COMPUTER IMPLEMENTATION OF THE ROBERT'S METHOD OF WATERFLOOD CALCULATIONS

A Thesis Submitted to the Department of
Petroleum Engineering

African University of Science and Technology Abuja, Nigeria

In Partial Fulfilment of the Requirements for the Degree of

Master of Science in Petroleum Engineering.

By

Uzokwe Paul Onyebuchi

December 2020.

CERTIFICATION

This is to certify that the thesis titled “Computer Implementation of the Robert’s Method of Waterflood Calculations” submitted to the school of postgraduate studies, African University of Science and Technology (AUST), Abuja, Nigeria for the award of the Masters’ degree is a record of original research carried out by Uzokwe Paul Onyebuchi in the Department of Petroleum Engineering.

COMPUTER IMPLEMENTATION OF THE ROBERT'S METHOD OF WATERFLOOD CALCULATIONS

By

Uzokwe Paul Onyebuchi

A THESIS APPROVED BY THE PETROLEUM ENGINEERING DEPARTMENT

RECOMMENDED:

Supervisor, Prof. Ikiensikimama Sunday

The name of the first co-supervisor

The second co-supervisor

Head, Department of Petroleum Engineering

APPROVED:

Chief Academic Officer

August 24th, 2021

Date

COPYRIGHT PAGE

ABSTRACT

This study presents a computer implementation of Robert's method for waterflooding, a critical technique in petroleum engineering for optimizing oil recovery. The implementation aims to enhance the accessibility and usability of Robert's method for industry professionals. Utilizing Python and a user-friendly graphical interface, the application allows users to input various parameters and visualize waterflooding scenarios effectively. Through comparative analysis with traditional waterflooding methods, the software provides insights into the performance and efficiency of Robert's approach. Preliminary results demonstrate that the implementation aligns closely with existing simulation tools, showcasing its potential as a practical resource for engineers. This work contributes to the ongoing efforts to refine waterflooding strategies and improve oil recovery outcomes in the field.

Keywords and Phrases: Waterflood, Robert’s method, Computer, Python, Enhanced oil recovery

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Prof. Sunday S. Ikiensikimama, for his continuous support during my research and the writing of my thesis.

Special thanks to my faculty members: Prof. David Ogbe, Dr. Alpheus Igbokoyi, the head of the department, Prof. Djebbar Tiab, and Prof. Wumi Iledare, for their invaluable education and the profound impact they have had on my life.

I extend warmest regards to my colleagues for the memorable moments shared and the academic contributions and support they provided during my time at the African University of Science and Technology. I am especially grateful to my partner and friend, Lekia Prosper for our stimulating discussions.

Last but not least, I would like to thank my family—my parents and siblings—for their unwavering spiritual, financial, and emotional support throughout the writing of this thesis and in my life in general.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

D= parameter defined by Eq 2.61,dimensionless

E_A = the areal sweep efficiency.

E_D = displacement efficiency

E_V = Vertical sweep efficiency.

F wop = water-to-oil ratio (WOR), *RB/RB*[res *m 3 /res* m3]

h= bed thickness, ft [m]

k = bed permeability, md

kr = relative permeability, dimensionless

L= bed length, ft [m]

M = mobility ratio, dimensionless

n = number of beds

N = oil in place in the pore volume

NPR = cumulative oil recovered, res bbl [res m3]

O= bed-ordering parameter, *cp/md* [Pa' *s/md]*

p = pressure, psia [kPa]

q= injection or production rate, *RB/D*[res *m3 /d]*

S= fluid saturation, dimensionless

Sor = residual oil saturation, dimensionless

S*wi* = irreducible water saturation, dimensionless

t= real or process time, days

W= cumulative injected water, **RB** [res m3]

y= bed width, ft [m]

z= point of flood-front coincidence

{3 = fluid formation volume factor, *RB/STB*[res *m3 /std* m3]

Aj= reservoir fluid mobility, *md/cp (md/Pa'* s)

Ar = reservoir mobility ratio, dimensionless

f1- = fluid viscosity, cp [Pa' s]

Φ = porosity, dimensionless

Subscripts

BT= breakthrough point

i,j,o= bed indices with range *L.n*

o= oil

τ = identification of a cumulative quantity attime *t*

T= total

w= water

Chapter 1: Introduction

1.1 Introduction

The oil and gas industry has been a phenomenon influencing the world’s economy. The World’s proven reserve amounts to 1.65 trillion barrels as of 2016. Proven reserve is the amount of oil that can be recovered under current and available technology, economically, at a specified date. It is the recovered oil that is valued, and technological and economic efforts are aimed at getting the oil out of the subsurface with maximum efficiency.

Waterflooding is a prominent enhanced oil recovery (EOR) technique employed in the petroleum industry to optimize the extraction of crude oil from reservoirs. As conventional oil sources become increasingly depleted, the significance of waterflooding grows, playing a critical role in maintaining production rates and extending the life of oil fields. This introduction will provide a comprehensive overview of waterflooding, its importance in oil recovery, and delve into Robert's method, a refined approach to optimizing waterflooding processes.

Waterflooding involves injecting water into oil reservoirs to maintain reservoir pressure and displace oil toward production wells. The efficiency of this technique is influenced by several factors, including fluid properties, reservoir characteristics, and the dynamics of fluid flow in porous media (Akhmetov et al., 2018). The primary objective of waterflooding is to enhance oil recovery by maintaining reservoir pressure and improving sweep efficiency—the effectiveness with which the injected water displaces oil.

The overall oil recovery from a reservoir can be characterized by two main components: volumetric sweep efficiency and displacement efficiency. Volumetric sweep efficiency refers to the fraction of the reservoir volume that is effectively contacted by the injected water, while displacement efficiency pertains to the ability of the injected water to displace the oil from the pore spaces of the reservoir rock (Lake et al., 2014).

In heterogeneous reservoirs, where variations in permeability and porosity exist, optimizing water injection patterns becomes a challenge. Engineers must consider factors such as fluid mobility, rock properties, and reservoir geometry to maximize oil recovery while minimizing water production (Chen et al., 2019).

The significance of waterflooding in the oil industry is substantial. It is estimated that waterflooding can recover an additional 20-50% of the original oil in place (OOIP) after primary recovery methods have been employed (Butler et al., 2013). This capability to enhance recovery rates is crucial as operators seek to extend the economic life of oil fields amid declining conventional reserves.

Moreover, maintaining reservoir pressure is essential for sustaining oil flow to production wells and preventing premature reservoir depletion. By injecting water, operators can counteract the natural decline in pressure, thus ensuring a more stable production profile over time (Santos et al., 2017). This not only contributes to higher recovery rates but also improves the economic viability of oil production, especially in a volatile market where price fluctuations can significantly impact profitability.

Robert's method of waterflooding is an advanced approach that enhances traditional waterflooding techniques by incorporating mathematical modeling to optimize water injection strategies. This method utilizes analytical solutions to provide a framework for understanding and predicting fluid behavior in reservoirs.

At the heart of Robert's method is the concept of piston-like displacement, which assumes that the injected water moves through the reservoir similarly to a piston. This assumption allows for the development of explicit solutions for key performance parameters under various conditions, such as constant injection pressure and overall injection rate (Robert, 1991).

One of the primary advantages of Robert's method is its ability to address complex reservoir conditions more accurately than traditional models. While conventional waterflooding models often rely on simplified assumptions, Robert's method can account for reservoir heterogeneity, varying fluid properties, and changes in injection rates, providing a more comprehensive understanding of the waterflooding process (Sheng, 2015).

The implementation of Robert's method allows petroleum engineers to design and optimize waterflooding strategies tailored to specific reservoir conditions. By utilizing the analytical solutions provided by this method, engineers can make informed decisions regarding water injection rates, well placements, and overall field management (Robert, 1991).

Furthermore, the predictive capabilities of Robert's method enhance the ability to forecast production outcomes. Accurate modeling of fluid flow and displacement efficiency allows operators to better anticipate production trends and optimize resource allocation, which is critical in an industry characterized by uncertainty and volatility (Pawar et al., 2020).

In this work, focus will be on the theories of waterflooding, which could also be applicable in enhanced oil recovery applications such as polymer flooding, or other chemical immiscible displacement technique. The theoretical advances of this theories, and its foregoing applications even in enhanced oil recovery, will be discussed in the adjoining chapter. But this will be with greater focus on Robert’s waterflood performance prediction method, decorated with an encompassing literature review.

1.1 Aim and Objectives.

The aim of this study is to develop a computer program to implement the Robert’s Waterflood performance technique.

The objectives of the study include:

1. Writing a computer program implementing the Robert’s waterflood method using Python 3.0.
2. Integrating the computer program with a graphical user interface.
3. Converting the resultant computer program to a desktop application.

CHAPTER TWO

2 Theories in Waterflood Performance Evaluation

2.1 Introduction

Waterflooding is a secondary recovery process in which water compatible with the reservoir of interest is injected into the reservoir to displace residual oil. Waterflooding is more commonly employed than immiscible gas injection – the other secondary recovery process – because water, which is the injectant, is relatively inexpensive compared to gas injectant. Waterflooding has two effects on the reservoir; it reduces the rate of reservoir pressure decline during production, possibly increasing the reservoir pressure with continued injection; and water injected into the reservoir sweeps the oil towards the producers, thus increasing oil production and consequently, cumulative oil production (Rose et al., 1989; Abdel-Kareem et al., 2009). Although waterflooding is a relatively inexpensive and mature technology, several potential problems may arise during a the process. Some of the problems include inefficient recovery due to varying permeabilities and anisotropy, reservoir heterogeneities affecting fluid transport within the reservoir, early water breakthrough that may cause production and surface processing problems, etc. (Rose et al., 1989).

