THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

JOHN AND WILLIE LEONE FAMILY DEPARTMENT OF ENERGY AND MINERAL ENGINEERING

STUDYING INJECTIVITY OF CO2 INTO SUBSURFACE FORMATIONS FOR GEOLOGIC SEQUESTRATION USING MRST
CYCLIC INJECTION VS. SIMULTANEOUS INJECTION

ISHTIAQUL ISLAM
SPRING 2021

A thesis
submitted in partial
fulfillmentof the
requirements
for a baccalaureate degree
in Petroleum and Natural Gas Engineering
with honors in Petroleum and Natural Gas Engineering

Reviewed and approved* by the following:

Sanjay Srinivasan
Head, John and Willie Leone Family Department of Energy and Mineral Engineering
Professor of Petroleum and Natural Gas Engineering
Thesis Supervisor

Eugene Morgan
Associate Head for Undergraduate Education
Assistant Teaching Professor
Honors Adviser

* Electronic Approvals on file

ABSTRACT

Reduction in global greenhouse gas concentration in the atmosphere is one of the key challenges confronting mankind that threatens our survival. Reducing carbon dioxide from the atmosphere, which is one of the key contributors to global warming is thus an important focus. As of today, given the demand for energy and the unavailability of technology for the direct conversion of CO2, storing the gas somewhere beneath the Earth's surface is always an option, possibly, currently the best option at our disposal for reducing the greenhouse gas concentration (1).

There are studies that involve optimizing CO2 sequestration, but most of which involve simultaneous and continuous injections. Doing so will lead the reservoir to reach undesirable pressures within a short period of time. Therefore, in this study, we are analyzing the pressure trend using cyclic injection and comparing the results against simultaneous injections while holding other factors constant. The simulation will run cases of both simultaneous and cyclic injections using four vertical injectors in a closed-flow boundary reservoir. The location, rates, and duration of the injector wells have been optimized to obtain greater results in both cases.

The injectors are operated until the bottom hole pressures reach values of fracture pressure. The results obtained showed that the cyclic injection is better at injecting CO2 for a longer duration of time in comparison to simultaneous injection. This yields a greater volume of CO2 injected at maintainable pressures.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Firstly, I would like to thank my family for sacrificing so much to get me here, my professors who have given me the knowledge and skills I needed to enter the real world. I am thankful for my friends who have supported me. They have helped me to reach heights I never thought of reaching on my own.

I would like to especially thank Dr. Sanjay Srinivasan for being a great instructor and mentor. He has been very patient with my setbacks and continued to support me through this journey.

Last, I would like to acknowledge SINTEF for developing the MRST library and generously making it accessible to the public. This has helped me further my knowledge in my field of interest. Knowledge should be made free and available. I have attached the source code used to do this study, in Appendix B. It's important to note that the code is copyrighted and belongs to SINTEF. The source code only has been modified to fit the needs of the study.

Chapter 1

Introduction

CO_2 sequestration is a process via which CO_2 collected from the atmosphere or industrial sources is injected into subsurface geologic formations. The goal of sequestration is to mitigate the climate impacts from greenhouse gas emission. One of the main problems is excessive pressure build-up in the formation. High pressure build-up in the reservoir or sub-surface formations creates potential fractures and has negative impact on the integrity of the caprock, creates fault slippage, and even allows leakage of fluids such as brine and CO_2 to transport into the nearby freshwater source. This may negatively impact the surrounding environment, and also the nearby community. Therefore, the pressures of operation of CO2 sequestration must be maintained without disturbing the formation (6).

This study will cover optimization of CO_2 injection and storage all while being able to maintain pressures below fracture pressures. It will focus primarily on differentiating bottomhole pressure outcomes expected from simultaneous versus cyclic injections using vertical injectors in a typical sandstone formation. The simultaneous well injection case (our base case) will involve having all four wells injecting simultaneously for a certain duration of time whereas our opponent case will involve cyclic well injections where the four wells are operated at different time intervals for the same duration of time each. Both processes will be done over the same period of time, however, the cyclic injection will have the first well to be re-operated for additional 2 years. To achieve the desired pressure goal (making sure not to fracture the

formation), the quantity, the injection rates, the location, and the schedules of the injectors are adjusted. This is done in an attempt to optimize CO_2 storage while minimizing the potential risks.

It is important to note this study contains a mixture of S.I. units and field units. For reference, unit conversion is listed at the end of the paper.

1.1: Simulator Description

The simulator we are using for this project is MRST (MATLAB Reservoir Simulation Toolbox) developed primarily by the Computational Geosciences group in the Department of Mathematics and Cybernetics at SINTEF Digital. MRST offers a wide range of data structures and computational methods that, when combined, can be customized according to the user's needs. MRST is organized into a minimal core module consisting of basic data structures and functionality, and a large set of add-on modules that consist of discretization, built-in solvers, physical models, and a wide variety of simulators and workflow tools (3).

MRST is ideal for CO_2 sequestration and is very convenient to use as numerous cases can be run by simply adjusting the codes on MATLAB. To run the simulation, one must download MRST on Sintef.no website and extract the downloaded folder. Once done, one must then type "startup" in the command window to import the MRST library. Once the library is imported, any simulation using the appropriate functions and variables known within the MRST library can be run.

MRST uses S.I. units, thus all the calculations and equations are in as such.

1.2: Testing Simulator Credibility and Capacity

1.2.1: Basic Reservoir Example (SINTEF)

The basic 3D example has been taken from the sample examples made by SINTEF. The objective of the model is to inject CO_2 at the bottom of the reservoir that is initially filled with brine. This is the model which we have used as our base to build upon according to the needs of the entire study. This section of this paper describes the pressure profile observed from injecting CO_2 for two years followed by a post-injection period of another two years (well shut-in).

There is only one well that is situated in the center of the reservoir grid and is injecting to the very bottom of the reservoir at the depth of 1600m. The reservoir is initially filled with brine. This brine is displaced by injecting CO_2 at the bottom of the reservoir which is located at a depth of 1600 m. The top of the aquifer is located at a depth of 1500 m. This is done in an attempt to observe the bottomhole pressure trend at the bottom of the injector wells.

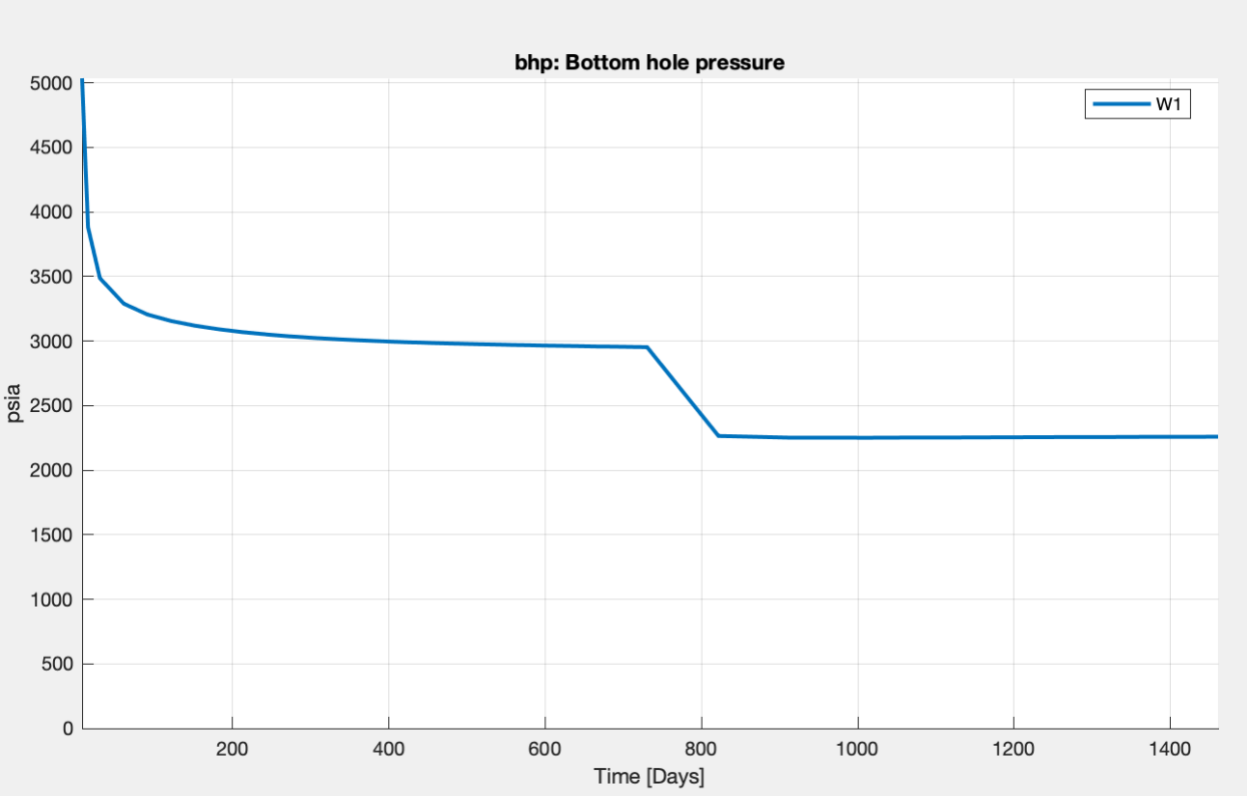

Figure 1: BHP (psia) profile for the unmodified version of the basic 3D example (SINTEF) - center injector

The initial reservoir pressure (based on hydrostatic considerations) is 2250 psia and so the rise in pressure to 3000 psia represents the pressure in the reservoir due to the successful injection and spreading of CO2 from the well. Once the well is shut in, the bhp dips to hydrostatic pressure (initial pressure). Assuming hydrostatic gradient of 0.433 psi/ft, the initial reservoir pressure is computed to be 2250 psia.

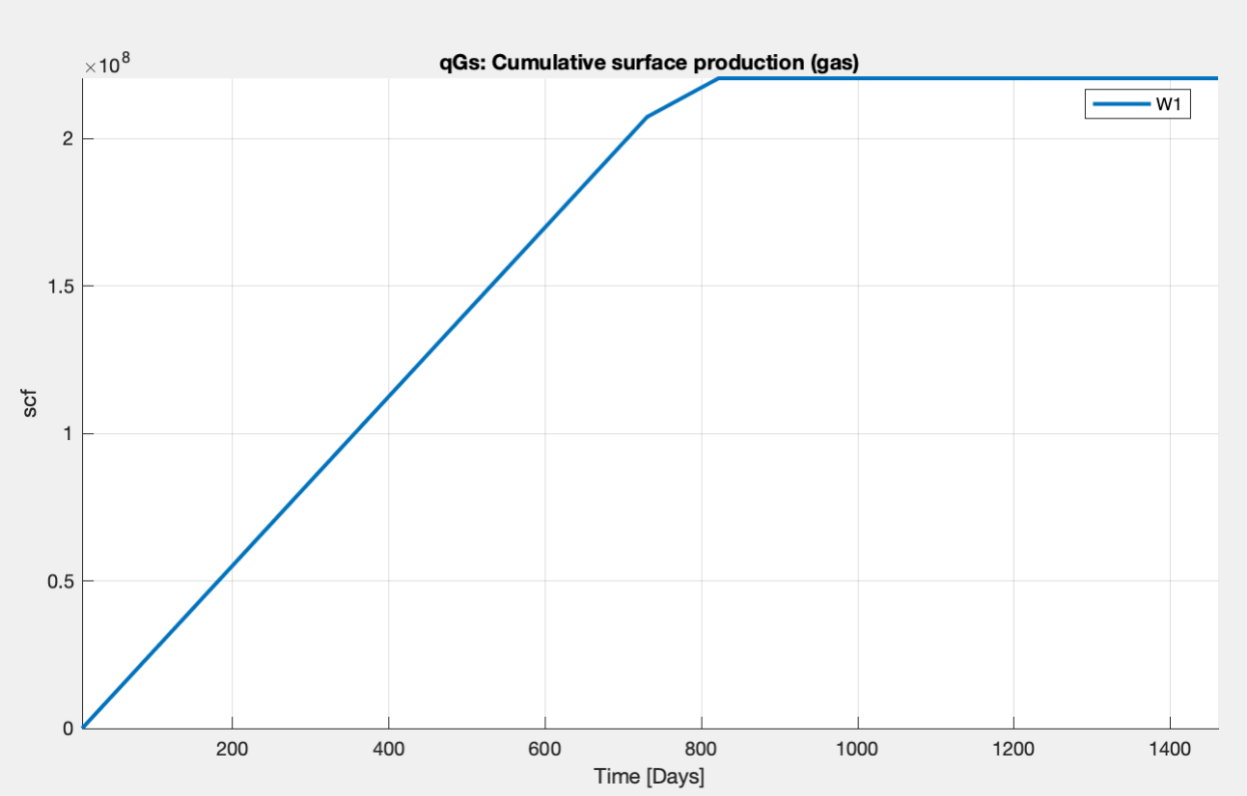

Figure 2: Cumulative Surface Injection Volume of CO2 (SCF) in the unmodified version of the basic 3D example (SINTEF) - center injector

The injection of CO_2 rises to around 2.25e+8 SCF for the first two years and then flattens out once the well is shut in. The objective of these graphs is to illustrate the viability and credibility of the simulation in accordance with what is expected in a heterogeneous, fairly porous formation.

