

**A NUMERICAL SIMULATOR TO ESTIMATE
HYDROLOGICAL PARAMETERS OF A FRACTURED POROUS MEDIUM
USING FLUORESCEIN THERMAL DECAY CORRECTION**

BY

**NG'ANG'A BENSON WANG'OMBE, B. Ed (Science)
REGISTRATION NUMBER I56/10390/2007**

A thesis submitted in partial fulfilment for the requirements of the degree of Master
of Physics of Kenyatta University.

Ng'ang'ga Benson
*A numerical simulator
to estimate*



2012/383398

KENYATTA UNIVERSITY LIBRARY

OCTOBER 2011

DECLARATIONS

This thesis is my original work and has not been presented for the award of a degree at any other university.

Ng'ang'a Benson Wang'ombe

Signature.....

Date.....25-10-2011

This thesis has been submitted for examination with our approval as university supervisors.

Supervisors

1. **Dr. Willis J Ambusso**

Physics Department

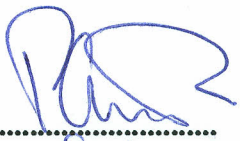
Kenyatta University

Signature.....

Date.....25/10/2011

2. **Dr Peter Omenda**

Kenya Geothermal Development Company

Signature.....

Date.....25/10/2011

DEDICATION

This thesis is dedicated to my children Eunice (8), Lucy (10) and Peter (12) who have eagerly looked forward to my graduation, like Isaac Newton said; hope God will help them stand on the shoulders of giants. Ann Wanjiru for moral support during my low point on 2.2.2011.

ACKNOWLEDGEMENTS

Like Mwawongo said, I feel honoured to be a student of Dr. Ambusso, Dr. Omenda and a “grand student” of professor Rathore. Particularly, Dr. Ambusso who in the course of this study was always available and accessible at all time. I owe him all my computer application skills that I have acquired from “control copy” to programming in C++. I am greatly indebted to him for knowledge. Lastly I remember him for his uncommon push to complete this study whenever I relaxed.

I appreciate the following personalities, Professor Rathore who remained fatherly during the post election violence especially in my home area, Molo and his guidance and drive of my proposal to graduate school. Professor Okumu for his assurance that we would still pass his complicated Quantum Mechanics. Dr. Charles Migwi, Chairman Physics Department, whose office was open to us, at all time. My former Headmistress Edna C Ruto in Moi Kipsitet Girls Secondary school, Kericho, who beat odds by allowing me to proceed for study leave despite shortage of teachers in the school.

Lastly I acknowledge the physics department community who always treated me as one of their own.

TABLE OF CONTENTS

TITLE.....	i
DECLARATIONS.....	ii
DEDICATIONS.....	iii
ACKNOWLEDGMENTS.....	iv
TABLE OF CONTENTS.....	v
LIST OF FIGURES.....	vii
ABSTRACT.....	viii
CHAPTER ONE.....	1
1.0 INTRODUCTION	1
1.1 Background to the study.....	1
1.2 Statement of the research problem.....	6
1.3 Objectives of the research project.....	6
1.3.1 Main objective	6
1.3.1.1 Specific objectives	6
1.4 Rationale of the research project	6
CHAPTER TWO.....	7
2 LITERATURE REVIEW.....	7
2.1 General overview	8
2.2 Properties of fluorescein and its application as a geothermal tracer	9
2.2 Re-injection and tracer tests history of Olkaria, Kenya	11
2.4 Previous attempt to correct fluorescein thermal decay in Olkaria	14
CHAPTER THREE.....	17
3 THEORY BEHIND THE PROBLEM.....	17
3.1 Introduction	17
3.2 The residence time distribution	17
3.2 Transport by Advection and by Diffusion.....	19
3.4 The governing equations	21
3.4.1 The Darcy Equation in geothermal context	21
3.4.2 Derivation of the Pressure and Chemical Balance Equations	22
CHAPTER FOUR	25
4 RESEARCH MATERIALS AND METHODOLOGY.....	25
4.1 Introduction	25
4.2 The conceptual model	25
4.3 Numerical solution implementation.....	27
4.3.1 Discretization of the reservoir.....	27

4.3.2 Discretization of the governing equations and the Gauss Seidal method of solution.....	28
4.3.3 The material balance Equation.....	29
4.3.4 The Tracer decay equation	32
4.3.5 The Tracer balance Equation	32
4.3.6 The Numerical Simulator structure.....	34
4.3.6.1 Reservoir Dimensions (reservoir.txt file)	35
4.3.6.2 Rock Types	35
4.3.6.3 The property table	35
4.3.6.4 The computer program	36
CHAPTER FIVE.....	44
5 RESULTS AND DISCUSSIONS.....	44
5.1 Introduction	44
5.2 Generic Tests.....	44
5.2.1 Pressure test	45
5.2.2 Tracer test.....	47
5.2.2.1 Concentration levels in neighbouring blocks.....	47
5.2.2.2 Effect of advection and dispersion transport mechanisms.....	49
5.2.2.4 Effect of increasing porosity and fracture size.....	51
5.3 The process of matching the simulated results with field data.....	52
5.3.1 The field data	53
5.4 Tracer Decay correction.....	54
5.5 Model validation.....	55
5.6 Determination of the fracture Parameters.....	56
5.6.1 Porosity.....	57
5.6.2 Permeability.....	57
5.6.3 Thickness of the fracture	57
5.6.4 Extent of the fractured media.....	57
5.6.4.1 Reservoir pore volume (overall reservoir volume).....	57
5.6.4.2 Rate of aquifer flow.....	58
CHAPTER SIX.....	60
6 CONCLUSIONS AND RECOMMENDATIONS.....	60
6.1 CONCLUSION	60
6.2 RECOMMENDATIONS	61
REFERENCES.....	62
APPENDIX 1; THE COMPUTER PROGRAM	64
APPENDIX 11; FIELD DATA OW-18 AN OW-19	64

LIST OF FIGURES

Figure 1-1. Greater Olkaria geothermal field (GOGA) (Ofwona, 2003).....	5
Fig 2-3; Fluorescein thermal decay correction for well OW-18 and OW-19 respectively (Mwawongo 2004).....	15
Figure 3-1: Variation in concentration of a tracer flowing through a fracture for different media.....	18
Figure 3-2; Illustration of pore channel velocity.....	20
Fig 3-3; Fracturing in rocks (Gregory, 2006).....	20
Fig 3-4; Illustration of the Darcy law.....	21
Fig 4-3; Discretization of the reservoir.....	28
Figure 4-4; Block j and its neighbours.....	29
Fig 4-5; Flow of information in the numerical simulator.....	34
Figure 4-6; The output file of reservoir dimensions function.....	37
Fig 4-7; The output of the function SetBlockDimensions.....	37
Figure 4-8; Illustration of how block positions were set.....	38
Figure 4-9; The output of SetBlockPositions function showing positions of the each block.....	38
Fig 4-10; Setting up the grid and identifying the neighbours of each block.....	39
Fig 4-11; Viscosity against temperature graph.....	40
Fig 4-13; Viscosity from temperature 340°C to 370°C.....	41
Fig 5-2; Pressure in injector block and its neighbour before injection.....	46
Fig 5-4; Tracer concentration in block 66 next after the injector block.....	47
Fig 5-5; Variation in concentration of the tracer with distance.....	48
Figure 5-6; Tracer concentration in symmetrical blocks, blk66 is symmetrical to blk68, blk65 with blk69. Injector is block 67.....	48
Fig 5-7; 1. Effect of transport by both advection and dispersion. 2. Transport by advection only and 3. Transport by dispersion only.....	50
Fig 5-8; Effect of different levels of porosity on concentration of the tracer.....	51
Fig 5-9; Effect of increasing permeability raised the tailing portion of the breakthrough curve.....	52
Fig 5-10; Match of field data and the simulated breakthrough curve.....	54
Fig 5-11; Fluorescein thermal decay correction.....	55
Figure 5-12; A plot of field data versus simulated results.....	56
Figure 5-13; Matching simulated results on the field data.....	56
Figure 5-14; Y-intercept of the extrapolated linear tailing portion give steady state concentration.....	58
Fig 5-15; The slope of the linear tailing portion represent rate of aquifer flow.....	59

ABSTRACT

A method that can be used to estimate hydrological parameters of a fracture using fluorescein thermal decay correction is presented. These parameters are porosity, permeability, fracture size, entire reservoir pore volume and reservoir recharge rate. A tracer was pumped into an injection well and its arrival characteristics with respect to time in the production wells monitored. The data was used to generate breakthrough curves which gives tracer effluent history. These curves yielded important information about the reservoir. Resulting fluorescein decays were accounted for by application of numerical modelling and properties of the geothermal system were estimated. The estimates were used to simulate a fracture connecting two wells. A mathematical model describing the flow of the tracer was solved to generate its flow profile in a fracture. This was done by developing a computer package in Microsoft Visual C++. The profile was matched with the field data by trying out different values of the fracture parameters. The values that almost matched the profile on the field data curve were the required parameters. Porosity was observed to vary between 13 % to 15 %, permeability was found to be 1.8 mm^2 and permeability thickness of 950 m^2 . The fracture thickness was estimated to vary between 3 m and 10 meters. Rate of aquifer flow was $1.956 \times 10^{-9} \text{ kg/m}^3/\text{sec}$ while entire reservoir pore volume was obtained to be 653520 litres/sec

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background to the study

Hydroelectricity is not a reliable source of energy due to changing rainfall and climatical patterns. On the other hand electricity from hydrocarbons is expensive and leads to depletion of ozone layer. The best alternative is therefore geothermal potential, especially for countries traversed by the Great Rift Valley. Kenya and Ethiopia have well tapped geothermal power as compared to their neighbours in Eastern Africa; the task remains how its exploitation can be sustained (Karsten, 2002). This can be done through proper re-injection strategies.

Re-injection started as a waste management scheme but experience has shown that it is a requirement for optimum operation of a field and calls for better understanding of hydrological characteristics of a reservoir. Reinjection is critical in maximizing power production and extending the lifetime of a reservoir. Based on this grounds tracers are injected to provide a record of what happens underground when a fluid flows between two or more wells (Gudmundsson et al, 1985). This is achieved by generating a tracer breakthrough curve which is simply a graph of concentration of the tracer against time. Tracer survey provides a number of information regarding bulk and regional formation properties of a reservoir, and by extension geological structures over a wide area than any other field test, which otherwise to obtain could be an expensive undertaking (Ambusso, 2007).

Traditionally tracers have been used for the purpose of tracking down steam, liquid and two phase flow from one location to another such as from an injection to a production well. The primary emphasis in tracer research have been to find compounds that will act as conservative tracers, that is, compounds that will not be removed from the fluid by chemical reaction or delayed by retention on the solid. A number of tracers have been developed for use in geothermal systems which include natural tracers and introduced tracers like Halogenated Alkenes, Halides, Radioisotopes, Sulphur Hexafluoride SF₆, two phase tracers and fluorescent dyes

Natural tracers are indigenous to the system, they already occur in the fluid in some part of the system and the changes in them from one zone to the other may be relevant. These changes may give some sort of time scale for the movement of fluid from one location to another, or they may indicate a mixing of fluids from different zones or the dilution of thermal waters with cold waters. For example, the low chloride content of cool groundwater surrounding a geothermal reservoir may ultimately be the tracer that indicates that the recharge water is reaching the production zone. Examples of natural tracers include hydrogen and oxygen isotopes like tritium, deuterium and oxygen-18 (Grant, 1982).

Introduced tracers are the most common means of tracking the diverse nature of a reservoir. Some of such tracers include Halogenated compounds which are volatile although some are stable at high temperatures and have a disadvantage in that they require specially equipped gas chromatography for analysis. Some of them especially the ones that contain chlorine and bromine for example

Dichlorodifluoromethane impart the worst ozone layer depletion. The most successful artificial tracers, chlorofluorocarbons, were taken off the market because of their deleterious effect on ozone concentrations in the upper atmosphere (UNEP, 1993). R-134a ($\text{CF}_3\text{CH}_2\text{F}$, 1,1,1, 2-Tetrafluoroethane) and R-23 (CHF_3 , Trifluoromethane), both hydrofluorocarbons, were proposed in 1997 as substitute geothermal tracers for the chlorofluorocarbons (Adams, 1997).

The halides are stable and inert but they are toxic and have natural background. Radioisotopes are detectable at low concentrations but toxic and have natural background count (Tester et al 1986). Two phase tracers have a weakness in that they do not always follow the same path as the injected water that does not immediately boil as it enters the reservoir. Alcohols especially ethanol are a class of compounds that form a good candidate of two phase tracers, they are stable at geothermal temperatures and have low toxicity. Fluorescent dyes have a well defined kinetics, detectable at low concentrations and simple field analysis. Examples include fluorescein and rhodamine-WT which decay rapidly, at high temperatures. Fluorescein was the subject of this research.

Fluorescein is an organic dye and has been used as a tracer to monitor fluid return rates, preferential flow paths and well output characteristics of a reservoir. Its wide applications in geothermal systems can be attributed to its low detection limits, inexpensive, strong colour at low concentrations and ease of analysis using fluorimeters. However it is subject to thermal decay at elevated temperatures (Adams and Davis, 1991). Thus its breakthrough curve should be corrected for thermal degradation if at all accurate fracture

parameters are to be obtained. Fluorescein can be used as a relatively conservative tracer in reservoirs with temperatures less than 210°C, where the duration of the test is expected to last less than a month. At reservoirs with temperatures above 210°C, decay is significant and Olkaria is one of them with temperatures as high as 300°C. Therefore, finding how to correct this decay using numerical modelling was the subject of this study.

Previously, attempts to correct fluorescein thermal decay especially in Olkaria had been made by Mwawongo (2004). In his work he obtained a breakthrough curve with a rising tail that could not be explained and his porosity was relatively high up to 50% of the rock and this was not realistic, indicating that tools in the market are not satisfactory.

Olkaria geothermal field is located in Naivasha along the Rift Valley and 120km northwest of Nairobi, Kenya. Also referred as to the Greater Olkaria Geothermal Area (GOGA), constitutes of seven geothermal fields namely the Olkaria North East, Olkaria North West, Olkaria East, Olkaria Central, Olkaria South East, Olkaria South West and Olkaria Domes geothermal fields (Figure 1-1). This study was based in this field (specifically Olkaria East) where re-injection has been done since 1987, aimed at disposing used brine, re-pressurizing the reservoir, preventing subsidence, enhancing heat recovery, providing barrier for cold water inflow and reducing steam decline rates (Kariuki, 2004). Cold re-injection have been found applicable only along the peripheral of the field primarily to dispose waste water otherwise its reinjection in the middle of the field is considered risky as it could cool the formation.

1.2 Statement of the research problem

Traditionally tracers have been used to determine fluid flow paths, return rates and existence of fractures but a method to quantify the degree of hydrologic parameters like porosity have been elusive especially within Olkaria. In other words, Olkaria being a high temperature field, fluorescein cannot be used to elucidate fracture parameters due to thermal degradation. A method to correct this thermal decay was therefore necessary and hence the purpose of this study.

1.3 Objectives of the research project

1.3.1 Main objective

The main objective of this study was to correct thermal decay of fluorescein by numerical modelling and use it to estimate levels of hydrological parameters of a fracture.

1.3.1.1 Specific objectives

- (i) To setup mathematical equations that governs tracer flow between two or more wells.
- (ii) To formulate and use computer program codes to solve the equations.
- (iii) Perform simulations to estimate hydrological parameters of a fracture.

1.4 Rationale of the research project

This study was intended to enhance an inexpensive single test that can be used to deduce reservoir characteristics because tools already in the market are not to the expected standards as discussed in section 3.2. Secondly fluorescein though decays at high temperatures was made to be applicable as a geothermal tracer. The primary emphasis in tracer research has been to find compounds

that are cost effective, non biodegradable, environmentally friendly and act as conservative tracers i.e. compounds that will not be removed from the fluid by chemical reaction or delayed by retention on the solid phases of the reservoir. Fluorescein is a good candidate although it decays at elevated temperatures and for tests lasting for several months. This research exploited this weakness by correcting the breakthrough curve and using it to calculate parameters of a fracture.

CHAPTER TWO

2 LITERATURE REVIEW

2.1 General overview

Traditional reservoir engineering and geology studies can establish existence of fractures but cannot provide information about their properties. Tracer tests, however, can accomplish this successfully. To avoid premature thermal breakthrough, tracers are employed for detection and evaluation of preferential path networks, to detect and quantify the timing and mass fraction of water movement from an injection to a production well. Further, they reveal information on a real coverage of water recharge and offer a unique opportunity to analyse reservoir behaviour (Shook, 2003).

Tracer tests make better re-injection strategies, hence help maintain reservoir pressure, enhance thermal recovery and eliminate possible compactional subsidence.

The term tracer signifies a material whose properties make it possible to follow the dynamic behaviour of a flowing system. Tracers are categorized as either natural or artificial tracers. Natural tracers are indigenous to the system under study while artificial tracers are deliberately introduced (Chrysikopoulos, 1992). Artificial tracers include radioactive isotopes, fluorescent dyes and substituted aromatic acids. A tracer once added should remain unchanged, providing a tag to the fluid, allowing it to be traced back to its source; however it has been found out that fluorescent dyes decay or adsorb (Adams and Davis 1991).

Tracers that display “imperfect” behaviour can be used to produce information on reservoir properties. For instance decaying tracers have been used to deduce the effective temperature of the injection –production flow path (Tester et al 1986; Adams et al 1989). The effective reservoir temperature can be obtained when two tracers with different thermal degradation are used simultaneously by monitoring the change in the ratio of the tracer concentrations. In this study fluorescein, which is an organic fluorescent dye and decays in certain conditions, will be the geothermal tracer, there exists other tracers but not considered here like rhodamine-B and rhodamine WT which decays at lower temperatures than fluorescein. However, using the tracers is expensive as it requires expensive sets of equipments.

2.2 Properties of fluorescein and its application as a geothermal tracer

Fluorescein is an organic dye with molecular formula $C_{20}H_{10}O_2$, absorbs light in the blue range of the visible spectrum with absorption peaking at 490 nm (blue) and fluoresces green in blue light. It emits light at 530 nm (yellow) and fluoresces much better under blue light. Fluorescein is normally added to rainwater in environmental testing simulation and although it is used in bulk, it is less expensive in comparison to other tracers. Behaviour of fluorescein in different situations is well known; It is resistant to biodegradation and it is not affected by variations in water chemistry (Smart and Laidlaw, 1977). Temperature and salinity has little effect on its fluorescence, however it decays at elevated temperatures.

Kinetics of Fluorescein decay and its applications as a geothermal tracer are well documented by Adams and Davis (1991). In their analysis, thermal decay

of Fluorescein is significant at high temperatures as from 210° C and for tracer tests lasting several months. It decays at constant pH governed by first order rate law given by equation 2-1

$$C = C_0 e^{-kt} , \quad 2-1$$

Where k – Thermal decay constant given by Arrhenius relation

$$k = Ae^{-E_a/RT} \quad 2-2$$

C – Fluorescein concentration (mg/l) at time t

C_0 -initial Fluorescein concentration (mg/l) at time $t=0$

A – Exponential constant equal to $= 18.25 \pm (1.44) s^{-1}$

E_a –Activation Energy = $143,300 \pm 6,620$ J/mol

R – Universal gas constant = 8.31 J/mol K

T - Sample absolute temperature (K)

According to Levenspiel (1972) fluorescein tracer history must therefore be corrected for thermal degradation using Arrhenius equation. Effective temperature of a flow path can be calculated from tracer flow data when fluorescein is used in conjunction with a tracer whose decay rates are known (Pruess et al 1984), this way other physico-chemistry parameters of injection flow path can be determined. Further, geothermal production fluid should be flashed prior to re-injection, to lower oxygen content and increase pH level, since oxygen reacts with fluorescein and its effect is greatly diminished in

alkaline fluids. Hence, because boiling lowers the oxygen concentration, fluorescein is a suitable tracer in geothermal systems. Furthermore, fluorescein has been shown not to adsorb on reservoir rock under geothermal conditions (Rose, 1999).

2.2 Re-injection and tracer tests history of Olkaria, Kenya

In Olkaria geothermal field fluorescein has been used as a tracer several times. Some of the tests were done in wells OW-3, OW-704 and OW-12 illustrated in Figure 2-1. The first of the tests was done in well OW-3 from April to September 1993 (Ambusso, 1994). For 172 days, cold fresh water at 18°C from Lake Naivasha was injected incessantly at an average rate of 100 tonnes / hour. Forty five days later 125kg of sodium fluorescein dye was introduced as a slug. Production changes were observed in wells OW-2, OW-4, OW-7, OW-8, OW-10 and OW-11. Tracer returns were observed in wells OW-4, OW-2 and OW-7 with OW-4 registering the highest of about 38% (Ofwona, 1996). Also, hot re-injection of separated brine from wells OW-27, OW-31 and OW-33 had been going on in well OW-3 since May 1995 at approximately 13 tonnes / hour.

of 100 tonnes/hour. After 27 days, 500kg of sodium fluorescein was introduced as a slug and wells close to well OW-R3 were monitored for output changes. Very little tracer returns were obtained from wells OW-33 and OW- 34.