2.1.1 Types of Waterflooding

In selecting a waterflooding pattern for the reservoir of interest, several factors are considered, such as reservoir heterogeneities – including directional permeability and formation fractures; injection fluid availability; anticipated maximum oil recovery and flood life; and well spacing, productivities, and injectivities. There are two types of waterflooding; patterned waterflooding and non-patterned waterflooding. In non-patterned waterflooding, there are basically two types;

generally irregular well placements and peripheral or flank waterflooding. In patterned waterflooding, where the well placements are done in some repetitive fashion, there are several types namely; regular 4-spot and skewed 4-spot, 5-spot, 7-spot, 9-spot, direct line drive and staggered line drive patterns. Another type of flooding that can be utilized – depending on reservoir geometry and properties is the crestal and basal pattern. This involves perforating the injector wells up-structure (for gas injection) or down-structure (for water injection) relative to the producer wells, thus utilizing the effect of gravity segregation in the displacement process (Ahmed, 2006). Based on some assumptions, Crawford in 1960 obtained the efficiencies for several pattern floods. According to him, assuming a unit mobility ratio, steady-state condition, homogeneous and uniform reservoir, and ignoring capillary and gravity effects, the efficiencies were 45% to 90% for 9-spot, 72% for 5-spot, and 56% for direct line drive pattern. The unit mobility ratio used was mobility ratio before water breakthrough. Older fluid injection projects involved maximizing oil recovery via understanding and utilizing the reservoir heterogeneities. Well placements were irregular in both secondary and tertiary recovery processes. Eventually, it became a norm that well placements be in some repetitive 10 pattern (Ogali, 2011). Several patterns were analyzed to determine the optimum patterns used in secondary and tertiary recovery processes. Recently, reservoir engineers, even after selecting a pattern, do not use the same pattern throughout the reservoir. This is because of two of the criteria considered in pattern selection which are reservoir heterogeneity and overall project economics (Rose et al., 1989). Waterflood recovery is sensitive to heterogeneity. Mobility ratios and well configuration also influence it. During waterflooding, the effect of heterogeneity becomes less severe upon reducing the well spacing as the mobilized oil has to travel a relatively shorter distance to the nearest producing well (Singhal et al., 2005). The key objective of selecting a flooding pattern is to maximize the contact between the injection fluid and the hydrocarbon system and hence improve oil recovery and the economics of the project. This is a critical step and can be achieved by either drilling infill injector wells or converting existing production wells to injectors. Figure 2.1 shows various patterns used in waterflooding. In this illustration, there are two types of each pattern; the normal pattern and the inverted pattern types. In the normal pattern type, for a set of injectors and producers, there are several injectors and one producer. In the inverted pattern type, for each set of injectors and producers, there are several producers and one injector. This means that each of the patterns shown in Figure 2.1 has the normal and the inverted pattern types. The success of a waterflood flood project can be predicted from proper selection of waterflood patterns. The primary objective is to attain a balance between injection and producer wells within a pattern and minimize oil migration to adjacent patterns and loss into the formation. Balancing patterns essentially means that for every barrel of water injected, a barrel of hydrocarbon is 11 recovered from the production wells surrounding the injector. An unbalanced pattern leads to poor sweep, premature breakthrough, and high-water cycling. An effective pattern balancing leads to better areal sweep and higher oil recovery (Oyebimpe, 2010). A wide variety of flood patterns (injection-production well arrangement) have been studied with efficiencies for various confined well patterns at breakthrough indicating the effect of the pattern. Figure 2.1: Various patterned flood arrangements – triangles are injectors and circles are producers (Tarek Ahmed, 2006). A comparison of the data for the two direct line drive patterns indicate that sweep is a function of spacing ratio, with the greater ratio resulting in higher breakthrough sweep efficiency (Singh and Kiel, 1982). The effect of off-pattern wells was studied by Prats et al., 1962 and they found that the oil recovery at breakthrough is always lower with an off-pattern injection well. Sweep out 12 beyond normal pattern was studied by Caudle et. al., 1955.They found that at least 90 percent of the area lying outside the last row of wells and within one well spacing of these wells would ultimately be swept by the injected water. Landrum and Crawford (1960) studied the effect of directional permeability on sweep efficiency at unit mobility ratio, for a five-spot and direct line drive (square pattern). Their results show that a 45˚ rotation of patterns could result in approximately 100 percent sweep for the five-spot and approximately zero sweep for a line drive

2.1.2 Factors to Consider in Waterflooding

In determining how suitable a reservoir of interest is for waterflooding, Thomas, Mahoney and Winter (1989) pointed out that several reservoir characteristics must be considered. i. Reservoir geometry – includes areal geometry influences on well and facility locations, number and location of platforms for offshore operations. ii. Lithology and rock properties – includes rock types, clay type and content, porosity (single and dual porosity systems), permeability, well spacing, pressure history. iii. Reservoir depth – involves considering drilling costs especially for new injector wells, dual porosity systems, temperature gradient, fracturing (injection pressure versus depth) and fracture types. iv. Fluid properties, fluid saturations, and fluid distribution – includes consideration of the saturation and distribution of the phases (oil, water and gas) throughout the reservoir, oil viscosity, oil mobility, and mobility ratio, areal sweep and flood efficiencies. v. Reservoir uniformity and pay continuity – includes considerations of thief zones, areal continuity of pay zones, faults, fractures, reservoir anisotropy. vi. Primary reservoir driving mechanisms – includes review of the hydrocarbon recovery that can be achieved via waterflooding, and considering how the primary drive mechanisms will affect the waterflooding process. For instance, solution gas drive reservoirs are the best candidates for waterflooding. Waterflooding has been employed successfully in many reservoirs. Extensive research and development in this technology have improved the efficiency of this secondary recovery process. Also, waterflooding in most reservoirs is considered a cheaper option for secondary recovery compared to immiscible gas injection. However, waterflooding has several potential problems; for instance, poor fluid transport within the reservoir due to reservoir heterogeneities, production and surface processing problems caused by early water breakthrough. Every reservoir is unique, and so there is no standard “recipe” for waterflooding a reservoir. Also, although considered a cheaper secondary recovery option, a waterflood project is a huge investment. To maximize oil recovery and economics from a given waterflood project while minimizing injection water volumes and the effects of reservoir heterogeneities, flood management is employed. Extensive research and development of various flood management techniques have been carried out. This research involves employing one of the waterflood management techniques. The optimization objective for a field undergoing water flooding is to maximize expected net present value (NPV) or expected cumulative oil production from the reservoir (Jackson et al., 2017). These objectives are maximized through proper well placement and optimal well control. While well placement and well control are often performed separately, there is increasing interest in performing coupled optimization of both decision variables. This research examined well placement, well control and coupled well placement-well control optimization.

2.2 Sweep Efficiencies.

It has been shown that knowing the oil in place, the areal efficiency, displacement efficiency, and the vertical sweep efficiency, one can determine the determine the cumulative recovery at any time using Equation 2.1.

$$N_p = \text{N} \, (E_A \times E_D \times E_v) \tag{2.1}$$

Where

N = oil in place in the pore volume

E_A = the areal sweep efficiency.

E_D = displacement efficiency

E_V = Vertical sweep efficiency.

The product of the areal and vertical sweep efficiency gives the **volumetric sweep efficiency**. While the overall recovery factor or efficiency is the product of the areal sweep efficiency, vertical sweep efficiency and the displacement sweep efficiency as expressed in Equation 2.2.

$$RF = E_A \times E_D \times E_v \qquad (2.2)$$

Areal sweep efficiency is defined as the reservoir area fraction contacted by the displacing fluid during the waterflooding operation. The areal sweep efficiency is a function of the relative flow properties of oil and water, the well pattern, pressure distribution and the directional permeability. The vertical sweep efficiency is the fraction of the formation in the vertical plane which injected water for waterflood operation will contact. It depends primarily on the vertical stratification of the reservoir. Figure 2.1 is a representation of areal and vertical sweep of waterflood in a stratified reservoir, showing the swept and the un-swept zone.

The displacement sweep efficiency represents that portion of movable oil displaced by the displacing fluid, expressed mathematically in Equation 2.3 – 2.8.

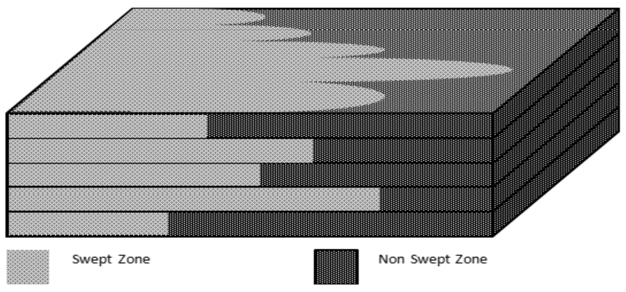


Figure 2.1 – 3-Dimensional view of vertical and areal sweep efficiency

Figure 2.1–3-Dimensional view of vertical and areal sweep efficiency

$$E_D = \frac{Change\ in\ oil\ saturation\ in\ the\ water\ swept\ zone}{Initial\ oil\ saturation\ at\ start\ of\ waterflood} \qquad (2.3)$$

$$E_D = \frac{volume\ of\ oil\ at\ start\ of\ flood - remaining\ oil\ volume}{volume\ of\ oil\ at\ start\ of\ flood} \qquad (2.4)$$

Where

S_{oi} = Initial oil saturation

Soi = 1-Swc – Sgi (2.5)

And

$$\overline{S_o} = 1 - \overline{S_w} \qquad (2.6)$$

Therefore,

$$E_D = \frac{(\frac{Soi}{Boi}) - (\frac{\bar{S}_O}{Bo})}{(Soi/Boi)} \qquad (2.7)$$

The areal sweep efficiency can also be analytically determined without the use of chart from the set of equation shown in Equation 2.8 – 2.11.

$$E_A = E_{ABT} + 0.2749 \ln(\frac{W_{inj}}{W_{iBT}}) \qquad (2.8)$$

$$E_{ABT} = 0.54602036 + \frac{0.03170817}{M} + \frac{0.30222997}{e^M} - 0.00509693M \qquad (2.9)$$

$W_{iBT} = E_{ABT}$ per pore volume (2.10)

$W_{inj} = \frac{E_{ABT}}{E_D}$ per pore volume (2.11)

2.2.1 Mobility Ratio

According to Tiab (1986), it is also found that waterflooding performance in layered composite reservoirs is essentially controlled by the mobility ratio.

