GEOLOGIC AND STOCHASTIC MODELING OF FRACTURE SYSTEMS IN ROCKS

by

VIOLETA MINTCHEVA IVANOVA

Master of Science in Civil and Environmental Engineering
Massachusetts Institute of Technology, February 1995

Civil Engineer: Structures
University of Architecture, Civil Engineering, and Geodesy
Sofia, Bulgaria, 1991

Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY
IN GEOENVIRONMENT AND GEOTECHNOLOGY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 1998

© 1998 Massachusetts Institute of Technology. All rights reserved.

Signature of Author _____
Department of Civil and Environmental Engineering
January 15, 1998

Certified by _____
Professor Herbert H. Einstein, Thesis Co-Supervisor

Certified by _____
Professor Daniele Veneziano, Thesis Co-Supervisor

Accepted by _____
Professor Joseph M. Sussman
Chairman, Department Committee on Graduate Studies

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
FEB 13 1998
LIBRARIES
ARCHIVES

GEOLOGIC AND STOCHASTIC MODELING OF FRACTURE SYSTEMS IN ROCKS

by

Violeta Mintcheva Ivanova

Submitted to the Department of Civil and Environmental Engineering
on January 15, 1998, in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy in Geoenvironment and Geotechnology

ABSTRACT

Fracture systems in the geologic environment consist of multiple sets of discontinuities that often interconnect to create conditions for mechanical failure and flow. A comprehensive study of natural fracture systems resulted in classification of five major fracture-producing geologic settings: folds, crustal faults, crustal extension, thermal contraction, and central intrusive and extrusive structures. The geometries of the fracture systems in different geologic settings, determined by the geologic sequence of fracture formation, rock lithology, and regional and local principal stress directions, can be described in terms of fracture intensity, orientation, and region of existence.

Since the field sampling methods are essentially 1D (logs and cores) and 2D (outcrop maps), there is usually great uncertainty about the actual 3D geometry and connectivity of natural rock fracture systems. To reduce that uncertainty, the model, proposed in this thesis, incorporates the geology into the inference procedure. The model reproduces rock fracture systems through superposition of hierarchical fractures sets, using three stochastic processes that reflect inherent relationships between the fracture system geometry and the underlying mechanics. The model uses Poisson plane process, Poisson line tessellation, and random 3D rotation and translation, to represent orientations of potential fracture planes, fracture intensity, and relationships to local geologic structures. The 3D fracture system model was implemented in the computer program GEOFRAC, written in C/C++ for UNIX.

The geologic and numerical model was applied in a case study of the fracture system in Permian sedimentary rocks in West Texas. The study included development of a conceptual model for the geologic evolution of the fracture system in the reservoir formations of the Yates oil field. Numerical simulations involved generation of fracture sets related to the regional stresses and depositional trends, and to the asymmetric anticlinal structure of reservoir strata. Comparison of the numerically generated fracture system to field data showed a good match by dip, strike, and spacing of fractures intersected by boreholes.

Thesis co-supervisors:
Herbert H. Einstein, Professor of Civil and Environmental Engineering
Daniele Veneziano, Professor of Civil and Environmental Engineering

ACKNOWLEDGEMENTS

The contribution of many people and organizations to this thesis is greatly appreciated.

First of all, I would like to thank the members of my doctoral committee: Prof. Herbert H. Einstein, Prof. Daniele Veneziano, and Prof. Patricia Culligan from MIT, and Dr. William S. Dershowitz from Golder Associates Inc. (Seattle, WA).

Professor Einstein guided me throughout my graduate work at MIT, which resulted in a Master's degree in 1995, and in this Doctoral dissertation. His knowledge of engineering geology, rock mechanics, and fracture system modeling, and his talent to motivate and inspire his students, helped me more than anything else to accomplish the objectives I set when I started graduate school.

Professor Veneziano was always available to answer questions on the theoretical aspects of my research. His ability to clearly explain even the most complex mathematical and statistical problems was invaluable help in the search of efficient modeling approaches.

Professor Culligan proof-read the thesis several times and suggested numerous changes that improved the final layout.

Dr. Dershowitz (long distance from Seattle) had a great impact on many aspects of my work, and most of all on the case study project. His critique of the thesis helped me learn a lot about the practical application of academic research.

I thank the engineers and geologists of Marathon Oil Company for their help during the Yates field case study. Most of all, I would like to thank Eugene E. Wadleigh who by phone, email, and in person during my visit to the office in Midland, Texas, answered a multitude of questions about sedimentary and petroleum geology, and about the geology of the Yates field. Brendan Curran and Tom Fitzsimmons helped me with the data collection in Midland, and organized a memorable field trip. Suzie Thompson in Midland made numerous plots and maps, which facilitated the modeling at MIT. Scott Tinker and Mike Uland in the Petroleum Technology Center in Littleton, Colorado, helped me a lot with the geologic Stratamodel of the Yates field.

I thank Thomas Meyer who ran hundreds of computer simulations and spent many hours in library research, thus providing great help during the last, most intense, semester of my Ph.D. research at MIT. Baffour Abedi also ran many computer simulations as part of the MIT Undergraduate Research Opportunity Program. I thank my office-mate Jennie Hango for the beautiful photographs of columnar joints (included in Chapter 2), and for inspiring my interest in geology. I also thank Prof. Hans Herda from the University of Massachusetts for his help with the statistical analysis during his visits to MIT.

From 1993 to 1996 the research project was sponsored by the US Superfund Hazardous Substances Basic Research Program administered by the Center for Environmental Health Sciences at MIT. During 1996 and 1997 the research project was sponsored by the National Institute for Petroleum and Energy Research of the US Department of Energy through a grant to the project on "Fractured Reservoir Discrete Feature Network Technologies" on which MIT, Golder Associates Inc., and Marathon Oil Company worked in collaboration.

DEDICATION

After more than twenty-two years of structured education, including five and a half years in one of the best graduate schools in the world, and two advanced degrees, the most important lesson that I have learned is that the only thing that really matters in life is the kindness and love one gives to others and is fortunate to receive from family and friends. This dissertation is dedicated to three people that more than anybody else helped me learn that lesson.

To my best friend Zlatozara Kuzmova-Takeva for never giving up telling me (even if only by phone during the last five years) that one can find happiness only within oneself and no titles or money can buy it.

To my brother Vladimir, for his constant loving support and encouragement.

And to Professor Herbert H. Einstein, for his infinite patience and generosity.

Many thanks to all others who, in one way or another, helped me learn in school and, most importantly, in life: mom and dad, and the rest of my family in Bulgaria - I would not be where I am without you; the friends I met at MIT, especially Beth Henderson-Rasmussen and her family, Marika Santagata-Sinfield, Joe Sinfield, and Emo Todorov; Julie Siromahova, who would not let me forget (long distance from Europe) that one should have some fun in life too; MIT sailing master Hatch Brown and others who taught me how to sail - nothing like planing a laser in 20 knots to take one's mind off mundane worries!; Elaine and Andrew from "the Group" who listened; the extremely helpful administrative staff at the MIT Civil Engineering headquarters, especially Cynthia Stewart and Pat Dixon; and last, but not least, my fellow graduate students Tony Simone, Patrick Kinnicutt, Jennie Hango, Antonio Bobet, Greg Da Re, and many others, who provided a very enjoyable study and work environment during the years at MIT.

***

We are here to learn. Everything that happens to us helps our learning.

Dr. M. Scott Peck
Further Along the Road Less Traveled

***

Imagination is more important than knowledge.

Albert Einstein

***

If I speak in the tongues of men and of angels, but have not love, I am a noisy gong or a clanging cymbal. And if I have prophetic powers, and understand all mysteries and all knowledge, and if I have all faith, so as to remove mountains, but have not love, I am nothing ...
... Love is patient and kind; love is not jealous or boastful; it is not arrogant or rude. Love does not insist on its own way; it is not irritable or resentful; it does not rejoice at wrong, but rejoices in the right ...
... Love never ends; as for prophecies, they will pass away; as for tongues, they will cease; as for knowledge, it will pass away... So faith, hope, love abide, those three; but the greatest of them is love.

1 Corinthians: **13**
The Holy Bible

***

GEOLOGIC AND STOCHASTIC MODELING OF FRACTURE SYSTEMS IN ROCKS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Rocks usually are considered to be materials of high compressive and shear strength and low permeability. Such assumptions may be too optimistic in the engineering design of constructed facilities, such as radioactive waste disposal sites, tunnels, and dams, since rocks are often fractured. In the geologic environment, fractures interconnect and create conditions for mechanical failure and fluid flow in rocks which, when intact, would have high strength and low permeability. For example, **Figure 1.1** shows a rock cut at Libby Dam, Montana, where fractures formed shear failure surfaces producing slides of the overlying rock material. Similar problems may occur in tunnels where rock wedges, bound by fractures, can fail due to loss of kinematic and kinetic stability. **Figure 1.2** shows an example of fluid flow though fractures in Cambridge Argillite in an excavation in Newton, Massachusetts. In low temperatures, the water can freeze in the fractures, thus causing additional problems such as impoundment of groundwater and further cracking through ice expansion.

The mechanics of brittle fracturing has been studied through laboratory experiments on small specimens and numerical models of the observed stress-strain-strength behavior [e.g. Bobet 1997]. There are two major types of fractures: 1) tensile fractures (joints) which open in planes parallel to the maximum and intermediate principal stresses, σ_1 and σ_2, and orthogonal to the minimum principal stress, σ_3; and 2) shear fractures (faults), which form in the plane of σ_2, and make an angle of less than 45° with the direction of σ_1. The interaction of two fractures in a compressive stress has also been studied through laboratory experiments and numerical modeling [Reyes & Einstein 1991; Bobet 1997]. Laboratory experiments give essential knowledge about the brittle behavior of rocks; however, methods to reproduce the conditions of rock fracturing in nature exist to a limited extent only.

In nature, fractures form in heterogeneous materials, under stresses that vary in time and space due to various geologic mechanisms that act simultaneously or in sequence. As a result, fracture systems consist of multiple sets of joints and faults with various intensity, orientation, and connectivity. The three-dimensional geometry of the entire fracture system, i.e. cumulative fracture area, orientations of fractures relative to one another and fracture connectivity, controls the flow and mechanical stability in fractured rocks. Therefore, in order to reliably predict the stability and conductivity of fractured rocks, one needs to have a good model that reproduces rock fracture systems as three-dimensional interconnected networks.

The flow chart in **Figure 1.3** illustrates the research approach that is followed in this thesis. The phenomenon of interest is the 3D geometry of natural rock

fracture systems. Hence, the objective is to develop a model for realistic 3D representation of the fracture system geometry, based on the underlying geologic mechanisms. **Figure 1.4** illustrates how the development of such a 3D model builds on the prior research in the field of fracture system modeling.

There are three major types of models of rock fracture systems: mechanical, geometric, and geometric-mechanical. Mechanical models represent the true mechanics of fracture initiation and propagation. However, because of the complex analysis needed to express fracturing even in a relatively simple stress field, mechanical models have so far been developed only in two dimensions [e.g. Wu & Pollard 1992]. Hence, they have limited application to practical cases that involve 3D visualization and the study of natural rock fracture systems.

Geometric models, on the other hand, focus on the representation of the final 3D geometry of a rock fracture system, rather than on the mechanisms that have created it. Avoiding analysis of the fracturing mechanisms, the most advanced geometric models use stochastic methods to generate randomly located and oriented fractures in 3D space, so that the statistics of the numerically generated systems match the statistics of the modeled real fracture systems [e.g. models implemented in the software FRACMAN by Golder Associates, Inc.; Dershowitz et al. 1994]. Geometric models have a wide range of practical applications. Their only shortcoming is that they rely largely on field sampling of fractures for inference of the geometric parameters of the 3D system. Field sampling methods are usually one-dimensional (intersections of fractures by logs and cores) or two-dimensional (trace maps of fracture outcrops). Since fracture systems with different 3D geometry and connectivity may have very similar 1D or 2D exposures, there is great uncertainty and potential for error when the inference of the parameters of a fracture system model are based on such field data.

Geometric-mechanical models attempt to combine the advantages, and avoid the shortcomings of both mechanical and geometric models. Geometric-mechanical models represent the geometry of rock fracture systems on the basis of relationships between mechanics and geometry, without modeling the mechanics itself. Examples of such models in two dimensions are the hierarchical models by Lee (1990) and Yu (1992). These models have been successfully applied to represent fracture trace outcrops, but they do not represent the geometry of the actual 3D fracture systems. An early three-dimensional geometric-mechanical model [Ivanova 1995] used Poisson plane and line stochastic processes (first suggested for fracture system modeling by Veneziano 1978) to incorporate the concept of hierarchical fracture sets into three dimensions. However, this early 3D geometric-mechanical hierarchical model was only conceptual and did not explore in depth the relationships between rock fracture system geometry, and the geologic mechanisms that create it, on the one hand, or the stochastic methods that represent it, on the other hand.

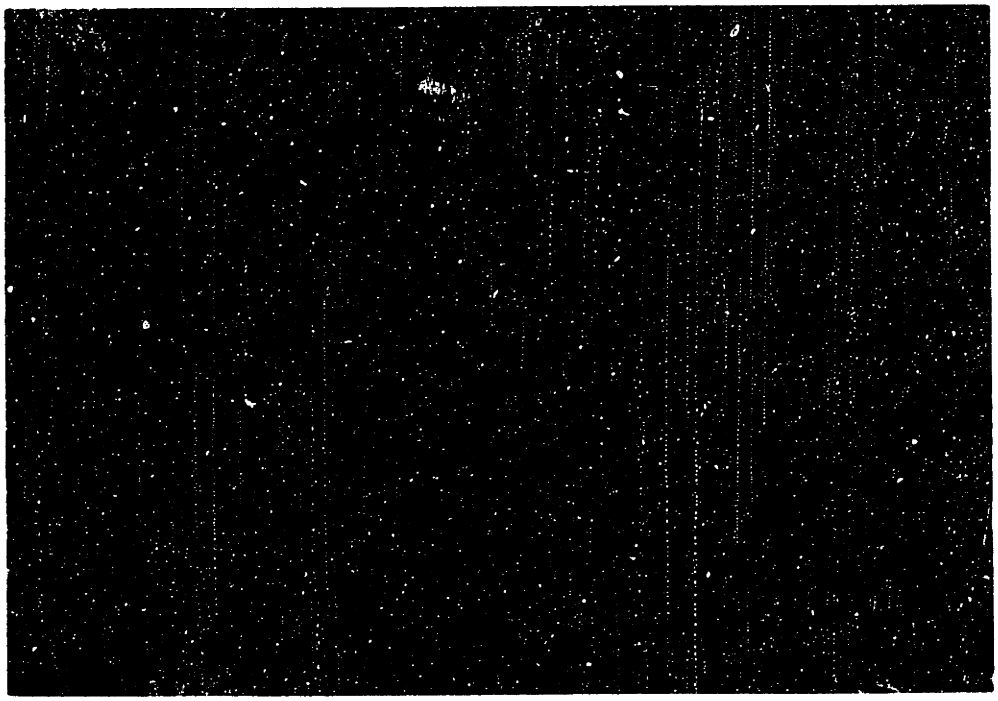


FIGURE 1.1 Fractures at Libby Dam, Montana, that formed surface for mechanical failure and sliding of the overlying rock wedge [photo by F. Descoeudres].

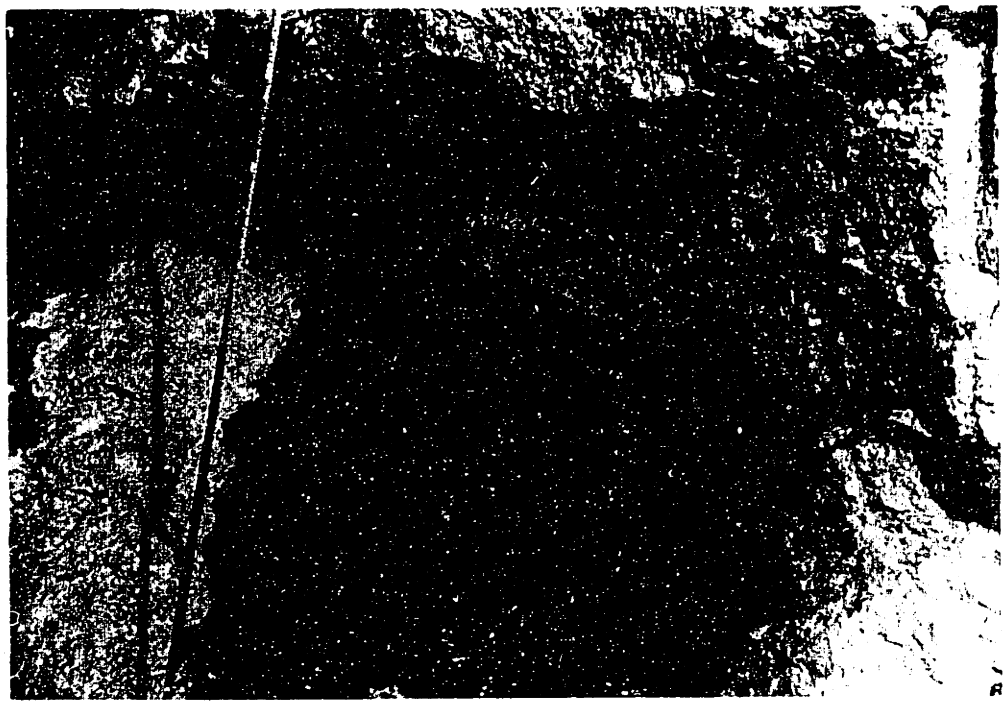


FIGURE 1.2 Flow along fractures in Cambridge Argillite in an excavation in Newton, Massachusetts [photo by H.H. Einstein].

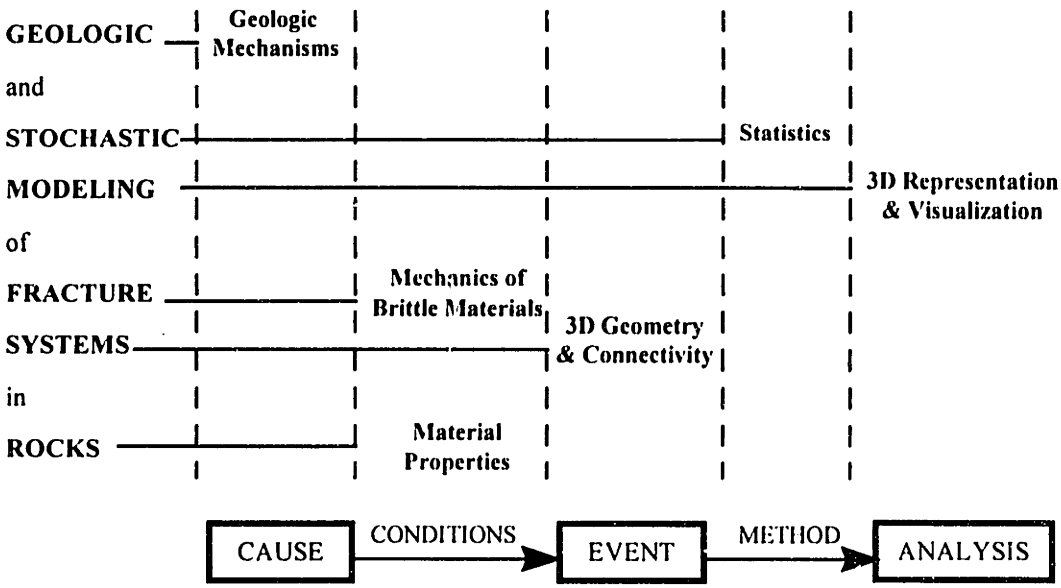


FIGURE 1.3 Research approach, followed in the thesis, for modeling of natural rock fracture systems.

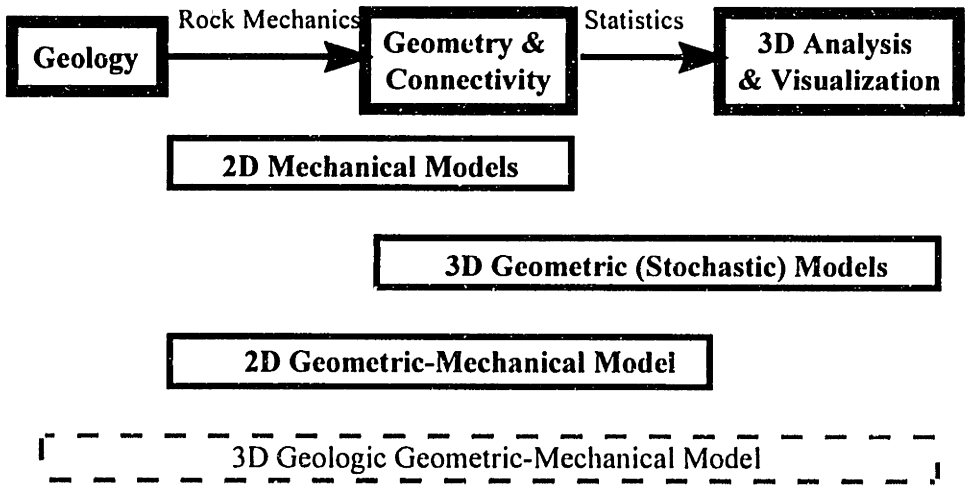


FIGURE 1.4 Prior research on fracture system modeling and objective of the thesis.

In the light of the above discussion, the author believes that a good model of natural rock fracture systems should be geometric-mechanical, hierarchical, and stochastic, and should include algorithms that can represent the geometry of rock fracture systems in different geologic settings. This thesis presents research which attempts to cover the entire path from interpretation of the geologic mechanisms that create natural fracture systems, to a practical tool for 3D analysis and visualization of the fracture system geometry and connectivity.

In order to accomplish the research objective, first, a comprehensive study of natural fracture systems is performed to determine the major geologic settings for rock fracturing. Chapter 2 presents a classification of five major fracture-producing geologic environments (folds, crustal faults, remote tension, thermal contraction, and central structures), and their characteristic fracture system geometries. Next, the 3D geometric-mechanical hierarchical model is developed theoretically. Poisson plane and line processes, as well as other stochastic procedures in 3D space, are incorporated into methods for representation of fracture intensity and fracture orientation related to a general stress field and to local geologic structures. The theory of the 3D model is presented in Chapter 3. Finally, in a case study, the model is applied to represent an actual fracture system. Chapter 4 presents the case study which includes development of a geologic model of fracture system genesis, and numerical simulation of the fracture system in the oil-producing sedimentary rocks of the Yates field in West Texas. Chapter 5 summarizes the conclusions from the geologic and stochastic modeling of rock fracture systems presented in the thesis, and makes suggestions for future research in this area. The Appendix contains a user manual and the source code for the computer program GEOFRAC, developed during the research presented here to implement the 3D fracture system model.

Chapter 2

ROCK FRACTURE SYSTEMS IN NATURE

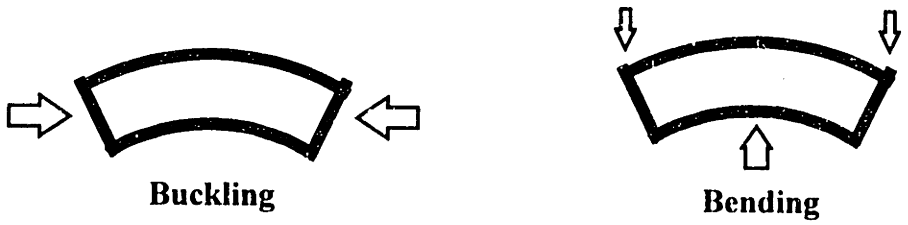
This chapter reviews observations on fracture systems of various geologic history. Five major geologic settings for brittle fracturing have been identified. Every setting is characterized by a specific combination of stresses and strains, leading to a unique geometric configuration of the associated fracture system. Section 2.1 presents fracturing related to folds. Section 2.2 discusses fracturing associated with the three types of crustal faults: normal (Section 2.2.1), strike-slip (Section 2.2.2), and thrust (Section 2.2.3). Section 2.3 describes fracturing in areas of crustal extension. Section 2.4 presents contraction fracturing due to cooling processes. Finally, Section 2.5 describes fracturing associated with central diapiric and collapse structures. Despite the great variety of fracture systems in terms of geologic origin and history, their final geometric configurations have some important common characteristics that are summarized in Section 2.6.

2.1 FOLDS AND RELATED FRACTURES

Folds are common geologic structures created by tectonic or non-tectonic processes. There are two major mechanisms of folding: flexure and shear [De Sitter 1956; Billings 1972; Suppe 1985; Ramsay & Huber 1987]. Flexural folding (**Figure 2.1.1**a) involves bending or buckling of rock strata. Buckling is caused by compressive forces, acting parallel to the bedding, for example, tectonic forces at convergent continental plate boundaries. Bending may result from solution of underlying beds or from differential compaction of sediments. Shear folding (Figure 2.1.1b) results from small displacements along closely spaced planes, perpendicular or oblique to the bedding. Flow is a type of shear folding in which the slip planes are infinitesimally close. The type of folding characteristic for a given rock is strongly dependent on the rheological and mechanical properties of the material, i.e. on the so-called "rock competence" [Billings 1972; Suppe 1985]. Flexural folding is characteristic of competent layers whereas flow and shear folding usually affects incompetent beds.

Rock competence is a relative property. A stiffer, and usually more brittle, rock is more competent than a softer, ductile, rock. A thicker formation is more competent than a thinner formation of the same rock. A previously fractured layer of brittle rock may be significantly less competent that the original intact material. **Table 2.1.1** summarizes a ranking by Suppe (1985) of commonly encountered rocks in terms of their relative competence. According to the same author, the most common combinations of stiff/soft pairs in multilayered rock formations are: sandstone/shale, limestone/shale, chert/shale, sandstone/limestone, gneiss/schist, dolomite/limestone, and calcium silicate/marble.

a) FLEXURAL FOLDING



b) SHEAR FOLDING

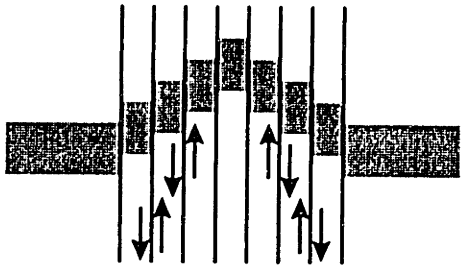


FIGURE 2.1.1 Mechanisms of folding: a) flexure; b) planar shear.

COMPETENCE	NON-CRYSTALLINE ROCKS	CRYSTALLINE ROCKS
Most competent	Dolomite	
↓	Arkose	Metabasaltic rock
	Quartzose sandstone	Coarse-grained granitic rock
	Greywacke	Fine-grained granitic rock
	Coarse-grained limestone	Quartzofeldspatic micaceous
	Fine-grained limestone	gneiss
	Siltstone	Quartzite
	Marl	Marble
	Shale	Mica schist
	Anhydrite	
Least competent	Halite	

TABLE 2.1.1 Ranking of rock competence [after Suppe 1985].

Besides the relative competency of the layers, the following factors determine if a rock formation responds to compressive stresses by flexural or shear folding: rock anisotropy, rate of increase of the deformative stress, confining pressure, thermal action by intrusive rocks at depth, and fluid content [De Sitter 1956].

A widely accepted classification of folds is that according to dip isogons (lines connecting points of equal dip) [Ramsay & Huber 1987; Suppe 1985; Price & Cosgrove 1990]. Three major classes of folds are illustrated in **Figure 2.1.2** and described in **Table 2.1.2**. A simpler classification subdivides cylindrical folds into two main categories: (1) concentric or parallel folds, in which the bed thickness is preserved (Class 1B in Figure 2.1.2); and (2) similar or non-parallel folds (Classes 1C, 2, and 3 in Figure 2.1.2), in which the limbs are thinned and the hinges are thickened. In general, concentric folds develop by flexure of competent strata, whereas similar folds form in incompetent rock due to bedding-oblique shear [De Sitter 1956; Billings 1972; Suppe 1985].

Table 2.1.3 summarizes some characteristics of concentric and shear folds and the fractures associated with them.

2.1.1 Fractures due to flexural folding of a single competent bed

Flexural folding usually involves buckling of the most resistant and rigid (competent) rock layers. It is generally agreed that the most stable fold shape of a stiff elastic layer, buckling within a less stiff elastic medium, is sinusoidal, i.e. a series of anticlines and synclines [Ramsay 1967; Suppe 1985]. The amplitude and the wavelength of the fold train depend mainly on two factors: (1) the competency contrast, i.e. the viscosity ratio μ_1/μ_2, where μ_1 and μ_2 are the shear viscosity coefficients of the competent bed and the incompetent matrix, respectively; and (2) the thickness of the imbedded stiff layer [Ramsay & Huber 1987].

The complex stress field associated with flexural folds varies with time and leads to the propagation of numerous sets of tension and shear fractures. **Figure 2.1.3** illustrates a simplified schematic representation of stress distribution in a cross profile of a folded sheet: tension at the outer arcs, compression in the inner arcs, and shear inside the sheet. Ramsay (1967) studied in detail the folding and fracturing of rocks and defined two possible states of strain within a buckled competent layer (**Figure 2.1.4**): (a) flexural flow, and (b) tangential longitudinal strain. In nature, although each type of deformation is characteristic for different rocks, most folded strata are deformed by both flexural flow and tangential longitudinal strain.

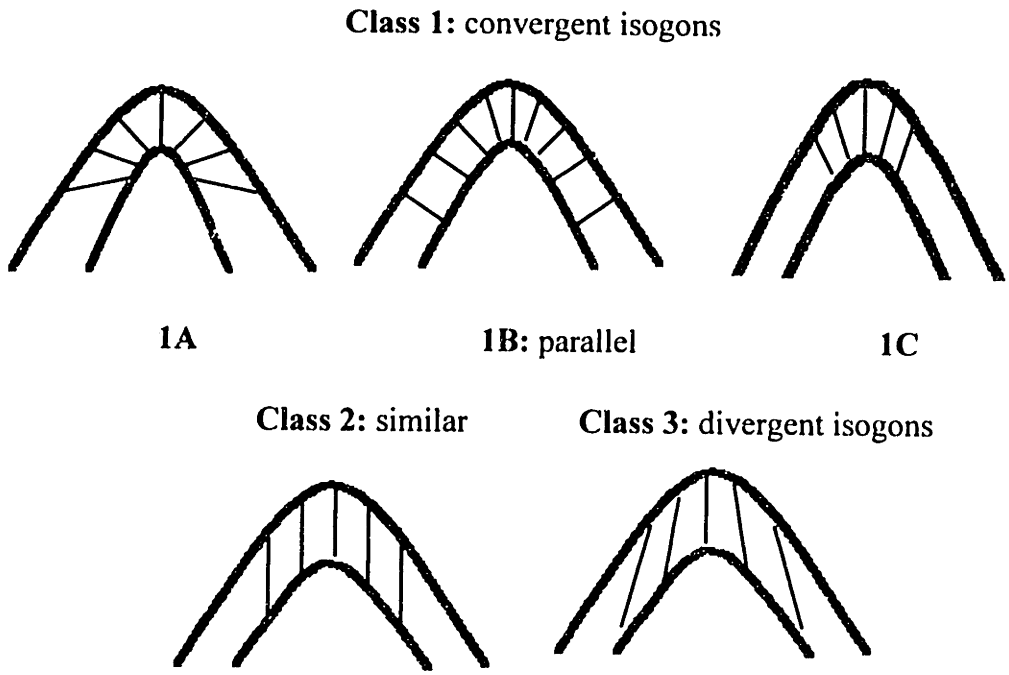


FIGURE 2.1.2 Fold classes according to dip isogons [after Ramsay & Huber 1987].

CLASS 1	Folds with convergent dip isogons; curvature of the inner surface everywhere exceeds that of the outer surface.
• ***Sub-class 1A***	• Folds with greater orthogonal layer thickness on limbs than in the hinge zone; quite rare.
• ***Sub-class 1B***	• Folds with constant orthogonal layer thickness (parallel folds); very widely developed in unmetamorphosed rock.
• ***Sub-class 1C***	• Folds with greater layer thickness in the hinge zone than on the limbs; very widely developed in relatively competent layers in ductile rock.
CLASS 2	Folds with parallel dip isogons; identical shapes of inner and outer surfaces (similar folds); developed in rocks of low ductility or low contrast ductility between adjacent layers.
CLASS 3	Folds with divergent dip isogons; curvature of the inner surface is always less than the curvature of the outer surface; widely developed in incompetent layers interbedded with more competent layers exhibiting folds of Classes 1B or 1C.

TABLE 2.1.2 Classification of folds by dip isogons [Ramsay & Huber 1987].

CHARACTERISTICS	CONCENTRIC (PARALLEL) FOLDS	PLANAR SHEAR (SIMILAR) FOLDS
Position of shear planes	Curved and concentric, parallel to the bedding.	Planar, sub-parallel to the axial plane of the fold; also along bedding-planes.
Position in the Earth's crust	Superficial	Superficial and deeper strata
Competent or incompetent rock	Competent beds	Competent and incompetent beds
State of rock metamorphism	Non-metamorphic rocks	Metamorphic and non-metamorphic rocks
Elastic / plastic	Elasto-viscous	Elasto-viscous
Accompanying fractures	Many, in varying directions, but always in relation to the fold	Fewer, mostly parallel to the fold axis or to oblique shear planes

TABLE 2.1.3. Characteristics of folds [after De Sitter 1956].

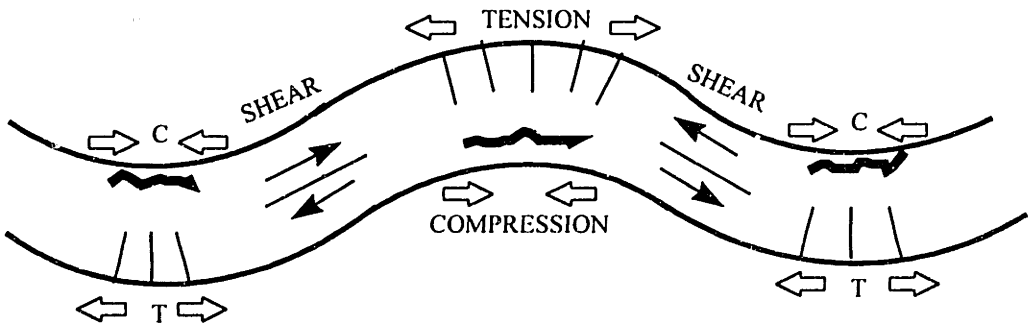
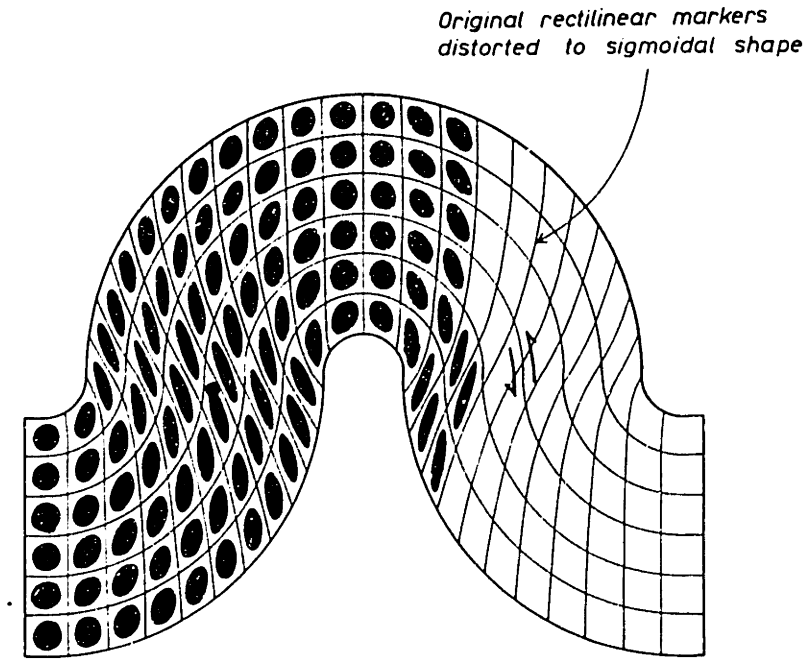


FIGURE 2.1.3 Stresses in a folded rock layer [after De Sitter 1956].

a) FLEXURAL FLOW



b) TANGENTIAL LONGITUDINAL STRAIN

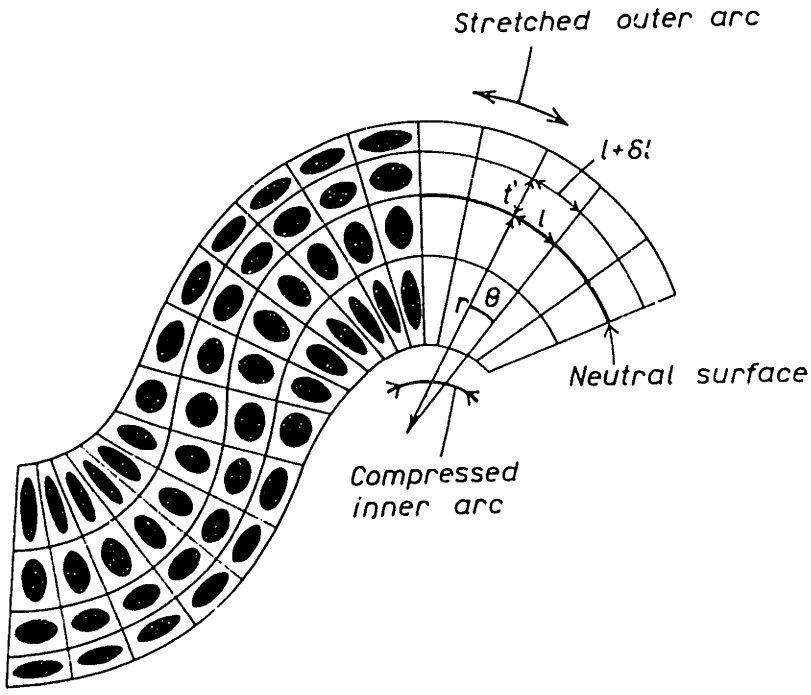


FIGURE 2.1.4 States of strain within a buckled layer: a) internal strain in a layer deformed by flexural flow; b) internal deformation developed by tangential longitudinal strain [from Ramsay 1967].

Fractures induced by bedding-parallel shear in flexural flow folds

Flexural flow (Figure 2.1.4a) is typical for sedimentary rocks with a well-developed plane-parallel stratification and an inherent weakness of the rock along the bedding plane contacts [Ramsay 1967]. True parallel folds (Class 1B in Figure 2.1.2) usually develop in such rocks. When the rock strata are flexed, the outermost layers slip over the inner layers towards the fold hinges. Bedding-parallel shear fractures (thrust faults) commonly develop due to the movement along the contact planes between the sedimentary beds (**Figure 2.1.5**).

According to De Sitter (1956), bedding-parallel shear planes develop first in the flanks of the fold, and, as soon as they are formed, the original bent sheet is subdivided into two elastically folded sheets; this elastic process is repeated continuously during further folding. Also, the slip in opposite directions along the top and bottom of a competent bed induces a shear couple perpendicular to the bedding planes. This may cause the opening of fractures (so-called rotational joints) that are perpendicular to the beds and parallel to the trend of the fold axial plane (Figure 2.1.5). **Figure 2.1.6**a and b show well developed bedding-plane faults in flexural-slip folds. In Figure 2.1.6b, some rotational joints can be seen within individual folded shale beds.

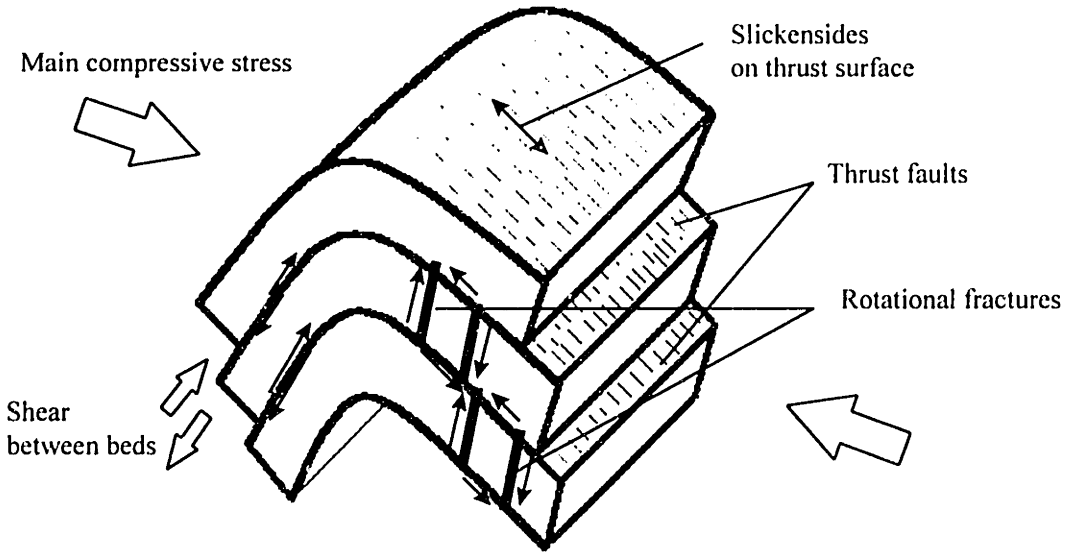


FIGURE 2.1.5 Fracturing induced by bedding-parallel shear in a flexural-slip fold [after Ramsay 1967 and De Sitter 1956].

En echelon tension fissures (often filled with quartz or calcite) are another type of fractures that may be created by flexural flow [Ramsay 1967; Ramsay & Huber 1987]. **Figure 2.1.7** illustrates the development of such tension fissures which always dip at low to moderate angles toward the fold axial surface. They begin to open on the flanks of the fold where the strata dips are the highest, and become progressively deformed during further flexure.

Fractures induced by tangential longitudinal strain in flexural folds

While stratified rocks usually deform by flexural flow and slip along bed-parallel shear planes, buckled non-stratified rocks generally develop tangential longitudinal strain [Ramsay 1967]. In this type of internal deformation the principal strains are oriented tangentially and perpendicularly to the folded layer, and their magnitudes are zero along a neutral surface within the buckled layer (Figure 2.1.4b). The layers are subjected to extension on the outer arc and to compression on the inner arc of the neutral surface. Under tangential longitudinal strain the layers become thinner and the dip isogons converge in the outer arcs of the folded formation (Class 1A in Figure 2.1.2), whereas in the inner arcs the layers become thicker and the dip isogons diverge (Class 1C).

Tangential longitudinal strain in folded strata leads to development of various sets of tensile and shear fractures closely related to the fold geometry. It is important to note that the fracture sets do not form simultaneously. Fold development, including initial fracturing, causes rotation of the principal stress axes and subsequent opening of younger fractures related to the new stress orientations.

Einstein and Dershowitz (1990) discuss the mechanism of formation of tension and shear fractures in a compressive stress field. The general direction of maximum compression (perpendicular to the fold axial plane) parallels the tensile fractures and bisects the acute angle between the shear fractures (**Figure 2.1.8**a). Once folding and fracturing occur, there is a local release of compression perpendicular to the anticlinal axis. This may lead to opening of tension fractures parallel to the fold axis, and to development of a new conjugate pair of shear fractures, the acute angle of which is bisected by the fold axis (Figure 2.1.8b). Once the second system of tensile and shear fractures is created, the local maximum compressive stress rotates to approximately 45° to the fold axis (Figure 2.1.8c). The shaded wedges tend to move out; since they are constrained a new set of tensile fractures may open parallel to the new maximum compressive stress.

If conditions of tensile brittle failure (see Equation 2-1 on p. 112) exist during various stages of fold development, dilational fractures open due to hydraulic fracturing of the rock (i.e. tensile fracturing under negative effective stress due to high fluid pressure). **Figure 2.1.9** illustrates the relationship of the major types of dilational fractures (joints) to the fold geometry and the associated principal stress directions [Price & Cosgrove 1990; Einstein & Dershowitz 1990].

a)

b)

FIGURE 2.1.6 Fractures in flexural-slip folds [photo by H.H. Einstein]: a) close view of non-planar bedding plane faults, Oslo, Norway; b) well developed bedding-plane faults and rotational joints in folded shale, Brienz, Switzerland.

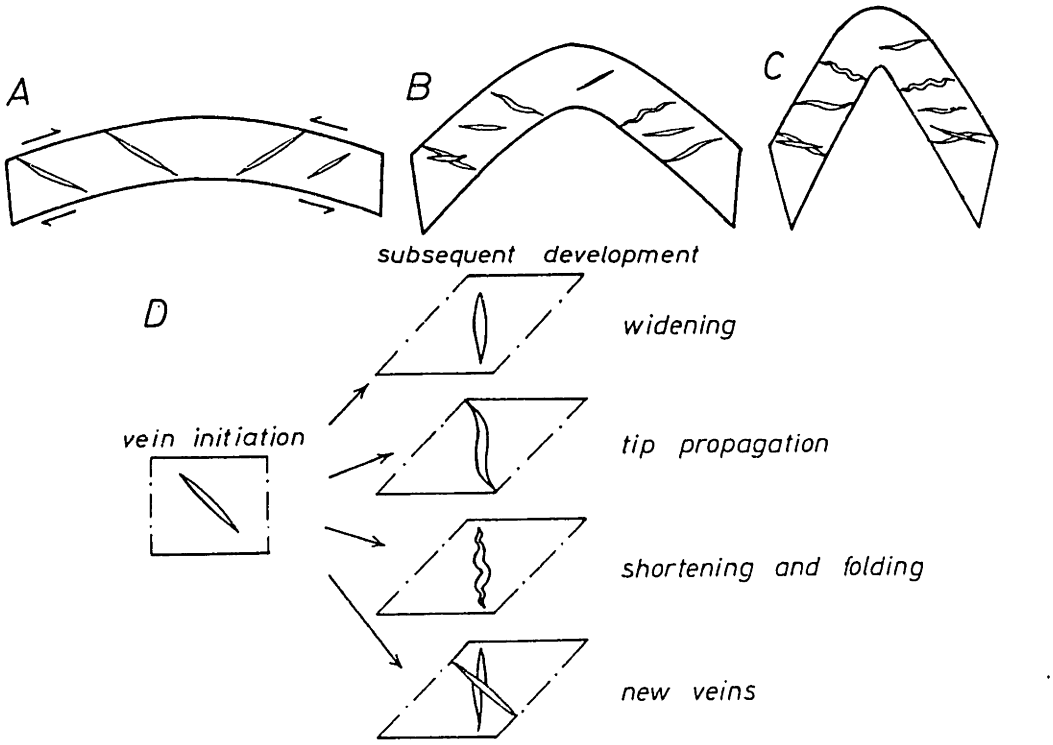


FIGURE 2.1.7 The geometry of tension fissures by flexural flow during subsequent folding stages A, B, and C. Different types of geometric modification of initially planar veins as a result of increased shear strain are illustrated in D [from Ramsay & Huber 1987].

Two major sets of tension joints form during different stages of flexural folding. A primary set (joint set J_1 in Figure 2.1.9) forms parallel to the direction of the general maximum compression even before the actual folding. These joints are usually vertical or very steeply dipping since they are orthogonal to both the fold axis and the bedding. With increased flexure and stretching of the beds, there is a decrease of the compressive stress above the neutral surface of a folded unit, so that locally the minimum principal stress becomes parallel to the general compressive direction. Then a secondary set of tension joints (J_2 in Figure 2.1.9) develops parallel to the anticlinal axes. These joints are perpendicular to the bedding and therefore their dips vary depending on the curvature of the fold crest. When the crest is well rounded, the joints form a fan. In a chevron-type fold with relatively straight limbs the axis-parallel joints form two distinct sets.

The extent of the dilational joints J_2 parallel to the fold trend may be considerable whereas perpendicular to the bedding they are usually restricted to not more than a single bed. However, once they open, the effective surface of the unit is reduced and the neutral surface migrates downward (**Figure 2.1.10**a) which causes the fractures to propagate at depth [Price & Cosgrove 1990]. Extensive lateral

compression may rotate the original layers on the flanks of the fold to an almost vertical position so that the principal compressive stress becomes oblique or even perpendicular to the bedding (Figure 2.1.10b). Due to the different dips of the beds along the fold profile, in the hinge the fractures propagate only in the uppermost layers while on the flanks they cut cross the entire thickness of the competent beds [De Sitter 1956]. Dilational fractures, similar to those illustrated in Figure 2.1.9 and Figure 2.1.10, also develop as release tension joints in the latest stage, when the stress causing folding has already vanished. Release tension joints are either parallel, or orthogonal to the fold axis, depending on whether they release the main or the secondary stress [De Sitter 1956].

When the conditions for hydraulic fracturing are not present, various shear fractures, including strike-slip, normal, and thrust faults may develop in complex relationships to the fold geometry. Shear fractures open in planes parallel to the intermediate principal stress σ_2, often forming conjugate pairs in which the acute angle is bisected by the direction of the maximum principal stress σ_1.

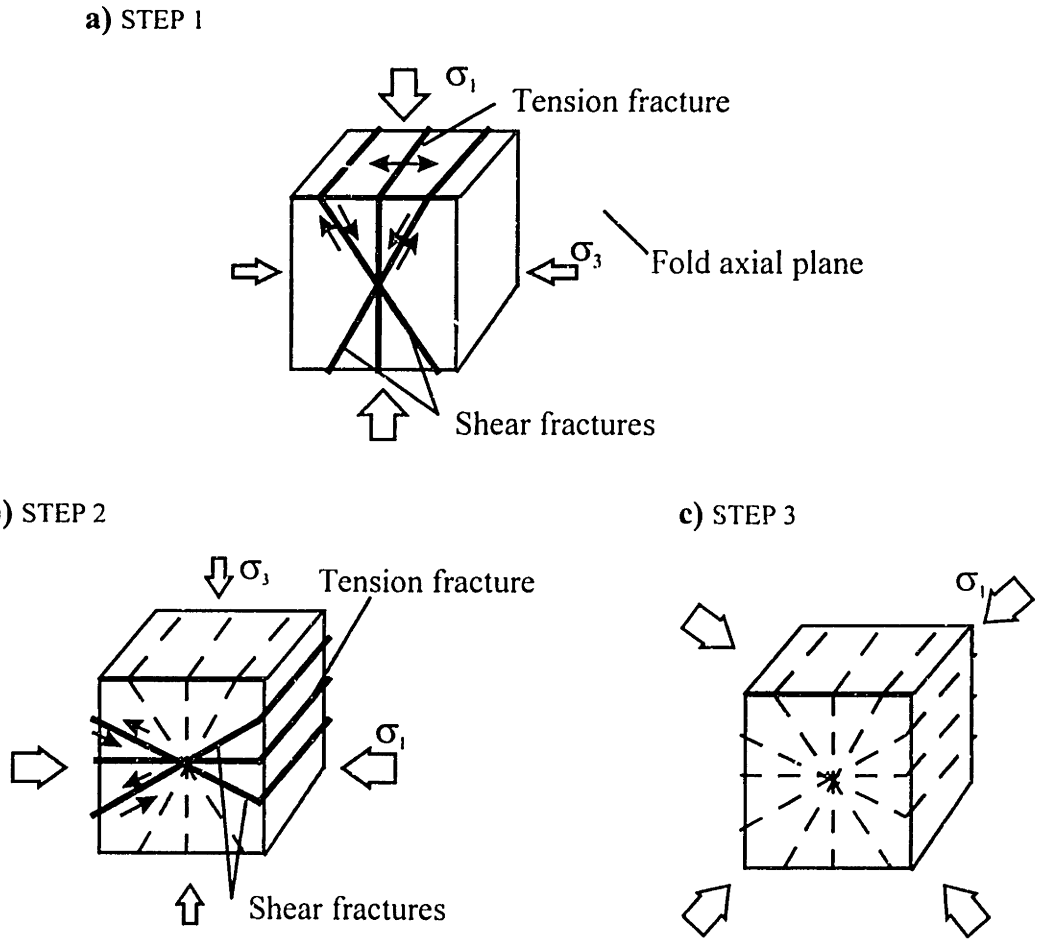


FIGURE 2.1.8 Complex stress and fracture pattern in compressive stress field [after Einstein & Dershowitz 1990].

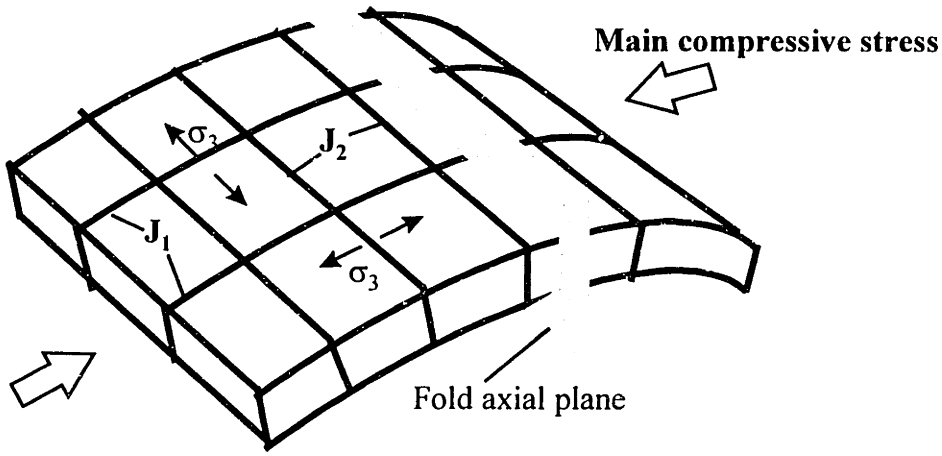
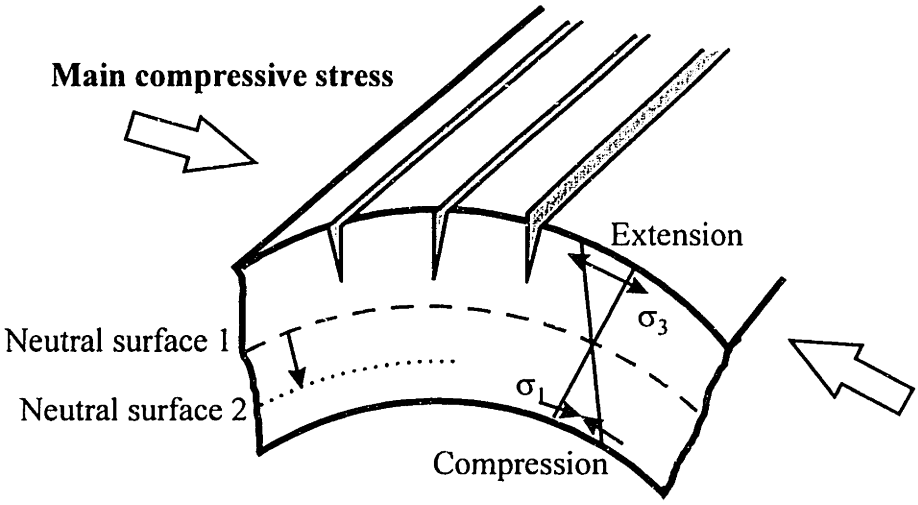


FIGURE 2.1.9 Typical relationship of dilational fractures to a fold [after Price & Cosgrove 1990].

Figure 2.1.11 illustrates typical relationships of shear fractures to the geometry of a flexural fold in a thin, bedded, competent layer [Price & Cosgrove 1990]. The most common set of conjugate strike-slip faults (S_1 in Figure 2.1.11) forms in the initial stage of folding. The deformative stress direction bisects the acute angle between these conjugate faults. In the more advanced, flexural, stage of folding, another set of conjugate strike-slip faults (S_2 in Figure 2.1.11) forms in response to the reorientation of σ_1 and σ_3. Strike-slip faults of types S_1 and S_2 develop when the intermediate principal stress σ_2 is the vertical stress of gravity. If σ_3 is the gravity stress, some minor thrusts (T_1 in Figure 2.1.11) also may develop.

Figure 2.1.12 shows typical orientations of shear fractures that may form in a thick, competent unit if in the advanced stages of flexure the horizontal stress parallel to the fold axis is greater than the vertical stress of gravity [Price & Cosgrove 1990]. Before the beds are bent and during initial folding, conjugate strike-slip faults of type S_1 bisected by the direction of deformative compression, often form (similar to the case in Figure 2.1.11). Due to further flexure, the upper layers are subjected to local extension while in the layers below the neutral surface σ_1 continues to act perpendicularly to the fold axial plane and increases in magnitude relative to the main deformative compressive stress. Strike-slip faults of type S_2, the acute angle between which is bisected by the trend of the fold axial plane, develop in the layers above the neutral surface. Below the neutral surface, low-angle thrust faults (T_1 in Figure 2.1.12) may develop to accommodate the increased local compression.

a)



b)

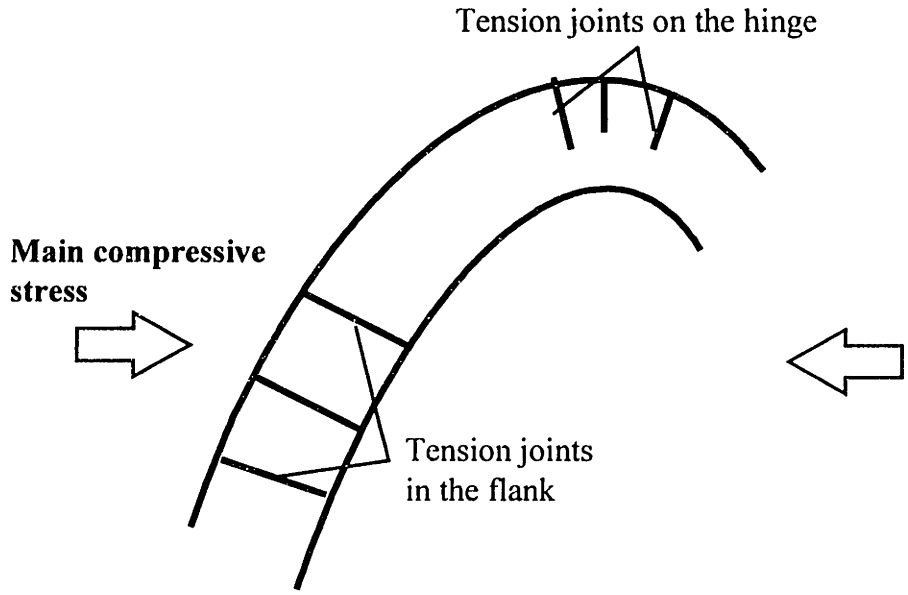


FIGURE 2.1.10 Dilational fractures parallel to the axial trend of a flexural fold: a) migration of the neutral surface downward [after Price & Cosgrove 1990]; b) fractures cutting across the flanks of folded beds [after De Sitter 1956].

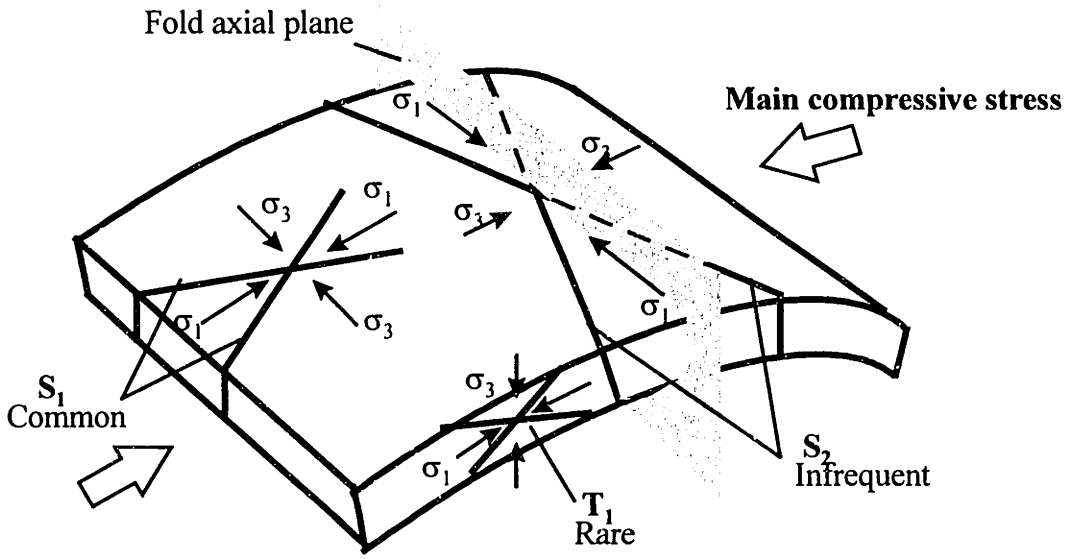


FIGURE 2.1.11 Typical orientations of shear fractures in a thin, bedded, competent layer. S_1 and S_2: strike-slip faults, T_1: thrust faults [after Price & Cosgrove 1990].

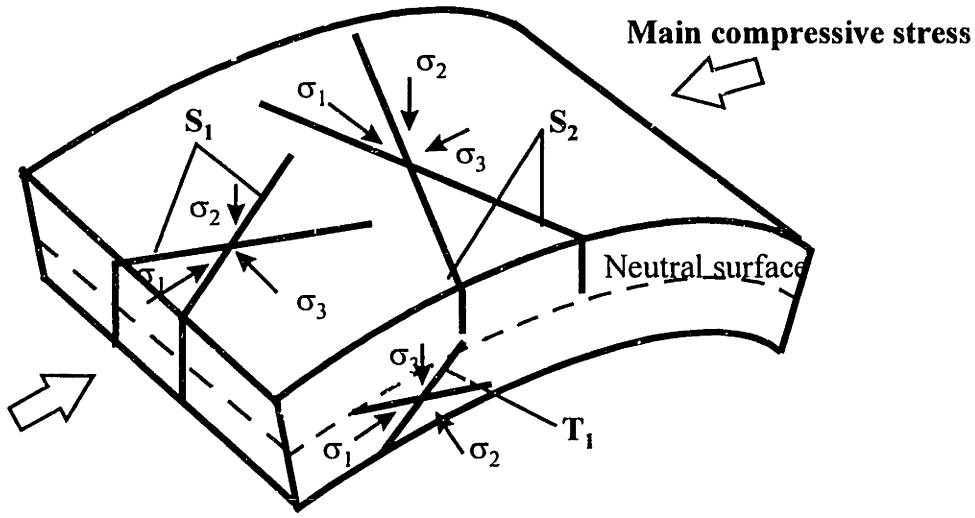


FIGURE 2.1.12 Faults above and below the neutral surface of a thick unit. Case 1: during flexure the horizontal stress parallel to the fold axis is larger than the vertical stress of gravity. S_1 and S_2: strike-slip faults (S_1 develops under the main compression before flexure); T_1: thrust faults [after Price & Cosgrove 1990].

Figure 2.1.13 depicts typical orientations of shear fractures that develop in a thick competent unit in the case when during flexure the horizontal stress parallel to the fold axis is smaller than the vertical stress of gravity [Price & Cosgrove 1990]. Normal faults, parallel to the trend of the axial plane (N_1 in Figure 2.1.13) develop in the stretched layers above the neutral surface. Once these normal faults form, the value of σ_3 increases in magnitude and may exceed the value of σ_2. This new stress configuration leads to development of another set of normal faults (N_2 in Figure 2.1.13) which trend orthogonal to the fold axial plane. In the layers below the neutral surface, strike-slip faults (type S_3 in Figure 2.1.13) may form to accommodate the increased compression perpendicular to the axial plane.

Fracturing of non-cylindrical flexural folds

The fracture patterns illustrated in Figures 2.1.9-2.1.13 are related to the geometry of an ideally-shaped cylindrical fold: i.e. symmetric, concentric anticline. Natural folds, however, seldom have such an ideal shape. For example, an anticline usually plunges at its end where the crest becomes convex upward parallel to the fold axial plane. The fold geometry affects the stress field so that the anticlinal crest is stretched in a plane perpendicular to the fold axis in the middle part of the fold, and in a plane parallel to the fold axis in the peri-anticline. If σ_1 is vertical, longitudinal normal faults may develop in the middle portion of the fold whereas normal cross faults open in the peri-anticline (**Figure 2.1.14**). As the fold grows, it extends along its axis so that the cross-curvature, now reduced in the central part of the fold, reaches new areas and more normal faults, both perpendicular and parallel to the axial trend, may develop [Ramsay & Huber 1987]. As faulting occurs, the neutral surface migrates downward which often causes the faults from the upper layers to propagate and intersect with faults in the lower layers. **Figure 2.1.15** shows an example of a well-developed set of longitudinal normal faults converging and cutting one another at depth.

Figure 2.1.16a schematically illustrates three types of faults typical for plunging peri-anticlines: (1) normal cross faults; (2) oblique shear faults (i.e. faults with both dip-slip and strike-slip) which allow for even further stretching of the peri-anticline; and (3) oblique shear faults that accommodate compressional stress in the inner flank and tensional stress in the outer flank of a curving in plan view peri-anticline [De Sitter 1956]. Figure 2.1.16b shows an example of a mapped fault system of cross faults and diagonal faults in a peri-anticline.

In nature, many flexural folds are markedly asymmetric with an inclined axial plane and one flank steeper than the other. At a certain depth, it is not possible to maintain concentricity of the fold which results in fracturing of the compressed bottom layers along a thrust fault that is usually subparallel to the inclined axial plane. Further compression may result in propagation of the thrust fault upward and even beyond the competent layer, and opening of a set of additional smaller thrust faults [**Figure 2.1.17**; De Sitter 1956].

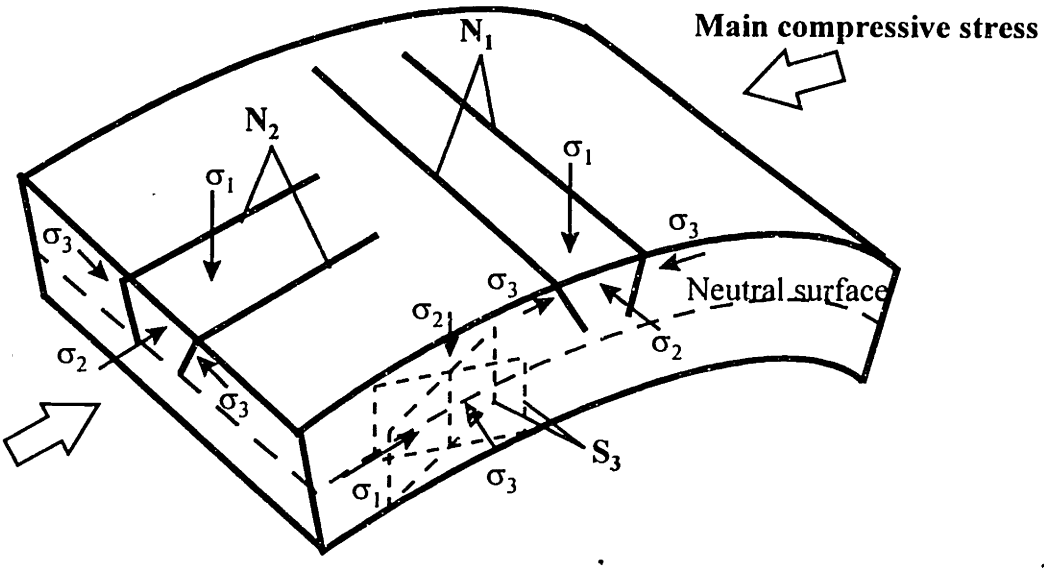


FIGURE 2.1.13 Faults above and below the neutral surface of a thick unit. Case 2: during flexure the horizontal stress parallel to the fold axis is smaller than the vertical stress of gravity. N_1 and N_2: normal faults; S_3: strike-slip faults [after Price & Cosgrove 1990].

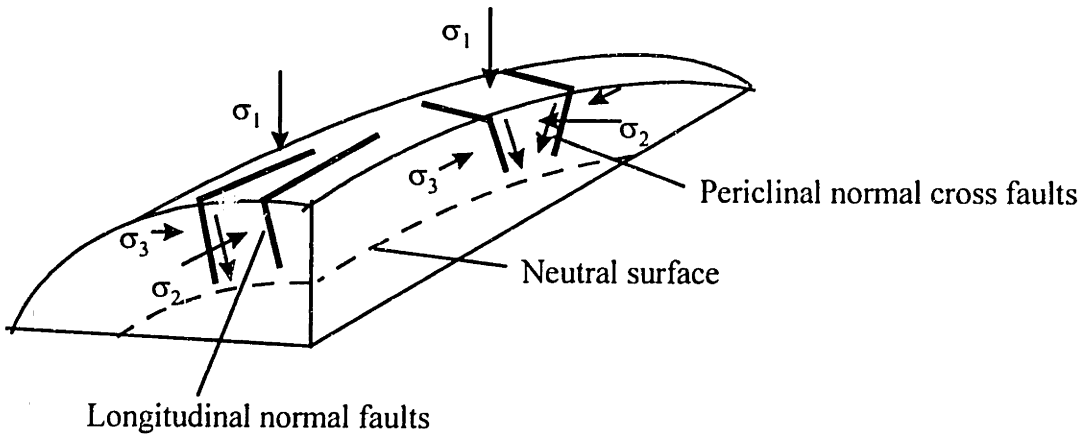


FIGURE 2.1.14 Change of stress magnitude related to fold curvature and development of normal faults parallel and perpendicular to the fold axis [after Ramsay & Huber 1987].

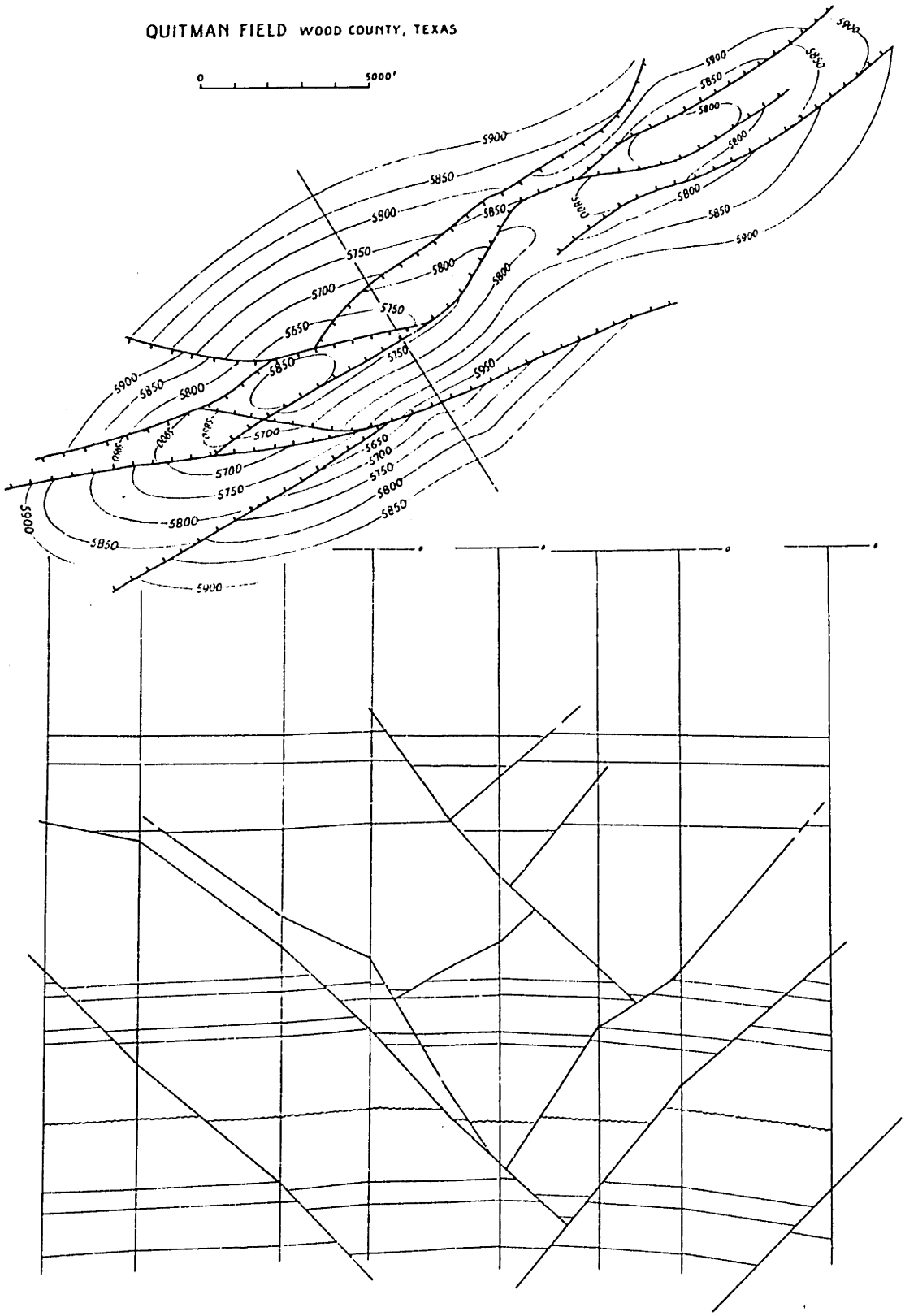
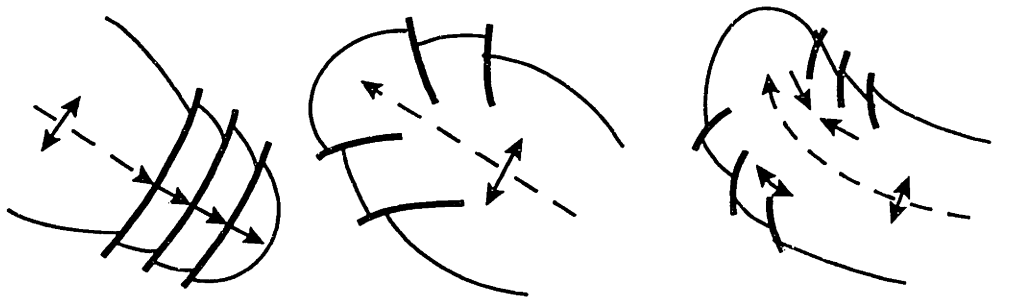


FIGURE 2.1.15 Crestal longitudinal normal faults converging at depth: Quitman Oilfield, Texas [De Sitter 1956].

a)



(1) (2) (3)

b)

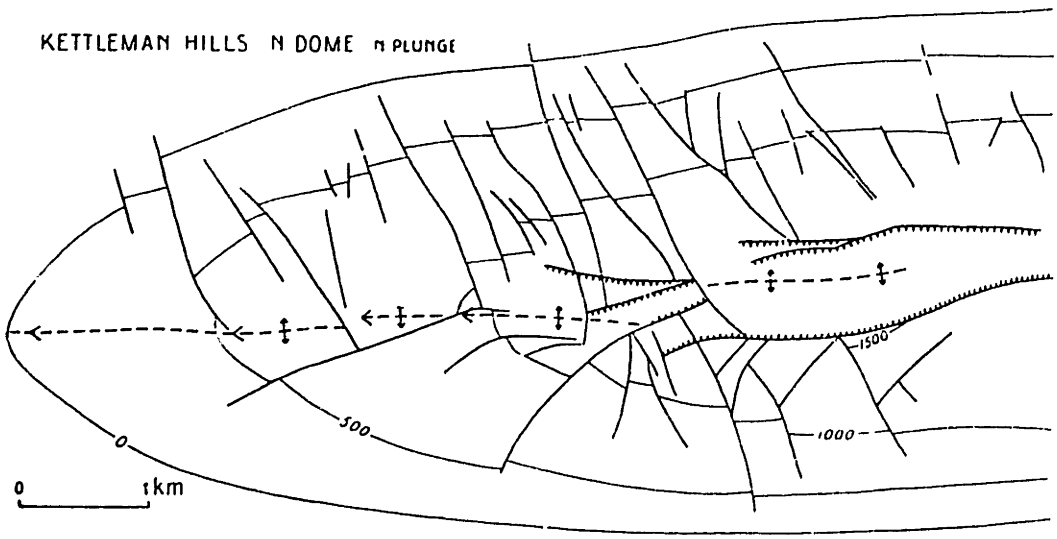


FIGURE 2.1.16 Peri-anticlinal faults: a) schematic representation of (1) normal cross faults, (2) oblique shear faults, and (3) oblique shear faults in a curving peri-anticline; b) peri-anticline of Kettleman Hills, California [after De Sitter 1956].

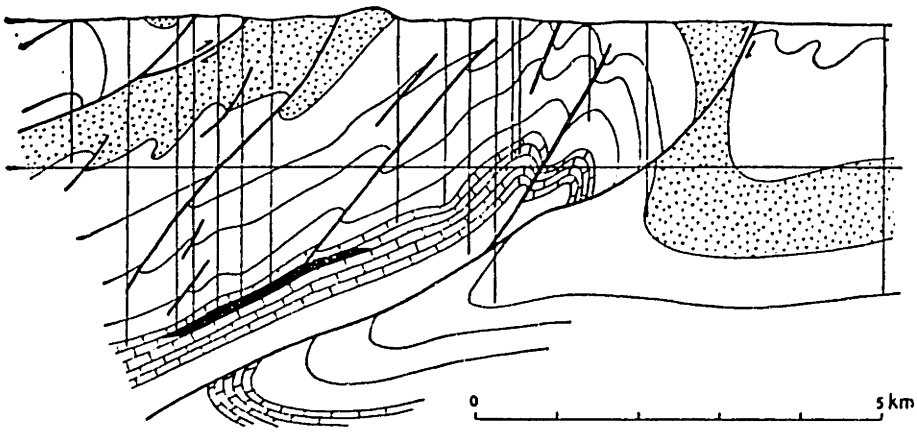


FIGURE 2.1.17 Low-angle thrusts in an asymmetric anticline: Turner Valley [from De Sitter 1956].

The fracture patterns that develop in a syncline are different from those of similar age in an adjacent anticline. **Figure 2.1.18** illustrates fracture outcrops in adjacent syncline and anticline in limestone described by De Sitter (1956). The fracture orientations are consistent with a stress field in the top folded layers such that σ_2 is the vertical stress of gravity whereas σ_3 in the anticline and σ_1 in the syncline are the stresses parallel to the general compressive stress (and orthogonal to the fold axis). In the anticline and the syncline alike, the fracture pattern consists of tensile joints orthogonal to σ_3 and conjugate strike-slip faults the acute angle between which is bisected by σ_1.

Unlike buckle folds discussed so far here, flexural folds formed by bending of sedimentary strata (drape folds) often do not have cylindrical shape. Drape folds develop in response to vertical movement due to either faulting or differential compaction of the underlying basement rocks. In general, the folding strata are stretched which leads to reduction of the confining horizontal stress and possible tensile fracturing. This is illustrated in **Figure 2.1.19** which shows stretching of strata a) over differentially compacted rocks, and b) over a normal fault in the basement. In the flanks, where the extension is the greatest, the bent layers usually become thinner and are often cut by tensile fractures (joints) parallel to the

fold hinge (J_1 in Figure 2.1.19). Figure 2.1.19c illustrates a different case where the folding strata are locally subjected to increased compression over a reverse fault in the basement. In this case σ_1 which acts parallel to the topographic surface and perpendicular to the fold axis may cause opening of both tensile and shear fractures (J_2 and S_1 in Figure 2.1.19). The folded strata are thinner over the sharp edge of the hanging wall of the underlying reverse fault. At this location, the compressive stress acting perpendicular to the bedding may become the maximum principal stress and cause tensile fracturing (J_1) parallel to the fold axial trend and orthogonal to the bent strata. Fracturing associated with drape folds is discussed in more detail in Chapter 4 in the case study of the Yates oil field.

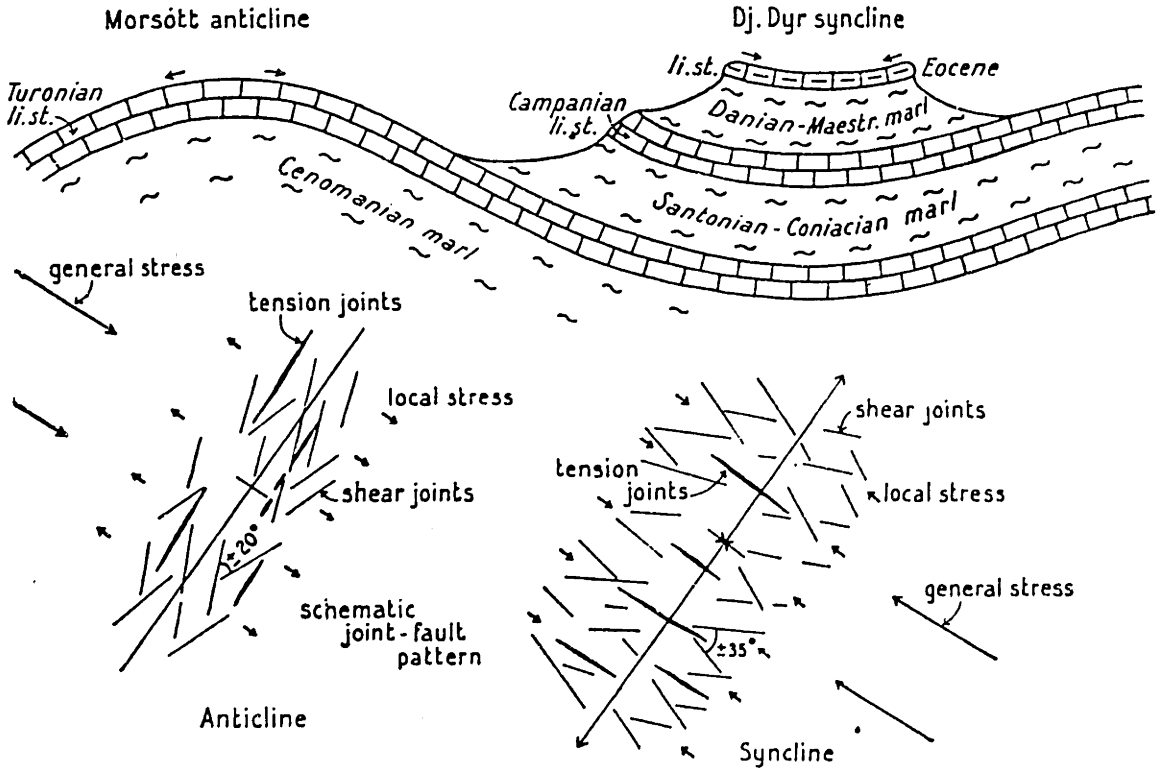
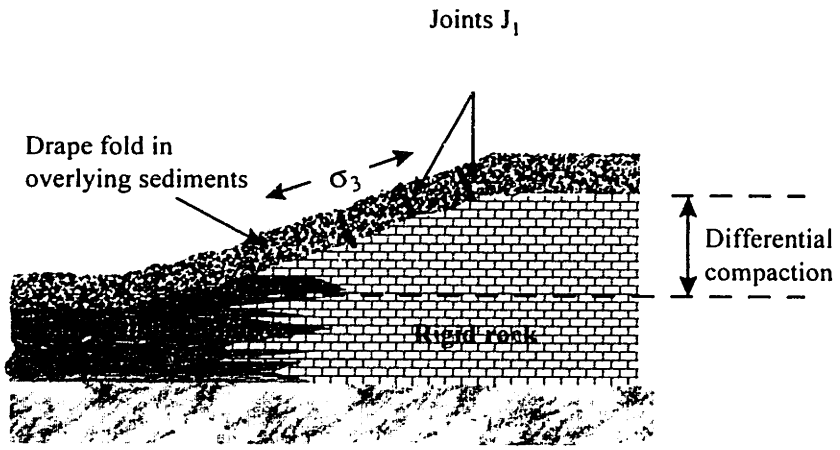
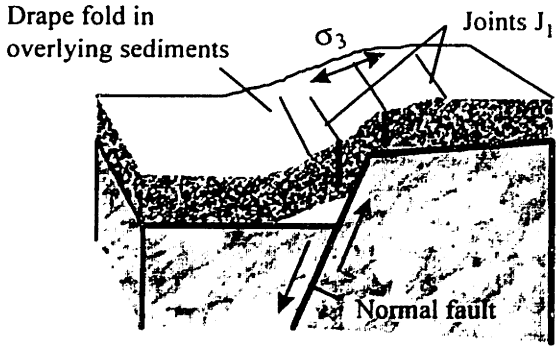


FIGURE 2.1.18 Fracture patterns in adjoining syncline and anticline in limestone in south-east Algeria [De Sitter 1956].

a)



b)



c)

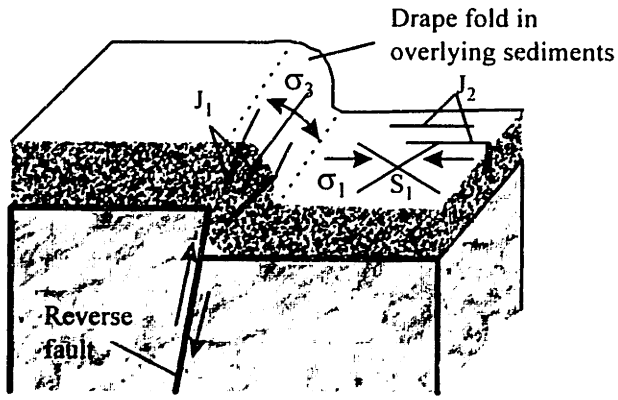


FIGURE 2.1.19 Stress orientation and possible fracturing of strata deformed by bending (drape folding) due to: a) differential compaction, b) normal faulting, or c) reverse faulting of the basement rocks [after Price & Cosgrove 1990].

2.1.2 Fractures related to cleavage and oblique shear in folds

Rock cleavage is defined as a planar structure, distinct from stratification, which permits rocks to be fractured or cleaved into thin slices [Wilson 1946]. Cleavage is often encountered in unfolded and folded rocks alike. It is generally accepted that, if a folded formation exhibits axial plane cleavage (i.e. cleavage subparallel to the fold axial plane), folding and cleavage have been caused by the same tectonic event [Price & Cosgrove 1990].

Cleavage folding is characteristic of incompetent beds which are more ductile than the rigid competent beds. According to De Sitter (1956), cleavage planes develop in ductile rocks since they respond to lateral compression not by flexure but by dilation and thickening of the strata perpendicular to the deformative stress (**Figure 2.1.20**a). In profile, the characteristic shape of cleavage folds is similar, i.e. they are nonparallel, cylindrical or angular, with thinned limbs and thickened hinges (Classes 1C, 2, and 3 in Figure 2.1.2). An ideal similar shape is that of a Class 2 fold with constant layer thickness measured along the dip isogons parallel to the axial plane. The cleavage planes always strike parallel to the trend of the fold axial plane (Figure 2.1.20b). Usually the cleavage is strictly parallel to the axial plane and perpendicular to the bedding only in the hinges. The cleavage planes may fan (Figure 2.1.20c) or refract as they pass from one bed to another (Figure 2.1.20d), thus being oblique to the bedding in the fold flanks.

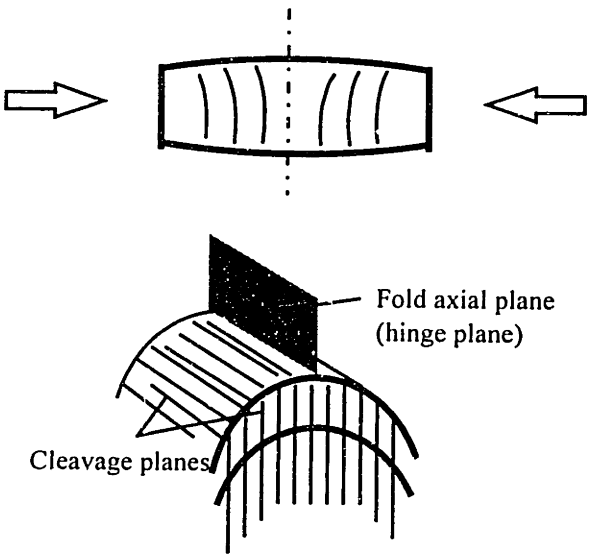
There are two types of folds that are created by shear along planes that are not parallel to the bedding and thus are closely related to the true cleavage folds [De Sitter 1956]. In accordion (also called regular chevron) folds (**Figure 2.1.21**a), cleavage is much more pronounced in the hinges than in the flanks which remain straight. In chevron folds, shear (and cleavage) develop along planes that are oblique to the bedding in one flank whereas the other flank is simply tilted (Figure 2.1.21b). These two types of shear folds are similar and exhibit the characteristic thickening of the hinges and shear (cleavage) parallel to the fold axial plane.

Compared to flexural folds, cleavage and oblique shear folds are accompanied by a much smaller number and variety of joints and faults. The most common type of fracturing associated with cleavage folds is the so-called fracture cleavage [Price & Cosgrove 1990; De Sitter 1956]. Unlike true continuous (also called flow or slatey) cleavage, in which the rock can be split into progressively thinner and thinner sheets, the orthogonal spacing between fracture cleavage planes is larger and the rock between the fractures exhibit little or no weakness parallel to the cleavage direction.

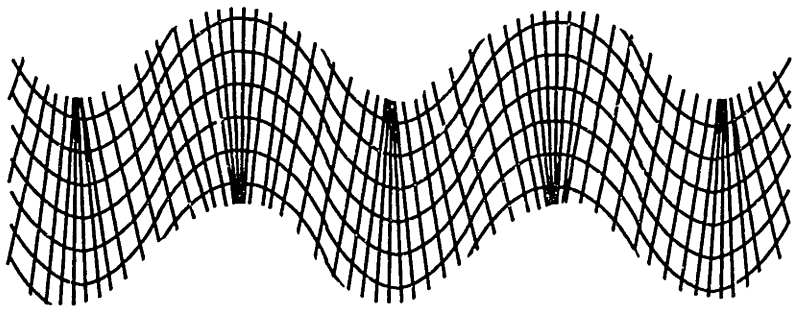
Figure 2.1.22 illustrates the closely spaced fracture cleavage characteristic of folded incompetent beds [Price 1962; Price & Cosgrove 1990]. Fracture cleavage, like slatey cleavage, is always parallel to the trend of the fold hinge.

a)

b)



c)



d)

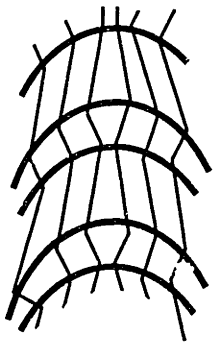


FIGURE 2.1.20 Cleavage folding [after De Sitter 1956 and Price & Cosgrove 1990]: a) origin: thickening of strata and shearing perpendicular to the deformative stress; b) relationship of axial plane cleavage to a fold; c) cleavage fanning in similar cleavage folds; d) cleavage refraction.

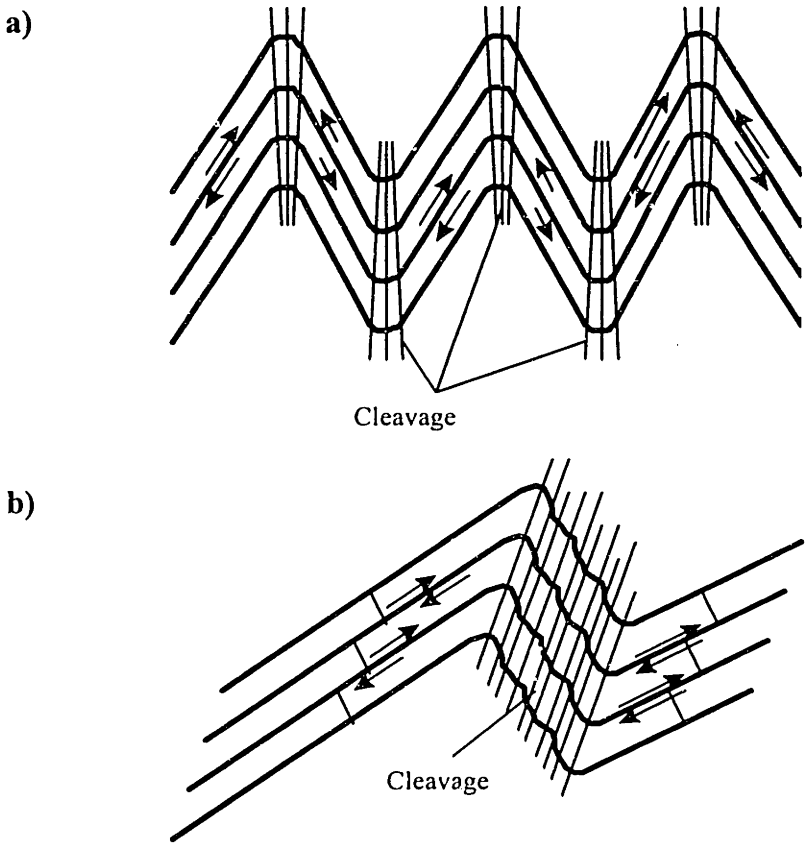


FIGURE 2.1.21 Folds created by bedding-oblique planar shear: a) accordion (regular chevron) fold; b) (irregular) chevron fold [after De Sitter 1956].

In profile, the fracture cleavage planes are usually arranged in an inward diverging fan around the hinge plane (Figure 2.1.22a). The degree of cleavage development depends on the rock lithology and is also related to the geometry of the fold. For example, the cleavage fracturing may be better developed in the flanks than it is near the hinge (Figure 2.1.22b), or it may be better developed in the short limb of an asymmetric fold (Figure 2.1.22c).

Figure 2.1.23 illustrates axial plane fracture cleavage observed in competent rocks by Price & Hancock (1972). These authors attribute spaced cleavage in competent rocks to hydraulic fracturing induced by compressive stress drop, combined with high fluid pressure, that exists at various portions of the fold during its amplification and sequential development. Unlike that in incompetent rocks, the cleavage fracture planes in the folded competent units form a fan that converges towards the inner arcs of the fold. Figure 2.1.23 also depicts two types of fractures that cut across the cleavage: a) pairs of conjugate shear fractures, and b) tensile joints parallel to the direction of the deformative stress. Most of the cross-fractures are restricted to individual beds.

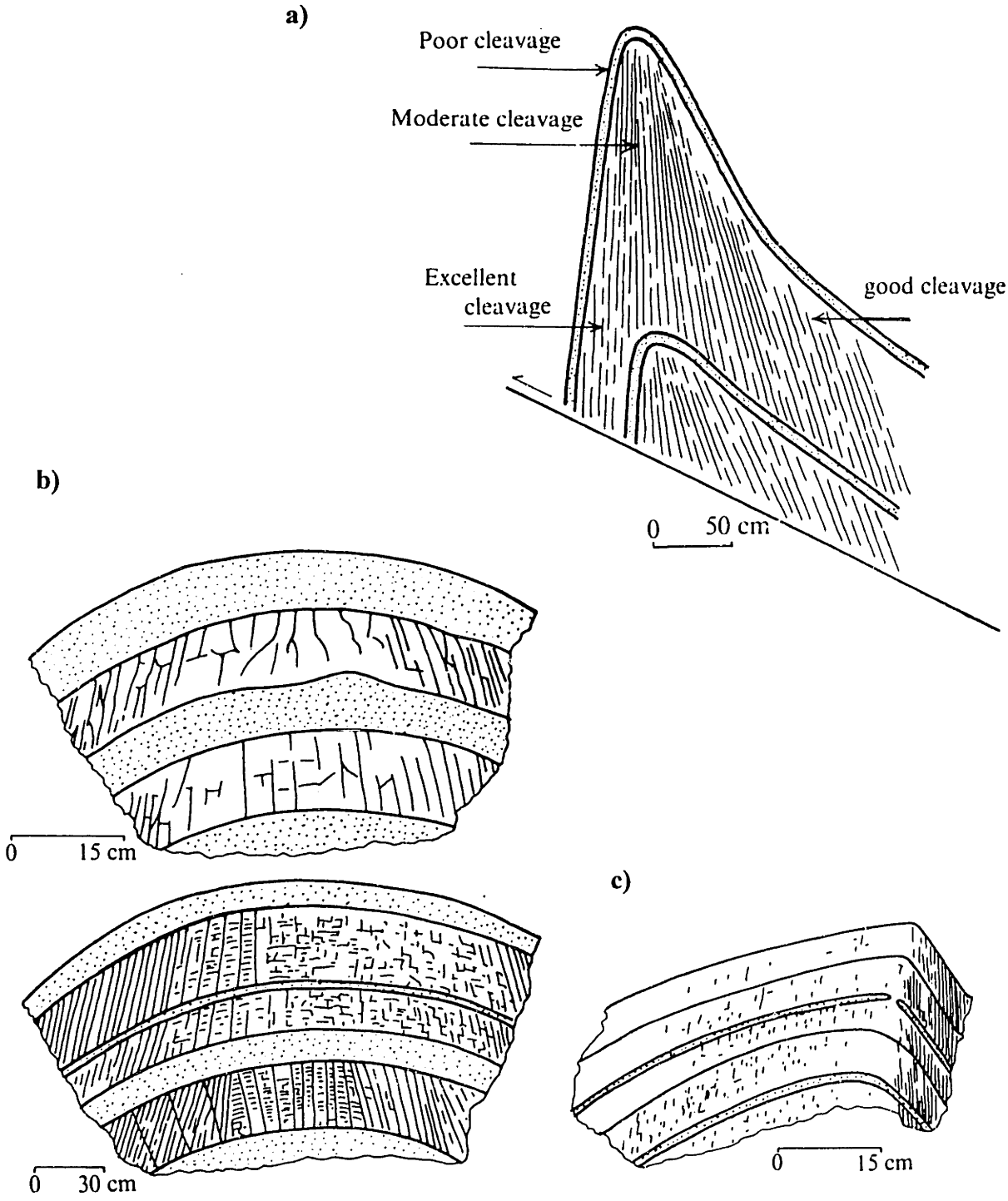


FIGURE 2.1.22 Field sketches of fracture cleavage in incompetent units in the Aberystwith Grits, Wales: a) divergent fanning around the hinge plane; b) well developed fracture cleavage in the flanks and more rudimentary cleavage in the hinge; c) good fracture cleavage in the short limb of an asymmetric fold [from Price 1962 and Price & Cosgrove 1990].

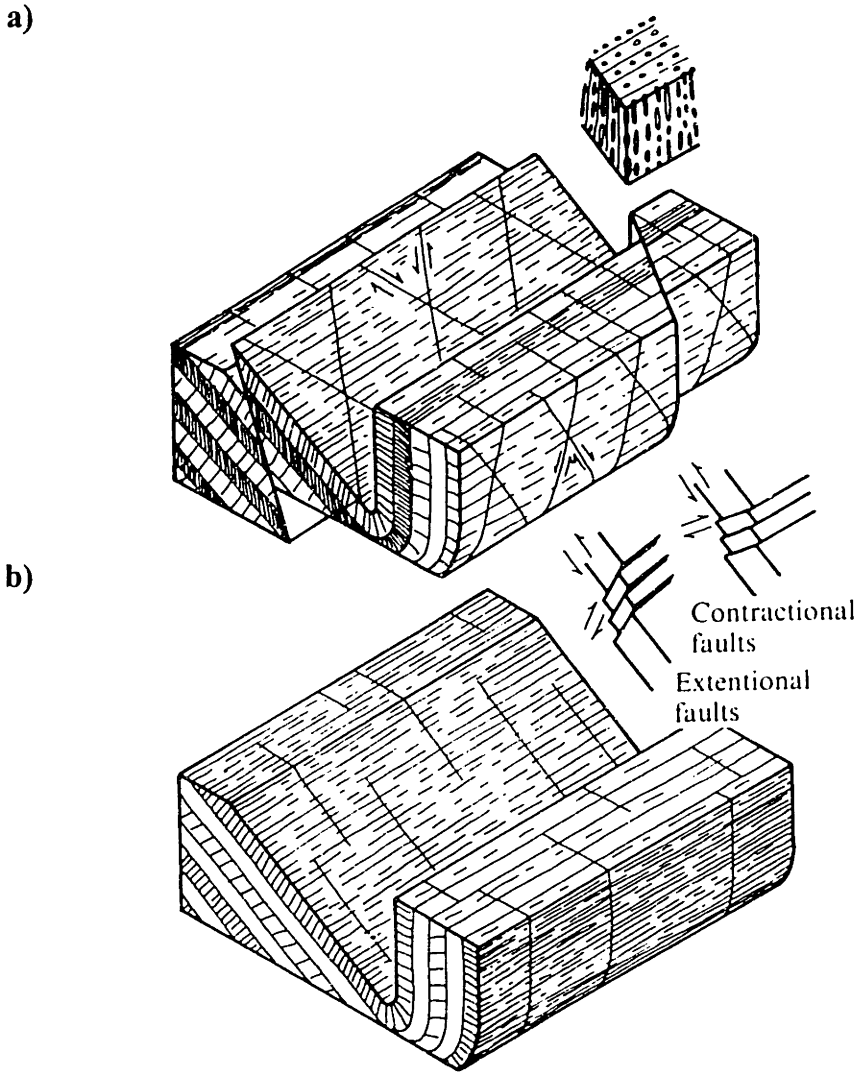
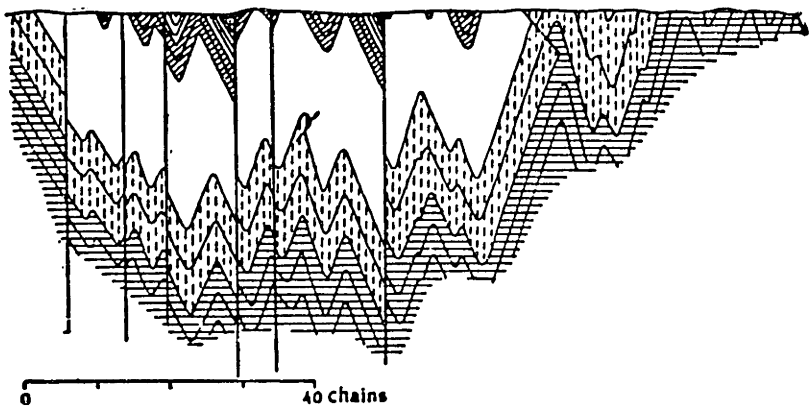


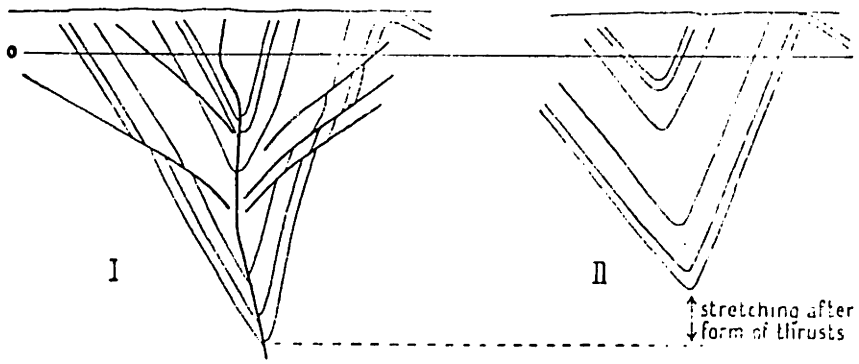
FIGURE 2.1.23 Schematic diagrams showing the relationships between folds and hydraulic fractures (cleavage) and fractures that cut the cleavage: a) Dyfed, Wales; b) Aragon, Spain [from Price & Hancock 1972 and Price & Cosgrove 1990].

Of the multitude of fracture types associated with flexural folds (discussed in Section 2.1.1) only two types are commonly encountered in planar shear folds [De Sitter 1956]: 1) bedding parallel thrusts that may form when the beds in the straight flanks of accordion and chevron folds slip past one another (Figure 2.1.21); and 2) thrust faults sub-parallel to the axial plane. **Figure 2.1.24**a shows an example of vertical faults parallel to the axial plane and cleavage in a true cleavage fold. Figure 2.1.24b illustrates a well-developed axial plane thrust fault in an accordion fold, accompanied with some minor thrusts in the flanks; the faults accommodate vertical dilation. Figure 2.1.24c depicts thrust faulting near the axial plane of a chevron fold.

a)



b)



c)

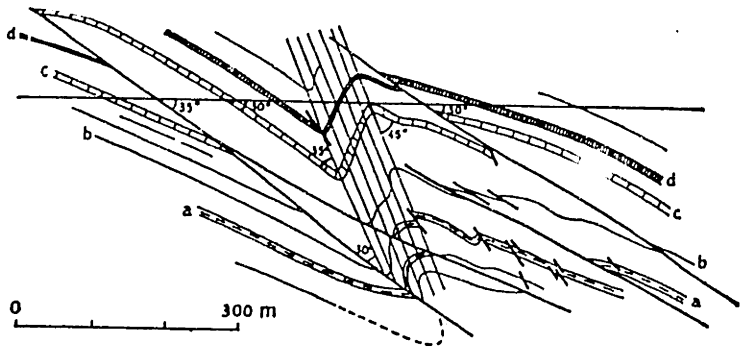


FIGURE 2.1.24 Thrust faults in shear folds [from De Sitter 1956]: a) vertical faults in true cleavage folds: Bendigo Goldfield, Victoria, Australia; b) thrusts in an accordion fold: Ruhr coal district, Westphalia, Germany; c) thrusts in a chevron fold: Domaniale Colliery, Belgium.

2.1.3 Fracturing of a folded sequence of competent and incompetent beds

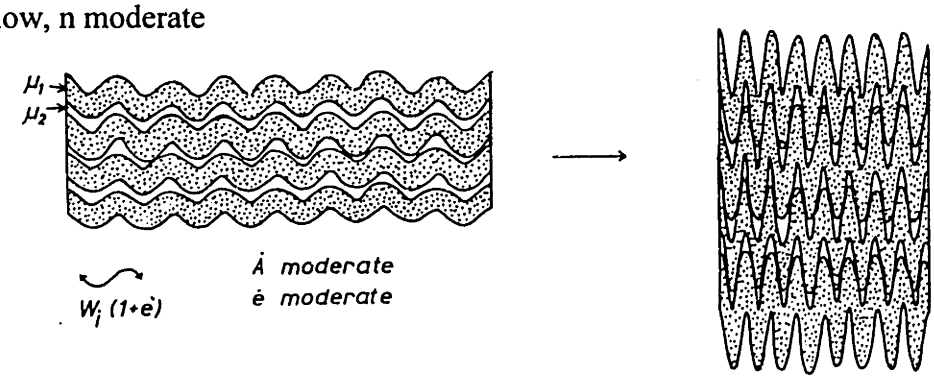
In a multilayered rock formation, relatively stiff, unmetamorphosed rocks of low ductility (competent layers) are usually interbedded with relatively soft, ductile rocks (incompetent layers). The geometry of folds in formations of alternating competent and incompetent rocks depends mainly on two factors: (1) the viscosity contrast μ_1/μ_2, where μ_1 and μ_2 are the viscosity coefficients of the competent and incompetent beds, respectively; and (2) the thickness ratio $n=d_2/d_1$, where d_1 and d_2 are the thicknesses of the competent and incompetent beds, respectively. Ramsay & Huber (1987) discuss in detail the characteristic shapes of folded multilayers as a function of μ_1/μ_2 and n.

Figure 2.1.25 shows three examples from Ramsay & Huber (1987) of different fold shapes that can develop as a function of the viscosity contrast μ_1/μ_2 and the thickness ratio d_2/d_1 of two alternating lithologies. If the competency of the two rocks is not very different, and n is moderate, both competent and incompetent layers develop similar folds: thick in the hinges and thin in the flanks (Figure 2.1.25a). In the case of great difference in competency and similar thickness of the two rocks (Figure 2.1.25b) the competent layers develop parallel folds (Class 1B in Figure 2.1.2) whereas the incompetent layers become thicker in the hinges and thinner in the flanks and form Class 3 folds. If the competency contrast is high, but the incompetent layers are much thinner than the competent ones, the characteristic chevron or accordion folds have narrow hinge zones and straight flanks (Figure 2.1.25c).

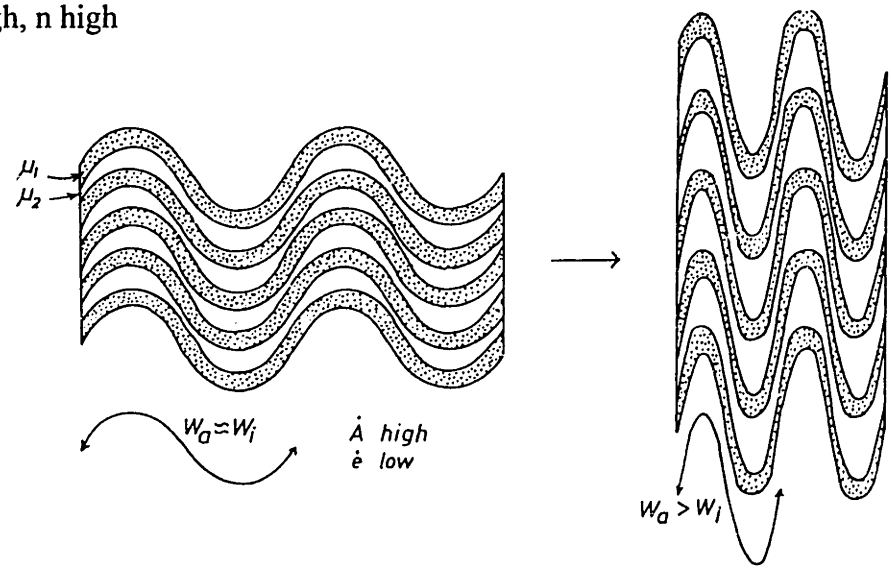
When a multilayer is folded, the incompetent rocks generally develop cleavage whereas the competent layers may fracture in a way typical for flexural folds. Two examples from De Sitter (1956) are shown in **Figure 2.1.26** and **Figure 2.1.27**. Figure 2.1.26 shows a portion of a fold flank, in which fracture cleavage has developed in the shale (light color in the figure), while perpendicular rotational joints can be observed in the competent sandstone layers (dark in the figure). Figure 2.1.27 depicts a hand specimen microfold of incompetent limestone (bands A, B, and C) banded with competent chert (bands a, b, and c). The following structures can be observed in this specimen: (1) in the chert: tension cracks on the anticlinal hinge and in the flanks, rotational joints in the flanks, small normal faults in the flanks and crestal thrusts; (2) in the limestone: flow lines parallel to the bedding, drag folds, and shear fracture traces. The structure is consistent with a general stress field of horizontal compression. There is evidence of concentric folding in both chert and limestone, cleavage folding in limestone, tension parallel to the bedding, and shearing independent of folding.

In some cases, under very high lateral compressive stress, the cleavage shearing can penetrate into competent beds, underlying and overlying soft incompetent layers. The amount of cleavage that develops in a competent bed depends on the magnitude of the lateral compression and the induced layer parallel shortening.

a) μ_1/μ_2 low, n moderate



b) μ_1/μ_2 high, n high



c) μ_1/μ_2 high, n moderate

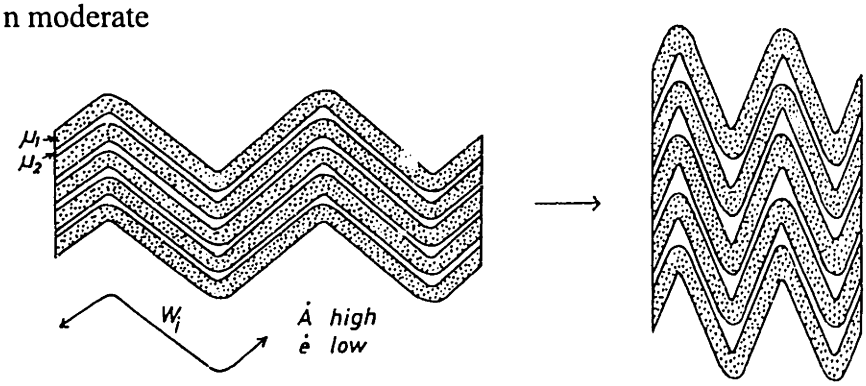


Figure 2.1.25 Models of folds developed in regularly alternating competent layers of thickness d_1 and viscosity μ_1, and incompetent layers of thickness d_2 and viscosity μ_2 [from Ramsay & Huber 1987]: a) similar folds in both competent and incompetent beds; b) parallel folds in competent beds and similar folds in incompetent beds; c) chevron (accordion) folds. $n=d_2/d_1$ is the layer thickness ratio, μ_1/μ_2 is the viscosity contrast, W_i is the initial wavelength of the fold, $\dot{A}$ is the buckle fold amplification rate, $\dot{e}$ is the layer parallel shortening rate, and e is the total layer parallel extension.

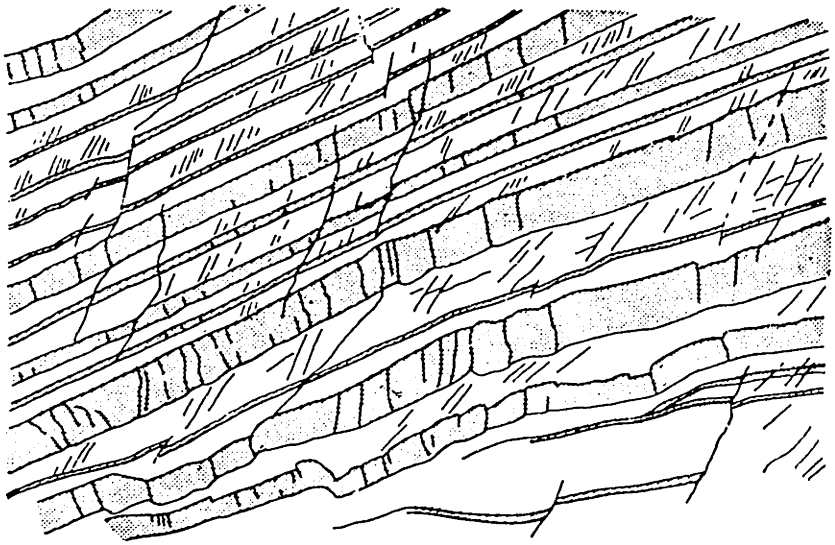


Figure 2.1.26 Fracture cleavage in shale and rotational joints in sandstone, drawn from photograph of Ordovician rocks near Aberystwyth, Wales [from De Sitter 1956].

