5-1994

An Inverse Approach to a Probability Model for Fractured Networks

Stacy G. Vail
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

Part of the Mathematics Commons

Recommended Citation
https://digitalcommons.usu.edu/etd/7128

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.
AN INVERSE APPROACH TO A PROBABILITY MODEL FOR FRACTURED NETWORKS

by

Stacy G. Vail

A thesis submitted in partial fulfillment of the requirements for the degree

of

MASTER OF SCIENCE

in

Mathematics

UTAH STATE UNIVERSITY
Logan, Utah
1994
ACKNOWLEDGMENTS

Dr. Kevin Hestir has shown incredible patience throughout the development of this thesis and throughout my master’s program. His ability to guide me along while letting me learn things on my own was a great strength to me, and helped me start to acquire a taste for research. I thank him for the time, patience, and encouragement he invested.

I also thank my other committee members, Dr. Joseph Koebbe and Dr. James Evans. I felt extremely fortunate to have a committee who truly wanted me to succeed, and offered help wherever needed. Dr. Koebbe was very helpful by being persistent and by offering advice on my thesis and on continuing my math education. Dr. Evans’ advice during my defense and comments on this thesis were very helpful.

This thesis was funded in part by the Assistant Secretary for Fossil Energy, Office of Oil, Gas and Shale Technologies of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. Funding also came from the National Science Foundation under Grant No. DMS-920054. I thank Jane Long at Lawrence Berkeley Laboratory and Dr. Hestir for making this support available to me.

Finally, I thank my family for their constant support and confidence. Their encouragement is a great influence on me.

Stacy Vail
# CONTENTS

ACKNOWLEDGMENTS ................................................. ii
LIST OF FIGURES ................................................... v
ABSTRACT ........................................................... viii

1 INTRODUCTION ..................................................... 1

2 THE PROBABILITY MODEL ........................................... 5
   2.1 Geologic Observations ....................................... 5
   2.2 Creating the Probability Model .............................. 7
   2.3 Analysis of the Model ....................................... 11

3 THE INVERSE MODEL ............................................... 14
   3.1 What Is an Inverse Approach? ............................... 14
   3.2 Dealing with Nonuniqueness ................................ 18
      3.2.1 Constraint Method .................................... 19
      3.2.2 Probability Method ................................... 19
   3.3 The Inverse Model ......................................... 20
   3.4 The Simulated Annealing Algorithm ......................... 21
      3.4.1 Fundamental Concepts of Simulated Annealing ....... 21
      3.4.2 Justification for the Metropolis Algorithm .......... 23
      3.4.3 Markov Chains ....................................... 23
      3.4.4 The Annealing Schedule ............................... 24
   3.5 The Conditional Coding Method ............................. 25

4 FLOW EQUATION AND NUMERICAL METHODS ....................... 27
   4.1 Equation Modeling Flow ..................................... 27
   4.2 Numerical Methods ........................................ 29
      4.2.1 Galerkin Finite Element Method ....................... 30
      4.2.2 Finite Difference Method .............................. 32

5 NUMERICAL RESULTS ............................................. 33
   5.1 Data Set 1 ................................................ 33
      5.1.1 Observations ......................................... 36
   5.2 Data Set 2 ................................................ 61
      5.2.1 Observations ......................................... 64
   5.3 Data Set 3 - Real Data .................................... 74
      5.3.1 Observations ......................................... 75

6 CONCLUSIONS .................................................... 82
# LIST OF FIGURES

2.1 Map view of surface fracture patterns at Conoco’s Borehole Test Facility. . 8
2.2 A sample of the starting fractures generated from the probability model. . 10
2.3 Illustrated growth process from the probability model: (a) first stage of growth with dominant starting fractures only, (b) completion of growth of these dominant fractures, (c) nondominant starting fractures added, and (d) completion of the configuration by growing the nondominant fractures. 12

3.1 Actual reservoir fracture pattern and exact head response from the wells (no noise in these head values). ................................. 16
3.2 Limited area of configuration from which observed response comes, and observed response from the wells, including noise. ................................. 17
3.3 Illustration of the nonuniqueness of the inverse problem. ................................. 18

5.1 Simulated fracture pattern for a 50x50 grid. This pattern represents the actual structure for all tests run on this grid. ........................................ 34
5.2 Pressure curves for each well, obtained by calculating flow on the simulated fracture pattern on the 50x50 element grid, with noise added to simulated actual observations. ........................................ 35
5.3 First sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was d=0.995. ........................................ 37
5.4 Energy profile from the first run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was d=0.995. ................................. 38
5.5 Pressure curves for each well, obtained by calculating flow on the first fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1. ........................................ 39
5.6 Second sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was d=0.995. ........................................ 40
5.7 Energy profile from the second run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was d=0.995. ................................. 41
5.8 Pressure curves for each well, obtained by calculating flow on the second fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1. ........................................ 42
5.9 Third sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was d=0.995. ........................................ 43
5.10 Energy profile from the third run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was d=0.995. ................................. 44
5.11 Pressure curves for each well, obtained by calculating flow on the third fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1. ........................................ 45

5.12 Fourth sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.99575$. ........................................ 46

5.13 Energy profile from the fourth run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.99575$. ........................................ 47

5.14 Pressure curves for each well, obtained by calculating flow on the fourth fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1. ........................................ 48

5.15 Fifth sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. ........................................ 49

5.16 Energy profile from the fifth run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. ........................................ 50

5.17 Pressure curves for each well, obtained by calculating flow on the fifth fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1. ........................................ 51

5.18 Sixth sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.998$. ........................................ 52

5.19 Energy profile from the sixth run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. ........................................ 53

5.20 Pressure curves for each well, obtained by calculating flow on the sixth fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1. ........................................ 54

5.21 Seventh sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. ........................................ 55

5.22 Energy profile from the seventh run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. ........................................ 56

5.23 Pressure curves for each well, obtained by calculating flow on the seventh fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1. ........................................ 57

5.24 Eighth sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. ........................................ 58

5.25 Energy profile from the eighth run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. ........................................ 59

5.26 Pressure curves for each well, obtained by calculating flow on the eighth fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1. ........................................ 60

5.27 Simulated fracture pattern for a 200x200 grid. This pattern represents the actual structure for all tests run on this grid. ........................................ 62
5.28 Pressure curves for each well, obtained by calculating flow on the simulated fracture pattern on the 200x200 element grid, with noise added to simulated actual observations. ........................................... 63

5.29 First sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. ........................................... 65

5.30 Energy profile from the first run of the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. ... 66

5.31 Pressure curves for each well, obtained by calculating flow on the first fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 2. ........................................... 67

5.32 Second sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. ........................................... 68

5.33 Energy profile from the second run of the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. ... 69

5.34 Pressure curves for each well, obtained by calculating flow on the second fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 2. ........................................... 70

5.35 Third sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. ........................................... 71

5.36 Energy profile from the third run of the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. ... 72

5.37 Pressure curves for each well, obtained by calculating flow on the third fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 2. ........................................... 73

5.38 Well tests taken from Conoco Site. ........................................... 74

5.39 First sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 3. The temperature decrease factor was $d=0.9975$. ........................................... 76

5.40 Energy profile from the first run of the Conditional Coding Method on Simulated Data Set 3. The temperature decrease factor was $d=0.9975$. ... 77