Mobility is defined as the ease of flow of fluid. It relates the effective permeability of a fluid with its viscosity.

Mobility is expressed as in Equation 2.12,

$$\lambda_{fluid} = \left(\frac{k}{\mu}\right)_{fluid} \qquad (2.12)$$

The mobility ratio is now defined as the ratio of the mobility of the displacing fluid phase to the mobility of the displaced fluid phase.

Therefore, for a waterflood where injected water is displacing oil, the mobility ratio is given in Equation 2.13 and 2.14.

$$M = \frac{K_w/\mu_w}{K_o/\mu_o} = \frac{K_w\mu_o}{K_o\mu_w} \qquad (2.13)$$

In terms of the relative permeability of the fluid, considering that absolute permeability is the same for both fluids, we have the expression in Equation 2.14.

$$M = \frac{K_{rw}\mu_o}{K_{ro}\mu_w} \qquad (2.14)$$

The relative permeability defined above, for water is the relative permeability in the water-contacted zone of the formation, and for oil, the relative permeability is defined for the un-swept oil zone.

A waterflood operation is successful as the areal sweep efficiency is sufficiently high, and this is only possible at favorable mobility ratio (M less than unity). This implies that the mobility of oil is greater than that of water.Where the mobility ratio is less than unity, the velocity of the waterfront in each layer will be impaired as the flood advances, which somewhat stabilizes the flood front. While the reverse applies to the case where the mobility ratio is greater than unity.

The mobility ratio also influences the fluid injectivity, which is defined as the rate at which fluid can be injected per unit pressure difference between injection and producing wells. At favorable mobility ratios, the fluid injectivity decreases with increase in areal sweep efficiency. While the reverse goes for unfavorable mobility ratios (M greater than unity).

2.3 Theories and Advances in Waterflood Performance Predictions.

Several theories have been proposed to predict waterflood performance. This is quite credited to certain authors and researchers as BuckleyS. E. et al (1942) who were the first to describe and include a saturation distribution using the frontal advancement theorem in immiscible displacement in homogeneous formation. The frontal advance equation incorporates the fractional flow equation in a material balance considering the fraction flow of water at the different sections of the reservoir being invaded by the flooded water. The result is an equation that relates the saturation with position and time at a given fluid injection rate as seen in Equation 2.15. With this equation, one can determine the position of the flood front at any time knowing the fractional flow slope as a function of saturation, which can be obtained from the fractional flow curve.

$$x = \frac{5.615W_i}{\phi A}\frac{df_w}{dS_w} \qquad (2.15)$$

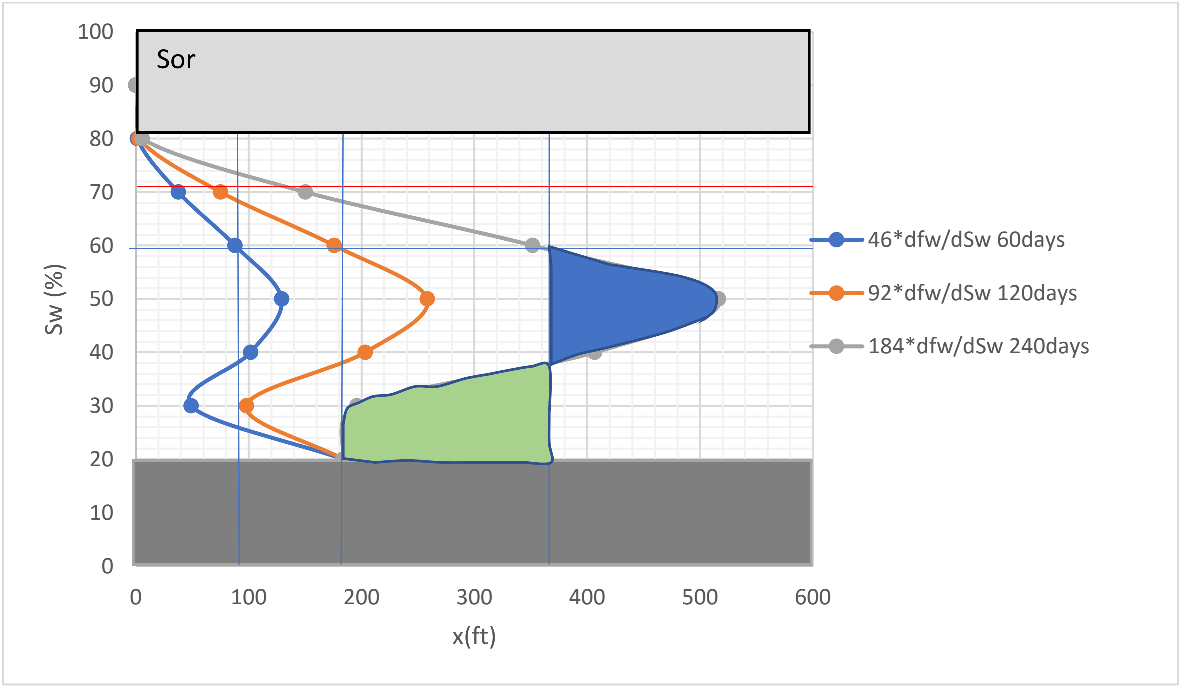


Figure 2.2– Waterflood frontal advancement saturation distribution (Source: B. C. CRAFT, M. H. (1991))

As shown from Figure 2.2 it could be observed that more than one saturation can occur at the same location. This is not possible in a practical sense. To resolve this issue, BuckleyS. E. et al. (1942) suggested that a portion of the calculated saturation distribution curve is imaginary, and the real curve contains a discontinuity at the front.

In a waterflood study, the reservoir is sectioned into three zones. Starting at the injection point, is the free water zone, having 100% water saturation, then a transition zone from which oil and water can be produced, and the oil zone with constituent connate water. The oil zone is considered as the un-swept zone, while the water zone is called the swept zone. During water breakthrough at the producer, indicating a complete sweep by the injected water, residual oil is left in the reservoir.

Welge H. J. (1952) proposed an improved method of achieving the same result as BuckleyS. E. et al. (1942), by integrating the saturation distribution from the injection point to the front and obtained the water saturation behind the front expressed in Equation 2.16. But requires that the initial water saturation be uniform. A graphical interpretation of the result show that a line drawn tangent to the fractional flow curve from the initial water saturation (S_{wi}), meets at the flood front (f_{wf}, S_{wf}) as shown in Figure2.3.

$$\left[\frac{df_w}{dS_w}\right]_{S_{wf}} = \frac{{}^{f_w}/_{S_{wf}} - {}^{f_w}/_{S_{wi}}}{S_{wf} - S_{wi}} \qquad (2.16)$$

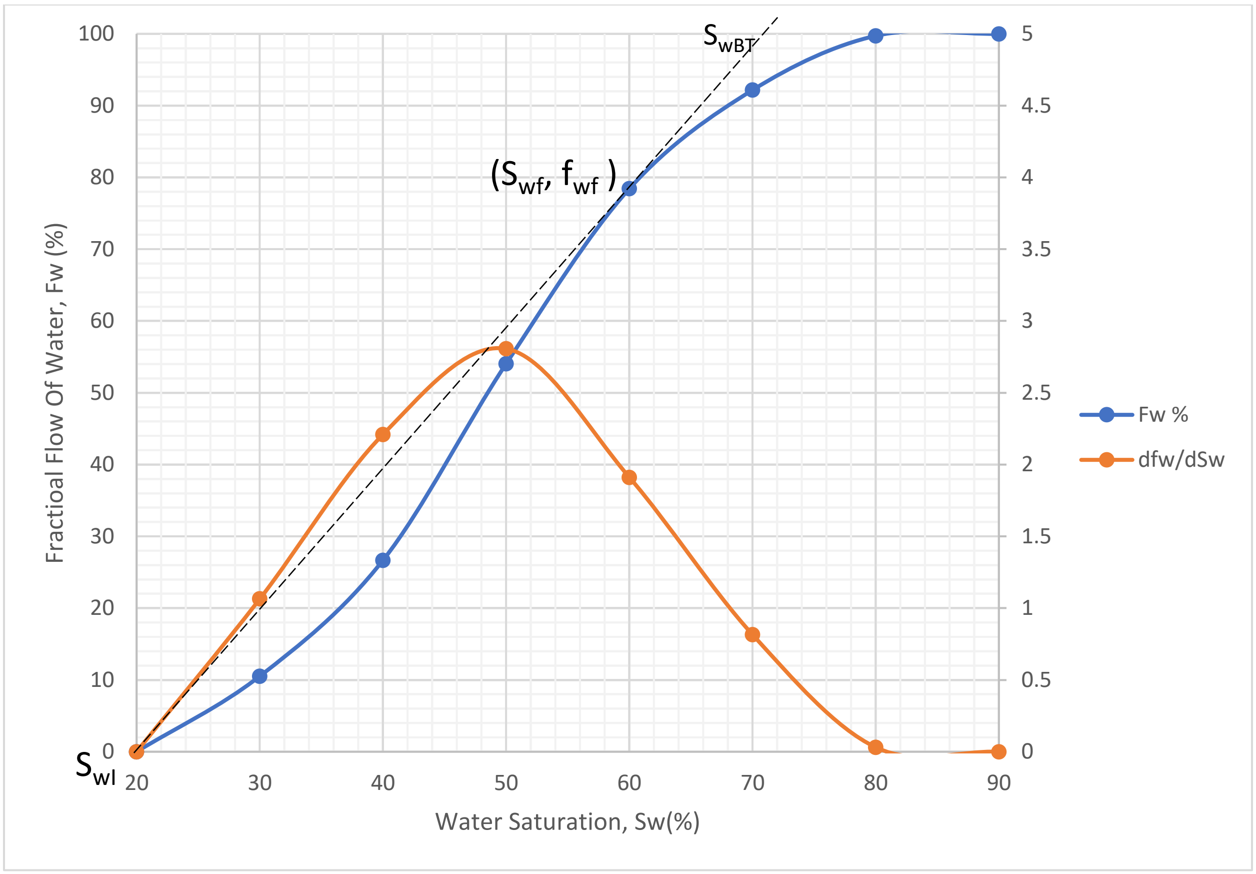


Figure 2.3–Fractional flow curve and derivative of fractional flow curve (Source: B. C. Craft, M. H. (1991))

CHAPTER THREE

3 Computer Implementation Process

3.1 Introduction

In this work, the waterflood performance shall be investigated using the Roberts method using Python programming language.

