MODELING GROUNDWATER FLOW IN THE VICINITY OF DIP-SLIP FAULTS

by

ROBERT A. PINE

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

New Mexico Institute of Mining and Technology

FALL 1995

ABSTRACT

This thesis details a study on the effects that a single idealized dip-slip fault can have on groundwater flow. Various geometric and hydrogeologic fault parameters are considered. These are: fault displacement, reverse drag, fault dip, permeability distribution along the fault and enhancement of permeability parallel to the fault plane within the fault zone. Consideration is given to the ability to detect these effects with sparse well data and to the possible misinterpretations of sparse well data in the vicinity of a fault.

MODFLOW was used to construct a three-dimensional finite-difference computer model simulating steady-state groundwater flow in a region containing an idealized dip-slip fault. A pre-processor was written to generate the model input files that define the region with the imbedded fault. A post-processor was written to slice the resulting head file in any plane parallel to a coordinate direction; these cross sections were then contoured using SURFER. Various statistics were also generated for each run by the post-processor to enable comparison of each set of fault parameters and so to measure the sensitivity of flow patterns to each fault parameter. Hypothetical observation wells were randomly placed in the aquifer and the resulting sparse well data was contoured and gradients were computed.

Simulation results indicate that the most important fault characteristics are fault displacement and permeability distribution within the fault zone. Vertical gradients are sensitive to fault dip. The various summary statistics computed are not very sensitive to the width of reverse drag.

Actual ground water gradients and flow patterns in the vicinity of a fault may be somewhat obscured by attempting to determine the potentiometric surface from a limited number of observation wells. A blind fault and its effects could easily escape detection due to a placement of wells that is inadequate in location and number. Additionally, an incorrect determination of regional ground water gradients can be made.

ACKNOWLEDGEMENTS

I would first like to thank my main advisor, Dr. John L. Wilson for all the time, advice and direction given me; he helped me keep this project focused. I would also like to thank him for helping to secure funding for the project.

For funding, moral support and a great deal of advice and guidance, I must thank another advisor, Dr. Jim McCord of Sandia National Labs. This project certainly wouldn't have happened without him. Of course, extreme gratitude to Sandia National Labs for funding this project.

Thanks also to my other committee members, Dr. Fred Phillips and Dr. Peter Mozley of New Mexico Tech for their time and suggestions.

I would like to express my gratitude to my wife, Barbara, for enduring all the I time spent in front of the computer.

My deep thanks to Dr. Dave Johnson, Chairman of the Department of Geoscience at New Mexico Tech, for working to ensure the completion of this thesis.

And finally, thanks to Tom Christensen for assistance with some of the figures.

TABLE of CONTENTS

LIST OF FIGURES

PAGE

LIST OF SYMBOLS

D	Displacement of a point at a given X and Y value
D_{max}	Maximum displacement along the fault plane
$GRAD_5$	Minimum gradient magnitude greater than 5% of computed gradient magnitudes over a well field
$GRAD_{95}$	Minimum gradient magnitude greater than 95% of computed gradient magnitudes over a well field
$GRAD_{max}$	Maximum gradient magnitude over a set of triplets (X,Y,h_{int})
h	Head at a point
h_{int}	Head at a point vertically integrated over the aquifer
K_1	Hydraulic conductivity of material in -Y direction from fault plane within one MODFLOW cell in fault zone
K_2	Hydraulic conductivity of material in +Y direction from fault plane within one MODFLOW cell in fault zone
K_f	Hydraulic conductivity of fault zone
K_{min}	The minimum conductivity occurring for a given hydrogeologic layer. This value occurs in the model in the fault zone at a point of maximum displacement
Kx_{ef}	Effective hydraulic conductivity in X direction of a MODFLOW cell containing the fault zone
Ky_{ef}	Effective hydraulic conductivity in Y direction of a MODFLOW cell containing the fault zone
Kz_{ef}	Effective hydraulic conductivity in Z direction of a MODFLOW cell containing the fault zone
$\mathbf{K}_{ef}$	The effective conductivity tensor $(Kx_{ef},Ky_{ef},Kz_{ef})$
KF	Conductivity factor by which conductivity is increased in the X and Z directions for cells containing the fault zone
L	The length of the fault in X direction

r	normalized perpendicular distance from the line of maximum displacement of a point within the fault plane, i.e. $r = X/(L/2)$
W	Width of the fault, i.e. distance perpendicular to fault plane from the fault plane to tip-line ellipsoid
W_{max}	Maximum fault width
Y_f	Thickness of the fault zone
Z_{max}	Z value at the upper aquifer/aquitard contact at a point
Z_{min}	Z value at the lower aquifer/aquitard contact at a point

CHAPTER 1

INTRODUCTION

It is well known that geologic faults can have a substantial effect on groundwater flow, in some cases acting as conduits to flow, in other cases acting as barriers to flow. Observations of hydraulic head discrepancies across a fault zone of as much as 50 meters are not uncommon (Huntoon, 1981, 1985; Maclay and Small, 1983; Newcomb, 1969; Pine, 1991; Wallace and Morris, 1986; McCord et al., 1993). However, little has been done to systematically study these effects.

The purpose of this thesis is to address the following questions: (1) What is the response of groundwater flow patterns to changes in various hydraulic and geometric fault properties? (2) How readily can groundwater flow patterns in the vicinity of a fault be observed through data from a limited number of wells? (3) To what degree could a limited amount of well data result in an incorrect interpretation of groundwater flow patterns in the vicinity of a fault?

The general approach chosen to address these questions was to create three-dimensional steady-state computer models of groundwater flow in the vicinity of idealized faults, varying the different fault properties with each simulation. The variable fault properties considered are:

1) amount of displacement on the fault;
2) extent of deformation perpendicular to the fault plane, i.e., reverse-drag profile;

3) variation of permeability along the fault plane in the direction perpendicular to slip;
4) enhancement of permeability in the fault zone in directions parallel to the fault plane;
5) fault dip.

In addition, various sets of observation wells are placed in the aquifer in the vicinity of the fault to see how the results of certain simulations might be "observed in the field".

A small amount of work has been done to model groundwater flow in the vicinity of faults. Townley and Wilson (1980) describe a technique for representing a no-flow fault in their 2-D finite element code, AQUIFEM. Forster and Smith (1988) represented a high-angle fault as a planar zone of increased permeability in a cross-sectional finite element model. Forster and Evans (1991) used a somewhat more sophisticated representation for two-dimensional modeling of thrust faults by using more realistic and varying permeabilities in the fault zone. Hsieh and Freckleton (1992) wrote a module for the three-dimensional finite-difference code MODFLOW that can incorporate barriers to horizontal flow. This would enable one to simplistically model a fault in two dimensions. However, it would be inadequate and cumbersome for simulating a fault with truly three-dimensional properties.

To narrow the scope of the present study, it was decided to model only dip-slip faults. This was, in part, because a reasonable amount of information was found on the geometry of normal faults, a class of dip-slip fault. In particular, a single idealized vertical dip-slip fault is described and serves as the primary conceptual fault model, though some attention is given to subvertical normal faults.

Faults are inherently three-dimensional structures with characteristics that vary with depth, along the length of and in a direction normal to the fault plane. Therefore, it is reasonable to assume that a two-dimensional model might be inadequate to accurately study the effects on groundwater flow. Thus, the fault models for this study are three-dimensional.

CHAPTER 2

FAULTS and FAULT ZONES

Faults and fault systems can be geometrically and hydrogeologicly complex. In order to model groundwater flow in the vicinity of a fault, one must have at least a basic understanding of faults and the relationships between the different geometric and hydrogeologic characteristics of faults. This chapter will discuss these matters.

2.1 Fault Movement and Slip Surfaces

Faults are frequently categorized based on the relative direction of movement, or slip, i.e. motion of one side of the fault relative to the other. Faults for which motion is approximately parallel to the dip are known as *dip-slip faults*, whereas, when motion is approximately parallel to the strike of the fault, they are referred to as *strike-slip faults*. Inclined dip-slip faults (i.e. when the fault dip is less than 90°) are divided into two types: *reverse faults* where the *hanging wall* fault block (i.e. the upper fault block) moves up relative to the *footwall* (i.e. the lower fault block) and *normal faults* where the hanging wall moves down relative to the footwall (see Figure 2.1). Normal faults commonly occur in opposing pairs or in sets in a horst and graben configuration.

In the case of a simple fault system consisting of a single fault, the line of no deformation in a cross section

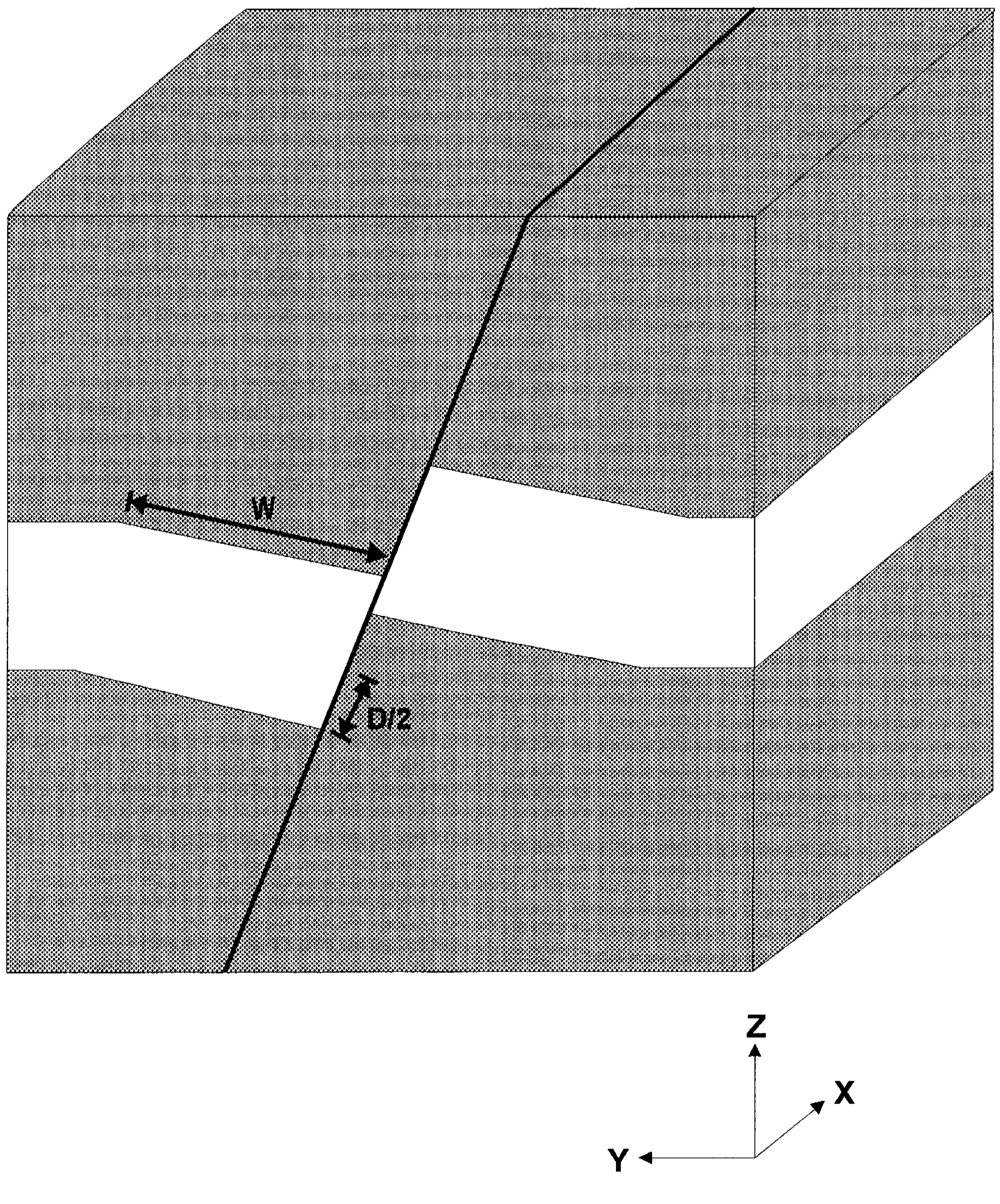


Figure 2.1 - Vertical cross section through modeled region showing hydrogeologic layers, fault and displacement, and coordinate system. As described in Section 2.3, D represents the fault displacement and W is the fault block width (the perpendicular distance from the fault plane to the tip line).

parallel or perpendicular to the direction of slip, known as a *tip line* (or tip-line loop), is often approximately elliptical (Twiss and Moores, 1992, pg. 70). The three-dimensional surface constructed of all such tip lines would then approximately form an ellipsoid, and is sometimes referred to as the tip-line ellipse.

The slip surface of a fault is generally not planer over the entire area of displacement. Normal and thrust faults often form a concave upward surface, the dip of the fault decreasing with depth. This is known as *listric* faulting and has various causes. It is common, however, for faults in vertical cross section to be approximately linear at shallow depths (depths typically of interest for hydrologic studies).

Faults may vary from the planar in the direction of strike as well. A fault plane may splay off into smaller *en échelon* faults. Motion on one fault plane may transfer over to another fault plane via a *step-over zone*, "an en échelon arrangement of offset faults" (Stewart and Hancock, 1991). Faults may generally trace a somewhat curved path due, in large part, to horizontal variations in lithology and rock strength as well as variations in stress. Despite the above, many faults can be reasonably approximated as planar.

2.2 Fault Zones

According to Twiss and Moores (1992), faults formed at depths less than about 10 to 15 kilometers typically have cataclastic rocks present. *Cataclasis* is a process by which grains are rotated and rock is fractured into clasts and granulated into powder. Below 10 to 15 kilometers, rock deformation is ductile, the process being referred to as *mylonitization* (at shallower depths, soft rocks can be deformed by mylonitization). The zone containing rock that is altered by fault movement is referred to as the *deformation zone* or *fault zone.* This zone may be characterized by the presence of cataclastic rock, numerous fractures and mylonitization. Within the fault zone one may find *gouge*, fine-grained rock flour, *breccia,* rock matrix with imbedded clasts of sizes ranging from less than a millimeter to over a meter, and areas of parent rock with an increased fracture density (Twiss and Moores, 1992).

The presence of fine-grained gouge, the presence of poorly sorted rock material and grain realignment can all cause a significant reduction in permeability across a fault zone. Pittman (1981) observed decreases in permeability of two to three orders of magnitude in fault zones within quartz sandstones. Morrow et al (1984) found that gouge permeabilities vary with clay content, average particle size, and with confining pressure. In their experiments, permeabilities of a variety of tested gouges, including non-clay gouges, tended to be quite low ranging from around 10^{-18} m^2 to 10^{-21} m^2 depending on the specific gouge and the confining pressure. Unfortunately, they give no indication of the permeabilities of the parent rock.

In addition to a fault zone forming a barrier to flow perpendicular to the fault plane, it is common for fracturing to take place in the deformation zone which could enhance permeability; with a decrease in permeability across the fault, the net result of fracturing may be an increase in permeability in directions parallel to the fault plane (Smith et al, 1990). Thus, permeability in fault zones can exhibit a large degree of anisotropy. Logan (1991) reports both field studies and laboratory experiments that demonstrate this phenomenon. The extent, geometry and permeability of the fracture network can change over time with changes in the applied stress and with cementation in and sealing of fractures (Bruhn and Yonkee, 1988; Mozley and Goodwin, 1995).

Wallace and Morris (1986) describe observations of various fault zones where they are intersected by underground mines. They report a great deal of variation in the distribution and thickness of the gouge, breccia and fractures within a single fault. They observed that the gouge may lie anywhere within the breccia; it may form "several discrete bands woven throughout the (breccia)" or form an anastomosing pattern within the fault zone.

The permeability across a fault may also be modified as a result of fault movement bringing into juxtaposition water-bearing strata of differing permeability (Maclay and Small, 1983; Huntoon, 1985).

2.3 Displacement

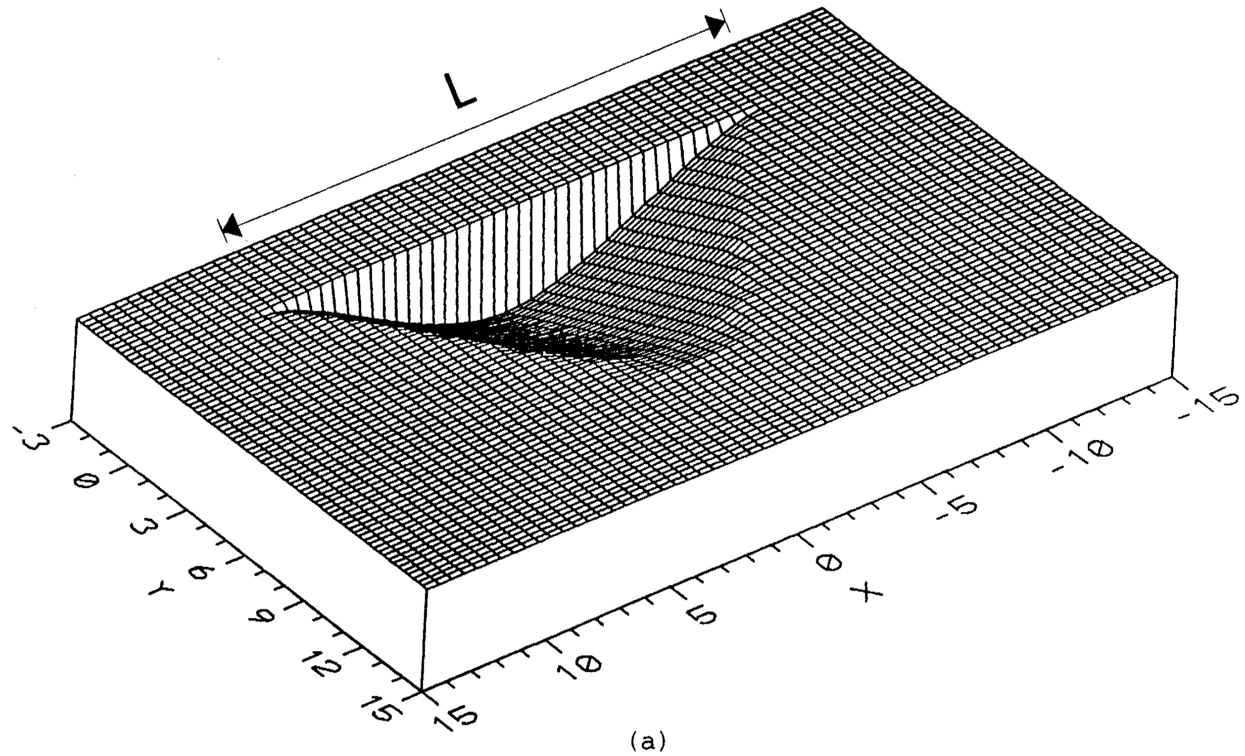
Measurements of the amount of displacement in various faults has shown a range of values from millimeters up to thousands of meters (Robertson, 1983; Hull, 1988). As all faults must terminate, displacement will vary along a single fault from 0 to the maximum for a particular fault.

Of interest to this study are relationships of displacement to other geometric and physical characteristics. Robertson (1983) and Hull (1988) examined data from several normal and reverse faults relating fault displacement (D) to the thickness (T) of the fault zone (where D and T are taken at any point in the fault zone). They found that for many faults and for a wide range of D values, the relationship can be approximated by $D = \beta T^{\alpha}$ where α is close to 1 and β ranged from 10 to 1000 (note that this equation is based on a regression analysis of D and T data from various faults and, therefore, α and β are unitless constants and one must take the units of D to be length even with α not equal to 1). The constants α and β vary from fault to fault and are functions of such factors as lithology and stress history.

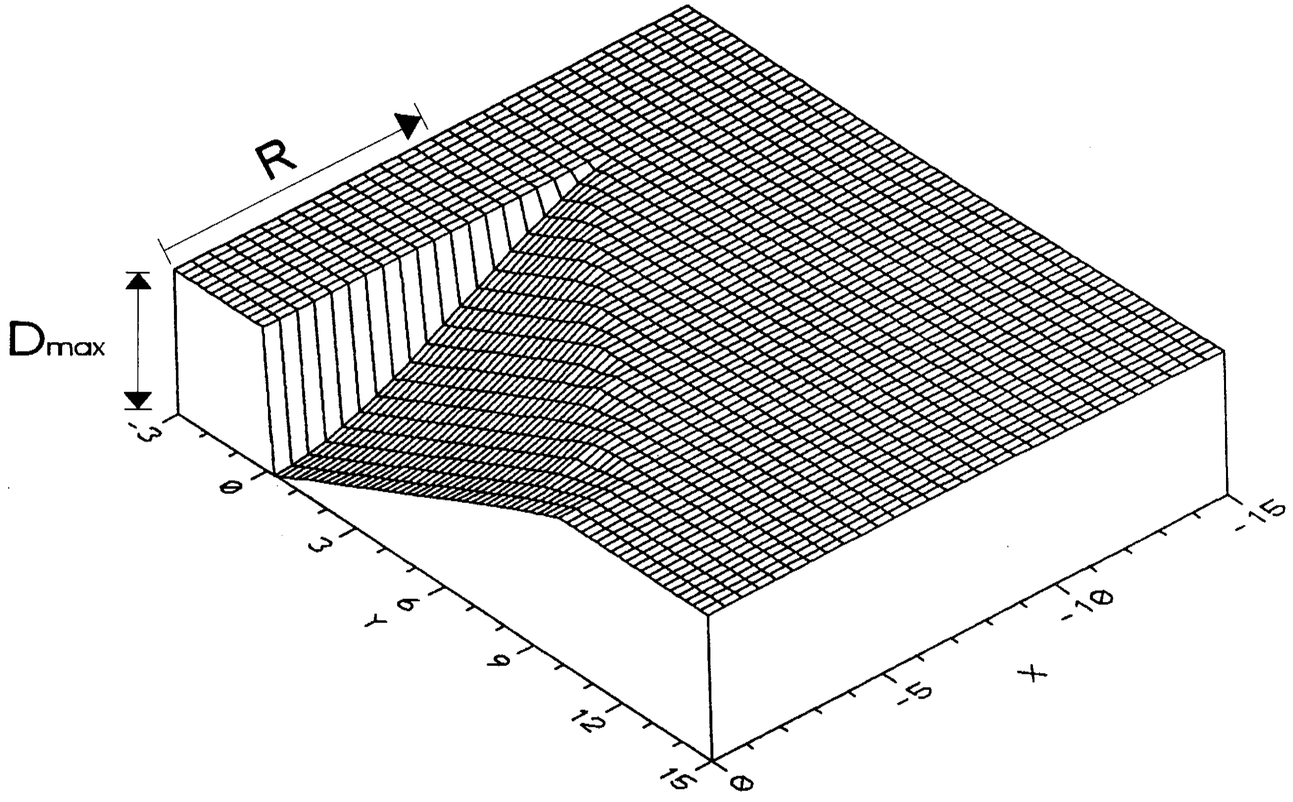
Walsh and Watterson (1988) looked at displacement versus various dimensions of the fault zone. In examining data from several faults, they found that the ratio L/D_{max} ranged from about 10 to 2000 where D_{max} is the maximum displacement along a fault and L is the length of the fault, i.e. the maximum distance along the fault plane perpendicular to the direction of slip between points of zero displacement (note that Walsh and Watterson refer to L as width). They observed that this ratio tended to decrease with increasing L.

Another relationship of interest is between displacement in the fault plane and displacement profiles perpendicular to the fault plane. Barnett et al (1987) described such profiles, referred to there as reverse drag profiles. They observed a relationship between width, W (the perpendicular distance from the fault plane to the point where there is effectively no deformation), and displacement, D. They found that the ratio W/D tends to be greater than 20 for faults with D_{max} less than 10 m and less than 10 for faults with D_{max} greater than 10 m (see Figures 2.1 and 2.2. Figure 2.2 shows the displacement of a horizontal surface on one side of a vertical fault).

Gibson et al (1989) note that until recently, it was believed that the footwall of a normal fault was passive, i.e. that there was no apparent displacement on the footwall side of the fault. Barnett et al (1987) claim that the footwall is not passive, but that reverse drag profiles are not generally symmetric across a fault plane. They argue that there is maximum hanging wall reverse drag in the lower part of the fault and minimum in the upper part of the fault. They also claim that the asymmetry increases with decreasing angle between fault and local bedding with more reverse drag appearing in the hanging wall. Thus, a lower angle fault that intersects the surface may appear to have primarily hanging



(a)



(b)

Figure 2.2 - (a) Displacement of a horizontal surface by a 90° fault; (b) the same as (a) but sliced at X = 0.

wall reverse drag.

2.4 Ideal Fault Model

This section discusses an idealized single blind-normal fault described by Watterson (1986) and Walsh and Watterson (1987,1989) (a blind fault is one that does not intersect the surface at any time in its active history). For such a fault, two-dimensional contours of bed displacement in a cross-section perpendicular to the direction of slip form concentric half-ellipses on either side of the fault centered about the point of maximum displacement; they form half-ellipses on either side of the fault because the direction and magnitude of deformation is different on either side of the fault plane as discussed above.

The point of maximum displacement is assumed to occur at the centroid of the tip-line ellipsoid, the three-dimensional contour of zero displacement. For any cross section perpendicular to the direction of slip, the fault would appear as a line with end points intersecting the tip-line ellipsoid with the point of maximum displacement (along that line) at the center; the distance from any point on this line to the center is denoted by R. The total length of this line is denoted by L. Variation of displacement, D, in the fault plane <u>in the direction perpendicular to slip</u>, based on an arithmetic growth model, is given by

$$D = 2D_{max}\{[(1 + r)/2]^2 - r^2\}^{1/2}(1 - r) \qquad (2.1)$$

where r is the normalized distance along the line of the point from the center of the line, r = R/(L/2); note that this description of variation is neither elliptical nor parabolic. This profile has been closely matched in observations of actual faults and can be seen in Figure 2.2 which shows the displacement of a horizontal surface perpendicular to a vertical fault. Displacement in the fault plane also varies in the direction of slip so that contours of displacement at the fault plane, for either the hanging wall or the footwall, would form concentric ellipses with the outer ellipse being the tip-line ellipse.

The ideal reverse drag profile is given by the equation

$$(1 - \delta)^2 + (1 - \mu)^2 = 1 \tag{2.2}$$

where μ is the normalized perpendicular distance from the fault plane (normalized by the distance from the plane to the tip-line ellipsoid) and δ is the normalized bedding displacement (normalized by the displacement on the fault plane for the given value of X and the location along the fault plane in the direction of slip). Figure 2.3 shows a graph of this profile for a vertical fault.

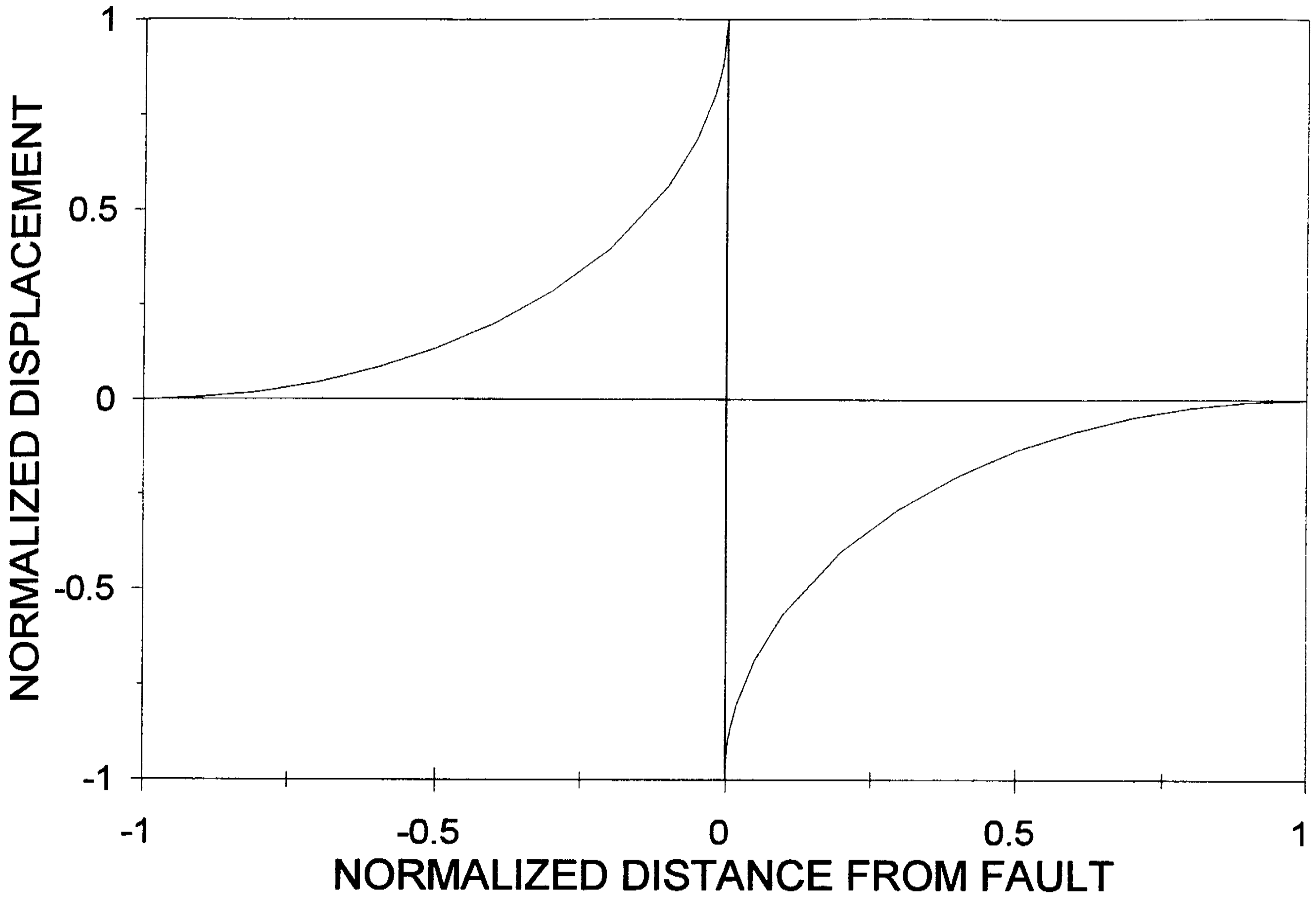


Figure 2.3 - Ideal reverse drag profile showing normalized displacement versus normalized distance from the fault plane.

CHAPTER 3

METHODS

A primary objective of this study was to create a computer simulation containing the essential geometric and hydrogeologic features of a fault. The code chosen for the flow simulation was MODFLOW, a finite-difference code developed for the USGS by McDonald and Harbaugh (1988). The reasons for the choice were i) the availability of the code; ii) MODFLOW is the single most commonly used code and so it is likely that MODFLOW (or some finite-difference code) would be used in the future to incorporate faults; iii) ease of use in the case of the vertical fault which is the primary case explored in this thesis. Thus, it seemed reasonable to uncover any associated problems with using such a code for this problem.

A pre/post-processor was written in FORTRAN to create the MODFLOW input files describing the region containing the fault and to process the MODFLOW output. The pre/post processor and various aspects of using MODFLOW for this problem are described in Appendix C. The FORTRAN code for the pre/post processor is found in Appendix A.

3.1 The Fault Model

The fault model used is based on the ideal fault model described in Chapter 2. Certain simplifications were made either for ease of coding or for purposes of comparison of different values of a given variable. This model is for a normal fault which is by definition sub-vertical. Although the majority of faults simulated in this study are vertical, the model is considered applicable as a vertical fault is the limiting case as the dip of a normal fault approaches 90°.

The units for length or for hydraulic conductivity are not specifically written in the following description. One does not have to specify the actual units when using MODFLOW. Only the relative changes in hydraulic conductivity affect the heads, although groundwater flux is affected by the actual conductivity values. It is assumed that the reader can scale the input and results as appropriate. One reasonable scenario for the simulations in this study would be to let one MODFLOW length unit equal 100 meters; in this case, the length of the fault, as described below, would be 2 kilometers. Length, then, will be in "MODFLOW units", referred to merely as "units". Units for hydraulic conductivity will not be explicitly used and any scale conversion will be left to the reader.

The region modeled is a box with length of approximately 100 MODFLOW units in the X and Y directions and 40 units in the Z direction (see Figure 2.1; it shows a portion of the region with the fault cross sectioned. The coordinate axes show the directions of the fault dip and displacement). The origin of the coordinate system is found at the centroid of the region; this always corresponds to the centroid of the fault plane.

The region contains three geologic layers which were horizontal before insertion of the fault: an aquifer confined by an aquitard above and below. The aquifer thickness was fixed at two units with the unfaulted aquifer centered about the plane Z = 0.

The length of the fault, L, was also fixed at 20 units; all other geometric features of the fault were variable (the range of values for the variable parameters being chosen relative to this value of L). The centroid of the fault within the modeled region is at coordinates (X,Y,Z) = (0,0,0). For the 45° fault, fault rotation is about the line (X,0,0) with dip in the +Y direction, i.e. the hanging wall is taken to be in the +Y direction from the fault plane. For the case of a vertical fault, the fault plane is contained in the plane Y = 0 and slip is in the -Z direction.

The variation in displacement along the fault in the X direction is given by Equation 2.1. A simplification of the ideal fault model used in these simulations was that there is no variation in displacement in the direction of slip; e.g., for a vertical fault, the amount of displacement in the fault plane is a function of X only. This implies that for a given geologic unit, permeability in the fault plane varies only in the direction perpendicular to slip. As the aquifer thickness is relatively small compared to the length of the fault, the amount of variation in displacement in the direction of slip within the aquifer would be small relative to the total variation and so neglecting it is taken to be a reasonable simplification.

Another simplification made for this model is that the reverse drag profile is linear instead of parabolic as described by Equation 2.2. This simplification eased the coding of the preprocessor and, based on the results presented in Chapter 4, likely had little effect.

The various features of the fault model can be seen in Figures 2.1 and 2.2. Figure 2.2 shows the displacement of a horizontal surface on one side of a 90° fault. Figure 2.1 shows displacement on both sides of the fault plane.

Finally, displacement of both walls was compared with displacement in only one wall. As will be seen, results for both displacement models are similar. For the purpose of the sensitivity analysis where different fault properties are varied, displacement in one wall only was used.

3.2 Fault Zone Permeability

The original idea was to pick a fault zone thickness, y_f, and then assign an effective conductivity tensor to each finite-difference cell intersected by the fault. This assignment is based on the thickness of the fault zone relative to the cell thickness and the permeability of the fault zone at that location in the principle directions. This is a simple matter for the case of a vertical fault as the principle directions of conductivity align with the MODFLOW coordinates. In this case, the effective conductivity tensor is $\mathbf{K}_{ef}$ = (Kx_{ef}, Ky_{ef}, Kz_{ef}) where Ky_{ef} is the harmonic mean

$$Ky_{ef} = \frac{\Delta y}{\frac{y_1}{K_1} + \frac{y_f}{K_f} + \frac{y_2}{K_2}} \tag{3.1}$$

and Kx_{ef} = Kz_{ef} is given by the arithmetic mean

$$Kx_{ef} = \frac{y_1\ K_1 + y_f\ K_f + y_2\ K_2}{\Delta y} \tag{3.2}$$

where all variables are described in Figure 3.1. Figure 3.1 shows a YZ cross-section of a finite-difference cell containing the fault zone for a vertical fault.

For reasons described in Appendix C, a different approach was used to simulate a sub-vertical fault zone. This approach, in the case of a 45° fault, is illustrated in Figure 3.2, where the fault zone is represented by a 45° "band" of homogeneous cells (Figure 3.2 shows a Y-Z cross section of the finite-difference grid where the displaced lower aquifer contact intersects the fault). Clearly, a different grid would be required for every fault angle (though any grid could be used for a dip of 90°). For this reason, and others described below, it was decided to compare only fault dips of 90° and 45°.

As described in Chapter 2, both the fault zone thickness (y_f in Figure 3.1) and permeability, K_f, are, in part,

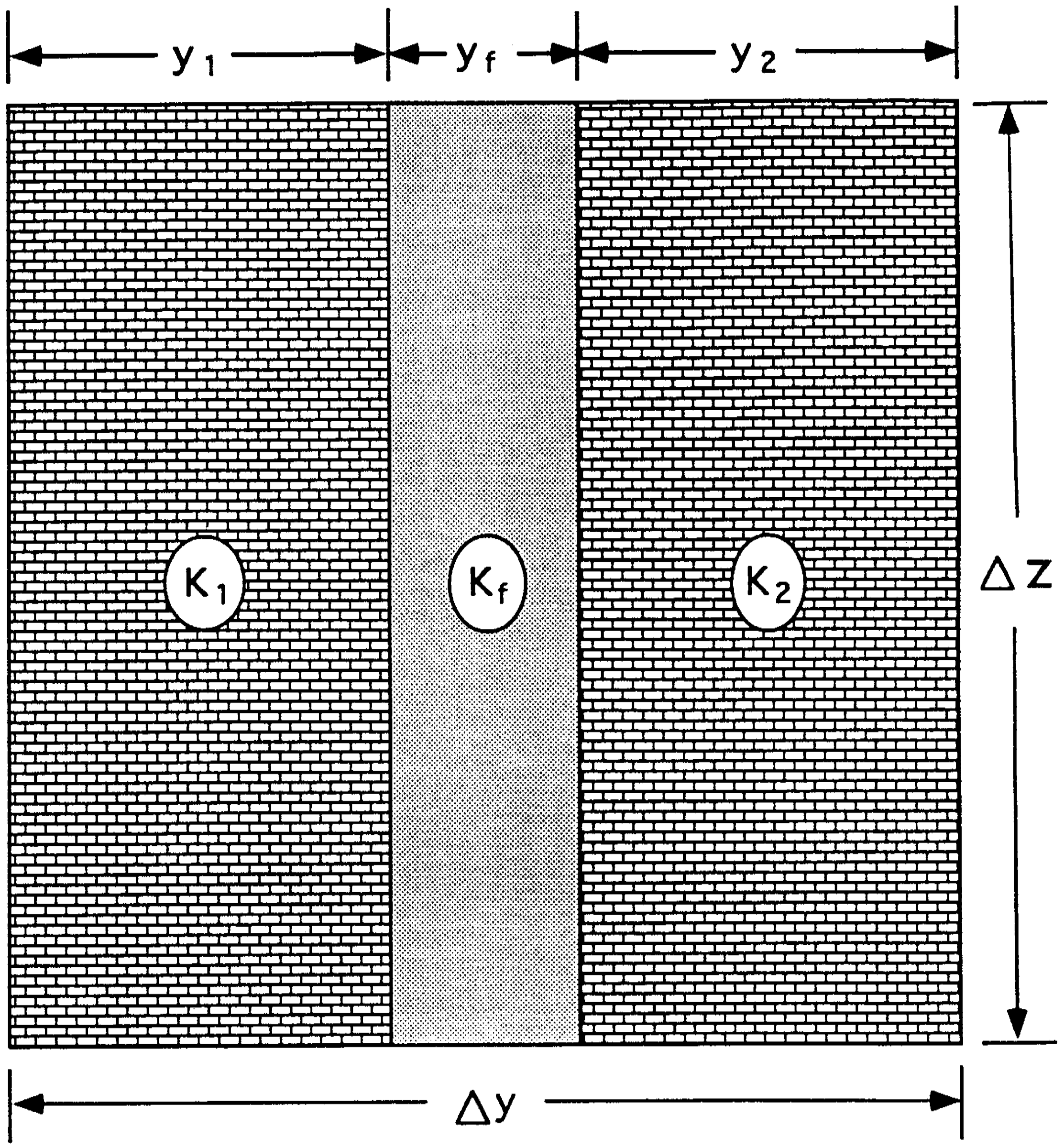


Figure 3.1 - YZ cross section of a finite-difference cell containing a 90° fault. K_f is the fault zone conductivity.

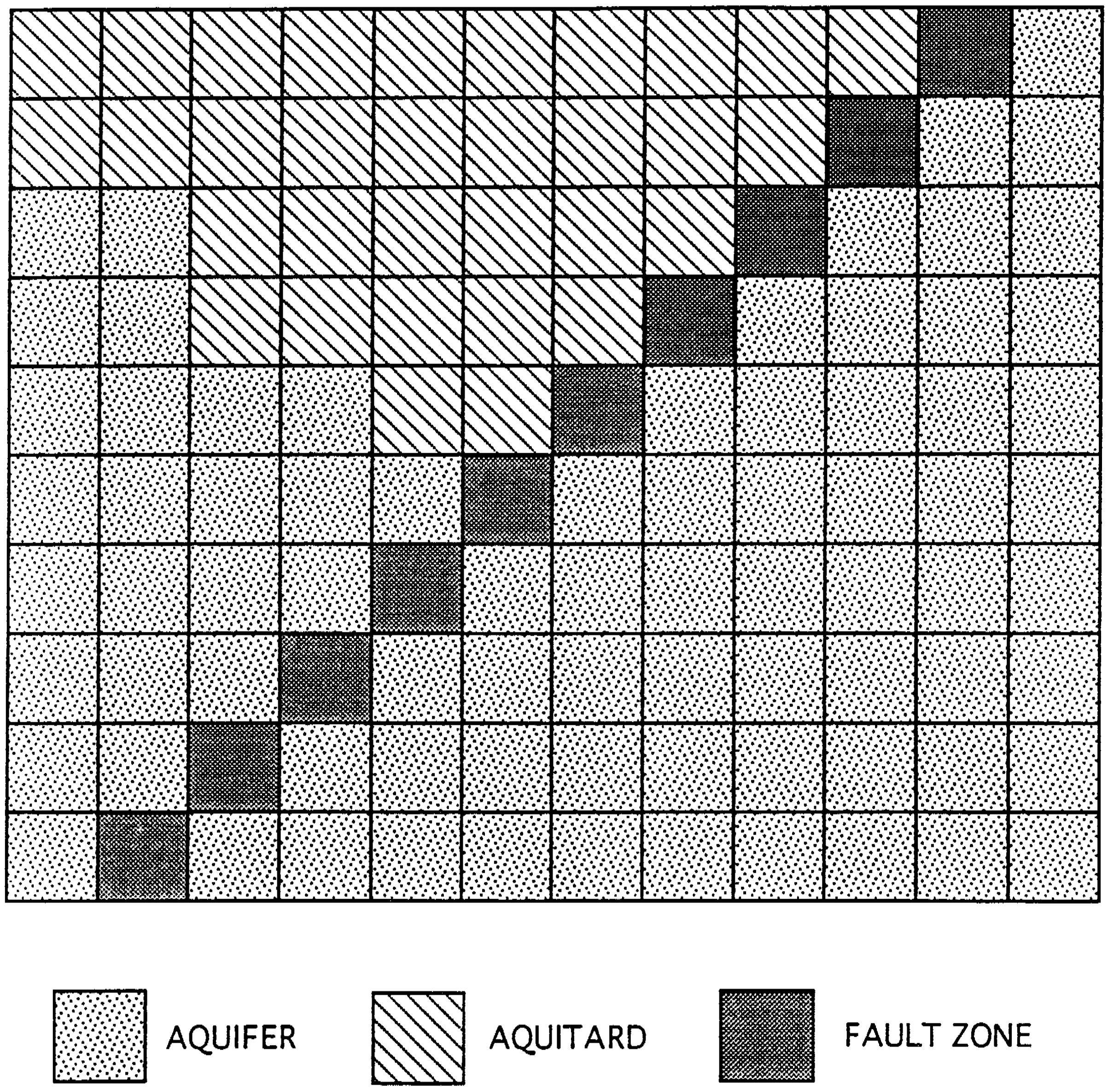


Figure 3.2 - Portion of a YZ profile of the finite-difference grid containing a 45° fault. Fault zone is represented by a diagonal of homogeneous cells.

functions of displacement. Though there is information on the relationship between D and y_f, no information could be found on the relationship between D and fault zone permeabilities. So two approaches were taken to generate $\mathbf{K}_{ef}$ distributions. One approach was to hold K_f constant in Equations 3.1 and 3.2 and to vary the fault zone thickness, y_f, as a function of D: $y_f = \mu D$ where $\mu = y_{fmax}/D_{max}$. The second approach was to let y_f remain constant with D, but vary K_f as follows:

$$K_f = EXP((D/D_{max})*Ln(K_{min}) + (1 - (D/D_{max}))Ln(K_{max}))$$

where K_{min} is the minimum K value achieved in the fault zone for a particular hydrogeologic layer (this occurs along the line of maximum displacement), K_{max} is the K value of the undisturbed layer and D/D_{max} is described by Equation 2.1. This distribution was chosen, first because it ranges from $K_f = K_{min}$ for $D = D_{max}$ to $K_f = K_{max}$ for $D = 0$ and, secondly, it results in a more gradually varying $\mathbf{K}_{ef}$ distribution than for the variable y_f distribution. The two resulting Ky_{ef} distributions, the variable y_f and the variable K_f distributions, are shown in Figure 3.3 where $K_{min} = .001$, $K_{max} = .1$ and the minimum value for $y_f = .01$, using normalized coordinates. In an actual fault zone, both y_f and K_f would vary with D, but lacking data on such relationships, the above distributions were created in order to observe the variation in heads and gradients with the different effective conductivity distributions.