Tracer injection test was also done in well OW-12 from 12th July 1996 to 1st September 1997. About 137,000 tons of cold water was injected at a rate of 100 tons / hour for a period of 416 days until 1st August 1997. Then, after 20 days of re-injection, 500 kg of sodium fluorescein dissolved in 20,000 litres of water was injected as a slug. All production wells were monitored daily for tracer returns, which were recovered only from wells OW-15, OW-16, OW-18 and OW-19 (Mwawongo, 2004; Ofwona, 2004). Tracer breakthrough time was 3 days in well OW-15, 46 days in well OW-16, 20 days in well OW-18 and 14 days in well OW-19. Only wells OW-18 and OW-19 recorded substantial returns, this is illustrated in Figure 2-2.

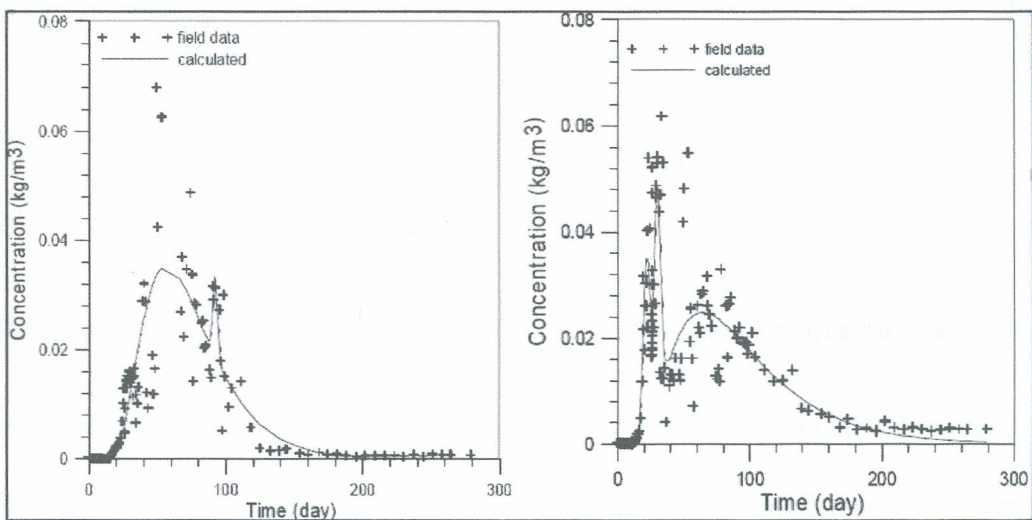


Fig 2-2; Tracer returns in well OW-18 and OW-19 respectively (Mwawongo 2004)

In all of the tracer tests done in Olkaria, they were qualitative and inadequate

- i. Attempts to correct fluorescein thermal decay failed as the rising tail could not be explained.
- ii. Obtained unrealistic high porosity of 50%.
- iii. Used 2-Dimension, which was inadequate
- iv. Worked on a homogeneous media which was not practical

2.4 Previous attempt to correct fluorescein thermal decay in Olkaria

Previously, attempts to correct fluorescein thermal decay had been made by one a Mr. Mwawongo, in his thesis (2004); he modelled tracer flow through a fracture between well OW-12 and wells OW-18 and OW-19 and corrected fluorescein thermal decay using programs called TOUGH2 and TRINV. These programs are basically based on solving the following equation

$$D \frac{\partial^2 C}{\partial x^2} = \varphi \frac{\partial C}{\partial x} + \frac{\partial C}{\partial t} \quad 3$$

whose analytical solution is

$$C(x,t) = \varphi \frac{M}{Q\sqrt{\pi Dt}} \exp \frac{-(x-\varphi t)}{4Dt} \quad 4$$

Where $\varphi = \frac{f}{\rho A \emptyset}$ - mean fluid velocity, A - cross section area of the flow channel, x-distance from injection point, t-time after injection, f - flow rate, \emptyset - porosity, ρ -fluid density, D - dispersion coefficient, C-tracer concentration at production well, c-tracer concentration at the injection well, M-mass of tracer injected.

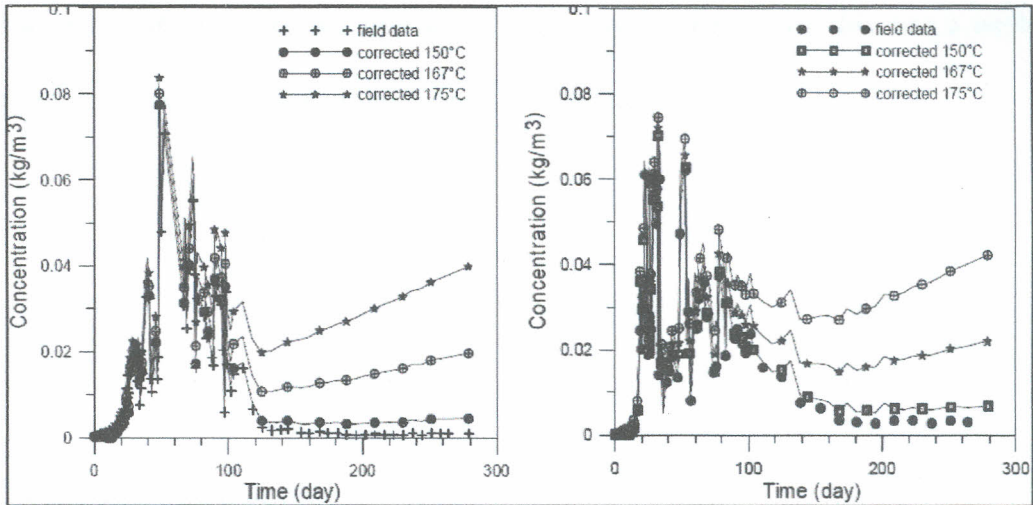


Fig 2-3; Fluorescein thermal decay correction for well OW-18 and OW-19 respectively (Mwawongo 2004)

His findings are illustrated in Figure 2-3, where he modelled tracer flow in 2-D which is inadequate compared to 3-D and used homogeneous media which is not practical. In his analysis the decay correction behaved well up to a temperature of 167° C and it was even much better at lower temperatures but above this, the rising tail could not be explained especially for a contaminant undergoing dilution, whose concentration should decrease with time. Further he obtained dimensions of the fracture as 800m high x400m long x 2m wide, with a porosity of 50%, which implies that, for every 1 cm³ rock, there exists a corresponding 1cm³ pore, such a formation has not been observed in practice and cannot be stable and there is a likelihood of subsidence. The width of the channel could not be 2 metres, probably it is this assumption that resulted to unrealistic 50 % porosity in his model in an attempt to account for the large tracer returns experienced, since as expected the narrower the conduit the larger the flow. Even from physical surface observations of the Olkaria

formation, the fracture is more than 2 metres wide. Therefore, this was a well posed problem and formed the subject of this research work.

CHAPTER THREE

3 THEORY BEHIND THE PROBLEM

3.1 Introduction

In this section residence time distribution, transport processes by advection and by diffusion, conservative, important equations and decaying tracers particularly fluorescein are discussed.

3.2 The residence time distribution

The distribution of fractures in a geothermal formation cannot be determined by direct observation, however their effects on a tracer can be analysed. Residence time distribution (Grant, 1982) is one method that can be used to do this. Residence time is the amount of time required for a volume of water (and/or contaminant) to travel through a given medium. It is a function of distance travelled and the velocity of seepage through the system. Ground water residence times are normally very long up to 10's or 1000's of years (Freeze and Cherry, 1979). For instance in Figure 3-1, the dotted curve is as a result of a one dimensional flow, through a homogeneous block, without diffusion or dispersion, a good example is flow due to a piston. All the particles cross the block at same time. Dispersion or diffusion would spread the spike (Figure 3-1)

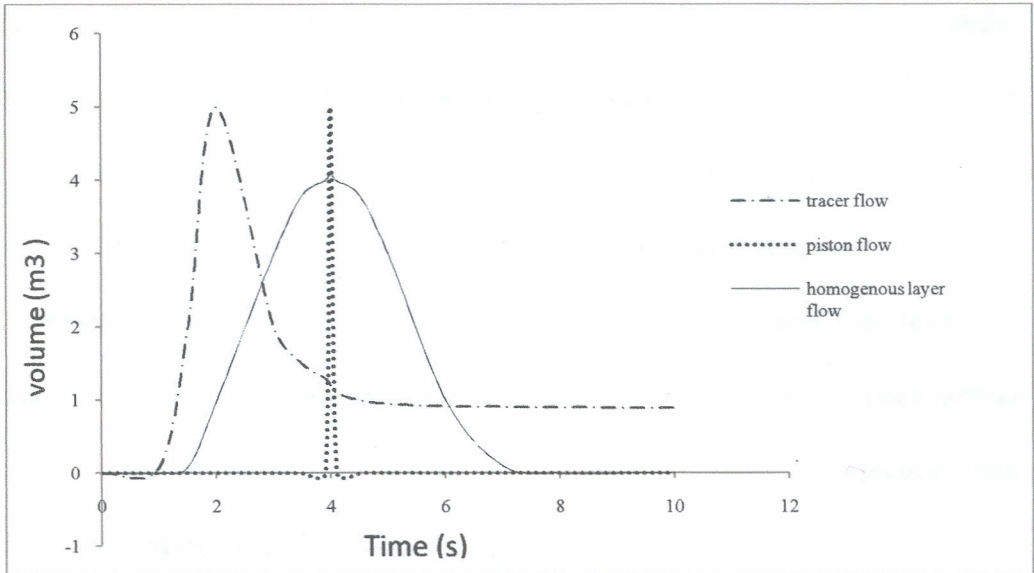


Figure 3-1: Variation in concentration of a tracer flowing through a fracture for different media.

The solid curve is produced by production and injection at two points in a layer of homogenous material. While the dashed curve is of the form assumed by a tracer, which represents an early peak then a long, slowly decaying tail. The peak and tail indicates fast transmission along the fractures and a slower transmission through the surrounding medium as some tracer enters and leaves blocks adjacent to fractures. A late peak arrival indicates low fracturing; a short tail would mean that the formation is less porous. Multiple peaks indicate multiple fluid flow paths. It is worth to note that the linear tailing portion of the return curves of a conservative tracer can be used to estimate both the effective reservoir fluid volume and the rate of reservoir recharge by surrounding aquifers (Rose, 1997).

Effects of varying reservoir properties are well documented. In his study, Ambusso (2007) analysed the behaviour of fluorescein as a conservative tracer (without thermal decay), under several scenarios. First he investigated effects of increasing porosity, diffusivity, injection rate and reservoir boundary. Increasing porosity led to increase in peak arrival time and low levels of concentrations. A decrease in diffusivity resulted in an increase in peak arrival time and a decrease in concentration. The decrease in injection rate corresponded in increase of concentration levels.

3.2 Transport by Advection and by Diffusion

There are two main transport mechanisms involved in solute transport which include advection and diffusion/dispersion. In advection, the dissolved substances are carried along with the bulk fluid flow, thus the solute travel with the velocity of the fluid. While in dispersion the solute spreads out of the path that would have been followed by advection alone. There exists 3 dispersion mechanisms namely molecular diffusion given by $(\phi\tau)$ whereby τ is tortuosity, longitudinal dispersion and pore channel velocity. In molecular diffusion the molecules spread out from regions of high concentration to those of low concentrations. Tortuosity is when molecules branch in different paths of different lengths, for instance molecules through path A cover longer distance than those through path B, (Figure 3-2).

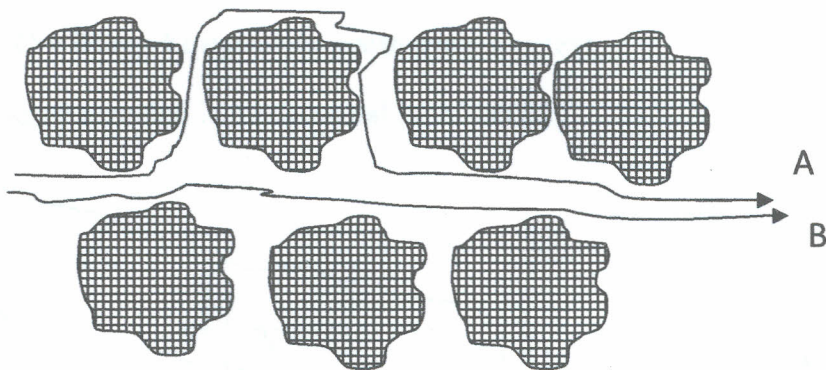


Figure 3-2; Illustration of pore channel velocity

In longitudinal dispersion pore channel velocity, molecules travel in different velocities depending on the channel

The above structure is analogous to fractured rocks as illustrated in Figure 3-3. Fractures provide permeability for fluid movement, such as water. Highly fractured rocks can make good aquifers since they possess significant permeability and porosity.



Fig 3-3; Fracturing in rocks (Gregory, 2006)

3.4 The governing equations

3.4.1 The Darcy Equation in geothermal context

The Darcy equation is fundamental in this study

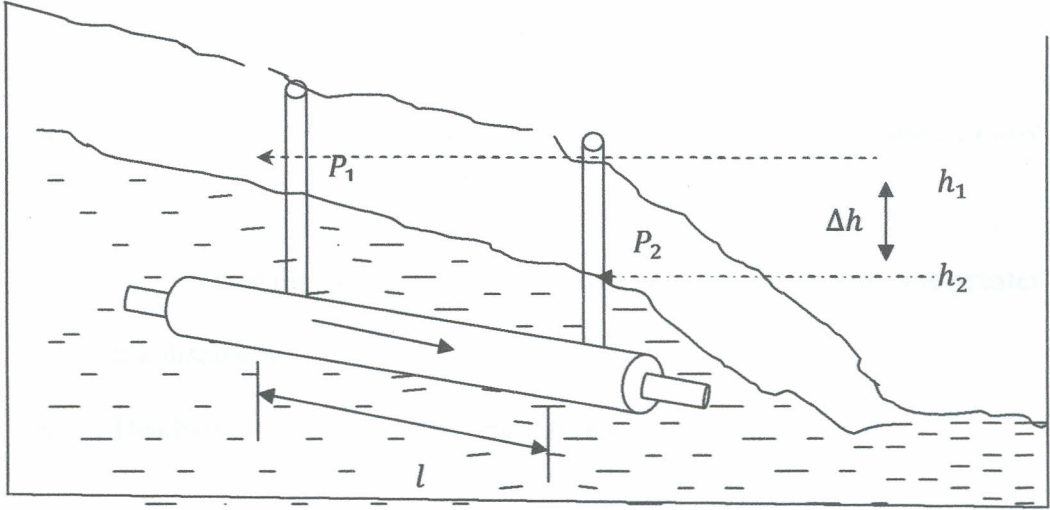


Fig 3-4; Illustration of the Darcy law

The fluid discharge in volume q (m^3/s) through a porous medium is directly proportional to product of permeability (m^2) of the medium and the pressure change due to re-injection and inversely proportional to viscosity of the liquid. Pressure drop due to elevation is taken care by the last term, Equation 3-5.

$$q = -\frac{k}{\mu_w} \left[\frac{P_2 - P_1}{l} + \frac{\rho g (h_2 - h_1)}{l} \right] \quad \dots\dots\dots 3-1$$

In three dimensions this equation becomes

$$q = -\frac{k}{\mu_w} \nabla (P + \rho_w g h) \quad \dots\dots\dots 3-2$$

Where k -permeability of the medium, P - pressure drop due to re-injection, $\rho_w g h$ -pressure drop due to change in elevation and μ_w -viscosity of water.

Darcy law is a simple mathematical statement which neatly summarises several familiar properties that ground water flowing in aquifers exhibit, including;

- i. If there is no pressure gradient over a distance, no flow occurs.
- ii. If there is pressure gradient, flow will occur from high pressure to low pressure.
- iii. The greater the pressure gradient (through same formation), the greater the discharge.
- iv. Discharge rate will be different through different formations.

3.4.2 Derivation of the Pressure and Chemical Balance Equations

The equations of fluid flow through porous media have been developed in petroleum, groundwater and soil science. Their application to geothermal reservoirs requires that in addition to fluid transport, also heat and chemical species transport equations be incorporated (Ambusso, 2007). In this research work the mass and tracer balance equations will be solved concurrently. These equations are derived from the continuity equation which states that for any quantity X, which is conserved, in a medium with sources or sinks, the flow per unit volume is given by

$$\text{Rate of gain (at one point) + net outflow} = q_s \text{ (point source)} \quad 3-3$$

$$\frac{\partial \rho}{\partial t} + \nabla(\rho q) = q_s \quad 3-4$$

where ρ and q are the density and velocity of the quantity X respectively, ρq is material flux (mass per unit time), q_s is point source or sink. Equation (3-4)

applies to a unit volume, for entire volume of rock with porosity \emptyset the equation becomes.

$$\int \frac{\partial(\emptyset\rho)}{\partial t} dV + \int \nabla(\rho_i q) dV = \int q_s dV \quad 3-5$$

Writing the first term in two phases of the fluid and changing second term to surface integral using the Gauss divergence theorem, the second term becomes;

$$\int \nabla(\rho_i q) dV = \int \rho_i q dA \quad 3-6$$

Thus equation (3-5) can be written as

$$\frac{\partial}{\partial t} \int \emptyset(\rho_i S_i) dV + \int \rho_i q dA = Q \quad 3-7$$

Where $Q = \int q_s dV$ and substituting equation 3-2 in 3-7, the mass balance equation becomes

$$\int \frac{\rho_i k_{rik}}{\mu_i} \nabla(P + \rho_i gh) dA = \frac{\partial}{\partial t} \int (\emptyset \rho_i S_i) dV + Q \quad 3-8$$

The equation of chemical species (tracer) can be obtained by substituting ρ by C in equation 3-4. The tracer transportation is due to both diffusion and advection.

$$\emptyset \int \frac{\partial C}{\partial t} dV + \int \nabla(\nabla DC) \cdot dV + \int \nabla(q \nabla C) dV = Q \quad 3-9$$

Where i is the phase, q is the fluid discharge in m^3/s , q_s is sink or source, ρ_i is the density, μ_i is viscosity, P is pressure, S is phase fraction, \emptyset is porosity, k is

permeability, Q_i represent total effect due sources of various phases, A is area, V is volume and C is concentration of the tracer.

CHAPTER FOUR

4 RESEARCH MATERIALS AND METHODOLOGY

4.1 Introduction

In this study, a numerical simulator to model tracer flow in three dimensions in a highly fractured geothermal system was developed. This was achieved by solving material and tracer equations implicitly, using finite difference methods and incorporating thermal decay correction. The discretization of the reservoir and the governing equations, Gauss Seidal method of solution and the computer codes implementation, are also discussed in this section.

4.2 The conceptual model

In order to solve the study problem a conceptual model had to be developed. This was arrived from the formation surface observations and from interference tests carried out on the area which had identified potential feed points. As Mwawongo 2004 puts it, there exist a conduit between well OW-12 and OW-18 and OW-19. The medium which channelled the tracer to OW-19 is approximately 850 m high by 400 m long by approximately 140 m wide. These details were obtained from well drilling data and the task therefore was to find the width of the channel. The 50% porosity got by Mwawongo was very high than expected especially after assuming a 2 m width; the conduit was so thin such that it had to be highly porous to account for the large arrivals of the tracer. Other dimensions were okay, the 850 m was got from height of the permeable layer and 400 m was the inter-well distance. This information can be illustrated as shown in Figure 4-1, This model was derived from the map of Olkaria East production field showing location of wells.