It is also important to note that despite the process being an injection, the title depicts surface production of gas which is a small error in MRST.

1.2.2: Basic Reservoir Example (SINTEF) with closed-flow boundaries

The previous results (and the default in MRST) is for the boundaries to be open, i.e., the CO2 plume can spread unimpeded by the presence of boundaries. In this alternate case, the boundary conditions are adjusted so that the water initially present inside the reservoir has no path to escape. We observe a different trend in the bhp curve for the injector well. The pressure starts building up during the injection phase to approximately 6000 psia and then decreases to 5000 psia when the injection is stopped.

For the cases in this study, we are going to be using closed-flow boundaries but also make the reservoir sufficiently big so that the wells do not sense the presence of the boundary until after sufficient volume of CO2 is injected. This is likely to mimic the performance of CO_2 into a large aquifer zone. It seems MRST is ready to be used for our purpose.

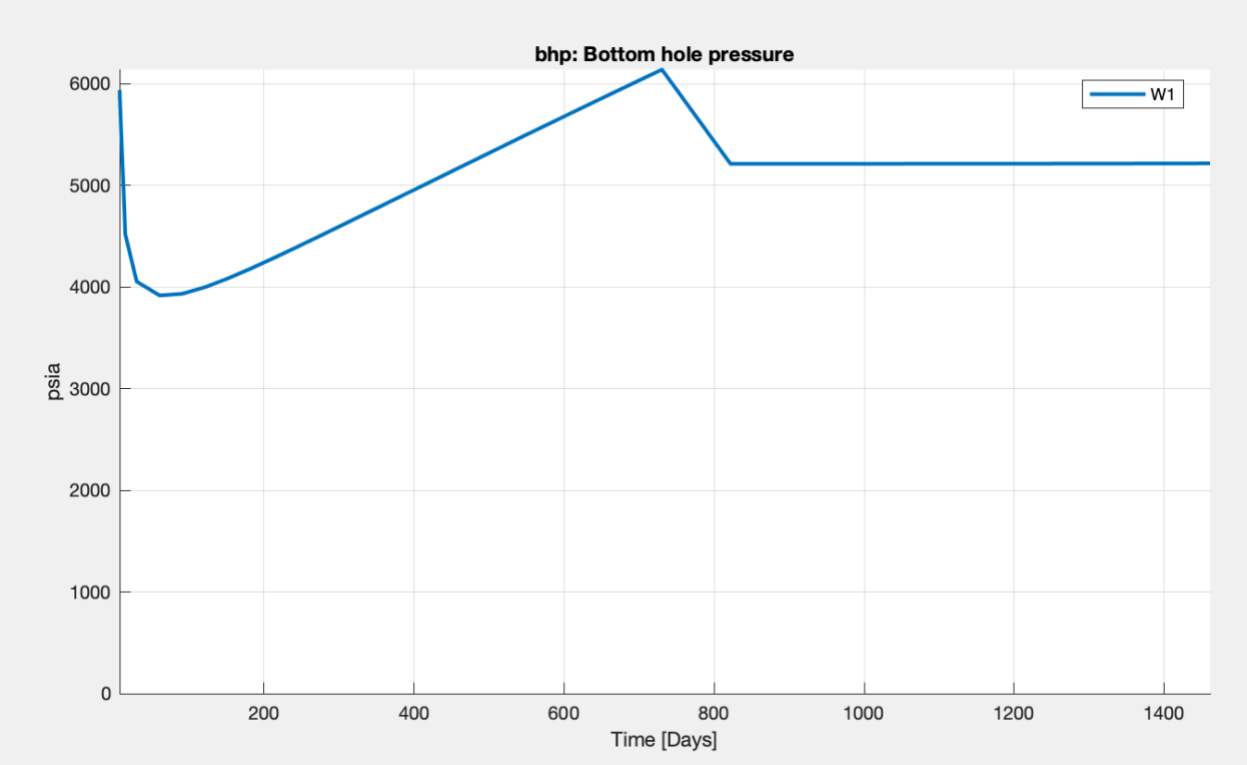

Figure 3: BHP (psia) profile for the basic 3D example (SINTEF) with no-flow boundaries - center injector

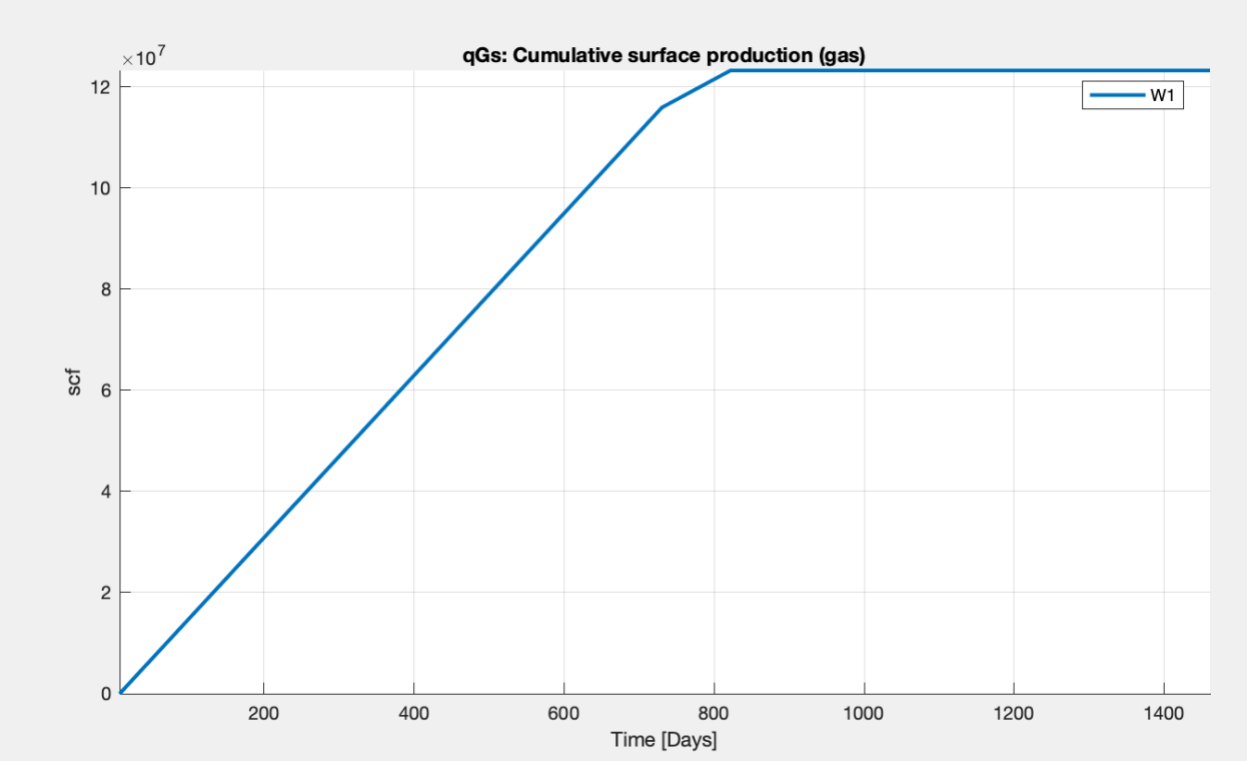

Figure 4: Cumulative Surface Injection Volume of CO2 (SCF) in the basic 3D example (SINTEF) - center injector

We also notice that the volume of CO_2 injected seems to flatten out at around 1.2E+8 SCF which is half of what we had previously observed. This is due to the immobility of irreducible water that does not have much room to escape (no-flow boundaries).

Based on this example created by the SINTEF team, we can finally build our reservoir model to fit our needs. In the next chapters, we will be playing around with the cases to draw distinctions between the performances of simultaneous versus cyclic injections.

Chapter 2

Methodology

2.1: Base Reservoir Model (Modified Version)

The model has been modified to our needs and is acting as the base to perform the reservoir simulation cases. The model for the reservoir we are running our MRST simulation consists of a rectangular grid of 35 x 35 x 20 blocks in the x-, y-, and z-directions respectively. The lengths in the x-, y-, and z-directions are 8,500 m, 8,500 m, and 100 m respectively. Our model reservoir grid has been modified to accommodate the four wells we have for injectors. The grid blocks around the wells are made smaller in comparison to the grid blocks near the edge of the reservoir. This is done in an attempt to recreate an infinite-boundary acting model. The grid blocks around the injector wells have dimensions 50 x 50 m in the x and y directions, while the intermediate grid blocks between the ones around the wells and the ones near the edges are of dimensions 200 x 200 m in the x and y directions. Finally, the boundary grid blocks are of dimensions 500 x 500 m. The thickness of each grid block is that of the layer in which it resides. The layers are 5 m each, which makes the total thickness of the reservoir to be 100 m (20 layers).

The depth of the aquifer is chosen to be as low as 1500 m to avoid complexities of pressure and temperature on the injected gas phase and volume. The reservoir is also heterogeneous meaning varying permeability and porosity with locations to make the model more realistic. Permeability in the x-, y-, and z-directions are considered the same for our intent and purposes (thus isotropic).

The permeability is calculated using the Kozeny-Carman equation that varies with porosity. The porosity values are randomized between 0.2 and 0.3, and the permeability values are calculated based on that. At depths between 1500 to 1600 meters, the initial pressure ranges between 1.471e-7 and 1.565e-7 Pascal (across the depth). The initial pressures are due to the hydrostatic column in the formation.