One set of hydraulic conductivity values was selected for

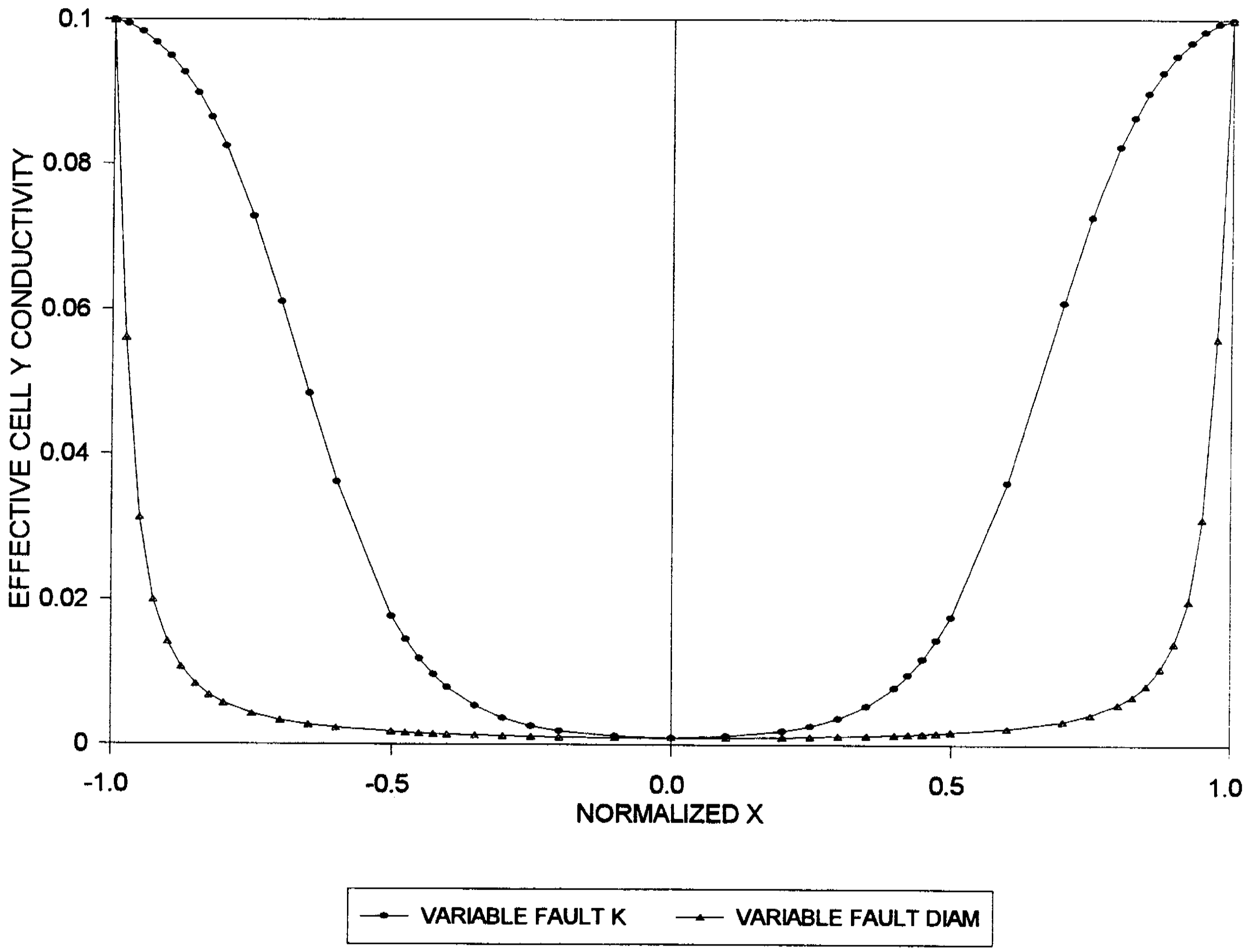


Figure 3.3 - Graph of two Ky_{ef} distributions versus relative distance from the fault centroid in a direction perpendicular to displacement and within the fault.

the 90° fault simulations and one set when comparing 90° and 45° faults for the sensitivity analysis (described below). The following K values were chosen for the 90° fault simulations: for the aquifer, K_{max} = .1, K_{min} = .0001; for the aquitards, K_{max} = .001, K_{min} = .0001. When comparing 90° and 45° faults, K_{min} was set to .001 (the reason for this is that, with a 3 order of magnitude difference in conductivities, the grid density used, as described in the next section, resulted in excessively long computer runs). As one of the main objectives of this study is to look at how geometric properties and conductivity variations along the fault affect groundwater flow, this choice, though arbitrary, was within the realm of possibility and gave enough relative contrast in conductivities to yield clear results.

For some of the simulations, permeability in the X and Z directions were enhanced for the 90° fault simulations to simulate an unmineralized fracture zone on either side of the gouge zone. To do this, a conductivity factor, KF, was used as a multiplier of Kx_{ef} and Kz_{ef}. The conductivity factor, KF, for a particular cell in the fault is taken to be a function of the amount of relative displacement, d, as follows:

$$KF = 1.0 + d(KF_{max} - 1.0) \qquad (3.3)$$

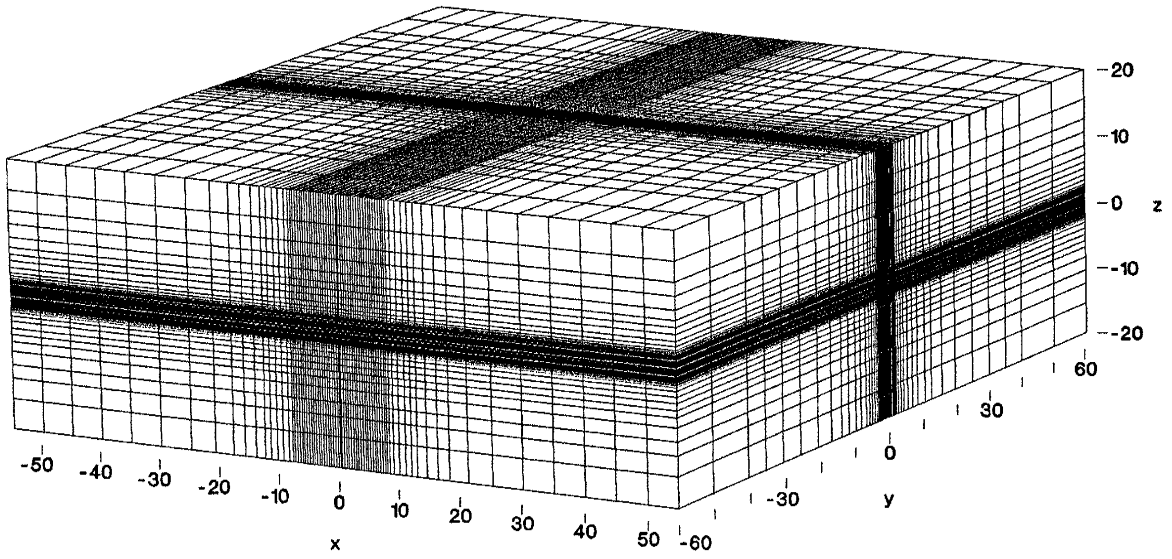
where KF_{max} is the maximum value of KF. The enhanced hydraulic conductivity tensor for the cells containing the fault zone is then taken to be $\mathbf{K}_{ef} = (KF \cdot Kx_{en}, Ky_{ef}, KF \cdot Kz_{en})$ where Kx_{ef}, Ky_{ef} and Kz_{ef} are given by Equations 3.1 and 3.2. With enhanced vertical flow along the fault, the no-flow boundary conditions at the top and bottom of the fault plane are not generally realistic and so these results should be viewed with caution.

3.3 The Numerical Model

MODFLOW, the USGS modular, three-dimensional finite-difference code was run on a SUN SPARC 2 with the Preconditioned Conjugate Gradient 2 (PCG2) equation solver module to simulate steady-state flow through the fault zone. MODFLOW was modified by the author to allow for cell-by-cell XY anisotropy (i.e. $Kx \neq Ky$) instead of layer-by-layer as in the original code. This requires a file of Ky/Kx values for each cell in the grid (the Ky/Kz values are indirectly defined by the VCONT file).

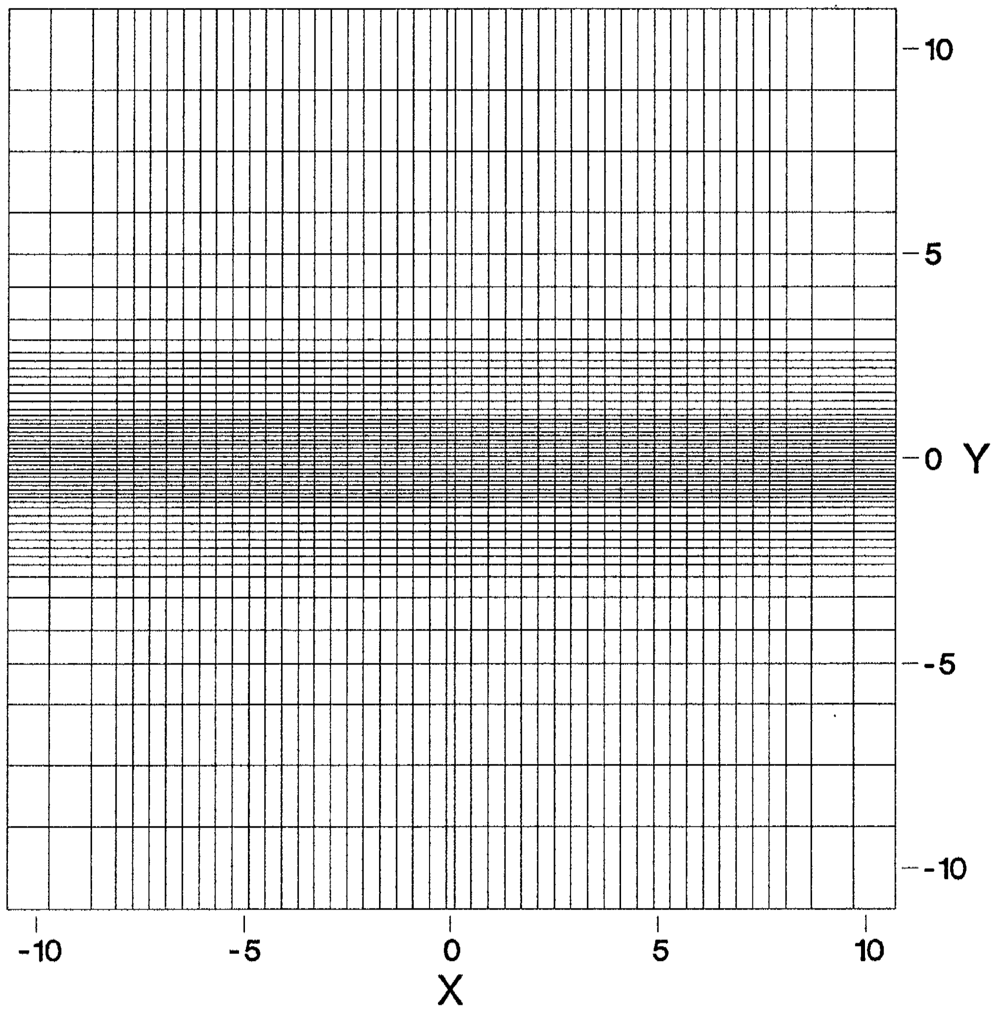
Two different grids were used, one for 90° fault simulations, and one when comparing 90° and 45° faults. The grid used for the 90° fault is shown in Figure 3.4(a) where $\Delta Y = \Delta Z = .1$ in the area of the fault and the fault zone thickness is .01. Figures 3.4(b) and (c) show highlights of the XY and YZ grids.

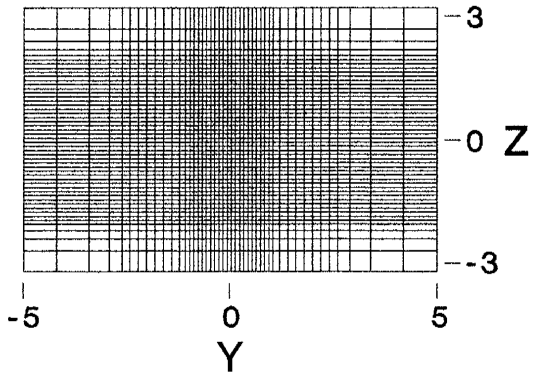
In order to simulate a fault zone that is not disproportionately thick, the grid must be reasonably fine in the region of the fault for the 90° and 45° fault comparison. However, the denser the grid, the slower each MODFLOW



(a)

Figure 3.4 - (a) Three-dimensional finite-difference grid used in 90° fault simulations; (b) highlight of XY and YZ finite-difference grid.





(b)

iteration and the more iterations required. The grid size used in the vicinity of the fault for this comparison was $\Delta Y = \Delta Z = .3$.

In comparing the 90° and 45° faults, it is assumed that the same fault used for the 90° fault is rotated by 45° to achieve the 45° fault. With the 45° fault simulated as in Figure 3.3 and for a given square cell length in the Y direction, there is the question of what fault zone thickness is represented; does the diagonal across the cell represent the fault thickness or does the cell length in the Y direction represent the distance across the 45° fault zone in the Y direction? It is unclear what the correct answer to this question is. This being the case, and with the two choices of fault zone thickness differing by only a factor of $2^{1/2}$, the later of the above two choices was selected arbitrarily. The fault zone thickness, T, for the comparable 90° fault with a cell length of .3 is then given by

$$T = (0.3)\ \mathrm{SIN}(45°) = .212$$

i.e. this is the value of y_f that was used in Equations 3.1 and 3.2. for the 90° fault. As it is not clear that this is the correct value, the results of the 90° and 45° fault comparison should be regarded with caution.

3.4 The Simulations

Table 3.1 shows the range of values used for the 90° fault simulations. The net flow gradient was in the +Y direction. In addition to these simulations, one simulation was run with the flow gradient in the -X direction, with D = 2 and W = 10. The results were predictable and not terribly interesting and are not shown (flow directions mirrored the aquifer geometry). The values used for the 90° and 45° fault comparison are found in Table 3.2.

The net flow gradient was generated by imposing constant head boundary conditions on the YZ planes that formed regional boundaries and assigning them initial head values (H = 200 units at X = -55 units and H = 100 units at X = 55 units resulting in a regional gradient of .91). All other boundaries were no-flow boundaries. To insure the boundary conditions were not constraining the results, simulations were run with the size of the region doubled in each variable individually. This was compared with the simulation in the original region. The parameters used in these simulations were $\theta = 90°$, $D_{max} = 2$, $W_{max} = 10$ and $\mathbf{K}_{ef}$ variation along the fault was determined by the varying K model. The size of the original region was deemed acceptable if the maximum difference in head values between the two simulations divided by the net regional head gradient was small compared with the maximum normalized head change between fault and no-fault simulations.

TABLE 3.1
Parameter Values for 90° Simulations

DISPLACEMENT	WIDTH	FAULT Ky MODEL	KF_{max}
0.0	na	na	na
0.0	na	Variable Perm	1
0.3	10	Variable Perm	1
0.5	10	Variable Perm	1
1.0	10	Variable Perm	1
1.5	10	Variable Perm	1
2.0	10	Variable Perm	1
2.0	7	Variable Perm	1
2.0	5	Variable Perm	1
2.0	4	Variable Perm	1
2.0	3	Variable Perm	1
0.0	na	Variable Diam	1
0.3	10	Variable Diam	1
0.5	10	Variable Diam	1
1.0	10	Variable Diam	1
1.5	10	Variable Diam	1
2.0	10	Variable Diam	1
2.0	7	Variable Diam	1
2.0	5	Variable Diam	1
2.0	4	Variable Diam	1
2.0	3	Variable Diam	1
2.0	10	Variable Perm	10
2.0	10	Variable Perm	100
2.0	10	Variable Perm	1000

DISPLACEMENT	WIDTH	FAULT Ky MODEL	KF_{max}
0.0	na	na	na
2.0	10	Variable Diam	10
2.0	10	Variable Diam	100
2.0	10	Variable Diam	1000

TABLE 3.2
Parameter Values for 90° vs. 45° Faults

DIP	DISPLACEMENT	y_f
45°	0.9	0.3
90°	0.9	0.212
45°	1.8	0.3
90°	1.8	0.212

3.5 Representation of Results

MODFLOW was modified to output a file that contains the steady-state head values for each layer beginning with the top layer. The post processor can then take a slice of the head file parallel to a XY-plane, YZ-plane or XZ-plane and output a file in the form (a,b,h) were a and b give the point location in the plane and h is the head value.

As the grids used had irregular spacing, and because SURFER does a poor job of contouring when the grid is not regular, the slice file was then interpolated onto a regular grid. This was done using a technique described by Macedonio and Pareschi (1991). First, a desired regular grid is specified. Then, an optimal triangulation of the (a,b) points is formed. The h value for each point of the regular grid is then interpolated: the h value for a point, P, on the regular grid is interpolated by linear interpolation over the triangle that contains P. SURFER can then be used to generate contour plots.

The X = 0 plane was used for YZ slices as this plane contains maximum displacement along the fault. The Z = 0 plane was used for XY slices.

3.6 Sensitivity Analysis

A variety of statistics were compiled for each simulation. These are i) maximum head change from the no-fault simulation; ii) maximum head gradient across the fault (i.e. in the Y direction). A cell on either side of the cell containing the fault was used; iii) maximum Z gradient over the entire region determined by calculating the vertical gradient between all pairs of vertically adjoining cells; iv) flow, Q, through the aquifer across the fault (through the undisplaced wall of the aquifer). The magnitude of the net regional gradient from the unfaulted case (.91) was used to normalize all gradients and head values as these values are dependent on the net regional gradient. Flow through the aquifer was normalized by flow through the same portion of the aquifer without a fault. These statistics were then graphed, showing the results for both $\mathbf{K}_{ef}$ distributions on the same graph, versus various maximum displacement and maximum fault width. These statistics were also graphed versus the logarithm of KF_{max}.

3.7 Well Fields

The pre/post processor has the capability to generate a set of (X,Y,Z_{max},Z_{min}) values where Z_{max} and Z_{min} are the Z values of the upper and lower aquifer/aquitard contacts, respectively, at (X,Y). The set can either be for the entire set of (X,Y) points in the MODFLOW grid, or it can be randomly generated using a uniform random number generator over a subdomain of the simulated region. The pre/post processor can then take that set and, for any simulation, return a set of triplets, (X,Y,h_{int}), where h_{int} is the head at (X,Y) integrated over the thickness of the aquifer from Z_{max} to Z_{min}, thus simulating the head in a well screened over the thickness of the aquifer.

Using the results of simulations with two-wall displacement, maximum displacement $D_{max} = 2$ (1 on each wall), maximum fault width $W_{max} = 10$ and the variable fault diameter effective conductivity distribution, several sets of triplets (X,Y,h_{int}) were generated with 5, 10, 30 and 200 triplets per set over the subdomain from $X = -10$ to $X = 10$ and $Y = -10$ to $Y = 10$ with $K_f = .0001$ for the aquifer and the aquitard. Sets of 200 triplets (X,Y,h_{int}) were also generated for simulations with K_f = .01, .001 and $K_1 = K_2$ = .1 for the aquifer and K_f = .0001 and $K_1 = K_2$ = .001 for the aquitard (refer to Figure 3.1). This was repeated for the subdomain from X = -2 to X = 2 and Y = -2 to Y = 2. For each subdomain, a dense set of wells corresponding to each grid node was also generated.

The question of how adequately the fault's effects can be detected from a given well density is approached graphically, contouring each set of well data as well as contouring the dense set of wells. Visual comparisons are then made.

The question of the degree to which a limited amount of well data can lead to misinterpretations of flow patterns in the vicinity of the fault is approached in two ways. This question is of some interest as decisions on where to place a small number of monitor wells when monitoring the storage of non-hazardous materials in areas with limited well information are fairly common. The first approach is to compare the contour plots of sparse data, as described above.

The second approach was as follows: for each set of 200 wells, the 3-point problem was solved for each subset of three non-collinear points. The collection of gradient magnitudes for each set was ordered from largest to smallest. The 95% confidence limit for the maximum gradient (magnitude), $GRAD_{95}$, is then obtained by selecting the smallest gradient magnitude that is larger than 95% of the gradients (as there are several sets generated for each number of triplets, the average over all $GRAD_{95}$ values for a given number of triplets is taken). These values for the different number of triplets are compared with the actual maximum gradient over the domain. This done for each sub-domain. Similarly, $GRAD_5$ is found representing the 5% confidence limit.

In addition to gradient magnitudes, the gradient directions are calculated for each triplet. The gradient directions are tabulated in the form of a histogram to consider probabilities of obtaining a given gradient direction from 3 random wells over the given sub-domain.

The gradient resulting from solving the three-point problem for a given well triplet will depend on the size and location of the triangle formed by the triplet in relation to the fault. This aspect is not really captured in the analysis described above. It was thought that one approach to consider the size and location factors would be one used by Steve Silliman of the University of Notre Dame for a different problem. That is to graph both the gradient magnitude and direction from well triplets versus triangle area. The shape of the graph would then give some insight into the connection between gradient and triangle size. This however does not factor in the location of the triangle in relation to the fault. To do this, triangle area multiplied by the distance between the fault centroid (in the XY plane) and the closest triangle vertex was plotted against both the gradient magnitude and direction. To keep the number of points under control, sets of 30 wells were generated over the 20 x 20 domain and the 4 x 4 domain described above.

Another consideration of this approach is whether these graphs might be unique in some way to these types of faults and, thus, be a useful tool for either identifying a fault, discerning certain fault properties or inferring that a known fault is significantly impacting ground water flow. This question is briefly considered.

CHAPTER 4

RESULTS

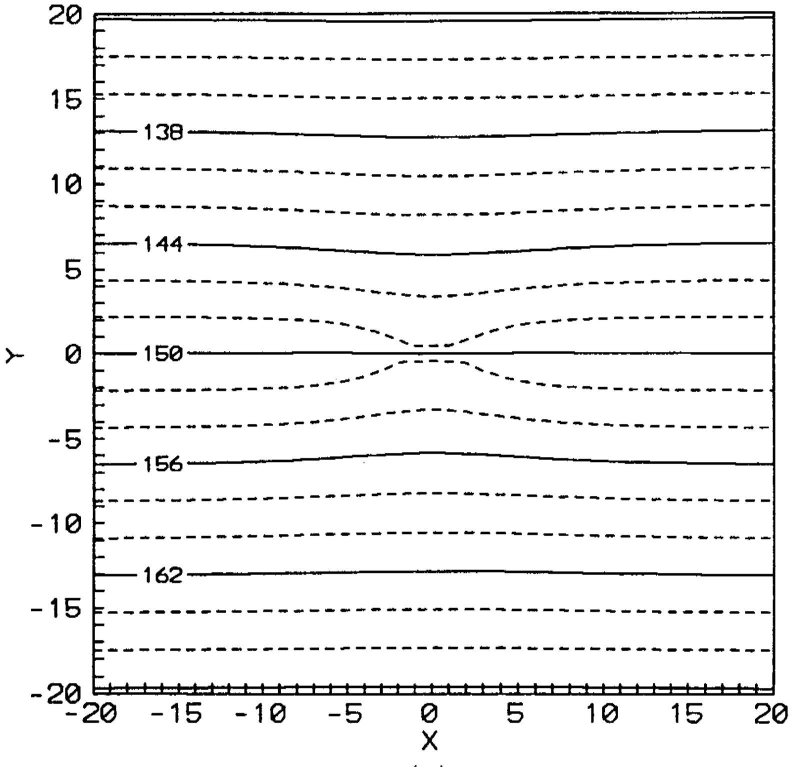
The maximum head differences between the simulations for the different size regions, normalized by the net no-fault gradient, are shown in Table 4.1. Considering the large gradients and head changes from the fault to the no-fault simulations (described below), these differences are sufficiently small so that they would not be noticed in either the contour plots or the performance measures. Therefore, boundary effects with the smaller grid were taken to be insignificant.

TABLE 4.1

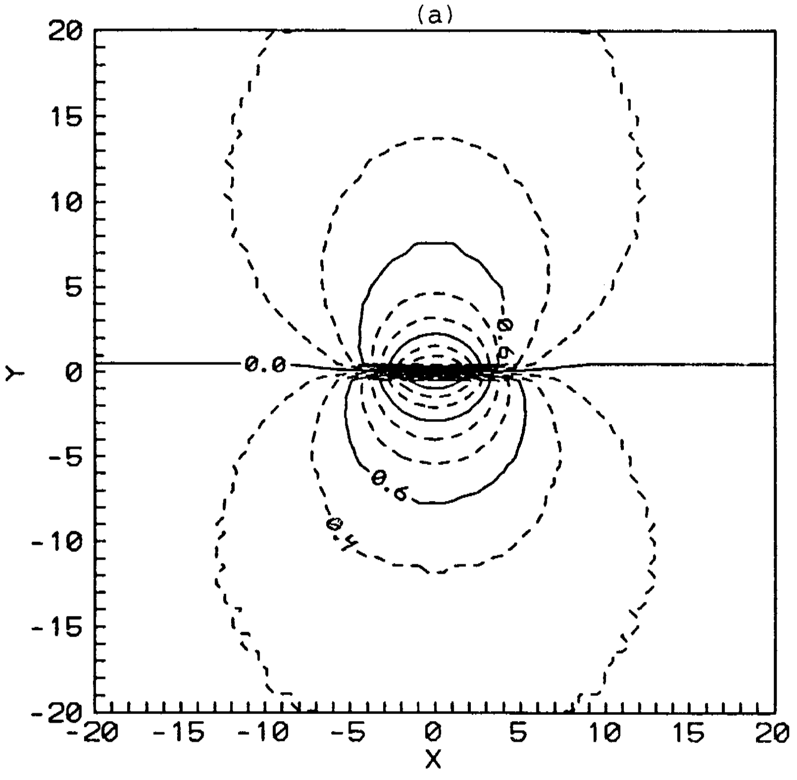
Normalized Head Difference in Enlarged Region

VARIABLE INCREASED		
X	Y	Z
0.022	0.16	0.086

A set of contour plots for four fault simulations are given in Figures 4.1 - 4.4. For both faults, θ = 90° and W = 10; however, for one simulation D_{max} = .5 (Figures 4.1 and 4.2) and for the other D_{max} = 2 (Figures 4.3 and 4.4). Each fault is simulated with $\mathbf{K}_{ef}$ along the fault calculated by both variable fault zone conductivity, K_f, and by variable fault zone thickness, y_f (see Figure 3.4), and with K_{min} = .0001 and K_{max} = .1 for the aquifer, and K_{min} = .0001 and K_{max} = .001 for the

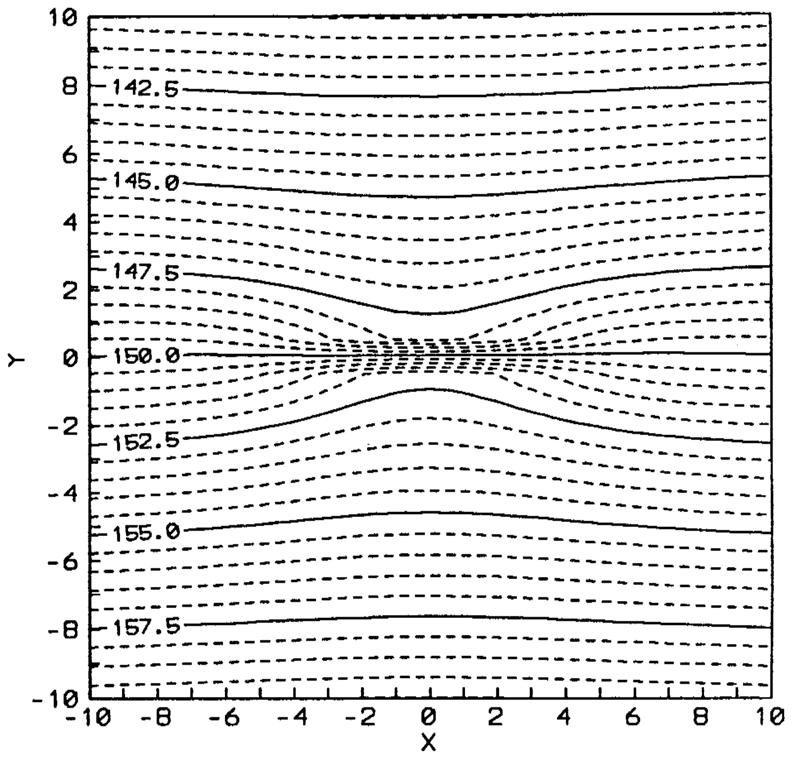


(a)



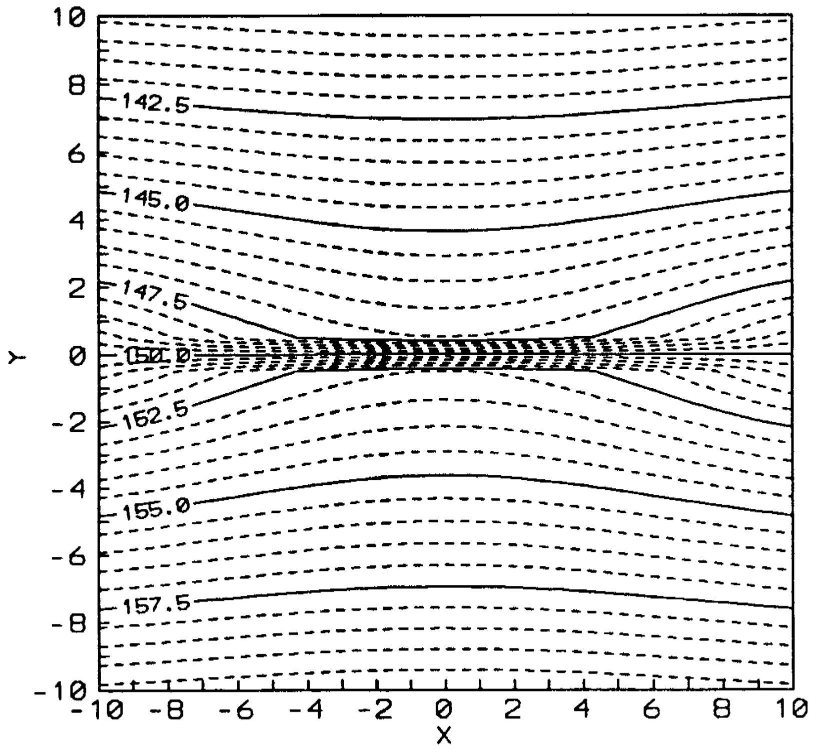
(b)

Figure 4.1 - θ = 90°, D_{max} = 0.5, W_{max} = 10, variable Kx distribution. (a) Head contours at Z = 0 cross section; (b) contours of head difference between fault and no fault simulations; (c) highlight of (a); (d) with variable y_f distribution.



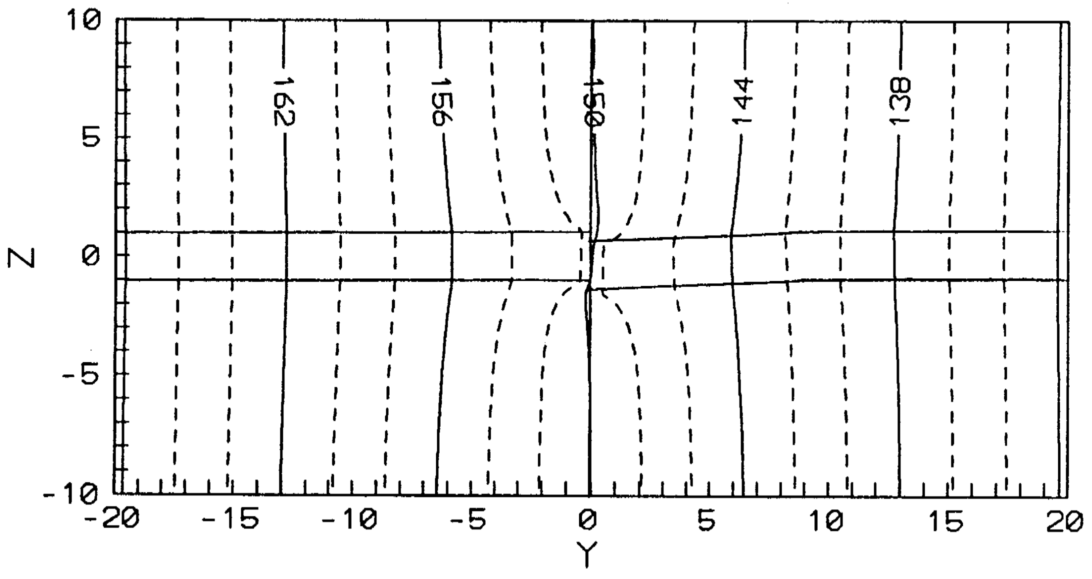
142.5
145.0
147.5
150.0
152.5
155.0
157.5
X
Y

(c)

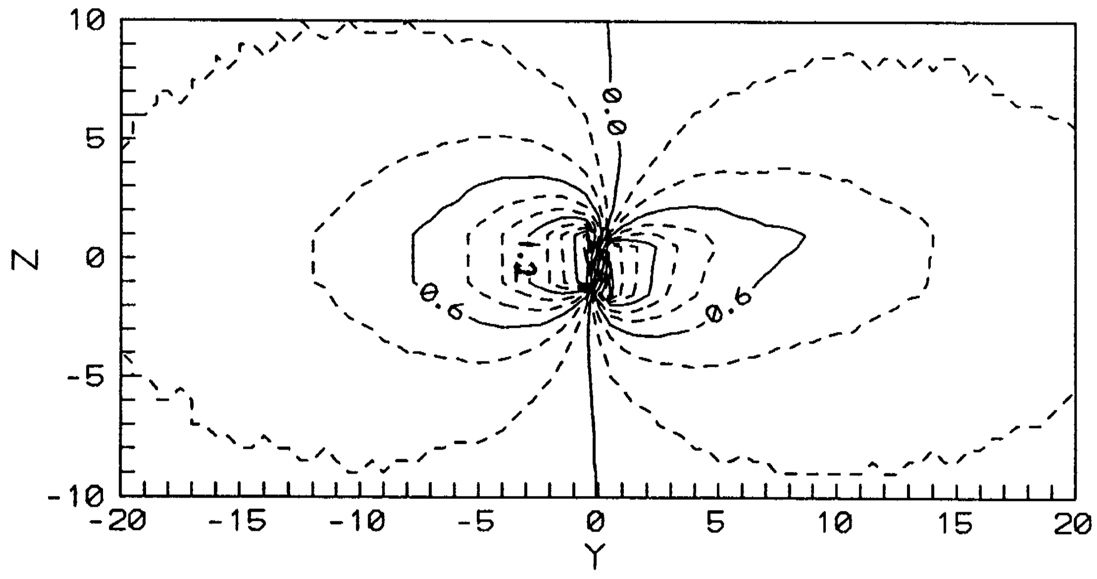


142.5
145.0
147.5
150.0
152.5
155.0
157.5
X
Y

(d)

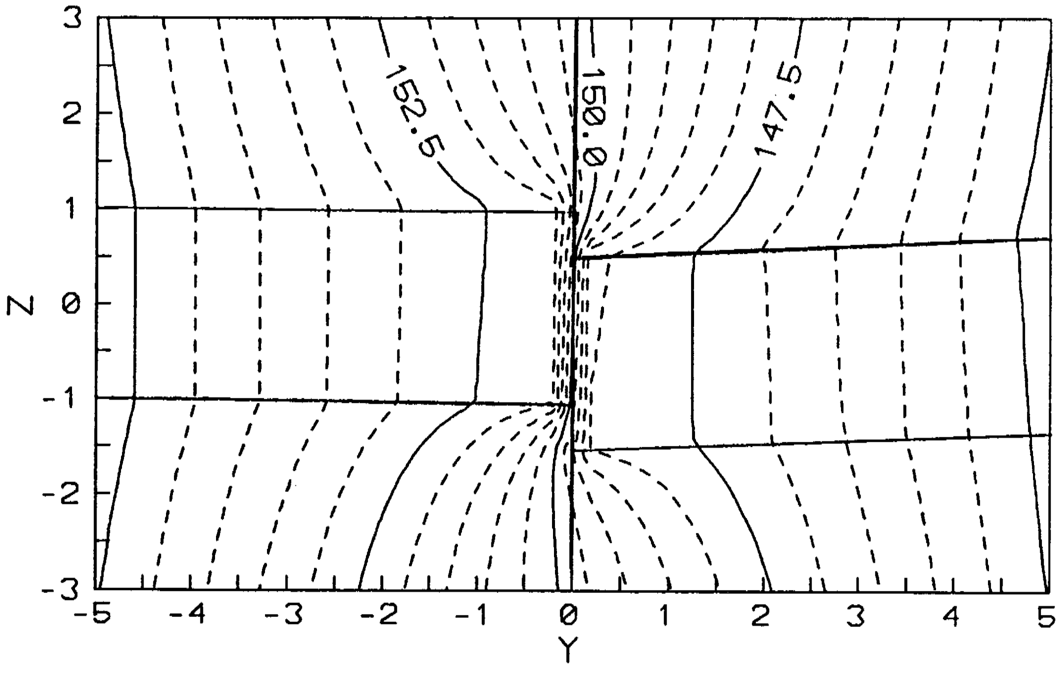


(a)

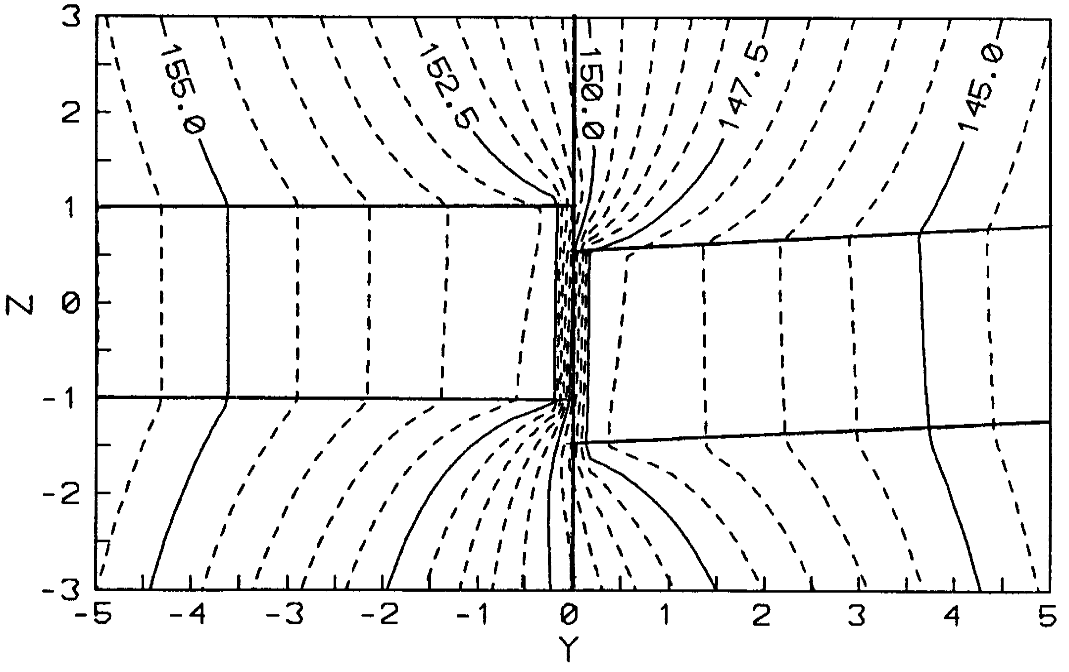


(b)

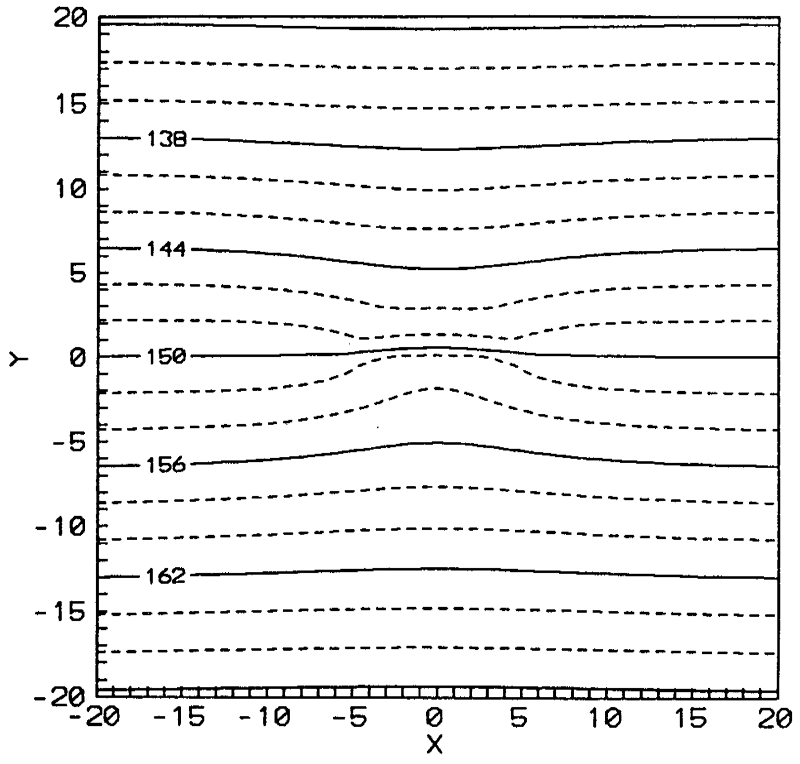
Figure 4.2 - YZ cross sections (a) head contours with variable K_f distribution; (b) contours of head difference between fault and no fault simulations; (c) highlight of (a); (d) with variable y_f distribution.