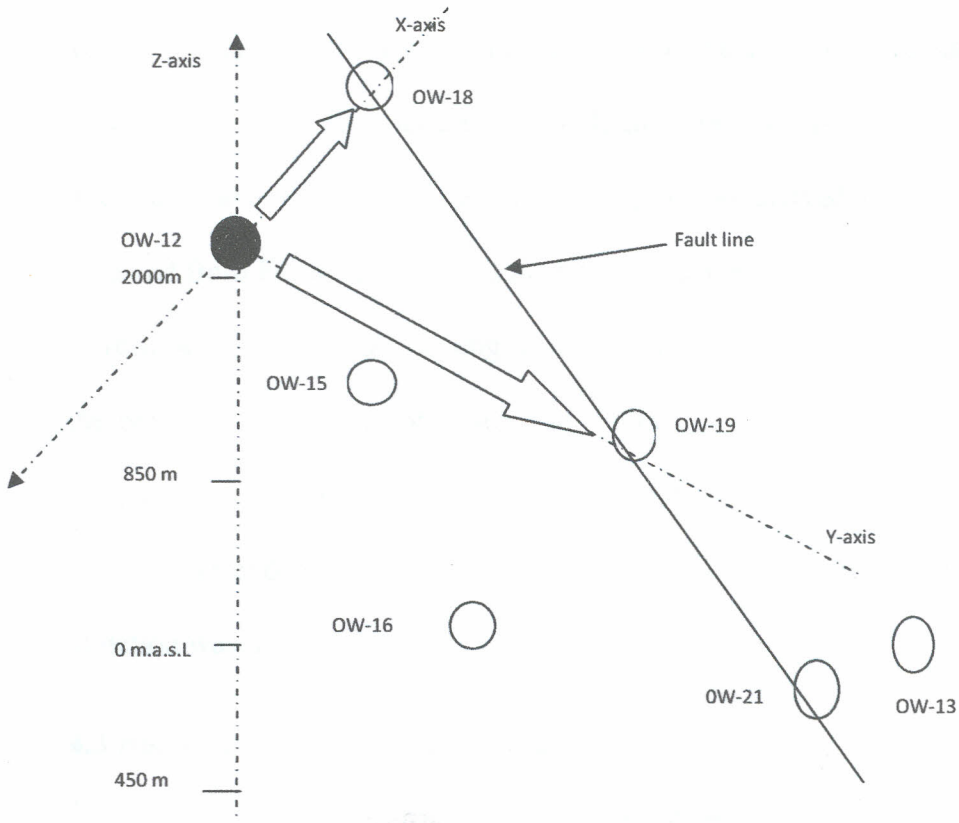


Figure 4-1: Conceptual fluid flow model between well OW-12 and wells

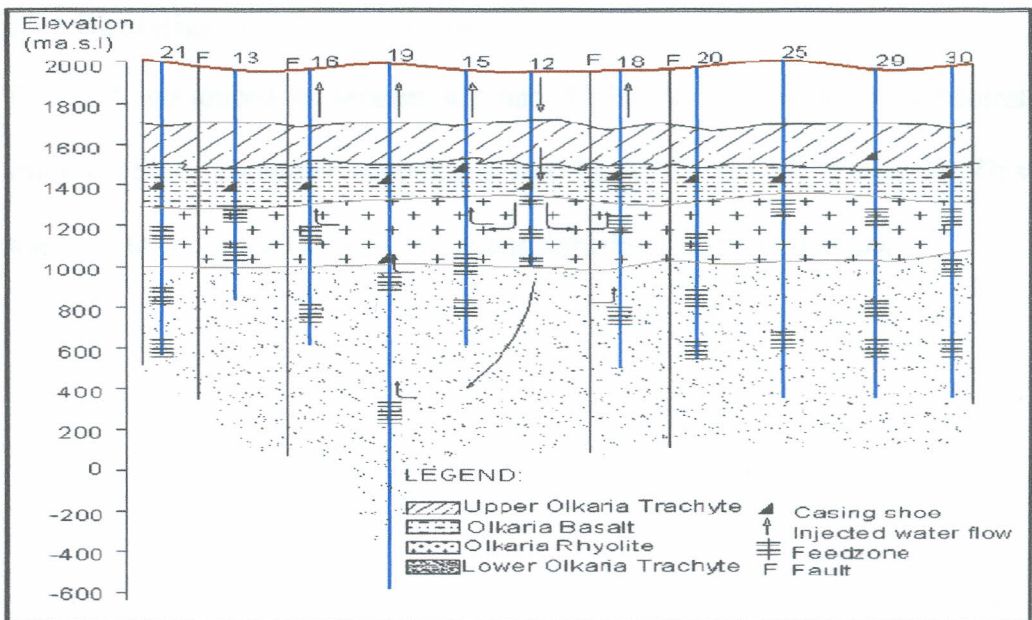


Figure 4-2; Vertical cross-section of the wells under study showing depth and feed points (Mwawongo 2004)

The grid (Figure 4-3) representing the conduit between OW-12 and OW-19 was approximately 30 m long x 2400 m high x 2000 m wide. This was divided into three layers along the vertical. The grid was arrived from information obtained from Figure 4-1 and Figure 4-2. These were information like location of feed points, well depth, casing level and distance between the wells. Hence the permeable layer was at a depth of 400m.a.s.l to 1200m.a.s.l leading to a region 800m high. Therefore if the height (2400m) of the grid was divided into 3 layers we ended up with the middle permeable layer 800m high where the injection was made at block 22.

4.3 Numerical solution implementation

This section entailed translating the mathematical equations into solving the real problem. This was attained by discretizing the reservoir, the governing equations and setting up the computer codes.

4.3.1 Discretization of the reservoir

The grid introduced in **section 4.2** had 45 blocks also referred as control volumes. Since computer counting is zero based the last block was 44. This was seen to represent the real continuous domain as illustrated below

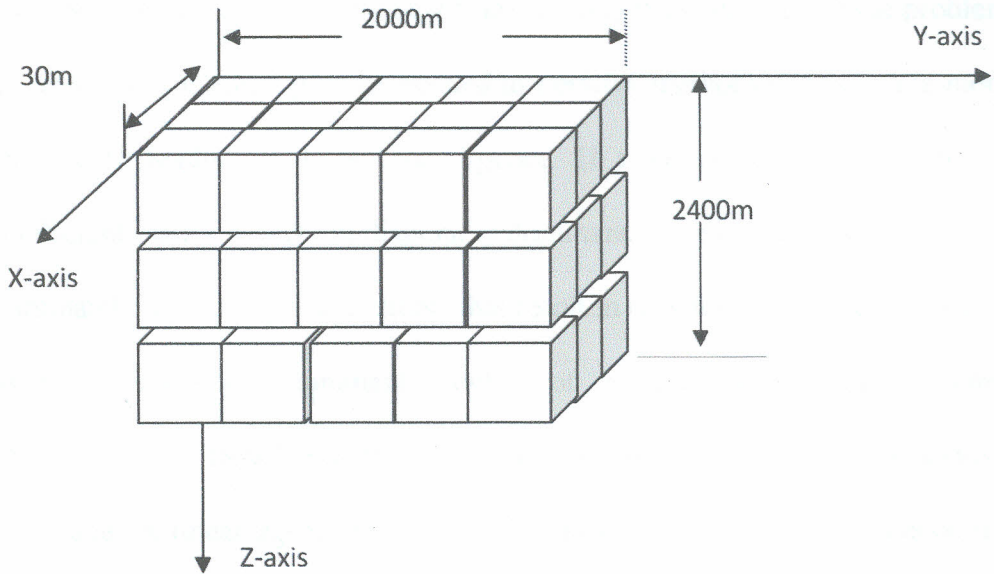


Fig 4-3; Discretization of the reservoir

Each block had four neighbours except those at the boundary. Discretization of the reservoir meant that it would be possible to assign each block different rock properties in the program.

4.3.2 Discretization of the governing equations and the Gauss Seidal method of solution.

Discretization of the governing equations involved converting them from integrals into summations or simply from continuous to discrete entities. This made it possible to code the mathematical equations and thus solve them by a computer program. However, the equations were first expressed into Gauss Seidal form. In certain cases, such as when a system of equations is large, iterative methods of solving equations are more advantageous. Elimination methods, such as Gaussian elimination, are prone to large round-off errors for a large set of equations. Iterative methods, such as the Gauss-Seidel method,

give the user control of the round-off error. Also if the physics of the problem are well known, initial guesses needed in iterative methods can be made more judiciously leading to faster convergence. If a system of equations has a coefficient matrix that is not diagonally dominant, it may or may not converge. Fortunately, many physical systems that result in simultaneous linear equations have a diagonally dominant coefficient matrix, which then assures convergence for iterative methods such as the Gauss-Seidel method of solving simultaneous linear equations. A similar scenario existed here and hence the method was adopted.

4.3.3 The material balance Equation

The material balance equation (3-8) was first discretized by assuming that every block j has four neighbours (Figure 4-4), denoted by k_i , as far as horizontal permeability was concerned n denotes current time step and $n+1$ the next time step, the material balance equation can therefore be written as in equation 4-1

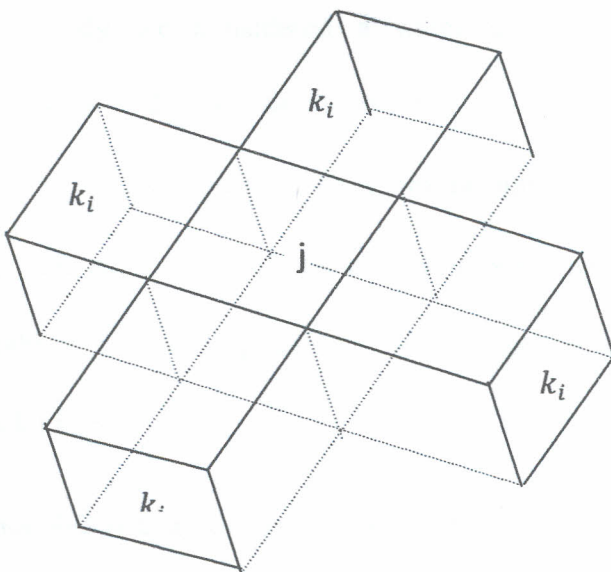


Figure 4-4; Block j and its neighbours

$$\sum_K \frac{\rho_w K_w (P_k^{n+1} - P_j^{n+1}) dA_{kj}}{\mu_w \Delta X_{kj}} + \frac{\nabla g \rho_w^2 h k_w dA_{kj}}{\mu_i} - C \rho_w \phi_j V_j \frac{(P_j^{n+1} - P_j^{n+1})}{\Delta t} = Q_s \quad 4-1$$

Further re-arranging, we get

$$\left(C \rho_w \phi_j \frac{V_j}{\Delta t} + \sum_k \frac{\rho_w K_w dA_{kj}}{\mu_i \Delta X_{kj}} \right) P_j^{n+1} - \sum_k \frac{\rho_w k_w dA_{kj}}{\mu_i \Delta X_{kj}} P_k^{n+1} = C \rho_w \phi_j V_j P_j^n + Q_s \dots 4-2$$

Equation 4-2 can then be expressed in Gauss Seidal method of solution as shown below

$$P_j^{n+1} = \frac{C \rho_w \phi_j V_j P_j^n + Q_s + \sum_k \frac{\rho_w k_w dA_{kj}}{\mu_i \Delta X_{kj}} P_k^{n+1}}{\left(C \rho_w \phi_j \frac{V_j}{\Delta t} + \sum_k \frac{\rho_w K_w dA_{kj}}{\mu_i \Delta X_{kj}} \right)} \dots 4-3$$

Where, the subscript w or s refers to phase under consideration water or steam, in this study we considered a water dominated reservoir only; C - is compressibility, ϕ_j - porosity in block j; Q_s -sources or sinks, V_j -volume of block j, P_j^n -pressure in block j in current time step, P_k^{n+1} pressure in block k in the next time step, A_{kj} -cross section area between block k and j, k_w - permeability of water, ρ_w -density of water and Δx_{kj} -distance between centres of block j and k.

Equation 4-2 is a square penta-valent matrix comprising of 45 equations with pressures in the next time step as the only unknowns, while the terms on the right hand side are all known. It was coded as follows.

Let $\beta = C\rho_w\phi_j \frac{V_j}{\Delta t}$,4-4

$\alpha = \sum_k \frac{\rho_w K_w d A_{kj}}{\mu_i \Delta X_{kj}}$ and4-5

$\gamma = C\rho_w\phi_j V_j P_j^n + Q_s$ 4-6

Then equation 4-2 became

$(\beta + \sum_k \alpha) P_j^{n+1} - (\sum_k \alpha) P_k^{n+1} = \gamma$ 4-7

This was further devolved as follows

$$\left(\beta + \sum_k \alpha \right) P_{11}^{n+1} - (\alpha)P_{12}^{n+1} - (\alpha)P_{13}^{n+1} - (\alpha)P_{14}^{n+1} - +0 \dots \dots + \dots \dots$$

$$= \gamma$$

$$0 + (\beta + \sum_k \alpha)P_{22}^{n+1} - (\alpha)P_{23}^{n+1} - (\alpha)P_{24}^{n+1} - (\alpha)P_{25}^{n+1} - +0 \dots \dots + \dots = \gamma$$

.....

.....

0..... + $(\beta + \sum_k \alpha)P_{n141}^{n+1} - (\alpha)P_{n142}^{n+1} - (\alpha)P_{n143}^{n+1} - (\alpha)P_{n144}^{n+1} = \gamma$

.....4-8

Equation 4-7 can be re-arranged as follows

$$P_j^{n+1} = \frac{\gamma + (\sum_{k \neq j} \alpha) P_k^{n+1}}{\beta + \sum_k \alpha} \dots\dots\dots 4-9$$

This equation is in Gauss Seidal method of solution gives pressure in block j in the next time which is then solved by application of Gauss Seidal method, which immediately updates the newly calculated values of pressure/concentration and uses them in the next calculation.

4.3.4 The Tracer decay equation

Differentiating Equation 2-1 we get

$$\frac{\partial C}{\partial t} = -k C_0 e^{-kt} \dots\dots\dots 4-10$$

But concentration on block j in the current time step was given by

$$C_j^n = C_0 e^{-kt} \dots\dots\dots 4-11$$

Thus on discretization and substituting equation 4-11 in 4-10 and writing in the implicit form, where calculations are done in the next time step, we get

$$\frac{C_j^{n+1} - C_j^n}{\Delta t} = -k C_j^{n+1} \dots\dots\dots 4-12$$

4.3.5 The Tracer balance Equation

Discretization of the chemical species equation is discussed in this section.

Substituting equation 4-12 in equation 3-9 and incorporating tracer decay, we get

$$V\phi \left(\frac{C_j^{n+1} - C_j^n}{\Delta t} \right) \sum_k D \left(\frac{C_k^{n+1} - C_j^{n+1}}{\Delta X_{kj}} \right) dA_{kj} + \sum_k q_{kj} \left(\frac{C_k^{n+1} - C_j^{n+1}}{\Delta X_{kj}} \right) dA_{kj} -$$

$$V \left(\frac{C_j^{n+1} - C_j^n}{\Delta t} \right) = Q_s \quad \dots 4-13$$

Where the last term on the left hand side is as result of tracer decay and advective velocity q_{kj} is given by the Darcy equation which can be written as

$$q_{kj} = -\frac{k}{\mu} \nabla P = -\frac{k}{\mu} \left(\frac{P_k^{n+1} - P_j^{n+1}}{\Delta X_{kj}} \right) \quad \dots 4-14$$

Solving for concentration in the next time step, and incorporating tracer decay, equation 4-13 becomes

$$\left(V\phi - \Delta t \sum_k D \frac{dA_{kj}}{\Delta X_{kj}} + \Delta t \sum_k q_{kj} \frac{dA_{kj}}{\Delta X_{kj}} - V k \Delta t \right) C_j^{n+1}$$

$$+ \Delta t \sum_k D \frac{dA_{kj}}{\Delta X_{kj}} C_k^{n+1} + \Delta t \sum_k q_{kj} \frac{1}{\Delta X_{kj}} dA_{kj} C_k^{n+1} = V\phi C_j^n + Q_s \Delta t$$

$$\dots 4-15$$

This is also a system of 45 equations with values of concentration in the next time step as the unknown terms, while all the rest are known. Just like the mass balance equation, this expression is also solved by application of Gauss Seidal method.

$$C_j^{n+1} = \frac{V\phi C_j^n + Q_s \Delta t - \Delta t \sum_k D C_k^{n+1} \frac{dA_{kj}}{\Delta X_{kj}} + \Delta t \sum_k q_{kj} \frac{C_k^{n+1}}{\Delta X_{kj}} dA_{kj}}{V\phi - \Delta t \sum_k D \frac{dA_{kj}}{\Delta X_{kj}} + \Delta t \sum_k q_{kj} \frac{dA_{kj}}{\Delta X_{kj}} + V k \Delta t} \quad \dots 4-16$$

Where D is dispersion/diffusion coefficient, C is concentration of the tracer; q is advective flux due to fluid movement and k decay rate constant.

4.3.6 The Numerical Simulator structure

In this section, an independent module of programming instructions used to solve the discretized mathematical equations is discussed. The program was written in C++ which is a high-level object oriented programming language. The entire program was formed from pieces called classes and functions. Each class contained objects of same attributes and none or several member functions that manipulated the attributes of the objects.

The problem under the study comprised the input information, the mathematical algorithm and the output. The input constitutes information like the size of the reservoir, rock and water properties such as porosity, density, permeability and diffusivity. The mathematical model involved the procedure to solve the material and tracer balance equations. While the output was as a result obtained after execution of the program. This either was displayed or stored in excel spreadsheets for further analysis. The entire structure of the simulator is illustrated in Figure 4-5.

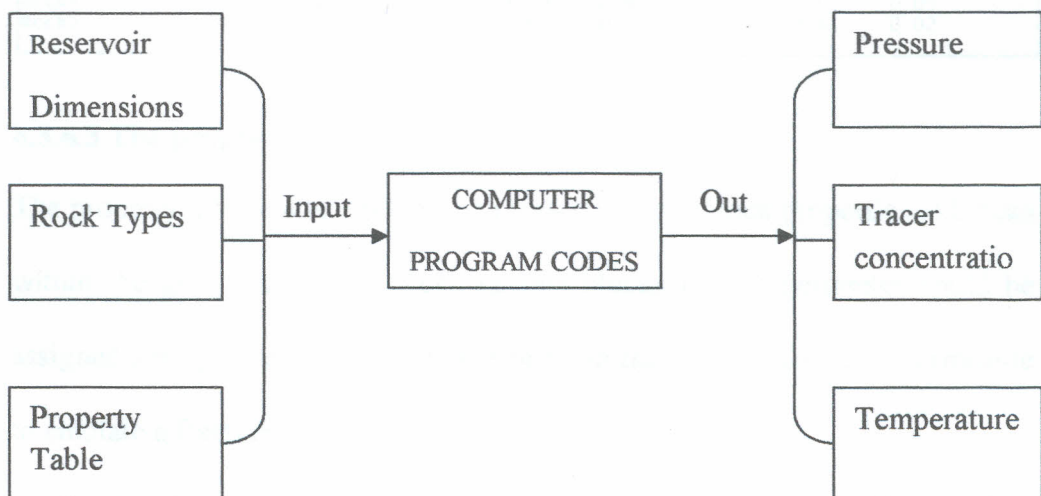


Fig 4-5; Flow of information in the numerical simulator

4.3.6.1 Reservoir Dimensions (*reservoir.txt* file)

This file contained the dimensions of the reservoir along the xyz directions and the number of blocks along each axis (Table 4-1). The simulator used this information to generate a grid from which all other aspects of the problem were founded.

Table 4-1; Reservoir dimensions input file

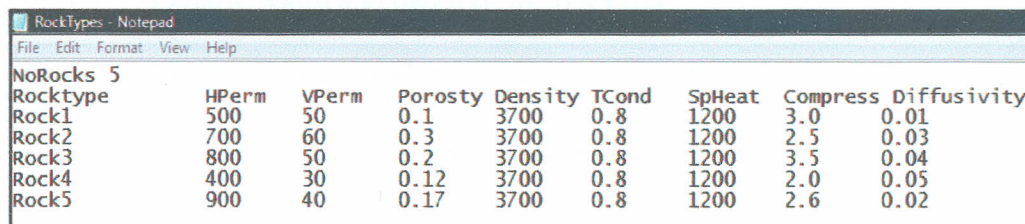


Reservoir - Notepad			
File	Edit	Format	View Help
RLengthXYZ	60	45	3
RBlocksXYZ	5	3	3

4.3.6.2 Rock Types

This was an input file where, rock properties like porosity, density, diffusivity, horizontal and vertical permeability were read from, for use in the mathematical models being solved by the simulator (Table 4-2)

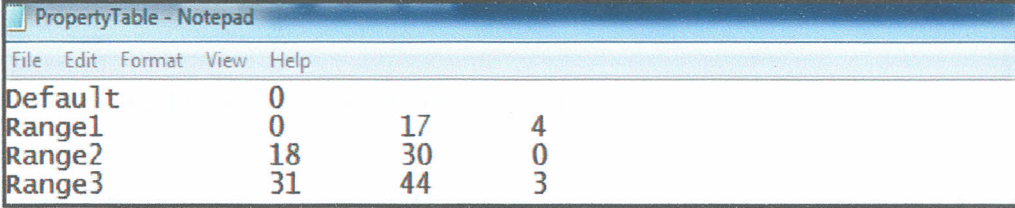
Table 4-2; Rocktypes table input file



RockTypes - Notepad								
File	Edit	Format	View	Help				
NoRocks	5							
Rocktype	HPerm	VPerm	Porosity	Density	TCond	SpHeat	Compress	Diffusivity
Rock1	500	50	0.1	3700	0.8	1200	3.0	0.01
Rock2	700	60	0.3	3700	0.8	1200	2.5	0.03
Rock3	800	50	0.2	3700	0.8	1200	3.5	0.04
Rock4	400	30	0.12	3700	0.8	1200	2.0	0.05
Rock5	900	40	0.17	3700	0.8	1200	2.6	0.02

4.3.6.3 The property table

The property table made it possible to assign different rock properties to blocks within the grid (Table 4-3). For example blocks on the perimeter could be assigned zero permeability, while the ones on the middle remained permeable to emulate a fracture.