5.41 Pressure curves for each well, obtained by calculating flow on the first fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 3. ........................................... 78

5.42 Second sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 3. The temperature decrease factor was $d=0.9975$. ........................................... 79

5.43 Energy profile from the second run of the Conditional Coding Method on Simulated Data Set 3. The temperature decrease factor was $d=0.9975$. ... 80

5.44 Pressure curves for each well, obtained by calculating flow on the second fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 3. ........................................... 81
ABSTRACT

An Inverse Approach to a Probability Model for Fractured Networks

by

Stacy G. Vail, Master of Science
Utah State University, 1994

A common problem in science and engineering applications deals with finding information about a system where only limited information is known. One example of this problem is determining the geometry of an aquifer or oil reservoir based on well tests taken at the site. The Conditional Coding Method attacks this type of problem. This method uses the Simulated Annealing Algorithm in conjunction with a probability model which generates possible solutions based on a uniform random number list. The Annealing Algorithm generates a conditional probability distribution on all possible solutions generated by the probability model, conditioned on the observed data set. The problem is attacked by sampling from this distribution. This method accounts for the noise inherent in the data set as well as the uncertainty due to the limited amount of data available.
A common problem in science and engineering applications deals with finding information about a system where only limited data are available from the system. The desired information may include qualities representing portions or all of the system. A formal definition of this problem follows, and will be referred to throughout this paper as Problem P1.

**Definition 1 (Problem P1)** Given a limited amount of measured data from some system, determine other qualities of the system.

An example illustrating this problem comes from geology. Geologists often study aquifers and oil reservoirs, and want to know the spatial distribution of the physical characteristics which allow fluid flow through this structure. However, the information available from the system is generally limited to well tests taken at a few well locations, or some other type of limited flow measurements taken at the site. How does one use this limited amount of information from the system to determine the solution to Problem P1? In terms of the example stated above, how does one use the available well tests to determine where fluid flow occurs throughout the structure?

There are some obstacles to deal with in solving Problem P1. Due to the limited amount of data, there may not be a unique solution to Problem P1. Many approximations may represent reasonable matches to the given observation set. Additionally, one would expect noise to be found in the observed data set. How does one account for these factors
when solving Problem $P1$? This thesis presents a method to address Problem $P1$. This method addresses the nonuniqueness and noise issues. The type of geologic formation addressed in this paper is a fractured rock system. Again, a formal definition describes this specific case of Problem $P1$. This case will be referred to as Problem $P1^*$:

**Definition 2 (Problem $P1^*$)** *Given well tests at a few distinct locations in a fractured rock system, determine where the fractures lie, and thus where fluid flows through the system.*

This method can be applied to other problems which fit the definition of Problem $P1$.

Problem $P1$ (and thus Problem $P1^*$) is well-suited for an inverse approach. Given the location of all fractures in a reservoir, the flow (head) measurements at any location in the reservoir can be determined. However, the question at hand requires making the inverse calculation: given the head measurements at a few distinct locations, the location of the fractures must be determined. This constitutes an inverse problem.

The method presented in this paper is comprised of two separate algorithms. The first is an inverse model used to solve the inverse problem. The inverse model incorporates the Simulated Annealing Algorithm. The second is a probability model, which generates configurations, or in other words, patterns of fracture locations in the system. The approximate solution is then obtained by sampling from a conditional distribution on this probability model. This process is called the Conditional Coding Method. The Conditional Coding Method accounts for the nonuniqueness of the solution, and the noise inherent in the data set. Through sampling from a conditional distribution and finding several reasonable matches, the algorithm quantifies the uncertainty involved in solving Problem $P1$. 
The probability model generates configurations to feed into the inverse model. This probability model is based on physical understanding of how the actual structure was created. By observing a specific fractured reservoir and the presumed geologic rules associated with its creation, representative probability rules are assigned in the algorithm to build an approximate configuration. There may be observations about how these fractures started, how they reacted to intersection, and how the growth rate was affected by length. Each of these aspects is incorporated into the algorithm through probability rules. Within the algorithm, each step of this creation requires a uniform random number for the probability rule. The uniform random numbers used to create this configuration are stored in sequence, or as a list of uniform random numbers. Thus, the list of random numbers determines the configuration. This list of uniform random numbers which represents a geometric configuration is the coded form of the configuration.

The next concept of the algorithm is to condition this probability model based on the observed flow measurements. The probability model uses random number lists to generate configurations, and the flow is calculated for each of these. This predicted flow is compared to the observed flow. This comparison can be considered as defining a function $f$ on the random number list assigning how well it matches the observed flow. We use Simulated Annealing to minimize $f$. Incorporating the Annealing Algorithm creates a conditional probability distribution on the space of possible solutions, conditioned on the flow measurements. By using the Annealing Algorithm as our inverse method, we are sampling from this conditional distribution.

This thesis presents in detail each aspect of the Conditional Coding Method. The probability model used to generate configurations is described in Chapter 2. Chapter 3 is
a description of the inverse method, including a description of the Simulated Annealing Algorithm, and the justification for the Conditional Coding Method. Chapter 4 is a discussion of the equation describing flow through porous media and the numerical method used to calculate flow. Finally, the results of testing this algorithm on simulated and real data are provided in Chapter 5.
CHAPTER 2
THE PROBABILITY MODEL

The first concept discussed is the probability model used to create fracture patterns, or in other words, fracture configurations. The purpose of the model is to computationally generate a geometric pattern which reasonably approximates the actual geometric structure. The model could simply randomly generate configurations with no restrictions placed on how each configuration is created; however, proceeding in this unsystematic manner would result in many useless configurations which would not reasonably match the actual structure. A wiser approach would incorporate into the model known information about the real structure to make a more realistic match of possible fractures, and thus give more credibility to the model-generated pattern. For example, incorporating into the model physical constraints associated with the actual structure could reduce the probability of producing patterns which are physically not possible. The type of information which may be incorporated includes how the fractures began, the age and growth rate of the fractures, interaction between fractures, fracture spacing, etc. We apply these types of observations into the creation of each configuration by assigning probability rules to each step of the pattern-creating model. This process creates a probability distribution on the space of all possible geometric fracture patterns.

2.1 Geologic Observations

Because the model creates a fracture pattern from beginning to end, it is reasonable to incorporate into the model geologic observations about how the site in question formed. It
would not be worthwhile to analyze patterns which do not match the physical properties associated with the structure. This type of information varies with different structures. For our fractured network, we restrict our attention to two-dimensional fractures in a homogeneous isotropic material.

One useful item of geologic interest is determining what causes the fractures to grow. Existing fractures propagate in the direction which minimizes the energy release rate. It can be shown [5] that this release rate is proportional to the length of the fracture. Thus, the growth rate of a fracture is a function of its existing length. Associated with the material is a critical level of release rate for fractures. If the release rate for a fracture exceeds this critical level, the fracture will grow rapidly. If the release rate is below this level, growth can occur but generally occurs at a slower rate. This critical level of release rate can also be converted to a critical fracture length.

