inner)[1],
                        Tuple(inner)[2], z) for z in 2:ann.n])
    end
    push!(ann.inner_ring, z_inners...);
    unique!(ann.inner_ring)
end

calculate_area(ann::Annulus) = sum(π .* (ann.radiuses .* ann.unit_radius).^2)
calculate_max_area(ann::Annulus) =   π * (ann.outer_radius^2 - ann.inner_radius^2)
calculate_max_area(ann::Annulus{3}) = ann.n * π * (ann.outer_radius^2 - ann.inner_radius^2)
calculate_clay_area(ann::Annulus) = sum(π .* (ann.radiuses .* ann.unit_radius
                                        .* (ann.particle_types .== clay) ).^2)
```

C.4 Simulate.jl

```
function swelling_probability(radius::Float64, temperature::Float64, σ::Float64)
    return exp(-1 * radius^2 / (temperature * σ))
end

function check_neighbour_swelling(I::CartesianIndex, ann::Annulus)
    neigs = grid_neighborhood(I, ann.swelling_counts)
    near_swelling = sum(I -> ann.swelling_counts[CartesianIndex(I)], neigs)
    near_swelling / length(collect(neigs))
end

function check_neighbour_porosity(I::CartesianIndex, ann::Annulus)
    neigs = grid_neighborhood(I, ann.swelling_counts)
    near_porosity = count(I -> ann.particle_types[CartesianIndex(I)] == pore, neigs)
    near_porosity / length(collect(neigs))
end


function simulate(ann::Annulus, ss::SimulationSetup, max_iter::Int = 200)

    areas = []

    for _ in 1:max_iter

        calculate_area(ann) > calculate_max_area(ann) && break
        uniq = unique(ann.swelling_counts)
        ss.max_swells ∈ uniq && length(uniq) == 2 && break

        Threads.@threads for I in CartesianIndices(ann.swelling_counts)
            if ann.particle_types[I] == clay && ann.swelling_counts[I] < ss.max_swells
                swell_prob = swelling_probability(ann.radiuses[I], ss.temperature, ss.σ)

                if ss.neighbor_effects
```

```
                    neighbour_swelling = check_neighbour_swelling(I, ann)
                    neighbour_porosity = check_neighbour_porosity(I, ann)
                    swell_prob = (swell_prob
                                  * (1 - (neighbour_swelling / ss.max_swells))
                                  * (1 + neighbour_porosity))
                end