The data for the program was taken from C.H. Wu (1988). The data includes a relative permeability of water and oil, and saturation table, and a table showing bed parameters, thickness, porosity, and permeability for each bed layer. There were no individual bed viscosities, thus an average viscosity for oil and water was used for the program. The saturation distribution was not also on per bed basis. The relative permeability of oil for the determination of the mobility ratio was taken at the initial water saturation (SWI), and that of water at 1-SOR. Where SOR is the residual oil saturation. The solution and step by step approach for the two cases was implemented with Python 3 software in Anaconda integrated development environment, shown in the schematic in Figure 3.1. The process employs three steps.

- Writing the computer program
- Development of a graphical user interface.
- Conversion of the python file to an executable format.

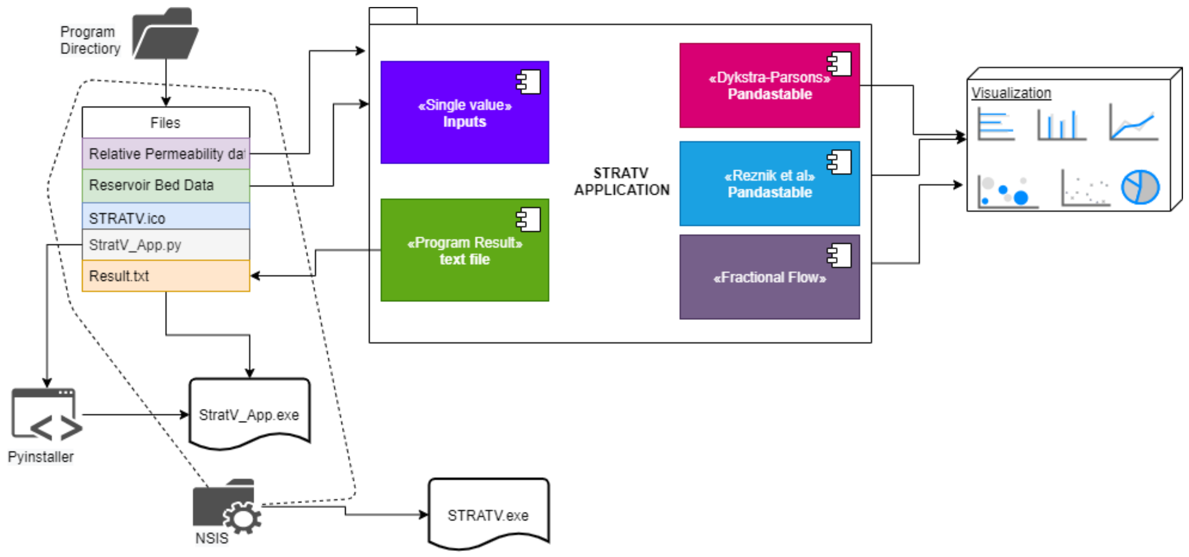


Figure 3.1 – Computer Implementation Process Schematic

3.2 Writing the Computer Program.

The computer program was written in python programming language for Robert's analytical solution, based on the process and waterflood performance parameters proposed by the authors.

3.2.1 Defining Inputs

STEP1: The first step for this process is input initialization. All the inputs required for the calculation were first initialized. The input data in Table 3.1 for the initialization step was also obtained from C.H. Wu (1988) paper.

Table 3.1– Defining Program inputs

```
Length_of_bed_ft = 2896
width_of_bed_ft = 2000
average_porosity = 0.25
VISO = 3.6
VISW = 0.95
OFVF = 1.11
WFVF = 1.01
SWI = 0.2
SGI = 0.16
SOI = 0.65
SOR = 0.35
Residual_gas_saturation_unswept_area = 0.06
Residual_gas_saturation_swept_area = 0.02
Residual_gas_saturation = Residual_gas_saturation_unswept_area+Residual_gas_saturation_swept_area
Constant_injection_rate = 1800
Inj_Pressure_differential = 700
```

3.2.2 Importing Python Modules

STEP 2: The next step requires importing all necessary modules from the python library. Some of which requires installation such as the Pandastable. Installing files or libraries into your python path is mostly accomplished with the ‘pip’ command. By simply typing ‘pip install filename’ into your command prompt or using windows PowerShell for windows users. Pandastable is a software with graphical user interface, for viewing and working with tables, and making custom plots. It was developed byFarrel. D(2019). The following Python modules and libraries in Table 3.2 were imported for this work.

Table 3.2– Algorithms for Importing Python Module

```
from tkinter import*
from tkinter import filedialog
from tkinter import ttk
import pandas as pd
from tkinter.messagebox import*
from tkinter.filedialog import askopenfilename
import csv
import math
from decimal import*
import numpy as np
from pandastable.core import Table
from pandastable.data import TableModel
import matplotlib.pyplot as plt
from scipy import integrate
```

Tkinter is a Python default application for building a graphical user interface. Pandas is the Python default library for working with tables, while numpy is for arrays. Python uses the matplotlib for making plots.

STEP 3: The tabular data for the saturation distribution, relative permeability of water and oil, and the reservoir bed parameters were arranged in a csv file stored in the same directory or folder as the Python file or executable file. This file serves as a template for working with the application. When trying to input new data, the column/ table header title must correspond with the csv file template. The data is as shown in Table 3.3:

Table 3.3- Oil relative permeability data

SW	KRW	KRO
0.2	0	1
0.25	0.003	0.68
0.3	0.008	0.46
0.35	0.018	0.32
0.4	0.035	0.2
0.45	0.054	0.124
0.5	0.08	0.071
0.55	0.105	0.038
0.6	0.14	0.017
0.65	0.18	0

As a matter of rule, the oil and water relative permeability must be (1) saved in the same folder as the executable file (2) must be saved with the name **'Oil_Water_Relative_Permeability_data.csv'**. (3) The file must be saved with the same column heading as shown in Table 3.3 and 3.4

Same rule also applies for the reservoir bed data. This should be saved with the name **'Permeability_Porosity_distribution_data.csv'**, and with same column heading as shown in Table 3.4:

Table 3.4 - Bed data permeability distribution

LAYER	DEPTH	THICKNESS	PERMEABILITY	POROSITY
1	4359.5	1	593	0.301

2	4361	1	500	0.288
3	4363.5	1	366	0.283
4	4365.5	1	1464	0.309
5	4367.5	1	2790	0.311
6	4369.5	1	5940	0.305
7	4371.5	1	9230	0.322
8	4373.5	1	2860	0.306
9	4375.5	1	3080	0.322
10	4377.5	1	594	0.297
11	4379.5	1	2370	0.323
12	4381.5	1	526	0.286

The csv (comma-separated values) files were imported into the python file with the following algorithms in Table 3.5.

Table 3.5 – Algorithm for reading input data csv files

```
bed_data = pd.read_csv('Permeability_Porosity_distribution_data.csv')
RPERM_data = pd.read_csv('Oil_Water_Relative_Permeability_data.csv')
```

STEP 4: After the above steps, the program is written based on the step by process outlined by Roberts. The following were determined from the calculations:

General Calculations

1. Construct a fractional flow curve and determine the average water saturation behind the front.

2. Draw several tangents to the fractional flow curve at Sw values greater than the breakthrough saturation. Determine Sw and dFw/dSw and corresponding to these values. Plot fw' versus Sw and construct a smooth curve through the points.

3. Define the layers within the reservoir and determine the average permeability, porosity, and thickness for each layer.

4. Compute the capacity, kh, and fraction of total capacity, ΔC, for each layer.

5. Compute the injection rate into each layer.

$$i_{wj} = i_{wt} \times \Delta C$$

6. Calculate the cumulative water injection, Wij, into each layer to reach each Sw point chosen in Step 2

$$W_{ij} = \frac{7758 \ \times A_j h_j \emptyset_j}{f_w^1}$$

7. Calculate qoj and qwj for each layer at each Sw point. Before breakthrough in a given bed,

$$q_{0j} = \frac{i_{wj}}{B_o}$$

9. Calculate the recovery, Npj, and the time, tj , to each point.

10.Plot the oil production rate for each layer as a function of time. Use this plot to construct a graph of total oil production rate versus time.

11.Repeat step 10 for the water production rate.

12.Use the total oil and water production rates to construct a plot of WOR versus time.

13. Plot cumulative oil recovery from each layer as a function of time and use this plot to construct a graph of total recovery versus time.

14. Based on estimated expenses, decide on an appropriate WOR cutoff and from the data in Step 12, estimate the life of the project.

15. Use the WOR-time cutoff to determine the projects ultimate recovery from data in Step 13.

It is worth noting that the column and row numbering are based on their index and not necessarily on the reservoir bed layers.

Table 3.6 – Index versus Layer definition

Index	**Layers**
0	7
1	6
2	9
3	8
4	5
5	11
6	4
7	10
8	1
9	12
10	2
11	3

The codes written for the implementation of the solution is found in the Appendix A.

3.2.1 The Fractional Flow Curve

A fractional flow curve menu tab was incorporated into the program. Generating this curve followed four encompassing steps.

To begin this calculation, the following Python modules in Table 3.7 are imported:

Table 3.7– Python modules for making fractional flow curve.

```
import numpy as np
from scipy.optimize import*
import math
import random
import matplotlib
import matplotlib.pyplot as plt
```

Generating the fractional flow equation.