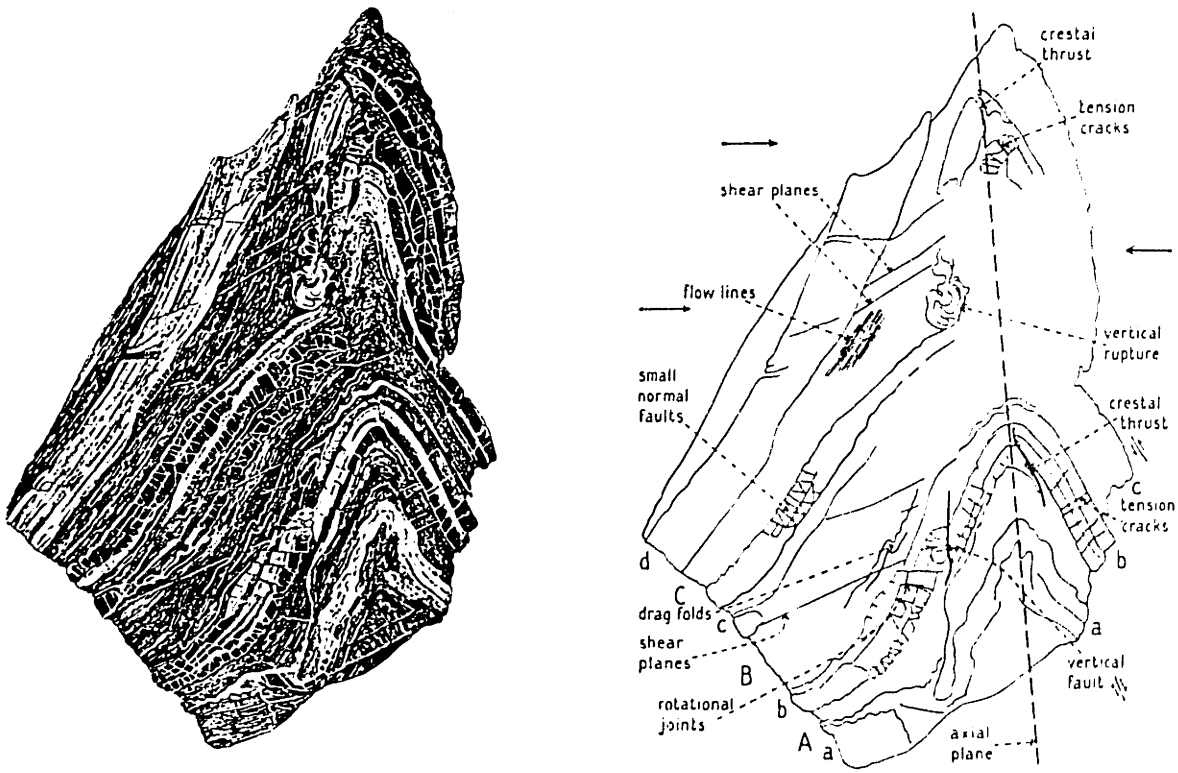


Figure 2.1.27 Fracturing of a microfold in limestone with chert bands: Devonian, Valle de Aran, Spanish Pyrenees [from De Sitter 1956]. a, b, and c are chert bands; A, B, and C are limestone layers.

The cleavage planes usually form convergent fans around the fold axial plane in competent beds and divergent fans in incompetent rocks [**Figure 2.1.28**; Ramsay & Huber 1987, De Sitter 1956, Suppe 1985].

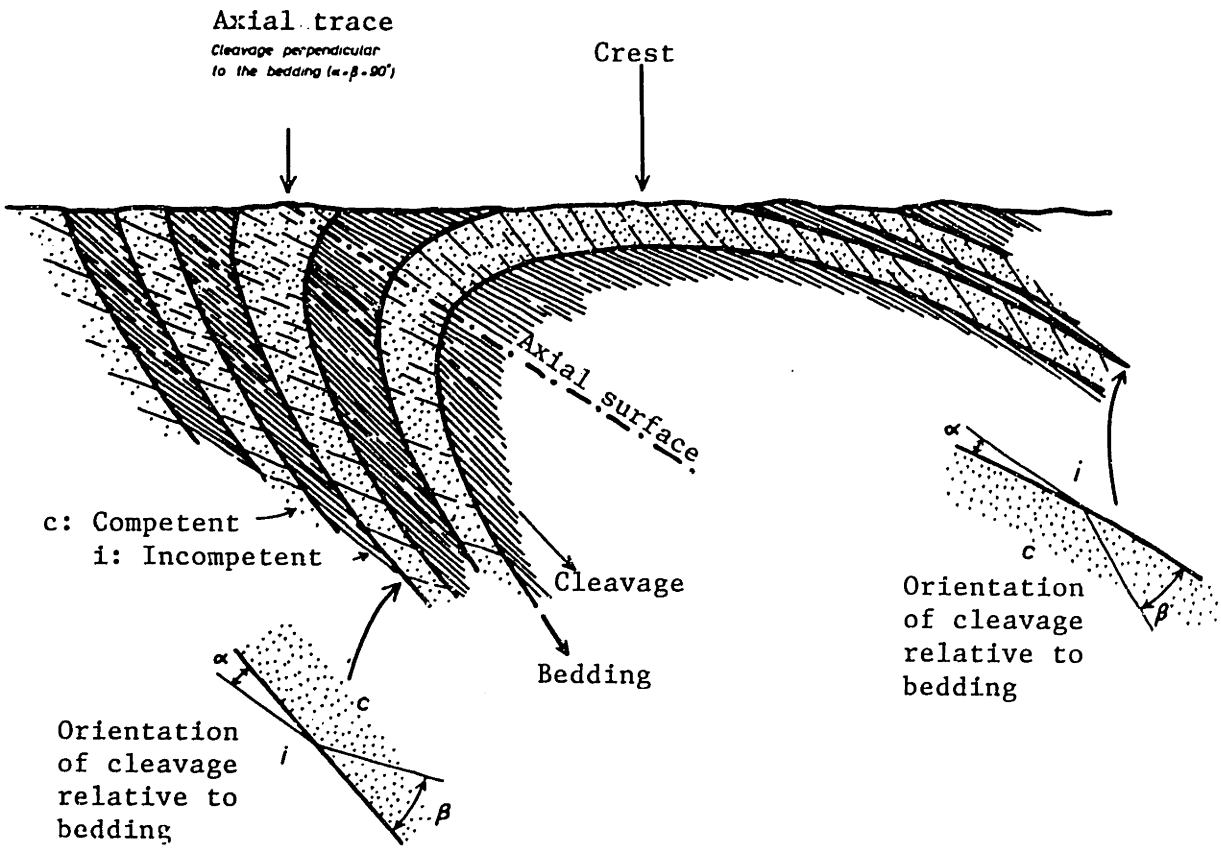


Figure 2.1.28 Cleavage in alternating competent and incompetent beds in an overturned fold [from Ramsay & Huber 1987].

The last figure in Section 2.1 shows fracturing characteristic for the angular folds in Figure 2.1.25c. **Figure 2.1.29**a illustrates the saddle reef structures (empty spaces in the hinges) that may form in a chevron fold if one of the competent layers is significantly thicker than the other layers. Figure 2.1.29b illustrates the possible solution of such a volumetric problem by thrust faulting along bedding planes in the flanks [Ramsay & Huber 1987, Price & Cosgrove 1990].

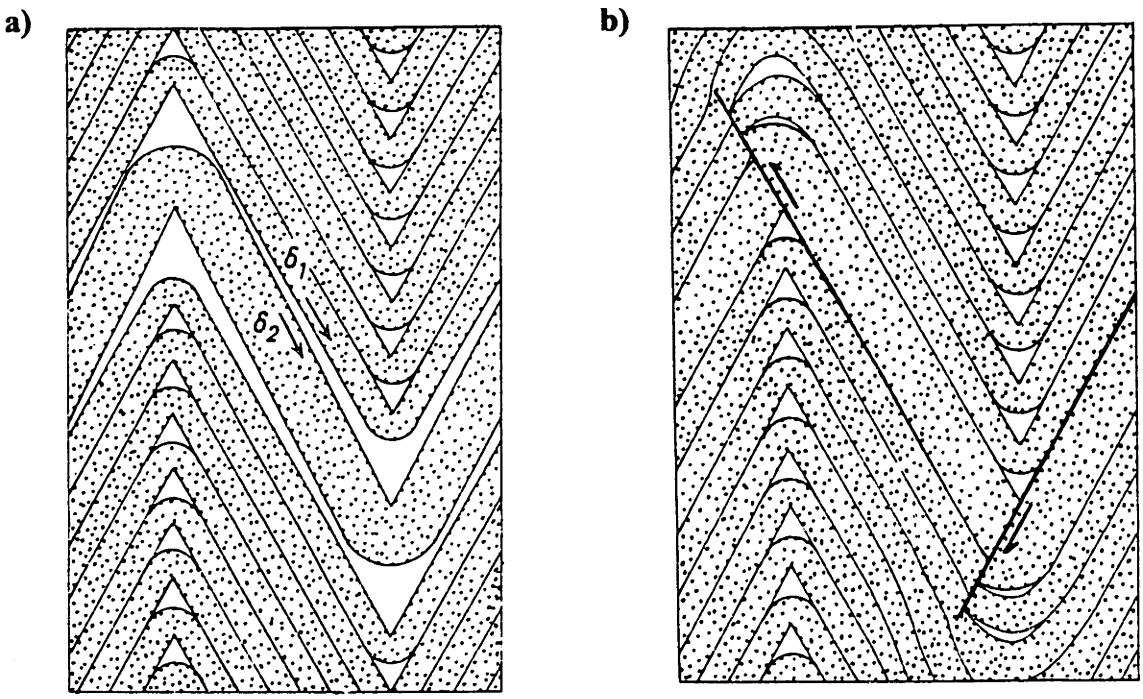


Figure 2.1.29 Thrust faults along bedding planes in the flanks of chevron folds [Ramsay & Huber 1987]: a) development of saddle reef structures in the fold hinge; b) thrust faults accommodating the volumetric increase.

2.2 FRACTURES RELATED TO SHALLOW-DEPTH CRUSTAL FAULTS

Faults are planes of shear failure along which significant displacement of the walls past one another has occurred. Fault planes are parallel to the direction of the intermediate stress σ_2 and form an acute angle (function of the rock friction angle) with the maximum principal stress σ_1. Despite a great variety of fault classifications (see, for example, Suppe 1985), for most purposes there are three major types of faults: normal, strike-slip, and thrust. **Figure 2.2.1** illustrates the three fault types and the characteristic principal stress orientations according to a simplified theory of shallow-depth faulting by Anderson (1951). According to this generally accepted theory, near the surface of the earth one of the principal stresses is vertical and the other two are horizontal. Normal, strike-slip, or thrust faults develop when σ_1, σ_2, or σ_3, respectively, is vertical.

Plate-boundary slip is the most important setting for fault development, although faulting also occurs in response to folding, gravity sliding, dike intrusion, and other processes. The boundaries between the earth's tectonic plates, especially in the continental crust, are composed of many-kilometer-wide zones of active faults. Normal, strike-slip, or thrust faulting is dominant in each of the fundamental types of plate boundaries: divergent, transform, or convergent [Suppe 1985].

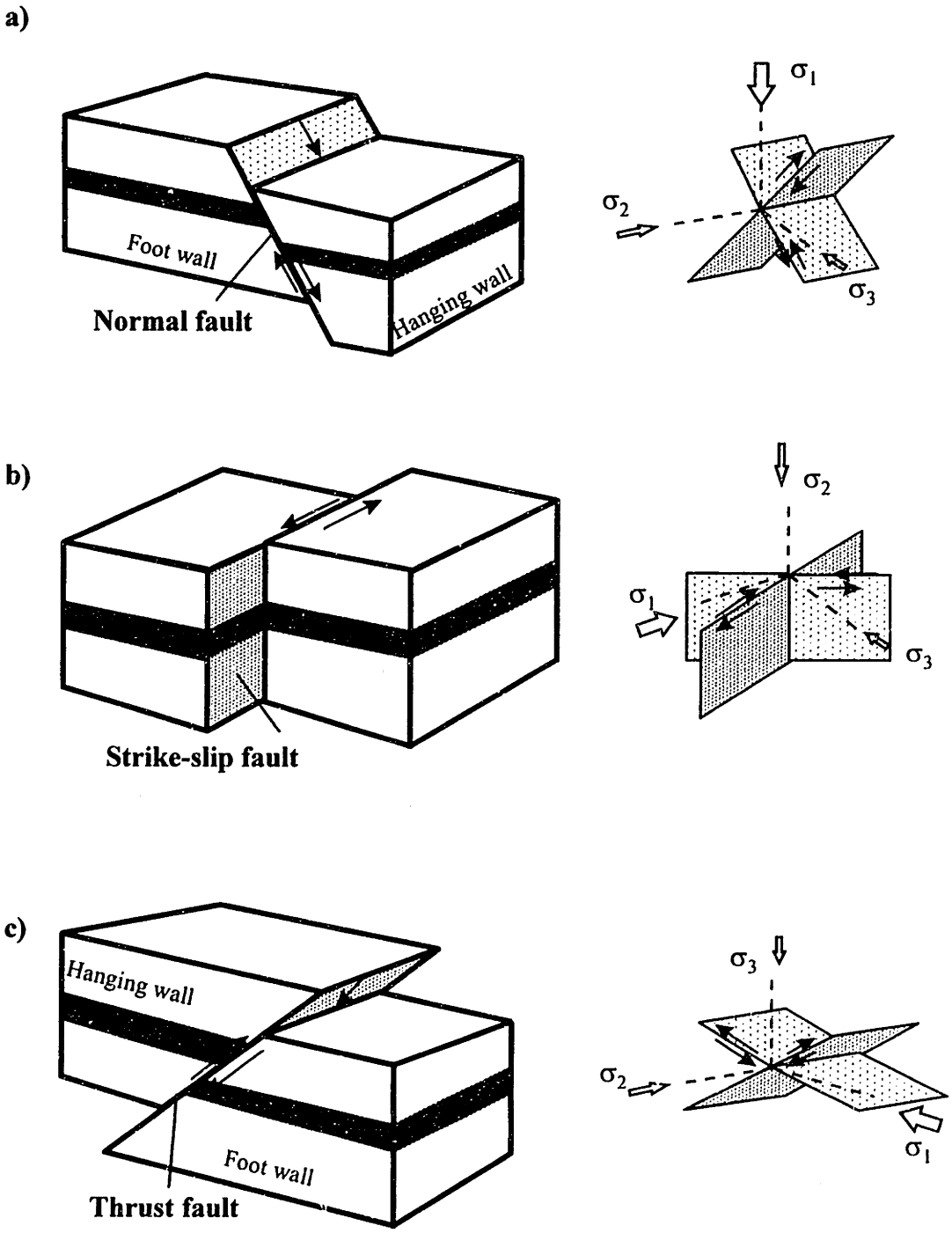


FIGURE 2.2.1 Three major types of crustal faults [after Anderson 1951, Ramsay 1967, and Suppe 1985]: a) normal faults, formed when σ_1 is vertical and σ_3 is horizontal; b) strike-slip faults, formed when σ_1 and σ_3 are horizontal and σ_2 is vertical; c) thrust faults, formed when σ_1 is horizontal and σ_3 is vertical.

Numerous tensile and shear fractures often develop within a broad zone of deformation in the vicinity of a large crustal fault. **Figure 2.2.2** illustrates two types of fractures that are commonly observed near normal, strike-slip, and thrust faults alike [Suppe 1985]. Tensile fractures (joints) form perpendicular to the regional minimum principal stress σ_3. Shear fractures (also called Riedel shears), having the same sense of shear as the main fault and thus adding to the slip, develop along planes oriented between the main fault orientation and the σ_1-σ_2 plane. Many other types of minor fractures may develop as well in the zone of deformation in the vicinity of the main fault. The next three sections present typical fracturing associated with the three major types of crustal faults: normal (Section 2.2.1), strike-slip (Section 2.2.2), and thrust (Section 2.2.3), respectively.

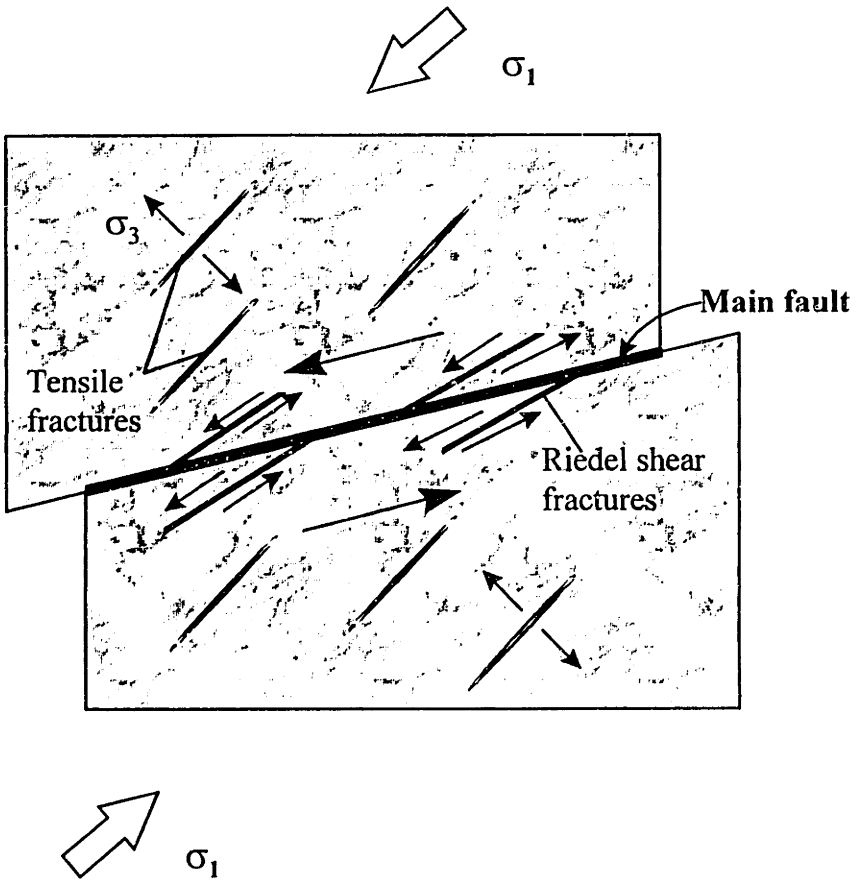


FIGURE 2.2.2 Schematic representation of the orientations and arrangement of tensile fractures and Riedel shears relative to the maximum compressive stress σ_1 and a through-going fault [after Suppe 1985].

Faults often occur in two conjugate sets, the acute angle between which is bisected by the direction of the maximum principal stress σ_1. According to Price & Cosgrove (1990), conjugate faults usually do not intersect. The fault sets may occur as "packets", i.e. regions with one set or the other alternate throughout the faulted rock volume (**Figure 2.2.3**a). Often, one fault set is dominant whereas the conjugate set, if it exists at all, is composed of relatively minor shear fractures (Figure 2.2.3b). Also, since the development of a fault causes decrease in differential stress within its immediate vicinity, new faults in any of the conjugate sets generally form at a certain distance from older faults.

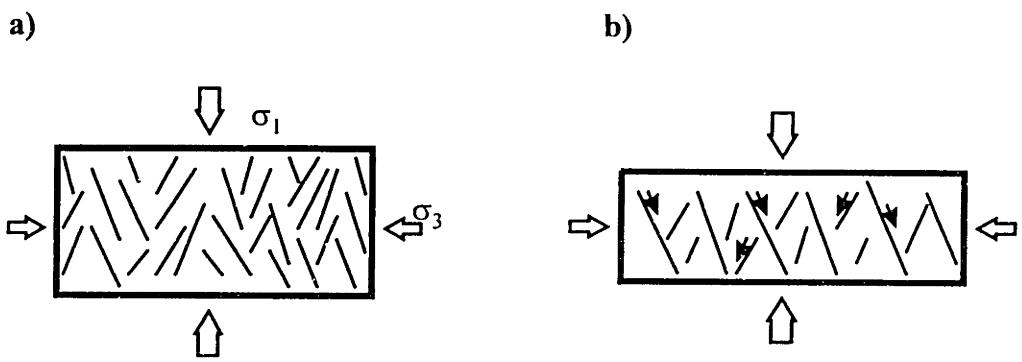


FIGURE 2.2.3 Conjugate fault sets [after Price & Cosgrove 1990]: a) random development of shear faults of one or the other set; b) development of one dominant fault set and another minor conjugate set.

2.2.1 Normal faults and associated fracture systems

Normal faults (Figure 2.2.1a) are dip-slip faults in which the hanging wall (the block overlying the fault plane) has dropped relative to the footwall (the block underlying the fault plane). Normal faults represent shearing planes that are parallel to the horizontal intermediate principal stress σ_2, and make acute angles with the vertical maximum principal stress of gravity, σ_1, and the horizontal minimum principal stress of small compression or tension, σ_3.

Suppe (1985) identifies two distinct kinematic situations that may lead to normal faulting: (1) net horizontal extension of the rock mass, and (2) collapse of rock masses as a result of removal of underlying material. **Table 2.2.1** summarizes geologic settings where these kinematic situations may exist. The table also gives references to other sections of Chapter 2 where one can find more details about how normal faults develop in various geologic environments. Independent of where it forms, a system of normal faults and associated minor fractures exhibits some typical geometric characteristics, which are discussed in this section.

KINEMATIC SITUATION	EXAMPLES	REFERENCE IN THE THESIS
Net horizontal extension	• Continental rifts • Midocean ridges • Large-scale gravity slides on continental margins • Flexural folding	• This section • Section 2.5.2 • Section 2.1.1
Collapse without net extension	• Removal of magma from subvolcanic igneous centers • Removal of underlying evaporites by dissolution • Differential compaction of underlying sediments	• Section 2.5.2 • Section 2.5.2 • Section 2.5.2

TABLE 2.2.1 Geologic settings for normal faulting [after Suppe 1985].

The sequential development of a system of conjugate normal faults (illustrated in **Figure 2.2.4**a) ultimately accommodates horizontal crustal extension and vertical shortening. The movement along conjugate faults inclined towards or away from each other causes relative sinking of some rock blocks, called grabens, and relative uplift of other rock blocks, called horsts (Figure 2.2.4b).

In an existing fracture system the truncated lower fault surface of an old normal fault can be reactivated (point X in Figure 2.2.4a) to generate new faults in the overlying block. Similarly, the truncated upper part of an old fault can be reactivated to cut the underlying block (point Y in Figure 2.2.4a). In most rocks it is easier to reactivate preferentially either the lower or the upper part of truncated old faults. **Figure 2.2.5**a illustrates the horst-graben structure resulting from preferential reactivation of the lower sections of truncated faults, which has led to the marked uplift of a central block of basement into the center of the graben. Figure 2.2.5b depicts the structure resulting from preferential reactivation of the upper parts of truncated faults, which has caused the sinking of a simple graben and the relative uplift of two blocks of basement along the side walls of the horst [Ramsay & Huber 1987].

The dip of normal faults is steep, usually in the range from 55° to 70°. The dip of some near surface faults is even greater, approaching 90°. However, normal faults with dips substantially less than 45° also exist, their low-angle dips being attributed to one of the following reasons: (1) rotation of rock blocks bounded by originally steeper normal faults; or (2) flattening of normal faults in zones of plastic flow or in particularly soft or incompetent layers [De Sitter 1956; Suppe 1985; Ramsay & Huber 1987].

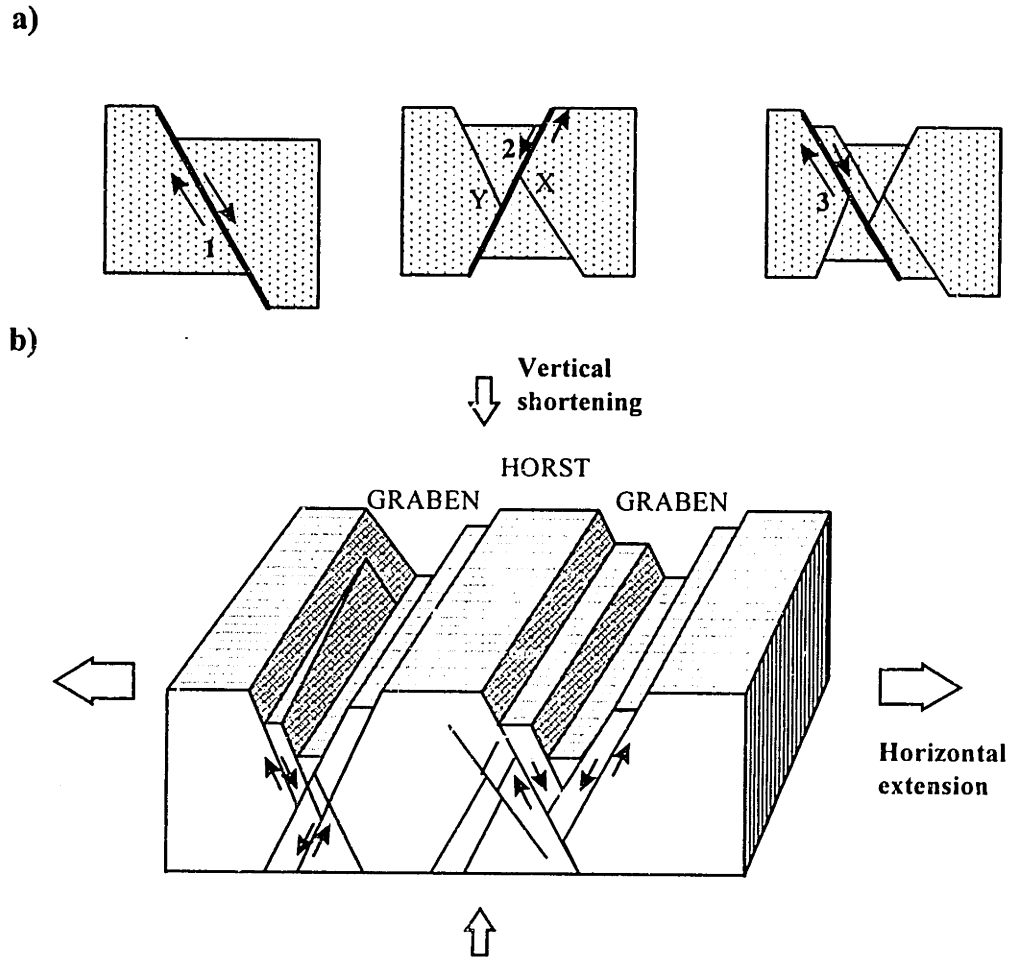


FIGURE 2.2.4 Development of a conjugate system of normal faults [after Ramsay & Huber 1987]: a) sequential movement along faults: fault 1 is displaced by fault 2 and therefore truncated at positions X and Y; fault 3, parallel to fault 1, truncates fault 2; b) typical geometric features of horst and graben.

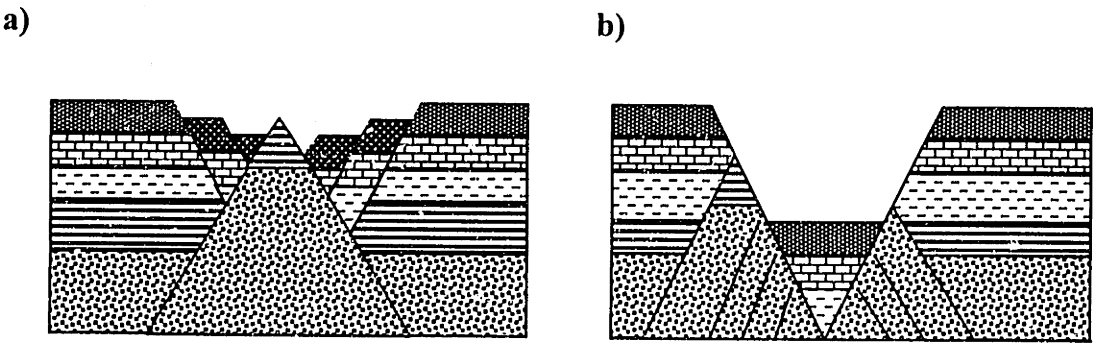


FIGURE 2.2.5 Development of horst and graben structures by reactivating truncated a) lower, or b) upper sections of pre-existing faults [after Ramsay & Huber 1987].

Figure 2.2.6 illustrates tilting of rock blocks, enclosed by a set of normal faults, which accommodates further horizontal stretching [Suppe 1985, Ramsay & Huber 1987]. The structure, bounded by a single set of faults on one side of a tilted fault block, is called half-graben. The uppermost basins are usually filled by sediment composed of eroded material from the uplifted edges of the rotated blocks. At depth, the spaces between rotated blocks are accommodated either by ductile flow of the underlying material or by magma infillings. Once the blocks have rotated, frictional resistance may inhibit further slip along the flattened fault surfaces. Further horizontal stretching is then accommodated by development of a new set of steep faults, cutting across the old flatter faults (Figure 2.2.6b), followed by rotation of the newly bounded rock blocks. Figure 2.2.6c illustrates a system of normal faults, created by such a mechanism. The older faults in the system formed at 55° to 70° to the horizontal bedding; then the beds rotated to steeper dips and the faults became nearly horizontal. The younger normal faults of the system are less rotated and have dips closer to the initial dip.

The dip of a normal fault, generally steep near the surface, may vary at depth due to different competency of the faulted strata. The dip is flatter in soft incompetent layers than it is in brittle competent layers. An abrupt change of dip, so-called steep-and-flat structure, may develop when the fault crosscuts strata of distinctly different lithology. For example, the fault in **Figure 2.2.7** is steeper in the brittle Austin chalk, less steep in the soft Eagle Ford shale, and steeper again in the sandstone. If the rheological properties of the strata change more gradually at depth, the fault dip exhibits continuous flattening. In such a case, a normal fault develops a curved, concave upward, so-called listric, profile in cross section [De Sitter 1956; Ramsay & Huber 1987].

Typical fracture systems associated with listric normal faults include step faulting, rotation of blocks, and antithetic faults that counter dip towards the main fault (**Figure 2.2.8**). Figure 2.2.8a shows a system of concave upward listric faults and the rock blocks enclosed by the curved fault surfaces. Figure 2.2.8b illustrates the structure that forms after the blocks slide progressively along the sub-parallel normal faults (a process called step faulting) and also along a flat underlying fault (detachment fault). Some vertical subsidence and thinning of the strata also take place to accommodate the rotation of rock blocks. The flexure of the upper strata may result in open fissures, perpendicular to the layers, as shown in Figure 2.2.8b. In brittle rocks, progressive sliding of rigid rock blocks along a system of listric faults may cause development of a counter dipping set of faults, called antithetic faults. Figure 2.2.8c depicts a conjugate set of antithetic faults with listric profile, dipping towards the main set of listric faults. [De Sitter 1956; Ramsay & Huber 1987]. Detachment faults, such as those in Figure 2.2.8b and c are special types of flat or low-angle normal faults that accommodate horizontal crustal extension. Detachment faults commonly develop as a low angle base to a complex normally faulted terrain. They often run close and parallel to the contact between a crystalline basement complex and an overlying faulted sedimentary cover.

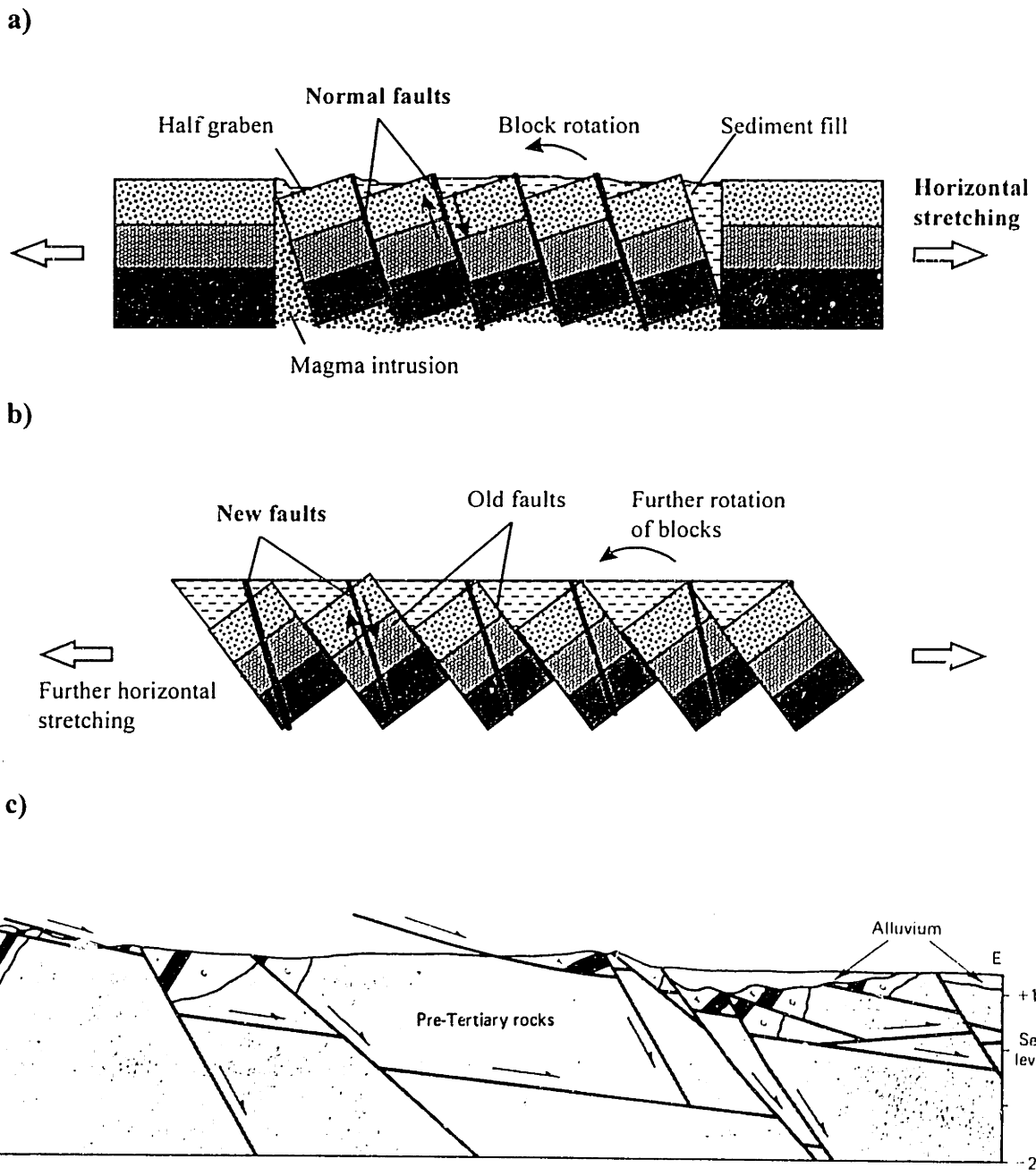


FIGURE 2.2.6 Rotation of blocks bounded by a set of normal faults, accommodating horizontal crustal extension: a) initial faulting and development of a half-graben structure by tilting of blocks, and b) development of a second set of steep normal faults, cutting across the flattened old faults [after Ramsay & Huber 1987]; c) normal fault structure of the Yerington district, Nevada, developed by the mechanism illustrated in (a) and (b) [from Suppe 1985].

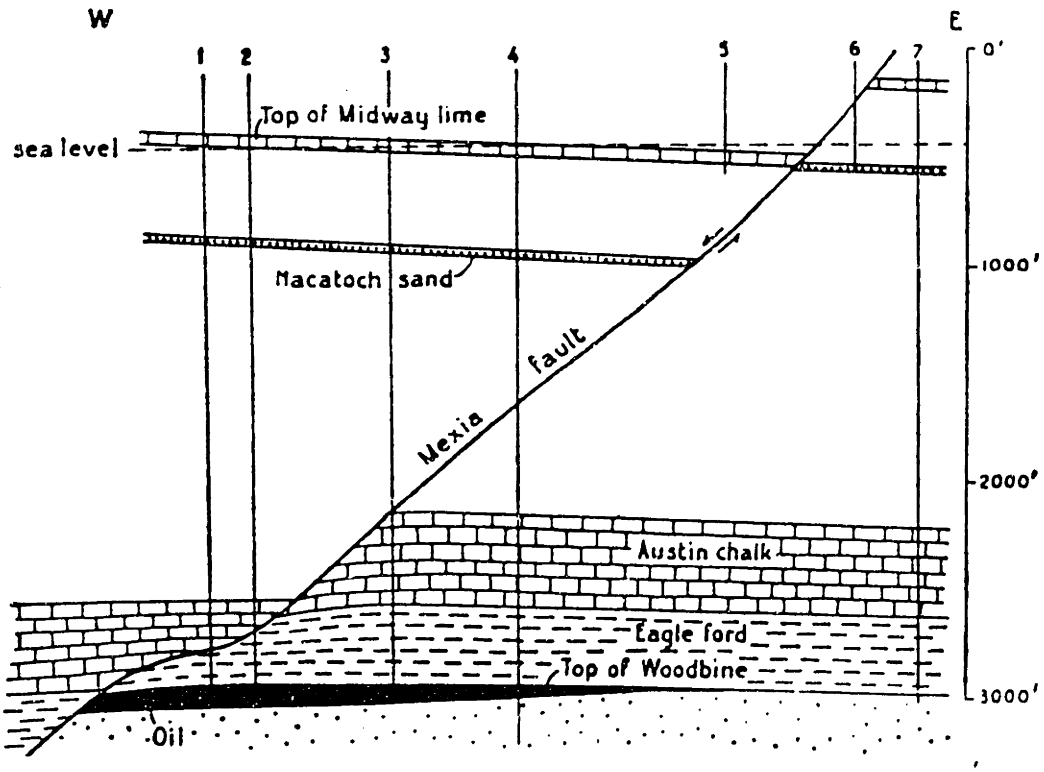


FIGURE 2.2.7 Section through the Mexia fault, Texas, showing variation of dip with different nature of rock [from De Sitter 1956].

Xiao et al. (1983) studied the mechanics of so-called "extensional wedges" and characterized them by: "1) a gently dipping underlying detachment fault along which the tapered hanging wall slides down dip; 2) a related system of steeply dipping normal faults in the overlying wedge; and 3) relatively little deformation in the footwall beneath the basal detachment fault". Normal fault systems typical for extensional wedges are illustrated in **Figure 2.2.9**.

Dilational fractures (tensile joints) have been observed in relation to normal faults. **Figure 2.2.10** illustrates a system of large normal faults accompanied by numerous parallel small joints, described by De Sitter (1956). The normal faults have a general NW-SE trend but often deviate from this direction thus joining and separating again. The joint sets are steeply dipping and very consistently parallel to the nearest normal fault whatever its strike might be. Between two main faults of different direction, both directions are represented in the joint pattern. The number of joints does not increase near the faults. The rock in this case is a very soft rock (Miocene lignite). According to De Sitter, in other cases, when the rocks were folded, cemented and lithified before faulting started, no joints can be related to normal faulting.

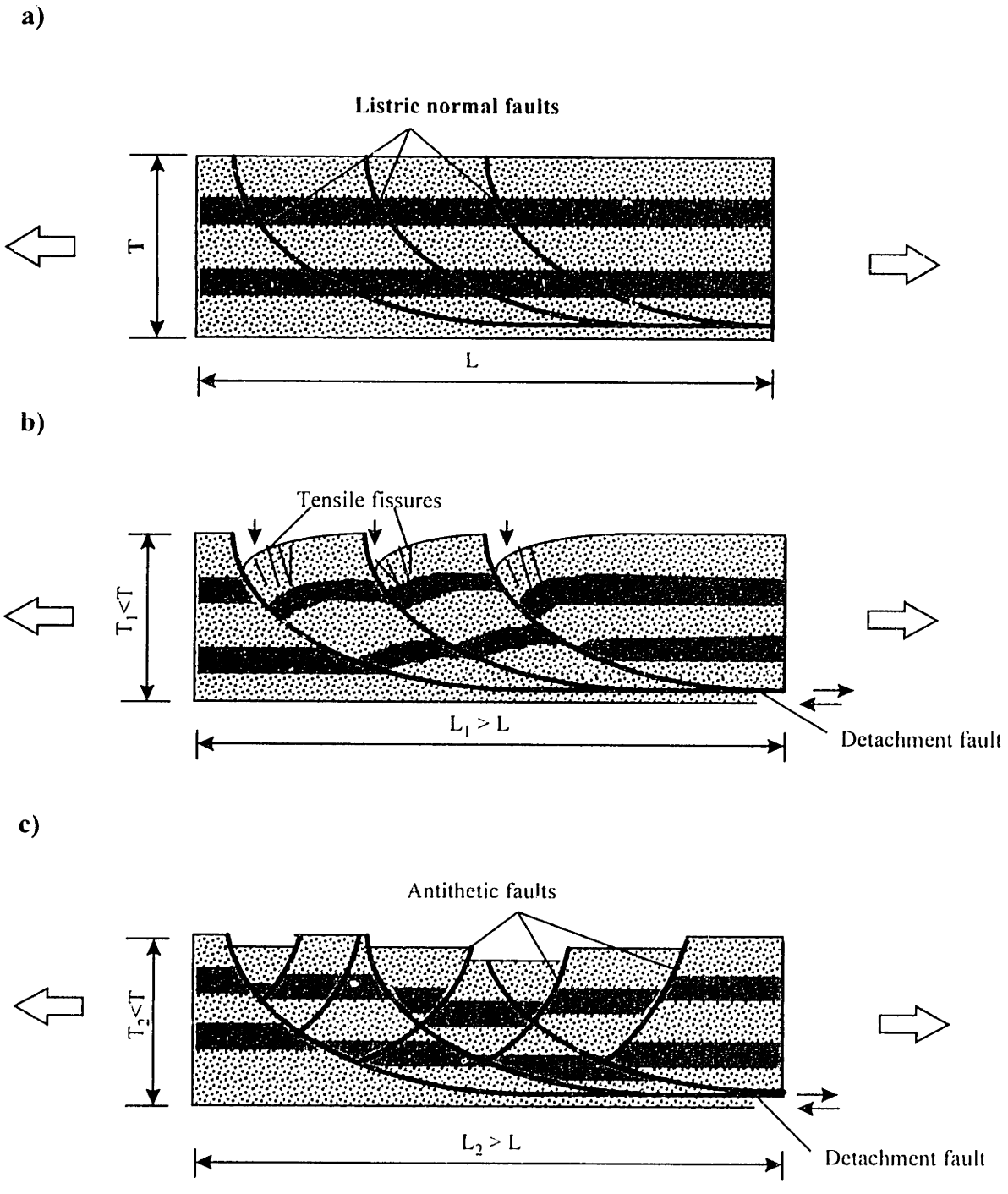


FIGURE 2.2.8 Fracture systems associated with listric normal faults [after Ramsay & Huber 1987]: a) initial position of sub-parallel listric faults; b) movement of rock blocks with vertical subsidence of strata within the blocks; c) development of conjugate listric faults (antithetic faults), counter dipping to the main set.

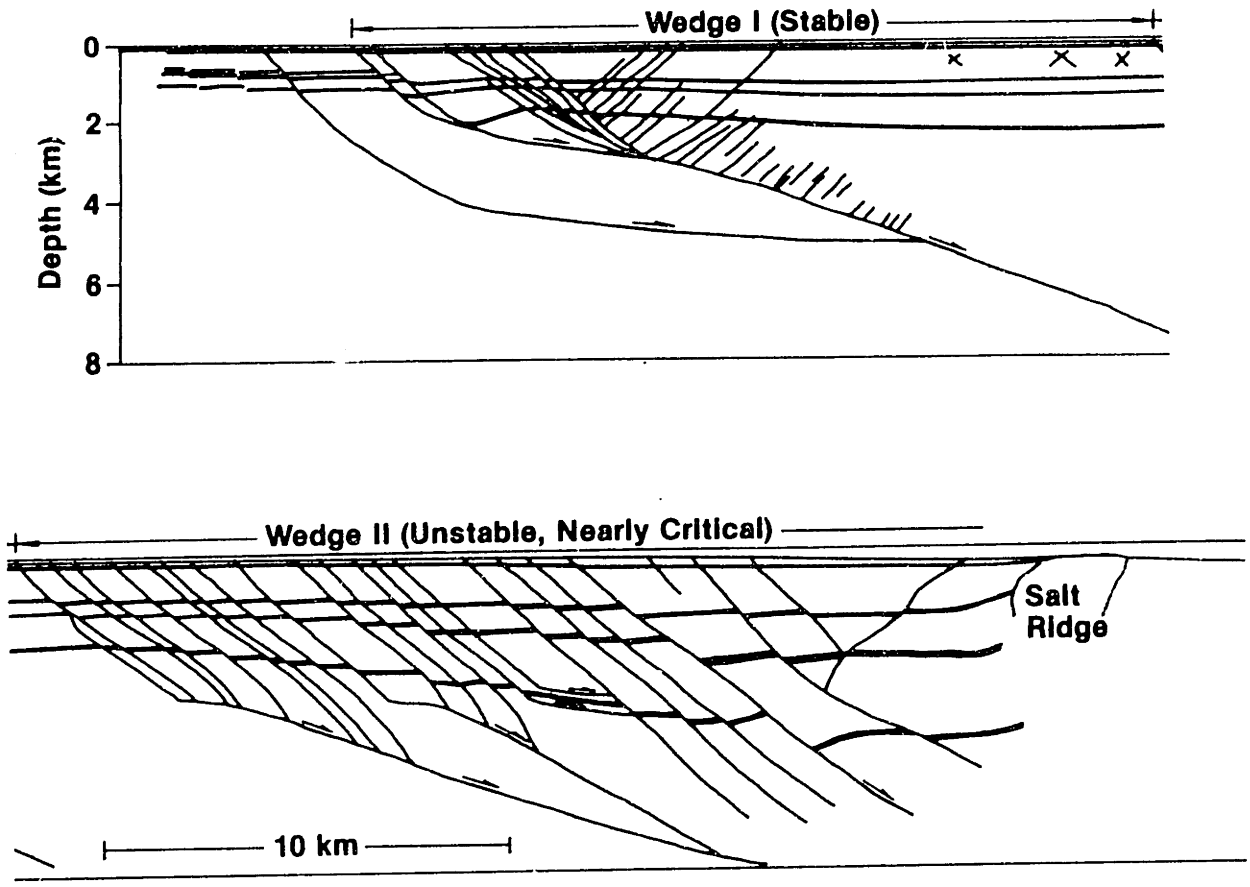


FIGURE 2.2.9 Two extensional wedges in Brazos area, offshore Texas [from Xiao et al. 1983].Wedge I is stably sliding on the Brazos Ridge detachment fault, whereas wedge II is unstable and nearly critical. Profile is obtained by interpretation of seismic reflection profile.

The down-dip displacement on the face of a normal fault that terminates into intact rock is not constant: along the strike of a fault, it is maximum in the middle and dies out near the edges (**Figure 2.2.11**a). In a sedimentary sequence, composed of layers of different thickness and material properties, the varying displacement is usually associated with flexure of one or more strong units that control the development of the fault [Price & Cosgrove 1990]. Figure 2.2.11b illustrates the flexure of a competent unit in the down-thrown wall of a normal fault and the corresponding stresses acting parallel to the fault plane. In the upper layers of the flexed unit there is local increase of compression in the middle portion of the fault, whereas tensile stresses may develop near the edges. In the layers below the neutral surface the stress distribution is reversed: increased

compression near the edges, and tension in the middle. If the stress is high enough to exceed the strength of the rock, secondary faults, adjacent to the major normal fault, may develop (Figure 2.2.11c). Secondary normal faults (N_1 and N_2 in Figure 2.2.11c), striking at 90° and terminating at the main fault, form in the zones of local tension. Minor strike-slip faults (S_1 and S_2 in Figure 2.2.11c) and possibly thrusts, oriented so that the acute angle between conjugate faults is bisected by σ_1 which acts parallel to the major fault plane, develop in the layers subjected to increased compression.

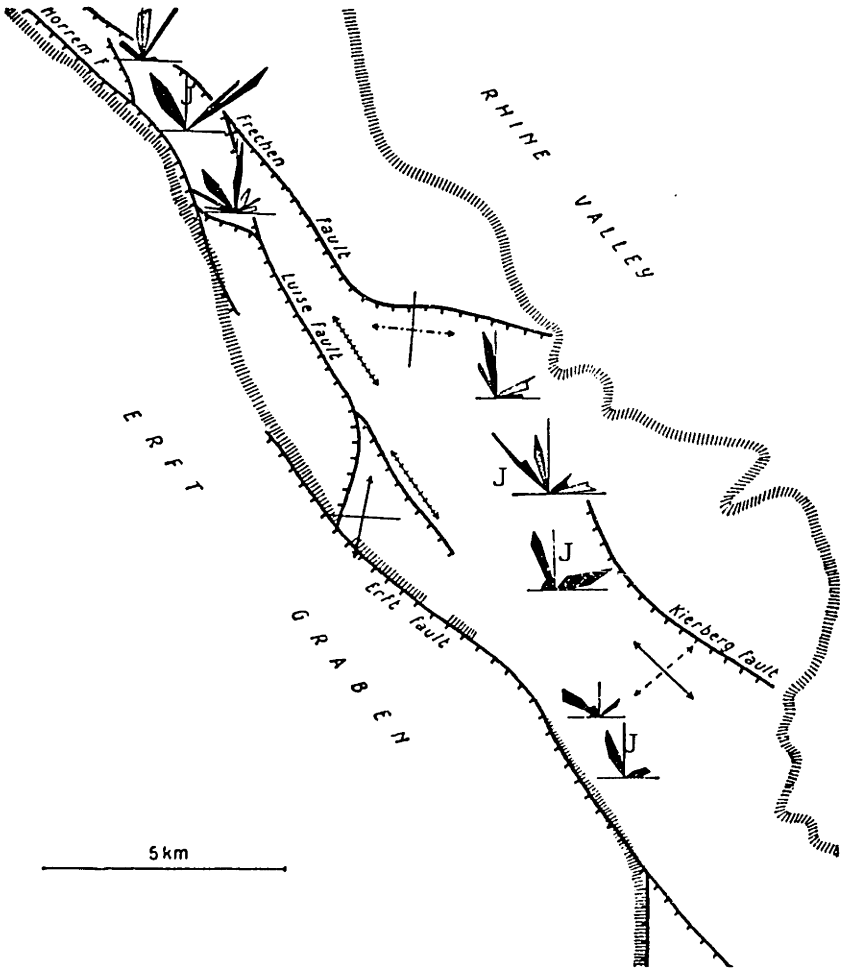
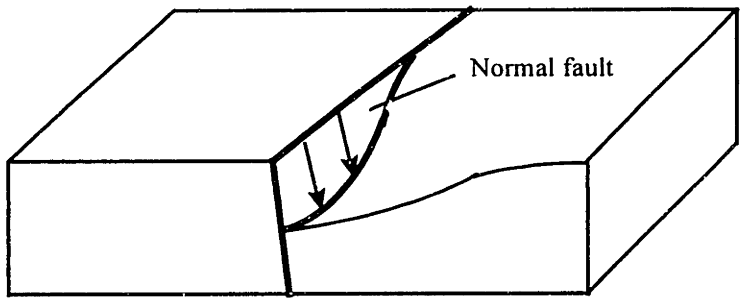
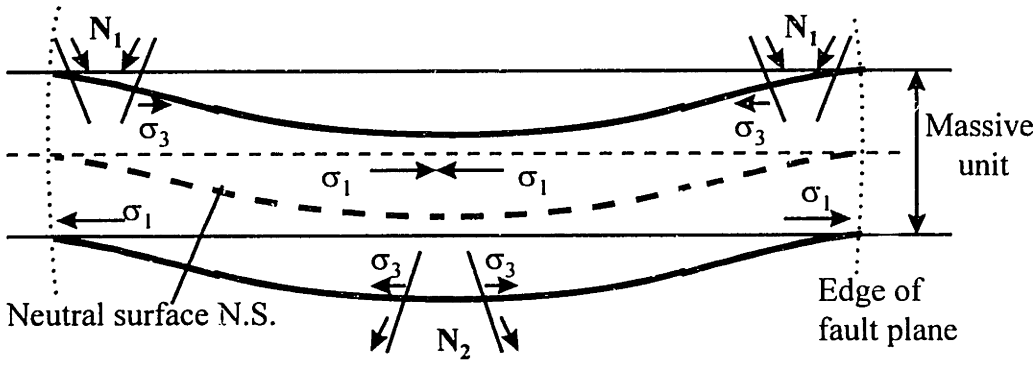


FIGURE 2.2.10 Joints parallel to normal faults (indicated as J) in Miocene lignite, Cologne, Germany [from De Sitter 1956].

a)



b)



c)

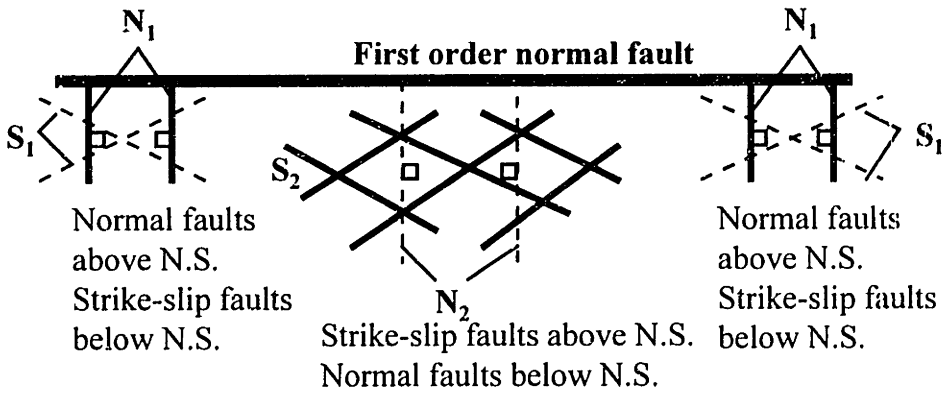


FIGURE 2.2.11 Secondary faults associated with varying down-dip displacement along the strike of a normal fault [after Price & Cosgrove 1990]: a) general form of displacement along half of the fault face; b) compressive and extensional stresses induced by the flexure of a massive control unit; c) plan view of minor normal and strike-slip faults associated with the major normal fault.

Normal faulting may occur in en-echelon arrangement (**Figure 2.2.12**). In a system of en-echelon normal faults the vertical displacement across the region is rather constant, although the local down-dip movement on an individual fault varies. The build-up of local stresses within the fault bridges between the fault segments may lead to splaying of the edges and subsequent sideways propagation of the fault tips towards adjacent faults [Ramsay & Huber 1987].

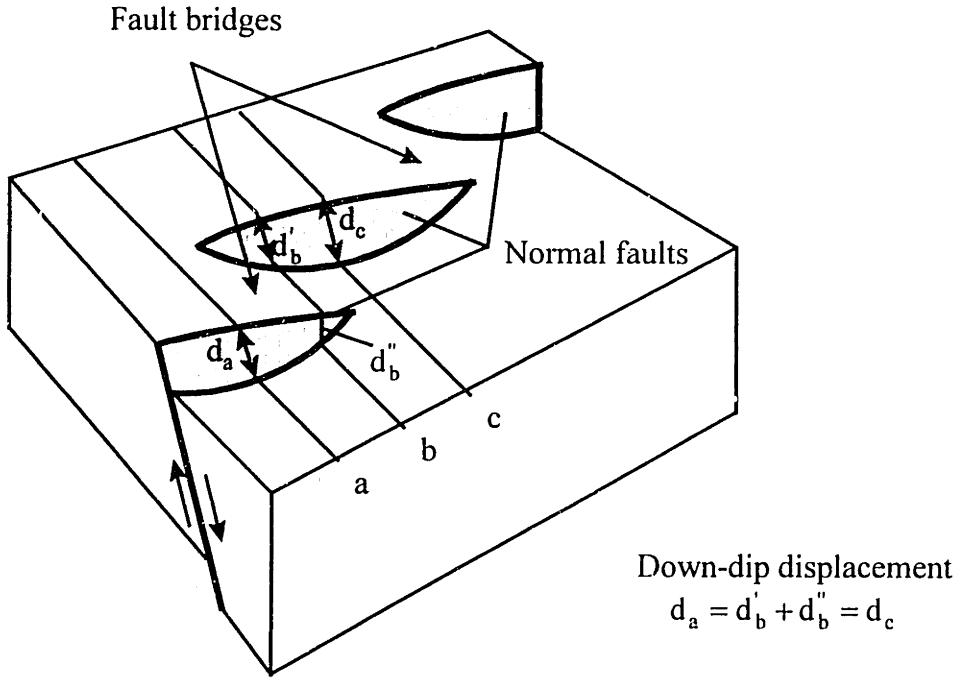


FIGURE 2.2.12 Possible en-echelon arrangement of normal faults [after Ramsay & Huber 1987].

2.2.2 Strike-slip faults and associated fracture systems

Strike-slip faults are steeply dipping, often vertical, planes of shear failure, along which the relative displacement of the walls past one another is predominantly horizontal (i.e. along the strike). Strike-slip faults develop when the maximum and the minimum principal stresses, σ_1 and σ_3, respectively, are horizontal, and the intermediate principal stress, σ_2, is vertical (Figure 2.2.1b). A strike-slip fault usually makes an angle of approximately 30° with the direction of σ_1 [Ramsay 1967; Billings 1972]. If a second conjugate set develops, it strikes at about 60° to the first set. Conjugate strike-slip faults have the opposite sense of relative movement of the walls, determined by the direction in which the wall opposite of the observer moves. One of two conjugate faults is right-hand (dextral) and the other one is left-hand (sinistral), as illustrated in **Figure 2.2.13**.

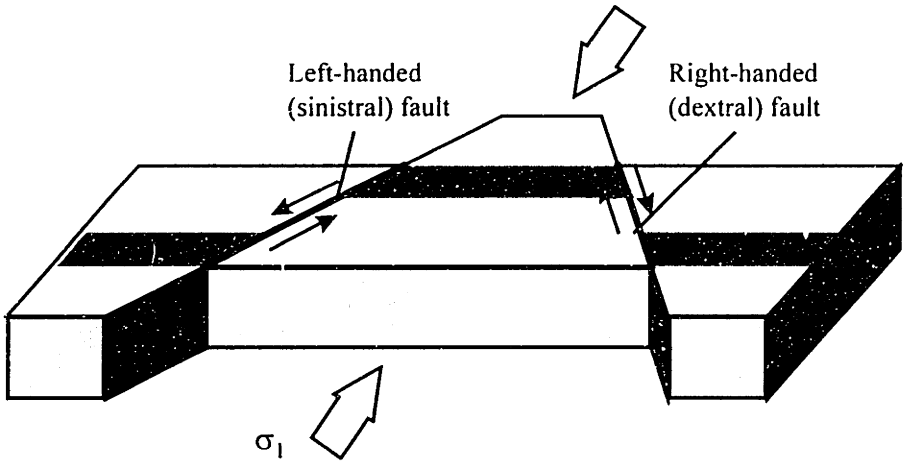


FIGURE 2.2.13 Formation of a conjugate pair of strike-slip faults showing opposite sense of shear displacement [after Billings 1972].

There are two important geologic settings of strike-slip faulting. First, major strike-slip faults exist along the transform boundaries between tectonic plates in the continental and oceanic crust. Examples of strike-slip faults of regional or continental extent (called transform faults) are the San Andreas fault in California (total horizontal displacement of about 500 km), the Great Glen fault in Scotland (total displacement of about 150 km), the Alpine fault in New Zealand, etc. [Price & Cosgrove 1990; Suppe 1985]. Second, strike-slip faults of relatively smaller scale (several meters to a few tens of kilometers) commonly develop in the upper levels of the crust in relation to other structures such as flexural folds or normal and thrust faults. For example, strike-slip fault sets characteristic of different stages of the folding process are discussed in Section 2.1.1 (see Figures 2.1.11 and 2.1.12).

In all geologic settings, strike-slip faults of all scales are accompanied by numerous secondary and younger faults and tensile fractures. **Figure 2.2.14** schematically illustrates secondary structures that may develop adjacent to first order strike-slip faults in basement rocks (Figure 2.2.14a) and in cover rocks (Figure 2.2.14b) in the earth's crust. The figure represents compilation by Price & Cosgrove (1990) that reflects the work of many authors [for example, Riedel 1929, Anderson 1951, McKinstry 1953, Moody & Hill 1956, Price 1968]. The rest of this section reviews in detail the genesis of the secondary fractures, illustrated in Figure 2.2.14, and the characteristic geometry of the complex fracture systems associated with major strike-slip faults.

Secondary faults adjacent to strike-slip faults in basement rocks

Figure 2.2.15 schematically illustrates the two types of second order faults that most commonly develop adjacent to a primary strike-slip fault in basement rocks [Price & Cosgrove 1990]: a) splay faults, and b) secondary strike-slip faults.

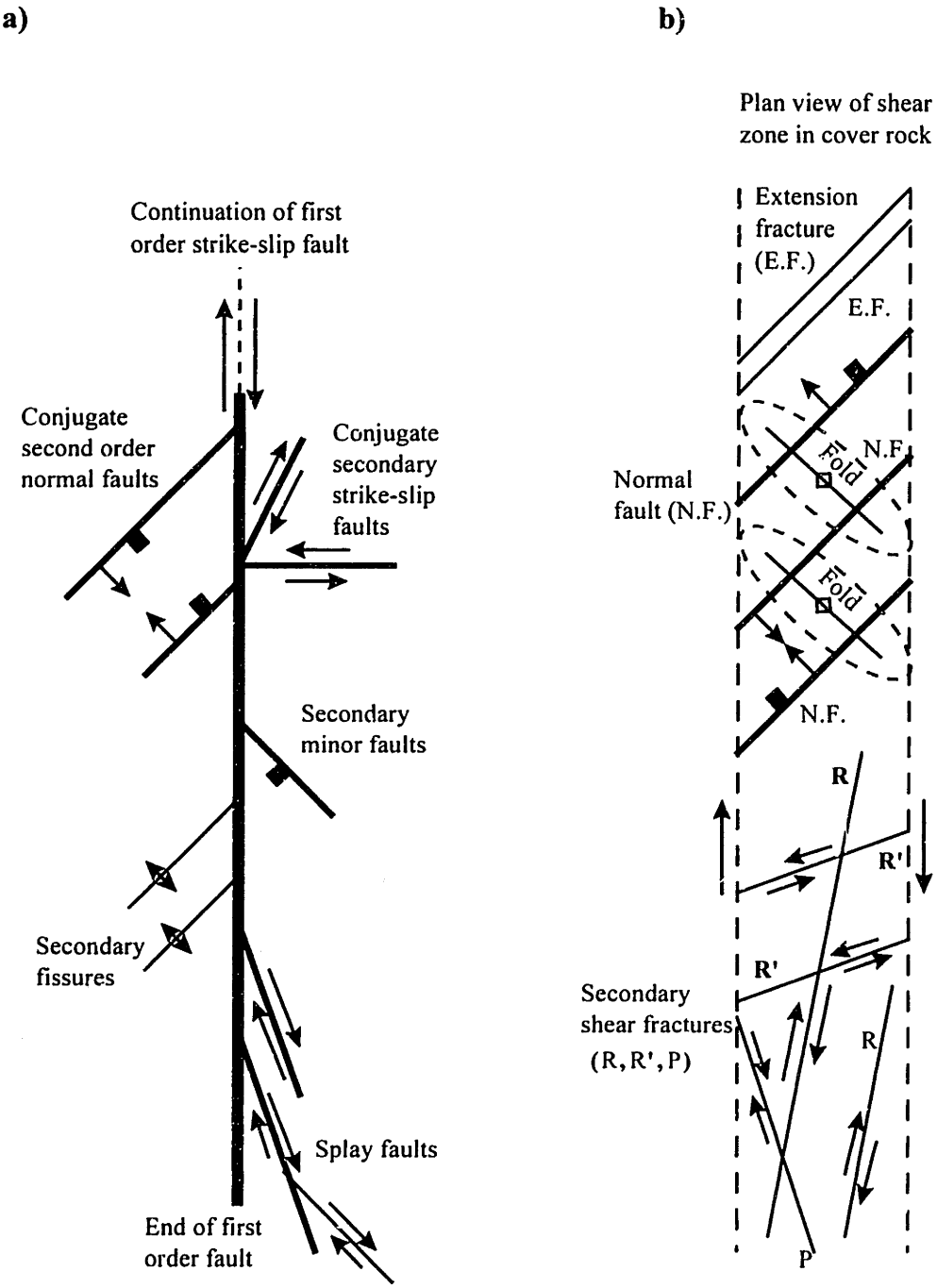
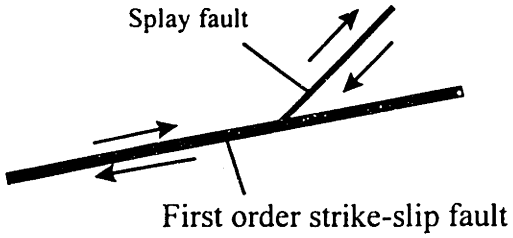


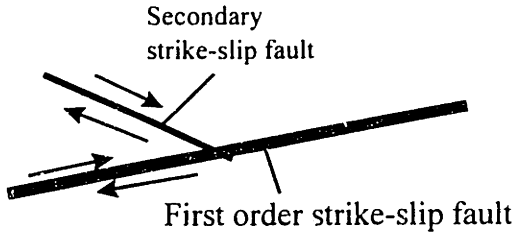
FIGURE 2.2.14 Compilation of secondary features that commonly develop due to shearing along a first order strike-slip fault [after Price & Cosgrove 1990]:
a) secondary faults adjacent to a major strike-slip fault in basement rocks;
b) secondary fractures and folds in cover rocks above a strike-slip fault in the basement.

a)



Example:
Ingleborough faults, England
[Anderson 1951]

b)



Example:
Yellowknife District faults, Canada
[McKinstry 1953]

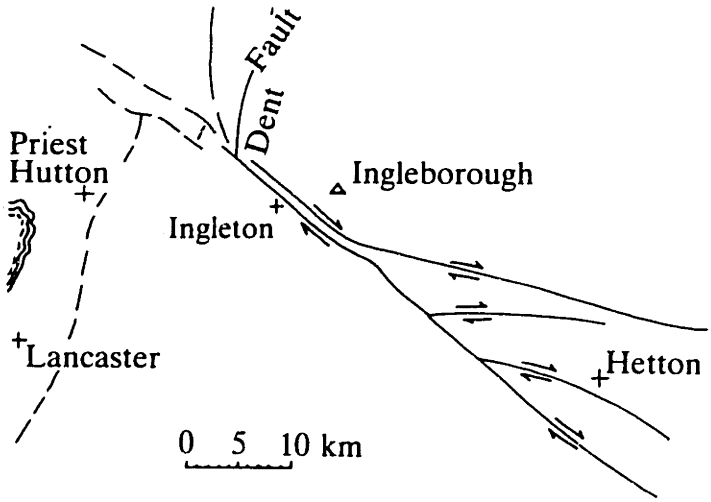


FIGURE 2.2 '5 Secondary faults adjacent to a major strike-slip fault in basement rocks [after Price & Cosgrove 1990]: a) splay faults which form near the end of a major fault, b) secondary strike-slip faults which develop in regions remote from the ends of the first order fault.

Splay faults (Figure 2.2.15a) form near the ends of major strike-slip faults. Secondary strike-slip faults of the type illustrated in Figure 2.2.15b develop in regions that are remote from the ends of the primary strike-slip fault. Examples of splay and secondary strike-slip faults, adjacent to a major strike-slip fault in basement rocks, are also shown in Figure 2.2.15a and b, respectively.

Price (1968) speculates that secondary faults develop adjacent to a first order strike-slip fault due to transient stresses, created by elastic waves induced in the unfractured rock by the shear movement along the primary fault. **Figure 2.2.16** illustrates the orientations of the static principal stresses, σ_1, σ_2, and σ_3, that cause shearing along the primary fault, and that of the transient principal stresses, σ'_1, σ'_2, and σ'_3. The type and orientation of the secondary faults depend on the value of the static intermediate principal stress σ_2. Secondary strike-slip faults (S in Figure 2.2.16) develop if the vertical σ_2 lies between σ'_1 and σ'_3. If σ_2 is greater than the maximum horizontal transient principal stress, σ_2 becomes σ'_1, and secondary normal faults may form (N in Figure 2.2.16). According to Price & Cosgrove (1990), once a first order strike-slip fault has been initiated, the vertical stress may become the least principal stress σ_1 and further movement along the fault can lead to thrust faulting.

Secondary normal or thrust faulting can also be induced in zones of local extension or compression, respectively, adjacent to first-order strike-slip faults. For example, strike-slip faults may extend between two zones of crustal extension. This is typical of transform faults between tectonic plates since they often connect zones of sea floor spreading at divergent plate boundaries (**Figure 2.2.17**a). Similarly, a strike-slip fault may develop on a smaller scale in the earth's crust between two graben structures defined by a system of normal faults (Figure 2.2.17b). The faults illustrated in Figure 2.2.17 are usually characterized by a constant amount of horizontal displacement along the surface of the fault. When a strike-slip fault terminates into intact rock, the horizontal displacement varies along the strike, being maximum in the central part of the fault and gradually decreasing near the ends. The variation of horizontal displacement may lead to development of secondary normal or thrust faults across the surface of the primary strike-slip fault, as illustrated in **Figure 2.2.18**.

En-echelon arrangement of strike-slip faults is common [Ramsay & Huber 1987; Suppe 1985; Price & Cosgrove 1990]. **Figure 2.2.19** schematically illustrates two cases of en-echelon sets of strike-slip faults. The faults in both cases are sinistral whereas the en-echelon arrangement can be classified as dextral (right-hand) in Figure 2.2.19a, and sinistral (left-hand) in Figure 2.2.19b. When left-hand faults step en-echelon to the right (or vice versa), compression stresses are created in the overlap zone. This compression can be relieved by formation of thrust faults in the overlap zone (Figure 2.2.19a). In the case of left-hand en-echelon arrangement of left-hand faults, extension takes place in the overlap zone. This may cause development of secondary normal faults striking obliquely to the

main faults (Figure 2.2.19b). In both cases, the orientation of the secondary faults depends on pre-existing weaknesses of the rock such as previous fracturing. According to Ramsay & Huber (1987), if the overlap zones are initially unfractured, the secondary faults form at about 45° to the main fault traces. Pull-apart basins in extensional zones or overlap ridges in compressional zones, and the associated normal or thrust faulting, respectively, may also result from strike-slip movement along a kinked or curved strike-slip fault, as shown in Figure 2.2.19c.

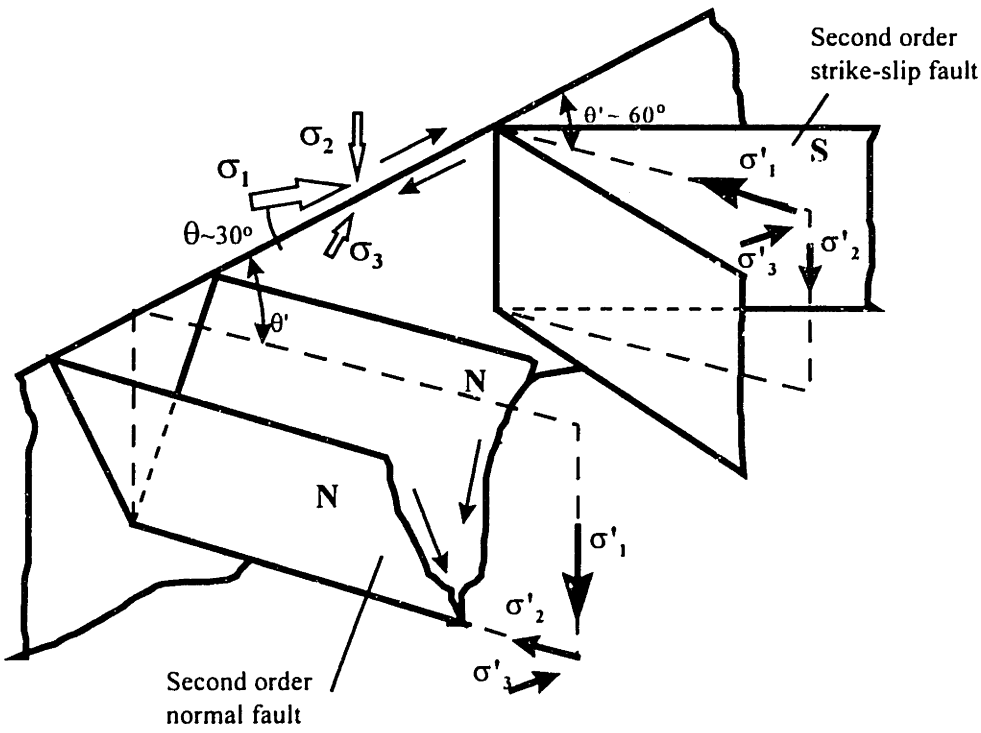


FIGURE 2.2.16 Secondary strike-slip fractures created by transient stresses [after Price & Cosgrove 1990]: second order strike-slip faults (S) if the vertical stress is the temporary intermediate principal stress, and second order normal faults (N) if the vertical stress is the temporary maximum principal stress.

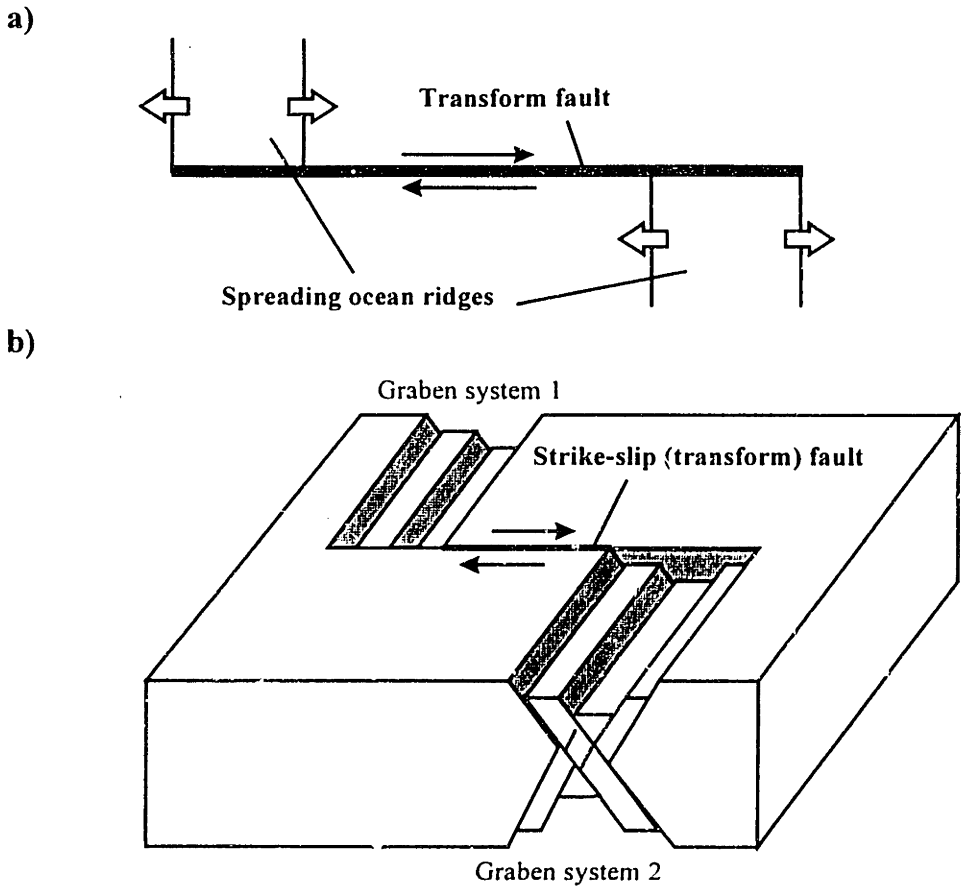


FIGURE 2.2.17 Strike-slip faults linking extensional zones: a) diagrammatic representation of a transform fault connecting two, off-set, spreading oceanic ridges [after Price & Cosgrove 1990]; b) a strike-slip fault linking two graben systems, bounded by normal faults [after Ramsay & Huber 1987].

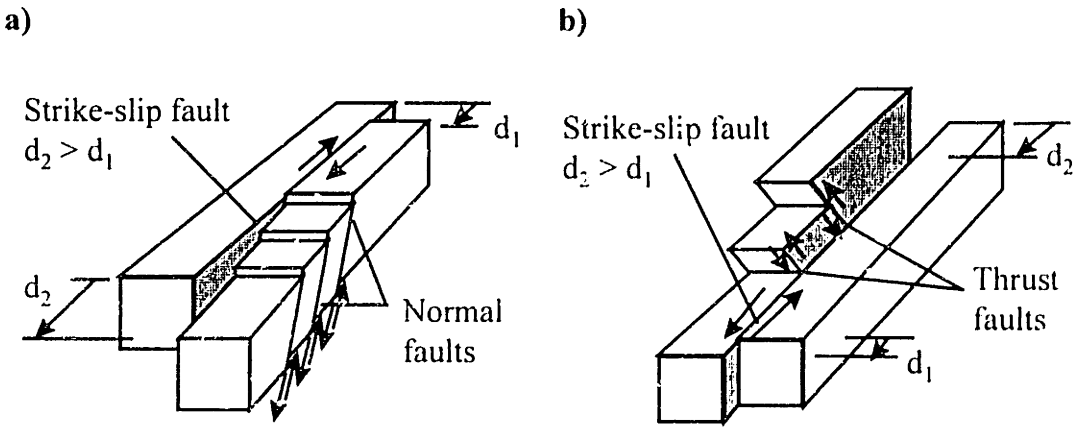


FIGURE 2.2.18 Secondary a) normal, or b) thrust faults due to different amount of horizontal movement along the surface of a primary strike-slip fault [after Ramsay & Huber 1987].

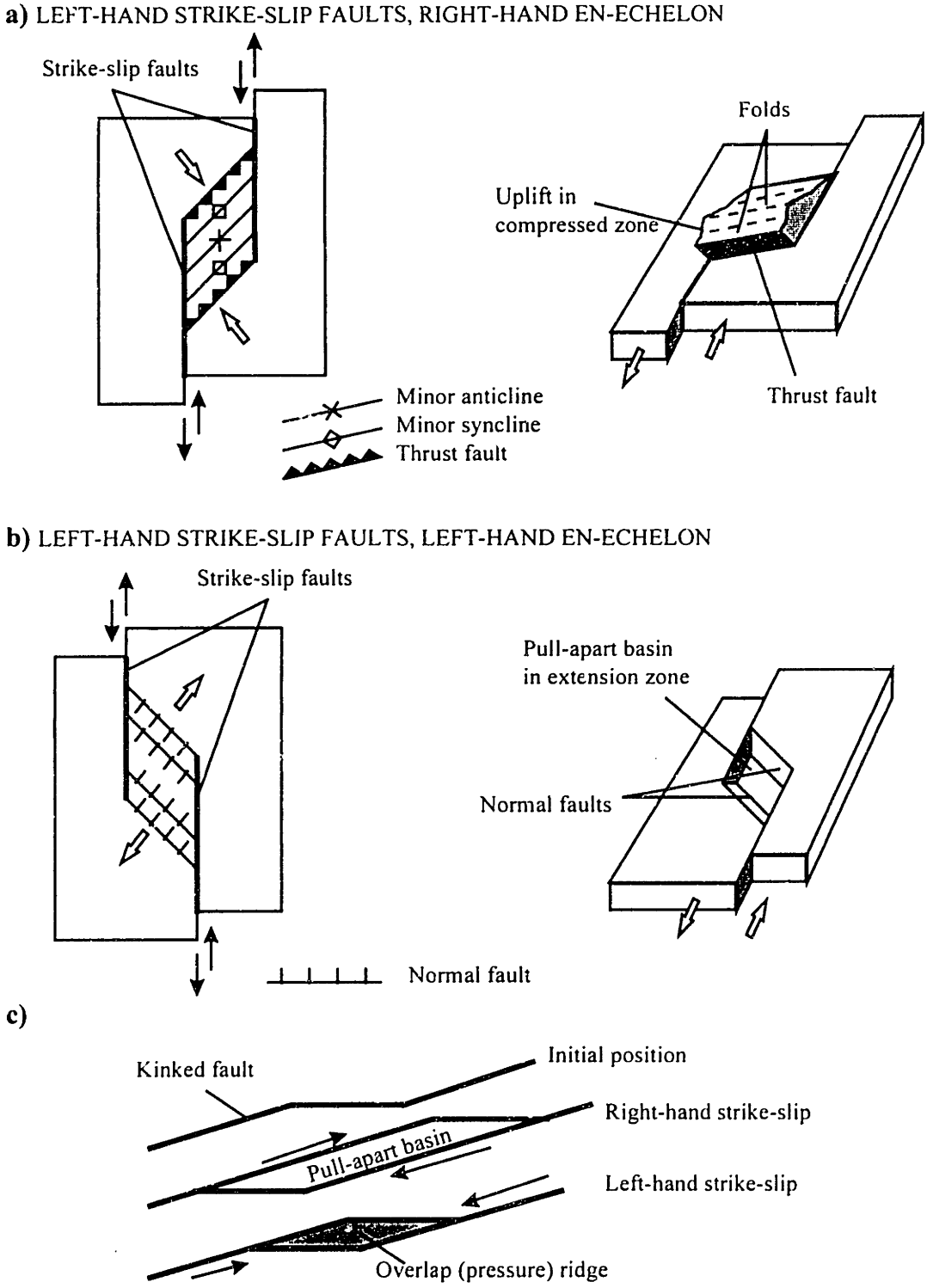


FIGURE 2.2.19 Formation of secondary structures in the overlap zones of strike-slip faults [after Price & Cosgrove 1990 and Ramsay & Huber 1987]: a) compression in the overlap zone of en-echelon faults resulting in thrust faulting and vertical uplift; b) increased extension in the overlap zone of en-echelon faults resulting in normal faulting and pull-apart basin; c) development of a pull-apart basin or a pressure ridge due to movement along a kinked strike-slip fault.

Figure 2.2.20 shows examples of secondary faults, similar to those schematically represented in Figure 2.2.19, that have formed in the overlap zones of en-echelon strike-slip faults. Figure 2.2.20a shows two examples of overlap zones subjected to extension due to a right-hand slip along strike-slip faults arranged in right-hand en-echelon arrays. The extension has caused transverse normal faulting and formation of pull-apart basins (occupied by a lake and a valley, respectively). Figure 2.2.20b depicts the large scale thrust faulting in the zone adjacent to a bend in the San Andreas fault in California. The zone is under compression due to the opposite directions of the right-hand shear along the fault and the left-hand bend in the fault trend.

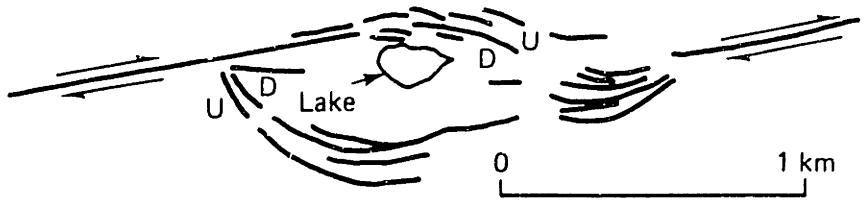
Fracturing in cover rocks due to strike-slip faulting in the basement

Shear along a strike-slip fault (or fault zone) in the basement commonly results in development of an intensely fractured shear zone in the overlying rocks. Typical structures in the cover rocks include: secondary folds, extension fractures, and several types of shear fractures (Figure 2.2.14b). **Figure 2.2.21** shows a large scale example of en-echelon folds created by compression in the cover rocks, induced by shear in the basement along the San Andreas fault in California. The inset shows possible fracture patterns induced by folding due to a shear couple.

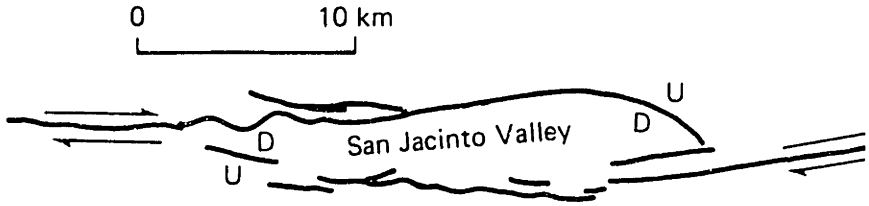
Figure 2.2.22 illustrates the major types of fractures induced in the cover by shear of the basement in laboratory experiments: (1) extensional fractures, parallel to the direction of σ_1 within the shear zone, and (2) two sets of shear fractures (also called Riedel shears). Figure 2.2.22a depicts the experimental setup in the classical work by Riedel (1929) who studied fracture patterns induced in clay sheets by direct shear in the support block. Simple shear experiments on clay sheets were performed by Cloos (1955). Figure 2.2.22b illustrates the orientations of extension fractures and Riedel shear fractures (R and R') in the shear zone. An additional set of shear fractures (P in Figure 2.2.22b) was observed by Tchalenko (1968). **Table 2.2.2** summarizes the experimental results.

In nature, strike-slip faulting usually leads to the development of fracture systems that have much more complex geometry than that of the simple patterns produced in laboratory experiments. Once secondary fracturing of the type illustrated in Figures 2.2.14, 2.2.15a, and 2.2.22b has occurred, the principal stresses rotate and further shearing along the first order faults produces secondary fractures oriented according to the new principal stress directions. The lithology of the rock also affects the characteristic fracturing in strike-slip shear zones. The rest of Section 2.2. reviews some well documented natural fracture systems that demonstrate the characteristic geometry of fracturing associated with strike-slip faults in crystalline and in layered rocks.

a)



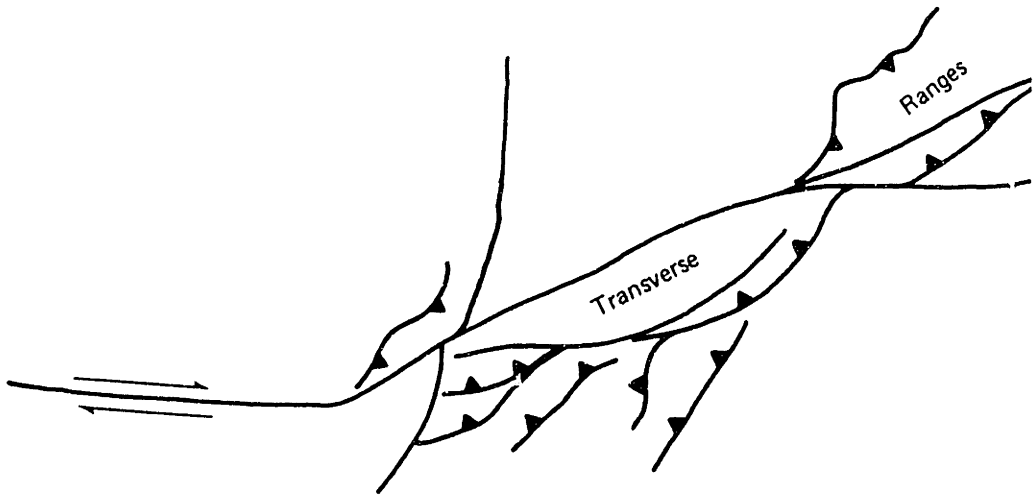
Hope fault, New Zealand



San Jacinto fault, California

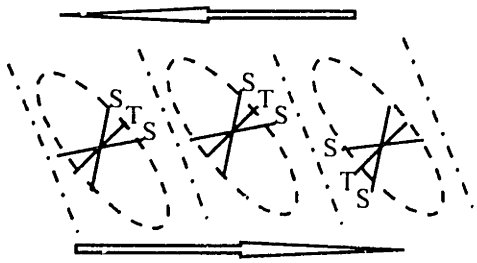
U: upthrown D: downthrown

b)



San Andreas fault, California

FIGURE 2.2.20 Examples of secondary structures between en-echelon segments of strike-slip faults [from Suppe 1985]: a) normal faulting and pull-apart basins in the extensional overlap zones of the Hope fault, New Zealand, and the San Jacinto fault, California; b) large scale thrust faulting and compressional uplift due to a bend in the San Andreas fault, California.



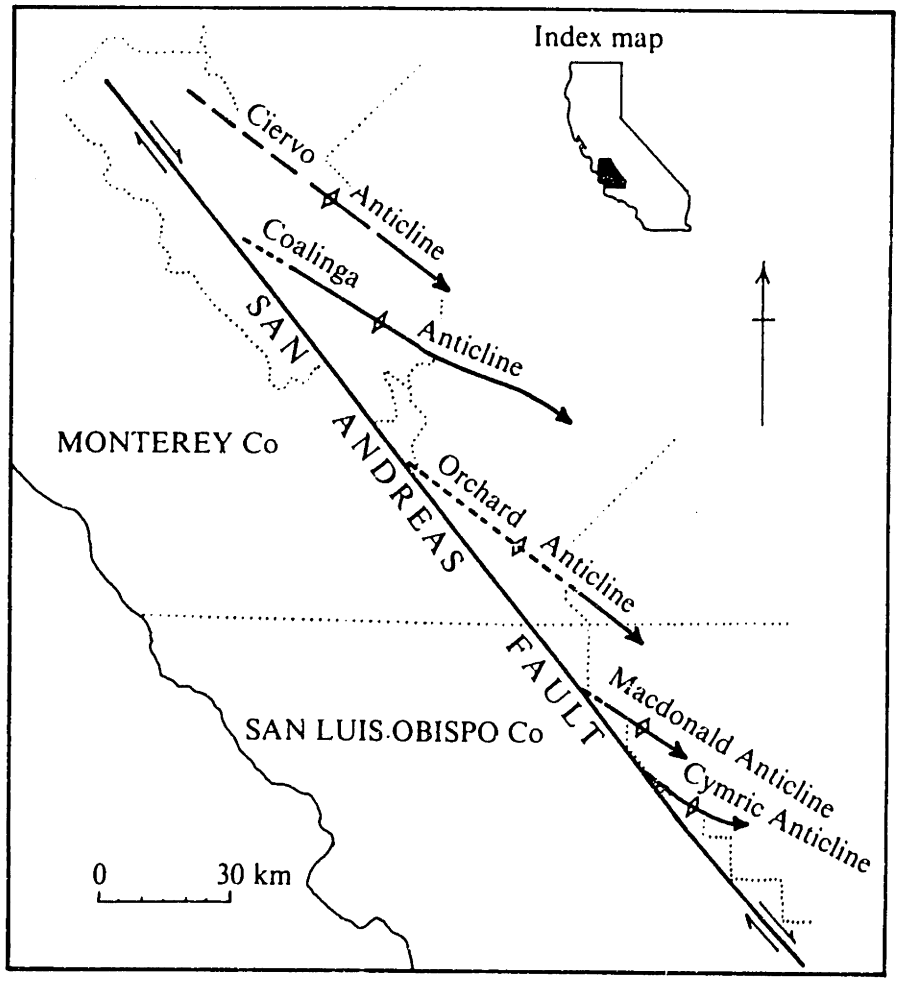
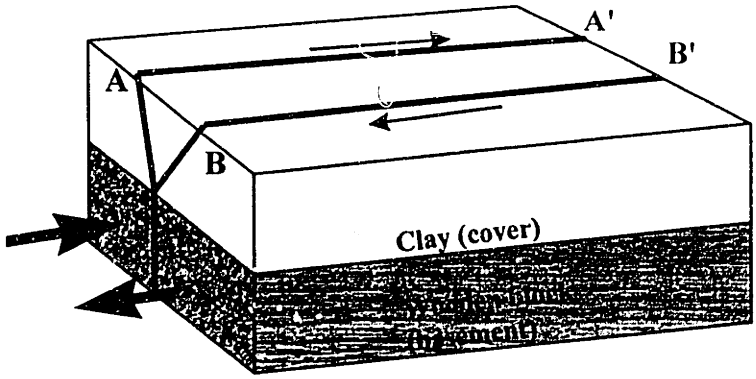


FIGURE 2.2.21 En-echelon folds in the cover rocks due to compression induced by shear along the San Andreas fault, California [from Price & Cosgrove 1990]. Inset: possible fracture patterns in en-echelon folds [after Einstein & Dershowitz 1990].

a)



b)

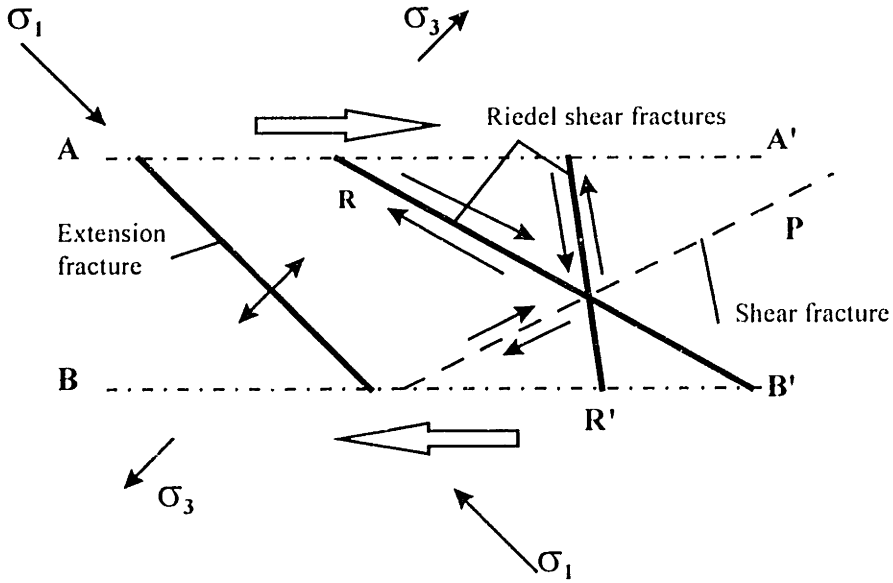


FIGURE 2.2.22 Summary of laboratory experiments on simulation of cover-rock fracturing as the result of strike-slip movement along fault in the basement [after Price & Cosgrove 1990]: a) experimental setup by Riedel (1929); b) fracture patterns induced in the shear zone in the cover as a result of shearing in the basement.

TYPE OF TEST	FRACTURE SETS	REFERENCE
Direct shear of the support	*In wet specimens (low tensile strength)* • Tension fractures at 45° to the applied shear direction • Shear fractures at 10-15° to the shear direction (R in Figure 2.2.22) *In dry specimens (low shear strength)* • Only the shear fractures (set R)	Riedel (1929)
Simple shear	*In wet specimens* • Tension fractures at 45° to the simple shear direction *In dry specimens* • Shear fractures at 10° to the shear direction (set R in Figure 2.2.22) • Shear fractures at 70° to the shear direction (set R') • Additional set of shear fractures (P)	Cloos (1955) Tchalenko (1968)

TABLE 2.2.2 Summary of extensional and shear fracture patterns observed in laboratory shear experiments [after Einstein & Dershowitz 1990 and Price & Cosgrove 1990].

Strike-slip fault zones in igneous rock

Strike-slip shear may occur along the surfaces of pre-existing fractures. For example, Segall & Pollard (1983b) describe outcrops in the Sierra Nevada where nucleation of strike-slip faults has occurred on earlier formed joints in granite (**Figure 2.2.23**). The strike-slip fault system in Figure 2.2.23 consists of a single, prominent set of subparallel fractures that strike uniformly N50°-70°E and dip steeply to the south. Some of the fractures in the outcrop in Figure 2.2.23 are joints (hairline fractures, 1 mm or less wide, containing undeformed fillings), and others are small faults (1 mm to more than 1 cm wide, filled with crushed material). Field observations suggest that the faults have developed through localization of shear deformation (left-lateral slip) on the surfaces of earlier joints. The fractures are from a few meters to several tens of meters long. Individual fractures are composed of en-echelon segments, tens of centimeters to several meters long, parallel to the overall fracture trend. Local spacing varies from centimeters to many meters. The depths of at least some fractures are comparable to their outcrop lengths. Most of the fractures are planar; also, there is evidence that the curved ones initially formed as planar surfaces and were later warped.

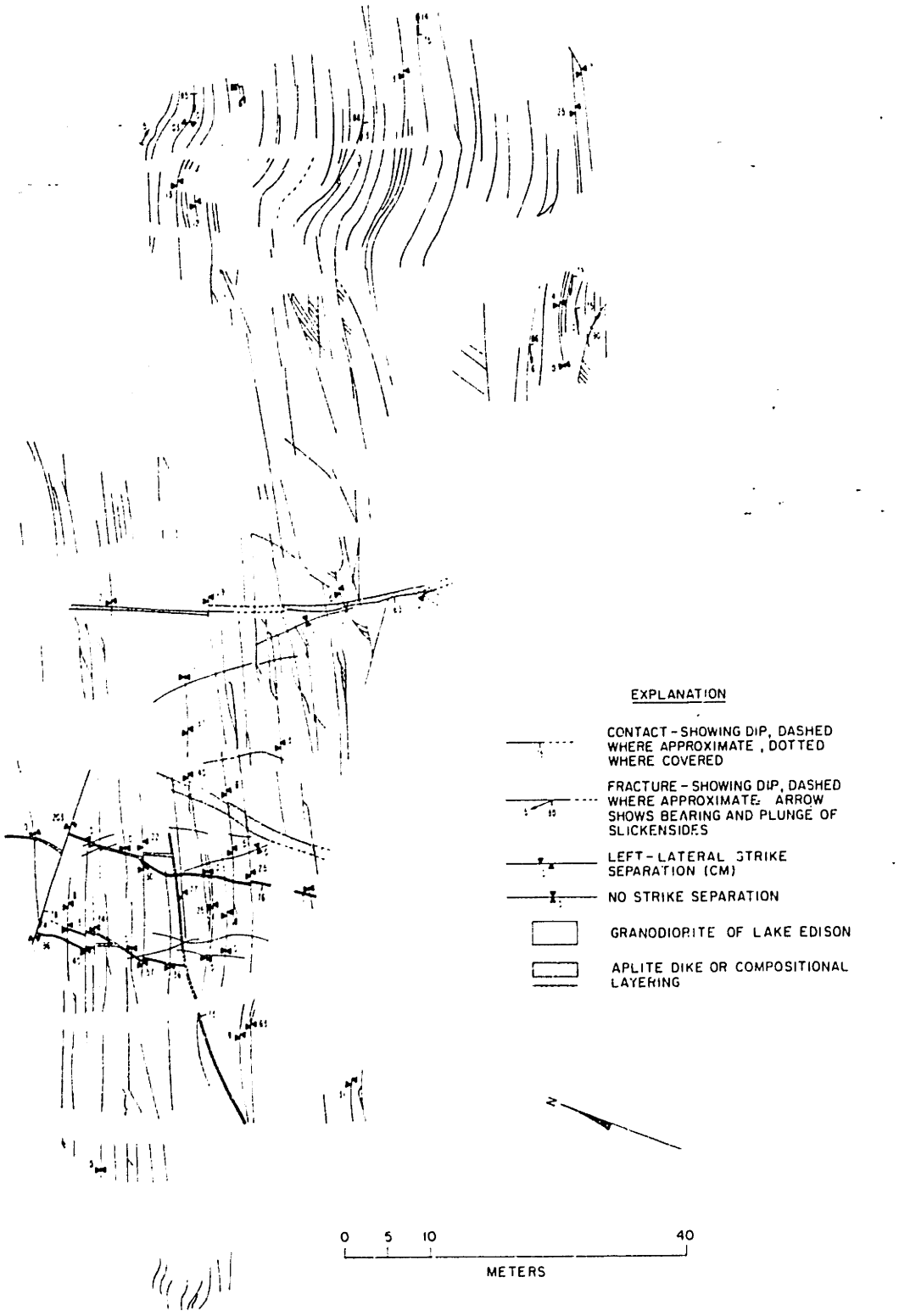


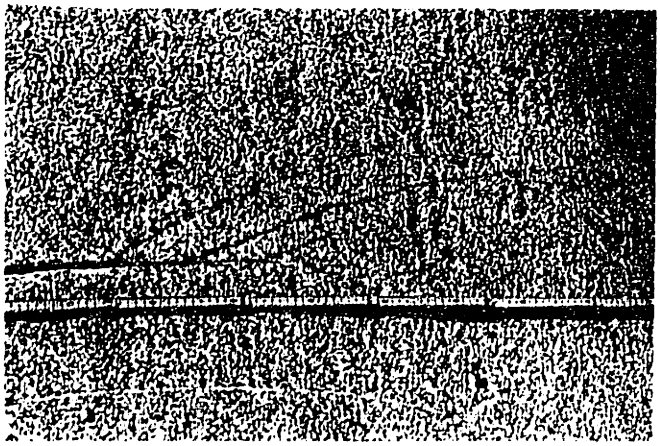
FIGURE 2.2.23 Strike-slip faults, created by localization of shear on earlier formed joints in granite in the Sierra Nevada [from Segall & Pollard 1983b].

Figure 2.2.24a depicts secondary fractures that have formed adjacent to the strike-slip faults of the system shown in Figure 2.2.23. Short dilational splay fractures exist near the ends of many strike-slip faults that have greater than a centimeter strike separations. The splay fractures dip steeply and strike N20°-60°E, i.e. about 15°-35° counterclockwise from the left-lateral faults. The lengths of splay fractures range from tens of centimeters to several meters and some of them terminate at adjacent left-lateral faults. Wedge-shaped cavities exist at the ends of individual faults, whereas rhomb-shaped cavities have developed between pairs of strike-slip fault segments. Figure 2.2.24b schematically illustrates the formation of splay fractures during initial shearing along joint surfaces, a process that creates the strike-slip faults, and the transformation of splay fractures into wedge and rhomb-shape cavities due to further shear along the strike-slip faults.

Figure 2.2.25 schematically represents the formation of simple and compound fault zones in crystalline rocks through linking of en-echelon strike-slip faults end-to-end by splay fractures and side-to-side by oblique internal fractures. Figure 2.2.25a represents the first step of deformation: a system of steeply-dipping subparallel joints. In the second stage of deformation (Figure 2.2.25b) some joints slip and become left-lateral faults; also, splay fractures, striking in the direction of the maximum compressive stress, form and may link some originally unconnected left-stepping echelon faults. In the third phase of deformation (Figure 2.2.25c), larger left-lateral displacements are accommodated as abundant oblique fractures link strike-slip faults side-to-side to form simple fault zones [Segall & Pollard 1983b; Martel et al. 1988; Martel & Pollard 1989]. The internal oblique fractures are dilatant fractures parallel to the local direction of the maximum principal stress. As illustrated in the figure, the ends of parallel en-echelon faults and the ends of fault segments with different strikes are linked by steps and bends, respectively. At the latest stage of deformation (Figure 2.2.25d), long oblique fractures connect some of the small strike-slip faults and simple fault zones, thus creating compound fault zones [Martel 1990].

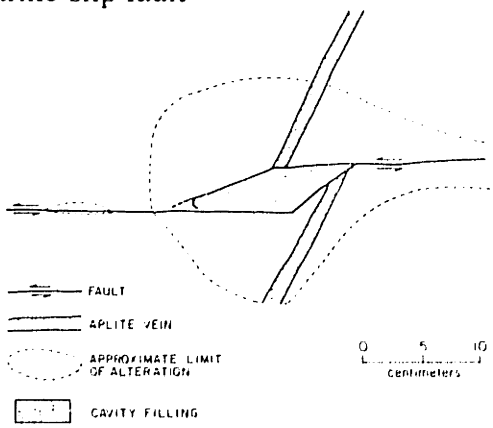
Figure 2.2.26 depicts typical strike-slip zones, observed by Segall & Pollard (1983b) in the Sierra Nevada granite. When the shear displacement along the boundary strike-slip faults has been relatively small, closely spaced, oblique, dilatant fractures divide the rock in the shear zone into rhombohedral blocks (Figure 2.2.26a). When greater displacements have occurred along the boundary faults, the pattern of internal fracturing is usually more complex and includes numerous secondary fractures that divide the rock into angular blocks (Figure 2.2.24 b). In the strike-slip fault zones in Figure 2.2.26, the most prominent internal fractures, either straight or very gently curved, are typically spaced 1-20 cm apart. Most of the net slip through the simple strike-slip zones is accommodated by the bounding primary strike-slip faults, the maximum strike-slip displacement observed being about 10 m. The internal fractures strike at acute counterclockwise angles from the boundary faults, and commonly intersect but do not cross or offset them.

a)



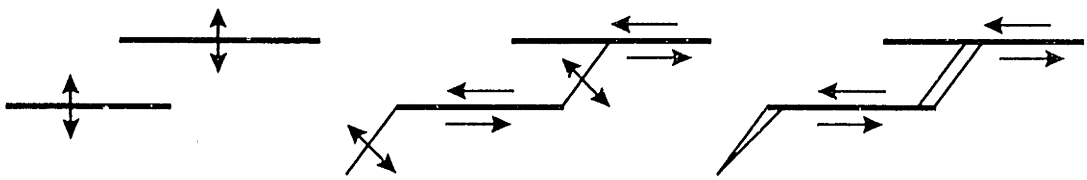
Splay fractures at the end of a strike-slip fault

Wedge-shaped cavity



Rhomb-shaped cavity

b)



Initial jointing

Nucleation of faults and opening of splay fractures

Further shearing and development of cavities

FIGURE 2.2.24 Secondary fractures adjacent to the strike-slip faults in the system shown in Figure 2.2.23 [after Segall & Pollard 1983b]: a) splay fractures at the ends of fault segments, and wedge and rhomb-shaped cavities; b) schematic representation of the generation of the fractures shown in (a).

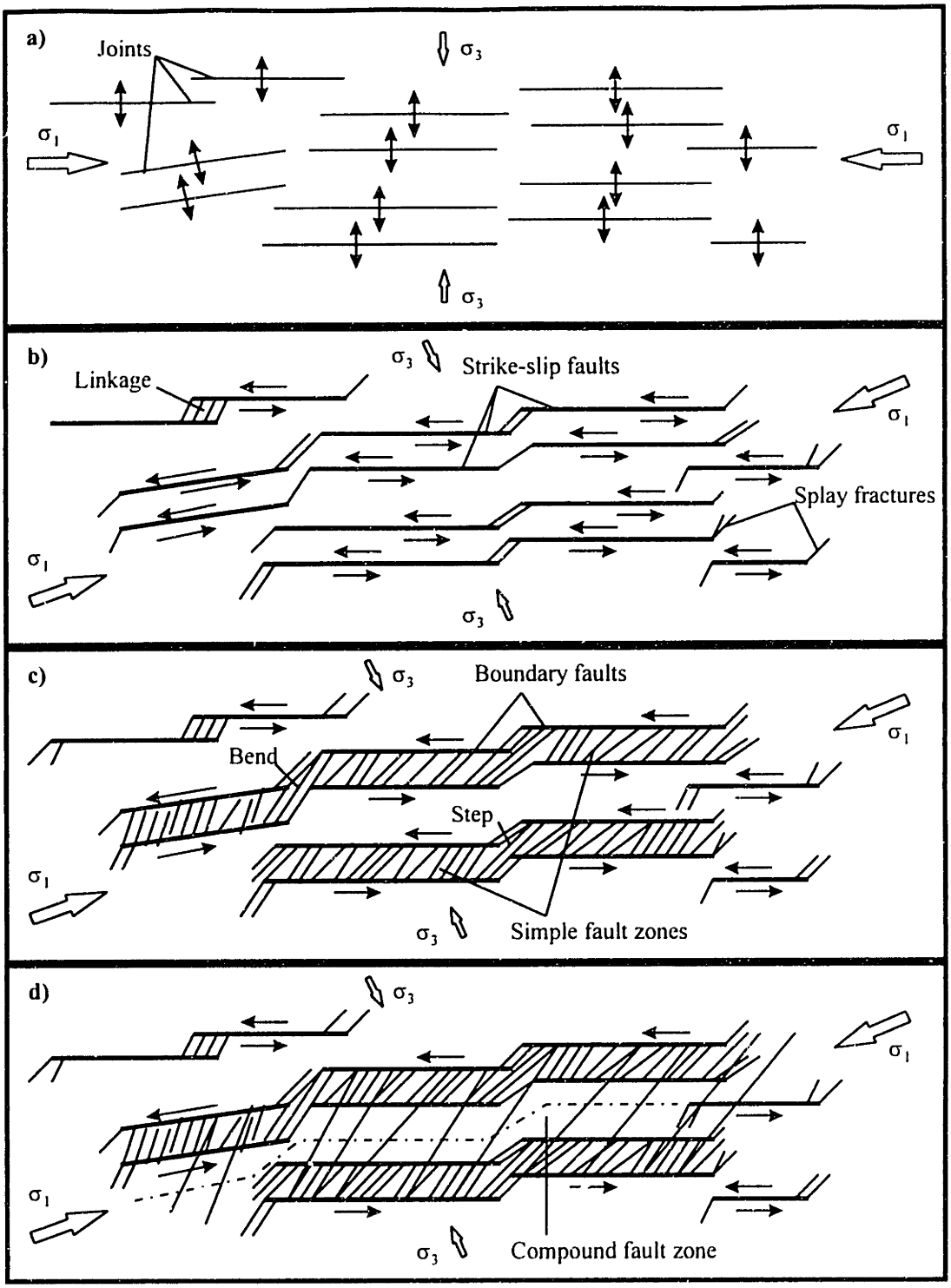
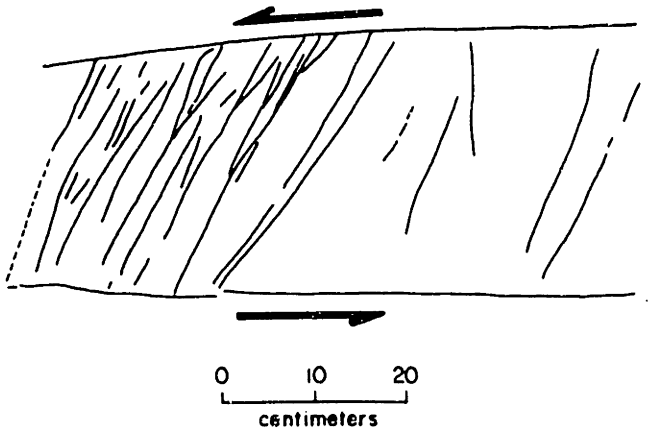


Figure 2.2.25 Formation of strike-slip fault zones [after Martel & Pollard 1989 and Martel 1990]: a) opening of joints; b) development of small left-lateral strike-slip faults, and linking of some faults end-to-end by splay cracks; c) development of simple fault zones through extensive fracturing between closely spaced faults; d) formation of compound strike-slip fault zones through linking of faults and simple strike-slip zones by long oblique internal fractures.

a)



b)

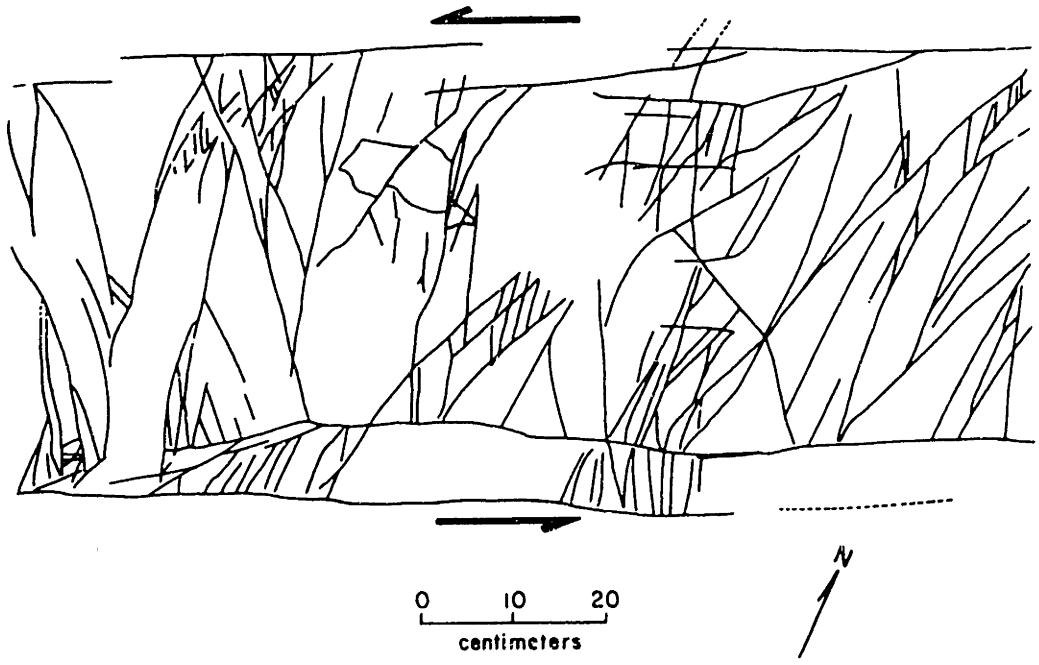


Figure 2.2.26 Internal structure of fault zones [from Segall & Polla d 1983b]: a) closely spaced secondary fractures initially divide the rock within a fault zone into rhombohedral blocks; b) additional fracturing and slip on previously formed fractures leads to a more complicated fracture pattern inside the fault zone.