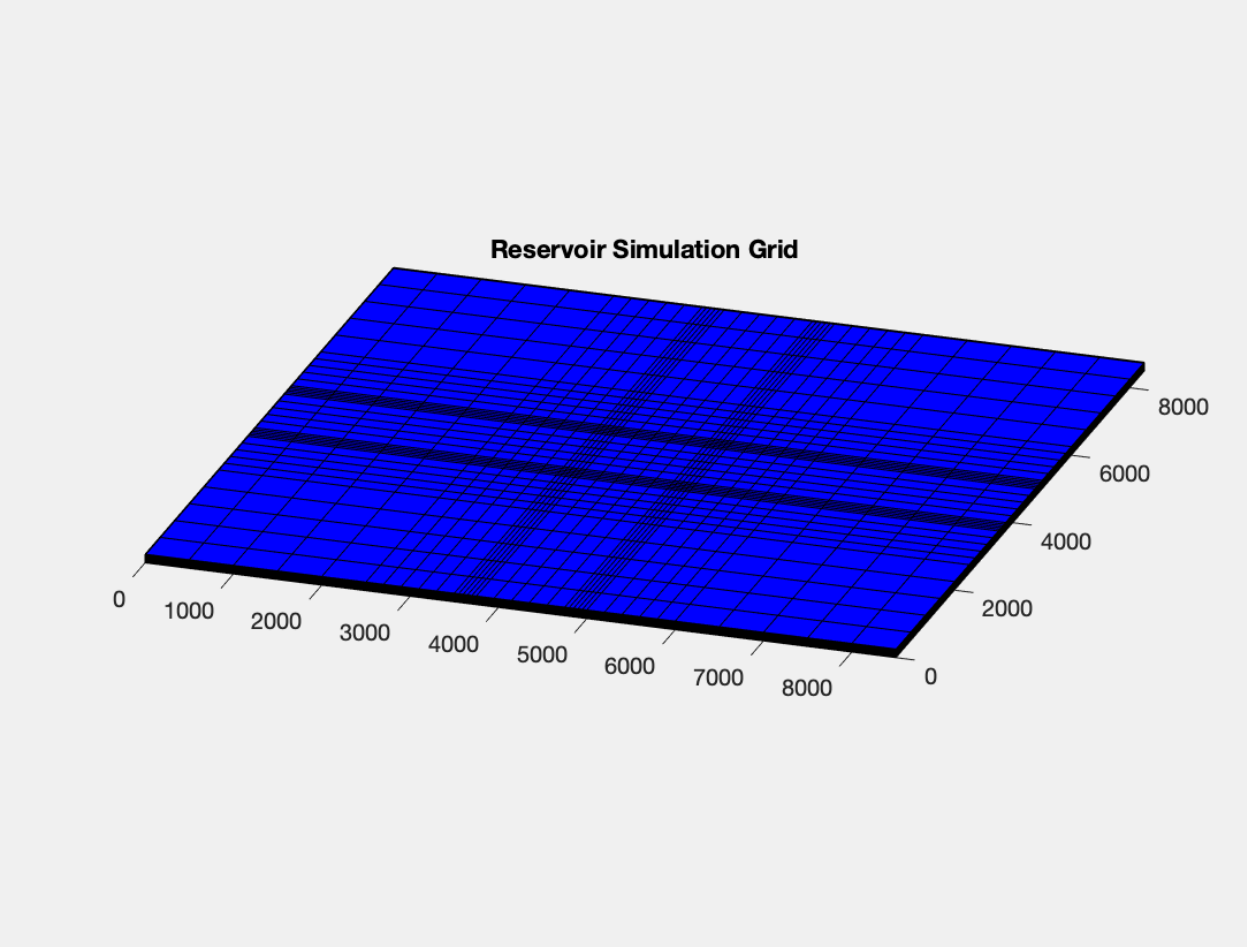

Figure 5: Reservoir Simulation Grid (No-flow boundaries; infinite-acting)

2.2: Fluid Model and Initial Conditions

The reference pressure and temperature chosen are 2175 psia and 343.15 Kelvin. The density and the compressibility of CO_2 can then be calculated. The reference pressure and temperature are just there to simulate the plume distribution of the injected CO_2.The density and the compressibility of brine are 1000 kg/m3 and 0. The viscosity of water is chosen to be 8e-4 Pa.s while the viscosity of CO_2 is calculated at the reference pressure and temperature. The rock compressibility is set at 4.35e-10 Pa^-1. The petrophysical data imported should reflect that of a typical sandstone formation.

The fluid saturation of water is given 0.27 (irreducible) and that of carbon dioxide gas is given 0.20 (residual), based on which the relative permeability curves of water and gas are generated. The process through which the saturations change throughout the simulation is based on Brooks-Corey relative permeability while the capillary pressure is generated using Leverett J-function. These are the default correlations used by MRST SINTEF.

Table 1: Reservoir and fluid properties

Reservoir Reference Pressure	2175 psia (15000000 Pa)
Reference Top Depth	1500 m
Reservoir Thickness Total	100 m
Reservoir Temperature	343.15 K
Maximum Residual Gas Saturation	0.2
Residual Water Saturation	0.27
Reservoir Dimensions	35 x 35 x 20
Total Injection Time	4 years
Total Simulation Time (including post-injection period)	6 years
Number of Injectors	4
Rock Compressibility	4.35e-10 Pa^-1

2.3: Injector Wells

We have chosen 4 vertical injector wells each with an injection rate that ensures the greatest CO_2 injection over a span of time. Thus, the variables we have had to adjust were the injection rates and the time period of the injection.

We also have to make sure that storing CO_2 does not cause fractures in the formation, thus the rates of injections and period of injection would need to be adjusted accordingly.

The positions of the wells can be seen below on the grid (7x7), Figure 6, which has been scaled 5 times smaller than the actual size (35x35). The actual grid has 35 blocks in the x-direction, 35 in the y-direction, and 20 in the z-direction. In Figure 6, the red blocks are finer-grained (50 m in x and y; 4 red blocks for 4 injector wells), blue blocks transition between fine and course (200 m in x and y), and green blocks are coarse-grained (500 m in x and y). This imitates the effect of an infinite reservoir.

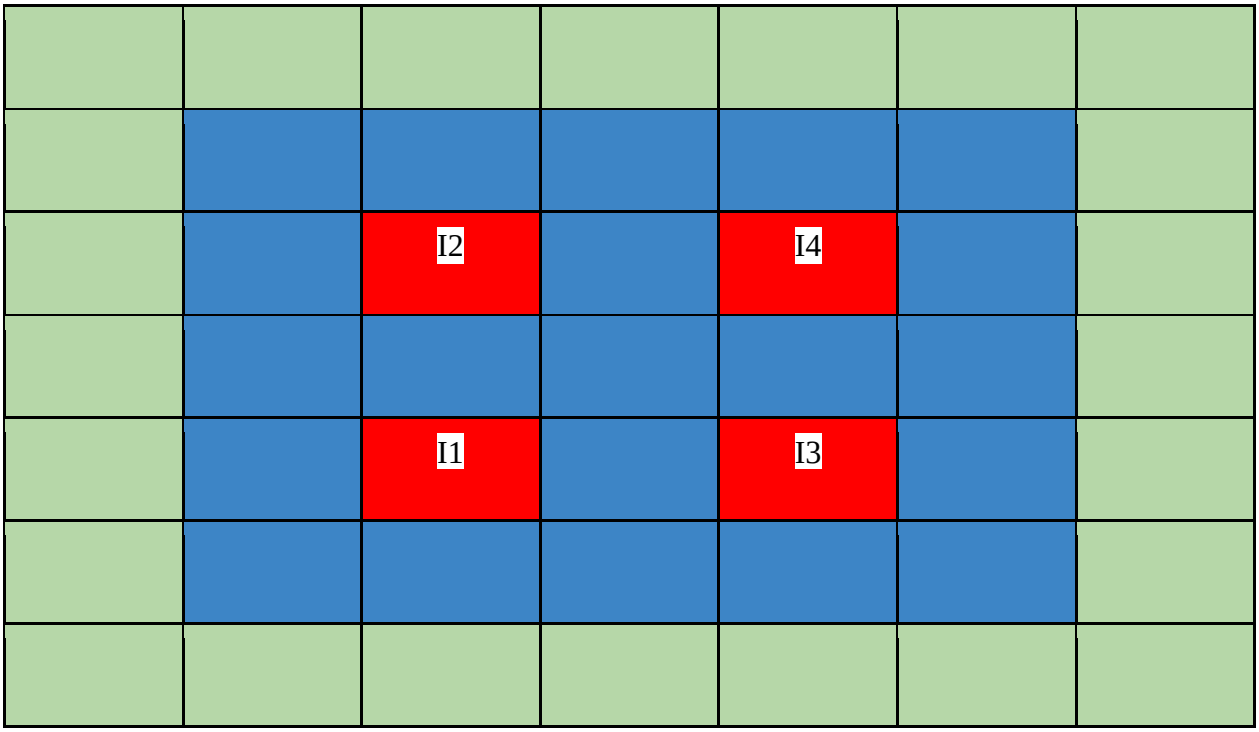

Figure 6: Reservoir Grid with injector wells (scale reduced by a factor of 5)

2.4: Fracture Gradient Calculation

It's important for our simulated formation to not incur fracture problems, thus we have to make sure that the pressures our simulation is operating under do not exceed a certain range. This is the most important constraint we have on the process of CO_2 sequestration. Thus, we would need to use realistic petrophysical values of the formation to calculate the fracture gradient.

From prior petroleum engineering course knowledge, we have decided to use the Ben Eaton Method in order to calculate the fracture gradient and hence the corresponding fracture pressure at every 5 m depth interval between 1500 m and 1600 m. First, we need to obtain the overburden gradient, S/D from the overburden stress gradient chart, Figure 7.

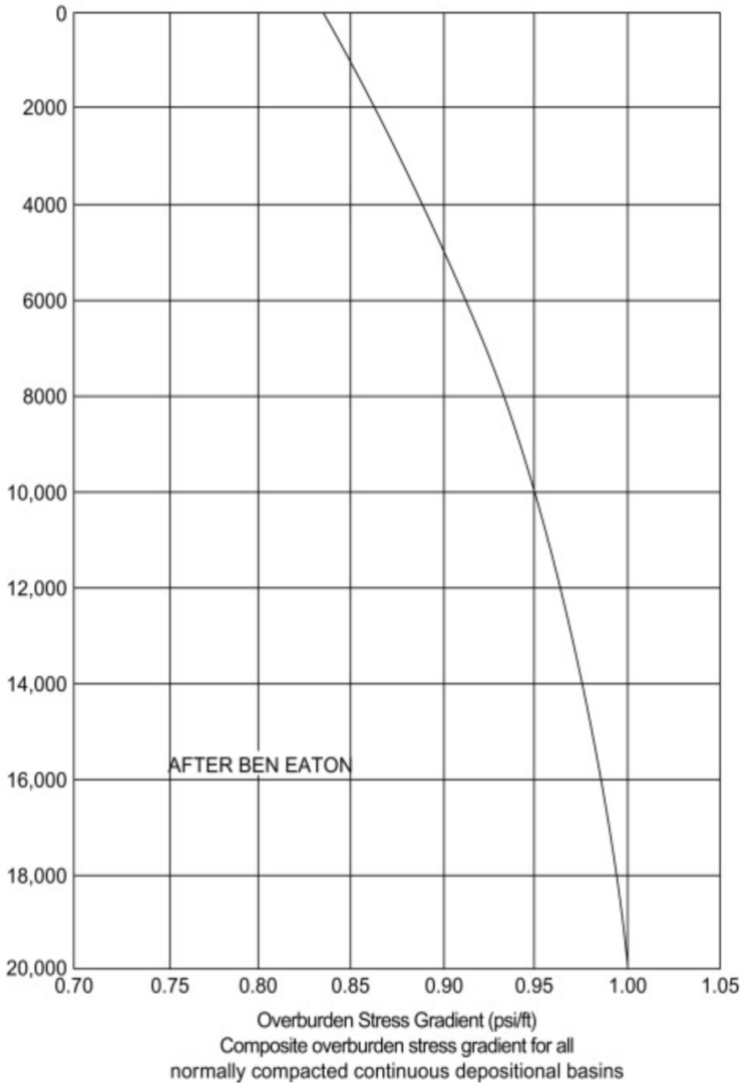

Figure 7: Overburden Stress Gradient Chart

After which, we have assumed a pore pressure gradient, Pf/D of 0.657 psi/ft given the depth difference is not too much and the value is typical in a sandstone formation such as this.

The Poisson's ratio, v is taken from the Poisson's ratio chart, Figure 8, each depth interval.