The fractional flow equation is generally given by:

$$f_w = \frac{1}{1+\frac{K_{ro}\,\mu_w}{K_{rw}\,\mu_o}} \tag{3.1}$$

Where,

$$\frac{K_{ro}}{K_{rw}} = ae^{-bS_w} \tag{3.2}$$

This imply that the coefficients and constants *'a'* and *'b'*, can be expressed as:

$$a = K_1 e^{bS_w} \tag{3.3}$$

$$\mathrm{b} = \frac{lnK_1 - lnK_2}{S_{w2} - S_{w1}} \tag{3.4}$$

The Python program for the determination of *a* and *b* was written as shown in Table 3.8:

Table 3.8– Defining function for fractional flow calculation.

```
b = (np.log((KRO/KRW)[2]) - np.log((KRO/KRW)[3]))/(SW[3] -SW[2])
a = (KRO/KRW)[2]*math.exp(b*SW[2])
def fw(SW):
fw = 1/(1+a*(VISW/VISO)*np.exp(-b*SW))
return(fw)
```

Note: The extreme points are not chosen because log of the relative permeability ratio does not form straight lines at the extremes, which will give erroneous result for the relative permeability ratio correlation in Equation (3.2)

After successful determination of *'a'* and *'b'*, the fractional flow data can be generation by substituting *'a'*, and *'b'* into the fractional flow equation.

Generating the Tangent to the fractional flow curve.

Generating the tangent to the fractional flow curve is the tricky part of plotting the fractional flow curve. Customarily, to generating a tangent is easier when the point of tangency is given or known. In this case the only point that is known is the initial water saturation (S_{wi}, 0) from where, the tangent line is drawn.

With the point (S_{wi}, 0) known, the tangent equation can be expressed as:

$$y = m(S_w - S_{wi}) \tag{3.5}$$

where,

m is the slope of the tangent line,

and S_w the water saturation

The concept involves generating several tangent lines that will intercept the fractional flow curve at several points. The line (drawn from point (S_{wi}, 0)) with the maximum slope touching the fractional flow curve will give the suitable tangent line.

This necessitated equating the tangent line equation, and the fractional flow equation.

$$m\,(S_w - S_{wi}) = \frac{1}{1+ae^{-bS_w}\frac{\mu_w}{\mu_o}} \tag{3.6}$$

which gave m to be:

$$m = \frac{1}{(S_w - S_{wi})\left(1+ae^{-bS_w}\frac{\mu_w}{\mu_o}\right)} \tag{3.7}$$

For this program, ten thousand of uniformly distributed random points were generated to the required slope. Using the algorithm in Table 3.9.

Table 3.9 – Algorithm for drawing tangent to the fractional flow curve

```
''' To calculate a suitable slope for the tangent to the fractional flow curve
Drawn from the initial water saturation'''
# STEP1: Generate a list of uniformly distributed random numbers from a water saturation
# greater than the initial water saturation to 1
xList = []
for i in range(0, 10000):
x = random.uniform(SWI+0.1, 1)
xList.append(x)
xs = np.array(xList)
# STEP2: Calculate different slopes of tangents or lines intersecting the fractional
# flow curve using the array generated in step 1 as the water saturation.
m = 1/((xs-SWI)*(1+(VISW/VISO)*a*np.exp(-b*xs)))
# STEP3: Calculate the maximum slope from different slopes generated in step 2.
# The value of this slope will be the slope of the tangent to the fractional flowcurve
```

```
tangent_slope = max(m)
```

Upon calculating the slope of the tangent line, the water saturation can be calculated from tangent equation by substituting the point (S_{wBT}, 1) into the tangent equation.

Where S_{wBT} is the water saturation at breakthrough.

The flood front water saturation is also determined by substituting the point (Swf, Fwf) into the tangent equation and the fractional flow equation. The resulting non-linear equation is then solve using Python *fsolve* by importing the *scipy* module. The algorithms are as shown in Table 3.10.

Table 3.10– Algorithm for Water saturation and fractional flow at flood front

```
Saturation_at_Breakthrough = SWI + 1/tangent_slope
def funct(SWF):
swf = SWF[0]
F[0] = ((tangent_slope*(swf-SWI)*(1+(VISW/VISO)*a*math.exp(-b*swf)))-1)
return F
SWF_Guess = np.array([SWI+0.1])
SWF = fsolve(funct, SWF_Guess)[0]
# Fractional flow at the flood front
Fwf = fw(SWF)
```

Differential of the fractional flow equation

The differential of the fractional flow equation was also plotted on the fractional flow curve. Differentiating the fractional flow equation with respect to water saturation gives the expression below:

$$\left(\frac{df_w}{dS_w}\right)_{S_w} = \frac{abe^{-bS_w}\frac{\mu_w}{\mu_o}}{\left(1+ae^{-bS_w}\frac{\mu_w}{\mu_o}\right)^2} \qquad (3.8)$$

The Python program for this is given as in Table 3.11.

Table 3.11 – Algorithm for fractional flow derivative

```
# Calculating the differential of the fractional flow equation
dfw_dSw = (VISW/VISO)*a*b*np.exp(-SW*b)/(1+(VISW/VISO)*a*np.exp(-SW*b))**2
# Generating the data for the tangent plot
tangent = (SW-SWI)*tangent_slope
```

Making the fractional flow curve

The fractional flow curve comprises the plot of the fractional flow equation, the derivative of the fractional flow equation, and the tangent on the same

plot. The Python algorithm in Table 3.12 was used to set up the plot.

Table 3.12– Algorithm for making the fractional flow curve.

```
# Making the plots
fig, ax = plt.subplots(constrained_layout=True)
fig.set_figheight(8)
fig.set_figwidth(12)
fractional_flow_curve = ax.plot(SW, fw(SW), 'b', label = 'Fractional Flow (Fw)')
tangent_curve = ax.plot(SW, tangent, 'k--')
ax.set_ylabel("Fractional Flow (fw)",fontsize=14)
ax.set_xlabel("Water Saturation (Sw)",fontsize=14)
ax.set_ylim([0,1])
ax.set_xlim([0,1])
# twin object for two different y-axis on the same plot
ax2=ax.twinx()
# make a plot with different y-axis using second axis object
dfw_dSw_curve = ax2.plot(SW, dfw_dSw, 'r', label ='dFw/dSw')
ax2.set_ylabel("dfw/dSw",fontsize=14)
ax.grid(True)
ax2.legend()
ax.legend(loc='upper left')
ax.annotate("  (Swf, Fwf)", (SWF, Fwf))
ax.annotate(" SwBT", (Saturation_at_Breakthrough, 1))
plt.show()
```

The resultant plot of the fractional flow curve is as shown in Figure 3.2.

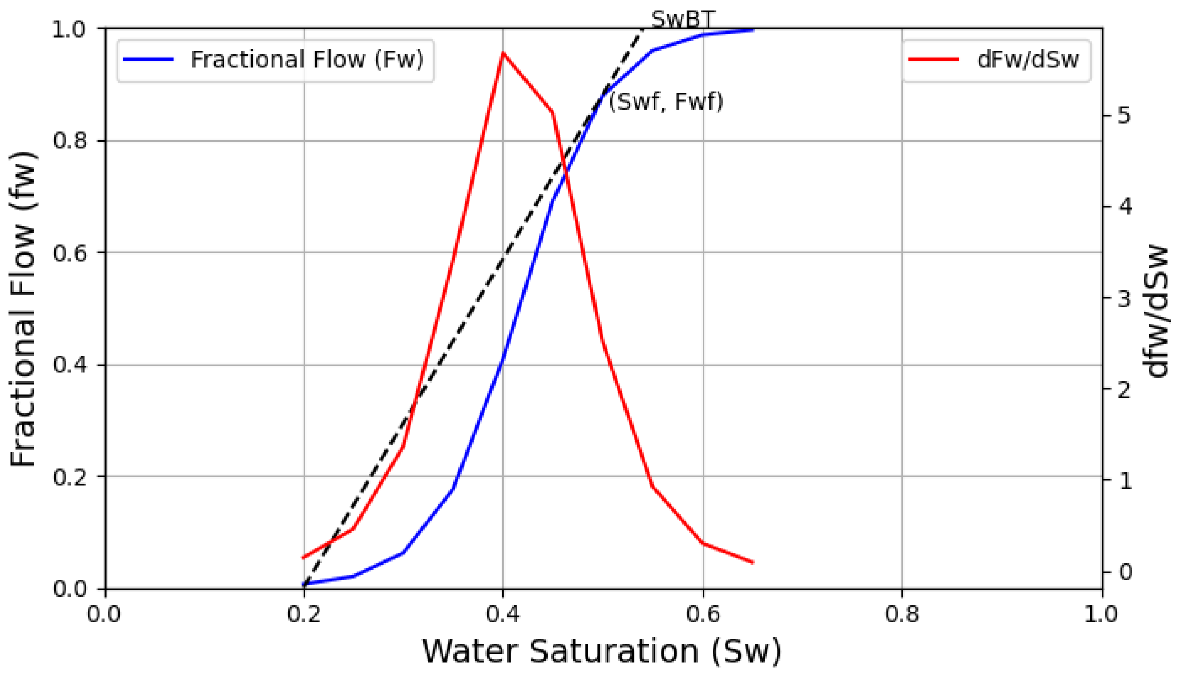


Figure 3.2 – The Fractional Flow Curve

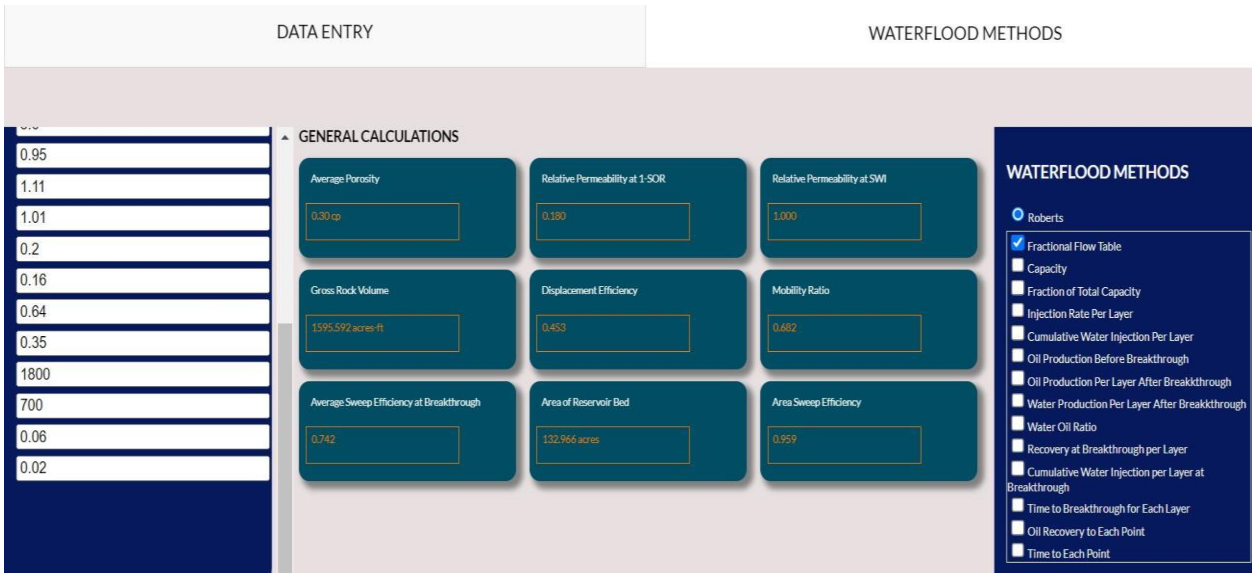
3.3 Development of The Graphical User Interface.