Martel & Pollard (1989) describe simple strike-slip fault zones in granitic rocks in the Mount Abbot quadrangle in California. At this location, strike-slip fault segments, a few tens of meters long, are joined by steps or bends into simple fault zones that are 0.5-3 m thick and 100 m to 1 km long. Fractures, oblique to the strike of the primary strike-slip faults and perpendicular to the local maximum

tensile stresses, are abundant inside the fault zones. The rock outside fault zones is highly fractured only at segment ends; away from the ends, external fractures are relatively scarce. In the same rocks, Martel (1990) describes compound strike-slip zones that have incorporated many pre-existing faults and simple fault zones. **Figure 2.2.27** illustrates an outcrop of a compound strike-slip zone in granite in the Mount Abbot quadrangle in California.

Fracture systems, similar to those described in the last three paragraphs, have been documented in high-grade gneiss, granite and quartzite. Martel et al. (1988) suggest that simple and compound strike-slip fault zones may be common in crystalline rocks in general.

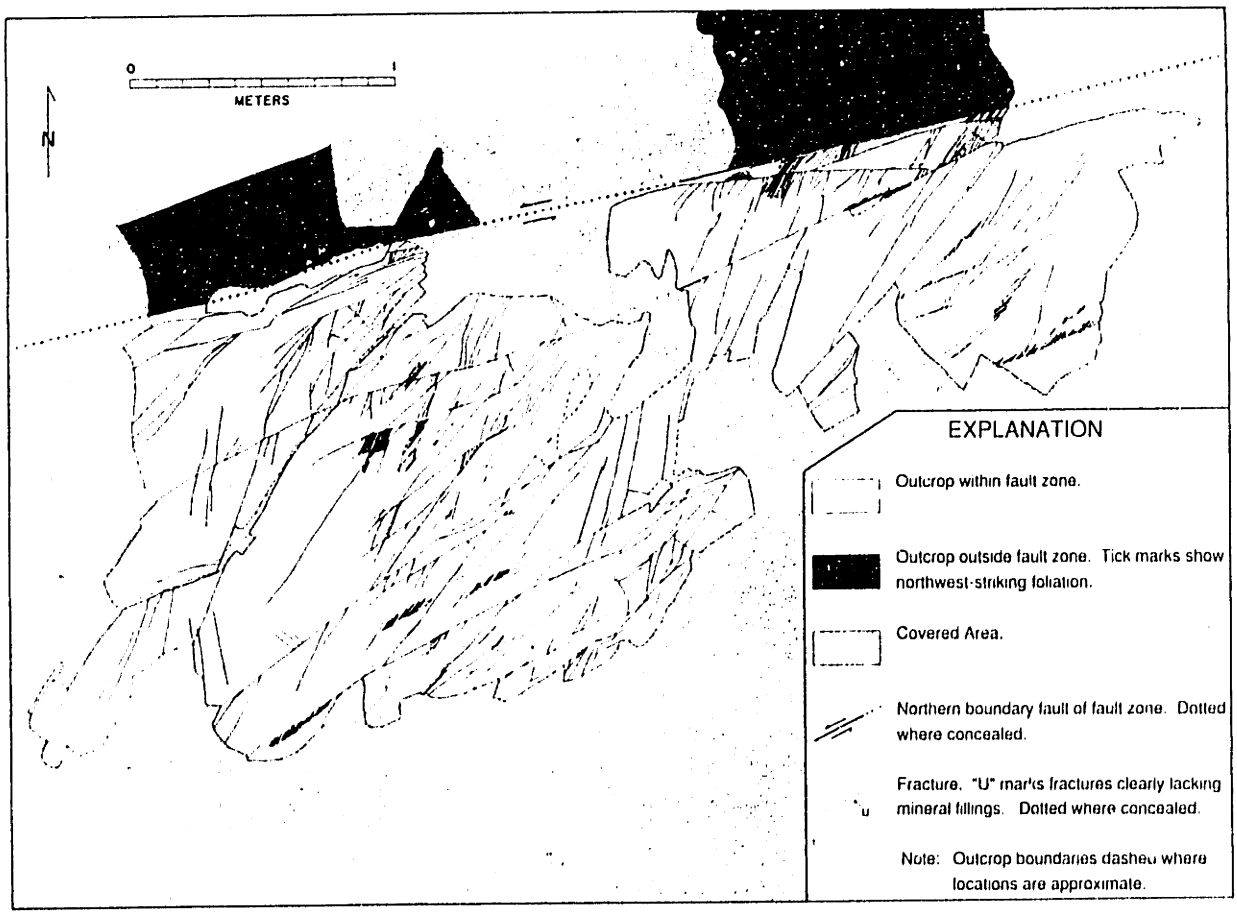


Figure 2.2.27 Map of fractures in a small outcrop of a compound fault zone at the Waterfall site in the Mount Abbot quadrangle, California [from Martel 1990].

Strike-slip band faults in sedimentary rocks

Deformation bands and band faults constitute a specific type of faulting (strike-slip, normal or thrust) that occurs only in sedimentary rocks. Deformation bands are surfaces across which relative displacements of a few millimeters to a few centimeters are distributed although there is no discrete surface of discontinuity. Aydin (1978) and Aydin & Johnson (1978) describe faults in the form of deformation bands in the Entrada and Navajo sandstones in the San Rafael Desert, Utah. The observed deformation bands are about one millimeter wide and tens of hundreds of meters long. Aydin (1978) defines two zones within a deformation band in sandstone (**Figure 2.2.28**). The outer zone consists of unfractured quartz and fractured feldspar grains that are more closely packed than the undisturbed sandstone. The inner zone, about 0.5 mm thick, is characterized by fractured and crushed grains, and by quite large consolidation that leads to negligible porosity.

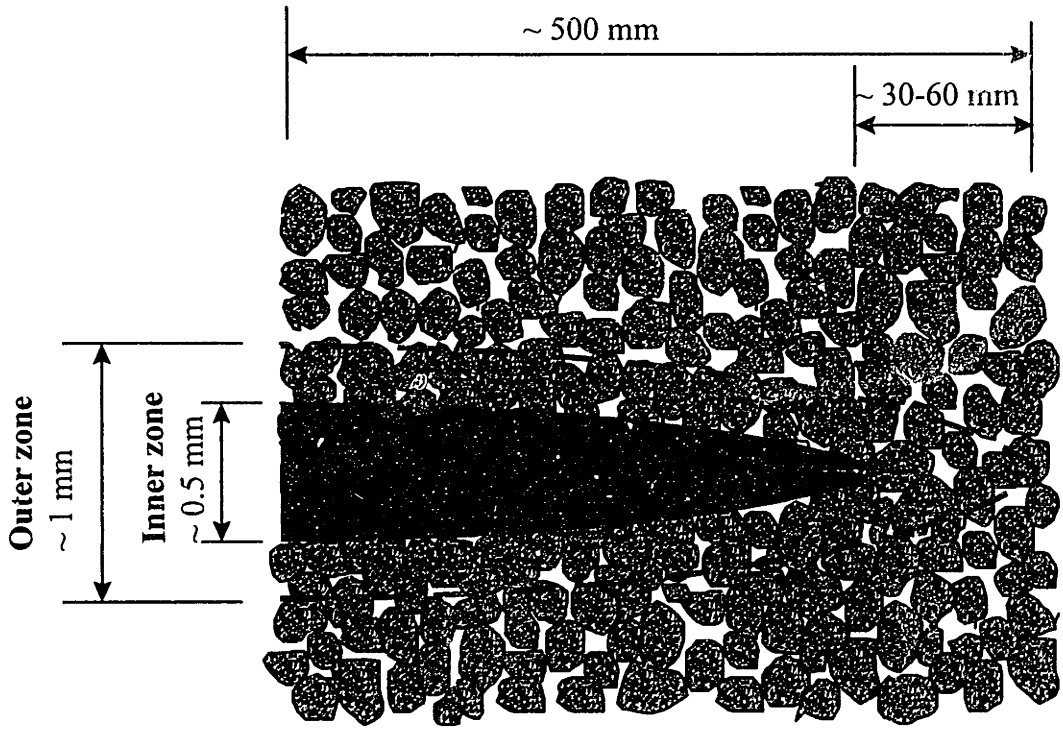


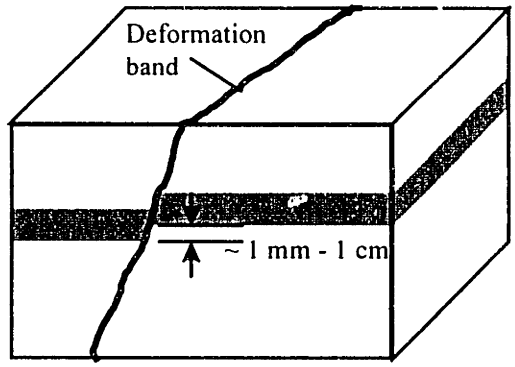
FIGURE 2.2.28 Idealized deformation band in porous sandstone: outer zone of deformed matrix and pores, and inner zone of fractured material and negligible porosity [after Aydin 1978].

The deformation bands in sedimentary rocks play an important role in the generation of larger faults with offsets of several tens of meters. The process is schematically illustrated in **Figure 2.2.29**. Two or more deformation bands adjacent to one another, having the same average strike and dip, form a zone of deformation bands (also called a band fault). Band faults are made up of many, closely spaced, individual deformation bands that are parallel or subparallel, commonly inosculating, but rarely crossing each other (Figure 2.2.29b). Generally, the parent rock retains its original texture between neighboring band faults. Total displacements across the zones of deformation bands depend on the number of individual bands; for example, a zone of 100 bands may accommodate a total displacement on the order of 25-30 cm [Aydin 1978]. Band faults may be accompanied by an actual slip surface: a discrete surface of discontinuity along which significant localized displacement takes place (Figure 2.2.29c). The slip surface develops in the region of highest concentration of deformation bands which is generally in the marginal areas of the zone of deformation bands. Slip surfaces may accommodate displacements on the order of several decimeters to several meters, i.e. much larger than those that are distributed across individual deformation bands or zones of deformation bands [Aydin & Johnson 1978].

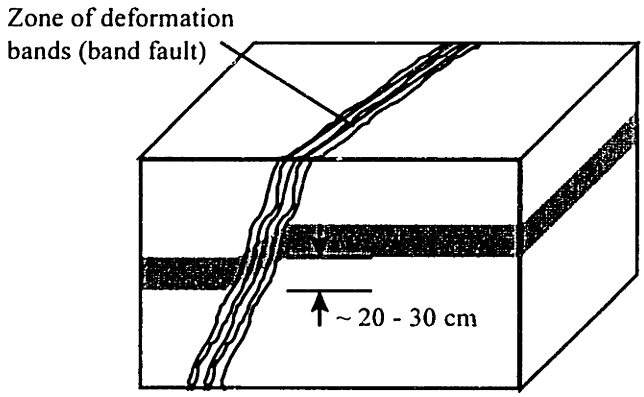
Figure 2.2.30 shows an outcrop in sandstone, observed by Cruikshank et al. (1991) in Entrada sandstone in the Arches National park, Utah. The outcropped fracture system consists of two conjugate sets of strike-slip band faults: a set of left-lateral faults strikes approximately N30°E, and a set of right-lateral band faults strikes about N60°E. The individual band faults, approximated in Figure 2.2.30 as continuous lines across distances of over 1 km, in reality are composed of segments of zones of deformation bands ranging in length from about 0.5 to 20 m. Fault segments are nearly collinear, but step short distances (from a few millimeters to a decimeter) to the left or right. Left-lateral faults have mostly right steps, and right-lateral faults have predominantly left steps. Adjacent fault segments may connect into a continuous fault via so called duplex structures: secondary band faults that strike at low angles to the general fault trend, and form patterns with various shapes, degrees of complexity and number of connectors (**Figure 2.2.31**). **Figure 2.2.32** shows maps of individual band fault traces.

Besides deformation bands and band faults, other more common types of strike-slip faults may also occur in sedimentary rocks. For example, faults may develop by a strike-slip shear along pre-existing joints, similar to the previously described process in igneous rocks. Cruikshank et al. (1991) point out that the structures connecting en-echelon fault segments are quite different depending on whether the faults are band faults or faulted joints. **Figure 2.2.33** schematically illustrates that difference: in right-lateral shear, the transfer zones between segments of faulted joints are bridged by tension cracks if the step is right, or remain unbridged if the step is left (Figure 2.2.33a); whereas transfer zones between segments of band faults are bridged by short ramp faults irrespective of whether the segments step to the left or to the right (Figure 2.2.33b).

a)



b)



c)

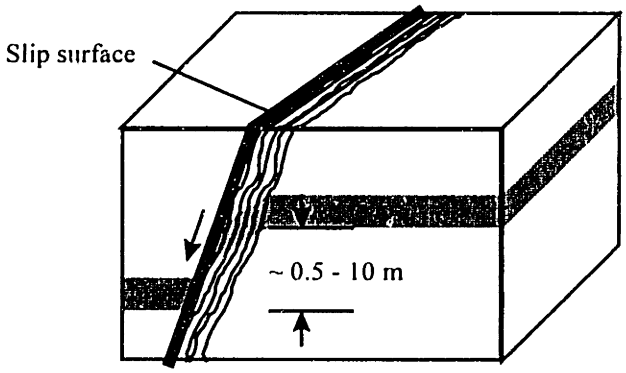


FIGURE 2.2.29 Diagrams, showing sequential development from a single band to a slip surface in sandstone [after Aydin & Johnson 1978]: a) a single deformation band; b) a zone of deformation bands; c) slip surface developed on the left-edge of the zone.

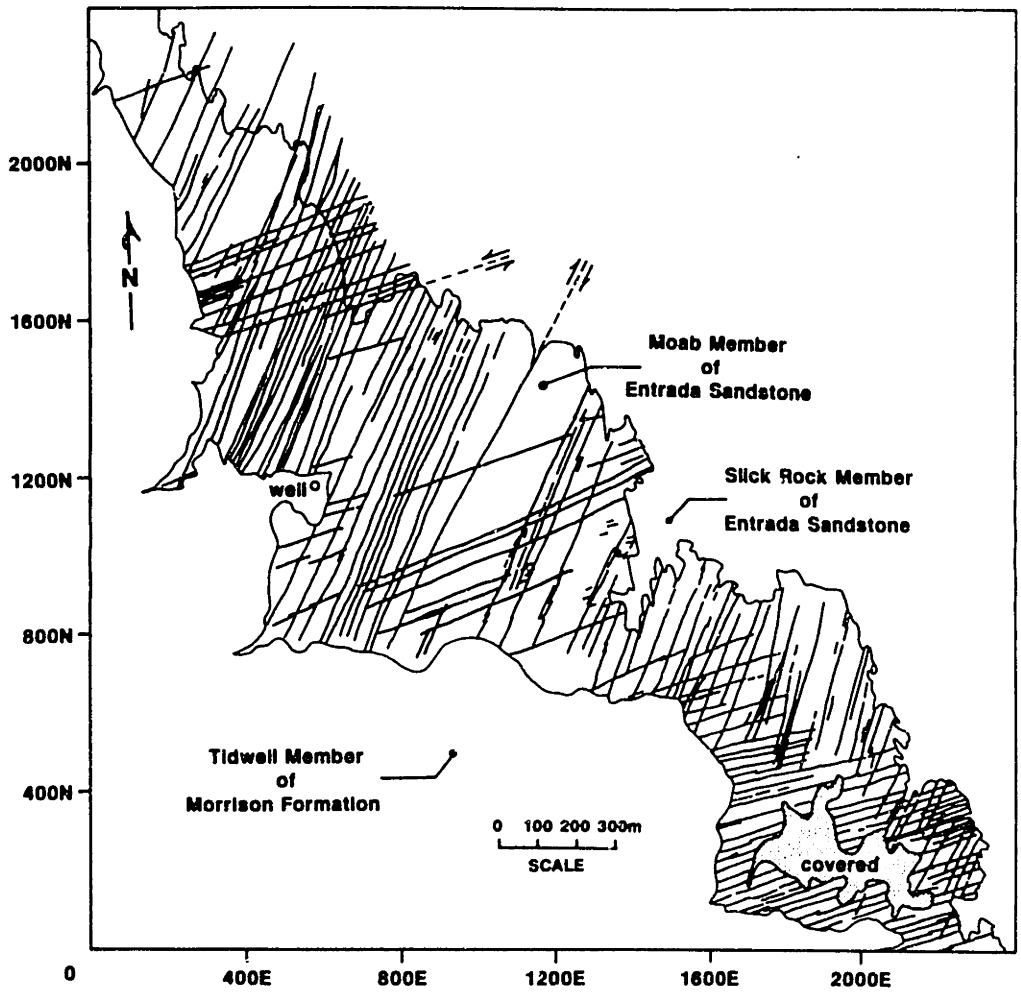


Figure 2.2.30 Traces of band faults in the Garden Area, Arches National Park, Utah [from Cruikshank et al. 1991].

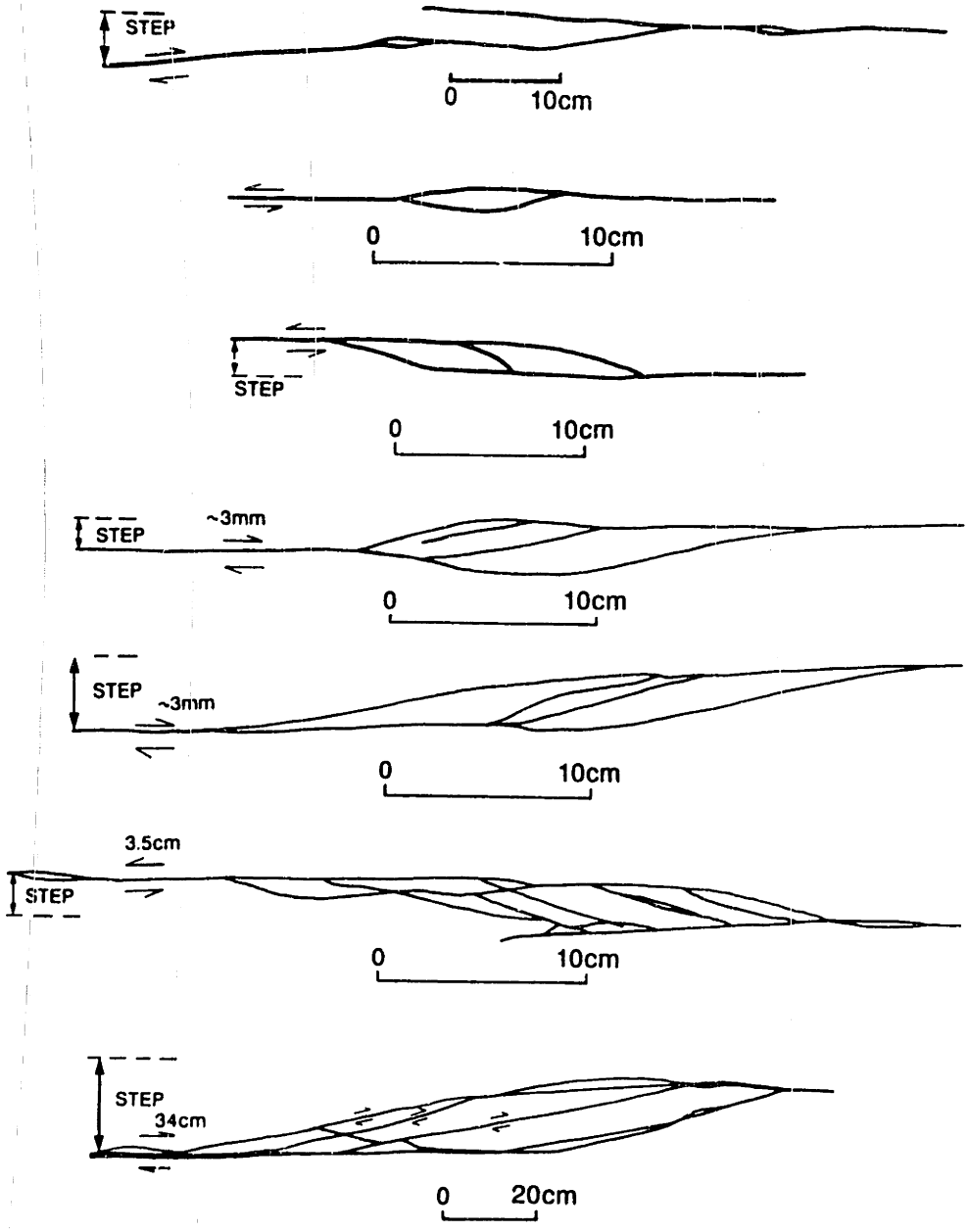


FIGURE 2.2.31 Maps of duplex-like structures in the Entrada Sandstone [from Cruikshank et al. 1991].

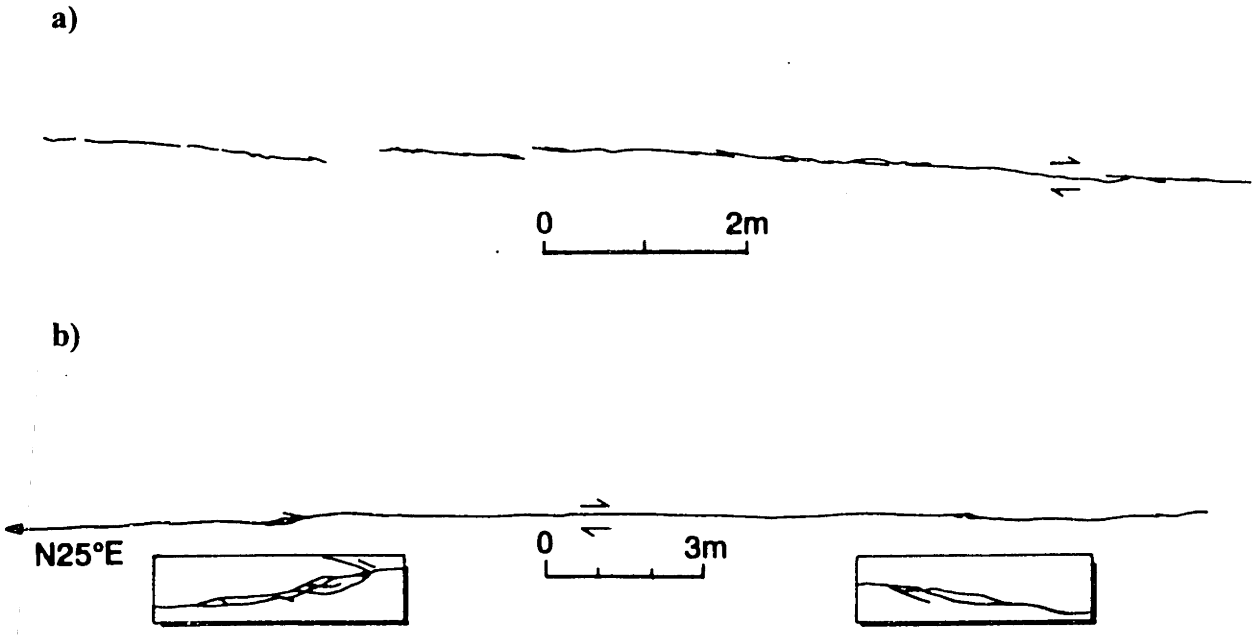


FIGURE 2.2.32 Traces of band faults showing their segmented nature [from Cruikshank et al. 1991]: a) some segments are still isolated, although the segments are themselves composed of smaller segments; b) a continuous band fault in which the segments are joined by ramp-like duplex structures.

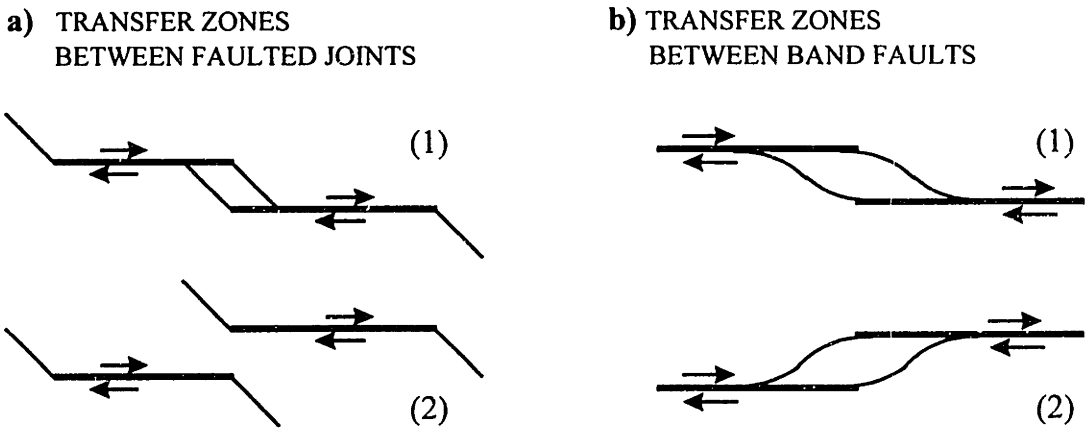


FIGURE 2.2.33 Difference between the structures in the overlap zones of faulted joints and band faults, subjected to right-lateral shear [after Cruikshank et al. 1991]: a) transfer zones between segments of faulted joints, bridged by tension cracks if the step is right (1), or unbridged if the step is left (2); b) transfer zones between segments of band faults, bridged by short ramps whether the segment steps to the right (1) or left (2).

2.2.3 Thrust faults, reverse faults and overthrusts, and associated fracture systems

Thrust faults are inclined planes of shear failure with a marked upward dip-slip displacement of the hanging wall relative to the footwall, and zero or very small strike-slip component of displacement. According to the theory by Anderson (1951), thrust faults form when the maximum and intermediate principal stresses, σ_1 and σ_2, are horizontal, and the minimum principal stress, σ_3 , is the vertical stress of gravity (Figure 2.2.1c). Thrust faulting is commonly associated with folding which occurs under the same principal stresses [De Sitter 1956; Billings 1972].

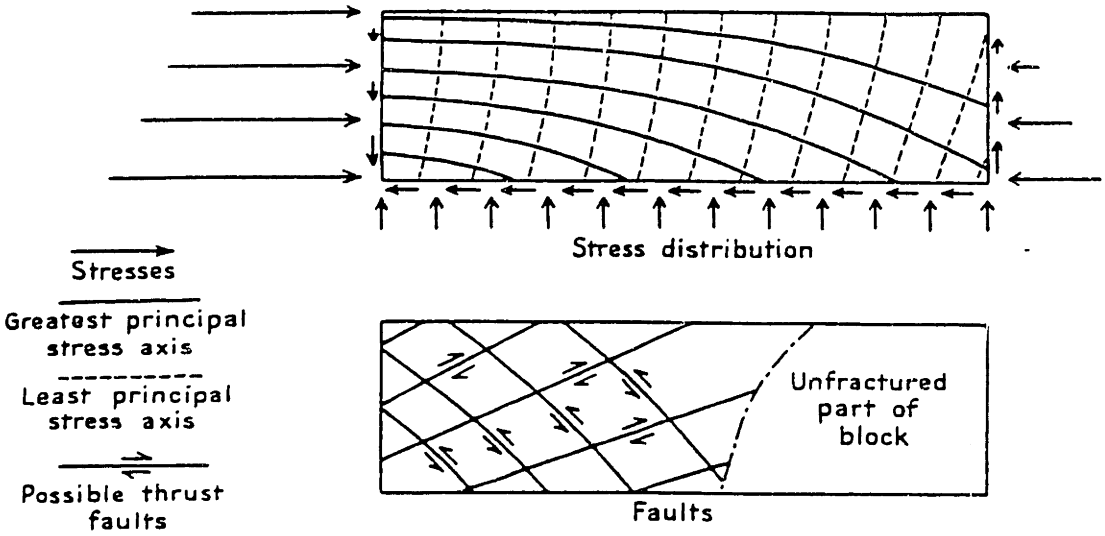
Compressive (convergent) tectonic plate boundaries in the continental and oceanic crust constitute the primary geologic setting for development of thrust faults [Suppe 1985]. This section deals primarily with the characteristic fault geometry in continental fold-and-thrust belts which are the best known and most important setting of thrust faulting along compressive plate boundaries. Thrust faulting characteristic of secondary geologic settings is reviewed in other sections and includes trust faults associated with strike-slip fault zones (Section 2.2.2), folds (Section 2.1), igneous or sedimentary intrusion (Section 2.5), landslides, etc.

Thrust and reverse faults

Thrust faults usually dip less than 45°, making an angle of 45°-ϕ/2 with the horizontal direction of σ_1 according to the Coulomb criterion (ϕ being the friction angle of the material which is normally less than 30°). Since in reality the maximum principal stress may deviate from the horizontal direction, thrust faults having dips higher than 45° (called reverse faults or upthrusts) also exist.

Hafner (1951) analyzed several cases of thrust fault plane orientations under additional stresses imposed on the so called standard state (i.e. lithostatic pressure which increases with depth at a constant rate, and is uniform on the surface of an imaginary sphere in the crust). **Figure 2.2.34**a and b illustrates the distribution of the stresses, given by trajectories of σ_1 and σ_3 (upper diagram), and the corresponding orientations of potential thrust faults (lower diagram) in two typical cases. In both of them σ_2 is assumed horizontal. In the first case (Figure 2.2.34a) the additional stress is horizontal pressure, linearly increasing downward, and linearly decreasing from left to right. At the left end of the fractured zone, where σ_1 is horizontal, one set of thrust fault planes dips 30° to the right, and another set dips 30° to the left. At the right end of the fractured zone, where σ_1 plunges to the right, one set of faults dips about 40° to the right and the other set dips 20° to the left. In the second case (Figure 2.2.34b) the supplementary horizontal stress decreases exponentially from left to right. The trajectories of the σ_1 axes plunge to the right, and the fractures are inclined 30° to them, forming within the fractured zone one set of high-angle reverse faults and another set of low-angle thrusts.

a)



b)

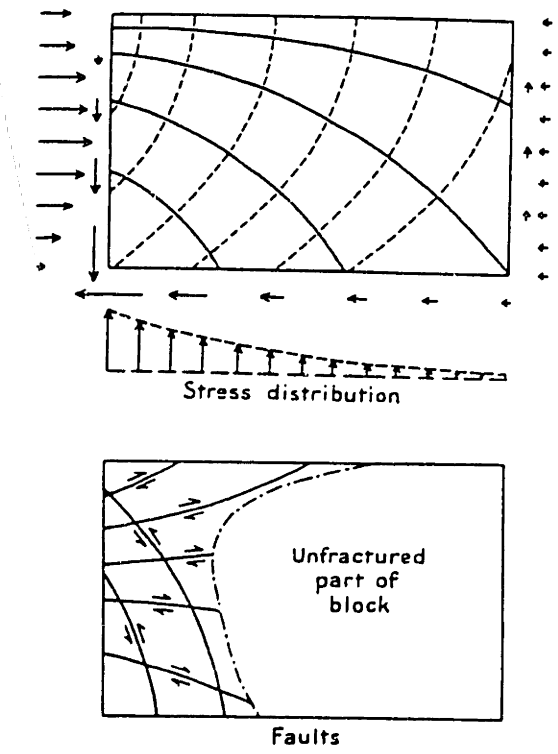
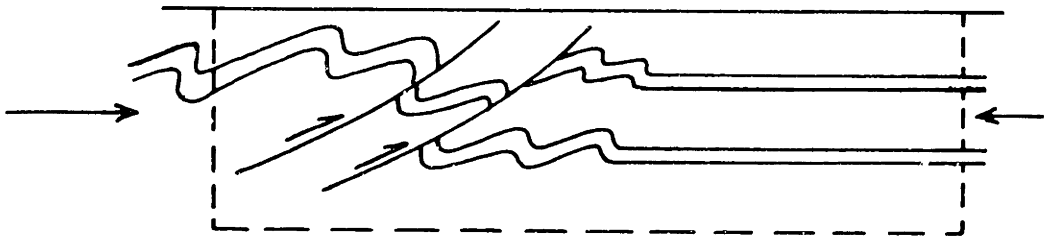


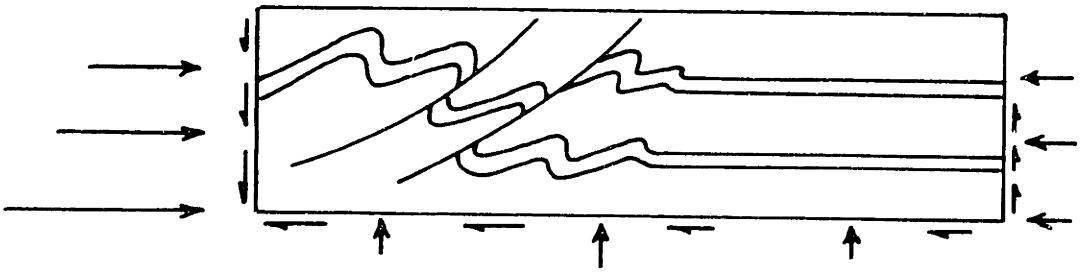
FIGURE 2.2.34 Stress distribution (upper diagram) and thrust fault orientations (lower diagram) under a combination of lithostatic pressure and superimposed horizontal stress [from Hafner 1951]: a) the supplementary stress linearly increases downward and decreases from left to right; b) the supplementary stress exponentially decreases from left to right.

Price (1966) and Price & Cosgrove (1990) point out that the stress system illustrated in Figure 2.2.34a is very similar to the conditions in continental thrust-and-fold belts. **Figure 2.2.35**a schematically illustrates the characteristic geometry of a fold-and-thrust belt featuring a zone of deformation (folded and cut by a set of concave upward thrust faults) that grades laterally into an undeformed foreland. Figure 2.2.35b depicts the stress system, as derived by Hubbert (1951), that can produce the deformation in Figure 2.2.35a (compare to the stress system in Figure 2.2.34a). Figure 2.2.35c diagrammatically represents sand box experiments in which boundary conditions of the type shown in Figure 2.2.35b resulted in the typical deformation of a thrust-and-fold belt.

a)



b)



c)

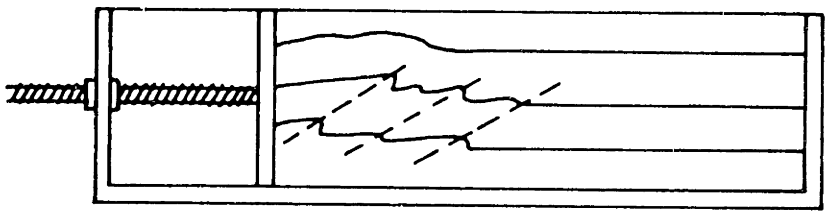
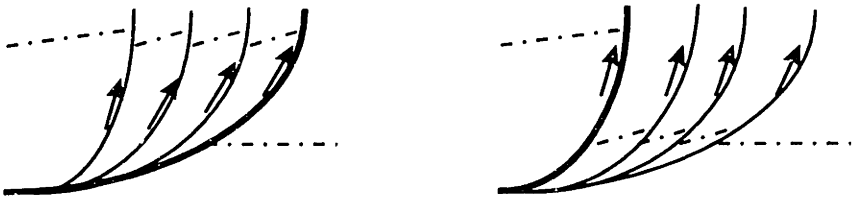


FIGURE 2.2.35 Stress distribution and deformation in a fold-and-trust belt [after Hubbert 1951 and Price 1966]: a) diagrammatic section through a fold belt showing concave upwards thrust faults and unfolded foreland; b) stress system that could give rise to the structures illustrated in (a); c) schematic representation of sand box experiments with boundary conditions similar to those shown in (b).

Figure 2.2.36 schematically illustrates two structures that are commonly associated with thrust faulting [Ramsay & Huber 1987; Price & Cosgrove 1990; Suppe 1985]. Imbricate structures (Figure 2.2.36a) develop when a series of reversed faults branches upwards from a floor thrust into a splay fault fan complex. An imbricate fan can be either (1) leading, or (2) trailing if the maximum displacement occurs along the frontal subsidiary fault or along the most internal one, respectively. A duplex structure (Figure 2.2.36b) develops when the reverse faults in an imbricate structure curve asymptotically towards a flat overlying thrust, thus enclosing the imbricate zone between a roof thrust and a floor thrust. The individual imbricate units (horses) in a duplex structure are completely enclosed by faults.

a)



b)

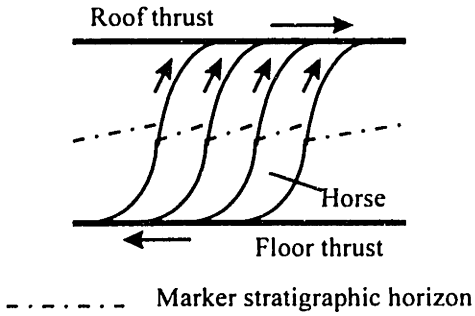


FIGURE 2.2.36 Typical thrust fault systems [after Ramsay & Huber 1987]: a) imbricate structures; b) a duplex structure.

In nature, the curved (listric) profiles of the thrust and reverse faults illustrated in Figures 2.2.34, 2.2.35 and 2.2.36 develop when the stress changes with depth but the material properties remain the same. However, thrust faults often cross cut layers of markedly different lithology. In such a case the fault dip changes abruptly and forms a structure that consists of flats and ramps (**Figure 2.2.37**a) [Ramsay & Huber 1987; Price & Cosgrove 1990]. Flats form near incompetent layers, and ramps develop where the thrust fault passes more steeply across competent beds. Figure 2.2.37b illustrates the position of the hanging wall after it has moved relative to the footwall along a flat-and-ramp thrust plane.

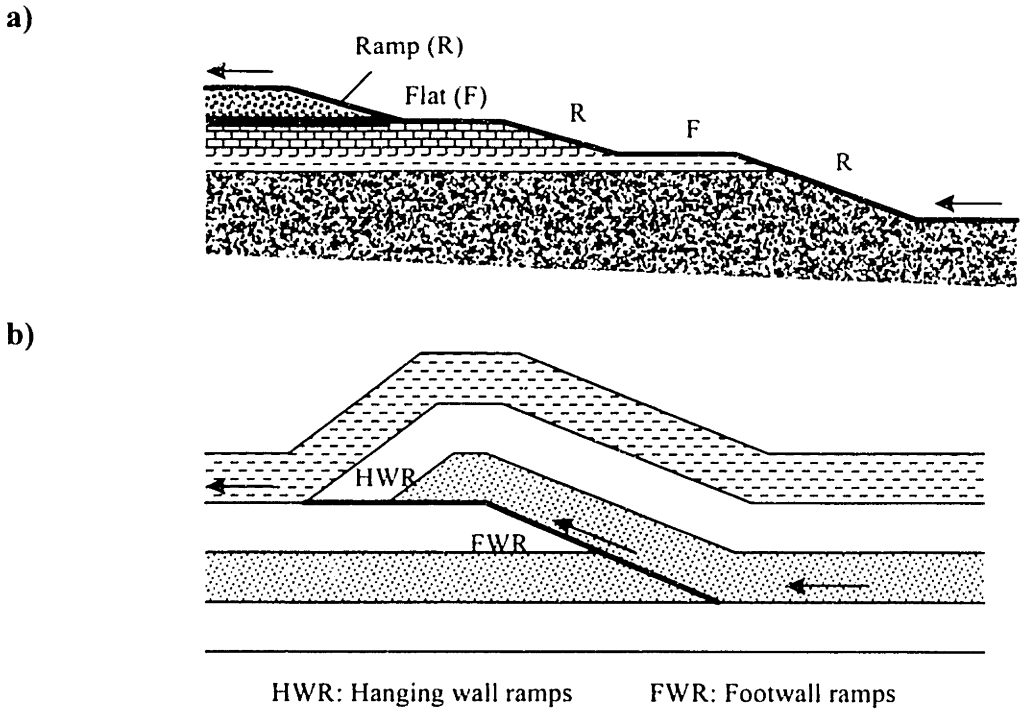


FIGURE 2.2.37 Geometry of a thrust fault cutting across layers of different lithology [after Price & Cosgrove 1990]: a) ramp-and-flat structure of the footwall; b) relationship between hanging wall and footwall ramps.

Figure 2.2.38 shows typical duplex structures associated with ramp-and-flat thrust faults [Boyer & Elliot 1972; Price & Cosgrove 1990; Ramsay & Huber 1987]. Figure 2.2.38a depicts the successive development of a duplex structure. Initially the base flat propagates under horizontal compression (from point P to point P_1 at a distance S_0), creating a new frontal thrust and an enclosed horse; this horse later is transported forward (from point P_1 to P_2 at a distance S_1), etc. Such a duplex structure in which the overall up-dip inclination of the imbricate splay faults and the beds enclosed between them is along the direction of movement is called a hinterland dipping complex. Its three-dimensional geometry is shown in Figure 2.2.38b. Figure 2.2.38c illustrates a stacked imbricate antiform which develops when the successive distances S_i of slip are about the same as the lengths of the enclosed horses. In such antiformal structures every successively activating horse provides an uplift to the previously formed and deactivated horses. Figure 2.2.38d illustrated a so called foreland dipping duplex, which forms when the slip along successive thrust faults is larger than the length of the enclosed horses. In this case the successive uplift of the roof thrust and enclosure of horses occur forward of the early formed horses. In this kind of duplex structure most thrust faults and beds enclosed between them have up-dip inclinations opposite to the direction of overall movement.

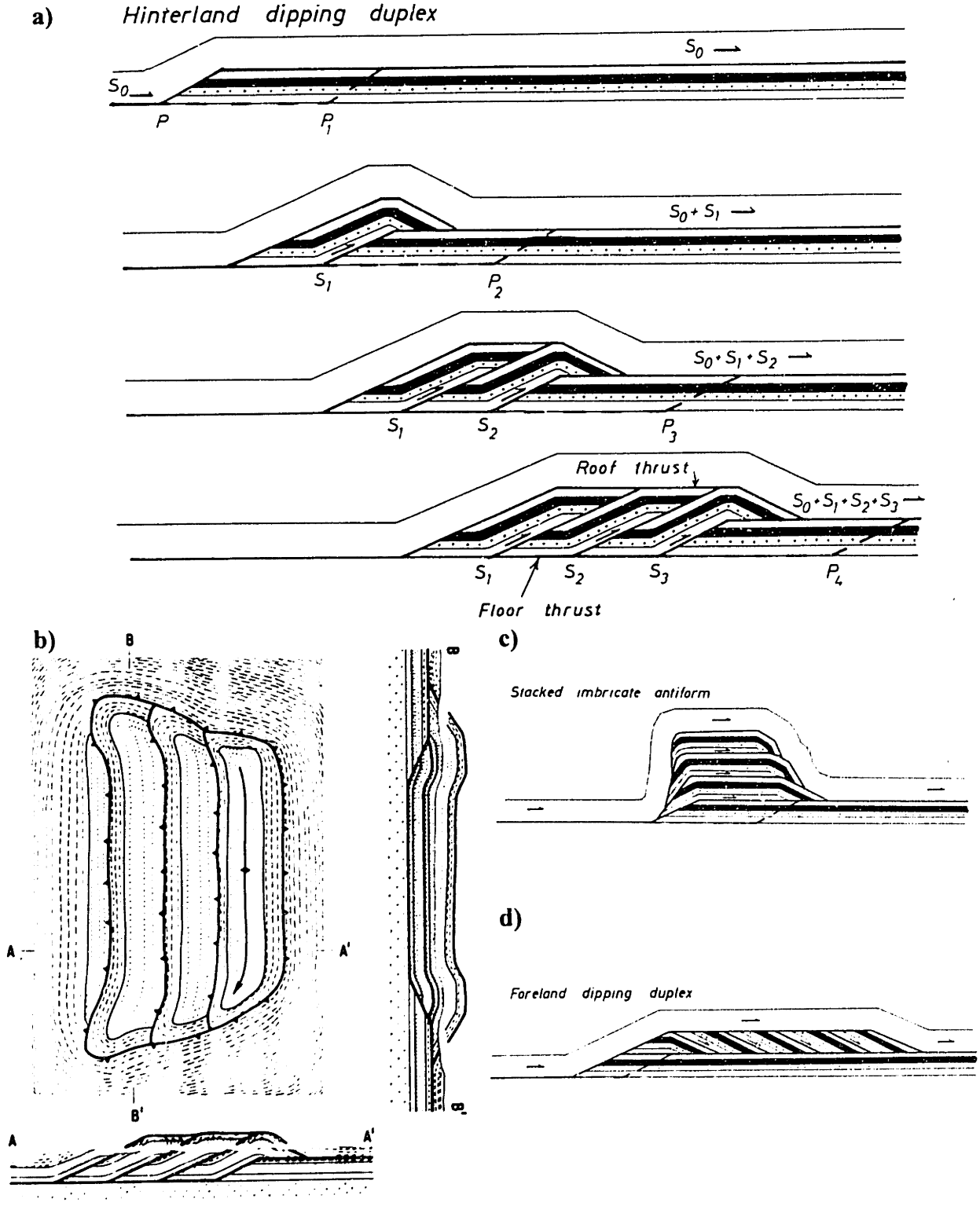


FIGURE 2.2.38 Geometry of duplex structures associated with ramp-and-flat thrust faults: a) development of a hinterland dippping duplex, and b) three-dimetional geometry of the duplex structure in (a) [from Boyer & Elliot 1982]; c) stacked imbricate antiform, and d) foreland dipping duplex [from Ramsay & Huber 1987].

Overthrusts and foreland fracturing

Overthrusts are low-angle faults along which the hanging wall rocks have been transported over significant distances (on the order of many miles), essentially at right angles to the regional trend of the fault [Billings 1972; Price & Cosgrove 1990]. When originated, the planes of overthrust faults are very flatly dipping, i.e. less than 5-10°. Models of the mechanics of overthrust formation, discussed in detail by Price & Cosgrove (1990), include push by an immense horizontal force, gravitational sliding (where the overthrust is essentially a slide, i.e. a low-angle normal fault grading laterally into a low-angle thrust fault), and gravitational spreading. Major overthrusts exist in the mountains at continental convergent boundaries, including the Alps, the Canadian and US Rocky Mountains, the Himalayas, the Appalachians.

In many regions there are many overthrusts that are arranged in the typical imbricate and duplex structures. The overthrusts enclose extensive, but thin, sheets of rock (called thrust nappes) that are transported at distances of many miles along the flat lying thrust palnes. **Figure 2.2.39** shows an example of an imbricate overthrust system in the Canadian Rocky Mountains. **Figure 2.2.40** illustrates (a) a schematic drawing and (b) the actual profile of the overthrust duplex structure of the Moine Thrust in Scotland.

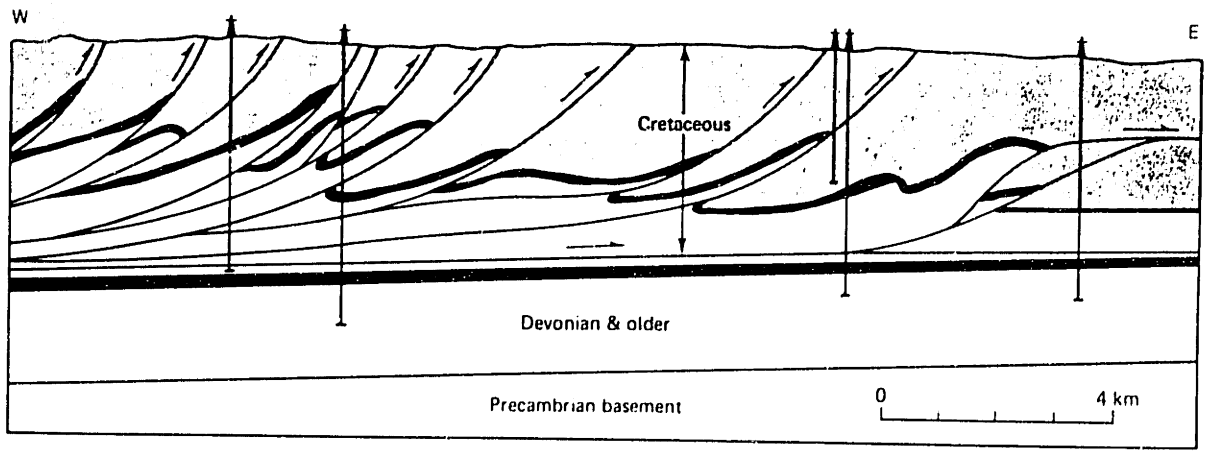
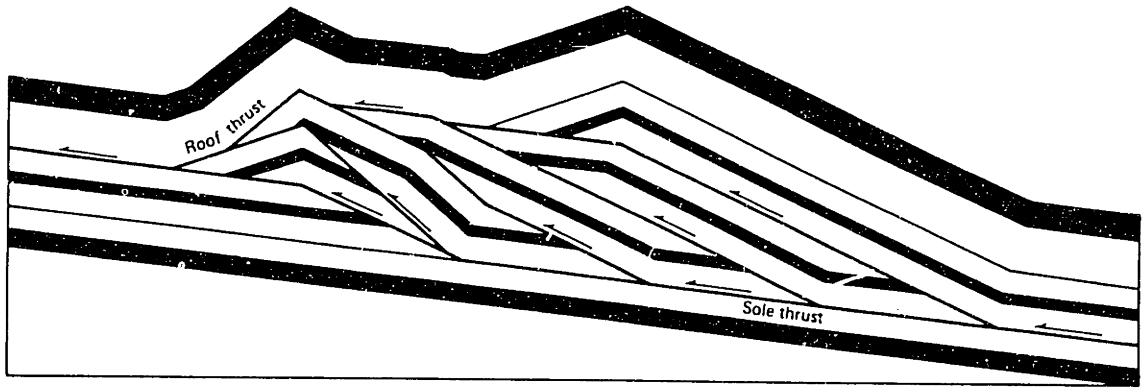


FIGURE 2.2.39 Detailed cross section of part of the Foothills fold-and-thrust belt in the Canadian Rocky Mountains, showing complex imbrication of the Mesozoic strata and undeformed Paleozoic at depth [from Suppe 1985].

a)



b)

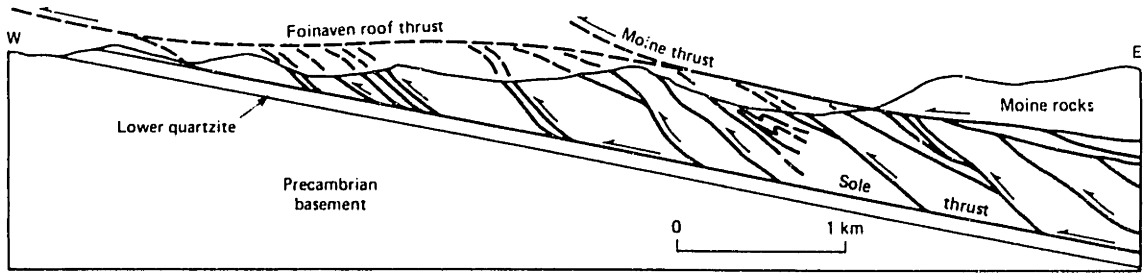
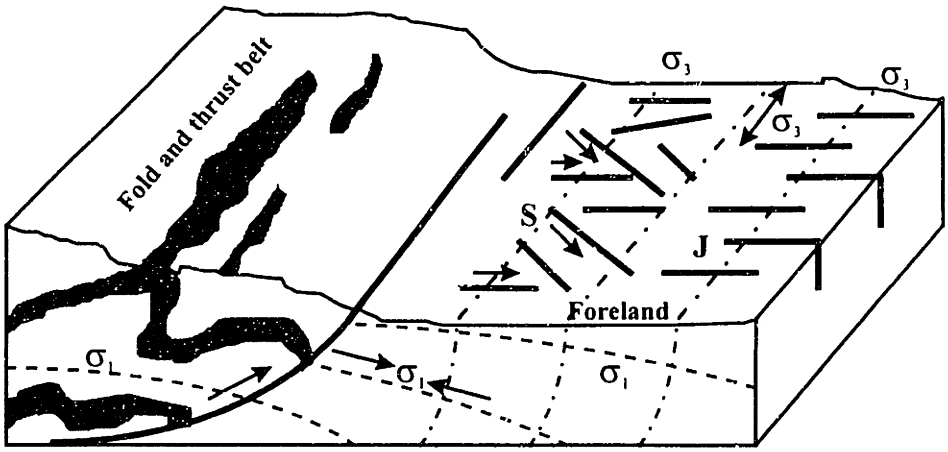


FIGURE 2.2.40 Duplex structure of the Moine thrust system, Scotland [from Suppe 1985]: a) schematic drawing; b) example of the actual thrust system.

The foreland in front of a fold-and-thrust belt is commonly fractured due to the horizontal compressive stress. According to Billings (1972), minor folds and thrusts may develop in the foreland in front of an overthrust system. Price & Cosgrove (1990) point out that thrusts in the foreland develop only when the intermediate principal stress σ_2 is such as that in the diagrams by Hafner (Figure 2.2.34a) and Hubbert (Figure 2.2.35b), i.e. orthogonal to the sections. However, it is possible that a stress of small extension develop perpendicular to the section due to a slightly convex trend of the fold belt towards the foreland. Such a stress system is illustrated in **Figure 2.2.41**: σ_1 is perpendicular to the trend of the fold belt, σ_2 is the vertical stress of gravity, and σ_3 is the extension parallel to the fold belt trend. According to Price & Cosgrove (1990), strike-slip faults (indicated as S in Figure 2.2.41) may form in the sediments adjacent to the fold belt where the differential stresses are higher, whereas vertical tensile joints (J in Figure 2.2.41) may open orthogonal, but distant from, the fold belt. Also, in a sedimentary sequence that consists of layers of different material properties, it is possible that at a certain distance from the fold belt in the foreland strike-slip faults form in some layers and tensile joints open in others.



S: strike-slip faults
J: joints

FIGURE 2.2.41 Block diagram showing fracture patterns likely to develop in the foreland forward from a fold-and-thrust belt [after Price & Cosgrove 1990].

2.3 FRACTURE SYSTEMS IN REMOTE TENSION

True tensile fractures (joints) represent discontinuous surfaces that have not experienced any shear movement of the walls past one another. Tensile joints are dilational fractures with very irregular surfaces. **Figure 2.3.1**a schematically represents the characteristic surface undulations on the face of a tensile joint: plumose and conchoidal structures in the central area, and en-echelon fracturing in the fringe zones. Figure 2.3.1b shows an example of upward radiating plumose structure perpendicular to conchoidal structure in greywacke sandstone.

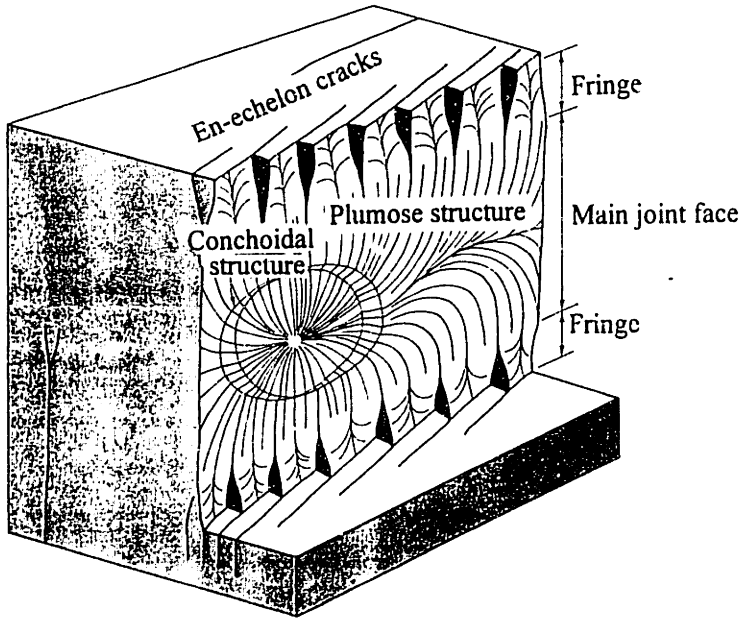
True tensile fracturing occurs when the minimum principal effective stress σ_3' is tensile (negative), and the following two conditions are met:

$$|\sigma_3'| = |\sigma_3 - \mathbf{p}| > |\mathbf{T}| \qquad (2\text{-}1a)$$

$$\sigma_1' - \sigma_3' = \sigma_1 - \sigma_3 \leq |\mathbf{4T}| \qquad (2\text{-}1b)$$

The first condition (2-1a) is that σ_3' (the difference between the minimum total stress σ_3 and the fluid pressure p) exceeds in magnitude the tensile strength T of the rock. The second condition (2-1b) is that the differential stress (the difference between the maximum and minimum principal stresses) is relatively small. When these requirements are met in the earth's crust tensile joints open on planes perpendicular to the direction of σ_3' due to hydraulic fracturing of the rock [Price & Cosgrove 1990; Suppe 1985].

a)



b)

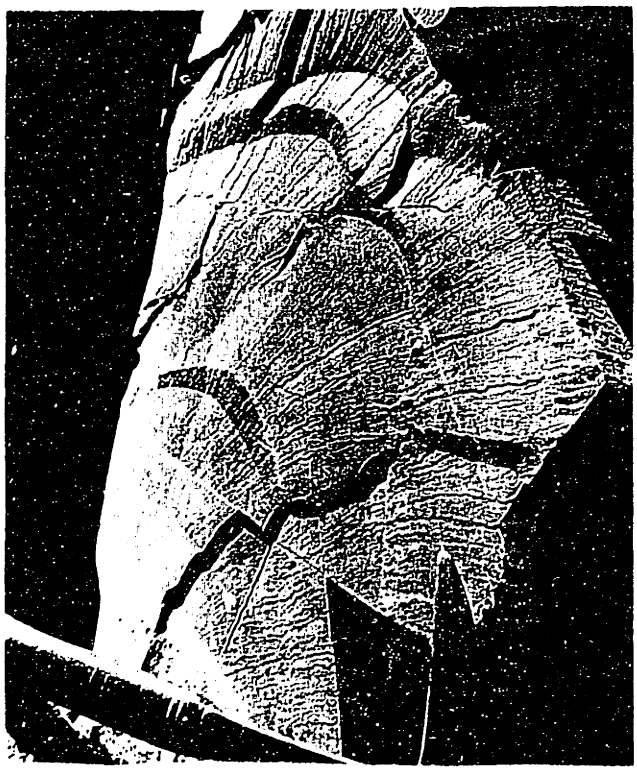


FIGURE 2.3.1 Typical irregular surface of a tensile joint [from Suppe 1985]: a) schematic representation showing some common morphological features; b) joint surface in Mezosoic graywacke sandstone showing upward radiating plumose structure, California.

Conditions such as that described by Equation 2-1 can be created by various geologic mechanisms, for example, by crustal extension, folding, etc. When the region of interest is considerably smaller than the geologic structure that defines the stress field within it, it is possible that the principal stress directions remain essentially constant throughout that region. If the minimum effective principal stress is tensile, the stress system can be defined as that of "remote tension" (an example is shown in **Figure 2.3.2**).

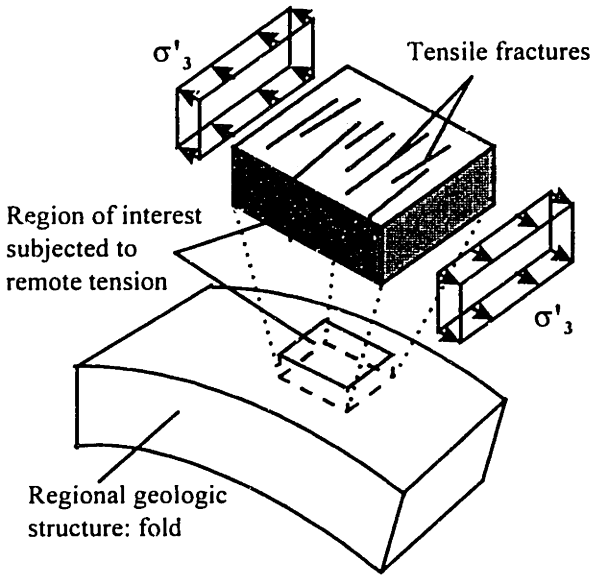


FIGURE 2.3.2 Schematic representation of conditions of remote tension in the earth's crust.

Suppe (1985) calculates the maximum possible depth of true tensile fracturing, consistent with the brittle failure envelope. If σ_1 is the vertical stress of gravity, the maximum depth of true tensile fractures can be calculated as:

$$\mathbf{Z} = \left| \frac{\mathbf{3T}}{\rho \mathbf{g}(\mathbf{1} - \lambda)} \right| \tag{2-2}$$

In Equation 2-2 λ is the fluid-pressure ratio, defined as:

$$\lambda = \frac{\mathbf{p}}{\rho \mathbf{g} \mathbf{z}} \tag{2-3}$$

where p is the fluid pressure at depth z, ρ is the density of the rock, and g is the acceleration of gravity. **Figure 2.3.3** presents a plot of Equation 2-2 for ρ=2700 kg/m^3 and different values of λ, where λ=0.4 corresponds to the hydrostatic fluid pressure. Figure 2.3.3 shows that in weaker rocks the higher the fluid pressure, the larger the maximum depth of tensile fracturing; very strong rocks (|T|~30 MPa) can experience tensile fracturing throughout most of the crust. Suppe (1985) also points out that: 1) if σ_2 or σ_3 is vertical, the maximum depth at which true tensile fractures may form is less than that when σ_1 is vertical; and 2) so-called transitional tensile joints (i.e. hybrid extensional and shear fractures) can form at depths about 40 % larger than the maximum depth of true tensile fractures.

The fracture systems, typical of remote tension conditions, consist of numerous subparallel tensile fractures (joints). The joint systems in nature are very similar to the joint patterns obtained in laboratory experiments on brittle materials subjected to uniaxial extension. **Figure 2.3.4** shows the development of an extensional joint system in experiments by Wu & Pollard (1992). Microcracks open first (Figure 2.3.4a); with increasing strain, they evolve into subparallel tensile joints orthogonal to the direction of extension (Figure 2.3.4b, c, and d).

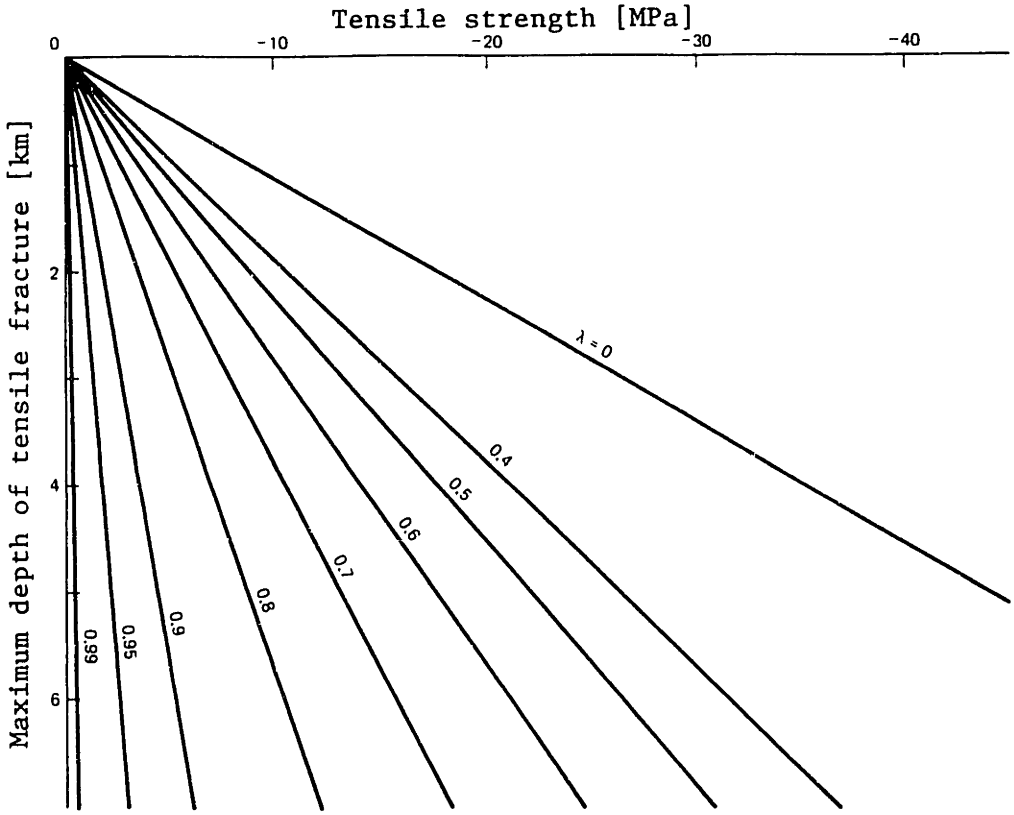


FIGURE 2.3.3 Maximum depth of tensile fracturing as a function of tensile strength T and fluid pressure ratio λ when σ_1 is vertical and ρ=2700 kg/m^3 [from Suppe 1985].

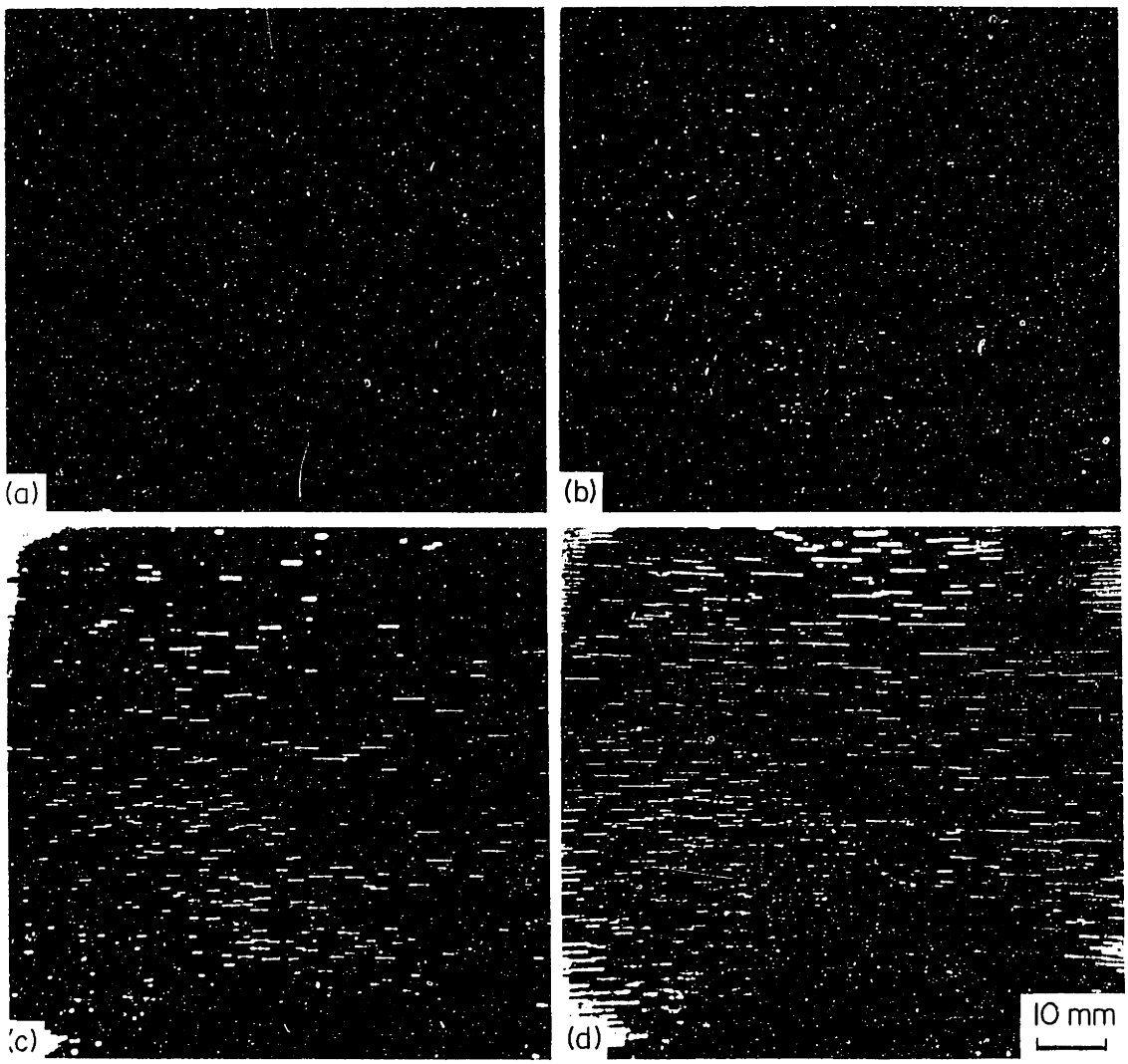


FIGURE 2.3.4 Development of a fracture set in brittle material: photographs at increasing extensional strain ε_{yy} [from Wu & Pollard 1992].

The rest of Section 2.3 reviews in detail the geometry of fracture systems that most commonly develop in remote tension in nature, including some well documented field examples. The characteristic geometry of fracture systems composed of subparallel vertical or steeply dipping joints is reviewed in Section 2.3.1. The effect of possible rotation of the remote stress direction on the fracture system geometry is discussed in Section 2.3.2. Section 2.3.3 reviews the characteristic geometry of extensional joints that are subparallel to the topographic surface.

2.3.1 Fracture systems composed of subparallel vertical joints

In nature, steeply dipping or vertical joint sets, formed in remote tension where σ_3' is approximately horizontal, are some of the most commonly encountered fracture systems in competent rocks of all lithologies and tectonic settings. Because of the significant effect of the lithology of the host rock, vertical joints in igneous and in layered or foliated rocks are reviewed separately.

Subparallel vertical or steeply dipping joints in igneous rocks

Figure 2.3.5 depicts a well-exposed outcrop in granodiorite in the Sierra Nevada. According to Segall & Pollard (1983a) who studied that outcrop in detail, the fractures initiated in response to regional horizontal tensile stresses of magnitude between 1 MPa and 40 MPa (the tensile strength of granodiorite), and dilation accommodated extensional strains on the order of $1x10^{-4}$ to $5x10^{-4}$. The fracture system in Figure 2.3.5 consists of a single set of subparallel, approximately planar, extensional joints, striking N10°E to N20°E and dipping steeply to the east 70° to 90°. The orthogonal spacing between mapped joints ranges from about 20 cm to nearly 25 m. The outcrop lengths vary from 1 or 2 m to about 70 m, shorter joints occurring more frequently than longer joints. In the vertical direction, the joints extend at least 3 to 4 m, possibly to depths comparable to their outcrop length. Joints in the area have formed at depths of at least several hundred meters.

The joints in Figure 2.3.5 are typically composed of multiple subparallel, straight or gently curved, en-echelon segments [Segall & Pollard 1983a]. The segments range from a few centimeters to approximately 1 m in length. Along a single joint, adjacent segments step to the right or left at steps of about 2 or 3 cm some of which are connected by traversing cracks. **Figure 2.3.6** shows a detailed map of a typical joint (indicated as inset II in Figure 2.3.5).

Another important characteristic of the outcrop in Figure 2.3.5 is the presence of numerous small cracks in the rock between the mapped joints [Segall & Pollard 1983a]. **Figure 2.3.7** illustrates the geometry of a population of small cracks (indicated as inset I in Figure 2.3.2). The small cracks are centimeters to a few tens of centimeters long and often less than 0.1 mm wide. They are parallel and adjacent to the joints, and contain similar fillings.

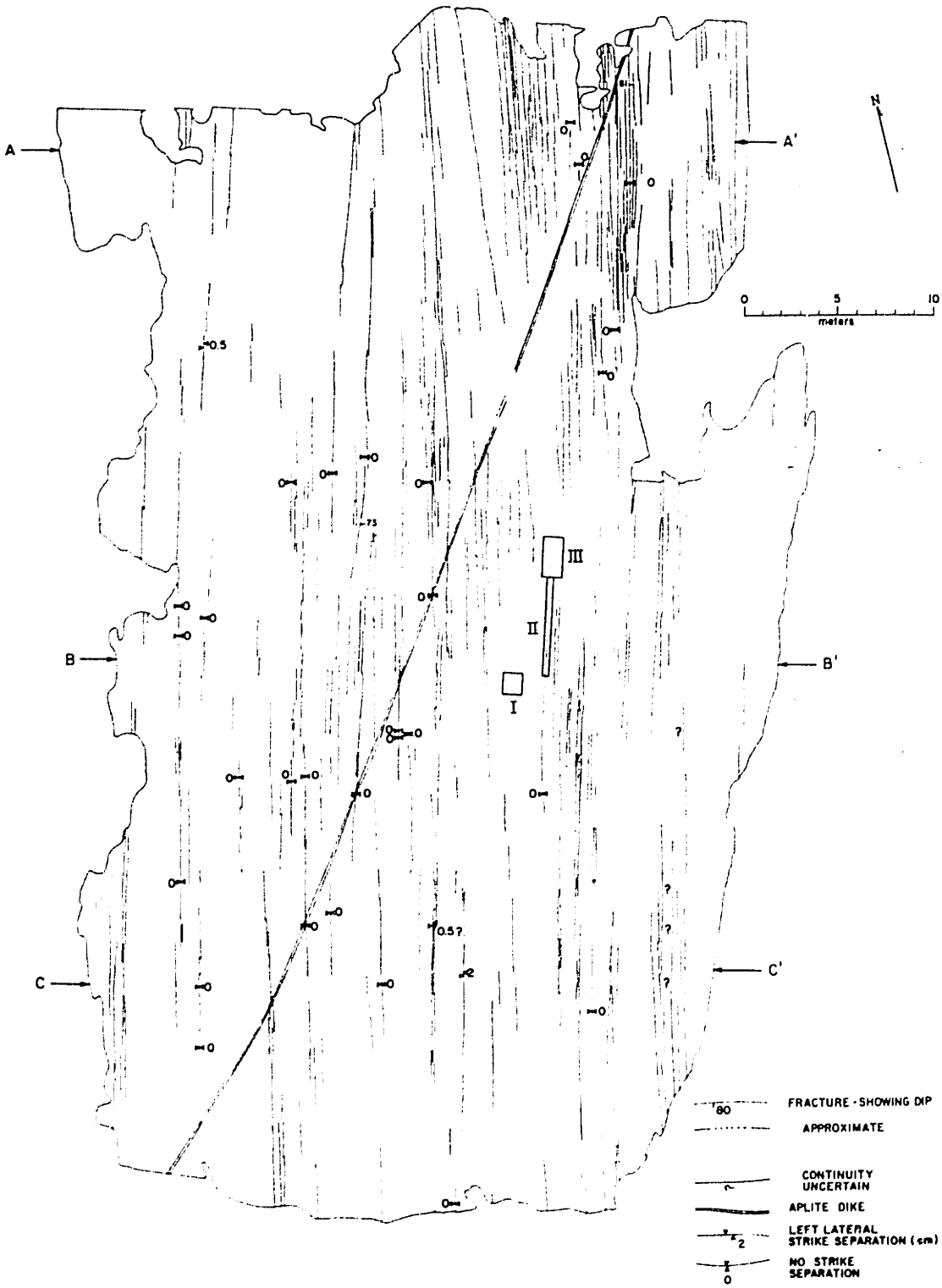


FIGURE 2.3.5 Outcrop map of a joint system, formed in remote tension in granodiorite, Sierra Nevada [from Segall & Pollard 1983a]. A-A', B-B', and C-C' indicate calculated strain traverses. I, II, and III are locations of detailed maps (Figures 2.3.6-2.3.8).

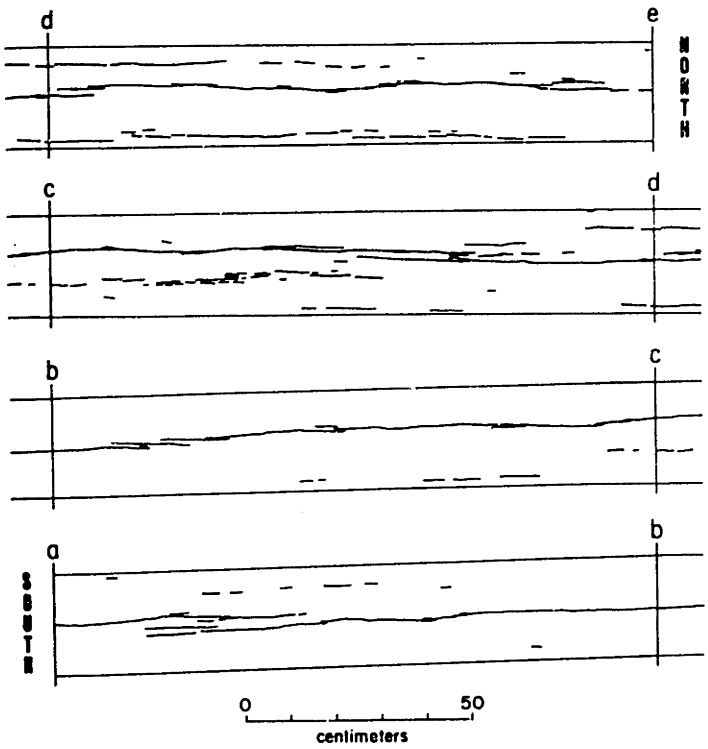


FIGURE 2.3.6 Detailed map of a joint trace [from Segall & Pollard 1983a]. Location of mapped joint is indicated as II in Figure 2.3.5.

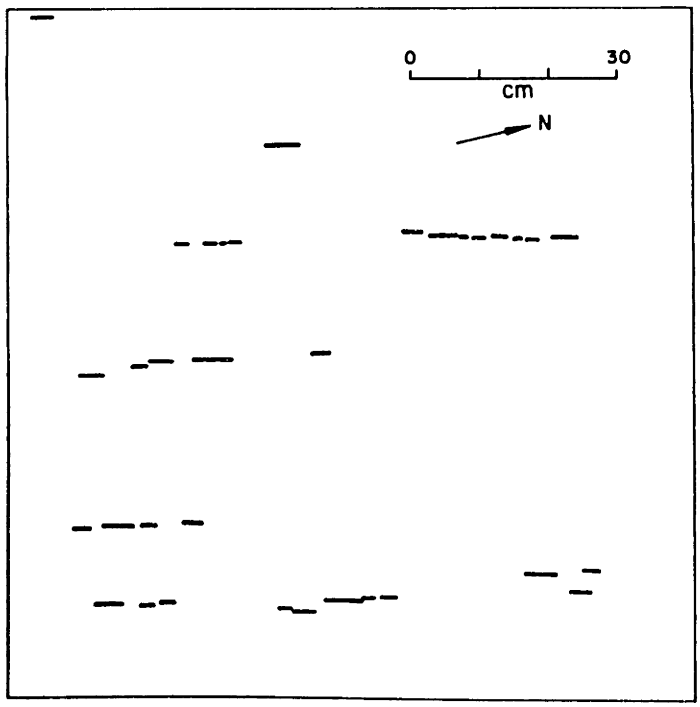


FIGURE 2.3.7 Detailed map of small cracks in the rock between map-scale joints [from Segall & Pollard 1983a]. Location is indicated as I in Figure 2.3.5.

Figure 2.3.8 depicts the end of a typical joint (inset III in Figure 2.3.2), made up of a number of very fine cracks in complex echelon patterns. The shape of the segment ends is illustrated in **Figure 2.3.9**: joint widths decrease gradually toward the end and there is no crack branching. According to Segall (1984), such joint ending implies stable propagation.

Fracture systems that form in remote tension in crystalline rocks have the typical geometry of the system shown in Figure 2.3.5: few widely spaced large joints and, clustered around and between them, numerous smaller joints and cracks. Segall & Pollard (1983a) and Segall (1984) speculate that the nature of interaction between joints in igneous rocks is such that the longest joints restrict the growth of shorter joints in their vicinity. According to Segall (1984), the joints that influence the propagation of a particular crack can be separated into two groups. Joints in the so called local group, near the particular crack, interact directly with that crack. Joints in the much larger remote group influence the particular crack only through their effect on the bulk mechanical properties of the rock mass. When the favored joints propagate, they interact with increasingly distant neighbors. As the joint set evolves, more and more joints are inhibited from propagation. Therefore, whether a crack propagates or not is determined by the complex joint interaction, the length distribution of neighboring joints, as well as by the local grain-scale heterogeneities and the increasing remote strains.

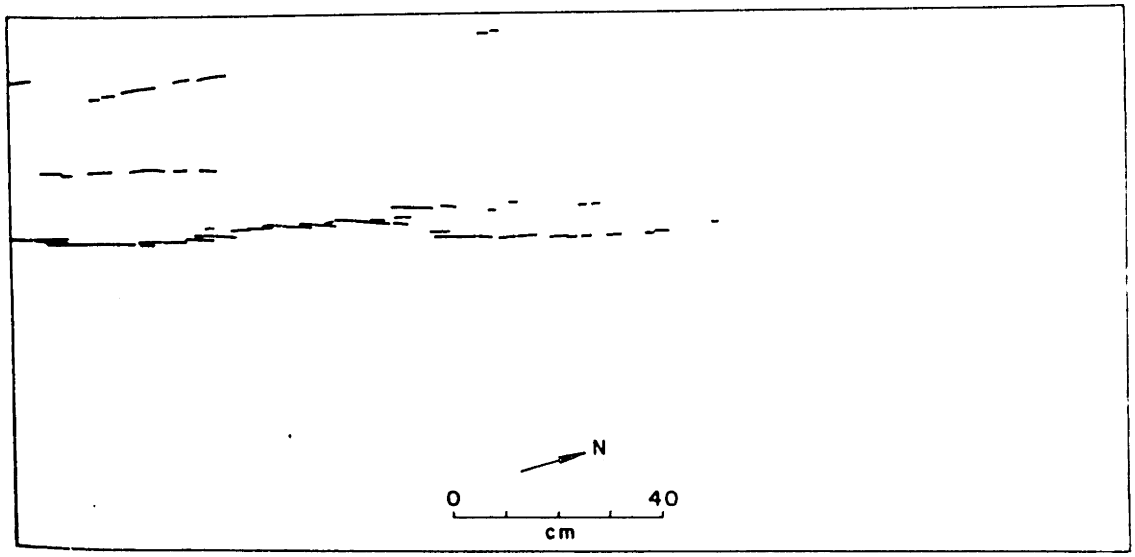


FIGURE 2.3.8 Detailed map of joint termination [from Segall & Pollard 1983a]. Location is indicated as III in Figure 2.3.5.

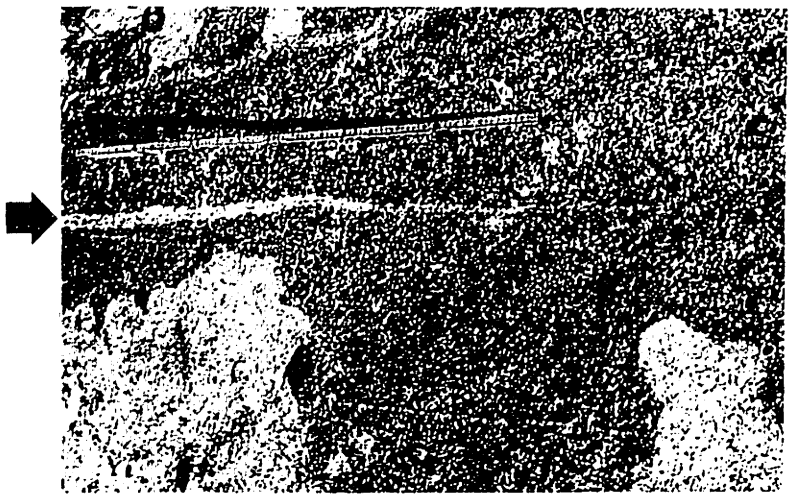


FIGURE 2.3.9 Termination of a joint in granodiorite [from Segall 1984]. The arrow indicates a joint (parallel to 1-m-long scale), bordered by light-colored, altered granodiorite. The joint and altered zone thin toward the joint termination. No branching is observed.

The fact that the joints in the outcrop in Figure 2.3.5 are mostly planar or very gently curving suggests that they formed under a significant remote differential stress. This conclusion can be made in the light of results from laboratory experiments and numerical analysis of joint traces by Thomas & Pollard (1993). In their study they found out that straight joints formed under significant remote differential stress (measured as the difference between the stresses parallel and perpendicular to the joint traces), whereas curving traces developed under nearly isotropic remote stresses (**Figure 2.3.10**).

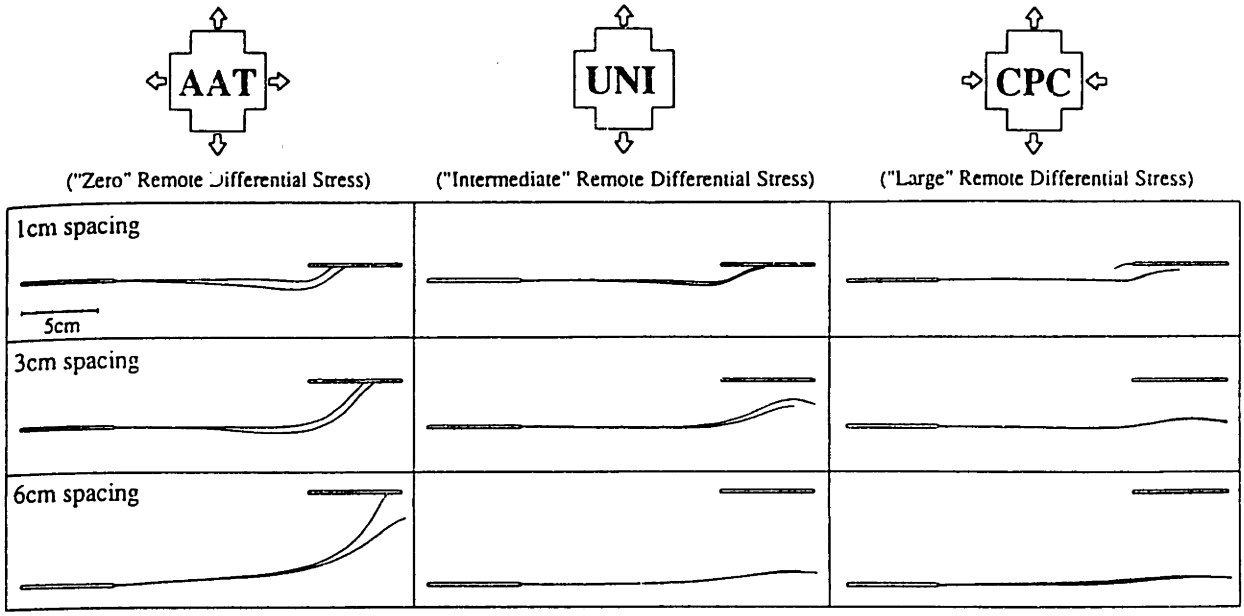


FIGURE 2.3.10 Fracture paths generated in the laboratory under different remote differential stress conditions [from Thomas & Pollard 1993]. AAT: all-around tension; UNI: uniaxial extension; CPC: crack-parallel compression. Two paths are shown for each loading.

Subparallel vertical or steeply dipping joints in layered or foliated rocks

Figure 2.3.11 illustrates an example of jointing under remote tension in adjacent schists and gneisses in Russel, Massachusetts. The exact mechanism that has created that system is not known. The joints may have formed on the flank of a large asymmetric regional fault; or they may have opened parallel to regional normal faults. In any case, the joint system in Figure 2.3.11 illustrates the typical geometry of fracture systems in remote tension. The fractures are sub-parallel, the spacing between them being much smaller in the schists than it is in the gneisses. Also, the closely spaced joints seem to be of similar (in this case relatively large) size which is typical of layered or foliated rocks.

Vertical joints in sedimentary rocks are usually orthogonal to the horizontal bedding. Joint spacing is relatively constant within a sedimentary unit and is a function of rock type and bedding thickness [Suppe 1985; Price 1966]. For example, **Figure 2.3.12** shows an outcrop in limestone in which thin beds are cut by closely spaced joints and thicker beds are cut by more widely spaced joints. Similar to joints in sedimentary rocks, joints in anisotropic rocks in general are orthogonal to schistosity, foliation, or other surfaces of pronounced anisotropy.

Narr & Suppe (1991) determine a nearly constant ratio of layer thickness to joint spacing of about 1.3 in siliceous shale, chert, and other sedimentary rocks. Harris et al. (1960), after extensively studying joints in dolomite, determine that the concentration of joints is inversely related to the bed thickness. Bogdanov (1947) and Novikova (1947) obtain the following relationship between mean joint spacing, S, and bed thickness, D, in limestone and sandstone:

$$\mathbf{S = k'D} \qquad (2\text{-}4)$$

where k' is a constant related to the lithology of the bed. Model experiments by Sowers (1973) and theoretical analyses by Hobbs (1967) and Price (1966) also predict a linear relationship between fracture spacing and bed thickness.

In a simple model, Price (1966) considers a uniform competent bed of thickness D and tensile strength T, enclosed between two incompetent units, and subjected to a uniform horizontal tensile stress σ_t (**Figure 2.3.13**a). When a tensile fracture opens (AA in Figure 2.3.13b), the horizontal stress is relieved at the free surface, but increases away from the fracture due to a shear stress ($\tau=c+\mu\sigma_n$) along the bedding planes. At a certain distance W from the fracture where the horizontal effective stress is equal to σ_t, a new tensile fracture may form (W is approximately equal to the mean spacing S in Equation 2-4). The equilibrium of the portion W of the competent unit can be expressed as $\sigma_t D=\tau D$ (Figure 2.3.13c). Price concludes that, if the stresses (σ_n and σ_t) and the material properties (μ) are the same, increase in the bed thickness D implies increase in the fracture spacing S.

In nature, joints in massive competent beds are often closely spaced. Price & Cosgrove (1990) point out that, besides bed thickness, other factors influence the spacing between joints in massive sedimentary rocks. For example, Ladeira & Price (1980) emphasize the importance of hydraulic fracturing. In many cases, initiating or extending a fracture leads to a drop of the fluid pressure within that fracture (**Figure 2.3.14**). Away from the fracture the fluid pressure increases and, at a distance L, reaches the high value that determines a negative effective stress. This distance L (smaller than the distance W in the friction analysis in Figure 2.3.13) is the minimum distance from the older fracture where a new hydraulic fracture may develop. **Figure 2.3.15** shows actual relationships between bed thickness and joint spacing, established by Ladeira & Price (1980) who measured joint spacing in thin-bedded and massive greywacke and limestone.

FIGURE 2.3.11 Inclined joints in adjacent gneiss and schist, Russel, Massachusetts [photo by H.H. Einstein].

FIGURE 2.3.12 Closely spaced joints in thin beds and more widely spaced joints in thicker beds in Cretaceous limestone, Yates oil field, Texas.

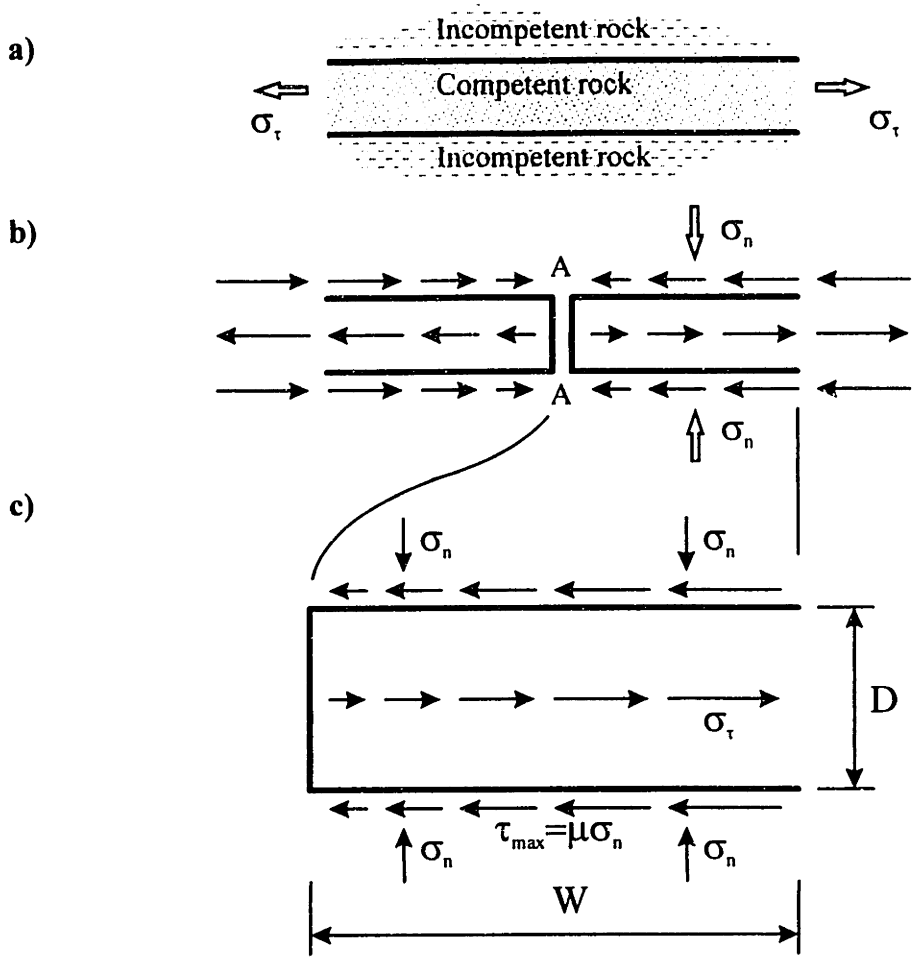
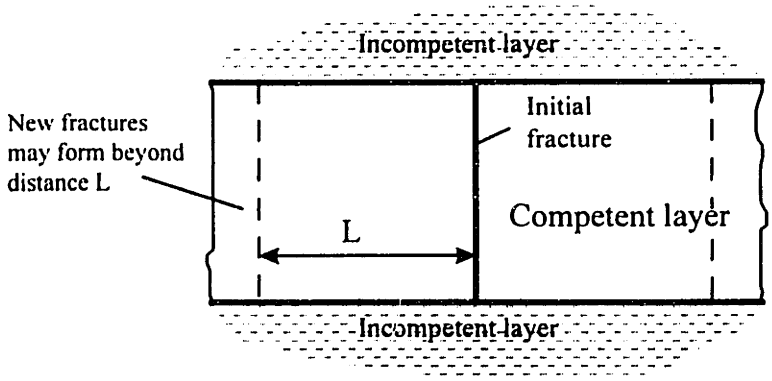


FIGURE 2.3.13 Relationship between bed thickness and joint spacing defined by friction along bedding boundaries [after Price 1966]: a) a single competent unit subjected to a uniform effective horizontal stress; b) generation of a single tensile fracture, resulting in reduction of lateral effective tensile strength and increase of bedding parallel shear stress; c) equilibrium conditions for portion of the competent unit of length W adjacent to the tensile fracture.

a)



b)

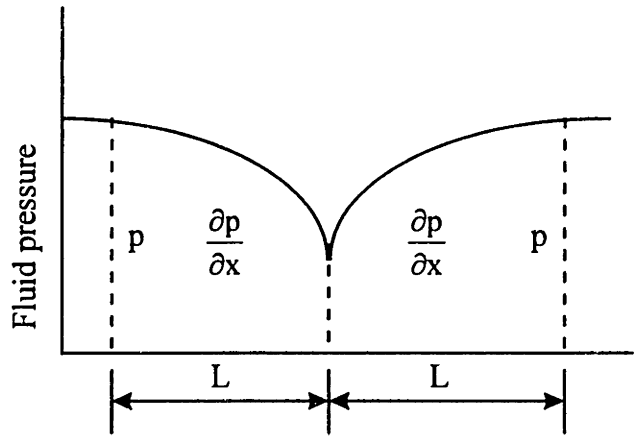


FIGURE 2.3.14 Spacing of hydraulic fractures defined by the permeability of the rock unit and fluid pressure gradient [after Ladeira & Price 1980].

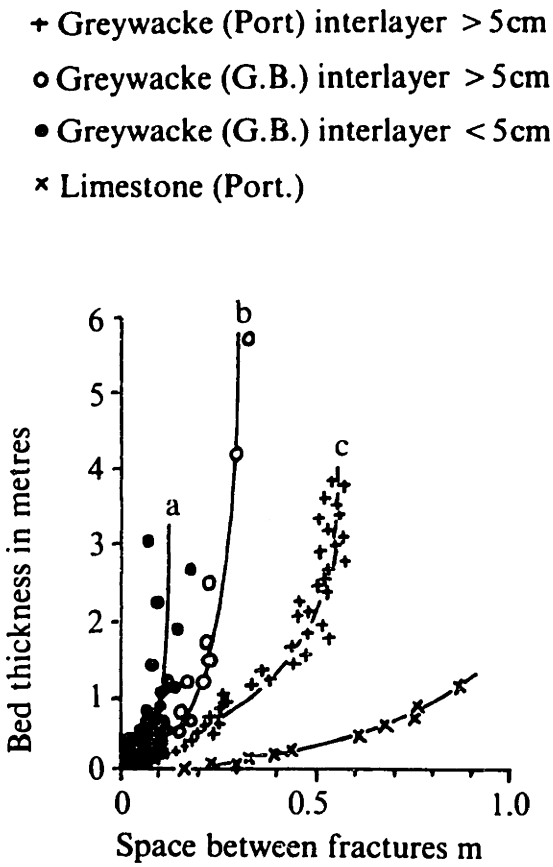


FIGURE 2.3.15 Relationships between joint spacing and bed thickness for thin-bedded and massive sedimentary rock units [from Ladeira & Price 1980].

When rocks of different lithology and material properties are interbedded, bed-perpendicular jointing occurs only in the formations where the remote tensile stress exceeds the strength of the rock. For example, Helgeson & Aydin (1991) have studied in detail an alternating siltstone and shale turbidite sequence in the Appalachian Plateau in central New York in which joints are confined mostly to the siltstone layers. Joint initiation points in the fractured siltstone are almost always located at voids, cusps, fossil inclusions, pyrite concretions and other flaws at the bedding interfaces.

Helgeson & Aydin analyze the various modes in which joints communicate across layer interfaces in the siltstone-shale sequence. Individual joints in the fractured siltstone are often well aligned vertically and sometimes laterally offset, thus forming composite joints (**Figure 2.3.16**). Four major types of vertical alignment between the joints in siltstone layers as a function of the thickness of the interbedded shale are illustrated in **Figure 2.3.17**. Figure 2.3.17a shows the in-plane alignment of segments in a stack of siltstone layers of similar mechanical properties. Figure 2.3.17b depicts joint segments that are still remarkably stacked in a vertical plane, but aligned slightly out-of-plane across shale lenses and thin

the shale layer. Figure 2.3.17c shows a case of practically absent communication between joints in siltstone layers which are interbedded with relatively thick shale layers. Figure 2.3.17d depicts the case when the relatively thick shale is also jointed. The joints in shale initiate at the ends of pre-existing joints in siltstone and propagate at small angles to them. Examples of joint interactions such as those illustrated in Figure 2.3.17 are shows in **Figure 2.3.18** and **Figure 2.3.19**. In Figure 2.3.18, vertical joints cut without offset across stacked layers of flat-lying Cretaceous limestone (similar to Figure 2.3.17a). Figure 2.3.19 shows interbedded jointed limestone and unjointed marl (similar to Figure 2.3.17c).

2.3.2 En-echelon cracks originating from a parent joint

Figure 2.3.20 schematically represents an array of cracks initiating from the edge of a single parent joint. According to Pollard et al. (1982), an en-echelon crack array grows from a single extensional parent joint in result to a change in the orientation of the remote stress. Initially, the (parent) extensional joint propagates in a plane orthogonal to the remote tensile effective stress. A slight temporal or spatial rotation of the remote principal stress direction introduces a rotation in the orientation of the local principal stresses near the tip of the parent joint. Since breakdown into a crack array requires less energy than would be necessary for the entire parent joint to grow into the new orientation, en-echelon cracks propagate along twisting surfaces and into planes that are orthogonal to the local tensile stress. The width of echelon cracks (indicated in Figure 2.3.20) is usually a small portion of the length of the parent joint. En-echelon cracks may grow to lengths that are relatively great compared to their width. **Figure 2.3.21** shows an en-echelon array of cracks along the edge of a parent joint in sandstone.

The width, thickness, and relative orientation of discrete echelon segments can be observed in cross sections that are perpendicular to the fracture propagation direction. **Figure 2.3.22**a and b depict two examples by Pollard et al. (1982) of en-echelon segments at two different outcrop scales. In Figure 2.3.22a, en-echelon segments of a quartz vein that are about 10 cm wide and 1 cm thick strike at approximately 5° to the trend of the vein. In Figure 2.3.22b, en-echelon segments of a minette dike, about 100 m wide and 2 m thick strike at about 6° to the average trend of the dike. Despite the different scales, the two echelon arrays exhibit some characteristic similarities, namely: 1) the crack segments within an en-echelon array are of comparable width and thickness, the ratio of thickness to width being small; 2) cracks systematically strike at small angles to the trend of their array; 3) there is a pronounced asymmetry of the crack form near crack tips; and 4) adjacent cracks overlap at their tips at distances that are approximately equal to the steps between the cracks. Figure 2.3.22c schematically illustrates the 3D geometry of a dike the upper edge of which is exposed on the surface as an en-echelon array similar to that in Figure 2.3.22b.

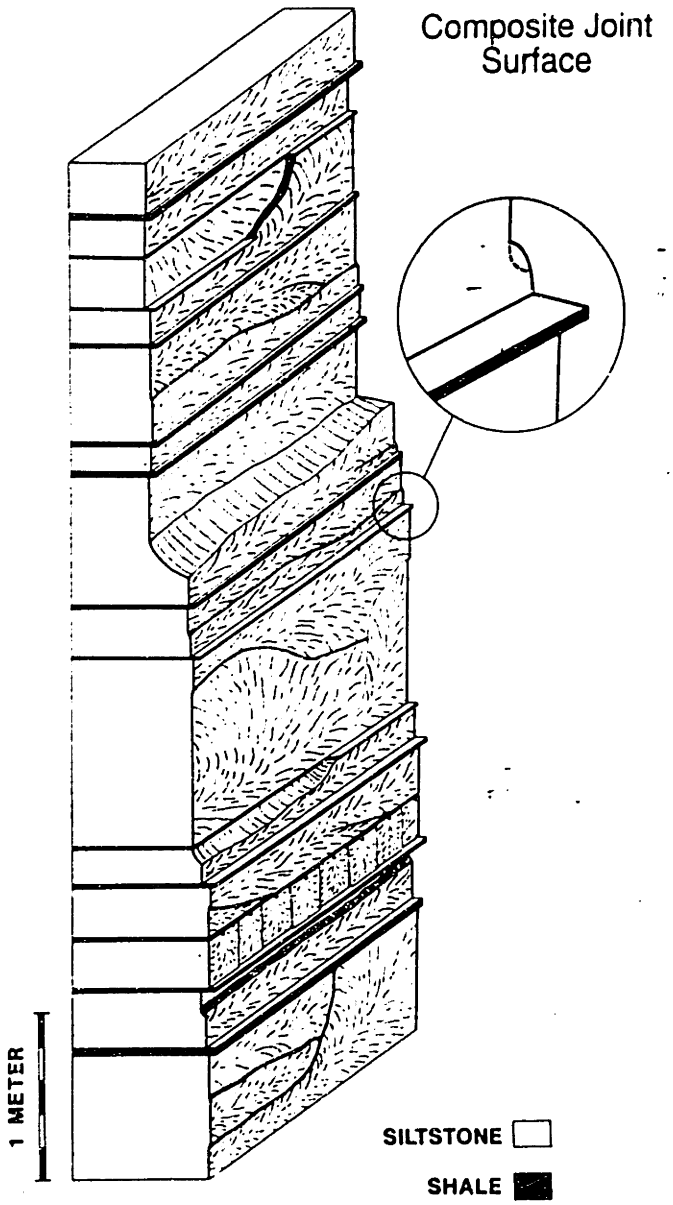


FIGURE 2.3.16 Large composite joint surface in alternating siltstone and shale sequence [from Helgeson & Aydin 1991]. The siltstone layers are individually jointed. The joints in adjacent siltstone layers are aligned out of plane across the shale layers (as shown in the detail). This out-of-plane alignment gives the overall vertical joint a discontinuous appearance.

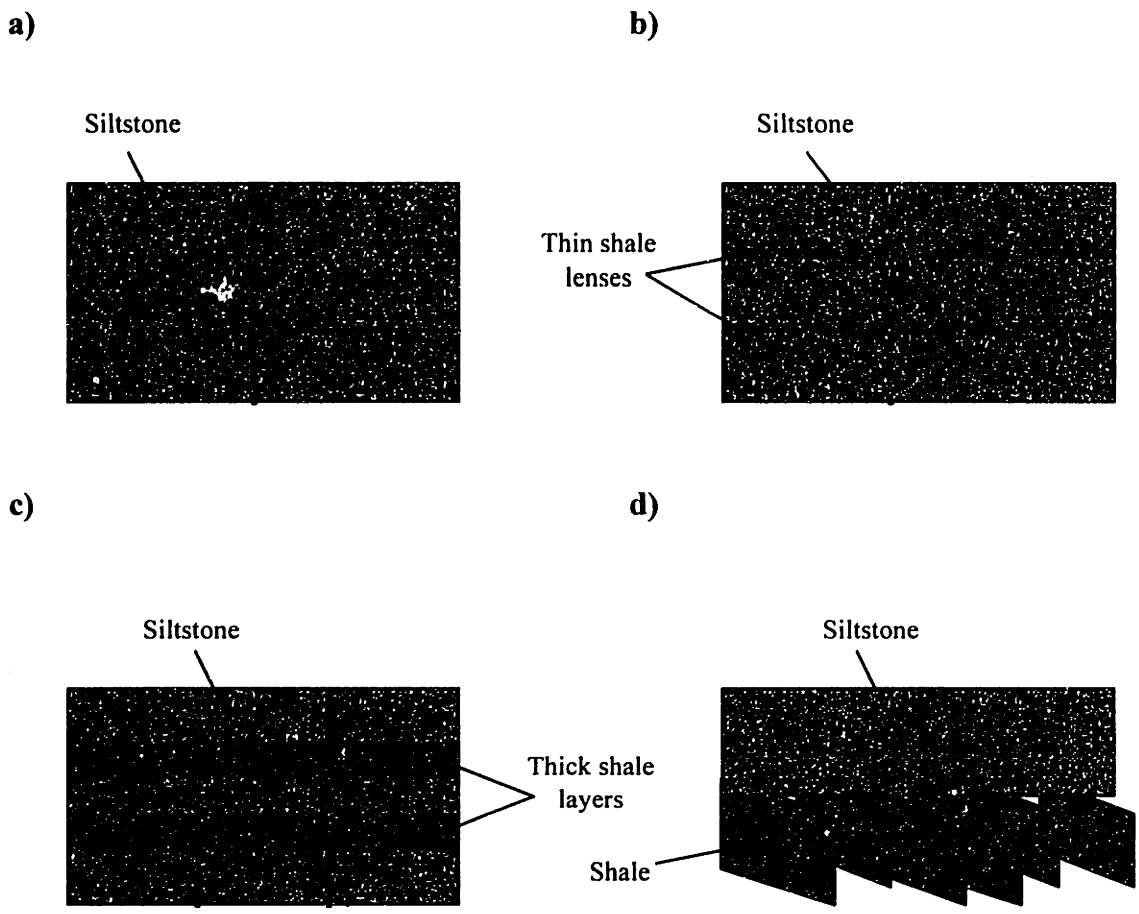


FIGURE 2.3.17 Schematic summary of observed characteristics of joint propagation in layered sedimentary rocks [after Helgeson & Aydin 1991]: a) In-plane addition of sequential joint segments in siltstone layers; b) out-of-plane addition of sequential joint segments across shale lenses and thin shale layers; c) independently formed joints in siltstone layers separated by thick shale layers; d) non-planar addition of joint segments in shale (lower layer) around the periphery of a pre-existing joint in siltstone (upper layer).

FIGURE 2.3.18 Vertical joints cutting across one or several beds in Cretaceous limestone, Yates oil filed, Texas.

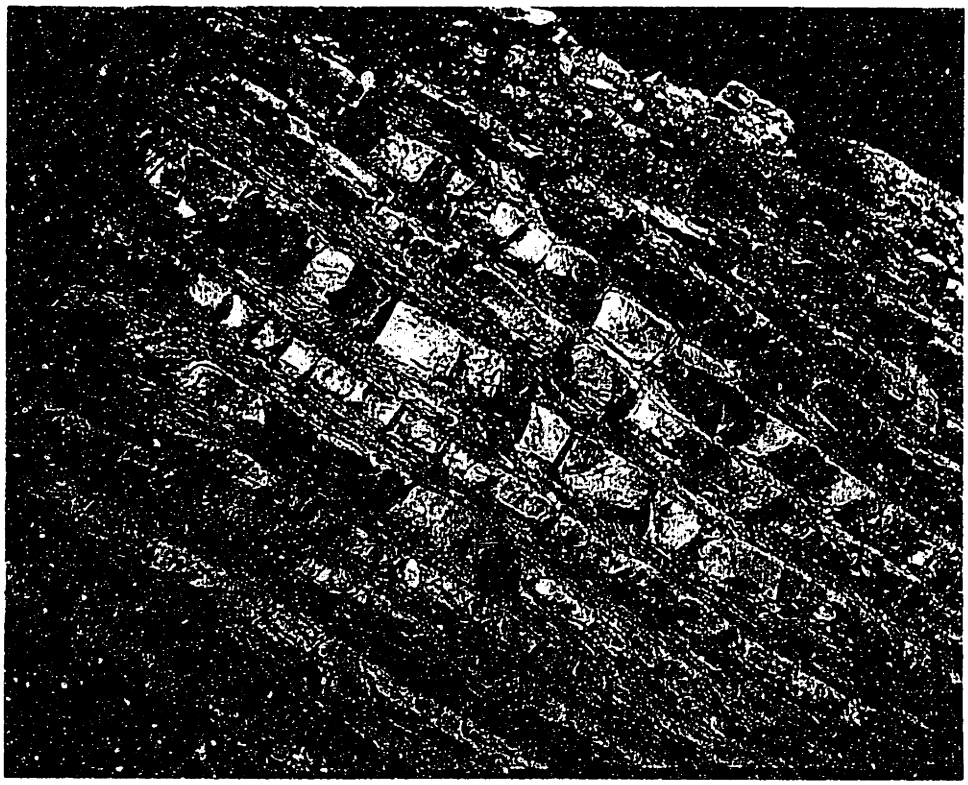


FIGURE 2.3.19 Well bed[illegible] limestone and marl with joints developed only in the limestone, Castellane, France [from Ramsay & Huber 1987].

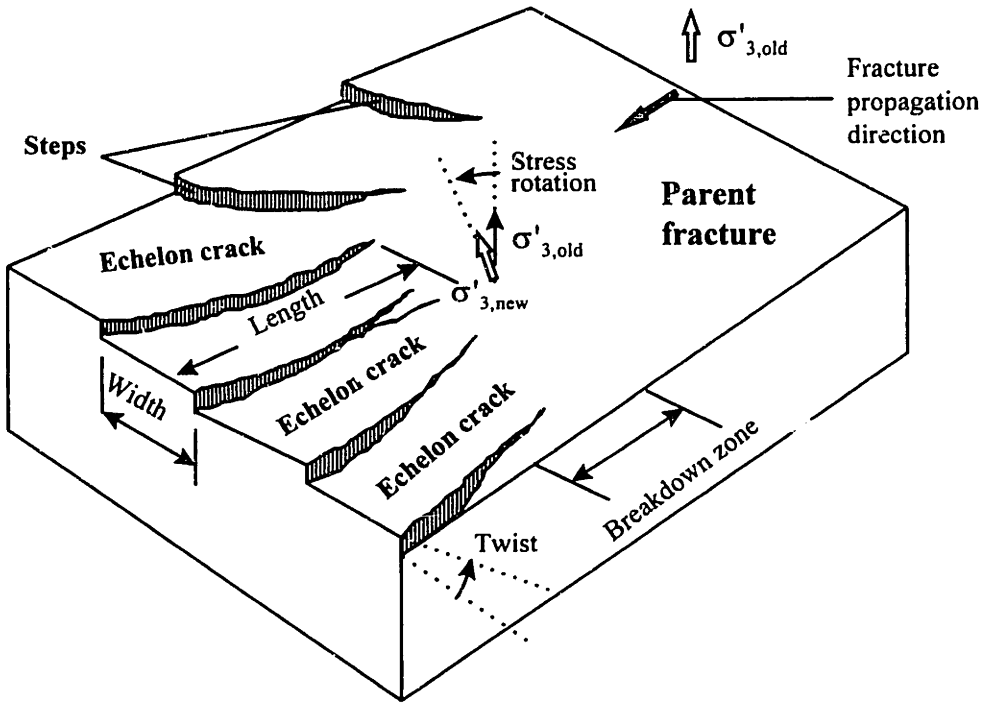


FIGURE 2.3.20 Idealized block diagram illustrating the geometry of parent and echelon cracks [after Pollard et al. 1982].