152.5
150.0
147.5
Z
Y
3
2
1
0
-1
-2
-3
-5
-4
-3
-2
-1
0
1
2
3
4
5

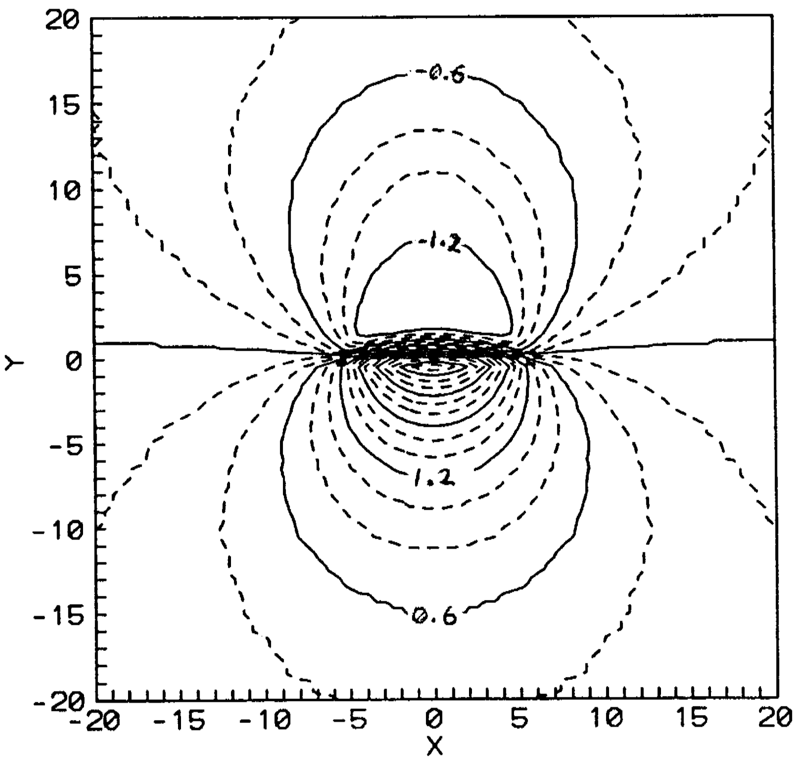
(c)


155.0
152.5
150.0
147.5
145.0
Z
Y
3
2
1
0
-1
-2
-3
-5
-4
-3
-2
-1
0
1
2
3
4
5

(d)

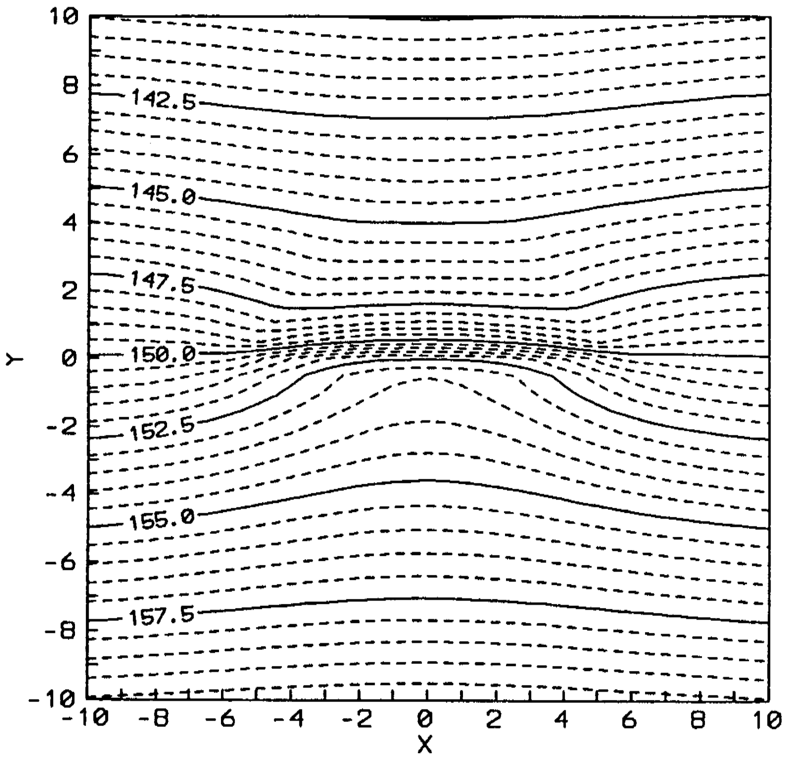


(a)

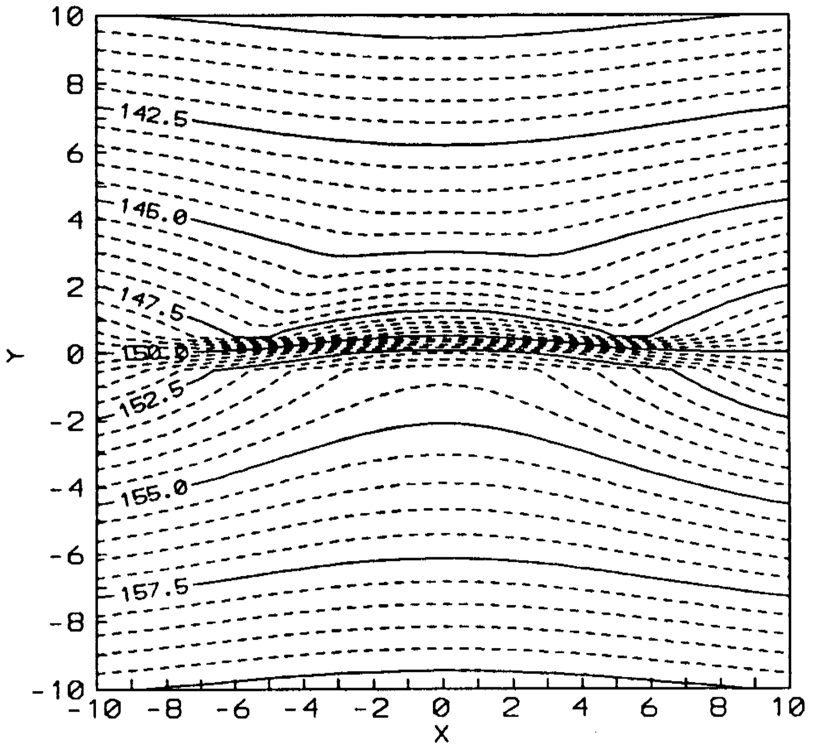


(b)

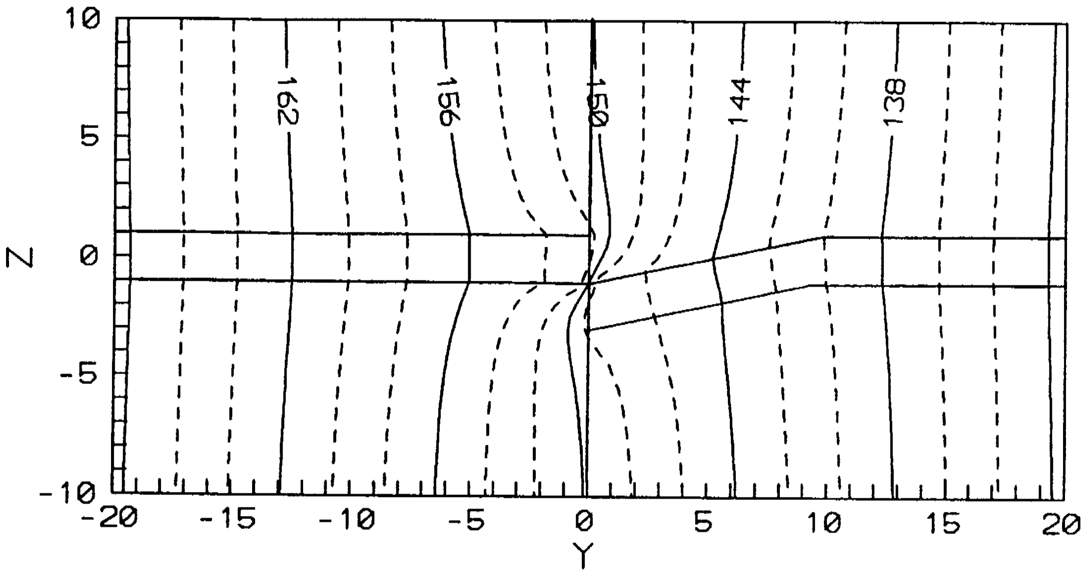
Figure 4.3 - Same as Figure 4.1 except with $D_{max} = 2$.


142.5
145.0
147.5
150.0
152.5
155.0
157.5
Y
X

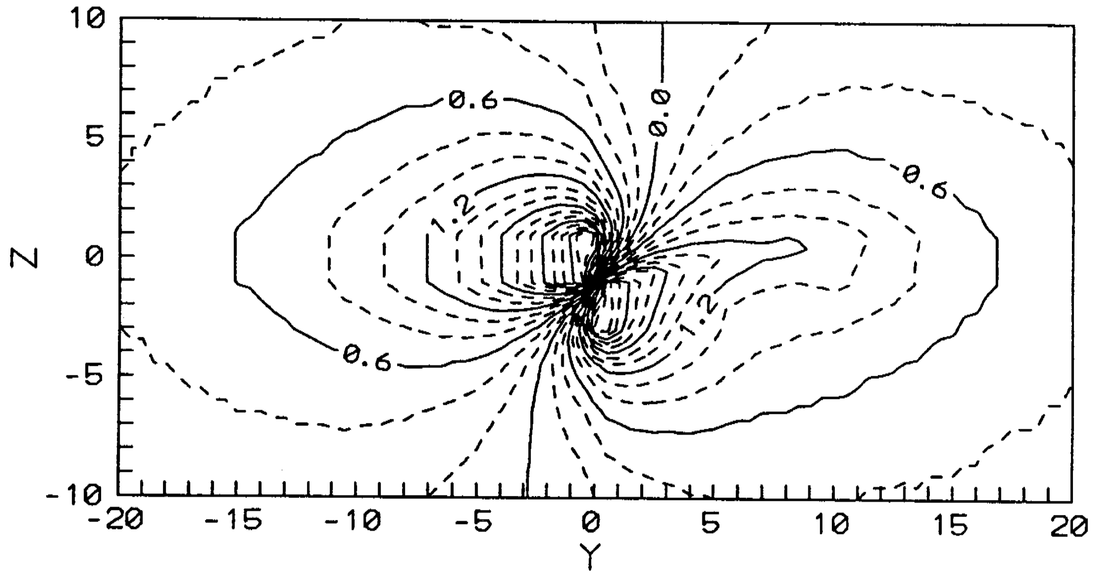
(c)


142.5
145.0
147.5
150.0
152.5
155.0
157.5
Y
X

(d)

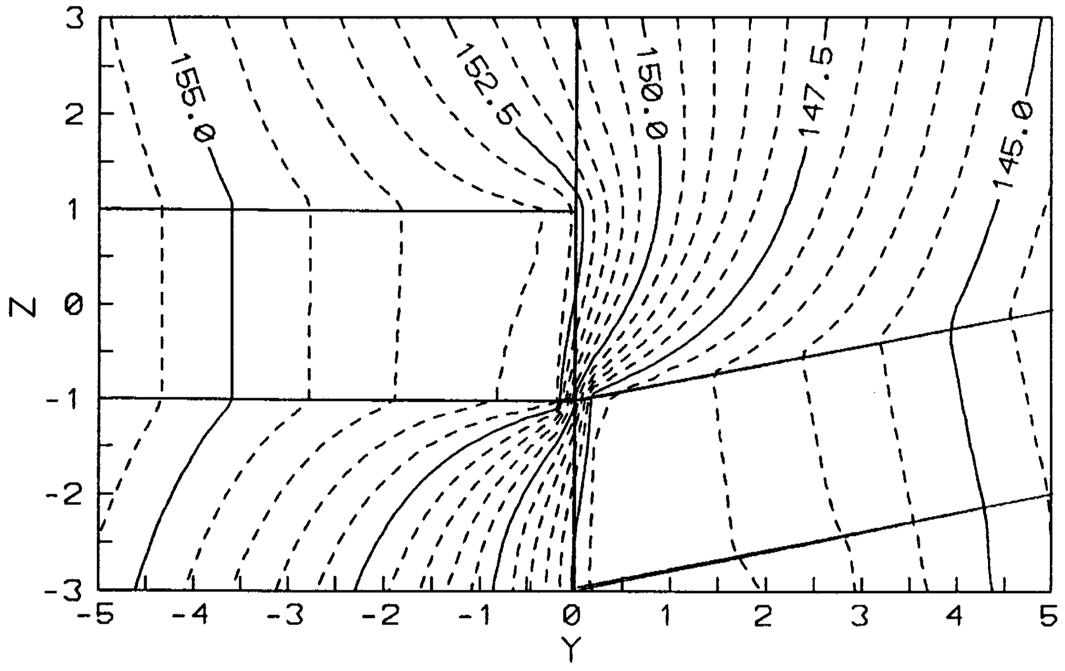


(a)

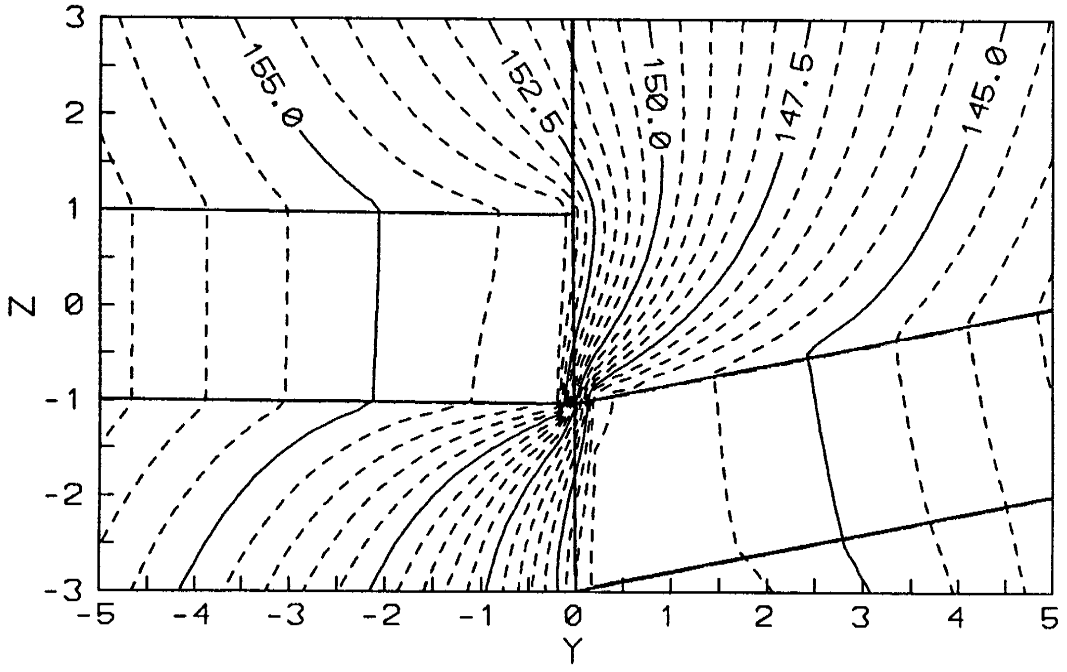


(b)

Figure 4.4 - Same as 4.2 except with D_{max} = 2.



(c)



(d)

aquitard.

Figure 4.1(a) shows a 40 x 40 portion of the regional map view through the horizontal slice Z = 0 with $\mathbf{K}_{ef}$ computed with the variable fault-zone conductivity model; the contour interval is 2 units. Figure 4.1(b) shows contours of the difference in head between the fault and no fault condition. The area influenced by the fault can be seen to lie approximately 10 units from the point (0,0,0) in the ±X direction and 20 units in the ±Y direction. A considerable head difference can be seen across the fault with a head increase on the footwall side and a decrease on the hanging wall side. Figure 4.1(c) shows an enlargement of the area of Figure 4.1(a) with a contour interval of 0.5 units. Slight asymmetry in the contours across the fault can be seen mirroring the geometric asymmetry.

The net regional head gradient has a magnitude of approximately 0.91; the maximum head gradient in the Y direction across the fault, normalized by the regional gradient (i.e., the gradient obtained with the same boundary conditions and no fault inserted), has a value of 22.4, a considerable increase from the no fault condition. In addition, the maximum normalized vertical head gradient increases from 0 to 8.8, and the maximum change in normalized head from the no-fault condition is 2.3 units.

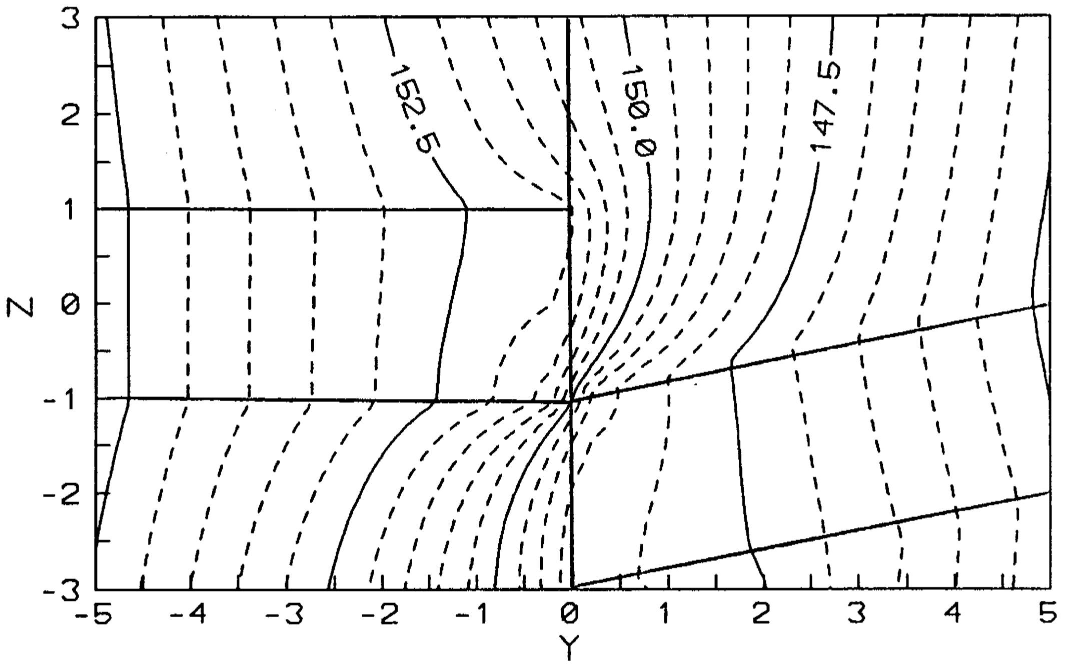
Figure 4.1(d) shows the equivalent of 4.1(c) except with $\mathbf{K}_{ef}$ determined with the variable y_f model. By comparing 4.1(d) with the contours of 4.1(c), it can be seen that Y gradients are increased even further; the maximum normalized head gradient in the Y direction across the fault is 30.6. In addition, the increase in Y gradient is sustained along more of the length of the fault. This is not surprising in light of the distributions graphed in Figure 3.4. The maximum normalized vertical gradient is 11.9 and the maximum normalized head difference with the no-fault condition for any one cell is 3.11 units.

Figure 4.2(a) shows the head contours in a YZ cross-section along the slice X = 0 from the simulation with a variable K_f distribution. The fault and aquifer can be seen. Figure 4.2(b) shows contours of variation from the no-fault condition; the contour interval is 0.2 (the jaggedness of the contours are assumed to be an artifact of the SURFER interpolation process). One can clearly see the depression of head values on the displaced side of the fault and elevated heads on the other. This cross section is enlarged in Figure 4.2(c). It is clear that the contours are determined by the decrease in permeability in the fault zone and by the geometry of the aquifer. This will be even more apparent for $D_{max} = 2$.

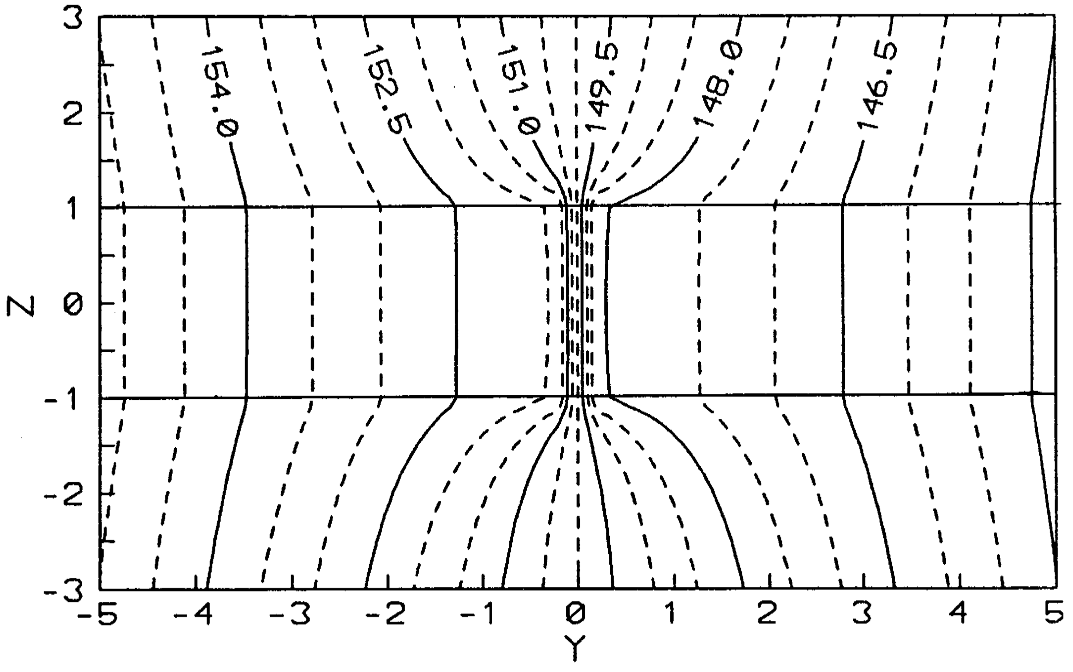
Figure 4.2(d) is the equivalent of 4.2(c) where $\mathbf{K}_{ef}$ is determined by the variable y_f distribution. Although $\mathbf{K}_{ef}$ is the same for both models at X = 0, the different distributions significantly impact the head contours at X = 0 as one might suspect based on the difference in values of maximum head gradient across the fault for the two models.

Figures 4.3 through 4.4 show an analogous set of contour plots for $D_{max} = 2$. As the aquifer thickness is 2 units, the aquifer is displaced the full thickness of the aquifer. The disturbance in the head field can be seen to be considerably greater than for $D_{max} = 0.5$. Because the $\mathbf{K}_{ef}$ values along the fault are the same as for $D_{max} = 0.5$, the head distribution is clearly quite sensitive to the geometry of displacement of the aquifer. One way to qualitatively separate the effects of geometry and those due to permeability variations along the fault is to run simulations of one without the other. Figure 4.5(a) is an X = 0 cross section with $D_{max} = 2$, but with no decrease in permeability along the fault so that head distortions are entirely due to displacement geometry. Figure 4.5(b) shows an X = 0 cross section with no displacement, but with $\mathbf{K}_{ef}$ determined with the variable K_f distribution. In both cases, there is considerable local head distortion, but one can see that the decreased permeability tends to cause considerably more head drop across the fault. The displacement tends to cause a considerable amount of head distortion and vertical gradients. Compare these Figures with Figures 4.2(a) and 4.4(a).

Figures 4.6(a) and (b) show contours for displacement of both walls with $\mathbf{K}_{ef}$ determined with the variable K_f distribution. Each wall is displaced by 1 unit giving a total displacement of 2 units. Simulations were also run with lesser

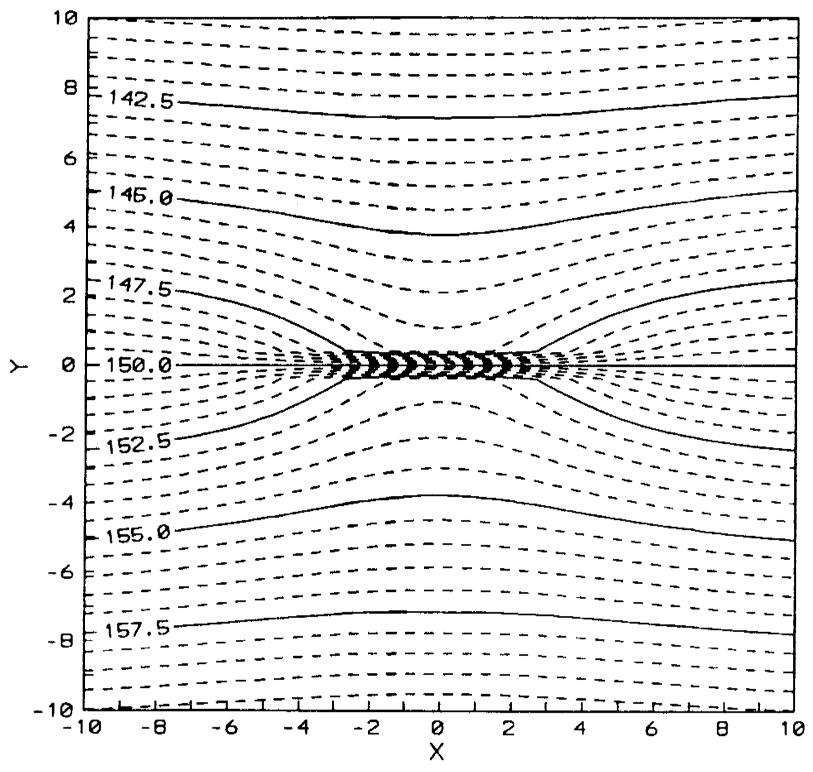


(a)

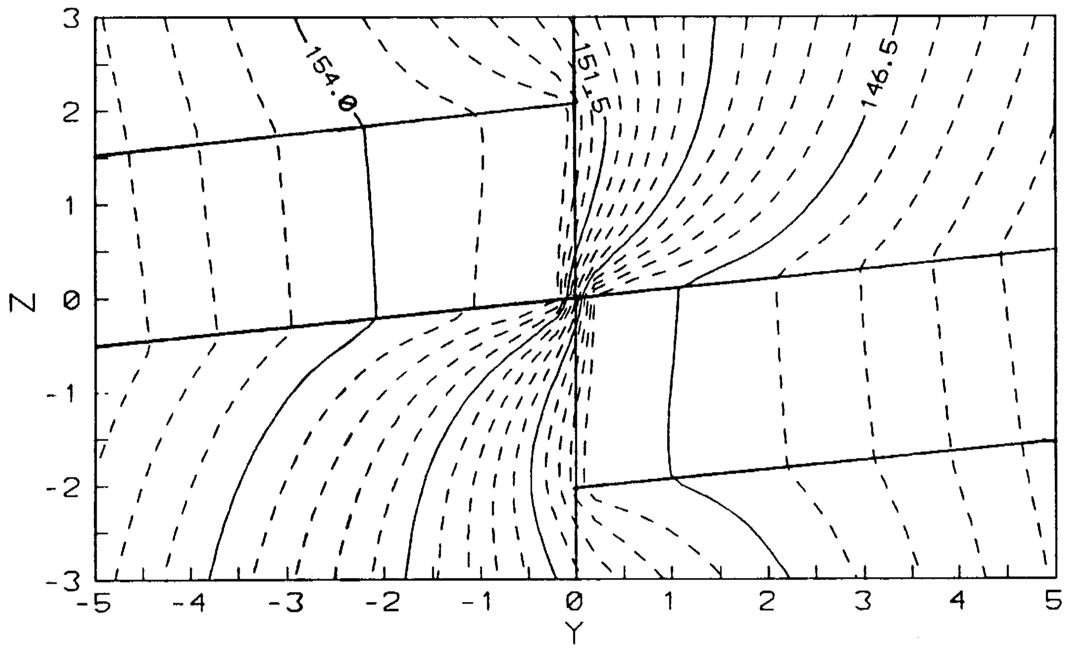


(b)

Figure 4.5 - (a) X = 0 cross section with displacement, but no change in fault K values; (b) X = 0 cross section with D_{max} = 0, but with y_f conductivity distribution.



(a)



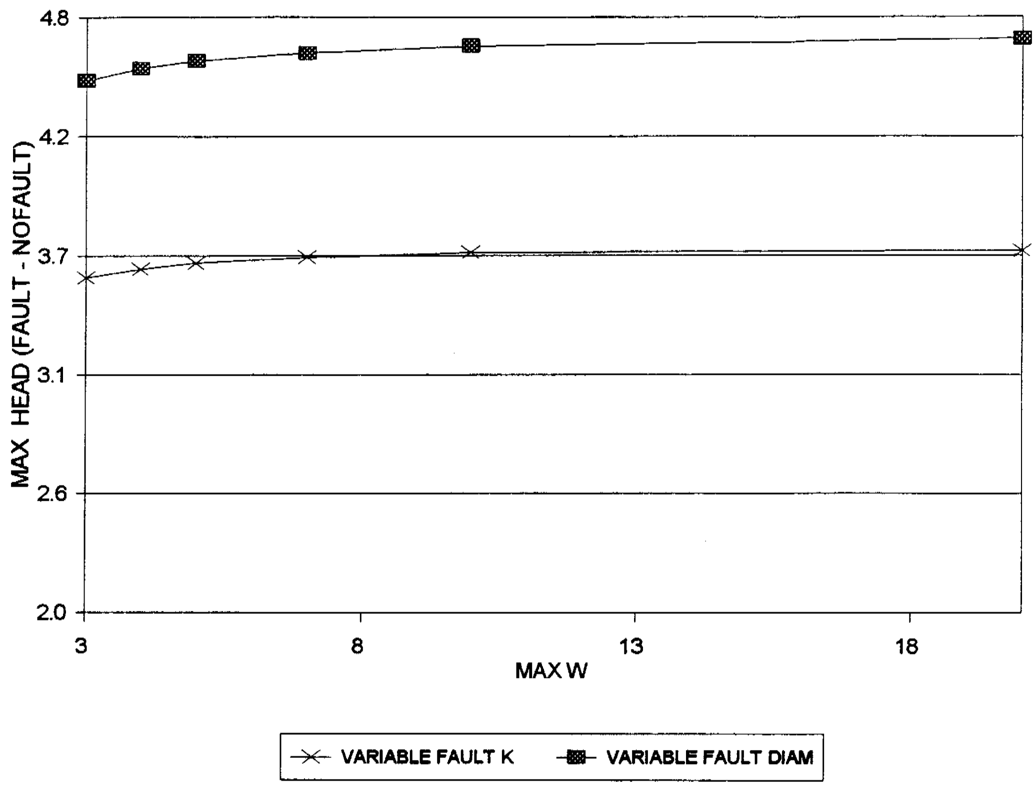
(b)

Figure 4.6 - Head contours with double fault-block displacement (a) XY cross section; (b) YZ cross section.

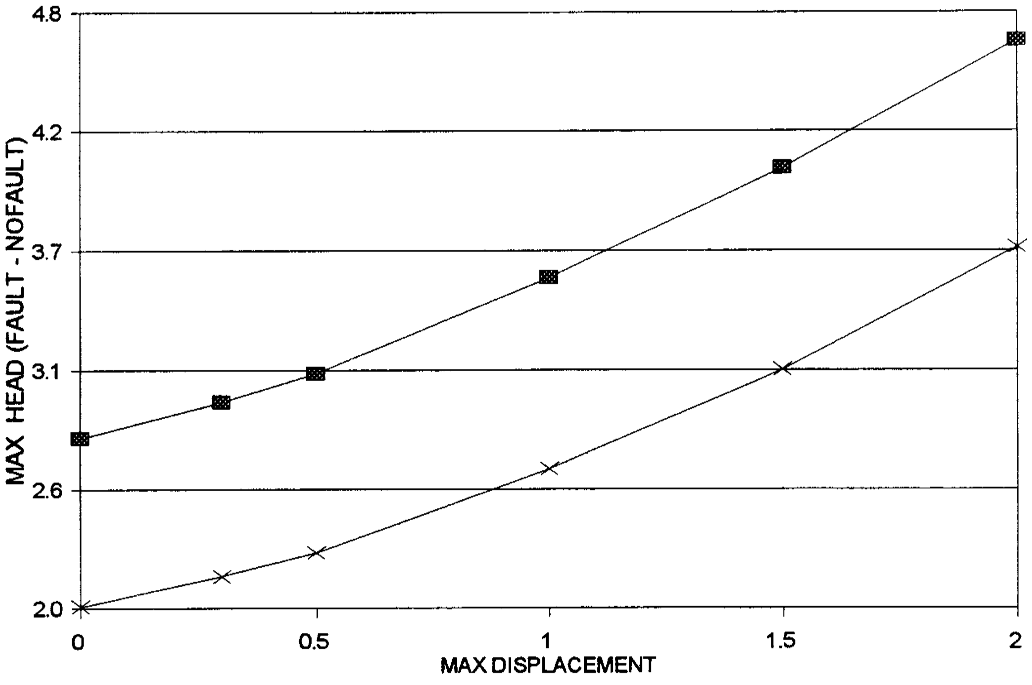
amounts of displacement. All the summary statistics described in Chapter 3 were calculated for these simulations. None differed from the statistics for the simulation with displacement in the hanging wall only (D_{max} = 2, variable K_f distribution) by more than 4%. Thus, based on these statistics, similar results are obtained whether one displaces one wall or both walls. However, one would expect the flow path of a particle to differ somewhat between these to simulations.

Figures 4.7 through 4.10 summarize the results for the simulations described in Table 3.1. Each figure is a graph of one of the summary statistics described in Chapter 3 versus either W_{max} (Figure (a)) or D_{max} (Figure (b)). In each figure, the summary statistics of the simulations using both models of $\mathbf{K}_{ef}$ variation are graphed. One can see that in each case, the change in effective conductivity model results in a vertical shift of the curve; that is, the trend is essentially the same, but the range changes (the shape of the curves are the same except for the case of vertical gradient were the two curves vary in shape somewhat). The shift is fairly large with the maximum Y gradient across the fault shifting by as much as 10 units. This implies that the head field is fairly sensitive to the variation in permeability along the fault.

Perhaps of most consequence is that all of these statistics are rather sensitive to the amount of displacement, but not very sensitive to W_{max}. For example, in the case of

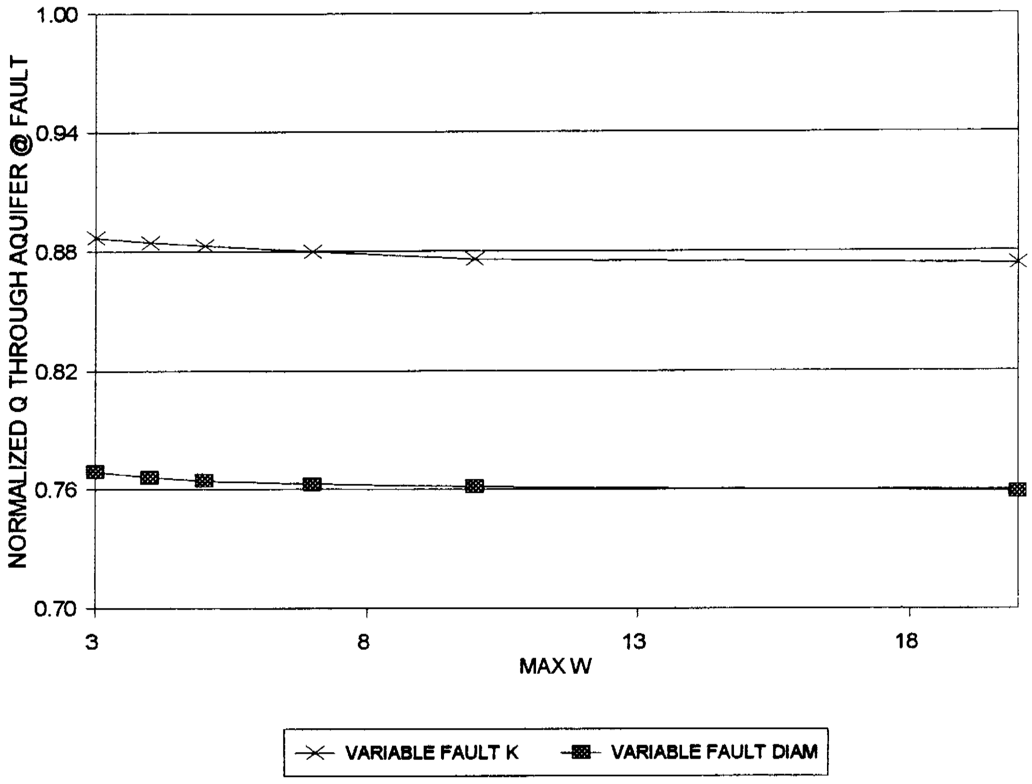


(a)

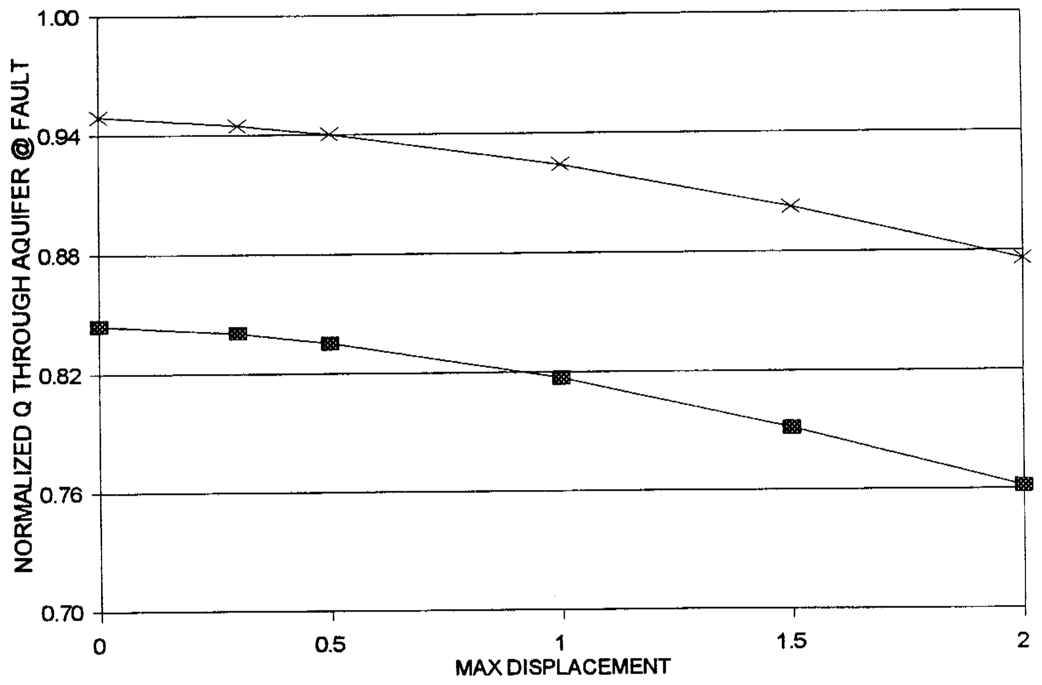


(b)

Figure 4.7 - Maximum head difference between fault and no-fault simulations versus (a) W_{max} and (b) D_{max}.

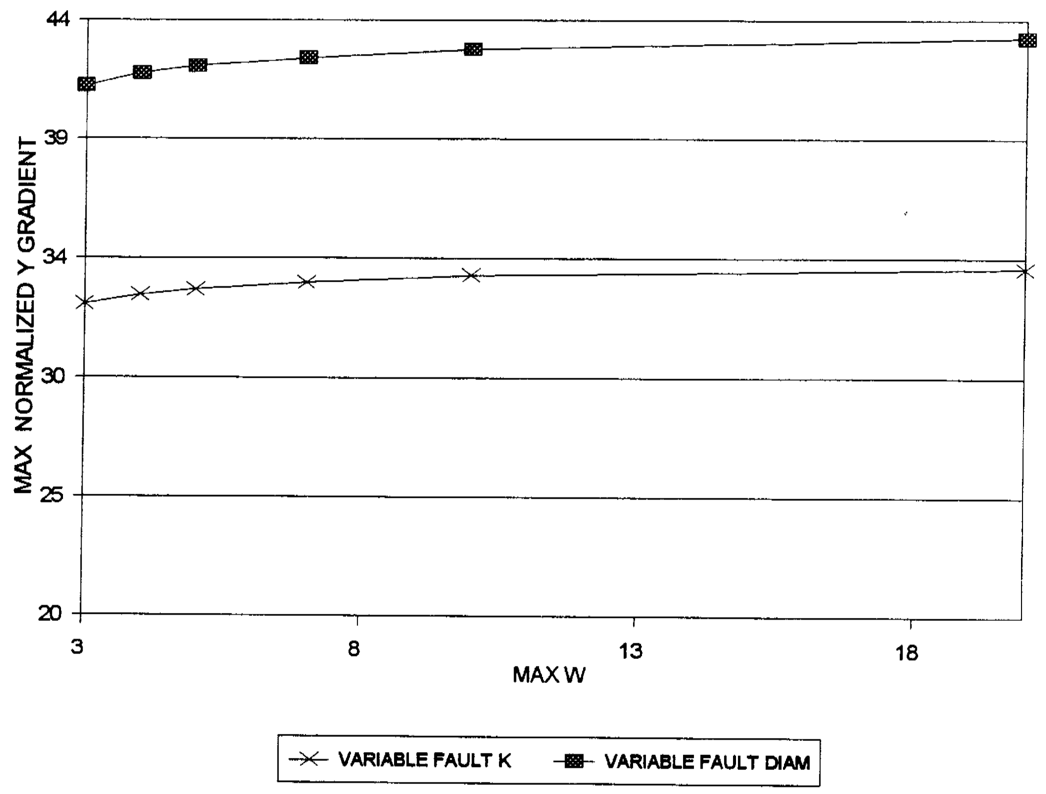


(a)

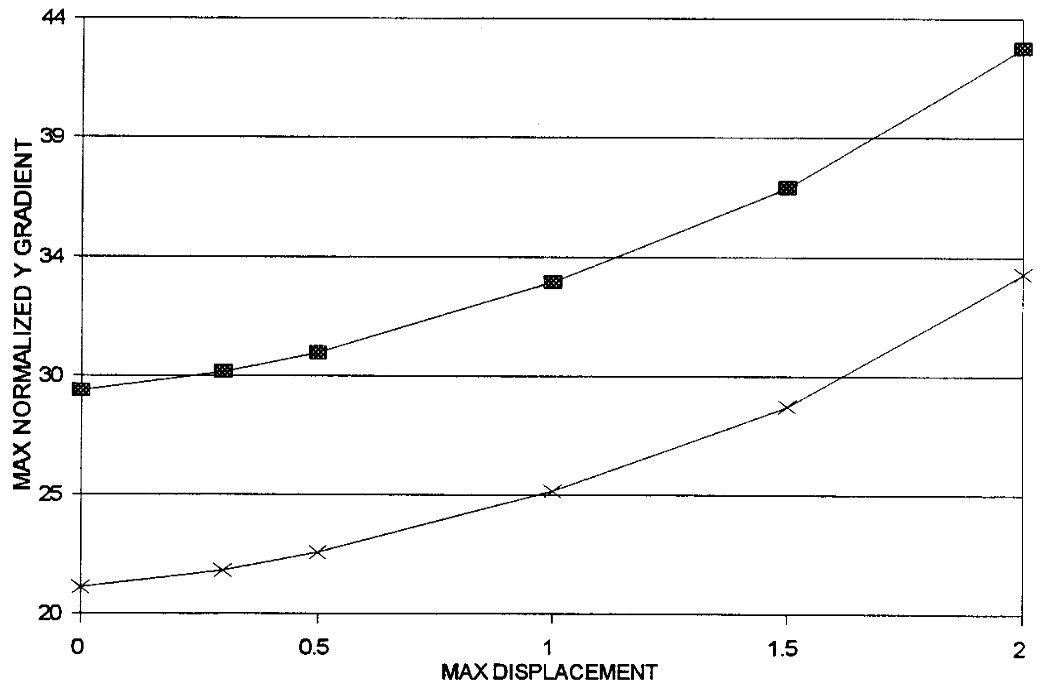


(b)

Figure 4.8 - Flow through aquifer at the fault versus (a) W_{max} and (b) D_{max}.