Fractures of this nature can also cause other cracks to form, which are called “daughter” fractures. The daughter fracture growth depends on the stresses present in and near an existing “parent” fracture. The stresses present in fractures can be released in two different ways. One way is to release the stress through the parent fracture by increasing the length of the fracture. Another way of release is through a transfer of the stress to a nearby location. This generally happens near the tip of the parent crack off to one side. This will be the location of the daughter crack, which then acts as a parent fracture in terms of propagation and reaction with other fractures. See [5] for a more complete description of these concepts.

Another consideration is how the fractures intersect with one another. The reaction varies with regard to the orientation and propagation direction of the fractures. When two
fractures oriented in the same direction intersect, one larger fracture results. Due to the increased length, the resulting fracture should grow faster, as discussed above. When the intersecting fractures propagate in different directions, the results can vary depending on the material and on the fractures themselves.

We restrict our attention now to a specific fractured network, and the remainder of the discussion and results correspond to analysis on this type of structure. This site, Conoco’s Borehole Test Facility located in Kay County, Oklahoma, was investigated by Queen and Rizer [9]. They summarized some of the physical properties associated with this site. The material at the site consists mainly of limestone and shale. Figure 2.1 shows a map view of the visible fractures at this site. This structure has two distinct, orthogonal fracture sets with the NE-striking set dominating, in terms of fracture length and intersections. The propagation of the less-dominant set, those running NW, appears to have stopped when these fractures intersected the dominant set. From this observation, it is concluded that the NE-striking fractures formed and propagated first, then afterward the NW-striking set was formed. These observations, along with fracture mechanics principles discussed in the previous paragraphs, constitute the information which is incorporated into the probability model.

2.2 Creating the Probability Model

The idea of the probability model is to translate these physical conditions and observations into probability rules which will be used to compute fracture patterns. An essential component of the model is that each decision in the growth process is performed using uniform random numbers. Section 3.5 describes the importance of this component.
Figure 2.1: Map view of surface fracture patterns at Conoco’s Borehole Test Facility.
These numbers, which are used to create the pattern, are stored in sequence. This list of uniform random numbers will be denoted as \( X = x_1, x_2, x_3, \ldots, x_n \). The list determines a geometric configuration, and is referred to as the “coded” form of the pattern. We discuss below each component of the probability model. Chapter 3 relates the significance that the random number list has on the outcome of the algorithm.

Due to the fact that the fracture sets are generally orthogonal at the Conoco test site, we use a rectangular grid as the basis for the geometric approximation. The line segments are the elements of the grid, and the points of intersection are the nodes. The elements represent possible fracture segment locations in the geologic structure. The portions of the grid not represented by elements can be thought of as nonporous areas of the structure. Flow only through line elements is considered with this grid.

The first step of the algorithm selects the location of starting fractures. The algorithm selects a fixed number \( N \) of elements on the grid, and marks each as a starting crack if it corresponds to the dominant direction, as illustrated in Figure 2.2. These initial elements are randomly selected through the values contained in the uniform random number list \( X \). The first \( N \) entries in the list \( X \) determine which elements are selected and tested. This process is then repeated by randomly testing \( M \) elements for the nondominant-direction set. Thus, the first \( N+M \) values of \( X \) characterize the starting locations of fractures.

The next step is to allow the fractures to grow. This occurs through an iteration process. In each iteration, the length and location of each fracture are individually examined and possibly propagated. As discussed above, the probability of growth of a fracture is a function of its length and the critical growth length associated with the system. At each iteration, the probability of growth is given by
where $L$ is the length of the fracture and $L_c$ is the critical length. From this we see that if $L > L_c$, then growth always occurs. Growth may occur if $L < L_c$. For each growth decision, a value is sequentially taken from the random number list $X$ to compare to the value $L/L_c$. If growth occurs, another value from $X$ determines which side of the fracture to extend. Each growth step occurs by increments of one element on the grid. Motivated by the physical observations from the site, we require that the dominant-direction set of fractures be grown completely first; then afterward the second set is propagated. The second set has the added condition that if a fracture intersects another fracture perpendicularly, then any growth which takes place must extend the end away.
from the intersection. Physically, such intersections cause the fracture to act as though its length is doubled in terms of the probability of propagation. Presently, the creation of daughter fractures is not implemented into the code. Figure 2.3 illustrates the process of growth. The growth probability rule 2.1 is used for all fractures through the growth iterations. From this growth iteration process, we see that the tail end of our list $X$ characterizes the amount and direction of growth for each starting fracture.

Perturbations of $X$ are required for the Simulated Annealing Algorithm explained in Chapter 3. The perturbations are accomplished by making changes to the values in $X$. Because of the way the members of the list are sequentially accessed, changes to the first $N+M$ values in $X$ make significant changes in the resulting configuration since these values determine where fractures start. The tail end of the list determines growth of existing fractures. Thus, the change in the pattern decreases as the position where changes to $X$ are made approaches the end of the list. The values of $N$ and $L_c$, as well as the probabilities for growth on each side of a fracture, and the number of growth iterations to perform for each set of fractures clearly affect the outcome of the probability model. Each of these parameters is user-controlled when executing the probability model.

### 2.3 Analysis of the Model

There are several advantages to creating configurations using this probability model. First, it incorporates observed physical information about the structure into the model, thus eliminating physically unreasonable patterns. Second, the principles of fracture mechanics are taken into account, such as the critical level of release rate and allowing fracture growth only near the tip of the fracture. Adding the possibility of daughter fractures will
Figure 2.3: Illustrated growth process from the probability model: (a) first stage of growth with dominant starting fractures only, (b) completion of growth of these dominant fractures, (c) nondominant starting fractures added, and (d) completion of the configuration by growing the nondominant fractures.
further add to this advantage. Third, the number of parameters that control the model is small. Fourth, the model takes relatively little computer time to execute. Fifth, representing the configuration with a list $X$ of uniform random numbers allows making small perturbations to the pattern very simple. And sixth, representing the configuration by the uniform random number list $X$ also assists in the justification for the Conditional Coding Method as discussed in Section 3.5.
CHAPTER 3
THE INVERSE MODEL

The goal of the algorithm is to solve Problem $P1$. Specifically, the goal is to approximate the configuration of a geologic reservoir based on observed head measurements at a few locations in the reservoir. Problem $P1$ can be approached using an inverse model. The inverse model used to achieve this goal utilizes a conditional probability distribution on all configurations produced by the probability model discussed in Chapter 2. This distribution is obtained through the use of the Simulated Annealing Algorithm and the uniform random number list $X$, also discussed in Chapter 2. This method, which incorporates the probability model and the Simulated Annealing Algorithm, we call the Conditional Coding Method. In this chapter each part of the inverse model is described, including an overview of inverse approaches and the Simulated Annealing Algorithm. The justification for the Conditional Coding Method is also given.