2.3.3 Sheet joints parallel to the topographic surface

Sheeting, or sheet structure, is a fracture system composed of extensional joints that run approximately parallel to the topographic surface. There is a controversy on the origin of sheet jointing [Twidale 1973]. The most widely advocated hypotheses are that sheet joints develop in response either to (1) unloading perpendicular to the land surface (for example, due to erosion), or (2) compressive stresses acting parallel to the ground surface. In both cases there is a net release of pressure perpendicular to the topographic surface. Strictly speaking, sheet joints do not form in remote tension but in compression. Nevertheless, sheet jointing is reviewed here since the characteristic geometry of the joint system is very similar to that in remote tension: numerous extensional joints, sub parallel to each other, and perpendicular to the direction of the minimum principal effective stress.

Sheet joints have been observed in a wide variety of igneous, sedimentary and metamorphic rocks: especially frequently in granitic rocks, but also in gneiss, rhyolite, sandstone, arkose, conglomerate, limestone, etc. [Suppe 1985]. Sheet structure is characteristic for massive rocks, i.e. rocks which either are otherwise unfractured, or at least do not display open joints. The occurrence of sheet joints only in massive rocks is attributed to the fact that in previously fractured rocks the stress release would be accommodated by slip along pre-existing fractures

Some sheet joints near the land surface are very closely spaced and exist only at relatively shallow depths [Twidale 1973; Suppe 1985]. They separate the rock in thin layers that are less than a meter and sometimes as little as a millimeter to a few centimeters thick. Such superficial joint sets are essentially discontinuous and relatively young in age. They are consistently related to the shape of the land surface. **Figure 2.3.23** illustrates typical closely spaced sheet joints in granite. Other sheet joints may extend to depths of a few hundreds of meters, the spacing between them regularly increasing with depth [Suppe 1985]. For example, **Figure 2.3.24** depicts sheet joints in granite, roughly elliptic in plan, that separate sheets ranging in thickness from less than a meter near the surface to between 5 and 10 m at depth of about 30 to 40 m. **Figure 2.3.25** illustrates a phenomenon that may be a proof that at least some sheet joints form in compression: sheets, enclosed by joints, that have buckled at the ground surface.

2.4 THERMAL CONTRACTION POLYGONS

Tensile fractures (joints), orthogonal to the boundary surface of a rock unit, can be created in response to rock contraction caused by cooling, desiccation, chemical reaction, or phase change. For example, igneous rocks contract due to cooling of lava; mud and silt contract in response to desiccation. In plan view, contraction joints usually bound randomly or preferentially oriented polygonal patterns. The predominant shape of polygons may be tetragonal or hexagonal, depending on the rheological behavior of the medium and the nature of induced volume change.

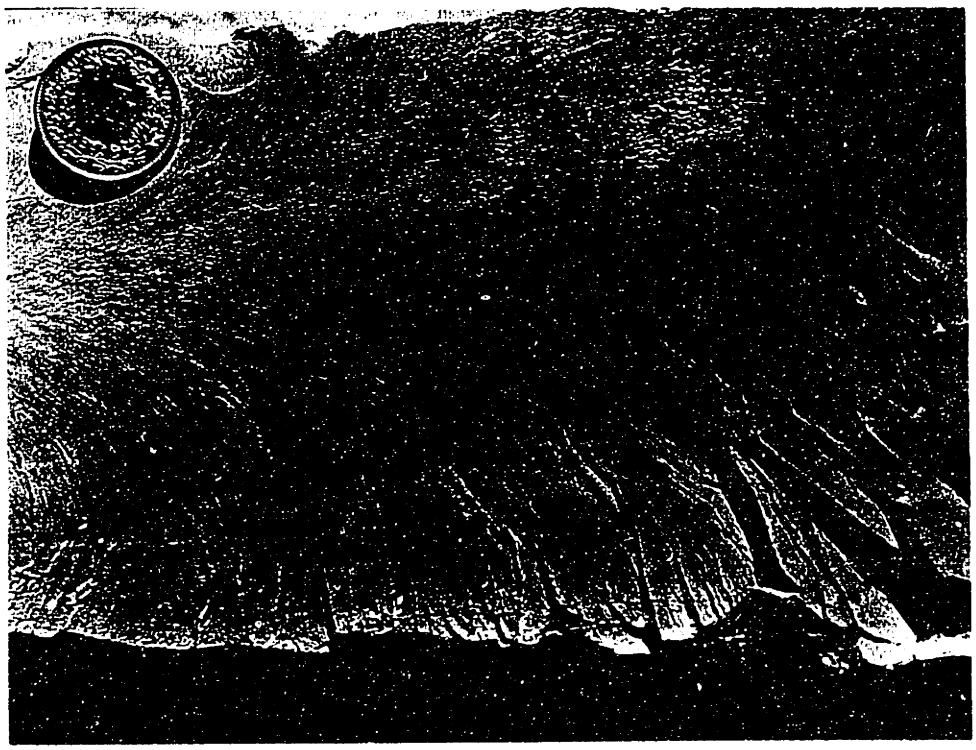
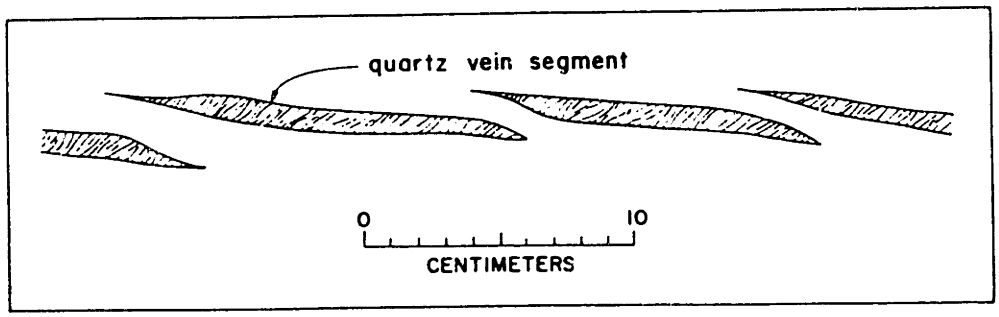
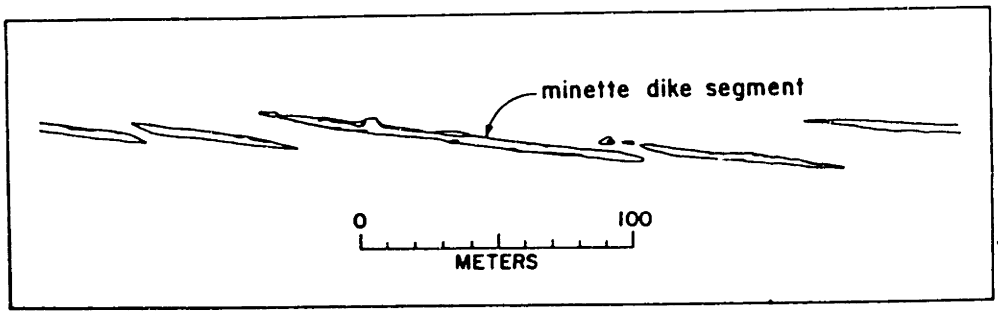


FIGURE 2.3.21 En-echelon cracks developing from a plumose marking in Flysch sandstone, Switzerland [from Ramsay & Huber 1987].

a)



b)



c)

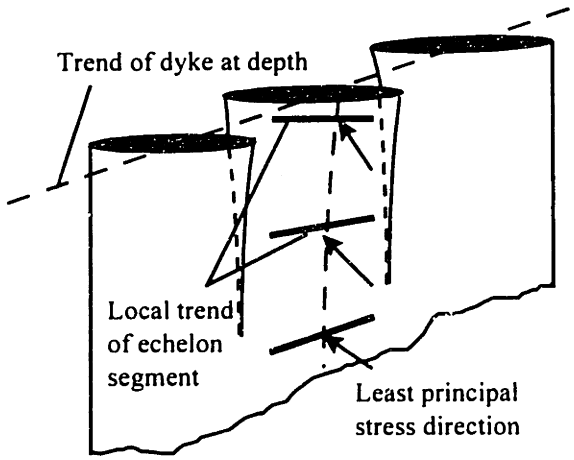


FIGURE 2.3.22 Extensional en-echelon fractures at different scales: a) map of a quartz-filled vein array from Carboniferous rocks, England; and b) map of part of a minette dike, New Mexico [from Pollard et al. 1982]; c) change of orientation of stresses at depth that causes the development of the en-echelon dikes in (b) [after Price & Cosgrove 1990].

The mechanics of contraction jointing and the development of polygonal patterns have been studied extensively by Lachenbruch (1962) in relation to thermal contraction of frozen soil (permafrost). This classical work, briefly discussed here, gives valuable insight on the development of thermal-contraction fracture systems of greatest interest in rocks: columnar joints due to contraction of cooling lava flows. Review of columnar joints in igneous rocks, including well documented field examples, constitutes the major subject of Section 2.4.

Thermal contraction cracks and polygonal patterns in permafrost and rocks

Figure 2.4.1 illustrates a typical polygonal pattern that can be observed in the Arctic. The irregular polygons, usually four-sided, 10 to more than 100 feet in diameter, are separated by several feet wide shallow trenches. A vertical ice wedge, such as that illustrated in the inset in Figure 2.4.1, extends about 10 to 20 feet downward below every trench. According to the generally accepted theory, an individual ice wedge originates as a thin vertical crack that opens in the permafrost during the arctic winter. During the spring, water from the melting snow in the top "active layer" of soil penetrates the crack and freezes there thus widening the ice wedge. The repetition of this cycle every year for centuries results in development of a vertical ice wedge.

Lachenbruch (1962) analyzes in detail the evolution of polygonal patterns in response to thermal contraction of permafrost and other materials. **Table 2.4.1** summarizes the geometrical types of polygonal patterns due to thermal contraction and the characteristic medium where each of them may develop.

According to Lachenbruch's model, when a fracture forms, it inhibits fracturing in the surrounding material in two ways: 1) by increase of the strength of the unfractured material since the points of greatest weakness have been removed; and 2) by reduction of the stress. However, while tension at right angles to the fracture is reduced to zero in its immediate vicinity, only minimal stress relief occurs parallel to the fracture (**Figure 2.4.2**). Since tension cracks propagate in the direction perpendicular to the maximum tensile stress, a new crack that advances into the zone of stress relief tends to form a right angle with the old fracture. Orthogonal systems are favored in non-homogeneous materials, where, due to irregularities of strength and stress, cracks originate at many points, propagate towards one another, and intersect at right angles. On the other hand, in homogeneous materials maximum uniform tensile stresses develop before any fracturing occurs. Cracks originate at very few points, then rapidly propagate and bifurcate many times before the branches intersect. Therefore, highly homogeneous, usually brittle, materials favor non-orthogonal (most commonly hexagonal) fracture patterns.

FIGURE 2.3.23 Sheeting joints in granite, Grimsel, Switzerland [photo by H.H. Einstein].

FIGURE 2.3.24 Sheet structure in granitic rock with decreasing number of fractures at depth, Norfolk County, Massachusetts [from Suppe 1985].

FIGURE 2.3.25 Buckled exfoliation sheet in granitic rock, Cloud's Rest, Yosemite National Park, California [from Suppe 1985].

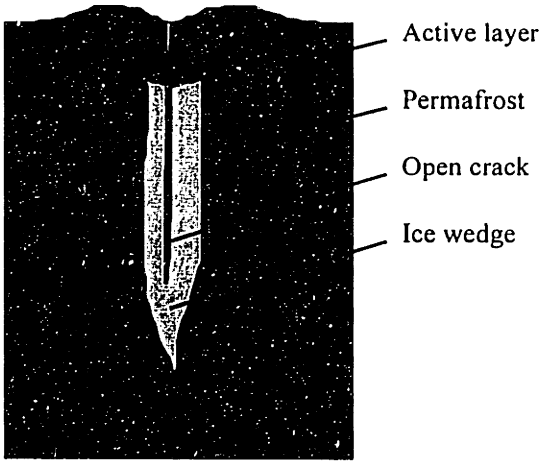
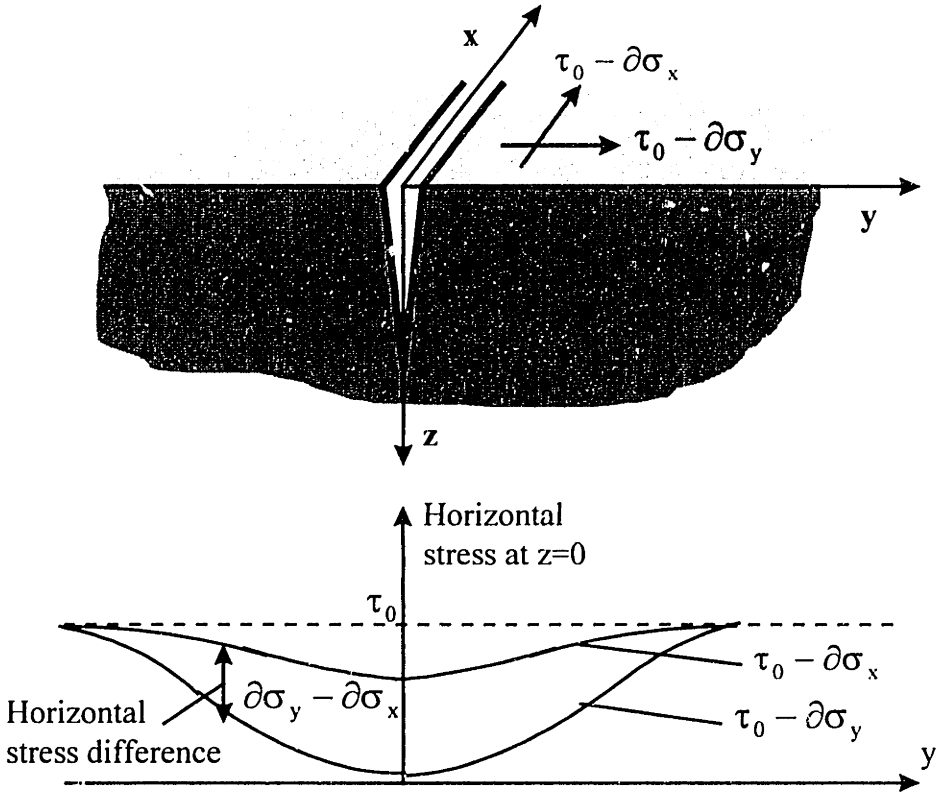


FIGURE 2.4.1 Plan view of ice-wedge polygonal patterns in northern Alaska, approximate scale: 1 inch = 100 yards [from Lachenbruch 1962]. Inset: cross section of an ice wedge.

POLYGONAL PATTERN	TYPICAL MEDIUM	FRACTURE PROCESS
Orthogonal: polygon intersections at right angles. • ***Random*** No preferred direction of the fractures. • ***Oriented*** Usually parallel and normal to the edges of water bodies.	Inhomogeneous or plastic medium. • Random loci of material heterogeneiy • Oriented thermal gradients (normal to the edges of water bodies).	The stress builds up gradually. The cracks originate at numerous points of low strength or high stress concentration. The cracks do not form simultaneously. Each new crack tends to join pre-existing ones at orthogonal intersections.
Non-orthogonal: mostly tri-radial (Y-type) intersections, commonly forming three angles of about 120°.	Very homogeneous, relatively non-plastic medium.	Under uniform cooling, fractures propagate and branch at obtuse angles, then the branches bifurcate, etc. All elements of non-orthogonal intersections are generated virtually simultaneously.

TABLE 2.4.1 Geometric characteristics of polygonal patterns in permafrost and other materials [after Lachenbruch 1962].

Lachenbrugh (1961) used a simple model of a semi-infinite solid to study the relationships between fracture depth and spacing, material properties, and stress distribution in basalt lava flows. The depth of crack penetration is related to the rate of energy dissipation at the advancing crack tip. Fracture spacing depends on the relief of stress by every fracture at the ground surface. Lachenbruch's analysis showed that the initial depth and spacing of contraction joints are of the same order of magnitude, and can be calculated as a function of the material properties of the contracting rock and the magnitude of the imposed thermal stresses. The fracture depth init ally does not exceed the thickness of the zone subjected to tension beneath the surface of the cooling mass (at depth, due to gravitational load, the stresses are compressive). During cooling, thermal tensile stresses penetrate to greater depths, and the initial fractures propagate downward. The final depth of contraction joints in igneous rocks often exceeds the spacing between them by two orders of magnitude, hence their name: columnar joints.



τ_0 = uniform horizontal tension before fracturing
$\tau_0 - \partial\sigma_x$ = component of tension parallel to crack after fracturing
$\tau_0 - \partial\sigma_y$ = component of tension normal to crack after fracturing

FIGURE 2.4.2 Stress at the ground surface near an isolated contraction crack [after Lachenbruch 1962].

Columnar joints in igneous rocks

Figure 2.4.3 illustrates an example of columnar joints in rhyolite. Such long polygonal columns, bounded by tensile joints, approximately perpendicular to the outer surfaces of the igneous body, are commonly encountered in lava flows and other shallow, quickly cooled igneous intrusions, e.g. sills and dikes [Suppe 1985; Price & Cosgrove 1990].

FIGURE 2.4.3 Columnar jointing in rhyolite, Rouyn-Noranda, Quebec, Canada [photo by J. Hango].

Suppe (1985) points out that there are two mechanisms that induce tension and tensile fracturing in a cooling igneous body: 1) inhomogeneous cooling of the lava, and 2) fixed boundaries between the country rock and the shrinking lava. Tensile stresses develop parallel to the boundary at the top of a lava flow which initially is cooler than the inside. Tensile stresses also develop parallel to the contacts of an intrusion or the bottom of a lava flow with the cooler country rock. When the temperature at the margins of the cooling igneous body falls below a certain value (900° for basalt), the material cannot flow further and the stress is relieved by initial tensile fracturing perpendicular to the outer surfaces. With gradual cooling of the interior of the igneous body, the fractures propagate from the margins inward and define long polygonal columns. The columns may be curved and tilted due to irregularities in the isotherms during cooling, caused most likely by topographic irregularities on the bedrock surface over which the lava flows. Fractures do not propagate into the country rock which is subjected to compression since it cannot expand freely under the increased temperature.

Columnar jointing in many basalt flows consists of lower and upper colonnades that are separated by a zone of distorted columns, called entablature [Suppe 1985; Peck & Minakami 1968]. The columns of the lower colonnade are shorter, larger in diameter and mostly hexagonal; the columns of the upper colonnade are longer, thinner, and mostly tetragonal. The joints of the upper colonnade grow downward from the top and the joints of the lower one grow upward from the bottom. The interaction of the two joint sets approaching from below and above creates the irregular stress field of the entablature in the middle.

Peck & Minakami (1968) describe the formation of columnar joints in the upper layers of lava lakes in Hawaii in the course of two years after the eruption. Cracks open within minutes after the exposure of the molten lava at the surface, and form random or oriented orthogonal networks. Within months after the eruption, as the crust cools and increases in thickness, the existing cracks extend downward and new cracks open, subdividing the crust into smaller polygons. Within two years after the eruption, additional fractures open at depths of tens of feet, propagate laterally and upward, and feather out near the surface. These new fractures outline large hexagonal polygons that do not correspond to the pre-existing polygons on the surface of the lava lake. Peck and Minakami point out that the polygonal patterns in the upper part of lava lakes resemble that of the irregular, random, orthogonal ice-wedge polygons described by Lachenbruch.

Figure 2.4.4 illustrates the configuration of the polygonal pattern in a 100 ft by 100 ft area on the surface of a lava lake, mapped by Peck and Minakami two months after the eruption. Fractures that formed immediately after the eruption (indicated by heavy solid lines) outline polygons that are 15 ft on the average across. Cracks that formed within twenty days after the eruption (light solid lines) divide the original pattern into smaller polygons, 5 to 10 ft across. Later cracks (dotted lines) have opened near the centers of the polygons.

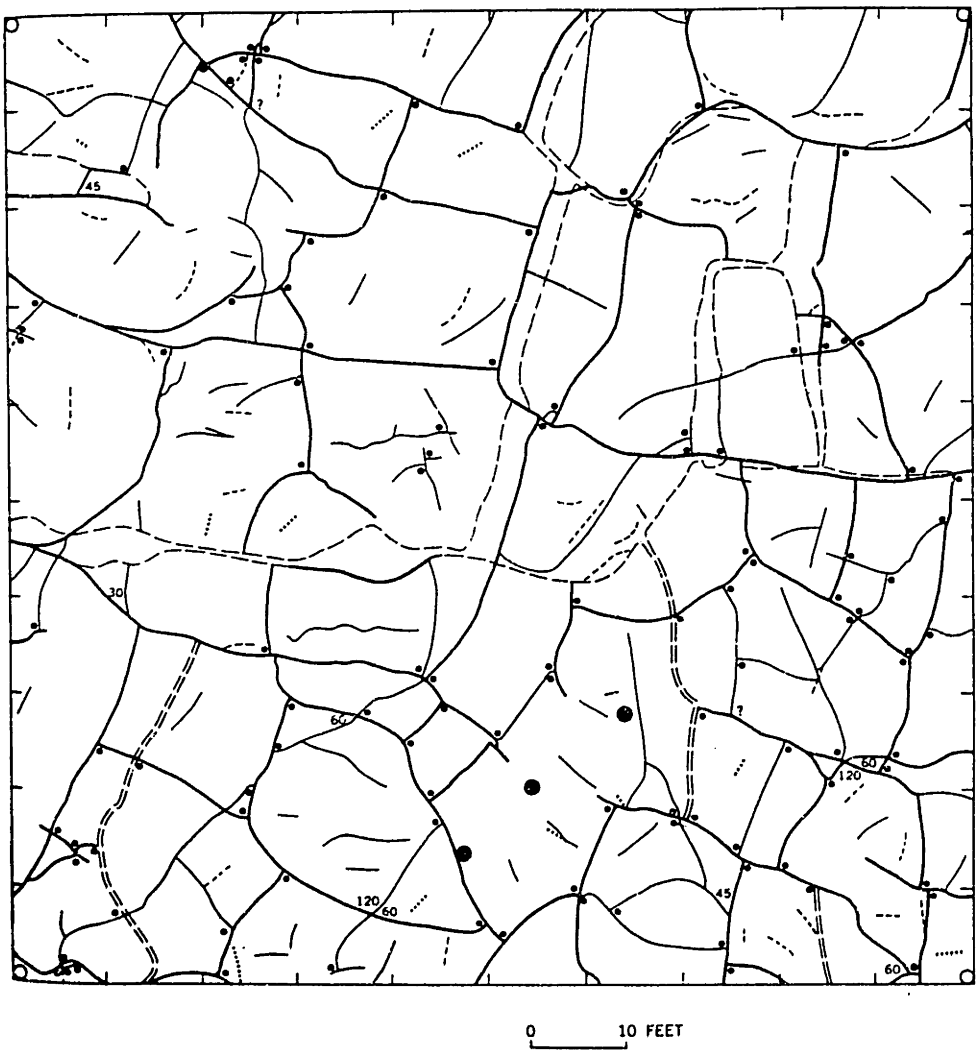


FIGURE 2.4.4 Cracks on a part of the crust of the March 1965 Makaopuhi lava lake, Hawaii [from Peck & Minakami 1968]. Crust formed at 5 pm, March 19. Small dots: orthogonal crack intersections. Heavy solid lines: early formed cracks. Light solid lines: other cracks present on April 8, 1965. Dashed lines: new cracks present on August 13, 1965.

In the base of lava flows fracture initiation is delayed due to slow cooling, absence of strong fluctuating temperatures and gradual build-up of stress throughout a rather thick layer in the lower part of the flow [Suppe 1985; Peck & Minakami 1968; Aydin & DeGraff 1988]. When cracks finally form, they propagate rapidly enough to bifurcate and form nonorthogonal, regular hexagonal and pentagonal columns. For example, **Figure 2.4.5** illustrates a regular hexagonal joint pattern in rhyolite that most likely formed in the interior of a lava flow that was later exposed due to erosion of the overlying rock layers.

According to Aydin & DeGraff (1988), the evolution of tetragonal networks at flow surfaces to hexagonal networks in flow interiors occurs by a gradual change of most orthogonal intersections to intersections of about 120°. Also, as the joints grow toward the flow interior, termination of some joints causes a systematic increase of joint spacing and column diameter. Similar gradual change of the shape and size of the columns occurs upward from the base which is in contact with the country rock and cools faster than the interior. The final product of these evolutionary changes is a joint system that bounds small tetragonal columns at flow surfaces and larger hexagonal columns in flow interiors.

Some authors describe cross joints that open perpendicular to the axes of the columns [Billings 1972; Price & Cosgrove 1990]. The cross joints usually originate near impurities or inclusions in the cooling rock, and often have convex or concave shape (**Figure 2.4.6**). According to Price and Cosgrove, the interior of the columns is free to shorten upon cooling; however, shrinkage of the "skin" layers of the columns is slowed down due to the penetration of water and hydration reactions along the columnar joints. The differential longitudinal shrinkage of the columns results in the opening of cross joints.

2.5 FRACTURE SYSTEMS AROUND CENTRAL INTRUSIVE AND EXTRUSIVE STRUCTURES

The origin of many geologic structures with approximately circular shape in plan view is due to the presence of underlying plastic material, e.g. magma, salt, or clay [De Sitter 1956, Suppe 1985, Price & Cosgrove 1990]. Diapiric structures, for example, are created by the upward flow of plastic material which, due to its low density or high pressure, has the tendency to rise and displace the overlying country rock. Collapse structures, on the other hand, are formed by downward caving-in of rock material into voids created either by volcanism or by solution of underlying beds. Diapiric and collapse structures commonly exhibit symmetry around a vertical axis, i.e. around a center in plan view, hence the name central structures. Typical fracture systems include radial and concentric joints and faults, cross cutting the strata that overlie and surround the rising or subsiding plastic material. The fracturing characteristic of diapiric and collapse structures is reviewed in Section 2.5.1 and Section 2.5.2, respectively.

2.5.1 Diapiric structures and associated fracture systems

Diapiric structures develop when a body of relatively low density rises through the overlying rocks of higher density. The rising plastic material can be rock salt, magma, peat, ice, mud, overpressured clay or shale [Price & Cosgrove 1990]. In the following paragraphs, typical fracture systems, related to the diapiric structures of greatest interest, salt domes and igneous domes, are reviewed.

Salt domes and typical fracturing of the surrounding sedimentary strata

Hundreds of salt domes exist in the Gulf Coast of Louisiana, Texas, and Mexico, in northwestern Germany, and in many other areas in the world. The ability of rock salt to flow and form salt domes (or stocks) is due to a combination of the rheological properties of the material and the geometry of the mother salt layer from which the domes originate [De Sitter 1956; Suppe 1985]. Development of salt domes begins with accumulation of salt in beds underlying sedimentary rocks. When a certain thickening of the salt in one place has occurred, the salt starts flowing from the surrounding area towards the salt stock. Price & Cosgrove (1990) point out that, at temperature of about 250° C in the crust, the salt layer must be at least 500 m thick, and a horizontal hydraulic gradient must be induced within it, before significant lateral migration can occur. The stock is then driven upward by buoyant forces due to the difference between the density of the lighter salt and that of the surrounding and overlying rock strata.

A great variety of shapes of salt stocks has been observed [Halbouty 1967]. **Figure 2.5.1** illustrates the two most commonly encountered shapes: bollard-type and bell-type. In some salt domes the salt stock may pierce completely the overlying sediments and outcrop on the surface. Structurally all salt domes show marked upward bending of the surrounding beds against the salt stock. The beds above the salt dome, when they are preserved, are also domed and stretched [De Sitter 1956; Suppe 1985]. As the dome grows, the stretching of surrounding and overlying strata leads to formation of numerous normal faults as well as some reverse and thrust faults. The faulted rocks around salt domes commonly form traps for hydrocarbon accumulation.

Figure 2.5.2 shows a bollard-type salt dome and the fracturing of the adjacent sedimentary strata, including a system of normal faults that dip toward the salt stock. **Figure 2.5.3** illustrates a more pronounced system of normal faults that has developed in the stretched sedimentary beds overlying a bell-shaped salt dome. In Figure 2.5.3, an earlier set of normal faults (dipping to the right in the cross section) has been offset by a younger set of normal faults (dipping to the left). **Figure 2.5.4** depicts the complex pattern of normal faults, exposed on a horizontal outcrop of the sediments overlying a salt dome. According to De Sitter (1956), the fault types characteristic of salt domes are similar to that observed in peri-anticlines of flexural folds (Section 2.1.1).

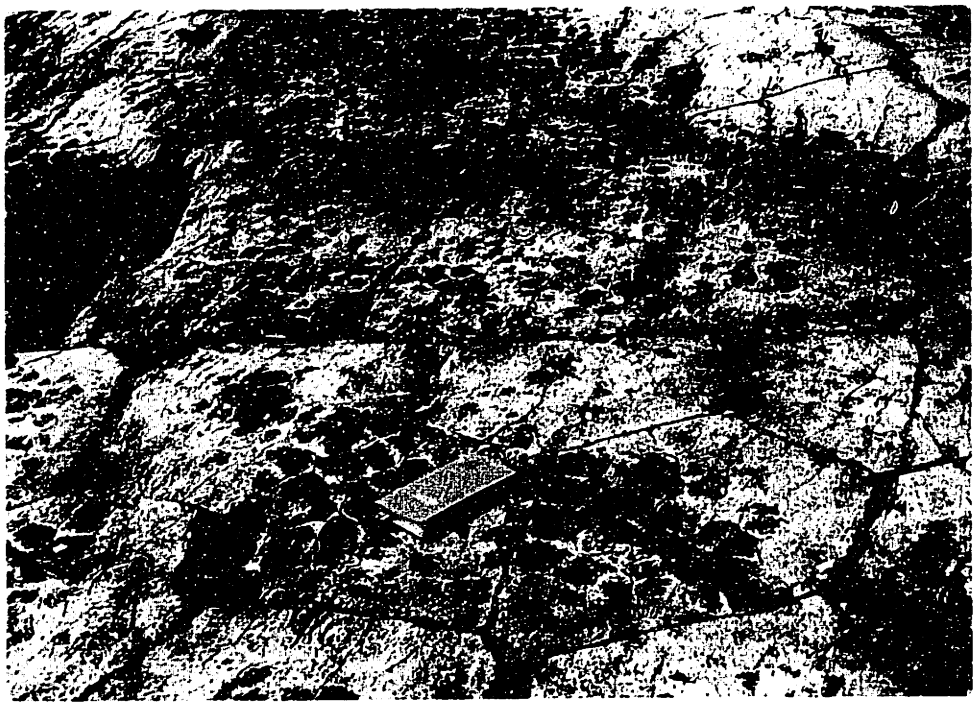


FIGURE 2.4.5 Plan view of mostly hexagonal columns in rhyolite, probably formed in the interior of a lava flow and later exposed through erosion, Rouyn-Noranda, Quebec, Canada [photo by J. Hango].

FIGURE 2.4.6 Columns from the Giant's Causeway, Northern Ireland, showing concave and convex cross fractures [from Price & Cosgrove 1990].

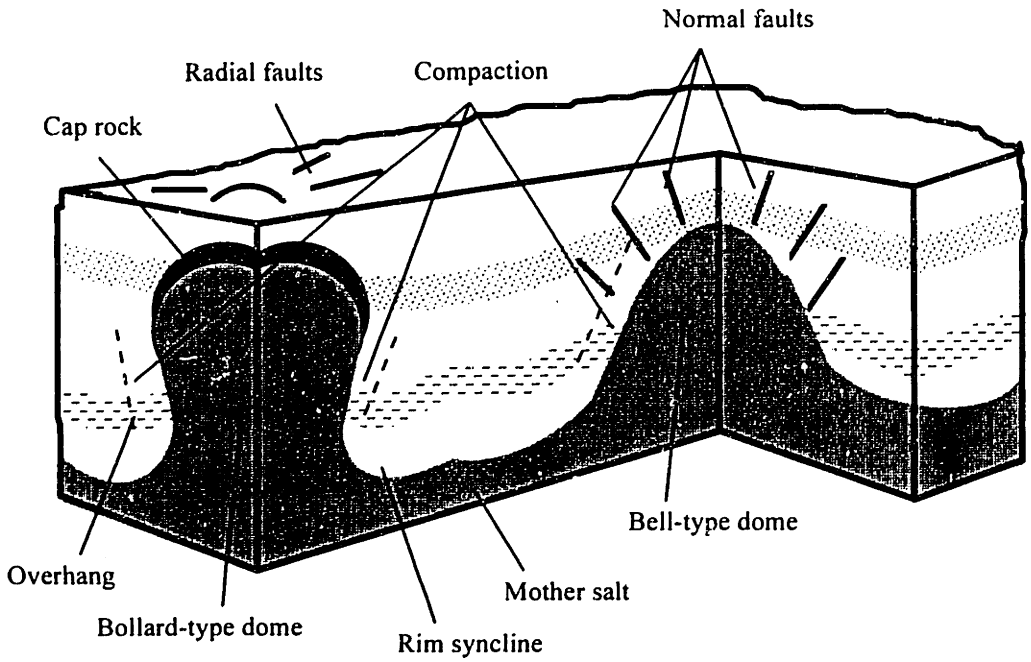


FIGURE 2.5.1 Diagrammatic representation of bollard-type and bell-type salt domes [after Price & Cosgrove 1990].

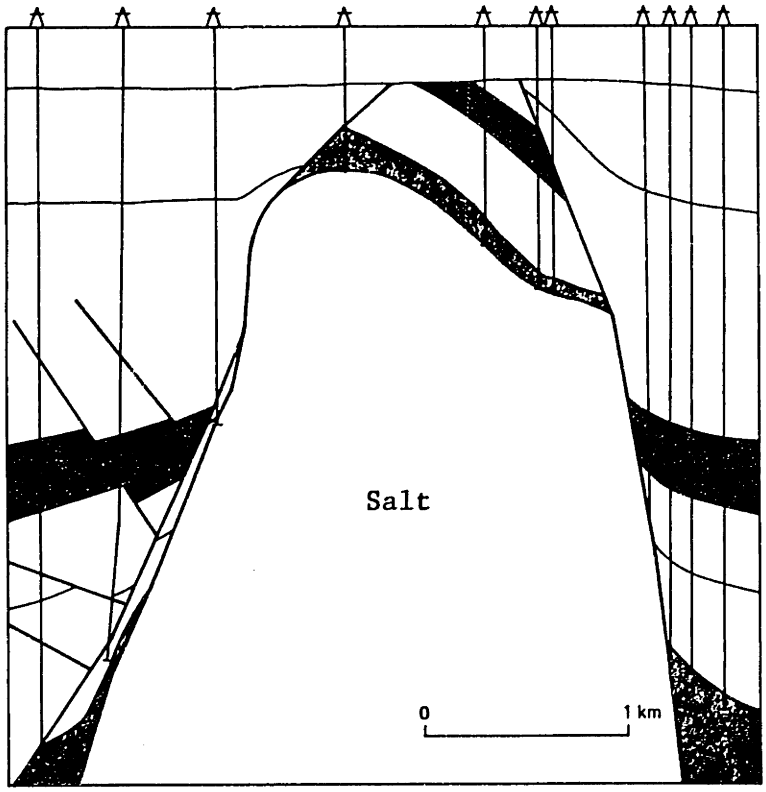


FIGURE 2.5.2 Cross section of the White Castle salt dome, Louisiana [from Suppe 1985].

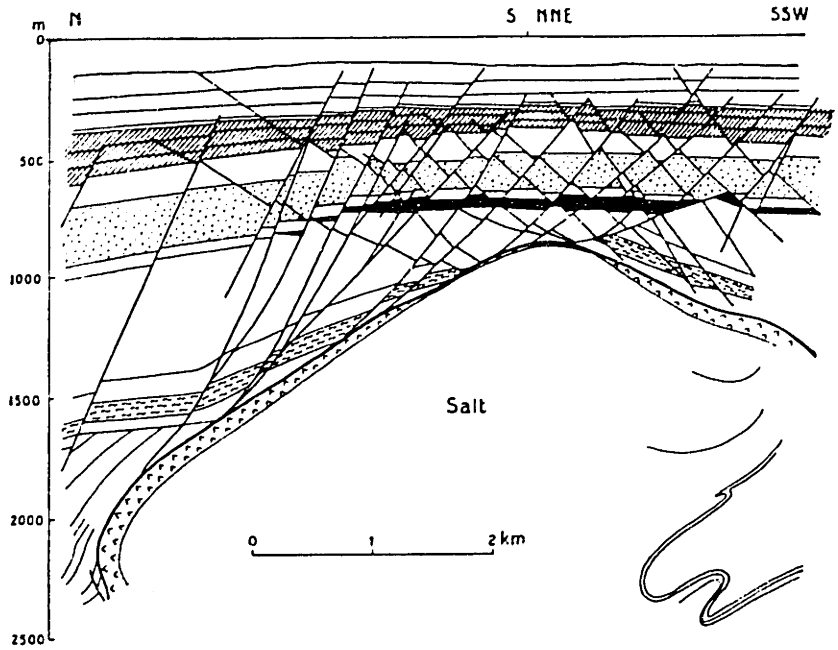


FIGURE 2.5.3 Cross section of the Reitbrook salt dome, Germany [from De Sitter 1956].

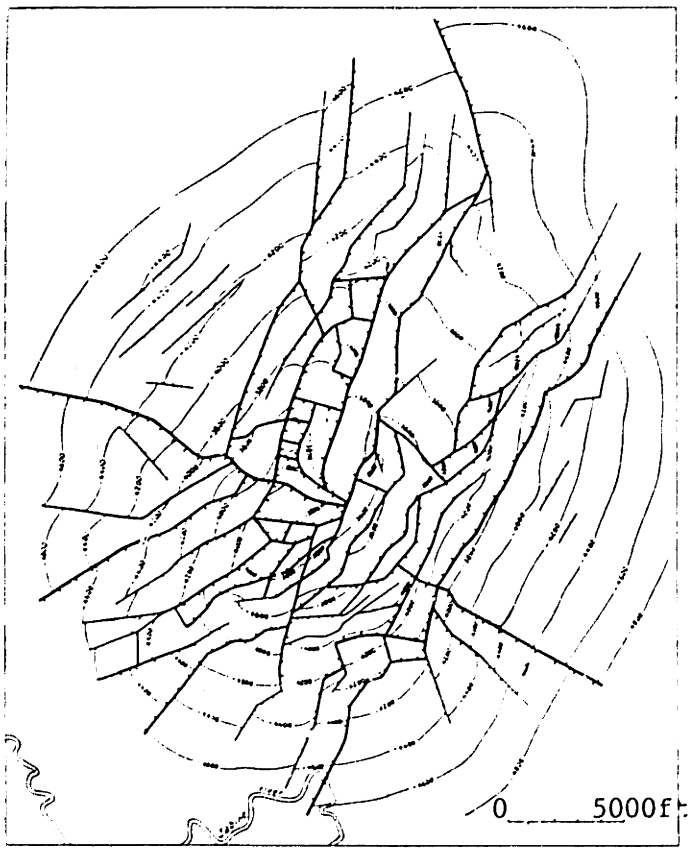


FIGURE 2.5.4 Structure map of the Hawkins Oilfield salt dome, Texas [from De Sitter 1956].

Igneous intrusions and associated fracture systems

Central sub-volcanic intrusive complexes (**Figure 2.5.5**) are structures that include a circular magma chamber and three types of sheet intrusions around it: radial dikes, cone sheets, and ring dikes [De Sitter 1956, Suppe 1985, Price & Cosgrove 1990]. Radial dikes are vertical dikes radiating from the common center (the magma chamber). Cone sheets occupy concentric fissures that are usually concave upward and dip inwards toward the center. Radial dikes and cone sheets are created when the pressure of the magma chamber is increased above that of the overburden. Ring dikes consist of circular segments, concentric around the magma center, either vertical or dipping steeply outward. Formation of ring dikes can be attributed to low chamber pressure and subsidence of the chamber roof (discussed in Section 2.5.2).

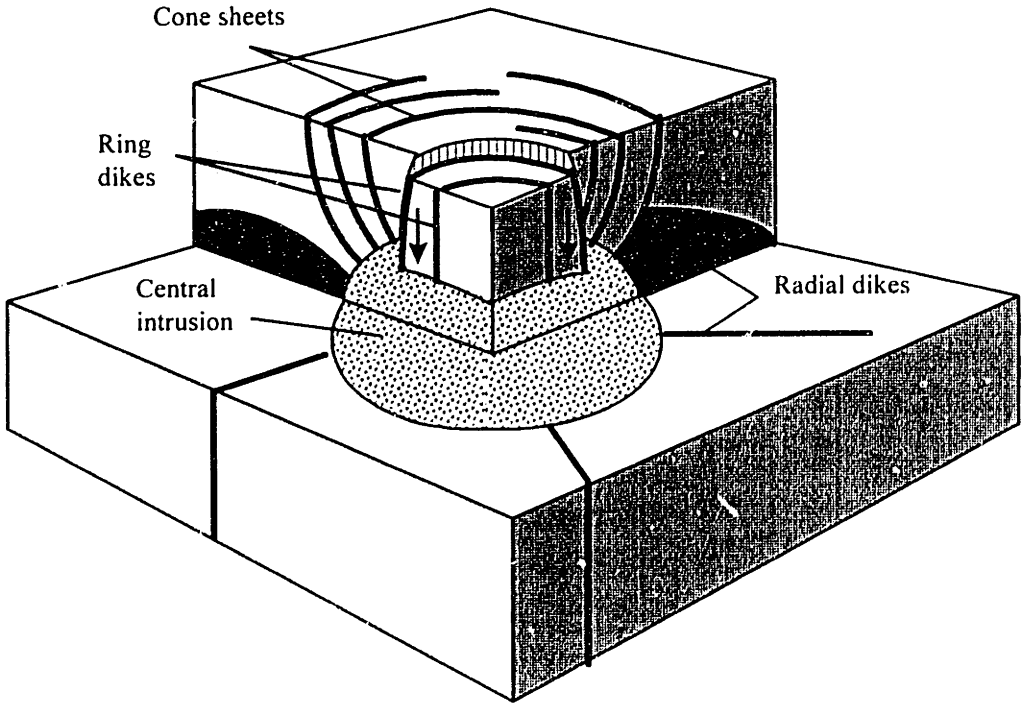


FIGURE 2.5.5 Block diagram showing three types of sheet intrusions around central sub-volcanic complexes: radial dikes, cone sheets, and ring dikes [after Suppe 1985].

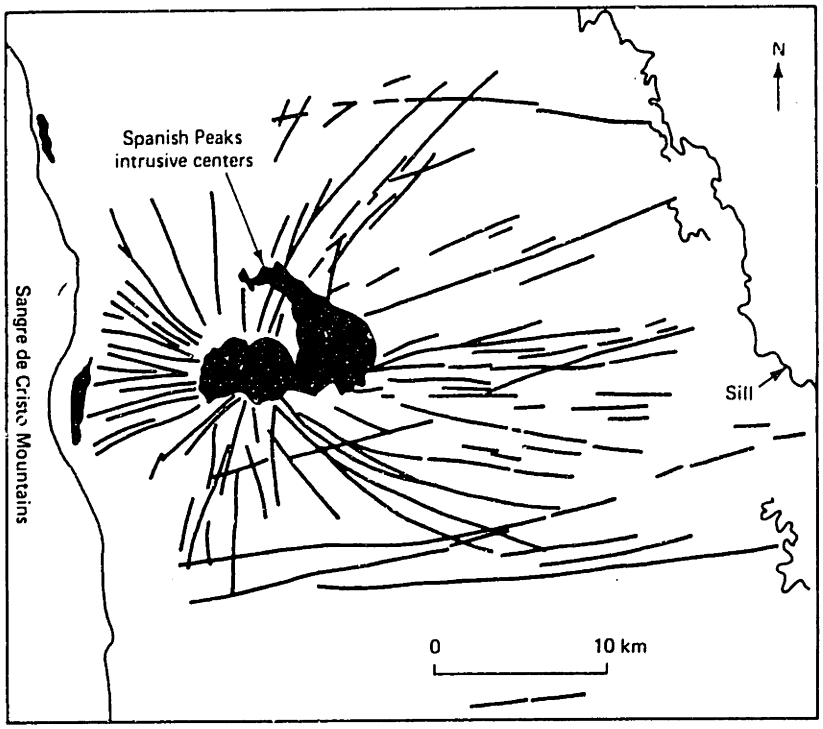
The orientations of sheet intrusions around an igneous center depend on three major factors [Anderson 1951; Suppe 1985]: 1) stress orientations at the earth's surface; 2) regional stress orientations away from the center; and 3) stress orientations at the surface of the magma chamber. When the pressure of the upward pushing magma becomes higher than that of the overburden, tensile stresses are induced in the country rock and the minimum principal stress σ_3 becomes oriented parallel to the chamber wall. If the magma pressure is sufficiently high, sheet intrusions are injected into hydraulic fractures orthogonal to σ_3. In the case when the regional deviatoric stress, $(\sigma_1-\sigma_3)_{regional}$, is small relative to the excess chamber pressure P_m, vertical radial dikes are injected at depth and cone sheets are injected into the roof. In the case when the regional deviatoric stress is relatively large, it affects more strongly the orientation of the sheet intrusions. For example, if $(\sigma_1-\sigma_3)_{regional}>P_m$ and σ_3 is vertical, cone sheets are injected above the chamber, whereas emplacement of sills, parallel to the horizontal beds, is favored at greater depths.

Figure 2.5.6 shows a system of radial dikes around an intrusive center. The pattern of dikes reflects both the regional stress field and the local stresses in the vicinity of the intrusive center. Dikes are most abundant in directions closely parallel to the approximately east-west direction of the maximum compressive stress, σ_1 [Suppe 1985].

Intense fracturing is often associated with central intrusive complexes. The igneous body may fracture upon cooling of the magma, whereas the country rock may fracture due to flexure and stretching of the strata. For example, Price & Cosgrove (1990) describe the mechanism of formation of fractures at the margins of the igneous body with the country rock (**Figure 2.5.7**). The magma which is at a very high temperature cools and solidifies most rapidly adjacent to the cooler country rock. The central part of the intrusion where the magma still flows rapidly upward as a liquid acts against the solidified material and disrupts the outer parts of the intrusion as well as the country rock. Tensile fractures (Figure 2.5.7a) or minor thrust faults (Figure 2.5.7b), respectively, may develop at the margins of the intrusion, depending on whether the intrusion has a constant or increasing diameter upward.

Four major classes of fractures, shown in **Figure 2.5.8**a, have been related to the flow lines of large igneous intrusive bodies [Cloos 1922; Price & Cosgrove 1990]: 1) steep cross fractures, 2) steep longitudinal fractures, 3) flat-lying fractures, and 4) inclined diagonal fractures. Cross fractures, perpendicular to the flow lines, are some of the earliest fractures that form in igneous intrusions. They are usually hybrid extension and shear fractures, often occupied by aplite dikes. Their formation is attributed to drag of the fluid core against the wall and roof of the intrusion, and shrinkage of the solidified magma as it cools. Longitudinal fractures are orthogonal to the cross fractures, and strike parallel to the flow lines. Longitudinal fractures are extensional joints due to cooling of the igneous body.

DIKE PATTERNS AT SPANISH PEAKS, COLORADO



THEORETICAL STRESS FIELD

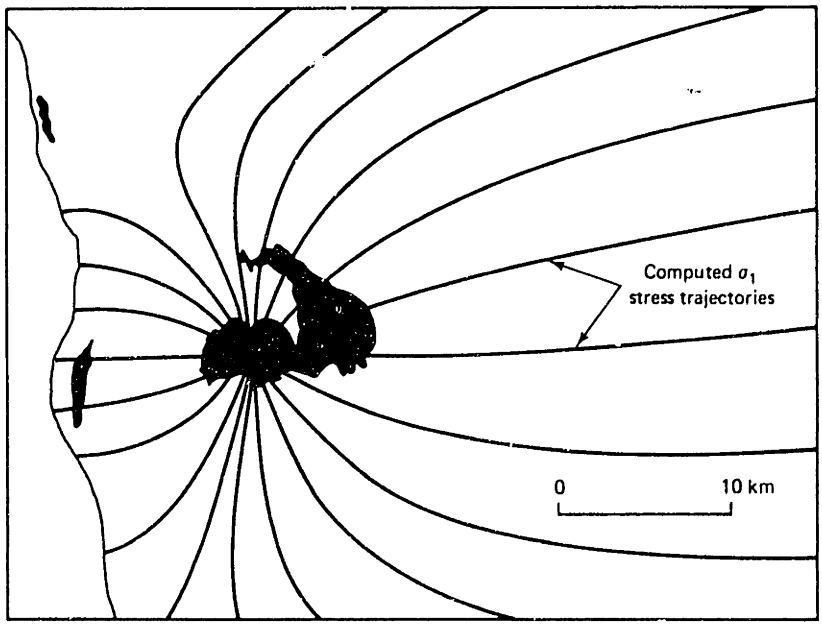


FIGURE 2.5.6 Dike patterns and theoretical regional stress field at Spanish Peaks, Colorado [from Suppe 1985].

a) b)

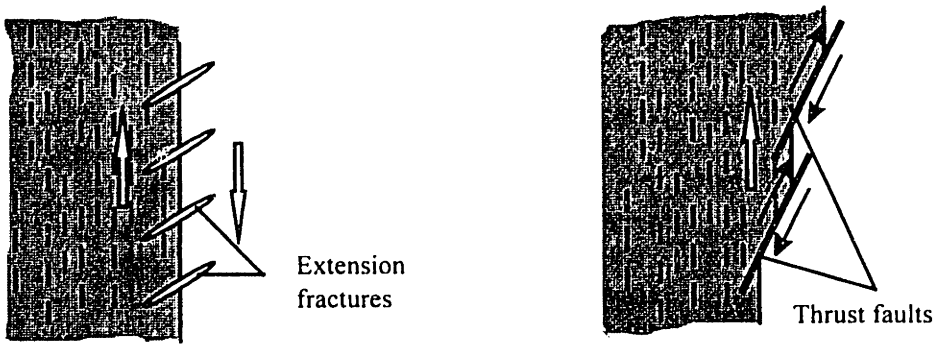
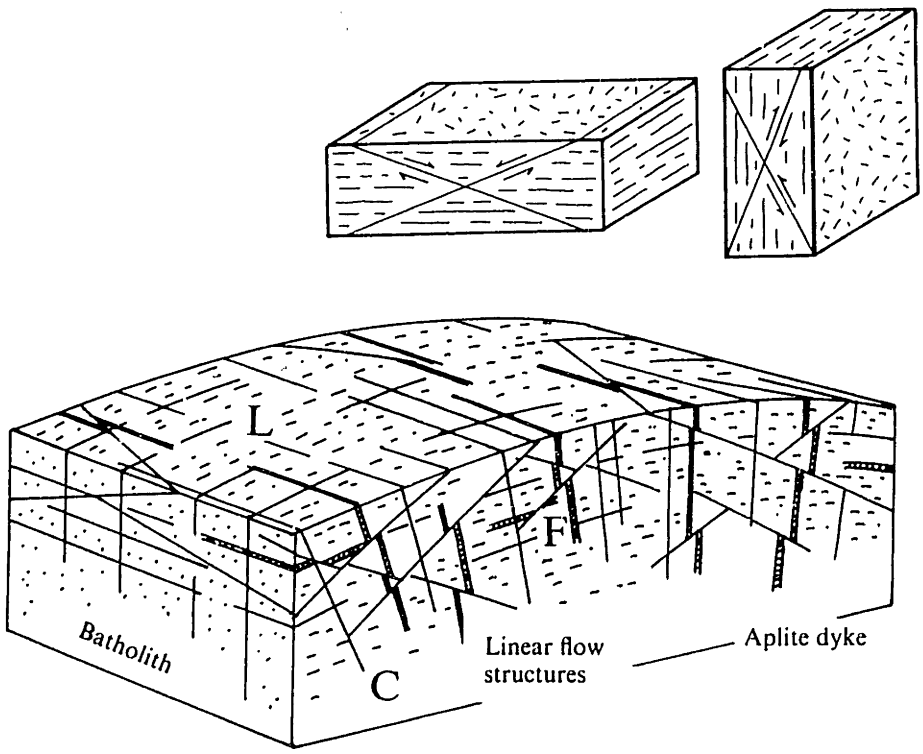


FIGURE 2.5.7 Fractures at the interface between an intrusion and the country rock [after Price & Cosgrove 1990]: a) extension fractures; b) thrust faults.

Flat-lying fractures, abundant predominantly near the apex of the igneous dome, form relatively early as extension joints and are usually filled with aplite. Price & Cosgrove (1990) explain their formation with decrease in the magma pressure after injection of radial dikes and cone sheets or eruption, and "sinking" of the flat-lying, solidified roof into the core magma. Diagonal fractures (inset in Figure 2.5.8) are normal or reverse faults that form an angle of less than 45° with the flow lines. Price & Cosgrove (1990) attribute their formation to ductile deformation of the intrusion when it is subjected to compression orthogonal and extension parallel to the flow lines. Figure 2.5.8b illustrates a system of cooling fractures (mostly joints), similar to the fractures described in Figure 2.5.8a, in a large granitic intrusion (batholith).

Sedimentary rocks, overlying and adjacent to igneous intrusions are often intensely fractured. For example, sheet intrusions such as dikes, are often accompanied by systematic dike-parallel tensile joints. It is important to distinguish between older joints invaded by dikes, and younger joints formed during dike emplacement. Delaney et al. (1986) who mapped joints in the vicinity of vertical dikes in three areas on the Colorado plateau, found out that the intensity of regional and adjacent dike-parallel joints can indicate whether the dikes ascended along older joints, or caused opening of new tensile joints. For example, in **Figure 2.5.9** the intensity of the set of dike-parallel joints is very high in all strata adjacent to the vertical dikes, whereas on a regional scale this set is present only in the limestone. According to Delaney et al., the joints parallel and adjacent to the dikes at this location formed during dike emplacement due to tensile stresses induced in the sedimentary rocks by the wedging action of magma.

a)



b)

FIGURE 2.5.8 Fractures in large intrusive bodies: a) typical orientations and interrelationships between primary fractures and flow lines [from Price & Cosgrove 1990; after Cloos 1922]. C: cross fractures; L: longitudinal joints; F: flat-lying joints; inset: conjugate diagonal faults; b) two joint sets created in response to cooling of granite, Ghottard, Switzerland [photo by H.H. Einstein].

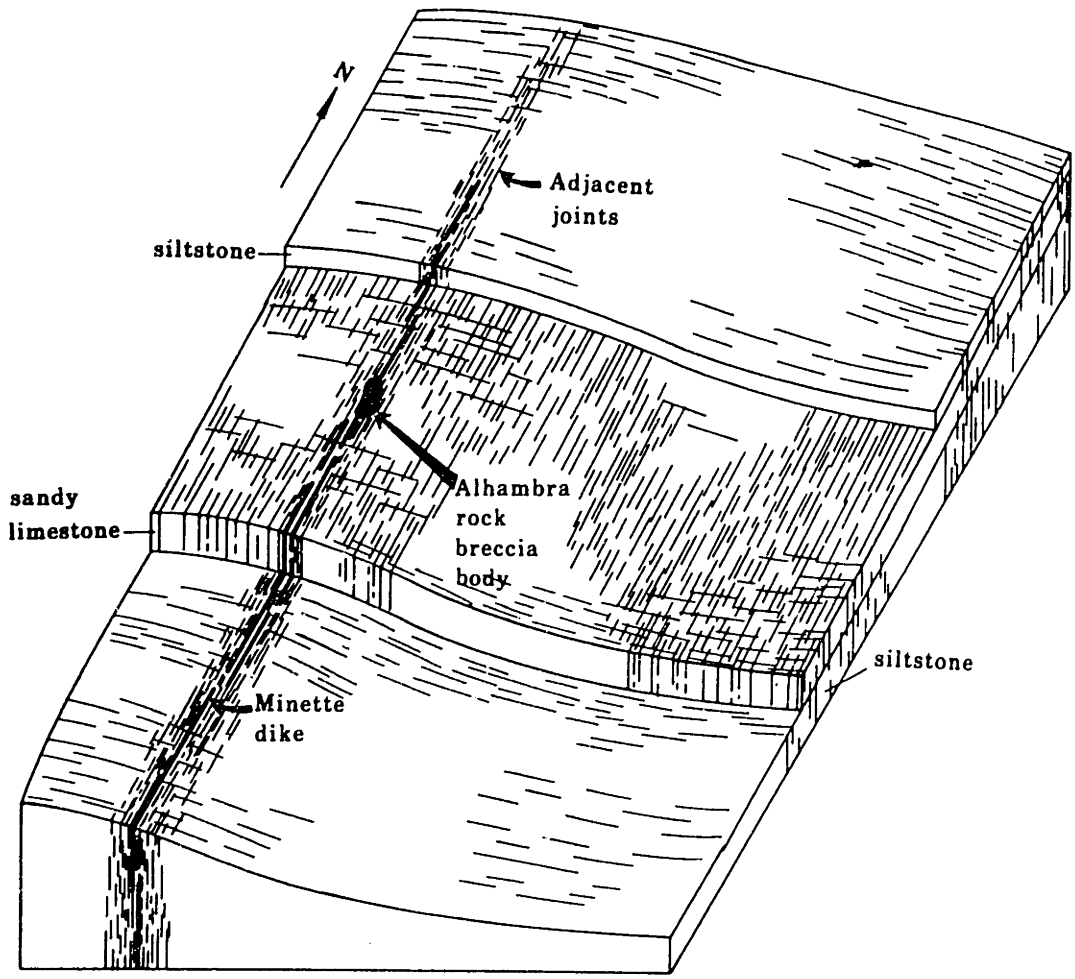


FIGURE 2.5.9 Schematic block diagram showing relations between dikes, systematic joints, and sedimentary strata near Alhambra rock, Utah [from Delaney et al. 1986]. A set of vertical joints are adjacent and parallel to north-south striking dikes that cut upper Pennsylvanian interbedded siltstone, limestone, and sandstone. Away from the dikes, joints of this set are present only in the limestone. An east-west striking regional joint set cuts all strata.

Typical fracturing in sedimentary strata above circular igneous domes is described by Jackson & Pollard (1990). They observe two major types of faults in the sandstone above three domes in the Henry Mountains on the Colorado Plateau: (1) large bedding-plane faults; and (2) smaller, sub-radial, steeply-dipping normal, reverse and oblique-slip faults. The faults, primarily in the form of deformation bands and band faults (discussed in Section 2.2.2), accommodated bending and stretching of the sedimentary host rock as the intrusive domes inflated.

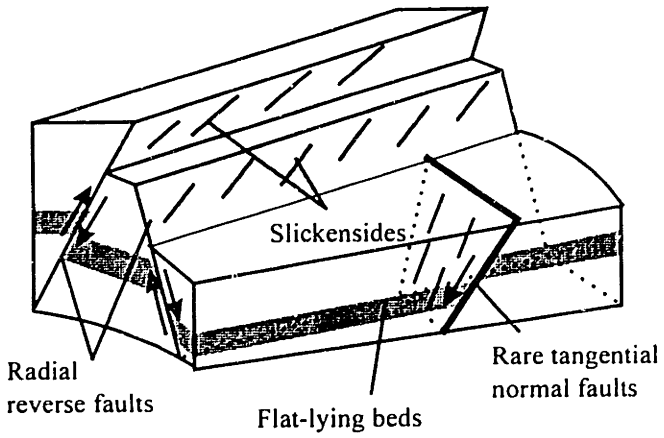
According to Jackson & Pollard (1990), during the initial stage of doming, the 4 km thick overburden behaved as a stack of mechanical layers that slipped over one another. Bedding-plane faults, spaced less than 200 m orthogonal to the beds, developed within the thick sandstone layers and at the formation contacts. Typical bedding-plane band faults in the Henry Mountains consist of up to 1 m thick stacks of deformation bands, some minor bedding-plane deformation bands being present between the thicker zones. The bedding-plane faults have the orientation of the beds which at the initial stage of doming is horizontal at the periphery of the domes, and very gently dipping above the hinge.

During the later, flexural, stage of doming in the Henry Mountains, networks of small (outcrop-scale) normal, reverse and oblique-slip faults developed and offset most bedding-plane faults [Jackson & Pollard 1990]. The small faults dip steeply and strike radially from the central intrusion. The greatest density of small radial faults occurs near the bedding-plane faults, whereas there are only a few or no faults within the thick sandstone layers. The faults cross cut at high angles the sandstone beds which strike tangentially to the dome and generally dip gently except for where they are bent against the dome. **Figure 2.5.10** depicts typical orientations of radial faults relative to (a) flat-lying, and (b) more steeply-dipping sedimentary beds. The outcrop traces of the small faults are few meters to several tens of meters long, 0.5 mm to 1 cm wide, and offset from 0.01 to 1 m. The faults are composed of en echelon segments, tens of centimeters to many meters long. The segments link through multiple secondary splay fractures to create very complicated outcrop patterns (**Figure 2.5.11**).

2.5.2 Subsidence of rock masses and associated fracture systems

Calderas are circular collapse structures of volcanic origin. They form through subsidence or collapse of the roof of a magma chamber when the pressure of the magma falls below that of the surrounding rock. The magma pressure decreases either after eruption of magma, ash, or other volcanic material to the surface, or because of emplacement of radial dikes, cone sheets and sills during preliminary magma rise. The roof collapses along outward-dipping shear fractures which are often intruded by ring dikes (Figure 2.5.5). **Figure 2.5.12** illustrates a system of ring dikes and cone sheets in Scotland, concentric around an intrusive center. Most of the ring dikes are vertical or dip steeply about 70° to 80° to the southwest, i.e. outwards from the intrusive center [De Sitter 1956; Suppe 1985].

a)



b)

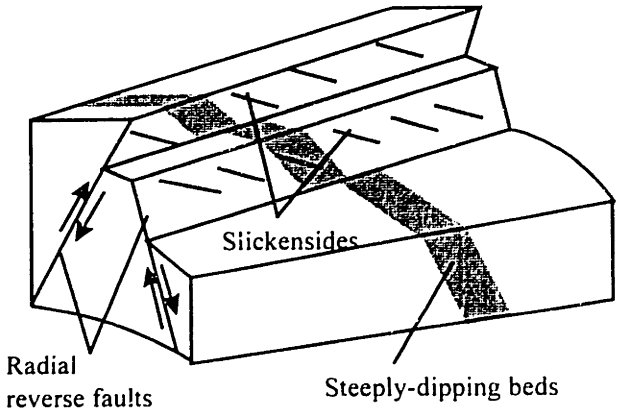


FIGURE 2.5.10 Block diagrams illustrating typical orientations of radial faults relative to the sedimentary beds [from Jackson & Pollard 1990]: a) Mount Holmes, Henry Mountains: initial phase of doming. The beds dip gently. The high-angle radial faults have slickenlines that plunge in the direction of the bed normal. Rare tangentially striking faults have normal or reverse displacements. b) Mount Ellsworth and Mount Hillers, Henry Mountains: advanced doming. The beds dip more steeply. The slickenlines plunge approximately parallel to the dip direction of the beds.

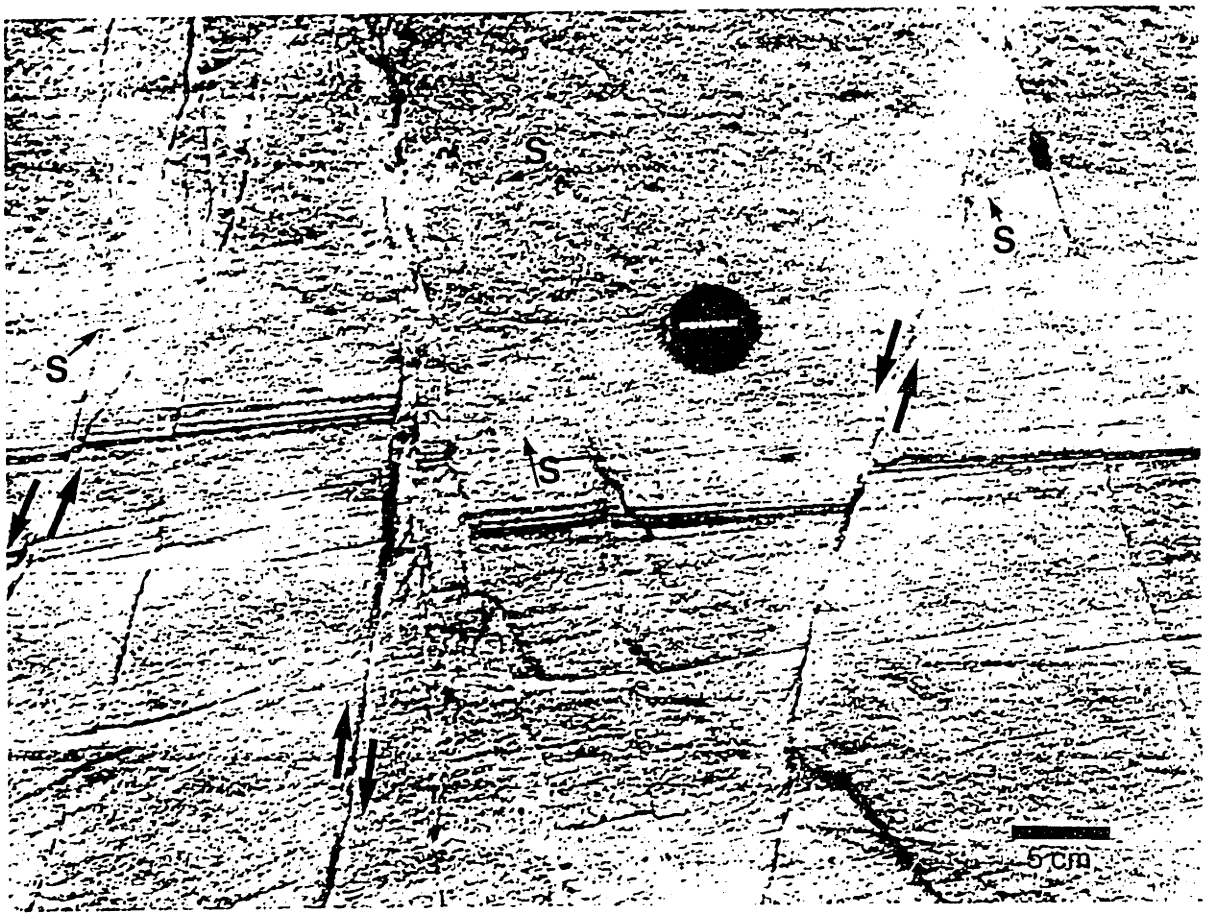


FIGURE 2.5.11 Network of radial faults within the Kayenta Formation, Mount Ellsworth, Henry Mountains [from Jackson & Pollard 1990]. Many splay faults (S) diverge from the primary fault planes.

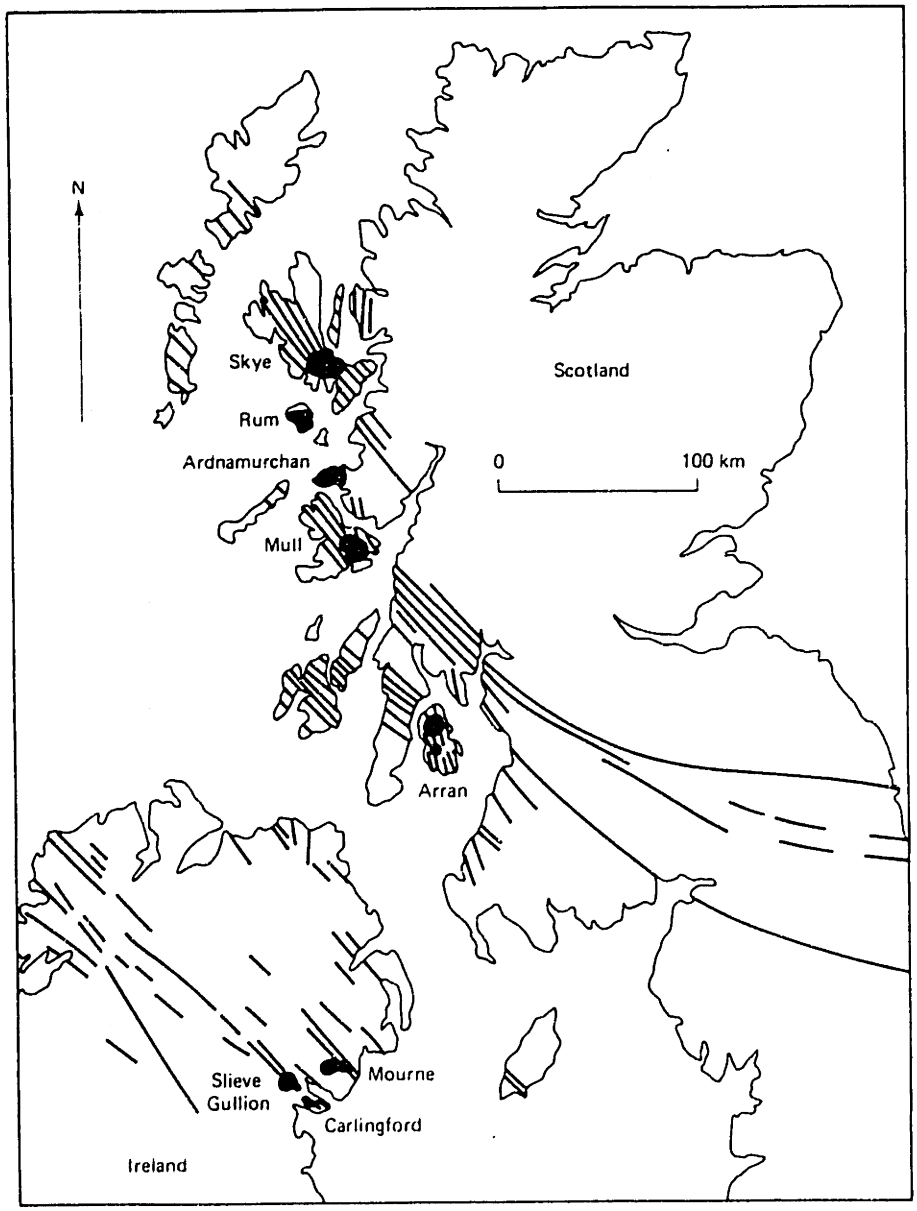


FIGURE 2.5.12 Distribution of early Tertiary basaltic ring dikes and cone sheets in central intrusive complexes in Scotland [from Suppe 1985].

The chamber collapse is often accompanied by intense normal faulting of the rocks above and around the sub-volcanic igneous center [Suppe 1985]. The normal faults, vertical or steeply-dipping inward, tend to strike concentrically to the igneous center. Since normal faults may also be produced due to stretching of the country rock strata during initial rise of the magma, the net result of the cycles of magma influx, eruption, and collapse is often an intense system of normal faults in the vicinity of an igneous center. **Figure 2.5.13** shows a caldera complex with high density of normal faults with various strikes in a roughly circular area, in contrast to the regional pattern of north-trending normal faults.

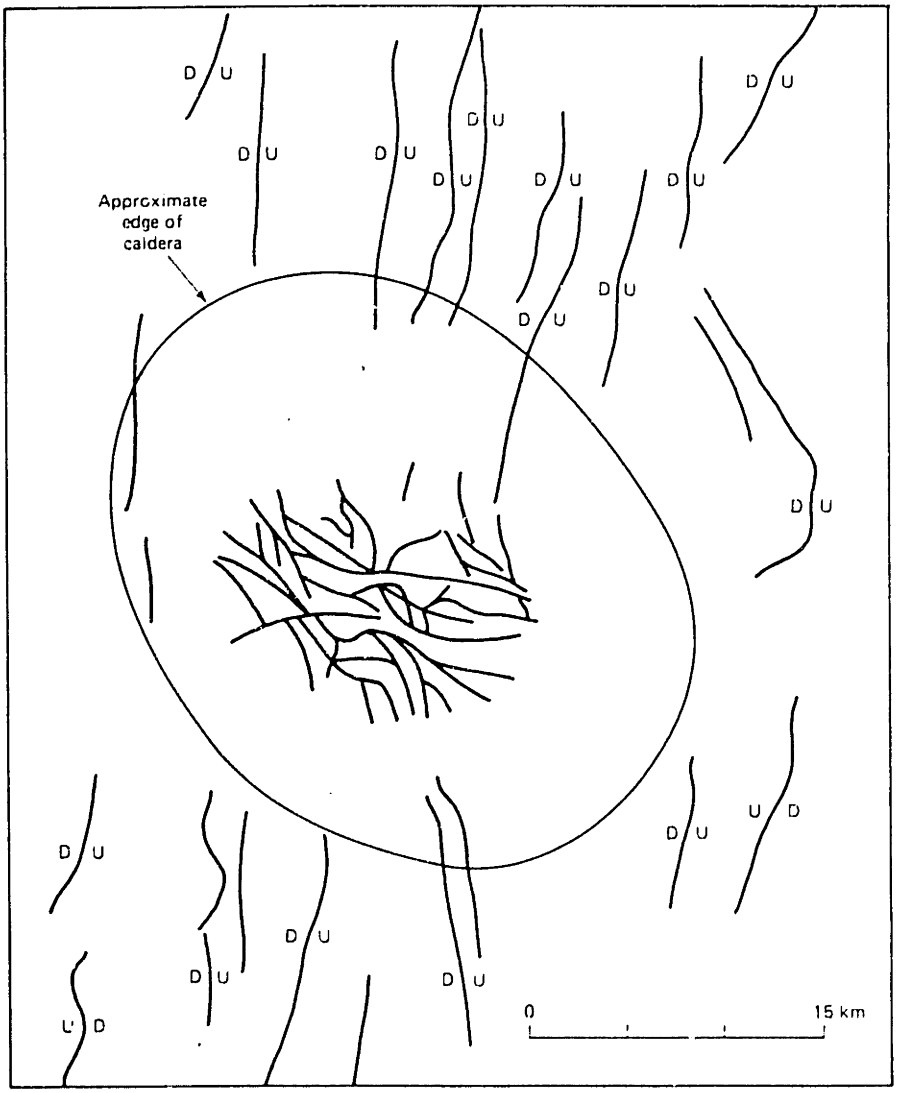


FIGURE 2.5.13 Map of Tertiary normal faults in the vicinity of Timber Mountain caldera, Nevada [from Suppe 1985].

Collapse structures may also develop through subsidence due to solution of limestone, gypsum, salt, or other soluble rock, differential compaction of shales, melting of ice, or landslides [De Sitter 1956; Suppe 1985]. For example, McDonald & Shilts (1973) describe intense normal and reverse faulting that occurred in fluvio-glacial and glacio-lacustrine sediments as they collapsed into voids created by molten ice. Fracture systems associated with all subsidence and collapse structures commonly involve numerous normal faults, concentric to and dipping towards the collapse center. For example, Figure **2.5.14** illustrates the system of normal faults that has developed in response to a regional scale sliding of coastal sediments down toward the Gulf of Mexico.

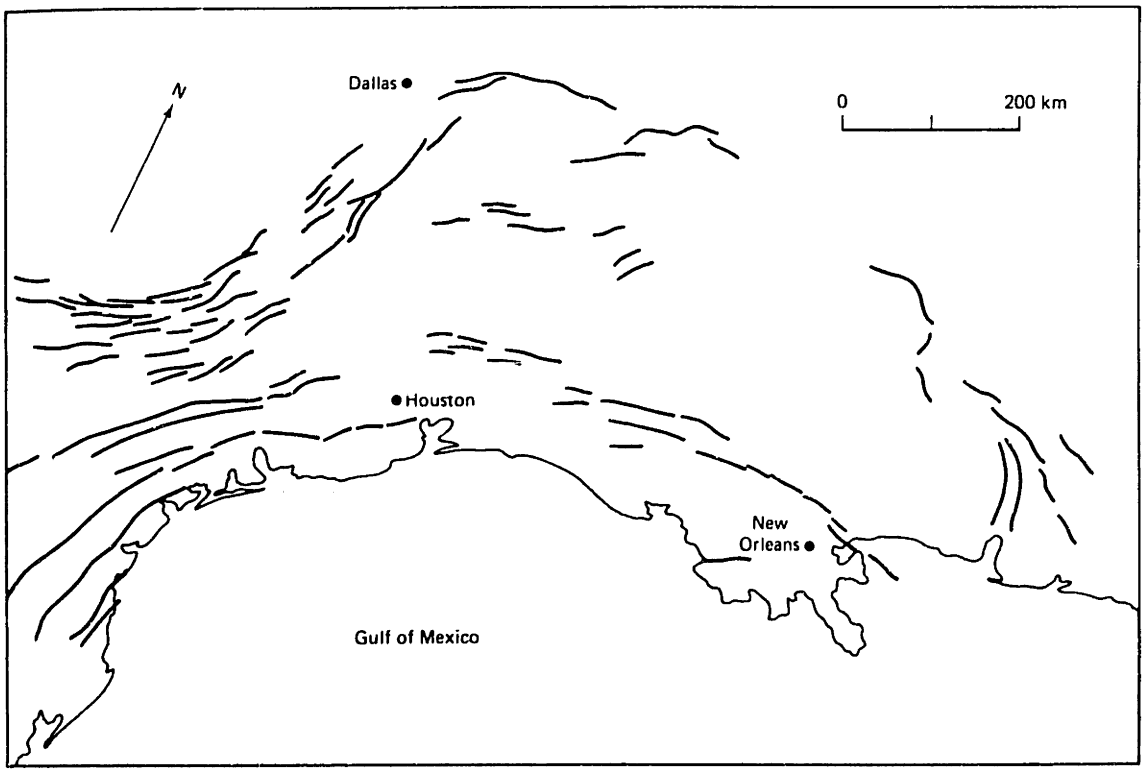


FIGURE 2.5.14 Map of normal faults in the sediments of the U.S. Gulf Coast [from Suppe 1985]. Most of the slip is down toward the Gulf of Mexico. A number of smaller antithetic faults also exist. The normal faulting is the result of large-scale sliding of continental-margin sediments out into the Gulf of Mexico.

2.6 SUMMARY OF THE GEOMETRY OF NATURAL FRACTURE SYSTEMS

A great number of natural fracture systems were reviewed in the preceding sections of Chapter 2. Five major geologic settings of fracturing have been identified, every one representing a unique stress field and resulting in a characteristic fracture system geometry. The classification of fracture-producing environments has been performed by giving primary consideration to the final geometric relationships between fractures and geologic structures (rather than from the point of view of the structural geologist). **Table 2.6.1** summarizes the most important geometric characteristics of the fracture systems associated with the five natural geologic settings.

GEOLOGIC SETTING	TYPICAL FRACTURE SYSTEM GEOMETRY	REFERENCE IN THE THESIS
I. ***Folds***	Fracture orientations and intensity are function of location relative to a curved fold surface.	Section 2.1
II. ***Crustal faults***	Fracture orientations are related to the primary shear direction. High fracture intensity occurs only within fault zones, enclosed between boundary faults.	Section 2.2
III. ***Remote tension***	Fractures are subparallel to one another and orthogonal to the remote tension direction.	Section 2.3
IV. ***Thermal contraction***	Fractures are orthogonal to the margins of a cooling rock body, and form polygonal patterns in plan view.	Section 2.4
V. ***Central intrusive and extrusive structures***	Fracture orientations are radial or concentric around an axis through the central intrusion, or related to radial and concentric structures (e.g. dikes).	Section 2.5

TABLE 2.6.1 Major geologic settings of fracture formation and typical geometry of the characteristic fracture systems.

Natural fracture systems consist of one or multiple sets of tensile fractures (joints) and shear fractures (faults). A fracture set is defined as an assembly of joints or faults with related orientations, shapes, and locations, occurring within a rock formation where a particular stress field determines the variation of fracture orientations and intensity. This section presents a simple classification of natural fracture sets according to three important geometric characteristics: region of existence within the rock formation (Section 2.6.1); fracture orientations (Section 2.6.2); and fracture intensity (Section 2.6.3). Section 2.6.4 contains concluding remarks on the geometry of natural fracture systems.

2.6.1 Types of fracture sets according to the region of existence

A fracture set develops within a portion of a geologic formation, roughly enclosed between several planar or non-planar surfaces. For every fracture set it is important to identify where exactly it forms in relation to other geologic structures. Two major types of fracture sets can be distinguished: (1) primary sets which are created either in intact rock, or without relation to other fracture sets in previously fractured rock; and (2) secondary sets which develop in close relation to primary sets.

Examples of primary sets are: bedding-parallel faults in folded competent beds (discussed in Section 2.1.1); subparallel joint sets in a rock formation subjected to remote tension (Section 2.3.1); columnar joints in igneous rocks (Section 2.4); etc. The region of existence of a primary set is defined by the boundaries of the geologic structure. For example, columnar joints form within the entire volume of the cooling lava flow. The region of existence of a primary set is independent of other fracture sets.

Examples of secondary sets are: antithetic faults dipping against normal faults (Section 2.2.1); splay fractures and secondary faults adjacent to major strike-slip faults (Section 2.2.2); en-echelon cracks originating from a parent tensile joint (Section 2.3.2), etc. The region of existence of a secondary fracture set is defined by one or several primary fracture sets. Some secondary sets are parallel to the previously formed fractures; for example, numerous small cracks are sub-parallel to outcrop-scale fractures in a remote tension field. Other secondary sets develop at an angle to the primary fractures and may terminate at the intersections with them; for example, oblique fractures in strike-slip shear zones usually terminate at the boundary faults. In either situation, secondary sets develop only in the vicinity, and sometimes enclosed between the fractures of a primary set.

2.6.2 Types of fracture sets according to fracture orientations

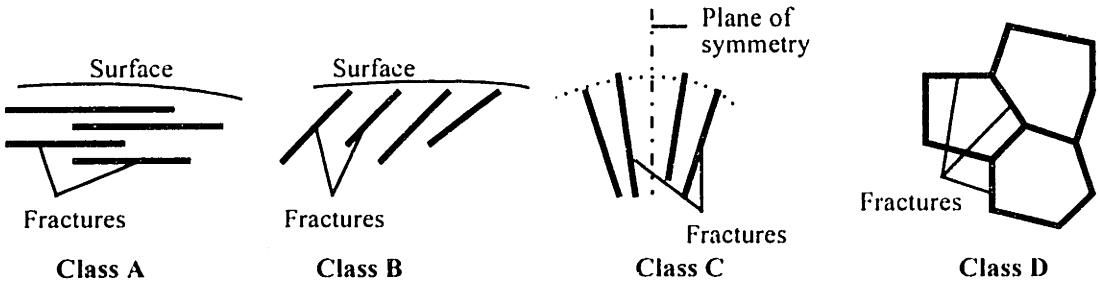
In the geologic literature a fracture set is often defined as a group of subparallel fractures. In the present classification fracture sets are identified on the basis of

common origin and characteristic geometry, rather than in terms of orientation alone. In this respect, some fracture sets indeed consist of subparallel fractures; however, many others are made up of non-parallel fractures, usually with orientations symmetric around a plane or an axis. In some cases it may even be appropriate to assign subparallel fractures to different sets, based on the geologic sequence of fracture genesis. **Table 2.6.2** and **Figure 2.6.1** describe six fracture set classes in terms of characteristic relationships of the orientations of fractures that belong to a given set to one another and to major geologic axes and surfaces.

CHARACTERISTIC VARIATION OF FRACTURE ORIENTATIONS WITHIN A SET	
GROUP I: FRACTURE SET ORIENTATIONS RELATED TO A PLANE OR SURFACE	
Class A	The fractures within the set are subparallel to a plane or surface. ***Example:*** Sheet joints, parallel to the topographic surface.
Class B	The subparallel fractures of the set are oblique (possibly orthogonal) to a plane or surface. ***Example:*** Secondary fractures in strike-slip faults oblique to the boundary faults.
Class C	The orientations of fractures within the set are symmetric around a plane. ***Example:*** Tensile longitudinal joints on an anticline crest symmetric around the fold axial plane.
Class D	Special case: sets of non-parallel connected fractures, perpendicular to a plane or surface. ***Example:*** Columnar joints in igneous rocks, orthogonal to the margins of the cooling igneous body.
GROUP II: FRACTURE SET ORIENTATIONS SYMMETRIC AROUND AN AXIS	
Class E	Fracture sets, striking radially from an axis. ***Example:*** Radial dikes, originating from a central igneous intrusion.
Class F	Fracture sets, concentric around an axis. ***Example:*** Cone sheets and ring dikes, concentric around a central igneous intrusion.

TABLE 2.6.2 Fracture set classification in terms of variation of fracture orientations within the set.

GROUP I: FRACTURE SET ORIENTATIONS RELATED TO A PLANE OR SURFACE



GROUP II: FRACTURE SET ORIENTATIONS SYMMETRIC AROUND AN AXIS

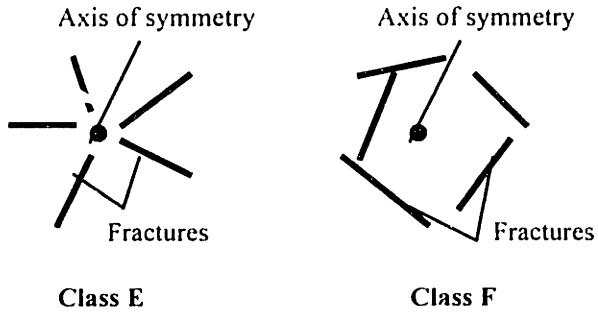


FIGURE 2.6.1 Fracture set classification in terms of variation of fracture orientations within the set.

2.6.3 Types of fracture sets according to fracture intensity

The intensity of a fracture set is a measure of the ratio of fractured and intact rock. A convenient expression of fracture intensity is, for example, the ratio of the cumulative fracture area over the volume of the rock mass where this set has developed (Dershowitz & Herda 1992). One can distinguish three types of fracture intensity (**Figure 2.6.2**): 1) low; 2) high, due to a small number of large fractures; and 3) high, due to a large number of small fractures.

The intensity of a fracture set is low only if the set consists of a small number of small fractures. An example of a low intensity fracture set is a set of small cracks in the rock between outcrop-scale fractures in remote tension (Section 2.3.1). Such small cracks are parallel to the outcrop fractures and do not interconnect.

On the other hand, a rock may be highly fractured either through a system of large fractures, or though a very large number of small fractures. Examples of high intensity sets of large fractures are: bedding-plane faults in concentric folds in competent beds (Section 2.1.1); cleavage on the crests of shear folds in incompetent beds (Section 2.1.2); sheet joints (Section 2.3.3), etc. An example of

a fracture set of high intensity due to a large number of fractures, whereas the size of an individual member is small compared to the entire fractured rock volume, is a set of bed-orthogonal joints in thin-bedded, competent rock unit. The joint size and spacing are approximately equal to the bed thickness (Section 2.3.1).

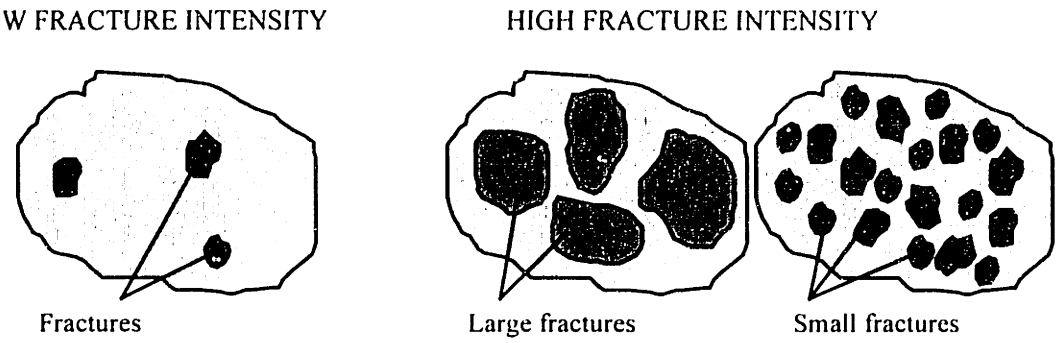


FIGURE 2.6.2 Types of fracture sets according to fracture intensity.

2.6.4 Conclusion

Natural fracture systems generally evolve through sequential genesis of fracture sets in heterogeneous and anisotropic rocks, subjected to stresses that vary in time and space. The final geometry of a natural fracture system is usually very complex and includes hierarchical relationships between fracture sets in terms of their orientation and intensity. Field exposures are often very limited. Therefore, modeling of natural fracture systems usually involves making many assumptions about their true three-dimensional geometry.

The classification of natural fracture systems, presented in this chapter, can facilitate mathematical modeling and numerical simulation of natural rock fracture systems in two major ways. First, identification of the geologic setting can be used for reliable assumptions about the existing fracture sets, and about their intensity and orientation. Second, a natural fracture system can be more easily modeled through a superposition of hierarchically related fracture sets, each of which is generated according to its major geometric properties: region of existence, variation of orientations, and intensity. Chapter 3 presents a three-dimensional, geology-based, hierarchical, stochastic model with which one can reproduce the geometry of natural fracture systems.

Chapter 3

THE THREE-DIMENSIONAL GEOLOGIC STOCHASTIC MODEL OF ROCK FRACTURE SYSTEMS

This chapter presents the three-dimensional model proposed in this thesis for reproducing geologic fracture systems. The model is based on previous developments that followed a detailed review of earlier 2D and 3D fracture system models and their advantages and shortcomings [Ivanova 1995]. This chapter contains a summary of the original model and detailed explanation of the new developments. Section 3.1 presents the basic concepts of the model. Section 3.2 describes the generation of an individual stochastic fracture set via algorithms related to the geometry of natural rock fracture systems. Section 3.3 explains the modeling of fracture systems by superposition of numerically generated fracture sets, and presents specific algorithms for modeling of fracture systems in the major geologic settings, discussed in Chapter 2. Section 3.4 contains concluding remarks on the capability of the model to represent fracture systems in nature.

3.1 BASIC CONCEPTS

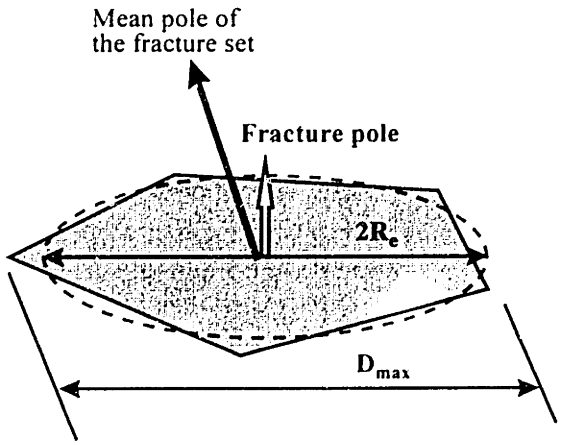
The conceptual model, presented in this chapter, is a three-dimensional geometric-mechanical model of rock fracture systems. The model employs simple geometric procedures to reproduce the 3D geometry of fracture systems, created by complex mechanical processes, on the basis of inherent relationships between mechanics and geometry (characteristic geometries of fracture systems in different geologic environments were reviewed in Chapter 2). Thus, the model combines the advantages of the purely geometric and the purely mechanical models. On the one hand, it has the capability of geometric models to reproduce complex fracture systems in 3D space by using various statistical and geometric methods. On the other hand, geology-based geometric algorithms provide a realistic representation of natural fracture systems, similar to mechanical models that reproduce the true mechanisms of fracture initiation and propagation.

The conceptual model is hierarchical. Based on the field data and the known geologic history of the region, fractures are grouped into hierarchically related fracture sets. Primary fractures that originate either in intact rock, or without relation to other sets in previously fractured rock, are grouped into independent sets. Fractures that are created as secondary, tertiary, etc. in relation to earlier ones with respect to location, orientation, size or other geometric characteristics are grouped into dependent sets. A rock fracture system is represented in three dimensions through superposition of hierarchically related fracture sets.

Finally, the model is stochastic. The model represents natural fracture systems on the basis of the available (usually very limited) geologic information. Fracture

systems in nature consist of multiple interconnected fractures, only a small percentage of which can be sampled with field methods. That is why statistical methods in general have proved to be particularly good for modeling purposes. The proposed model has the additional advantage of using stochastic procedures that represent important relationships between fracture system geometry and underlying mechanics.

In the model, fractures are convex polygons (**Figure 3.1.1**), randomly oriented and randomly located in three-dimensional space. The pole orientation and the coordinates of the center and the vertices of an individual fracture are determined indirectly when the fracture is generated as a member of a set (details follow in Section 3.2). The model incorporates Poisson plane and line stochastic processes, and is thus related to earlier 3D fracture network models [Veneziano 1978; Dershowitz 1979]. Good references on the fundamental theory of Poisson point, line, and plane processes are Diggle (1983), Stoyan et al. (1987), Upton & Fingleton (1985), Miles (1969 and 1973).



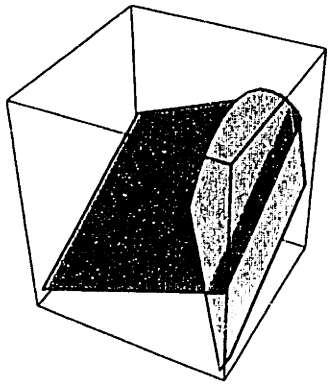
Elongation:

$$e = \frac{D_{max}}{2R_e}$$

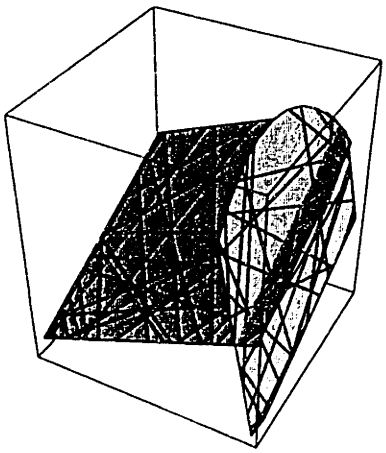
FIGURE 3.1.1 Dimensions and shape of an individual fracture in the 3D model.

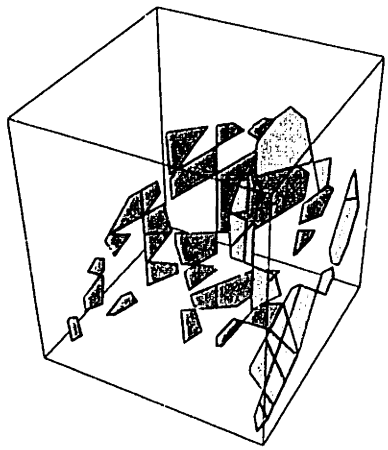
Figure 3.1.2 illustrates the three major stochastic processes of the conceptual model. The primary process (Figure 3.1.2a), a random plane network, models the orientation of the stress field which creates fractures along planes of maximum shear and tension in rocks. The secondary process (Figure 3.1.2b), subdivision of the potential fracture planes into intact and fractured regions, is accomplished through a line tessellation and a polygon marking procedure. The combined primary and secondary processes reproduce fracture intensity. The tertiary process (Figure 3.1.2c), random translation and rotation of polygon-fractures, represents the relationship of fracture intensity and orientation to local structures that modify the general stress field. The three stochastic processes are explained in detail in Section 3.2 in the context of fracture set generation with the 3D model.

a) PRIMARY PROCESS: POISSON PLANE NETWORK

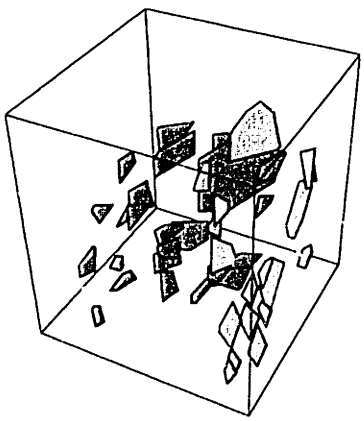


b) SECONDARY PROCESS: POISSON LINE TESSELLATION AND POLYGON MARKING





c) TERTIARY PROCESS: RANDOM POLYGON TRANSLATION / ROTATION



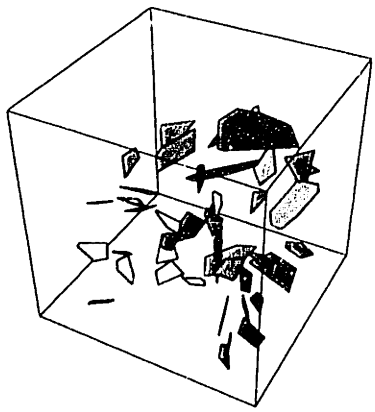


FIGURE 3.1.2 Stochastic processes of the 3D fracture system model (computer generation with program GEOFRAC): a) primary process, Poisson plane network; b) secondary process, subdivision of planes through Poisson line tessellation and polygon marking; c) tertiary process, random rotation and translation of polygons.

3.2 FRACTURE SET MODELING

Geometrically, a fracture set is a collection of fractures with related orientations, sizes, and locations. A fracture set is characterized by two parameters:

- probability density function (PDF) describing the variation of fracture plane orientations;
- intensity of fracturing as it varies in space.

The geometry of a fracture set is reproduced though the three stochastic processes defined in Section 3.1.

3.2.1 Modeling of stress field orientation: primary stochastic process

Stress field orientation is defined as the orientations of planes of maximum shear and tension that are related to fracture plane orientations. The orientations of the potential fracture planes are represented by the primary stochastic process: a homogeneous, anisotropic, Poisson plane network (Figure 3.1.2a).

Frame of reference and orientations of potential fracture planes

The mean orientation of a fracture set is specified in polar coordinates in terms of an azimuth Θ and a latitude Φ (**Figure 3.2.1**) in a global frame of reference (OXYZ). The axes of the global coordinate system coincide with some relevant global directions; usually the positive directions of OX and OY are east and north, respectively, and OZ is vertical.

In the frame of reference of a fracture set (Oxyz in Figure 3.2.1) a plane is defined by the equation:

$$x\sin\theta\sin\phi + y\cos\theta\sin\phi + z\cos\phi = d \qquad (3\text{-}1)$$

where d is the distance from the global origin O to the plane, and θ and ϕ are the azimuth and latitude of the normal vector (pole Oz' in Figure 3.2.1), respectively. The orientations of potential fracture planes are randomly generated as pairs of polar coordinates (θ, ϕ) in the frame of reference of the fracture set (Oxyz in Figure 3.2.1). The directional pairs (θ, ϕ) follow a spherical PDF, inferred from the measured data. Possible PDFs, illustrated in **Figure 3.2.2**, include uniform or partial uniform, one-parameter or two-parameter Fisher, and Bingham. In some cases all fracture planes may be assumed parallel and deterministically assigned the same orientation. The application of the distributions illustrated in Figure 3.2.2 for generation of directional data is reviewed in more detail in Dershowitz (1979), Dershowitz et al. (1994), and Ivanova (1995). Discussion of PDFs, appropriate for representation of potential fracture plane orientations in different geologic environments, can be found in Sections 3.3.2 - 3.3.6.

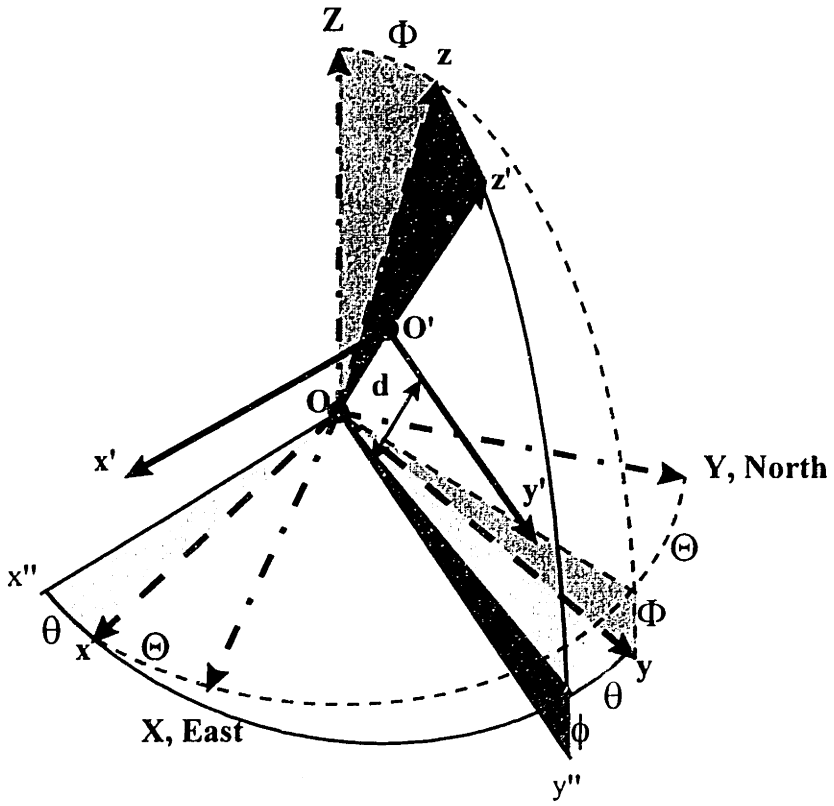


FIGURE 3.2.1 Primary stochastic process: frames of reference of a fracture set. OXYZ: global frame of reference (f.o.r.); Oz: mean pole direction of the fracture set, defined by global azimuth Θ and global latitude Φ; Oxyz: f.o.r. of the fracture set; Oz' : pole (normal vector) of a (potential) fracture plane, defined by local azimuth θ and local latitude ϕ; d = OO' : orthogonal distance of the fracture plane from the global origin O; O'x'y'z' : f.o.r. of the fracture plane (O'x' is parallel to Ox", and O'y' is parallel to Oy").

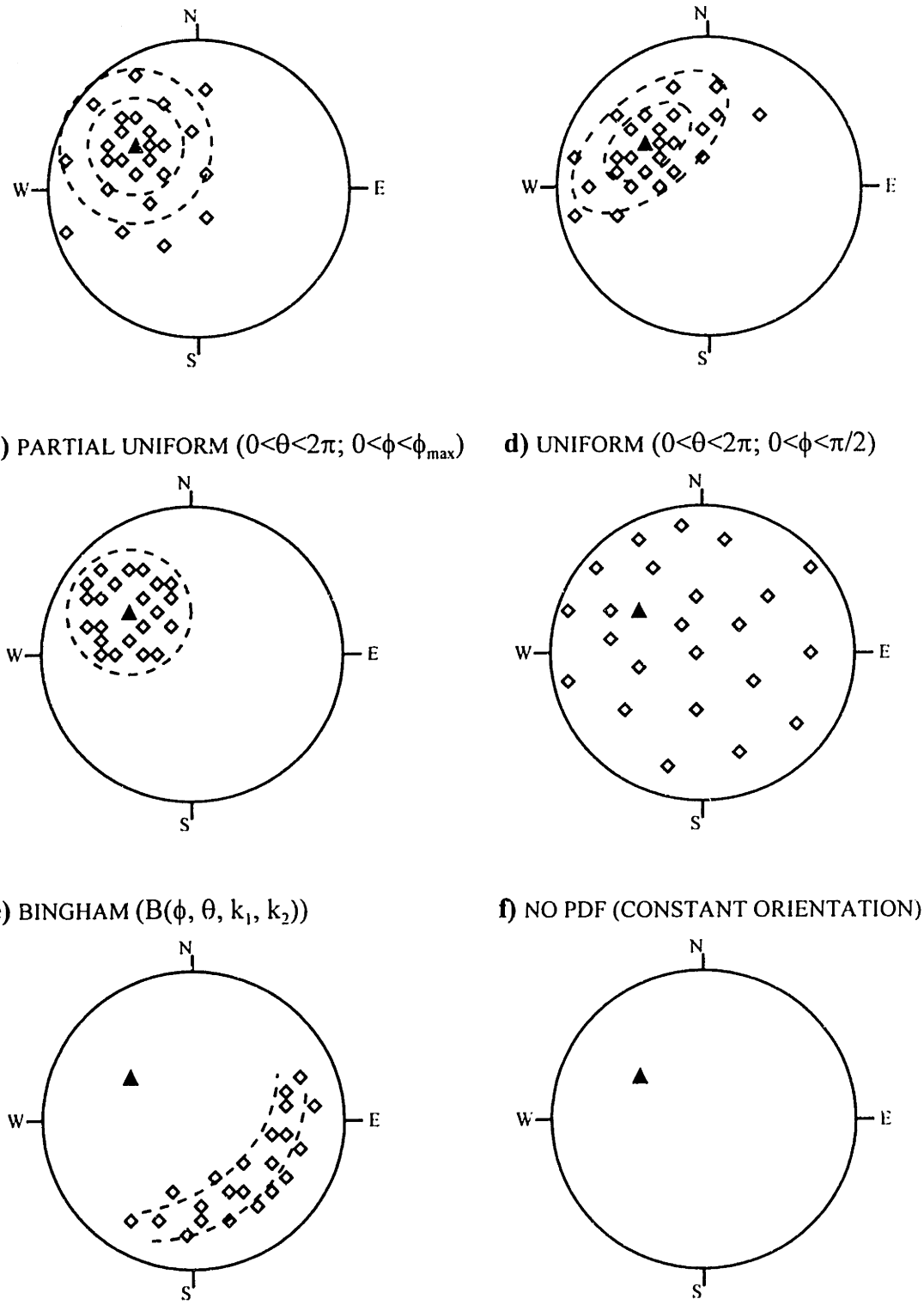


FIGURE 3.2.2 Possible spherical PDFs of fracture plane orientations: a) one-parameter Fisher; b) two-parameter Fisher; c) partial uniform; d) uniform; e) Bingham (girdle); f) constant orientation of all planes.

Modeling volume of a fracture set

The Poisson plane network is generated in a modeling volume that represents the rock mass where the fracture network of interest develops. A generic modeling volume is bound by several planes and surfaces in 3D space. For example, the modeling volumes of independent sets are defined by the boundaries of geologic structures (such as the topographic surface, bedding planes, fold surfaces, or datum planes) over a certain area. The boundaries of the modeling volumes of dependent fracture sets are usually defined by previously generated independent sets. An example of a possible modeling volume for independent fracture sets is shown in **Figure 3.2.3**. The volume in Figure 3.2.3 is enclosed between four vertical planes at $X=X_m$, $X=-X_m$, $Y=Y_m$, $Y=-Y_m$, one horizontal plane at $Z=0$, and a top surface described by a quadratic function:

$$Z = C_1X^2 + C_2XY + C_3Y^2 + C_4X + C_5Y + C_6 \tag{3-2}$$

The top boundary may represent, for example, the topographic surface.

Intensity of the Poisson plane network

The number of planes in the primary process is a Poisson number which is related to the modeled fracture intensity (discussed in more detail in Section 3.2.2). In summary, a homogeneous Poisson plane network of intensity μ corresponds to a Poisson point process in the region:

$$\{(d,\theta,\phi): -\infty < d < \infty, 0 \le \theta \le \pi, 0 \le \phi \le \pi\} \tag{3-3}$$

with non-homogeneous intensity function of the type:

$$\mu(d,\theta,\phi) = \mu f_{\theta,\phi}(\theta,\phi) \tag{3-4}$$

where $f_{\theta,\phi}$ is the joint PDF of θ and ϕ and μ is a positive constant [Veneziano 1978].

The ordered distances from an arbitrary point to the planes of a Poisson network with intensity μ define a Poisson point process on a line with intensity 2μ [Miles 1969; Veneziano 1978]. The expected number of planes, $E[N_p]$ that intersect a sphere of radius R is therefore:

$$E[N_p] = 2\mu R = 2\mu R \tag{3-5}$$

The expected number of Poisson planes that intersect a 3D region of irregular shape is theoretically expressed by Miles (1969) as a function of the surface area of the region.

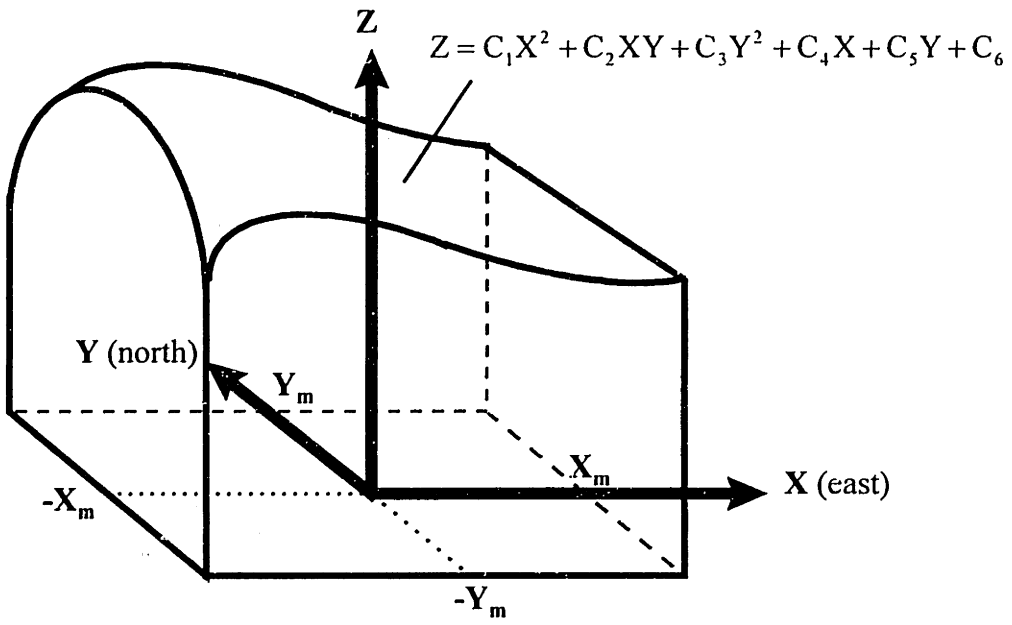


FIGURE 3.2.3 Primary stochastic process: example of a modeling volume for independent fracture sets.

3.2.2 Modeling of fracture intensity: primary and secondary stochastic processes

In the model, fracture intensity is defined as the cumulative fractured area per unit volume of rock:

$$P_{32} = \frac{\sum_{i=1}^{N} A_{f,i}}{V} \qquad (3\text{-}6)$$

where N is the total number of fractures and $A_{f,i}$ is the area of an individual fracture inside the volume V (P_{32} is notation after Dershowitz & Herda 1992

who defined various measures of fracture intensity). For statistical modeling of a system that consists of a large number of fractures, P_{32} is the best intensity measure for two reasons: (1) it represents the two-dimensional nature of fractures, and (2) it does not depend on the size and shape of the region (as long as the fractures are smaller than the region itself). Fracture intensity, modeled by the combined primary and secondary processes, incorporates fracture size distribution and cumulative fracture area.

A random subdivision of the planes, generated by the primary process, into a fractured region and its complementary region of intact rock constitutes the basis of the secondary process. This subdivision is accomplished by a Poisson line tessellation on every plane and a process of marking the so-created polygons as fractured or intact rock (Figure 3.1.2b). The secondary process produces sets of fractures that have a certain size and shape variation and are arranged in clusters.

The model incorporates fracture intensity algorithms that are based on previously published second and first moments of Poisson plane and line processes [Miles 1973; Veneziano 1978], and some geometric properties of the population of polygons, herewith established via numerical simulations. The rest of Section 3.2.2 presents in detail the application of Poisson point, line and plane processes for modeling of fracture intensity (defined by Equation 3-6) through generation of populations of polygon-fractures with desired size and shape variation.

Poisson line tessellation

In the local 2D frame of reference O'x'y' on a plane, a line from the Poisson network is defined as:

$$\mathbf{x'\cos\alpha + y'\sin\alpha = D} \tag{3-7}$$

in terms of an angle α on A, measured counterclockwise from the axis O'x', and a distance D from the origin O' to the line (**Figure 3.2.4**).

A homogeneous Poisson line network with intensity λ corresponds to a Poisson point process in the region:

$$\{(\mathbf{D},\alpha):\mathbf{0} \le \mathbf{D} \le \infty, \mathbf{0} \le \alpha \le \mathbf{2\pi}\} \tag{3-8}$$

with intensity function of the points of the type:

$$\lambda(\mathbf{D},\alpha) = \lambda \mathbf{f}_{\alpha}(\alpha) \tag{3-9}$$

where λ is a positive constant and $f_\alpha(\alpha)$ is the PDF of α in $[0, 2\pi]$. The ordered distances from an arbitrary point on the plane to the lines of a Poisson line network with intensity λ form a Poisson point process with intensity 2λ [Miles 1973; Veneziano 1978]. The expected number of lines that intersect a circle of radius R is $2\lambda R$. The expected number of Poisson lines that intersect a 2D region of irregular shape is theoretically expressed by Miles (1973) as a function of the perimeter of the region.

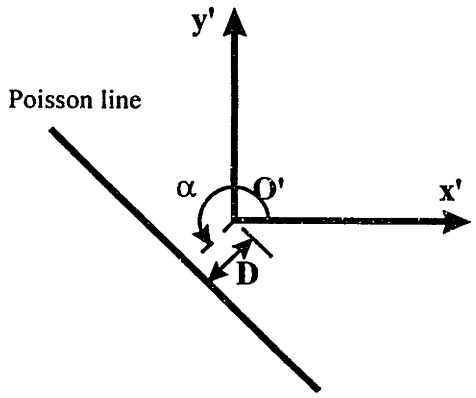


FIGURE 3.2.4 Secondary stochastic process: generation of a Poisson line.

The points of intersection of all pairs of lines determine the vertices of polygonal tiles. First and second moments of the polygons created by a Poisson line tessellation of intensity λ are summarized in **Table 3.2.1**.