                if rand() < swell_prob
                    ann.radiuses[I] *= ss.radius_increase_per_swelling
                    ann.swelling_counts[I] += 1
                end
            end
        end
        push!(areas, calculate_area(ann))
    end
    return areas
end
```

C.5 Movement.jl

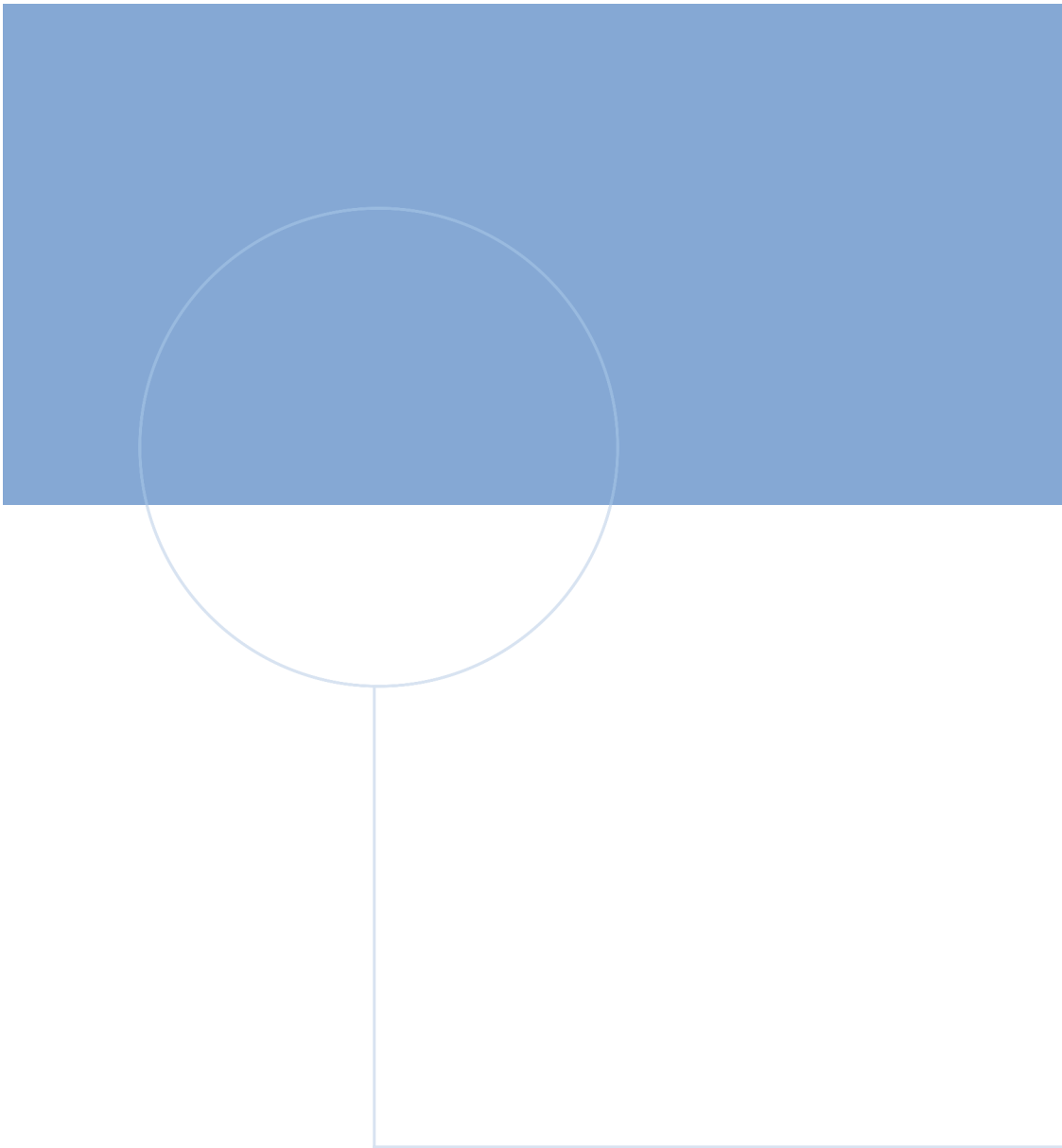
```
abstract type Walk end
struct randomWalk <: Walk end
struct biasedWalk <: Walk end

function move_particle!(from::CartesianIndex, to::CartesianIndex, ann::Annulus)
    ann.particle_types[to] = ann.particle_types[from]
    ann.particle_types[from] = pore

    ann.radiuses[to] = ann.radiuses[from]
    ann.radiuses[from] = 0.0

    ann.swelling_counts[to] = ann.swelling_counts[from]
    ann.swelling_counts[from] = 0
end

function possible_locations(from::CartesianIndex, ann::Annulus)
    [
        neighbor
        for neighbor in grid_neighborhood(from, ann.particle_types)
        if ann.particle_types[CartesianIndex(neighbor)] == pore
    ]
end

function force_vector(position::CartesianIndex, ann::Annulus{2})
    center = (ann.n/2, ann.n/2)

    force_direction = center .- position.I
    force_direction = normalize(collect(force_direction))
    force_direction
end

function force_vector(position::CartesianIndex, ann::Annulus{3})
    center = (ann.n/2, ann.n/2, position.I[3])

    force_direction = center .- position.I
    force_direction = normalize(collect(force_direction))
    force_direction
end


function choose_location(locations)
    len = length(locations)
    prob_dirs = fill(1/len, len)
```

```
    to_index = sample(Weights(prob_dirs))
    locations[to_index]
end

function choose_location(locations, from, ann::Annulus, ss::SimulationSetup)

    force_dir = force_vector(from, ann)
    prob_dirs = Vector{Float64}(undef, 0)

    for loc in locations
       push!(prob_dirs, exp(ss.σ * dot(normalize(collect(loc .- from.I)), force_dir)))
    end

    normalize!(prob_dirs, 1)
    to_index = sample(Weights(prob_dirs))
    to = locations[to_index]

end


move_ann(ann::Annulus, ss::SimulationSetup) =  move_ann(ann, ss, randomWalk)

function move_ann(ann::Annulus, ss::SimulationSetup, walk::Type{T}) where {T<:randomWalk}
    clays = [I for I in CartesianIndices(ann.particle_types)
            if ann.particle_types[I] == clay]
    shuffle!(clays)
    for from in clays
        if rand() > ss.λ
            locations = possible_locations(from, ann)
            length(locations) == 0 && continue
            to = choose_location(locations)