3.1 What Is an Inverse Approach?

To outline the inverse approach in a general sense, we begin with the example at hand: approximating geologic reservoir configurations. There is a direct correspondence linking the geometry and permeability of a reservoir system to the head values at any location of the reservoir. Thus, we can interpret this correspondence as a function $f$ associating the configuration of the reservoir to the head measurements. This relationship $f$ may be given implicitly by an equation describing flow through a porous medium. The differential equation used in this application is explained in Chapter 4. However, we
wish to find the inverse relationship $f^{-1}$; that is to say, given the head measurements we wish to find the configuration of the reservoir. The greater amount of data obtained, the better the configuration can be approximated. Due to practical and physical constraints, the data set consisting of the head measurements over the entire region is impossible to obtain. Generally, the available data set consists of head measurements at a relatively small number of wells in the reservoir. Another obstacle is that this data set involves noise, perhaps due to instrument limitations, human error, or other uncontrollable factors. Figures 3.1 and 3.2 demonstrate these points. How does one account for this noise when looking at the inverse relationship? There is still a direct correspondence between the reservoir's geometry and this observed data set, i.e., an associated function $f$ mapping the exact configuration to the pressure observations. However, the inverse image of this set $f^{-1}(\text{observation})$ contains many possible configurations, due both to the small region from which the set was obtained and the noise inherent in it. This data set could be duplicated by many (possibly infinitely many) distinct reservoir configurations. We wish to find an approximate geometry from this set $f^{-1}(\text{observation})$.

This specific example has a straightforward generalization. Given a function $f : Y \rightarrow Z$ for arbitrary spaces $Y$ and $Z$ and given some $y \in Y$, $f(y)$ can be determined. This process is called solving the forward problem. The situation at hand, however, is as follows: given a function $f : Y \rightarrow Z$ and an observation $z \in Z$ where $z = f(y) + \text{noise}$ for some $y \in Y$, we wish to examine the inverse image set $f^{-1}(z)$ and then from this set select an element $\hat{y}$ in $f^{-1}(z)$ which best represents the element $y$ that produced the element $z$. Figure 3.3 illustrates this generalization. The process of finding $\hat{y}$ is known as solving the inverse problem, and appears in a variety of science and engineering applications.
Figure 3.1: Actual reservoir fracture pattern and exact head response from the wells (no noise in these head values).
Figure 3.2: Limited area of configuration from which observed response comes, and observed response from the wells, including noise.
3.2 Dealing with Nonuniqueness

How does one go about selecting an element \( \hat{y} \) from this set \( f^{-1}(z) \) to represent \( y \)? In terms of the application addressed in this paper, how does one find a reasonable approximate configuration of the reservoir from all the configurations which could produce the observed data set?

There are two approaches to deal with this problem of nonuniqueness in the set \( f^{-1}(z) \). One approach we refer to as the constraint method. The other approach, the one which is implemented in this algorithm, we call the probability method. Each approach is discussed below.
3.2.1 Constraint Method

As discussed above, the inverse image of the element $z \in Z$ contains multiple elements of the space $Y$. The solution of the inverse problem requires obtaining a single element of this set. The idea behind the constraint method involves putting further restraints on this set, thus forcing the process to produce a unique solution. These restraints require assumptions about the spaces $Y$ and $Z$.

The assumptions made with the constraint method may or may not be reasonable assumptions. These assumptions can be thought of as assigning a penalty function $h$ on the set $f^{-1}(z)$. This method, therefore, amounts to finding an element $\hat{y}$ from $f^{-1}(z)$ which minimizes the penalty function $h$. The solution $\hat{y}$ found with this method depends upon the assumptions made on the system, whether they were reasonable assumptions or poor ones. Also, this method does not necessarily account for the noise contained in the data set.

3.2.2 Probability Method

A second method in dealing with the nonuniqueness of $f^{-1}(z)$ takes a probability-based approach. The element $z$ in our application represents an observation of pressure measurements taken from wells. As stated previously, this response realistically has noise, or in other words, an error inherent in the data set. This means that the set $f^{-1}(z)$ also inherits this error. When applied to the differential equation, the actual configuration may not reproduce identically the recorded data set. To account for this error, this probabilistic approach forces a conditional probability distribution on the space $Y$ conditioned on $z$.

In our application, we condition the space of all possible configurations generated by the
probability model discussed in Chapter 2, based on the observed pressure measurements. The solution to this inverse problem results from sampling from this conditional probability distribution.

This method has advantages over the constraint method. One major advantage of this approach is that through sampling from this distribution the error included in the observed data set is accounted for. The effect due to the noise can be analyzed by examining the sample. It also gives more certainty to the resulting solution since it is based on several possible solutions rather than on restrictive assumptions placed on the inverse model. In other words, this method quantifies the uncertainty in solving the inverse problem. The remainder of this chapter discusses how this concept of sampling from the conditional probability distribution is incorporated into the algorithm.

3.3 The Inverse Model

Models to solve the inverse problem can take many different forms. This section outlines the steps of our inverse model. We first limit the configurations considered as possible solutions. Those we consider are the configurations which fit the geologic conditions of the reservoir resulting from the probability model discussed in Chapter 2. We then condition this probability model on the pressure measurements obtained. We obtain possible solutions to the problem by sampling from this conditional probability distribution of the possible configurations.

The process of sampling from this distribution is based on the uniform random number list $X$ discussed in Chapter 2, and on the Simulated Annealing Algorithm. Recall that the list $X$ represents a configuration based on the probability rules associated with
the reservoir's creation. This list \( X \) is a "coded" representation of the configuration. To sample from the conditional distribution, it is necessary to construct a function \( E \) which can compare the response expected from a given configuration to the actual data set. The Simulated Annealing Algorithm uses this function, as described in the following section. This function, in a sense, assigns an energy (or distance) for each configuration representing how well the response of each configuration matches the actual data response. Since the configurations are represented by uniform random number lists, the function \( E \) assigns an energy for each random number list. Our objective in sampling from the conditional distribution can be accomplished by minimizing this energy function \( E \). By using Simulated Annealing to minimize this function, we are in fact sampling from the conditional distribution. This method is called the Conditional Coding Method. The following sections cover in greater detail the Annealing Algorithm and the Conditional Coding Method.

3.4 The Simulated Annealing Algorithm

The method used in this algorithm to sample from the conditional distribution of the space \( Y \) of possible reservoir configurations is Simulated Annealing. The Annealing Algorithm is a global-minimization technique used in many applications. The fundamental concepts involved with Simulated Annealing are given below. A brief discussion of its mathematical justification also follows.

3.4.1 Fundamental Concepts of Simulated Annealing

The goal of Simulated Annealing is to reach a global minimum of some function. An analogy which serves well to describe the goal and process of the Annealing Algorithm
comes from thermodynamics [8]. When liquids or metals cool slowly, the state of the substances reaches a global-minimum energy. Liquids form crystals, and metals anneal. If cooled too rapidly, the state which is reached will have a higher energy than the slowly cooled state. The crystallized state of liquids, for example, is a state of minimum energy. The Boltzmann probability distribution describes the distribution of the energy in this type of system,

\[ P(E) \propto \exp \left\{ -\frac{E}{kT} \right\} \]

where \( E \) is the energy, \( T \) is the temperature, and \( k \) is Boltzmann’s Constant. A similar distribution is used in the Simulated Annealing Algorithm.