Table 4-3; Property table input file


Property	Default	Range1	Range2	Range3
Default	0			
Range1	0	17	4	
Range2	18	30	0	
Range3	31	44	3	

4.3.6.4 The computer program

The computer program (**Appendix 1**) consisted of the computer codes that gave the machine instructions to read the input files, solve the mathematical models and store the results in the output files for further analysis. It was made of five objects namely *CBlock*, *CBlockGrid*, *CBlockGridm*, *CBlockPro*, *CPosition* and *CReservoirDim* and several functions such as *GetResDimension()*, *SetBlockDimensions()* *SetBlockPositions()* etc.

The function *GetResDimension()* was used to read the *reservoir.txt* file (Table 4-1), the following information.

- (i) Length, width and depth of the reservoir
- (ii) Number of blocks along x-axis, y-axis and z-axis.
- (iii) And converted the distances from kilometres into metres and

Then this information was displayed on the prompt window Figure 4-9. Note that this was for a grid of 60 km x 45 km x 3 km divided into 8 x 6 x 3 blocks with 144 blocks. This grid consumed a lot of computer time and hence later in the study a 5 x 4 x 3 grid (Figure 4-3) was used.

The screenshot shows a window titled "G:\FluosimCopy2\Debug\CentralFile.exe" with the following text output:

LengthX	60000
LengthY	45000
LengthZ	3000
BlocksX	8
BlocksY	6
BlocksZ	3

Figure 4-6; The output file of reservoir dimensions function

Then the *SetBlockDimensions()* as the name suggests was used to set the number of blocks in the reservoir (Figure 4-7), their length, width, height, volume and number in layer

The screenshot shows a window titled "G:\FluosimCopy2\Debug\CentralFile.exe" with the following text output:

n	No. In Res	BLayer	LengthX	LengthY	LengthZ	Volume
0	0	0	7500	7500	1000	5.625e+010
1	1	0	7500	7500	1000	5.625e+010
2	2	0	7500	7500	1000	5.625e+010
3	3	0	7500	7500	1000	5.625e+010
4	4	0	7500	7500	1000	5.625e+010
5	5	0	7500	7500	1000	5.625e+010
6	6	0	7500	7500	1000	5.625e+010
7	7	0	7500	7500	1000	5.625e+010
8	8	0	7500	7500	1000	5.625e+010
9	9	0	7500	7500	1000	5.625e+010

The continuation of the text window shows the following text output:

125	125	2	7500	7500	1000	5.625e+010
126	126	2	7500	7500	1000	5.625e+010
127	127	2	7500	7500	1000	5.625e+010
128	128	2	7500	7500	1000	5.625e+010
129	129	2	7500	7500	1000	5.625e+010
130	130	2	7500	7500	1000	5.625e+010
131	131	2	7500	7500	1000	5.625e+010
132	132	2	7500	7500	1000	5.625e+010
133	133	2	7500	7500	1000	5.625e+010
134	134	2	7500	7500	1000	5.625e+010
135	135	2	7500	7500	1000	5.625e+010
136	136	2	7500	7500	1000	5.625e+010
137	137	2	7500	7500	1000	5.625e+010
138	138	2	7500	7500	1000	5.625e+010
139	139	2	7500	7500	1000	5.625e+010
140	140	2	7500	7500	1000	5.625e+010
141	141	2	7500	7500	1000	5.625e+010
142	142	2	7500	7500	1000	5.625e+010
143	143	2	7500	7500	1000	5.625e+010

Fig 4-7; The output of the function SetBlockDimensions

Secondly this function set up an array of blocks in the reservoir and allocated memory space to dynamic data structures that increased or decreased during program execution, this array is called *ReservoirBlocks* while the changing data include calculations for pressure, viscosity or tracer concentrations. This was achieved by use of pointers and a *new* expression

Then it was necessary to set positions of the blocks in the reservoir, this was done by the function *SetBlockPositions()* which calculated the xyz coordinates of the centre of each block as illustrated in (Figure 4-8 and 4-9). Along Z-axis we had 3000 m divided by 6 (3 blocks each with 2 parts) =500 m. X-axis- $60000/8 \times 2 = 3750$ m. Y- axis $45000/6 \times 2 = 3750$ m.

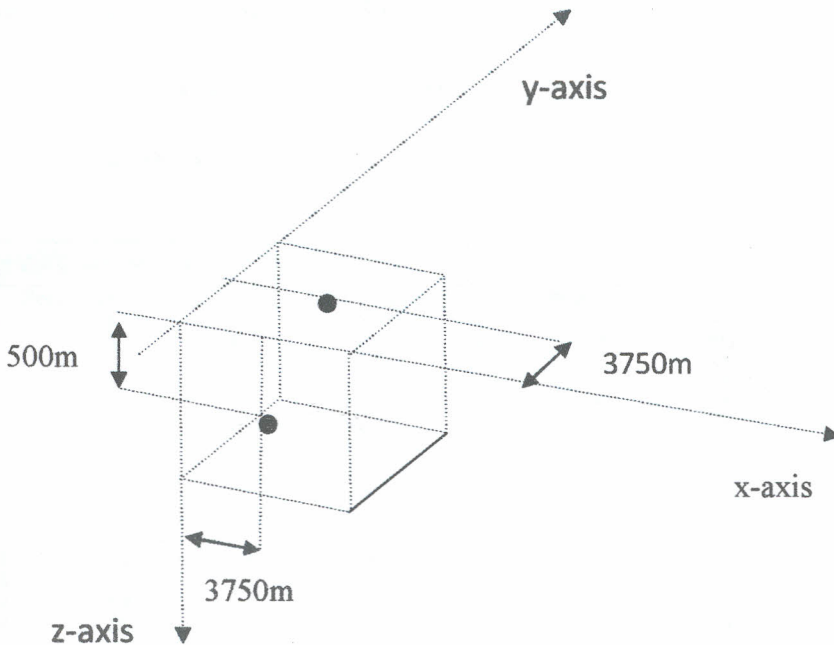


Figure 4-8; Illustration of how block positions were set

n	PstnX	PstnY	PstnZ
0	3750	3750	500
1	11250	3750	500
2	18750	3750	500
3	26250	3750	500
4	33750	3750	500
5	41250	3750	500
6	48750	3750	500
7	56250	3750	500
8	3750	11250	500
9	11250	11250	500
10	18750	11250	500
11	26250	11250	500
12	33750	11250	500
13	41250	11250	500
14	48750	11250	500

Figure 4-9; The output of SetBlockPositions function showing positions of the each block

The preceding functions were the basis of setting up the grid, done by use of the *SetGrid* function. For this goal to be achieved, all the neighbours of each block were identified. The blocks inside the grid had neighbours all around but those on the boundary did not, therefore blocks on the perimeter were assigned negative numbers on those sides that neighbours did not exist. For instance block 0 had block 1 in front, 8 on the left, -2 on the rear, and 3 on the right, 5 on the upper side and 48 on the lower side. Neighbours on the other blocks are generated as shown in Figure 4-10

n	FrontBlk	RearBlk	RightBlk	LeftBlk	UpBlk	LowBlk
0	1	-2	-3	8	-5	48
1	2	0	-3	9	-5	49
2	3	1	-3	10	-5	50
3	4	2	-3	11	-5	51
4	5	3	-3	12	-5	52
5	6	4	-3	13	-5	53
6	7	5	-3	14	-5	54
7	-1	6	-3	15	-5	55
8	9	-2	0	16	-5	56
9	10	8	1	17	-5	57
10	11	9	2	18	-5	58
11	12	10	3	19	-5	59
12	13	11	4	20	-5	60
13	14	12	5	21	-5	61
14	15	13	6	22	-5	62
15	-1	14	7	23	-5	63
16	17	-2	8	24	-5	64
17	18	16	9	25	-5	65
18	19	17	10	26	-5	66
19	20	18	11	27	-5	67
20	21	19	12	28	-5	68

Fig 4-10; Setting up the grid and identifying the neighbours of each block

SetRockProperties() function had three roles, first it opened and read *RockTypes.txt* and *PropertyTable.txt* input files, created a dynamic array called *RockTypes* which contained various rock properties like porosity, permeability, diffusivity and density.

The next function (single computer operation), was *SetThermondynConditions()* computed values of pressure at various depths using Jonathan Leaver (1984), correlations. It also called the functions

ComputeTemp() and *ComputeViscosity()* which generated the values of temperature, water density and viscosity with change in depth for all blocks in the reservoir, which were used later in the program. These functions were first calculated elsewhere before they were called here. Temperature change with depth was first calculated using the Leaver correlations. On the other hand Correlations for calculation of viscosity were first derived using excel spreadsheets. This was done by first plotting a graph of viscosity against temperature (Figure 4-11) using the values from the steam tables and then adding a trend line to develop mathematical functions.

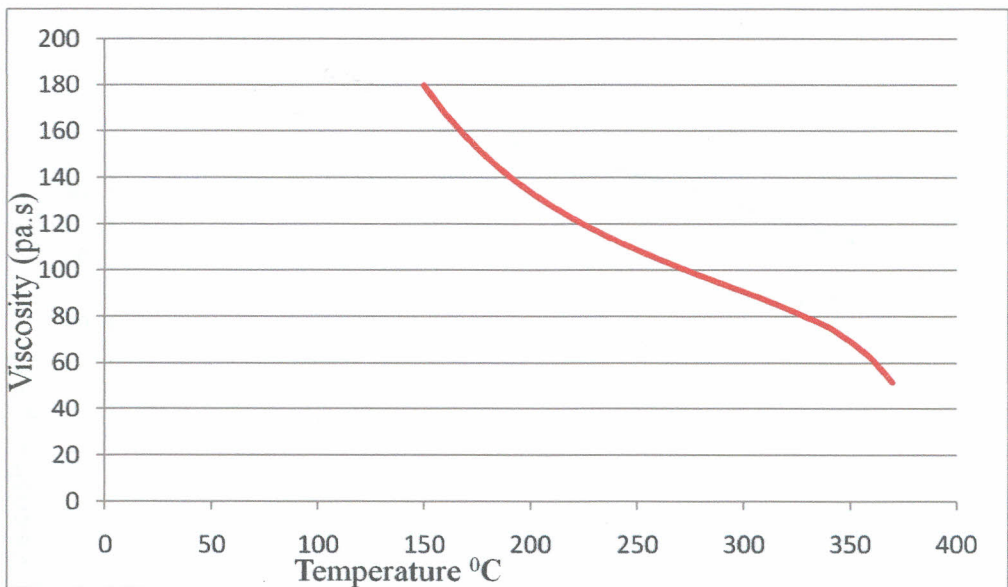


Fig 4-11; Viscosity against temperature graph

This graph was first broken into two sections so that a better estimation of a trend line could be added as shown in Figure 4-12 and 4-13 using Ms office excel.

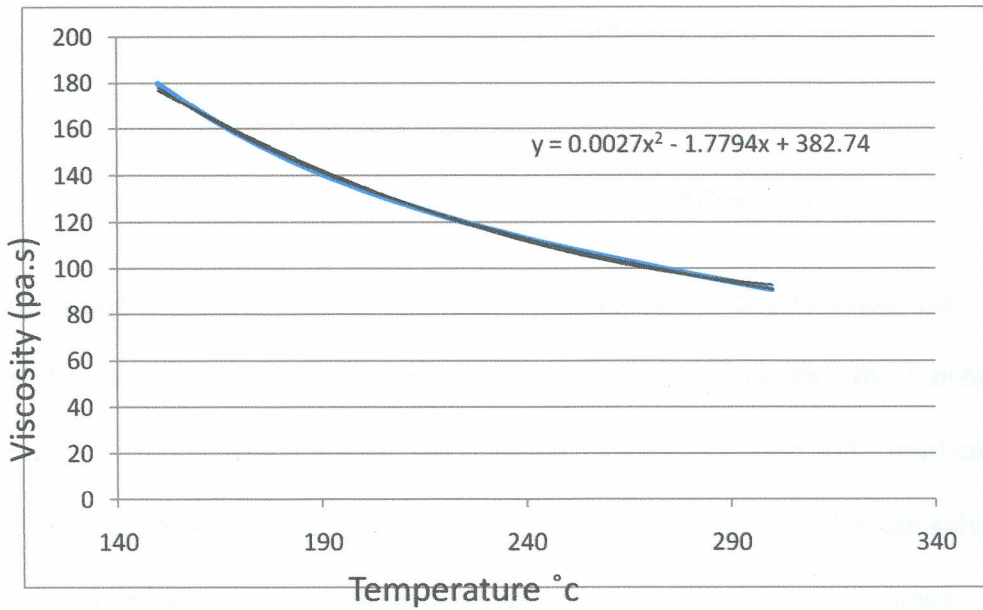


Fig 4-12; Viscosity from temperature 150°C to 300°C

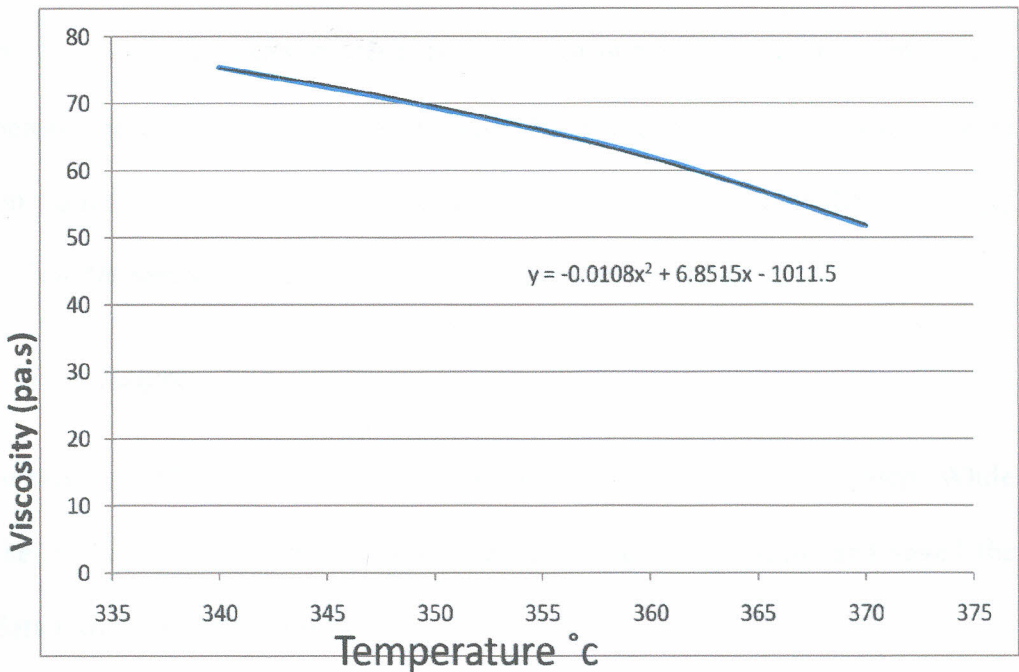


Fig 4-13; Viscosity from temperature 340°C to 370°C

Here y and x represent viscosity and temperature respectively, thus it was possible to code in the program as follow.

if (Temperature <= 300) Viscosity = 0.002 * pow(Temperature, 2) -
1.779 * Temperature + 382.7

else Viscosity = -0.010 * pow(Temperature, 2) + 6.851 * Temperature - 1011.

Tracer concentration, sources and sinks were also set in this function. Having set the basics, the main solvers were then set up, starting with the function *Solve Pressure()* which solved the material balance equation 4-9 implicitly using Gauss Seidal method whereby 45 equations for each block were solved and updated after every iteration. The symbols were first coded as follows

γ – gamma β – beta α – alpha and $\sum_k \alpha$ – alphasum

A “for loop” was used to solve the codes for every four neighbour of block j before the entire equation was solved at the end of the solver and updated immediately. This equation was $InterMed2[j] = (\beta * ReservoirBlocks[j].Pressure + alphasum + ReservoirBlocks[j].SFluid) / \alpha$

where $InterMed2[j]$ was the pressure of block j in the next time step. While the following code updated the operation during the iterations and saved the data into the file *Pressure.txt*

$InterMed[j] = InterMed1[j];$

$InterMed1[j] = InterMed2[j];$

Similarly, the next main solver was the *SolveTracer()* which computed (equation 4-16) implicitly and stored the results in an output file called

Tracer.txt

```
InterMed2[j]=(alpha+beta*TimeStep+ReservoirBlocks[j].STracer*TimeStep  
+gammasum*TimeStep)/(volume*gamma*TimeStep+gammasum2*TimeStep+  
ReservoirBlocks[j].
```

```
RateConstant*TimeStep*ReservoirBlocks[j].LengthY*ReservoirBlocks[j].Len  
gthZ*ReservoirBlocks[j].LengthX);)
```

The next two lines implements the Gauss Seidal method of solution which immediately updates the newly calculated values of concentration and uses them in the next calculation unlike the other methods which through all calculations before updating. Gauss Seidal method therefore takes much fewer iterations and thus saves much of computer time and memory

```
InterMed[j]=InterMed1[j];
```

```
InterMed1[j]=InterMed2[j];//Updates InterMed1[j].
```

Fluorescein thermal decay was incorporated in this equation and coded *RateConstant* which was given by the Arrhenius equation coded in c++ as follows

```
RateConstant=exp(18.25)*exp(-143300/(8.31*Temperature));
```

CHAPTER FIVE

5 RESULTS AND DISCUSSIONS

5.1 Introduction

This simulator was developed to model tracer flow in a highly fractured porous medium of a geothermal system. This chapter is hereby presented by first looking at the generic tests and then the match of the simulated results with the field data. The generic tests were intended to determine the workability of the simulator while in the later; the objective was to deduce parameters of the fracture. Therefore, before matching the field data, generic tests were carried out to ascertain that it operated as expected. Field parameters like porosity, permeability, size of the medium and injection rates were varied and sensitivity of the simulator was observed.

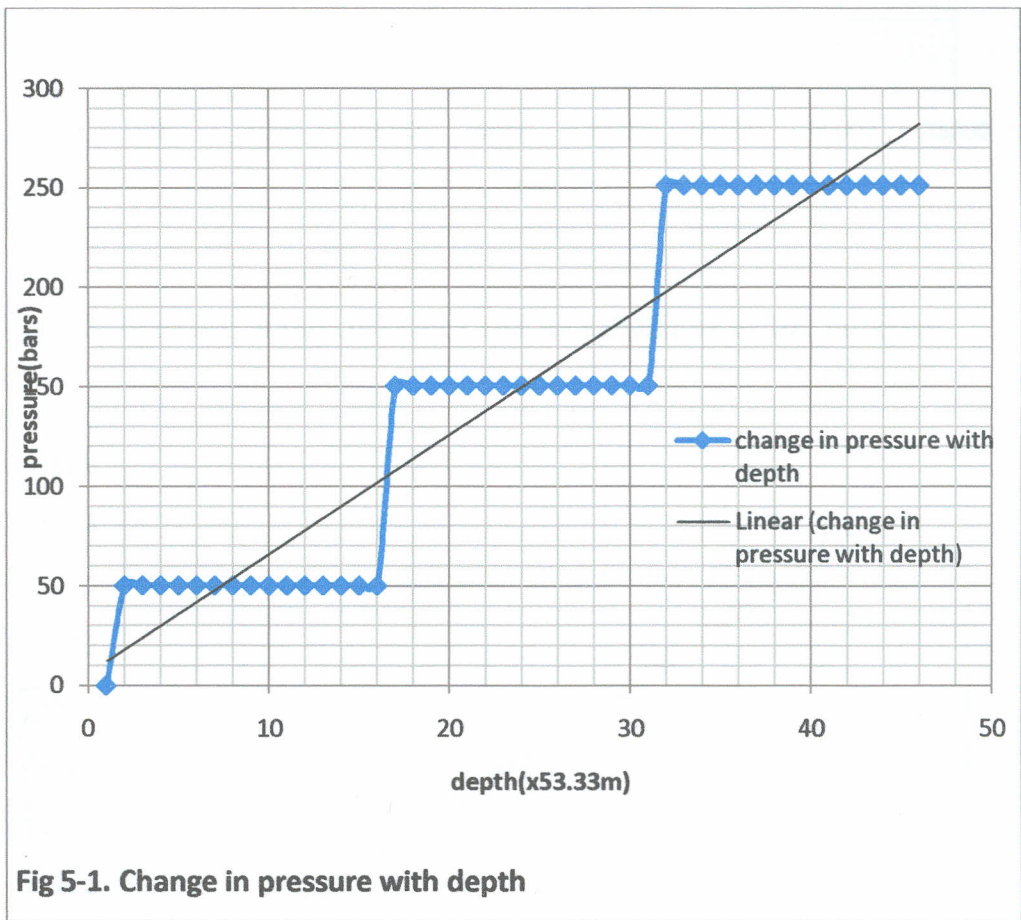
The process involved solving two modules, the pressure and the tracer. The pressure being important as it formed the basis of the movement of the tracer. The second was the tracer module from which the breakthrough curve was obtained which had to be matched with field data later.