            move_particle!(from, CartesianIndex(to), ann)
        end
    end
end

function move_ann(ann::Annulus, ss::SimulationSetup, walk::Type{T}) where {T<:biasedWalk}
    clays = [I for I in CartesianIndices(ann.particle_types)
             if ann.particle_types[I] == clay]
    shuffle!(clays)
    for from in clays
        if ss.fixed_λ
            if rand() > ss.λ
                locations = possible_locations(from, ann)
                length(locations) == 0 && continue
                to = choose_location(locations, from, ann, ss)

                move_particle!(from, CartesianIndex(to), ann)
            end
        else
            if rand() > ss.λ * exp(-1 * ss.σ)
                locations = possible_locations(from, ann)
                length(locations) == 0 && continue
                to = choose_location(locations, from, ann, ss)

                move_particle!(from, CartesianIndex(to), ann)
            end
        end
    end
end
```

C.6 HoshenKopelman.jl

```
HoshenKopelman(ann::Annulus; ss::SimulationSetup) = HoshenKopelman(ann, [clay], ss)

function HoshenKopelmanNeighbours(label_grid::AbstractArray{Int64},
                                  x::Int, y::Int, diag::Bool)
    W = y > 1 ? label_grid[x, y - 1] : 0
    N = x > 1 ? label_grid[x - 1, y] : 0

    if diag
        NW = (x > 1 && y > 1) ? label_grid[x - 1, y - 1] : 0
        SW = (x < size(label_grid, 1) && y > 1) ? label_grid[x + 1, y - 1] : 0
        return (W, N, NW, SW)
    end
    return (W, N)
end

function HoshenKopelmanNeighbours(label_grid::AbstractArray{Int64},
                                  x::Int, y::Int, z::Int, diag::Bool)
    W = y > 1 ? label_grid[x, y - 1, z] : 0
    N = x > 1 ? label_grid[x - 1, y, z] : 0
    D = z > 1 ? label_grid[x, y, z - 1] : 0

    if diag
        NW = (x > 1 && y > 1) ? label_grid[x - 1, y - 1, z] : 0
        SW = (x < size(label_grid, 1) && y > 1)
              ? label_grid[x + 1, y - 1, z] : 0

        DNW = (x > 1 && y > 1 && z > 1) ? label_grid[x - 1, y - 1, z - 1] : 0
        DN = (x > 1 && z > 1 ) ? label_grid[x - 1, y, z - 1] : 0
        DNE = (x > 1 && y < size(label_grid, 2) && z > 1 ) ?
             label_grid[x - 1, y + 1, z - 1] : 0
        DE = (y < size(label_grid, 2) && z > 1) ?
             label_grid[x, y + 1, z - 1] : 0
        DSE = (x < size(label_grid, 1) && y < size(label_grid, 2) && z > 1) ?
             label_grid[x + 1, y + 1, z - 1] : 0
        DS = (x < size(label_grid, 1) && z > 1) ?
             label_grid[x + 1, y, z - 1] : 0
        DSW = (x < size(label_grid, 1) && y > 1 && z > 1) ?
             label_grid[x + 1, y - 1, z - 1] : 0
        DW = (y > 1 && z > 1) ? label_grid[x, y - 1, z - 1] : 0

        return (W, N, NW, SW, DNW, DN, DNE, DE, DSE, DS, DSW, DW)
    end

    return (W, N, D)
end

function HoshenKopelman(ann::Annulus, particles::AbstractVector{Particle},
                        ss::SimulationSetup)
    labels = IntDisjointSets(0)
    label_grid = fill(0, size(ann.particle_types.s))

    for I in eachindex(label_grid)
        if ann.particle_types[I] in particles
            label_grid[I] = 1
        end
    end


    @inbounds for I in CartesianIndices(label_grid)
        if label_grid[I] != 0
            ngs = HoshenKopelmanNeighbours(label_grid, I.I... , ss.diag)
            if sum(ngs) == 0
                label_grid[I] = push!(labels)
            elseif count(!isequal(0), ngs) == 1    # Check exactly one neighbour has label
                label_grid[I] = sum(ngs)           # Only one is nonzero
            else
                ngs = filter(!isequal(0), ngs)
```

```
                label_grid[I] = min(ngs...)
                for i in 1:length(ngs) - 1
                    union!(labels, ngs[i], ngs[i + 1])
                end
            end
        end
    end

    @inbounds for I in eachindex(label_grid)
        if label_grid[I] != 0
            label_grid[I] = find_root!(labels, label_grid[I])
        end
    end

    return label_grid

end
```



NTNU

Norwegian University of
Science and Technology