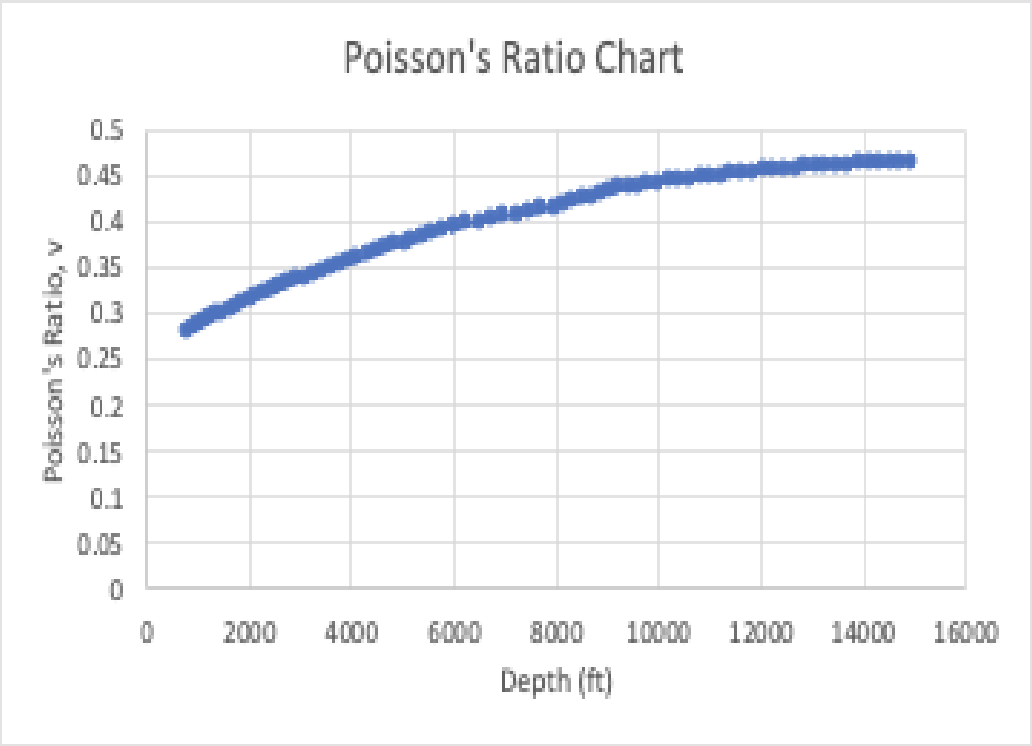

Figure 8: Poisson's Ratio Chart

We finally use the Fracture Gradient equation below to calculate the fracture gradient given the appropriate steps above mentioned have been properly executed (5). Fracture Pressures can be calculated by multiplying the fracture gradient with the depth of interest.

The fracture pressures yielded results from 2.8e+7 to 3.03e+7 Pascal (4061 – 4393 psia) with increasing depth. Therefore, the base of the bhp of the injector wells should not go beyond this range. Since the injectors are injecting at the bottom of the reservoir (the last layer), it's safe to operate the bhp of the injectors below 4300 psia without disturbing the formation. The pressure gradient window can be seen in the figure below.

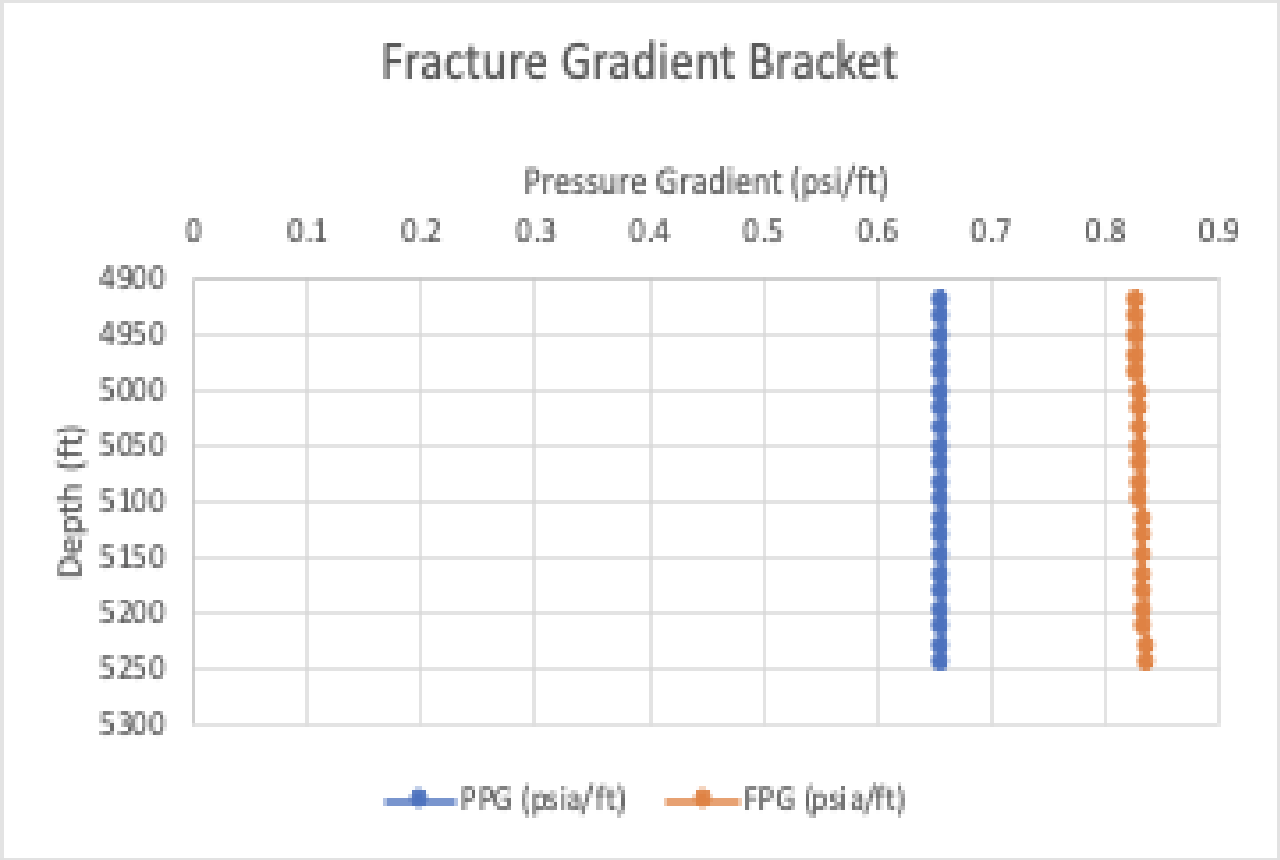

Figure 9: Fracture Gradient Window

$$F = \left(\frac{S}{D} - \frac{P_f}{D}\right) * \left(\frac{\mu}{1-\mu}\right) + \left(\frac{P_f}{D}\right)$$

Where:

$\frac{S}{D}$ = Overburden Gradient, psi/ft

$\frac{P_f}{D}$ = Formation Pressure Gradient at depth of interest, psi/ft

μ = Poisson's Ratio

Chapter 3

Results and Discussion

3.1: Simultaneous Injection Case (Closed Flow Boundaries)

After successfully calculating fracture pressures, we have simulated the simultaneous injection using the four vertical wells.

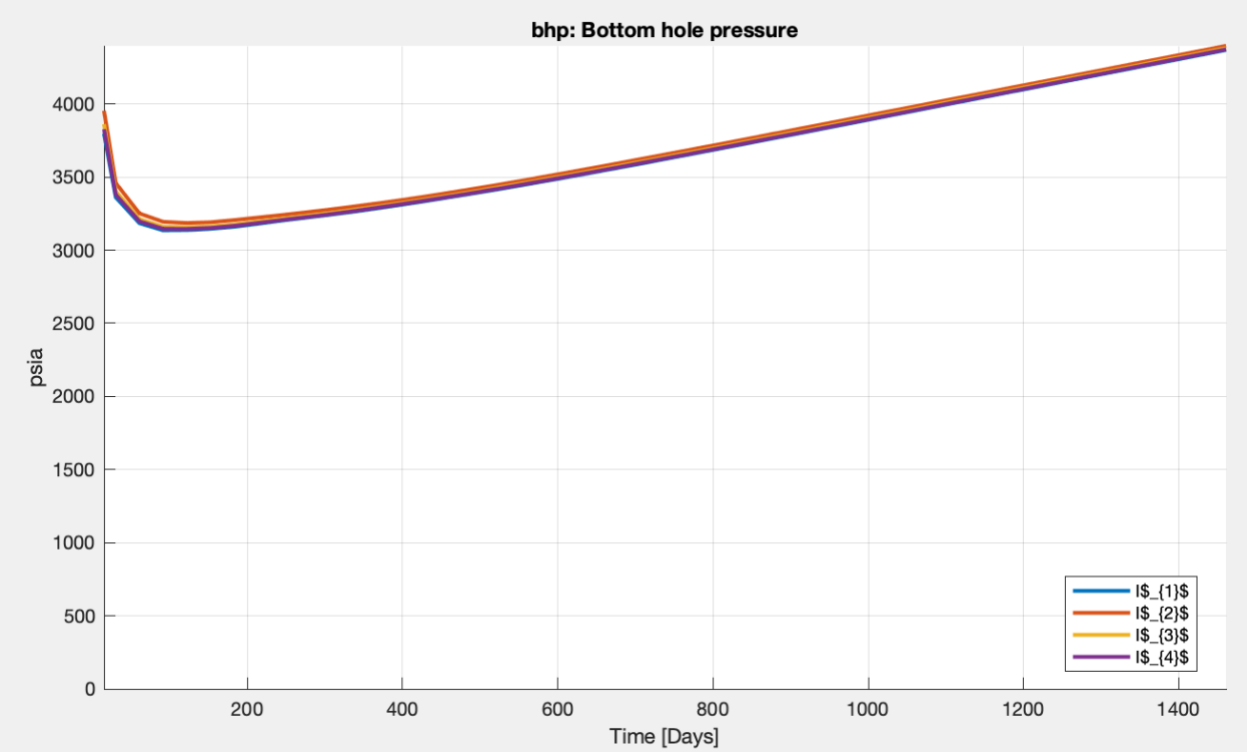

Figure 10: Bottomhole Pressure of 4 injectors simultaneously operating for 4 years

In Figure 10, we can see that the bottom hole pressure reaches around 4300 psia which is the maximum we can allow the pressures to go to (due to the fracture pressures constraint). After the initial dip in the bottomhole pressure level, the bhp seems to rise which equates to the

successful storage of CO_2. This effect is caused by 4 injectors simultaneously operating. The pressure rise is due to the reservoir having closed-flow boundaries, thus no water or gas is escaping. However, how much is stored can be seen by shutting in the 4 injector wells and observe how much extra pressure has built up due to the storage of CO_2 in the reservoir. Doing so will also inform us if the system is losing undesirable pressure. It would tell us if CO_2 were escaping or if there are potential fractures in the formation in a realistic scenario. Due to the nature of the simulation, we have a reservoir with no-flow external and no existing faults. Thus, we should expect a flat curve during the period in which we shut in the wells.

3.2: Stimulating the Well

The “initial spike” in Figure 10 is due to the rapid buildup in CO2 saturation in the vicinity of the well. It is clear that this initial buildup in pressure above the fracture gradient will induce fractures to be initiated in the vicinity of the well. In order to manage this initial pressure rise better, we will stimulate the well or make changes to the reservoir in the immediate vicinity of the well to mitigate this “initial spike”. We could either change the skin factor of the well or change the petrophysical properties such as permeability such as what would result when various stimulations such as injection of hydro fluoride acid are performed.

However, the MRST toolbox does not have any such function that allows stimulating the well by adjusting the skin factor. The wells are by default without any skin factor. Thus, our only option would be to increase the permeability in the grid blocks near the wellbores.

In the base case, permeability has been computed using the Karmen-Cozeny correlation and the corresponding permeability values are in the range of 20 to 100 milli Darcies. This automatically makes the reservoir not very permeable. The permeability values have been increased by a magnitude of 10 which bring us within the range between 2.0e-13 m^2 and 1.0e-12 m-^2 (approx. 200 mD to 1000 mD). In Figure 10, we can see that the “initial spike” is not as significant as before.