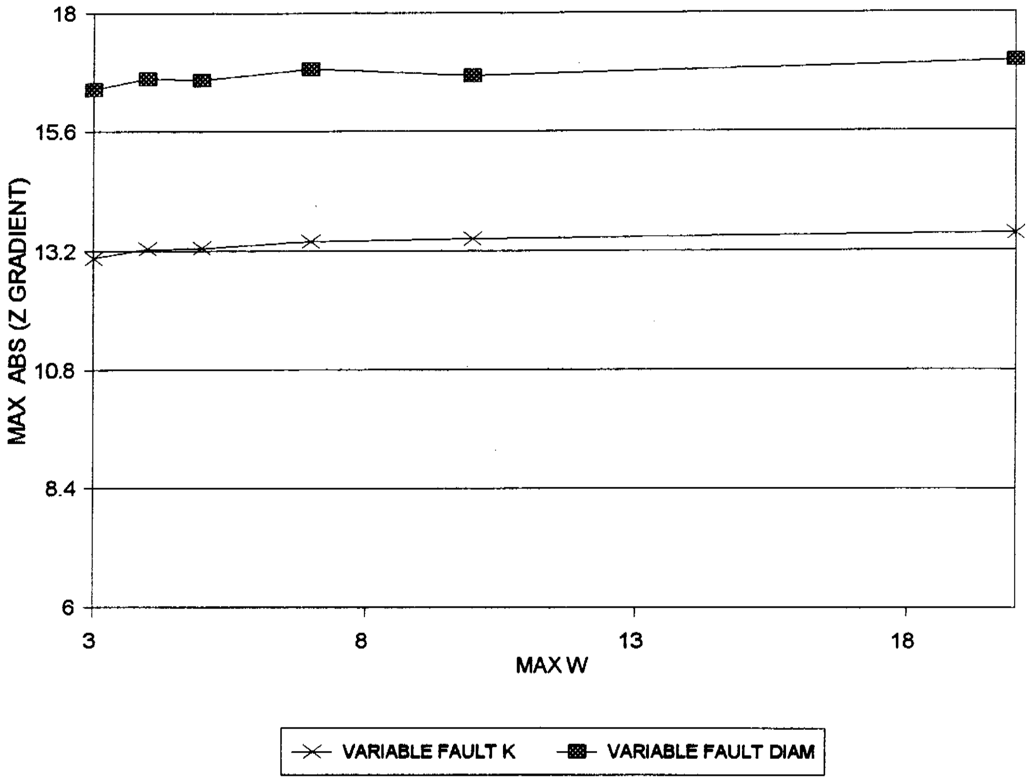


(a)

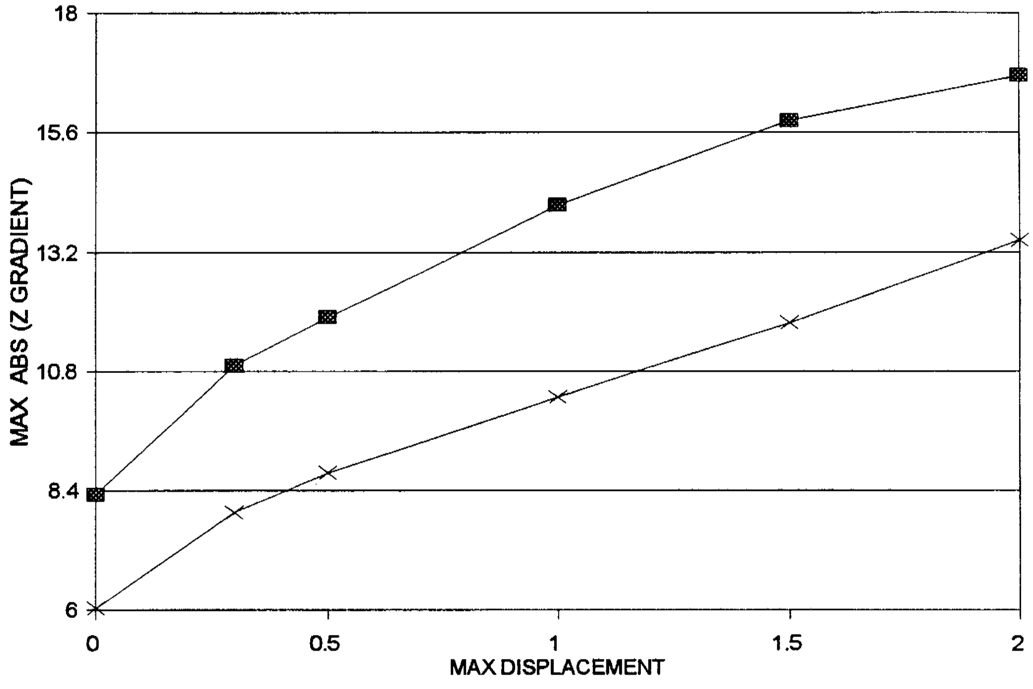


(b)

Figure 4.9 - Maximum Y gradient magnitude versus (a) W_{max} and (b) D_{max}.



(a)



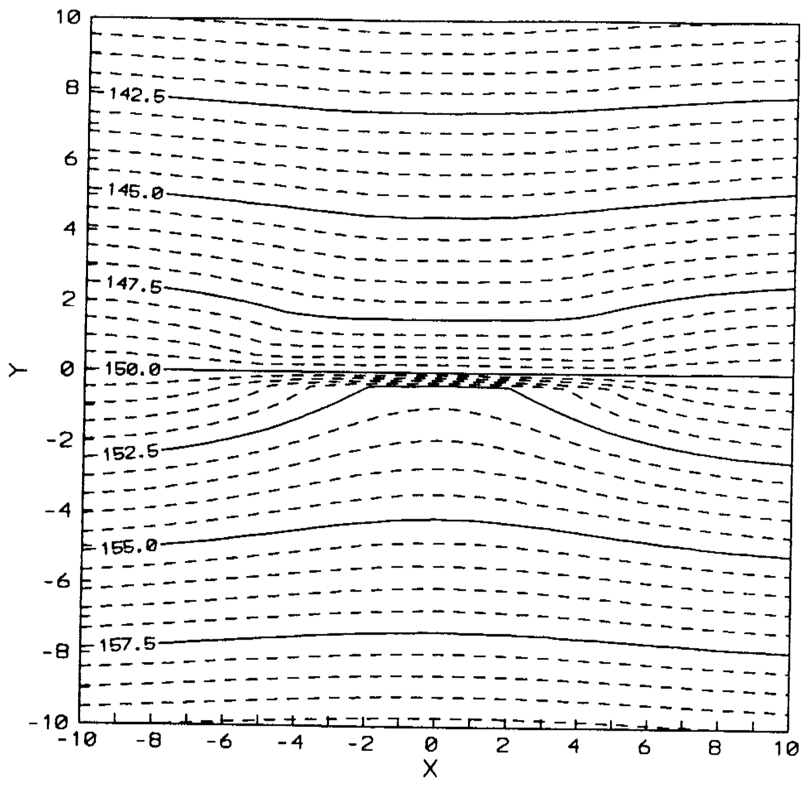
(b)

Figure 4.10 - Maximum of absolute value of Z gradient versus (a) W_{max} and (b) D_{max}.

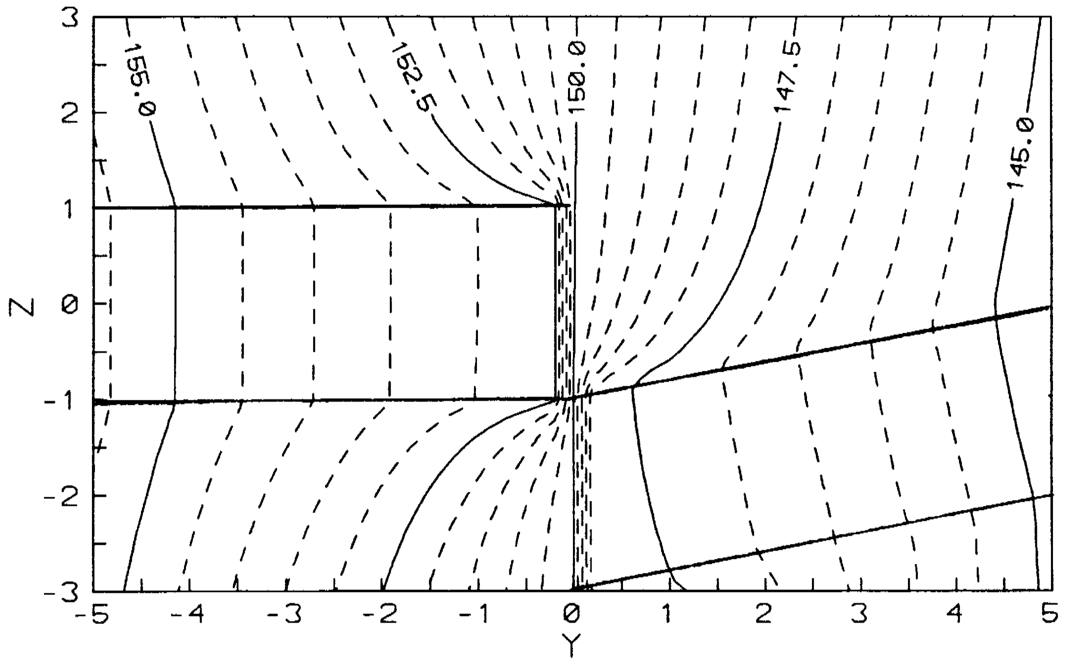
maximum vertical gradient, there is a 125% variation over the range of D_{max} values, but only a 3% variation over the range of W_{max} values. Inspection of Figure 4.5(a) shows that head contours are indeed impacted by reverse drag. However, gradients in the vicinity of the fault plane are not very sensitive to W_{max}.

Another possible statistic would be maximum X gradient. This gradient would be largely dependent on, and proportional to, the ratio K_{max}/K_{min} in the aquifer. As this ratio was not varied in the simulations, this statistic was not computed. One can, however, clearly observe, in the XY cross sections, significant flow of groundwater around the fault in response to the decrease in permeability in the fault zone. In the limit of $K_{min} = 0$, flow just either side of the fault centroid would be parallel to the fault plane.

Figures 4.11(a) and (b) show Z = 0 and X = 0 cross sections, respectively, for the case of enhanced X and Z permeability with $\theta = 90°$, $D_{max} = 2$, $W_{max} = 10$ and $KF_{max} = 1000$. This cross section should be compared with Figures 4.3 and 4.4 where $KF_{max} = 1$ (as defined in equation 3.3). There is considerably less deflection of the contours with enhanced permeability as there is increased flow along the fault plane. Figure 4.12 shows graphs of the summary statistics versus the natural log of KF. The decreased gradients reflect what was observed in the cross sections. Head contours appear to be mostly affected by geometry; significant vertical flows can

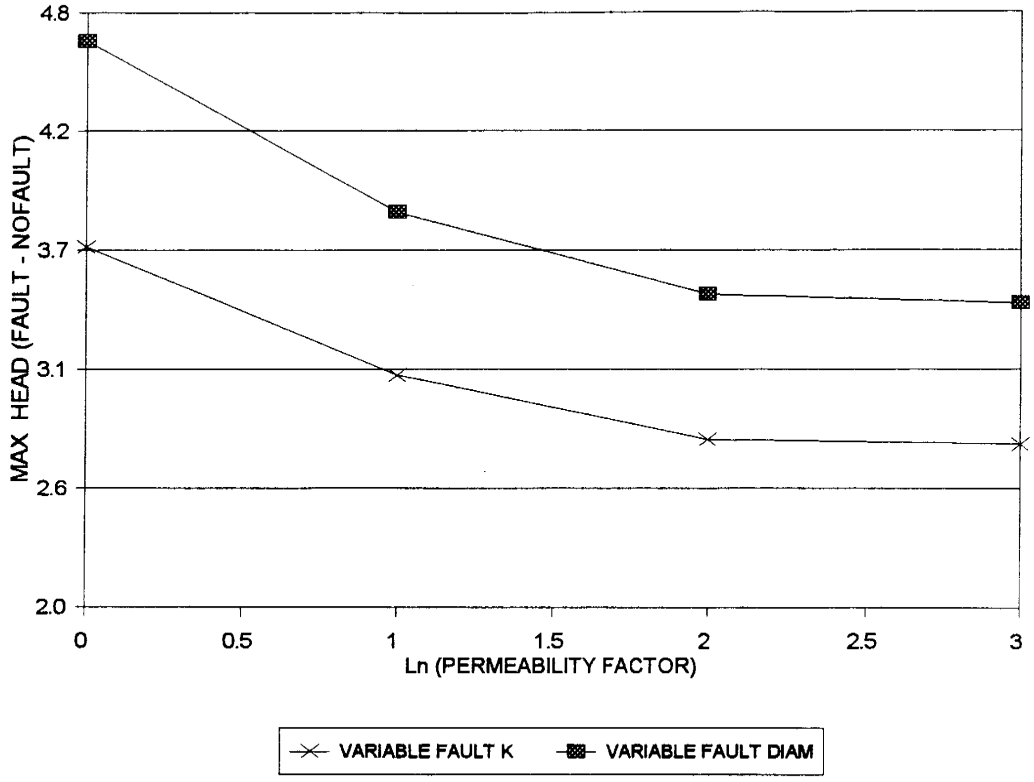


(a)

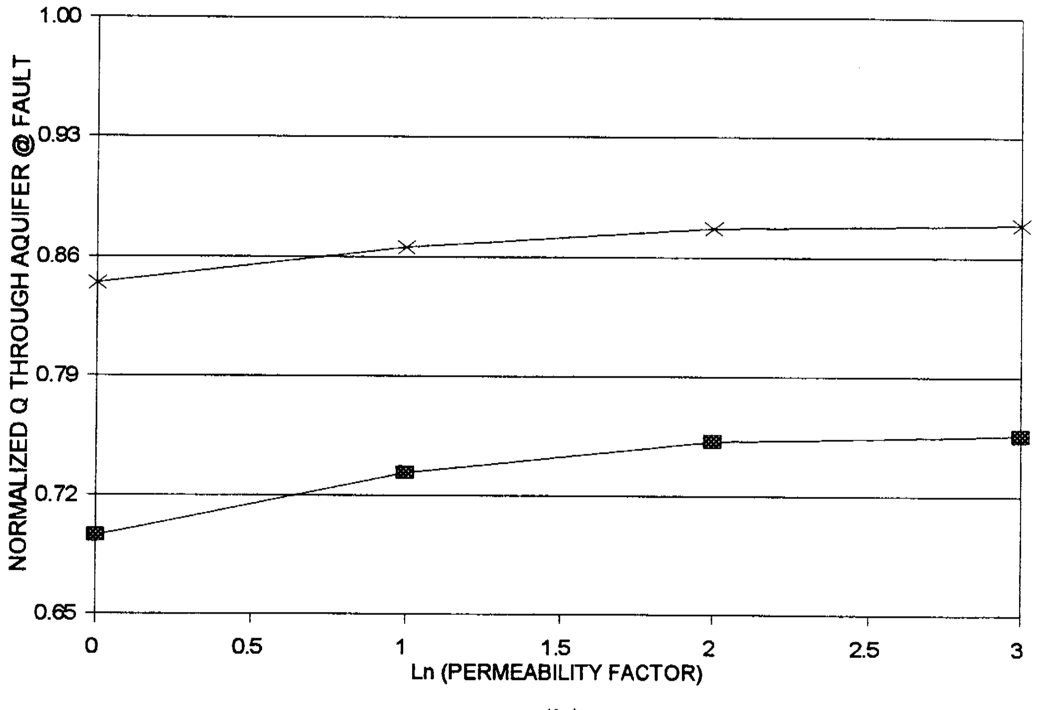


(b)

Figure 4.11 - D_{max} = 2, variable K_f distribution, KF_{max} = 1000 (a) XY cross section; (b) YZ cross section.

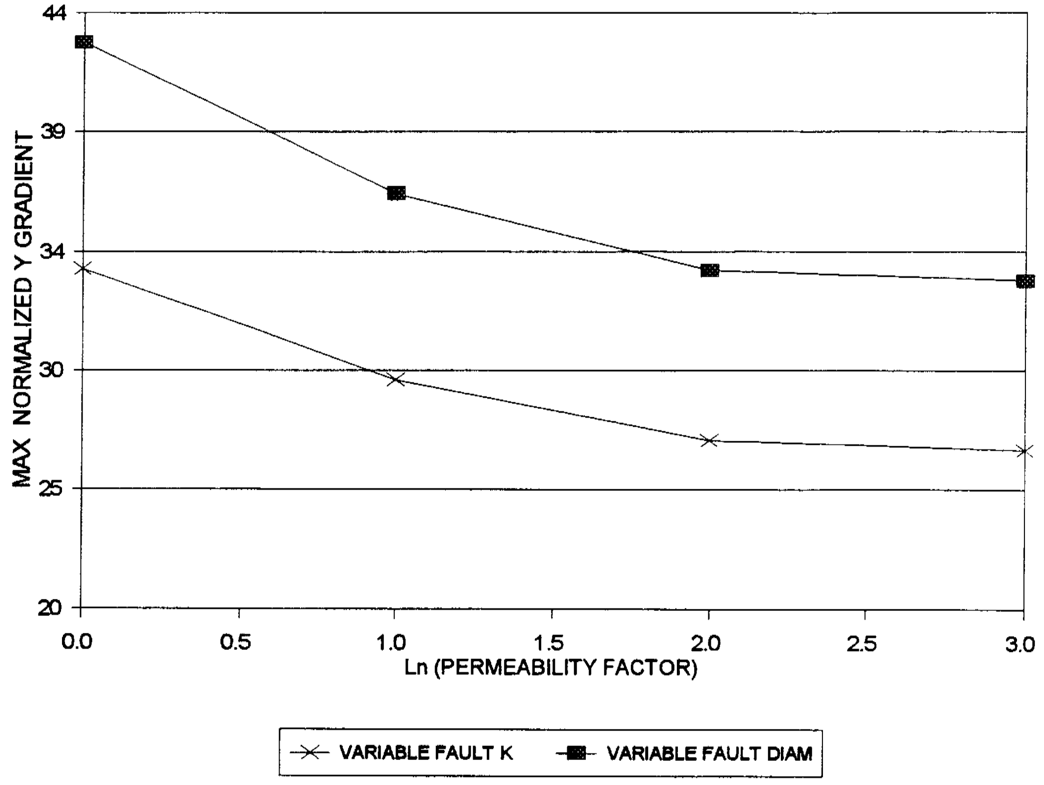


(a)



(b)

Figure 4.12 - Log of KF_{max} versus (a) Maximum head difference between fault and no fault simulations; (b) Q through aquifer at fault; (c) Maximum Y gradient.


MAX NORMALIZED Y GRADIENT
44
39
34
30
25
20
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Ln (PERMEABILITY FACTOR)
VARIABLE FAULT K
VARIABLE FAULT DIAM

(c)

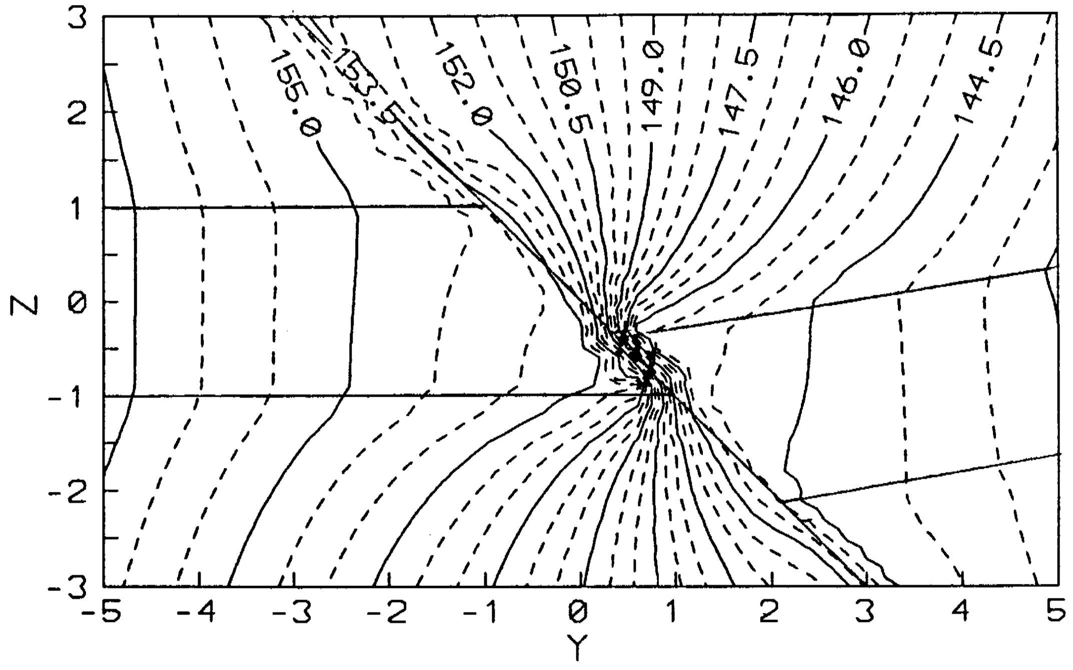
result despite relatively small vertical gradients due to the greatly enhanced vertical hydraulic conductivity.

Figures 4.13(a) and (b) show X = 0 cross sections for a 90° and a 45° fault, respectively. For both fault angles $D_{max} = 1.8$ and the effective conductivity distribution is determined by the variable fault conductivity model. One can see once again that head contours significantly reflect the physical geometry of the fault zone as well as the hydrologic characteristics of the region. Table 4.2 gives the comparison of the normalized summary statistics for these two cases.

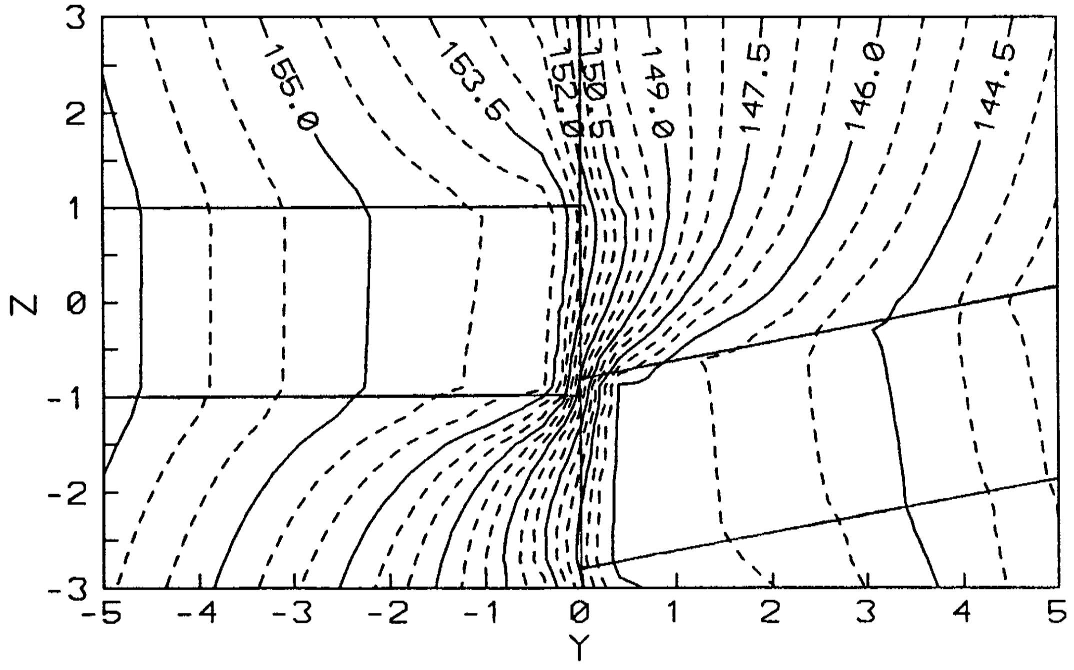
TABLE 4.2
Statistics for 90° and 45° Simulations

DIP	D_{max}	MAX ΔH	MAX Y GRAD	MAX Z GRAD
45°	0.9	3.6	12.5	-11.5
45°	1.8	4.1	14.7	-13.7
90°	0.9	3.7	13.1	6.1
90°	1.9	4.4	14.9	7.8

From Table 4.2, we see that the maximum vertical head gradient is rather sensitive to fault dip, whereas the other statistics do not show much sensitivity to fault dip. This implies that, depending on one's objectives, one may reasonably ignore fault dip when including simple faults in a groundwater model. If, on the other hand, one is doing three-dimensional particle tracking, fault dip can clearly make a significant difference.



(a)



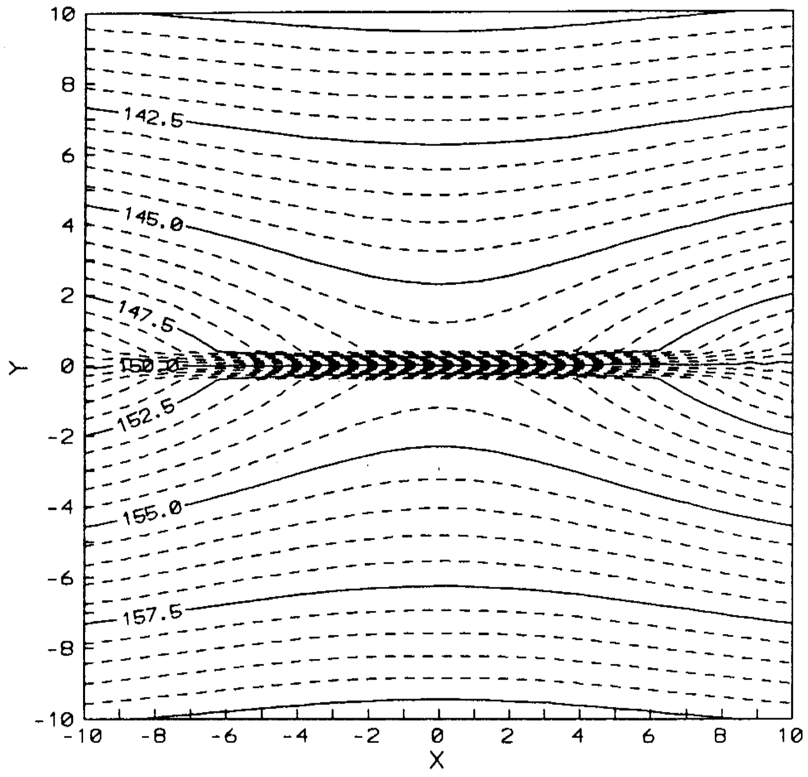
(b)

Figure 4.13 - YZ cross sections for (a) θ = 45° and (b) θ = 90°. D_{max} = 1.8, W_{max} = 10 and variable K_f distribution.

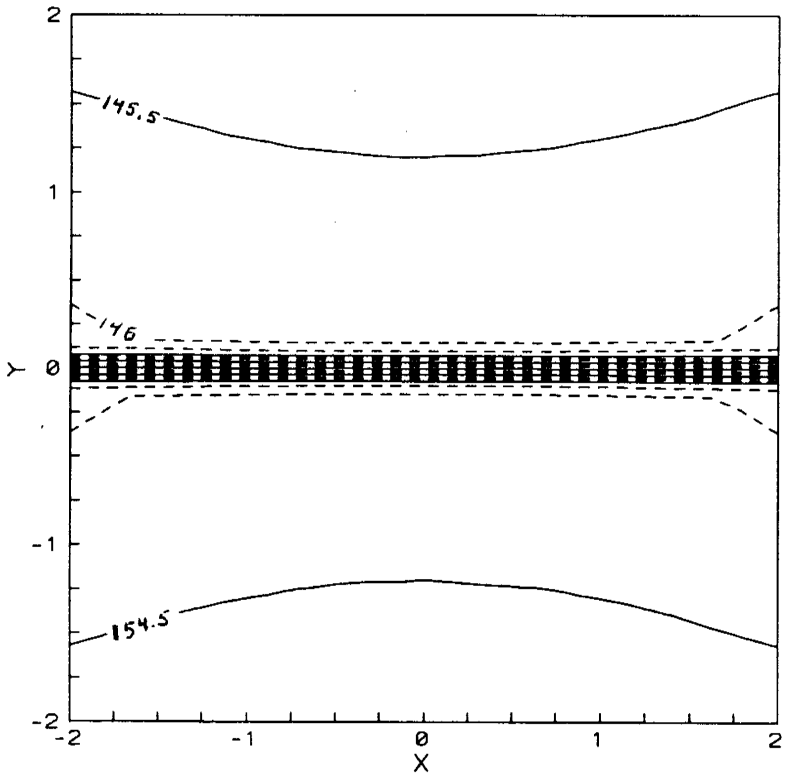
Figures 4.14(a) and (b) show contour maps interpolated from the heads in a dense set of wells, one well for each cell, 4.14(b) being over a smaller domain. The variable y_f model is used to determine the $\mathbf{K}_{ef}$ distribution. Figures 4.15 show contours interpolated from the heads in varying numbers of wells randomly generated over the XY domain defined by $-10 \le X \le 10$, $-10 \le Y \le 10$. Figures 4.16 are similar except that they are over the XY domain $-2 \le X \le 2$, $-2 \le Y \le 2$. Each well location is indicated by an asterisk. For both domains, there are two contour maps each for sets of 5, 10 and 30 wells (several more sets of wells were generated, but those shown are representative). The contour plots of Figures 4.15 and 4.16 are then an approximation of the contour plots of Figure 4.14.

It should be noted that there are diversions from the actual contour pattern due to the contour interpolation algorithm in areas with few or no wells. This can take the form of contour lines curving away from the correct direction or concentric contours as in 4.15(b) and 4.16(f). Also note that some bias can be detected in the random number generator by observing the well placements in Figures 4.15 and 4.16.

It can be seen that with a very low density of wells, the actual variation in gradient is largely lost. Additionally, if the wells tend to be aligned in a particular direction, the apparent gradient may be in that direction as can be seen in Figures 4.15(a), (b) and (d). In Figures 4.15(a) and (b), with an average well density of 5 wells per 100 square units

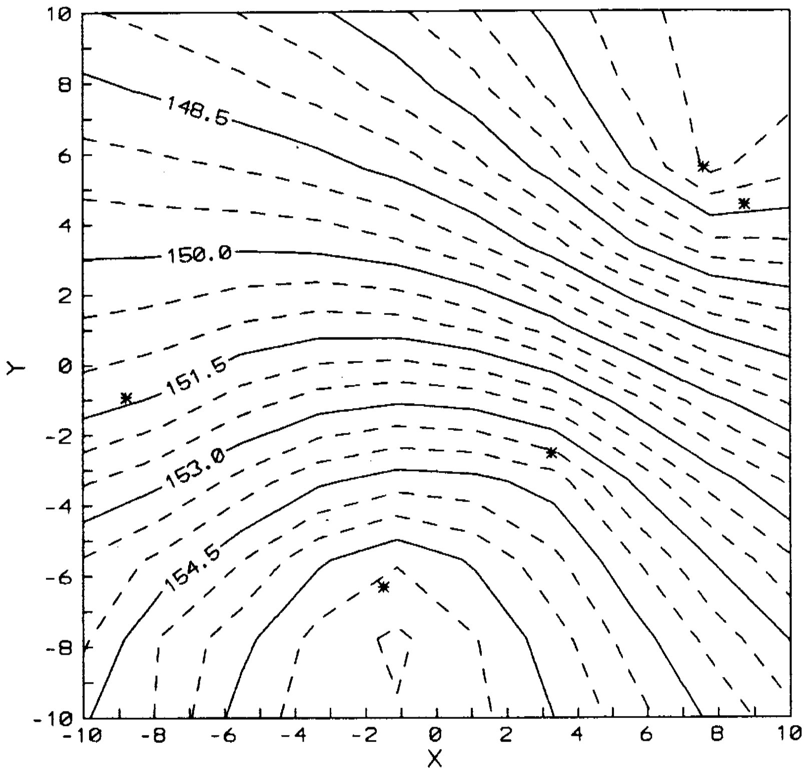


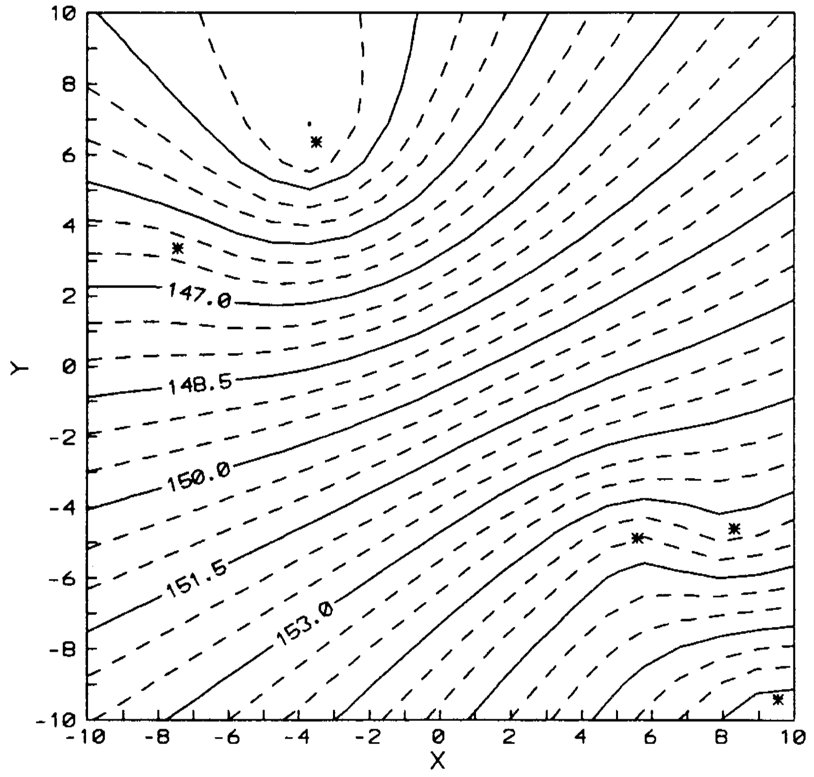
(a)



(b)

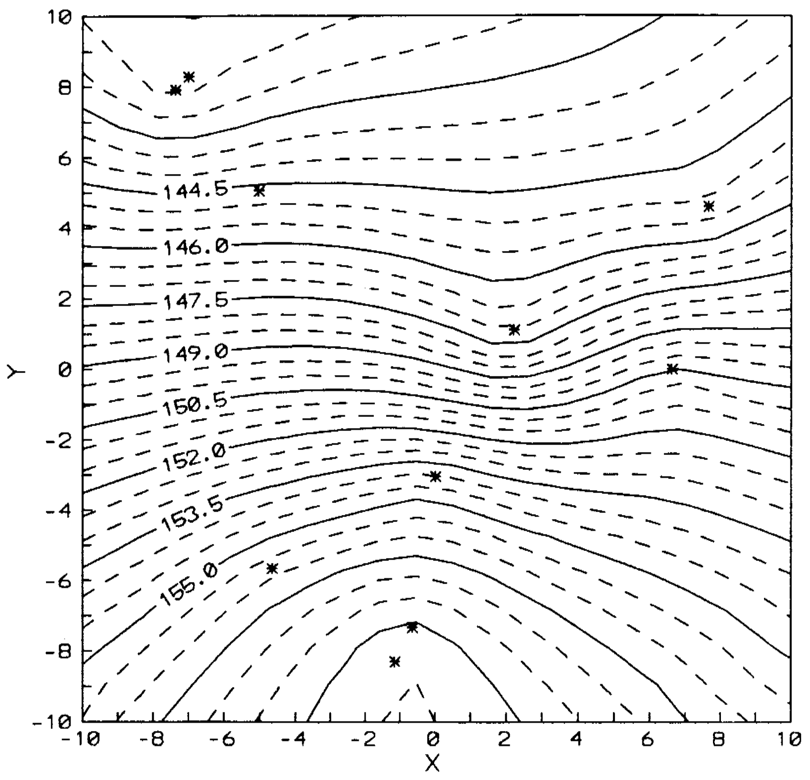
Figure 4.14 - (a) Contours of heads from a dense set of wells; (b) highlight of (a).

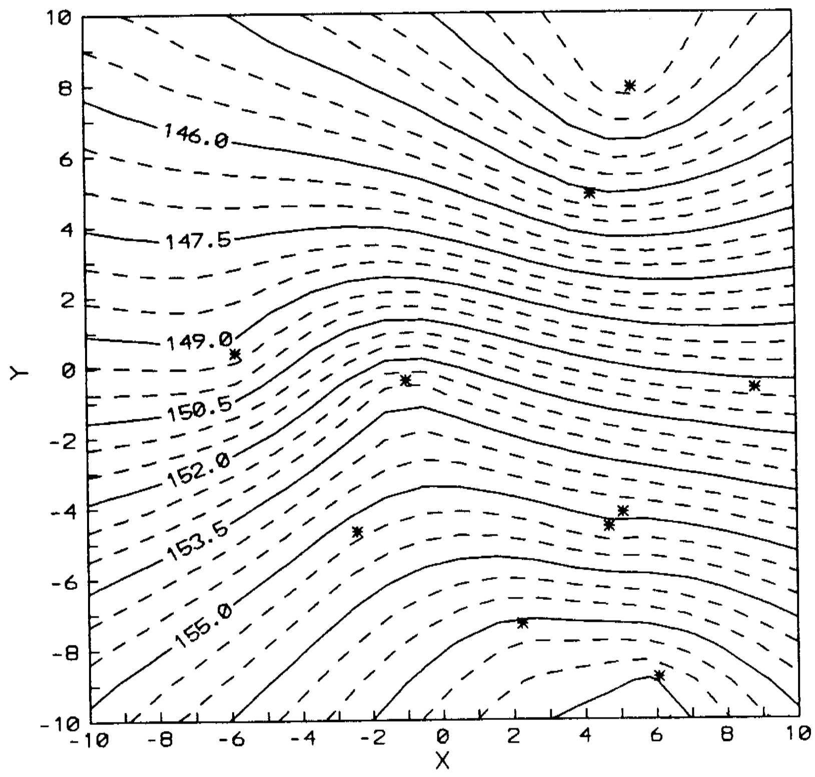




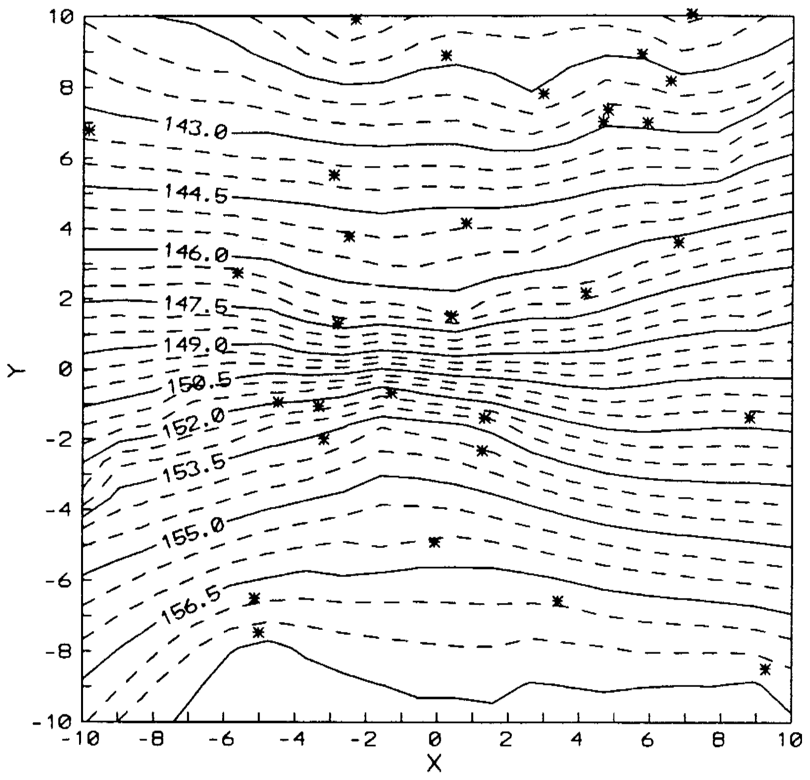
(a)

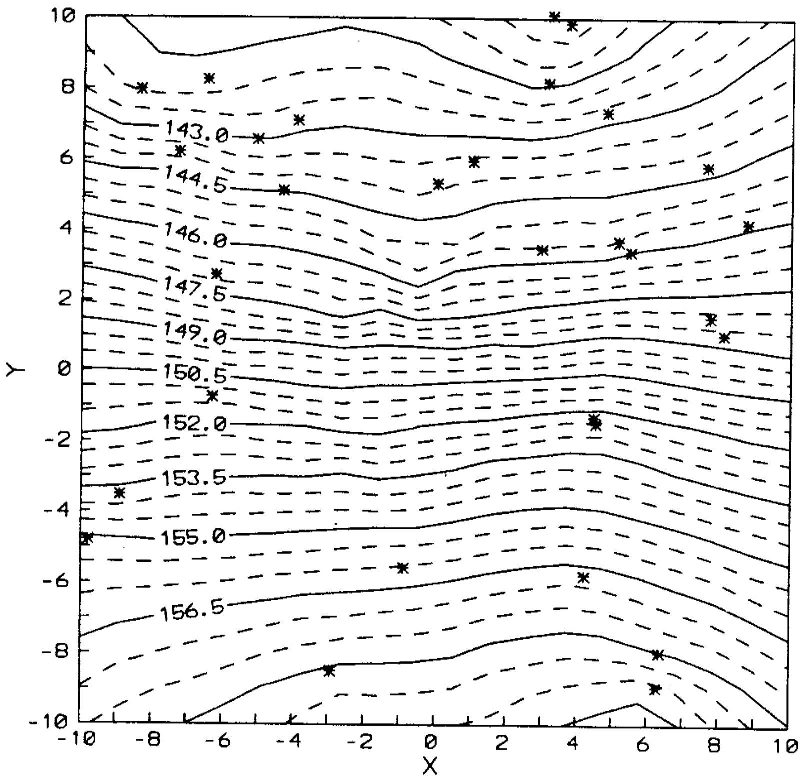
Figure 4.15 - Contours of sparse well data from the following well densities (a) 5 wells/400 units^2; (b) 10 wells/400 units^2; (c)30 wells/400 units^2.