GEOMETRIC PROPERTY	MEAN	VARIANCE, COVARIANCE	
		N	A
Number of vertices, **N**	4	$\frac{(\pi^2-8)}{2}$	$\frac{\pi(\pi^2-8)}{2\lambda^2}$
Polygon area, **A**	$\frac{\pi}{\lambda^2}$	$\frac{\pi(\pi^2-8)}{2\lambda^2}$	$\frac{\pi^2(\pi^2-2)}{2\lambda^4}$

TABLE 3.2.1 Theoretical statistics of polygons obtained through a Poisson line tessellation with intensity λ [after Miles 1973, and Veneziano 1978].

The standard deviation of polygon areas can be calculated as:

$$\sigma_A = \sqrt{\mathbf{var}[A]} = \sqrt{\frac{\pi^2(\pi^2-2)}{2\lambda^4}} = \sqrt{\frac{\pi^2-2}{2}}\left(\frac{\pi}{\lambda^2}\right) \approx \mathbf{1.98E[A]} \tag{3-10}$$

Polygon marking procedure and distribution of fracture sizes and shapes

The process of dividing each plane into a fractured region and its complementary region of intact rock is homogeneous in a statistical sense. The probability of marking a polygon as fractured, P_f, is calculated individually for every polygon:

$$\mathbf{P_f = P_f(size, shape, location)} \tag{3-11}$$

It can be expected that, if only polygons with certain shapes are retained as possible fractures, the mean and variance of their areas will be different from those of the entire population of polygons created by a Poisson line tessellation. For example, analysis according to the Best Linear Unbiased Estimation (BLUE) theory calculates conditional mean and variance of the areas, A, of polygons that have a certain number of vertices, N=n, as:

$$\mathbf{E[A|N=n] \cong E[A] + \frac{cov[A,N]}{var[N]}(n - E[N])}$$

$$\mathbf{var[A|N=n] \cong var[A]\left(1 - \frac{(cov[A,N])^2}{var[N]\,var[A]}\right)} \tag{3-12}$$

According to the BLUE formula, the mean area of the polygons with five vertices, created by a Poisson line tessellation of intensity λ, is:

$$\mathbf{E[A|N=5]} \approx \frac{\pi}{\lambda^2} + \frac{\frac{\pi(\pi^2-8)}{2\lambda^2}}{\frac{(\pi^2-8)}{2}}(5-4) = \frac{\pi}{\lambda^2} + \frac{\pi}{\lambda^2} = 2\frac{\pi}{\lambda^2} = \mathbf{2E[A]} \tag{3-13}$$

Similarly, E[A|N=6]=3E[A], E[A|N=7]=4E[A], etc. Although the analysis is approximate, it illustrates an important characteristic of the polygonal shapes created by a Poisson line tessellation: that larger polygons tend to have more vertices. Larger polygons also tend to have larger angles, since the mean angle of an n-sided polygon is $\pi(n-2)/n$.

In the 3D model, only polygons that have shapes similar to that of natural fractures are retained from the population of polygons created by a Poisson line tessellation. The shape of a "good" polygon is that of a typical non-elongated natural fracture which is here defined as follows:

(1) the polygon is convex and has at least four vertices;
(2) all angles are at least $\pi/3$ (60°);
(3) the polygon elongation (as defined in Figure 3.1.1) is not more than 1.6 (modeling of elongated shapes is discussed by Ivanova 1995).

Based on the BLUE analysis above, one can conclude that larger polygons are more likely than smaller ones to be retained as potential fractures. **Figure 3.2.5** compares the size distribution of 84191 polygons obtained by a Poisson line tessellation, to the size distribution of 14378 polygons marked as "good" in the simulation. The Poisson line tessellation creates a large number of small polygons. The polygon size PDF has a "long tail": a small number of polygons with large areas are widely spread in the range above the mean E[A]. The PDF of the sizes of "good" polygons has a similar shape, but a much lower peak in the range of the smallest polygon sizes. The polygon size distributions in Figure 3.2.5 were obtained through numerical simulations of the Poisson line tessellation, in which the desired mean area E[A] of all polygons was given as input and the line intensity λ was calculated as:

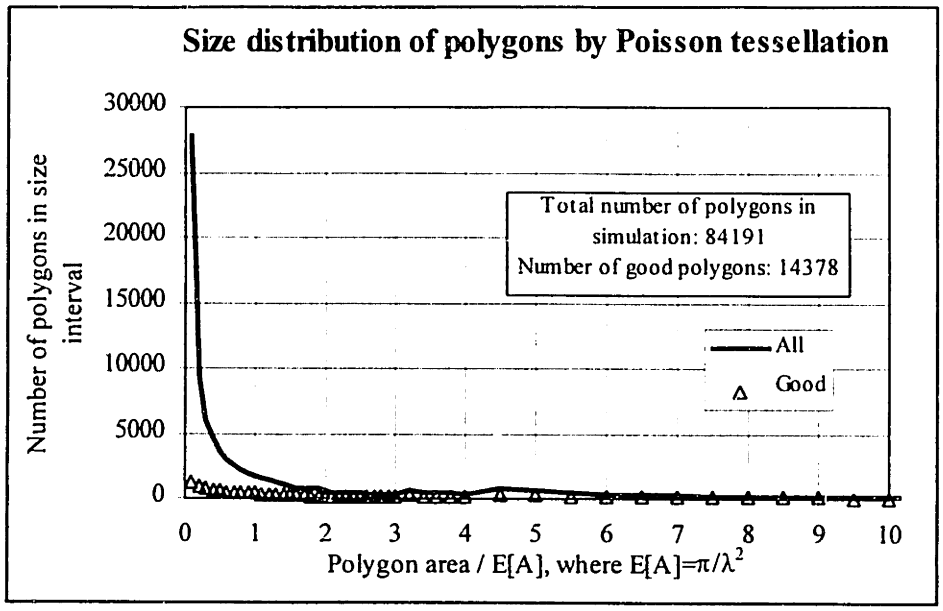
$$\lambda = \sqrt{\frac{\pi}{\mathbf{E[A]}}} \tag{3-14}$$

Since most of the polygons, discarded in the shape marking process have small sizes, the expected area of "good" polygons is larger than the expected area of the entire population of polygons created by a Poisson line network. **Table 3.2.2** summarizes the relationship of the sizes of "good" polygons (including mean, standard deviation, median, and cumulative area) to the intensity λ of the Poisson line tessellation.

Two correction coefficients best express the relationship of the areas of "good" polygons to the intensity of the underlying Poisson line tessellation:

(1) the ratio C_A of the mean area of "good" polygons to the mean area of all polygons produced by a tessellation with intensity λ;
(2) the ratio $\gamma = A'_T / A_T$, where A'_T is the sum of the areas of all "good" polygons that are created within a finite region of total area A_T by a Poisson line tessellation of intensity λ.

a)



b)

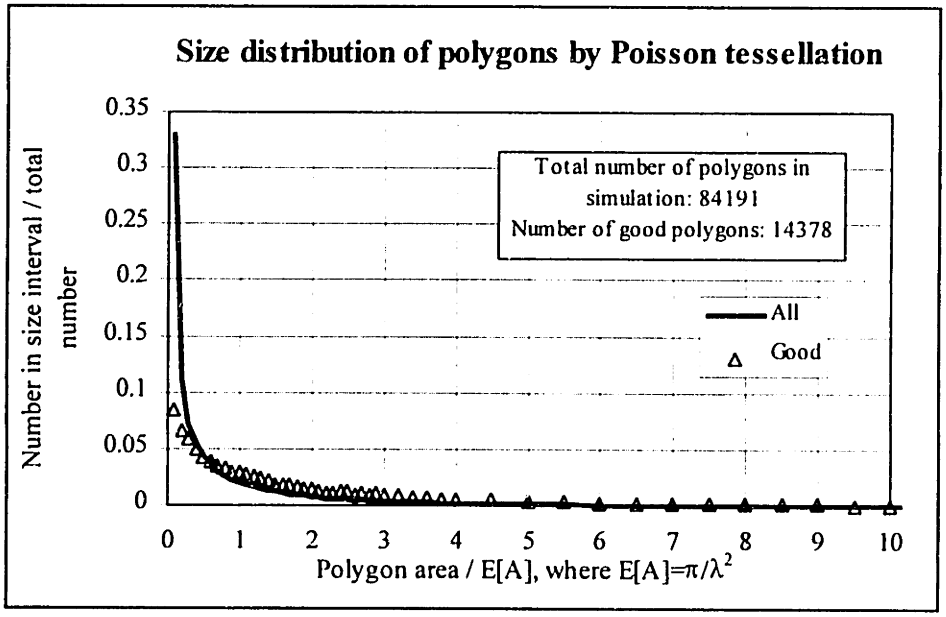


FIGURE 3.2.5 Distributions of areas of all polygons and "good" polygons obtained through a Poisson line tessellation of intensity λ: a) number of polygons vs. normalized polygon area; b) normalized number of polygons vs. normalized polygon area.

STATISTICAL PROPERTY	VALUE BASED ON SIMULATIONS
Mean fracture area, $E[A']$	$E[A'] = C_A \frac{\pi}{\lambda^2}$, where C_A=2.2
Standard deviation of fracture area, $\sigma_{A'}$	$\sigma_{A'} = 1.4E[A']$
Median of fracture area, $M_{A'}$	$M_{A'} = 0.5E[A']$
Cumulative fracture area, A'_T	$A'_T = \gamma A_T$, where γ=0.4

TABLE 3.2.2 Statistics of the distribution of areas of "good" polygons (i.e. polygons with fracture-like shapes), produced by a Poisson line tessellation of intensity λ in a region with total area A_T.

Theoretically, the coefficients C_A and γ do not depend on the intensity λ of the Poisson line network but only on the rule by which polygons are marked as "good" (Veneziano, pers. comm.). Therefore, the coefficients C_A=2.2 and γ=0.4, given in Table 3.2.2, are valid for any intensity λ of the Poisson lines, as long as "good" polygons are marked according to the three rules listed above.

Figure 3.2.6 illustrates the fit of Gamma PDF, $\Gamma(\alpha, \beta)$, to the distribution of sizes (areas) of "good" polygons generated with a Poisson line tessellation. In the empirical PDF, the polygon size intervals on the abscissa are normalized by the mean area of "good" polygons, whereas on the ordinate the number of polygons in each size interval is normalized by the total number of "good" polygons in the simulation. The coefficients α and β of the Gamma PDF are calculated so that the mean, m_Γ, and variance, σ_Γ^2, are equal to the mean and variance of the empirically obtained normalized PDF of "good" polygon areas (see Table 3.2.2):

$$\begin{aligned} \mathbf{m}_\Gamma &= \alpha\beta = \mathbf{1.0} \qquad \alpha = \mathbf{0.51} \\ \sigma_\Gamma{}^2 &= \alpha\beta^2 = \mathbf{(1.4)^2} \Rightarrow \quad \beta = \mathbf{1.96} \end{aligned} \tag{3-15}$$

The distribution of polygon sizes, illustrated in Figure 3.2.6 and described in Table 3.2.2, is representative of natural fracture systems in the sense that they usually include few large fractures and many small fractures. However, in many cases one may want to represent a smaller variation of fracture sizes (i.e. a smaller standard deviation $\sigma_{A'}$ of sizes of "good" polygons). In order to do so, the model currently implements a simple procedure of marking "good' polygons not only by shape, but also according to their size relative to the mean equivalent radius of "good" polygons $E[R'_e]$, and relative to the mean polygon area E[A] of the

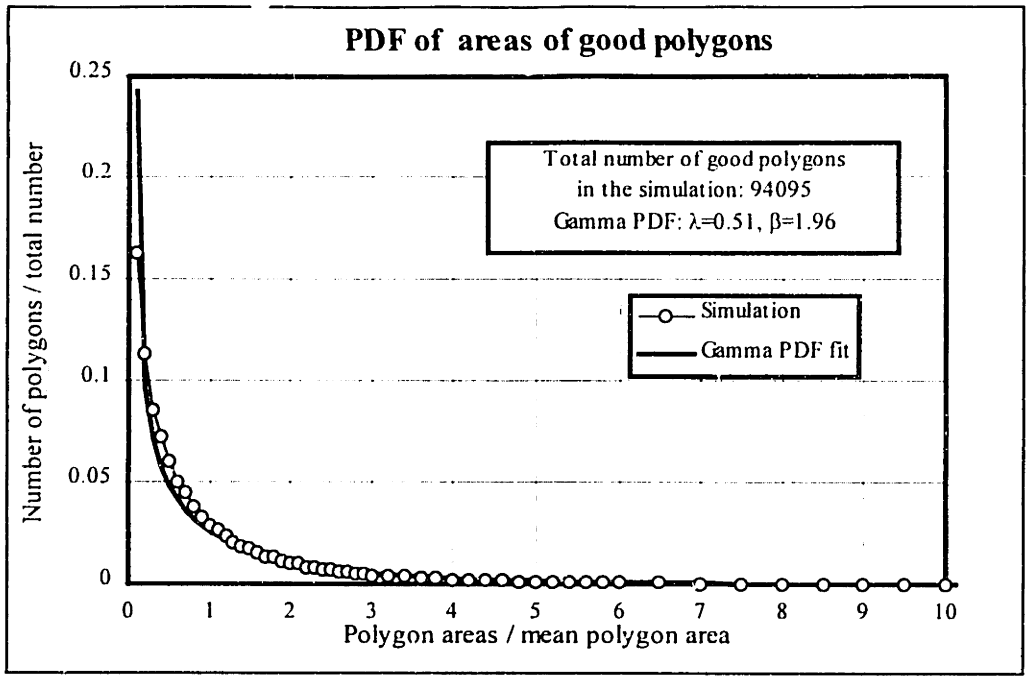
underlying Poisson line tessellation. **Table 3.2.3** summarizes statistics of the PDF of areas of "good" polygons obtained through a Poisson line tessellation and different shape-and-relative-size marking processes (for comparison, the characteristic values for the marking process only by shape are also shown).

Applying a marking rule according to relative polygon size essentially leads to truncation of the size PDF, hence to a decrease in the spread of sizes of the remaining polygons, i.e. to a smaller $\sigma_{A'}$ (more flexible algorithms for control of the size distribution are discussed in Section 3.4). Size PDFs of populations of polygons marked according to some of the rules enlisted in Table 3.2.3 are included in the Appendix. As Table 3.2.3 shows, the correction coefficients C_A and γ are different for different marking rules. Theoretically, as long as the marking is only according to shape and relative size, the coefficients C_A and γ are independent of the intensity λ of the Poisson line tessellation. A marking process according to relative size of polygons leads to decrease of the cumulative fractured area on a plane (i.e. a smaller coefficient γ). The maximum possible value in the current version of the model is γ =0.4 (corresponding to marking of all "good" polygons). This value of γ represents the maximum possible intensity of coplanar fractures on a given plane that can be reproduced with the model.

MARKING RULE	$C_A = \frac{E[A']}{E[A]}$	$\gamma = \frac{\sum_{i=1}^{N} A'_i}{A_T}$	$\frac{\sigma_{A'}}{E[A']}$	$\frac{M_{A'}}{E[A']}$
All with good shape	2.2	0.4	1.4	0.5
$A'_i > E[A]$	3.6	0.36	0.93	0.7
$A'_i > 2E[A]$	5.0	0.30	0.74	0.75
$R'_{e,i} < 3E[R'_e]$	1.8	0.38	1.26	0.6
$R'_{e,i} < 2E[R'_e]$	1.4	0.23	0.96	0.7
$A'_i > E[A]$, $R'_{e,i} < 3E[R'_e]$	3.6	0.36	0.90	0.8
$A'_i > E[A]$, $R'_{e,i} < 2E[R'_e]$	3.4	0.36	0.72	0.75

TABLE 3.2.3 Statistics of "good" polygons, marked by shape and relative size, obtained through a Poisson line tessellation of intensity λ in a finite region with total area A_T. $E[A]=\pi/\lambda^2$: theoretical mean area of all polygons; N: total number of "good" polygons; $E[A']$, $\sigma_{A'}$, and $M_{A'}$: expected value, standard deviation, and median, respectively, of the areas of "good" polygons; A'_i and $R'_{e,i}$: area and equivalent radius (radius of a circle with area equal to A'_i) of the i-th "good" polygon; $E[R'_e]$: expected value of the equivalent radius of "good" polygons.

a)



b)

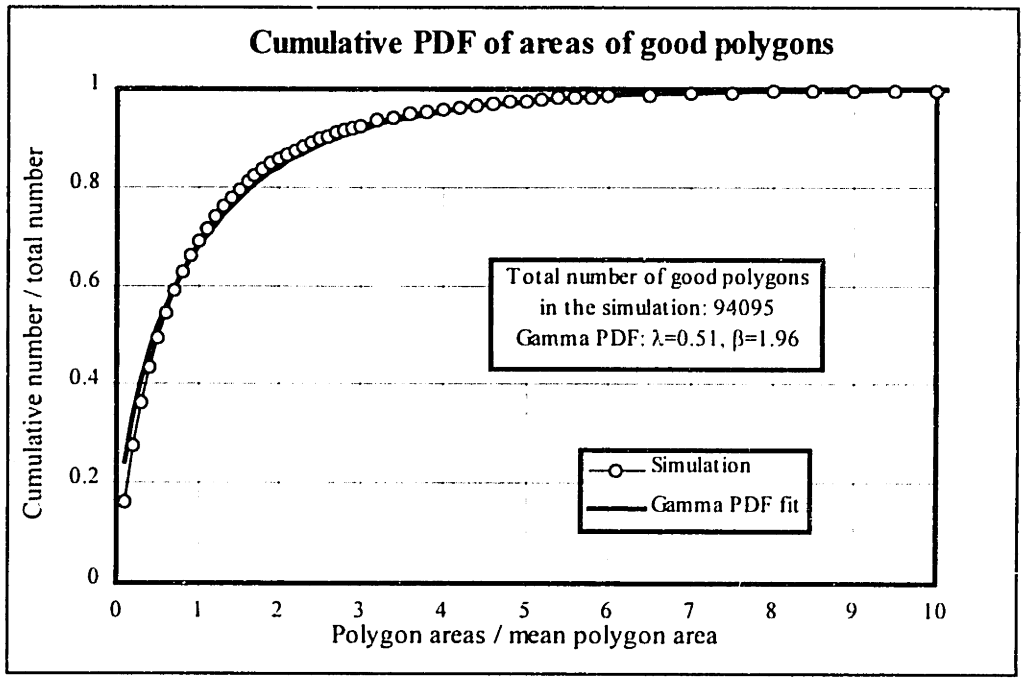


FIGURE 3.2.6 Empirical distribution of areas of "good" polygons obtained though a Poisson line tessellation, and fit of Gamma PDF: a) PDF of polygon areas; b) cumulative PDF of polygon areas.

Poisson plane network intensity and total fracture area in the modeling volume

In order to obtain the expected value of fracture intensity P_{32} (Equation 3-6), one needs to know not only γ but also the expected value of the cumulative total area $\sum A_{T,i}$ that is cut from a volume V by a Poisson network of planes with intensity μ. **Figure 3.2.7** illustrates the derivation of the $\sum A_{T,i}$ for a sphere. The plane at a random distance d_i cuts from the sphere with radius R a circle with radius:

$$r_i = \sqrt{R^2 - d_i^2} \tag{3-16}$$

and area:

$$A_{T,i} = \pi r_i^2 = \pi\left(R^2 - d_i^2\right) \tag{3-17}$$

The expected area $E[A_T]$ of a circle cut by a Poisson plane from a sphere of radius R can be calculated as:

$$E\left[A_T\right] = \int_0^R pdf(d)\pi\left(R^2 - d^2\right)dd = \int_0^R \frac{1}{R}\pi\left(R^2 - d^2\right)dd = \pi\frac{1}{R}\frac{2}{3}R^3 = \frac{2}{3}\pi R^2 \tag{3-18}$$

where pdf(d)=1/R is the probability density function of d. Using equations (3-5), (3-6), and (3-18), the expected cumulative area of potential fractures per unit volume (equal to the fracture intensity P_{32}) is:

$$E\left[\frac{\sum A_{f,i}}{V}\right] = P_{32} = \frac{E\left[N_p\right]\gamma E\left[A_T\right]}{V} = \frac{2\mu R\gamma E\left[A_T\right]}{V} = \frac{2\mu R\gamma\frac{2}{3}\pi R^2}{\frac{4}{3}\pi R^3} = \gamma\mu \tag{3-19}$$

The expected cumulative area of polygon-fractures per unit volume does not depend on R, i.e. it is independent of the shape and size of the total volume in which the Poisson plane process is generated. Therefore the modeled fracture intensity P_{32} only depends on the intensity of the Poisson line network μ and on the rule by which polygons are marked as potential fractures.

If the fracture sizes are comparable to the size of the region of interest, truncations of fractures by the boundaries of the region would affect the coefficient γ, and hence the geometric property described by Equation 3-19. In such a case, if one uses this stochastic model, the fracture system needs to be generated in a

modeling volume much larger than the region of interest. However, it is important to know the exact locations of fractures as large as or larger than the region of interest. Therefore, it is recommended that they are not stochastically generated, but deterministically established if at all possible.

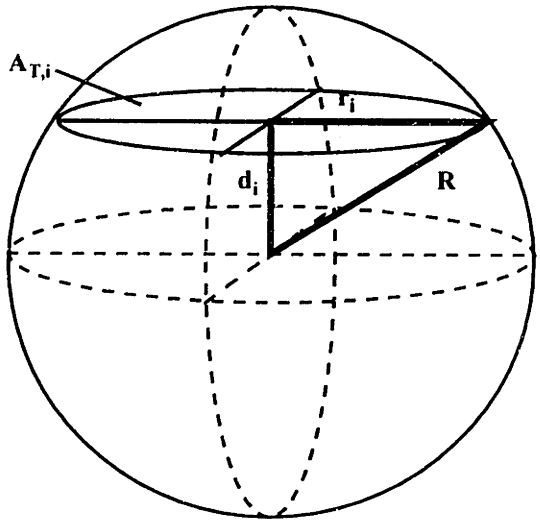


FIGURE 3.2.7 Diagram illustrating the derivation of total fractured area in a sphere.

Zone marking of polygons and varying fracture intensity in space

There are two ways to obtain a smaller percentage γ^* of the potentially fractured planes (i.e. a lower percent of coplanar fractures). One can either use the tertiary stochastic process (Section 3.2.3), or mark the "good polygons" with a probability P such that:

$$\gamma^* = \mathbf{P}\gamma \tag{3-20}$$

where $0<P<1$.

Then a Poisson plane network with intensity $\mu^*=\mu/P$ preserves the fracture intensity:

$$\mathbf{E}\left[\frac{\mathbf{A}_{T,f}}{\mathbf{V}}\right] = \gamma\mu = \gamma\mathbf{P}\frac{\mu}{\mathbf{P}} = \gamma^*\mu^* \tag{3-21}$$

In the model, the probability of marking a polygon as fractured can be a function of the distance to some specified cluster centers, for example, other fractures or local faults (**Figure 3.2.8**). A zone is defined by the maximum and minimum value of the distance from the face of a polygon-fracture to the face of another polygonal feature. Within every zone j the zone probability P_j is constant, but that constant is generally different in different zones. Polygons in zone j are retained with probability P_j and discarded with probability $1 - P_j$. To obtain different fracture intensity within the same fracture set in different portions V_1, V_2, etc. of the volume V, one has to calculate the different probabilities P_1, P_2, etc. for polygon marking in the different regions. Volume V_i does not have to be continuous; it can be defined as a function of any continuous rock property. For example, V_i may be defined as "the regions where the fractures of the set are at a distance not more than L" from the fractures of a primary set or from a fault face. Alternatively, V_i can be defined as "the regions where the rock is dolomite" or "the regions where the porosity of the rock is not more than n%", etc. It is only the intensity of fractures, and not the orientation that can be controlled by the marking probability process. Any marking according to polygon orientation would affect the function $f_{\theta,\phi}(\theta,\phi)$ in Equation (3-4) and the constant μ itself.

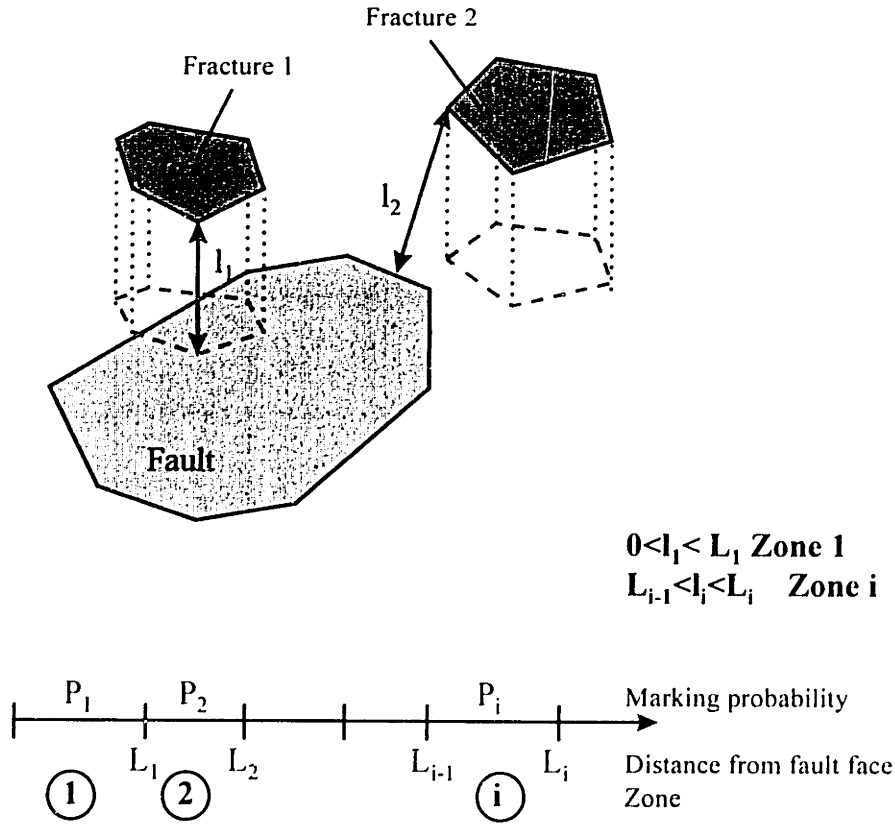


FIGURE 3.2.8 Secondary stochastic process: definition of fracture zones and zone marking probability.

Figure 3.2.9 and **3.2.10** illustrate examples of varying fracture intensity in space, generated with the numerical code GEOFRAC that implements the 3D hierarchical model. Figure 3.2.9a illustrates a fracture network with higher intensity in the vicinity of a fault. Figure 3.2.9b shows the relative locations of the fault, and two outcrop planes: vertical and horizontal. Figures 3.2.9c and d depict the trace outcrops of the fracture network on the horizontal and vertical outcrop planes, respectively. Figure 3.2.10 illustrates the trace outcrops of two fracture systems: one exists only in a layer enclosed between two bedding surfaces but not in the surrounding rock (Figure 3.2.10a); the other one has higher intensity in the rock below a bedding plane than above it (Figure 3.2.10b).

Summary of fracture intensity modeling

In summary, desired fracture intensity is modeled in the following order:

1) Desired mean fracture size $E[A']$, fracture intensity $P_{32,i}$ in various regions V_i and the extent of those regions are given as input.
2) The intensity of the Poisson plane process is calculated as $\mu = P_{32,max} / \gamma$.
3) Poisson planes are generated in the total volume V.
4) The expected polygon area of the Poisson line network is calculated as $E[A] = E[A'] / C_A$.
5) Poisson line tessellation with intensity $\lambda=(\pi/E[A])^{1/2}$ is generated on the planes.
6) The probability of marking "good polygons" is calculated as $P_i=P_{32,i}/P_{32,max}$.
7) Polygons with bad shapes are discarded and polygons with good shapes are marked as fractures (with probability P_i in region V_i).

The above algorithm ensures that the expected value of the intensity of the fracture set will be the desired intensity P_{32} (or $P_{32,i}$ in regions V_i, respectively).

3.2.3 Modeling of stress field variation: tertiary stochastic process

The translation procedure of the tertiary stochastic process (Figure 3.1.2c) controls the ratio of coplanar fractures and fractures that are parallel but not coplanar. The rotation algorithm (Figure 3.1.2c) represents possible deviations of fracture orientations from the regional directions due to variations of the stress field near local geologic structures.

Random polygon translation

In the tertiary stochastic process translation is performed in the frame of reference of the fracture plane ($O'x'y'z'$ in Figure 3.2.1). Translation is accomplished by assigning a non-zero coordinate z'_i to the center and the vertices of a polygon and hence to the entire polygon. The algorithm, currently implemented in the model, allows for translation of a polygon at a maximum distance:

$$dz'_{max} = C\frac{E[R'_e]}{R'_e}E[R'_e] \qquad (3\text{-}22)$$

where R'_e is the equivalent radius of a "good" polygon, and $E[R'_e]$ is the expected value of that radius. Thus larger polygons are shifted closer to their original positions than smaller polygons. C in Equation (3-22) is a coefficient of fracture coplanarity: the smaller C, the more coplanar fractures can be expected. **Figure 3.2.11** illustrates the effect of increasing C to decrease the coplanarity of fractures. The fracture systems shown on horizontal outcrops in Figure 3.11a and b are statistically the same (in terms of P_{32}, mean size, orientation, etc.); only C in Figure 3.2.11b is higher, hence there are fewer coplanar fractures.

Random polygon rotation

The random rotation of polygons in the tertiary process is used for cases when the local conditions modify the stress field so much that the most likely fracture orientation deviates from the most likely orientation defined by the general stress field. The algorithm, currently implemented in the model, checks the orientations of numerically generated polygons against a specified relation of the real fracture orientations to the geologic structures, bedding planes, or other features that may influence the direction of fracture propagation. **Figure 3.2.12** illustrates the rotation process. In Figure 3.2.12a the fracture system is generated with intensity P_{32} through superposition of two sub-vertical sets: one striking NW, and another one striking NE. In Figure 3.2.12b, after rotation, the fractures strikes are concentric or radial to a circular dome with apex in the center of the rectangular area. Rotation preserves the fracture intensity P_{32} of the set. The rotation process is discussed in detail in Section 3.3.2 in the context of fold-related fractures.

3.3 FRACTURE SYSTEM MODELING

3.3.1 General model algorithm

In the model, a fracture system is reproduced by superposition of independent and dependent fracture sets. Two fracture sets can be considered independent of one another and reproduced as such if the inferred statistics of the first set are not related to the statistics of the second set. Each independent set is modeled with the stochastic sequence, described in the previous section (therefore there is a correlation between the geometric characteristics of its members). The same three stochastic processes generate a dependent fracture set. Dependence on previously generated sets can be obtained in many ways; for example, the zone probability can be defined as a function of the distance to primary fractures. Also, dependence can be obtained by defining a function for termination of fractured polygons from the secondary set at the intersections with fractures from the primary set.

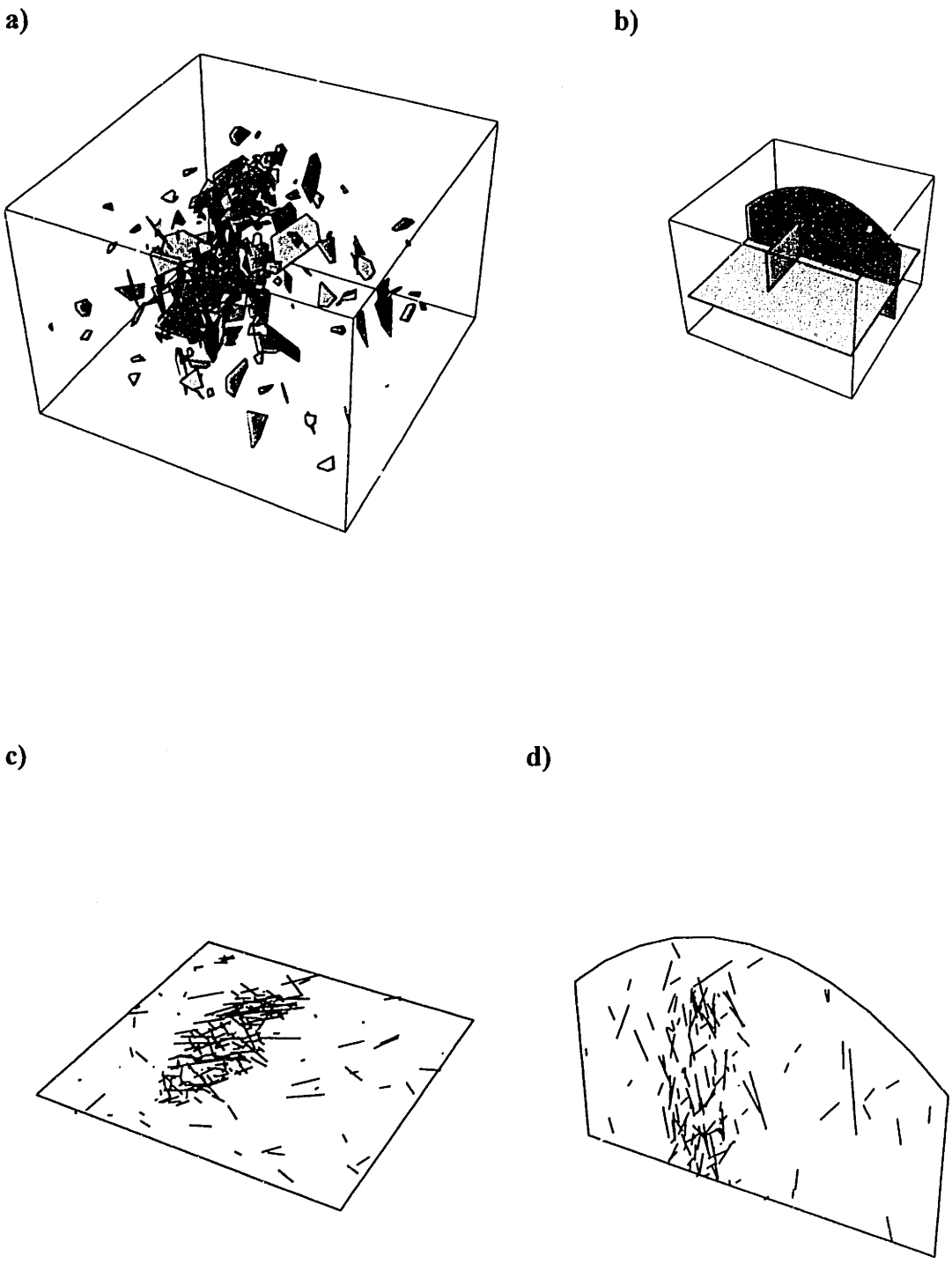
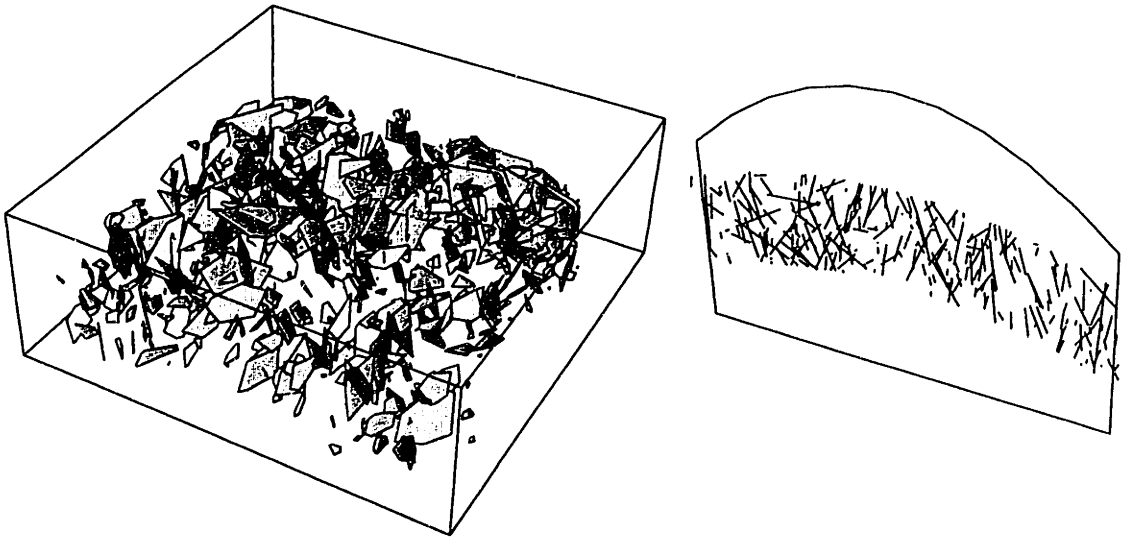


FIGURE 3.2.9 Fracture system with highest intensity near a fault (generation with computer program GEOFRAC): a) 3D fracture system ; b) fault and outcrop planes; c) horizontal trace outcrop; d) vertical trace outcrop.

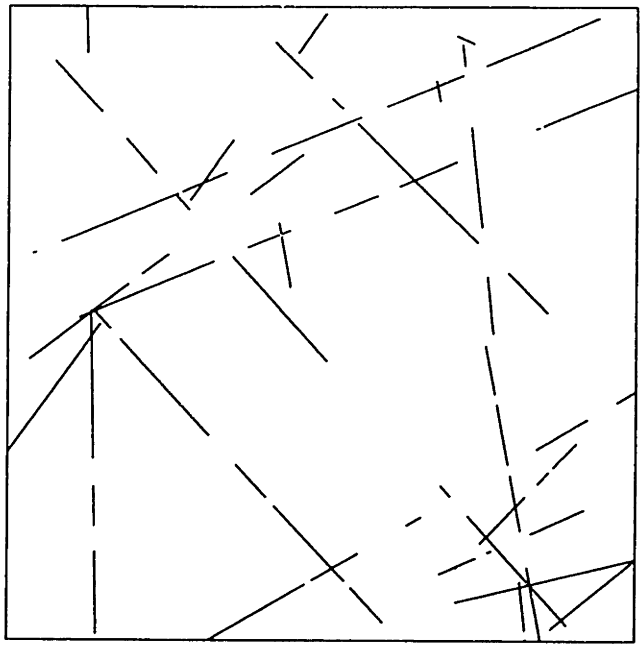
a)



b)

FIGURE 3.2.10 Fracture systems with different intensity in different layers (generation with computer program GEOFRAC): a) 3D view and vertical outcrop of a fracture system that exists only in one layer; b) vertical outcrop of a fracture system with intensity increasing with depth.

a)



b)

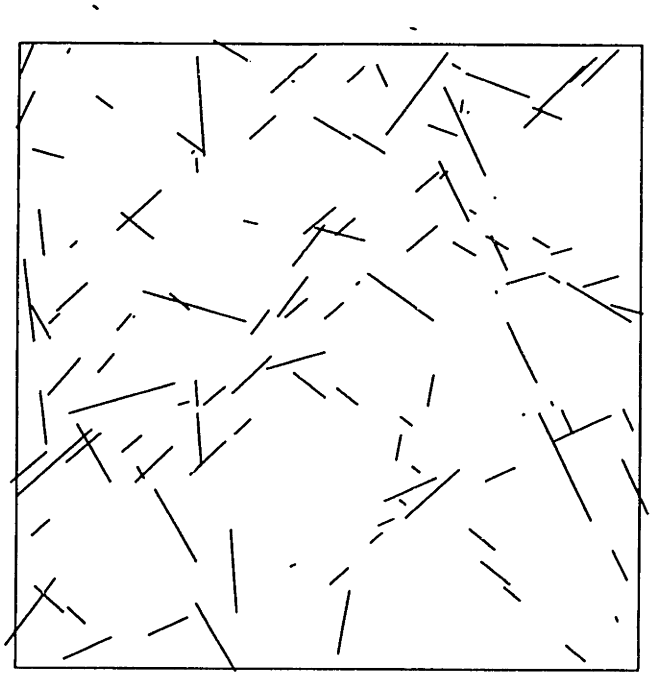
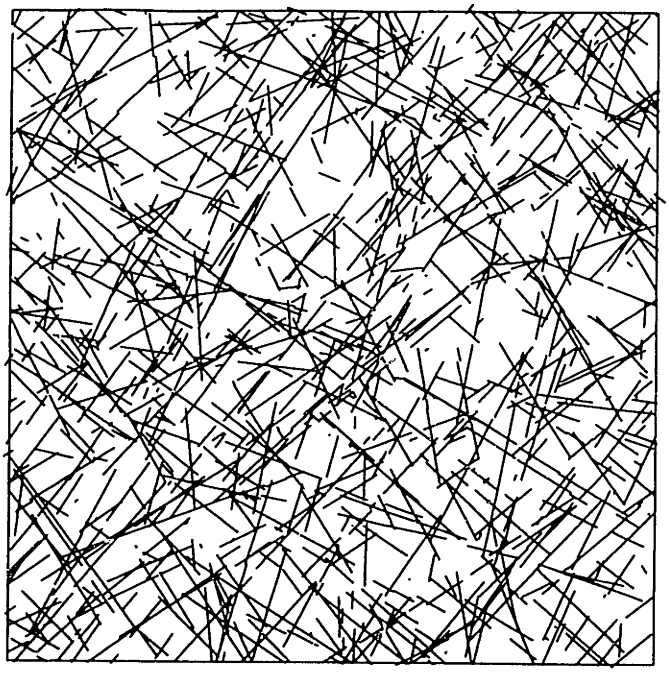


FIGURE 3.2.11 Tertiary stochastic process, translation of polygons (generation with computer program GEOFRAC): a) trace outcrop of coplanar fractures, C=0; b) trace outcrop of non-coplanar fractures, C=0.5. C: coefficient of coplanarity.

a)



b)

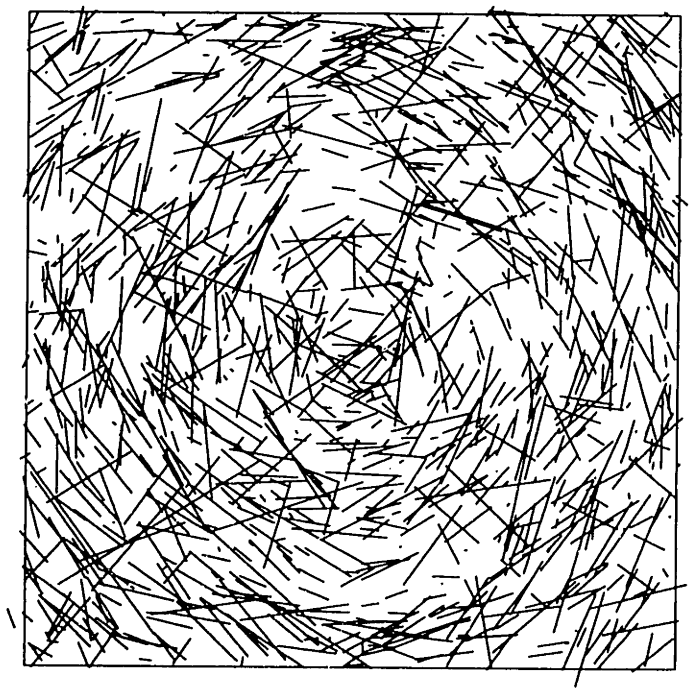


FIGURE 3.2.12 Tertiary stochastic process, rotation of polygons (generation with computer program GEOFRAC): a) trace outcrop of fractures with orientations related to the general stress field directions; b) trace outcrop of the fractures from (a) with their orientations related also to a circular dome.

Figure 3.3.1 shows a fracture system, generated by superposition of two sets: Set 1 (blue), an independent set of large, sub-horizontal primary fractures, and Set 2 (yellow), a dependent set of smaller, sub vertical secondary fractures, produced in the vicinity of Set 1. Figure 3.3.1 illustrates the capability of the model to represent fracture clustering by assigning appropriate zone probabilities. In the generation of Set 2, a zone function has been defined so that the closer a polygon is to a fracture from Set 1, the higher the probability is that the polygon will be marked as fractured.

The rest of Section 3.3 presents specific algorithms that can be used for representation of rock fracture systems in the major geologic environments: folds (Section 3.3.1), crustal faults (Section 3.3.2), remote tension (Section 3.3.3), thermal contraction (Section 3.3.4), and central structures (Section 3.2.5). The recommended methods for representation of the orientation variation of potential fracture planes include application of theoretical PDFs on a unit hemisphere (schematically illustrated in Figure 3.2.2). With the exception of Bingham, all PDFs shown in Figure 3.2.2 are implemented in the computer program GEOFRAC. Sections 3.3.2-3.3.6 also contain suggested procedures for modeling of fracture intensity in the different geologic settings. Most of the fracture intensity procedures are already implemented in the computer code GEOFRAC.

3.3.2 Fracture systems related to folds

All fracture sets that form during folding are related to the curved fold surface in terms of both orientation and intensity (Section 2.1). Therefore, the first step in modeling a fold-related fracture system is representation of the geometry of the fold itself. Currently in the 3D hierarchical model fold geometry is described by a cubic function of the type:

$$Z = c_1X^3 + c_2X^2Y + c_3XY^2 + c_4Y^3 + c_5X^2 + c_6XY + c_7Y^2 + c_8X + c_9Y + c_{10} \quad (3\text{-}23)$$

The coefficients c_i can be derived through a polynomial fit to elevations of formation contacts of folded strata. Such elevations may be obtained, for example, from log data or from mapped exposures of folds on outcrop planes. By varying the coefficients c_i in Equation 3-23 one can represent various fold shapes. For example, if all coefficients except c_{10} and c_5 are zero, Equation 3-23 represents a cylindrical fold with vertical axial plane parallel to the global axis OY (a syncline if $c_5>0$, or an anticline if $c_5<0$).

The orientations of fractures in folded strata are related to the varying strike and dip of the fold surface. **Figure 3.3.2** illustrates the calculation of strike and dip at a given point $P_0(X_0; Y_0; Z_0)$ on the fold surface. The coordinates of the normal vector N_f are defined by the directional first derivatives of the function $F(X,Y,Z)=0$, describing the fold surface:

$$\vec{N}_f = \begin{pmatrix} F_X \\ F_Y \\ F_Z \end{pmatrix} = \begin{pmatrix} \frac{\partial F}{\partial X} \\ \frac{\partial F}{\partial Y} \\ \frac{\partial F}{\partial Z} \end{pmatrix}_{\substack{X=X_0 \\ Y=Y_0}} = \begin{pmatrix} -3c_1X^2 - 2c_2XY - c_3Y^2 - 2c_5X - c_6Y - c_8 \\ -c_2X^2 - 2c_3XY - 3c_4Y^2 - c_6X - 2c_7Y - c_9 \\ 1 \end{pmatrix}_{\substack{X=X_0 \\ Y=Y_0}}$$

(3-24)

The unit normal vector n_f at the point P_0 has coordinates:

$$\vec{n}_f = \begin{pmatrix} n_X \\ n_Y \\ n_Z \end{pmatrix} = \begin{pmatrix} \frac{F_X}{\sqrt{F_X^2 + F_Y^2 + F_Z^2}} \\ \frac{F_Y}{\sqrt{F_X^2 + F_Y^2 + F_Z^2}} \\ \frac{1}{\sqrt{F_X^2 + F_Y^2 + F_Z^2}} \end{pmatrix}$$

(3-25)

The strike and dip of the fold surface at point P_0 (illustrated in Figure 3.3.2) can be calculated as:

$$\mathbf{strike} = \theta_f - \frac{\pi}{2} = \mathbf{Arc\,tan}\left(\frac{n_Y}{n_X}\right) - \frac{\pi}{2}$$

$$\mathbf{dip} = \phi_f = \mathbf{Arc\cos}\left(n_Z\right)$$

(3-26)

To compare the orientation of a fracture-polygon to that of a fold, one can calculate the angle α between the unit normal vector of the fracture plane, $n_p = (n_{X,p}, n_{Y,p}, n_{Z,p})$, and the unit normal vector of the surface that describes the fold curvature at the polygon center, $n_f = (n_X, n_Y, n_Z)$. This angle can be calculated from the dot product of the two vectors:

$$\vec{n}_p \bullet \vec{n}_f = \left|\vec{n}_p\right|\left|\vec{n}_f\right|\cos\alpha = (1)(1)\cos\alpha = n_{X,p}n_X + n_{Y,p}n_Y + n_{Z,p}n_Z$$

$$\alpha = \mathbf{Arc\cos}\left(n_{X,p}n_X + n_{Y,p}n_Y + n_{Z,p}n_Z\right)$$

(3-27)

To compare the strike of a fracture-polygon to the local strike of a fold, one can calculate the difference α_{strike} between the angle of azimuth of the polygon, θ_p, and the angle of azimuth of the fold surface at the polygon center (θ_f in Figure 3.3.2):

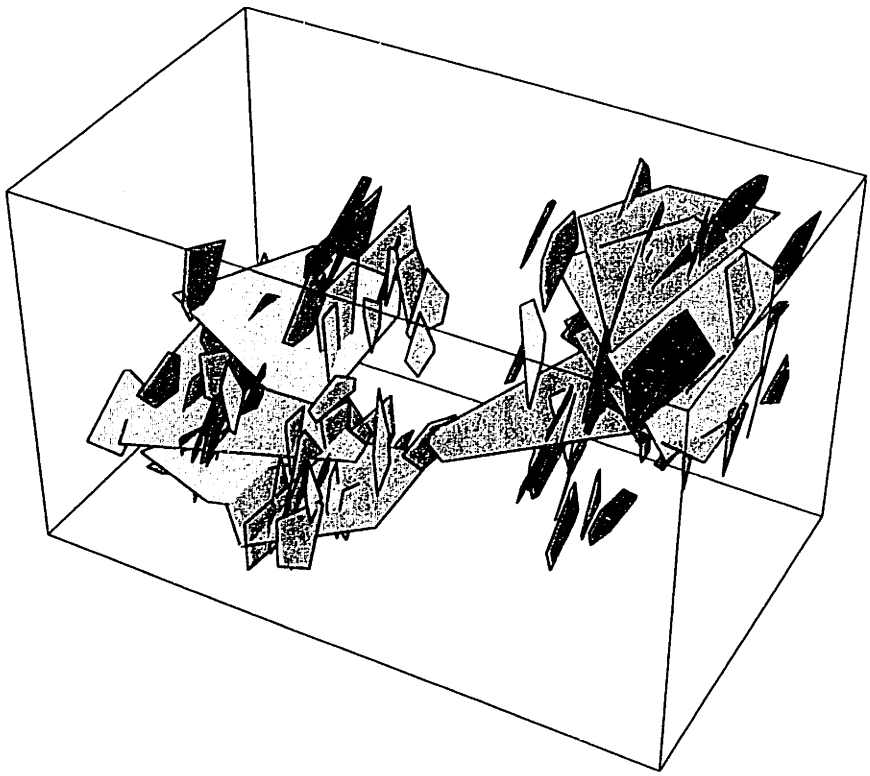


FIGURE 3.3.1 Fracture system modeling via superposition of hierarchically related fracture sets (generation with computer program GEOFRAC). Set 1 (blue): an independent set; Set 2 (yellow): a dependent set, clustered around Set 1.

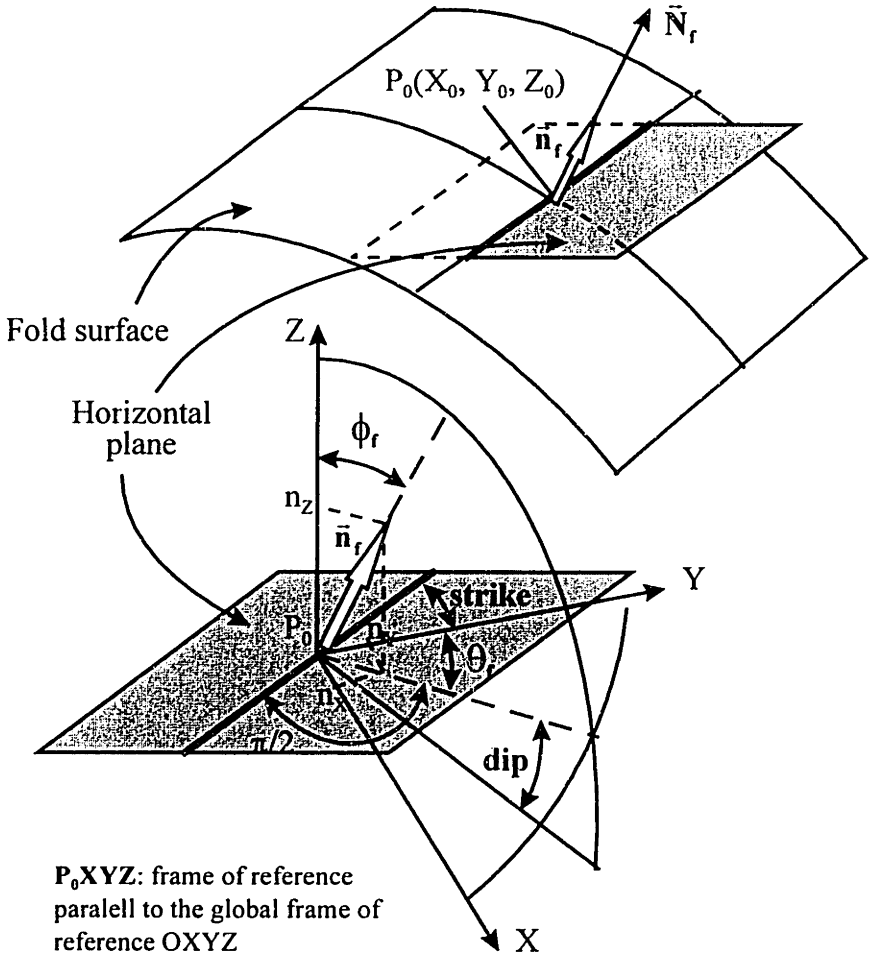


FIGURE 3.3.2 Diagram illustrating the calculation of local dip and strike of fold.

$$\alpha_{strike} = \left|\theta_p - \theta_f\right|$$

(3-28)

To compare the dip of a fracture-polygon to the local dip of a fold, one can calculate the difference α_{dip} between the angle of latitude of the polygon, ϕ_p, and the angle of latitude of the fold surface at the polygon center (ϕ_f in Figure 3.3.2):

$$\alpha_{dip} = \left|\phi_p - \phi_f\right|$$

(3-29)

Once these angles are calculated, they need to be checked against the specified relationships of the fractures to the fold. For example, a small angle α in Equation 3-27 indicates that the fracture is subparallel to the fold surface. An angle α_{strike} close to 90° in Equation 3-28 means that the fracture strikes approximately orthogonal to the local fold strike (i.e. parallel to the local slope of the fold surface). An angle α_{strike} close to zero or 180° in Equation 3-28 means that the fracture strikes approximately parallel to the local fold strike (i.e. orthogonal to the local slope of the fold surface). A small angle α_{dip} in Equation 3-29 indicates that the fracture has approximately the same dip as the fold at that location.

If the orientation of a fracture-polygon does not conform to the specified relationships between fractures and fold, the polygon is rotated. For example, rotation by dip (so that the new fracture dip is subparallel to the local fold dip) is performed by assigning a new latitude angle ϕ_{new} to the polygon:

$$\phi_{new} \propto \mathbf{U}\left[\phi_f - \delta\alpha_{dip}, \phi_f + \delta\alpha_{dip}\right]$$

(3-30)

where $\delta\alpha_{dip}$ is a small angle of allowed deviation of the fracture dip from the local dip of the fold. Rotation by strike is performed by assigning a new azimuth angle θ_{new} to the polygon:

$$\theta_{new} \propto \mathbf{U}\left[\theta_f - \delta\alpha_{strike}, \theta_f + \delta\alpha_{strike}\right] + \mathbf{C}_s \frac{\pi}{2}$$

(3-31)

where $\delta\alpha_{strike}$ is a small angle of allowed deviation of the fracture strike from the local strike of the fold, or from the horizontal direction orthogonal to the local strike of the fold. The coefficient C_s is either C_s=0 or C_s=1.0, depending on whether the fracture is rotated to be concentric or radial to the fold, respectively. If necessary, a completely new orientation of the fracture can be generated (in a frame of reference where the local normal vector to the fold is assumed to be the mean polar direction). After calculating the new dip or/and strike, the fracture-polygon is rotated. Rotation of a polygon means that new 3D coordinates are

calculated for every vertex (formulas for rotation in the global frame of reference of the model are given in Ivanova 1995).

In summary, the following algorithm for generation of a fracture set related to a fold by strike and/or dip is currently implemented in the model:

1) Primary stochastic process: fracture planes are generated according to a spherical PDF around a mean pole direction related to the fold axial plane. For example, the mean direction of tensile joints is either orthogonal to the axial plane, or parallel to the fold hinge (see Figure 2.1.9). The mean orientation of a set of strike-slip faults is oblique to the fold axial plane (see Figure 2.1.11). Fracture orientation variation can be modeled with Fisher PDF (assuming a large coefficient k for subparallel fractures) or partial uniform PDF (assuming small angle ϕ_{max} for subparallel fractures). The fracture planes generated in this way are related to the general stress field: compression orthogonal to the fold axial plane.

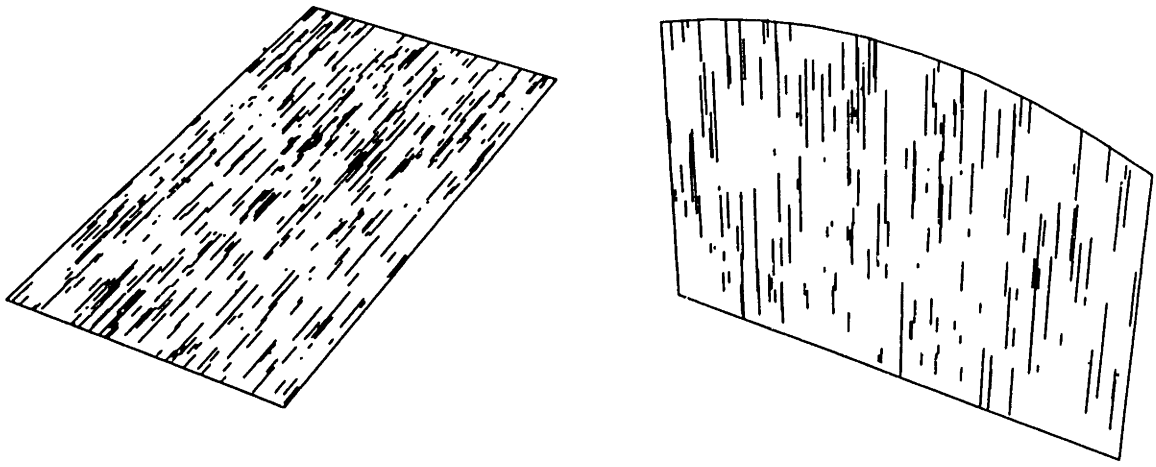
2) Secondary stochastic process: the fracture planes generated with the primary process are subdivided into polygons via the tessellation and marking procedures. Different fracture intensity (hence different zone marking probability of polygons) can be assigned in strata bounded by curved surfaces which represent formation contacts. For example, in the generation of the fracture system illustrated in Figure 3.2.10a only polygons between two bounding surfaces have been retained. Also, the polygons that intersect either surface have been truncated and only the portions above the underlying and below the overlying surface have been retained.

3) Tertiary stochastic process: the strike and dip of every polygon-fracture produced with the primary and secondary processes are compared to the strike and dip of the fold surface at the center of the polygon. Fractures with orientations that do not conform to the specified relationship between the fracture set and the fold are assigned new strikes and/or dips and are rotated.

Figure 3.3.3 shows an example, generated with the program GEOFRAC, of a set of tensile joints parallel to the hinge of a cylindrical fold. Figure 3.3.3a illustrates the initial step in which the polygons were generated following a partial uniform PDF with mean pole orientation orthogonal to the fold axial plane. The horizontal cross section shows that the fractures strike parallel to the fold axial plane. The vertical cross section shows that the fractures are very steeply dipping. Figure 3.3.3b depicts the system from Figure 3.3.3a after rotation has been applied. The horizontal cross section has not changed much: the traces still strike parallel to the fold axial plane. However, the vertical cross section shows that the fractures are not only steeply-dipping, but also approximately orthogonal to the fold surface.

a) HORIZONTAL CROSS SECTION

VERTICAL CROSS SECTION



b) HORIZONTAL CROSS SECTION

VERTICAL CROSS SECTION

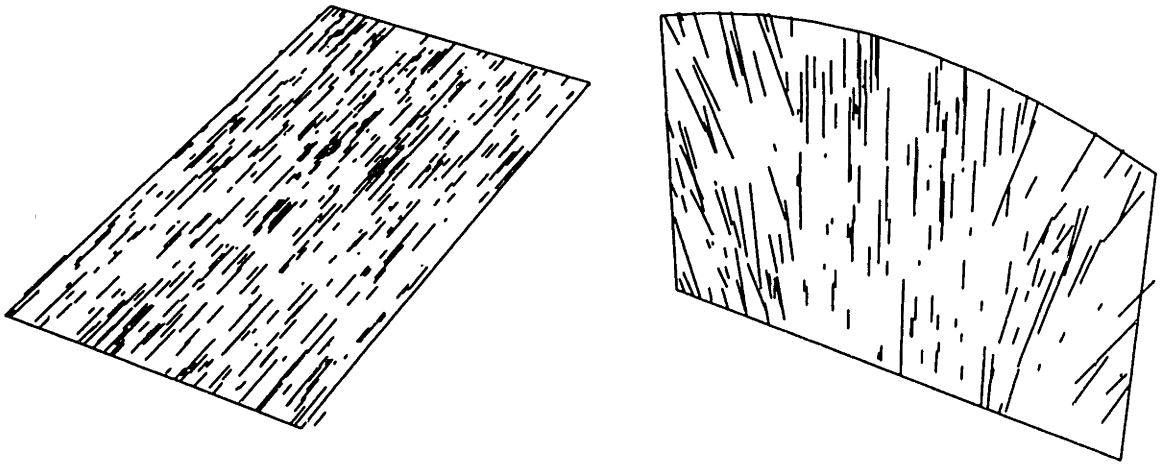


FIGURE 3.3.3 Simulation of a set of tensile fractures parallel to the axial plane of a fold (computer generation with program GEOFRAC): a) initial fractures related to the general stress field; b) rotation of fractures by dip.

Figure 3.3.3 illustrates the importance of correct identification of the geologic setting before fracture modeling. The two horizontal cross sections are very similar, but the three-dimensional geometry of the systems in Figure 3.3.3a and b is different. If the inference procedure were based only on horizontal trace outcrops without considering the existence of a fold, it could be incorrectly assumed that all fractures were vertical and did not intersect at depth.

Because of the great variety of fracture sets related to folds, many geometric algorithms, other than the rotation described above, can be incorporated in the 3D model. **Figure 3.3.4** schematically illustrates a simple geometric procedure for generating bedding-plane fractures in flexural folds. The contact surfaces between the beds are described by 3D functions of the type of Equation 3-23. All bedding surfaces are "unfolded" to planar polygonal regions. Then every planar region is subdivided into fractured polygons and intact zones by a Poisson line tessellation. Finally, all planar regions are transformed back to their folded shapes and the fracture centers are automatically located along the bedding surfaces.

3.3.3 Fracture systems associated with crustal faults

Fracture systems associated with crustal faults (discussed in Section 2.2) include primary faults and numerous secondary fracture sets. Modeling of the geometry of sets of primary faults (normal, strike-slip, or thrust) is relatively easy. Usually there are one or two sets that form angles of less than 45° with the direction of the maximum compressive stress (see Figure 2.2.1 and Figure 2.2.3). Commonly, one of the sets is major and the other one is minor. The faults in a set are subparallel to one another and usually do not intersect the faults from the other set. The variation of orientations in a primary fault set can be described by Fisher PDF (with a high coefficient k) or by partial uniform PDF (with a small angle ϕ_{max}). The mean pole orientation is orthogonal to the planes of maximum shear (**Figure 3.3.5**). When modeling the minor set, one should also apply a procedure to check fault intersections with the primary set. This procedure involves a termination probability, and high probability of rejection of minor faults that intersect major faults.

The stress field associated with crustal faults usually varies with time. Creation of new fractures often changes the local principal stress directions in the vicinity of primary faults, although the general shear direction along them remains the same. Numerous secondary, tertiary, etc. fault and joint sets form between and near primary faults. Therefore, the first important step in modeling the secondary and younger sets is to determine their hierarchy, i.e. the sequence of their formation.

Once the hierarchy of fracture genesis is established, the modeling volume and the mean orientation of a fracture set is defined by the fractures of older sets, including the primary faults. **Figure 3.3.6** illustrates this process with two examples.

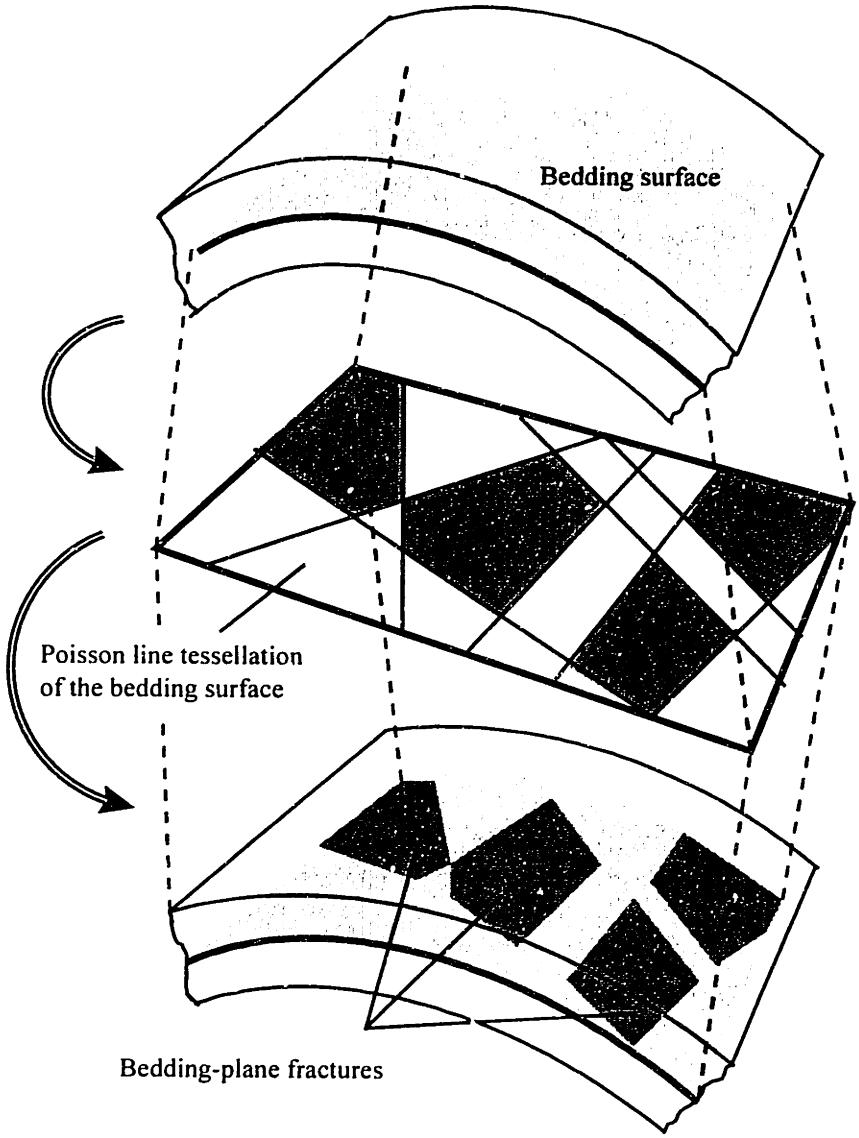


FIGURE 3.3.4 Algorithm for generation of bedding-plane fractures in flexural folds.

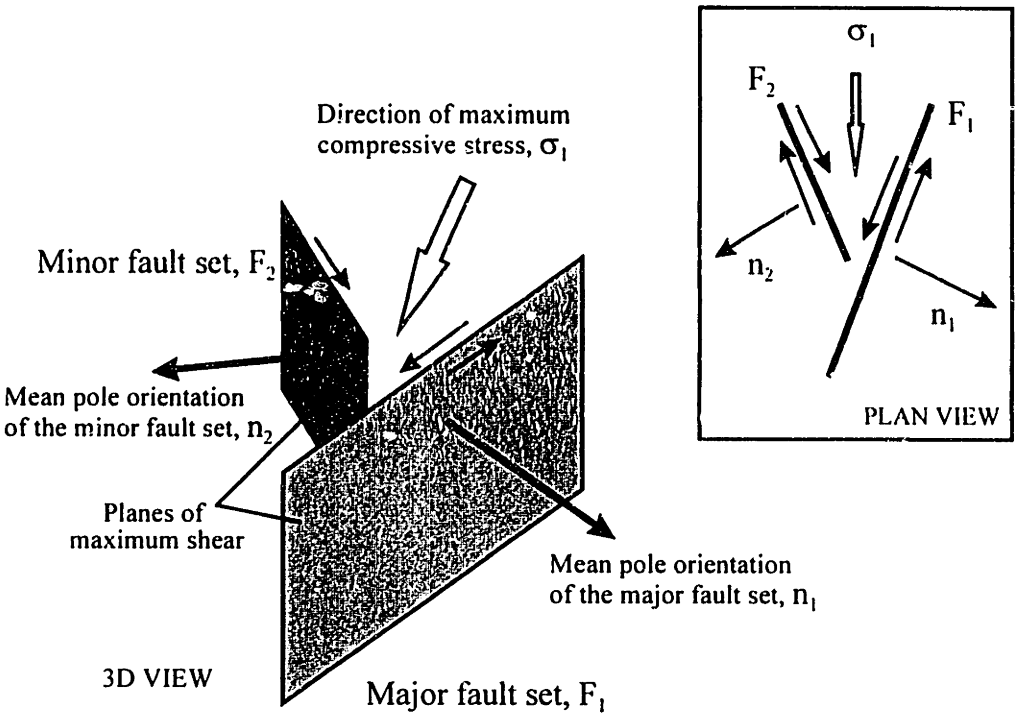
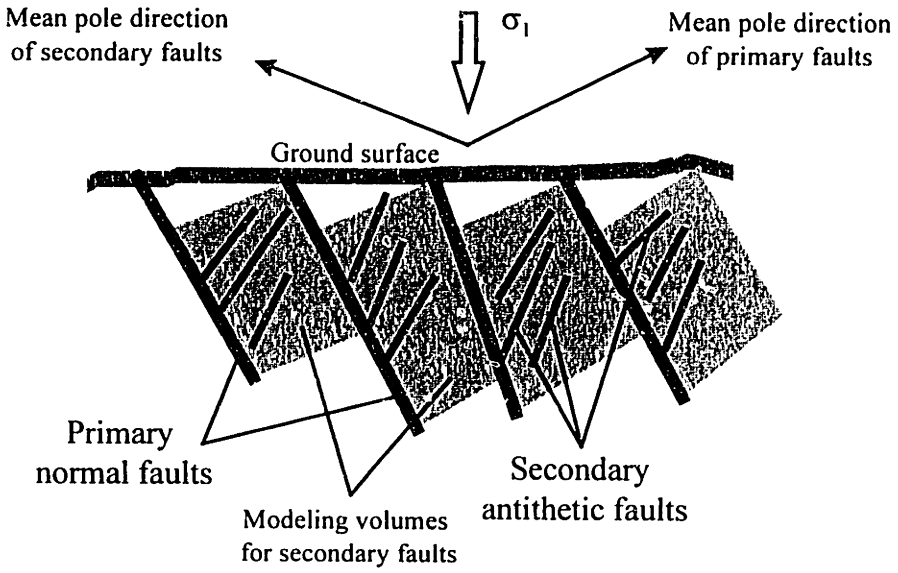


FIGURE 3.3.5 Mean pole orientation of primary fault sets related to the directions of maximum compression and maximum shear.

Figure 3.3.6a depicts the modeling volumes and mean pole orientations of a set of secondary antithetic faults, dependent on a set of primary normal faults. The secondary faults form inside the blocks enclosed between the primary faults. The intensity of the antithetic fault set is highest immediately above the normal faults. Most antithetic faults terminate at an underlying normal fault. Figure 3.3.6b illustrates the modeling volumes and mean pole orientations for secondary and tertiary fractures in strike-slip fault zones (similar to those shown in Figure 2.2.26). The modeling volumes of secondary and tertiary fractures are enclosed between the primary and secondary ones, respectively.

After determining the modeling volume and the mean pole orientation, the variation of fracture orientations in a dependent set can be modeled with Fisher or partial uniform PDF. The fracture intensity is generally highest in the regions adjacent to older fractures. The translation procedure of the tertiary process of the model can be applied to avoid coplanarity of fractures. Rotation is usually not necessary since the fractures are not related to any other structures except to the older faults (which is accounted for by choosing the correct mean pole orientation of the set).

a) NORMAL AND ANTITHETIC FAULTS



b) STRIKE-SLIP FAULT ZONES

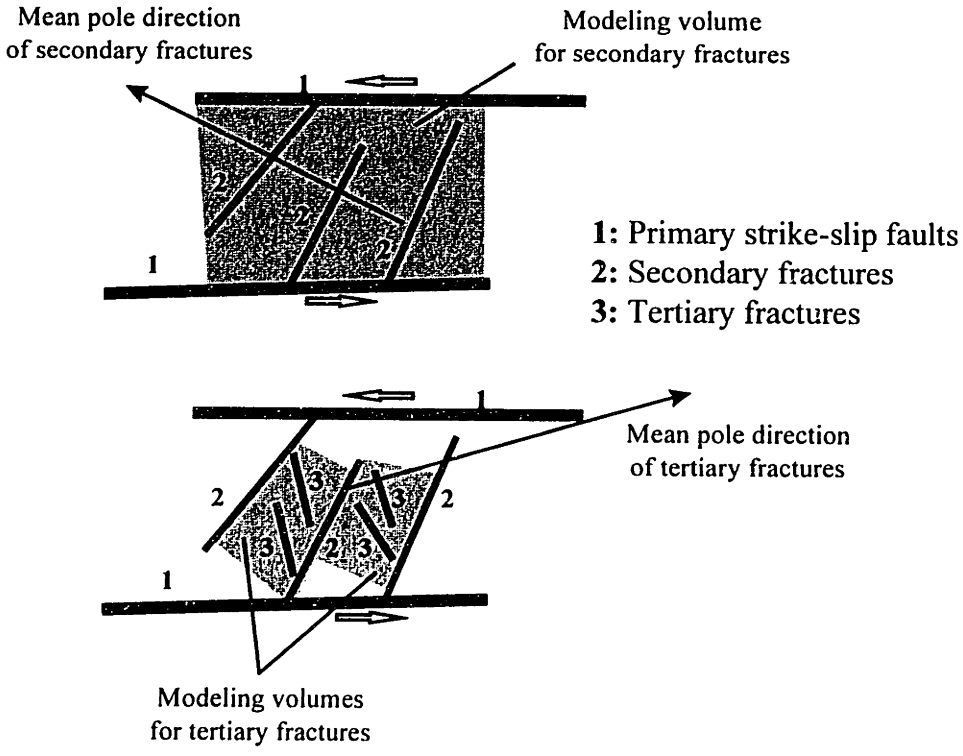


FIGURE 3.3.6 Determination of modeling volumes and mean orientations for fracture sets dependent on primary faults: a) normal faults and secondary antithetic faults; b) strike-slip faults, secondary and tertiary fractures.

The main emphasis in modeling fracture systems related to crustal faults has to be placed on calculating modeling volumes of various 3D shapes. The currently implemented modeling volume (Figure 3.2.3) has a fixed shape: it is bounded by a horizontal (datum) plane, four vertical planes, and a cubic top surface. The next step should be to program a generic modeling volume, enclosed between surfaces of any shape. Currently the computer program GEOFRAC implements some algorithms that calculate intersections of polygons and projections of polygons onto one another. Using these algorithms, secondary modeling volumes can be determined by calculating the 3D regions enclosed between random polygons (primary fractures) that either intersect or overlap (hence can be projected onto one another).

3.3.4 Fracture systems in remote tension

The fracture systems, typical for remote tension (reviewed in Section 2.3), consist of numerous subparallel joints, orthogonal to the direction of the least principal stress σ_3. Therefore, the mean pole orientation of fracture sets can be assumed to coincide with the direction of σ_3. The orientations of all fracture planes in the primary stochastic process can be assumed to be the same (orthogonal to σ_3). More realistically, however, the variation of fracture plane orientations can be described by a spherical PDF that produces clustering of the poles close to the mean pole direction, for example:

- One-parameter Fisher PDF with a high value of the coefficient k (e.g. k>20);
- Partial uniform PDF where the latitude angle ϕ varies between 0 and a small value ϕ_{max}.

Both spherical PDFs generate fracture plane normal vectors that form small angles with the mean pole direction. The one-parameter Fisher PDF is the best option: when the coefficient k is high, the probability of generating orientations that are very close to the mean direction is much higher than the probability of generating orientations that deviate by a larger angle from the mean.

During modeling of fracture intensity it is important to take into account the lithology of the rock. For example, typical tensile fracture systems in igneous rocks contain a few widely-spaced large joints and numerous smaller joints and extension cracks clustered around them. In sedimentary rocks, both the extent of tensile fractures and their spacing are often approximately equal to the bedding thickness. Two algorithms are suggested in the next two paragraphs in order to realistically represent the intensity of fractures in remote tension.

First, when modeling large primary fractures which inhibit the development of other large fractures in their vicinity, one can apply zone marking probability (as defined in Figure 3.2.8). In the generation of new fractures, low probability should

be assigned to the zones immediately surrounding already generated fractures. Thus it is ensured that the distances between large primary fractures are not smaller than a specified value (for example, equal to the bedding thickness in sedimentary rocks).

Second, small cracks and secondary fractures, clustered around the large primary fractures, can be reproduced according to the procedure illustrated in **Figure 3.3.7**. Once the primary fractures are generated, new modeling volumes are calculated adjacent to them, and the secondary fractures are generated only in these secondary volumes. **Figure 3.3.8** illustrates the cross section of a tensile fracture system produced with GEOFRAC according to this algorithm. Figure 3.3.8a shows the outcrop traces of relatively large primary fractures. Figure 3.3.8b shows the outcrop traces of all fractures, after generation of secondary, relatively small, fractures in secondary volumes adjacent to the fractures in Figure 3.3.8a. The cross section in Figure 3.3.8b is very similar to horizontal outcrop traces of vertical joints observed in igneous rocks (for example, Figure 2.3.5).

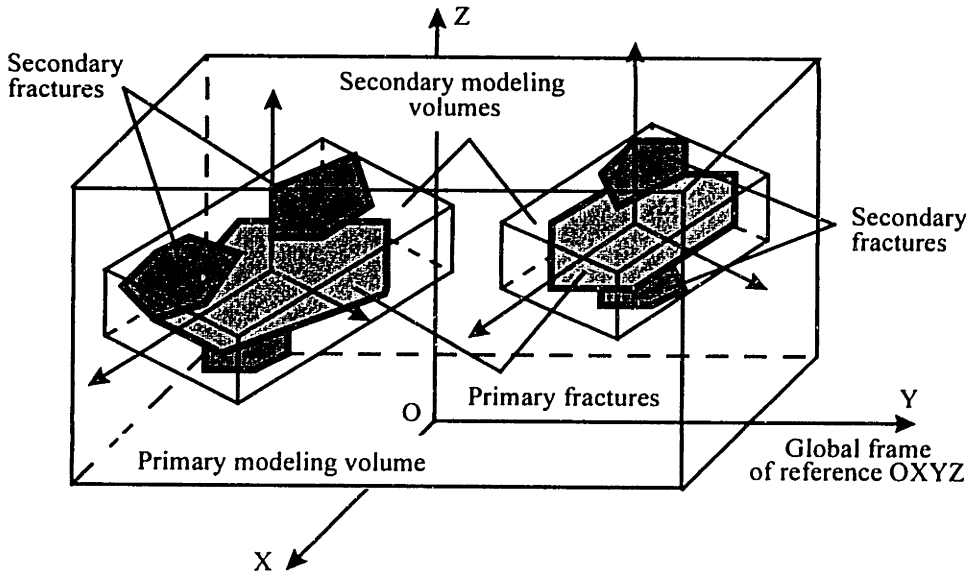
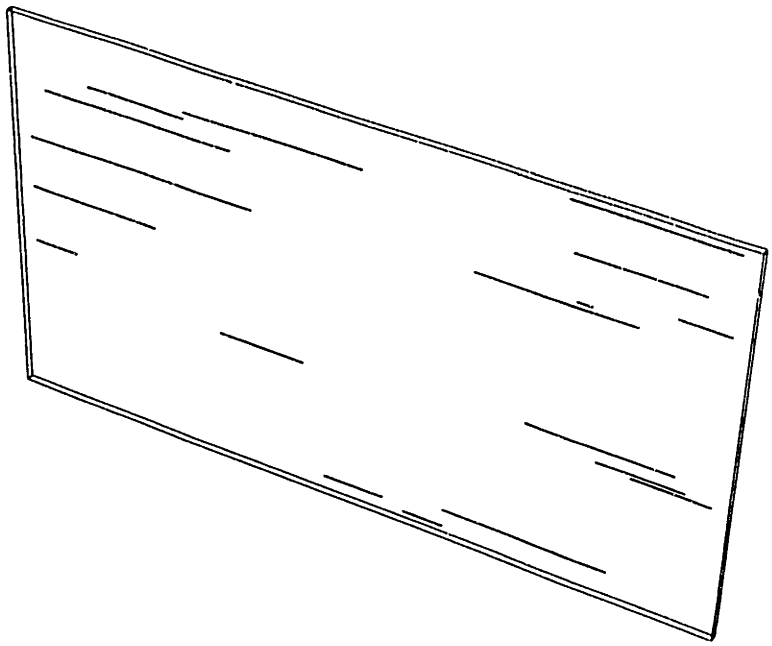


FIGURE 3.3.7 Modeling of fracture intensity in remote tension: primary and secondary modeling volumes.

a)



b)

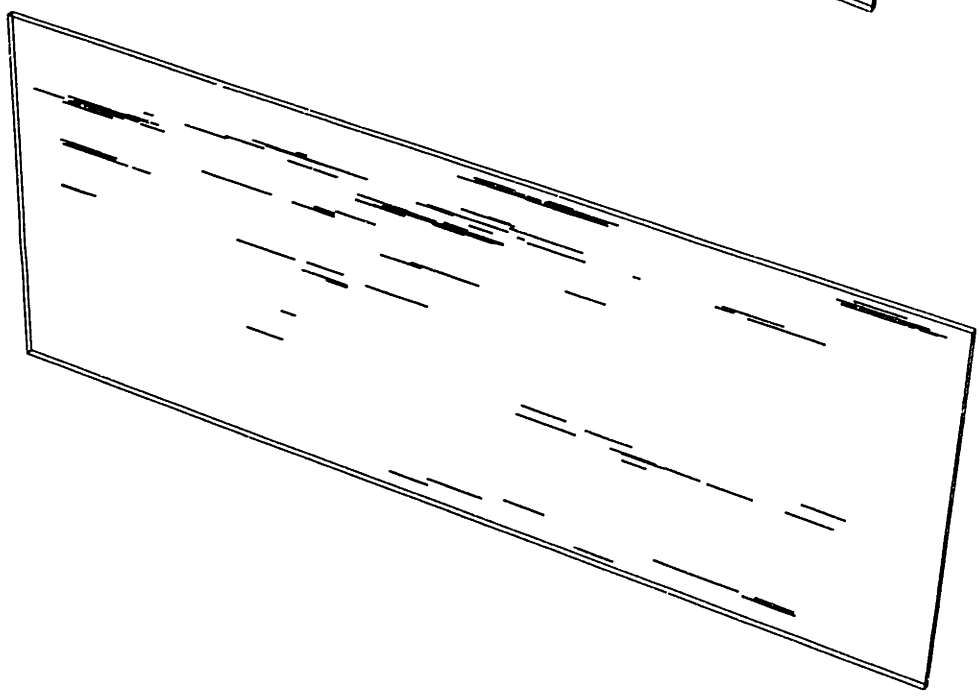


FIGURE 3.3.8 Trace outcrops of a tensile fracture system, generated with program GEOFRAC according to the algorithm in Figure 3.3.7: a) original tension fractures; b) fracture propagation and formation of small secondary fractures clustered around the primary ones.

3.3.5 Thermal fracture polygons

Thermal contraction joints that form polygonal patterns in plan view (described in Section 2.4) are rather unique fracture systems, and as such they deserve a unique model. The orientations of the joints are known: they are orthogonal to the free surface of the cooling material and to the contact surfaces of that material with the country rock. Therefore, the emphasis should be on developing procedures for modeling the polygonal patterns in plan view.

The polygonal patterns defined by cooling joints are strikingly similar to polygons created by the Voronoi-Dirichlet tessellation on a plane, illustrated in **Figure 3.3.9** [Upton & Fingleton 1985]. This tessellation is based on an underlying Poisson point process and Delaney triangulation (created by drawing all line connectors between nearest neighbor-points). The Voronoi polygons are created by the intersections of the orthogonal bisectors of the triangulation segments.

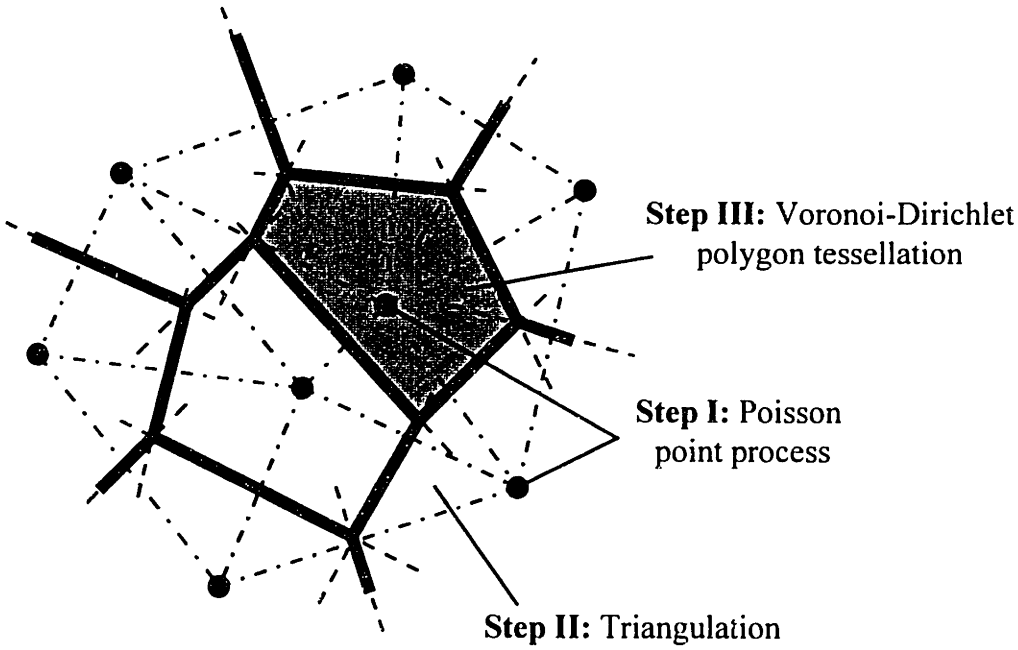


FIGURE 3.3.9 Algorithm of generation of Voronoi-Dirichlet polygon tessellation.

The expected number of polygon vertices, E[N], and the mean and standard deviation, E[A] and σ_A, of the polygon size (area) of the Voronoi-Dirichlet tessellation are:

$$E[N] = 6$$

$$E[A] = \frac{1}{\rho}$$

$$\sigma_A = \sqrt{E[A^2] - (E[A])^2} = \sqrt{\frac{1.28}{\rho^2} - \left(\frac{1}{\rho}\right)^2} = \sqrt{\frac{0.28}{\rho^2}} = \frac{0.529}{\rho} \qquad (3\text{-}32)$$

where ρ is the intensity (number of points per unit area) of the underlying Poisson point process on the plane [Meijering 1953; Gilbert 1962].

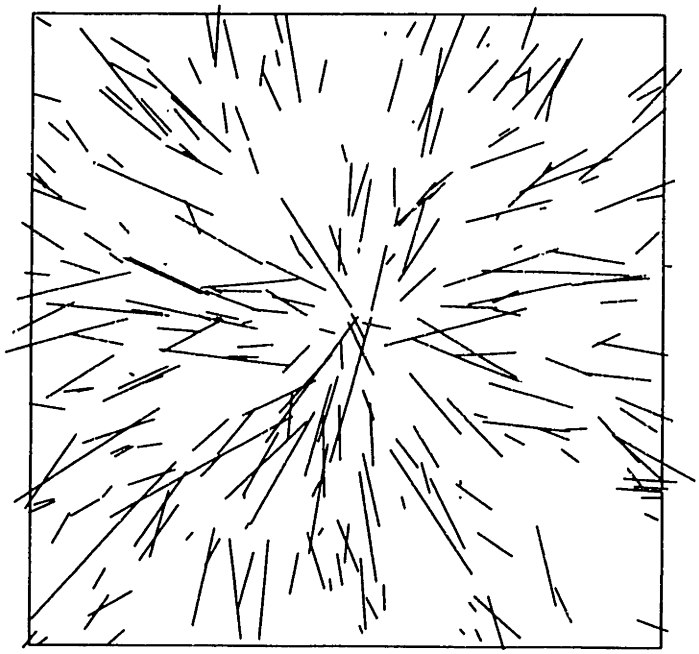
In order to model polygons with more regular hexagonal shapes, one needs to control the minimum distances between Poisson points via an inhibition process [Diggle 1983]. Inhibition can be performed by assigning circular zones around the Poisson points, and discarding every newly-generated point that falls within such a zone. Regular hexagonal intersections are typical for columnar joints in the inside of lava flows. The diameters of the columns in the inside of lava flows are generally larger than the diameters of the columns near the free surface.

In summary, columnar joints can be modeled with the following procedure. First, Voronoi-Dirichlet tessellation is generated on the free surface. Another Voronoi-Dirichlet tessellation, based on an inhibited Poisson point process of lower point intensity, is generated for the inside of the modeling volume. Next, the joints are assumed to be perpendicular to the surfaces. It is important to define joint termination, so that closely spaced joints in the upper part of the modeling volume gradually evolve into more widely spaced joints at depth. Joint terminations can be assigned to a given horizontal layer at depth (representing entablature in cooling lava flows). Alternatively, every column can be modeled as a prism, bounded by a small polygon at the top, a large polygon at the bottom, and inclined joint surfaces on the sides.

3.3.6 Fracture systems in central structures

Fracture systems in and around central intrusive and extrusive structures (reviewed in Section 2.5) include various sets of joints and faults (some of them are observed in other geologic environments as well). Some algorithms, developed in the context of geologic settings such as folds, faults, and remote tension, can be applied for modeling of specific fracture sets related to central structures. For example, the 3D shape of a dome can be described by a function of the type of Equation 3-23. Rotation algorithms developed for folds (Section 3.3.2) can be successfully applied for generation of radial and concentric fractures (dipping outwards or inwards toward the center) around a dome. **Figure 3.3.10** illustrates horizontal cross sections of computer-generated fractures related to a dome: a radial pattern (Figure 3.3.10a), and a concentric pattern (Figure 3.3.10b).

a)



b)

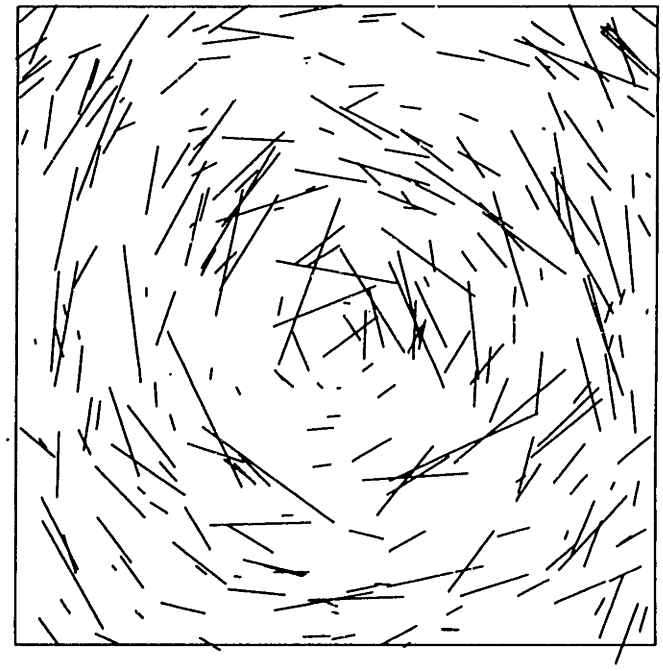


FIGURE 3.3.10 Horizontal outcrops of fracture systems related to a circular dome (generated with computer code GEOFRAC): a) radial fractures; b) concentric fractures.

In Figure 3.3.10, the dome surface is part of a hemisphere, centered in the middle of the horizontal area. The surface is described by Equation 3-23 using coefficients $c_5 = c_7$, and $c_1 = c_2 = c_3 = c_4 = c_6 = c_8 = c_9 = 0$. The synthetic systems in Figure 3.3.10a and b are similar to natural systems of radial dikes (Figure 2.5.6a) and ring dikes (Figure 2.5.12), respectively.

Doming creates numerous intersecting normal and reverse faults in the strata overlying the intrusion (Figure 2.5.3 and Figure 2.5.4). Therefore algorithms designed for fracture systems associated with crustal faults (Section 3.3.3) can be applied for generation of fracture systems related to central structures. Also, the algorithms designed for fractures in remote tension (Section 3.3.4) can be used for modeling of tension joint sets associated with sheet intrusions (such as the dikes in Figure 2.5.9).

Figure 3.3.11 depicts two algorithms, suggested for modeling of fracture sets typical for large igneous intrusions. For example, the orientations of tension joints in the host rock, created by upward pressure of expanding magma, can be modeled with Bingham PDF (Figure 3.3.11a). On the other hand, cooling of the magma creates several sets of fractures, including flat-lying joints inside the igneous intrusion (see Figure 2.5.8). The variation of pole orientations of flat-lying joints can be described with Fisher PDF (Figure 3.3.11b).

3.4 DISCUSSION OF THE 3D MODEL

The stochastic processes implemented in the model are inherently related to the underlying mechanics and geology. The input parameters for generation of fracture system geometry are quantities known from rock mechanics and geology. Model parameters include the number of fracture sets, inferred from field data, and for every one of them:

1) Mean orientation, relative to some global directions or to the orientation of geologic structures (folds, faults);
2) Possible variation of the orientations of potential fracture planes around the mean direction;
3) Fracture intensity: cumulative fractured area per unit rock volume;
4) Mean fracture size and size variation;
5) Relationships of fracture intensity and orientation to geologic structures (folds, faults), bedding planes, formation contacts, etc.

It is very important to develop an inference procedure for the model parameters from field data. One aspect of the inference procedure, the relationship of the geometry of a synthetic fracture system to the mathematical processes that are used to generate this system, was addressed in this chapter. Desired fracture-polygon size, shape, and cumulative area per volume were linked to the intensity of the Poisson line and plane processes.

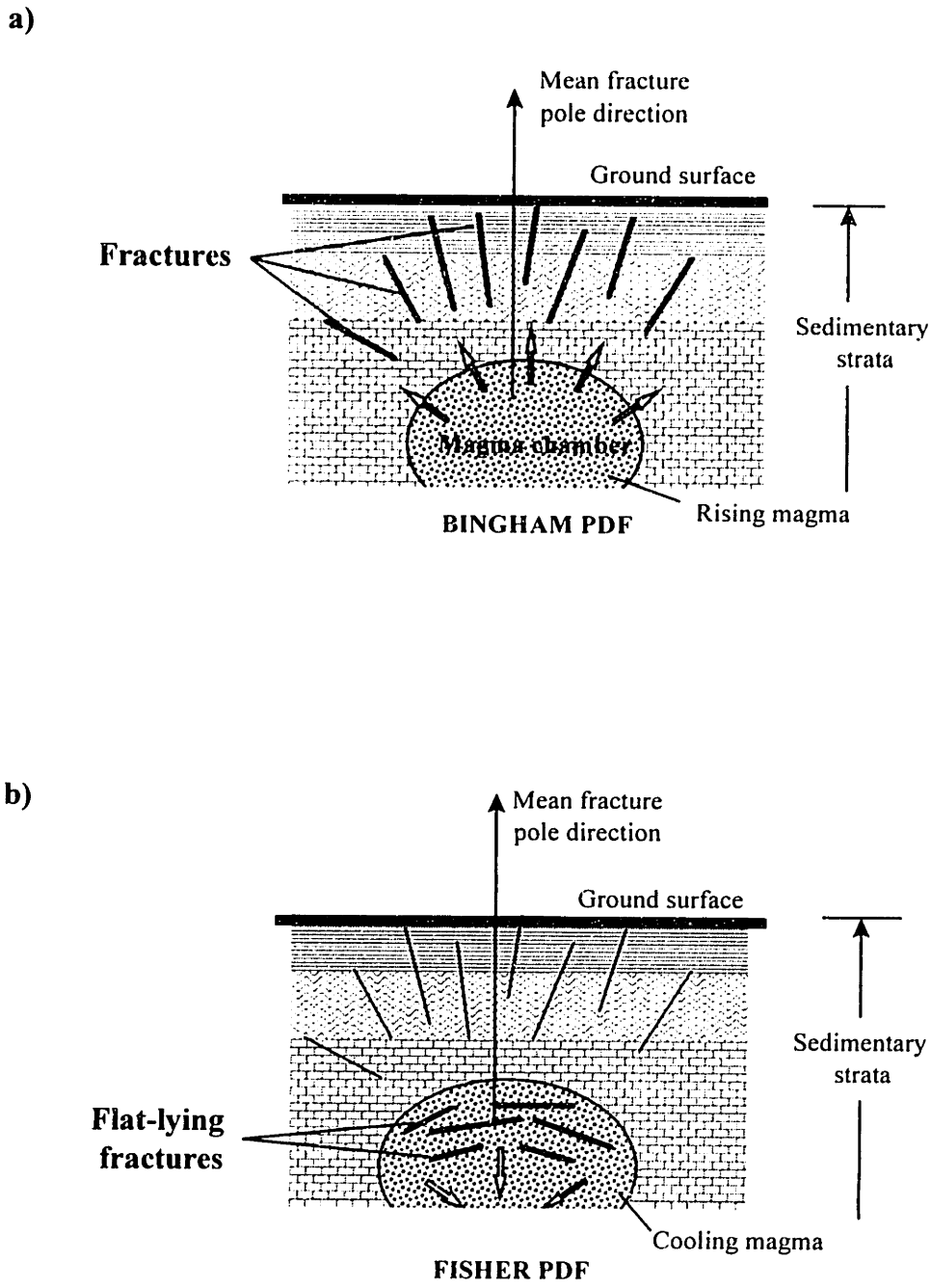


FIGURE 3.3.11 Suggested PDFs that can describe the variation of fracture poles related to central structures: a) Bingham PDF: steep fractures in overlying sediments due to rising of magma; b) Fisher PDF: flat-lying joints in the intrusion due to cooling of magma.

Another aspect of the inference procedure is the inference of fracture orientations from the principal stress directions, defined by the geologic mechanisms, and the inference of fracture intensity from the magnitude of stresses and strains and the mechanical properties of the host rock. To compare a fracture system, generated with the 3D model, to an actual fracture system, one can use available field exposures of fractures in boreholes (logs and cores) and surface outcrops (see case study in Chapter 4).

The stochastic processes of the model represent important relationships between fracture system geometry and underlying mechanics, in terms of fracture orientation, fracture intensity, and relation of fractures to geologic structures. Further development of the stochastic processes, as well as use of new algorithms can enhance the representation of important fracture system characteristics. There are two major areas for further improvement of the model: (1) development of specific procedures for modeling of fracture systems in different geologic settings (discussed in Section 3.3), and (2) enhancement of the methods for modeling of fracture intensity.

Poisson line tessellation is currently implemented in the 3D model for subdivision of planes into intact and fractured areas. However, some other tessellations may enable even greater flexibility in the modeling of fractures with various shapes and sizes. For example, the Voronoi/Dirichlet tessellation (Figure 3.3.9) produces polygons with expected number of vertices E[N]=6 (this may be a better shape for representing fractures). The sizes of polygons, produced by the Voronoi tessellation depend on the intensity of the underlying point process. Small, clustered fractures can be produced by a high intensity of the point process; whereas large polygons of uniform sizes can be created by an inhibited point process (in which points are not closer than a certain distance from one another).

Figure 3.4.1 illustrates another method that can enable modeling of various fracture size distributions. The desired distribution (e.g. lognormal, exponential, or, if available, a field histogram of fracture sizes) can be normalized so that it fits under the numerically established PDF (Figure 3.2.5). Then in every size interval polygons can be retained as fractures with probability P_f defined by the value of the desired PDF, or discarded with probability $1 - P_f$, defined by the difference between the desired PDF of fracture sizes and the empirical PDF of "good" polygon sizes.

Marking of polygons according to relative size, such as that illustrated in Figure 3.4.1 and in Table 3.2.3, decreases the possible cumulative area of coplanar fractures per unit area on potential fracture planes (coefficient γ). A higher value of γ can be obtained if, instead of discarding polygons with bad shapes, they are combined into aggregates to form larger polygons, as shown in **Figure 3.4.2**.

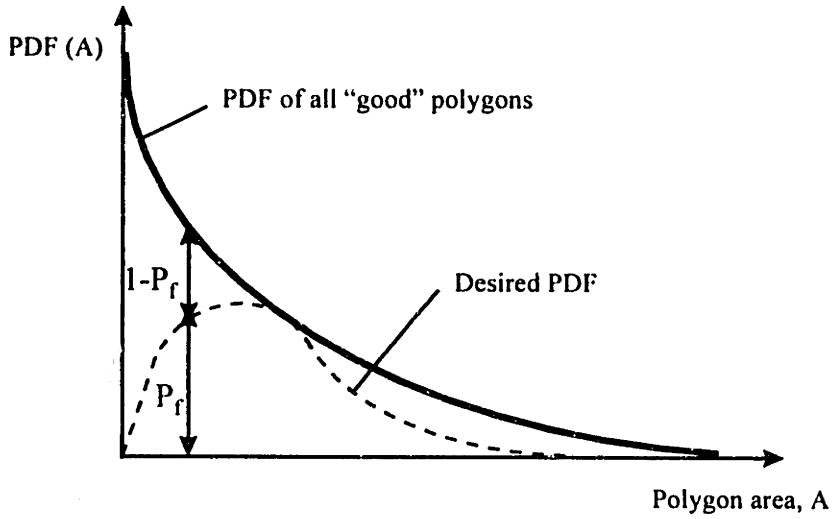


FIGURE 3.4.1 Marking of polygons with probability P_f in order to obtain a specified PDF of fracture sizes.

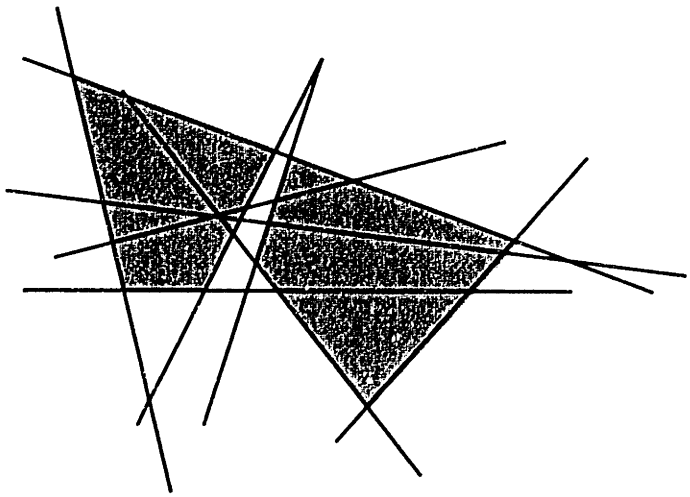


FIGURE 3.4.2 Aggregation of adjacent polygons into large fractures.

Another method for subdividing planes into polygons is the newly developed variation of the Poisson line tessellation, called Fractal Line Tessellation (FLT). **Figure 3.4.3** illustrates the FLT algorithm. First, the original plane is divided into large polygons by a small number of Poisson lines, and some of the polygons are marked as fracture zones according to Equation (3-11) (stage I in Figure 3.4.3). Next, a Poisson line tessellation and a marking process is performed only in the fracture zones (stage II in Figure 3.4.3). This procedure is repeated in a number of iterations (stages III, IV, etc., in Figure 3.4.3).

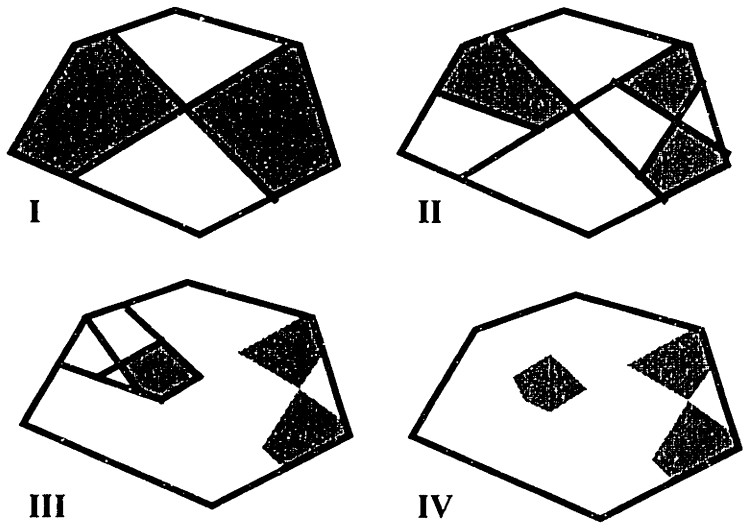


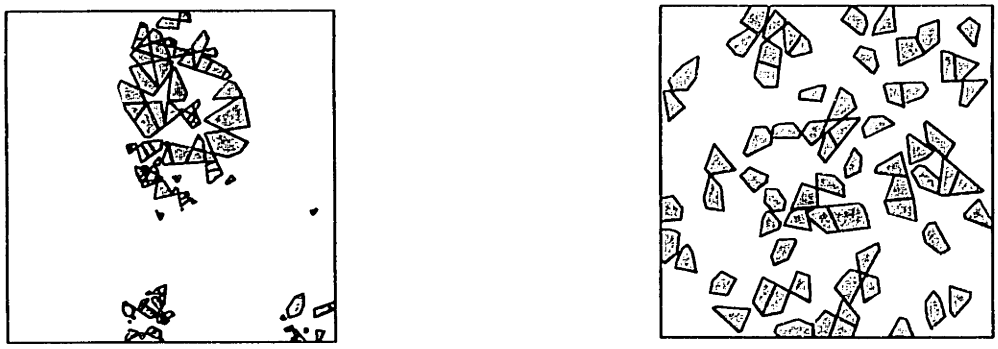
FIGURE 3.4.3 Secondary stochastic process: algorithm of the Fractal Line Tessellation.

The fractal line tessellation can be defined in terms of three parameters: (1) number of iterations, N; (2) fractal dimension, $F=E[A_{i-1}]/E[A_i]$, where $E[A_{i-1}]$ and $E[A_i]$ are the expected polygon areas in two consecutive iterations; and (3) probability P for marking polygons as fractured in each iteration. The FLT allows for modeling of various types of fracture intensity. For example, **Figure 3.4.4**a shows clustering of fractures, obtained in the second iteration of FLT with low P and high F. Figure 3.4.4b shows polygons with approximately the same sizes ($\sigma_A=0.2E[A]$), obtained through five FLT iterations in which only the large polygons were tessellated at every stage. Figure 3.4.4c shows polygons that have very different sizes: some very large polygons, and numerous very small polygons. The size variation in Figure 3.4.4c was obtained through three FLT iterations in which F=2 and only the small polygons were tessellated.

The Poisson line tessellation and the other suggested methods for subdividing planes into fractured and intact areas enable modeling of fracture intensity of type P_{32} (cumulative fracture area per unit volume). This is an advantage of the 3D model, presented here, compared to other models which reproduce separately fracture sizes and locations. For a given PDF of fracture orientations, P_{32} is linearly related to the spacing of fracture intersections by boreholes and to cumulative length of fracture traces per outcrop area. In the model, geologic history is incorporated in the inference of fracture orientations, which allows for high confidence in the assumptions of orientations. Another advantage of the 3D rock fracture system model is its capability to reproduce important fracture system characteristics such as hierarchical fracture genesis and clustering near faults or primary fractures. Chapter 4 presents the application of the model for generation of measured field data.

a)

b)



c)

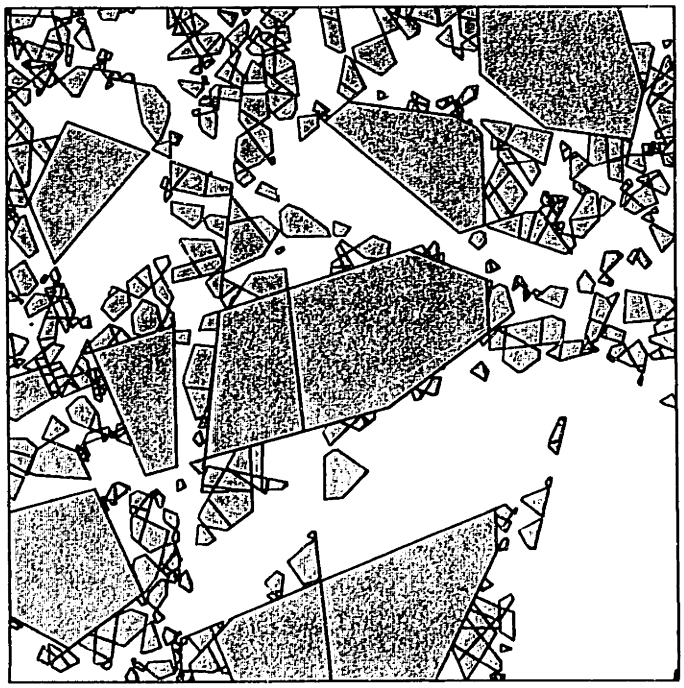


FIGURE 3.4.4 Modeling of fracture intensity with Fractal Line Tessellation (generation with computer program GEOFRAC): a) fracture clustering; b) fractures with similar sizes; c) fractures with dissimilar sizes: very large fractures, surrounded by very small fractures.

Chapter 4

CASE STUDY: YATES OIL FIELD, TEXAS

This chapter presents the application of the 3D fracture system model (Chapter 3) for representation of the fracture system in the oil-producing sedimentary rocks in the Yates field in west Texas. Background of the case study, including location of the site, is presented in Section 4.1. A review of the geologic setting of the Yates field is given in Section 4.2, which includes regional depositional geology of the Permian Basin (Section 4.2.1), and stratigraphy and structure of the reservoir (Section 4.2.2). Available data from previous field testing and numerical modeling are summarized in Section 4.3. Discussion of the field geomechanics in Section 4.4 includes hypotheses on regional stress directions (Section 4.4.1), local drape folding (Section 4.4.2), and fracture-inducing geologic mechanisms in the Yates reservoir (Section 4.4.3). Section 4.5 summarizes the results from the quantitative application of the 3D model for numerical generation of the fracture system in two study areas in the Yates field: Tract 49 (Section 4.5.1), and Tract 17 (Section 4.5.2). Discussion of the results and recommendations for future work on the case study can be found in Section 4.6.

4.1 LOCATION AND BACKGROUND

The Yates oil field is located in the eastern part of Pecos County in west Texas, near the town of Iraan, at approximately 30°52' N latitude and 101°55' W longitude, about 90 miles (144 km) south of the city of Midland [Craig 1990; Tinker et al. 1995]. **Figure 4.1.1**a shows the location of the Yates field and other oil and gas fields (indicated as black areas) within the regional framework of Permian basins and shelves of west Texas and southeast New Mexico. The three major paleogeographic elements, the Delaware and Midland Basins and the Central Basin Platform, are commonly known as the Permian Basin of the United States. Today the two basins and the platform are buried under thousands of feet of younger sediments. The Yates field location on the southeastern tip of the Central Basin Platform is the highest present position of the structure of Permian strata (Figure 4.1.1b).

Figure 4.1.2 shows the horizontal extent of the Yates Field Unit, operated by Marathon Oil Company (MOC). Figure 4.1.2 also represents the structure on the top of the most prolific reservoir formation in the Yates field, the San Andres, deposited in the Middle Permian. The two rectangular areas, Tract 17 and Tract 49, are the selected regions for a joint project between MOC (Midland, TX), Golder Associates, Inc. (Seattle, WA), and MIT, sponsored by the U.S. Department of Energy [Dershowitz et al. 1996, 1997]. The case study on fracture system modeling is part of this broader project on "Fractured Reservoir

Discrete Feature Network Technologies" which aims at development of new technologies for secondary recovery of oil and gas from fractured reservoirs. Data from field testing and the case study itself are limited to the formations above a datum of 800 ft (243 m) above sea level which is the extent of operational rights of MOC (hence to the reservoir known as Yates Field Unit). The objective of the case study, presented in this chapter, is to identify the geologic mechanisms that caused fracturing in the Yates field and to model the fracture system in the reservoir rocks in Tracts 17 and 49.

Table 4.1.1 defines the locations of Tract 17 and Tract 49 within the Yates Field Unit. The values of the coordinates in Table 4.1.1 are from the local survey coordinate system used in the Yates oil field. Horizontal distances are measured in the old Spanish measure "vara" (33.3 in.; 86.64 cm). Conversion to Texas Central State Plan can be accomplished using the transformation rules in **Table 4.1.2**.