Simulated Annealing incorporates the Metropolis Algorithm. The Metropolis Algorithm is an iterative process which allows some iterates to be accepted and others to be rejected. The probability rule which determines whether an iterate \( X_i \) is to be kept or rejected is given by

\[ P(\text{keep } X_i) = \exp \left\{ \frac{\min\{0, -(E_i - E_{\text{cur}})\}}{T} \right\} \]

where \( E_i \) represents the “energy” of \( X_i \), \( E_{\text{cur}} \) is the energy attained from the current iterate, and \( T \) again represents the “temperature.” This energy value is the function \( E \) discussed previously which assigns a value representing the closeness of a configuration \( X \) to the exact structure. This assignment comes by comparing the observed head response to the expected head response of \( X \), using the rule

\[ E(X) = \sum_{i=1}^{m} (\text{obs}_{t_i} - \text{resp}_{t_i})^2 \]

where \( \text{obs}_{t_i} \) represents the observed data and \( \text{resp}_{t_i} \) represents the expected response from \( X \), at time \( t_i \). The \( t_i \)'s correspond to the \( m \) measurement times in the observed data set.
The temperature $T$ is used to control the rate of convergence of the Metropolis Algorithm.

Note that if $E_i \leq E_{\text{cur}}$, then the probability to keep $X_i$ equals 1, and thus a decrease in energy is always accepted. If $E_i > E_{\text{cur}}$, $X_i$ may still be accepted according to this probability distribution, which is dependent upon $T$. This can be interpreted to mean that the sequence of $X_i$'s should not get trapped in a local minimum. By keeping some iterates which have an increase in energy, this process allows the sequence to step out of a local minimum, thus ideally allowing a global minimum to be obtained. See [8] for more details.

### 3.4.2 Justification for the Metropolis Algorithm

As described above, the Metropolis Algorithm is an iterative method, and creates a sequence of some of the iterative values. The algorithm dictates how iterates are formed and which iterates are added to the sequence. Given an iterate $X_i$, a small random perturbation of $X_i$ creates a new value $X^*_i$. Based on the probability rule in Equation 3.2, this value may be accepted or rejected. If $X^*_i$ is accepted, then it becomes the next iterate $X_{i+1}$ and the process is repeated by making a random perturbation of $X_{i+1}$. If $X^*_i$ is rejected, then the process is repeated with a random perturbation of $X_i$ again until the perturbed member is accepted. Thus, this creates a sequence of values $X_0, X_1, X_2, \ldots$. This sequence is a Markov Chain.

### 3.4.3 Markov Chains

A sequence is a Markov Chain if the following relationship holds:

$$P(X_{n+1} = i_{n+1} \mid X_n = i_n, \ldots, X_0 = i_0) = P(X_{n+1} = i_{n+1} \mid X_n = i_n).$$
In other words, the probability of obtaining $X_{n+1}$ depends solely on the previous value $X_n$ and not upon any of the other preceding values. Thus, the $(n + 1)^{th}$ iterate depends only upon the $n^{th}$ iterate, and not upon how the $n^{th}$ iterate itself was obtained. The sequence generated by the Annealing Algorithm fits this definition.

A Markov Chain which is nonperiodic recurrent will converge to a stationary distribution [4]. The sequence obtained from the Metropolis Algorithm meets this condition, and thus the sequence generated by this method converges to a stationary distribution which is given by Equation 3.1. The details of this property are not included in this thesis, but are explained more thoroughly in [2] and [7].

3.4.4 The Annealing Schedule

The probability rule for the iterations given by Equation 3.2 depends not only on the configuration, but also on the temperature $T$. A change in $T$ results in a different probability rule for keeping iterates. The Annealing Algorithm incorporates the Metropolis Algorithm along with a so-called annealing schedule, which defines a rule for changing the value of $T$. A common schedule involves gradually reducing the temperature $T$. By lowering the temperature, the sequence converges to distributions with a low value of $T$. This is due to the probability rule in Equation 3.2. As $T$ decreases, the probability of accepting an iterate with an increased energy state also decreases. Thus the parameter $T$ gives some control over the convergence rate. However, the analogy stated earlier offers some insight into choosing an appropriate annealing schedule. If $T$ is lowered too quickly, or in other words, if the system is cooled too rapidly, the system may not converge to a global-minimum energy state. It may get trapped in a local-minimum state of energy.
Incorporating this slow-cooling concept increases the number of iterations required to run the Annealing Algorithm. The annealing schedules which were selected in the Annealing Algorithm in this application, as well as the results obtained with each schedule, are discussed in Chapter 5.

3.5 The Conditional Coding Method

The heart of the algorithm presented in this thesis is the Conditional Coding Method. It is this concept which allows and justifies the process of sampling from the conditional probability distribution. Many of the key factors in this method have appeared in previous sections and chapters of this thesis. In this section some of these concepts are restated in a more probabilistic sense, and the equations justifying these concepts are also presented.

As stated in the beginning of this chapter, there is a correspondence linking the configuration of a reservoir and the observed pressure data obtained from well tests. The observed data set, which will be referred to as $obs$, consists of the exact response plus noise, i.e.,

$$\text{(3.5)} \quad obs = \text{response} + \text{noise}.$$  

Given a configuration $C$ for a reservoir, we can calculate the probability of the observation given a configuration based on a normal distribution for noise,

$$\text{(3.6)} \quad P(\text{obs} \mid C) = \exp \left\{ -\frac{E(\text{obs}, C)}{2\sigma^2} \right\}$$

where $E$ represents how well the response from $C$ matches $obs$,

$$\text{(3.7)} \quad E(\text{obs}, C) = \sum_{i=1}^{m} (\text{obs}_{t_i} - \text{resp}_{t_i})^2$$

(where $\text{obs}_{t_i}$ is the observed pressure at time $t_i$ and $\text{resp}_{t_i}$ is the calculated response from $C$ at time $t_i$). This function $E$ is equivalent to the energy function $E$ used in the Simulated
Annealing Algorithm. Thus, by letting the temperature $T = 2\sigma^2$ and implementing Simulated Annealing, we are in fact sampling from this conditional probability distribution.

However, the goal of this model is to find the configuration given the observation, i.e., to find $P(C \mid obs)$. Since we are in fact using a uniform random number list $X$ to represent the configuration, we want to find $P(X \mid obs)$.

From the conditional probability definition

$$
(3.8) \quad P(A \mid B) = \frac{P(A \cap B)}{P(B)}
$$

we have that

$$
(3.9) \quad P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)}.
$$

This is the simple result which provides the means to reach the objective,

$$
(3.10) \quad P(X \mid obs) = \frac{P(obs \mid X)P(X)}{P(obs)}.
$$

Since $P(obs)$ is a constant, this term is noncritical. Also since $X$ has a uniform distribution, then $P(X)$ is constant as well. Therefore, if $X$ has a uniform distribution,

$$
(3.11) \quad P(X \mid obs) \propto P(obs \mid X).
$$

Thus, by sampling from the conditional probability distribution of Equation 3.6 by using Simulated Annealing, we are in fact sampling from the conditional distribution on the configuration space. Sampling from this distribution solves the problem of approximating the geometry of the system. This justifies the steps of the algorithm presented in this thesis.
CHAPTER 4
FLOW EQUATION AND NUMERICAL METHODS

The Metropolis Algorithm described in Chapter 3 requires modeling flow through a configuration representing a porous medium. That is to say, it requires solving the forward problem. This is critical to analyze how well the configuration approximates the actual structure. This chapter addresses the method applied to calculate this flow. The equation which models flow through porous media is presented and discussed, and the numerical methods which solve this equation are also described.