The graphical user interface for the program was created with Tkinter^{TM1}. Tkinter makes use of widgets such as labels, buttons, treeviews, entries, frames, canvas, and lots more to build graphical user interface.

The graphical user interface is as seen in Figure 3.3 to Figure (3.9).

3.3.1 The Application Window.

When the program is turned on or opened the window in Figure 3.2 comes up. This is the application from which you can navigate to other part of the application.



0.95
1.11
1.01
0.2
0.16
0.64
0.35
1800
700
0.06
0.02

Average Porosity: 0.30 cp
Relative Permeability at 1-SOR: 0.180
Relative Permeability at SWI: 1.000
Gross Rock Volume: 1595.592 acres-ft
Displacement Efficiency: 0.453
Mobility Ratio: 0.682
Average Sweep Efficiency at Breakthrough: 0.742
Area of Reservoir Bed: 132.966 acres
Area Sweep Efficiency: 0.959

WATERFLOOD METHODS

○ Roberts

- [✓] Fractional Flow Table
- [ ] Capacity
- [ ] Fraction of Total Capacity
- [ ] Injection Rate Per Layer
- [ ] Cumulative Water Injection Per Layer
- [ ] Oil Production Before Breakthrough
- [ ] Oil Production Per Layer After Breakkthrough
- [ ] Water Production Per Layer After Breakkthrough
- [ ] Water Oil Ratio
- [ ] Recovery at Breakthrough per Layer
- [ ] Cumulative Water Injection per Layer at Breakthrough
- [ ] Time to Breakthrough for Each Layer
- [ ] Oil Recovery to Each Point
- [ ] Time to Each Point

Figure 3.3–Graphical user interface

3.3.2 The Menu Bar

The application consists of five menu bars.

- Load Data
- Fractional flow
- Print Result
- Robert: clicking on this tav brings up the table (*the pandastable*) on the left.

3.4 Conversion of The Python File to An Executable Format.

To convert the python file to an installable executable file, the following process were taken.

- Run command prompt in the directory with the python file and all other related file. Install python installer, by typing the command '*pip install pyinstaller*[2]'. This command will be success if python is installed the system path. With pyinstaller installed, the command *'pyinstaller --onefile -w filename.py'* is issued. If some dependencies (such as pandas, pandastable, etc.) are absent, they can be installed with the command *"pyinstaller -F –hidden-import 'name of module absent'* *filename.py".* The possibility of pyinstaller not finding a dependency already used in the development of the application is traceable to the fact that these dependencies were installed in an IDE and not in the path where Python is in the computer.

- The NSIS[3] (nullsoft scriptable install system) is used to bundle any additional document or file the application may require, into an executable zip file.

CHAPTER FOUR

4 RESULT

The example used to test the application/program with saturation and relative permeability data as shown in Table 3.3, and bed parameters as shown in Table 3.4 was gotten from Langnes etal (1985). Tabular results were obtained for Robert's method, and the necessary waterflood descriptive and performance plots were made