144.5
146.0
147.5
149.0
150.5
152.0
153.5
155.0
Y
X

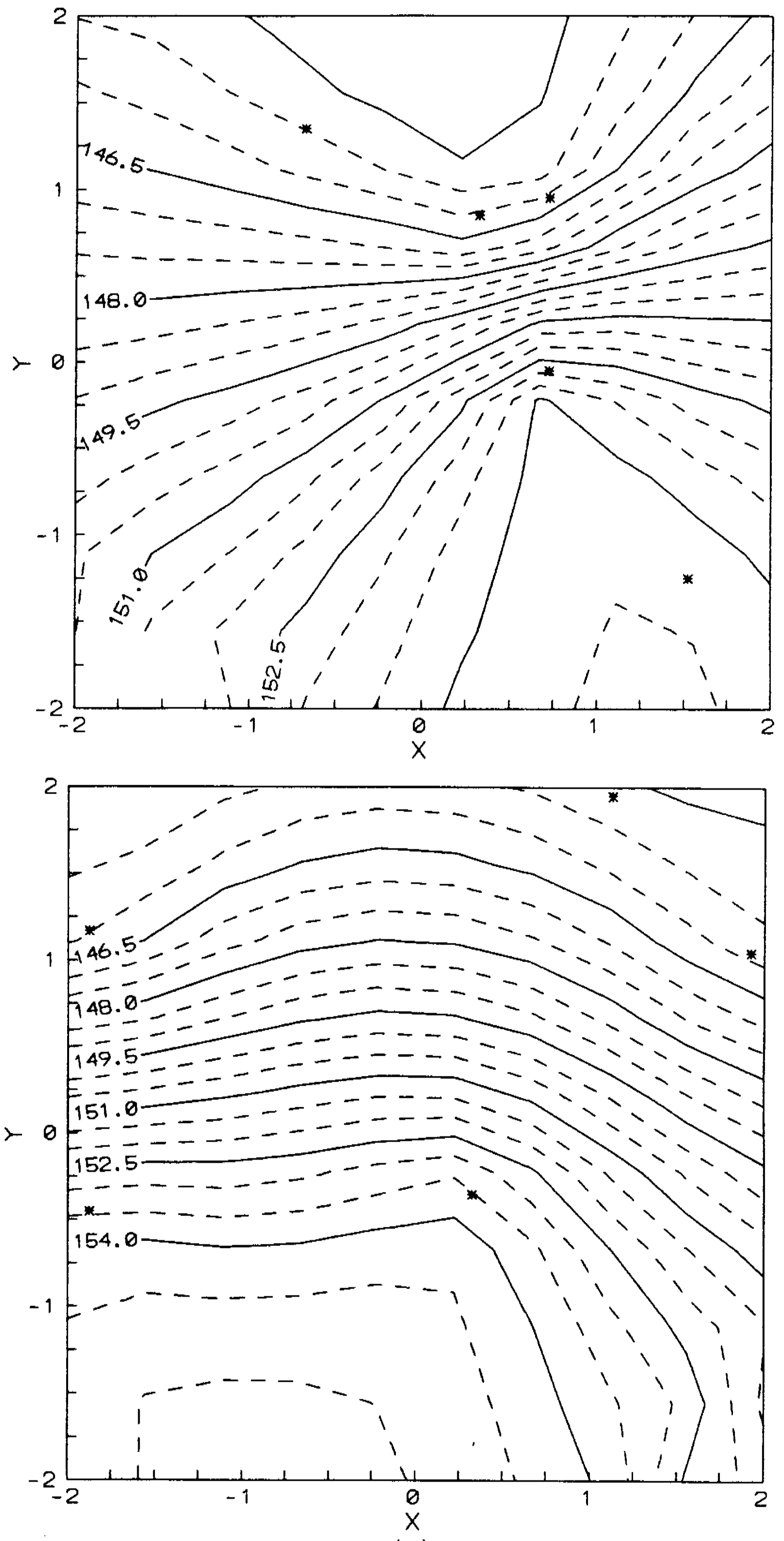

146.0
147.5
149.0
150.5
152.0
153.5
155.0
Y
X

(b)


143.0
144.5
146.0
147.5
149.0
150.5
152.0
153.5
155.0
156.5
X
Y

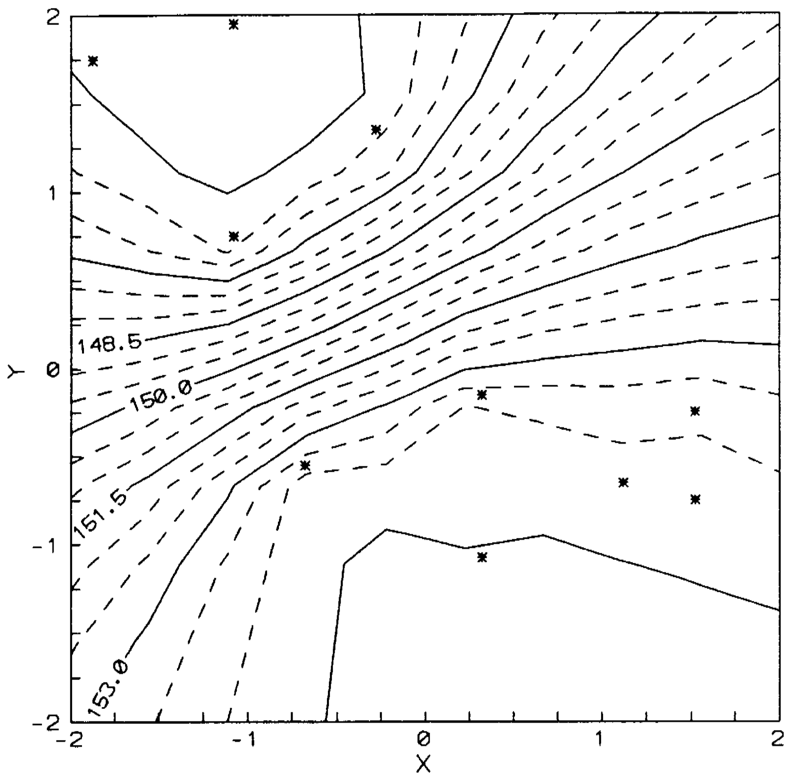

143.0
144.5
146.0
147.5
149.0
150.5
152.0
153.5
155.0
156.5
X
Y

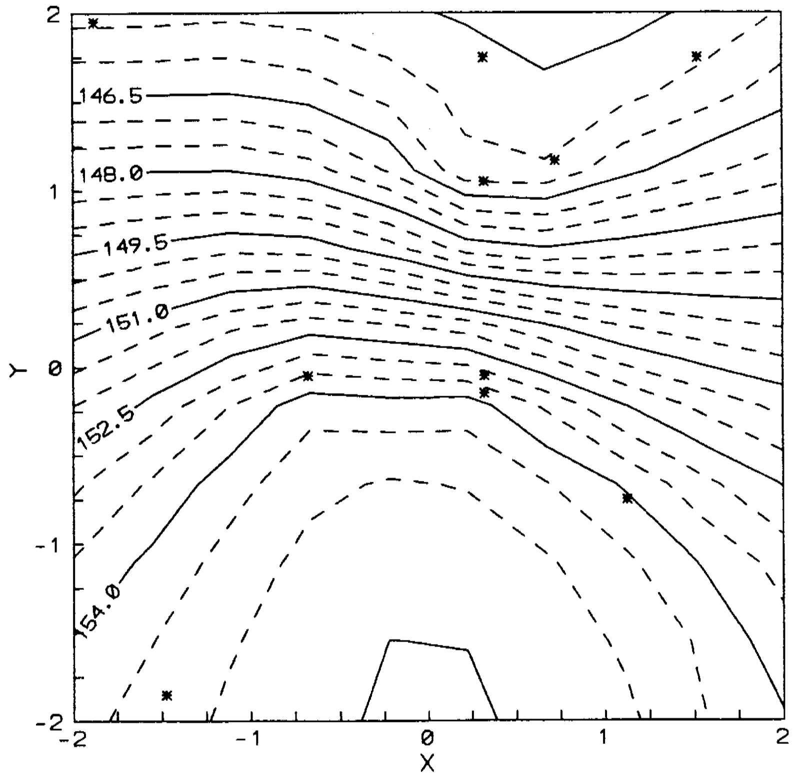
(c)



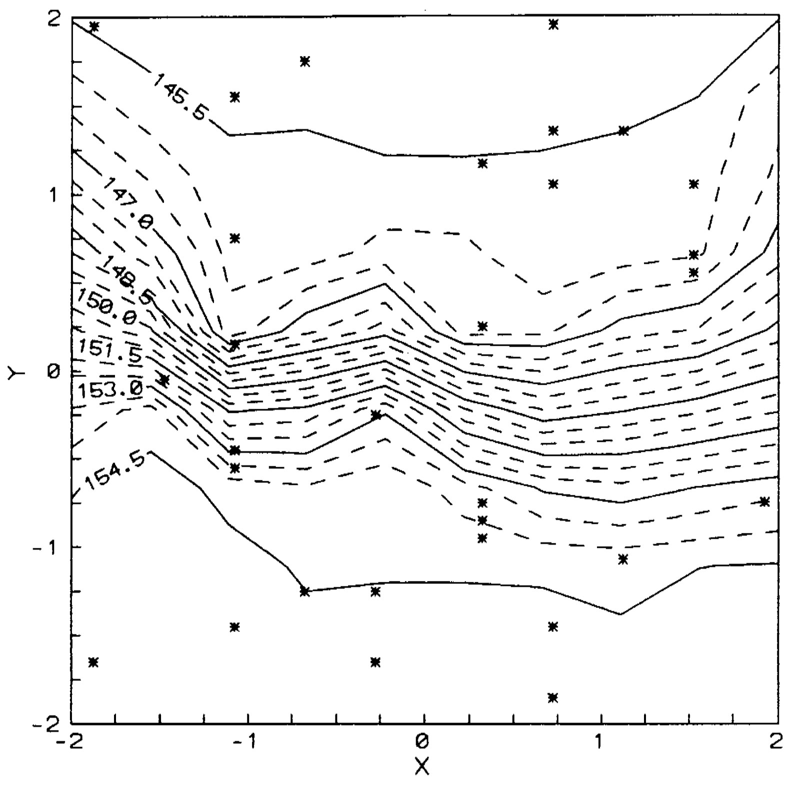
(a)

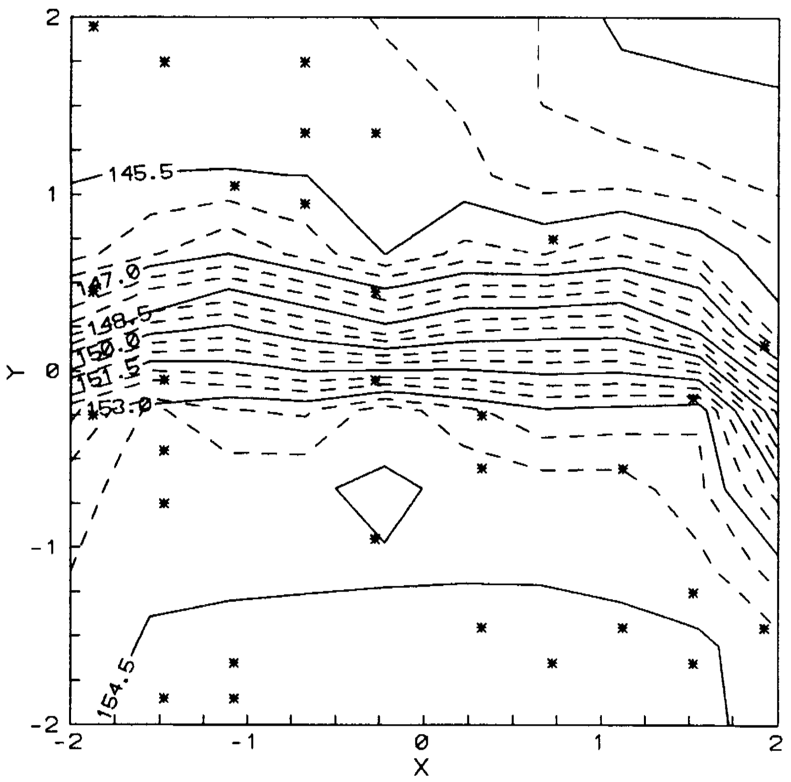
Figure 4.16 - Contours of sparse well data from the following well densities (a) 5 wells/16 units^2; (b) 10 wells/16 units^2; (c) 30 wells/16 units^2.


148.5
150.0
151.5
153.0
Y
X
2
1
0
-1
-2
-2
-1
0
1
2


146.5
148.0
149.5
151.0
152.5
154.0
Y
X
2
1
0
-1
-2
-2
-1
0
1
2

(b)


145.5
147.0
148.5
150.0
151.5
153.0
154.5
Y
X


145.5
147.0
148.5
150.0
151.5
153.0
154.5
Y
X

(c)

centered about the fault, there is really no indication that there may be some variation in gradient and, thus, no indication of the geologic structure.

When the well density is doubled to 10 wells per 100 square units, there is still little visual indication of the actual gradient variation, as can be seen in Figures 4.15(c) and (d). A slight increase in gradient toward the centroid of the fault can be seen, but not enough to indicate that the gradient increases significantly toward the second. In all likelihood, one could distribute 10 wells over the domain in such a way that there would be an indication of the actual gradient variation occurring. However, in the 10 sets of 10 wells generated over this domain, it was not the case. It is intuitive that the gradient variation will be largely lost unless there is a sufficient number of wells in the vicinity of that portion of the fault where the gradient is greatest and in areas of smaller gradients.

Figures 4.15(e) and (f) show two examples of contour plots for an average well density of 30 wells per 400 units^2. Figure 4.15(e) has considerably more wells in the vicinity of the fault's centroid and the contours show a clear gradient increase in this area. On the other hand, in Figure 4.15(f) there are no wells near the fault centroid and so there is no indication of the fault's effects.

In Figures 4.16 the domain is $-2 \leq X, Y \leq 2$. The average well density in Figure 4.16(a) and (b) is 5 wells per 16 square units centered about the fault centroid, in Figures 4.16(c) and (d) is 10 wells per 16 square units and in Figures 4.16(e) and (f) is 30 wells per 16 square units. In most of these contour plots, some variation in gradient is apparent, but this variation does not reflect the actual variation very well until the density reaches 30 wells per 16 square units. Again, with a random well distribution, there must be enough wells so that an adequate number occur in the vicinity of the fault and away from the fault.

To more quantitatively consider the effects of the ideal fault on groundwater flow apparent from sparse well data, sets of 200 wells were generated over an area of 400 units^2 (average well density of 0.5 wells/unit^2) and over an area of 16 units^2 (average well density of 12.5 wells/unit^2). Gradients were calculated from all sets of three non-collinear wells. Table 4.3 summarizes the results for the gradient magnitudes (gradients are normalized by the no-fault Y gradient).

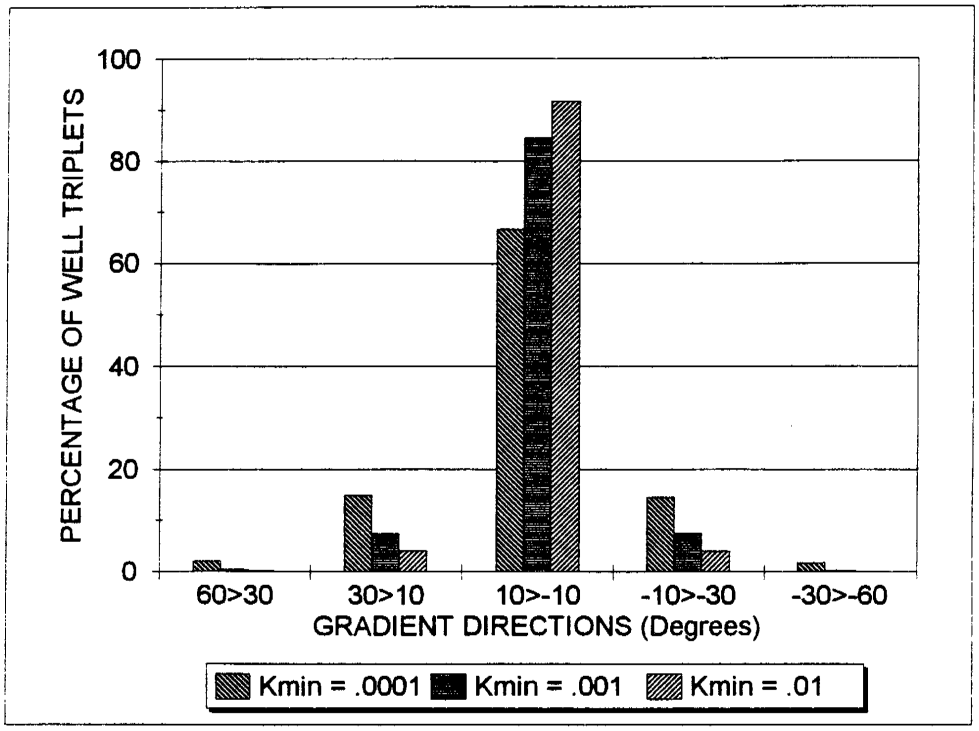
TABLE 4.3
Gradients/Std. Dev. for Various Simulations

K_{min}	WELL DENSITY	95%	5%	MAX	MIN
.0001	0.5	2.03/.084	.75/.028	13.3/3.5	.25/.128
.001		1.48/.049	.86/.013	7.7/1.9	.38/.063
.01		1.32/.033	.90/.01	5.7/1.54	.49/.063
.0001	12.5	9.60/.26	.40/.006	50.5/.65	.083/.083
.001		5.74/.061	.52/.002	29.9/1.4	.055/.1
.01		4.37/.042	.60/.002	28.8/3.5	.24/.038

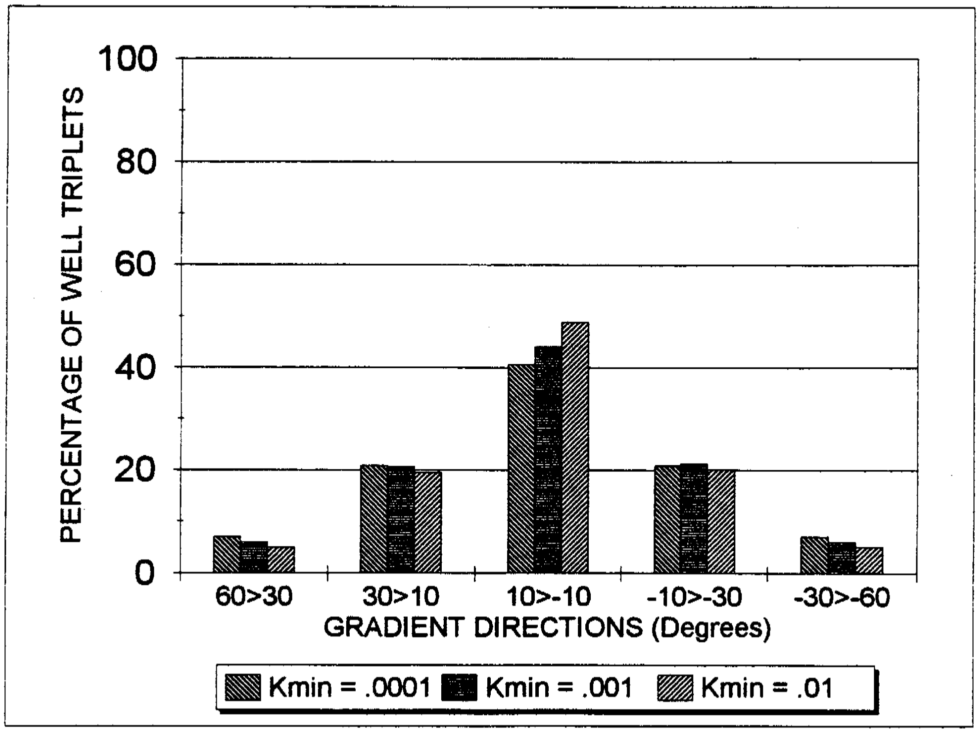
The actual maximum normalized gradient in the Y direction is approximately 43 (as determined from cells on either side of the fault, see Figure 4.9) for the case of K_{min} = .0001. The maximum gradient calculated from the 200 wells is considerably smaller than this. The 90% confidence interval for an average well density of 0.5 wells/unit^2 is (.75, 2.03). Thus, the probability of a small number of wells in the vicinity of the fault with an average well density of 0.5 wells/unit^2 indicating the large gradient magnitudes occurring near the fault is much less than 0.05.

When the well density is increased to 12.5 wells/unit^2, with K_{min} = .0001, the 90% confidence interval becomes (.4, 9.6) and the maximum gradient magnitude is 50.5 (which is larger than the actual maximum Y gradient because this gradient is not constrained to be in the Y direction). The maximum is indeed a reflection of the large gradients near the fault, but the 95% confidence level shows that there is less than a 5% probability of detecting a gradient magnitude larger than 9.6.

Figures 4.17(a) and (b) show histograms of gradient directions for well densities .5 wells/unit^2 and 12.5 wells/unit^2, respectively. The gradient directions are given in ranges (e.g., 30>10 means all directions between 30° and 10° from the Y direction). For the case of the lower well density and K_{min} = .0001, there is a 66.7% probability of the gradient direction being within 10° of the average direction,



(a)



(b)

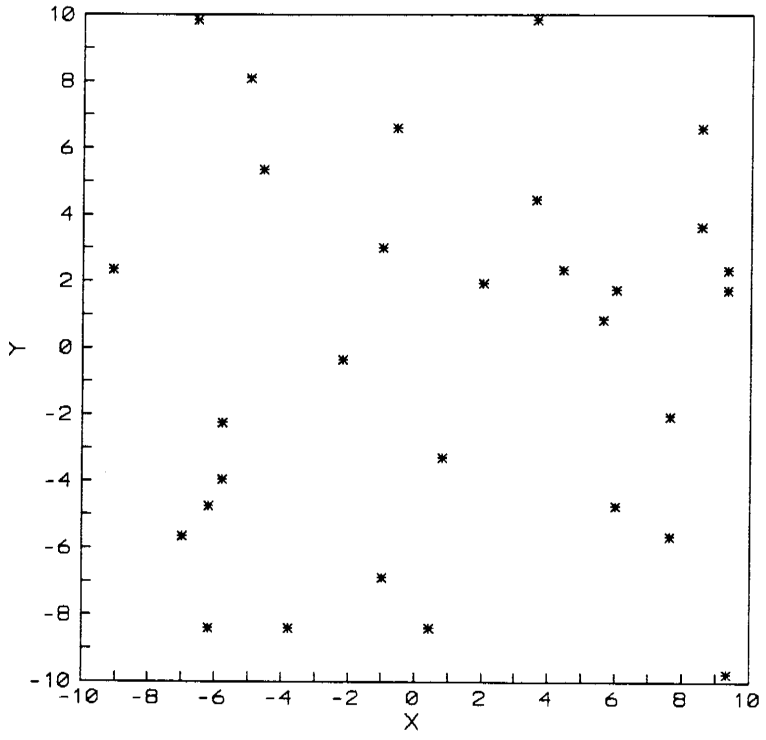
Figure 4.17 - Histograms of head gradient direction from well sets with average densities (a) 0.5 wells/unit^2; (b) 12.5 wells/unit^2.

along the +Y direction, and a 29% probability of being from 10° to 30° from the +Y direction. These probabilities change to 41% and 41.5% for the higher well density. There is also a 14% probability of the gradient being from 30° to 60° or -30° to -60° from the +Y direction. The direction determined from a well triplet may or may not be a good reflection of the gradient local any of the three wells. Either way, the possibility of misinterpreting the regional (average) gradient by extrapolating from the local gradient is significant. Clearly, the possibility of considerable variation in local gradient must be considered in the vicinity of a fault. Not surprisingly, as K_{min} increases, the probability of misinterpretation decreases. This can be seen in Figure 4.17.

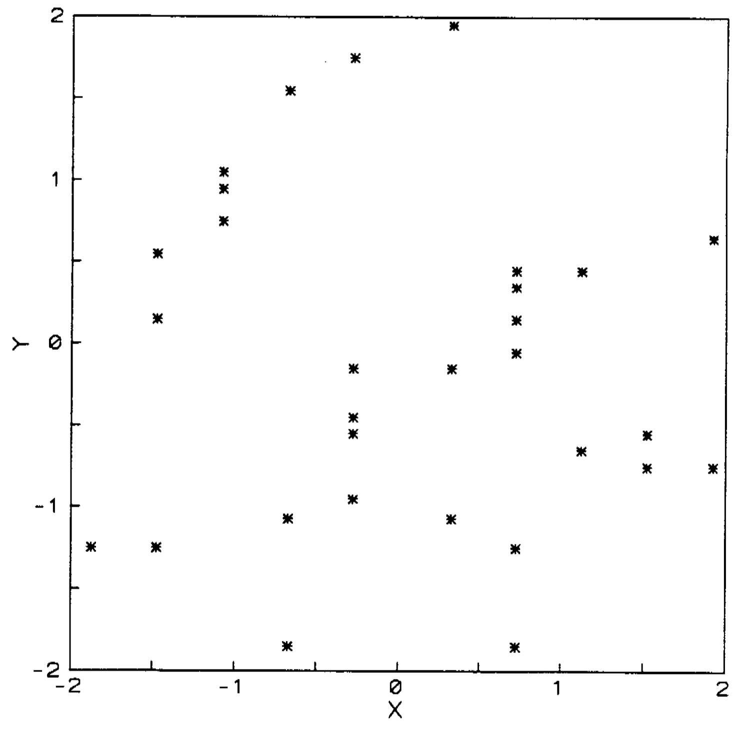
Detecting the presence of the fault through sparse well data depends on the density of the wells and on their placement in relation to the fault. If one is not aware of the presence of a fault, the probability of detecting its presence and its actual effects is low if using data from a small number of randomly placed wells. If one is aware of the presence of a fault and is trying to determine the groundwater head field in the vicinity of the fault, it is important to know something about the structure of the fault, such as fault size and displacement, before determining well placements. This is certainly the case if one is locating wells to monitor a waste storage or disposal facility where the number of wells may be small and their location is to reflect local groundwater gradients. Unfortunately, structural information about a fault may not be readily available. If the facility is small in relation to the fault, this may not be a problem as the close spacing of wells should reflect the local gradient somewhat accurately.

Consideration must be given not only to the number of wells, but to their location in relation to a fault. A set of 30 wells generated over a 20 x 20 area and a 4 x 4 area (centered on the fault centroid) are shown in Figure 4.18. From these well sets, plots of triangle area and triangle area times distance between the fault centroid (the point X = 0, Y = 0) and the nearest triangle vertex versus both the gradient magnitude and direction for are found in Figures 4.19 (several sets of 30 wells were generated with very similar results and so only the results from one set for each domain are included).

Figures 4.19 (a)-(f) show the graphs for the 20 x 20 domain. Figure 4.19 (a) is the graph of triangle area versus gradient magnitude (normalized by the regional gradient). The peak is found at the value of the average normalized gradient magnitude. As the triangle area increases, the spread around the average gradient magnitude decreases. The plot is not symmetric; it tails out to the right indicating, in part, that as the gradient increases, the average size of the triangle decreases. Based on the results for 200 wells tabulated in Table 4.3, as the number of wells increases, so does the

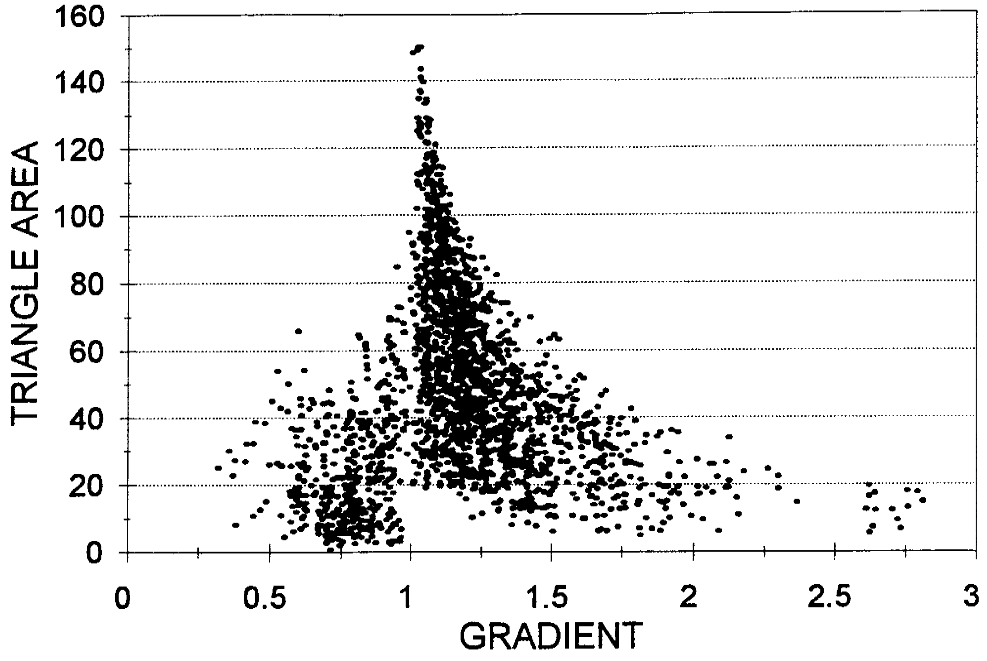


(a)

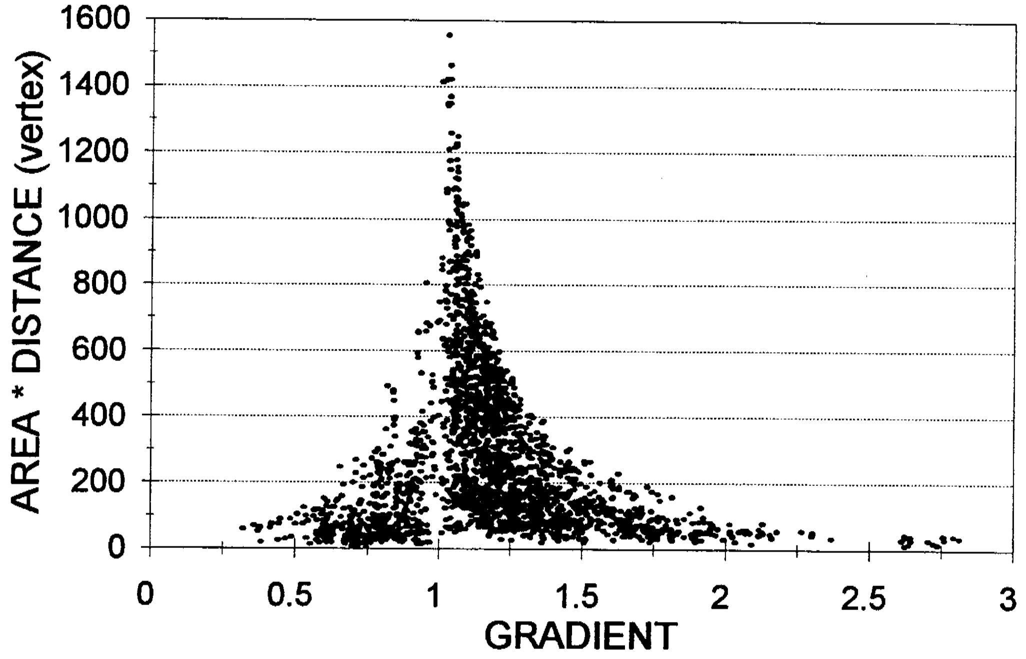


(b)

Figure 4.18 - (a) Well distribution used to generate Figures 4.19 (a) - (f); (b) Well distribution used to generate Figures 4.19 (g) - (j).

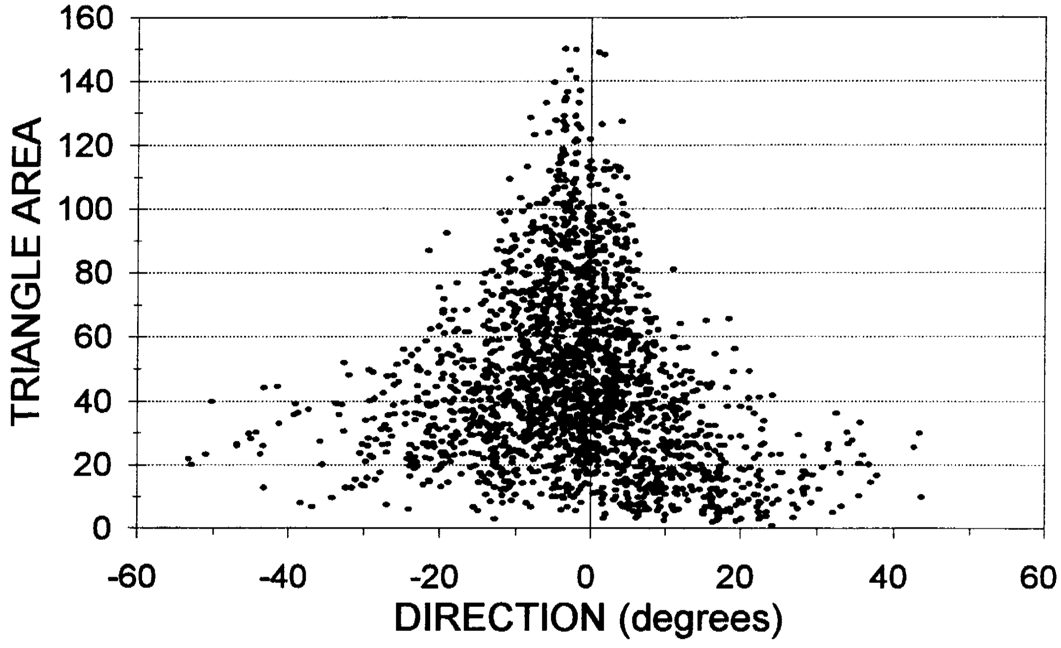


(a)

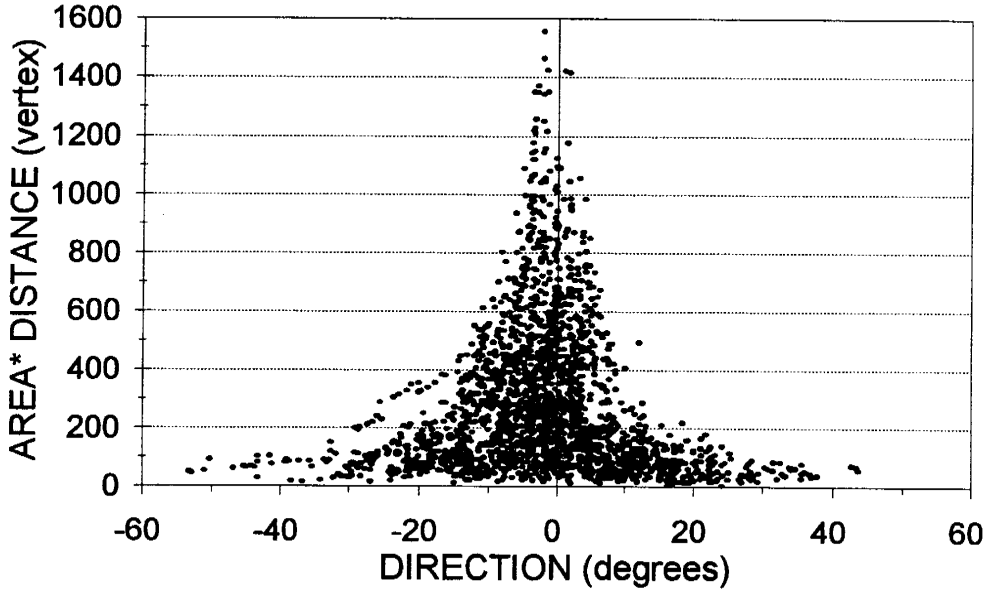


(b)

Figure 4.19 - (a) Graph of well triplet triangle area versus gradient magnitude; (b) as in Figure 4.19 (a) but area is multiplied by distance between closest vertex and fault centroid. Well set used is that of Figure 4.18 (a).

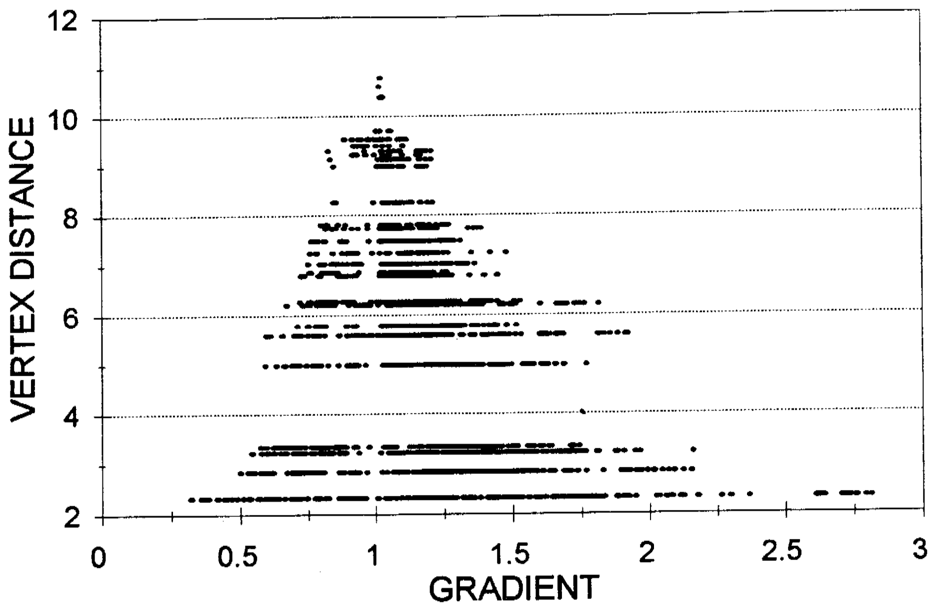


(c)

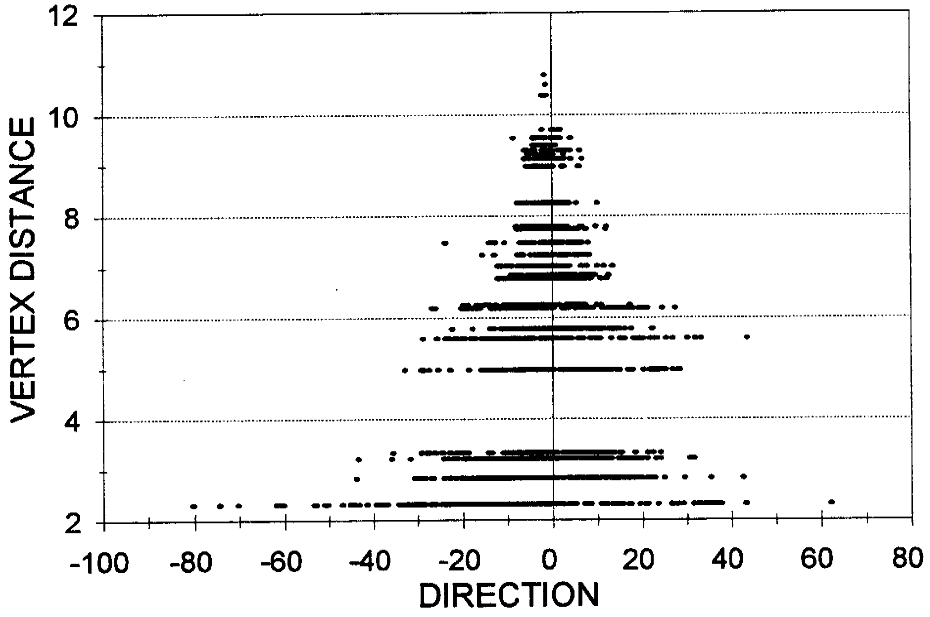


(d)

Figure 4.19 - (c) and (d) as in Figure 4.19 (a) and (b) except using gradient direction instead of magnitude.