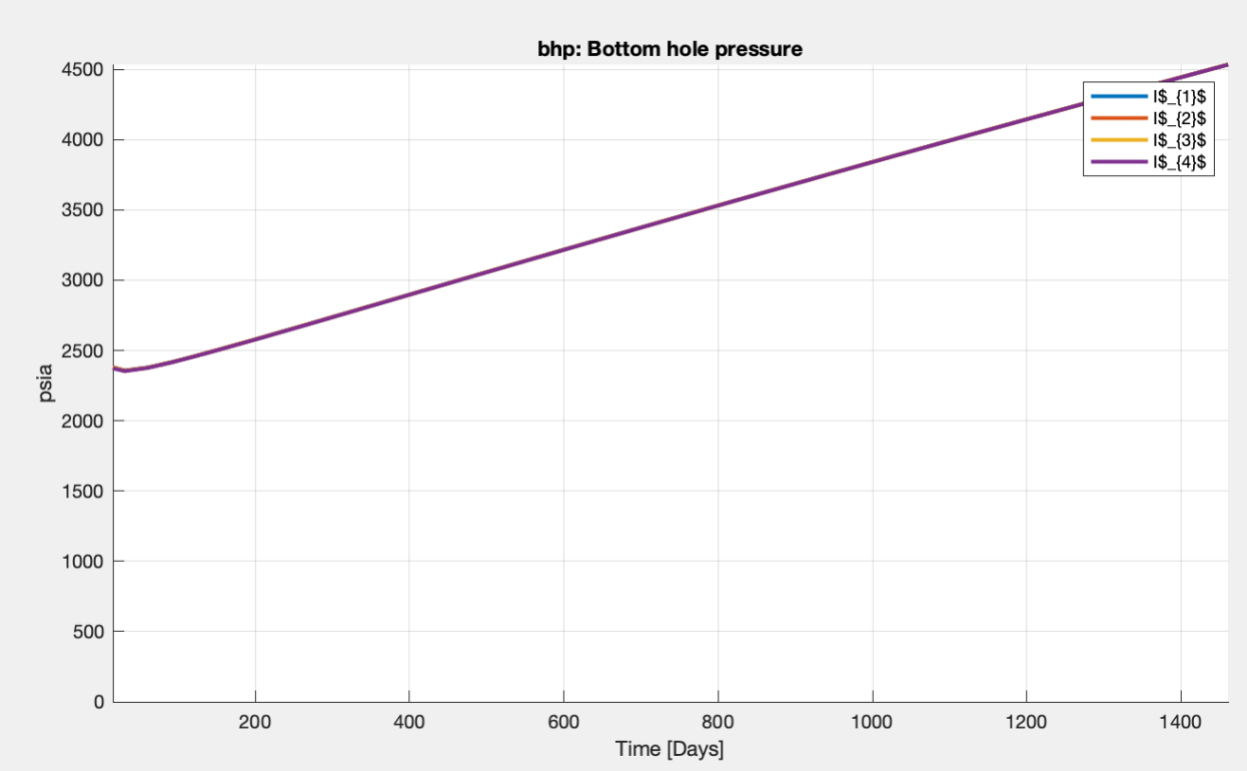

Figure 11: Corrected Bottomhole Pressure of 4 injectors simultaneously operating for 4 years

We will be using the new, greater permeability values for the onwards simulations.

Next, we have run the simulation where we shut in the 4 injector wells for 2 years. Right after the injection period, there's a small dip in the trend of the bottomhole pressure and then a flat line that we can see in Figure 12. This represents the accumulation of the CO_2 gas injected. The difference between the current pressure and the initial reservoir pressure (due to hydrostatics) equates to that due to the CO_2 storage.

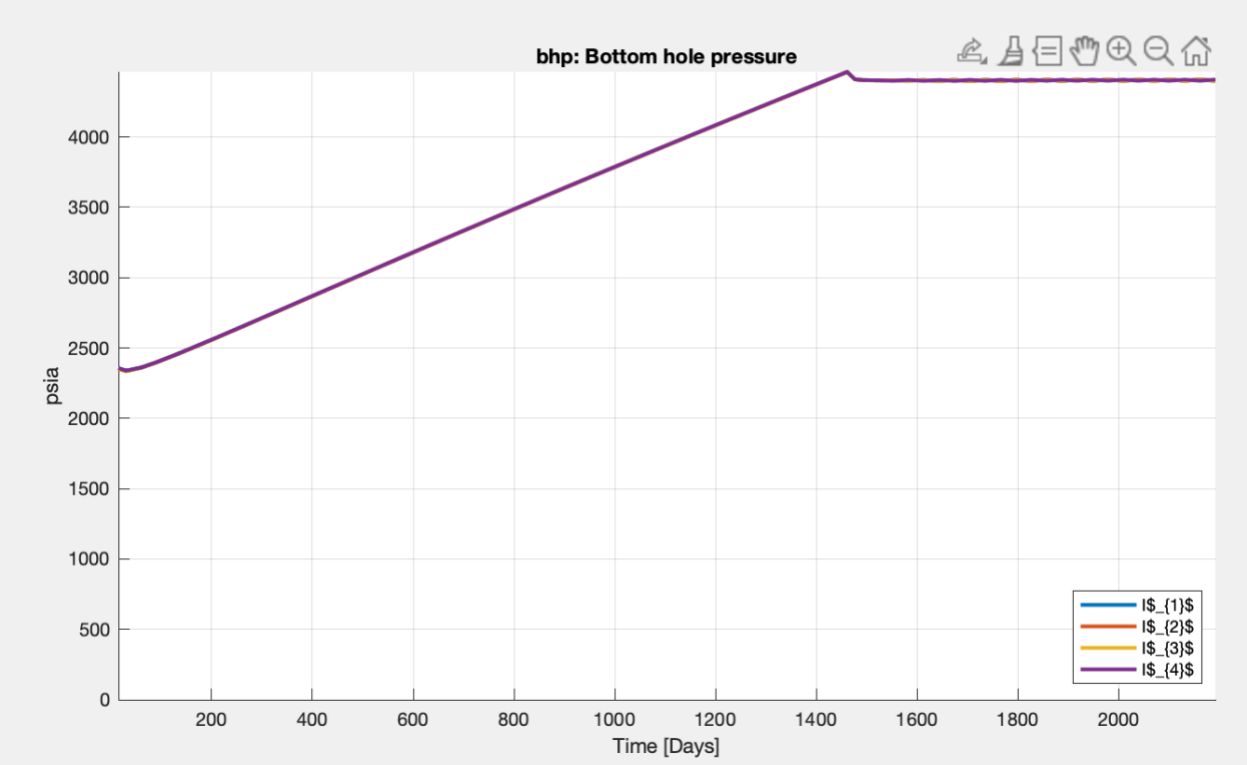

Figure 12: Bottomhole Pressure of 4 injectors simultaneously operating for 4 years and simultaneously shut in for 2 years

In this case, the near well bore regions are assumed to be stimulated and consequently, the initial pressure spike is better managed.

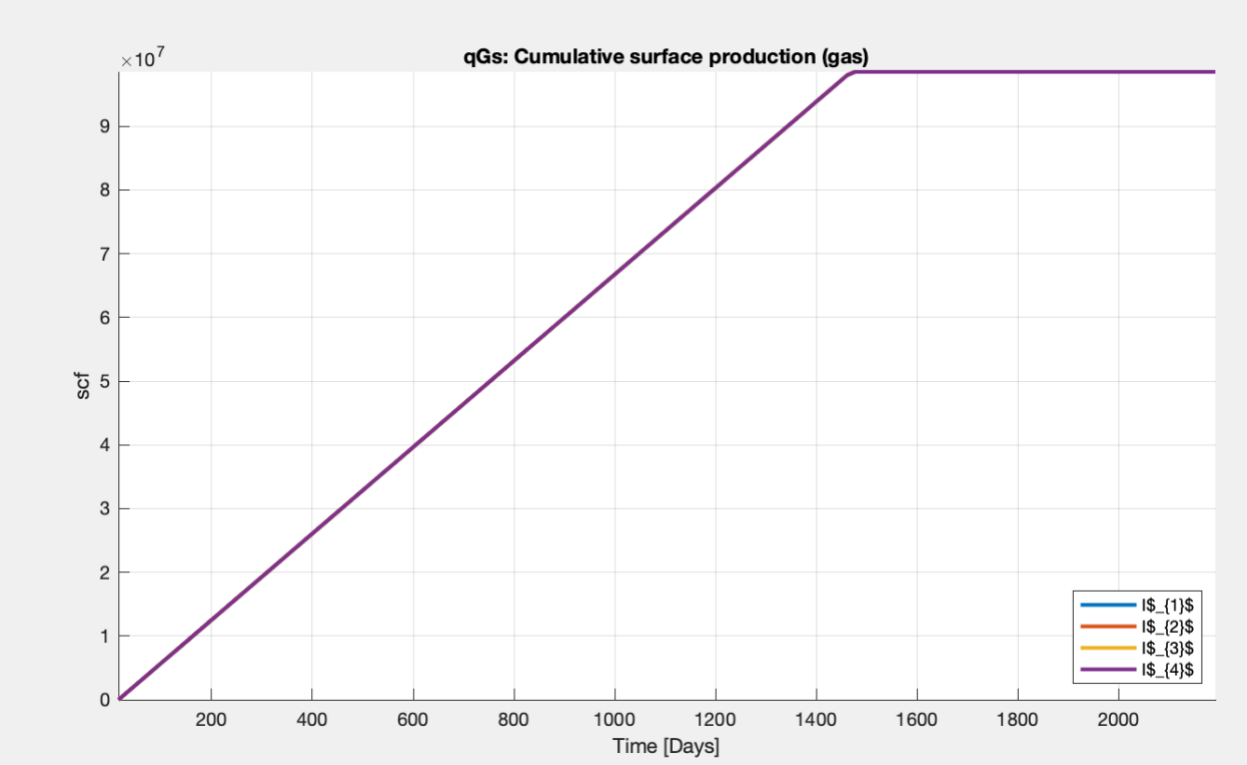

Figure 13: Cumulative Surface Gas Injected using 4 injectors simultaneously operating

The volume injected by each of the injectors is 9.854e+7 SCF (2.79e+6 m^3), thus the total gas injected is 3.946+8 SCF (1.116e+7 m^3) over 4 years. The rate of injection is 1920 cubic meter per day for every injector.

3.3: Cyclic Injection Case (better option for maintaining pressure)

With everything held constant except for the injection rates, the simulation using injectors operated in a cyclic manner is run. All the injectors are injecting 5000 cubic meter per day. Injector 1 is initially run for the first year and then is shut in, injector 2 injects for the second year and then shut in, injector 3 is turned on in the third year and then shut in, and injector 4 is run for the fourth year and then shut in. After this cycle, injector 1 is run again for another 2 years with the same injection rate. Thus, while one injector runs, the others are shut in. By doing so, the injection rates can be increased over twice as much as the previous case.

All the injectors are shut in for additional 2 years. This is done in an attempt to observe the noticeable change in pressure difference and if the reservoir stays at that pressure. This represents the period of accumulation of the CO2. The trend in pressures is shown in Figure 13.