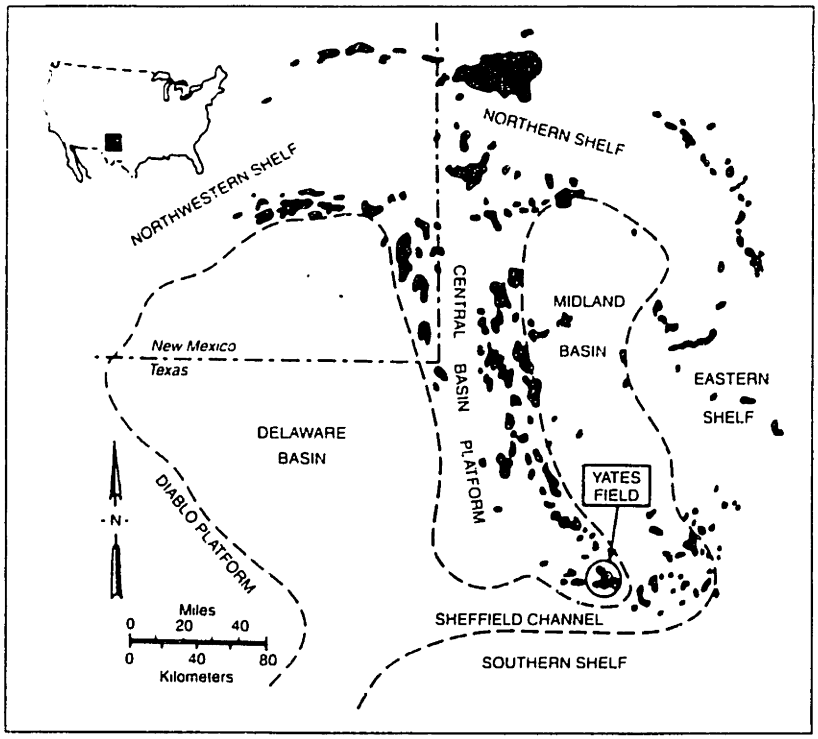
BOUNDARY	TRACT 17 AREA	TRACT 49 AREA
Northern limit, $\mathbf{Y}_{max}$ [vara]	114,696	109,966
Southern limit, $\mathbf{Y}_{min}$ [vara]	112,967	108,123
Eastern limit, $\mathbf{X}_{max}$ [vara]	110,141	114,762
Western limit, $\mathbf{X}_{min}$ [vara]	106,271	112,876
South-to-north dimension	1729 vara [1498 m]	1843 vara [1597 m]
West-to-east dimension	3870 vara [3353 m]	1886 vara [1634 m]

TABLE 4.1.1 Location of the case study areas: Tract 17 on the west side and Tract 49 on the east side of the Yates oil field. 1 vara = 2.777 ft = 0.846 m.

SCALE FACTOR	2.7771151 feet/vara, 0.846 m/vara
CLOCKWISE ROTATION	1.162667 degrees
POINT OF ROTATION	**Yates** [vara] → **Texas Central** [feet] X=112,475.983 → X=1,498,103.600 Y=107,276.091 → Y=441,536.200

TABLE 4.1.2 Conversion factors for horizontal coordinates of the Yates field study areas (Table 4.1.1) from local system to Texas Central State Plan.

a)



b)

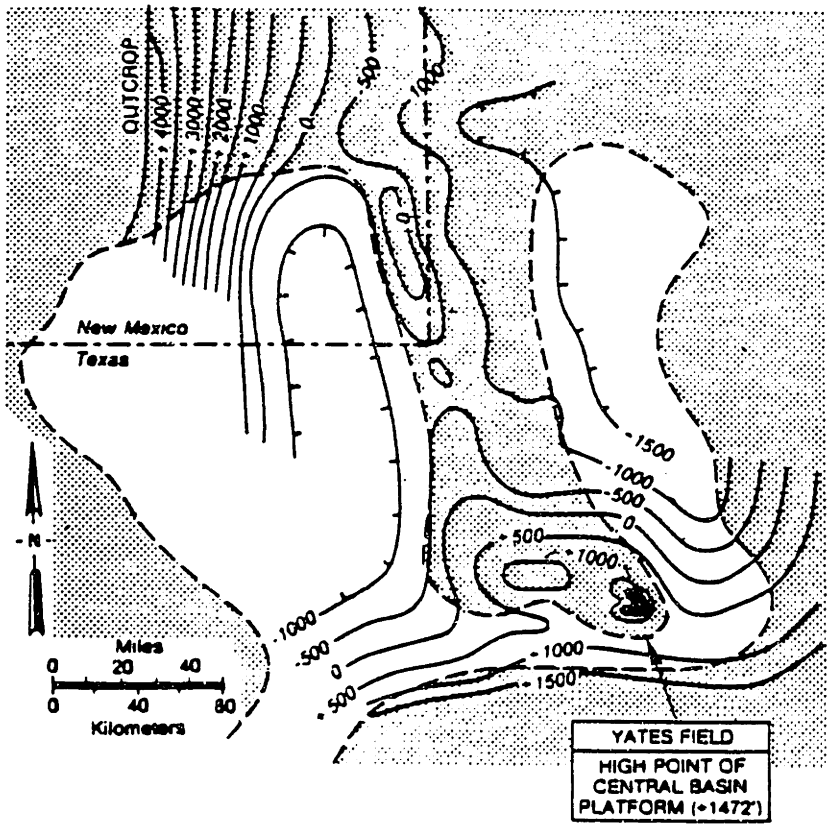


FIGURE 4.1.1 Location of the Yates field within the paleogeographic structural framework (shelves and basins) of the Permian Basin [from Craig 1990]: a) distribution of oil and gas reservoirs of Late Permian age (shown as black areas) in west Texas and southeast New Mexico; b) present day structure of Permian strata in the region and position of the Yates field on the Central Basin Platform.

4.2 GEOLOGIC SETTING

The regional geology of the Permian Basin and the local geology of the Yates field have been investigated in detail by geologists of the Midcontinent Division of Marathon Oil Company in Midland and Iraan, Texas, [Craig 1963; Craig 1990; Wadleigh, pers. comm.]. A computerized 3D geologic model of the Yates field reservoir has been built at the MOC Petroleum Technology Center (PTC) in Littleton, Colorado, [Tinker 1996; Tinker et al. 1995; Tinker & Mruk 1995]. The reservoir characterization model incorporates geologic and engineering data from 1800 wells, 118 cores (total of 23000 ft, i.e. 7010 m, of quantified core description and analysis), and historical production records. The computer model, containing 6.8 million cells and 40 attributes per cell, is created with Landmark's SGM® software (Stratigraphic Geocellular Modeling ®, called Stratamodel). The regional geologic setting of the Yates field in the Permian Basin is reviewed in Section 4.2.1. Section 4.2.2 presents the depositional model and the present day stratigraphy and structure of the Yates field reservoir.

4.2.1 Regional geology of the Permian Basin

A generalized column of the stratigraphy and lithology of the Permian Basin of West Texas and Southwest New Mexico is shown in **Figure 4.2.1**. The inset in Figure 4.2.1 illustrates the position of the reservoir strata of the Yates field within the regional depositional sequence. The rest of Section 4.2.1 presents five major phases of the regional geologic development of the Permian Basin [Craig 1963].

First, second and third phases: Pre-Permian regional deposition and tectonism

During the first phase, Upper Cambrian through Early Ordovician, 3000 ft (914 m) of sediments, mostly conglomerate, coarse sandstone, and sandy carbonates, were deposited on a broad, low relief shelf of Precambrian rocks. During the second phase, Middle Ordovician through Mississippian, relatively uniform deposition of sediments occurred within the limits of a broad marine basin (the Tobosa Basin) that spanned the entire west Texas and southeast New Mexico. Sediments from the second phase included: up to 2000 ft (610 m) Ordovician fine sandstone, green shale and minor limestone; more than 1500 ft (457 m) Silurian and Devonian limestone and dolomite; and up to 1200 ft (366 m) Mississippian limestone, interbedded with shale. The sediments were greatly reduced in thickness and geographical extent by erosion at three regional unconformities.

Intense structural deformation (mostly folding and normal faulting), characterized the third, Pennsylvanian, phase in the Permian Basin [Craig 1963]. Based on regional fold and fault patterns in southeast New Mexico and west Texas, Hills (1970) derived the predominant stress directions during the period of tectonic activity from the Late Mississippian through the Pennsylvanian and to the Early Permian.

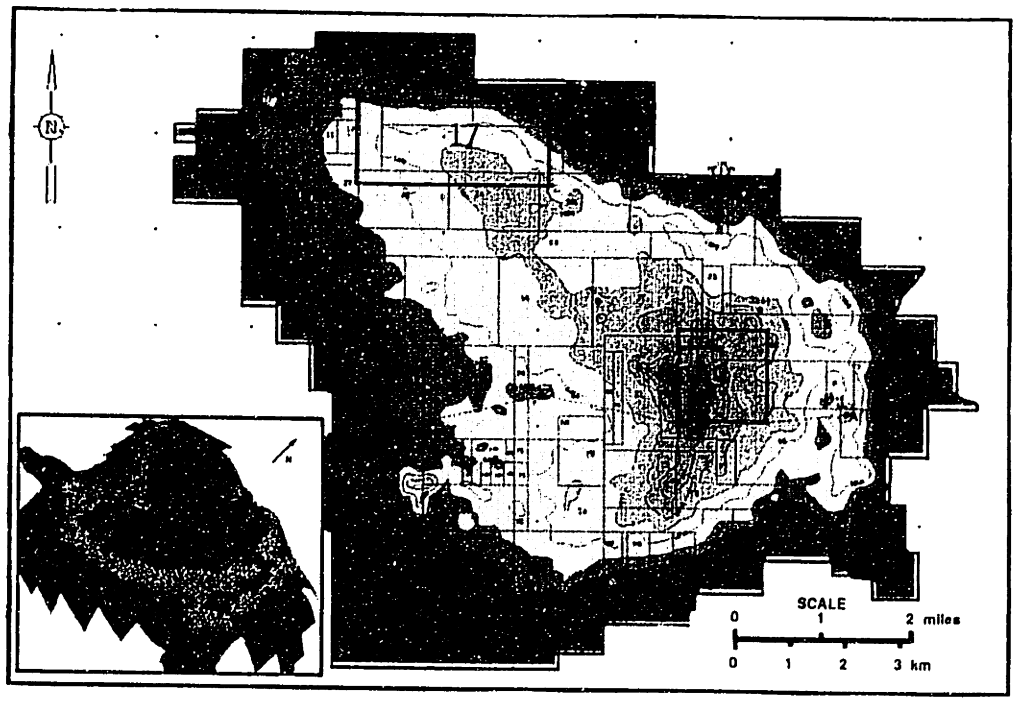


FIGURE 4.1.2 Horizontal extent of the Yates Field Unit and locations of the case study areas, Tract 17 and Tract 49, superimposed on a structure map of the San Andres formation top [from Tinker et al. 1995]. Inset: 3D view of the San Andres top. Colors represent approximate present day fluid levels. Blue: water zone; green: oil-water transition zone and oil column; orange and red: secondary gas cap.

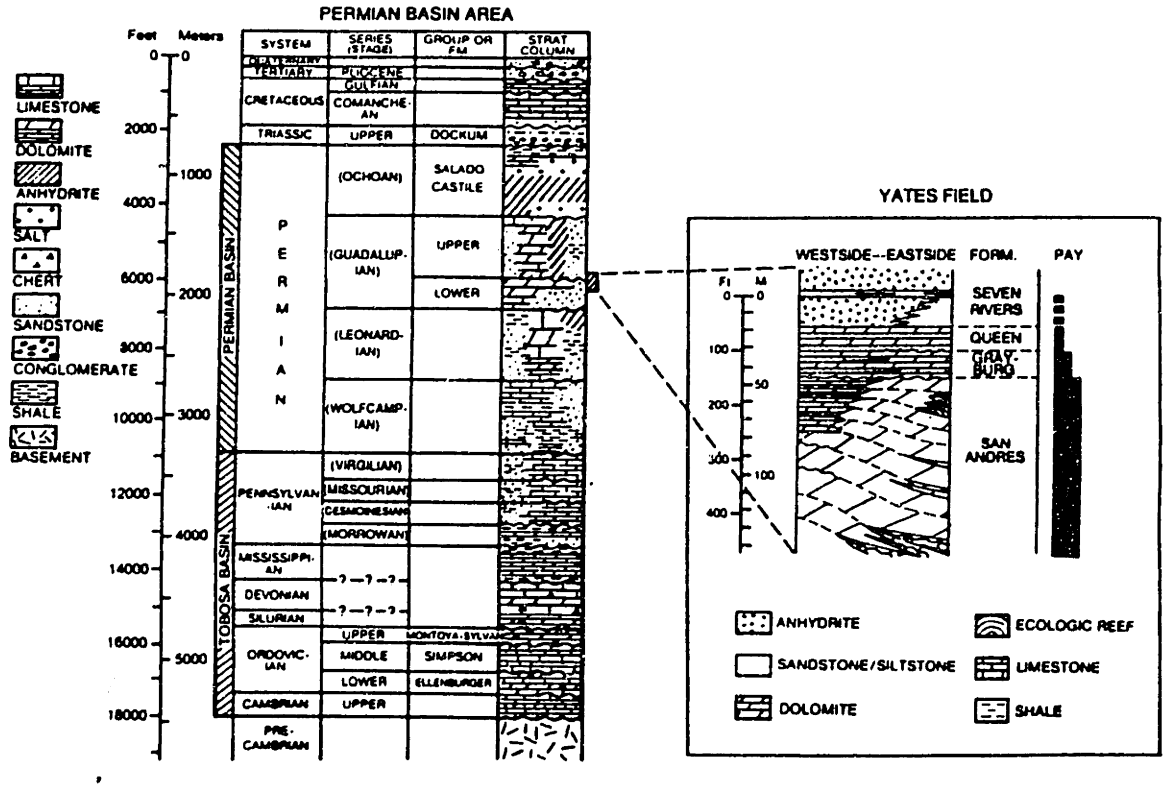


FIGURE 4.2.1 Generalized stratigraphic and lithologic column for the Permian Basin [from Craig 1990; after Galley 1958]. Inset: stratigraphic column for the Yates field with depth measured from Seven Rivers M-horizon (silty unit deposited as a horizontal layer during the Upper Guadalupe).

Figure 4.2.2a illustrates the tectonic forces active from the Late Mississippian to the Middle Pennsylvanian. During that period the maximum compressive stress initially acted approximately east-west and later rotated to about N65°E. The regional structures, created by the older east-west regional compression, include two (possibly conjugate) sets: a major fault set, striking N50°-65°W, and another widely spaced set of faults striking N55°-80°E. The later northeast-southwest compression created a system of folds striking N23°-35°W in Paleozoic rocks buried under Early Pennsylvanian sediments.

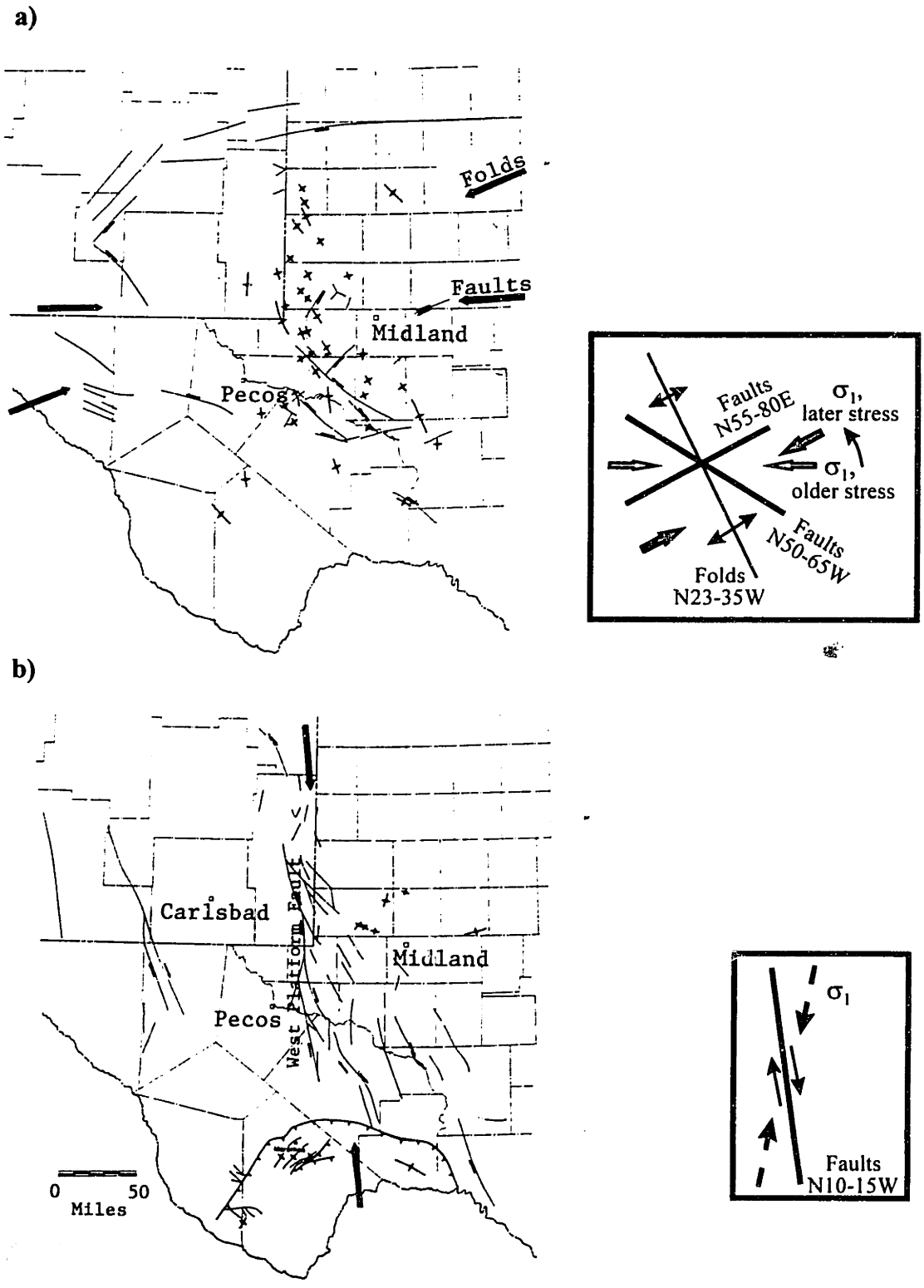


FIGURE 4.2.2 Stress directions in the Permian Basin during the last period of regional tectonism [after Hills 1970]: a) Late Mississippian to Middle Pennsylvanian; b) Late Pennsylvanian to Early Permian. Solid lines with arrows show directions of maximum compression derived by Hills. Broken line with arrow indicates maximum compressive direction, interpreted in the current study.

According to Hills (1970), from the Late Pennsylvanian to the Early Permian there was an east-west relaxation and a north-south increase of horizontal compressive stresses. At that time, crustal extension in east-west direction, associated with the sinking of the Midland and the Delaware basins, left only the Central Basin platform emerged above the ancient seas. This last period of tectonic activity in the Permian Basin created a system of N10°-15°W strike-slip faults (shown in Figure 4.2.2 b), the most prominent being a major fault along the west boundary of the Central Basin Platform (called the West Platform Fault). Hills speculates that the maximum principal stress at that period acted slightly west of north, after reorientation of the regional stresses radially around the Marathon salient (in the southernmost part of the area). The strike of the faults, however, is more consistent with a regional maximum compression, acting slightly east of north (according to rock mechanics principles, a fault develops in the plane of the intermediate principal stress, here the vertical stress of gravity, and makes an acute angle with the maximum principal stress direction).

Upwarping of the Central Basin Platform during the Pennsylvanian divided the regional sedimentary basin into the Midland and the Delaware basins, thus creating the structural framework of the Permian Basin, bounded by contiguous shelves on the north, east and west (Figure 4.1.1). According to Craig (1963), in the two basins Early Pennsylvanian coarse sands and shales were overlain by Late Pennsylvanian fine shale, siltstone, and limestone, with a maximum total thickness of 2000 ft (610 m). During the Late Pennsylvanian, for the first time in the Paleozoic, carbonate buildups (predecessors of barrier reefs) formed in the shallow waters seaward from the shelf margins along the eastern flank of the Midland basin and along the Central Basin Platform. A regional unconformity separated the Pennsylvanian rocks from the Permian.

Fourth phase: Permian regional sedimentation

The primary geologic process during the fourth, Permian, geologic phase of the region was sedimentation [Craig 1963; Craig 1990; Hills 1970]. Deposition of sediments continued in the basins and platforms of the tectonic framework, established during the Pennsylvanian. Permian rocks, which had total thickness of more than 20000 ft (6096 m) before erosion, can be grouped into four series: Wolfcamp, Leonard, Guadalupe, and Ochoa. The most important formations are the carbonates, clastics, and evaporites of the Lower and Upper Guadalupe which constitute the reservoirs and seals of the Permian oil and gas fields throughout the region (see Figure 4.1.1a and Figure 4.2.1).

Wolfcamp sediments in the region include sands and clays in the basins and some carbonate buildups along the shelves. By the end of the Wolfcamp, the Permian Basin became structurally stable. An erosional unconformity separates the Wolfcamp from the Leonard on the Central Basin Platform and along the shelves.

For the first time in the Paleozoic history of the region, facies typical for barrier reef environments were deposited during the Leonard. Such "true reef" deposits, unlike the carbonate buildups, form in the zone of surf and restrict the circulation of sea water over the platforms (**Figure 4.2.3**). As a result, fine-grained clastics (e.g. shales) and carbonates, as well as salt, anhydrite, and other evaporites are deposited behind the reefs in the environment of restricted water circulation (called back-reef). Leonard rocks in the back-reef are composed of green, red and gray shale, thin-bedded limestone and dolomite, with some anhydrite. Leonard reefs around the basin rims consist of irregularly bedded, fossiliferous limestone and dolomite, which grade basinward into fore-reef deposits of silliceous shale, clay shale, sandstone and thin-bedded limestone (more than 4000 ft, i.e. 1219 m, in the middle of the Delaware Basin). An unconformity on the Central Basin Platform separates the Leonard from the overlying Guadalupe rocks.

The Guadalupe Series of the Permian Basin can be divided into two units, Lower and Upper Guadalupe, locally separated by an erosional unconformity [Craig 1963]. The vertical and lateral relationships between the major Guadalupe lithofacies are shown in **Figure 4.2.4** and **Figure 4.2.5**, respectively.

The Lower Guadalupe (Figure 4.2.4 and Figure 4.2.5a) consists of three groups characteristic of the major marine depositional environments, i.e. fore-reef, reef, and back-reef [Craig 1963]: 1) the Cherry Canyon Sandstone of the Delaware Basin, a 2000 to 2500 ft (610 to 762 m) sequence of dark, fine-grained basinal sandstone and siltstone; 2) the Goat Seep Reef of the Delaware Basin margins, up to 1200 ft (366 m) limestone and dolomite reef and carbonate buildup; and 3) the San Andres on the Central Basin Platform, approximately 1000 ft (305 m) lagoonal (back-reef) dolomite, grading landward into progressively more anhydritic, shaly, and sandy carbonates.

The Upper Guadalupe (Figure 4.2.4 and Figure 4.2.5b) also includes sediments from the three major environments of marine deposition [Craig 1963]: 1) the Bell Canyon Formation (fore-reef and basin), dark siltstone and fine-grained sandstone with some thin-bedded, dark limestone; 2) the Capitan reef, up to 1200 ft (366 m), irregularly bedded, fossiliferous limestone and dolomite; 3) the Artesia Group, carbonates, clastics and evaporites deposited in the Midland basin which during that period became part of the back-reef environment. The Artesia Group consists of five formations (Figure 4.2.4): a) Grayburg, up to 300 ft (91 m), sandy or silty, locally oolitic dolomite; b) Queen, a 300 ft (91 m) sequence of interbedded red and gray sandstone and sandy dolomite, with some anhydrite and salt; c) Seven Rivers, anhydrite, about 1400 ft (427 m) in the Midland Basin, thinning to 400 ft (122 m) on the Central Basin Platform, with some salt and polyhalite, and thin interbeds of siltstone and silty dolomite; e) Yates, up to 350 ft (107 m) fine quartz sandstone and siltstone, with some shale, anhydrite, salt, and thin-bedded dolomite; f) Tansill, 60 to 320 ft (18 to 98 m) dolomite, changing to anhydrite and salt shoreward.

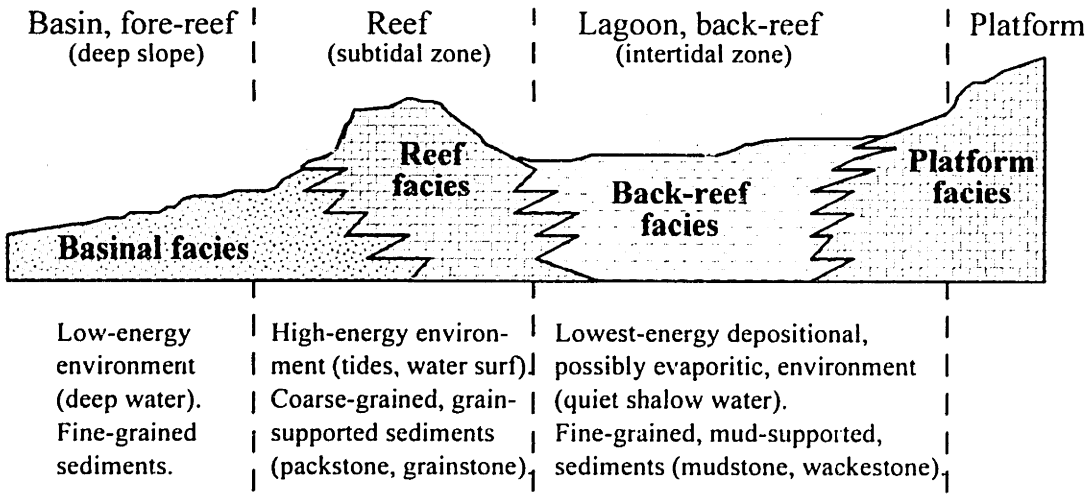


FIGURE 4.2.3 Schematic representation of the reef, back-reef, and fore-reef depositional environments in the Permian Basin.

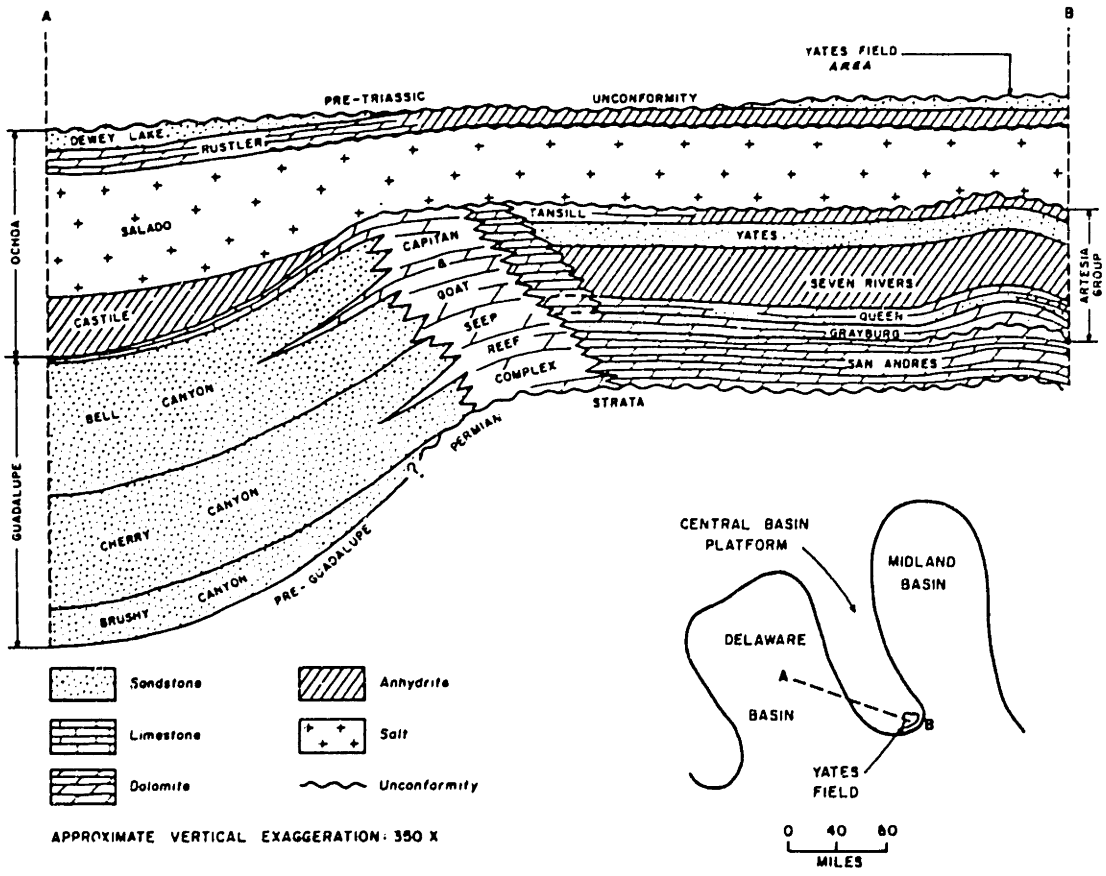
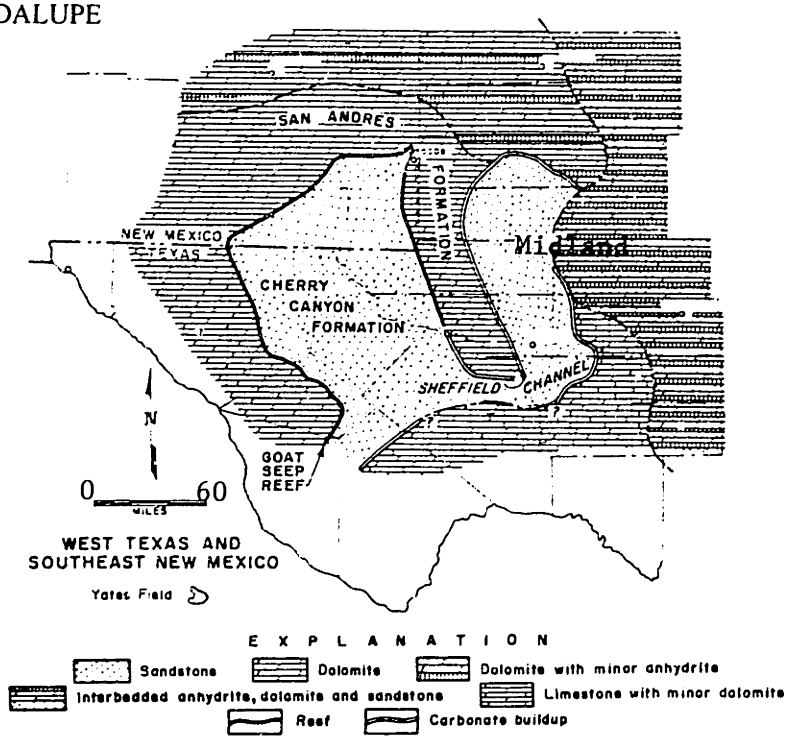


FIGURE 4.2.4 Diagrammatic cross section of Guadalupe and Ochoa series from Delaware Basin to Yates field [from Craig 1963].

a) LOWER GUADALUPE



b) UPPER GUADALUPE

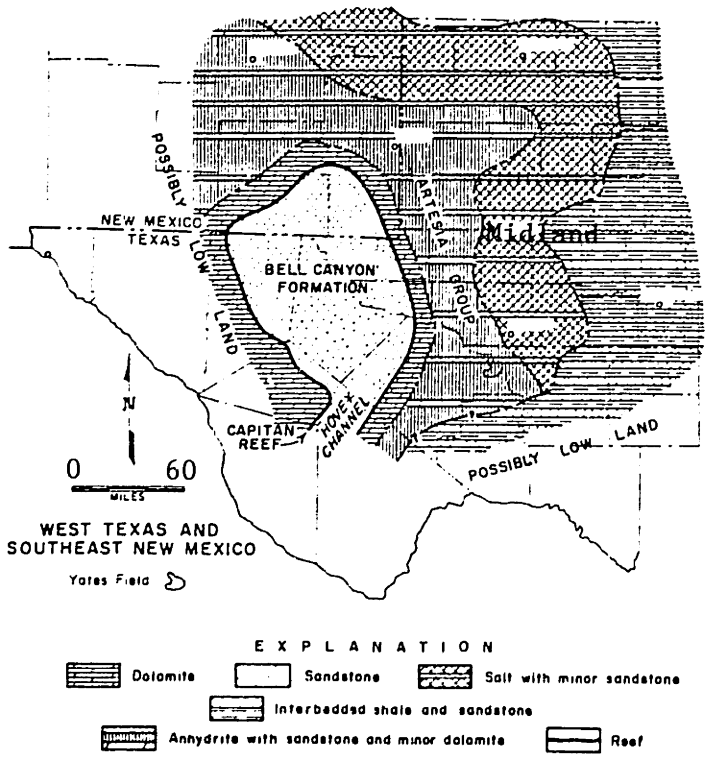


FIGURE 4.2.5 Lithofacies of the Guadalupe series in the Permian Basin [from Craig 1963]: a) Lower Guadalupe; b) Upper Guadalupe.

At the end of the Guadalupe, the entire area became back-reef to reef or carbonate buildup depositional environment. The Late Permian Ochoa Series (shown in Figure 4.2.4) consists of four formations [Craig 1963]: 1) Castille, up to 2100 ft (640 m) in the Delaware basin, banded anhydrite (alternating laminae of anhydrite and calcite); 2) Salado, up to 2200 ft (670 m) in the Delaware basin, salt with small amounts of anhydrite and dolomite; 3) Rustler, 375 ft (114 m) sequence of thin-bedded sandstone, conglomerate, and variegated shale, overlain by interbedded dolomite, anhydrite and salt; 4) Dewey Lake, 280 ft (85 m) in the Midland basin, siltstone and fine-grained sandstone, partly cemented by anhydrite and gypsum. Between the Salado and the Rustler, there was a period of emergence and mild folding that led to deposition of Rustler sediments on the upturned and eroded surface of the Salado salt.

Fifth phase: post-Permian regional sedimentation

During the fifth, post-Permian, phase, the region remained stable. In the Late Triassic, more than 2000 ft (610 m) of stream and lake variegated clay and silt, with some sand and gravel, were deposited on a broad alluvial plain [Craig 1963]. Some Triassic rocks were removed by erosion during the Jurassic. Cretaceous seas submerged the flat regional plain and deposited coarse basinal sands and gravels, overlain by limestone, with total thickness from 800 to 1000 ft (244 to 305 m). At the end of the Cretaceous and during the Early Tertiary, the area west of the Delaware Basin was locally uplifted, folded, and faulted to produce a system of small mountains and detritus-filled valleys. The youngest deposits in the region (Tertiary and Quaternary) included up to 5000 ft (1524 m) alluvial shale, sandstone, conglomerate, sand, and gravel.

4.2.2 Stratigraphy and structure of the Yates field reservoir

The stratigraphic record in the Yates Field area, with an indication of rock types and thicknesses, is summarized in **Table 4.2.1**. The table enlists all formations that are present in the Yates field. Some formations, deposited elsewhere in the Permian Basin, are absent in the Yates field due to non-deposition or erosion.

Depositional model, stratigraphy and lithology of the Permian reservoir

The Yates field reservoir produces from four formations of Middle Permian (Guadalupian) age [Craig 1990; Tinker & Mruk 1995]. The oldest and most prolific reservoir formation is the San Andres which consists of more than 400 ft (122 m) dolomite. The San Andres is the top unit of a 4000-foot-thick (1219 m) sequence of carbonates, deposited on the eroded surface of pre-Pennsylvanian rocks; the sequence also includes 2500 to 3000 ft (762 to 914 m) Early Permian (Wolfcamp and Leonard) rocks, mostly limestone with some shale.

Fifth Phase • **CRETACEOUS (Comanche)** ⇒ WASHITA-FREDERICKSBURG ⇒ TRINITY • **TRIASSIC (Upper)**	 ⇒ 630 ft (192 m) maximum; limestone, cherty limestone; minor shale and sandstone. ⇒ 290 ft (88 m) maximum; variegated sandstone; minor conglomerate, shale, limestone. • Absent over apex of field; up to 350 ft (107 m); red and green shale, coarse conglomeratic sandstone.
Fourth Phase • **PERMIAN** ◊ **Ochoa** ⇒ RUSTLER ⇒ SALADO ◊ **Upper Guadalupe** ⇒ TANSILL ⇒ YATES ⇒ SEVEN RIVERS ⇒ QUEEN ⇒ GRAYBURG ◊ **Lower Guadalupe** ⇒ SAN ANDRES ◊ **Leonard** ◊ **Wolfcamp**	 ⇒ 100 ft (30 m) maximum; absent from highs; interbedded anhydrite, siltstone, sandstone. ⇒ 100 ft (30 m) maximum; present only on southern side; salt with minor anhydrite, shale, and sandstone; upper surface an unconformity. ⇒ 60 ft (18 m) maximum, 30 ft (9 m) average; anhydrite and dolomite; eroded on high parts. ⇒ 85 ft (26 m) maximum; sandstone, siltstone, and shale; minor anhydrite, dolomite and salt. ⇒ 427 ft (130 m) average; anhydrite; minor siltstone, silty dolomite, and gypsum. ⇒ 47 ft (14 m) average; siltstone, dolomite; minor anhydrite cement in siltstone. ⇒ 53 ft (16 m) average; dolomite; some silty and oolitic dolomite; absent in 1 mile by 1/4 mile area about 1 mile north-northwest from apex. ⇒ 345-450 ft (105 to 137 m); dolomite; upper surface an unconformity; uncertain lower contact due to unreliable deep well control. ◊ 1700-2100 ft (518 to 640 m); limestone (carbonate buildup, minor reefing); shale, chert and dolomite. ◊ 940-1780 ft (286 to 243 m); limestone (carbonate buildup); shale.
Third Phase • **PENNSYLVANIAN**	• 900-1250 ft; fossiliferous, cherty limestone; up to 320 ft sequence of sandstone and shale, interbedded with upper portion of limestone.
Second Phase • **DEVONIAN and SILURIAN** • **ORDOVICIAN (Middle)**	• 330 ft (100 m) maximum; dolomite, some cherty dolomite. • At least 910 ft (277 m); variegated shale, sandstone.
First Phase • **CAMBRIAN (Upper) and ORDOVICIAN (Lower)**	• Estimated 1100 ft (335 m); conglomerate, sandstone, limestone, dolomite.

TABLE 4.2.1 Stratigraphic record of the Yates field [after Craig 1963].

The three oil-producing formations above the San Andres are: the Grayburg silty dolomite, averaging about 50 ft (15 m) within the productive limits of the field; the Queen siltstone and dolomite, about 50 ft (15 m); and a clastic layer, up to 100 ft (30 m), at the base of the Seven Rivers anhydrite [Craig 1990; Tinker et al. 1995]. The sediments of the Grayburg, Queen, and lower Seven Rivers thicken into topographic depressions and thin over topographic highs on the erosional surface of the San Andres. The Seven Rivers anhydrite, averaging about 400 ft (122 m) within the productive limits of the field, forms the seal of the Yates field reservoir. The middle and upper Seven Rivers, and the overlying Yates sandstone and Tansill dolomite have relatively constant thicknesses and lithologies over the field. The Salado salt is present in the Yates field only as a thin stratum surrounding the field anticline. According to Craig (1963), the total thickness of Permian rocks in the Yates field is about 5100 ft (1555 m).

Figure 4.2.6a illustrates a west-to-east stratigraphic section of the 3D geologic model (Stratamodel) of the Yates field reservoir. At reservoir elevations, overlying the datum at 800 ft (244 m) above sea level, the upper San Andres formation is composed of a ramp complex, overlain by four major cycles of carbonate shoals [Tinker & Mruk 1995; Tinker et al. 1995]. The present day geometry of the shoals (typical clinoforms) reflects eastward progradation (i.e. lateral spreading) and aggradation (vertical build-up) during the Permian. In Figure 4.2.5a, the third and fourth cycles are collectively referred to as Cycle 3, since Cycle 4 was almost entirely removed by subaerial erosion which defined the San Andres top as a major unconformity. Paleoelevations of the cycle boundaries (i.e. elevations prior to the formation of the field anticline) are reconstructed according to depths from a datum in a thin layer of silty dolomite in the Seven Rivers formation about 200 ft (61 m) above the top of San Andres, referred to as "M horizon". This layer, which can be mapped throughout the field and along the Central Basin Platform, is considered to have been essentially horizontal during deposition [Craig 1988].

Figure 4.2.6b illustrates the west-to-east distribution of the three dominant lithofacies of the San Andres dolomite [Tinker & Mruk 1995; Tinker et al. 1995]. Each major cycle in the San Andres represents a sequence of three to five minor cycles. The cycle boundaries in the geologic model have been correlated between wells in the west side of the field, based on interpretation of log profiles and core data. The lithofacies in Figure 4.2.6b are described according to the accepted oil industry classification of carbonate facies by Duhnam (1962) in terms of grain size and energy of the depositional environment (illustrated in Figure 4.2.3).

The first major lithofacies of the San Andres (shown in green in Figure 4.2.6b) is present on the upper west side of the Yates field. These sediments consist of vertically stacking, compacted, carbonate and argillaceous mudstones, deposited in a lagoonal and intertidal low-energy environment (see Figure 4.2.3). The

lithology of the San Andres on the upper west side includes clean fine-grained dolomite, interbedded with thin layers of shaley dolomite ("shales") with low porosity and high gamma ray response. Shales are thickest and most extensive at the base of each major cycle, and become thinner and laterally discontinuous near the cycle top. The vertical distribution of shale is explained by the deposition of clays (90% illite and 10% chlorite) in lagoons behind the shoal islands during sea level fall. The shoals graded upward into shallow subtidal mudstone and wackestone, and shallow shelf packstone and grainstone during subsequent sea level rise.

The second major lithofacies of the San Andres, present in the deep west side and dominant on the upper east side of the Yates field, consist of hundreds of vertical feet of clean carbonates with high porosity and low gamma ray response (i.e. low shale content). These carbonates (shown in Figure 4.2.6b in varying colors, representing the porosity) were deposited as higher-energy, shallow subtidal shoals of fusulinid packstones and grainstones (see Figure 4.2.3), and subsequently dolomitized. The San Andres porous, coarse-grained, dolomite on the upper east side is the best reservoir rock in the Yates field. The major cycles, shown in Figure 4.2.6a, cannot be correlated from log and core data on the east side, since the shale layers are completely missing. The major cycles, correlated in the west, are projected into typical clinoformal shoal geometries to the east according to the depositional model of eastward carbonate shoal progradation and aggradation [Tinker et al. 1995; Tinker & Mruk 1995; Tinker 1996].

The rigid grainstones on the east (paleo basinward) side were deposited over deep marine carbonates with significant potential for dewatering and compaction. This third lithofacies (blue in Figure 4.2.6b), present in the lower east side of the field, consist of low porosity, low gamma ray wackestone, typical for the eastern slope and deep ramp. The distribution of the three major lithofacies with different grain sizes and compactibilities has been observed in logs and cores throughout the field, as well as in other fields on the Central Basin Platform [Craig 1990], and in outcrops of the San Andres formation and the Capitan Reef in southeast New Mexico [Sarg and Lehmann 1986].

Table 4.2.2 summarizes the distribution of dominant lithofacies, rock textures, and porosity within the San Andres formation in the Yates field [Tinker & Mruk 1995]. On the west side of the field (Tract 17 area), two major lithofacies are present: (1) argillaceous dolomite ("shales"), and (2) clean dolomites (95% dolomite per rock volume). On the east side of the field (Tract 49 area), dolomite is the predominant mineral (96% of the rock volume). Rock matrix porosity in both areas, generally low in the west and high in the east, is dominated by moldic, fusumoldic, and vuggy types, in addition to the crystalline porosity typical for dolomite (porosity classification by Choquette & Pray 1970).

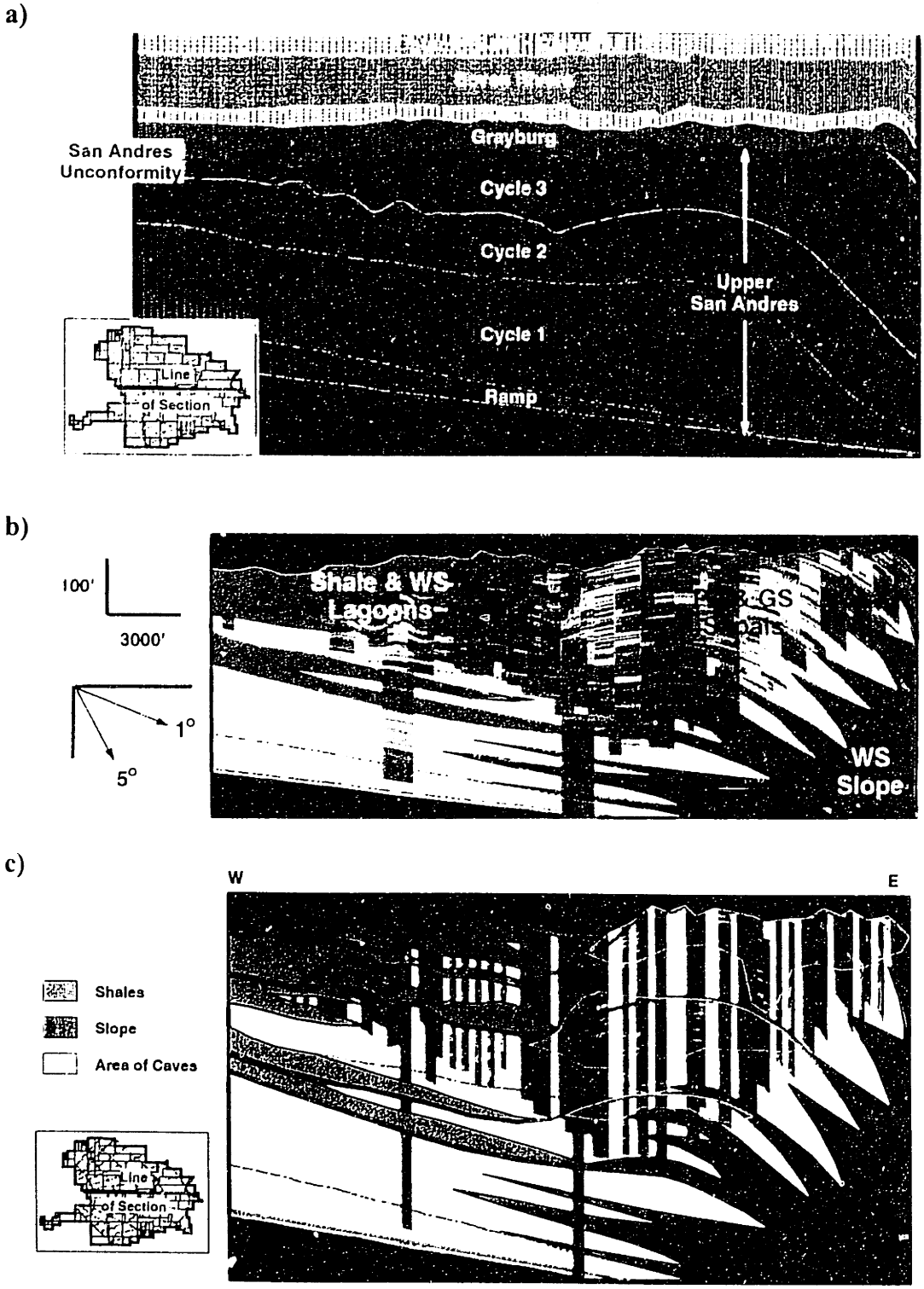


FIGURE 4.2.6 Depositional model of the Yates field reservoir [from Tinker et al. 1995]: a) stratigraphic cross section cut from the 3D geologic model (datum on Seven Rivers M); inset: location of cross section in the field; b) the three major San Andres lithofacies: (1) lagoonal shales and dolomitic mudstone/wackestone (green); (2) dolomitic packstone and grainstone shoals; (3) slope and deep ramp wackestone (blue); c) position of caves (yellow) in the stratigraphic framework.

	SAN ANDRES WEST (TRACT 17)	SAN ANDRES EAST (TRACT 49)
Lithology	"Shales" (argillaceous dolomites; silt and shale 40% of rock volume); clean dolomite; limestone (calcite cement).	96% clean dolomite; limestone (calcite cement).
Texture	Mostly mudstone and wackestone; some packstone.	Packstone and grainstone.
Porosity type	Intercrystalline and micro-intercrystalline; moldic, fusumoldic, vuggy.	Intercrystalline and micro-intercrystalline; moldic, fusumoldic, vuggy.
Comment	Low porosity; high gamma ray.	Higher porosity; low gamma ray (corrected for uranium); best quality reservoir rock.

TABLE 4.2.2 Lithology, texture, and porosity of the San Andres formation in the Yates field [after Tinker & Mruk 1995].

Besides dolomite, limestone is the only other significant lithologic/mineralogic component, present on both west and east side of the field, principally as secondary calcite cement filling the pores of the rock matrix [Tinker & Mruk 1995]. The 3D geologic model of the field (Stratamodel) includes spatial distribution and a model for the origin of the secondary calcite. Crystallization of secondary calcite in pores and fractures is explained by bio-degradation of oils at the paleo water-oil contacts. Under this process, a flat layer of calcite would be deposited at the oil-water contact elevation during a relatively long period of constant sea level.

The top surface of the San Andres formation is marked by a pronounced regional karst unconformity [Craig 1988; Tinker et al. 1995]. The Grayburg sediments (shown in Figure 4.2.6a) onlapped the karst paleosurface of the San Andres during marine transgression, filling topographic lows and thinning over topographic highs. The Grayburg and Queen sediments were deposited as clastic and carbonate mudstones and wackestones in shallow subtidal, intertidal and tidal-flat environments [Tinker & Mruk 1995; Tinker et al. 1995]. Grayburg sediments, 10 to 115 ft (3 to 35 m), consist of approximately equal amounts of clastics (mostly silt) and carbonates (90% dolomite). The Queen formation, about 45 ft (14 m), also consists of clastics (sand and silt) and carbonates (mostly

dolomite). The predominant porosity of the Grayburg and Queen sediments is of micro-intercrystalline type; some vuggy and fenestral pores are also present. The Seven Rivers predominantly evaporite facies (mostly anhydrite) were deposited in hypersaline environments, created by sea water evaporation [Tinker & Mruk 1995; Tinker et al. 1995]. The Seven Rivers anhydrite, 300 to 950 ft (91 to 290 m) has extremely low porosity and forms the seal of the reservoir. Some thin clastic facies, present at the base of the Seven Rivers formation on the east side of the field, were deposited in a subaerial environment of a low-relief beach and dune complex over the buried San Andres topographic high. The clastic portion of the Seven Rivers, up to 100 ft (30 m) arc of sandstone, siltstone, and dolomite cement, has mostly micro-crystalline and intercrystalline porosity.

Karstification of the San Andres formation

Paleokarst in the San Andres formation is recognized by lithologic features in cores, extremely high flow rates from early wells, bit drops and other anomalies. Karsting by solution of porous limestone created numerous caves in the San Andres formation prior to its dolomitization. 1050 caves, ranging in height from 1 to 21 ft (0.3 to 6 m), or a total of 4477 vertical cave feet (1365 m), are recognized in wells throughout the Yates field [Tinker et al. 1995; Tinker & Mruk 1995]. Figure 4.2.6c shows the position of caves (yellow) in the San Andres formation, superimposed on the sequence-stratigraphic framework from Figure 4.2.6a. The greatest concentration of caves occurs on the highest parts of the carbonate shoals in the three major depositional cycles (the highest parts of the shoal clinoforms shift to the east from Cycle 1 to 3).

The spatial distribution of caves in the San Andres is best explained by the island hydrologic model [Craig 1988]. According to this model, karst is formed by the action of dynamic freshwater lenses beneath low-relief limestone islands. Such meteoric processes occurred several times during the Middle Permian when the carbonate shoals were subaerially exposed as limestone islands during relative sea level fall [Tinker et al. 1995]. The caves in the present-day San Andres formation progressively decrease in number and size under the paleo sea levels. Since the Yates reservoir is relatively shallow (1500-2000 ft; 457-610 m), many caves in the grain-supported reservoir rocks have not collapsed mechanically under the overburden and remain open in the reservoir. On the other hand, due to the rapid increase in effective loading due to dewatering in the Middle Permian, mud-supported sediments probably experienced failure karsting and subsequent compaction under less than 200 ft (61 m) of overburden (Wadleigh, pers. comm.).

Lineaments of caves and associated high well production from the reservoir have been related to intensified karstification along regional joints in the San Andres formation during the Permian [Craig 1988]. Solution and cave formation must have been especially intense along open joints on the crests of the carbonate shoals, exposed as limestone islands. According to Craig, paleo topographic

depressions (sinkholes) developed on the San Andres unconformity due to maximum solution at the joint intersections. A second derivative map of the present thickness of strata between the San Andres top and the Seven Rivers M horizon, shown in **Figure 4.2.7**, suggests that the alignment of regional joints was in two orthogonal directions: to the northwest and to the northeast.

Figure 4.2.8 shows the present thickness of sediments between the San Andres top and the M horizon in the case study area: Tract 17 (Figure 4.2.8a) and Tract 49 (Figure 4.2.8b). Thickness is plotted as the depth of the San Andres top under the M horizon. High negative values indicate thick overlying sediments, i.e. topographic lows (sinkholes or compactional sags in the carbonate blocks) that existed on the erosional surface of the San Andres top before deposition of Grayburg sediments. Coupled closed highs and lows possibly indicate early differential settlement of carbonate blocks. Since the M silty layer was horizontal during deposition, Figure 4.2.8 essentially illustrates the shape of the erosional surface (with different elevations). The karst surface has a maximum variation in elevations of about 80 ft (24 m), the steepest slopes being towards the sinkholes.

Dolomitization of the San Andres in the Late Permian

Most of the Middle Permian limestone ($CaCO_3$) of the San Andres and Grayburg formations was subjected to dolomitization (substitution of Ca by Mg in $CaCO_3$) during the Late Permian. According to the seepage-reflux model, in an evaporitic environment, hipersaline Mg-rich brines seep downward from saline lakes (salina) and salt flats (sabkha), displacing the less dense water from the pores of permeable back-reef limestones, and converting them into dolomite [Adams & Rhodes 1960; Tucker & Wright 1990]. This model is commonly accepted today to explain the dolomitization of intertidal and subtidal facies, underlying sabkha and salina evaporites. During the deposition of the Seven Rivers evaporites in the Yates field, as well as throughout the Central Basin Platform, magnesium-rich brines descended through the limestone of the San Andres and Grayburg and converted it into dolomite [Craig 1990].

Core description shows that the original fabric of the San Andres limestone was preserved during dolomitization [Tinker & Mruk 1995]. This indicates that dolomitization must have occurred at abundant nucleation sites, since this is an essential factor for retention of the crystal forms of the limestone [Tucker & Wright 1990]. Abundance of dolomite nucleation sites is common in fine limestone, such as the mudstone on the west side of the Yates field. Dolomitization of the coarser-grained limestone on the east side probably occurred at fewer nucleation sites. Some of the large pores of the east side grainstone may have been produced by leaching of limestone which was not dolomitized. The carbonate microcrystals "shrink", when calcite (the mineral that forms limestone) is converted into dolomite via replacement of Ca by Mg. This process can explain why dolomite is more brittle than limestone. Dolomitization

creates intercrystalline volumetric strains which causes the rock to fracture more readily under imposed stresses.

Anticlinal structure of the Permian reservoir

The present-day structure in the Yates field Permian rocks (shown in Figure 4.1.2) is a broad anticlinal asymmetrical dome with a northwest oriented anticlinal axis [Craig 1990; Tinker & Mruk 1995]. **Figure 4.2.9** illustrates the field structure together with a field-scale system of faults and drape folds which possibly exist in the Yates field. The fault/drape fold system in Figure 4.2.9 was derived by MOC geologists, based on interpretation of the thickness of sediments overlying the San Andres (Figure 4.2.7), and dip variation on flexure maps of the Permian strata [Wadleigh, pers. comm.].

The apex of the anticlinal structure on the east side of the field (in the Tract 49 area) is associated with the greatest accumulation of relatively incompactible, grain-supported, sediments. The structure is gently dipping toward the region of fine-grained, compacted mudstones on the west side of the field (Tract 17 area). **Figure 4.2.10** shows structure maps of the field anticline on the top of the San Andres formation in Tract 17 and Tract 49. The variation in elevations in Tract 17 is up to 500 ft (152 m), the steeper slopes being to the northeast towards the paleo basin. The variation in elevations in Tract 49 is up to 250 feet (76 m).

In the absence of significant regional tectonism during the Permian and later periods, it is accepted that the anticlinal structure of the Yates reservoir, and the structures of other Permian reservoirs on the Central Basin Platform, were created by drape folding. Two geologic mechanisms can possibly explain the drape folding [Craig 1990; Tinker & Mruk 1995; Wadleigh, Tinker, pers. comm.]. One is the differential compaction between the highly compactible mudstones and the relatively rigid shoal grainstones. Craig (1990) suggests that "differential compaction during and following deposition of shelf margin skeletal grainstone bars and associated lagoonal and intertidal mudstones was the dominant structure-forming process along the San Andres - Grayburg oil reservoirs". Another possible mechanism for drape folding of Permian strata is dip-slip movement along faults in Pennsylvanian rocks about 5000 ft (1524 m) below the reservoir.

The most significant structure-forming events in the Yates field can be dated to the end of the Late Permian. Field evidence from oil fields throughout the Permian Basin indicates that the Late Permian Salado salt is upturned and truncated beneath the Rustler formation [Craig 1963]. In the Yates field, there is a remarkable similarity between the structure on the top of the San Andres and in the overlying Permian strata, which indicates that they were folded together. This similarity is illustrated in **Figure 4.2.11** and **Figure 4.2.12** which show three-dimensional views of the present surfaces of (a) the Seven Rivers M horizon, and (b) the top of the San Andres formation in Tract 17 and Tract 49, respectively.

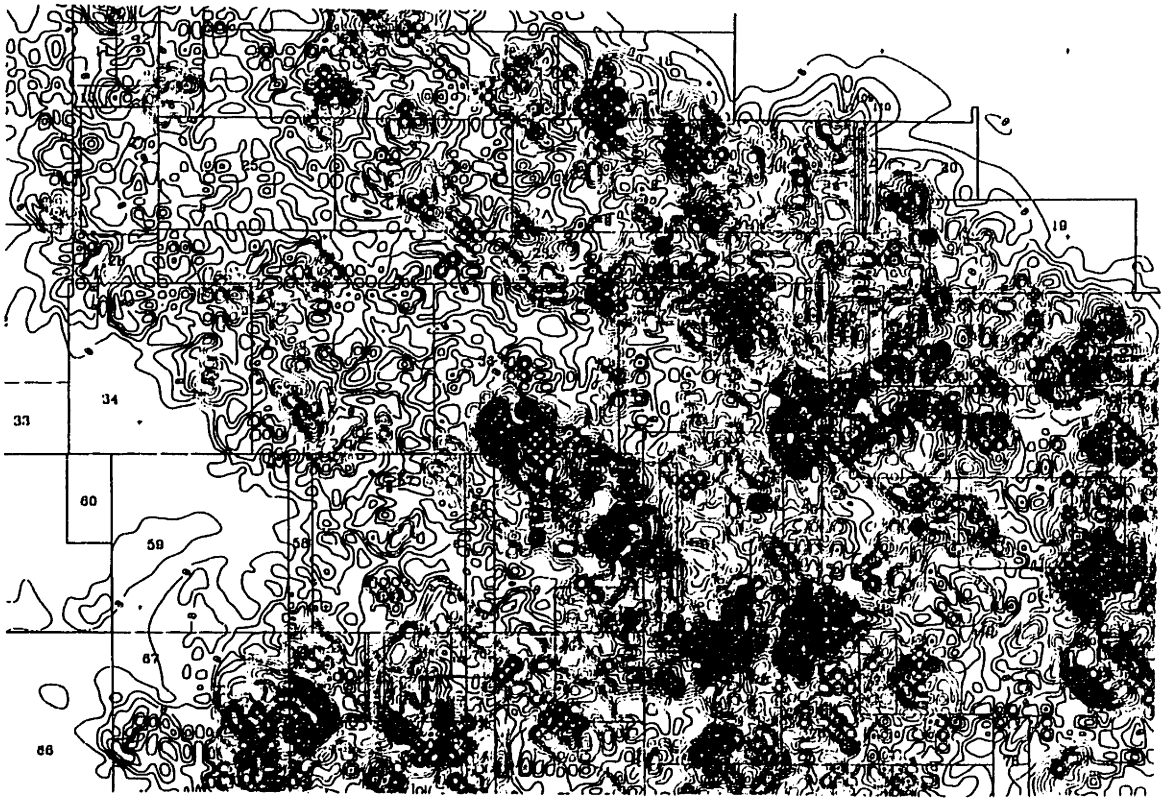
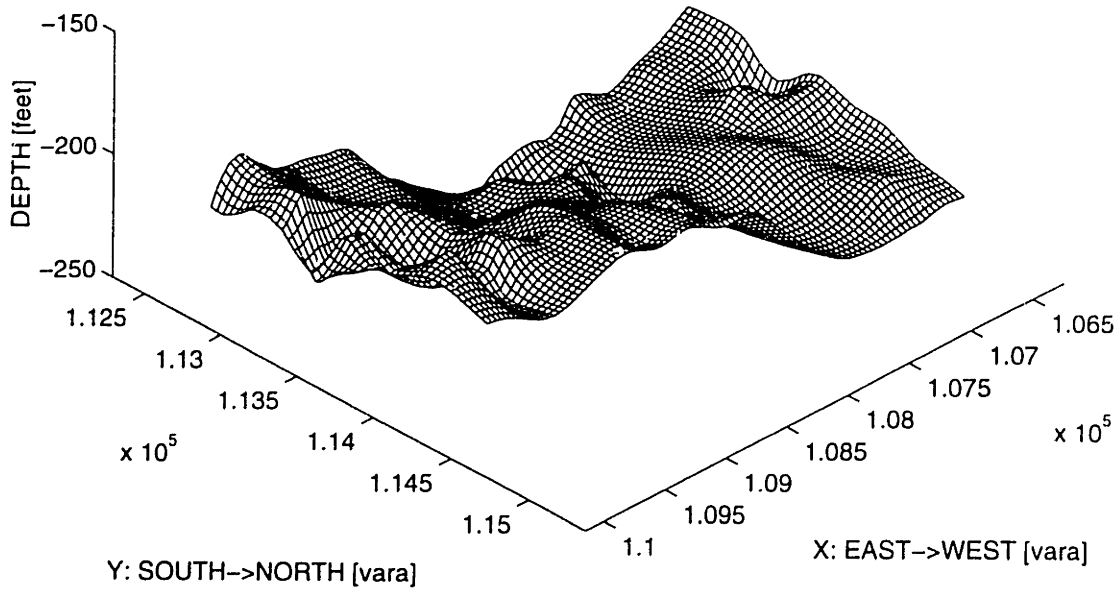


FIGURE 4.2.7 Present day thickness of sediments between the top of the San Andres formation and the Seven Rivers M datum in the Yates field [map provided by E. E. Wadleigh]. Alignment of thick sediments suggests intense karsting along regional joints, aligned approximately NE/SW and NW/SE, during the Permian.

a) TRACT 17



b) TRACT 49

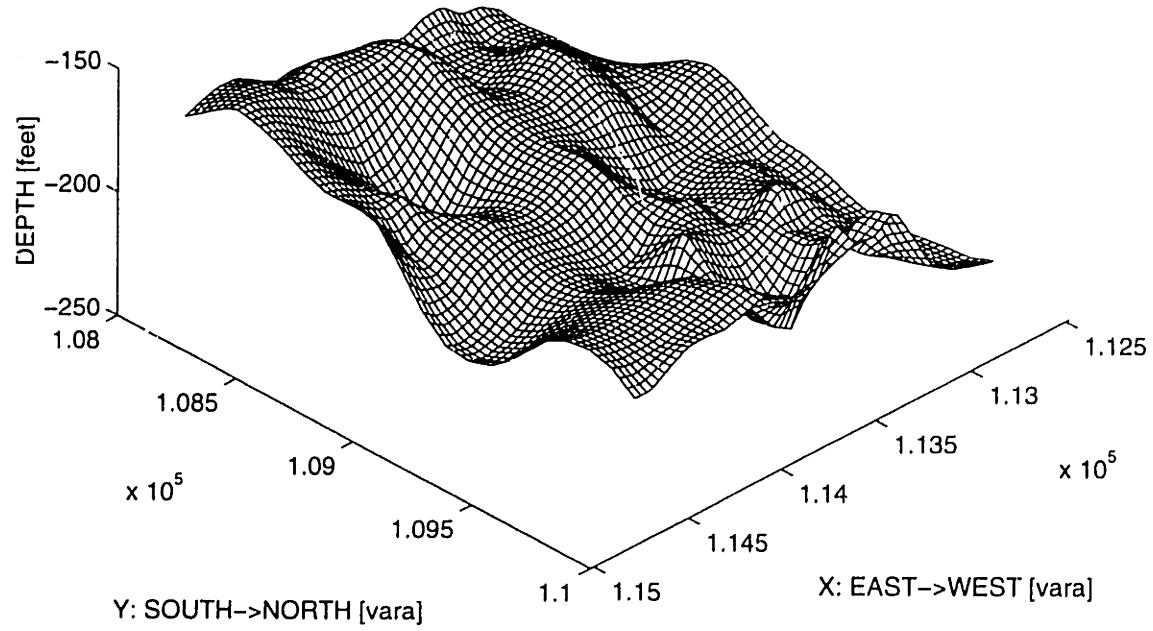


FIGURE 4.2.8 Present depth of the San Andres top from the Seven Rivers M datum in (a) Tract 17, and (b) Tract 49. Larger depths indicate thick sediments between the two levels (i.e. sinkholes on the paleokarst surface of the San Andres unconformity).

The present day structure of the internal shoals in the San Andres formation (mapped on the west side of the Yates field as tops of carbonate layers and bases of shale beds) is also very similar to that of the San Andres top and the Seven Rivers M datum. The field evidence and the present day geometric configuration of the reservoir strata suggest that the drape folding of the Yates field reservoir rocks occurred at the end of the Late Permian, after deposition of Salado salt and before deposition of Rustler carbonates and clastics.

4.3 AVAILABLE QUANTITATIVE DATA

This section summarizes the interpretation of field data from the case study area (Tracts 17 and 49), incorporated in the numerical modeling of the fracture system (discussed in Section 4.5). Chemical composition and physical properties of the reservoir rocks have been studied in detail prior to the case study. However, little field information has been collected on mechanical properties and brittle fracturing of the reservoir rocks. Section 4.3.1 summarizes the quantitative data on reservoir stratigraphy, lithology and structure in Tracts 17 and 49. Section 4.3.2 presents all available field data on fracturing of the Yates reservoir rocks.

4.3.1 Stratigraphy, lithology and structure

In the 3D fracture system model (Chapter 3), fracture orientations can be related to geologic structures, whereas fracture intensity is usually a function of the spatial distribution of material properties of the host rock. All data relevant to Tracts 17 and 49, extracted from the 3D geologic database of the Yates field reservoir, were incorporated in two mini-Stratamodels by MOC geologists and engineers. **Table 4.3.1** summarizes the parameters of the two Stratamodels. The following data from the Stratamodels of Tract 17 and Tract 49 are used in the numerical modeling of the fracture system:

- Structure maps of the contact surfaces between major and minor formation sequences, the two most important being the San Andres top surface and the surface of the Seven Rivers M-horizon (plotted in Figure 4.2.11 for Tract 17 and in Figure 4.2.12 for Tract 49). Elevations of the formation contacts are available from all wells in Tracts 17 and 49 (well locations are shown in Figure 4.2.10).
- Complete spatial distribution (i.e. readings in the geocells of the two Stratamodels) of four attributes: gamma ray response (corrected for uranium in Tract 49), porosity, calcite content, and cave locations. In the geologic Stratamodel of the reservoir, the spatial distribution of the four attributes is extrapolated from data available from all 1800 logged wells and 118 cores in the Yates field. The spatial arrangement of Stratamodel geocells is shown in **Figure 4.3.1**a. Figure 4.3.1b shows the distribution of porosity on a cross section from the 3D Stratamodel of Tract 17.

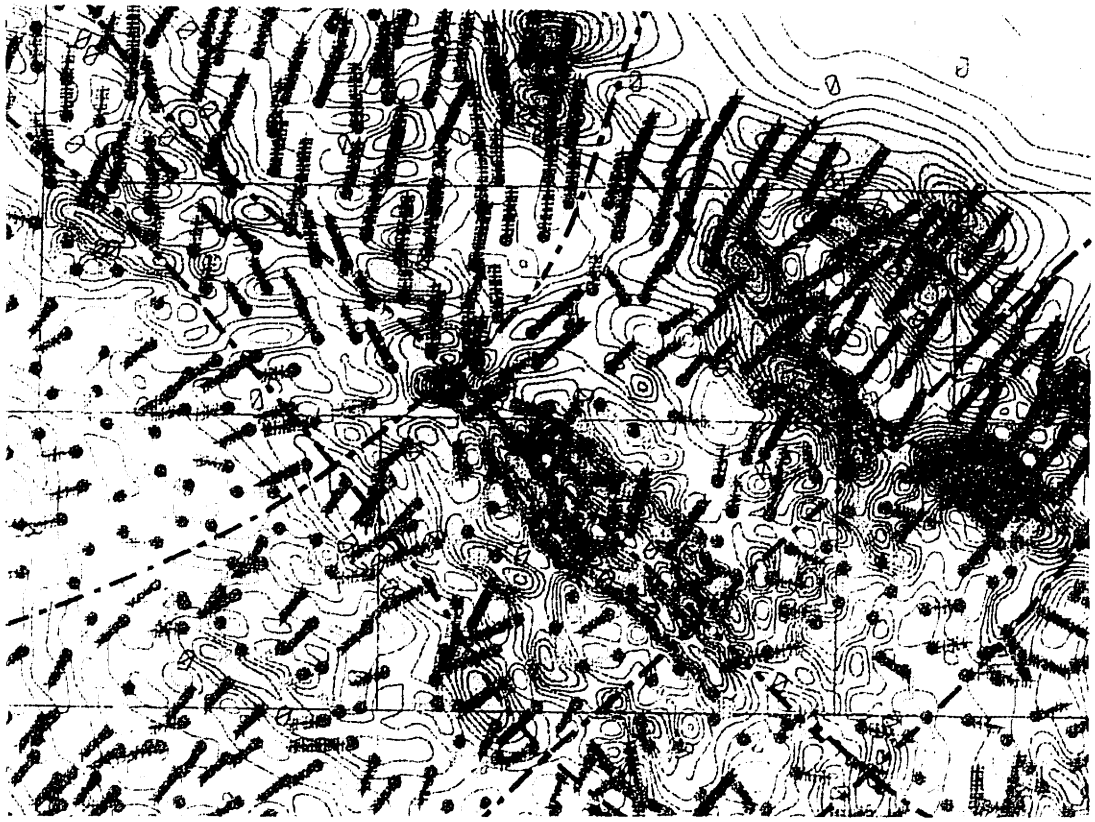
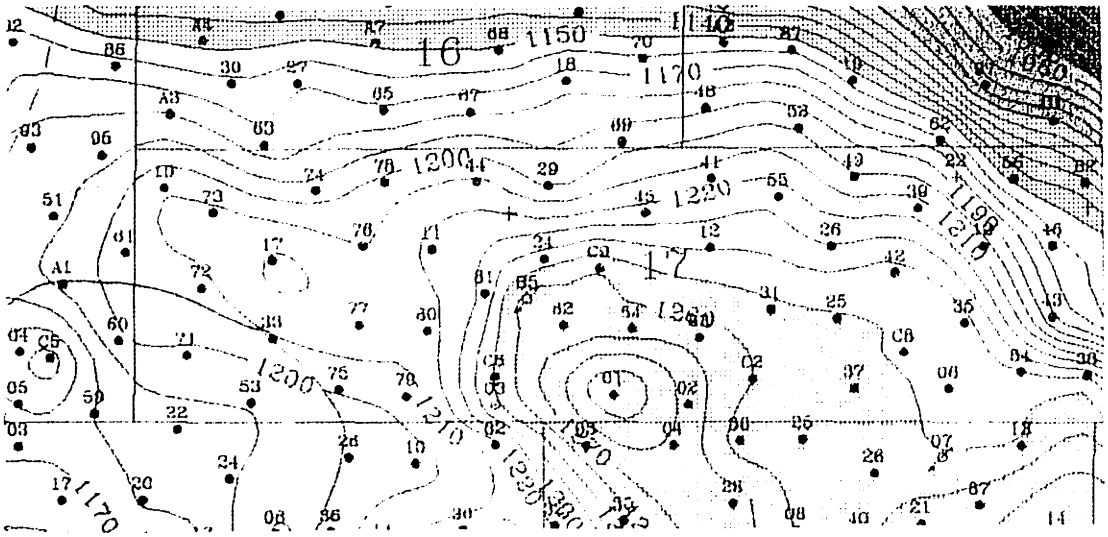
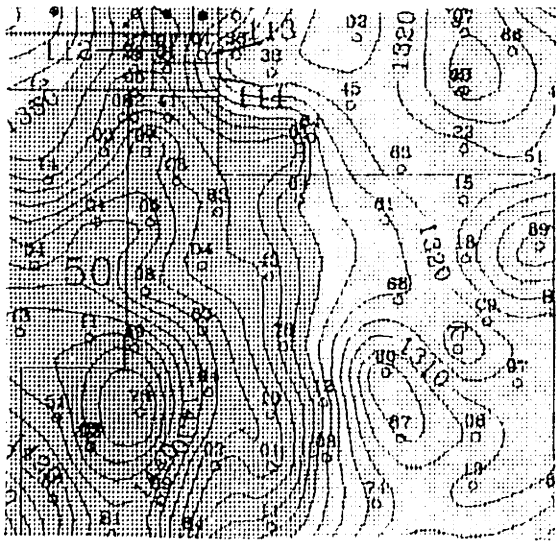


FIGURE 4.2.9 Anticlinal structure on the top of the San Andres formation with locations of faults and drape folds that possibly exist in the Yates field reservoir [map provided by E. E. Wadleigh]. Red dash-and-dot lines: faults / drape fold axes. Short solid lines: direction of maximum flexure; color indicates flexure magnitude (red: highest, blue: intermediate, green: lowest).

a) TRACT 17



b) TRACT 49



LEGEND

Structure Elevation
(Feet Above Sea Level)

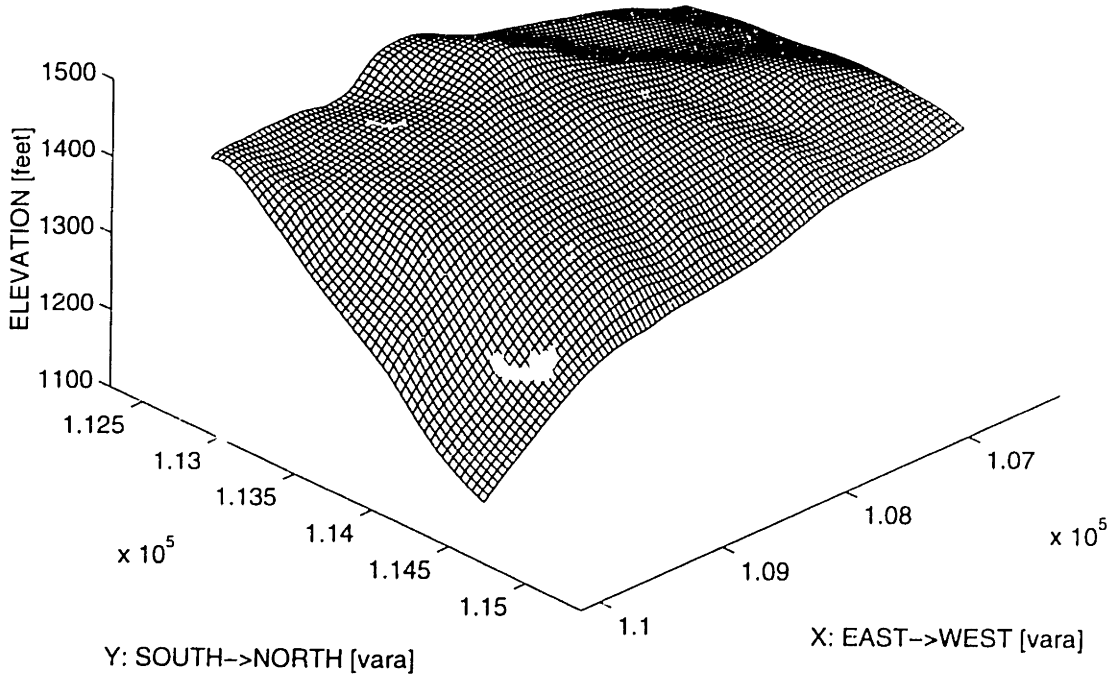
>1600	1200–1150
1600–1550	1150–1100
1550–1500	1100–1050
1500–1450	1050–1000
1450–1400	1000– 950
1400–1350	950– 900
1350–1300	900– 850
1300–1250	850– 800
1250–1200	<800

Well Symbols

- Producing oil well
- Water or gas injection well
- Observation well
- Plugged & abandoned
- Water supply well

FIGURE 4.2.10 Structure maps of the San Andres formation: a) in Tract 17, and b) in Tract 49.

a) SEVEN RIVERS M



b) SAN ANDRES TOP

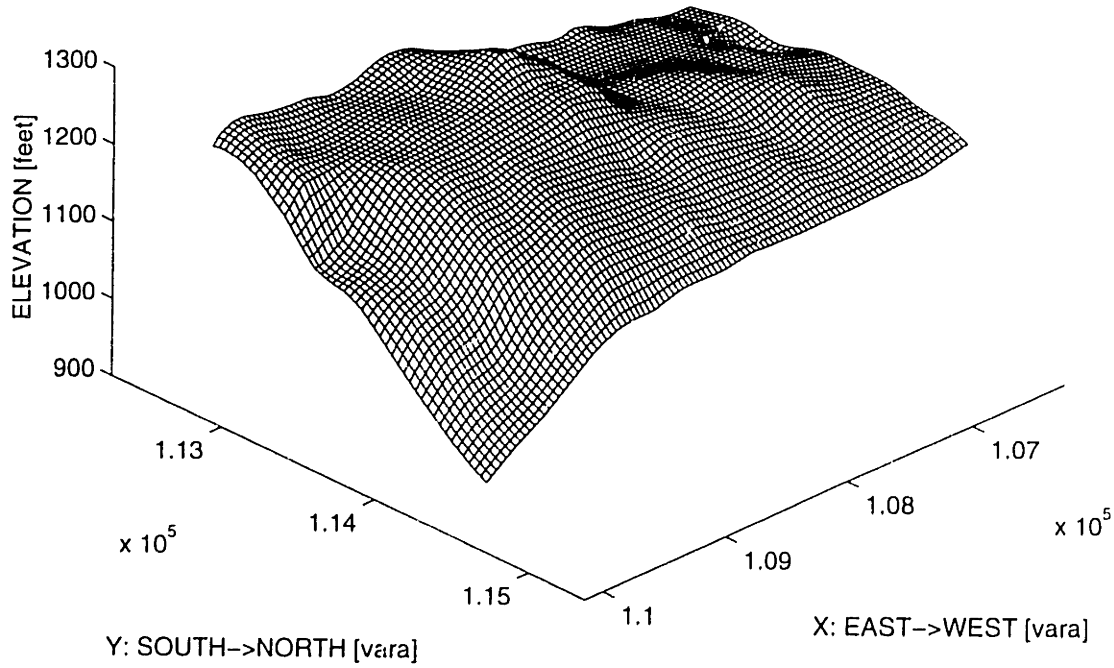
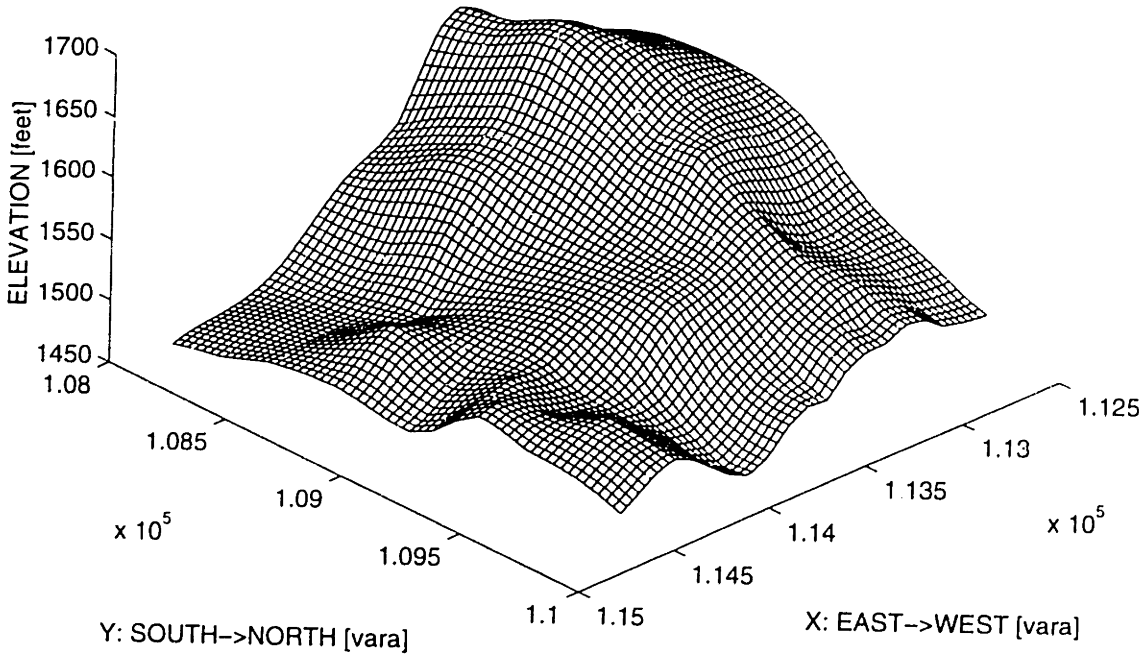


FIGURE 4.2.11 Reservoir anticlinal structure in Tract 17: a) Seven Rivers M-horizon; b) San Andres top.

a) SEVEN RIVERS M



b) SAN ANDRES TOP

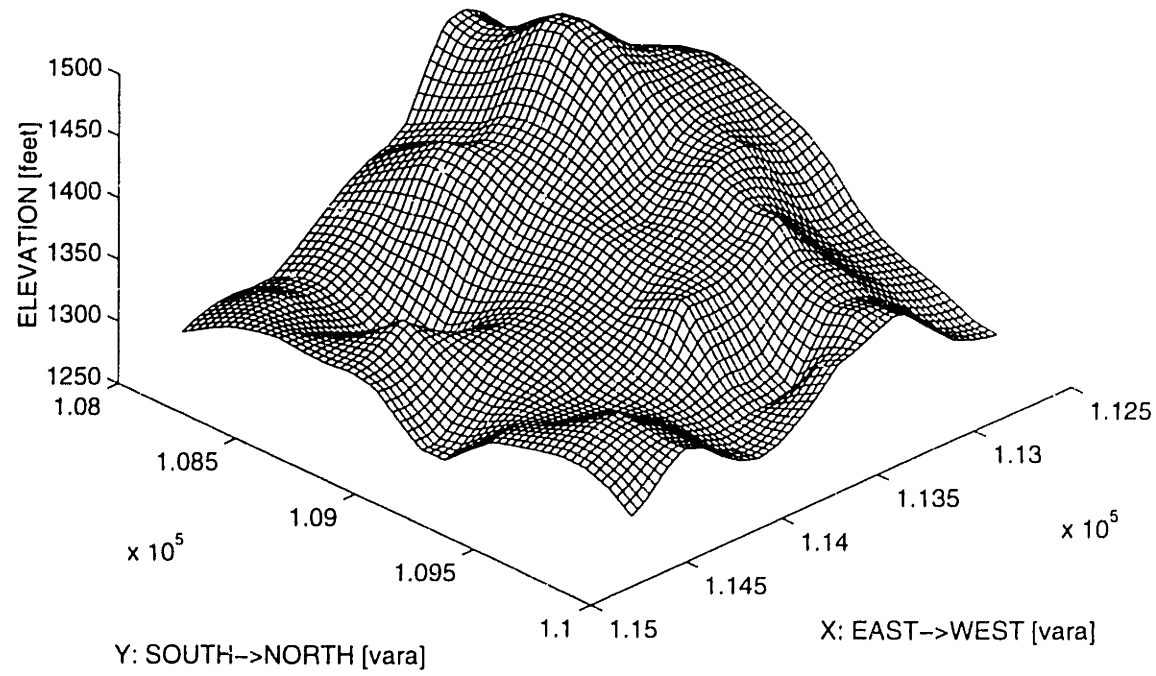


FIGURE 4.2.12 Reservoir anticlinal structure in Tract 49: a) Seven Rivers M-horizon; b) San Andres top.

PARAMETER	TRACT 17	TRACT 49
West boundary, X_{min} [vara]	106,250	112,877
East boundary, X_{max} [vara]	110,150	114,777
South boundary, Y_{min} [vara]	112,931	108,091
North boundary, Y_{max} [vara]	114,725	109,991
West-east data column size, **dX** [vara]	39	31.667
South-north data column size, **dY** [vara]	39	31.667
West-to-east number of data columns	100	60
South-to-north number of data columns	46	60
Total number of data columns	4,600	3,600
Minimum elevation, Z_{min} [feet]	464.034	742.826
Maximum elevation, Z_{max} [feet]	1515.45	1660.69
Number of formation sequences	22	22
Number of stratigraphic layers	243	216
Total number of geocells in the model	1.12 million	0.78 million
Number of attributes per cell	28	65

TABLE 4.3.1 Stratamodel parameters in Tract 17 and Tract 49. Data columns are vertical stacks of geocells, the number of geocells in each column being equal to the number of stratigraphic layers present at the (X,Y) location of the column.

Distribution of gamma ray, porosity, calcite content, and caves can give evidence for the variation of fracture intensity and permeability in the reservoir. Gamma ray response (corrected for high values due to uranium) is an indicator of shale content in the San Andres dolomite. According to Tinker & Mruk (1995), some fractures in the west side (Tract 17 area) are possibly associated with compaction of argillaceous dolomite and dolomitic shales in the San Andres. Typically, these fractures affect the more brittle dolomite between the shales, but terminate at the more ductile shales. On the west side of the field 40 API gamma ray (GR) corresponds to 10% argillaceous content. **Figure 4.3.2** shows the distribution of GR response in Tract 17 and Tract 49. In the figure, red color indicates locations where GR>20 API. The color scheme in Figure 4.3.2a indicates that in Tract 17 the San Andres formation consists of massive dolomite, interbedded with shale layers. In Figure 4.3.2b the red color is almost completely absent, indicating massive clean dolomite in Tract 49.

Porosity distribution can be a good indicator of fracture intensity in the Yates field reservoir, since prior interpretation of field data has suggested that lower-porosity rocks are more brittle and higher-porosity rocks are more ductile [Tinker & Mruk 1995; Wadleigh, pers. comm.]. **Figure 4.3.3** illustrates the distributions of fracture intensity (fracture counts), superimposed on histograms of porosity distribution, for four major lithology classes in the Yates reservoir. In the San Andres "shales" on the west side, fracture counts are generally low. The greatest fracture counts are observed in the low-porosity dolomite mudstones and wackestones typical for the upper west side of the Yates field (Tract 17 area). Fracture counts increase up to 8-10% porosity, and then decrease with increased porosity. Yates field geologists have concluded that the thin-bedded carbonates on the west side have undergone intense fracture failure with numerous short, low conductivity, fractures. As illustrated in Figure 4.3.3, fewer fracture counts are observed in the more ductile, high-porosity, dolomite packstones and grainstones in the Yates field east side (Tract 49 area). These massive carbonate shoal build-ups are mechanically stronger which has resulted in a smaller number of fractures with larger apertures. Relatively few fracture counts have been observed in the ductile sandy dolomite of the Grayburg (marked as "sand" in Figure 4.3.3).

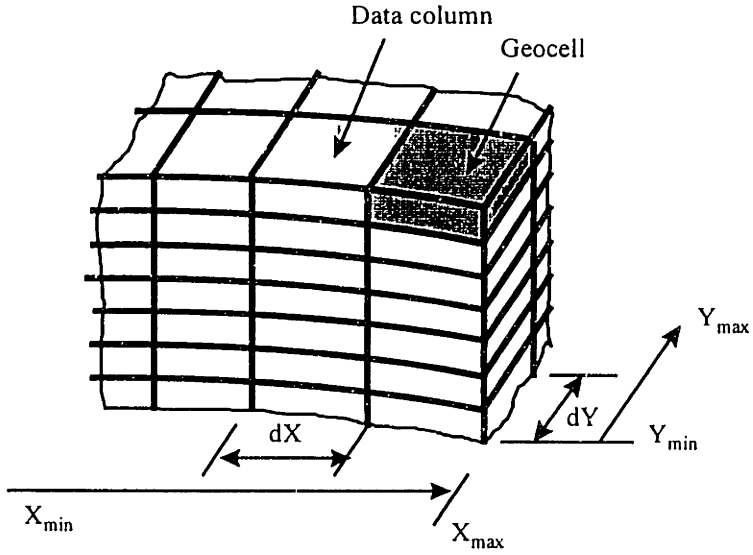
The distribution of caves (discussed in Section 4.2.2 and illustrated in Figure 4.2.6c) can give indication about the locations of early fractures which served as water conduits for karstification during and after deposition of the San Andres formation. The presence of caves increases, whereas precipitated calcite decreases, the permeability along fractures in the reservoir rocks. Although permeability is not a subject of the current case study, detailed calcite and cave data have been collected in the context of future verification of the fracture system model though simulations of flow in the reservoir. **Figure 4.3.4** shows the distribution of secondary calcite in the San Andres formation in Tract 49.

4.3.2 Fracture data from field testing

The immense network of interconnected pores in the San Andres formation, including fractures, is a major factor for the high productivity of the Yates field reservoir [Craig 1990; Tinker & Mruk 1995]. Although comprising only a small portion of the Yates field storage capacity (about four percent of the total porosity), the fracture system in the Permian reservoir rocks may provide significant pathways for anisotropic permeability and secondary recovery of oil.

Fractures in the Yates field reservoir rocks have been directly observed in cores, and also identified on FMI/FMS (Schlumberger's Formation Micro Image and Formation Micro Scanner) profiles of logged wells. **Figure 4.3.5** shows the locations of wells in or near Tract 17 and Tract 49 where core or log fracture identifications are available. The well locations are superimposed on structure maps of the San Andres top in Tract 17 (Figure 4.3.5a) and Tract 49 (4.3.5b), respectively.

a)



b)

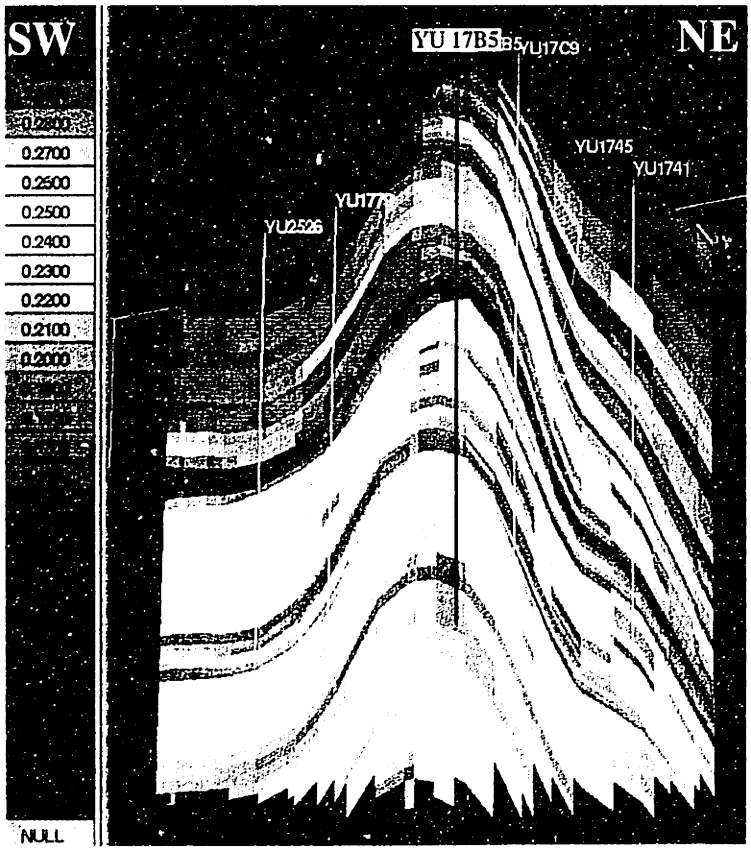
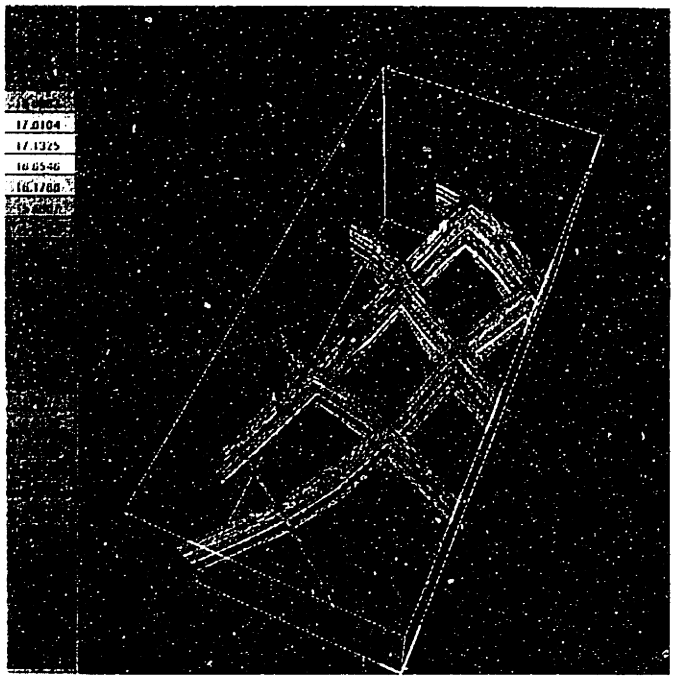


FIGURE 4.3.1 The Stratigraphic Geocellular Model (Stratamodel) of the Yates field reservoir in Tracts 17 and 49: a) schematic representation of the spatial arrangement of data cells (geocells); b) porosity distribution in geocells, shown on a cross section of the Stratamodel of Tract 17.

a) TRACT 17 GAMMA RAY



b) TRACT 49 GAMMA RAY

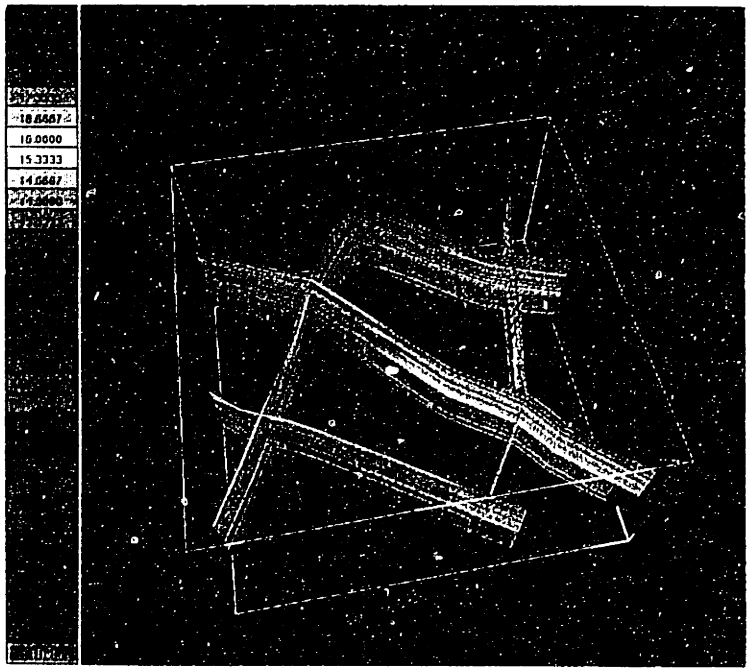


FIGURE 4.3.2 Gamma ray response in the San Andres formation: a) Tract 17; b) Tract 49. Red color indicates GR>20API, corresponding to 5% shale content.

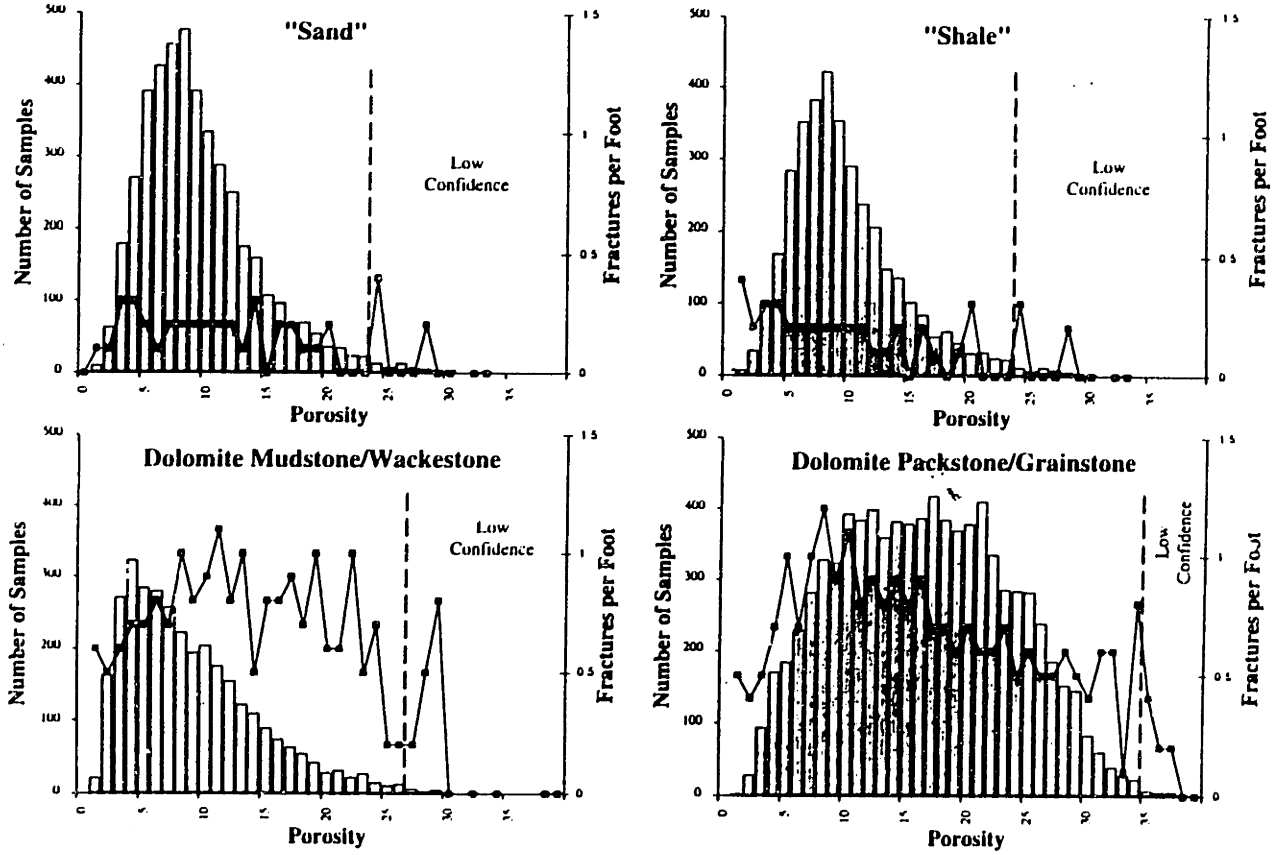


FIGURE 4.3.3 Fracture intensity (solid line), superimposed on porosity histograms in four Permian formations in the Yates field. High porosity values where field specimens are too few for a representative statistical sample are marked as "low confidence" [from Tinker & Mruk 1995].

Core photographs in **Figure 4.3.6** illustrate some typical textures and fractures of the Yates field reservoir rocks. Figure 4.3.6a and b show the core at two different elevations in the same well, located in the middle of the Yates field. Figure 4.3.6a illustrates typical dense mudstone (core porosity 8.3%), characteristic of the dolomitic shale marker beds in the upper San Andres. Figure 4.3.6b shows dolomitic grainstone of relatively low matrix porosity (10.7%). The core is cut by an oil-stained, totally open, penetrating fracture (TOPN) that controls the core permeability. Figure 4.3.6c shows typical dolomite packstone/ grainstone of higher porosity (16.9%). The core is cut by a solution-enhanced, partially filled with calcite fracture (FILO), and by several dendritic, non-penetrating, open fractures (PONP). Figure 4.3.6d shows dolomite packstone with dense matrix and fusumoldic porosity (total porosity 16.8%). The core is cut by a vertical calcite-filled fracture (FILL), and by a horizontal, drilling-induced, fracture (INDU).

Table 4.3.2 summarizes fracture observations in continuous cores from Tract 17 and Tract 49. For every cored well the table lists well location, elevation at which core was taken, number of continuous cored feet, number of fractured cored feet, and predominant apparent dip of fractures at that location (fracture strikes are not available since the coring device is not oriented).

CORED WELL	CORE ELEVATION [FT]		CORED FEET			APPARENT DIP
	BOTTOM	TOP	TOTAL	FRAC-TURED	FRACT. / TOTAL	
Tract 17						
YU14D4	1069	1159	90	36	0.40	60°-85°
YU1776	1053	1240	187	82	0.44	60°-90°
YU17C9	1077	1130	53	18	0.34	80°-90°
YU2416	1011	1320	204	14 ?	0.07?	80°-85°
YU2433	967	1285	250	79	0.32	65°-90°
YU2437	1095	1145	48	18	0.38	60°-80°
YU2509	1000	1270	270	22 ?	0.08?	85°-90°
YU2530	1111	1190	15	6	0.40	80°-85°
Tract 49						
YU4001	1116	1199	72	28	0.39	60°-85°
YU4020	1127	1211	68	28	0.41	60°-85°
YU4033	1111	1157	34	11	0.32	60°-85°
YU4910	1106	1212	81	12	0.15	80°-85°
YU4936	944	1139	195	82	0.42	80°-90°
YU4942	1112	1206	94	19	0.20	60°-85°
YU4951	1290	1253	27	8	0.30	60°-80°
YU4957	1155	1218	40	9	0.22	60°-85°
YU49C3	997	1159	139	63	0.45	60°-85°
YU5005	1009	1171	129	45	0.35	60°-85°
YU5008	1111	1180	69	21	0.30	70°-85°
YU5123	1110	1211	85	26	0.32	70°-85°
YU5131	1050	1418	361	15 ?	0.04?	80°-85°
YU5152	1194	1389	140	14 ?	0.10?	70°-90°

TABLE 4.3.2 Core description in Tract 17 and Tract 49. Total cored feet equal the total core length core minus the missing core. Fractured cored foot is a foot of core with at least one natural fracture penetrating the core wall to wall. Question mark (?) indicates unusually low fracture count possibly due to poor core quality.

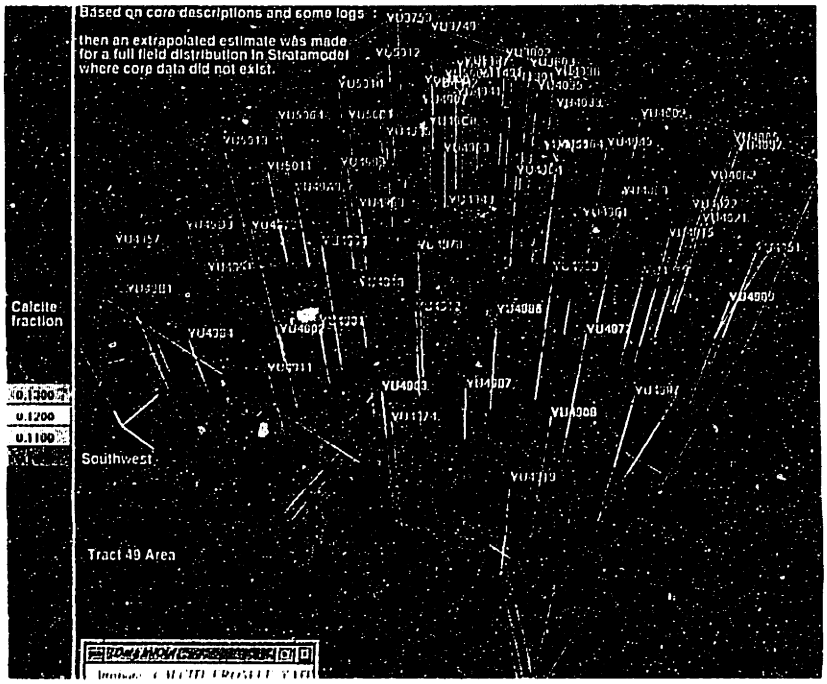
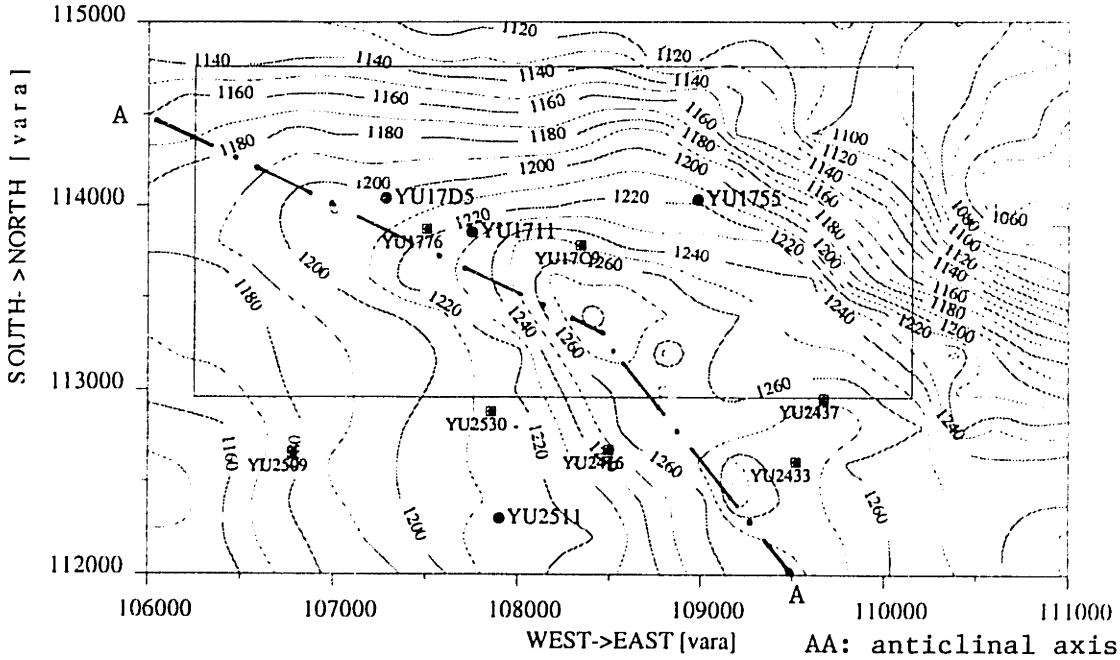


FIGURE 4.3.4 Distribution of secondary calcite in Tract 49 [figure by M. Uland].

a) TRACT 17



b) TRACT 49

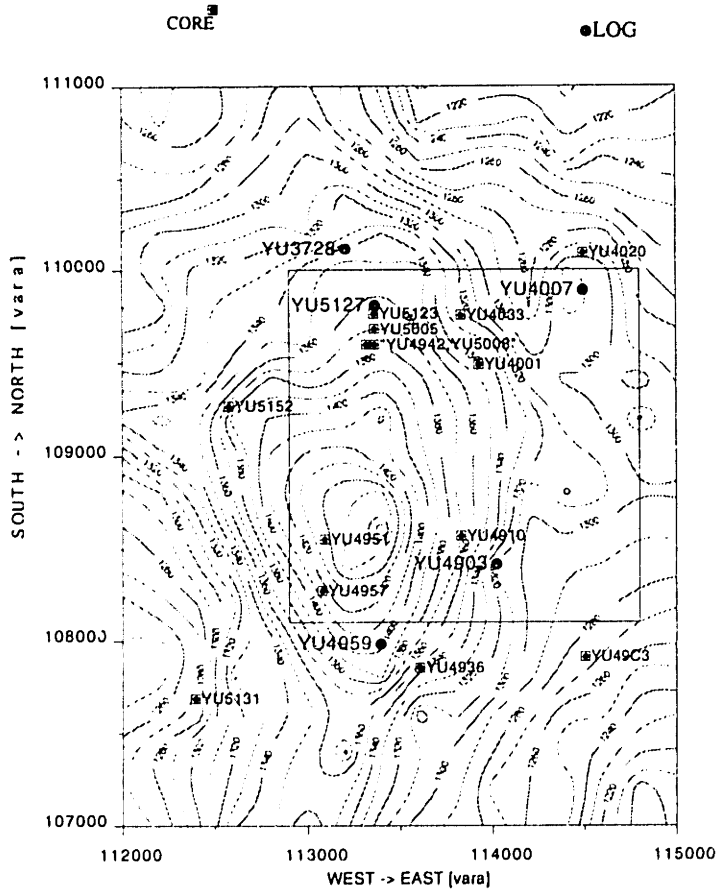


FIGURE 4.3.5 Locations of logged and cored wells, superimposed on structure maps of the San Andres top in a) Tract 17, and b) Tract 49 area.

In the count of fractured cored feet in Table 4.3.2, only fractures of the types TOPN, FILO, and FILL (Figure 4.3.6) are considered. Calcite-filled fractures, if large enough, possibly transform into partially open and even totally open over their vertical extent (since calcite precipitated at relatively thin horizontal layers). Interconnected totally open and partially filled fractures contribute to the conductivity of the fractured reservoir (which is of ultimate interest to the Fractured Reservoir Technologies project). Fractures of type PONP that do not completely penetrate the core are too small to affect the directional conductivity; their effect can be accounted for by an increase of the matrix porosity and permeability. Fractures of type INDU are created by drilling stresses and exist only in close vicinity of the cored well; they do not affect significantly the conductivity of the reservoir.

Table 4.3.3 summarizes the fractures identified on FMI/FMS profiles of logged wells in Tract 17 and Tract 49: a horizontal well (YU17D5) and eight vertical wells. For every logged well the table shows well location, elevation and length of the FMI/FMS profile, and number of identified fractures. **Figure 4.3.7** and **Figure 4.3.8** show distribution of bedding boundaries and fractures at depth at two logged wells in Tract 17 (YU1711 and YU1755). **Figures 4.3.9 - 4.3.11** show distribution of bedding boundaries and fractures at depth at three logged wells in Tract 49 (YU4007, YU4903, and YU5127). Fracture dip distribution at the logged wells in Tract 17 and Tract 49 are shown in **Figure 4.3.12** and **Figure 4.3.13**, respectively.

LOGGED WELL	ELEVATION [FEET]		LOGGED FEET	NUMBER OF FRACTURES		
	BOTTOM	**TOP**		**TOTAL**	**OPEN**	**HEALED**
Tract 17						
• YU1711	810.03	1027.63	217.60	124	75	49
• YU1755	817.10	1099.74	282.64	163	133	30
• YU17D5	1087	1087	1550	671	599	72
• YU2511*	813.59	1032.03	218.44	85	57	25
Tract 49						
• YU3728*	798	1089.47	291.47	192	110	82
• YU4007	821.46	1070.44	248.98	105	76	29
• YU4903	824.24	1019.99	195.75	83	67	16
• YU4959*	813.11	1114.48	301.37	86	56	30
• YU5127	810.91	1023.37	212.46	122	112	10

TABLE 4.3.3 Fractures identified on FMS/FMI profiles of logged wells in Tract 17 and Tract 49. The asterisk (*) sign indicates that the well is located outside of the study area, in the close vicinity of either Tract 17 or Tract 49.