5.2 Generic Tests

The reservoir used here had 144 blocks (before the one with 45 blocks was used as it saved computational time) injector was block number sixty five, its neighbours being block numbers sixty four to the rear, sixty six in front, seventy three the left, fifty seven on the left, seventeen on top and one hundred and thirteen to the bottom. On injection observations were made on changes in pressure and tracer concentration on block sixty five and its neighbours. These are discussed in the next subheadings of this section.

5.2.1 Pressure test

Observations from the generic pressure tests indicated that there was an increase in pressure with depth before even injection was done which was as expected. The horizontal steps represented blocks on same layer hence on same depth (Figure 5-1).



Without injecting, pressure remained steady in the injector block and also in its neighbours at a depth of 1600 metres as illustrated in Figure 5-2. But during injection pressure build up linearly with time (Figure 5-3).

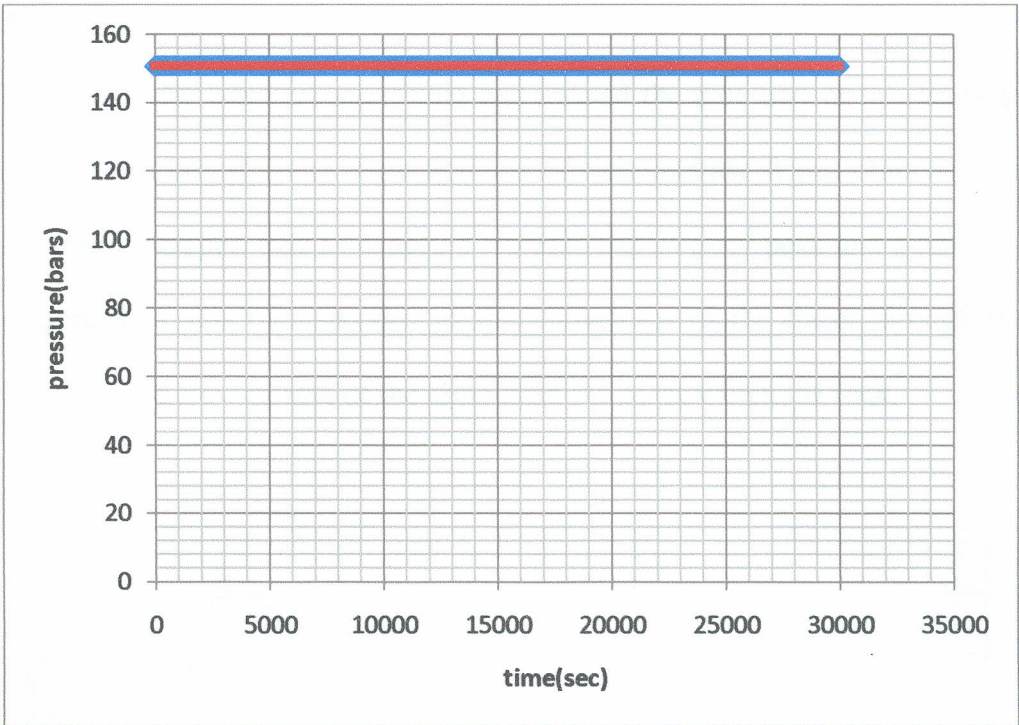


Fig 5-2; Pressure in injector block and its neighbour before injection

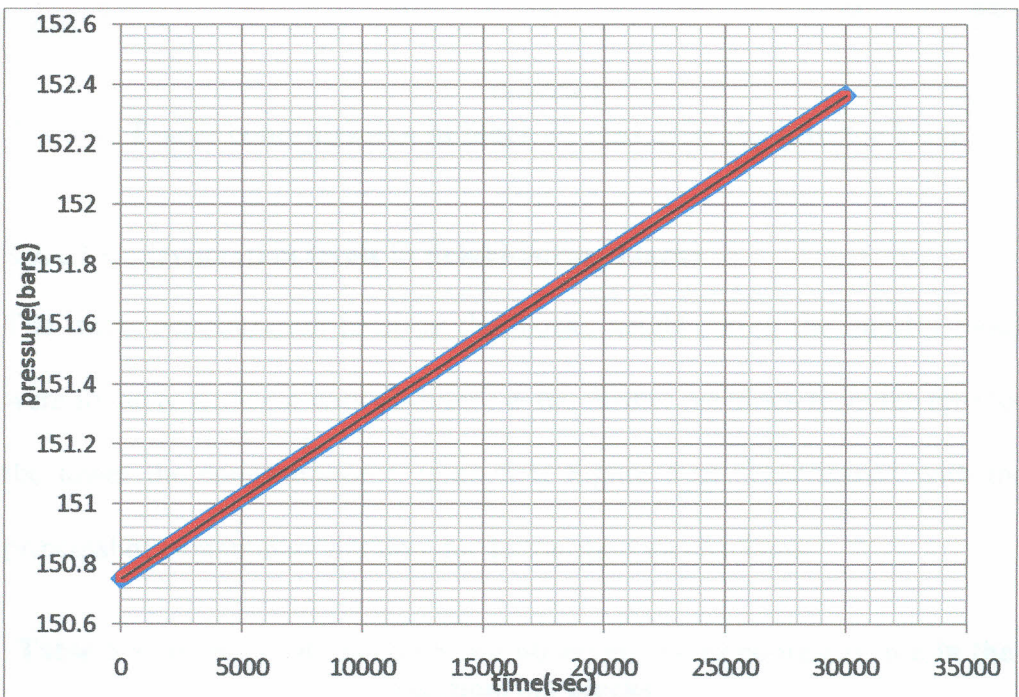


Fig 5-3; Pressure in injector block during injection

5.2.2 Tracer test

The tracer test was very significant and formed the core part of this study as fluorescein thermal decay correction was of main interest. It was observed that the tracer transport was as expected representing an early peak and a later tail (Figure 5-4), this corresponded to first movement of the tracer through the fractures and slow release of the same from the pores within the rock.

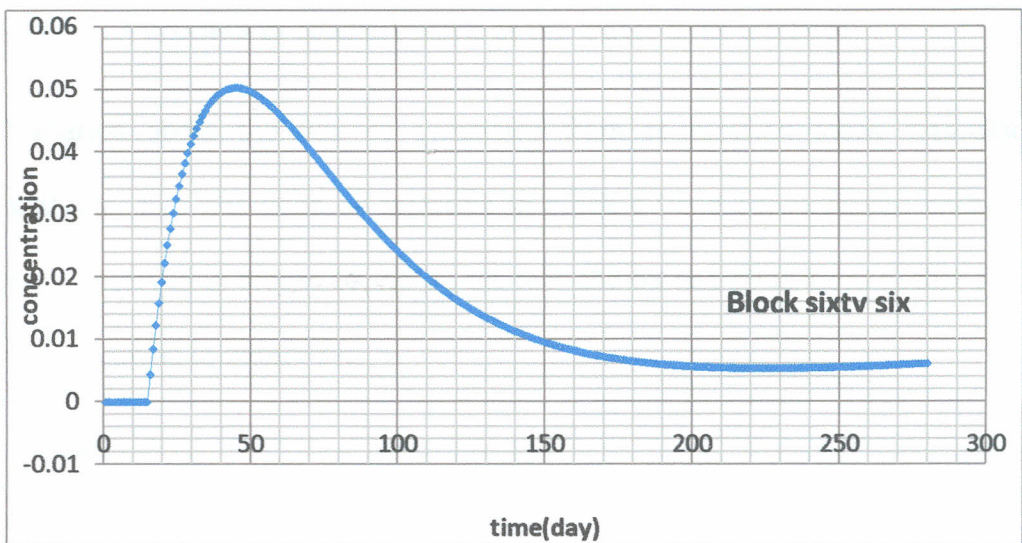


Fig 5-4; Tracer concentration in block 66 next after the injector block

5.2.2.1 Concentration levels in neighbouring blocks

The tracer concentration level on the blocks neighbouring the injector block were found to decrease with distance i.e. the further the block from the injector the lower the concentration (Figure 5-5; Figure 5-6). This agreed with the principal of contaminant dilution due to transport mechanisms.

Table 5-1; Injection in block 65 and observed tracer concentration in the neighbouring blocks

Blk63	Blk64	InjectorBlk65	Blk66	Blk67	Blk68	Blk69	Blk70
-------	-------	---------------	-------	-------	-------	-------	-------

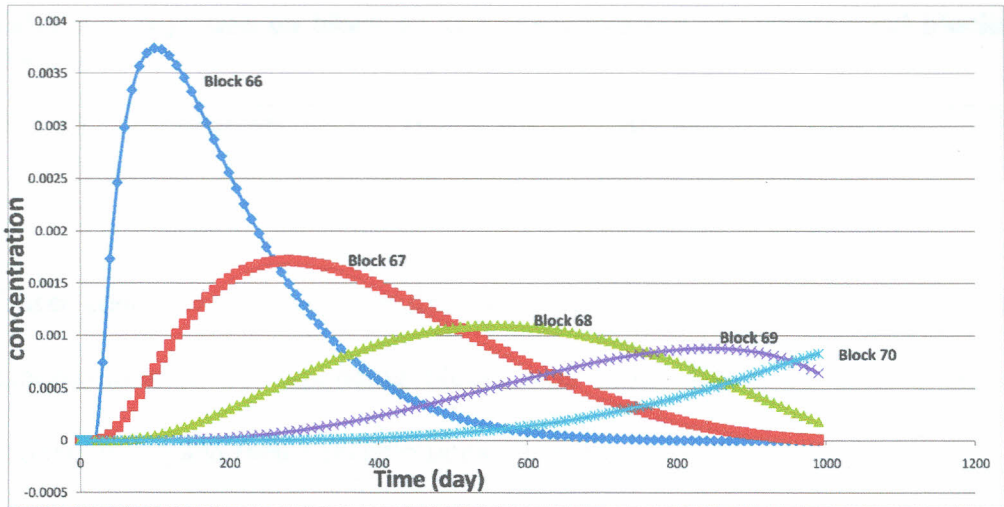


Fig 5-5; Variation in concentration of the tracer with distance

Variation about the line of symmetry from re-injection block was also analysed, this was done by injecting in block 67 and concentration changes were observed in blocks 65, 66 on the rear and 68 and 69 in front (Table 5-1) and displayed in Figure 5-6

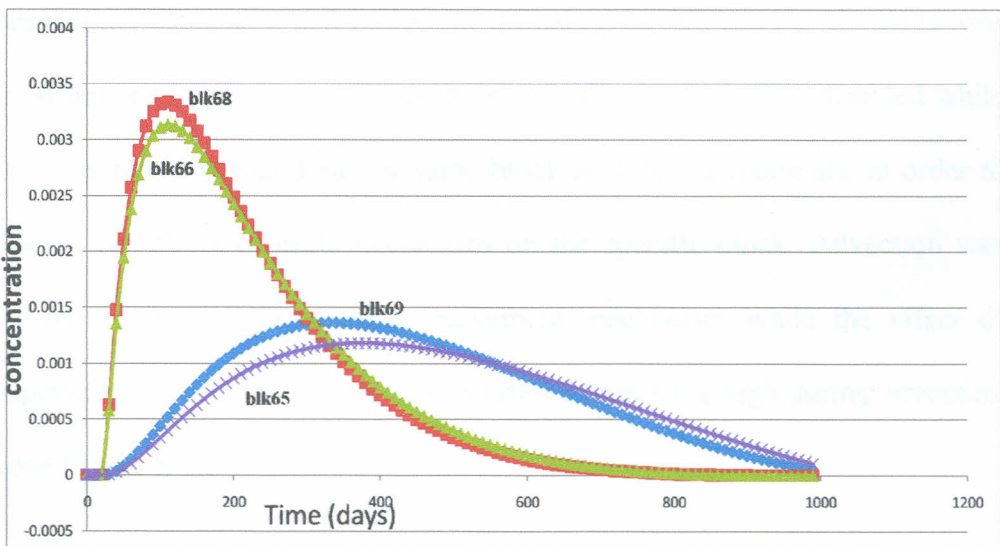


Figure 5-6; Tracer concentration in symmetrical blocks, blk66 is symmetrical to blk68, blk65 with blk69. Injector is block 67

Table 5-2: injection on block 67 and concentration on symmetrical blocks observed

Blk 65	Blk 66	Injetor Blk 67	Blk 68	Blk 69
--------	--------	----------------	--------	--------

Tracer concentration along the line of symmetry was also as expected since concentration in corresponding blocks was same. Block 66 and 68 were corresponding and their concentrations were same.

5.2.2.2 Effect of advection and dispersion transport mechanisms

The tracer transport is by two main mechanisms, namely advection and dispersion. In advection the dissolved substances are carried along with bulk fluid flow and depend on the velocity of the fluid. Dispersion constitute of both longitudinal dispersion and molecular diffusion. In this test the simulator was run when one of the transport mechanisms in the simulator was disabled while the other ran, this was done for same block at different moments, in order to observe the effect of each mechanism on the specific block. Advection was realized to be the major tracer movement mechanism while the effect of dispersion was minimal, since concentration levels were high during advection alone Figure 5-7.

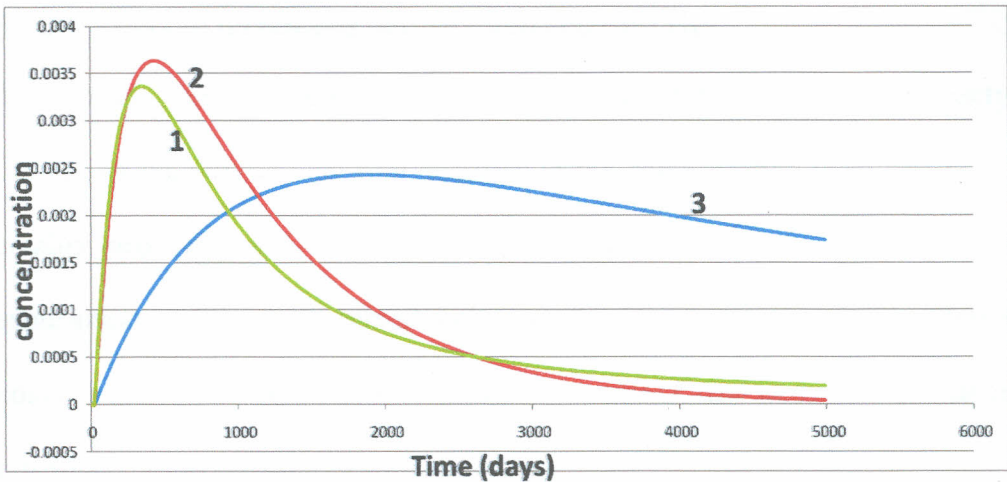


Fig 5-7; 1. Effect of transport by both advection and dispersion. 2. Transport by advection only and 3. Transport by dispersion only

5.2.2.3 Effect of changing field parameters

Different values of field parameters were tried out on the simulator and their effect was compared with that in literature. The table 5-1 shows range of some of these parameters from literature.

Table 5-1; Source of parameters Mwawongo, 2004 and Ambusso, 2007

Quantity	Magnitude (range in literature)	Units
Porosity	7 to 30	%
Permeability	200×10^{-15} to 59.16×10^{-13}	m^2
Dispersion	0.67	
Fluid injection rate	6	Kg/s
Tracer injection rate	8	Kg/s
<i>Reservoir temperature</i>	250	$^{\circ}C$
<i>Bottom hole pressure</i>	170	bars
<i>Water density</i>	0.601583	

5.2.2.4 Effect of increasing porosity and fracture size

The tracer concentration was found to vary inversely as the porosity. Increasing the porosity led to a decrease in the concentration level and hence smaller peak (Figure 5-8). This meant that, due to an increase in storage then same amount of the tracer mixed with more fluid hence translating into a lower concentration. The effect of increasing the reservoir size was same as that of porosity.

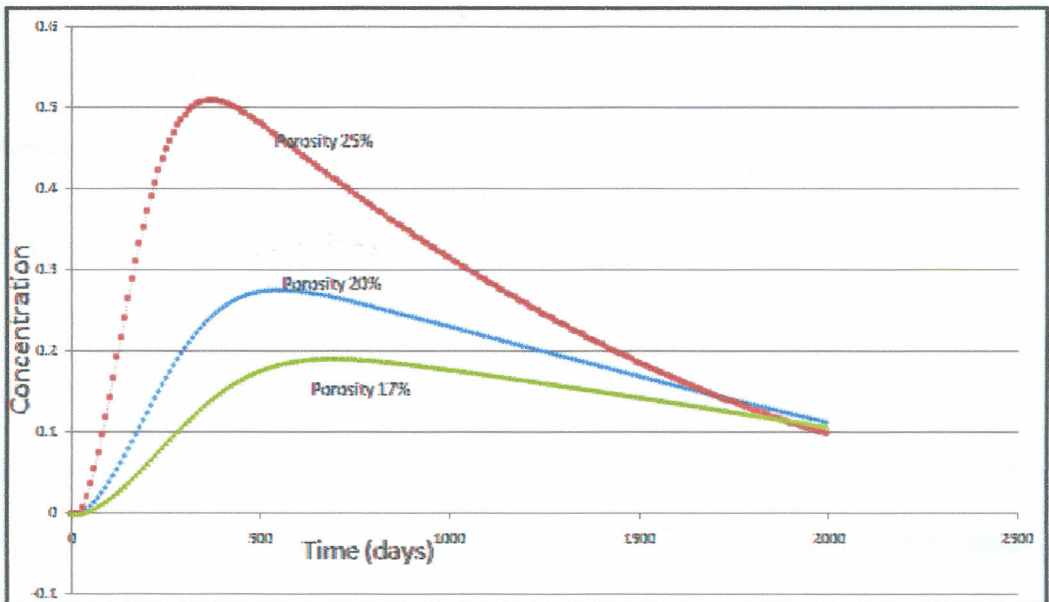


Fig 5-8; Effect of different levels of porosity on concentration of the tracer

5.2.2.4 Effect of increasing horizontal permeability

Increase of permeability resulted to an increase in the height of the tailing portion of the breakthrough curve. This meant that permeability has direct effect on fracturing. That is the higher the permeability the higher the fracturing and thus a high and longer tail (Figure 5-9).

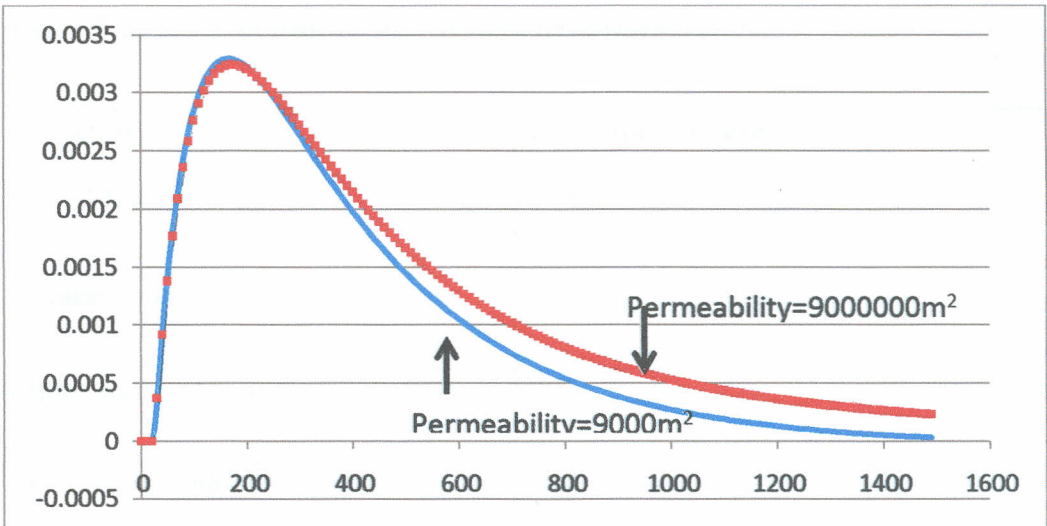


Fig 5-9; Effect of increasing permeability raised the tailing portion of the breakthrough curve.

5.3 The process of matching the simulated results with field data

The simulation was done over a maximum time of three hundred days covering the total period of the test without incorporating the decay correction term. The parameters shown in Table 5-2 were varied until the simulated breakthrough curve matched the field data. The simulation was run and matched with data from tracer returns in well OW-12 and OW-19 dipole.