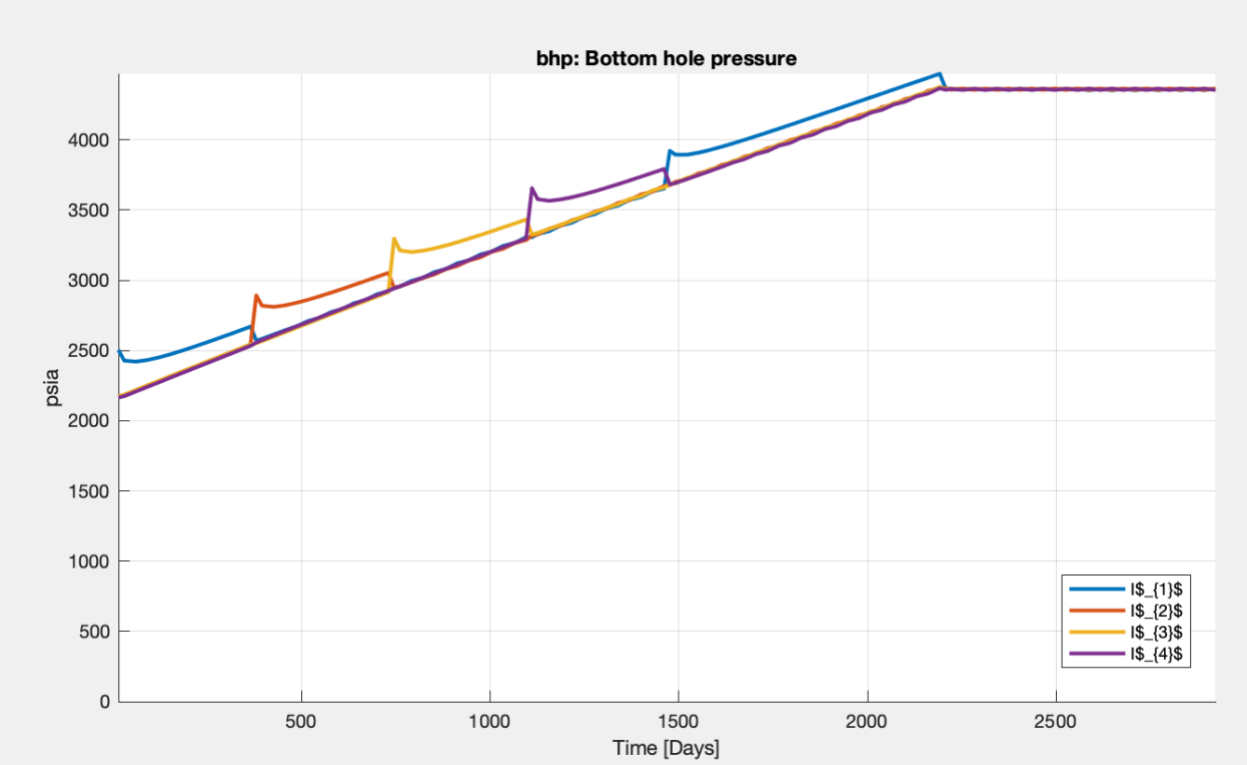

Figure 14: Bottomhole Pressure of 4 injectors operating at different time intervals (cyclic injection) for a total of 5 years and 2 years post injection

It's important to note that we could fit in all these injectors operating at different times and with greater injection rates, and still keep the pressures under the fracture pressures as is evident in Figure 14.

The cumulative gas injected per well and that for the entire project is shown in Figure 15.

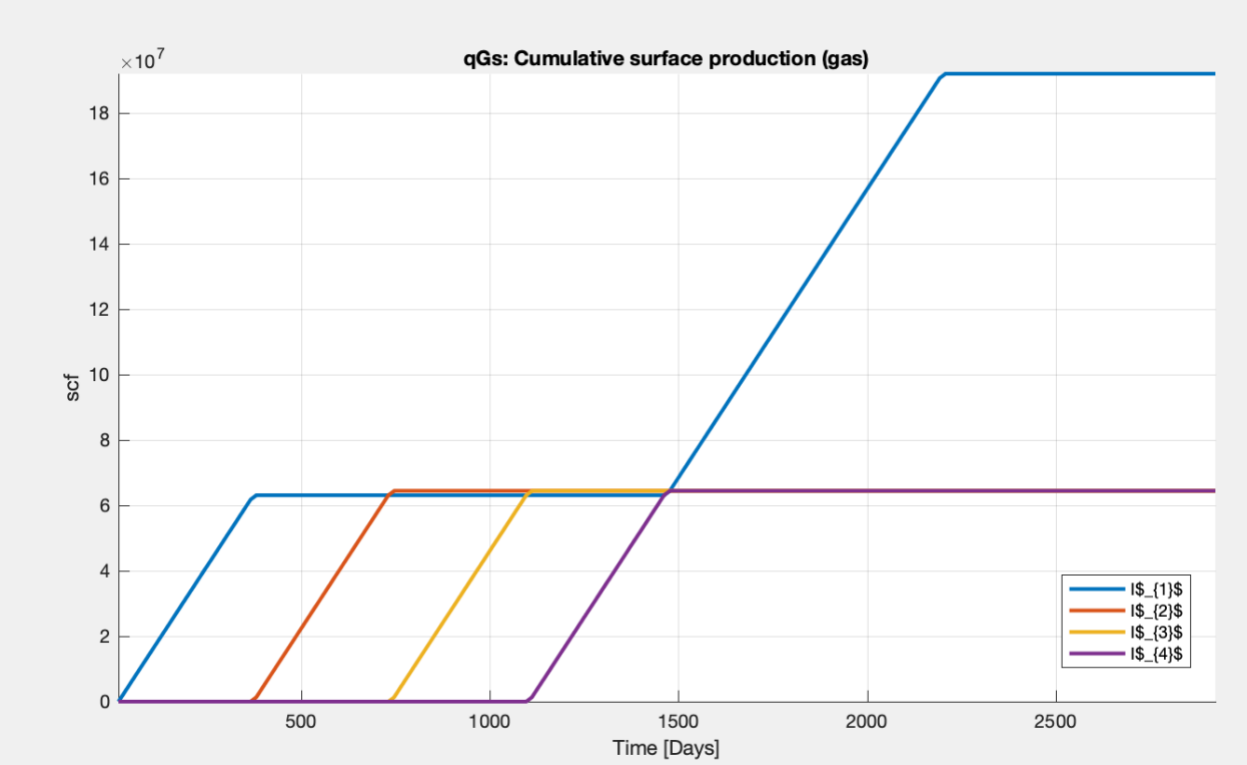

Figure 15: Cumulative Surface Gas Injected using 4 injectors operating cyclically

Within the first 4 years, injector 1 has injected 6.315e+7 SCF of gas while injectors 2, 3, and 4 have injected 6.449e+7 SCF of gas each, totaling of 2.5662e+8 SCF of gas (7.266e+6 m^3). This is less than the case where we injected using all 4 wells simultaneously. However, being able to maintain the bottomhole pressures below the fracture pressures, we could run injector 1 again with the same rate as before (5000 m^3/d) to obtain additional 1.921e+8 SCF of gas which brings the value to the total injected gas over a period of 6 years to be 4.4872e+8 SCF (1.2707e+7 m^3) which is more than the 4.0e+8 SCF cumulative volume that was observed for the base case with all wells injecting simultaneously.

3.4: Potential Strategies, Key Improvements, and Overlooked costs

Our simulation consisted of a simple, yet heterogeneous rectangular reservoir. For better representation of CO2 storage projects, we could potentially create scenarios where the reservoir has an impermeable caprock at the top of the formation which should prevent the gas from rising to the surface and possibly escaping.

In our simulation, we have increased our permeability tenfold to make the reservoir more permeable and avoid the "initial spike" in the bottomhole pressure. We could also try to use horizontal injectors which promise to increase the well injectivity and hence volume of gas injected (2). To maintain pressures better, we can also implement producer wells that will extract the brine in the reservoir and create more space for greater volumes of CO2 gas. The injection of CO2, and the extraction of brine will create a balance in pressure in the reservoir (6). Having the ability to maintain the pressures, greater volumes of CO2 can be injected with higher rates and for longer duration of time.

While it may be true that cyclic injection promises better results in a longer period of time, it may also be costlier because the rates of injection are greater in comparison to those used for simultaneous injection. This implies that there may be an increase in compression costs. It's also true that horizontal wells can help increase well injectivity, but at the same time add on more cost to the project of well completion. The feasibility and optimization of CO2 storage largely depends on the financial costs associated with them. As the CO2 is being sequestered as a waste fluid, there is no commercial revenue to be expected by injecting the gas, unless policies

such as the imposition of a carbon tax are enforced. In order to make the sequestration projects viable, it will be imperative to keep the cost for capturing CO2 and subsequently sequestering the CO2 down to the minimum possible.

Chapter 4

Conclusion

We have performed two cases – one in which we injected in all injection wells simultaneously and second, in which we injected cyclically using the same four 4 vertical injectors. The rates of injection could be doubled for the cyclic injection case because using the same rates of injection as that for simultaneous injection case, it takes much longer for the cyclic injection case to reach fracture pressures. Consequently, the cumulative gas volume injected is greater for the cyclic injection case compared to the simultaneous injection case while still maintaining the reservoir pressure below the fracture pressure. However, if the time for injection is held constant then the simultaneous injection case yields greater injected CO2 volume. This can be very well overcome by the cyclic injection if the injectors are run for longer durations.

We can deduce that greater volumes of CO2 can be injected using cyclic injection while maintaining reasonable pressures below the fracture pressures in comparison to simultaneous injection, thus cyclic injection is better than simultaneous injection for CO2 sequestration. With the cyclic injection, we can maintain pressures much better, thus avoid the risk of fractures in the formation, leakage of CO2 and/or brine into the nearby fresh water sources, and overall, address and mitigate environmental concerns of global greenhouse gasses.

Future extensions of this work can include exploration of other schemes for managing the pressure, such as brine withdrawal from the reservoir and disposal in other intervals or the premixing of CO2 in the extracted brine and subsequent injection. The effectiveness of

horizontal injectors for injecting large volumes of CO2 while still managing the pressure, should also be studied. Lastly, there may more esoteric schemes for managing pressure in the reservoir, such as the inducement of a small number of fractures in the immediate vicinity of the well that will serve to stimulate the region around the well and allow for injection of larger volumes of CO2. However, such a scheme would also require the development of a strategy for arresting the propagation of the fracture too far into the injection interval.

Appendix A

Unit Conversion Table (SI – Field)

Table 2: Unit Conversion Table

	SI	Field
Pressure	**1 Pa**	**0.000145038 psia**
Volume	**1 m3**	**35.3146667 SCF**
Length	**1 m**	**3.28084 ft**

Appendix B

Source Code

Below one could find the source code to run on MATLAB, however the MRST library containing CO_2 modules must be downloaded first via the MRST SINTEF website - https://www.sintef.no/projectweb/mrst/download/

The source code below belongs to SINTEF and is copyrighted. It has been modified to fit the needs of the project.