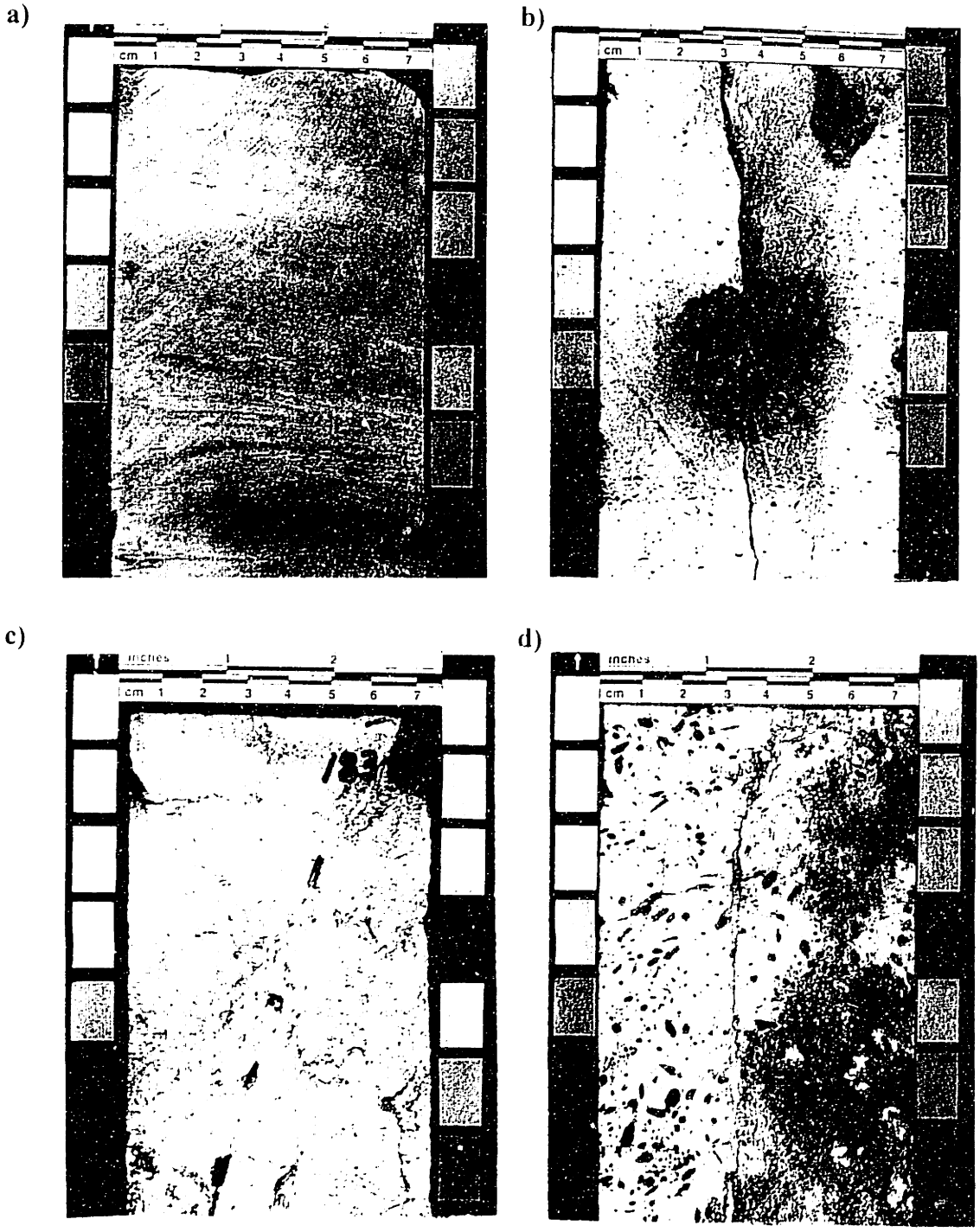


FIGURE 4.3.6 Texture and fracturing of the San Andres formation (core photographs): a) dense, dolomitic shale mudstone; b) dolomitic grainstone cut by a totally open penetrating (TOPN) oil-stained fracture; c) dolomitic packstone/ grainstone cut by a partially filled with calcite penetrating (FILO) fracture; d) dolomitic packstone with fusumoldic porosity, cut by a vertical calcite-filled (FILL) fracture (stained red), and a horizontal drilling-induced (INDU) fracture.

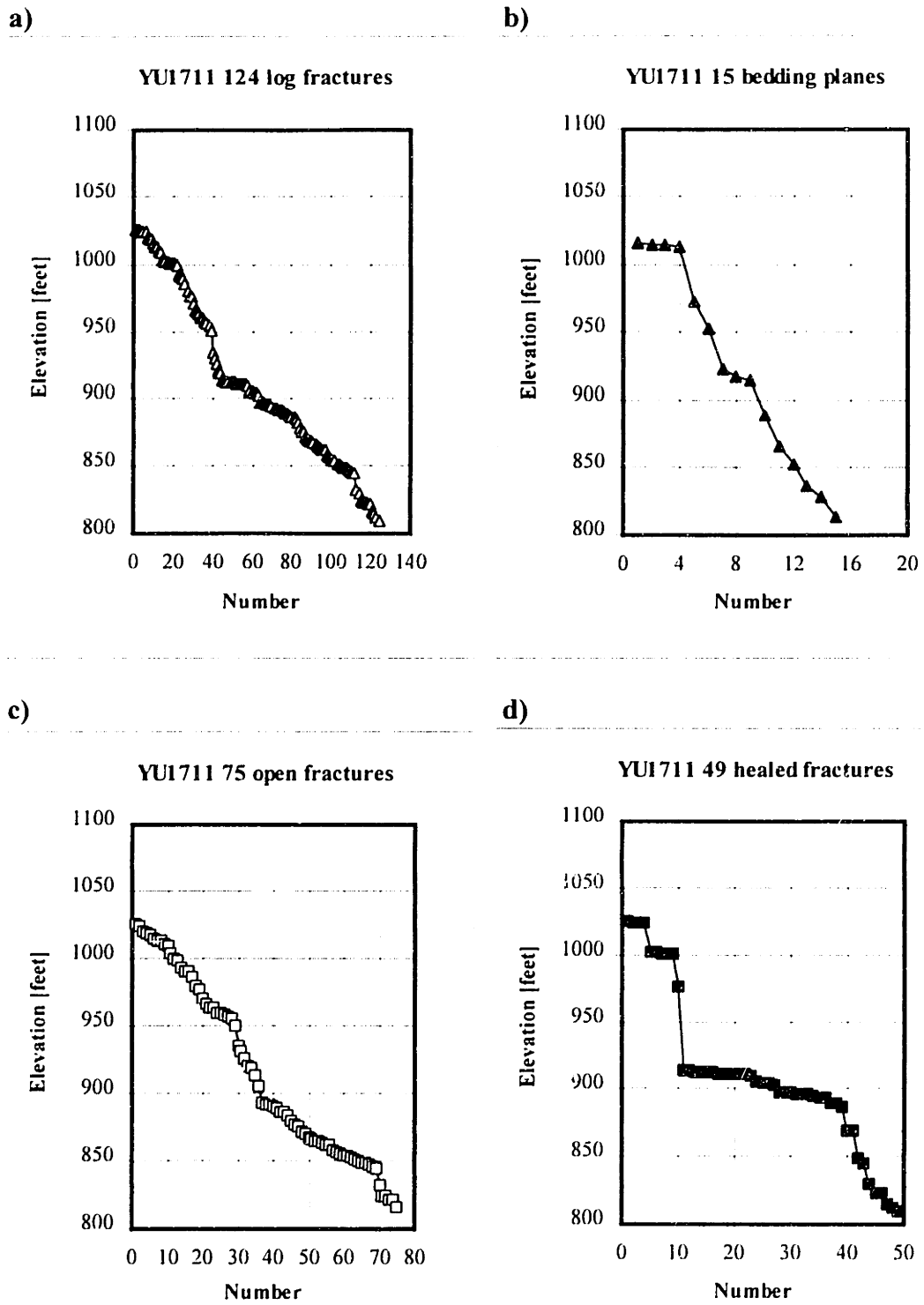


FIGURE 4.3.7 Distribution of fractures with depth at logged well YU1711: a) all fractures (including open and healed fractures); b) bedding boundaries; c) open fractures; d) healed fractures.

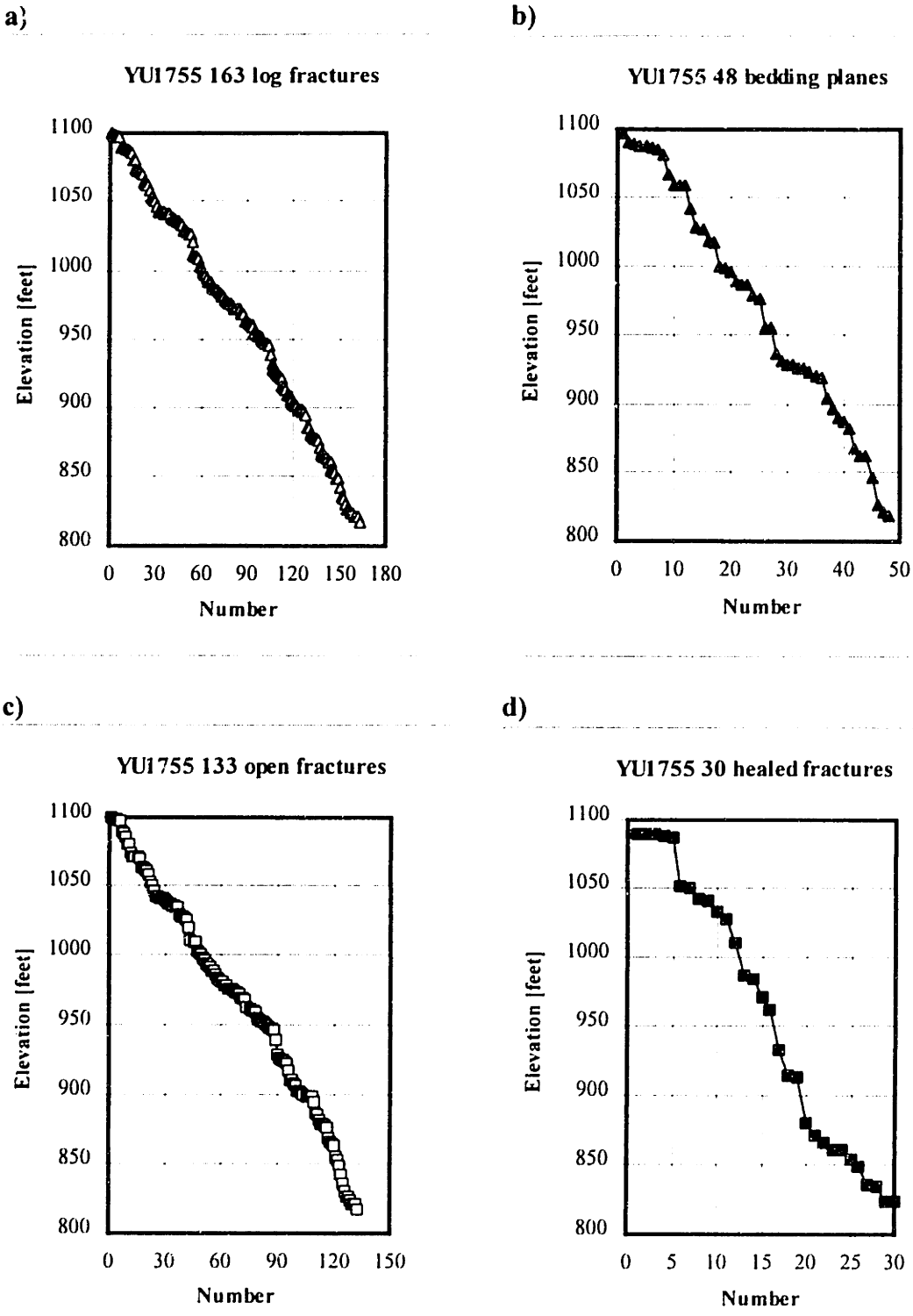


FIGURE 4.3.8 Distribution of fractures with depth at logged well YU1755: a) all fractures (including open and healed fractures); b) bedding boundaries; c) open fractures; d) healed fractures.

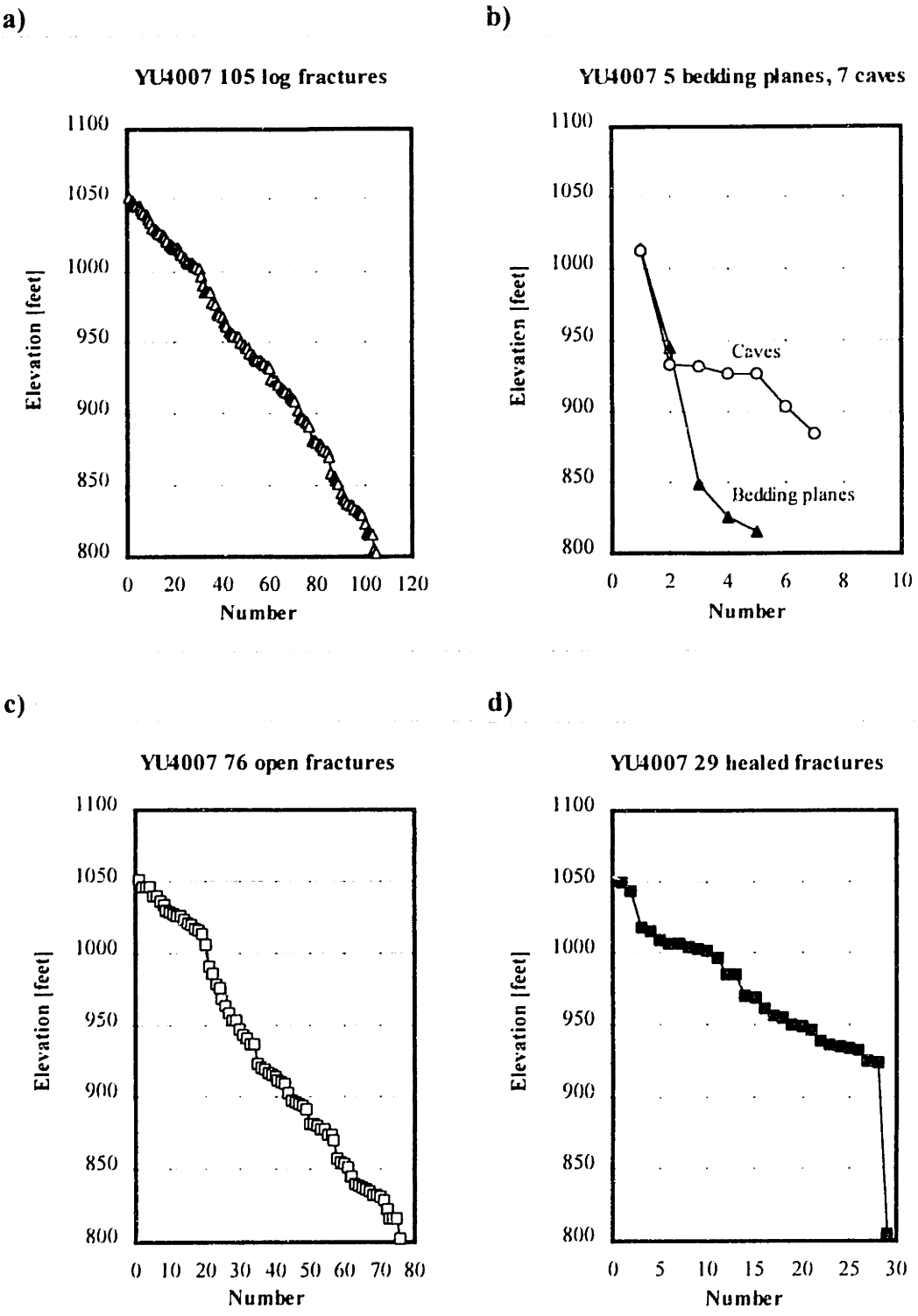


FIGURE 4.3.9 Distribution of fractures with depth at logged well YU4007: a) all fractures (including open and healed fractures); b) bedding boundaries and caves identified on the log profile; c) open fractures; d) healed fractures.

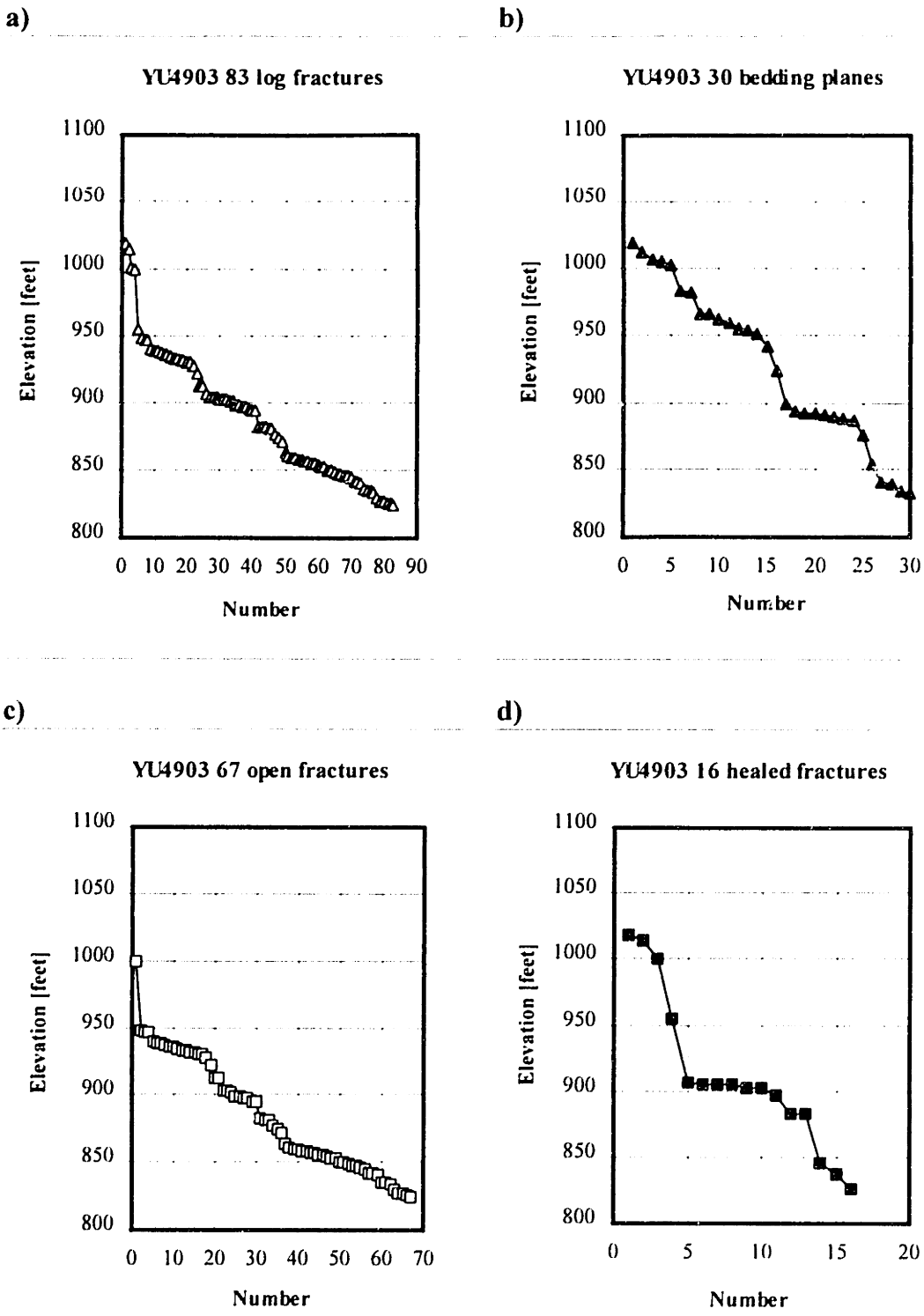


FIGURE 4.3.10 Distribution of fractures with depth at logged well YU4903: a) all fractures (including open and healed fractures); b) bedding boundaries; c) open fractures; d) healed fractures.

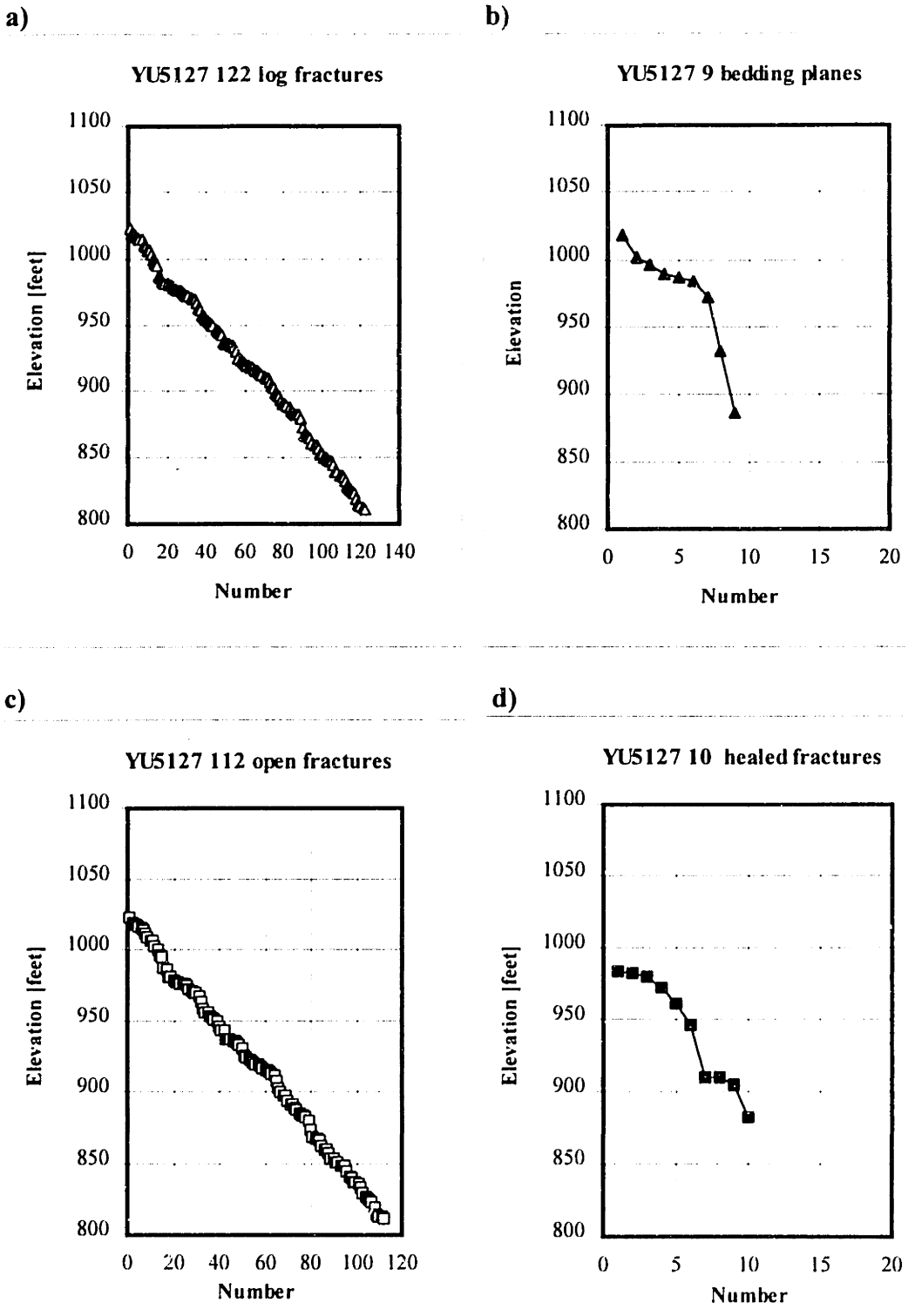
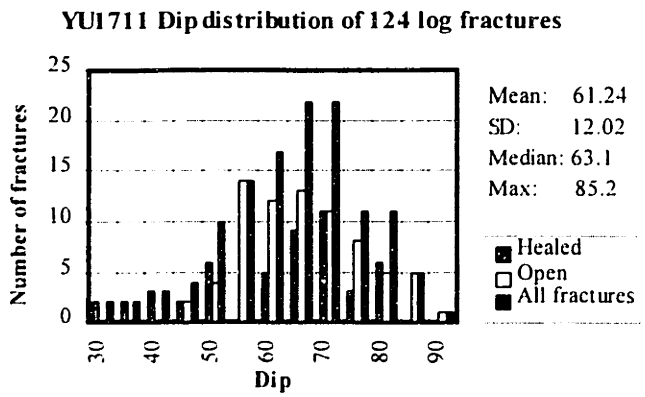
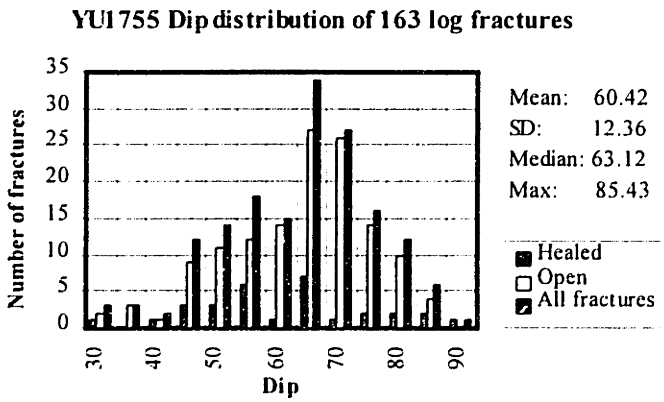


FIGURE 4.3.11 Distribution of fractures with depth at logged well YU5127: a) all fractures (including open and healed fractures); b) bedding boundaries; c) open fractures; d) healed fractures.

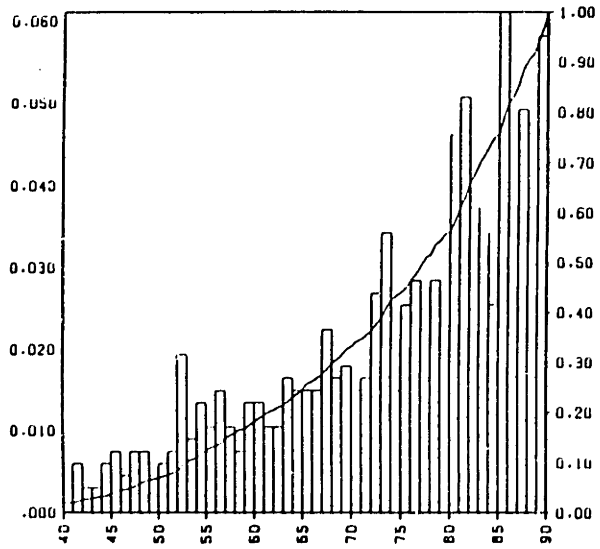
YU1711



YU1755



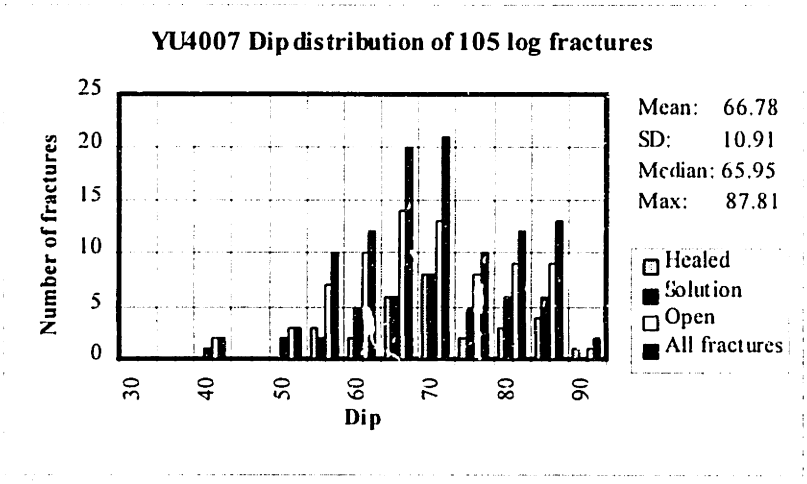
YU17D5



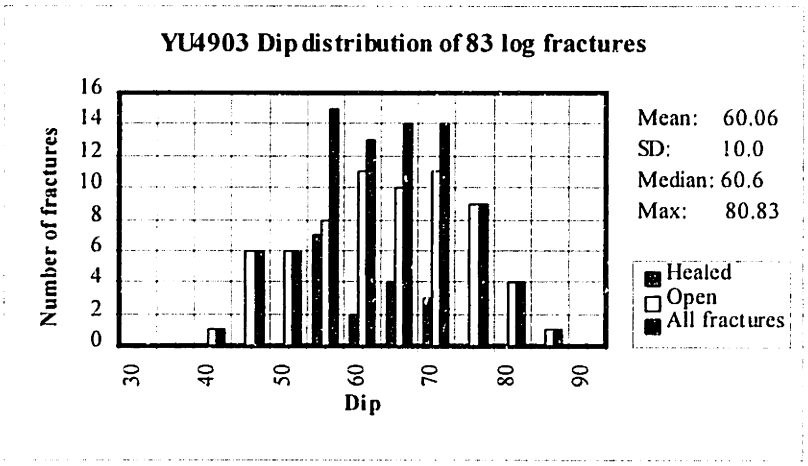
Number of fractures: 671
Mean fracture dip: 73.57
Standard deviation: 13.38
Median of dip: 77.18
Mode of dip: 85.5
Maximum dip: 90

FIGURE 4.3.12 Distribution of fracture dips at logged wells in Tract 17.

YU4007



YU4903



YU5127

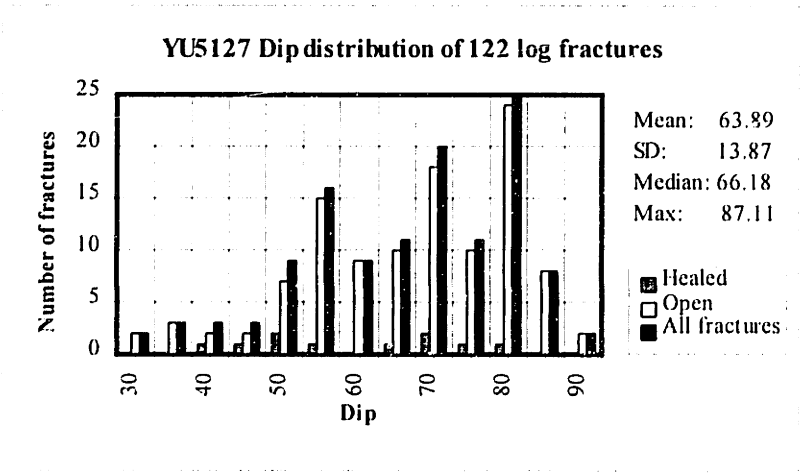


FIGURE 4.3.13 Distribution of fracture dips at logged wells in Tract 49.

Table 4.3.4 summarizes statistics of the spacing between the fracture well intersections, identified on FMS/FMI log profiles at eight locations: three wells in or near Tract 17, and five wells in or near Tract 49. The table includes calculated average and standard deviation of the spacing between (1) bedding planes, (2) all fractures, (3) open fractures, and (4) healed fractures. What the values in Table 4.3.4 mean is illustrated in Figures 4.3.7-4.3.11. A small standard deviation (smaller or similar to the average spacing) corresponds to a relatively uniform distribution of fractures per depth (typical of open fractures, for example, Figure 4.3.9c and Figure 4.3.11c). A standard deviation that is quite larger than the average spacing indicates fracture spacing that significantly varies with depth. Such varying spacing is typical for healed fractures (for example, Figure 4.3.7d and Figure 4.3.10d). Healed fractures are clustered near the elevations of paleo oil-water contacts where the predominant calcite precipitation occurred.

WELL SPACING		BEDDING PLANES	OPEN FRACTURES	HEALED FRACTURES	ALL FRACTURES
Tract 17	$\bar{s}$ [ft]	14.435	2.82	4.47	1.75
YU1711	σ_s [ft]	12.42	2.96	10.56	2.29
Tract 17	$\bar{s}$ [ft]	5.91	2.14	9.19	1.74
YU1755	σ_s [ft]	6.11	2.21	10.22	1.82
Tract 17	$\bar{s}$ [ft]	18.20	3.75	7.98	2.57
YU2511	σ_s [ft]	16.79	4.75	15.90	2.66
Tract 49	$\bar{s}$ [ft]	49.58	3.32	8.73	2.39
YU4007	σ_s [ft]	40.51	3.26	22.27	2.23
Tract 49	$\bar{s}$ [ft]	6.47	1.89	12.81	1.67*
YU4903	σ_s [ft]	7.26	2.42	16.74	2.22
Tract 49	$\bar{s}$ [ft]	16.42	1.90	11.16	1.75
YU5127	σ_s [ft]	17.21	1.58	11.70	1.45
Tract 49	$\bar{s}$ [ft]	6.11	2.61	2.89	1.49
YU3728	σ_s [ft]	6.55	2.96	5.44	2.04
Tract 49	$\bar{s}$ [ft]	9.08	5.40	6.69	3.52
YU4959	σ_s [ft]	12.65	6.14	8.65	5.25

TABLE 4.3.4 Calculated spacing of bedding planes and fractures identified on well log profiles in the Tract 17 and Tract 49 area. $\bar{s}$: average spacing; σ_s: standard deviation of spacing. The number of all fractures includes open and healed fractures. * In the top portion of well YU4903 there is a 45 ft zone without any fractures, not included in the calculation of average spacing.

Rosette diagrams of fracture strikes at all nine logged wells, listed in Table 4.3.2, are shown in **Figure 4.3.14**. In **Figure 4.3.15**, the predominant fracture strikes at the nine logged wells are superimposed on structure maps of the Seven Rivers M-horizon. Since this layer was deposited horizontally, its present day structure indicates as closely as possible the geometry of the fold that deformed the reservoir rocks at the end of the Late Permian. In Tract 49 (on the apex of the field structure) log fracture strikes are predominantly concentric to the field anticlinal dome. In Tract 17, the log fractures have a more complex relationship to the peri-anticline of the field structure. In general, at each well in Tract 17, fractures belong predominantly either to a NW/SE set or to a NE/SW set. The predominant fracture strikes are either parallel (YU1711) or orthogonal (YU17D5, YU1755, YU2511) to the peri-anticlinal hinge.

Besides direct observations of fractures in cores and outcrops, and fracture identification on log profiles, evidence on the fracture system that exists in the Yates field reservoir can come from water injection field tests. In such tests (available from four locations in or near the Tract 17 and Tract 49 area) water is pumped in boreholes and its dissipation at depth is monitored in order to identify large conductive features. **Figure 4.3.16** shows the water flow profiles at the four wells: YU14C4 and YU2233 in the vicinity of Tract17; YU49C6 near and YU4903 in Tract 49. The horizontal portions of the profiles indicate that at these elevations large conductive features (possibly fractures) are present. The inclined portion of the profile indicate that in these intervals water dissipation occurs gradually, possibly along numerous smaller fractures.

In the Yates field fractures in Cretaceous limestone (several hundred feet above the San Andres dolomite) are exposed in outcrops and road cuts. The depositional environment of Cretaceous rocks was similar to that of the San Andres formation in the Middle Permian. Therefore, the fracture system geometry in Cretaceous rocks may give indication of the geometry and connectivity of fractures, created in the San Andres prior to its dolomitization in the Late Permian. Most of the Cretaceous fractures are joints (tensile fractures), some of which are filled with precipitated calcite.

Fractures on outcrops of Cretaceous limestone in the Yates field are shown in Figure 2.3.12 and 2.3.18 (in Section 2.3 on fracturing in remote tension). Some joints in Cretaceous limestone in the Yates field are solution enhanced (**Figure 4.3.17**a), a few of them to openings as large as small caves. Solution-enhanced joints tend to be sub-vertical and north-south striking. Figure 4.3.17b shows a large fracture in a road cut of Cretaceous limestone in Pecos County, West Texas. If fractures such as those shown in Figure 4.3.17a and b exist in the Permian reservoir, they can provide pathways for significant flow. The top of the Cretaceous limestone, shown in Figure 4.3.17c, has moldic porosity that may be similar to that of the erosional top surface of the San Andres.

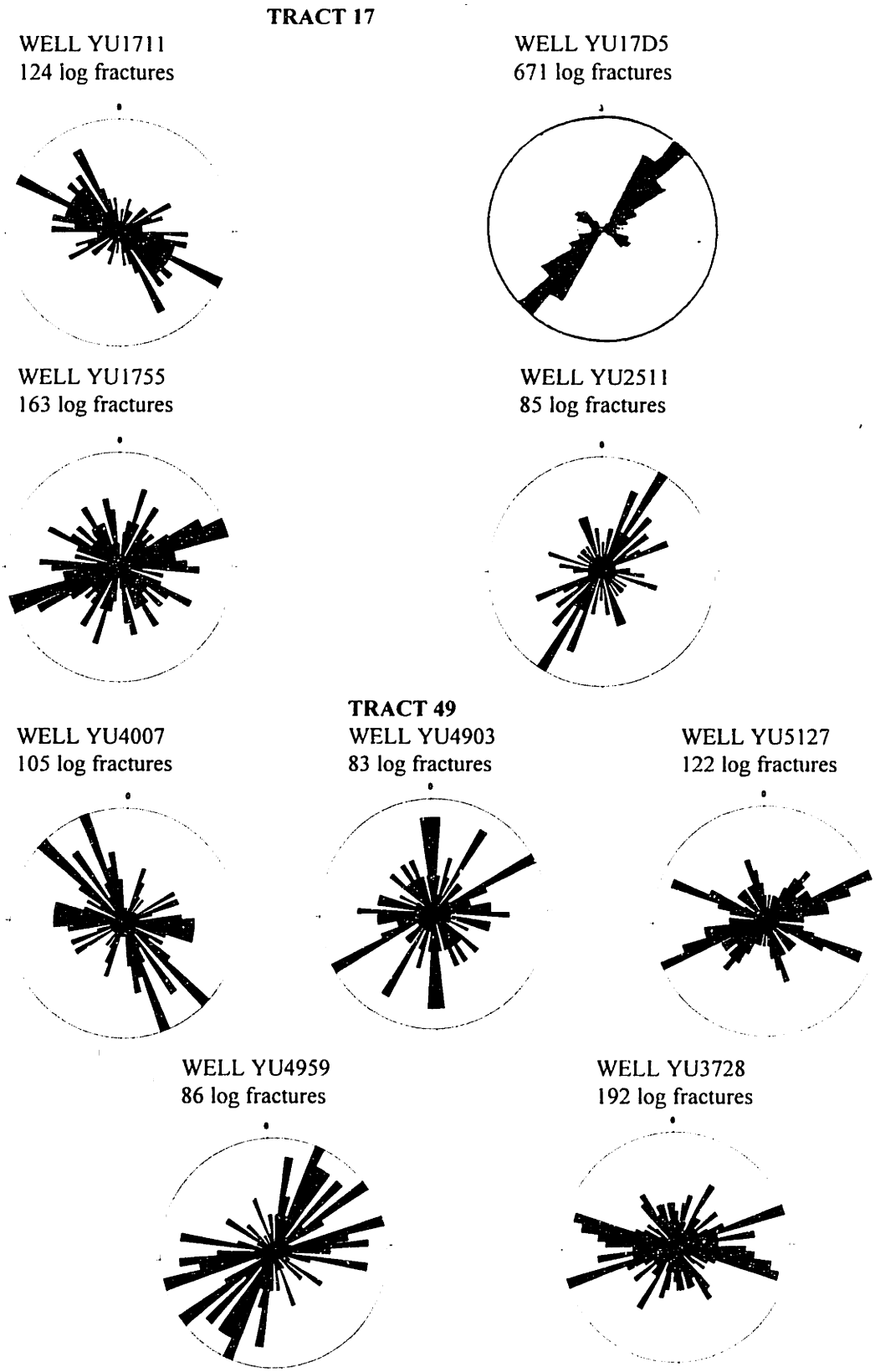
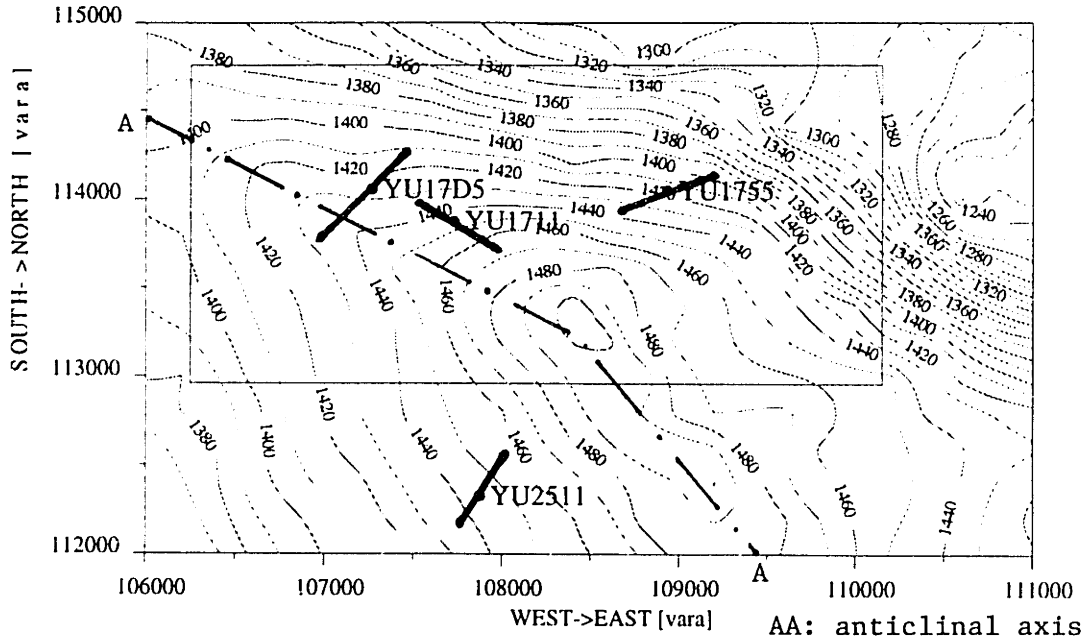


FIGURE 4.3.14 Rosette diagrams of fracture strikes at logged wells in and near the study area of Tract 17 and Tract 49.

a) TRACT 17



b) TRACT 49

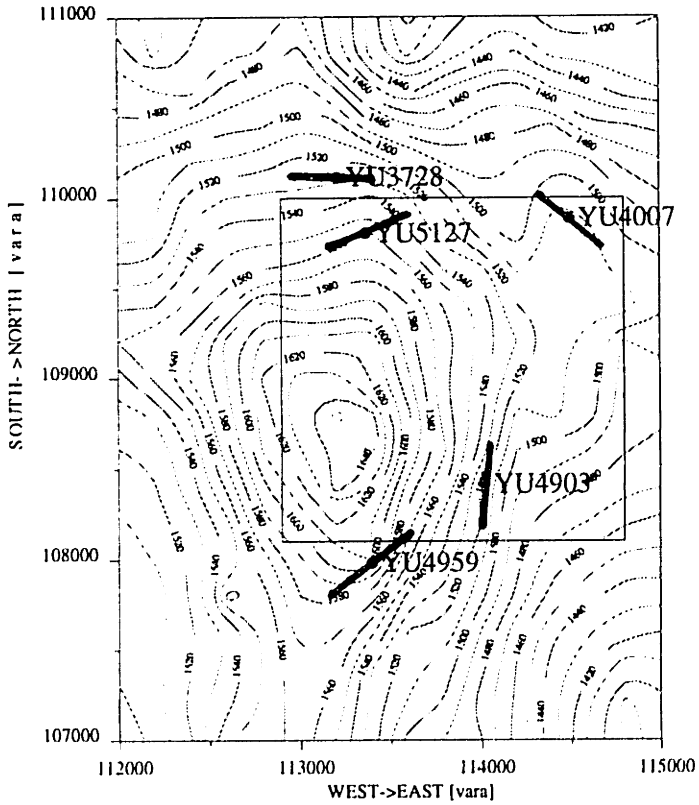
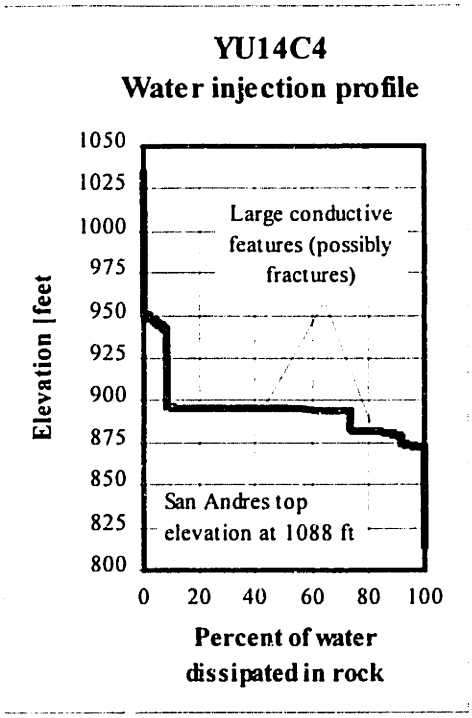
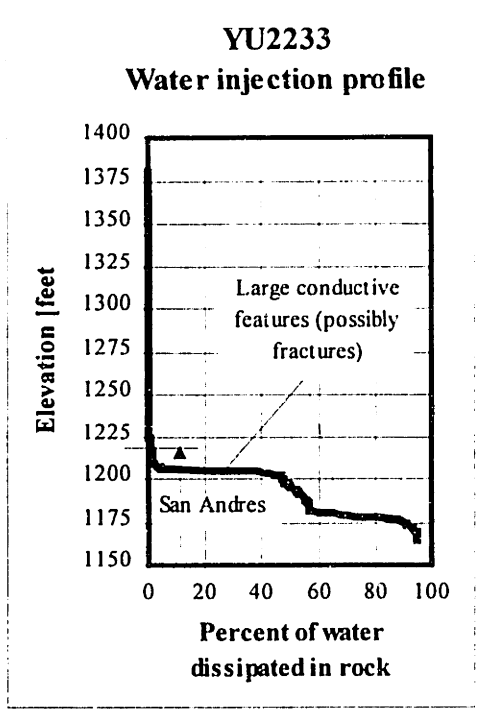


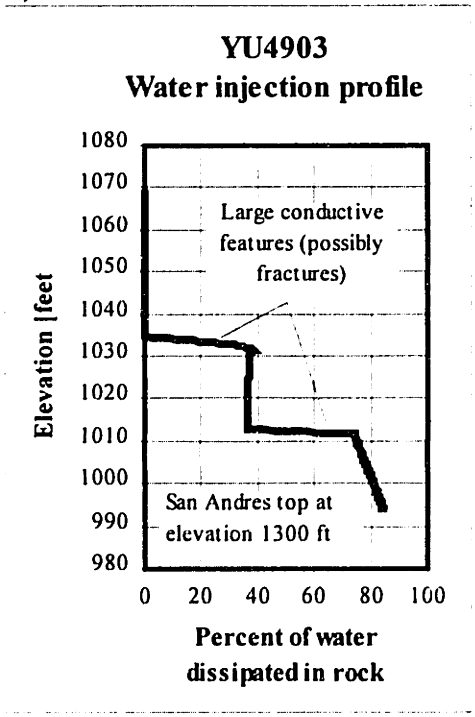
FIGURE 4.3.15 Predominant strike of fractures in the San Andres formation identified on well log profiles, superimposed on structure map of the Seven Rivers M datum: a) in Tract 17; b) in Tract 49.

a) TRACT 17





b) TRACT 49



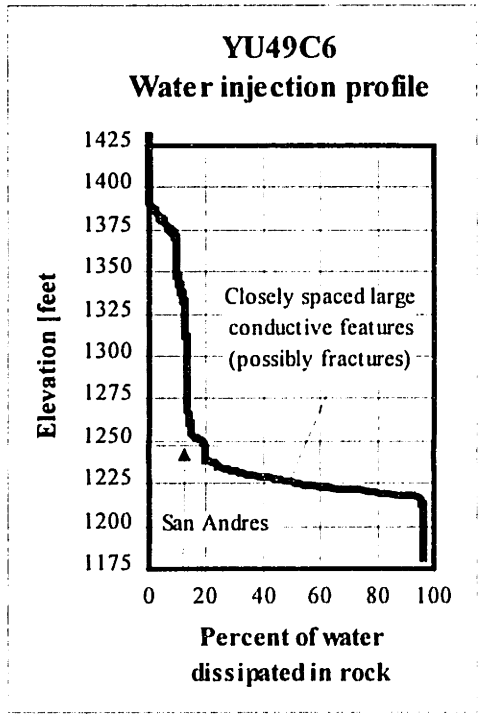
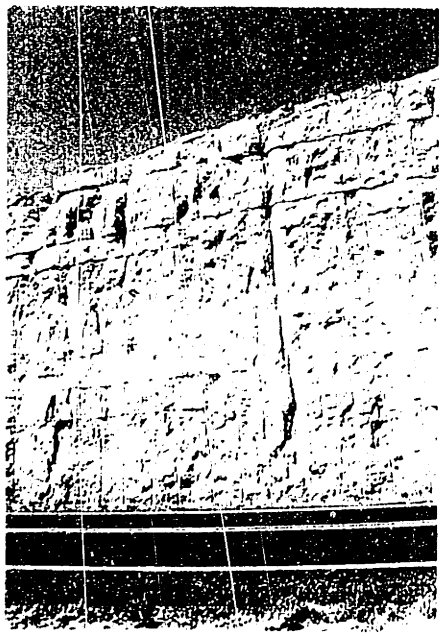


FIGURE 4.3.16 Water injection profiles: a) in two wells in the vicinity of Tract 17; b) in two wells in Tract 49. Flat portions of the profiles indicate that at these elevations a large amount of flow occurred from the borehole into the surrounding rock (suggesting the presence of large conductive features, possibly fractures).

a)

b)

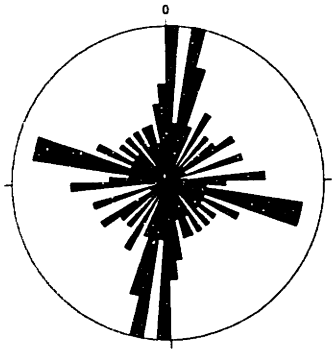


c)

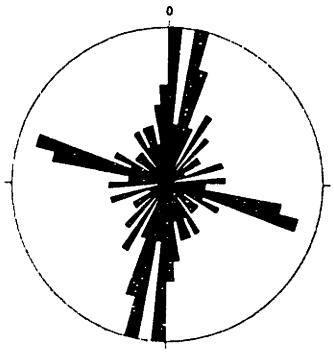
FIGURE 4.3.17 Exposures of Cretaceous limestone: a) solution enhanced joints in the Yates field; b) a large vertical joint exposed on a road cut in Pecos county, west Texas; c) moldic porosity of the erosional top of the limestone.

During the case study, strikes and dips of fractures in Cretaceous limestone were measured at five outcrops in the Yates field. The predominant outcropping sets are: 1) bedding-plane fractures, parallel to the limestone beds which are approximately horizontal or slightly dipping; and 2) two or three sets of vertical or very steeply dipping fractures, some cutting across many limestone beds (similar to the joints in Figure 4.3.17a and b). **Figure 4.3.18** shows rosette strike diagrams of the vertical fractures, including strike diagrams of (a) all fractures, (b) only joints, and (c) fractures identified as minor faults (relatively small, cutting across only one to a few beds). The most prominent joint set strikes slightly east of north; another set is approximately orthogonal to the first one. Such orthogonal joint patterns in sedimentary rocks in tectonically quiet regions are relatively common [Stearns & Friedman 1972]. Their development can be explained by temporary rotation of the directions of the horizontal principal stresses [Price 1966; Price & Cosgrove 1990; Einstein & Dershowitz 1990; similar to the mechanism illustrated in Figure 2.1.8].

a) ALL 111 FRACTURES IN CRETACEOUS LIMESTONE



b) 98 CRETACEOUS JOINTS



c) 13 CRETACEOUS MINOR FAULTS

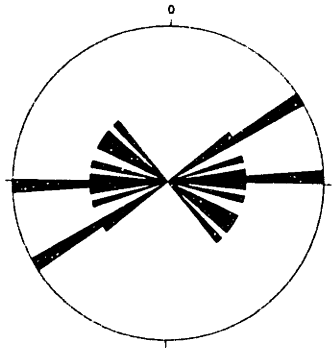


FIGURE 4.3.18 Rosette diagrams of strikes of fractures in Cretaceous limestone, compiled from five locations in the Yates field: a) all fractures; b) only joints; c) only faults.

4.4 GEOMECHANICS

Fracturing of the Yates field reservoir can be attributed to two main mechanisms: a regional stress system and a local stress system. At any moment in time and at any given location during the period of greatest interest, the Middle to Late Permian, fracturing occurred under the combined action of local and regional stresses, if these stresses exceeded the strength of the rock. Section 4.4.1 reviews the regional stresses that acted in the Yates field since deposition of the reservoir rocks in the Middle Permian to present day. Section 4.4.2 discusses the geomechanics of formation of the anticlinal structure of the reservoir strata in the Late Permian. Section 4.4.3 presents a hypothesis about the fracture genesis in the Yates field, based on the regional and local geological mechanisms.

4.4.1 Regional principal stress directions

The reservoir rocks were deposited during the Middle Permian which was a tectonically quiet period. Nor was there any major tectonism in the area after the Permian and to present day. The tectonic framework (Midland Basin - Central Basin Platform - Delaware Basin), which controlled the deposition of Permian rocks, developed during the earlier, tectonically active, Pennsylvanian period (Craig 1963; Hills 1970; discussed in more detail in Section 4.2.1). During the last period of regional tectonism which ended in the Early Permian, the maximum compressive stress, σ_1, acted slightly east of north (Figure 4.2.2b). At that time, east-west crustal extension constituted the minimum principal stress, σ_3, in the Permian Basin.

Figure 4.4.1 schematically illustrates the likely evolution of the regional stress system during the tectonically quiet geologic time from the Early Permian to present day. Since there have been no strong regional tectonic forces on the Central Basin platform after the Early Permian, σ_1 has most likely been the vertical stress of gravity. The minimum principal stress, σ_3, has continued to be in the direction of an east-west regional crustal extension. The intermediate principal stress, σ_2, has been oriented slightly east of north, following the direction in which σ_1 acted in the period of tectonic activity that ended in the Early Permian.

The orientation of the regional principal stress system in Figure 4.4.1 is supported by the orthogonal joint sets in Cretaceous limestone in the Yates field (Figure 4.3.18). According to Price (1966) and Einstein & Dershowitz (1990), vertical orthogonal joint sets develop in areas where σ_1 is vertical, and involve rotation of the horizontal σ_2 and σ_3. In the Yates field, the major set of nearly vertical joints, striking north to N10°E, probably developed parallel to the vertical σ_1 and orthogonal to the east-west σ_3. After relaxation of strains in east-west direction, σ_2 and σ_3 temporarily switched their directions. This caused formation of the second joint set, striking east-west, orthogonal to the new σ_3. After strain release in north-south direction, σ_2 and σ_3 rotated to their original directions.

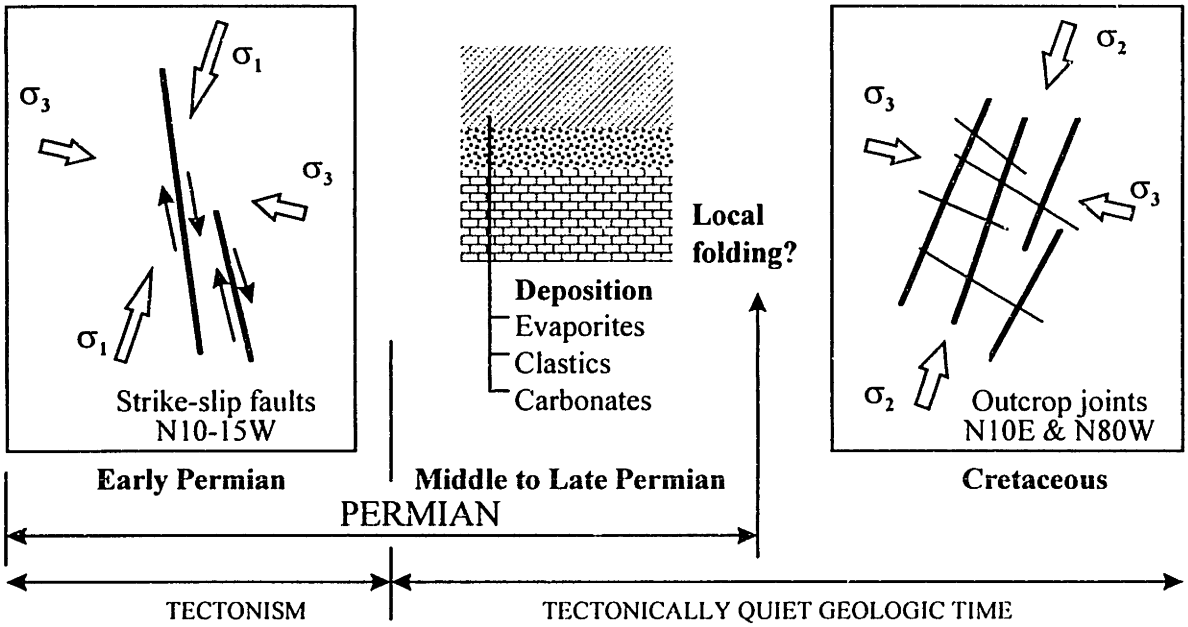


FIGURE 4.4.1 Regional stress directions in the Permian Basin after the last events of regional tectonism in the Early Permian.

4.4.2 Genesis of structure in Permian rocks

The present day structure of the Permian reservoir strata in the Yates field (illustrated in Figure 4.1.2, and in Figures 4.2.9-4.2.12) is an asymmetrical dome with its major anticlinal hinge striking to the northwest. This section presents a hypothesis about the geologic mechanism that created the structure of the Yates reservoir. This hypothesis is the basis for the conceptual model of fracture system genesis in the Yates field (discussed in Section 4.4.3), and the numerical simulation of the fracture system (presented in Section 4.5).

Differential compaction during deposition and shallow burial: Middle Permian

Minor structure formation, under vertical strains and effective stresses building at slow rates, likely occurred during deposition and shallow burial of the San Andres formation, due to gradual differential compaction of sediments. The mudstones on the west side of the Yates field compacted relatively more than the rigid, skeletal, cemented and minimally compressible grainstone shoals to the east. Also, to the east the carbonate shoals had prograded over highly compressible basinal lime muds (Figure 4.2.6b) which gradually failed and compacted as a function of their thickness, and caused bending of the overlying strata.

Intensified differential compaction during island exposure: Middle Permian

At times of relative sea level falls during the Middle Permian, increased effective stresses in the San Andres formation due to subaerial exposure of the limestone shoals must have led to greater compaction of the lime muds under the islands. This process, superimposed on the island hydrologic model of the Yates field [Craig 1988], is illustrated **in Figure 4.4.2**. During sea level fall, the total vertical stress, σ_v, remained the same in the shoals exposed as limestone islands. The pore pressure, p, decreased because of the lowered water table. The effective stress (σ_v -p) increased in all formations underlying the subaerially exposed islands. In the portions of the shoals that remained submerged under sea level, σ_v and p decreased by the same amount; hence there was no increase in effective stress. At that time, the highly-compressible basinal muds under the exposed islands must have experienced larger settlement and compaction compared to the muds under the submerged parts of the shoals. The stiff grain-supported limestone shoals did not undergo significant volume change, but stretched and bent over the differentially compacted lime muds.

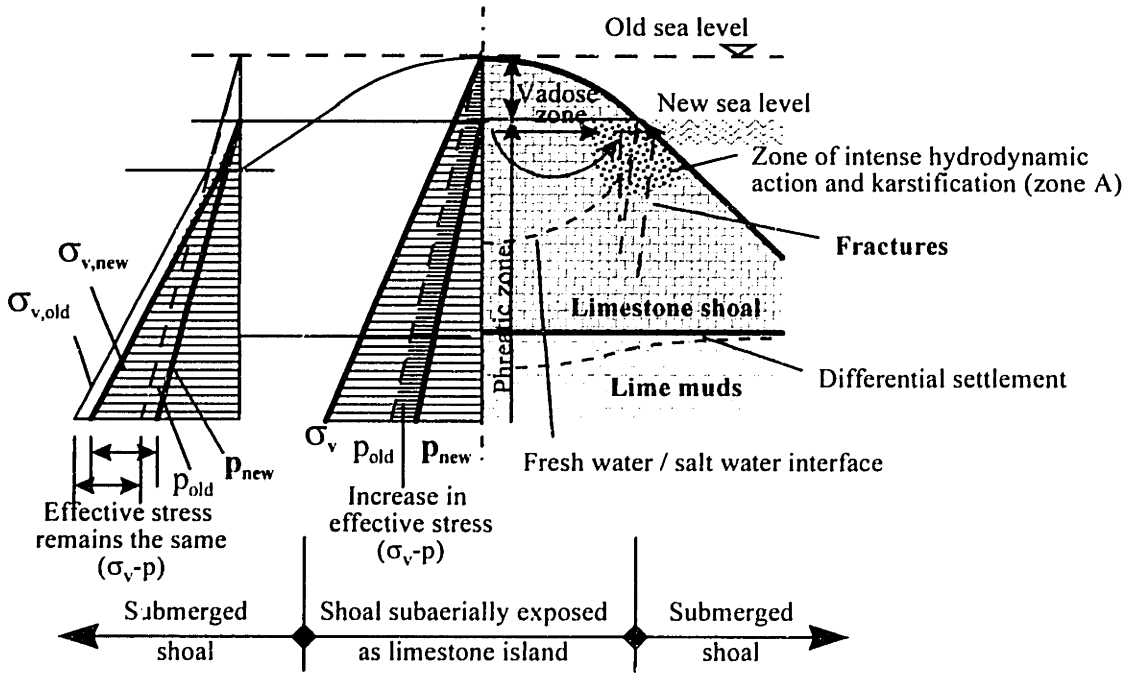


FIGURE 4.4.2 Stresses in the San Andres formation during island exposure in the Middle Permian.

Intense structure formation due to drape folding: end of Late Permian

By the end of the Late Permian, the Middle Permian sediments were already differentially compacted to some extent, and a minor structure existed in the San Andres formation. The limestones were brittle under the relatively thin, about 700 ft (213 m), overburden (limestone becomes ductile at much larger depths on the order of at least several thousands of feet; Stearns & Friedman 1972). Also, according to the seepage-reflux model (discussed in Section 4.2.2), most of the limestone had already been dolomitized in the evaporitic environment of the Late Permian. If folding with high loading and strain rates occurred in the brittle dolomite and the strength of the rock was exceeded, a significant structure-related fracture system would develop.

Field evidence (presented in Section 4.2.2) indicates that during the Late Permian Ochoa period (after Salado and before Rustler deposition) drape folding deformed the Middle Permian reservoir strata (San Andres, Grayburg, Queen, and Seven Rivers formations). In the absence of major regional tectonism after the Early Permian, two geologic events on the Central Basin Platform could have triggered the mechanism of drape folding [Craig 1990; Wadleigh, pers. comm.]: (1) movement along regional faults in the underlying Pennsylvanian rocks (5000 ft, 1524 m, below the reservoir); and (2) differential compaction of Permian sediments.

The process of drape folding of sedimentary strata due to fault movement in deeper strata is schematically illustrated in Figure 2.1.19. According to Hills (1970), during the Late Permian some downward movement may have occurred on the west side of the West Platform Fault (Figure 4.2.2b) and along other Pennsylvanian faults. However, there is no evidence anywhere on the Central Basin Platform that dip-slip movement along old Pennsylvanian faults was strong enough for the deep faults to penetrate into Permian reservoir rocks. Throughout the platform, the strikes of structure axis elongation in individual fields is very consistent, but generally differs from the predominant strikes of Pennsylvanian faults. The structure axes of Permian reservoirs on the Central Basin Platform are aligned along depositional strikes [Craig 1990]. Both Hills and Craig conclude that it is very unlikely faults in older strata to have controlled the structure in Late Permian reservoirs. Furthermore, most core fractures in the Yates field are tensile joints, while significant faulting would have produced numerous shear fractures in the Permian rocks.

Therefore, one can conclude that most of the structure of the Yates field Permian reservoir was formed by intense drape folding due to differential compaction of sediments at the end of the Late Permian. Why most of the differential compaction occurred after the Salado salt deposition can be explained by the combination of stresses, acting on the Central Basin Platform at the end of the Late Permian.

One of the main reasons for the high production from the Yates reservoir is the presence of natural high fluid pressures in the reservoir rocks, controlled by a regional aquifer on the Central Basin Platform. Field data from oil fields on the platform indicate that paleo flow occurred from the northwest to the southeast [Craig 1990]. Average fluid pressures in Yates reservoir rocks at a datum of 1050 ft (320 m) above the sea level were 800 psi (5.5 MPa) when the field was discovered. Today, after large amounts of oil and water have been extracted from the reservoir, the average fluid pressures at the same datum are 500 psi (3.4 MPa) [Wadleigh, pers. comm.]. However, during the Permian Guadalupe period, when the reservoir rocks were deposited, the aquifer in the San Andres limestone was still a phreatic aquifer. High pore fluid pressures most likely developed while the aquifer slowly changed from phreatic into confined during the deposition of the Seven Rivers anhydrite, which sealed the Yates reservoir.

By the end of the Salado period sediments with total thickness of about 600 ft (182 m) had been deposited above the San Andres limestones/dolomites. This number was calculated, using today's average thicknesses of the formations above the San Andres in the Yates field area, as follows (**Figure 4.4.3**):

Grayburg		Queen		Seven Rivers		Yates-Tansill		Salado		Total
50 ft	+	**50 ft**	+	**400 ft**	+	**50 ft**	+	**50 ft**	=	**600 ft**
(15 m)		**(15 m)**		**(122 m)**		**(15 m)**		**(15 m)**		**(182 m)**

(4-1)

In Tract 17 the San Andres formation has an average thickness of about 150 ft (46 m) overlying the datum 1050 ft (320 m) above sea level. Using a bulk density of ρ=2450 kg/m^3 for the medium compacted carbonates, clastics, and evaporites, the specific weight of the overburden above the San Andres at the end of Salado can be calculated as:

$$\gamma = \rho g = 2450\frac{\mathbf{kg}}{\mathbf{m^3}}9.81\frac{\mathbf{m}}{\mathbf{sec^2}} = 24\frac{\mathbf{kPa}}{\mathbf{m}} \tag{4-2}$$

where g is the acceleration of gravity. The vertical stresses at the datum at elevation 1050 ft (320 m) were:

$$\sigma_v = \gamma z = 0.024\frac{\mathbf{MPa}}{\mathbf{m}}(600+150)\mathbf{ft}\left(0.3048\frac{\mathbf{m}}{\mathbf{ft}}\right) = 5.5\mathbf{MPa}\,, \tag{4-3}$$

which is equal to the pore fluid pressure.

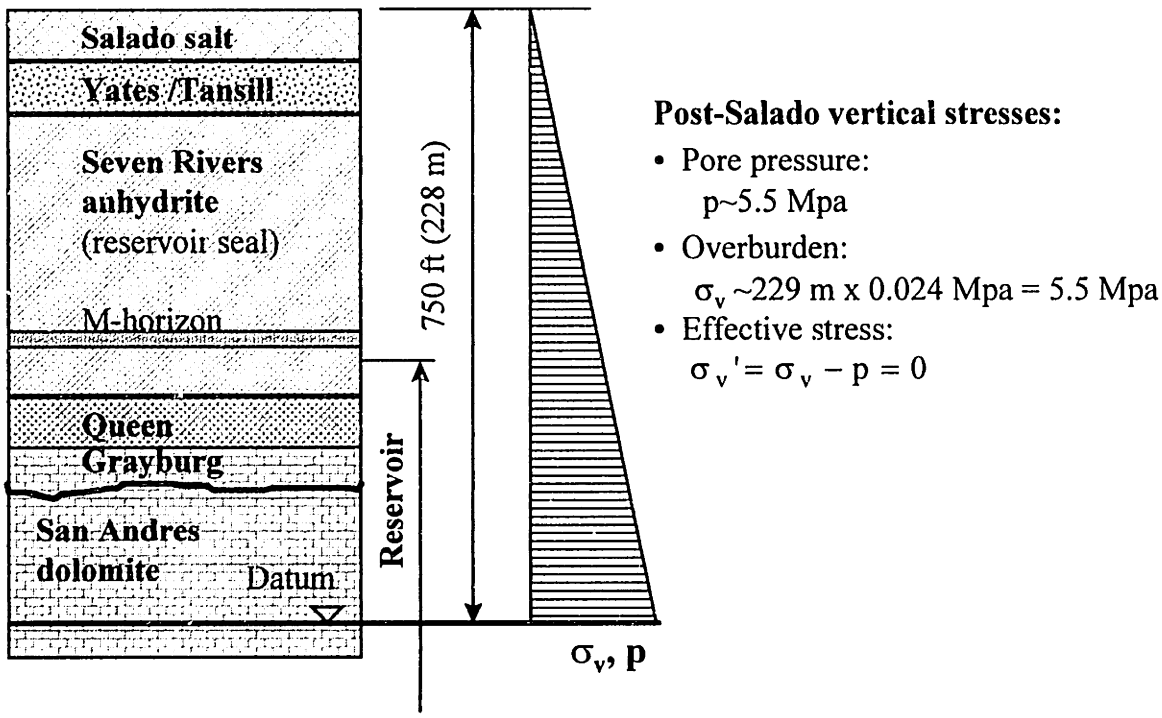


FIGURE 4.4.3 Calculation of total and effective stresses in the San Andres formation at the end of the Late Permian.

These calculations are only approximate since the density of the material and the fluid pressures in the San Andres varied with depth and location. Nevertheless, the calculations show that at the end of Salado the lithostatic stress and the pore fluid pressure were equal at various points within the San Andres formation. Therefore conditions of zero effective stresses occurred at different locations throughout the San Andres, where:

$$\sigma_v' = \sigma_v - p = 0 \qquad (4\text{-}4)$$

and $\sigma_V = p \approx 5.5\text{MPa}$ at the datum used in the calculations. Due to the loss of effective stress, under the relatively light overburden, portions of the San Andres formation experienced expansion and heave. With increase of the overburden, the strata that had expanded were subjected to higher effective stresses and started again to settle and compact. Meanwhile, at shallower depths the rocks were subjected to zero effective stresses and heave.

Due to temporary expansion and heave, followed by increased overburden and effective stress, the mud-supported sediments likely experienced rearrangement of grains due to collapse into voids of grains unsupported by other grains. This must have led to maximal volume reduction in the mudstones on the shallow west side and the deep east side and their transformation into the present matrix of very low porosity. Minimal total volume change occurred in the incompressible grain-supported shoals on the upper east side. During expansion and heave of the underlying basinal muds, the grainstone layers were pushed upward, stretched and fractured. During subsequent compaction of the muds, the grainstone strata above them stretched and bent to form the field anticlinal dome. The formations above the San Andres stretched to accommodate the convex upward shape of the dome, but retained their thickness.

The temporary conditions of zero vertical effective stresses, followed by settlement and differential compaction under increased overburden, does not exclude the possibility for faulting in deeper strata. Only additional field data from deep wells can indicate whether reactivation of Pennsylvanian faults also played a major role into the formation of the Yates field structure. Therefore, differential compaction of limestones and dolomites, due to a combination of overburden and high fluid pressure in the San Andres, following sealing of the reservoir at the end of the Late Permian, is accepted herewith to explain the formation of the Yates field structure. The drape folding occurred at strain rates high enough to produce a significant fracture system in the brittle carbonate rock of the San Andres. Some of the fractures probably propagated into the younger formations which were stretched and bent together with the San Andres. It is also possible that some, but not many, fractures propagated into the strata underlying the San Andres.

4.4.3 Geology-based hypothesis for the fracture system geometry

This section contains a conceptual model of the evolution of the fracture system in the Yates field Permian reservoir under the regional and local stresses, described in Section 4.4.1 and Section 4.4.2. **Figure 4.4.4** schematically illustrates the sequence of fracture system genesis in the Yates reservoir.

Crack initiation in limestone: Middle Permian

Figure 4.4.5 illustrates the possible mechanism of fracture initiation in the San Andres formation during deposition and shallow burial of limestone in the Middle Permian (Lower Guadalupe). Figure 4.4.5a represents the differential deposition of limestone lithofacies: compactible mudstones ("shales") to the west, and rigid grainstones, prograded over marine muds, to the east. Differential compaction caused by the different compactibilities of the lithofacies led to some minor drape folding (Figure 4.4.5b). Due to stretching of the upper layers, cracks opened on the crest of the shoal on the east side. Increased vertical strains due to mudstone

settlement on the west side initiated cracks in the limestone between the shale layers. The same sequence of deposition, compaction and crack initiation continued in all subsequent shoal cycles. Stretching of the beds on the top of the carbonate shoals, associated with the minor drape folding, defined planes of weakness, orthogonal to the beds, for fracture propagation and coalescence (Figure 4.4.5c). The strikes of these planes were parallel to depositional lineaments in the region, i.e. predominantly to the northwest and northeast.

Systematic regional fractures: Middle Permian

Regional fractures possibly formed by coalescence of cracks due to stresses induced in the San Andres limestone during relative sea level fall and island exposure in the Middle Permian (Figure 4.4.2). The differential settlement of the muds, underlying the limestone shoals, induced at the island margins strains that could have been high enough to cause fracturing. Such fractures would strike sub-parallel to the alignment of the limestone islands. According to Craig (1988), intense karstification occurred along such regional fractures and created sinkholes on the topographic surface and caves at depth.

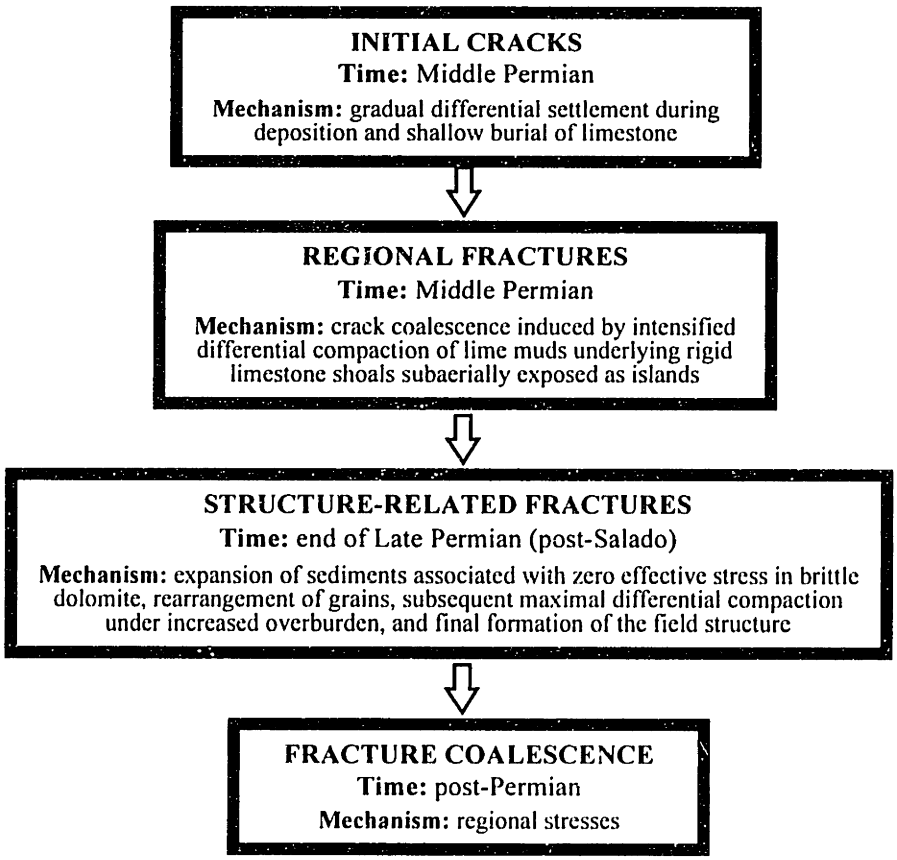


FIGURE 4.4.4 Sequence of fracture system genesis in the Yates reservoir rocks.

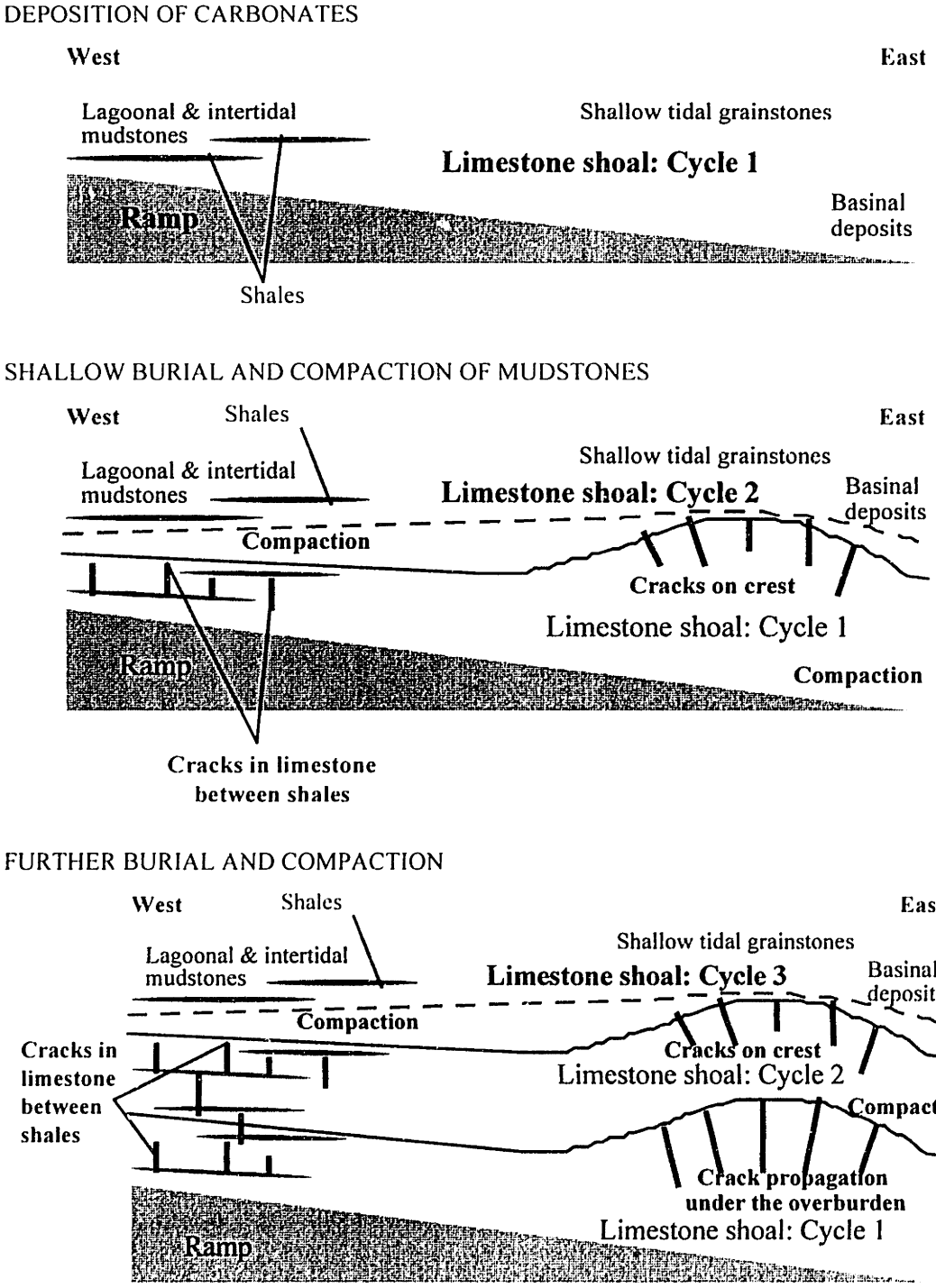


FIGURE 4.4.5 Crack initiation during deposition and shallow burial of limestone shoals in the Middle Permian.

Figure 4.4.6a shows the hypothetical pattern of two orthogonal joint sets (striking approximately N15°E and N75°W), fit by Craig (1988) to lineaments of sinkholes in the San Andres formation. Such a fracture system is consistent with

the regional stresses on the Central Basin Platform (Figure 4.4.1). Also, this hypothetical regional fracture system is very similar to the one in the Yates field Cretaceous limestone, shown in Figure 4.3.18 (the San Andres formation prior to its dolomitization had a depositional history very similar to that of the Cretaceous limestone). Figure 4.4.6b shows a more recent interpretation of the regional fracture trends with N50°W and N40°E lineaments by Tinker & Mruk (1995). 3D cave distribution in the reservoir, fracture counts in cores, and variation of the thickness of sediments above the San Andres (shown in Figure 4.2.7) were used in the interpretation of the regional fracture sets in Figure 4.4.6b. NE/SW and NW/SE regional fracture orientations can be explained with preferential orientation of maximum tension on nearly vertical planes striking parallel to depositional strikes (i.e. to the northwest and to the northeast).

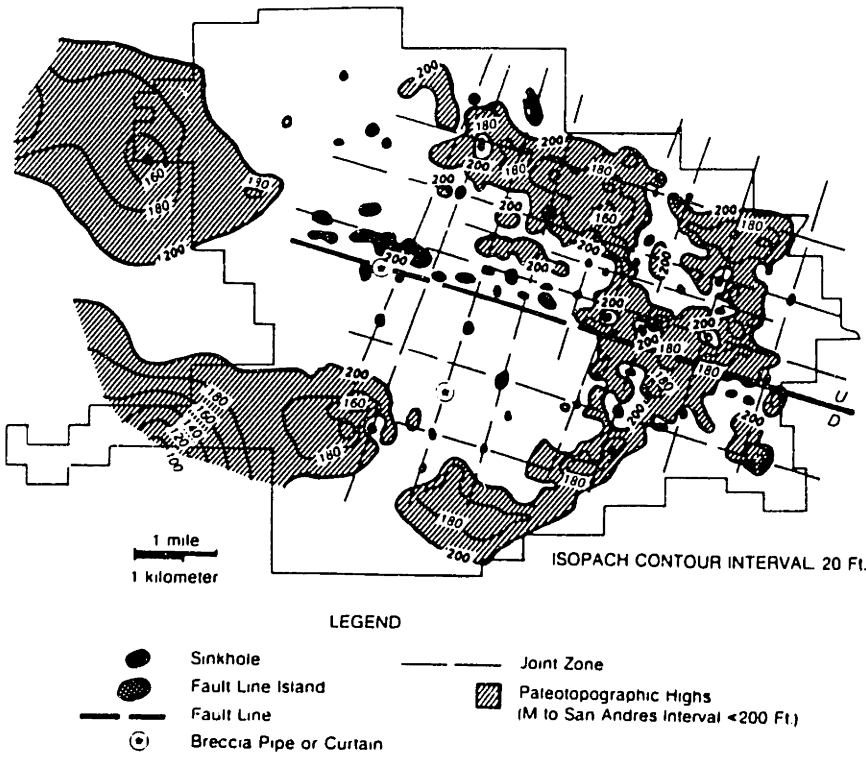
The regional patterns most likely include fractures with orientations similar to the ones suggested in both interpretations shown in Figure 4.4.6. On the one hand, the predominant fracture strikes are N50°W and N40°E, sub-parallel to the regional trends and the alignment of limestone islands. On the other hand, on the apex of the structure (on the east side), which was exposed as a cluster of islands (rather than two distinct trends of alignment) there is probably an additional north-south striking set. The orientaiton of such a set is consistent with the regional stress system (vertical σ_1 and east-west σ_3). A north-south striking set of solution-enhanced and calcite-filled fractures has been noted by Merkel (1992) in the FMS interpretation of well YU4007, located on the east side of the Yates field.

Dome-related fracture system: Late Permian

Most of the present day fracture system in the Permian reservoir formed during folding at the end of the Late Permian under high strain rates in brittle dolomite. **Figure 4.4.7**a schematically illustrates the Yates reservoir rocks at the end of the Late Permian, prior to the folding. By that time, the San Andres formation had been completely dolomitized in the evaporitic environment of the Middle Permian (Upper Guadalupe: Seven Rivers). Dolomitization increased the brittleness of the San Andres formation and likely initiated a great number of microcracks.

According to the hypothesis presented in Section 4.4.2, intense post-Salado folding occurred due to the maximal compaction of argillaceous mudstone on the upper west side and on the lower east side of the field. The compaction followed a process of swell and heave of reservoir rocks, and was caused by subsequent increase of overburden and effective stress. Under such stress conditions, it is very likely that the brittle San Andres dolomite experienced intense fracturing. Several factors must have enhanced the fracturing at that time, namely: the high fluid pressure, the high strain rates, the high intensity of microcracks, and the relatively low confining pressure (600 ft, 182 m, of overburden). The effect of fluid pressure, strain rate, and confining pressure on the brittle behavior of rocks is discussed by many authors [e.g. Singh 1981].

a)



b)

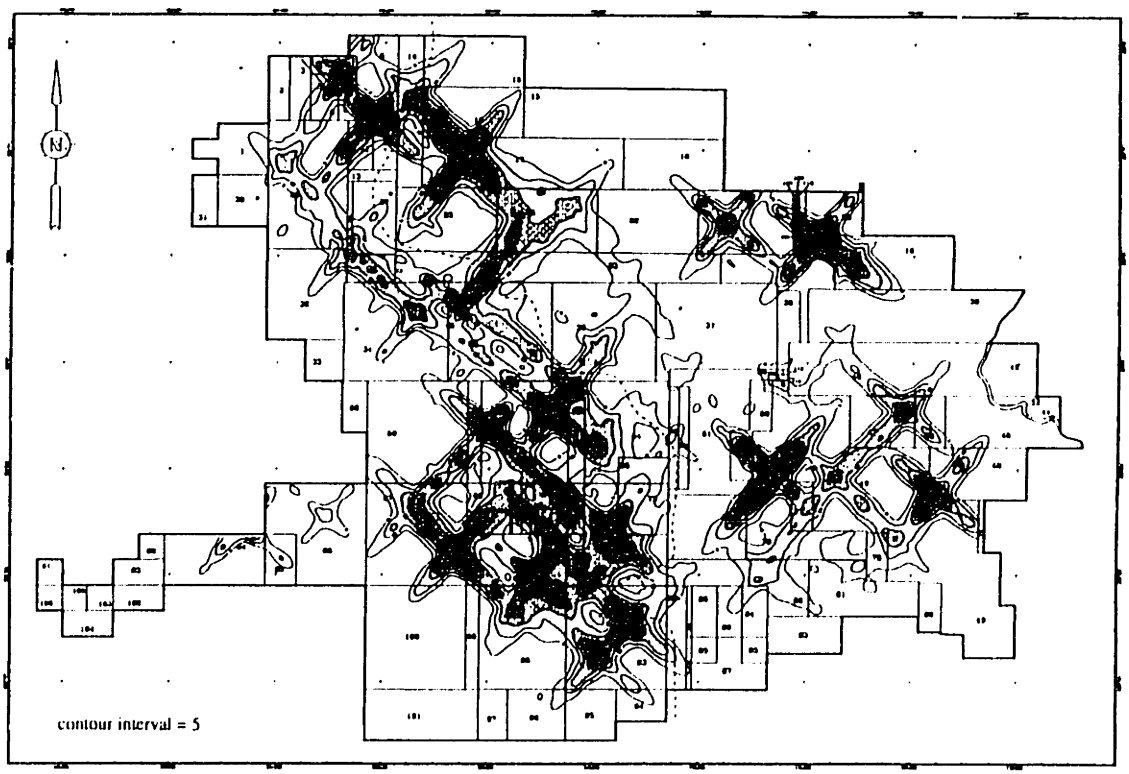
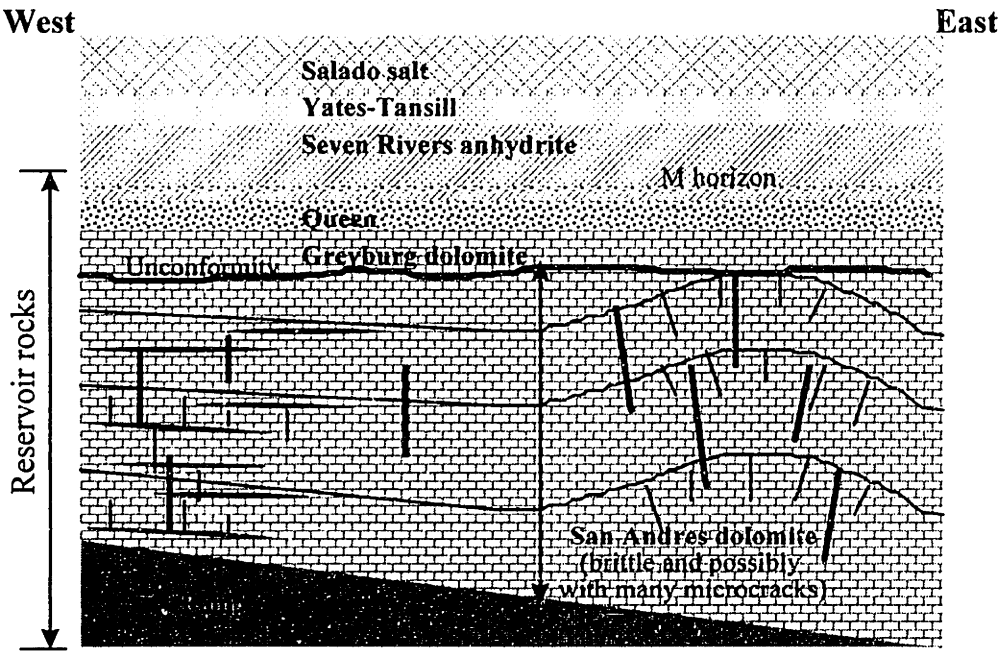


FIGURE 4.4.6 Regional fracture patterns, based on interpretation of paleo island alignments and cave distribution in the San Andres formation: a) interpretation by Craig (1988); b) interpretation by Tinker & Mruk (1995).

a)



b)

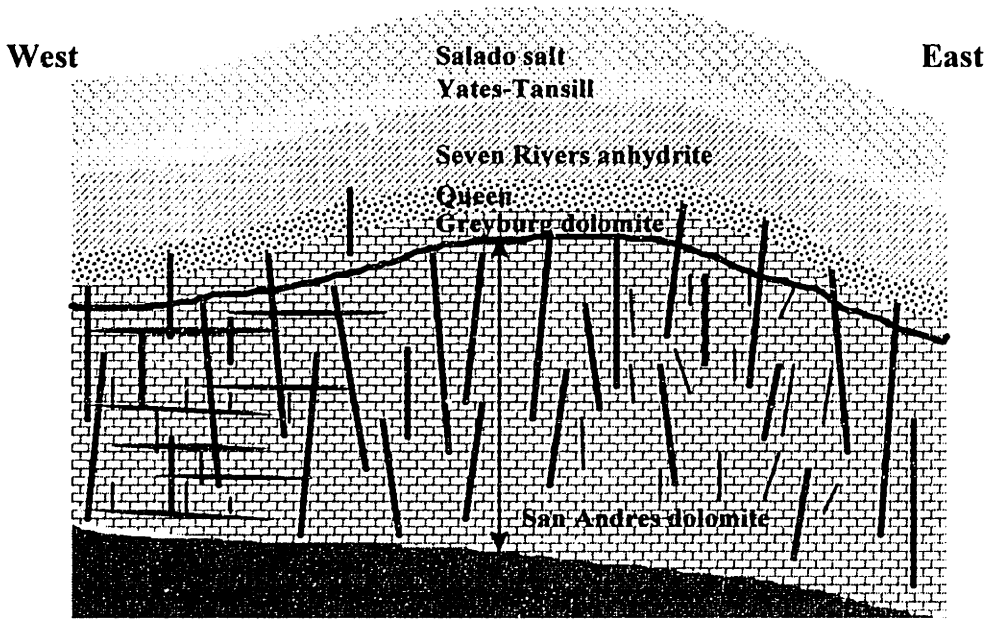


FIGURE 4.4.7 Development of the fracture system related to the Yates field anticlinal structure at the end of the Late Permian: a) the reservoir rocks prior to folding; b) the folded and fractured reservoir rocks.

Due to the high pore water pressures at the end of the Permian, tensile horizontal effective stresses, hence conditions for vertical hydraulic fracturing (see Section 2.3), developed in the San Andres formation. The horizontal stresses, σ_h, in normally consolidated deposits are smaller than the vertical stress of gravity, σ_v, which constitutes the maximum principal stress, σ_1. The horizontal effective stress, equal to σ_h less the pore fluid pressure, p, can be calculated as:

$$\sigma_h' = \sigma_h - \mathbf{p} < \mathbf{0} \qquad (4\text{-}5)$$

where $\sigma_h < \sigma_v$ and $\sigma_v \sim p$ at the end of the Late Permian (Section 4.4.2).

Additional tensile stresses were induced during the formation of the dome structure due to the stretching of dolomite beds. At the end of the Late Permian, the tensile strength of the San Andres dolomite (as it is typical for brittle rocks) was most likely very low (on the order of 4-6 MPa). Tensile stresses associated with the post-Salado folding of the San Andres formation exceeded the strength of the dolomite, and created a high-intensity system of tensile joints, related to the curvature of the newly-formed dome (Figure 4.4.7b).

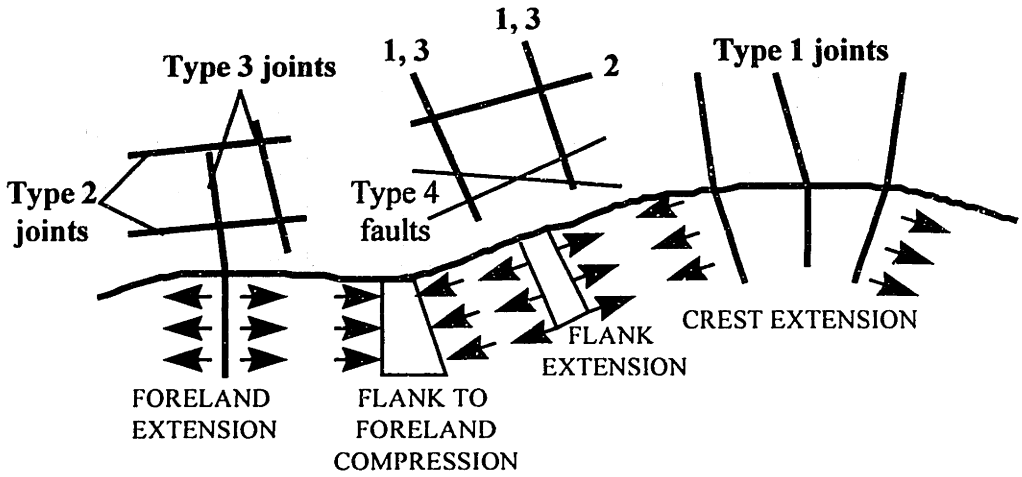
Figure 4.4.8a shows four types of fractures that can be associated with the field asymmetric anticline. On the crest of the structure, most fractures are concentric tensile joints created by the radial stretching of the dome (type 1 in Figure 4.4.8a). Most of the fractures in Tract 49 on the apex of the dome are of this type; they strike approximately orthogonal to the slope of the structure (Figure 4.3.15b).

The top of Tract 17 is in the zone of transition from rigid grainstones (including a local peak of the structure) on the east side to compactible mudstones to the west; downward the reservoir rocks in Tract 17 grade into better quality, more rigid, grainstone. The fractures in Tract 17 are of types 1, 2 and 3, shown in Figure 4.4.8 on the flank of the anticline and in the foreland. On the flank of the anticline, most fractures are extensional joints of type 1. Because of the tendency of the higher standing material to slide towards the compacting material, slope-parallel compression develops in the foreland and locally on the flanks. This stress creates the joints of type 2 in Figure 4.4.8a. According to the model of fracturing of mildly deformed sedimentary rocks by Price & Cosgrove (1990), tension joints of type 1 and 2 constitute the predominant type of fracturing in rocks of low tensile strength. According to the same model, if some slight buckling occurs in the foreland due to the compression from higher parts, then an additional joint set (type 3 in Figure 4.4.8a) may develop. Only in the strongest rocks, under high differential stress, shear fractures (type 4 in Figure 4.4.8a) develop as well.

Figure 4.4.8b schematically shows the relation of fractures to the anticline in plan view in Tract 17 (crest axis is indicated as A-A in Figure 4.3.5 and Figure 4.3.15). Since the anticline curves to the northwest, there is additional crest-parallel

extension along the east fold flank. The asymmetrical shape of the field structure in Tract 17 suggests that the fractures observed in wells YU1711 and YU2511 represent crest type 1, and flank to foreland type 2 fractures, respectively. The fractures in wells YU1755 and YU17D5 probably represent foreland type 3 fractures. Which type is predominant at a given location depends on the local maximum curvature of the structure.

a)



b)

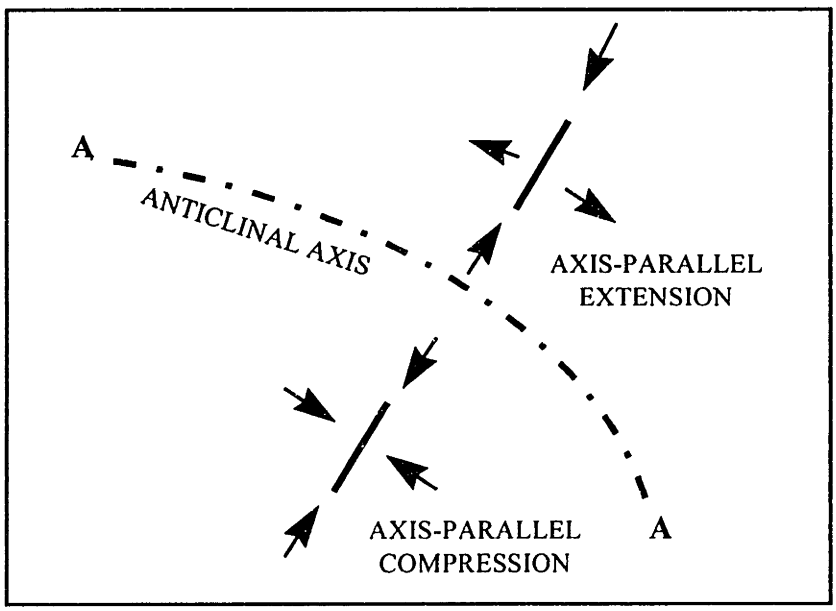


FIGURE 4.4.8 Relationship of fractures to the surface of a drape fold: a) vertical cross section; b) plan view.

Besides tensile fracturing of high intensity, field-scale faults possibly developed in the Permian reservoir as well (Figure 4.2.9). Such faults could have developed in the zones of maximum strain concentration to bound blocks that moved relative to one another during drape folding. The field-scale faults (if they exist) are most likely fracture zones of high intensity and connectivity. Considering the Late Permian conditions which favored hydraulic fracturing, the majority of the fractures in the vicinity of the field lineaments are possibly tensile joints oblique to the direction of relative movement between the blocks and orthogonal to the local direction of the minimal (tensile) principal stress (see Figure 2.2.2).

Regional fracture lineaments: post-Permian

Under the regional stresses some of the structure-related fractures possibly connected by secondary fractures in post-Permian time, and formed field-scale lineaments. Once the structure-related fracture system developed and the folding-induced strains were accommodated, the regional stress system in the Yates field became predominant again. In the regional system, the vertical stress of gravity is the maximum principal stress, σ_1, and an east-west relaxation of compressive stress constitutes the minimum principal stress, σ_3. This stress regime favors a system of vertical tensile joints with predominantly north-south strikes (but also including joints with different strikes). It is possible that such joints developed after the Permian and connected the structure-related fractures to form field-scale lineaments. Most of the fractures, related to the anticlinal structure (types 1, 2 and 3 in Figure 4.4.8), were parallel or perpendicular to the general trends of the asymmetric anticline (northwest and northeast). Field-scale lineaments, created by the fracture coalescence, would have the same predominant orientations.

Present day geometry of the reservoir fracture system

Considering the hierarchical development, discussed above, the present day fracture system in the reservoir rocks in the Yates field can be divided into two major systems. System 1 includes regional fractures that developed prior to the formation of the field anticlinal structure. System 2 includes fractures which developed in relation to the reservoir structure, created by drape folding at the end of the Late Permian.

System 1 consists of vertical or very steep fractures which initiated predominantly on the crests of the major shoaling cycles and propagated downward. These fractures were affected by the karstification of the San Andres during the Permian, and by calcite precipitation at the paleo oil-water contacts. Also, since System 1 fractures formed in limestone prior to dolomitization, many might not have survived the diagenesis. The ones that have, however, today are likely to be solution enhanced and partially filled with calcite. System 1 includes northwest and northeast striking fractures, and possibly an additional north-striking set on the apex of the reservoir structure on the east side.

The fractures of System 2 are by far the most numerous in the reservoir due to the combination of factors, discussed in Section 4.4.2, which acted at the end of the Late Permian. Most of the fractures, identified by log analysis and observed in cores, belong to System 2. Fractures of System 2 are open, since they formed in the Late Permian, after the last precipitation of calcite. Most fractures are probably relatively small in size. However, their great number can allow for a good connectivity. The intensity of System 2 fractures is greatest in the areas of maximum curvature of the anticlinal structure (possibly coinciding with field-scale lineaments of faults/drape folds, shown in Figure 4.2.9). The orientations of the fractures are related to the local strike and dip of the dome. The connectivity of System 2 is increased by the presence of additional connecting fractures that developed after the structure-forming events in the Late Permian.

4.5 NUMERICAL APPLICATION OF THE 3D MODEL

This section summarizes the results from the numerical application of the 3D mathematical model, presented in Chapter 3, for simulation of the fracture system in the Yates field reservoir rocks. The modeled fracture system includes fractures that are subparallel to the regional depositional trends (System 1, as defined in Section 4.4.3), and fractures that are related to both regional depositional trends and field anticlinal structure (System 2, also defined in Section 4.4.3). Section 4.5.1 and Section 4.5.2 contain the results from the numerical simulations of the fracture system in Tract 49 and Tract 17, respectively. Presentation of the numerical modeling for each area includes the following:

- Inference of the model parameters from geology and field data, namely inference of number of fracture sets, fracture intensity P_{32}, fracture plane orientation PDF, fracture size variation, and relation of fractures to the field structure.
- Development of specific algorithms, incorporated in modules of the program GEOFRAC.
- Results from numerical generations and comparison to field data.

Mean and PDF of fracture plane orientations

The number of fracture sets, and their mean orientations, defined by the regional stresses, are the same in Tract 17 and Tract 49. Throughout the Yates field, field data consistently show that there are two major fracture sets. One set strikes to the northwest and the other one strikes to the northeast. The expected value for the strike of the northwest set is N50°W, and for the northeast set is N40°E (parallel to the two major regional depositional trends on the Central Basin Platform). Both fracture sets are composed of vertical or steeply dipping fractures. In the primary process of the numerical model, the mean pole direction (θ, ϕ) of fracture planes

that belong to the first set is defined as (40°, 90°), i.e. orthogonal to a vertical plane striking N50°W. The mean pole direction of fracture planes in the second set is defined as (-50°, 90°), i.e. orthogonal to a vertical plane striking N40°E. Fisher distribution with Fisher constant k=20 is used for the generation of fracture planes in each set. This spherical PDF (with the relatively high k=20) preferentially produces planes with poles at small angles with the specified mean pole orientation, as they occur in reality (mostly steep fractures in a narrow dip range).

Relation of fractures to the field structure

At different locations in the Yates field, the predominant fracture strikes often deviate from the two regional directions (see Figure 4.3.14 and Figure 4.3.15), since the majority of the fractures are also related to the curvature of the reservoir structure. The structure-defined variation of fracture orientations around the mean directions of the two major sets is relatively simple in Tract 49 (discussed in Section 4.5.1), and quite complex in Tract 17 (Section 4.5.2).

Fracture size variation and P_{32}

The inference of size variation and P_{32} from geologic data is discussed in Section 4.5.1 for Tract 49, and in Section 4.5.2 for Tract 17. In general, the inference of precise fracture size distribution from the available field data is practically impossible since fractures are sampled only by one-dimensional field tests: logs and cores. Fracture size is assumed solely on the basis of known relationships of fracture sizes to layers in sedimentary rocks (discussed in detail in Section 2.3).

In the 3D geometric-mechanical model, fracture intensity P_{32} should be inferred from the magnitude of geologic stresses and strains, and from the mechanical properties (strength, deformation moduli) of the host rock. However, since this information is not available yet, P_{32} is inferred from field data (i.e. following a procedure that is typical for a geometric, rather than for a geometric-mechanical, model). Inference of cumulative fracture intensity (Equation 3-6) from logs and cores is straightforward since P_{32} (as long as it is not too low) is linearly related to the mean spacing of fracture intersections in boreholes [Dershowitz 1984; Dershowitz & Herda 1992; Low 1986]. In the inference procedure, the emphasis is placed on making assumptions about which field-sampled fractures are significant and which ones are not. **Table 4.5.1** summarizes advantages and drawbacks of the field sampling methods, used in the Yates field, in terms of the information which they provide on fracture intensity.

Well water injection and monitoring gives very good, direct evidence of where the conductive fractures are (Figure 4.3.16). The shape of the water dissipation profile indicates if a major conductive fracture exists at a given elevation, or if numerous conductive fractures intersect the well along a certain depth interval. However, since only sparse data is available from water injection tests in the Yates field, one cannot rely on this information alone for inference of fracture intensity.

FIELD TEST	ADVANTAGES	DRAWBACKS
Water injection	Gives direct indication of where the major conductive features are.	Very limited testing, not enough data to form a representative field sample.
Continuous cores	Fractures, as well as rock lithology, porosity, calcite filling, etc. are directly observed and measured.	Core description (available only in electronic form) includes numerous minor fractures and microcracks, difficult to distinguish from major conductive fractures.
FMS/FMI logs	Logs penetrate almost the entire thickness of the San Andres formation.	Fractures are not directly observed, but identified (possibly very subjectively) on resistivity profiles of the logged wells.

TABLE 4.5.1 Field tests and their limitations as indicators of fracture intensity in the Yates reservoir.

There is a dramatic difference between the intensity of conductive fractures, suggested by water injection tests, and the intensity of fractures identified on log profiles or observed in cores. For example, Figure 4.3.16b shows that there are only two large conductive features (possibly fractures) in a 50 ft water profile at well YU4903. Figure 4.3.10, on the other hand, depicts 83 fractures and 30 bedding planes, identified on a FMS/FMI log at the same well, in the depth range immediately below the one where the water test was performed. Well YU4903 is in the middle of Tract 49 where at elevations between 800 ft and 1200 ft the San Andres consists of massive dolomitic grainstone. It is unlikely that the lithology and fracturing of the San Andres formation at well YU4903 vary so much. Hence most fractures on log profiles are most likely non-conductive and relatively small.

Identification of fractures on FMS/FMI profiles is a subjective procedure. For example, **Figure 4.5.1** shows the rosette diagrams of fracture strikes at one well in Tract 17 (YU1711) and two wells in Tract 49 (YU4007, YU5127), determined from two independent analyses of the same FMS/FMI logs. For all wells, the number of fractures, identified in the first log analysis, is quite different from the fracture number in the second log analysis. Only the predominant strikes of fractures are approximately the same in the two analyses.

Core description is the only test where fractures are actually observed and not assumed based on other data. Since there are not enough water injection tests,

whereas the log analyses give ambiguous information on the number of intersected fractures, fracture intensity P_{32} is determined through analysis of core data from the Tract 17 and Tract 49 areas. Locations in and near the two tracts where continuous cores were taken are indicated in Figure 4.3.5. However, only a computer database (not the actual cores) is available to the case study. The database includes description of every available foot of core, including fracture count and comment on whether the fractures penetrate or not across the entire cored foot. Inference of fracture intensity from the core database is discussed in Section 4.5.1 for Tract 49 and in Section 4.5.2 for Tract 17. Also, Section 4.5.1 and Section 4.5.2 present some specific algorithms, implemented in the 3D fracture system model, which enable reading of Stratamodel porosity and shale data in order to account for their effect on fracture intensity in Tract 49 and Tract 17, respectively.

4.5.1 Numerical simulation of the fracture system in Tract 49

Inference of model parameters

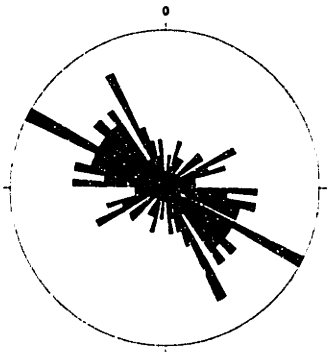
Fracture intensity P_{32} is inferred from the mean spacing of large significant fractures intersected by continuous cores (core locations are shown in Figure 4.3.5b). **Table 4.5.2** illustrates the types of fractures, defined as "significant" in the inference procedure, through examples from the core database of Tract 49.

Fracture 1 in Table 4.5.2 is totally open, and penetrates more than one cored foot (type TOPN). It is stained by oil, which indicates that it is conductive. This fracture is of the type shown in Figure 4.3.6b. Fracture 2 in Table 4.5.2 is partially filled with calcite and has aperture of 5 mm which is much larger than that of similar fractures. The large aperture is due to solution enhancement. This fracture, which penetrates the entire core diameter, is of the type FILO shown in Figure 4.3.6c. It very likely belongs to System 1 fractures, created prior to folding of the reservoir strata, and affected by solution enhancement and calcite precipitation. Fracture 3 in Figure 4.5.2 is filled with calcite (type FILL). Even though at its intersection with the core this fracture is not conductive, the fact that its aperture (thickness) is so large (4 mm) and its dip is so steep (80°) indicates that it probably extends to elevations above and below the local level of calcite precipitation. Therefore, this fracture is considered significant since portions of it are likely to be conductive. Fractures of type FILL with smaller thickness or flatter dips are not considered significant. Fractures of types PONP (non penetrating the entire thickness of the core) and INDU (drilling-induced) are ignored in the inference procedure.

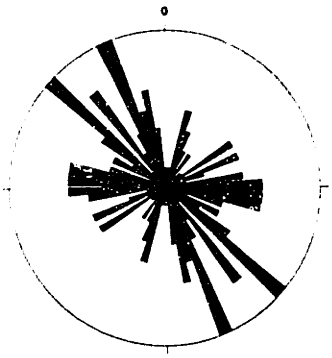
In order to gather information on fracture size, vertical fractures in cores were specifically described during the inference procedure. **Table 4.5.3** shows two examples of "vertical fractures" identified in continuous core.

LOG ANALYSIS 1

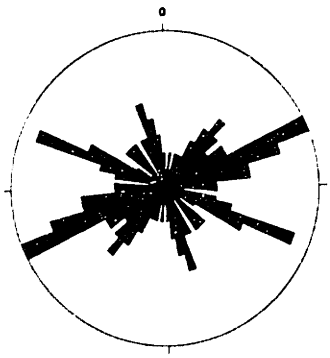
YU1711
124 log fractures



YU4007
105 log fractures

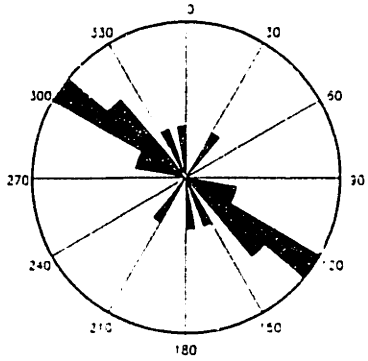


YU5127
122 log fractures

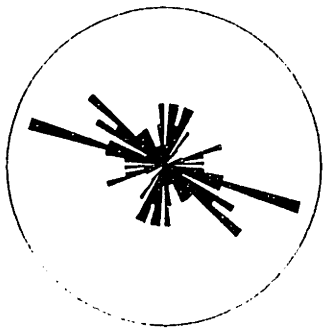


LOG ANALYSIS 2

YU1711
20 log fractures



YU4007
71 log fractures



YU5127
47 log fractures

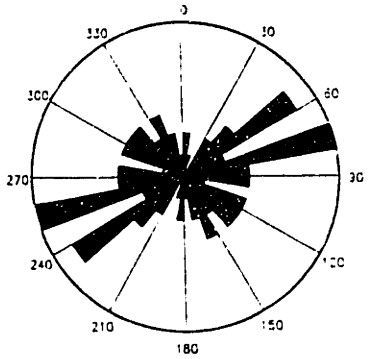


FIGURE 4.5.1 Comparison of number and strikes of fractures, identified on FMS/FMI profiles during two independent log analyses.

WELL	DEPTH	ELEV.	POROS.	APER.	DIP	TYPE	COMMENT	FRACTURE
FRACTURE 1								
YU5005	1425	1157	9.7	N/A	80	TOPN	Light oil stain	
YU5005	1426	1156	12.2	N/A	85	TOPN	TOPN continues	BIG TOPN 85
YU5005	1427	1155	14.3	N/A	85	TOPN	TOPN continues	
YU5005	1428	1154	10.7	N/A			Light oil stain	
FRACTURE 2								
YU4936	1536	1035	5.6	5	50	FILO	Calcite; cont. below	BIG FILO 50-70
YU4936	1537	1034	6.5	5	70	FILO	Solution; calcite	
FRACTURE 3								
YU5008	1508	1118	9.6	4	80	FILL	Calcite filling	
YU5008	1509	1117	N/A	4	80	FILL	FILL continues	BIG FILL 80

TABLE 4.5.2 Significant fractures identified on cores from Tract 49: a) totally open, penetrating fracture (type TOPN); b) solution-enhanced, partially filled with calcite fracture (type FILO); c) steep fracture filled with calcite at its intersection with the well (type FILL). The shaded column includes the interpretation of significant fractures in the current case study. The rest of the table presents data from the core database of the Yates field.