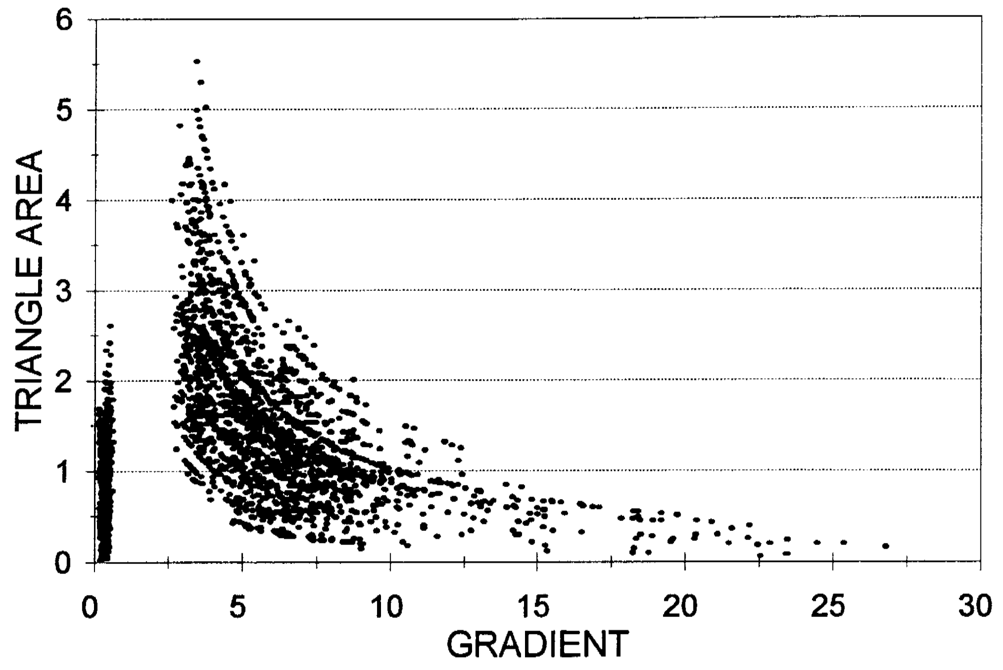


(e)

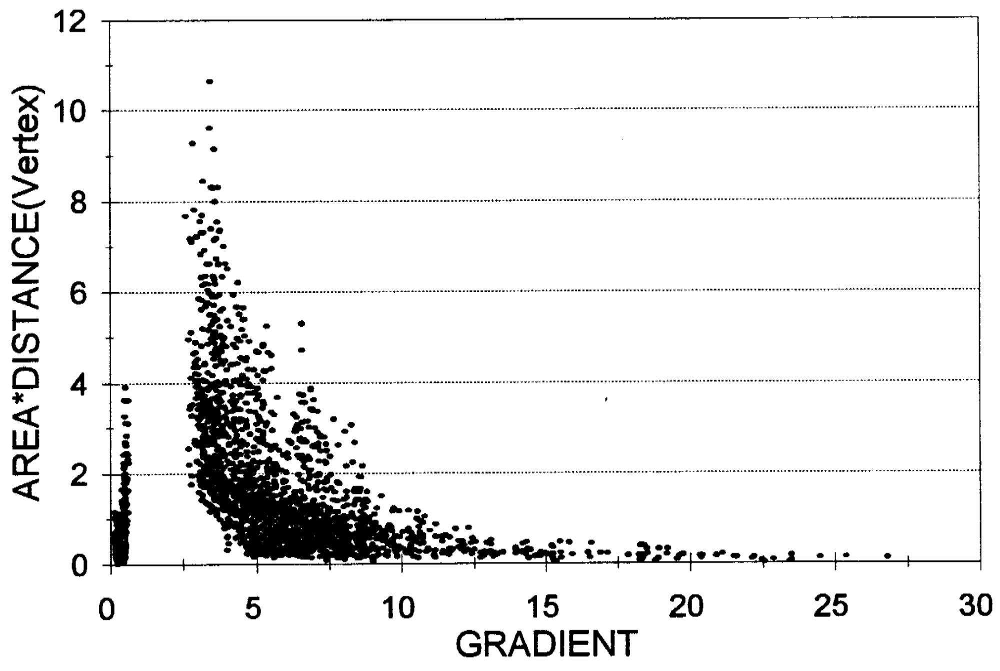


(f)

Figure 4.19 - (e) Graph of vertex distance from fault centroid versus gradient magnitude; (f) graph of vertex distance versus gradient direction

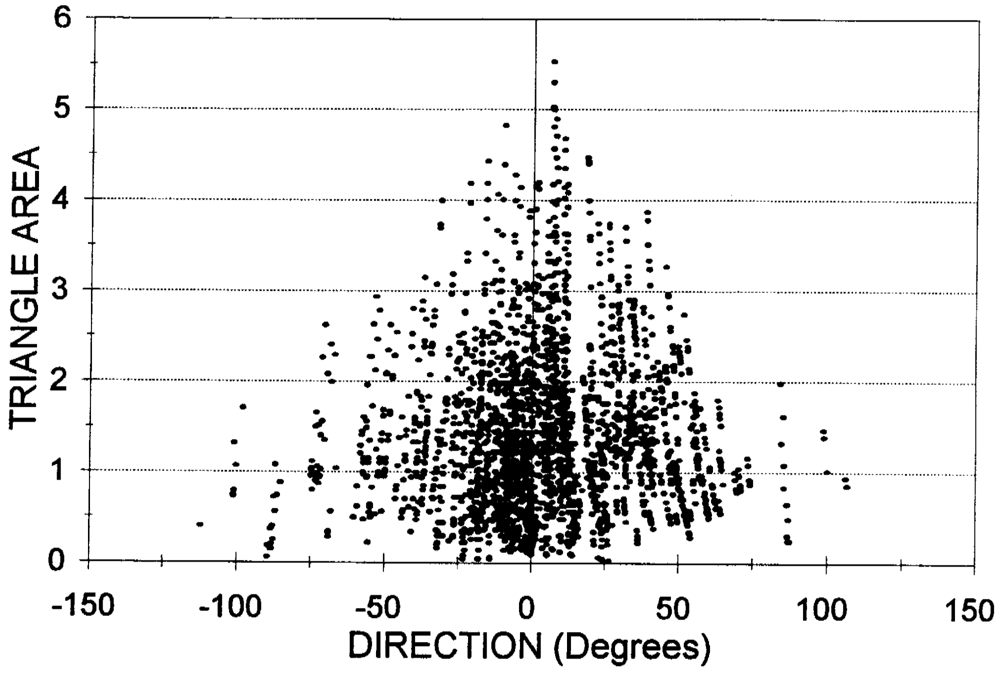


(g)

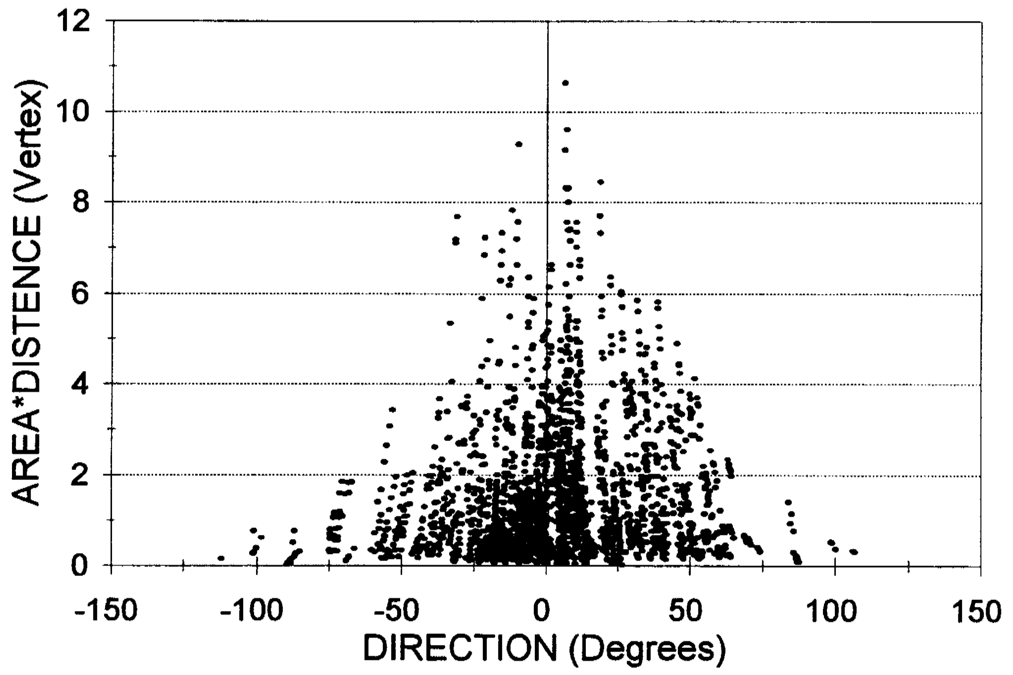


(h)

Figure 4.19 - (g) and (h) as in Figure 4.19 (a) and (b) except using well set of Figure 4.18 (b).



(i)



(j)

Figure 4.19 - (i) and (j) as in Figures 4.19 (c) and (d) except with well set of Figure 4.19 (b).

probability that the length of the tail will increase. The envelope of the plot is somewhat "fuzzy". When triangle area is multiplied by the distance of the closest vertex, as in Figure 4.19 (b), the envelope becomes more clearly defined. The shape changes little for larger triangle areas; this is to be expected since a large triangle within the domain necessarily implies a larger distance from the centroid of the closest vertex. Points on the left and right sides of the graph are noticeably compressed implying that for a well triplet to indicate a below or above average gradient magnitude, it is likely to have a vertex close to the fault centroid.

Figures 4.19 (c) and (d) are similar to (a) and (b) except gradient direction is plotted instead of gradient magnitude. The average direction for this set is -1.9° which can be seen somewhat in the graph. One would expect this graph to be symmetric; it would likely become more so with an increasing number of wells. As with gradient magnitude, we see less spread when triangle area is multiplied by vertex distance. It can be seen that the further from the average the gradient direction is, the more likely it is that the triangle area will be small and that there will be a triangle vertex close to the fault centroid. This relationship between gradient magnitude or direction and triangle vertex distance can be seen more clearly in Figures 4.19 (e) - (f).

Figures 4.19 (g)-(j) are similar to Figures 4.19 (a)-(b) except that the set of 30 wells was generated over a 4 x 4 domain. The pattern seen in Figure 4.19 (g) is similar to that seen in Figure 4.19 (a), being appearing less symmetric. The peak this time does not correspond well with the average normalized gradient magnitude which is 5.1. This is to be expected with a skewed distribution. The peak also does not correspond with the net regional gradient of .91 which is to be expected because head contours in the small region about the fault are considerably impacted by the fault.

Perhaps most noticeable is the lack of points found between gradient values of approximately 1 and 2.5. The reason for this is not clear, but it is likely that this gap would get smaller with an increasing number of wells. There is also a slight gap found in Figures 4.19 (a) and (b). Again, lower and higher gradient magnitudes correspond to smaller triangles located close to the fault centroid.

Figures 4.19 (i) and (j) are graphs of gradient direction versus triangle area and area times closest vertex distance. These graphs have a much less distinct envelope than Figures 4.19 (c) and (d) as there is much more spread about 0° (this difference can also be seen in the histograms of Figure 4.17).

These graphs in Figure 4.19 demonstrate that the location of a triplet of wells in relation to a fault and the size of the triangle they form are important factors when interpreting the gradient implied by a well triplet. Large triangles with vertices located far from a fault will, not surprisingly, reveal little impact by the fault. However, triplets that form small triangles and are located somewhat close to a fault can yield a very large range of gradient values, both in magnitude and direction.

The question arises of whether graphs of triangle area, or area times distance from the fault, versus gradient magnitude or direction can be used to uniquely identify a fault or fault type. This question is not answered by this study. It is possible that other situations could result in similar looking distributions. For example, a fold in an aquifer or a groundwater mound could possibly result in a graph of similar form. Possibly, the graph combined with other information (e.g. maximum gradient or geologic information) could aid in either the identification of a fault or the estimate of the impacts of a known fault. Additional work should be done in this area.

CHAPTER 5

CONCLUSIONS and RECOMMENDATIONS

5.1 Conclusions

- Faults are generally three-dimensional structures which have effects on groundwater flow that can be locally significant. Specifically, vertical and horizontal head gradients are formed in response to the fault's geometric and hydrologic characteristics.
- Certain information should be known about a fault in order to model it. This should include information on maximum fault displacement and displacement distribution along the fault, fault dip and fault-zone permeability distribution.
- The summary statistics studied are rather sensitive to fault displacement, fault-zone permeability distribution and fault dip. Changes in fault dip can cause significant changes in vertical head gradients, both in magnitude and direction. Horizontal head gradients are not as sensitive.
- As fault zones are anisotropic, it becomes difficult to realistically model them using a finite-difference model if the fault dip is other than 90°. One faces either the problem of determining the values of the full conductivity tensor for each cell intersected by the

fault plane, or the need for too dense a grid in order to keep each cell homogeneous and isotropic.

- Solving a three-point problem for well triplets with at least one well located relatively close to the area where groundwater flow is most impacted by a fault can result in a large range of values of groundwater gradient magnitude and direction.
- Contours of hydraulic head determined from sparse well data in the vicinity of a fault are not likely to adequately indicate the effects of the fault.
- Groundwater gradient directions determined from sparse well data in the vicinity of a fault may show considerable variability and may be misleading as to a regional gradient.
- Ideally, one should have some understanding of the structure of a fault prior to placing wells for the purpose of monitoring a facility in the vicinity of a fault.

5.2 Recommendations

A major difficulty in creating a good 3-D model is obtaining enough hydrogeologic information to make the model meaningful. Obtaining the additional information necessary to accurately represent a particular fault in a 3-D model could be prohibitive. One solution is more generalized data relating various fault and geologic properties.

With such general fault information and relationships available, one might be able, in certain modeling endeavors, to take a stochastic approach to the model as a way of dealing with the lack of specific information. For example, with a reasonable database on permeability distribution for given lithology, one could determine a likely probability density function for permeability for the given fault based on known lithology and displacement. This could then be used as any other stochastic parameter in a stochastic model.

Currently, there is little information available detailing reductions and enhancement of permeability throughout fault zones. In particular, there is a need for information relating lithologies to permeability variations and information relating permeability and displacement. It is strongly recommended that field studies be undertaken to look at these relationships within several different fault zones.

Because of the difficulties of using a rectangular grid for modeling a fault zone, as described above, it is recommended that finite-element models be constructed to simulate fault zones and compared with the results from the finite-difference model. Either very thin 3-D elements or 2-D, homogeneous, isotropic elements could be embedded in a 3-D model to simulate the fault plane thus eliminating the problem of anisotropy while allowing definition of the fault-plane thickness for any dip. Nothing is free, however. The disadvantage is that grid construction is more difficult and that the grid will change for each new fault geometry. It may be worth it, however, to see the results for more accurately defined properties.

A worthwhile endeavor would be to simulate more complex geologic scenarios. One way to do this would be to use the same fault model, but a different regional model. For example, multiple aquifers separated by aquitards. Or an upper unconfined aquifer. Or a fault which extends to the surface where the fault zone permeability is enhanced.

Another way to increase the complexity is to change the fault model. For example, one could add multiple faults into the region. Since there is an infinity of such compound fault models one could choose from, one should have a well defined objective in mind.

More study should be devoted to investigating fault and fault impact inference based on data from sparse sets of wells in the vicinity of faults. In particular, more study should be devoted to the potential for graphs of triangle area or triangle area times distance between the closest vertex of a triangle versus gradient magnitude or direction determined from well triplets formed from a set of wells. Can such graphs identify a fault or the extent of impacts to groundwater flow by a fault? Can such graphs distinguish between different structures and different impacts to groundwater?

REFERENCES

Barnett, J.A., Mortimer, J., Rippon, J.H., Walsh, J.J. and Watterson, J., 1987, *Displacement Geometry in the Volume Containing a Single Normal Fault.* AAPG Bulletin, 71(8), pp. 925-937

Bruhn, R.L. and Yonkee, W.A., 1988, *Fracture Networks in Fault Zones*. AGU Fall Meeting, EOS Transactions, 69(44), p. 1172

Forster, C.B. and Evans, J.P., 1991, *Hydrogeology of Thrust Faults and Crystalline Thrust Sheets: Results of combined Field and Modeling Studies*. Geophys Res. Lett., 18(5), pp. 979-982

Forster, C.B. and Smith, L., 1988, *Groundwater Flow Systems in Mountainous Terrain 2. Controlling Factors*. Water Resour. Res., 24(7) pp. 1011-1023

Gibson, J.R., Walsh, J.J. and Watterson, J., 1989, *Modelling of Bed Contours and Cross-Sections Adjacent to Planar Normal Faults*. J. Struct. Geol., 11(3), pp.317-328

Hsieh, P.A. and Freckleton, J.R., 1992, *Documentation of a Computer Program to Simulate Horizontal-Flow Barriers Using the Modular Three-Dimensional Finite Difference Ground-Water Flow Model*. Preliminary USGS Open File Report.

Hull, J., 1988, *Thickness-Displacement Relationships for Deformation Zones*. J. Struct. Geol., 10(4), pp. 431-435

Huntoon, P.W., 1981, *Fault Controlled Ground-Water Circulation Under the Colorado River, Marble Canyon, Arizona*. Ground Water, 19(1), pp. 20-27

Huntoon, P.W., 1985, *Fault Severed Aquifers Along the Perimeters of Wyoming Artesian Basins*. Ground Water, 23(2), pp. 176-181

Logan, J.M., 1991, *The Influence of Fault Zones on Crustal-Scale Fluid Transport*. AAPG 1991 annual convention, AAPG Bull., 75(3), p. 623

Maclay, R.W. and Small, T.A., 1983, *Hydrostratigraphic Subdivisions and Fault Barriers of the Edwards Aquifer, South-Central Texas, U.S.A.* J. of Hydrology, 61, pp. 127-146

Macedonio, G. and Pareschi, M.T., 1991, *An Algorithm for the Triangulation of Arbitrarily Distributed Points: Applications to Volume Estimate and Terrain Fitting*. Computers & Geoscience, 17(7), pp. 859-874

McCord, J.P., McCord, J.T., Gibson, J.D. and King, H.L., 1992 *Sandia National Laboratories Site-Wide Hydrogeologic Characterization Project Calendar Year 1992 Annual Report*. United States Department of Energy, Albuquerque Operations Ofice

McDonald, M.G. and Harbaugh, A.W., 1988, *A Modular Three-Dimensional Finite-Difference Ground Water Flow Model*. U.S. Geological Survey Technics of Water-Resource Investigations, Book 6, Chapter A1

Morrow, C.A., Shi, L.Q. and Byerlee, J.D., 1984, *Permeability of Fault Gouge Under Confining Pressure and Shear Stress*. J. Geophys. Res., 89(B5), pp. 3193-3200

Mozley, P.S., Goodwin, L.B., 1995, *Patterns of Cementation along a Cenozoic Normal Fault: A Record of Paleoflow Orientations*, Geology, In Press

Newcomb, R.C., 1969, *Effects of Tectonic Structure on the Occurrence of Ground Water in the Basalt of the Columbia River Group of the Dalles Area, Oregon and Washington*. USGS Prof. Paper 383-C

Pine, R.A., 1991, *Report to the State Engineer on Aspects of the Hydrogeology and Quality of Groundwater in the Sandia Basin, New Mexico*. New Mexico State Engineer Office

Pittman, E.D., 1981, *Effect of Fault-Related Granulation on Porosity and Permeability of Quartz Sandstones, Simpson Group (Ordovician), Oklahoma*. AAPG Bulletin, 65(11), pp. 2381-2387

Robertson, E.C., 1983, *Relationship of Fault Displacement to Gouge and Breccia Thickness*. Miner. Engr., 35(10), pp. 1426-1432

Smith, L., Forster, C.B. and Evans, J.P., 1990, *Interaction of Fault Zones, Fluid Flow, and Heat Transfer at the Basin Scale*. "Hydrology of Low Permeability Rocks", IAH Hydrology Selected Papers, Vol. 2

Stewart, I.S. and Hancock, P.L., 1991, *Scales of Structural Heterogeneity within Neotectonic Normal Fault Zones in the Aegean Region*. J. of Struc. Geol., 13(2), pp. 191-204

Townley, R.L. and Wilson, J.L., 1980, *Description of and User's Manual for a Finite Element Aquifer Flow Model, AQUIFEM-1*. Ralph M. Parsons Laboratory for Water Resources and Hydrodynamics Report No. 252

Twiss, R.J. and Moores, E.M., 1992, *Structural Geology*. W. H. Freeman and Co.

Wallace, R.E. and Morris, H.T., 1986, *Characteristics of Faults and Shear Zones in Deep Mines*. Pure Appl. Geophys., 124(1/2), pp. 107-125

Walsh, J.J. and Watterson, J,, 1987, *Distributions of Cumulative Displacement and Seismic Slip on a Single Normal Fault Surface*. J, of Struc. Geol., 9(8), pp. 1039-1046

Walsh, J.J. and Watterson, J., 1988, *Analysis of the Relationship Between Displacements and Dimensions of Faults*. J. of Struc. Geol., 10(3), pp. 239-247

Walsh, J.J. and Watterson, J., 1989, *Displacement Gradients on Fault Surfaces*. J. of Struc. Geol., 11(3), pp. 307-316

Watterson, J., 1986, *Fault Dimensions, Displacements and Growth*. Pure Appl. Geophys., 124(1/2), pp. 365-373