4.1 Equation Modeling Flow

Chapter 3 introduced the concept of a relationship, or function $f$, between the configuration of a system and the flow (i.e., head) measurements expected from that system. This relationship between the geometry and flow is clearly necessary to solve Problem $P1^*$. This section outlines the equations governing flow through a porous medium.

Certainly the geometry of a configuration is not the only factor involved in determining flow. Properties such as the permeability and storativity of the medium, the compressibility and concentration of the fluid, and the rate of fluid flowing through the system all contribute to the flow. Here we assume single-phase flow of an incompressible fluid with constant flux through a porous medium. In terms of the fractured rock network generated by the algorithm, the configuration can be thought of as a network of pipes filled with some type of porous medium such as sand, with liquid flowing through the system. We wish to model the flow of the liquid through this system. We essentially assume two
dimensional flow. However, in the next section an explanation is given which allows us to reduce this to a one dimensional model. By accounting for these factors and by satisfying conservation of mass, a differential equation describing flow can be obtained.

An equation relating the properties of the flow through a porous medium is obtained by considering a small unit or box within the system. The net flow into and out of this element must satisfy the conservation of mass; that is to say,

\[(4.1) \quad \text{[rate in]} - \text{[rate out]} + \text{[rate injected]} = \text{[rate of accumulation]}.\]

These rates are calculated by examining all faces of the box and its volume. Then, by taking the limit as the dimensions of the box decrease, the following partial differential equation is formed:

\[(4.2) \quad \nabla \cdot \underbrace{K \nabla h}_{\text{source/sink term}} + q = S \frac{\partial h}{\partial t},\]

over some domain $\Omega$ and for $t > 0$. In this equation, $q$ represents the source/sink term (i.e., fluid injection and production), $S$ is the storativity, $K$ is an array representing permeability, $h$ is the hydraulic head, $t$ is the time variable, and $\nabla$ represents the gradient operator $< \frac{\partial}{\partial x}, \frac{\partial}{\partial y} >$. See Peaceman [6] for details of this development.

The solution of Equation 4.2 requires conditions which specify values at all points of the closed boundary $\partial \Omega$ (closure of $\partial \Omega$). Thus, initial conditions at $t = 0$ and boundary conditions are necessary to uniquely solve the system for the head $h$. We assume a Dirichlet boundary condition $h(x, t) |_{\partial \Omega} = f(x, t)$ over the boundary $\partial \Omega$ for $t > 0$. The initial condition is $h(x, 0) = g(x)$ at $t = 0$. This gives values over all of $\partial \Omega$. The variable $x$ represents the spatial variable on $\Omega$.

The source/sink term $q$ represents fluid injected into and produced out of the system.
For our system, we assume there is no flow across the boundary of the system, so $q$ represents the well sources within the configuration. As a function of location on the domain, $q$ has a value of zero everywhere except at some well locations. Thus, the value of $q$ at some location $x$ can be calculated with a scaled Dirac delta function to account for this “zero-almost-everywhere” behavior.

$$q(x) = \sum_{j=1}^{n} Q_j \delta(x).$$

In this equation $Q_j$ is the strength of the $j^{th}$ well, and $\delta(x)$ is the Dirac delta function at $x \in \Omega$. The value $n$ represents the total number of wells in the system.

Given a configuration and associated permeability and storativity values, we can now use Equation 4.2 to find the head values over $\Omega$, and thus at the wells. The permeability and storativity values are assigned for configuration by shifting the results of the numerical simulation both vertically and horizontally to minimize the distance between the calculated values and the observed values of head in the log(t)-log(h) domain. The vertical shift corresponds to the permeability assignment, and the horizontal shift assigns the storativity. In terms of Equation 4.2, this corresponds to multiplying by a constant.

### 4.2 Numerical Methods

With the model equation discussed in the previous section, we can now address the method used to calculate flow (i.e. head values) through a configuration. We use a program called TRINET, written by Kenzi Karasaki at Lawrence Berkeley Laboratory, which computes this flow associated with a configuration. The mathematics involved with each step will be briefly explained, along with how it acts specifically upon Equation 4.2. For a more detailed discussion of TRINET, see [3].
Karasaki [3] points out that the two dimensional model for fractured rock systems may in this case be better represented by using a one dimensional model. To model fractured networks, the fractures are often thought of as representations of parallel plates with smooth walls. Fractures in rocks, however, generally do not fit this representation due to roughness of the surface. This roughness can cause dominant paths of fluid flow through the network. Thus, by treating the fractures as porous line segments rather than pairs of smooth parallel plates and by considering a one dimensional network of flow through a porous medium, the dominant flow path can be modeled. Our model uses this line segment representation of the fractures for a one dimensional model equation.

In one dimension the equation modeling flow (Equation 4.2) can be rewritten as

\[
\frac{\partial}{\partial x} K \frac{\partial h}{\partial x} + q = S \frac{\partial h}{\partial t} \tag{4.4}
\]

where \( K \) is now a one dimensional parameter for permeability, and \( x \) is now the one dimensional spatial variable. Our goal is to find the function \( h(x,t) \) which solves this differential equation.

The TRINET code, which in this application considers only transient flow without the advection or dispersion effects, solves Equation 4.4 on each element of the grid using a finite element method in \( x \) and a finite difference time stepping scheme. Both of these methods are described below.

### 4.2.1 Galerkin Finite Element Method

To explain this finite element method, a simplified form of Equation 4.4 will be used. Consider the second-order partial differential equation

\[
\frac{\partial}{\partial x} K \frac{\partial h}{\partial x} = \frac{\partial h}{\partial t}. \tag{4.5}
\]
The classical solution to this problem is the function \( h \) which satisfies this differential equation. A weak form of Equation 4.5 will be solved instead.

Before proceeding, it is important to define some function spaces which will be used:

\[
L^2(\Omega) = \{ h \mid \int_{\Omega} h^2 d\Omega < \infty \},
\]

\[
H^1(\Omega) = \{ h \in L^2(\Omega) \mid \nabla h \in (L^2(\Omega))^n \},
\]

\[
H_0^1(\Omega) = \{ h \in H^1(\Omega) \mid h = 0 \text{ on } \partial \Omega \}.
\]

Here \( n \) represents the dimension of \( \Omega \). To get the weak form of Equation 4.5, we multiply the equation by a “test function” \( \varphi \in H_0^1(\Omega) \) and integrate over the domain \( \Omega \).

\[
\int_{\Omega} \left( \frac{\partial}{\partial x} K \frac{\partial h}{\partial x} \right) \varphi d\Omega = \int_{\Omega} \frac{\partial h}{\partial t} \varphi d\Omega.
\]

Using Green’s Theorem on the left-hand-side produces the equation

\[
- \int_{\Omega} K \frac{\partial h}{\partial x} \frac{\partial \varphi}{\partial x} d\Omega + \int_{\Gamma} \varphi (K \frac{\partial h}{\partial x}) \cdot \vec{n} d\Gamma = \int_{\Omega} \frac{\partial h}{\partial t} \varphi d\Omega
\]

where \( \Gamma = \partial \Omega \) and \( \vec{n} \) is the outward normal.