Table 5-2; Parameters that were varied to match the field data

Parameter	Magnitude in literature	Units
porosity	7-30	%
permeability	950	m^2
Fracture width	3-10	m

Table 5-3; Olkaria geothermal system field parameters

Quantity	Magnitude in literature	Units
Fluid Injection rate	0.0038	m ³ /s
Tracer injection rate	0.027	m ³ /s
Inter well distance	400	m
Fracture Depth	800	m
Water density (ρ_w)	0.654	g/m ³
Bottom hole temperature	325	°C
Bottom hole pressure	120	bars
Water Compressibility (c_w)	1.2×10^{-9}	Pa ⁻¹
Viscosity (ν)	Calculated by simulator and changed with depth	Pa.s
Molecular diffusion (k)	2.0×10^{-9}	m ² /s
Dispersion ($\phi\tau$)	0.67	
Diffusivity (k)	0.670000002	
Injection Tracer concentration (c)	25	Kg/m ³

5.3.1 The field data

The field data was obtained from KENGEN (Kenya Electricity Generating Company) in Olkaria, Naivasha. The Figure 5-10 below shows the matching of breakthrough curves for the field data and the simulated one. The field data for tracer returns between well OW-12 and OW-18 is contained in **Appendix 2**

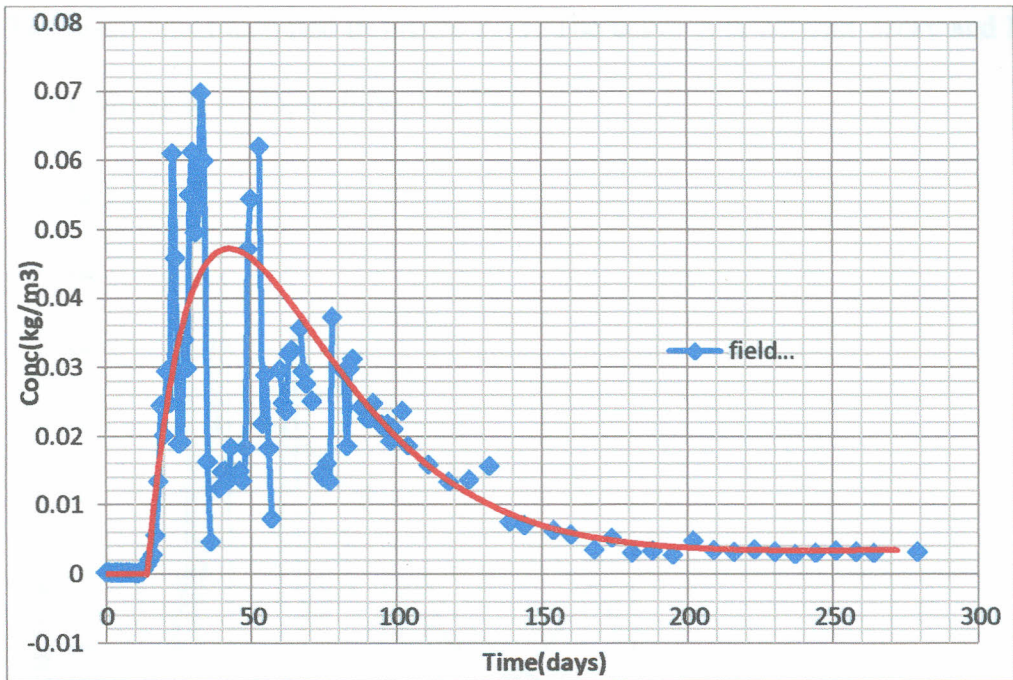


Fig 5-10; Match of field data and the simulated breakthrough curve

The peak indicates quick transmission of the fluid through the fractures. The action of continued release of dissolved tracer after the removal of the bulk of the free product causes what is called “tailing”. This results in small, but detectable and possibly significant contamination in the groundwater that can progress for long periods of time. This is due to fluid that enters and leaves the rock matrix. Due to capillary suction small amounts of the tracer may become trapped in pores of the rock matrix. Removal of these chemical is difficult to impossible because they are held in place by capillary suction.

5.4 Tracer Decay correction

All along, the simulator was ran when fluorescein thermal decay correction term was disabled but on its incorporation in the tracer equation, the breakthrough curve had a shift passing where a conservative tracer would pass.

The result was illustrated in Figure 5-11. The curve A is without decay and B is with decay at a temperature of 325°C.

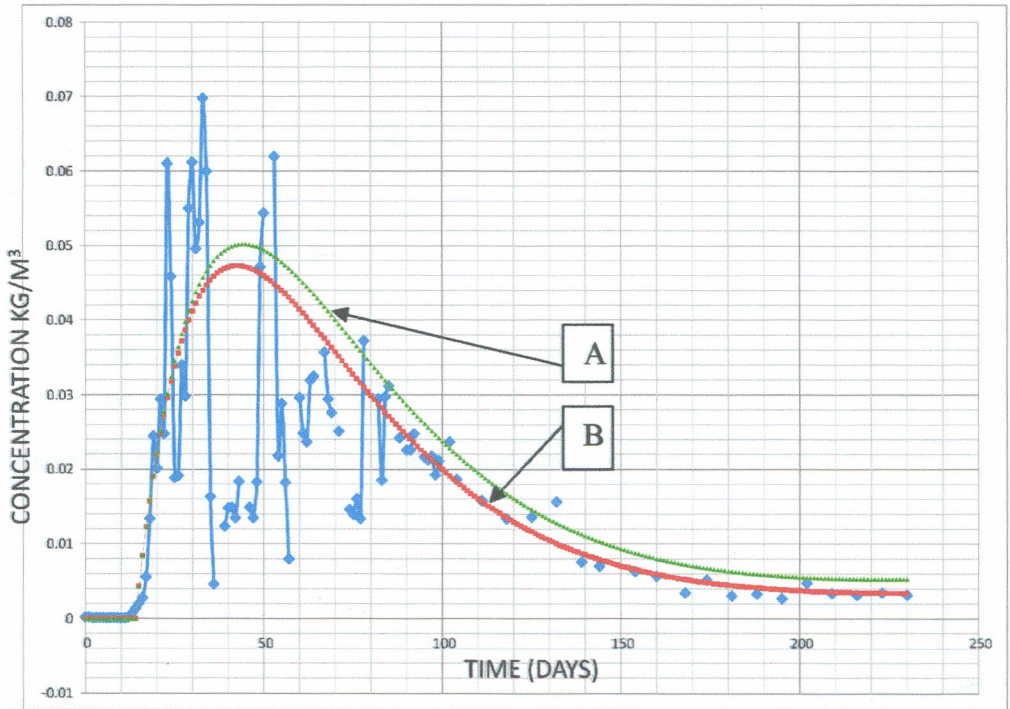


Fig 5-11; Fluorescein thermal decay correction

5.5 Model validation

The field data was plotted against the simulated data as illustrated in Figure 5-12. A linear relationship Equation 5-1 was obtained as expected.

$$\text{Concentration} = 0.9 \text{ SimulatedResults}$$

5-1

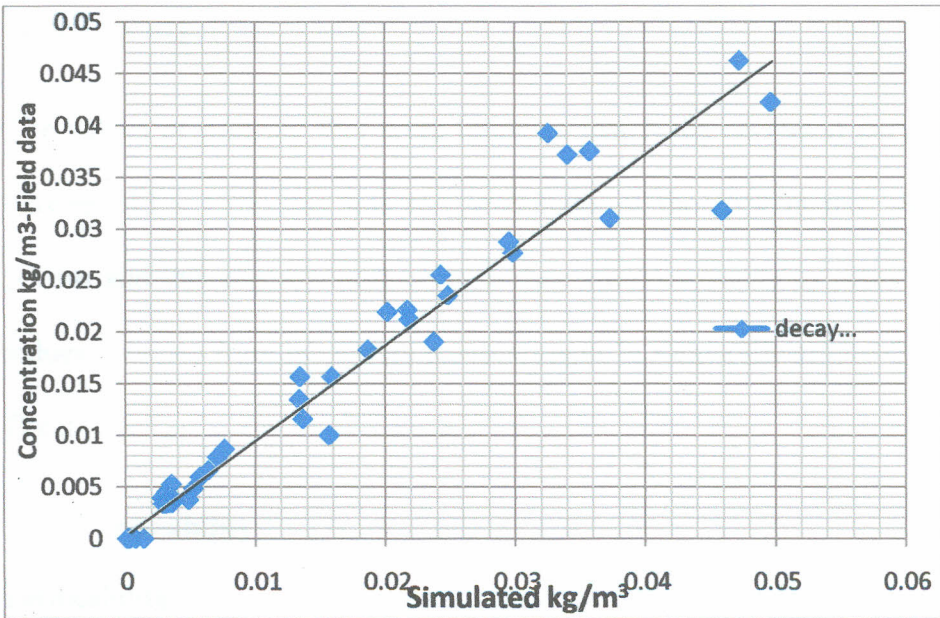


Figure 5-12; A plot of field data versus simulated results

5.6 Determination of the fracture Parameters

To investigate the parameters of the fracture from the breakthrough curve, simulations were ran and matched with the corrected curve. Values of porosity, permeability and fracture width were varied until the simulated results, matched the corrected curve in Figure 5-13

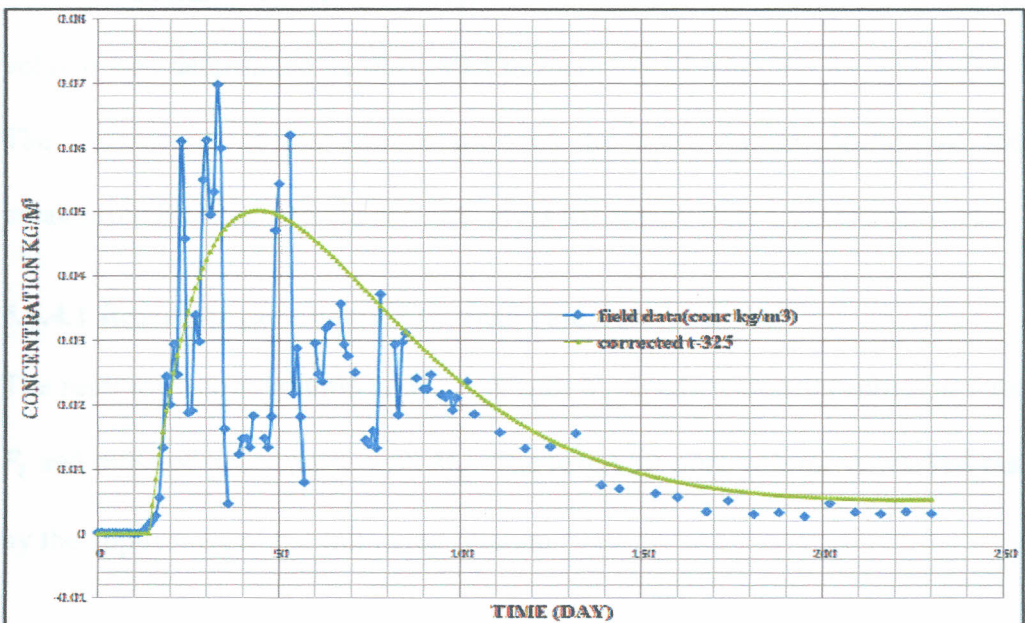


Figure 5-13; Matching simulated results on the field data

5.6.1 Porosity

The values of porosity that were found to match the corrected curve ranged between 13 - 15%, this was the percentage of the rock matrix which was porous and this was practical because they are comparable to petro graphical values obtained from laboratory measurements in Kenya Electricity Generating Company

5.6.2 Permeability

Permeability of the fracture was found to range between 1.8mm^2 while permeability thickness was 950 m^2

5.6.3 Thickness of the fracture

The thickness of the fracture was found to vary between 3 metres and 10 metres and this seemed to agree with Mwawongo, 2004 who obtained 2 m.

5.6.4 Extent of the fractured media

The linear tailing portion of the breakthrough curve can be used to calculate reservoir overall volume according to Rose et al (1997). Overall reservoir volume was determined as illustrated in Figure 5-14 from the formula below. The intersection of the extrapolation of the linear tailing portion of the breakthrough curve with the ordinate gives the steady state Concentration F_t .

5.6.4.1 Reservoir pore volume (overall reservoir volume)

The reservoirs pore volume is denoted with V , steady state concentration with F_t and mass of tracer pulse with M_p . The reservoir pore volume is worked out by the application of the following relation:

$$V = \frac{M_p}{F_t}$$

5-2

Given that M_p for this study was 20 tonnes /hour and F_t obtained from extrapolation of the curve Figure 5-14 = 0.0085kg/m^3 then

$$V = \frac{20 \text{ tonnes/hr}}{0.0085\text{kg/m}^3} = \frac{5.555\text{kg/s}}{0.0085\text{kg/m}^3}$$

$$= 653.52\text{m}^3/\text{s} = 653520 \text{ litres/sec}$$

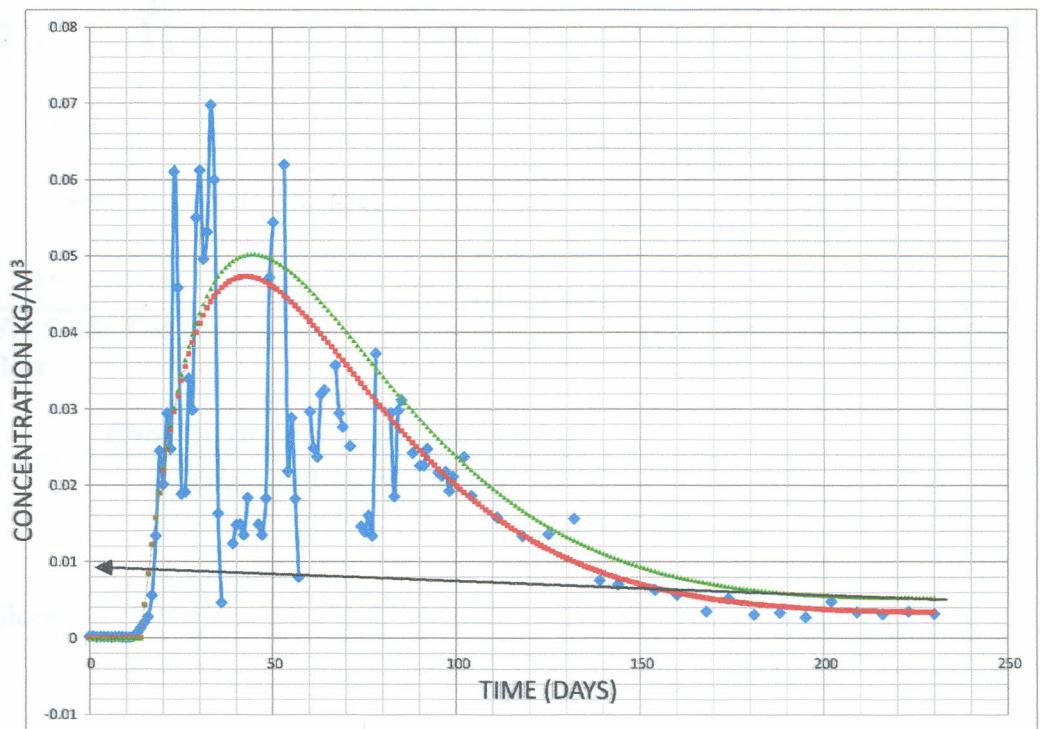


Figure 5-14; Y-intercept of the extrapolated linear tailing portion give steady state concentration

5.6.4.2 Rate of aquifer flow

Slope of the linear tailing portion of a breakthrough curve (Figure 5-15) represent rate of aquifer flow through the reservoir (Rose et al, 1997).

Microsoft excel was used to read out the small values to calculate slope as shown in the equation 5-3.

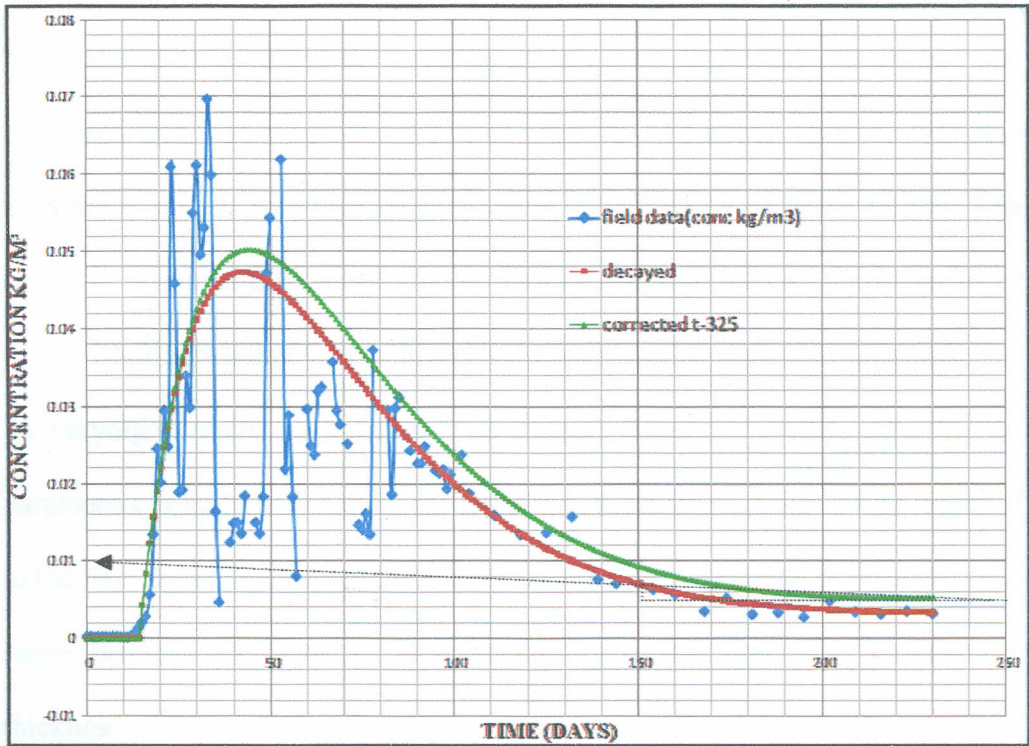


Fig 5-15; The slope of the linear tailing portion represent rate of aquifer flow

$$\text{Slope} = \frac{\text{Change in concentration } \text{kg/m}^3}{\text{change in time}(\text{secs})} \quad 5-3$$

$$\text{Slope} = \frac{(0.00761 - 0.003537) \text{kg/m}^3}{(12009600 - 24105600) \text{sec}}$$

$$= \frac{0.0041 \text{kg/m}^3}{2096000 \text{sec}}$$

$$= 1.956 \times 10^{-9} \text{kg/m}^3/\text{sec}$$

CHAPTER SIX

6 CONCLUSIONS AND RECOMMENDATIONS

6.1 CONCLUSION

The material balance and the tracer equations were first setup, discretized and expressed into Gauss Seidal method of solution. The decay equation was also discretized and incorporated into the tracer equation as a negative source. The reservoir was also discretized into 45 blocks resulting into 45 equations to be solved implicitly using C++, a high level programming computer language.

By varying porosity, permeability and fracture dimensions while other known parameters were kept constant, the simulated breakthrough curve was matched to the field data. This necessitated estimation of the magnitude of porosity as to range between 13 % to 15 %, permeability of 1.8 mm^2 and permeability thickness of 950 m^2 . The fracture thickness was estimated to be 3 m and not more than 10 meters. The model was also validated by plotting the simulated result against the field data and obtained a linear relation as expected. This was done by first filtering noise from the field data.

The steady state concentration obtained from y-intercept after extrapolating the linear tailing portion of the breakthrough curve was used to calculate entire reservoir pore volume, which was obtained to be 653520 litres/sec. While the slope of the same linear tailing portion was used to calculate rate of aquifer flow which was found to be $1.956 \times 10^{-9} \text{ kg/m}^3/\text{sec}$.

6.2 RECOMMENDATIONS

The study was basically involved with correction of fluorescein thermal decay, whereby a numerical simulator was developed which can be used to estimate properties of an aquifer. Hence maximise power production through sustained re-injection. However the following is recommended

1. It should be appreciated that no single test can reliably be conclusive alone; therefore other tests such as interference tests should be conducted to determine the boundaries, transmissivity and storativity of the porous medium.
2. Incorporate Geology of the area in the simulator. The feed zones, permeability and porosity of the rocks in the area (e.g. Olkaria Basalt, Rhyolite etc) should be determined in the laboratory using geological methods like petrography and various blocks in simulator should be assigned those rock properties. The extent of the fracture/aquifer should be estimated from analysis done during well drilling since no single test can reliably be conclusive.
3. Investigate the relationship that increase in permeability leads to a rising tail and vice versa, and if this can be of any significance.
4. Determine the temperature of the flow path from the extent of the tracer decay.