APPENDIX A

```
      program flt6
C*********************************************************************
C              VARIABLE DESCRIPTIONS
C
C   COND()      3D array used to hold the index into COND_XYZ() for each
C               point
C   COND_XYZ()  Stores the X (1), Y (2) and Z (3) K values for each point
C   DELX()      column spacing (1,*), cell centroid in X (2,*)
C   DELY()      row spacing (1,*), cell centroid in Y (2,*)
C   DELZ()      MODFLOW layer spacing (1,*), cell centroid in Z (2,*)
C               DELZ(*,1) is the top layer (DELZ(2,1) is largest)
C   DELX2()     As above but for slicing
C   DELY2()                  "
C   DELZ2()                  "
C   LIMIT()     Contains lower contact (1,*,*) and upper contact (2,*,*)
C               of aquifer after displacement
C   MESH()      2D array used to determine initial heads
C   DISPLACE    Maximum displacement on fault
C   FAULT_LN    Length of fault
C   FAULT_DP    Dip angle of fault in degrees
C   F_DIAM      Diameter (width) of gauge zone
C   F_WIDTH     Maximum width of fault elipse
C   F_MODEL     Fault model: 1 variable K distributio; 2 variable diam
C   LOW()       Z position of lower contact along fault plane
C   NUM_X       Number of cells in X direction (Columns)
C   NUM_Y       Number of cells in Y direction (Rows)
C   NUM_Z       Number of (MODFLOW) layers
C   SIDE_LN     Length of side of square region
C   UP()        Z position of upper contact along the fault plane
C   OUT_FILE    Out file name (no extension)
C   CHOICE      Menu option selected
C   FAULT_ON    Logical indicating if current file contains a fault
C   HAVE_DAT    Logical indicating if a data set has been defined
C   WALLS       TRUE if both walls deformed, FALSE if only hanging wall
C
C                            ASSUMPTIONS
C   -------------------------------------------------------------
C   1) Y-Z cell dimensions are fixed for angle
C   2) Top & Bottom geologic layers are aquitards, middle layer
C      is an aquifer
C   3) Y = min and Y = max sides are constant head boundaries
C   4) Fault dip is in direction of +Y
C   5) Fault strike in X direction
C   6) Fault "plane" goes through middle of aquifer and mid-X
C
C**********************************************************************

      parameter (MAX_X = 90, MAX_Y = 180, MAX_Z = 160)
      integer   COND(MAX_Y,MAX_X,MAX_Z)
      real      DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z),
     #          DELX2(2,MAX_X), DELY2(2,MAX_Y), DELZ2(2,MAX_Z),
     #          MESH(MAX_Y,MAX_X), LIMIT(2,MAX_X,MAX_Y), UP(MAX_X),
     #          LOW(MAX_X), HEADS(MAX_Y,MAX_X,MAX_Z)
      double precision COND_XYZ(MAX_X*MAX_Z,3)
      character*1 CHOICE, OUT_FILE*8, F_MODEL
      logical   HAVE_DAT, FAULT_ON, WALLS

      common /GRID/ NUM_X, NUM_Y, NUM_Z
      common /FAULT/ FAULT_LN, FAULT_DP, DISPLACE, F_WIDTH, F_DIAM,
     #               F_MODEL
```

```
      HAVE_DAT = .FALSE.

      CHOICE   = '1'

      do while (CHOICE .NE. '8')
        call Menu(CHOICE)
        if (CHOICE .EQ. '1') then
           FAULT_ON = .FALSE.
           call Get_data(OUT_FILE,HAVE_DAT,FAULT_ON,WALLS,DELX,DELY,
     #                   DELZ,MAX_X,MAX_Y,MAX_Z)
        else if (CHOICE .EQ. '2') then
          call K_File(OUT_FILE,HAVE_DAT,FAULT_ON,WALLS,DELX,DELY,DELZ,
     #                LIMIT,COND,COND_XYZ,UP,LOW,MAX_X,MAX_Y,MAX_Z)
        else if (CHOICE .EQ. '3') then
          call Initial(OUT_FILE,HAVE_DAT,DELX,DELY,DELZ,
     #                 MAX_X,MAX_Y,MAX_Z,MESH)
        else if (CHOICE .EQ. '4') then
          call Slice(DELX2,DELY2,DELZ2,MAX_X,MAX_Y,MAX_Z)
        else if (CHOICE .EQ. '5') then
          call Compare()
        else if (CHOICE .EQ. '6') then
          call Wells(HAVE_DAT,HEADS,DELX,DELY,DELZ,MAX_X,MAX_Y,
     #               MAX_Z)
        else if (CHOICE .EQ. '7') then
          call Stats(HEADS,DELX,DELY,DELZ,MAX_X,MAX_Y,MAX_Z)
      end if
      end do

      end


C***********************************************************************
C                        SUBROUTINE Bndry()
C
C   Bndry() constructs the boundary array for each MODFLOW layer and
C   outputs it to the boundary file FNAME. It assumes that the front
C   and back sides are constant head boundaries. The other sides are
C   variable head but impervious boundaries. The constant head
C   values are determined by I_Heads() define a non-fault gradient.
C   Called by Initial().
C
C***********************************************************************
      subroutine Bndry(FNAME)

      character FNAME*12

C***** LOCAL VARIABLES *****
      integer ROW(2,120)
      character*1 DIRECT

      common /GRID/ NUM_X, NUM_Y, NUM_Z

C****** Output formats *******
1     format(100I3,$)

C*****  Output the boundary file *****
      OPEN(UNIT=40,FILE=FNAME,STATUS='UNKNOWN')

      Write(*,10)
10    format('Flow in Y direction or X direction (Y/X)  >> ',$)
      Read '(A)', DIRECT


C***** Define the rows *****
```

```
      if (DIRECT .EQ. 'X'  .OR.  DIRECT .EQ. 'x') then
        do 20 I=2,NUM_X-1
          ROW(1,I) = 1
          ROW(2,I) = 1
20      continue
        ROW(1,1) = -1
        ROW(2,1) = -1
        ROW(1,NUM_X) = -1
        ROW(2,NUM_X) = -1
      else
        do 30 I=1,NUM_X
          ROW(1,I) = -1
          ROW(2,I) = 1
30      continue
      end if

C***** Output the boundary arrays for interior layers *****
      do 70 L=1,NUM_Z
        Write(40,1) (ROW(1,I), I=1,NUM_X)
        do 50 J=2,NUM_Y-1
          Write(40,1) (ROW(2,I), I=1,NUM_X)
50      continue
        Write(40,1) (ROW(1,I), I=1,NUM_X)
70    continue

      Close(40)
      Write(*,200) FNAME
200   format(/,' Boundary arrays written to file ',A12,/)
      end


C**********************************************************************
C
C       SUBROUTINE Compare()
C
C  Compare() takes two data files in SURFER format, and, if they are
C  the same length and have the same X,Y locations in the same order
C  (by X,Y is meant the 1st and 2nd entries in each file record which
C  correspond to location), will output a file with the same X,Y values
C  but Z1-Z2 for the Z value.
C  Called by Main.
C
C               LOCAL VARIABLES
C
C  FILE1    Name of first input file
C  FILE2    Name of second data file
C  F_EXIST  Logical = .TRUE. if data file exists
C  LENGTH   Logical = .TRUE. if both data files have same # of records
C  MATCH    Logical = .TRUE. if both data files have same X,Y values
C  OFILE    Name of ouput file
C  X1,Y1,Z1 Record from FILE1
C  X2,Y2,Z2 Record from FILE2
C
C**********************************************************************
      subroutine Compare()

C***** LOCAL VARIABLES *****
      real X1, X2, Y1, Y2, Z1, Z2, GRAD
      integer NUM1, NUM2
      character*12  FILE1, FILE2, OFILE, TRASH*1, E_TYPE*1
      logical F_EXIST, MATCH, LENGTH

1     format(F7.2,F7.2,F8.3)
```

```
C***** Open the 2 data files *****
      Write(*,20)
20    format(////,' Enter the name of the first data file  >> ',$)
      Read '(A12)', FILE1

      Inquire(FILE=FILE1,EXIST=F_EXIST)
      if (.NOT. F_EXIST) then
        Write(*,30) FILE1
30      format(////,' FILE ',A12,' DOES NOT EXIST!!!  press enter.')
        Read '(A)', TRASH
        RETURN
      end if
      Open(UNIT = 20, FILE = FILE1, STATUS = 'OLD')

      Write(*,40)
40    format(/,' Enter the name of the second data file  >> ',$)
      Read '(A12)', FILE2

      Inquire(FILE=FILE2,EXIST=F_EXIST)
      if (.NOT. F_EXIST) then
        Write(*,50) FILE2
50      format(////,' FILE ',A12,' DOES NOT EXIST!!!  press enter.')
        Read '(A)', TRASH
        RETURN
      end if
      Open(UNIT = 22, FILE = FILE2, STATUS = 'OLD')

      Write(*,55)
55    format(/,' Absolute error (A) or relative error (R)?  >> ',$)
      Read '(A)', E_TYPE

      if (E_TYPE .EQ. 'R'  .OR.  E_TYPE .EQ. 'r') then
        Write(*,60)
60      format(/,' Enter the gradient  >> ',$)
        Read(*,*) GRAD
      end if

C***** Open the output file *****
      Write(*,70)
70    format(/,' Enter the name of the OUTPUT file >> ',$)
      Read '(A12)', OFILE
      Open(UNIT=24, FILE=OFILE, STATUS='UNKNOWN')

C***** Check for same length of files *****
      LENGTH = .TRUE.
      NUM1 = 0
      do while (.TRUE.)
        Read(20,1,END=80)X1,Y1,Z1
        NUM1 = NUM1 + 1
      end do

80    NUM2 = 0
      do while (.TRUE.)
        Read(22,1,END=90)X2,Y2,Z2
        NUM2 = NUM2 + 1
      end do

90    if (NUM1 .NE. NUM2) then
        LENGTH = .FALSE.
      else
        Rewind(20)
        Rewind(22)
      end if
```

```
C***** Output the difference of each record if same location *****
      MATCH = .TRUE.
      do while (MATCH  .AND.  LENGTH)
        Read(20,*,END=100) X1,Y1,Z1
        Read(22,*,END=100) X2,Y2,Z2
        if (X1 .NE. X2  .OR.  Y1 .NE. Y2) MATCH = .FALSE.
        if (E_TYPE .EQ. 'R'  .OR.  E_TYPE .EQ. 'r') then
          Write(24,1) X1,Y1,(Z1-Z2)/GRAD
        else
          Write(24,1) X1,Y1,Z1-Z2
        end if
      end do

100   if (.NOT. MATCH) then
        Write(*,110)
110     format(//,' DATA LOCATIONS IN TWO FILES DON''T MATCH!',$)
      else if (.NOT. LENGTH) then
        Write(*,120)
120     format(//,' DATA FILES NOT THE SAME LENGTH! ',$)
      else
        Write(*,130) OFILE
130     format(//,' Difference data written to ',A12,$)
      end if

      Write(*,140)
140   format(' Press Enter...',$)
      Read '(A)', TRASH
      Close(20)
      Close(22)
      Close(24)

      end


C*********************************************************************
C
C                   FUNCTION Distance
C
C  Distance() returns the distance between the line AY + BZ + C = 0
C  and the point (Y,Z)
C
C*********************************************************************
      function Distance(A,B,C,Y,Z)

      real Y, Z, Distance
      double precision A, B, C

      Distance = Abs(A*Y + B*Z + C)/Sqrt(A*A + B*B)
      end



C*********************************************************************
C
C                        SUBROUTINE Fault_On()
C
C  Fault_On() assigns the fault conductivity to the cells along
C  the trace of the fault for cells Y = Y_NX, Z = Z_NX.  The
C  conductivity varies ellipticly from K(4) at the center of the fault
C  to K(2) at the tip-line. An effective conductivity is calculated
C  based on the fault diameter and how the fault bisects the cell.
C  Called by In_Fault().
C
```

```
C------------------------ LOCAL VARIABLES ---------------------------
C
C  BOT_NX Z index to bottom of undisterbed aquifer
C  DISP   Normalized displacement along fault assuming elliptic variation
C  K1     K value just upgradient of fault plane
C  K2     K value of fault
C  K3     K value just downgradient of fault plane in cell
C  KF     Permeability factor for given fault radius
C  RF     Radius of fault (1/2 the length)
C  RX     Normalized distance of cell centroid to fault center
C  TOP_NX Z index to top of undisterbed aquifer
C  XMID   X value of center of fault
C  ZMID   Z value at center of fault
C*******************************************************************
      subroutine Fault_On(COND,COND_XYZ,K,DELX,DELY,DELZ,LEFT,RGHT,
     #                    Y_NX,Z_NX,THICK,A,B,C,MAX_X,MAX_Y,MAX_Z,UP,
     #                    LOW,KF_MAX,K_CNT)

      double precision  COND_XYZ(MAX_X*MAX_Z,3), A, B, C
      real K(6), DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z),
     #     UP(MAX_X), LOW(MAX_X), KF_MAX
      integer  LEFT, RGHT, Y_NX, Z_NX, COND(MAX_Y,MAX_X,MAX_Z)
      character*1 F_MODEL

C*****  LOCAL VARIABLES *****
      integer TOP_NX, BOT_NX
      real    XMID, ZMID, RF, K1, K2, K3, KF
      double precision DISP, RX

      common /GRID/ NUM_X, NUM_Y, NUM_Z
      common /FAULT/ FAULT_LN, FAULT_DP, DISPLACE, F_WIDTH, F_DIAM,
     #               F_MODEL

C**** Find X and Z midpoints ****
      XMID = (DELX(2,NUM_X) + DELX(1,NUM_X)/2)/2.0
      RF = FAULT_LN/2.0
      ZMID = (DELZ(2,1) + DELZ(1,1)/2)/2.0
      Call Find_NX(DELZ,ZMID+THICK/2,TOP_NX,NUM_Z,2)
      Call Find_NX(DELZ,ZMID-THICK/2,BOT_NX,NUM_Z,2)

      do 100 I=LEFT,RGHT
        K_CNT = K_CNT + 1
        RX = Abs(DELX(2,I) - XMID)/RF
        DISP = 2*DSqrt(((RX+1)/2.0)**2 - RX**2)*(1 - RX)
        KF = 1.0 + DISP*(KF_MAX - 1.0)

C**** Find the fault plane conductivity ***
        if (F_MODEL .EQ. '1') then
          if (DELZ(2,Z_NX) .GT. UP(I)) then
            K2 = DExp(DISP*Log(K(6)) + (1.0 - DISP)*Log(K(3)))
          else if (DELZ(2,Z_NX) .LT. LOW(I)) then
            K2 = DExp(DISP*Log(K(4)) + (1.0 - DISP)*Log(K(1)))
          else
            K2 = DExp(DISP*Log(K(5)) + (1.0 - DISP)*Log(K(2)))
          end if
        else if (DELZ(2,Z_NX) .GT. UP(I)) then
          K2 = K(6)
        else if (DELZ(2,Z_NX) .LT. LOW(I)) then
          K2 = K(4)
        else
          K2 = K(5)
        end if
        K1 = COND_XYZ(COND(Y_NX-1,I,Z_NX),2)
        K3 = COND_XYZ(COND(Y_NX+1,I,Z_NX),2)

C**** Find the effective K for the cell ****
        COND(Y_NX,I,Z_NX) = K_CNT
        call K_Effct(COND_XYZ,K_CNT,K1,K2,K3,Y_NX,Z_NX,A,B,C,
     #              DELY,DELZ,MAX_X,MAX_Y,MAX_Z,DISP,KF)

C***** Write fault flow info for use in Summary Stats *****
        if(Z_NX .GE. TOP_NX  .AND.  Z_NX .LE. BOT_NX) then
          if (FAULT_DP .EQ. 90.0) then
            Write(75,10) DELX(2,I), DELY(2,Y_NX)-1, DELZ(2,Z_NX),
     #            COND_XYZ(COND(Y_NX-1,I,Z_NX),2),
     #            COND_XYZ(COND(Y_NX-2,I,Z_NX),2)
10          format(F7.2,F7.2,F7.2,F8.5,F8.5)
          else
            Write(75,20) DELX(2,I), DELY(2,Y_NX), DELZ(2,Z_NX),
     #            COND_XYZ(COND(Y_NX,I,Z_NX),2),
     #            COND_XYZ(COND(Y_NX-1,I,Z_NX),2),
     #            COND_XYZ(COND(Y_NX-1,I,Z_NX-1),2)
20          format(F7.2,F7.2,F7.2,F8.5,F8.5,F8.5)
          end if
        end if

100   continue
      end


C**********************************************************************
C
C                        SUBROUTINE Find_NX()
C
C Find_NX() returns the index, NX, into DEL of the cell that contains
C the value XYZ. If TYPE = 1, then DEL is increasing (DELX & DELY);
C if TYPE = 2, then DEL is decreasing (DELZ). It uses a bisection
C method to locate the index quickly.
C Called by Slice().
C
C LOCAL VARIABLES
C
C FOUND Logical is TRUE if XYZ is found within DEL
C LEFT Current left index of DEL bounding XYZ
C MID Current MID point between LEFT and RGHT
C RGHT Current right index of DEL bounding XYZ
C**********************************************************************
      subroutine Find_NX(DEL,XYZ,NX,NUM,TYPE)

      real DEL(2,NUM), XYZ
      integer NX,TYPE

C***** LOCAL VARIABLES *****
      integer LEFT, RGHT, MID
      logical FOUND

      LEFT = 1
      RGHT = NUM
      FOUND = .FALSE.

      do while (LEFT .LT. RGHT)
         MID = Nint((LEFT + RGHT)/2.0)
         if (XYZ .GT. (DEL(2,MID)+DEL(1,MID)/2)) then
           if (TYPE .EQ. 1) then
             LEFT = MID + 1
          else
            RGHT = MID - 1
          end if
        else if (XYZ .LT. (DEL(2,MID)-DEL(1,MID)/2)) then
          if (TYPE .EQ. 1) then
            RGHT = MID - 1
          else
            LEFT = MID + 1
          end if
        else
          LEFT = MID
          RGHT = MID
        end if
      end do

      if (LEFT .EQ. RGHT) then
         NX = LEFT
      else
         NX = -1
      end if

      end




C**********************************************************************
C
C                 SUBROUTINE Fold()
C
C Fold() deforms the hanging wall down. Displacement and fault width
C (perpendicular distance from fault plane to tip-line) varies along
C the fault elliptically.
C Called byIn_Fault().
C
C LOCAL VARIABLES
C-----------------------------------------------------------
C BOTZ Z value for lower limit of bottom contact for given X
C BOTZ_NX Z index for lower contact
C DSPL_BT Displacement of upper contact
C DSPL_TP Displacement of lower contact
C DSPL_X Displacement on fault at a given X value
C PHI Angle between horizontal and deformed contact for a given X
C R Normalized displacement or width
C RX Normalized radial distance from X to fault center, XMID
C TEMP Temporary storage of an expression
C TOPZ_NX Z index for lowest layer to check for altered conductivity
C WDTH_X Fault width at a given X value
C XMAX_NX Largest X value to determine deformation
C XMIN_NX
C YMAX_NX
C YMIN_NX
C
C**********************************************************************
      subroutine Fold(COND,COND_XYZ,K,XMID,YMID,ZMID,DELX,DELY,DELZ,
     # A,B,C,THICK,LAYERS,MAX_X,MAX_Y,MAX_Z,WALLS)

      real XMID, YMID, ZMID, DELX(2,NUM_X), DELY(2,NUM_Y),
     # DELZ(2,NUM_Z), THICK, K(6)
      double precision COND_XYZ(MAX_X*MAX_Z,3), A, B, C
      integer COND(MAX_Y,MAX_X,MAX_Z)
      character*1 LAYERS, F_MODEL
      logical WALLS
```

```
C***** LOCAL VARIABLES *****
      real PHI, DSPL_X, WDTH_X, DSPL_BT, DSPL_TP, BOTZ
      double precision RX, R
      integer TOPZ_NX, BOTZ_NX, XMAX_NX, XMIN_NX, YMAX_NX,
     # YMIN_NX, YMID_NX, ZMID_NX

      common /GRID/ NUM_X, NUM_Y, NUM_Z
      common /FAULT/ FAULT_LN, FAULT_DP, DISPLACE, F_WIDTH, F_DIAM,
     #               F_MODEL

      YMID_NX = Int(NUM_Y/2) + 1
      ZMID_NX = Int(NUM_Z/2) + 1

      Call Find_NX(DELZ,ZMID+THICK/2,TOPZ_NX,NUM_Z,2)
      DIP = FAULT_DP*3.141592/180
      COSN = Cos(3.141592/2 - DIP)
      SINN = Sin(3.141592/2 - DIP)

C***** Find X limits & MIN Y limit *****
      call Find_NX(DELX,XMID-FAULT_LN/2,XMIN_NX,NUM_X,1)
      call Find_NX(DELX,XMID+FAULT_LN/2,XMAX_NX,NUM_X,1)
      YMIN = YMID - (THICK/2)*SINN
      call Find_NX(DELY,YMIN,YMIN_NX,NUM_Y,1)

      PHI = ASin(DISPLACE*COSN/F_WIDTH)

C***** Deform the hanging wall *****
      do 100 I=XMIN_NX,XMAX_NX

C**** find displacement and fault width for given X value
         RX = Abs(2*(DELX(2,I) - XMID)/FAULT_LN)
         R = 2.0*DSqrt(((1.0+RX)/2)**2 - RX**2)*(1.0-RX)
         DSPL_X = DISPLACE*R
         WDTH_X = F_WIDTH*R

C**** Find the MAX Y limit *****
         TEMP = YMID + (THICK/2)*SINN + WDTH_X*Cos(PHI)
         call Find_NX(DELY,TEMP,YMAX_NX,NUM_Y,1)

         TEMP = WDTH_X*Cos(PHI) + DSPL_X*SINN

         do 80 J=YMIN_NX-2,YMAX_NX
           if (LAYERS .EQ. '3' .AND. DELY(2,J)-YMIN .LT. TEMP) then
             DSPL_TP = (TEMP - (DELY(2,J) - YMIN))*Tan(PHI)
           else
             DSPL_TP = 0
           end if
           DSPL_BT = (TEMP - (DELY(2,J)-(YMID+(THICK/2)*SINN)))*Tan(PHI)

           BOTZ = ZMID-(THICK/2)-DSPL_BT
           call Find_NX(DELZ,BOTZ,BOTZ_NX,NUM_Z,2)

           do 60 L=TOPZ_NX,BOTZ_NX
C**** determine if in hanging wall ****
             if ((DELY(2,J)-YMID)*COSN + (DELZ(2,L)-ZMID)*SINN .GT. 0)
     #        then
C**** Determine which geologic layer it is in ****
               if (DELZ(2,L) .LE. (ZMID+THICK/2-DSPL_TP)) then
                  COND(J,I,L) = 2
               else
                  COND(J,I,L) = 3
               end if

C**** Deform footwall if requested ****
```

```
                if (WALLS) then
                   COND(2*YMID_NX - J,I,2*ZMID_NX - L) = COND(J,I,L)
                end if
              end if
60          continue
80        continue
100     continue

        end



C**********************************************************************
C
C                         SUBROUTINE Get_Limts()
C
C   Get_Limts() returns the indecies into the DEL array of the interval
C   of length LENGTH centered in the DEL(2,*) array. It assumes a C  C
C   symmetric grid and takes the cell whose centroid is closest to the end
C   of the interval. Called by In_Fault() and K_File().
C
C                       LOCAL VARIABLES
C
C   MDL_NDX    The index of the middle layer if an odd number of layers
C              or the index of the lower of 2 middle layers an odd number
C              of layers
C   MIDDLE     The middle of the DEL grid
C
C**********************************************************************
        subroutine Get_Limts(CONT_L,CONT_U,LAYERS,LENGTH,DEL,NUM)

        integer CONT_L, CONT_U, NUM
        real LENGTH, DEL(2,NUM)
        character*1 LAYERS

C***** LOCAL VARIABLES *****
        integer MDL_NDX
        real MIDDLE

        MIDDLE = (Max1(DEL(2,1),DEL(2,NUM)) + DEL(1,1)/2)/2.0
        MDL_NDX = Nint(NUM/2.0 + 0.6)

        CONT_L = MDL_NDX
        do while (Abs(DEL(2,CONT_L) - MIDDLE) .LT. LENGTH/2)
          CONT_L = CONT_L + 1
        end do

C***** choose the closer of CONT_L & CONT_L-1 ******
        if (Abs(Abs(DEL(2,CONT_L) - MIDDLE) - LENGTH/2 + 0.0001) .GE.
       #    Abs(Abs(DEL(2,CONT_L-1) - MIDDLE) - LENGTH/2)) then
          CONT_L = CONT_L - 1
        end if

C***** Find index for CONT_U assuming symmetric grid *****
        if (LAYERS .EQ. '3') then
          CONT_U = 1 + (NUM - CONT_L)
        else
          CONT_U = 1
          call Find_NX(DEL,DEL(2,1)+DEL(1,1)/2-LENGTH,CONT_L,NUM,2)
        end if

        end
```

```
C**********************************************************************
C                         SUBROUTINE Get_data
C
C     Get_data() gets the fault and grid data.
C     Called by main program.
C
C**********************************************************************
      subroutine Get_data(OUT_FILE,HAVE_DAT,FAULT_ON,WALLS,DELX,DELY,
     #                    DELZ,MAX_X, MAX_Y, MAX_Z)

      real         DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z)
      integer      MAX_X, MAX_Y, MAX_Z, NUM_X, NUM_Y, NUM_Z
      character*8 OUT_FILE, IN_FILE, F_MODEL*1
      logical      HAVE_DAT, FAULT_ON, WALLS

C****  local variables ****
      character*12 YorN*1, FILEX, FILEY, FILEZ, DP_CHOICE*1
      logical   F_EXIST

      common /GRID/ NUM_X, NUM_Y, NUM_Z
      common /FAULT/ FAULT_LN, FAULT_DP, DISPLACE, F_WIDTH, F_DIAM,
     #               F_MODEL

      Write(*,10)
10    format(//////,' Enter the INPUT file name (no extension) >> ',$)
      Read '(A8)', IN_FILE


C*******  read in grid data *******
      FILEX = IN_FILE(1:(Index(IN_FILE,' ')-1)) // '.col     '
      FILEY = IN_FILE(1:(Index(IN_FILE,' ')-1)) // '.row     '
      FILEZ = IN_FILE(1:(Index(IN_FILE,' ')-1)) // '.lay     '

      call Get_Grid(FILEX,NUM_X,DELX,MAX_X,F_EXIST)
      if (.NOT. F_EXIST) Return

      call Get_Grid(FILEY,NUM_Y,DELY,MAX_Y,F_EXIST)
      if (.NOT. F_EXIST) Return

      call Get_Grid(FILEZ,NUM_Z,DELZ,MAX_Z,F_EXIST)
      if (.NOT. F_EXIST) Return

C**** rearrange DELZ(2,*) so DELZ(2,1) is largest value ****
      call Reverse(DELZ,NUM_Z)

      Write(*,50)
50    format(/,' Enter the OUTPUT file name (no extensions) >> ',$)
      Read '(A8)', OUT_FILE

C******** Determine if there is a fault; if so, get fault data *******
      Write(*,60)
60    format(/,' Do You want to insert a fault? (Y/N) >> ',$)
      Read '(A)', YorN
      FAULT_ON = (YorN .EQ. 'Y' .OR. YorN .EQ. 'y')

      if (FAULT_ON) then
        Write(*,70)
70      format(//,' Enter the LENGTH of the fault >> ',$)
        Read(*,*) FAULT_LN

C**** Adjust fault length so fault ends meet cell boundaries ****
        XMID = (DELX(2,NUM_X) + DELX(1,NUM_X)/2)/2
        call Find_NX(DELX,XMID-FAULT_LN/2,NX,NUM_X,1)
        FAULT_LN = 2*(XMID - (DELX(2,NX) - DELX(1,NX)/2)) - 0.001
        Write(*,72) FAULT_LN
72      format(' Actual fault length = ',F5.1)

C***** Choose one of the 3 available fault angles *****
        Write(*,80)
80      format(/,' Choose a fault DIP ANGLE (degrees) ',/,
     #   'Enter  1 for 45   2 for 90 >> ',$)
        Read '(A)', DP_CHOICE
        do while (DP_CHOICE .NE. '1' .AND. DP_CHOICE .NE. '2')
          Write(*,85)
85        format(//,' MUST CHOOSE 1 or 2!  REENTER  >> ',$)
          Read '(A)', DP_CHOICE
        end do
        if (DP_CHOICE .EQ. '1') then
          FAULT_DP = 45.0
        else
          FAULT_DP = 90.0
        end if

        Write(*,90)
90      format(/,' Enter the maximum DISPLACEMENT >> ',$)
        Read(*,*) DISPLACE

        Write(*,100)
100     format(/,' Enter the maximum fault-zone WIDTH >> ',$)
        Read(*,*) F_WIDTH

        Write(*,110)
110     format(/,' Enter the FAULT-PLANE DIAMETER  >> ',$)
        Read(*,*) F_DIAM

        WALLS = .FALSE.
        if (DP_CHOICE .EQ. '2') then
          Write(*,120)
120       format(/,' Deform both walls (B) or hanging wall only (H)?'
     #      '  >> ',$)
          Read '(A)', YorN
          WALLS = (YorN .EQ. 'B'  .OR.  YorN .EQ. 'b')
        end if

        Write(*,130)
130     format(/,' Fault Zone K Distribution:',
     #       ' Variable K (1) or Variable Fault Diameter (2)  >> ',$)
        Read '(A)', F_MODEL
      end if

      HAVE_DAT = .TRUE.
      end


C*********************************************************************
C                     SUBROUTINE Get_Geo()
C
C   Get_Geo gets hydrogeologic data.
C   Called by K_File().
C
C*********************************************************************
      subroutine Get_Geo(LAYERS,THICK,K,MAX_THK,KF_MAX,DELZ,
     #                   NUM_Z,MAX_Z,FAULT_ON)

      real THICK, K(6), MAX_THK, KF_MAX, DELZ(2,MAX_Z)
      integer NUM_Z
      character*1 LAYERS, ANS
      logical FAULT_ON
      Write(*,20)
20    format(////,' Enter number of geologic layers (2 or 3) >>',$)
      Read '(A)', LAYERS

      do while (LAYERS .NE. '2'.AND. LAYERS .NE. '3')
        Write(*,30)
30      format(//,' Number of layers must be 2 or 3! Reenter >> ',$)
        Read '(A)', LAYERS
      end do

      Write(*,40) MAX_THK
40    format(//,' Enter aquifer thickness ( < ',F5.1,') >> ',$)
      Read(*,*) THICK


      do while (THICK .GE. MAX_THK)
        Write(*,50) MAX_THK
50      format(//,' MUST BE LESS THAN ',F5.1,'! Reenter >> ',$)
        Read(*,*) THICK
      end do

C**** Adjust thickness to align with cell boundaries ****
      ZMID = (DELZ(2,1) + DELZ(1,1)/2)/2.0
      call Find_NX(DELZ,ZMID+THICK/2,NXU,NUM_Z,2)
      call Find_NX(DELZ,ZMID-THICK/2,NXL,NUM_Z,2)
      THICK = (DELZ(2,NXU) + DELZ(1,NXU)/2) -
     #         (DELZ(2,NXL) - DELZ(1,NXL)/2) - .0001
      Write(*,55) THICK
55    format(' Actual thickness of aquifer is ',F6.2)

      Write(*,60)
60    format(//,' Enter conductivity of LOWER AQUITARD >> ',$)
      Read(*,*) K(1)
      K(3) = K(1)

      Write(*,70)
70    format(/,' Enter conductivity of the AQUIFER >> ',$)
      Read(*,*) K(2)

      if (LAYERS .EQ. '3') then
        Write(*,80)
80      format(/,' Enter conductivity of UPPER AQUITARD >> ',$)
        Read(*,*) K(3)
      end if

      if (FAULT_ON) then
        Write(*,90)
90      format(/,' Enter minimum FAULT conductivity of LOWER ',
     #         'AQUITARD >> ',$)
        Read(*,*) K(4)

        Write(*,100)
100     format(/' Enter minimum FAULT conductivity of AQUIFER >> ',$)
        Read(*,*) K(5)

        Write(*,110)
110     format(/' Enter minimum FAULT conductivity of UPPER ',
     #         'AQUITARD >> ',$)
        Read(*,*) K(6)

        Write(*,120)
120     format(/,' Enhance X and Z permeability? (Y/N)  >> ',$)
        Read '(A)', ANS
```

```
C*********************************************************************
C                        SUBROUTINE I_Heads
C
C  I_Heads() determines and outputs the initial
C  head file. It is assumed that the heads along the y = CONST
C  boundaries are constant head boundaries and determine the pre-fault
C  flow pattern. It is assumed that there is no vertical gradient on
C  the boundaries (so have same initial heads for each MODFLOW layer).
C  Heads are linearly interpolated from 4 corner values.
C  Called by Initial().
C
C                  LOCAL VARIABLE DESCRIPTIONS
C
C     HD_LL   Initial head in Lower Left cell (row 1,col 1)
C     HD_LR   Initial head in Lower Right cell (row 1,col NUM_X)
C     HD_UL   Initial head in Upper Right cell
C     HD_UR   Initial head in Upper Right cell
C     GRAD*   Linear gradient of heads in direction of interpolation
C**********************************************************************
      subroutine I_Heads(FNAME, DELX, DELY, DELZ,
     #                   MAX_X, MAX_Y, MAX_Z, MESH)

      real DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z),
     #     MESH(NUM_Y,NUM_X)
      character*12 FNAME

C************ LOCAL VARIABLES **************
      real HD_LL, HD_LR, HD_UL, HD_UR, GRAD1, GRAD2

      common /GRID/ NUM_X, NUM_Y, NUM_Z

C******  Output formats ******
1     format(100F9.3,$)
2     format(F9.3)

C**** Get data to determine initial heads *****
      Write(*,10)
10    format(////,' Enter initial head for ROW 1, COLUMN 1 >> ',$)
      Read(*,*) HD_LL
      Write(*,20) NUM_X
20    format(//,' Enter initial head for ROW 1, COLUMN ',I3,' >> ',$)
      Read(*,*) HD_LR
      Write(*,30) NUM_Y
30    format(//,' Enter the initial head for ROW ',I3,' COLUMN 1 >> ',$)
      Read(*,*) HD_UL
      Write(*,40) NUM_Y, NUM_X
40    format(//,' Enter the initial head for ROW ',I3,' COLUMN ',I3,
     #       ' >> ',$)
      Read(*,*) HD_UR
      print *, ' '
      print *, 'WAIT'

      MESH(1,1) = HD_LL
      MESH(1,NUM_X) = HD_LR
      MESH(NUM_Y,1) = HD_UL
      MESH(NUM_Y,NUM_X) = HD_UR

C***** first interpolate along 2 constant y boundaries *****
      GRAD1 = (HD_LR - HD_LL)/(DELX(2,NUM_X) - DELX(2,1))
      GRAD2 = (HD_UR - HD_UL)/(DELX(2,NUM_X) - DELX(2,1))
      do 50 I=2,NUM_X-1
         MESH(1,I) = MESH(1,I-1) + GRAD1*(DELX(2,I) - DELX(2,I-1))
         MESH(NUM_Y,I) = MESH(NUM_Y,I-1) +
     #                  GRAD2*(DELX(2,I) - DELX(2,I-1))
```

```
50    continue

C***** Interpolate along each column *****
      do 70 I=1,NUM_X
        GRAD1 = (MESH(NUM_Y,I) - MESH(1,I))/(DELY(2,NUM_Y)-DELY(2,1))
        do 60 J=2,NUM_Y-1
          MESH(J,I) = MESH(J-1,I) + GRAD1*(DELY(2,J) - DELY(2,J-1))
60      continue
70    continue

C***** Output initial heads to output file *****
      OPEN(UNIT=40,FILE=FNAME,STATUS='UNKNOWN')
      do 100 L = 1,NUM_Z
        do 90 J=1,NUM_Y
          Write(40,1) (MESH(J,I), I=1,NUM_X)
90      continue
100   continue
      Close(40)
      Write(*,110) FNAME
110   format(///,' Initial heads written to file ',A12,/)

      end

C*******************************************************************
C
C                    SUBROUTINE In_Fault()
C
C  In_Fault() changes the conductivity in COND of those cells
C  which represent the fault "plane".
C  The algorithm takes the cell whose centroid is closest to the
C  mathematical fault plane (in the Y direction).
C  It's up to K_File() to vary conductivity along the fault trace.
C  The equation for the line which in x_section is the fault is
C  AY + BZ + C = 0  where A = TAN_DP, B = 1 and C = - (ZMID + TAN_DP*YMID)
C  unless the FAULT_DP = 90 degrees (vertical) in which case A = 1 ,
C  B = 0 and C = -YMID. Called by K_File().
C
C                     LOCAL VARIABLES
C
C  A         Coefficient of Y in expression of fault x-section line
C  B         Coefficient of Z "
C  C         Constant term in "
C  D1        Distance from current point to fault plane
C  D2        Distance from point over from current point to fault
C  D3        Distance from point above & over from current point to fault
C  TAN_DP    Tangent of the fault dip (angle in radians)
C  X_LEFT    Left-most (smallest) index of X on fault trace
C  X_RGHT    Right-most (largest) index of X on fault trace
C  XMID
C  YMID      Y value of midpoint in Y direction
C  YMID_NX   Y index of initial cell near midpoint
C  Y_NX      Y index of current cell
C  ZMID
C  ZMID_NX
C  Z_NX
C*******************************************************************
      subroutine In_Fault(COND,COND_XYZ,K,DELX,DELY,DELZ,MAX_X,MAX_Y,
     #                    MAX_Z,THICK,UP,LOW,LAYERS,KF_MAX,K_CNT,WALLS)

      real DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z), K(6),
     #     UP(MAX_X), LOW(MAX_X), KF_MAX
      double precision    COND_XYZ(MAX_X*MAX_Z,3)
      integer COND(MAX_Y,MAX_X,MAX_Z)
      character*1 LAYERS, F_MODEL
      logical WALLS

C***** LOCAL VARIABLES *****
      real  XMID, YMID, ZMID, D1, D2, D3
      double precision RX, DISP, A, B, C, TAN_DP, COSN
      integer X_LEFT, X_RGHT, XMID_NX, YMID_NX, ZMID_NX, Y_NX, Z_NX,
     #        COUNT
      logical END_FLT
      character*1 FLOW_NM*14

      common /GRID/ NUM_X, NUM_Y, NUM_Z
      common /FAULT/ FAULT_LN, FAULT_DP, DISPLACE, F_WIDTH, F_DIAM,
     #               F_MODEL


C***** Find the indecies into DELX giving the limits of the fault ****
      call Get_Limts(X_RGHT,X_LEFT,'3',FAULT_LN,DELX,NUM_X)

      XMID = (DELX(2,NUM_X) + DELX(1,NUM_X)/2)/2.0
      call Find_NX(DELX,XMID,XMID_NX,NUM_X,1)
      YMID = (DELY(2,NUM_Y) + DELY(1,NUM_Y)/2)/2.0
      call Find_NX(DELY,YMID,YMID_NX,NUM_Y,1)
      ZMID = (DELZ(2,1) + DELZ(1,1)/2)/2.0
      call Find_NX(DELZ,ZMID,ZMID_NX,NUM_Z,2)

C**** Define expression for fault-line in x-section ****
      if (FAULT_DP .LT. 90.0) then
        TAN_DP = DTan(Dble(FAULT_DP)*3.141592/180)
        A = TAN_DP
        B = 1.0
        C = -(ZMID + TAN_DP*YMID)
      else
        A = 1.0
        B = 0.0
        C = -YMID
      end if

C***** Fold hanging wall (& possibly footwall) down *****
      call Fold(COND,COND_XYZ,K,XMID,YMID,ZMID,DELX,DELY,DELZ,A,B,C,
     #          THICK,LAYERS,MAX_X,MAX_Y,MAX_Z,WALLS)

      Y_NX = YMID_NX
      Z_NX = ZMID_NX
      COUNT = 0

C***** Find Z values of upper & lower aquifer contacts @ fault intersec.
**
      RF = FAULT_LN/2.0
      COSN = DCos(3.141592*(1.0/2 - Dble(FAULT_DP)/180))
      do 10 I=X_LEFT,X_RGHT
        RX = Abs(DELX(2,I) - XMID)/RF
        DISP = COSN*DISPLACE*(2*DSqrt(((RX+1)/2.0)**2 - RX**2)*(1-RX))

        if (WALLS) then
          UP(I) = ZMID + THICK/2 + DISP
        else
          UP(I) = ZMID + THICK/2
        end if

        LOW(I) = ZMID - THICK/2 - DISP
10    continue
```

```
C***** get name of and open flow-through-fault file *****
      Write(*,20)
20    format(/,' Enter the name of the flow-through-fault file >> ',$)
      Read '(A)', FLOW_NM
      Open(UNIT=75,FILE=FLOW_NM,STATUS='UNKNOWN')

C***** Assign lower half of the fault *****
      END_FLT = .FALSE.
      do while (.NOT. END_FLT)
        COUNT = COUNT + 1
        D1 = Distance(A,B,C,DELY(2,Y_NX),DELZ(2,Z_NX))
        D2 = Distance(A,B,C,DELY(2,Y_NX+1),DELZ(2,Z_NX))
        D3 = Distance(A,B,C,DELY(2,Y_NX+1),DELZ(2,Z_NX-1))

C***** Select the cell closest to the fault plane *****
        if (D2 .LT. D1  .AND.  D2 .LT. D3) then
          Y_NX = Y_NX + 1
        else if (D3 .LT. D1  .AND.  D3 .LT. D2) then
          Y_NX = Y_NX + 1
          Z_NX = Z_NX - 1
        end if

C***** Insert this part of the fault *****
        call Fault_On(COND,COND_XYZ,K,DELX,DELY,DELZ,X_LEFT,X_RGHT,
     #                Y_NX,Z_NX,THICK,A,B,C,MAX_X,MAX_Y,MAX_Z,UP,
     #                LOW,KF_MAX,K_CNT)

C***** Make sure the upper 1/2 search leaves no gaps *****
        if (COUNT .EQ. 1) then
          YMID_NX = Y_NX
          ZMID_NX = Z_NX
        end if

C***** Make next cell below the current cell *****
        Z_NX = Z_NX + 1
        if (Z_NX .GT. NUM_Z) END_FLT = .TRUE.
      end do


C***** Assign upper half of the fault *****
      Y_NX = YMID_NX
      Z_NX = ZMID_NX - 1
      END_FLT = .FALSE.
      do while (.NOT. END_FLT)
        D1 = Distance(A,B,C,DELY(2,Y_NX),DELZ(2,Z_NX))
        D2 = Distance(A,B,C,DELY(2,Y_NX-1),DELZ(2,Z_NX))
        D3 = Distance(A,B,C,DELY(2,Y_NX-1),DELZ(2,Z_NX+1))

C***** Select the cell closest to the fault plane *****
        if (D2 .LT. D1 .AND. D2 .LT. D3) then
          Y_NX = Y_NX - 1
        else if (D3 .LT. D1  .AND.  D3 .LT. D2) then
          Y_NX = Y_NX - 1
          Z_NX = Z_NX + 1
        end if

C***** Insert this part of the fault *****
         call Fault_On(COND,COND_XYZ,K,DELX,DELY,DELZ,X_LEFT,X_RGHT,
     #                Y_NX,Z_NX,THICK,A,B,C,MAX_X,MAX_Y,MAX_Z,UP,
     #                LOW,KF_MAX,K_CNT)

C***** Make next cell above the current cell *****
        Z_NX = Z_NX - 1
        if (Z_NX .EQ. 0) END_FLT = .TRUE.
      end do

      Close(75)

C**** Vertical flow file for stats *****
      Write(*,30)
30    format(/,' Enter vertical flow file name  >> ',$)
      Read '(A14)', FLOW_NM
      Open(UNIT=75,FILE=FLOW_NM,STATUS='UNKNOWN')

      call Find_NX(DELZ,ZMID-THICK/2,NX_LZ,NUM_Z,2)
      Write(75,40)XMID,DELY(2,YMID_NX),DELZ(2,NX_LZ),
     #            COND_XYZ(COND(YMID_NX,XMID_NX,NX_LZ),3),
     #            XMID,DELY(2,YMID_NX),DELZ(2,NX_LZ-1),
     #            COND_XYZ(COND(YMID_NX,XMID_NX,NX_LZ-1),3)
40    format(F6.2,F6.2,F6.2,F9.3,/,F6.2,F6.2,F6.2,F9.3,/)
      close(75)

      end



C*********************************************************************
C                   SUBROUTINE Initial()
C
C  Initial() manages 2 things: 1) the output of initial
C  head file; 2) output of the boundary file.
C  It is assumed that the heads along the y = CONST
C  boundaries are constant head boundaries and determine the pre-fault
C  flow pattern. It is assumed that there is no vertical gradient on
C  the boundaries (so have same initial heads for each MODFLOW layer).
C  Heads are linearly interpolated from 4 corner values. Note that the
C  uppermost MODFLOW layer of cells is layer 1. Note that the top and
C  bottom layers and sides are taken to be constant head boundaries.
C  Called by Main.
C
C*********************************************************************
      subroutine Initial(OUT_FILE, HAVE_DAT, DELX, DELY, DELZ,
     #                  MAX_X, MAX_Y, MAX_Z, MESH)

      real DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z),
     #     MESH(NUM_Y,NUM_X)
      character*8 OUT_FILE
      logical HAVE_DAT

C***** LOCAL VARIABLES ******
      character*12 FNAME, TRASH*1

      common /GRID/ NUM_X, NUM_Y, NUM_Z

C**** Check to see if data has been entered. If not, abort ****
      if (.NOT. HAVE_DAT) then
        Write(*,5)
5       format(////,'        MUST FIRST ENTER DATA!!! press enter')
        Read '(A)', TRASH
        RETURN
      end if

C*****  Initial heads *****
      FNAME = OUT_FILE(1:(Index(OUT_FILE,' ')-1)) // '.int     '
      call I_Heads(FNAME,DELX,DELY,DELZ,MAX_X,MAX_Y,MAX_Z,MESH)

C***** Boundary arrays *****
```

```
      FNAME = OUT_FILE(1:(Index(OUT_FILE,' ')-1)) // '.bnd     '
      call Bndry(FNAME)

      Write(*,*) ' Press enter to continue'
      read '(A)', TRASH
      end


C*******************************************************************
C
C                        SUBROUTINE K_Effct()
C
C  K_Effct determines the effective conductivity for a cell.
C  It determines the effective diameter of a slice of the fault.
C  It assumes that for 45 degree dip the cells are square and for 63.34
C  the Z/Y ratio of the cells is 2 and for 90 degrees, the size is
C  variable DIAM is the effective diameter of the fault; it depends on
C  which K distribution was chosen.
C  Called by Fault_On().
C
C*******************************************************************
      subroutine  K_Effct(COND_XYZ,K_CNT,K1,K2,K3,Y_NX,Z_NX,A,B,C,DELY,
     #                 DELZ,MAX_X,MAX_Y,MAX_Z,DISP,KF)

      real K1, K2, K3, DELY(2,MAX_Y), DELZ(2,MAX_Z), KF
      double precision A, B, C, COND_XYZ(MAX_X*MAX_Z,3), DISP
      integer  K_CNT, Y_NX, Z_NX
      character F_MODEL

      double precision DIAM

      common /GRID/ NUM_X, NUM_Y, NUM_Z
      common /FAULT/ FAULT_LN, FAULT_DP, DISPLACE, F_WIDTH, F_DIAM,
     #               F_MODEL

      DIAM = F_DIAM
      if (F_MODEL .EQ. '2') DIAM = DISP*F_DIAM

      COND_XYZ(K_CNT,1)  = KF*((DELY(1,Y_NX)-DIAM)*K1/2 + DIAM*K2 +
     #                      (DELY(1,Y_NX)-DIAM)*K3/2)/DELY(1,Y_NX)
      COND_XYZ(K_CNT,2)  = DELY(1,Y_NX)/((DELY(1,Y_NX)-DIAM)/(2*K1)
     #                     + DIAM/K2 + (DELY(1,Y_NX)-DIAM)/(2*K3))
      COND_XYZ(K_CNT,3)  = COND_XYZ(K_CNT,1)

      end


C*******************************************************************
C                          SUBROUTINE K_File()
C
C  K_File() gets the number of geologic layers and conductivities of
C  the layers and fault. It assumes either 2 or 3 layers; if 2 layers,
C  then the upper is an aquifer, the lower is an aquitard. If 3, then
C  the middle is an aquifer, the others aquitards. NOTE that the top
C  MODFLOW layer coresponds to COND(*,*,1). It then assigns to each cell
C  its non-fault conductivity, determines region affected by fault and
C  then alters conductivity in this region to simulate the fault and
C  displacement. To determine displacement, it finds new position of
C  lower and, if 3 layers, upper contact, and then compares each cell
C  to determine which layer it is in.
C  After the final COND array is determined, the cell transmissivities
C  are computed and written. Finally, the leakance arrays are written.
C  Called by Main.
```

```
C
C                          LOCAL VARIABLES
C
C  CONT_L      Index into DELZ of non-fault lower contact. Cell of this
C              index is assumed part of aquifer.
C  CONT_U      Index into DELZ of non-fault upper contact. Cell of this
C              index is assumed part of aquifer.
C  K()         The conductivities of layers and fault (material types):
C              1 bottom aquitard; 2 aquifer; 3 upper aquitard; 4 lower
C              aquitard fault; 5 aquifer fault. 6 up aquitard fault
C              These indicies correspond to the material type of each cell
C              in COND()
C  KF_MAX      Max factor of inceased permeability in X and Z directions
C  FNAME       Name of output file
C  LAYERS      The number of geologic layers (2 or 3)
C  THICK       Thickness of aquifer
C**********************************************************************
      subroutine K_File(O_FILE,HAVE_DAT,FAULT_ON,WALLS,DELX,DELY,DELZ,
     #                  LIMIT,COND,COND_XYZ,UP,LOW,MAX_X,MAX_Y,MAX_Z)
      real DELX(2,NUM_X),DELY(2,NUM_Y),DELZ(2,NUM_Z),
     #         LIMIT(2,MAX_X,MAX_Y), UP(MAX_X), LOW(MAX_X)
      double precision COND_XYZ(MAX_X*MAX_Z,3)
      integer COND(MAX_Y,MAX_X,MAX_Z)
      character*8 O_FILE, F_MODEL*1
      logical  HAVE_DAT, FAULT_ON, WALLS

C***** LOCAL VARIABLES *****
      real K(6), THICK, KF_MAX
      integer CONT_L, CONT_U
      character*12 FNAME, LAYERS*1, TRASH*1

      common /GRID/ NUM_X, NUM_Y, NUM_Z
      common /FAULT/ FAULT_LN, FAULT_DP, DISPLACE, F_WIDTH, F_DIAM,
     #              F_MODEL

C***** OUTPUT FORMATS *****
1     format(120F8.5,$)
2     format(120F9.6,$)
3     format(120F10.5,$)

C**** Check to see if data has been entered. If not, abort ****
      if (.NOT. HAVE_DAT) then
        Write(*,10)
10      format(////,'        MUST FIRST ENTER DATA!!! press enter')
        Read '(A)', TRASH
        RETURN
      end if

C***** Get layer thickness and conductivity data *****
      call Get_Geo(LAYERS,THICK,K,(DELZ(2,1)+DELZ(1,1)/2)/2,
     #             KF_MAX,DELZ,NUM_Z,MAX_Z,FAULT_ON)

C***** Find the Z indecies for lower and upper contacts *****
      ZMID = (DELZ(2,1) + DELZ(1,1)/2)/2.0
      call Find_NX(DELZ,ZMID+THICK/2,CONT_U,NUM_Z,2)
      call Find_NX(DELZ,ZMID-THICK/2,CONT_l,NUM_Z,2)
      print *, 'CONT_L = ',CONT_L,' CONT_U = ',CONT_U

C***** Initialize the COND array. Start w/bottom layer & work up *****
      do 40 L = NUM_Z,CONT_L+1,-1
        do 30 I=1,NUM_X
          do 20 J=1,NUM_Y
            COND(J,I,L) = 1
20        continue
```

```
30          continue
40        continue
          COND_XYZ(1,1) = K(1)
          COND_XYZ(1,2) = K(1)
          COND_XYZ(1,3) = K(1)

          do 70 L=CONT_U,CONT_L
            do 60 I=1,NUM_X
              do 50 J=1,NUM_Y
                COND(J,I,L) = 2
50            continue
60          continue
70        continue
          COND_XYZ(2,1) = K(2)
          COND_XYZ(2,2) = K(2)
          COND_XYZ(2,3) = K(2)
          K_CNT = 2

          if (LAYERS .EQ. '3') then
            do 100 L=1,CONT_U-1
              do 90 I=1,NUM_X
                do 80 J=1,NUM_Y
                  COND(J,I,L) = 3
80              continue
90            continue
100         continue
            COND_XYZ(3,1) = K(3)
            COND_XYZ(3,2) = K(3)
            COND_XYZ(3,3) = K(3)
            K_CNT = 3
          end if

          print *, ' '
          print *, 'WAIT'

C***** Insert fault into COND() ******
          if (FAULT_ON) then
            call In_Fault(COND,COND_XYZ,K,DELX,DELY,DELZ,MAX_X,MAX_Y,
     #                    MAX_Z,THICK,UP,LOW,LAYERS,KF_MAX,K_CNT,WALLS)
          end if

C***** Output ROW transmissivities *****
          FNAME = O_FILE(1:(Index(O_FILE,' ')-1)) // '.cnd      '
          Open(UNIT=41,FILE=FNAME,STATUS='UNKNOWN')
          do 150 L=1,NUM_Z
            do 140 J=1,NUM_Y
              Write(41,1) (COND_XYZ(COND(J,I,L),1)*DELZ(1,L), I=1,NUM_X)
140         continue
150       continue
          Close(41)

          Write(*,160) FNAME
160       format(//,' Transmissivities written to file ',A12)

C***** Output column/row anisotropy factors *****
          FNAME = O_FILE(1:(Index(O_FILE,' ')-1)) // '.ans      '
          Open(UNIT=41,FILE=FNAME,STATUS='UNKNOWN')
          do 200 L=1,NUM_Z
            do 190 J=1,NUM_Y
              Write(41,2) (COND_XYZ(COND(J,I,L),2)/COND_XYZ(COND(J,I,L),1),
     #                     I=1,NUM_X)
190         continue
200       continue
          Close(41)
```

```
      Write(*,210) FNAME
210   format(/,' Anisotropy ratios written to file ',A12)

C***** Output Vcont values to .VCT file *****
      FNAME = O_FILE(1:(Index(O_FILE,' ')-1)) // '.vct     '
      Open(UNIT=41,FILE=FNAME,STATUS='UNKNOWN')

      do 240 L=1,NUM_Z-1
        do 220 J=1,NUM_Y
          Write(41,3) (1.0/((DELZ(1,L)/(2*COND_XYZ(COND(J,I,L),3))) +
     #                 (DELZ(1,L+1)/(2*COND_XYZ(COND(J,I,L+1),3)))),
     #             I=1,NUM_X)
220     continue
240   continue
      Close(41)

      Write(*,260) FNAME
260   format(/,' Vcont values written to file ',A12,/)

      Print *, 'Press ENTER to continue...'
      Read '(A)', TRASH

      end

C**********************************************************************
C                          SUBROUTINE MENU
C**********************************************************************

      subroutine Menu(CHOICE)

      character*1 CHOICE

      Write(*,5)
5     format(//////////////'                                  MENU',/,
     #      '        ______________________________________',/,
     #      '             1    ENTER PROBLEM DATA',/,
     #      '             2    CONDUCTIVITY & LEAKANCE FILES',/,
     #      '             3    INITIAL HEAD AND BOUNDRY FILES',/,
     #      '             4    SLICE FILE',/,
     #      '             5    COMPARE SLICES',/,
     #      '             6    WELL FIELD',/,
     #      '             7    SUMMARY STATISTICS',/,
     #      '             8    EXIT')
      Write(*,10)
10    format(//,'              Enter choice >> ',$)
      Read '(A)', CHOICE
      end

C*****************************************************************
C
C                    FUNCTION Odd()
C
C     If NUM is odd, then Odd() returns .TRUE., else .FALSE.
C
C******************************************************************
      Function Odd(NUM)

      integer NUM
      logical Odd

      Odd = (MOD(NUM,2) .EQ. 1)
      end
```

```
C*****************************************************************
C
C                        FUNCTION Rand()
C
C  Function Rand is a unifom random number generator.
C
C  SEED is the seed value for the generator. It is altered and
C  passed back.
C  Called by Well_Make()
C
C*****************************************************************


      Function Rand(IDUM)

      integer IDUM, IA, IM, IQ, IR, NTAB, NDIV
      real    Rand, AM, EPS, RNMX
      parameter (IA=16807, IM = 2147483647, AM = 1.0/IM,
     #          IQ = 127773, IR = 2836, NTAB = 32,
     #          NDIV = 1 + (IM-1)/NTAB, EPS = 1.2e-7,RNMX = 1.-EPS)
      integer J, K, IV(NTAB), IY
      save IV, IY
      DATA IV /NTAB*0/, IY /0/

      if (IDUM .LE. 0  .OR.  IY .EQ. 0) then
        IDUM = Max(-IDUM,1)
        do 10 J=NTAB+8,1,-1
          K = IDUM/IQ
          IDUM = IA*(IDUM - K*IQ) - IR*K
          if (IDUM .LT. 0) IDUM = IDUM + IM
          if (J .LE. NTAB) IV(J) = IDUM
10      continue
        IY = IV(1)
      end if

      K = IDUM/IQ
      IDUM = IA*(IDUM - K*IQ) - IR*K
      if (IDUM .LT. 0) IDUM = IDUM + IM
      J = I + IY/NDIV
      IY = IV(J)
      IV(J) = IDUM
      Rand = min(AM*IY,RNMX)

      return
      end



C***************************************************************
C
C                       SUBROUTINE Reverse()
C
C  Reverse() reverses the order of the elements in DEL(1,*) and
C  DEL(2,*). Called by  Get_data() and Slice().
C
C***************************************************************
      subroutine Reverse(DEL,NUM)

      real DEL(2,NUM), TEMP(200)

      do 10 I=1,NUM
        TEMP(NUM-I+1) = DEL(1,I)
10    continue
```

```
      do 20 I = 1,NUM
        DEL(1,I) = TEMP(I)
        TEMP(I) = DEL(2,NUM-I+1)
20    continue
      do 30 I=1,NUM
        DEL(2,I) = TEMP(I)
30    continue

      end


C***************************************************************
C
C                     SUBROUTINE Slice()
C
C   Slice takes a 3D head file and allows the ouput of a 2D slice
C   in either an XY, YZ or XZ plane. Output is in 3 column form
C   where first 2 columns are location and third column is head.
C   This is in a format that can be input to SURFER.
C   Called by Main.
C
C                 LOCAL VARIABLES
C -------------------------------------------------------
C  F_EXIST   Logical is .TRUE. if a FILE exists
C  IN_FILE   File name of grid files
C  LIMIT     Logical .TRUE. if output domain is limited
C  NUMX      Number of cells in X direction
C  NUMY
C  NUMZ
C  PLANE     1 if XY slice; 2 if YZ slice; 3 if XZ slice
C  OUT_FILE  Name of output file
C  TEMP      Stores 1 row of heads when input file is being read
C  XYZ       The constant value of the plane to slice
C  XYZ_NX    Index into DEL of closest cell to XYZ
C  XMIN      Minimum X value for output file
C  XMAX      Max X value for output file
C  YMIN
C  YMAX
C  ZMIN
C  ZMAX
C  XTRANS    Amount of translation of output domain in X direction
C  YTRANS
C  ZTRANS
C
C***************************************************************
      subroutine Slice(DELX,DELY,DELZ,MAX_X,MAX_Y,MAX_Z)

      real DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z)

C**** LOCAL VARIABLES ****
      real XYZ, TEMP(2000), XMIN, XMAX, YMIN, YMAX, ZMIN,
     #          ZMAX, XTRANS, YTRANS, ZTRANS
      integer  NUMX, NUMY, NUMZ, XYZ_NX,
     #              XMIN_NX, XMAX_NX, YMIN_NX, YMAX_NX, ZMIN_NX, ZMAX_NX
      character*8 IN_FILE
      character*16 YorN*1, PLANE*1, FILE_NM, OUT_FILE
      logical  F_EXIST, LIMIT, TRANS

1     format(F7.2,F7.2,F8.3)

      Write(*,10)
10    format(//////,' Enter GRID file name (no extension) >> ',$)
      Read '(A8)', IN_FILE
```

```
C*******  read in grid data *******
      FILE_NM = IN_FILE(1:(Index(IN_FILE,' ')-1)) // '.col     '
      call Get_Grid(FILE_NM,NUMX,DELX,MAX_X,F_EXIST)
      if (.NOT. F_EXIST) Return

      FILE_NM = IN_FILE(1:(Index(IN_FILE,' ')-1)) // '.row     '
      call Get_Grid(FILE_NM,NUMY,DELY,MAX_Y,F_EXIST)
      if (.NOT. F_EXIST) Return

      FILE_NM = IN_FILE(1:(Index(IN_FILE,' ')-1)) // '.lay     '
      call Get_Grid(FILE_NM,NUMZ,DELZ,MAX_Z,F_EXIST)
      if (.NOT. F_EXIST) Return
C**** rearrange DELZ(2,*) so DELZ(2,1) is largest value ****
      call Reverse(DELZ,NUMZ)

      print *
      print *,' NUMX:',NUMX,'   NUMY:',NUMY,'  NUMZ:',NUMZ

      Write(*,20)
20    format(/,' Enter the complete name of file to slice >> ',$)
      Read '(A16)', FILE_NM

C***** Get and open 3D heads file *****
      Inquire(FILE=FILE_NM,EXIST=F_EXIST)
      if (.NOT. F_EXIST) then
        Write(*,30) FILE_NM
30      format(////,' FILE ',A16,' DOES NOT EXIST!!!  press enter.')
        Read '(A)', YorN
        Return
      end if

      Open(UNIT = 25, FILE = FILE_NM, STATUS = 'OLD')

      YorN = 'Y'
      do while (YorN .EQ. 'Y'  .OR.  YorN .EQ. 'y')
        Rewind 25

        Write(*,40)
40      format(/,' Enter the name of the OUTPUT file >> ',$)
        Read '(A16)', OUT_FILE
        Open(UNIT = 30, FILE = OUT_FILE, STATUS = 'UNKNOWN')

C***** Determine which slice is desired *****
        Write(*,50)
50      format(/,' Enter 1 for XY plane, 2 for YZ and 3 for XZ  >> ',$)
        Read '(A1)', PLANE
        do while (PLANE .NE. '1'  .AND.  PLANE .NE. '2' .AND.
     #            PLANE .NE. '3')
          Write(*,60)
60        format(//,' ENTER 1, 2 or 3 >> ',$)
          Read '(A1)', PLANE
        end do

C***** Do they want to limit the output domain *****
        Write(*,61)
61      format(/,' Want to RESTRICT the output domain? (Y/N) >> ',$)
        Read '(A1)', YorN
        LIMIT = (YorN .EQ. 'Y'  .OR.  YorN .EQ. 'y')
        XMIN_NX = 1
        XMAX_NX = NUMX
        YMIN_NX = 1
        YMAX_NX = NUMY
        ZMIN_NX = NUMZ
        ZMAX_NX = 1
```

```
C***** Do they want to translate the domain *****
        Write(*,62)
62      format(/,' Want to TRANSLATE the domain? (Y/N)  >> ',$)
        Read '(A1)', YorN
        TRANS = (YorN .EQ. 'Y'  .OR.  YorN .EQ. 'y')
        XTRANS = 0.0
        YTRANS = 0.0
        ZTRANS = 0.0

C***** Find the specific plane and output data *****

C**** If XY plane ****
        if (PLANE .EQ. '1') then
          Write(*,100)
100       format(/,' Plane defined by Z = ',$)
          Read(*,*) XYZ
          do while (XYZ .GE. (DELZ(2,1)+DELZ(1,1)/2))
            Write(*,105) DELZ(2,1)+DELZ(1,1)/2
105         format(//,' MUST HAVE Z < ',F5.1,'. REENTER >> ',$)
            Read(*,*) XYZ
          end do
          call Find_NX(DELZ,XYZ,XYZ_NX,NUMZ,2)

          if (LIMIT) then
            Write(*,110)
110         format(/,' Enter XMIN  >> ',$)
            Read(*,*) XMIN
            Write(*,115)
115         format(/,' Enter XMAX  >> ',$)
            Read(*,*) XMAX
            if (XMAX .GT. DELX(2,NUMX)+DELX(1,NUMX)/2) then
              XMAX = DELX(2,NUMX)
            end if

            Write(*,120)
120         format(/,' Enter YMIN  >> ',$)
            Read(*,*) YMIN
            Write(*,125)
125         format(/,' Enter YMAX  >> ',$)
            Read(*,*) YMAX
            if (YMAX .GT. DELY(2,NUMY)+DELY(1,NUMY)/2) then
              YMAX = DELY(2,NUMY)
            end if

            call Find_NX(DELX,XMIN,XMIN_NX,NUMX,1)
            call Find_NX(DELX,XMAX,XMAX_NX,NUMX,1)
            call Find_NX(DELY,YMIN,YMIN_NX,NUMY,1)
            call Find_NX(DELY,YMAX,YMAX_NX,NUMY,1)
          end if

C***** Get translation vector *****
          if (TRANS) then
            Write(*,130)
130         format(/,' Enter value of X translation  >> ',$)
            Read(*,*) XTRANS
            Write(*,135)
135         format(/,' Enter value of Y translation  >> ',$)
            Read(*,*) YTRANS
          end if

          do 160 L=1,NUMZ
            do 150 J=1,NUMY
              Read(25,*) (TEMP(K),K=1,NUMX)
              if (L .EQ. XYZ_NX) then
              do 140 I=XMIN_NX,XMAX_NX
                if (J .GE. YMIN_NX  .AND.  J .LE. YMAX_NX) then
                  Write(30,1) DELX(2,I)+XTRANS, DELY(2,J)+YTRANS,
     #                        TEMP(I)
                end if
140           continue
            end if
150       continue
160     continue

C***** if YZ plane *****
        else if (PLANE .EQ. '2') then
          Write(*,200)
200       format(/,' Plane defined by X = ',$)
          Read(*,*) XYZ
          do while (XYZ .GE. (DELX(2,NUMX)+DELX(1,NUMX)/2))
            Write(*,205) DELX(2,NUMX)+DELX(1,NUMX)/2
205         format(//,' MUST HAVE X < ',F5.1,'. REENTER >> ',$)
            Read(*,*) XYZ
          end do
          call Find_NX(DELX,XYZ,XYZ_NX,NUMX,1)

          if (LIMIT) then
            Write(*,210)
210         format(/,' Enter YMIN  >> ',$)
            Read(*,*) YMIN
            Write(*,215)
215         format(/,' Enter YMAX  >> ',$)
            Read(*,*) YMAX
            if (YMAX .GT. DELY(2,NUMY)+DELY(1,NUMY)/2) then
              YMAX = DELY(2,NUMY)
            end if
            Write(*,220)
220         format(/,' Enter ZMIN  >> ',$)
            Read(*,*) ZMIN
            Write(*,225)
225         format(/,' Enter ZMAX  >> ',$)
            Read(*,*) ZMAX
            if (ZMAX .GT. DELZ(2,1)+DELZ(1,1)/2) then
              ZMAX = DELZ(2,1)
            end if

            call Find_NX(DELZ,ZMIN,ZMIN_NX,NUMZ,2)
            call Find_NX(DELZ,ZMAX,ZMAX_NX,NUMZ,2)
            call Find_NX(DELY,YMIN,YMIN_NX,NUMY,1)
            call Find_NX(DELY,YMAX,YMAX_NX,NUMY,1)
          end if

C***** Get translation vector *****
          if (TRANS) then
            Write(*,230)
230         format(/,' Enter value of Y translation  >> ',$)
            Read(*,*) YTRANS
            Write(*,235)
235         format(/,' Enter value of Z translation  >> ',$)
            Read(*,*) ZTRANS
          end if

          do 260 L=1,NUMZ
            do 250 J=1,NUMY
              Read(25,*) (TEMP(K),K=1,NUMX)
              if (L .GE. ZMAX_NX  .AND.   L .LE. ZMIN_NX   .AND.
     #             J .GE. YMIN_NX  .AND.  J .LE. YMAX_NX) then
                Write(30,1) DELY(2,J)+YTRANS, DELZ(2,L)+ZTRANS,
     #                        TEMP(XYZ_NX)
              end if
250         continue
260       continue

C***** if XZ plane *****
        else
          Write(*,300)
300        format(/,' Plane defined by Y = ',$)
          Read(*,*) XYZ
          do while (XYZ .GE. (DELY(2,NUMY)+DELY(1,NUMY)/2))
            Write(*,305) DELY(2,NUMY)+DELY(1,NUMY)/2
305         format(//,' MUST HAVE Y < ',F5.1,'. REENTER >> ',$)
            Read(*,*) XYZ
          end do

          call Find_NX(DELY,XYZ,XYZ_NX,NUMY,1)

          if (LIMIT) then
            Write(*,310)
310         format(/,' Enter XMIN  >> ',$)
            Read(*,*) XMIN
            Write(*,315)
315         format(/,' Enter XMAX  >> ',$)
            Read(*,*) XMAX
            if (XMAX .GT. DELX(2,NUMX)+DELX(1,NUMX)/2) then
              XMAX = DELX(2,NUMX)
            end if

            Write(*,320)
320         format(/,' Enter ZMIN  >> ',$)
            Read(*,*) ZMIN
            Write(*,325)
325         format(/,' Enter ZMAX  >> ',$)
            Read(*,*) ZMAX
            if (ZMAX .GT. DELZ(2,1)+DELZ(1,1)/2) then
              ZMAX = DELZ(2,1)
            end if

            call Find_NX(DELX,XMIN,XMIN_NX,NUMX,1)
            call Find_NX(DELX,XMAX,XMAX_NX,NUMX,1)
            call Find_NX(DELZ,ZMIN,ZMIN_NX,NUMZ,2)
            call Find_NX(DELZ,ZMAX,ZMAX_NX,NUMZ,2)
          end if

C***** Get translation vector *****
          if (TRANS) then
            Write(*,330)
330         format(//,' Enter value of X translation  >> ',$)
            Read(*,*) XTRANS
            Write(*,335)
335         format(/,' Enter value of Z translation  >> ',$)
            Read(*,*) ZTRANS
          end if

C***** Output the slice file *****
          do 360 L=1,NUMZ
            do 350 J=1,NUMY
              Read(25,*) (TEMP(K),K=1,NUMX)
              if (J .EQ. XYZ_NX  .AND.   L .GE. ZMAX_NX   .AND.
     #            L .LE. ZMIN_NX) then
                do 340 I=XMIN_NX,XMAX_NX
                  Write(30,1) DELX(2,I)+XTRANS, DELZ(2,L)+ZTRANS,
     #                        TEMP(I)
```

```
340             continue
              end if
350           continue
360         continue
          end if
          Close(30)
          Write(*,400)
400       format(///,' Want another slice from this file? (Y/N) >> ',$)
          Read '(A1)', YorN
        end do

        Close(25)
        end



C***************************************************************
C
C                SUBROUTINE Stats()
C
C  Stats takes the head file generated from MODFLOW and compiles
C  the following summary statistics:
C
C     i) Maximum head variation from no-fault
C    ii) Maximum head gradient across fault
C   iii) Flow through aquifer at fault plane
C   Called by Main.
C
C --------------------LOCAL VARIABLES ---------------------
C
C   K1   Ky for Cell read from flow file
C   K2   Ky for cell at y-1 of cell read from flow file
C
C***************************************************************
      subroutine Stats(HEADS,DELX,DELY,DELZ,MAX_X,MAX_Y,MAX_Z)

      real DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z),
     #     HEADS(MAX_Y,MAX_X,MAX_Z)


C***** LOCAL VARIABLES *****
      real MAX_DIF, MAX_HGRD, MAX_VGRD, MAX_XGRAD, XGRAD, NF_HEAD(100),
     #     FAULT_DP, FAULT_LN, YMID, K1, K2, K3
      double precision TAN_DP, A, B, C
      integer X_LEFT, X_RGHT, X_NX, Y_NX, Z_NX
      character*16 INFILE, FNAME, FNAME2, TRASH*1
      logical F_EXIST

      Write(*,10)
10    format(//////,' Enter the GRID file name (no extension) >> ',$)
      Read '(A8)', INFILE

C*******  read in grid data *******
      FNAME = INFILE(1:(Index(INFILE,' ')-1)) // '.col      '
      call Get_Grid(FNAME,NUMX,DELX,MAX_X,F_EXIST)
      if (.NOT. F_EXIST) Return

      FNAME = INFILE(1:(Index(INFILE,' ')-1)) // '.row      '
      call Get_Grid(FNAME,NUMY,DELY,MAX_Y,F_EXIST)
      if (.NOT. F_EXIST) Return

      FNAME = INFILE(1:(Index(INFILE,' ')-1)) // '.lay      '
      call Get_Grid(FNAME,NUMZ,DELZ,MAX_Z,F_EXIST)
      if (.NOT. F_EXIST) Return

      print *,' NUMX: ',NUMX,'  NUMY: ',NUMY,'  NUMZ: ',NUMZ

C**** rearrange DELZ(2,*) so DELZ(2,1) is largest value ****
      call Reverse(DELZ,NUMZ)

C***** Open fault and no-fault head files *****
      Write(*,20)
20    format(/,' Enter the name of FAULT/HEAD data file  >> ',$)
      Read '(A16)', FNAME
      Inquire(FILE=FNAME,EXIST=F_EXIST)
      if (.NOT. F_EXIST) then
        Write(*,30) FNAME
30      format(///,' FILE ',A16,' DOES NOT EXIST!. Press enter...',$)
        Read '(A1)', TRASH
        Return
      end if
      Open(UNIT=20,FILE=FNAME,STATUS='OLD')

C***** Choose one of the 2 available fault angles *****
      Write(*,40)
40    format(/,' Choose a fault DIP ANGLE (degrees) ',/,
     #   ' Enter  1 for 45   2 for 90 >> ',$)
      Read '(A)', DP_CHOICE
      do while (DP_CHOICE .NE. '1' .AND. DP_CHOICE .NE. '2')
        Write(*,50)
50      format(//,' MUST CHOOSE 1 or 2!  REENTER  >> ',$)
        Read '(A)', DP_CHOICE
      end do
      if (DP_CHOICE .EQ. '1') then
        FAULT_DP = 45.0
      else
        FAULT_DP = 90.0
      end if

      Write(*,55)
55    format(/,' Enter the fault LENGTH  >> ',$)
      Read(*,*) FAULT_LN

      Write(*,80)
80    format(//,' Enter the name of NO-FAULT/HEAD data file  >> ',$)
      Read '(A16)', FNAME
      Inquire(FILE=FNAME,EXIST=F_EXIST)
      if (.NOT. F_EXIST) then
        Write(*,90) FNAME
90      format(///,' FILE ',A16,' DOES NOT EXIST!. Press enter...',$)
        Read '(A1)', TRASH
        Return
      end if
      Open(UNIT=30,FILE=FNAME,STATUS='OLD')

C***** Read head data into HEADS array *****
      do 110 L=1,NUMZ
        do 100 J=1,NUMY
          Read(20,*) (HEADS(J,I,L), I=1,NUMX)
100     continue
110   continue


C******* Compute MAX FAULT/NO_FAULT head difference *******
      MAX_DIF = 0.0
      do 140 L=1,NUMZ
        do 130 J=1,NUMY
```

```
          Read(30,*) (NF_HEAD(I),I=1,NUMX)
          do 120 I=1,NUMX
            if (Abs(HEADS(J,I,L)-NF_HEAD(I)) .GT. Abs(MAX_DIF)) then
              MAX_DIF = HEADS(J,I,L) - NF_HEAD(I)
            end if
120       continue
130     continue
140   continue
      Close(20)
      Close(30)

C***** Find the indecies into DELX giving the limits of the fault ****
      XMID = (DELX(2,NUMX) + DELX(1,NUMX)/2)/2.0
      YMID = (DELY(2,NUMY) + DELY(1,NUMY)/2)/2.0
      ZMID = (DELZ(2,1) + DELZ(1,1)/2)/2.0
      call Find_NX(DELX,XMID-FAULT_LN/2,X_LEFT,NUMX,1)
      call Find_NX(DELX,XMID+FAULT_LN/2,X_RGHT,NUMX,1)

C**** Define expression for fault-line in x-section ****
      if (FAULT_DP .LT. 90.0) then
        TAN_DP = DTan(Dble(FAULT_DP)*3.141592/180)
        A = TAN_DP
        B = 1.0
        C = -(ZMID + TAN_DP*YMID)
      else
        A = 1.0
        B = 0.0
        C = -YMID
      end if

C** Find MAX Head Y-grad across fault plane & Z-grad to either side **
      MAX_HGRD = 0.0
      do 180 L=1,NUMZ
        Y = -(1.0/A)*(B*DELZ(2,L) + C)
        call Find_NX(DELY,Y,Y_NX,NUMY,1)
        do 170 I=X_LEFT,X_RGHT
          GRAD = Abs((HEADS(Y_NX-1,I,L)-HEADS(Y_NX+1,I,L))/
     #                (DELY(2,Y_NX-1)-DELY(2,Y_NX+1)))
          if (GRAD .GT. MAX_HGRD) MAX_HGRD = GRAD
170     continue
180   continue

C***** Find flow through aquifer at fault zone *****
      Write(*,200)
200   format(/,' Enter the flow/fault file name  >> ',$)
      Read '(A16)',FNAME

      if (Index(FNAME,' ') .GT. 1) then
        Inquire(FILE=FNAME,EXIST=F_EXIST)
        if (.NOT. F_EXIST) then
          Write(*,210) FNAME
210       format(///,' FILE ',A16,' NOT FOUND! Press enter...',$)
          Read '(A1)', TRASH
          Return
        end if
        Open(UNIT=75,FILE=FNAME,STATUS='OLD')

        Q = 0.0
        if (FAULT_DP .EQ. 90.0) then
          do while (.TRUE.)
            Read(75,*,END=280) X,Y,Z,K1,K2
            call Find_NX(DELX,X,X_NX,NUMX,1)
            call Find_NX(DELY,Y,Y_NX,NUMY,1)
            call Find_NX(DELZ,Z,Z_NX,NUMZ,2)
          GRADY = (HEADS(Y_NX-1,X_NX,Z_NX)-HEADS(Y_NX,X_NX,Z_NX))/
     #            (DELY(2,Y_NX) - DELY(2,Y_NX-1))
          CONDY = (DELY(1,Y_NX) + DELY(1,Y_NX-1))/
     #            (DELY(1,Y_NX)/K1 + DELY(1,Y_NX-1)/K2)

          Q = Q + CONDY*GRADY*DELX(1,X_NX)*DELZ(1,Z_NX)
        end do
      elseif (FAULT_DP .EQ. 45.0) then
        do while (.TRUE.)
          Read(75,*,END=280) X,Y,Z,K1,K2,K3
          call Find_NX(DELX,X,X_NX,NUMX,1)
          call Find_NX(DELY,Y,Y_NX,NUMY,1)
          call Find_NX(DELZ,Z,Z_NX,NUMZ,2)

          GRADY = (HEADS(Y_NX-1,X_NX,Z_NX)-HEADS(Y_NX,X_NX,Z_NX))/
     #            (DELY(2,Y_NX) - DELY(2,Y_NX-1))
          GRADZ = (HEADS(Y_NX-1,X_NX,Z_NX-1)-HEADS(Y_NX-1,X_NX,Z_NX))/
     #            (DELZ(2,Z_NX-1) - DELZ(2,Z_NX))
          CONDY = (DELY(1,Y_NX) + DELY(1,Y_NX-1))/
     #            (DELY(1,Y_NX)/K1 + DELY(1,Y_NX-1)/K2)
          CONDZ = (DELZ(1,Z_NX-1) + DELZ(1,Z_NX-1))/
     #            (DELZ(1,Z_NX-1)/K3 + DELZ(1,Z_NX)/K2)

          Q = Q + CONDY*GRADY*DELX(1,X_NX)*DELZ(1,Z_NX) +
     #            CONDZ*GRADZ*DELX(1,X_NX)*DELY(1,Y_NX)
        end do
      end if

280   Close(75)
      end if

C***** Get data and compute vertical gradient *****
      Write(*,300)
300   format(/,' Enter the name of vertical flow file  >> ',$)
      Read '(A14)', FNAME2

      if (Index(FNAME2,' ') .GT. 1) then
        Open(UNIT=75,FILE=FNAME2,STATUS='OLD')
        Read(75,*)X1,Y1,Z1,K1
        Read(75,*)X2,Y2,Z2,K2
        call Find_NX(DELX,X1,NX_X,NUMX,1)
        call Find_NX(DELY,Y1,NX_Y,NUMY,1)
        call Find_NX(DELZ,Z1,NX_Z1,NUMZ,2)
        VGRAD = (HEADS(NX_Y,NX_X,NX_Z1-1) - HEADS(NX_Y,NX_X,NX_Z1))/
     #          (DELZ(2,NX_Z1-1) - DELZ(2,NX_Z1))
        VCOND = (DELZ(1,NX_Z1-1) + DELZ(1,NX_Z1))/
     #          (DELZ(1,NX_Z1)/K1 + DELZ(1,NX_Z1-1)/K2)
        VQ = VCOND*VGRAD*DELX(1,NX_X)*DELY(1,NX_Y)
        Close(75)
      end if

C***** Find the max vertical gradient *****
      MAX_VGRD = 0.0
      do 360 L=1,NUMZ-1
        do 350 J=1,NUMY
          do 340 I=1,NUMX
            VGRAD = (HEADS(J,I,L) - HEADS(J,I,L+1))/
     #              (DELZ(2,L) - DELZ(2,L+1))
            if (Abs(VGRAD) .GT. Abs(MAX_VGRD)) then
              MAX_VGRD = VGRAD
            end if
340       continue
350     continue
360   continue
```

```
C*** Find MAX X Gradient ****
      MAX_XGRAD = 0.0
      do 460 L=1,NUMZ
        do 450 J=1,NUMY
          do 440 I=1, NUMX-1
            XGRAD = Abs((HEADS(J,I+1,L) - HEADS(J,I,L))/
     #                (DELX(2,I+1) - DELX(2,I)))
            if (XGRAD .GT. MAX_XGRAD) then
               MAX_XGRAD = XGRAD
            end if
440       continue
450     continue
460   continue

      Write(*,500) MAX_DIF, MAX_XGRAD,MAX_HGRD, MAX_VGRD
500   format(////,' MAX FAULT - NO_FAULT: ',F7.3,
     #        /,' MAX X GRAD: ',F7.3,
     #        /,' MAX Y GRAD: ',F7.3,
     #        /,' MAX Z GRAD: ',F7.3)

      if (Index(FNAME,' ') .GT. 1) then
        Write(*,510) Q
510     format(/,' FLOW THROUGH AQUIFER at FAULT: ',F6.3,/)
      end if

      Write(*,520) VGRAD,VQ
520   format(' VERTICAL GRAD at FAULT APEX: ',F6.3,/,
     #        ' VERTICAL FLOW at FAULT APEX: ',F9.6,/)

      print *
      Print *,' Press ENTER to continue...'
      Read '(A)',TRASH

      end


C***************************************************************
C
C                  SUBROUTINE WELLS()
C
C  Wells() controls the flow for well field operations
C
C***************************************************************
      Subroutine Wells(HAVE_DAT,HEADS,DELX,DELY,DELZ,MAX_X,MAX_Y,
     #                          MAX_Z)

      real DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z),
     #     HEADS(MAX_Y,MAX_X,MAX_Z)
      logical HAVE_DAT

      character*1 CHOICE

      CHOICE = "0"

      do while (CHOICE .NE. "3")
        Write(*,10)
10      format(////////,' Would you like to: ',
     #            /,'       1   CREATE a random well file '
     #            /,'       2   FIND HEADS from well file',
     #            /,'       3   QUIT',
     #           //,'       ENTER CHOICE  >> ',$)

        Read '(A)', CHOICE
```

```
          if (CHOICE .EQ. '1') then
            call Well_Make(DELX,DELY,DELZ,MAX_X,MAX_Y,MAX_Z)
          else if (CHOICE .EQ. '2') then
            call Well_Field(HAVE_DAT,HEADS,DELX,DELY,DELZ,MAX_X,MAX_Y,
     #                      MAX_Z)
          end if
        end do

        end


C***************************************************************
C
C                 SUBROUTINE Well_Bnds()
C
C
C  Subroutine Well_Bnds() takes the (X,Y) location of a well,
C  the HEAD file and fault info, and returns the upper and
C  lower boundaries of the aquifer.
C  Called by Well_Make()
C
C***************************************************************
        Subroutine Well_Bnds(X,Y,TOP,BOT,DISP,WIDTH,NUMX,NUMY,NUMZ,
     #                       DELX,DELY,DELZ,XMID,YMID,MAX_X,MAX_Y,
     #                       MAX_Z,DBL_DISP)

        real X, Y, BOT, TOP, WIDTH, DISP, DELX(2,MAX_X),
     #       DELY(2,MAX_Y),DELZ(2,MAX_Z), XMID, YMID
        integer NUMX, NUMY, NUMZ, MAX_X, MAX_Y, MAX_Z
        logical DBL_DISP

C****  LOCAL VARIABLES ****
        real ZMID, DISP_X, PHI, TEMP, R, RX, WDTH_X, Y_WDTH_X
        integer X_NX, Y_NX, NX_TEMP

        ZMID = (DELZ(2,1) + DELZ(1,1)/2)/2.0
        call Find_NX(DELX,X,X_NX,NUMX,1)
        call Find_NX(DELY,Y,Y_NX,NUMY,1)

        COSN = 1.0

C**** find displacement and fault width for given X value
        RX = Abs(2*(DELX(2,X_NX) - XMID)/20)
        R = 2.0*Sqrt(((1.0+RX)/2)**2 - RX**2)*(1.0-RX)
        DISP_X = DISP*R
        WDTH_X = WIDTH*R

        PHI = ASin(DISP_X/WDTH_X)

C**** Find the MAX Y limit *****
        TEMP = YMID + WDTH_X*Cos(PHI)
        call Find_NX(DELY,TEMP,NX_TEMP,NUMY,1)
        Y_WDTH_X = DELY(2,NX_TEMP) - YMID

        if (Y .GT. YMID+Y_WDTH_X  .OR.  Y .LT. YMID-Y_WDTH_X  .OR.
     #     (Y .LT. YMID  .AND.  .NOT.(DBL_DISP))) then
          TOP = ZMID + 1
          BOT = ZMID - 1
        else
          TEMP = WDTH_X*Cos(PHI)
          XY_DSPL = (TEMP - Abs(DELY(2,Y_NX) - YMID))*Tan(PHI)
          XY_DSPL = XY_DSPL*(DELY(2,Y_NX) - YMID)/Abs(DELY(2,Y_NX) - YMID)
          TOP = ZMID + 1 - XY_DSPL
          BOT = ZMID - 1 - XY_DSPL
        end if

        end


C************************************************************
C
C                SUBROUTINE Well_Field()
C
C  Well_Field takes a head file (3D) and a set of wells (X,Y
C  location and screen interval) and averages (integrates) over
C  the screened interval for each well. Results outputted to a file.
C  File give the limits of top and bottom of well screens.
C  Called by Wells().
C
C              LOCAL VARIABLES
C
C  FNAME    Name of file to open
C  F_EXIST  Logical = .TRUE. if file exists
C  INTG     The integrated (averaged) head
C  NUMX     Number of cells in X direction
C  NUMY
C  NUMZ
C  X,Y      X-Y location of well
C  X_NX     X index of cell containing X
C  Y_NX     Y index of cell containing Y
C  ZTOP     Z value of top of well screen
C  ZBOT     Z value of bottom of well screen
C
C************************************************************
      subroutine Well_Field(HAVE_DAT,HEADS,DELX,DELY,DELZ,MAX_X,MAX_Y,
     #                     MAX_Z)

      real DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z),
     #     HEADS(MAX_Y,MAX_X,MAX_Z)
      logical HAVE_DAT

C***** LOCAL VARIABLES *****
      real X, Y, ZTOP, ZBOT, INTG, h, XMID, YMID
      integer X_NX, Y_NX, ZTOP_NX, ZBOT_NX, NUMX, NUMY, NUMZ
      character*12 INFILE*8, FNAME, TRASH*1, ANS*1
      logical F_EXIST

1     format(F7.2,F7.2,F7.2)

      Write(*,10)
10    format(//////,' Enter the GRID file name (no extension) >> ',$)
      Read '(A8)', INFILE

C*******  read in grid data *******
      FNAME = INFILE(1:(Index(INFILE,' ')-1)) // '.col     '
      call Get_Grid(FNAME,NUMX,DELX,MAX_X,F_EXIST)
      if (.NOT. F_EXIST) Return

      FNAME = INFILE(1:(Index(INFILE,' ')-1)) // '.row     '
      call Get_Grid(FNAME,NUMY,DELY,MAX_Y,F_EXIST)
      if (.NOT. F_EXIST) Return

      FNAME = INFILE(1:(Index(INFILE,' ')-1)) // '.lay     '
      call Get_Grid(FNAME,NUMZ,DELZ,MAX_Z,F_EXIST)
      if (.NOT. F_EXIST) Return
```

```
C**** rearrange DELZ(2,*) so DELZ(2,1) is largest value ****
      call Reverse(DELZ,NUMZ)

      XMID = (DELX(2,NUMX) + DELX(1,NUMX)/2)/2.0
      YMID = (DELY(2,NUMY) + DELY(1,NUMY)/2)/2.0

      print *, 'XMID = ',XMID,'  YMID = ',YMID

C***** Open head, well data and output files *****
      Write(*,20)
20    format(/,' Enter the name of HEAD data file  >> ',$)
      Read '(A12)', FNAME
      Inquire(FILE=FNAME,EXIST=F_EXIST)
      if (.NOT. F_EXIST) then
        Write(*,30) FNAME
30      format(///,' FILE ',A12,' DOES NOT EXIST!. Press enter...',$)
        Read '(A1)', TRASH
        Return
      end if
      Open(UNIT=20,FILE=FNAME,STATUS='OLD')

      Write(*,40)
40    format(/,' Enter name of WELL data file >> ',$)
      Read '(A12)', FNAME
      Inquire(FILE=FNAME,EXIST=F_EXIST)
      if (.NOT. F_EXIST) then
        Write(*,50) FNAME
50      format(///,' FILE ',A12,' DOES NOT EXIST!. Press enter...',$)
        Read '(A1)', TRASH
        Return
      end if
      Open(UNIT=22,FILE=FNAME,STATUS='OLD')

      Write(*,55)
55    format(/,' Are these piezometers or screened wells? (P/W)  >> ',$)
      Read '(A)', ANS

      Write(*,60)
60    format(/,' Enter name of OUTPUT file >> ',$)
      Read '(A12)', FNAME
      Open(UNIT=24,FILE=FNAME,STATUS='UNKNOWN')

C***** Read head data into HEADS array *****
      do 90 L=1,NUMZ
        do 80 J=1,NUMY
          Read(20,*) (HEADS(J,I,L), I=1,NUMX)
80      continue
90    continue

C***** Loop through well file *****
      do while (.TRUE.)
        INTG = 0.0
        Read(22,*,END=200) X,Y,ZTOP,ZBOT
        call Find_NX(DELX,X,X_NX,NUMX,1)
        call Find_NX(DELY,Y,Y_NX,NUMY,1)
        call Find_NX(DELZ,ZBOT,ZBOT_NX,NUMZ,2)
        call Find_NX(DELZ,ZTOP,ZTOP_NX,NUMZ,2)

        if (ANS .EQ. 'W'  .OR.  ANS .EQ. 'w') then
C***** Integerate over Z *****
          do 100 L=ZTOP_NX,ZBOT_NX-1
            h = DELZ(2,L)-DELZ(2,L+1)
            INTG = INTG+(HEADS(Y_NX,X_NX,L)+HEADS(Y_NX,X_NX,L+1))*h/2.0
100       continue
```

```
            INTG = INTG/(DELZ(2,ZTOP_NX)-DELZ(2,ZBOT_NX))
            Write(24,1) X-XMID,Y-YMID,INTG
          else
            Write(24,1) X-XMID, Y-YMID, HEADS(Y_NX,X_NX,ZBOT_NX)
          end if
        end do

200     Close(20)
        Close(22)
        Close(24)

        Write(*,220) FNAME
220     format(//,' Integrated well data written to file ',A12,//,
     #         ' Press Enter...',$)
        Read '(A1)', TRASH

        end

C***************************************************************
C
C                 SUBROUTINE Well_Make()
C
C  Well_Make() Generates a uniformly random well field for a given head
C  field. For each well, i.e. each (X,Y), it then determines the
C  aquifer boundaries.
C  Fault assumed to be vertical and fault length assumed to be 20.
C  Aquifer thickness assumed to be 2.
C  Results outputted to a file.
C  Called by Wells().
C
C               LOCAL VARIABLES
C
C  DBL_DISP Logical. TRUE if displacement on both walls
C  FNAME    Name of file to open
C  F_EXIST  Logical = .TRUE. if file exists
C  NUMX     Number of cells in X direction
C  NUMY
C  NUMZ
C  X,Y      X-Y location of well
C  X_NX     X index of cell containing X
C  Y_NX     Y index of cell containing Y
C  ZTOP     Z value of top of well screen
C  ZBOT     Z value of bottom of well screen
C
C***************************************************************
        subroutine Well_Make(DELX,DELY,DELZ,MAX_X,MAX_Y,MAX_Z)

        real DELX(2,MAX_X), DELY(2,MAX_Y), DELZ(2,MAX_Z)

C***** LOCAL VARIABLES *****
        real DISP, BOT, TOP, WIDTH, X, Y, XMID, YMID, Rand
        integer WELL_NUM, FILE_NUM, X_NX, Y_NX,
     #          SEED, NUMX, NUMY, NUMZ
        character*16 INFILE*8, FNAME, ANS*1
        logical F_EXIST, DBL_DISP

1       format(F7.2,F7.2,F7.2)

        SIZE = 10.0

        Write(*,10)
10      format(//////,' Enter the GRID file name (no extension) >> ',$)
        Read '(A8)', INFILE
```

```
C*******  read in grid data *******
      FNAME = INFILE(1:(Index(INFILE,' ')-1)) // '.col     '
      call Get_Grid(FNAME,NUMX,DELX,MAX_X,F_EXIST)
      if (.NOT. F_EXIST) Return

      FNAME = INFILE(1:(Index(INFILE,' ')-1)) // '.row     '
      call Get_Grid(FNAME,NUMY,DELY,MAX_Y,F_EXIST)
      if (.NOT. F_EXIST) Return

      FNAME = INFILE(1:(Index(INFILE,' ')-1)) // '.lay     '
      call Get_Grid(FNAME,NUMZ,DELZ,MAX_Z,F_EXIST)
      if (.NOT. F_EXIST) Return

C**** rearrange DELZ(2,*) so DELZ(2,1) is largest value ****
      call Reverse(DELZ,NUMZ)

      XMID = (DELX(2,NUMX) + DELX(1,NUMX)/2)/2.0
      YMID = (DELY(2,NUMY) + DELY(1,NUMY)/2)/2.0

      print *, 'XMID = ',XMID,'  YMID = ',YMID

C***** Get data and open output file *****
      Write(*,40)
40    format(/,' Double displacement (D) or Single (S)  >> ',$)
      Read '(A)', ANS
      DBL_DISP = (ANS .EQ. 'D'  .OR.  ANS .EQ. 'd')

      Write(*,50)
50    format(/,' Enter the DISPLACEMENT >> ',$)
      Read(*,*) DISP

      Write(*,60)
60    format(/,' Enter the fault WIDTH >> ',$)
      Read(*,*) WIDTH

      Write(*,70)
70    format(/,' Enter the number of files to generate >> ',$)
      Read(*,*) FILE_NUM

C**** Loop through each well file ****
      SEED = -Int(Secnds(0)*0.01)
      do 120 I=1,FILE_NUM
        Write(*,80)
80      format(/,' Enter name of the output WELL file >> ',$)
        Read '(A16)', FNAME
        Open(UNIT=22,FILE=FNAME,STATUS='NEW')

        Write(*,90)
90      format(/,' Enter the number of wells in file >> ',$)
        Read(*,*) WELL_NUM

C*** Loop to find each well location and aquifer boundaries ****
        do 110 J=1,WELL_NUM
          X = XMID + 2*SIZE*RAND(SEED) - SIZE
          call Find_NX(DELX,X,X_NX,NUMX,1)
          do while (X .LT. XMID-SIZE  .OR.  X .GT. XMID+SIZE  .OR.
     #              XMID - DELX(2,X_NX) .EQ. 0)
            X = XMID + 2*SIZE*RAND(SEED) - SIZE
            call Find_NX(DELX,X,X_NX,NUMX,1)
          end do

          Y = YMID + 2*SIZE*RAND(SEED) - SIZE
          call Find_NX(DELY,Y,Y_NX,NUMY,1)
          do while (Y .LT. YMID-SIZE  .OR.  Y .GT. YMID+SIZE  .OR.
     #              YMID - DELY(2,Y_NX) .EQ. 0)
          Y = YMID + 2*SIZE*RAND(SEED) - SIZE
          call Find_NX(DELY,Y,Y_NX,NUMY,1)
        end do

        call Well_Bnds(X,Y,TOP,BOT,DISP,WIDTH,NUMX,NUMY,NUMZ,DELX,
     #                DELY,DELZ,XMID,YMID,MAX_X,MAX_Y,MAX_Z,DBL_DISP)
        Write(*,100) X,Y,TOP,BOT
100     format(F8.3,F8.3,F8.3,F8.3)
        Write(22,100)X, Y, Top, BOT
110   continue
      Close(22)
120 continue

    end
```