PERFORMANCE PLOTS

Plot of water injection per layer against Capacity

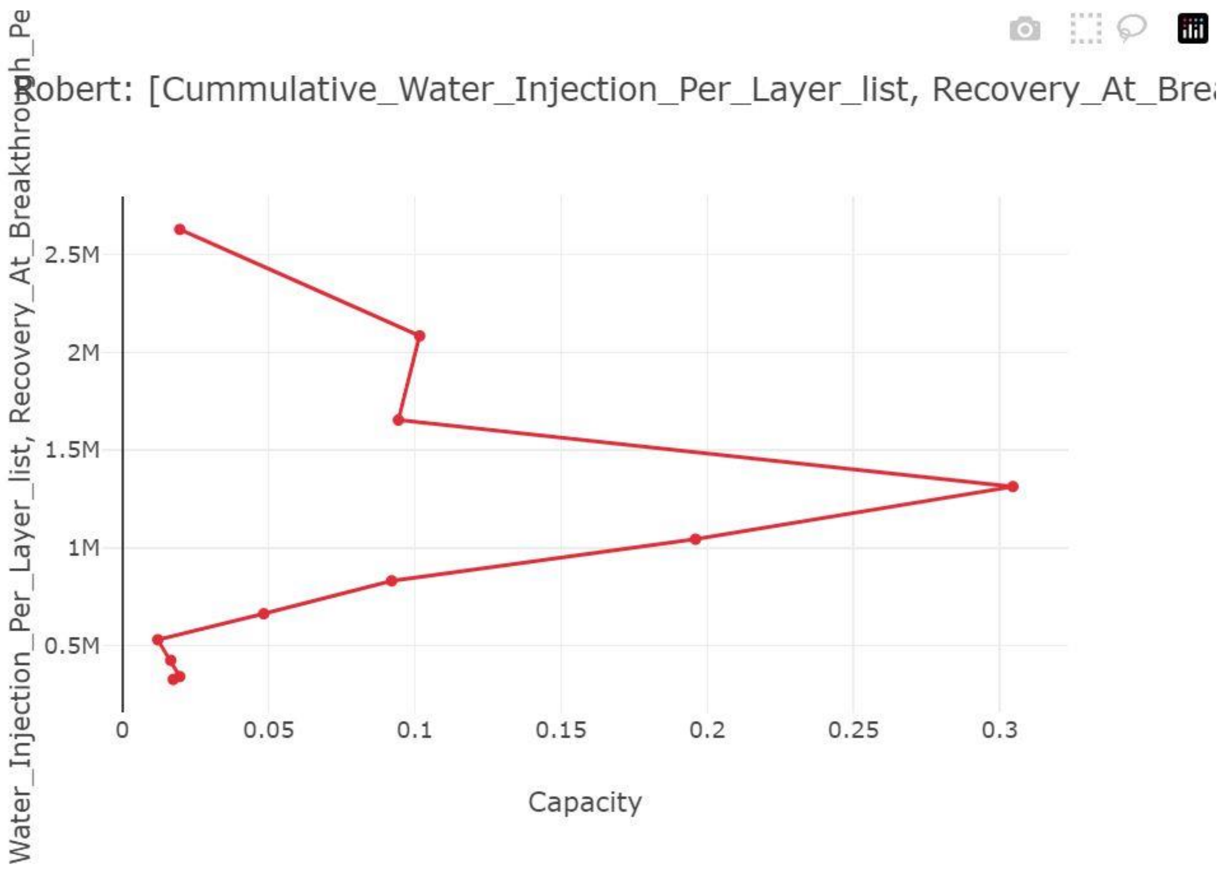


Figure 4.1–Plot of water injection per layer against Capacity

Oil Production after breakthrough Vs time to each point

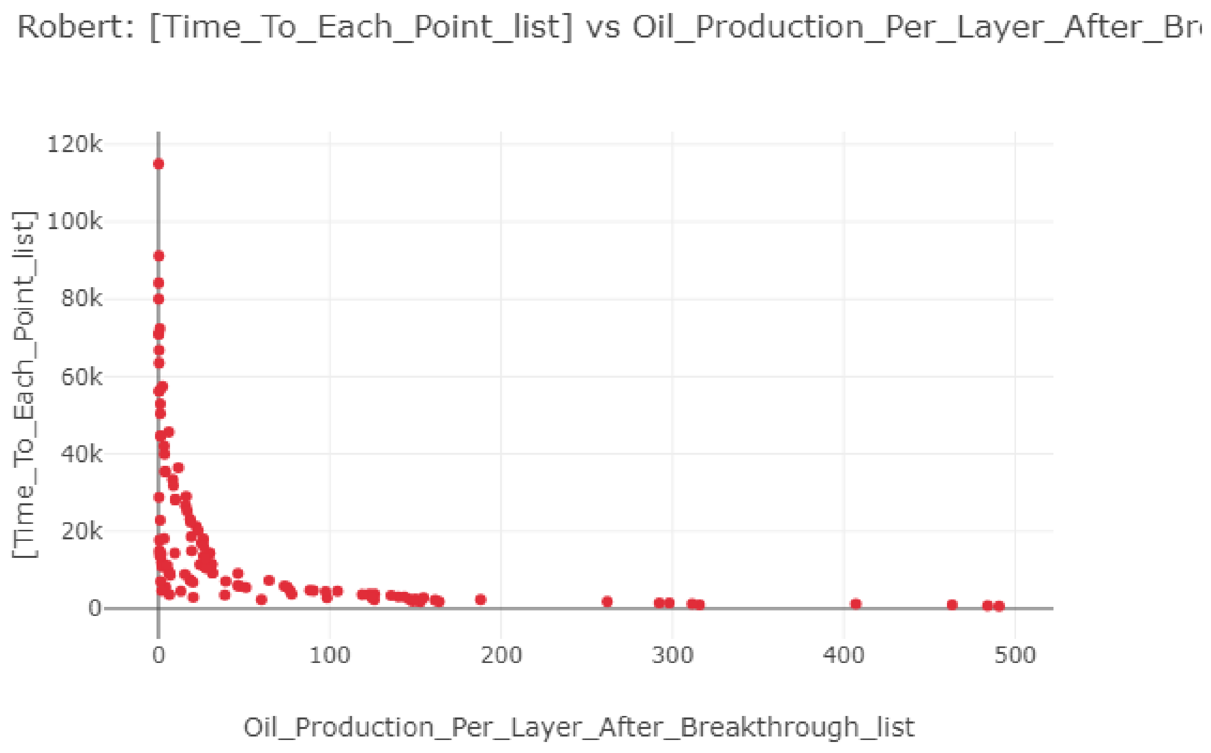


Figure 4.2 – Oil Production after breakthrough Vs time to each point

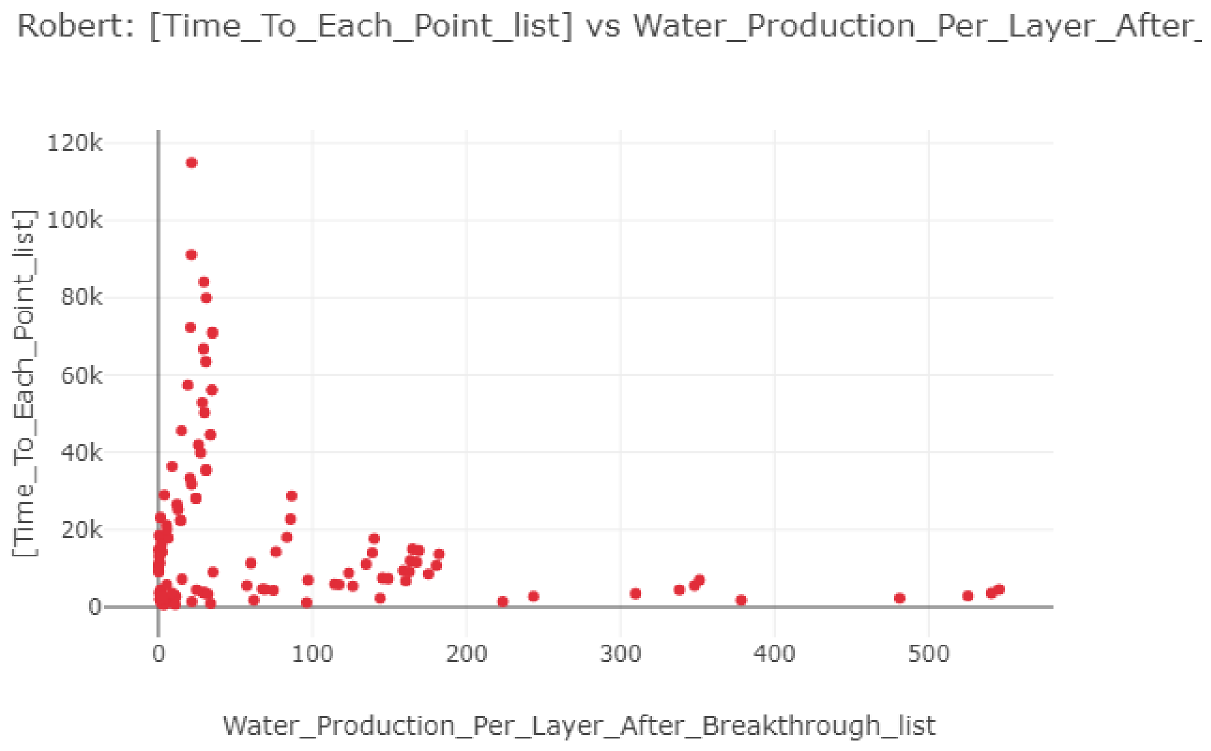


Figure 4.3 – Water Production vs time.

5 Conclusion and Recommendation

5.1 Conclusion

waterflooding is a crucial technique in enhanced oil recovery, significantly impacting the petroleum industry's ability to maximize resource extraction. As operators face the challenges posed by declining conventional reserves, the importance of effective waterflooding strategies becomes increasingly evident. Robert's method of waterflooding represents a significant advancement in this field, providing a robust analytical framework for optimizing water injection processes.

By leveraging mathematical modeling and sophisticated computer implementations, Robert's method enhances the efficiency and effectiveness of waterflooding operations. The adoption of advanced techniques like Robert's method will play a pivotal role in shaping the future of oil recovery, ensuring that valuable resources are extracted sustainably and economically.

5.2 Recommendation

Robert's method of waterflooding calculation integrates multiple parameters that influence water flooding, enabling a comprehensive assessment of reservoir performance. Its capacity to incorporate real-time data significantly enhances prediction accuracy and operational efficiency, facilitating dynamic adjustments to water injection strategies. Furthermore, Robert's model is versatile and applicable to various reservoir types and conditions, making it a valuable tool across diverse geographical locations and geological settings. A computer program that embeds all methods of predicting waterflooding performance can be developed, which can optimize the management of waterflooding operations and improve overall recovery outcomes.

APPENDIX

Python Codes

```
#Water Saturation
import numpy as np
from scipy.optimize import*
import math
import random
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd

SW = np.array([0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65])
SW_table = pd.DataFrame(SW, columns = ['SW'])
#print(SW_table)
# Relative permeability of Water
KRW = np.array([0, 0.003, 0.008, 0.018, 0.035, 0.054, 0.08, 0.105, 0.14,
0.18])

# Relative permeability of Oil
KRO = np.array([1, 0.68, 0.46, 0.32, 0.2, 0.124, 0.071, 0.038, 0.017, 0])

# Water and Oil Viscosity
VISW = 0.95 # unit is in centipoise
VISO = 3.6 # unit is in centipoise

# Initial water saturation
SWI = 0.2

# Using the correlation between relative permeability ratio and water
saturation
#print('Correlation:\n Kro/Krw = aexp(-bSw)\n')

# Calculating the coefficient b
b = (np.log((KRO/KRW)[2])-np.log((KRO/KRW)[3]))/(SW[3]-SW[2])
#print('b is:\n ', b)
#=========================================================================

# Calculating the coefficient a
a = (KRO/KRW)[2]*math.exp(b*SW[2])
#print('a is:\n ', a)
#=========================================================================
# Calculating the fractional flow
def fw(SW):
fw = 1/(1+a*(VISW/VISO)*np.exp(-b*SW))
return(fw)
#=========================================================================
''' To calculate a suitable slope for the tangent to the fractional flow
curve
Drawn from the initial water saturation'''

''' STEP1: Generate a list of uniformly distributed random numbers from a
water saturation
# greater than the initial water saturation to 1'''
xList = []
```

```
for i in range(0, 10000):
x = random.uniform(SWI+0.1, 1)
xList.append(x)
xs = np.array(xList)

'''STEP2: Calculate different slopes of tangents or lines intersecting the
fractional
flow curve using the array generated in step 1 as the water saturation.'''
m = 1/((xs-SWI)*(1+(VISW/VISO)*a*np.exp(-b*xs)))

'''STEP3: Calculate the maximum slope from different slopes generated in step
2.
The value of this slope will be the slope of the tangent to the fractional
flow
curve.'''
tangent_slope=max(m)
#print('slope of the tangent line is:\n ',tangent_slope)
#============================================================================
# Calculate the breakthrough saturation.
Saturation_at_Breakthrough = SWI + 1/tangent_slope
print(Saturation_at_Breakthrough)
#print('saturation at breakthrough is:\n ', Saturation_at_Breakthrough)
#=============================================================================
# Calculating the saturation at the flood front

def funct(SWF):
swf = SWF[0]
F = np.empty((1))
F[0] = ((tangent_slope*(swf-SWI)*(1+(VISW/VISO)*a*math.exp(-b*swf)))-1)
return F
SWF_Guess = np.array([SWI+0.1])
SWF = fsolve(funct, SWF_Guess)[0]
SWF
#==============================================================================
# Fractional flow at the flood front
Fwf = fw(SWF)
Fwf
#==============================================================================
=
# Fractional flow
Fw = fw(SW)
Fw_table = pd.DataFrame(Fw, columns = ['Fractional Flow (Fw)'])
#print(Fw_table)
#==============================================================================
=
# Calculating the differential of the fractional flow equation
def dFw_dSw(Sw):
dfw_dSw = (VISW/VISO)*a*b*np.exp(-Sw*b)/(1+(VISW/VISO)*a*np.exp(-Sw*b))**2
return dfw_dSw
dfw_dSw_table = pd.DataFrame(dFw_dSw(SW), columns = ['dFw/dSw'])
print(dFw_dSw(SW))
#print(dfw_dSw_table)
#==============================================================================
# Generating the data for the tangent plot
tangent = (SW-SWI)*tangent_slope
tangent_table = pd.DataFrame(tangent, columns = ['Tangent'])
#print(tangent_table)
```

```
#==============================================================================
==
'''Draw several tangents to the fractional flow curve at Sw values greater
than the
breakthrough saturation. Determine Sw and dFw/dSw and corresponding to these
values.
Plot fw' versus Sw and construct a smooth curve through the points '''
# Sw greater than SwBT
from numpy import*
Sw_greater_SwBT = arange(Saturation_at_Breakthrough+0.01,SW[len(SW)-1],0.01)
dFw_dSw_greater_SwBT = dFw_dSw(Sw_greater_SwBT)
print(dFw_dSw_greater_SwBT)
#==============================================================================
Fractional_flow_table = pd.concat([SW_table, Fw_table, dfw_dSw_table,
tangent_table], axis=1)
#print(Fractional_flow_table)
#==============================================================================
=
# Making the plots

fig, ax = plt.subplots(constrained_layout=True)
fig.set_figheight(4)
fig.set_figwidth(7)
fractional_flow_curve = ax.plot(SW, fw(SW), 'b', label = 'Fractional Flow
(Fw)')
tangent_curve = ax.plot(SW, tangent, 'k--')
ax.set_ylabel("Fractional Flow (fw)",fontsize=14)
ax.set_xlabel("Water Saturation (Sw)",fontsize=14)
ax.set_ylim([0,1])
ax.set_xlim([0,1])
# twin object for two different y-axis on the same plot
ax2=ax.twinx()
# make a plot with different y-axis using second axis object
dFw_dSw_curve = ax2.plot(SW, dFw_dSw, 'r', label ='dFw/dSw')
ax2.set_ylabel("dfw/dSw",fontsize=14)
ax.grid(True)
ax2.legend()
ax.legend(loc='upper left')
ax.annotate(" (Swf, Fwf)", (SWF, Fwf))
ax.annotate(" SwBT", (Saturation_at_Breakthrough, 1))
plt.show()

#Making the plots of Sw_greater_SwBT and dFw_dSw_greater_SwBT
dFw_dSw_Sw_greater_SwBT = plt.plot(Sw_greater_SwBT, dFw_dSw_greater_SwBT)
plt.show()

#==============================================================================
=
#Calculating capacity
Bed_data =
pd.read_csv("C:/file_location/Permeability_Porosity_distribution_data.csv")
permeability = Bed_data['PERMEABILITY']
thickness = Bed_data['THICKNESS']
porosity = Bed_data['POROSITY']
Capacity=permeability*thickness
Fraction_of_total_Capacity= Capacity/sum(Capacity)
```

```
#===========================================================================
=
#Calculating Injection Rate
Injection_Rate = 1800 #STB/d
Injection_Rate_Per_Layer = Injection_Rate*Fraction_of_total_Capacity

#===========================================================================
=
#Calculating Water injection rate per layer
Length = 2896 #feet
Breadth = 2000 #feet
Area = Length*Breadth

Cummulative_Water_Injection_Per_Layer_list = []
for j in range(len(thickness)):
Cummulative_Water_Injection_Per_Layer =
7758*Area*thickness[j]*porosity[j]/dFw_dSw_greater_SwBT
Cummulative_Water_Injection_Per_Layer_list.append(Cummulative_Water_Injection
_Per_Layer)
print(Cummulative_Water_Injection_Per_Layer_list)

#===========================================================================
=
#Oil Production Rate Before Breakthrough
Oil_Formation_Volume_Factor = 1.11 #RBL/STB
Oil_Production_Before_Breakthrough =
Injection_Rate_Per_Layer/Oil_Formation_Volume_Factor

#Oil Production Rate After Breakthrough
Oil_Production_Per_Layer_After_Breakthrough_list = []
for j in range(len(thickness)):
Oil_Production_Per_Layer_After_Breakthrough =
Oil_Production_Before_Breakthrough[j]*(1-Fw)
Oil_Production_Per_Layer_After_Breakthrough_list.append(Oil_Production_Per_La
yer_After_Breakthrough)

#Water Production
Water_Production_Per_Layer_After_Breakthrough_list = []
for j in range(len(thickness)):
Water_Production_Per_Layer_After_Breakthrough =
Injection_Rate_Per_Layer[j]*Fw
Water_Production_Per_Layer_After_Breakthrough_list.append(Water_Production_Pe
r_Layer_After_Breakthrough)

#Calculate the recovery at breakthrough and the time to breakthrough for each
layer
Recovery_At_Breakthrough_Per_Layer_list = []
for j in range(len(thickness)):
Recovery_At_Breakthrough_Per_Layer =
7758*Area*thickness[j]*porosity[j]*(Saturation_at_Breakthrough -
SWI)/Oil_Formation_Volume_Factor
Recovery_At_Breakthrough_Per_Layer_list.append(Recovery_At_Breakthrough_Per_L
ayer)

#Time to Breakthrough for each layer
#Water injection at Breakthrough
Cummulative_Water_Injection_Per_Layer_At_Breakthrough_list = []
```

```
for j in range(len(thickness)):
Cummulative_Water_Injection_Per_Layer_At_Breakthrough =
7758*Area*thickness[j]*porosity[j]/dFw_dSw(Saturation_at_Breakthrough)
Cummulative_Water_Injection_Per_Layer_At_Breakthrough_list.append(Cummulative
_Water_Injection_Per_Layer_At_Breakthrough)
Time_To_Breakthrough_For_Each_Layer =
Cummulative_Water_Injection_Per_Layer_At_Breakthrough/permeability[j]

#Oil recovery and time to each point.
Oil_Recovery_To_Each_Point_List = []
for j in range(len(thickness)):
Oil_Recovery_To_Each_Point = 7758*Area*thickness[j]*porosity[j]*(SW -
SWI)/Oil_Formation_Volume_Factor
Oil_Recovery_To_Each_Point_List.append(Oil_Recovery_To_Each_Point)
Time_To_Each_Point = Cummulative_Water_Injection_Per_Layer/permeability[j]
```