Since we picked \( \varphi \in H_0^1(\Omega) \), then \( \int_{\Gamma} \varphi (K \frac{\partial h}{\partial x}) \cdot \vec{n} d\Gamma = 0 \) and Equation 4.10 becomes

\[
- \int_{\Omega} K \frac{\partial h}{\partial x} \frac{\partial \varphi}{\partial x} d\Omega = \int_{\Omega} \frac{\partial h}{\partial t} \varphi d\Omega,
\]

which is the weak form of Equation 4.5.

The solution to the weak form of the partial differential equation is to find \( h \), which satisfies Equation 4.11 for all \( \varphi \in H_0^1(\Omega) \).

Since \( H_0^1(\Omega) \) is infinite dimensional, we restrict our attention to some finite dimensional subspace \( V \in H_0^1(\Omega) \) where a solution to the weak form exists. The basic process of this method is to use a basis of the space \( V \), \{\( \varphi_1, \varphi_1, \varphi_2, \ldots, \varphi_N \)\}, and to find some \( \hat{h} \in V \) as
our approximate solution, where \( \hat{h} = \sum_{i=1}^{N} \alpha_i \varphi_i(x) \). This constitutes the Galerkin Finite Element method. Generally, the basis for the space \( V \) is chosen first, then \( V \) is determined by the span of these basis elements. In this form, our goal now is to find the coefficients \( \{\alpha_i\}_{i=1}^{N} \). By using properties of the function space, it can be shown that \( \hat{h} \in V \) exists and is the best approximation to \( h \) (in the \( L^2 \) norm) on \( V \).

Substituting \( \hat{h} \) into Equation 4.11,

\[
(4.12) \quad - \int_{\Omega} K \frac{\partial}{\partial x} \left( \sum_{i=1}^{N} \alpha_i \varphi_i \right) \frac{\partial \varphi_j}{\partial x} d\Omega = \int_{\Omega} \frac{\partial}{\partial t} \left( \sum_{i=1}^{N} \alpha_i \varphi_i \right) \varphi_j d\Omega,
\]

for \( j = 1, 2, \ldots, N \). This gives a system of \( N \) equations and \( N \) unknowns to solve. By choosing the basis elements wisely, the resulting matrix can be forced to have computationally nice properties, such as being tridiagonal or sparse.

For more details of finite element methods, see [1].

### 4.2.2 Finite Difference Method

The solution of the finite-element system defines \( \hat{h} \) over the spatial domain \( \Omega \). The code then uses this in a finite-difference time stepping scheme to define the solution over the region \( \Omega \times [0, T] \) over some finite time interval \([0, T]\). The code implements the discretization of Equation 4.4 (in a simplified form)

\[
(4.13) \quad [K]^t \cdot (\theta \{h\}^{t+\Delta t} + (1 - \theta)\{h\}^t) = [S]^t \cdot \left( \frac{\{h\}^{t+\Delta t} - \{h\}^t}{\Delta t} \right)
\]

where \([K]^t\) is the permeability matrix, \([S]^t\) is the storage coefficient matrix, and \(\{h\}^t\) is the head vector at time \( t \). \( \theta \) is a user-controlled parameter dictating the weight of each of the implicit and explicit permeability terms. The value \( \theta = 0.66 \) was used in this application, as recommended by Karasaki.
CHAPTER 5
NUMERICAL RESULTS

The algorithm described in this thesis was tested on three different data sets. Two of these sets contained simulated data, and the other was a real data set. This chapter presents the results and observations of implementing the algorithm on each of these three data sets.

5.1 Data Set 1

The algorithm was first tested on a coarse grid of 50 elements x 50 elements. A configuration was selected as the “actual” structure, and flow from injection at the center well was calculated on this structure using the numerical flow model. Figures 5.1 and 5.2 show this simulated pattern and its pressure curves for each well. Note that the pressure curves from this simulation have noise which one would expect to see in an observed data set. This noise was added using a normal distribution to the flow calculated from the numerical solver discussed in Chapter 4. The center well (well 3) was the pumping well. The algorithm was started using a random configuration (i.e., an arbitrary uniform random number list) for its initial iteration value, and was allowed to run, comparing the flow of each configuration to the simulated flow test. The algorithm ends when an acceptably small energy level is obtained. This process of running the algorithm was repeated several times, each time starting either a different initial random number list, or making different perturbations to the same initial random number list, or using a different annealing schedule. With each run, the same parameters which control the probability...
Simulated Data Set 1

Figure 5.1: Simulated fracture pattern for a 50x50 grid. This pattern represents the actual structure for all tests run on this grid.
Figure 5.2: Pressure curves for each well, obtained by calculating flow on the simulated fracture pattern on the 50x50 element grid, with noise added to simulated actual observations.
model were used (see Section 2.2). The annealing schedule used for each run started at an initial temperature of 50, and decreased the current temperature by multiplying by a temperature decrease factor $d$ each time an iterate was accepted within the Annealing Algorithm. Figures 5.3 through 5.26 show the results of each of these runs, displaying the final configuration, its associated pressure curves, the energy profile of the Annealing Algorithm, and the temperature decrease factor $d$ for each run. On the pressure curves, the dot-marked curve represents the calculated flow.

5.1.1 Observations

Only the last three runs included noise in the “observed” head measurements. The first five used the exact head measurements from the numerical flow solver. In terms of matching the head measurements, each of these results gives a good approximation of the observed data set. The energy profiles show the effect of the Metropolis Algorithm, allowing the energy to step out of local minima, thus allowing lower values to be obtained. The energy profiles also exhibit the convergence of the algorithm, as well as the effect of the probability rule 3.2 where decreases in temperature allow fewer increases in energy to be kept. Note that several thousand iterations were performed before the algorithm terminated the process. This corresponds to the slow-cooling analogy stated previously.

The paths from each well to the pumping well match the characteristics found in the simulated configuration. Each run created configurations with short, direct paths from wells 2 and 4 to the pumping well. Well 1 is unconnected from the pumping well in the simulated configuration. Fluid injected at the pumping well must take a meandering path to show response at well 1. This characteristic appears in the approximations as well.
Figure 5.3: First sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.995$. 

Run 1.1
Run 1.1

Figure 5.4: Energy profile from the first run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.995$. 
Run 1.1

Figure 5.5: Pressure curves for each well, obtained by calculating flow on the first fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1.
Figure 5.6: Second sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.995$. 
Run 1.2

Figure 5.7: Energy profile from the second run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.995$. 
Run 1.2

Figure 5.8: Pressure curves for each well, obtained by calculating flow on the second fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1.
Figure 5.9: Third sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.995$. 

Run 1.3
Run 1.3

Figure 5.10: Energy profile from the third run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was \( d = 0.995 \).
Run 1.3

Figure 5.11: Pressure curves for each well, obtained by calculating flow on the third fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1.
Figure 5.12: Fourth sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.99575$. 

Run 1.4
Run 1.4

Figure 5.13: Energy profile from the fourth run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.99575$. 
Run 1.4

Figure 5.14: Pressure curves for each well, obtained by calculating flow on the fourth fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1.
Figure 5.15: Fifth sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. 

Run 1.5
Run 1.5

Figure 5.16: Energy profile from the fifth run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. 
Figure 5.17: Pressure curves for each well, obtained by calculating flow on the fifth fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1.
Figure 5.18: Sixth sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.998$. 

Run 1.6
Run 1.6

Figure 5.19: Energy profile from the sixth run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. 
Run 1.6

Figure 5.20: Pressure curves for each well, obtained by calculating flow on the sixth fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1.
Figure 5.21: Seventh sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. 