REFERENCES

- Adams, M.C., and Davis, J. (1991). **Kinetics of fluorescein decay and its application as a geothermal tracer**, University of Utah Research Institute 391A Chipeta Way, Salt lake USA
- Adams, M.C., Benoit, W.R., Bordvarsson, G.S. and Moore, J.N. (1989). **The Dixie Valley, Nevada tracer test**. Geothermal Resources Council Trans. 13, 215-220.
- Ambusso and Ouma (1991). **Assessment of Reservoir and Steam Status in Olkaria East Field**, A KPC internal report.
- Ambusso, W.J. (1994). **Results of injection and tracer tests in Olkaria geothermal field**. Proceedings, 19th Workshop on Geothermal Reservoir Engineering, Stanford University, Stanford, California.
- Ambusso, J.W. (2007). **Numerical simulation of fluid flow in a dual porosity geothermal system with a thin zone of high horizontal permeability**, PhD Thesis. Kenyatta University, Nairobi, KENYA.
- Chrysikopoulos, C. and Kruger, P. (1992). **Artificial Tracers for Geothermal Reservoir Studies**. Environmental Geology 22:60-70.
- Freeze, R.A. and J.A. Cherry. 1979. **Groundwater**. Prentice-Hall, Inc. Englewood Cliffs, NJ. 604 p.
- Grant, A.M, Donaldson, I.G, Bixley. (1982). **Geothermal Reservoir Engineering**. Academic Press.
- Gregory, 2006. **Cracks in rock at Sunrise-on-Sea beach**, Eastern Cape, South Africa
- Gudmundsson J.S. and Hauksson, T. (1985). **Tracer Survey in Svartsengi 1984**. Geothermal Resources Council. Vol. 9-PART II, Pg. 307-315. August 1985.
- Kariuki N.M. 2004. **Strategy for increasing hot re-injection in the Olkaria East field**. Proceedings of Olkaria geothermal conference. Pg 83-88.
- Kariuki, M., and Ouma, P., 2002: **Twenty years of exploitation of Olkaria East Field**, *Geoth. Res. Council, Trans.*, 26, 565-571.
- Karsten P (2002). **Numerical simulation of 'multiphase tracer transport in fractured geothermal reservoirs**. Geothermics 31 (2002) 475-499
- Levevspiel, O. (1972), **Chemical Reaction Engineering**, 2nd Edition, John Wiley and sons, New York, NY, Ch9.

Mwawongo, G.M. (2004). **Infield re-injection strategies in Olkaria, Kenya based on tracer studies and numerical modelling.** The United Nations University.

Ofwona, C.O.(1996). **Analysis on injection and tracer tests data from the Olkaria East geothermal field, Kenya.** UNU Geothermal training programme, Iceland.

Ofwona, C.O (2003). **A preliminary update of natural state numerical model of Olkaria geothermal system, Kenya.** Olkaria geothermal project, P.O. BOX 785, Naivasha, Kenya.

Ofwona O.Cornel (2004). **Predictions of exploitation of Olkaria I reservoir based on a lumped model.** Proceedings of Olkaria geothermal conference, 26-32.

Ouma, P.(2002). **Re-injection strategy in the Olkaria North East geothermal field.** Kenya Electricity Generating Company LTD, Kenya.

Rose, P.E., 1999. **Tracer Testing at Steamboat Springs, Nevada, Using fluorescein and 1,5- naphthalene disulfonate.** In: Proceedings of the 24th workshop on Geothermal Reservoir Engineering, S

Shook, G.M. (2003). **“A simple, Fast Method of Estimating Fractured Reservoir Geometry from Tracer tests”**, Trans., Geothermal Resources Council, 27, 407-411.

Smart, P.L., and Laidlaw, E.M.S. (1977). **An Evaluation of some fluorescent dyes for water tracing.** Research, V, 13.P.15-33.

Tsai R, Huang (2008). **Heat and mass transfer for Soret and Dufour's effects on Hiemenz flow through porous medium onto a stretching surface.** International Journal of Heat and Mass Transfer 52 (2009) 2399–2406

Tester,J.W., Robinson,B.A and Ferguson,J.H. (1986). **Inert and reacting tracers for reservoir sizing in fractured, hot rock systems;** Eleventh workshop on Geothermal Engineering, Stanford University, 149-159.

Pruess, K. and G.S.Bodvarsson (1984). **Thermal Effects of Re-injection in Geothermal Reservoirs with Major Vertical fractures.** J.Pet.Tech. 36 , 1567-1578

ReservoirDim.cpp

APPENDIX 1; THE COMPUTER PROGRAM

```

// ReservoirDim.cpp: implementation of the CReservoirDim class.//
//////////////////////////////////////////////////////////////////
/
#include      <iomanip.h>
#include      "ReservoirDim.h"
#include      <fstream.h>
#include      "BlockGrid.h"
#include      <math.h>
//////////////////////////////////////////////////////////////////
/
// Construction/Destruction
//////////////////////////////////////////////////////////////////
/
CReservoirDim::CReservoirDim()
{
    TimeStep=1998.5;
    TimeMax=280*86400;
}

CReservoirDim::~CReservoirDim()
{
}

void CReservoirDim::GetResDimension()
{
    ifstream infile; infile.open ("Reservoir.txt");

    char dummy[15];

    infile>>dummy;
    infile>>LengthX;LengthX*=1000;//length of grid along x-axis
    infile>>LengthY;LengthY*=1000;
    infile>>LengthZ;LengthZ*=1000;

    infile>>dummy;
    infile>>BlocksX;//number of blocks in x-axis
    infile>>BlocksY;
    infile>>BlocksZ;
}

void CReservoirDim::Display()
/*
    cout<<"LengthX"<<setw(15)<<LengthX<<endl;//display length of
reservoir along x-axis
    cout<<"LengthY"<<setw(15)<<LengthY<<endl;
    cout<<"LengthZ"<<setw(15)<<LengthZ<<endl;
    cout<<"BlocksX"<<setw(15)<<BlocksX<<endl;//displays number of
blocks in x-axis
    cout<<"BlocksY"<<setw(15)<<BlocksY<<endl;
    cout<<"BlocksZ"<<setw(15)<<BlocksZ<<endl;

    //Blocks dimensions
    cout<<"n"<<setw(10)<<"Block Num"<<setw(10)<<"Block
Num"<<setw(10)<<"BlockX " <<setw(10)<<"BlockY " <<setw(10)<<"BlockZ
"<<setw(10)<<"volume"<<endl;

    for (int n=0;n<BlockNumber;n++)
    {

    cout<<n<<setw(10)<<ReservoirBlocks[n].NumberInRes<<setw(10)<<Reservoir
rBlocks[n].BLayer<<setw(10)<<ReservoirBlocks[n].LengthX<<setw(10)<<Re
servoirBlocks[n].LengthY<<setw(10)<<ReservoirBlocks[n].LengthZ
<<setw(10)<<ReservoirBlocks[n].Volume<<endl;
    }

    cout<<"n"<<setw(10)<<"Front"<<setw(10)<<"Rear"<<setw(10)<<"Right
"<<setw(10)<<"Left " <<setw(10)<<"Up"<<setw(10)<<"Low"<<endl;

```



```

ReservoirDim.cpp
for ( n=0;n<BlockNumber;n++)
{
    cout<<n<<setw(10)<<ReservoirGrid[n].FrontBlock
<<setw(10)<<ReservoirGrid[n].RearBlock
<<setw(10)<<ReservoirGrid[n].RightBlock
<<setw(10)<<ReservoirGrid[n].LeftBlock
<<setw(10)<<ReservoirGrid[n].UpBlock
<<setw(10)<<ReservoirGrid[n].LowBlock <<endl;
}
cout<<endl<<Rocks<<endl;
for (int n=0; n<Rocks; n++)
{

cout<<n<<setw(10)<<RockTypes[n].HPermeability<<setw(10)<<RockTypes[n]
.VPermeability<<setw(10)<<RockTypes[n].Porosity<<setw(10)<<RockTypes[
n].Density<<setw(10)<<RockTypes[n].KThermCond<<setw(10)<<RockTypes[n]
.HeatCapacity<<setw(10)<<RockTypes[n].Compressibility
<<setw(10)<<RockTypes[n].Diffusivity <<endl;
}
cout<<endl<<Rocks<<endl;
for ( n=0;n<BlockNumber;n++)
{

cout<<n<<setw(10)<<ReservoirBlocks[n].properties.HPermeability<<setw(
10)<<ReservoirBlocks[n].properties.VPermeability<<setw(15)<<Reservoir
Blocks[n].properties.Porosity<<setw(15)<<ReservoirBlocks[n].WaterDens
ity<< endl;
}
cout<<endl<<Rocks<<endl;
for (int n=0;n<BlockNumber;n++)

//cout<<n<<setw(10)<<ReservoirBlocks[n].BPosition.PositionX<<setw(10)
<<ReservoirBlocks[n].BPosition.PositionY<<setw(10)<<ReservoirBlocks[n]
.BPosition.PositionZ<<setw(10)<<endl;
cout<<n<<setw(15)<<ReservoirBlocks[n].Pressure<<setw(15)<<ReservoirBl
ocks[n].Temperature<<setw(15)<<ReservoirBlocks[n].Viscosity <<endl;

cout<<endl<<"Pressure"<<endl;
for (int n=0; n<BlockNumber; n++)
cout<<n<<setw(15)<<ReservoirBlocks[n].Pressure<<setw(15)<<InterMed[n]
<<setw(15)<<InterMed1[n]<<setw(15)<<InterMed2[n]<<endl;
*/
for (int n=0;n<BlockNumber;n++)
cout<<n<<setw(15)<<ReservoirBlocks[n].Pressure<<setw(15)<<ReservoirBl
ocks[n].Temperature<<setw(15)<<ReservoirBlocks[n].Viscosity<<setw(15)
<<ReservoirBlocks[n].RateConstant<<setw(15)<<endl;
}
void CReservoirDim::SetBlockDimensions()
{
    BlockNumber=BlocksX*BlocksY*BlocksZ;//total number of blocks
in the reservoir

    ReservoirBlocks=new CBlock[BlockNumber];//allocates memory to
an array of type CBlock
    ReservoirGrid=new CBlockGrid[BlockNumber];//create memory for
ReservoirGrid
    InterMed1=new double[BlockNumber];//dynamic array for
pressure
    InterMed2=new double[BlockNumber];//allocation of memory
space for a dynamic variable InterMed which is of type double
    InterMed=new double[BlockNumber];
    //Set block dimensions

    for (int n=0;n<BlockNumber;n++)
    {

```

```

ReservoirDim.cpp
ReservoirBlocks[n].NumberInRes =n;//number of a block
in the reservoir
ReservoirBlocks[n].LengthX =LengthX/BlocksX;//length
of the block along an axis is given by length of the
ReservoirBlocks[n].LengthY
=LengthY/BlocksY;//reservoir divided by number of blocks along that
axis
ReservoirBlocks[n].LengthZ =LengthZ/BlocksZ;
ReservoirBlocks[n].NumberInLayer
=(n%(BlocksX*BlocksY));
ReservoirBlocks[n].BLayer =n/(BlocksX*BlocksY);
ReservoirBlocks[n].Volume
=ReservoirBlocks[n].LengthZ*ReservoirBlocks[n].LengthX
*ReservoirBlocks[n].LengthY ;
}
}
void CReservoirDim::SetBlockPositions()
{
double depth=LengthZ/(2*BlocksZ);//depth of reservoir divided
by 2*number of blocks along z axis(200/2*8)=12.5
double lengthx=LengthX/(2*BlocksX);//length of reservoir
divided by 2*number of blocks along x-axis (600/2*6)=50
double lengthy=LengthY/(2*BlocksY);//(300/2*3)=50

for (int n=0;n<BlockNumber;n++)
{
ReservoirBlocks[n].BPosition.PositionZ
=depth+ReservoirBlocks[n].BLayer *ReservoirBlocks[n].LengthZ;
ReservoirBlocks[n].BPosition.PositionX
=lengthx+(ReservoirBlocks[n].NumberInLayer%BlocksX)
*ReservoirBlocks[n].LengthX;
ReservoirBlocks[n].BPosition.PositionY
=lengthy+(ReservoirBlocks[n].NumberInLayer/BlocksX)
*ReservoirBlocks[n].LengthY;
}
}
void CReservoirDim::SetGrid()
{
for (int n=0;n<BlockNumber;n++)
{
//set front block
if(ReservoirBlocks[n].NumberInRes%BlocksX==(BlocksX-1))
{ReservoirGrid[n].FrontBlock=-1;
}
else{ReservoirGrid[n].FrontBlock=ReservoirBlocks[n].NumberInRes+1;}

//set rear block
if(ReservoirBlocks[n].NumberInRes%BlocksX==0)
{ReservoirGrid[n].RearBlock =-2;
}else{ReservoirGrid[n].RearBlock
=ReservoirBlocks[n].NumberInRes-1;}

//set right block
if(ReservoirBlocks[n].NumberInLayer<BlocksX)
{ReservoirGrid[n].RightBlock =-3;
}else{ReservoirGrid[n].RightBlock
=ReservoirBlocks[n].NumberInRes-BlocksX;}

//set left block
if(ReservoirBlocks[n].NumberInLayer>=(BlocksY*BlocksX-BlocksX))
{ReservoirGrid[n].LeftBlock =-4;
}else{ReservoirGrid[n].LeftBlock
=ReservoirBlocks[n].NumberInRes+BlocksX;}
}
}

```

```

        ReservoirDim.cpp
        //set top block
        if(ReservoirBlocks[n].NumberInRes <BlocksY*BlocksX)
        {ReservoirGrid[n].UpBlock   =-5;
        }else{ReservoirGrid[n].UpBlock
=ReservoirBlocks[n].NumberInRes-BlocksY*BlocksX;}
        //set low block
        if(ReservoirBlocks[n].NumberInRes
>=(BlockNumber-BlocksY*BlocksX))
        {ReservoirGrid[n].LowBlock   =-6;
        }else{ReservoirGrid[n].LowBlock
=ReservoirBlocks[n].NumberInRes+BlocksY*BlocksX;}
    }
}
void CReservoirDim::SetRockProperties()
{
    ifstream infile("RockTypesHigh.txt"); char buffer[15];
    infile>>buffer>>Rocks;

    RockTypes=new CBlockProp[Rocks]; //creation of an array
    containing rock properties

    for(int n=0; n<9; n++)infile>>buffer;

    for ( n=0; n<Rocks; n++)
    {
        infile>>buffer>>RockTypes[n].HPermeability>>RockTypes[n].VPermeabilit
y>>RockTypes[n].Porosity>>RockTypes[n].Density>>RockTypes[n].KThermCo
nd>>RockTypes[n].HeatCapacity>>RockTypes[n].Compressibility
>>RockTypes[n].Diffusivity;
        RockTypes[n].Compressibility
*=pow(10,-8); //successively multiply compressibility with pow(10,-8)
        RockTypes[n].HPermeability*=pow(10,-15);
        RockTypes[n].VPermeability*=pow(10,-15);
        RockTypes[n].Diffusivity*=pow(10,-5);
    }
    infile.close ();

    ifstream infile1("PropertyTable.txt");int defaultv,
lowerv,upperv;

    do {infile1>>buffer>>defaultv;
    } while (infile1>>buffer>>defaultv);

    for (int n=0; n<BlockNumber; n++)
    {
        ReservoirBlocks[n].properties=RockTypes[defaultv];
    }

    while(infile1>>buffer>>lowerv>>upperv>>defaultv)
    {
        for ( n=lowerv;n<=upperv;n++)
        {
            ReservoirBlocks[n].properties=RockTypes[defaultv];
        }
    }
}
void CReservoirDim::RunProgram()
{
    GetResDimension (); // functions call
    SetBlockDimensions();
    SetRockProperties();
    SetGrid();
    SetBlockPositions();
    SetThermoDynCondition();
    //set files
}

```

```

ReservoirDim.cpp
SetPressureFiles();
SetTemperatureFiles();
SetTracerFiles();

UpdatePressure(0);
UpdateTemperature(0);
UpdateTracer(0);

for(int time=86400; time<TimeMax;time++)
{
    if(14*86400<time&&time<((14*86400)+3600))
    {
        ReservoirBlocks[22].STracer =0.007;//tracer is injected only
when timestep is 25 else at any other timestep source tracer is zero
        ReservoirBlocks[22].TracerConc=25.0;
    }else{ReservoirBlocks[22].STracer =0.0;
    }

    PressureIterator();
    TracerIterator();

    if(time<86400||time%86400==0)
    {
        UpdatePressure(time);
        UpdateTemperature(time);

        UpdateTracer(time);
    }

    //set temperature files

    Display();

cout<<endl<<endl<<setiosflags(ios::fixed)<<setprecision(8)<<TestConve
rgence()<<endl;
}

void CReservoirDim::SetThermoDynCondition()
{
    double depth=LengthZ/(2*BlocksZ);

    for (int n=0;n<BlockNumber;n++)
    {
        ReservoirBlocks[n].Pressure=(depth*(
1+2*ReservoirBlocks[n].BLayer)/(9.95));
        ReservoirBlocks[n].ComputeTemp();
        ReservoirBlocks[n].ComputeViscosity();
        ReservoirBlocks[n].GenerateRateConstant();
        ReservoirBlocks[n].SFluid =0.0;
        ReservoirBlocks[n].SHeat =0.0;
        ReservoirBlocks[n].STracer=0.0;
        ReservoirBlocks[n].TracerConc=0.0;
    }

    ReservoirBlocks[22].SFluid =0.03;
    //ReservoirBlocks[20].STracer =2.0;
    // ReservoirBlocks[20].TracerConc=0.02;
}

void CBlock::ComputeTemp()
{
Temperature=69.314+21.823*sqrt(1.0/Pressure)+75.475*log10(Pressure)+8

```

```

ReservoirDim.cpp
.9496*sqrt(Pressure)-8.6596*pow(10,-7)*pow(Pressure,3);

double
SPV=0.99185+0.049269*sqrt(1.0/Pressure)+0.095166*log10(Pressure)+0.00
24859*Pressure+1.6329*pow(10,-10)*pow(Pressure,4);

WaterDensity=1.0/SPV;
}
void CBlock::ComputeViscosity()
{
    if (Temperature<=150.0) Viscosity=179.8*pow(10,-6);
    else if(Temperature<=330.0)
Viscosity=25772*pow(Temperature,-0.99)*pow(10,-6);
    else
Viscosity=(-0.010*pow(Temperature,2)+6.851*Temperature-1011)*pow(10,-
6);
}

void CReservoirDim::PressureIterator()//continues to solve pressure
until convergence is achieved
{
    for(int n=0;n<BlockNumber;n++)
    {
        InterMed1[n]=ReservoirBlocks[n].Pressure;//calculates
InterMed1[n] up to BlockNumber
    }
    do
    {
        solvePressure();
    }while(TestConvergence())>0.0001);
    for( n=0;n<BlockNumber;n++)
    {
        ReservoirBlocks[n].Pressure=InterMed1[n];//updates
pressure in blk j
        ReservoirBlocks[n].ComputeTemp();
    }
}
void CReservoirDim::SolvePressure()
{
    double alpha,beta,gamma,gammasum;

    for(int j=0;j<BlockNumber;j++)//repetition statement
    {
        //initialisation
        beta=ReservoirBlocks[j].properties.Porosity*ReservoirBlocks[j].Volume
*ReservoirBlocks[j].properties.Compressibility*ReservoirBlocks[j].Wat
erDensity/TimeStep;
        alpha=beta;//initialisation/copy value of beta onto
        alpha
        gammasum=0;//initilialisation

        //set front block
        if(ReservoirGrid[j].FrontBlock>=0)//test if the block
is on the perimeter
        {
            gamma=ReservoirBlocks[j].properties.HPermeability*ReservoirBlocks[j].
LengthY*ReservoirBlocks[j].LengthZ*ReservoirBlocks[j].WaterDensity/(R
eservoirBlocks[j].LengthX*ReservoirBlocks[j].Viscosity);
            alpha+=gamma;//add values of gamma on alpha
            successively
            gammasum+=gamma*InterMed1[ReservoirGrid[j].FrontBlock];

```

ReservoirDim.cpp

```

    }
    //set rear block
    if(ReservoirGrid[j].RearBlock>=0)
    {
gamma=ReservoirBlocks[j].properties.HPermeability*ReservoirBlocks[j].
LengthY*ReservoirBlocks[j].LengthZ*ReservoirBlocks[j].WaterDensity/(R
eservoirBlocks[j].LengthX*ReservoirBlocks[j].Viscosity);
        alpha+=gamma;

gammaSum+=gamma*InterMed1[ReservoirGrid[j].RearBlock];
    }
    //set right block
    if(ReservoirGrid[j].RightBlock>=0)
    {

gamma=ReservoirBlocks[j].properties.HPermeability*ReservoirBlocks[j].
LengthX*ReservoirBlocks[j].LengthZ*ReservoirBlocks[j].WaterDensity/(R
eservoirBlocks[j].LengthY*ReservoirBlocks[j].Viscosity);
        alpha+=gamma;

gammaSum+=gamma*InterMed1[ReservoirGrid[j].RightBlock];
    }

    //set left block
    if(ReservoirGrid[j].LeftBlock>=0)
    {

gamma=ReservoirBlocks[j].properties.HPermeability*ReservoirBlocks[j].
LengthX*ReservoirBlocks[j].LengthZ*ReservoirBlocks[j].WaterDensity/(R
eservoirBlocks[j].LengthY*ReservoirBlocks[j].Viscosity);
        alpha+=gamma;

gammaSum+=gamma*InterMed1[ReservoirGrid[j].LeftBlock];
    }

    /*
    //set top block
    if(ReservoirGrid[j].UpBlock>0)
    {

gamma=ReservoirBlocks[j].properties.VPermeability*ReservoirBlocks[j].
LengthX*ReservoirBlocks[j].LengthY*ReservoirBlocks[j].WaterDensity/(R
eservoirBlocks[j].LengthZ*ReservoirBlocks[j].Viscosity);
        //alpha+=gamma;
        gammaSum-=gamma*(ReservoirBlocks[j].Pressure
-ReservoirBlocks[ReservoirGrid[j].UpBlock].Pressure
-9.81*ReservoirBlocks[j].LengthZ*ReservoirBlocks[j].WaterDensity);

//gammaSum+=gamma*9.81*ReservoirBlocks[j].LengthZ*ReservoirBlocks[j].
WaterDensity;

        //
InterMed1[ReservoirGrid[j].UpBlock]-InterMed1[ReservoirGrid[j].LeftB
lock]) ==-5;
    }

    if(ReservoirGrid[j].LowBlock>0)
    {

gamma=ReservoirBlocks[j].properties.VPermeability*ReservoirBlocks[j].
LengthX*ReservoirBlocks[j].LengthY*ReservoirBlocks[j].WaterDensity/(R
eservoirBlocks[j].LengthZ*ReservoirBlocks[j].Viscosity);
        //alpha+=gamma;

gammaSum-=gamma*(ReservoirBlocks[ReservoirGrid[j].LowBlock].Pressure-