REFERENCES

Akhmetov, A., et al. (2018). "Fundamentals of Waterflooding in Oil Reservoirs." *Journal of Petroleum Science and Engineering*, 171, 113-124.

B. C. Craft, M. H. (1991). *Applied Petroleum Reservoir Engineering.pdf*.

Buckley, S. E. and Leverett, M. C. (1942) Mechanism of Fluid Displacement in Sands. Trans., AIME 146,107-16.

Butler, R., et al. (2013). "Waterflooding: Principles and Practices." *Society of Petroleum Engineers*.

Chen, J., et al. (2019). "Optimization of Waterflooding in Heterogeneous Reservoirs." *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, 41(3), 256-267.

C.H Wu. (1988). Waterflood Performance Projection Using Classical Waterflood Models Compared with Numerical Model Performance jpt forum.

D. Tiab, (1986). SPE 15020 Extension of the Dykstra-Parsons Method to Layered-Composite Reservoirs.

Dykstra, H. and Parsons, H.: "The Prediction of Oil Recovery by Waterflooding, " Secondary Recovery of Oil in the United States, API, New York City (1950) 160-74.

Enick, R. M., Pittsburgh, U., Reznik, A. A., Pittsburgh, U., Miller, R. A., & Pittsburgh, U. (1988). The Statistical and Dimensionless- Time Analog to the Generalized Dykstra-Parsons Method. February 313–319.

Farrell, D. (2019). *pandastable Documentation*.

Field, P., Saavedra, N. F., Peralta, R. C., Ltda, D. T. H., Cobb, W., & Cobb, W. M. (2003). SPE 81042 Distribution of Injected Water by Using CGM Method: A Case History in Palogrande-Cebu Field.

Gasimov, R. R. (2005), Modification of the Dykstra-Parsons Method to Incorporate Buckley-Leverett displacement Theory for Water Floods. Thesis Submitted to the Office of Graduate Studies of Texas A & M University in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE August 2005 Major Subject: Petroleum Engineering.

Gomez, V., Gomez, A., & Duran, J. (2009). SPE 121854 Analytical Simulation of the Injection / Production System of La Cira East and North Areas Using CGM Method.

Jensen, J. L., & Currie, D. (1990). A New Method for Estimating the Dykstra · Parsons Coefficient to Characterize Reservoir Heterogeneity. August, 369–374.

Johnson, C. E. (1956). Prediction of Oil Recovery by Water Flood - A Simplified Graphical Treatment of the Dykstra-Parsons Method. 2–3.

Lake, L. W., et al. (2014). "The Physics of Fluid Flow in Porous Media." *Petroleum Engineering Handbook*, Volume II.

Lewis, E., Dao, E. K., & Mohanty, K. K. (2008). Sweep Efficiency of Miscible Floods in a High-Pressure Quarter-Five-Spot Model. May, 24–27.

Li, D. (2004). SPE 88459 Comparative Simulation Study of Water Flood. 1–10.

Mahfoudhl, J. E., Enick, R. M., & Pittsburgh, U. (1990). Extension of the Generalized Dykstra-Parsons Technique to Polymer Flooding in Stratified Porous Media. August, 339–345.

Morrison, G. R. (n.d.). SPE 97645 Dykstra Parsons Water Flood Theory Adapted to Chemical Flood Modelling.

Nyga, J. I. (2020). Simulation of Immiscible Water-Alternating-Gas Injection in a Stratified Reservoir: Performance Characterization Using a New Dimensionless Number. November 2019, 1711–1728.

Omar, A. I., Chen, Z., & Khalifa, A. E. (2017). Predicating Water-flooding Performance into Stratified Reservoirs Using a data driven proxy model. 8(7), 60–78. https://doi.org/10.5897/JPGE2016.0240

Pawar, S., et al. (2020). "Forecasting Oil Production Using Advanced Waterflooding Techniques." *Journal of Petroleum Technology*, 72(12), 45-56.

Popa, A. S., Sivakumar, K., & Cassidy, S. (2012). SPE 154302 Associative Data Modeling and Ant Colony Optimization Approach for Waterflood Analysis. 1–14.

Reznik, A. A., Robert M. Enick, & Sudhir B. Panvelker. (1984). An Analytical Extension of the Dykstra -Parsons Vertical Stratification Discrete Solution to a Continuous, Real-Time Basis.

Richardson, J. G. (1957). The Calculation of Waterflood Recovery from Steady-State Relative Permeability Data. 64–66.

Santos, J., et al. (2017). "Reservoir Pressure Maintenance: Challenges and Solutions." *Energy Sources, Part B: Economics, Planning, and Policy*, 12(6), 554-570.

Sheng, J. (2015). "Modern Waterflooding: Technology and Applications." *Journal of Natural Gas Science and Engineering*, 23, 123-134.

Singh, S. P., & Kiel, O. G. (1982). Waterflood Design (Pattern, Rate, and Timing).

Stiles, W.E. (1949). Use of Permeability Distribution in Water-flood Calculations. Trans., AIME 186,9-13.

Warren, J.E., (1964), and Cosgrove, J.J.: Prediction of Waterflood Behavior in a Stratified System. Soc. Pet. Eng. · J., 149-15 7.

Welge, H. J. (1952). A Simplified Method for Computing Oil Recoveries by Gas or Water Drive. Trans. AIME, 195, 91-98.

Yazdani, S., et al. (2018). "Software Implementation of Enhanced Waterflooding Methods." *Journal of Petroleum Science and Engineering*, 172, 135-145.

Zhao, L., Li, L., Wu, Z., & Zhang, C. (2016). Analytical Model of Waterflood Sweep Efficiency in Vertical Heterogeneous Reservoirs under Constant Pressure. 2016.