APPENDIX B

As mentioned in Chapter 3, an attempt was made to implement the fault model by determining the effective hydraulic conductivity tensor for a finite-difference cell containing a fault of arbitrary dip and thickness. This initial attempt for a dip of 45° and a square 2-D cell is described here.

The principle axes of conductivity in a finite-difference cell with a fault running through opposite corners are aligned with the fault, i.e. one axis is parallel to the fault plane and one is perpendicular to it. Thus, in principle, to define the finite-difference problem, one can either specify the directions of the principle axes for each cell and give the principle values of conductivity, or one can use global coordinates and supply a full conductivity tensor for each cell. The latter approach was selected because it is a relatively straight forward matter to modify MODFLOW to use a full tensor for each cell.

Two problems then remain: i) how to determine the values of the elements of the full tensor for a given cell, and ii) how to define the cross terms of the continuity equation at cell boundaries. These two problems are discussed below.

Determining Tensor Values

An idea suggested by John Wilson and Peter Kitanidis for numerically approximating the values of the full conductivity tensor given K_1, K_f and K_2 (see Figure 3.2) is as follows:

simulate a single cell with a square finite-difference grid, the diagonal cells representing the fault. Cells above the diagonal are assigned a conductivity K_1, cells along the diagonal K_f and cells below the diagonal K_2.

It is assumed that the tensor values are independent of boundary conditions. The boundary conditions chosen were periodic in the Z direction and periodic with a fixed head drop in the Y direction. The periodic boundary conditions in the Z direction are given by $h_{i,1} = h_{i,10}$ where $h_{i,j}$ is the head at cell (i,j) with i the Y index and j the Z index. This boundary condition creates a net Z gradient of 0 yet allows for flow, Q_z, across the upper and lower boundaries. The Y-direction boundary condition is given by $h_{1,j} = h_{10,j} - \Delta h$ where Δh is a fixed head increment defining the net head gradient in the Y direction. To allow for a unique solution, one of the cells in the grid must be assigned an actual constant head value; the cell and the value can be chosen arbitrarily.

When the finite-difference model is run, the resulting head values can be used to approximate the vertical flow, Q_z, by:

$$Q_z = \sum_{i=1}^{10} Q_{zi} = \sum_{i=1}^{10} \Delta Y \frac{2K_{i,1}K_{i,2}}{K_{i,1}+K_{i,2}} \frac{(h_{i,1} - h_{i,2})}{\Delta Z}$$

For this exercise, $\Delta Y = \Delta Z = 1$.

Given Q_z, we can approximate $K_{yz} = K_{zy}$ of the tensor using Darcy's Law by $K_{yz} = Q_z/\Delta h_y$ where Δh_y is the head change across the simulated fault-containing cell in the Y direction; this is because the net Z gradient across the simulated fault-containing cell is 0. Using Q_y, we can approximate the diagonal tensor elements in an analogous manner with $K_{yy} = K_{zz}$ due to symmetry.

To test the obtained tensor values, the following was done: construct first a 100 x 100 regular finite-difference grid with diagonal elements assigned conductivity K_2, upper-diagonal elements assigned conductivity K_1 and lower-diagonal elements assigned conductivity K_3. Pick any set of boundary conditions. Run the model and calculate the resulting Q_y and Q_z over the 100 boundary cells. Next, set up a 10 x 10 grid where each cell represents a 10 x 10 unit of the previously-run 100 x 100 grid. Therefore the upper-diagonal cells of the 10 x 10 are assigned a conductivity K_1, the lower-diagonal cells of the 10 x 10 are assigned a conductivity K_3 and the diagonal elements of the 10 x 10 must be assigned the full conductivity tensor values previously determined. Thus, the finite-difference solver must be able to utilize the full tensor for each cell. If the tensor values were computed correctly, the head fields for the 10 x 10 and the 100 x 100 grids must be approximately the same as well as the computed values of Q_y and Q_z.

The results of the above experiment were poor. The head fields and Q values were poorly matched with better results obtained using a variety of different approximations to the K tensor (such as zero off-diagonal elements with harmonic means for the diagonal elements). For example, a 10 x 10 grid was set up with $K_x = K_y = .001$ on the diagonal and $K_x = K_y = .1$ otherwise. This yielded a $K_{xx} = K_{yy} = .048$ and $K_{xy} = K_{yx} = .036$ for the whole grid. A 100 x 100 grid was constructed with $K_x = K_y = .001$ on the diagonal and $K_x = K_y = .1$ otherwise resulting in $K_{xx} = K_{yy} = .072$ and $K_{xy} = K_{yx} = .026$ for the whole grid. This 100 x 100 grid was then simulated by replacing the 100 x 100 grid with a 10 x 10 grid with $K_x = K_y = .1$ for the offdiagonal elements and the full tensor defined for the diagonal elements with $K_{xx} = K_{yy} = .048$ and $K_{xy} = K_{yx} = .036$ (thus simulating the 10 x 10 blocks containing the diagonal in the 100 x 100 grid). If the tensor was correct and the simulation done properly, this simulation should have resulted in the same values for K_{xx} and K_{xy} as the 100 x 100 grid. Instead the resulting values were $K_{xx} = .082$ and $K_{xy} = .0095$.

It is unclear whether the problem lies in the logic or the implementation. A possible source of error was in the method used to incorporate the cross terms into the continuity equation (see below); several alternatives were tried with no improvement. It seems likely that one or more of the assumptions made is incorrect. Further study into this problem is recommended.

Incorporation of Cross Terms

The 2-D finite-difference equation for steady-state flow (assuming square cells and unit depth) used in the experiments detailed above is:

$$
\begin{gathered}
K_{yy_{i+\frac{1}{2},j}}(h_{i+1,j} - h_{i,j}) + K_{yy_{i-\frac{1}{2},j}}(h_{i-1,j} - h_{i,j}) + \\
K_{zz_{i,j+\frac{1}{2}}}(h_{i,j+1} - h_{i,j}) + K_{zz_{i,j-\frac{1}{2}}}(h_{i,j-1} - h_{i,j}) + \\
K_{yz_{i+\frac{1}{2},j}} \frac{\Delta h_{z_{i+\frac{1}{2},j}}}{m} + K_{yz_{i-\frac{1}{2}}} \frac{\Delta h_{z_{i-\frac{1}{2},j}}}{m} + \\
K_{zy_{i,j+\frac{1}{2}}} \frac{\Delta h_{y_{i,j+\frac{1}{2}}}}{m} + K_{zy_{i,j-\frac{1}{2}}} \frac{\Delta h_{y_{i,j-\frac{1}{2}}}}{m} = 0
\end{gathered}
$$

where $h_{i,j}$ is the head in cell (i,j), $Kyy_{i+1/2,j}$ is the principle conductivity in the Y direction at the boundary between cells (i,j) and (i+1,j) (this is generally taken as the harmonic mean of $Kyy_{i,j}$ and $Kyy_{i+1,j}$), $Kyz_{i+1/2,j}$ is the yz tensor cross term at the boundary between cells (i,j) and (i+1,j), $\Delta hz_{i+1/2,j}$ is an approximation to (dh/dz)Δz at the boundary between cells (i,j) and (i+1,j) and m is the number of ΔZ lengths over which the gradient is estimated. The first four terms represent the standard five-point finite-difference method and the next four terms are approximations to the cross terms. What must be decided is i) how to approximate Kzy and Kyz, and ii) how to measure the Z gradient at a Y cell boundary (as well as how to measure the Y gradient at a Z cell boundary).

In the experiments to verify the tensor terms for a fault containing cell, only the diagonal cells have non-zero cross terms. Therefore, one can not use a harmonic mean (or a geometric mean) of the Kzy values for cells (i,j) and (i+1,j) to approximate $Kzy_{i+1/2,j}$ because this will give $Kzy_{i+1/2,j} = 0$ (and similarly for other cross-term approximations along cell boundaries). The effect is to ignore tensor cross terms when solving the finite-difference equations. Some sort of weighted arithmetic mean would give non-zero cross-term approximations. However, this requires some justification.

APPENDIX C

MODFLOW, the U.S.G.S. finite-difference groundwater flow code, was chosen to do the computer simulations of the fault-containing region. A pre/post processor was written in FORTRAN to do the following:

i) build input files to MODFLOW that describe the particular fault being modeled;

ii) take 2D slices of the resulting 3D head field in any of the three coordinate directions;

iii) insert wells or piezometers into the 3D head field and look at the observed heads;

iv) calculate various summary statistics from the 3D head field.

The code for the pree/post processor can be found in Appendix A.

MODFLOW requires a coordinate system with the origin at one corner of the modeled region. For convenience, a reference coordinate system is used in this report that has its origin at the centroid of the modeled region; this point always correponds to the centroid of the fault plane. Thus, the two coordinate systems are related by the following translation

$$(X_r, Y_r, Z_r) = (X_{mf}-X_{max}/2, Y_{mf}-Y_{max}/2, Z_{mf}-Z_{max}/2)$$

where the subscripts r and mf indicate coordinates in the reference and MODFLOW coordinate systems, respectively, and X_{max} is the maximum X value of the modeled region in the MODFLOW coordinate system. Unless otherwise specified, all coordinates given in this text will be in the reference coordinate system and the r subscript will be dropped.

If the fault dip < 90°, the principle axes of conductivity are not aligned with the MODFLOW coordinate axes. In this case, one would have to give the values for the full conductivity tensor for each cell to MODFLOW. MODFLOW can be modified to accept these values. The problems are i) in determining what these tensor values are, and ii) how to express the cross terms at the cell boundaries. A good deal of time was spent trying to determine the tensor values without success. These efforts are described in Appendix B.

The Kx, Ky/Kx and the VCONT (vertical conductance) files for a particular simulation were constructed by the preprocessor. The general algorithm for determining the Kx and Ky values for a particular simulation is as follows:

1) assign pre-fault conductivities to each cell in the region;
2) deform hanging wall
   A) determine the limits in the X and Y directions, (X_{min}, X_{max}) and (Y_{min}, Y_{max}), of the deformed portion of the hanging wall over all layers;
   B) determine the equation, $AY + BZ + C = 0$, of the line for the trace of the fault in a YZ cross section;
   C) loop through each cell in the deformed region and determine which post-fault layer it is in. Assign it the appropriate Kx and Ky values;

```
3) insert the fault:
   A) Loop through each layer, determining, from the equation
      $AY + BZ + C = 0$, the correct Y value of the cells in
      that layer which contain the fault;
      i) loop through each cell from $X_{min}$ to $X_{max}$ in the layer
         with Y as above (i.e. in the fault)
         a) determine the amount of displacement for the given
            X value using equation 2.1;
         b) based on the displacement, use the appropriate
            model to determine $Kx_{ef}$, $Ky_{ef}$ and $Kz_{ef}$ for the cell.
            Store the values;
4) output files for Kx, Ky/Kx and VCONT.
```

The algorithm for step 2 C is as follows:

```
1) loop from $X_{min}$ to $X_{max}$:
   A) calculate the displacement, d, and the width, W, for the
      current value of X;
   B) loop from $Y_{min}$ to $Y_{max}$:
      i) determine the Z limits of the displaced aquifer, $Z_{min}$
         and $Z_{max}$, for the current X and Y;
     ii) loop from $Z_{max}$ to $Z_{min}$:
         a) determine if the current cell is in the hanging
            wall; if yes, find which geologic layer it is in
            by determining its location in relation to the
            top and bottom aquifer contacts. Assign that
            cell to the appropriate geologic layer.
```

The specifics of any step can be found in the code in Appendix A.