Cyclic Injection Code

```
%% Basic 3D simulation of a two-phase water and gas system
% This example shows the creation of a simple 3D grid (not VE) in mrst from
% scratch and basic usage of the the TwoPhaseWaterGasModel for modelling a
% CO2-H2O system. CO2 is injected into an intially brine filled reservoir
% from an injection well at the bottom of the reservoir. We model two years
% of injection and two years post injection. CO2 properties are taken from
% CO2lab's tabulated CO2props() function.

%% Load modules
mrstModule add CO2lab ad-core ad-props ad-blackoil mrst-gui;

%% Grid and rock

% Make 3D grid with cells
depth = 1500; % depth of aquifer top surface in m
H     =  100;      % thickness of aquifer
nx    =   35;      % number of blocks along x
ny    =   35;      % number of blocks along y
nz    =   20;      % number of blocks along z

xcoord = 0.0;
```

```
ycoord = 0.0;


% creating the length in the x and y directions for the reservoir
for i=1:nx
   if i>5 && i<=10
      xcoord(i+1) = xcoord(i) + 200;
   elseif i>=26 && i<31
      xcoord(i+1) = xcoord(i) + 200;
   elseif i>=11 && i<=15
      xcoord(i+1) = xcoord(i) + 50;
   elseif i>=21 && i<=25
      xcoord(i+1) = xcoord(i) + 50;
   elseif i>=16 && i<21
      xcoord(i+1) = xcoord(i) + 200;
   else
      xcoord(i+1) = xcoord(i) + 500;
   end
end


for i=1:ny
   if i>5 && i<=10
      ycoord(i+1) = ycoord(i) + 200;
```

```
    elseif i>=26 && i<31
        ycoord(i+1) = ycoord(i) + 200;
    elseif i>=11 && i<=15
        ycoord(i+1) = ycoord(i) + 50;
    elseif i>=21 && i<=25
        ycoord(i+1) = ycoord(i) + 50;
    elseif i>=16 && i<21
        ycoord(i+1) = ycoord(i) + 200;
    else
        ycoord(i+1) = ycoord(i) + 500;
    end
end

xcordtensor = xcoord;
ycordtensor = ycoord;
zcordtensor = linspace(0,H,nz+1);

G    = tensorGrid(xcordtensor, ycordtensor, zcordtensor, 'depthz', repmat(depth, 1, (nx+1) *
(ny+1)));
G = computeGeometry(G);


lower_limit_poro = 0.2;
```

```
upper_limit_poro = 0.3;


% - Generate a Gaussian distribution for porosity within the two limits above
%p    =  gaussianField(G.cartDims,[lower_limit_poro upper_limit_poro]);  %p    =
gaussianField([nx, ny, nz],[lower_limit upper_limit]);
p = gaussianField(G.cartDims, [lower_limit_poro upper_limit_poro]);
p    =  p(:);     % % FOR ISOTROPIC CASE, IT MUST BE CONVERTED INTO SINGLE
COLUMN VECTOR
k    =  10*(p.^3.*(1e-5)^2./(.81*72*(1-p).^2)); % permeability found using karmen cozeny
relationship however multiplied by 10 to make the reservoir more permeable


rock = makeRock(G,k,p);


plotGrid(G, 'FaceColor', 'b');
view(3)
axis  equal tight;
title('Reservoir Simulation Grid');



%%


% % subplot(1,2,1)
% % plotCellData(G,rock.poro,'EdgeColor','none'); %colorbar
```

```
% % colorbar('horiz');
% % pbaspect([2 1 .5]);
% % %axis equal tight
% % title('Reservoir Porosity ');
% % view(3)
% % axis off;
%
% % plot(1,2,2)
% plotCellData(G,rock.perm,'EdgeColor','none'); %colorbar
% colorbar('horiz');
% pbaspect([2 1 .5])
% % axis equal tight;
% title('Reservoir Permeability (sq. m)');
% view(3)
% axis off
%% Initial state
gravity on; % tell MRST to turn on gravity
g = gravity; % get the gravity vector
rhow = 1000; % density of brine kg/m3
initState.pressure = rhow * g(3) * G.cells.centroids(:,3); % initial pressure
initState.s = repmat([1, 0], G.cells.num, 1); % initial saturations
initState.sGmax = initState.s(:,2); % initial max. gas saturation (hysteresis)
```

```
%% Fluid model
CO2     = CO2props(); % load sampled tables of CO2 fluid properties
p_ref   = 15 * mega * Pascal; % choose reference pressure

t_ref   = 70 + 273.15; % choose reference temperature, in Kelvin
rhoc    = CO2.rho(p_ref, t_ref); % CO2 density at ref. press/temp
cf_CO2  = CO2.rhoDP(p_ref, t_ref) / rhoc; % CO2 compressibility
cf_wat  = 0; % brine compressibility (zero)
cf_rock = 4.35e-5 / barsa; % rock compressibility in barsa-1
muw     = 8e-4 * Pascal * second; % brine viscosity in Pa.s
muCO2   = CO2.mu(p_ref, t_ref) * Pascal * second; % CO2 viscosity

mrstModule add ad-props; % The module where initSimpleADIFluid is found

% Use function 'initSimpleADIFluid' to make a simple fluid object
fluid = initSimpleADIFluid('phases', 'WG'          , ...
                           'mu'  , [muw, muCO2]    , ...
                           'rho' , [rhow, rhoc]    , ...
                           'pRef', p_ref           , ...
                           'c'   , [cf_wat, cf_CO2] , ...
                           'cR'  , cf_rock         , ...
                           'n'   , [2 2]);
```

```
% Change relperm curves
srw = 0.27;
src = 0.20;
fluid.krW = @(s) fluid.krW(max((s-srw)./(1-srw), 0));
fluid.krG = @(s) fluid.krG(max((s-src)./(1-src), 0));

% Add capillary pressure curve
pe = 5 * kilo * Pascal;
pcWG = @(sw) pe * sw.^(-1/2);
fluid.pcWG = @(sg) pcWG(max((1-sg-srw)./(1-srw), 1e-5)); %@@



%% Wells

%Injectors


I_inj = [13 23 13 23];
J_inj = [13 13 23 23];
R_inj = [5 5 5 5]*1000*meter^3/day;

W = []; % creating empty set of wells
```

```
for i = 1 : numel(I_inj)

   W = verticalWell(W, G, rock, I_inj(i), J_inj(i), [], 'Type', 'rate', ...
'Val', R_inj(i), 'Radius', 0.05, 'Comp_i', [0 1], ...
'name', ['I$_{', int2str(i), '}$']);

end

%% Boundary conditions

% Start with an empty set of boundary faces
bc = [];

% identify all vertical faces
vface_ind = (G.faces.normals(:,3) == 0);

% identify all boundary faces (having only one cell neighbor
bface_ind = (prod(G.faces.neighbors, 2) == 0);

% identify all lateral boundary faces
```

```
bc_face_ix = find(vface_ind & bface_ind);

% identify cells neighbouring lateral boundary baces
bc_cell_ix = sum(G.faces.neighbors(bc_face_ix,:), 2);

% lateral boundary face pressure equals pressure of corresponding cell
p_face_pressure = initState.pressure(bc_cell_ix);

% % Add hydrostatic pressure conditions to open boundary faces
% bc = addBC(bc, bc_face_ix, 'pressure', p_face_pressure, 'sat', [1, 0]);

% closed-flow boundaries




%% Schedule

% Setting up two copies of the well and boundary specifications.
% Modifying the well in the second copy to have a zero flow rate.
schedule.control(1)   = struct('W', W, 'bc', bc);
schedule.control(2) = struct('W', W, 'bc', bc);
schedule.control(3) = struct('W', W, 'bc', bc);
```

```
schedule.control(4) = struct('W', W, 'bc', bc);
schedule.control(5) = struct('W', W, 'bc', bc);
schedule.control(6) = struct('W', W, 'bc', bc);
for i = 1:numel(W)
    schedule.control(6).W(i).val = 0; %post-injection
end
% schedule.control(3) = struct('W', W, 'bc', bc);
% schedule.control(4) = struct('W', W, 'bc', bc);
% schedule.control(5) = struct('W', W, 'bc', bc);
for i = 1:numel(W)

    if i==1

        schedule.control(2).W(i).val = 0;
        schedule.control(3).W(i).val = 0;
        schedule.control(4).W(i).val = 0;

%         schedule.control(3).W(i).status = false;
%         schedule.control(4).W(i).status = false;
    elseif i==2

        schedule.control(1).W(i).val = 0;
        schedule.control(3).W(i).val = 0;
```

```
        schedule.control(4).W(i).val = 0;
        schedule.control(5).W(i).val = 0;

%       schedule.control(1).W(i).status = false;
%       schedule.control(3).W(i).status = false;
%       schedule.control(4).W(i).status = false;
%       schedule.control(5).W(i).status = false;

    elseif i==3
        schedule.control(1).W(i).val = 0;
        schedule.control(2).W(i).val = 0;
        schedule.control(4).W(i).val = 0;
        schedule.control(5).W(i).val = 0;

%       schedule.control(1).W(i).status = false;
%       schedule.control(2).W(i).status = false;
%       schedule.control(4).W(i).status = false;
%       schedule.control(5).W(i).status = false;
    elseif i==4
        schedule.control(1).W(i).val = 0;
        schedule.control(2).W(i).val = 0;
        schedule.control(3).W(i).val = 0;
        schedule.control(5).W(i).val = 0;
```

```
%         schedule.control(1).W(i).status = false;
%         schedule.control(2).W(i).status = false;
%         schedule.control(3).W(i).status = false;
%         schedule.control(5).W(i).status = false;
    end



end

dT1 = rampupTimesteps(1*year,year/12,1);
%
% schedule.step.val = dT1;
% schedule.step.control = ones(numel(dT1),1);

dT2 = rampupTimesteps(1*year, year/12, 1);
dT3 = rampupTimesteps(1*year, year/12, 1);
dT4 = rampupTimesteps(1*year, year/12, 1);
dT5 = rampupTimesteps(2*year, year/12, 1);
dT6 = rampupTimesteps(2*year, year/12, 1);

% schedule.step.val = [dT1; ...
%                      dT2; dT3; dT4; dT5];
```

```
%
% % Specifying which control to use for each timestep.
% schedule.step.control = [ones(numel(dT1), 1); ones(numel(dT2),1)*2; ones(numel(dT3),1)*3;
ones(numel(dT4),1)*4; ones(numel(dT5),1)*5];% Specifying which control to use for each
timestep.



schedule.step.control = [ones(numel(dT1), 1); ones(numel(dT2),1)*2;
ones(numel(dT3),1)*3;ones(numel(dT4),1)*4;ones(numel(dT5),1)*5;ones(numel(dT6),1)*6];
schedule.step.val = [dT1; ...
                     dT2; dT3; dT4; dT5; dT6];



%% Model
model = TwoPhaseWaterGasModel(G, rock, fluid, 0, 0);

%% Simulate

[wellSol, states] = simulateScheduleAD(initState, model, schedule);
```

```
%% Plot plume at end of simulation
sat_end = states{end}.s(:,2);  % CO2 saturation at end state

% Plot cells with CO2 saturation more than 0.05
plume_cells = sat_end > 0.05;

clf; plotGrid(G, 'facecolor', 'none');  % plot outlinone of simulation grid
plotGrid(G, plume_cells, 'facecolor', 'red'); % plot cells with CO2 in red
view(35, 35);


% Inspect results interactively using plotToolbar
clf;
plotToolbar(G,states,'EdgeColor','none');
plotWell(G,W); axis off;  pbaspect([1.5 1 .5])
view(3);  %axis equal tight


plotWellSols(wellSol,cumsum(schedule.step.val))
%% Inspect results interactively using plotToolbar

clf;
plotToolbar(G,states)
```

```
%%
% <html>
% <p><font size="-1">
% Copyright 2009-2020 SINTEF Digital, Mathematics & Cybernetics.
% </font></p>
% <p><font size="-1">
% This file is part of The MATLAB Reservoir Simulation Toolbox (MRST).
% </font></p>
% <p><font size="-1">
% MRST is free software: you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation, either version 3 of the License, or
% (at your option) any later version.
% </font></p>
% <p><font size="-1">
% MRST is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
% </font></p>
% <p><font size="-1">
```

```
% You should have received a copy of the GNU General Public License
% along with MRST.  If not, see
% <a href="http://www.gnu.org/licenses/">http://www.gnu.org/licenses</a>.
% </font></p>
% </html>
```