Run 1.7
Run 1.7

Figure 5.22: Energy profile from the seventh run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. 
Run 1.7

Figure 5.23: Pressure curves for each well, obtained by calculating flow on the seventh fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1.
Figure 5.24: Eighth sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. 
Run 1.8

Figure 5.25: Energy profile from the eighth run of the Conditional Coding Method on Simulated Data Set 1. The temperature decrease factor was $d=0.9975$. 
Run 1.8

Figure 5.26: Pressure curves for each well, obtained by calculating flow on the eighth fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 1.
Each run except the sixth forced this disconnection between wells 1 and 3. The same result is true of well 5. It is more connected to the pumping well than is well 1, but still is not a direct path. The approximations inherited this quality as well. The distance of paths from the pumping well to observation wells can be adjusted by a storativity value correction.

The approximations where noise was included in the observation set produced more fluid paths and more direct channels than the actual pattern exhibited. This demonstrates the need to account for the effect of noise in the data set.

The energy profiles display another point of interest. Determining an acceptable minimum energy value to terminate the algorithm depends upon the observed data set. The runs in which no noise was present in the observed head measurements achieved reasonable matches to this set with energy values on the order of 1. However, with noise present the algorithm produced reasonable matches and convergence with energy values on the order of 100. The number of data points in the observation as well as the amount of noise present affects the acceptable minimum energy value.

5.2 Data Set 2

Next, we refined the fracture grid to the size of 200 elements \( \times \) 200 elements. Clearly, this grid required much more computer memory and time to store and manipulate the increased number of elements. The first experiment with this grid was also on a simulated data set. Figures 5.27 and 5.28 show the simulated configuration and its pressure curves. Because of the finer grid scale, this grid produces more realistic-looking fracture patterns. This simulated set is also less dense than the previous simulation, which more closely
Simulated Data Set 2

Figure 5.27: Simulated fracture pattern for a 200x200 grid. This pattern represents the actual structure for all tests run on this grid.
Simulated Data Set 2

Figure 5.28: Pressure curves for each well, obtained by calculating flow on the simulated fracture pattern on the 200x200 element grid, with noise added to simulated actual observations.
resembles actual fractured rock systems at the Conoco test site. The parameters used to create this simulated set were used in all runs on this set. The parameters were chosen in hopes to approximate the fracture system from the Conoco site discussed in Chapter 2. Once again, well 3 in the center of the grid was used as the pumping well.

Three separate runs were done on this simulated data set, with the results displayed in Figures 5.29 through 5.37.

Only the third run compared flow against the noise-included pressure set. The other two compared to the flow found from the differential equation. With this refined grid, since the initial energy was considerably higher for the first configurations, we experimented with both the temperature decrease factor and the initial temperature values.

5.2.1 Observations

Once again, the pressure curves indicate reasonable matches to the observed data set. Fewer iterations were performed on these runs before stopping the algorithm. Computation time was a factor in this decision.

The results shown in Figures 5.29 through 5.31 began with an initial temperature of 50, while the others started with a temperature of 500.

The energy profile from Figure 5.30, where the initial temperature was 50, indicates rapid cooling. This caused decreases in energy without the ability to jump out of the local-minimum energy values. Figure 5.36 indicates that the algorithm had a high temperature at the time of termination, and had not reached a stable state. As stated above, computation time was a motivating factor in terminating the algorithm.

The relationship between the pumping well and the observation wells in the actual
Figure 5.29: First sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. 
Run 2.1

Figure 5.30: Energy profile from the first run of the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. 
Run 2.1

Figure 5.31: Pressure curves for each well, obtained by calculating flow on the first fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 2.
Run 2.2

Figure 5.32: Second sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. 
Run 2.2

Figure 5.33: Energy profile from the second run of the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. 
Figure 5.34: Pressure curves for each well, obtained by calculating flow on the second fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 2.
Figure 5.35: Third sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. 
Run 2.3

Figure 5.36: Energy profile from the third run of the Conditional Coding Method on Simulated Data Set 2. The temperature decrease factor was $d=0.995$. 
Figure 5.37: Pressure curves for each well, obtained by calculating flow on the third fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 2.
configuration again appeared in the approximations. Wells 2 and 4 are more directly connected to the pumping well, while wells 1 and 5 take a less connected path.

5.3 Data Set 3 - Real Data

After testing the algorithm on the simulated sets, we incorporated the actual data obtained from the Conoco site in Oklahoma. Figure 2.1 shows the map view of the site, and Figure 5.38 shows the observed pressure measurements taken there. The pumping well test was not considered for these applications. We again used the 200×200 grid to
create configurations, and the parameters used in generating configurations from data set
2 were used in this run as well. There are five wells at this site, but well 2 was used as
the pumping well for these measurements.

Two runs were made with this data set. Each started with an initial temperature of
500 and a temperature decrease factor $d = 0.9975$. Figures 5.39 through 5.44 show the
configurations, energy profiles, and pressure curves for these runs.

5.3.1 Observations

As seen from the results, these processes were not run to completion, and did not give
pressure matches as closely as the other data sets. This may in part be due to the choice of
parameters used to build configurations. Errors in representing the site numerically may
also contribute to the difference. For instance, the choice of well locations and orientation
on the computer grid may not reasonably represent the site. These details require further
research.
Run 3.1

Figure 5.39: First sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 3. The temperature decrease factor was $d=0.9975$. 
Run 3.1

Figure 5.40: Energy profile from the first run of the Conditional Coding Method on Simulated Data Set 3. The temperature decrease factor was $d=0.9975$. 
Figure 5.41: Pressure curves for each well, obtained by calculating flow on the first fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 3.
Run 3.2

Figure 5.42: Second sample of a fracture pattern approximation, generated by the Conditional Coding Method on Simulated Data Set 3. The temperature decrease factor was $d=0.9975$. 
Run 3.2

Figure 5.43: Energy profile from the second run of the Conditional Coding Method on Simulated Data Set 3. The temperature decrease factor was $d=0.9975$. 
Run 3.2

Figure 5.44: Pressure curves for each well, obtained by calculating flow on the second fracture pattern approximation which was generated by the Conditional Coding Method on Simulated Data Set 3.
The testing of this Conditional Coding Method thus far gives promising results. The method accounts for both the nonuniqueness of the solution due to the limited amount of information as well as the noise in the observed data set. By representing iterates with a uniform random number list in the Simulated Annealing Algorithm, we can justify the convergence of the process. By sampling from the approximations produced from the algorithm, information about the actual structure can be obtained.

The Conditional Coding Method can be extended beyond this orthogonal-fracture application. The program can be modified to allow consideration of nonorthogonal fractures, and three-dimensional systems. As stated at the beginning of this thesis, this algorithm can also be applied to many other scientific applications requiring an inverse method. This makes it a very powerful tool.

There remain unanswered questions with this algorithm. Presently, the parameters used to generate configurations are chosen in a trial-and-error fashion. It is not certain that the chosen parameters such as the number of starting fractures and growth iterations will produce configurations similar to the actual structure. The choices of initial temperature and the annealing schedule also affect the outcome. Further study of these variables could greatly enhance the results of the algorithm.
REFERENCES