```

ReservoirDim.cpp

```

ReservoirBlocks[j].Pressure
-9.81*ReservoirBlocks[j].LengthZ*ReservoirBlocks[j].WaterDensity);

//gammasum+=gamma*9.81*ReservoirBlocks[j].LengthZ*ReservoirBlocks[j].
WaterDensity;

//
InterMed1[ReservoirGrid[j].UpBlock]-InterMed1[ReservoirGrid[j].LeftBl
ock]) ==-5;
    }*/
    //set source

InterMed2[j]=(beta*ReservoirBlocks[j].Pressure+gammasum+ReservoirBloc
ks[j].SFluid)/alpha;//Calculates pressure of blk j at the next time
step.
    InterMed[j]=InterMed1[j];//frees InterMed1[j] so that
new pressure of blk j can be stored in it.
    InterMed1[j]=InterMed2[j];//Updates InterMed1[j].
}
}

double CReservoirDim::TestConvergence()
{
    double value=fabs(InterMed1[0]-InterMed[0]);//declaration and
initialization of value

    for(int j=0;j<BlockNumber;j++)
    {
        if(fabs(InterMed1[j]-InterMed[j])> value)
value=fabs(InterMed1[j]-InterMed[j]);
    }

    return value;
}

void CReservoirDim::SetPressureFiles()
{
    ofstream        outfile("Pressure.txt");

    //Title
    outfile<<"Time"<<setw(10);

    for(int j=0;j<BlockNumber;j++)  outfile<<"Blk"<<j<<setw(15);
    outfile<<endl;
}

void CReservoirDim::UpdatePressure(int time)
{
    ofstream        outfile("Pressure.txt",ios::app);

    outfile.eof() ;
    outfile<<time<<setw(10);

    for(int j=0;j<BlockNumber;j++)
outfile<<ReservoirBlocks[j].Pressure<<setw(15);
    outfile<<endl;
}

void CReservoirDim::SetTemperatureFiles()
{
    ofstream        outfile("Temperature.txt");

```

```

ReservoirDim.cpp
outfile<<"Time"<<setw(10);

for(int j=0;j<BlockNumber;j++) outfile<<"Blk"<<j<<setw(15);
outfile<<endl;
}
void CReservoirDim::UpdateTemperature(int time)
{
ofstream outfile("Temperature.txt",ios::app);

outfile.eof() ;
outfile<<time<<setw(10);

for(int j=0;j<BlockNumber;j++)
outfile<<ReservoirBlocks[j].Temperature<<setw(25);
outfile<<endl;
}

void CReservoirDim::SolveTracer()
{
double alpha,beta,gamma,gammasum,gammasum2=0.0,volume;

for(int j=0;j<BlockNumber;j++)//repetition statement
{
beta=gamma=gammasum=gammasum2=0.0;

volume=ReservoirBlocks[j].properties.Porosity*ReservoirBlocks[j].LengthX*ReservoirBlocks[j].LengthY*ReservoirBlocks[j].LengthZ;

alpha=ReservoirBlocks[j].TracerConc*volume;

//set front block
if(ReservoirGrid[j].FrontBlock>=0)
{
gammasum+=ReservoirBlocks[j].properties.Diffusivity*InterMed1[ReservoirGrid[j].FrontBlock]*ReservoirBlocks[j].LengthX*ReservoirBlocks[j].LengthY/ReservoirBlocks[j].LengthX;

gammasum2+=ReservoirBlocks[j].properties.Diffusivity*ReservoirBlocks[j].LengthX*ReservoirBlocks[j].LengthY/ReservoirBlocks[j].LengthX;

if(ReservoirBlocks[j].Pressure>ReservoirBlocks[ReservoirGrid[j].FrontBlock].Pressure)
{
gamma+=(ReservoirBlocks[ReservoirGrid[j].FrontBlock].Pressure-ReservoirBlocks[j].Pressure)*ReservoirBlocks[j].properties.HPermeability*ReservoirBlocks[j].LengthY*ReservoirBlocks[j].LengthZ*TimeStep/(ReservoirBlocks[j].LengthX*ReservoirBlocks[j].Viscosity);
}else{

beta+=InterMed1[ReservoirGrid[j].FrontBlock]*(ReservoirBlocks[ReservoirGrid[j].FrontBlock].Pressure-ReservoirBlocks[j].Pressure)*ReservoirBlocks[j].properties.HPermeability*ReservoirBlocks[j].LengthY*ReservoirBlocks[j].LengthZ*TimeStep/(ReservoirBlocks[j].LengthX*ReservoirBlocks[j].Viscosity);
}
}

//set rear block
if(ReservoirGrid[j].RearBlock>=0)
{

gammasum+=ReservoirBlocks[j].properties.Diffusivity*InterMed1[ReservoirGrid[j].RearBlock]*ReservoirBlocks[j].LengthX*ReservoirBlocks[j].LengthY/ReservoirBlocks[j].LengthX;
}
}
}

```



```

lengthY/ReservoirBlocks[j].LengthX;

gammasum2+=ReservoirBlocks[j].properties.Diffusivity*ReservoirBlocks[
j].LengthX*ReservoirBlocks[j].LengthY/ReservoirBlocks[j].LengthX;

if(ReservoirBlocks[j].Pressure>ReservoirBlocks[ReservoirGrid[j].RearB
lock].Pressure)
{

gamma+=(ReservoirBlocks[ReservoirGrid[j].RearBlock].Pressure-Reservo
irBlocks[j].Pressure)*ReservoirBlocks[j].properties.HPermeability*Rese
rvoirBlocks[j].LengthY*ReservoirBlocks[j].LengthZ*TimeStep/(Reservoir
Blocks[j].LengthX*ReservoirBlocks[j].Viscosity);
}else{

beta+=InterMed1[ReservoirGrid[j].RearBlock]*(ReservoirBlocks[Reservo
irGrid[j].RearBlock].Pressure-ReservoirBlocks[j].Pressure)*ReservoirB
locks[j].properties.HPermeability*ReservoirBlocks[j].LengthY*Reservo
irBlocks[j].LengthZ*TimeStep/(ReservoirBlocks[j].LengthX*ReservoirBlok
s[j].Viscosity);
}

//set right block
if(ReservoirGrid[j].RightBlock>=0)
{

gammasum+=ReservoirBlocks[j].properties.Diffusivity*InterMed1[Reservo
irGrid[j].RightBlock]*ReservoirBlocks[j].LengthX*ReservoirBlocks[j].L
engthY/ReservoirBlocks[j].LengthX;

gammasum2+=ReservoirBlocks[j].properties.Diffusivity*ReservoirBlocks[
j].LengthX*ReservoirBlocks[j].LengthY/ReservoirBlocks[j].LengthX;

if(ReservoirBlocks[j].Pressure>ReservoirBlocks[ReservoirGrid[j].Right
Block].Pressure)
{

gamma+=(ReservoirBlocks[ReservoirGrid[j].RightBlock].Pressure-Reservo
irBlocks[j].Pressure)*ReservoirBlocks[j].properties.HPermeability*Rese
rvoirBlocks[j].LengthY*ReservoirBlocks[j].LengthZ*TimeStep/(Reservoir
rBlocks[j].LengthX*ReservoirBlocks[j].Viscosity);
}else{

beta+=InterMed1[ReservoirGrid[j].RightBlock]*(ReservoirBlocks[Reservo
irGrid[j].RightBlock].Pressure-ReservoirBlocks[j].Pressure)*Reservoir
Blocks[j].properties.HPermeability*ReservoirBlocks[j].LengthY*Reservo
irBlocks[j].LengthZ*TimeStep/(ReservoirBlocks[j].LengthX*ReservoirBlo
cks[j].Viscosity);
}

//set left block
if(ReservoirGrid[j].LeftBlock>=0)
{

gammasum+=ReservoirBlocks[j].properties.Diffusivity*InterMed1[Reservo
irGrid[j].LeftBlock]*ReservoirBlocks[j].LengthX*ReservoirBlocks[j].Le
ngthY/ReservoirBlocks[j].LengthX;

gammasum2+=ReservoirBlocks[j].properties.Diffusivity*ReservoirBlocks[
j].LengthX*ReservoirBlocks[j].LengthY/ReservoirBlocks[j].LengthX;

if(ReservoirBlocks[j].Pressure>ReservoirBlocks[ReservoirGrid[j].LeftB
lock].Pressure)

```

ReservoirDim.cpp

```

    {
gamma+=(ReservoirBlocks[ReservoirGrid[j].LeftBlock].Pressure-ReservoirBlocks[j].Pressure)*ReservoirBlocks[j].properties.HPermeability*ReservoirBlocks[j].LengthY*ReservoirBlocks[j].LengthZ*TimeStep/(ReservoirBlocks[j].LengthX*ReservoirBlocks[j].Viscosity);
    }else{
beta+=InterMed1[ReservoirGrid[j].LeftBlock]*(ReservoirBlocks[ReservoirGrid[j].LeftBlock].Pressure-ReservoirBlocks[j].Pressure)*ReservoirBlocks[j].properties.HPermeability*ReservoirBlocks[j].LengthY*ReservoirBlocks[j].LengthZ*TimeStep/(ReservoirBlocks[j].LengthX*ReservoirBlocks[j].Viscosity);
    }
}

InterMed2[j]=(alpha+beta*TimeStep+ReservoirBlocks[j].STracer*TimeStep+gammasum*TimeStep)/(volume-gamma*TimeStep+gammasum2*TimeStep-ReservoirBlocks[j].RateConstant*TimeStep*ReservoirBlocks[j].LengthY*ReservoirBlocks[j].LengthZ*ReservoirBlocks[j].LengthX);/*300000);/*0.00000000477669=4.77669p0w(10,-10)
InterMed[j]=InterMed1[j];//frees InterMed1[j] so that
new pressure of blk j can be stored in it.
InterMed1[j]=InterMed2[j];//Updates InterMed1[j].
    }
}

void CReservoirDim::TracerIterator()
{
    for(int n=0;n<BlockNumber;n++)
    {
        InterMed1[n]=ReservoirBlocks[n].TracerConc;//assigns
values of tracerconc onto InterMed1
    }
    int num=0;
    do
    {
        solveTracer();num++;

    }while(num<10);/*TestConvergence()>0.0000000001);

    for( n=0;n<BlockNumber;n++)
        ReservoirBlocks[n].TracerConc=InterMed1[n];
}

void CReservoirDim::SetTracerFiles()
{
    ofstream outfile("Tracer.txt");
    outfile<<"Time"<<setw(10);
    for(int j=0;j<BlockNumber;j++) outfile<<"Blk"<<j<<setw(15);
    outfile<<endl
}

void CReservoirDim::UpdateTracer(int time)
{
    ofstream outfile("Tracer.txt",ios::app);
    outfile.eof();
    outfile<<time<<setw(25);
}

```

ReservoirDim.cpp

```
for(int j=0;j<BlockNumber;j++)
outfile<<ReservoirBlocks[j].TracerConc<<setw(25);
outfile<<endl;
}
void CBlock::GenerateRateConstant()
{RateConstant=(exp(18.25)*exp(-143300/(8.31*210)))*pow(10,14);
//RateConstant=1.39*pow(10,-6);
}
```

APPENDIX 11 FIELD POWER OW-18

time (sec)	concentration
0	0.000225768
86400	0.000244582
86400	0.000263396
172800	0.000206954
172800	0.00018814
259200	0.000150512
259200	0.000150512
345600	0.000150512
345600	0.000150512
432000	0.00018814
432000	0.000131698
518400	0.000150512
518400	0.000150512
604800	0.000150512
604800	0.00018814
691200	0.000150512
691200	0.00018814
777600	0.00018814
777600	0.000112884
864000	0.000112884
864000	0.000112884
950400	0.000112884
950400	0.000112884
1036800	0.000112884
1036800	0.000112884
1123200	0.000112884
1123200	0.000112884
1209600	0.000112884
1209600	0.000150512
1296000	0.000225768
1382400	0.000714932
1468800	0.001204096
1555200	0.001655632
1641600	0.00206954
1641600	0.002220052
1728000	0.002370564
1728000	0.002784472
1814400	0.00310431
1814400	0.003047868
1900800	0.003254822
1900800	0.003424148
1987200	0.004440104
2073600	0.007788996
2160000	0.01458085
2160000	0.011514168

2246400	0.005531
2246400	0.010404
2332800	0.016199
2332800	0.014713
2419200	0.017215
2505600	0.01618
2505600	0.018287
2592000	0.018287
2592000	0.015917
2678400	0.017064
2678400	0.016763
2764800	0.013
2764800	0.016989
2851200	0.017234
2851200	0.018758
2937600	0.013396
2937600	0.007526
3024000	0.011495
3110400	0.014938
3369600	0.032718
3456000	0.036349
3542400	0.032548
3628800	0.013734
3715200	0.010536
3974400	0.021636
4060800	0.013546
4147200	0.01872
4233600	0.076874
4320000	0.047976
4579200	0.070778
5788800	0.030516
5875200	0.041654
5961600	0.025361
6134400	0.039321
6393600	0.0552
6480000	0.038155
6566400	0.01618
6652800	0.032435
6739200	0.031909
7084800	0.028221
7171200	0.028635
7257600	0.023028
7344000	0.023555
7603200	0.018513
7689600	0.01682

7776000	0.035803
7862400	0.032925
7948800	0.035521
8208000	0.030893
8294400	0.020394
8380800	0.005926
8467200	0.033922
8553600	0.017064
8812800	0.010799
8985600	0.014788
9590400	0.016105
10195200	0.006547
10800000	0.002314
11404800	0.001769
12009600	0.001881
12441600	0.00207
13305600	0.001204
13824000	0.000978
14515200	0.001392
15033600	0.001054
15638400	0.001091
16243200	0.000715
16848000	0.000527
17452800	0.000677
18057600	0.00079
18662400	0.000753
19267200	0.000677
19872000	0.000602
20476800	0.001016
21081600	0.000602
21686400	0.001072
22291200	0.000978
22809600	0.000941
24105600	0.000828

days	Time	field data(conc kg/m3)	WELL OW-19
0	0	0.00028221	46 3974400 0.014957
1	86400	0.000301024	47 4060800 0.013546
2	172800	0.00018814	48 4147200 0.018344
3	259200	0.000225768	49 4233600 0.047223
4	345600	0.000225768	50 4320000 0.054467
5	432000	0.000225768	51 4406400
6	518400	0.00018814	52 4492800
7	604800	0.000225768	53 4579200 0.062011
8	691200	0.000225768	54 4665600 0.021899
9	777600	0.00018814	55 4752000 0.028898
10	864000	0.000150512	56 4838400 0.018287
11	950400	0.000150512	57 4924800 0.008052
12	1036800	0.000263396	58 5011200
13	1123200	0.00075256	59 5097600
14	1209600	0.001392236	60 5184000 0.029688
15	1296000	0.002107168	61 5270400 0.02491
16	1382400	0.002859728	62 5356800 0.023743
17	1468800	0.0056442	63 5443200 0.031984
18	1555200	0.013433196	64 5529600 0.032548
19	1641600	0.024533456	65 5616000
20	1728000	0.020168608	66 5702400
21	1814400	0.029500352	67 5788800 0.035747
22	1900800	0.02483448	68 5875200 0.0295
23	1987200	0.061070244	69 5961600 0.027694
24	2073600	0.04590616	70 6048000
25	2160000	0.01890807	71 6134400 0.025173
26	2246400	0.01919028	72 6220800
27	2332800	0.034034526	73 6307200
28	2419200	0.02991426	74 6393600 0.014675
29	2505600	0.055087392	75 6480000 0.013998
30	2592000	0.061258384	76 6566400 0.016105
31	2678400	0.049631332	77 6652800 0.013433
32	2764800	0.053205992	78 6739200 0.037289
33	2851200	0.06979994	79 6825600
34	2937600	0.06001666	80 6912000
35	3024000	0.01636818	81 6998400
36	3110400	0.0047035	82 7084800 0.029519
37	3196800		83 7171200 0.018588
38	3283200		84 7257600 0.029839
39	3369600	0.01241724	85 7344000 0.031231
40	3456000	0.01486306	86 7430400
41	3542400	0.01495713	87 7516800
42	3628800	0.01354608	88 7603200 0.024289
43	3715200	0.018418906	89 7689600
44	3801600		90 7776000 0.022633
45	3888000		91 7862400 0.022633
			92 7948800 0.024834

93	3974400	0.01495713	140	12096000	
94	4060800	0.01354608	141	12182400	
95	4147200	0.01834365	142	12268800	
96	4233600	0.04722314	143	12355200	
97	4320000	0.05446653	144	12441600	0.007074
98	4406400		145	12528000	
99	4492800		146	12614400	
100	4579200	0.062010944	147	12700800	
101	4665600	0.021899496	148	12787200	
102	4752000	0.028898304	149	12873600	
103	4838400	0.018287208	150	12960000	
104	4924800	0.008052392	151	13046400	
105	5011200		152	13132800	
106	5097600		153	13219200	
107	5184000	0.029688492	154	13305600	0.006359
108	5270400	0.024909736	155	13392000	
109	5356800	0.023743268	156	13478400	
110	5443200	0.0319838	157	13564800	
111	5529600	0.03254822	158	13651200	
112	5616000		159	13737600	
113	5702400		160	13824000	0.005738
114	5788800	0.0357466	161	13910400	
115	5875200	0.029500352	162	13996800	
116	5961600	0.027694208	163	14083200	
117	6048000		164	14169600	
118	6134400	0.025173132	165	14256000	
119	6220800		166	14342400	
120	6307200		167	14428800	
121	6393600	0.01467492	168	14515200	0.003537
122	6480000	0.013997616	169	14601600	
123	6566400	0.016104784	170	14688000	
124	6652800	0.013433196	171	14774400	
125	6739200	0.037289348	172	14860800	
126	6825600		173	14947200	
127	6912000		174	15033600	0.00523
128	6998400		175	15120000	
129	7084800	0.029519166	176	15206400	
130	7171200	0.018588232	177	15292800	
131	7257600	0.029839004	178	15379200	
132	7344000	0.03123124	179	15465600	
133	7430400		180	15552000	
134	7516800		181	15638400	0.003123
135	7603200	0.024288874	182	15724800	
136	7689600		183	15811200	
137	7776000	0.022633242	184	15897600	
138	7862400	0.022633242	185	15984000	
139	7948800	0.02483448	186	16070400	

187	16156800		234	20217600	
188	16243200	0.00338652	235	20304000	
189	16329600		236	20390400	
190	16416000		237	20476800	0.002879
191	16502400		238	20563200	
192	16588800		239	20649600	
193	16675200		240	20736000	
194	16761600		241	20822400	
195	16848000	0.002784472	242	20908800	
196	16934400		243	20995200	
197	17020800		244	21081600	0.003067
198	17107200		245	21168000	
199	17193600		246	21254400	
200	17280000		247	21340800	
201	17366400		248	21427200	
202	17452800	0.004854012	249	21513600	
203	17539200		250	21600000	
204	17625600		251	21686400	0.003387
205	17712000		252	21772800	
206	17798400		253	21859200	
207	17884800		254	21945600	
208	17971200		255	22032000	
209	18057600	0.003461776	256	22118400	
210	18144000		257	22204800	
211	18230400		258	22291200	0.003236
212	18316800		259	22377600	
213	18403200		260	22464000	
214	18489600		261	22550400	
215	18576000		262	22636800	
216	18662400	0.00319838	263	22723200	
217	18748800		264	22809600	0.003085
218	18835200		265	22896000	
219	18921600		266	22982400	
220	19008000		267	23068800	
221	19094400		268	23155200	
222	19180800		269	23241600	
223	19267200	0.003537032	270	23328000	
224	19353600		271	23414400	
225	19440000		272	23500800	
226	19526400		273	23587200	
227	19612800		274	23673600	
228	19699200		275	23760000	
229	19785600		276	23846400	
230	19872000	0.003236008	277	23932800	
231	19958400		278	24019200	
232	20044800		279	24105600	0.003236