Simultaneous Injection Code

```
%% Basic 3D simulation of a two-phase water and gas system
% This example shows the creation of a simple 3D grid (not VE) in mrst from
% scratch and basic usage of the the TwoPhaseWaterGasModel for modelling a
% CO2-H2O system. CO2 is injected into an intially brine filled reservoir
% from an injection well at the bottom of the reservoir. We model two years
% of injection and two years post injection. CO2 properties are taken from
% CO2lab's tabulated CO2props() function.

%% Load modules
mrstModule add CO2lab ad-core ad-props ad-blackoil mrst-gui;

%% Grid and rock

% Make 3D grid with cells
depth = 1500; % depth of aquifer top surface in m
H     =  100;      % thickness of aquifer
nx    =   35;      % number of blocks along x
ny    =   35;      % number of blocks along y
nz    =   20;      % number of blocks along z

xcoord = 0.0;
```

```
ycoord = 0.0;


% creating the length in the x and y directions for the reservoir
for i=1:nx
   if i>5 && i<=10
      xcoord(i+1) = xcoord(i) + 200;
   elseif i>=26 && i<31
      xcoord(i+1) = xcoord(i) + 200;
   elseif i>=11 && i<=15
      xcoord(i+1) = xcoord(i) + 50;
   elseif i>=21 && i<=25
      xcoord(i+1) = xcoord(i) + 50;
   elseif i>=16 && i<21
      xcoord(i+1) = xcoord(i) + 200;
   else
      xcoord(i+1) = xcoord(i) + 500;
   end
end


for i=1:ny
   if i>5 && i<=10
      ycoord(i+1) = ycoord(i) + 200;
```

```
    elseif i>=26 && i<31
        ycoord(i+1) = ycoord(i) + 200;
    elseif i>=11 && i<=15
        ycoord(i+1) = ycoord(i) + 50;
    elseif i>=21 && i<=25
        ycoord(i+1) = ycoord(i) + 50;
    elseif i>=16 && i<21
        ycoord(i+1) = ycoord(i) + 200;
    else
        ycoord(i+1) = ycoord(i) + 500;
    end
end

xcordtensor = xcoord;
ycordtensor = ycoord;
zcordtensor = linspace(0,H,nz+1);

G    = tensorGrid(xcordtensor, ycordtensor, zcordtensor, 'depthz', repmat(depth, 1, (nx+1) *
(ny+1)));
G = computeGeometry(G);


lower_limit_poro = 0.2;
```

```
upper_limit_poro = 0.3;

% - Generate a Gaussian distribution for porosity within the two limits above
%p    =  gaussianField(G.cartDims,[lower_limit_poro upper_limit_poro]);  %p    =
gaussianField([nx, ny, nz],[lower_limit upper_limit]);
p = gaussianField(G.cartDims, [lower_limit_poro upper_limit_poro]);
p    =  p(:);     % % FOR ISOTROPIC CASE, IT MUST BE CONVERTED INTO SINGLE
COLUMN VECTOR
k    =  10*(p.^3.*(1e-5)^2./(.81*72*(1-p).^2)); % permeability found using karmen cozeny
relationship however multiplied by 10 to make the reservoir more permeable

rock = makeRock(G,k,p);

plotGrid(G, 'FaceColor', 'b');
view(3)
axis  equal tight;
title('Reservoir Simulation Grid');


%%

% % subplot(1,2,1)
% % plotCellData(G,rock.poro,'EdgeColor','none'); %colorbar
```

```
% % colorbar('horiz');
% % pbaspect([2 1 .5]);
% % %axis equal tight
% % title('Reservoir Porosity ');
% % view(3)
% % axis off;
%
% % plot(1,2,2)
% plotCellData(G,rock.perm,'EdgeColor','none'); %colorbar
% colorbar('horiz');
% pbaspect([2 1 .5])
% % axis equal tight;
% title('Reservoir Permeability (sq. m)');
% view(3)
% axis off
%% Initial state
gravity on; % tell MRST to turn on gravity
g = gravity; % get the gravity vector
rhow = 1000; % density of brine kg/m3
initState.pressure = rhow * g(3) * G.cells.centroids(:,3); % initial pressure
initState.s = repmat([1, 0], G.cells.num, 1); % initial saturations
initState.sGmax = initState.s(:,2); % initial max. gas saturation (hysteresis)
```

```
%% Fluid model
CO2     = CO2props(); % load sampled tables of CO2 fluid properties
p_ref   = 15 * mega * Pascal; % choose reference pressure

t_ref   = 70 + 273.15; % choose reference temperature, in Kelvin
rhoc    = CO2.rho(p_ref, t_ref); % CO2 density at ref. press/temp
cf_CO2  = CO2.rhoDP(p_ref, t_ref) / rhoc; % CO2 compressibility
cf_wat  = 0; % brine compressibility (zero)
cf_rock = 4.35e-5 / barsa; % rock compressibility in barsa-1
muw     = 8e-4 * Pascal * second; % brine viscosity in Pa.s
muCO2   = CO2.mu(p_ref, t_ref) * Pascal * second; % CO2 viscosity

mrstModule add ad-props; % The module where initSimpleADIFluid is found

% Use function 'initSimpleADIFluid' to make a simple fluid object
fluid = initSimpleADIFluid('phases', 'WG'           , ...
                           'mu'  , [muw, muCO2]     , ...
                           'rho' , [rhow, rhoc]     , ...
                           'pRef', p_ref            , ...
                           'c'   , [cf_wat, cf_CO2] , ...
                           'cR'  , cf_rock          , ...
                           'n'   , [2 2]);
```

```
% Change relperm curves

srw = 0.27;

src = 0.20;

fluid.krW = @(s) fluid.krW(max((s-srw)./(1-srw), 0));

fluid.krG = @(s) fluid.krG(max((s-src)./(1-src), 0));


% Add capillary pressure curve

pe = 5 * kilo * Pascal;

pcWG = @(sw) pe * sw.^(-1/2);

fluid.pcWG = @(sg) pcWG(max((1-sg-srw)./(1-srw), 1e-5)); %@@




%% Wells


%Injectors



I_inj = [13 23 13 23];

J_inj = [13 13 23 23];

R_inj = [4 4 4 4]*480*meter^3/day;


W = []; % creating empty set of wells
```

```
for i = 1 : numel(I_inj)

    W = verticalWell(W, G, rock, I_inj(i), J_inj(i), [], 'Type', 'rate', ...
'Val', R_inj(i), 'Radius', 0.05, 'Comp_i', [0 1], ...
'name', ['I$_{', int2str(i), '}$']);

end

%% Boundary conditions

% Start with an empty set of boundary faces
bc = [];

% identify all vertical faces
vface_ind = (G.faces.normals(:,3) == 0);

% identify all boundary faces (having only one cell neighbor
bface_ind = (prod(G.faces.neighbors, 2) == 0);

% identify all lateral boundary faces
```

```
bc_face_ix = find(vface_ind & bface_ind);

% identify cells neighbouring lateral boundary baces
bc_cell_ix = sum(G.faces.neighbors(bc_face_ix,:), 2);

% lateral boundary face pressure equals pressure of corresponding cell
p_face_pressure = initState.pressure(bc_cell_ix);

% % Add hydrostatic pressure conditions to open boundary faces
% bc = addBC(bc, bc_face_ix, 'pressure', p_face_pressure, 'sat', [1, 0]);

% closed-flow boundaries



%% Schedule

% Setting up two copies of the well and boundary specifications.
% Modifying the well in the second copy to have a zero flow rate.
schedule.control(1)   = struct('W', W, 'bc', bc);
schedule.control(2)   = struct('W', W, 'bc', bc);
for i = 1:numel(W)
```

```
    schedule.control(2).W(i).val = 0;
end



dT1 = rampupTimesteps(4*year,year/12,1); %injection period
dT2 = rampupTimesteps(2*year,year/12,1); % post-injection period

schedule.step.control = [ones(numel(dT1), 1); ones(numel(dT2),1)*2];
schedule.step.val = [dT1; dT2];



%% Model
model = TwoPhaseWaterGasModel(G, rock, fluid, 0, 0);

%% Simulate

[wellSol, states] = simulateScheduleAD(initState, model, schedule);

%% Plot plume at end of simulation
sat_end = states{end}.s(:,2);  % CO2 saturation at end state

% Plot cells with CO2 saturation more than 0.05
plume_cells = sat_end > 0.05;
```

```
clf; plotGrid(G, 'facecolor', 'none');  % plot outlinone of simulation grid
plotGrid(G, plume_cells, 'facecolor', 'red'); % plot cells with CO2 in red
view(35, 35);


% Inspect results interactively using plotToolbar
clf;
plotToolbar(G,states,'EdgeColor','none');
plotWell(G,W); axis off;  pbaspect([1.5 1 .5])
view(3);  %axis equal tight


plotWellSols(wellSol,cumsum(schedule.step.val))
%% Inspect results interactively using plotToolbar

clf;
plotToolbar(G,states)


%%
% <html>
% <p><font size="-1">
```

```
% Copyright 2009-2020 SINTEF Digital, Mathematics & Cybernetics.
% </font></p>
% <p><font size="-1">
% This file is part of The MATLAB Reservoir Simulation Toolbox (MRST).
% </font></p>
% <p><font size="-1">
% MRST is free software: you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation, either version 3 of the License, or
% (at your option) any later version.
% </font></p>
% <p><font size="-1">
% MRST is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
% </font></p>
% <p><font size="-1">
% You should have received a copy of the GNU General Public License
% along with MRST.  If not, see
% <a href="http://www.gnu.org/licenses/">http://www.gnu.org/licenses</a>.
% </font></p>
% </html>
```

BIBLIOGRAPHY

1. Cihan, A., Birkholzer, J., & Bianchi, M. (2014). Targeted pressure management During CO2 SEQUESTRATION: Optimization of well placement and Brine Extraction. *Energy Procedia*, *63*, 5325-5332. doi:10.1016/j.egypro.2014.11.564
2. Fracture gradient. (2019). Retrieved April 01, 2021, from https://www.sciencedirect.com/topics/engineering/fracture-gradient
3. Hoteit, H., Fahs, M., & Soltanian, M. R. (2019). Assessment of CO2 injectivity during sequestration in depleted gas reservoirs. *Geosciences, 9*(5), 199. doi:10.3390/geosciences9050199
4. Hoteit, H., Fahs, M., & Soltanian, M. R. (2019). Assessment of CO2 injectivity during sequestration in depleted gas reservoirs. *Geosciences, 9*(5), 199. doi:10.3390/geosciences9050199
5. Liang, Q. (2002). Application of quantitative risk analysis to pore pressure and fracture gradient prediction. *All Days*. doi:10.2118/77354-ms
6. The MATLAB Reservoir Simulation toolbox. (2019). *An Introduction to Reservoir Simulation Using MATLAB/GNU Octave,* 597-630. doi:10.1017/9781108591416.021

ACADEMIC VITA

ISHTIAQUL ISLAM

imi5022@psu.edu

OBJECTIVE

Looking for roles in Field, Project, Reservoir, Drilling, Production, Process, Well, Wireline, Pipeline, Mud, Engineering or Data Analyst roles to apply the skills gained as a well-rounded Engineer. Willing to relocate. Very adept in Microsoft Excel

EDUCATION

Bachelor of Science in Petroleum and Natural Gas Engineering. May 2021
Pennsylvania State University, Schreyer Honors College, University Park, PA
Thesis – "*Studying the injectivity of CO2 into subsurface formations for geologic sequestration"*
Coursework: Wireline Logging / Formation Evaluation, Fluid Mechanics, Drilling Engineering, Applied Reservoir Engineering, Production Completions Engineering, Strength of Materials, Engineering Dynamics, Applied Reservoir Analysis, Secondary Recovery, Fundamentals of Reservoir Simulation and Modeling, Geo-Resources Evaluation & Investment Analysis

TECHNICAL / NON-TECHNICAL SKILLS

- **Python, C++**
- **MATLAB (Simulation)**
- **SQL**
- **AutoCAD, SolidWorks**
- **Microsoft Office (inc. Excel, Word, PowerPoint)**
- **CMG, Petrel, TechLog**
- **VBA Excel**
- **Synergi Pipeline Fluids simulation**

EXPERIENCE

Teaching Assistant, Department of Energy and Mineral Engineering, Penn State August 2020 – December 2020

- Assisted in teaching undergraduate students on the basics of Fluid Mechanics, Bernoulli's equation, Reynold's number, Pipe flow Pressure heads, Major and Minor Frictional Head Losses, Laminar and Turbulent Flow characteristics, Engineering Assumptions
- Held TA office hours to help students solve problems

Student Researcher, Schreyer Honors College August 2019 – Present

- Extracted and interpreted geological and petrophysical data and findings and reported them directly to thesis adviser for later comprehensive analysis - Research
- Programmed codes on MATLAB to run CO2 flow simulation, adjusting boundary conditions and petrophysical properties to produce best results and optimization of CO2 sequestration
- Assessed the output for the code and diagnosed computer and human error from the output

Captain, Petrobowl – Society of Petroleum Engineers August 2018 – Present

- Recruited underclassmen for Petrobowl, and improved the number of participants from 7 to 11
- Developed team strategies to prioritize specialization on different topics for the competition
- Created a safe, friendly, communicative atmosphere to inspire positive personal and interpersonal growth among peers

Crew Leader, Penn State Residential Dining June 2017 – July 2019

- Administered and addressed issues at food stations and executed operations in the absence of managers which resulted in a promotion to a crew leader
- Encouraged co-workers to work efficiently which resulted in open communication during the shift
- Supervised the operation of food station to ensure on-time catering, clean work environment, and excellent customer service
- Trained 2 new recruits on the basis of professionalism, supportive work atmosphere, and health and environmental hazards, and encouraged them to effectively communicate if any problems arise

AWARDS

Scholarships: Matthew J. Wilson Honors, Bissy Memorial, Hammond Memorial, Newfield Exploration Company, John and Elizabeth Holmes Teas; The President's Freshman Award; Dean's List (6 semesters+)

CERTIFICATIONS: PCEP – Certified Entry-Level Python Programmer, JLG Aerial work Platform, Forklift Operation, LinkedIn Microsoft Excel