WELL	DEPTH	ELEV.	POROS.	APER.	DIP	TYPE	COMMENT	FRACTURE
YU4936	1433	1138	25.7	0.2	90	POPN		start POPN 90
YU4936	1434	1137	24	0.4	90	POPN		8 ft long
YU4936	1435	1136	12.6	.	.		Microfractures	
YU4936	1436	1135	10.6	.	.		Microfractures	
YU4936	1437	1134	10	.	90		Solution	
YU4936	1438	1133	19.2	0.4	90	POPN		
YU4936	1439	1132	22.9	0.4	90	POPN		
YU4936	1440	1131	24.2	.	90	POPN		end POPN 90
YU4936	1565	1006	18.9	.	90	TOPN	Rubble; cont. below	start TOPN 90
YU4936	1566	1005	16.3	.	90	TOPN		8 ft long
YU4936	1567	1004	19.1	.	.	TOPN	Rubble; cont. below	
YU4936	1568	1003	14.1	.	.	TOPN	As above	
YU4936	1569	1002	11.9	.	90	TOPN	Some calcite fill	
YU4936	1570	1001	11.1	.	90	TOPN	Microfractures	
YU4936	1571	1000	10.8	.	90	TOPN		
YU4936	1572	999	12.1	.	70	TOPN		end TOPN 90

TABLE 4.5.3 Vertical fractures, identified on continuous core. The shaded column includes interpretation of vertical fractures in the current case study. The rest of the table presents data from the core database of Tract 49.

Table 4.5.4 summarizes the spacing of significant fractures, identified in cores available from twelve wells located inside or in the vicinity of Tract 49. The average spacing of significant fractures varies from 6.5 ft to 14.9 ft (2-4.5 m) in dolomite that has porosity 6.9-20.4% (excluding the two extreme values for wells YU4910 and YU4957). There is a tendency of fewer well fracture intersections where the porosity of the rock is higher than 20% (see data for wells YU4910, YU4951, YU4936), although sometimes many intersections are observed in high-porosity rock (well YU5152). Nine vertical fracture intersections have heights from 4 ft to 40 ft (1.2-12 m), with all but two having heights less than 10 ft (3 m). It has to be noted that the real "diameter" of a vertical fracture is usually larger than the height of its well intersection. The dip distribution of significant fractures, intersected by cores in Tract 49, is shown in **Figure 4.5.2**.

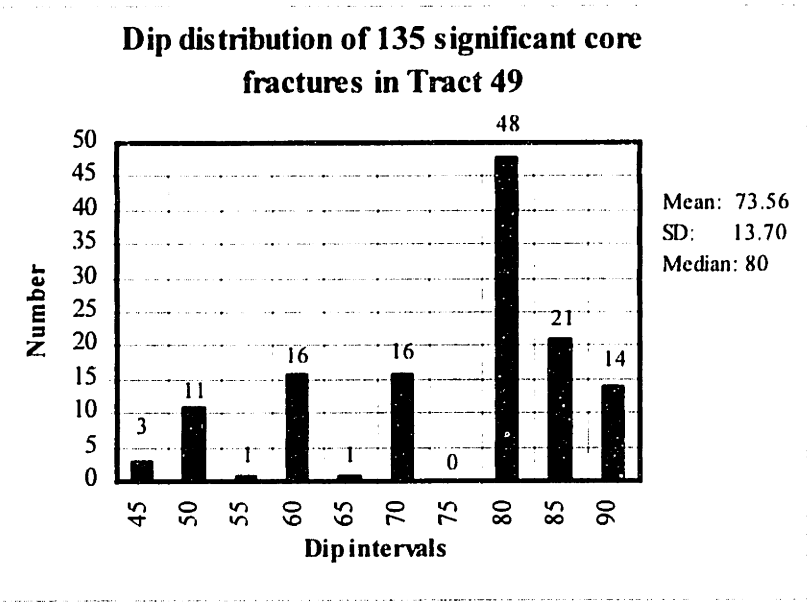


FIGURE 4.5.2 Dip distribution of 135 fractures, identified as "significant" on cores from twelve wells in the Tract 49 area.

The fact that some vertical fractures have been intersected at all indicates that the intensity of significant fractures in the Yates reservoir is very high. However, the probability of intersection of a large vertical fracture by the coring device (with diameter of about 3 in; 7.5 cm) is extremely low. Because of the great number of smaller fractures, some of the vertical ones have been intersected by the coring device. Therefore, the heights of vertical fractures intersected by cores mark the lower end of size variation of significant fractures. The upper end (largest sizes) can be estimated through analysis of the thickness of dolomite cycles in the San Andres formation (summarized in **Table 4.5.5**). In the middle and upper portions of the San Andres formation, there are three layers of similar thickness that could

have acted as mechanical units during folding when the San Andres dolomite strata bent over compacted underlying muds. The three mechanical units are: the relatively thick second dolomite cycle, combined with two very thin cycles above it (Unit 1 in Table 4.5.5); three cycles of approximately the same thickness combined together (Unit 2 in Table 4.5.5); and the top portion of the San Andres dolomite (Unit 3 in Table 4.5.5). The average thickness of the three units is about 60 ft (18 m) which can be assumed as an approximation for the average diameter of the largest fractures in Tract 49. Fractures of that size and larger possibly exist in the Yates reservoir, and, even if they are not too many, because of their large sizes, they may provide for significant reservoir conductivity.

WELL	n_{ave} [%]	N	s_{ave} [ft]	COMMENT
YU4001	12.54	13	7.27	SD=2.82 ft.
YU4033	18.85	4	13	
YU4910	20.52	3	25.5	No fractures in zone with n_{ave}=27.4%.
YU4951	12.27	3	12	No big fractures in zone with n_{ave}=21.5%.
YU4957	5.85	2	2	Spacing of smaller fractures s_{ave}=13 ft.
YU5005	8.05	3	7	All big fractures are in zone with n_{ave}=11.46%. Smaller fractures with s_{ave}=8.5 ft in n_{ave}=6.03%.
YU5008	6.9	6	13.6	SD=6.19 ft. Mostly calcite-filled.
YU5123	8.83	9	6.5	SD=5.85 ft. No big fractures below elevation 1150 ft.
YU4936	12.03	11	14.90	SD=10.32 ft. Spacing varies: s_{ave}=5.67-10.7 ft in zones with n_{ave}=10.10-12.6%; no big fractures in high porosity n_{ave} >20%. Vertical fractures: 4 ft in n_{ave}=9.73%; 7 ft in n_{ave}=18.65%; 40 ft in n_{ave}=10.1%.
YU49C3	18.24	9	12	SD=7.59 ft. Spacing varies: s_{ave}=8-8.5 ft in zones where n_{ave}=13.2-18.23%; s_{ave}=21.5 ft in n_{ave}=21.5%. Vertical fractures: 7 ft in n_{ave}=13.2%; 21 ft in n_{ave}=14.3%; 8 ft in n_{ave}=18.23%; 9 ft in n_{ave}=21.5%.
YU5131	11.51	4	9	No natural fractures below elevation 1191 ft.
YU5152	20.37	9	12.5	SD=11.90 ft. Spacing varies: s_{ave}=5.5-6.6 ft in n_{ave}= 18.72-25.75%; s_{ave}=12.67 ft in n_{ave}=18.76%. Vertical fractures: 4 ft in n_{ave}=21.82%; 6 ft in n_{ave}=18.72%.

TABLE 4.5.4 Spacing of significant fractures, identified in cores from Tract 49. n_{ave}: average core porosity; N: number of significant fractures identified at the given well; s_{ave}: average spacing of significant fractures; SD: standard deviation of spacing between significant fractures (given only where N>5).

	THICKNESS			
	MIN	AVERAGE	MAX	STRATIGRAPHIC LAYER
	250.35	309.1012	347.05	Cycle 1 dolomite
UNIT 1	38.1	51.7887	64.5	Dolomite layer 2
	1	5.9067	18.1	Dolomite layer 3
	1	3.3237	5.2	Dolomite layer 4
	40.1	61.0191	87.8	Total Unit 1
UNIT 2	10.6	19.2643	28	Dolomite layer 5
	9.4	17.4586	26.7	Dolomite layer 6
	7.9	17.6432	32	Dolomite layer 7
	27.9	54.3661	86.7	Total Unit 2
UNIT 3	29.4	71.6617	113	Dolomite to San Andres top
	6	34.3345	58.4	Greyburg
	19.3	24.4931	36.1	Queen

TABLE 4.5.5 Thickness of dolomite cycles in the San Andres formation.

In Tract 49 the relationship of the fractures to the field anticlinal structure is relatively simple: fractures are predominantly concentric. This is not only suggested by rosette diagrams of log fracture strikes (Figure 4.3.15b), but, more important, results from simple mechanical analysis. The San Andres formation in Tract 49 includes the apex of the field anticlinal dome. This indicates that during folding the reservoir strata in Tract 49 experienced radial extension which defined tangentially striking planes of maximum tension. The two regional fracture sets (striking to the northwest and northeast) exist in Tract 49, but which one of them is locally the major set depends on which one is "more concentric" to the dome.

Numerical algorithms

Most of the algorithms for generation of fold-related fracture sets, implemented in the 3D model and in program GEOFRAC, were developed in the context of the Yates field case study. The methods for generation of fracture sets related to a curved surface (fold) were presented in detail in Section 3.3.2.

An additional algorithm, implemented in a module of program GEOFRAC, reads porosity values from the Stratamodel and calculates the average porosity of the host rock that surrounds a numerically generated polygon. Then the polygon is retained as a fracture with probability P_f as a function of the average porosity:

$$\mathbf{P_f} = \mathbf{f_n}\left[\frac{\mathbf{1}}{\mathbf{N}}\sum_{\mathbf{i=1}}^{\mathbf{N}}\mathbf{n_i}\right] \qquad (4\text{-}6)$$

where N is the number of geocells from the reservoir Stratamodel intersected by the polygon, and n_i is the porosity of the i-th intersected cell.

Numerical simulations

Based on the inference procedure, presented above, the following parameters of the 3D fracture system model are selected for the simulations in Tract 49:

- Mean orientations of two fracture sets (inferred from regional geology): vertical planes, striking N50°W and N40°E.
- PDF of fracture plane orientations: Fisher (k=20) which produces close clustering of plane pole orientations around the mean pole, as explained on page 315-316).
- Modeling volumes: five modeling volumes for the five logged wells: YU4007, YU4903 and YU5127 inside Tract 49, and YU3728 and YU4959 in the vicinity of Tract 49. Each modeling volume is defined by the following boundaries: a horizontal datum surface at 700 ft above sea level; four vertical planes (two striking south-north and two striking east-west), each at a horizontal distance of 300 ft (91 m) from the vertical well; a quadratic upper surface, parallel to the San Andres top and located 100 ft (30 m) above it. The coefficients of the quadratic function are calculated through a polynomial fit with MATLAB to the elevations of the San Andres top at the wells in the Tract 49 area.
- Fracture intensity (inferred from field data, since the stress and strain magnitudes, and the mechanical properties of the San Andres dolomite are not known yet): 0.3 [ft^2/ft^3] for each set, i.e. the total $P_{32}=2(0.3)=0.6$ [ft^{-1}], established through simulations to produce expected spacing in vertical boreholes E[s]=10 ft (3 m).
- Fracture size distribution (approximate procedure, since field sampling of fracture sizes is not available): the mean of the equivalent radius is assumed $E[R'_e] = 30ft$ (9 m; equal to half the average thickness of mechanical units in the San Andres formation). Marking in terms of relative size is assumed of the type $A'_i > E[A]$, $R'_{e,i} < 3E[R'_e]$, where $E[A']=\pi(30)^2=2827$ [ft^2] is the expected area of fractures. This type of marking (see Section 3.2) eliminates the polygons with extremely small and extremely large sizes, and produces fracture areas that have a standard deviation $\sigma_{A'} = 0.9E[A']$ and median $M[A'] = 0.8E[A']$. 70% of the produced fractures have areas smaller than the average area.
- Effect of porosity (approximate procedure since the exact effect of high porosity on fracture intensity has not been studied yet): fracture intensity in areas with porosity n>20% is assumed to be half of the fracture intensity where n<20%, i.e. $P_{32}|n>20\%=50\%P_{32}=0.5(0.6)=0.3$ [ft^{-1}]. A

fracture is discarded with probability P_f=0.5 if the average porosity of the dolomite matrix around the fracture is n_{ave}>20%. This method results in discarding fractures with relatively small sizes, since they are the most likely to be located entirely in an area of high porosity. Thus in high porosity dolomite only relatively large fractures are retained which results in larger spacing of fractures intersected by a vertical well.

- Relationship to the anticlinal structure (inferred from local geology): fractures are predominantly concentric to the field dome. The shape of the field anticline is approximated by a cubic surface, fit to the shape of the Seven Rivers M horizon in Tract 49 (illustrated in **Figure 4.5.3**). The coefficients of the cubic surface are calculated through a polynomial fit with MATLAB to the elevations of the Seven Rivers M horizon in the wells in Tract 49. The strike of every fracture is compared to the strike of the cubic surface at the center of the fracture. If the fracture strike deviates more than 30° from the local fold strike, the fracture is rotated with a probability P=0.6. A smaller percent of fractures (P=0.2) are rotated to be orthogonal to the structure strike (and to the major concentric fracture set). The remaining fractures (P=0.2) retain their orientations defined by the regional stress. Rotation is not applied to the largest numerically generated fractures ($A'_i > 2E[A']$; 10% of the total number of fractures), since they possibly belong to System 1 which was formed prior to folding; hence they are related only to the regional depositional trends (this algorithm is approximate and can be modified when more detailed information is available on the intensities of System 1 and System 2 fractures).

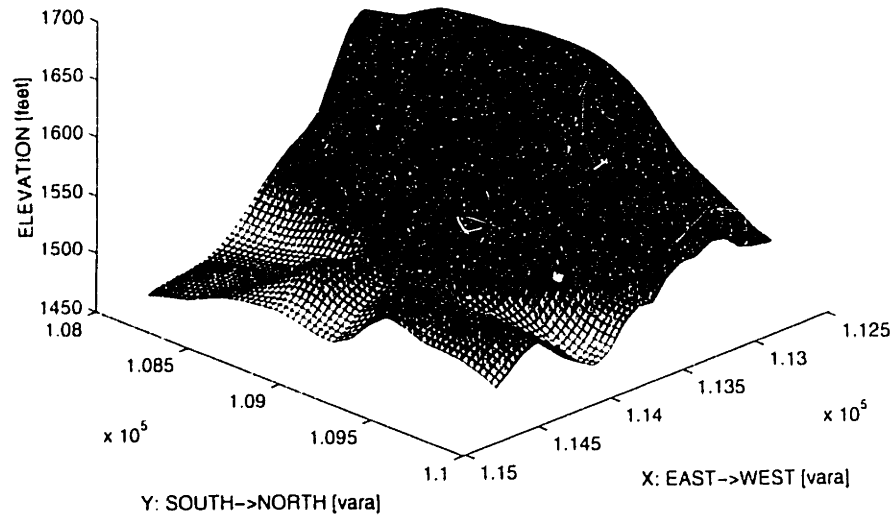
Table 4.5.6 summarizes results from numerical simulations of the fracture system around the three logged wells in Tract 49. These results are the last step in the simulations which established the correct fracture intensity P_{32} of the system in Tract 49. There is a linear relationship between the fracture intensity P_{32} and the fracture spacing in boreholes (similar to the linear relationships established by Dershowitz 1984 and Low 1986). The horizontal sections in **Figures 4.5.4a - 4.5.6a** illustrate the shape of the field anticlinal structure in the vicinity of the logged wells in Tract 49. Fracture traces on hypothetical horizontal outcrops in Figures 4.5.4b - 4.5.6b show how the numerically generated fracture system in Tract 49 relates to the field structure in the vicinity of the three logged wells in Tract 49 (YU4007, YU4903, and YU5127).

Figure 4.5.7 illustrates an example of fractures intersected by a a simulated borehole at well YU4007. The simulated boreholes intersect mostly large fractures, therefore the average size of fractures intersected by a borehole is larger than the mean size of the entire population of fractures in a given simulation.

#	N	P_{32} [ft^{-1}]	SIMULATED VERTICAL BOREHOLE		
			s_{ave} [ft]	σ_s [ft]	N_b
WELL **YU4007**: input P_{32}=2(0.3)=0.6 ft^{-1}, E[R'$_e$]=30 ft, 150 ft borehole					
1	45684	0.593718	7.42	7.45	17
2	45641	0.602779	6.88	6.59	20
3	**45454**	**0.607149**	**10.13**	**5.62**	**14**
4	45169	0.603828	12.84	10.43	10
5	48273	0.599406	9.68	10.04	12
6	46740	0.604518	6.47	4.48	19
7	43844	0.583571	7.41	6.45	13
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
9	43475	0.578401	10.20	7.84	14
10	46953	0.602952	11.66	11.02	13
WELL **YU4903**: input P_{32}=2(0.3)=0.6 ft^{-1}, E[R'$_e$]=30 ft, 200 ft borehole					
1	50321	0.592024	8.34	8.55	24
2	47780	0.571936	11.60	9.25	18
3	**49854**	**0.601455**	**12.37**	**11.68**	**16**
4	51041	0.611094	11.60	7.83	16
5	49802	0.600112	11.31	8.46	17
6	53181	0.604869	8.78	6.96	23
7	51964	0.610513	6.22	6.45	29
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
9	48544	0.583886	10.65	14.52	15
10	51105	0.589622	9.81	10.25	21
WELL **YU5127**: input P_{32}=2(0.3)=0.6 ft^{-1}, E[R'$_e$]=30 ft, 200 ft borehole					
1	49106	0.589063	10.64	10.64	19
2	48274	0.579433	9.22	9.12	21
3	**48708**	**0.58493**	**14.05**	**12.89**	**15**
4	51585	0.587576	8.11	6.29	21
5	47071	0.546611	11.86	7.29	17
6	48965	0.58914	8.75	7.09	22
7	49046	0.583544	10.48	13.12	18
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
9	48174	0.575173	10.3	5.18	18
10	49459	0.592878	10.71	7.53	18

TABLE 4.5.6 Results from numerical simulations of the fracture system in the vicinity of wells YU4007, YU4903, and YU5127 in Tract 49. P_{32}: fracture intensity defined as cumulative fracture area per unit volume; E[R'$_e$]: expected value of fracture equivalent radius; N: total number of fractures generated in a simulation; N_b: fractures intersected by a simulated borehole; s_{ave} and σ_s: average value and standard deviation of fracture spacing in the simulated borehole. Dip distributions of the fractures in the shaded rows are shown in Figure 4.5.11.

a) ACTUAL FOLD SHAPE IN TRACT 49: SEVEN RIVERS M HORIZON



b) REPRESENTATION OF SEVEN RIVERS M BY A CUBIC SURFACE

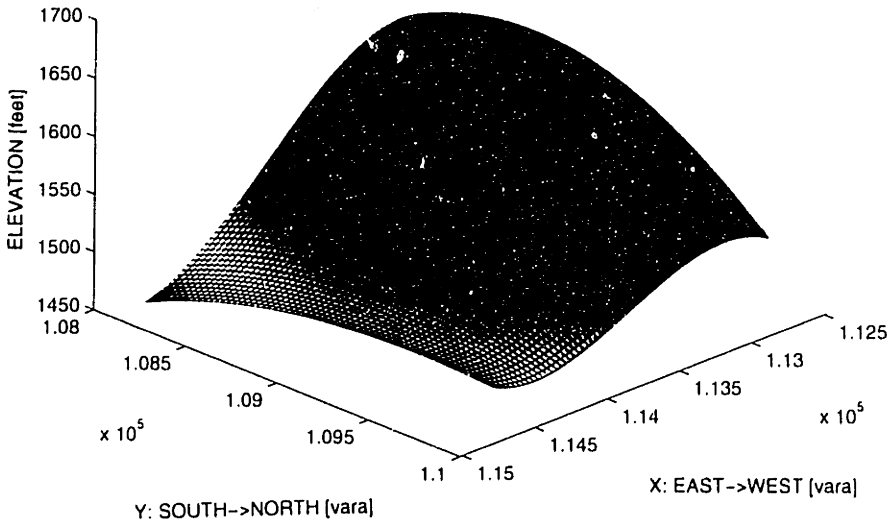
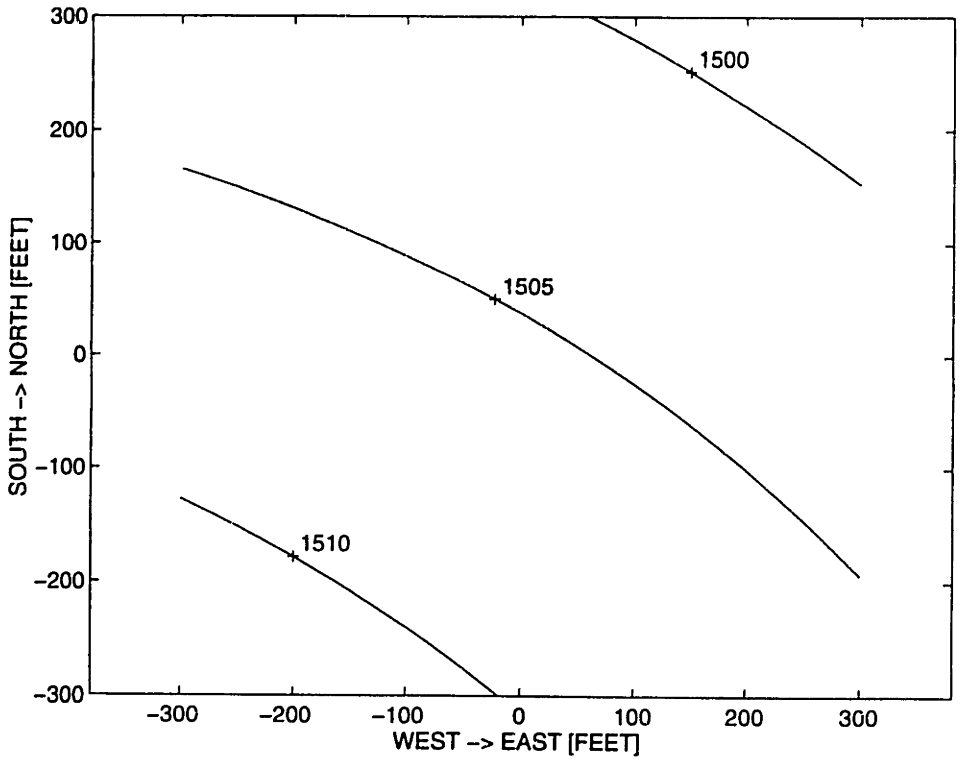


FIGURE 4.5.3 Fit of a cubic surface to represent the fold in Tract 49: a) actual shape, and b) cubic approximation of the Seven Rivers M horizon.

a) FOLD SURFACE



b) HORIZONTAL OUTCROP OF FRACTURE SYSTEM

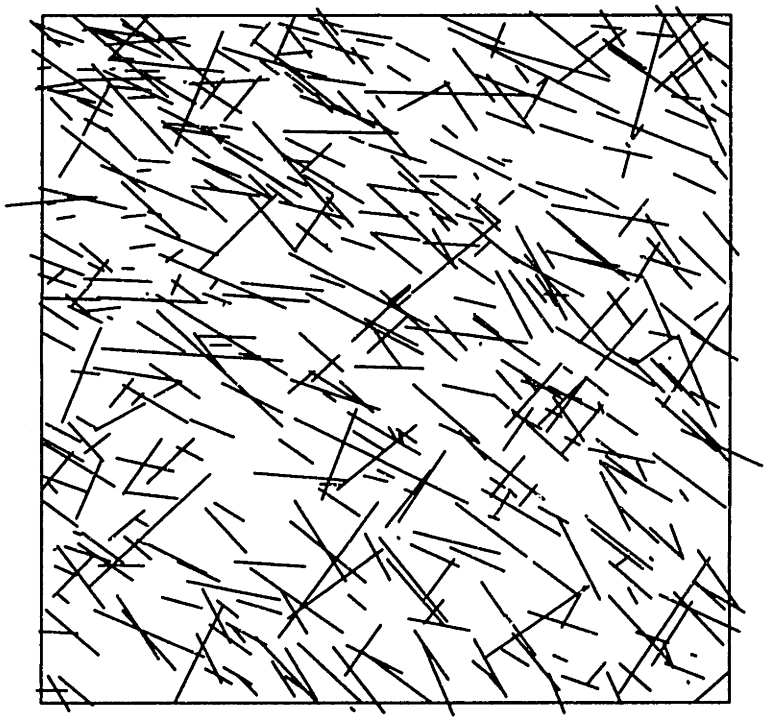
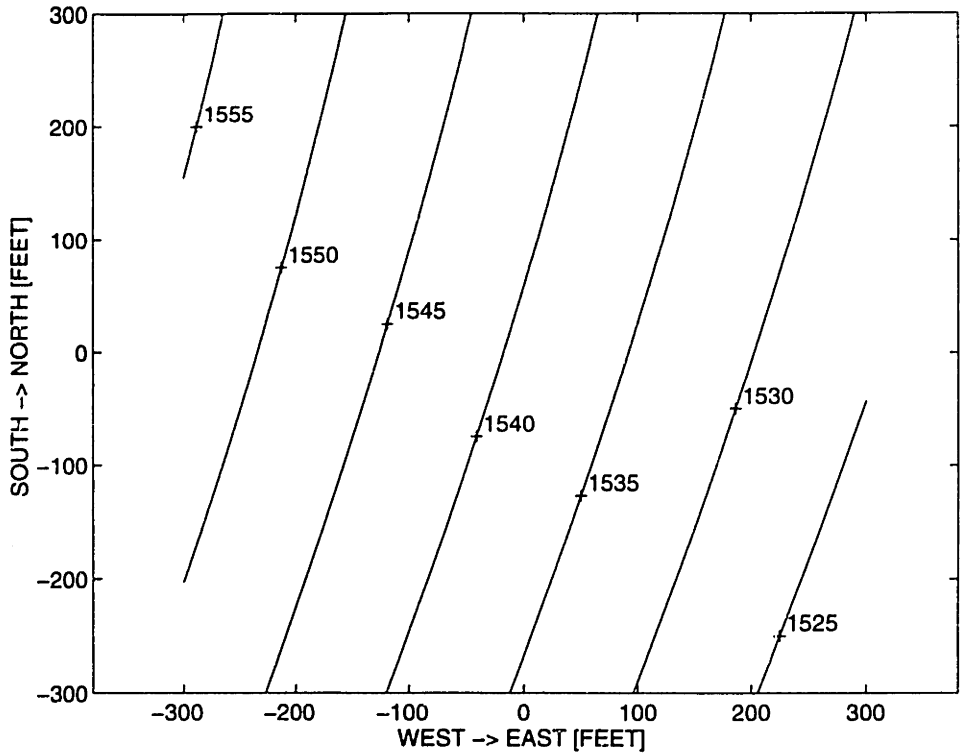


FIGURE 4.5.4 Relationship of the fracture system in the vicinity of well YU4007 to the local shape of the fold: a) fold structure map; b) traces of fold-related fractures on a hypothetical horizontal outcrop plane.

a) FOLD SURFACE



b) HORIZONTAL OUTCROP OF FRACTURE SYSTEM

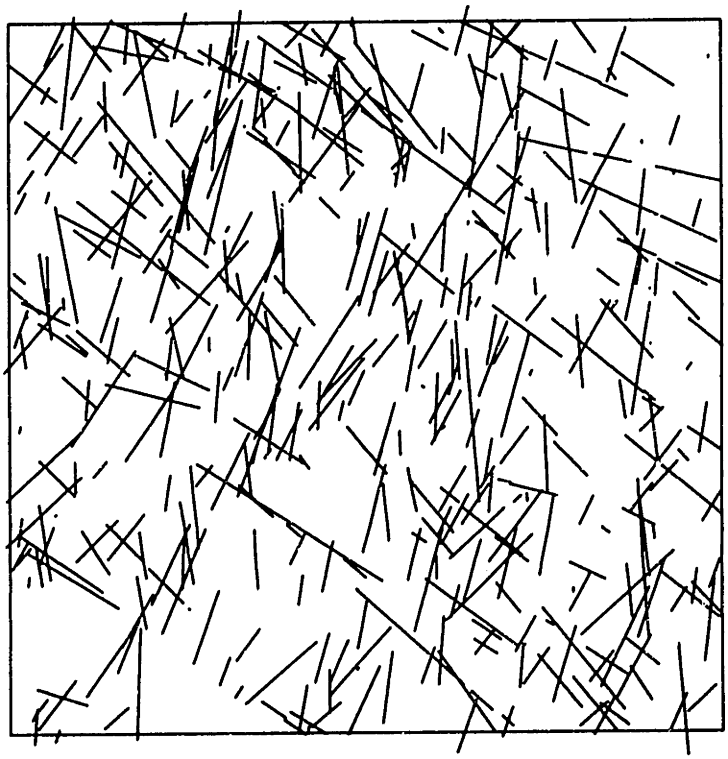
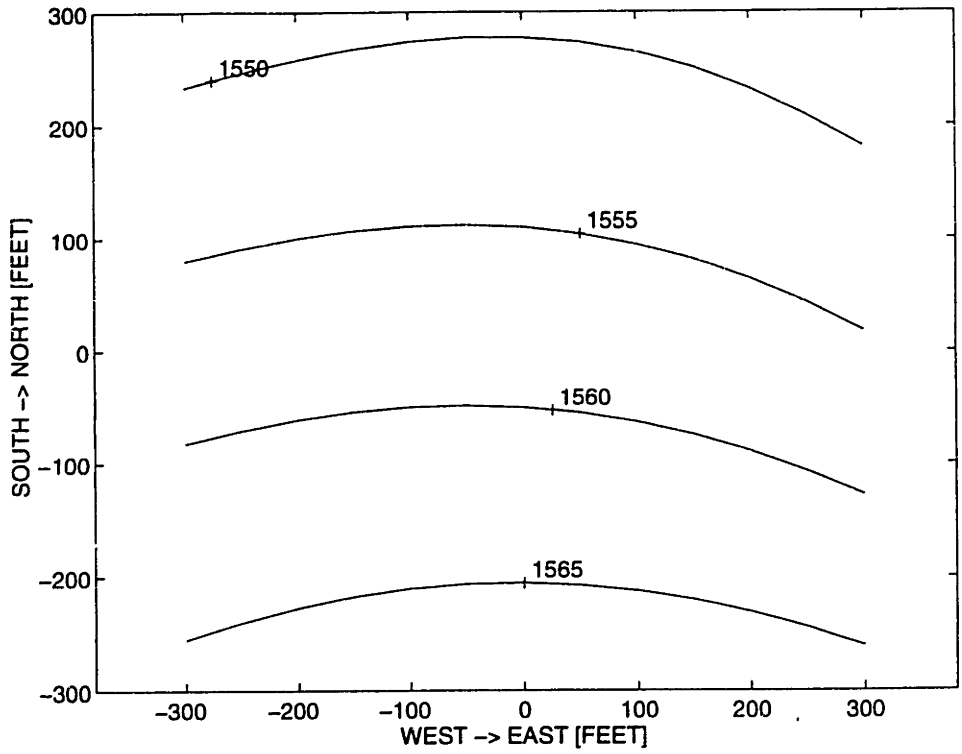


FIGURE 4.5.5 Relationship of the fracture system in the vicinity of well YU4903 to the local shape of the fold: a) fold structure map; b) traces of fold-related fractures on a hypothetical horizontal outcrop plane.

a) FOLD SURFACE



b) HORIZONTAL OUTCROP OF FRACTURE SYSTEM

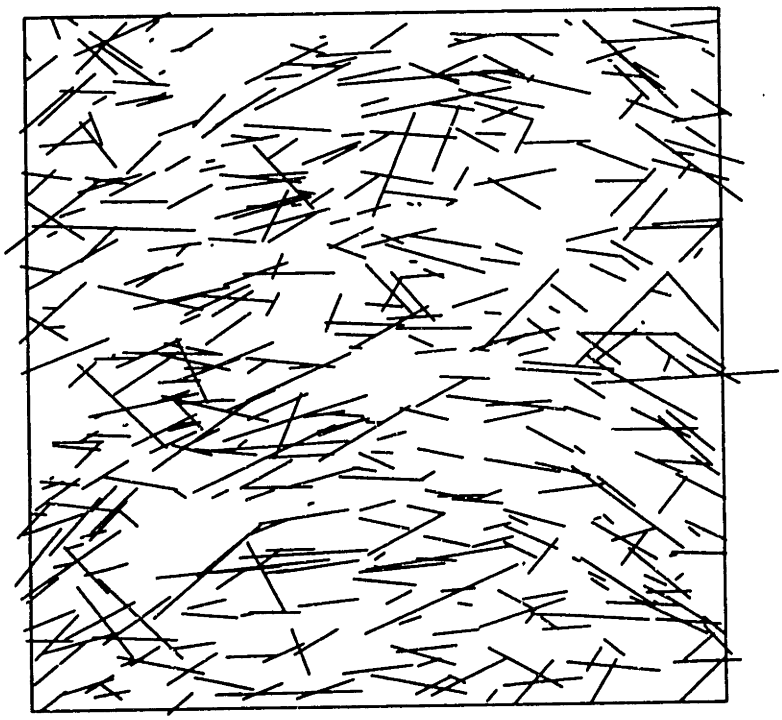


FIGURE 4.5.6 Relationship of the fracture system in the vicinity of well YU5127 to the local shape of the fold: a) fold structure map; b) traces of fold-related fractures on a hypothetical horizontal outcrop plane.

Figures 4.5.8 - 4.5.10 show rosette diagrams of fracture strikes in simulated boreholes at wells YU4007, YU4903, and YU5127. The strikes of numerically generated fracture well intersections match well the strikes of fractures, identified on the log profiles of the wells in Tract 49. **Figure 4.5.11** illustrates the normalized dip distribution of numerically generated fractures compared to the normalized dip distribution of significant fractures identified in cores in Tract 49. The figure shows three examples of fracture dip distribution in simulations of borehole intersections with wells YU4007, YU4903, and YU5127 (the three simulations correspond to the shaded rows in Table 4.5.6).

Table 4.5.7 shows results from simulations in which, after generating the population of fractures, they were also marked as a function of the porosity of the surrounding rock. As a result, in parts of the modeling volumes around the wells, the simulated fracture intensity is lower due to high porosity of the dolomite matrix which makes the material relatively ductile. The distinction between "low" and "high" porosity has been assumed to be at n=20%. Numerically generated fractures which lie entirely in regions with average porosity $n \geq 20\%$ are discarded with probability $P_f=0.5$. Thus, in the high-porosity dolomite the fracture system includes only relatively large fractures, whereas in dolomite with lower porosity both large and smaller fractures exist. The simulations in Table 4.5.7 were performed for illustration of the capability of the model to reproduce fracture intensity as a function of rock properties. Since the exact effect of the porosity of San Andres dolomite on the fracture intensity in the formation has not been studied yet, more precise simulations cannot be done at this stage of the Yates case study.

As Table 4.5.7 shows, due to high porosity, the total fracture number N of fractures decreases within the modeling volumes surrounding wells YU4007, YU4903, and YU5127, hence the fracture intensity P_{32} also decreases. The fracture spacing in the simulated boreholes is larger when porosity is accounted for in cases when the well cuts across some regions of high porosity (see data for wells YU4903 and YU5127). In the case when the well does not cut across high-porosity regions (YU4007) there is no change in the spacing of fractures in simulated boreholes before and after the porosity effect has been accounted for. However, the fracture number and the overall fracture intensity P_{32} in the entire modeling volume are smaller if the porosity is considered.

The results from the modeling of fracture intensity as a function of porosity once again confirms the importance of inferring model parameters from the entire geologic environment of a fracture system, rather than assuming geometric characteristics of the 3D system based on intersections by boreholes (or surface outcrops). **Figure 4.5.12** shows rosette diagrams of strikes of the numerically generated fractures intersected by simulated boreholes at wells YU4903 and YU5127 (a) before, and (b) after porosity effect has been considered.

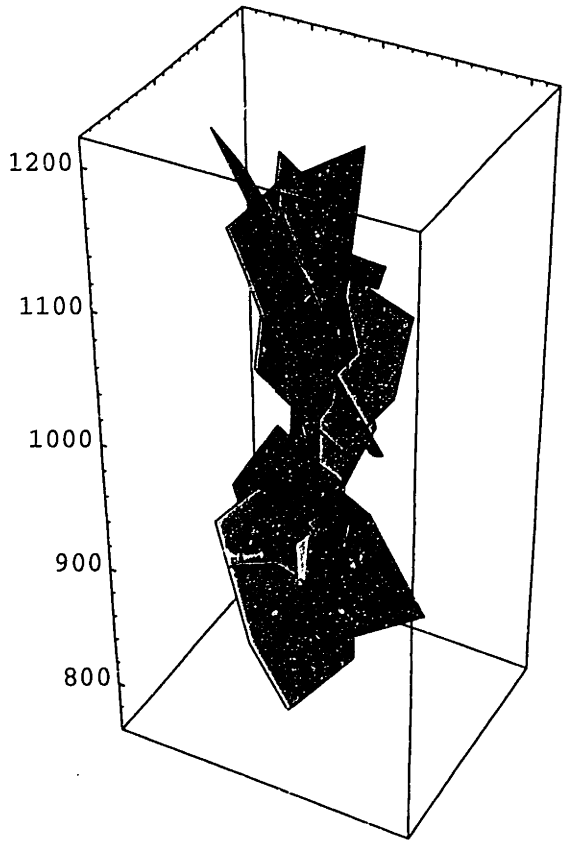


FIGURE 4.5.7 Numerically generated fractures intersected by a simulated borehole at well YU4007. The average size of the intersected fractures is larger than the average size of all fractures.

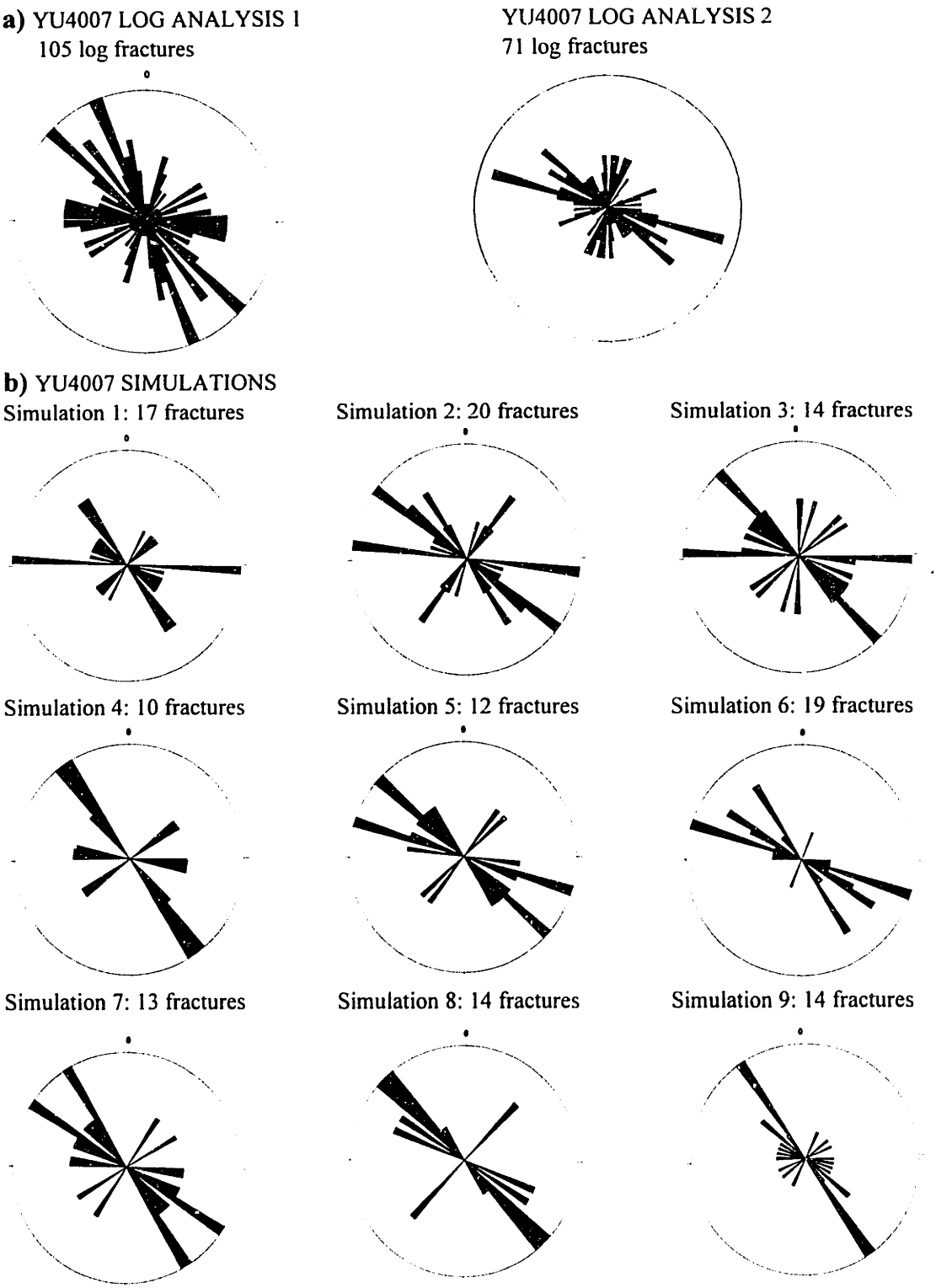
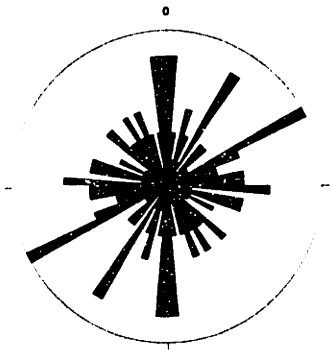


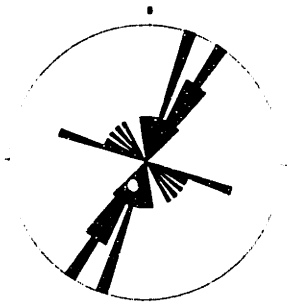
FIGURE 4.5.8 Rosette diagrams of fractures strikes at well YU4007 in Tract 49: a) strikes of log fractures (from two log analyses); b) strikes of numerically generated fractures (nine simulations).

a) YU4903 LOG ANALYSIS
83 log fractures

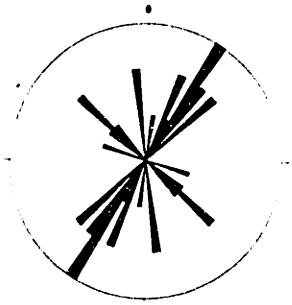


b) YU4903 SIMULATIONS

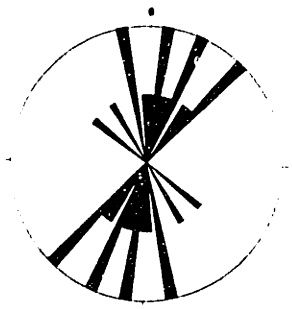
Simulation 1: 24 fractures



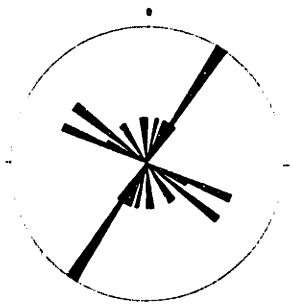
Simulation 2: 18 fractures



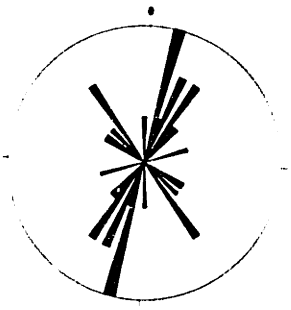
Simulation 3: 16 fractures



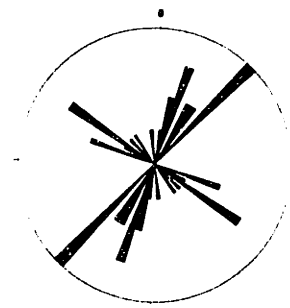
Simulation 4: 16 fractures



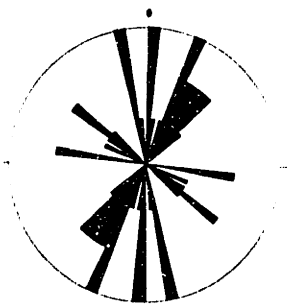
Simulation 5: 17 fractures



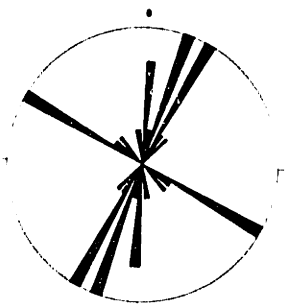
Simulation 6: 23 fractures



Simulation 7: 29 fractures



Simulation 8: 24 fractures



Simulation 9: 15 fractures

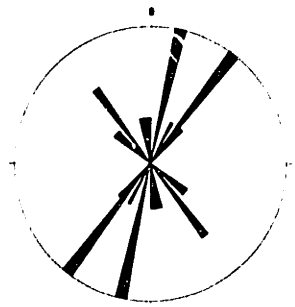


FIGURE 4.5.9 Rosette diagrams of fractures strikes at well YU4903 in Tract 49: a) strikes of log fractures; b) strikes of numerically generated fractures (nine simulations).

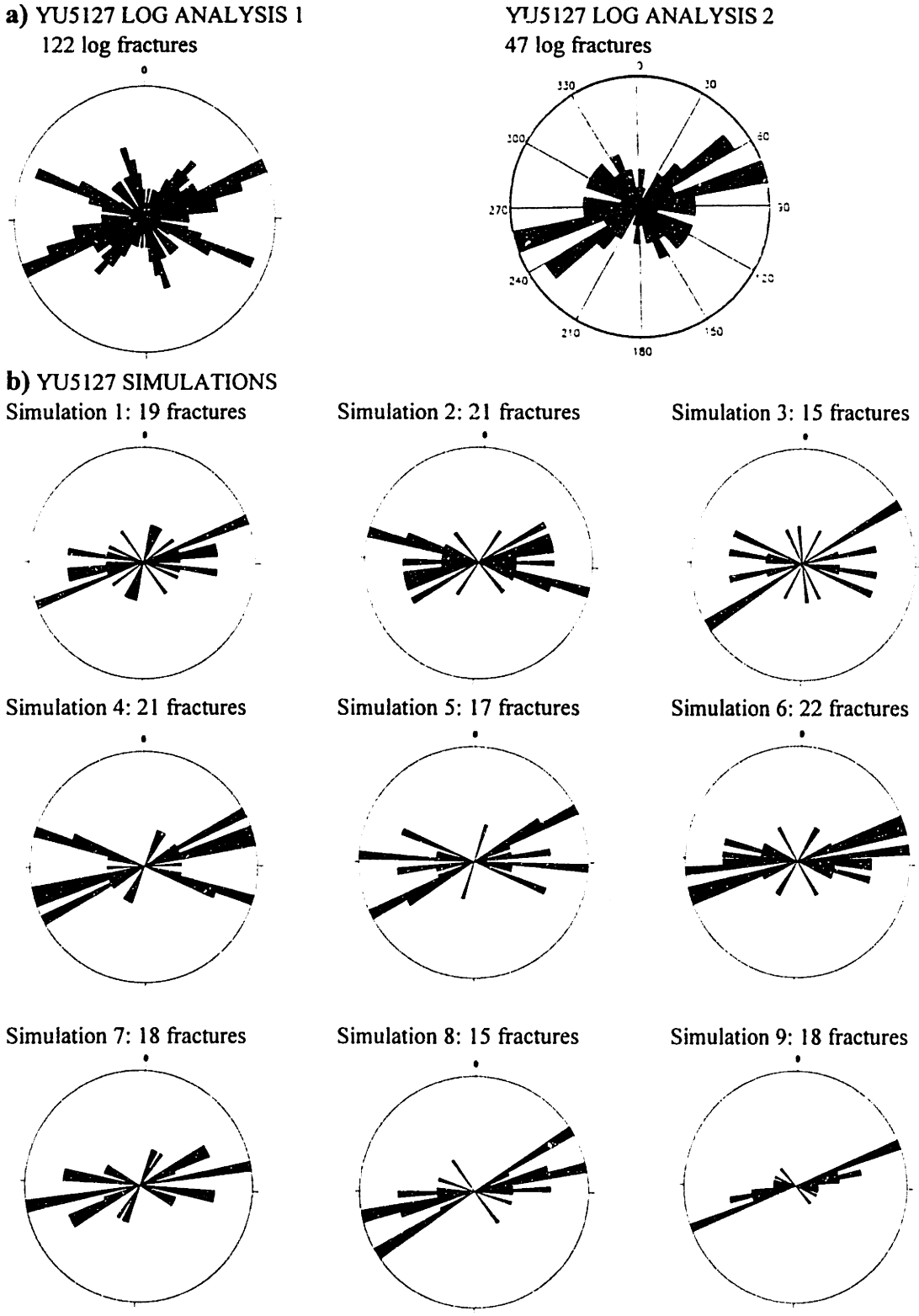
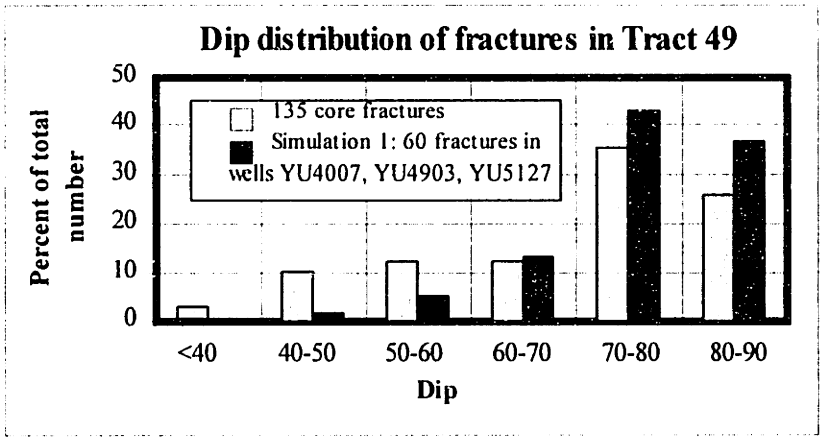
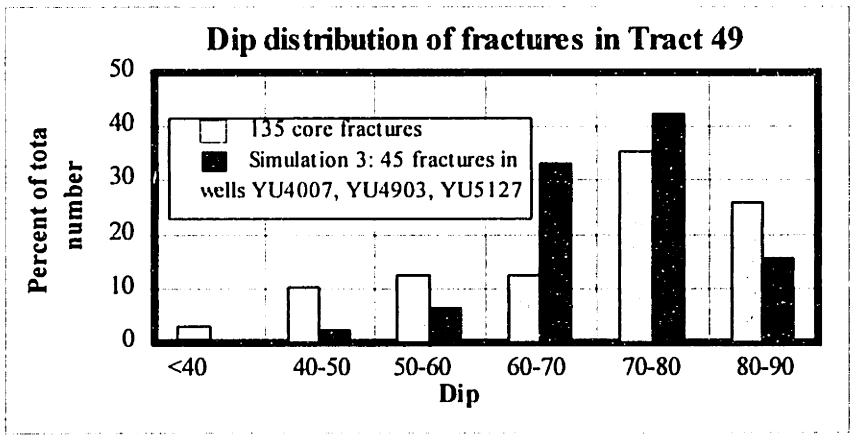


FIGURE 4.5.10 Rosette diagrams of fractures strikes at well YU5127 in Tract 49: a) strikes of log fractures (from two log analyses); b) strikes of numerically generated fractures (nine simulations).

SIMULATION 1



SIMULATION 3



SIMULATION 8

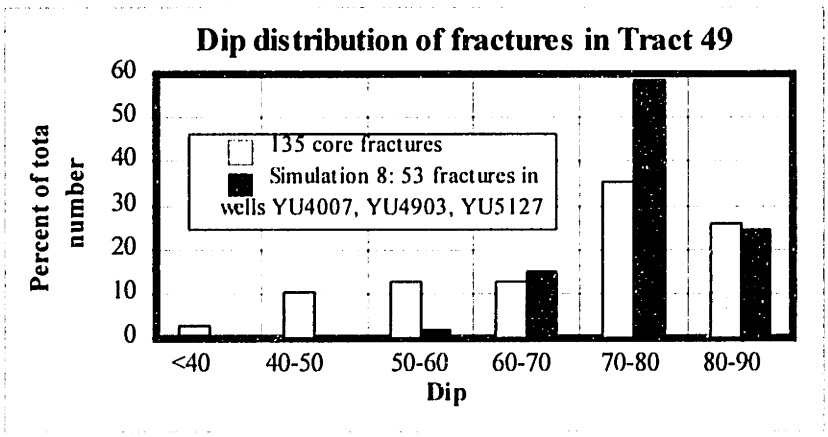


FIGURE 4.5.11 Comparison of fracture dip distribution from simulations to dip distribution of significant fractures in cores in Tract 49. The simulation number corresponds to the line number in Table 4.5.6 on page 326.

WELL	N	P_{32} [ft^{-1}]	N_b	s_{ave} [ft]
YU4007				
• **Simulation 1**				
Before porosity marking	46887	0.623321	12	12.24
After porosity marking	43405	0.587105	12	12.24
• **Simulation 2**				
Before porosity marking	46581	0.618255	10	6.58
After porosity marking	43062	0.580815	10	6.58
YU4903				
• **Simulation 1**				
Before porosity marking	52823	0.626901	17	11.80
After porosity marking	37678	0.458482	15	13.48
• **Simulation 2**				
Before porosity marking	50785	0.610058	19	10.28
After porosity marking	36071	0.444844	17	11.52
YU5127				
• **Simulation 1**				
Before porosity marking	49080	0.592736	18	11.18
After porosity marking	43330	0.533109	17	11.34
• **Simulation 2**				
Before porosity marking	48201	0.577111	17	11.90
After porosity marking	42682	0.518546	13	14.03

TABLE 4.5.7 Results from numerical simulations of fracture intensity as a function of porosity in Tract 49. After generation of the fracture system, P_{32} in porous dolomite (n>20%) is reduced to half of P_{32} in dense dolomite. N: total number of fractures generated in the modeling volume; P_{32}: fracture intensity (averaged for the entire modeling volume); N_b: number of fractures intersected by a simulated borehole; s_{ave}: average spacing between fractures in the simulated borehole.

4.5.2 Numerical simulation of the fracture system in Tract 17

Inference of model parameters

Similar to the inference procedure for Tract 49, fracture intensity P_{32} in Tract 17 is determined from the mean spacing of significant fractures intersected by cores. **Table 4.5.8** summarizes the spacing of significant fractures, identified in cores available from eight wells located inside or in the vicinity of Tract 17. Significant

fractures are determined from core analysis according to the method described in Section 4.5.1 for Tract 49 (see Table 4.5.2). However, while in Tract 49 the distinction between significant and insignificant fractures is relatively clear, in Tract 17 there are numerous fractures that cannot be ignored even if they are not of the type illustrated in Table 4.5.2. For example, there are numerous vertical TOPN fractures which are relatively small (having 1 ft or 2 ft intersections with the core). Also, there are numerous TOPN or POPN fractures of large aperture that cut wall to wall only one cored foot. In Table 4.5.8 such fractures are referred to as "smaller significant fractures".

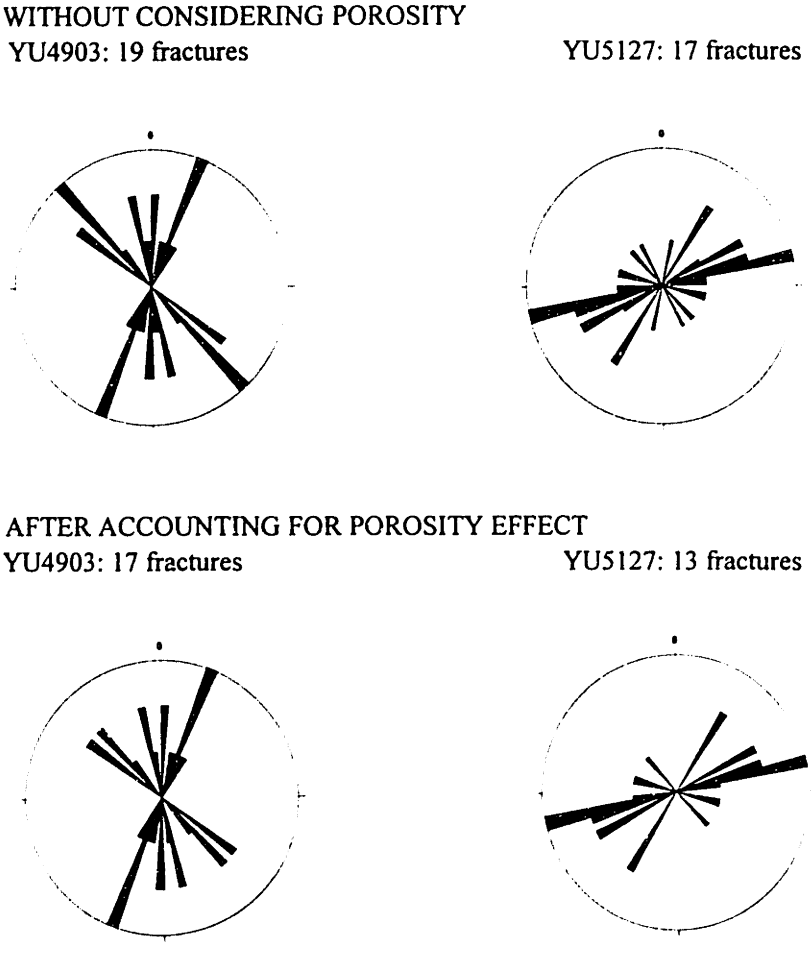


FIGURE 4.5.12 Rosette diagrams of strikes of numerically generated fractures intersected by simulated boreholes at wells YU4903 and YU5127: a) before, and b) after considering the effect of high porosity to decrease fracture intensity. After considering the effect of porosity, the number of fractures intersected by the borehole is smaller, but the predominant strike is the same.

WELL	n_{ave} [%]	N	s_{ave} [ft]	COMMENT
YU1776	13.2	13	7.50	SD=6.87 ft. Numerous smaller fractures. Alternating zones of low and high porosity. Spacing of all (not only the largest) significant fractures: s_{ave}=5.5-6.5 ft. Vertical fractures (from top to bottom): 4 ft in n_{ave}=7.9%; 44 ft in zones with n_{ave}=7.9%, 10.9% and 19.5%; 2 ft in n_{ave}=19.5%; 8 ft and 19 ft in n_{ave} =20.7%.
YU17C9	6.9	4	11.33	Spacing between all significant fractures: s_{ave}=7 ft.
YU2530	N/A	2	7	Only 15 ft of core recovered.
YU2437	19.4	8	4.5	SD=2.07 ft. Zones with very high porosity ϕ_{ave}=20-30%. Vertical fractures: 13 ft in n_{ave}=17.82%; 7 ft in n_{ave} =21.28%.
YU2416	21.62	6	13.8	SD=14.48 ft. Numerous smaller fractures. No fractures in zone with very high porosity n_{ave}=29.28%. Some thin layers of low porosity n_{ave}=7.87%. Spacing s_{ave}= 4.5 ft in n_{ave}=17.05%. No fractures below elevation 1098 ft (in 87 ft of core).
YU2433	19.4	36	6.4	SD=5.36 ft. Longest continuous core: 318 ft. Mostly high porosity n>20%. Spacing increasing with depth (from top to bottom): s_{ave}=3.25 ft in n_{ave}=7.81% (27 ft top zone); s_{ave}=3 ft in n_{ave}=21%; thin zone (4 ft) with n_{ave}=5.3%; s_{ave}=3.43 ft in n_{ave}=22.5%; s_{ave}=5 ft in n_{ave}=11.6%; s_{ave}=8.2 ft in n_{ave}=22.5%; s_{ave}=9 ft in n_{ave}=20.9%. Vertical fractures: 7 ft and 2 ft in n_{ave}= 7.81%; 7 ft start in n_{ave}=21%, terminate at n_{ave}=5.3%; 4 ft in n_{ave}=22.1%; 4 ft and 2 ft in n_{ave}=22.5%; 4, 1, 1, 1, 3, 2, 2, 5, 12 ft in n_{ave}= 20.9% (bottom zone).
YU2509	20.9	12	17.09	SD=9.43 ft. Numerous smaller significant fractures. Spacing including smaller significant fractures s_{ave}= 7.48 ft. In upper layers spacing s_{ave}=4.77 ft in n_{ave}= 5.07-14.7%; in lower layers s_{ave}=8.82 ft in n_{ave}=20.8%. Almost all vertical fractures: 3, 5, 11, 4, 15, 1, 2, 3, 4, 1, 2, 4. 6, 1, 2, 1, 2, 2 ft (top to bottom).
YU14D4	4.9	11	7.18	SD=7.36 ft. Well is located in the far west side of field.

TABLE 4.5.8 Spacing of significant fractures, identified in cores in Tract 17. n_{ave}: average core porosity; N: number of large significant fractures identified at the given well; s_{ave}: average spacing of large significant fractures; SD: standard deviation of spacing between large significant fractures (given only where N>5).

As Table 4.5.8 shows, the average spacing of intersections of significant fractures with the vertical wells in Tract 17 varies between 4.5 ft and 11.3 ft (1.4-3.4 m) where the rock porosity is n_{ave}<20%. The spacing increases where the rock

porosity is n_{ave}>20% (see data for wells YU2416 and YU2509). When the smaller significant fractures are also considered, the average spacing is between 4.5 ft and 7.5 ft (1.4-2.3 m). There may be a tendency of increasing spacing with depth, suggested by the longest available continuous core (318 ft; 97 m) at well YU2433. Forty vertical fracture intersections in Tract 17 cores have heights from 1 ft to 44 ft (0.3-13 m), with all but five having heights less than 10 ft (3 m).

The dip distribution of significant fractures, intersected by cores in Tract 49, is shown in **Figure 4.5.13**. Compared to Tract 49, there are more vertical or nearly vertical fractures in Tract 17. Dip distribution similar to that shown in Figure 4.5.13 has also been determined from the log analysis of the only horizontal well in the study area: well YU17D5 in Tract 17 (see Figure 4.3.12). According to Tinker & Mruk (1995), "the quality of the data in this well and the confidence in the fracture picks are very high". The average spacing of 671 fractures intersections in the 1550 ft (472 m) long horizontal well is s_{ave}= 2.31 ft (0.7 m). Since the fractures in the Yates reservoir rocks are nearly vertical, they are much more easily intersected by the horizontal well than by the vertical wells.

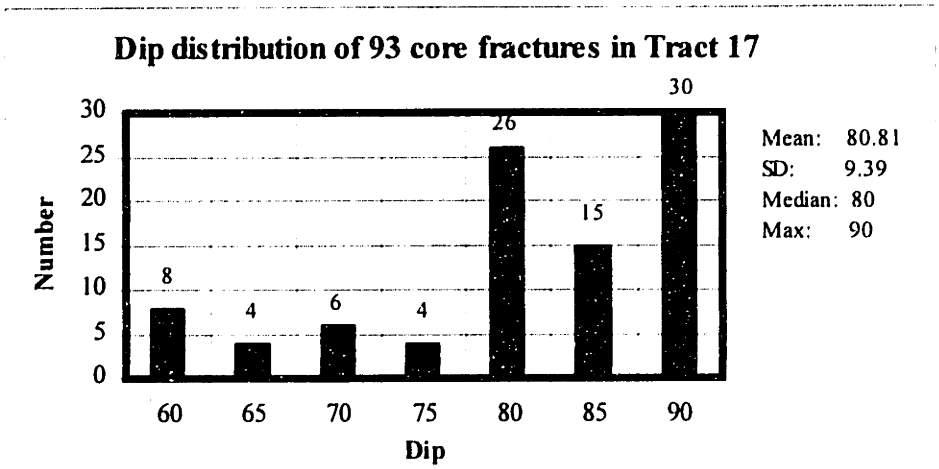


FIGURE 4.5.13 Dip distribution of 93 fractures, identified as "significant" on cores from eight wells in the Tract 17 area.

In Tract 17 the relationship of the fractures to the field anticlinal structure is more complex than that in Tract 49 (Figure 4.3.15). Tract 17 is on the peri-anticline of the reservoir dome: a zone of transition from uncompactible grainstones on the east side toward compacted mudstones to the west. In Tract 17, the northwest striking regional fracture set is sub-parallel to the major hinge of the peri-anticline and to a set of drape folds / faults. The northeast striking regional set is sub-parallel to the minor hinge of the peri-anticline (which defines the so called cross-curvature of the fold) and to a second set of drape folds / faults.

Numerical algorithms

In Tract 17, the algorithm, defined by Equation 4-6 in Section 4.5.1 to read porosity data from the Yates field Stratamodel, and to mark polygons as a function of the porosity of the surrounding rock, is used in the numerical generation of the fracture system. In Tract 17, there is an additional influence of the rock lithology on the fracture intensity: "shales" are considered to be more ductile (hence less fractured) than the brittle dolomite between them. An algorithm, implemented in a module of program GEOFRAC, reads from the Stratamodel not only porosity values, but also gamma ray (GR) data (indicator of shale content), and calculates the average porosity and the average GR of the host rock surrounding a given polygon. Then a polygon is retained as a fracture with probability P_f as a function of the average porosity and the average GR of the surrounding rock:

$$P_f = P_f\left(\frac{1}{N}\sum_{i=1}^{N} n_i, \frac{1}{N}\sum_{i=1}^{N} GR_i\right) \qquad (4\text{-}7)$$

where N is the number of geocells from the reservoir Stratamodel intersected by the polygon, n_i is the porosity and GR_i is the gamma ray response in the i-th intersected cell.

Numerical simulations

Based on the inference procedure, presented above, the following parameters of the 3D fracture system model are selected for the simulations in Tract 17:

- Mean orientations of two fracture sets (inferred from regional geology): vertical planes, striking N50°W and N40°E.
- PDF of fracture plane orientations: Fisher (k=20) which produces close clustering of plane pole orientations around the mean pole (as explained on page 315-316).
- Modeling volumes:
  - ◊ Vertical logged wells: YU1711 and YU1755 inside Tract 17, and YU2511 in the vicinity of Tract 17. The modeling volume for a vertical well is defined by the following boundaries: a horizontal datum surface at 700 ft above sea level; four vertical planes (two striking south-north and two striking east-west), each at a horizontal distance of 300 ft (91 m) from the vertical well; a quadratic upper surface, parallel to the San Andres top and located 100 ft (30 m) above it. The coefficients of the quadratic function are calculated through a

polynomial fit with MATLAB to the elevations of the San Andres top at the wells in the Tract 17 area.

◊ Horizontal logged well: YU17D5 in Tract 17 (north-south trend). The modeling volume for the horizontal well is defined by the following boundaries: a horizontal datum surface at 800 ft above sea level (about 200 ft, 61 m, below the elevation of the well); two vertical planes, striking south-north, at a horizontal distance of 200 ft (61 m) from the well; two vertical planes, striking east-west, at a horizontal distance of 800 ft (244 m) from the midpoint of the well (i.e. 50 ft, 15 m, from the two ends of the well); a quadratic upper surface, parallel to the San Andres top and located 100 ft (30 m) below it (i.e. at least 200 ft, 61 m, above the elevation of the well). The coefficients of the quadratic function are calculated through a polynomial fit with MATLAB to the elevations of the San Andres top.

- Fracture intensity (inferred from field data, since the stress and strain magnitudes, and the mechanical properties of the San Andres dolomite, are not known yet): total $P_{32}=0.8$ [ft^2/ft^3], established through simulations to produce expected spacing in vertical boreholes E[s]=7 ft (2.1 m). The intensities of two fracture sets contribute to the total P_{32}: a major set with assumed intensity $P_{32,1}=0.6$ [ft^{-1}], and a minor set with intensity $P_{32,2}=0.2$ [ft^{-1}]. The N50°W regional set is major in the vicinity of well YU1711 which is located on the crest of the major northwest striking anticlinal axis (Figure 4.3.15a). The N40°E regional set is major in the vicinity of the other three wells: YU17D5, YU1755, and YU2511, which are located in more peripheral areas of the peri-anticline.

- Fracture size distribution (approximate procedure, since field sampling of fracture sizes is not available): similar to Tract 49, the mean of the equivalent radius is assumed $E[R'_c]=30$ ft (9 m; related to the thickness of mechanical units in the San Andres formation). Marking in terms of relative size is assumed of the type $A'_i < E[A']$, where $E[A']=\pi(30)^2=2827$ [ft^2] is the expected area of fractures. This type of marking (see Section 3.2) eliminates the polygons with extremely large sizes, and produces fracture areas that have a standard deviation $\sigma_{A'} = 1.26E[A']$ and median $M[A'] = 0.6E[A']$. 65% of the produced fractures have areas smaller than the average area.

- Effect of porosity (approximate procedure since the exact effect of high porosity on fracture intensity has not been studied yet): fracture intensity in areas with porosity n>20% is assumed to be half of the fracture intensity where n<20%, i.e. $P_{32}|n>20\%=50\%P_{32}=0.5(0.8)=0.4$ [ft^{-1}]. A fracture is discarded with probability $P_f=0.5$ if the average porosity of the dolomite matrix around the fracture is $n_{ave}>20\%$. This method results in discarding fractures with relatively small sizes, since they are the most likely to be

located entirely in an area of high porosity. Thus in high porosity dolomite only relatively large fractures are retained which results in larger spacing of fractures intersected by the wells.

- Effect of shale (approximate procedure since the exact effect of high shale content on fracture intensity has not been studied yet): fracture intensity in areas with high shale content is assumed to be half of the fracture intensity in clean dolomite, i.e. $P_{32,\,shale}=50\%P_{32}=0.5(0.8)=0.4$ [ft^{-1}]. Rock with high shale content ($\geq$10%) is defined to be the one where the porosity $n_{ave}<20\%$, and the gamma ray response $GR_{ave}>40$ API. Thus a fracture is discarded with probability $P_f=0.5$ if the average porosity of the dolomite matrix around the fracture is $n_{ave}<20\%$ and the average gamma ray response is $GR_{ave}>40$ API. This methods results in discarding fractures with relatively small sizes, since they are the most likely to be located entirely in the relatively thin shale layers. Thus the clean dolomite in Tract 17 includes numerous small and fewer large fractures, but only the relatively large ones cut across the thin shale layers.
- Relationship to the anticlinal structure (inferred from local geology): in Tract 17, where the slope of field anticline is very gentle, the relationship of fractures to the local slope of the structure is not as pronounced as it is in Tract 49. The shape of the field anticline is approximated by a cubic surface, fit to the shape of the Seven Rivers M horizon in Tract 17 (illustrated in **Figure 4.5.14**). The coefficients of the cubic surface are calculated through a polynomial fit with MATLAB to the elevations of the Seven Rivers M horizon in the wells in Tract 17. The relation of fracture orientation to the reservoir structure is more complex than it is in Tract 49: predominant fracture orientations in Tract 17 are related to the shape of the entire peri-anticline rather than only to the local slope of the structure. This relationship is accounted for by choosing which one of the regional sets (related to the shape of the peri-anticline) is dominant in the vicinity of different wells. Only a small percent of fractures in Tract 17 are related to the local strike of the fold structure at their centers. In the numerical generation, such fractures are rotated to be predominantly parallel to the slope of the cubic surface near wells YU1711, YU1755, and YU2511, and predominantly orthogonal to the slope (i.e. parallel to the structure itself) in the vicinity of the horizontal well YU17D5).

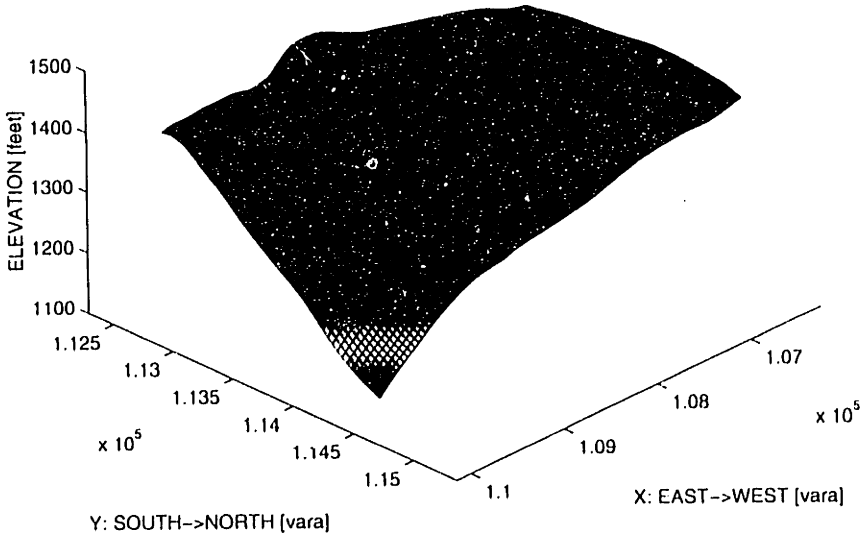
Table 4.5.9 summarizes results from numerical simulations of the fracture system around the three logged wells in Tract 17 (YU1711, YU1755, and YU17D5). These results reflect the last step of the simulations that established the correct fracture intensity P_{32} of the system in Tract 17. The spacing of fracture intersections in the simulated horizontal well YU17D5 (shaded column in Table 4.5.9), which is essentially equal to the known actual spacing of 2.31 ft, confirms that the fracture intensity P_{32} used in the simulations (based on fracture intersections with vertical wells) has been assumed correctly.

#	N	P_{32} [ft^{-1}]	SIMULATED VERTICAL BOREHOLE		
			s_{ave} [ft]	σ_s [ft]	N_b
WELL **YU1711:** input P_{32}=0.8 ft^{-1} E[R'$_e$]=30 ft, 250 ft borehole					
1	**52456**	**0.779530**	**7.51**	**8.51**	33
2	52995	0.786830	7.95	7.59	32
3	53475	0.802514	6.10	5.95	40
4	53865	0.836727	8.06	12.29	30
5	52005	0.820035	7.96	6.03	31
6	53222	0.823912	4.65	4.83	54
WELL **YU1755**: input P_{32}=0.8 ft^{-1}, E[R'$_e$]=30 ft, 300 ft borehole					
1	**49776**	**0.820790**	**7.24**	**10.65**	41
2	50009	0.829751	9.26	7.60	32
3	49632	0.802260	5.99	5.27	50
4	48966	0.811473	5.95	5.27	49
5	49333	0.816772	7.93	6.83	38
6	49349	0.807361	7.18	6.41	42
WELL **YU17D5**: input P_{32}=0.8 ft^{-1}, E[R'$_e$]=30 ft, 1500 ft borehole					
1	**77743**	**0.805955**	**2.23**	**2.49**	672
2	75648	0.830260	2.14	2.42	696
3	76732	0.833689	2.19	2.60	682
4	74975	0.810587	2.19	2.54	686
5	74593	0.780934	2.44	2.77	613
6	74938	0.804766	2.37	2.95	632

TABLE 4.5.9 Results from numerical simulations of the fracture system in the vicinity of wells YU1711, YU1755, and YU17D5 in Tract 17. P_{32}: fracture intensity defined as cumulative fracture area per unit volume; E[R'$_e$]: expected value of fracture equivalent radius; N: total number of fractures generated in a simulation; N_b: fractures intersected by the simulated borehole; s_{ave} and σ_s: average value and standard deviation of fracture spacing in the simulated borehole. Dip distribution of the fractures in the shaded row is shown in Figure 4.5.21. The shaded column represents a test of P_{32} (assumed on the basis of spacing of fracture intersections with vertical wells): compare to the actual spacing of 2.31 ft determined from log analysis of the horizontal well YU17D5.

The horizontal sections in **Figures 4.5.15**a - **4.5.17**a illustrate the shape of the field anticlinal structure in the vicinity of the logged wells inside and near Tract 17. Fracture traces on hypothetical horizontal outcrops in Figures 4.5.15b - 4.5.17b show how the numerically generated fracture system in Tract 17 relates to the field structure in the vicinity of the three logged wells in Tract 17 (YU1711, YU1755, and YU17D5).

a) ACTUAL FOLD SHAPE IN TRACT 17: SEVEN RIVERS M HORIZON



b) REPRESENTATION OF SEVEN RIVERS M BY A CUBIC SURFACE

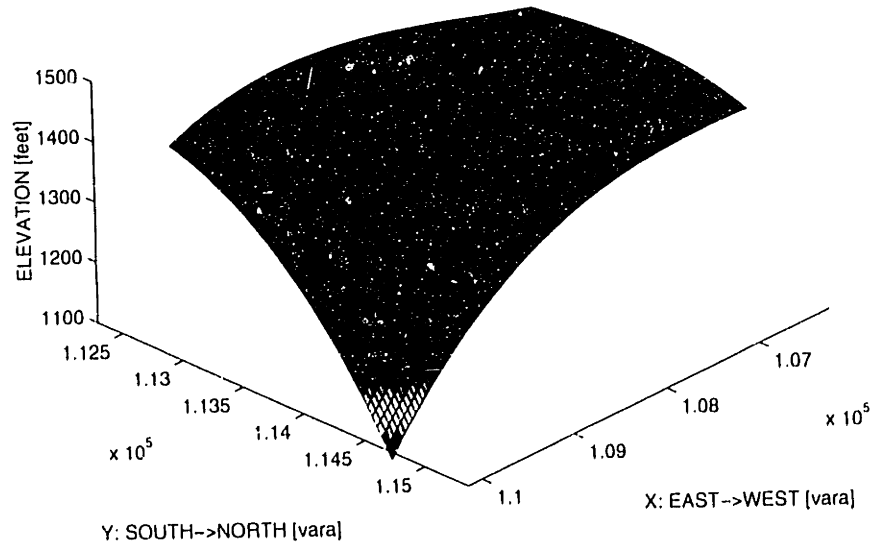
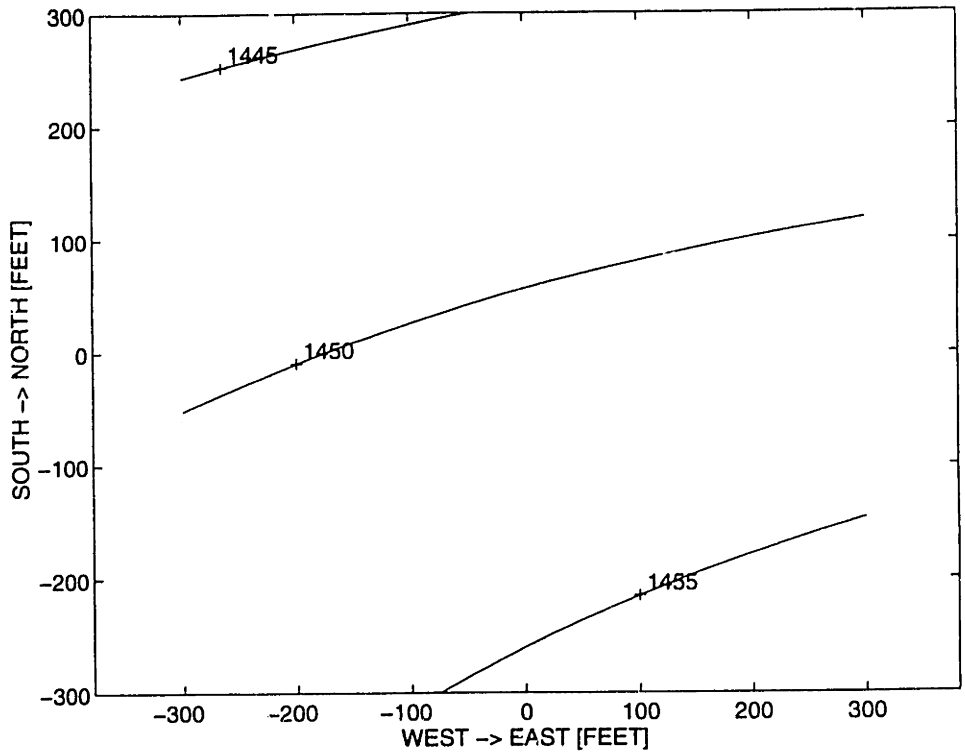


FIGURE 4.5.14 Fit of a cubic surface to represent the fold in Tract 17: a) actual shape, and b) cubic approximation of the Seven Rivers M horizon.

a) FOLD SURFACE



b) HORIZONTAL OUTCROP OF FRACTURE SYSTEM

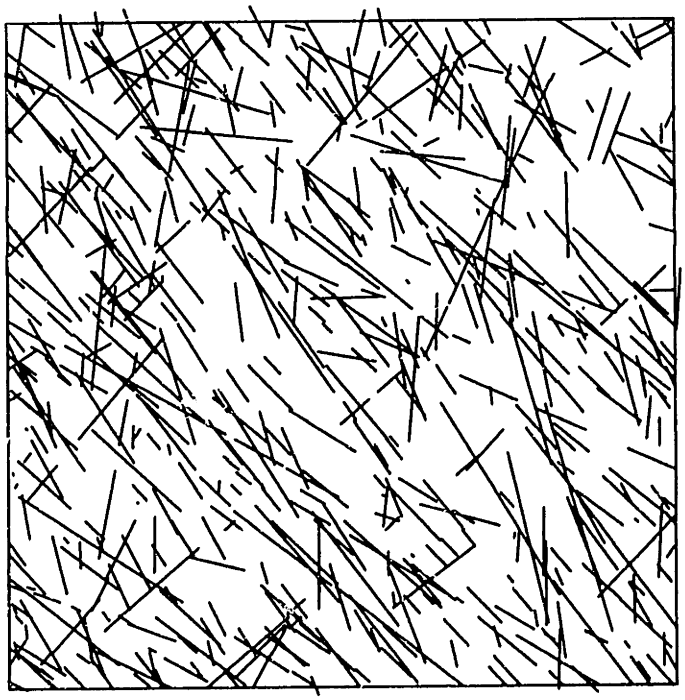
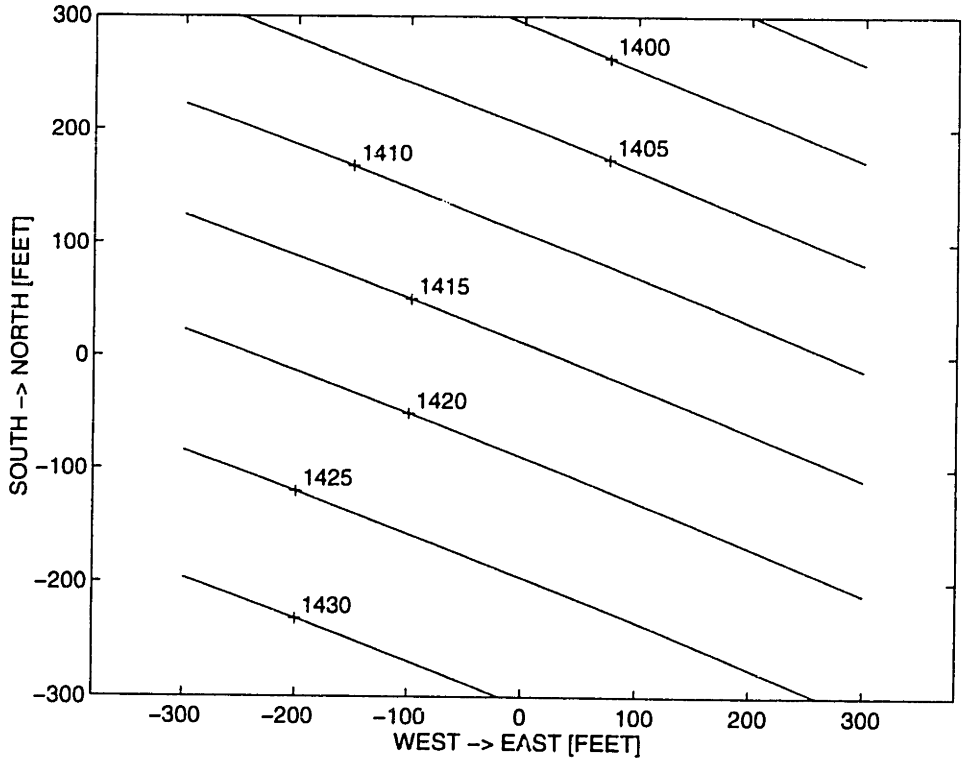


FIGURE 4.5.15 Relationship of the fracture system in the vicinity of well YU1711 to the local shape of the fold: a) fold structure map; b) traces of fold-related fractures on a hypothetical horizontal outcrop plane.

a) FOLD SURFACE



b) HORIZONTAL OUTCROP OF FRACTURE SYSTEM

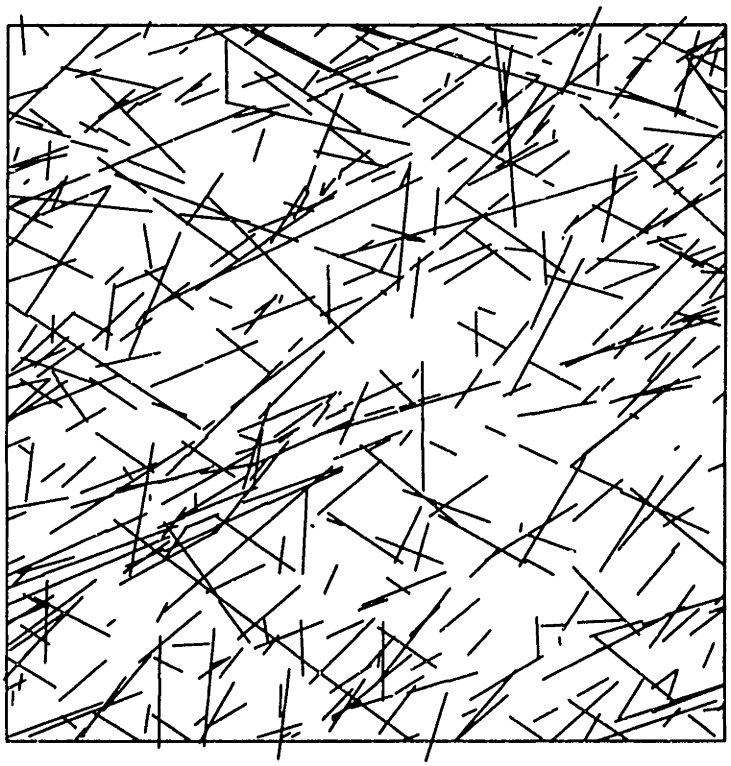


FIGURE 4.5.16 Relationship of the fracture system in the vicinity of well YU1755 to the local shape of the fold: a) fold structure map; b) traces of fold-related fractures on a hypothetical horizontal outcrop plane.

a) FOLD SURFACE

b) HORIZONTAL FRACTURE OUTCROP

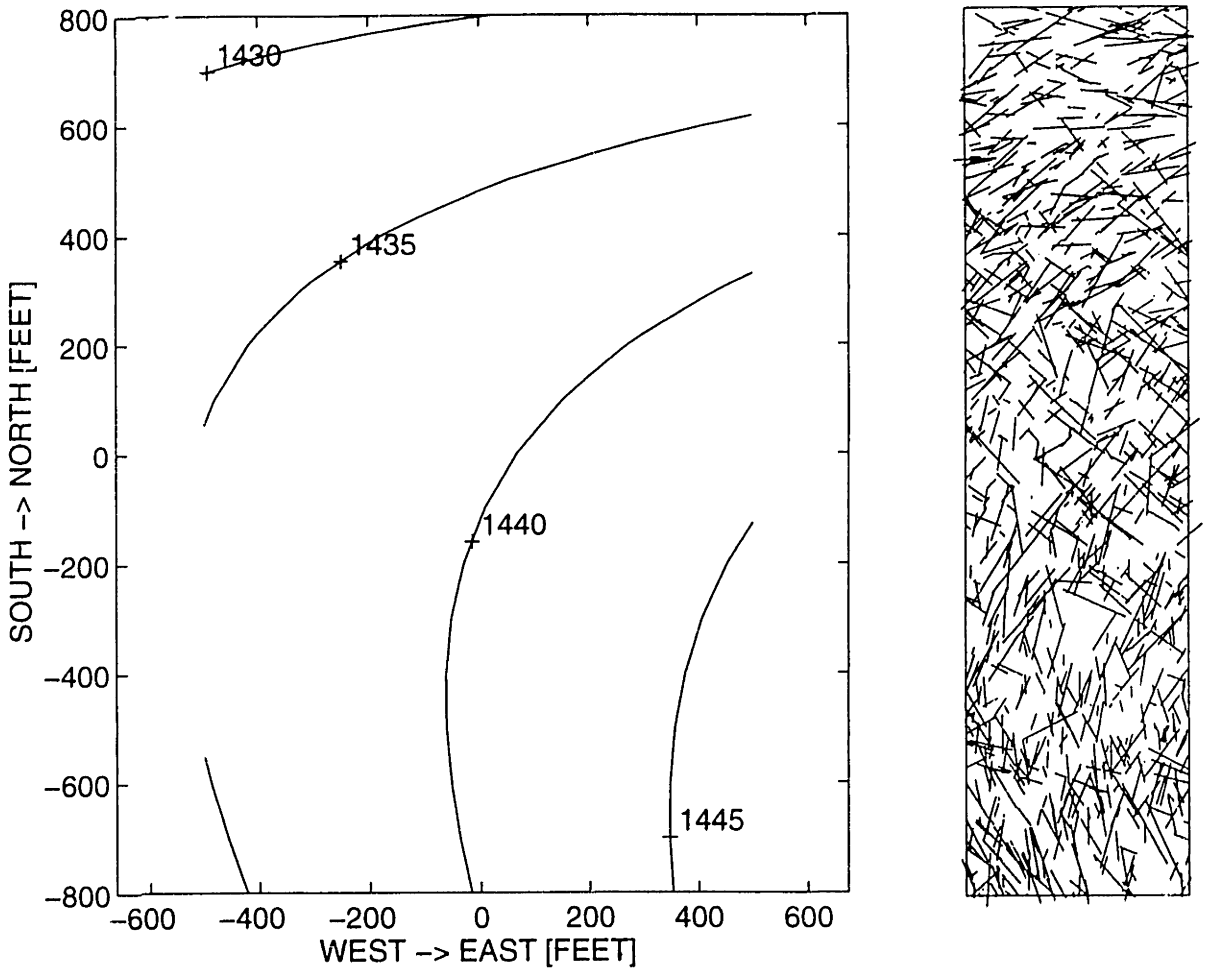


FIGURE 4.5.17 Relationship of the fracture system in the vicinity of well YU17D5 to the local shape of the fold: a) fold structure map; b) traces of fold-related fractures on a hypothetical horizontal outcrop plane.

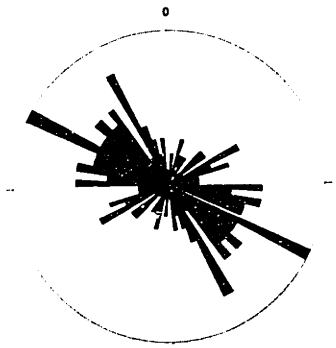
Figures 4.5.18 - 4.5.20 show rosette diagrams of fracture strikes in simulated boreholes at wells YU1711, YU1755, and YU17D5. The strikes of numerically generated fracture well intersections are compared to the strikes of fractures, identified on the log profiles of the wells in Tract 17. **Figure 4.5.21** illustrates the normalized dip distribution of numerically generated fractures compared to the normalized dip distribution of significant fractures identified in cores in Tract 17. The figure shows the fracture dip distribution in a simulation of borehole intersections with wells YU1711, YU1755, and YU17D5 (corresponding to the shaded row in Table 4.5.9).

Table 4.5.10 shows results from simulations in which the effect of high porosity and shale content on fracture intensity was also considered. In parts of the modeling volumes around the wells, the fracture intensity is lower due to either high porosity or high shale content of the dolomite matrix, both of which make the material relatively ductile. The distinction between "low" and "high" porosity has been assumed to be at n=20%. The distinction between "low" and "high" shale content has been assumed to be at 10% (GR=40 API). Numerically generated fractures which lie entirely in regions with either average porosity $n \geq 20\%$ or with average shale content of more than 10% are discarded with probability $P_f=0.5$. Thus, in the high-porosity dolomite the fracture system includes only relatively large fractures, whereas in dolomite with lower porosity both large and smaller fractures exist. Only the relatively large fractures cut across the thin layers with high shale content, whereas the intensity of small fractures in the shales is much lower than it is in the clean dolomite. The simulations in Table 4.5.10 were performed for illustration of the capability of the model to reproduce fracture intensity as a function of rock properties. Since the exact effect of the porosity and shale content of San Andres dolomite on the fracture intensity in the formation has not been studied yet, more precise simulations cannot be done at this stage of the Yates case study.

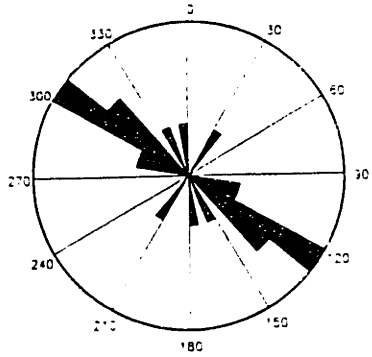
4.6 DISCUSSION OF RESULTS AND RECOMENDATIONS FOR FUTURE WORK ON THE CASE STUDY

The model for the development of the fracture system in the Yates reservoir is based on the depositional model of prograding and aggrading carbonate shoals, published data on the regional tectonic history of the Permian Basin, and other geologic information (including field data) provided by Marathon Oil Company. The model for fracture system development and the proposed algorithms of the 3D hierarchical model best suit the geological information available from the Yates field. Should any additional field data significantly differ from the data available so far, the model for fracture system evolution and the numerical hierarchical model may have to be modified. With this in mind, the following conclusions can be drawn regarding the modeling of the fracture system in the Yates reservoir rocks, presented in the preceding sections of this chapter.

YU1711 LOG ANALYSIS 1
124 log fractures

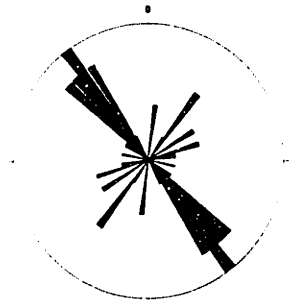


YU1711 LOG ANALYSIS 2
20 log fractures

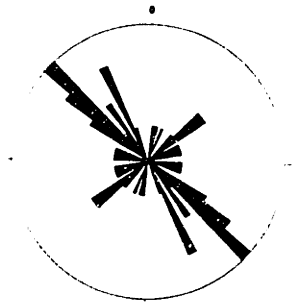


b) YU1711 SIMULATIONS

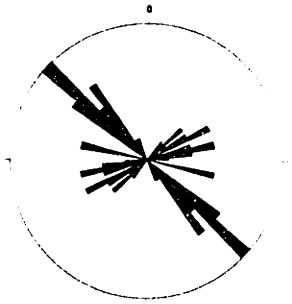
Simulation 1: 33 fractures



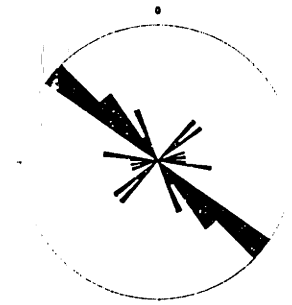
Simulation 2: 32 fractures



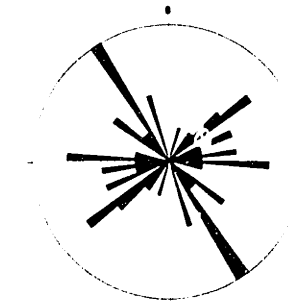
Simulation 3: 40 fractures



Simulation 4: 30 fractures



Simulation 5: 31 fractures



Simulation 6: 54 fractures

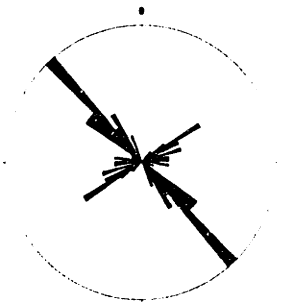
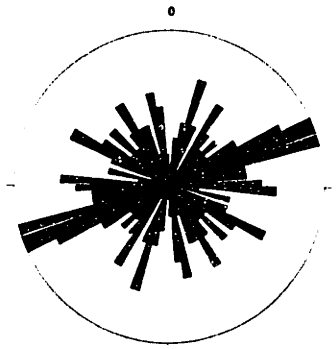


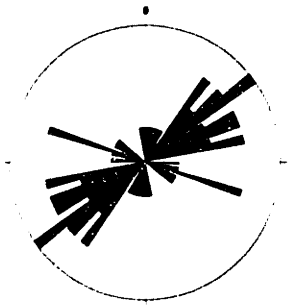
FIGURE 4.5.18 Rosette diagrams of fracture strikes at well YU1711 in Tract 17: a) strikes of log fractures (from two log analyses); b) strikes of numerically generated fractures (six simulations).

a) YU1755 LOG ANALYSIS
163 log fractures

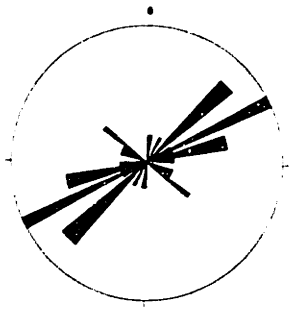


b) YU1755 SIMULATIONS

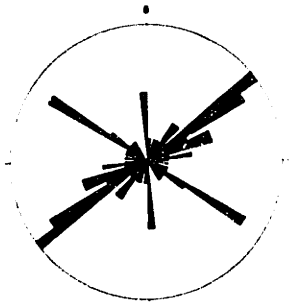
Simulation 1: 41 fractures



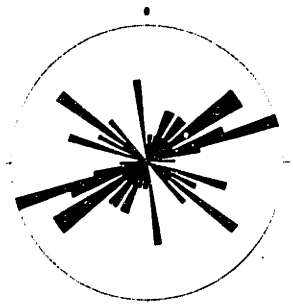
Simulation 2: 32 fractures



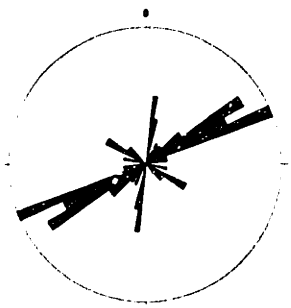
Simulation 3: 50 fractures



Simulation 4: 49 fractures



Simulation 5: 38 fractures



Simulation 6: 42 fractures

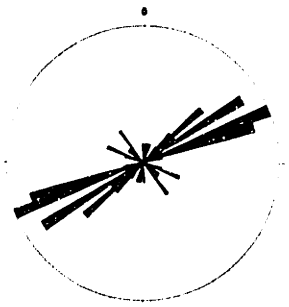
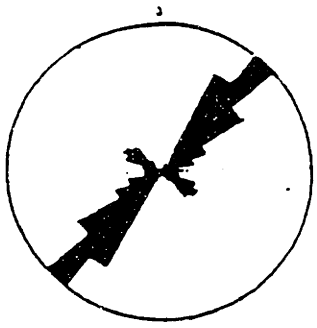


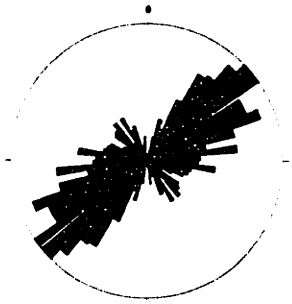
FIGURE 4.5.19 Rosette diagrams of fracture strikes at well YU1755 in Tract 17: a) strikes of log fractures; b) strikes of numerically generated fractures (six simulations).

a) YU17D5 LOG ANALYSIS
671 log fractures

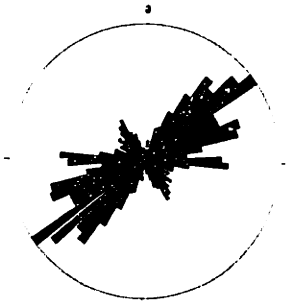


b) YU17D5 SIMULATIONS

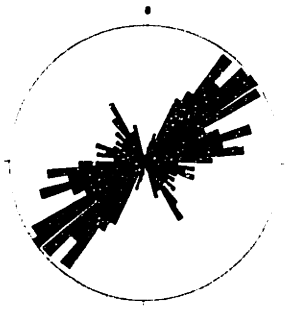
Simulation 1: 672 fractures



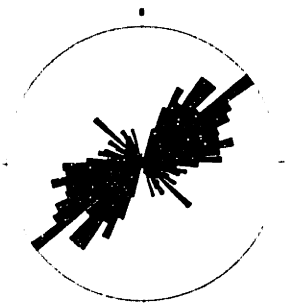
Simulation 2: 696 fractures



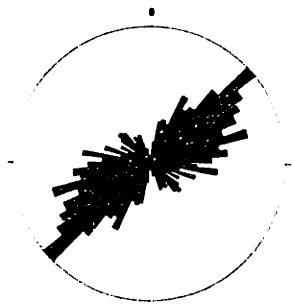
Simulation 3: 682 fractures



Simulation 4: 686 fractures



Simulation 5: 613 fractures



Simulation 6: 632 fractures

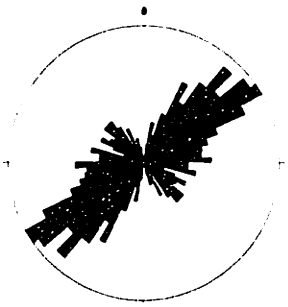


FIGURE 4.5.20 Rosette diagrams of fracture strikes at well YU17D5 in Tract 17: a) strikes of log fractures (from two log analyses); b) strikes of numerically generated fractures (six simulations).

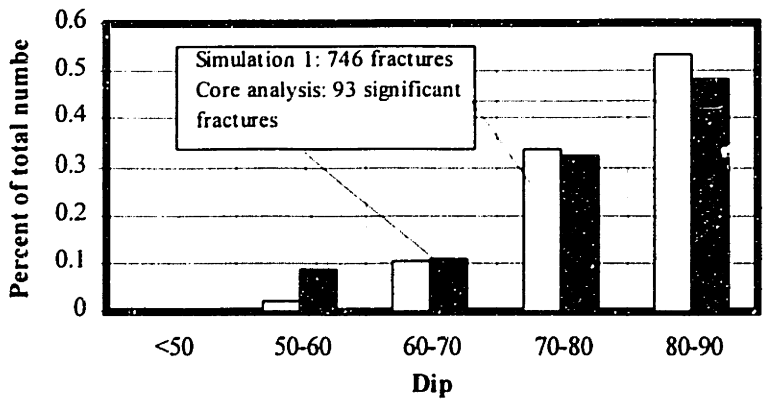


FIGURE 4.5.21 Comparison of fracture dip distribution from simulations to dip distribution of significant fractures in cores in Tract 17. The simulation number corresponds to the line number in Table 4.5.9 on page 346.

WELL	N	P_{32} [ft^{-1}]	N_b	s_{ave} [ft]
YU1711				
Before lithology marking	52736	0.788880	30	8.08
After lithology marking	33101	0.531829	20	11.39
YU1755				
Before lithology marking	51409	0.837606	49	6.02
After lithology marking	34771	0.62375	36	8.26
YU17D5				
Before lithology marking	74663	0.809764	639	2.34
After lithology marking	41600	0.494508	575	2.60

TABLE 4.5.10 Results from numerical simulations of fracture intensity as a function of lithology (including effect of porosity and shale content) in the vicinity of three wells in Tract 17. After generation of the fracture system, P_{32} in dolomite of high porosity (n>20%) is reduced to half of P_{32} in lower-porosity dolomite, and P_{32} in "shales" (shale content>10%) is reduced to half of P_{32} in clean dolomite. N: total number of fractures generated in the modeling volume; P_{32}: fracture intensity in the modeling volume; N_b: number of fractures intersected by a simulated borehole; s_{ave}: average spacing between fracture intersections with the simulated borehole.

Fracture orientations

In the case study, a model of the geologic genesis of the reservoir fracture system in the Yates field was developed. This model accounted for regional and local geologic mechanisms that determined the fracture orientations. The assumed variation of fracture orientations includes subvertical joints, related by strike to regional depositional trends and to the reservoir anticlinal structure. The fracture orientations are confirmed by the intersections in cored and logged wells in the Yates field. Since the geology has been considered in detail, there is great confidence that the assumptions regarding fracture orientations away from the wells are correct. The results from the numerical simulations show that the proposed 3D model has the capability of realistically reproducing various fracture orientation distributions, including relationships to both regional stress directions and local structures (folds).

In the light of a future study, one has to remember that besides the steeply-dipping fractures, flat-lying bedding boundaries most likely also contribute to the reservoir conductivity (possibly even more than the fractures themselves). Bedding-planes have not been modeled in this case study; however, their orientations should not be ignored in a study of flow though the fractured reservoir.

Fracture intensity: P_{32}

In the case study, fracture intensity P_{32}, expressed as cumulative fractured area per unit rock volume, has been assumed on the basis of field data (e.g. fracture core intersections), since field data on geologic stress and strain magnitudes, and experimental data on mechanical properties (strength, deformation moduli) of the reservoir formations, are not available yet. The effect of rock lithology (porosity and shale content) on fracture intensity has also been considered in some simulations. Given the available data, the general method of inference of fracture intensity is correct. Also, the results from the numerical simulations show that the 3D model has the capability of reproducing desired fracture intensity such as P_{32}, including its 3D variation as a function of rock properties. However, in terms of the exact values of P_{32} used in the simulations, there is more to be desired. The following recommendations should be incorporated in a future study of the fracture system intensity in the Yates reservoir.

- The fracture system modeling, presented in this chapter, included inference and numerical simulation of a "total" fracture intensity P_{32}. All fractures which were considered "significant" (i.e. totally open, having a large aperture, etc.), were reproduced. However, in reality it is possible that certain major (possibly the largest) fractures control the flow in the reservoir. If one is interested in the reservoir conductivity, a "conductive" P_{32} should be used in the simulations [Dershowitz 1993]. The field data available to the case study so far was not sufficient for inference of the

conductive P_{32}. However, a limited number of water profiles from Tract 17 and Tract 49 suggest that fluid injection field tests can be a good method to determine the conductive P_{32} which is likely to be lower than the total P_{32}. Also, numerical simulations of flow through the fracture system, generated with the 3D fracture system model, can be a good way to establish which fractures in the Yates field reservoir are conductive and which are not.

- In the Yates field case study, rock lithology is considered to affect the "ductility" or "brittleness" of the reservoir rocks, and hence the fracture intensity P_{32}. This is mostly based on assumptions by geologists and engineers: for example, since fewer fractures are observed in very porous dolomite in the San Andres formation, high porosity is considered to "increase rock ductility". Similarly, dolomites with high shale content on the west side of the field are considered more ductile (hence less fractured). It would be much better if the assumptions about the lithology effect were based on laboratory tests of the mechanical properties of the Yates reservoir rocks. If specimens with different porosity and shale content are tested for tensile and shear strength, and for deformability, more reliable assumptions can be made about which lithological characteristics really affect the mechanical properties of the San Andres dolomite and the other reservoir formations.

- The core database, used in the inference of total P_{32}, has been available to the case study only in electronic form (the actual cores are at MOC PTC in Littleton, Colorado). In a future study, it would be better to identify significant fractures on the actual continuous cores.

- In general, the extensive database available from the Yates field includes very detailed information on depositional sequences, rock lithology, porosity, and other reservoir properties that are traditionally of interest to the oil industry. However, since fractures are subject of a more recent interest, far less information has been collected on the mechanical properties of the reservoir formations. It would be good to develop a procedure for recording field information on the fracture system from a "more mechanical" point of view (for example, taking samples for laboratory experiments, etc.), rather than describing fractures by number or lithology of the mineral filling. The intersections of fractures by cores and logs do not necessarily indicate what fractures exist away from the wells; whereas a depositional model, combined with a database on the mechanical properties of the various formations, can enable one to make a good estimate of the variation of fracture intensity in the reservoir.

- Lastly, only P_{32} of vertical fractures was the subject of this case study. In a future study of flow through the reservoir, intensity of conductive features

along flatly-lying bedding boundaries (fractures or not) should be considered as well.

Fracture intensity: size variation

Fracture size distribution in the case study was assumed on the basis of limited exposures of vertical fractures in cores and a simplified analysis of major mechanical units in the reservoir. This procedure is approximate and does not give a precise estimate for distribution of fracture sizes in the Yates reservoir. Sizes are not considered in more detail since at this stage it is more important to first make a correct estimate of the fracture intensity P_{32}. This is particularly so since there is great uncertainty in the estimation of fracture sizes, especially in a case like the Yates field, where the field fracture data are essentially one-dimensional intersections by wells.

In a future study, fracture size distribution, if desired, can be correlated to distribution of bedding thicknesses in the reservoir formations. In sedimentary rocks, fractures usually propagate to the extent of the thickness of the beds, and align in-plane and out-of-plane across bedding boundaries. Therefore, fracture sizes in the Yates reservoir most likely vary as a function of the variation of the bed thickness, being larger in massive dolomite, and smaller and more closely spaced in thin-bedded dolomite. However, smaller fractures may not be conductive at all; hence a reliable estimate of the conductive P_{32} of the reservoir formations remains of primary importance.

Fracture intensity: zone variation

There is field evidence that suggests that field scale faults bounded carbonate blocks that moved relative to one another during drape folding of the reservoir strata. If such faults exist, it is important to know their exact locations, since they very likely affect the intensity of the fracture system. The "faults" themselves may be actually fault zones which consist of numerous interconnected fractures. The field-scale system of faults/drape folds in the Yates reservoir is currently being studied by Marathon Oil geologists [Wadleigh, pers. comm.].

The field-scale faults were not included in the numerical generation of the fracture system in the Yates reservoir in the case study. However, the 3D model incorporates algorithms for modeling of fracture intensity in zones defined according to distances from primary faults. In a future study, if the locations of the field scale faults/drape folds are known, they can easily be incorporated in the numerical generation of the Yates fracture system.

Conclusion

The first application of the 3D rock fracture system model showed that the model has the capability to realistically represent natural rock fracture systems. The

conceptual model is flexible in terms of implementing specific algorithms for representation of fracture systems in different geologic settings. For example, the algorithms for generating fracture orientations related to folds were developed in the course of the Yates field case study. The case study also showed how limited or ambiguous the fracture information from field sampling could be, even at a site like the Yates field which has been investigated for years. Fracture systems are discrete networks: there is a greater uncertainty about the field sampling and extrapolation between sampling points of their 3D geometry, than there is about the sampling and extrapolation of continuous rock properties such as porosity or lithology. This once again confirms how important it is to base the inference of the parameters of a fracture system model on the underlying geologic mechanisms (rather than on fracture exposures in boreholes or outcrops): a fundamental concept of the 3D geometric-mechanical model that was presented in this thesis.

Chapter 5

CONCLUSIONS

The objective of this thesis was to develop a model for realistic representation of the three-dimensional geometry of fracture systems in natural rocks. In order to do so, the geometry of fracture systems in different natural geologic settings was studied. The geologic study resulted in classification of five major fracture-producing environments, described in Chapter 2: folds, crustal faults, remote tension, thermal contraction, and central structures. The fracture system geometries, created in every one of these major geologic settings, have important characteristics that distinguish them from the geometries of fracture systems in other settings. The numerous types of fracture systems, created by different geologic mechanisms, can be described in terms of regions of existence of hierarchically related fracture sets, fracture intensity expressed as cumulative area per rock volume, and variation of fracture orientations. Therefore, the mechanical classification of typical structural geologic processes, presented in Chapter 2, can be used in the inference procedure of a fracture system model, when assumptions regarding fracture intensity and orientations are made.

The three-dimensional model of rock fracture systems was presented in Chapter 3. The model is geometric-mechanical: it represents the 3D fracture system geometry based on its relationship with the underlying geologic mechanisms, without modeling the mechanics itself. Specifically, the model uses a sequence of three geometric processes to represent relationships between geometry and mechanics: 1) a Poisson plane network that reproduces the orientations of potential fracture planes; 2) a Poisson line tessellation and a polygon marking process which model the ratio of fractured and intact areas on the potential fracture planes; and 3) random polygon translation and rotation which account for the possible local variation of the general stress field (leading to change in fracture orientations and intensity).

The 3D model extends earlier MIT research on fracture system modeling. The model is hierarchical: natural fracture systems are reproduced through superposition of hierarchically related fracture sets, based on the geologic sequence of fracture genesis. Thus the model continues the earlier 2D hierarchical modeling by Lee (1990) and Yu (1992).The model is also stochastic: it uses Poisson plane and line processes, and thus continues the 3D modeling of fracture systems by Veneziano (1978). The application of statistical methods accomplishes two tasks: 1) representation of interconnected fractures in natural rocks as a system, rather than finding the exact location and size of every individual fracture; and 2) consideration of the uncertainty about the 3D geometry of the fracture systems, which are commonly sampled in the field by 1D and 2D methods (boreholes and outcrops). By incorporating geology into geometrical

modeling, the model reduces that uncertainty, and a realistic representation of the 3D geometry of rock fracture systems becomes possible.

Chapter 4 presented a case study in which the 3D model for the first time was applied to an actual fracture system. Modeling of the fracture system in the sedimentary reservoir rocks of the Yates oil field in West Texas included development of a conceptual model of geologic fracture genesis, and numerical simulation of the fracture system. The main conclusion from the case study is that it demonstrated the importance of incorporating the geologic history into fracture system modeling. Even at a site like the Yates field, which has been investigated for many years, the collected field data can give only limited or ambiguous information about the 3D fracture system geometry. Fracture systems in rocks are discrete 3D networks: field sampling by 1D and 2D methods gives little indication on how to infer the true 3D fracture system geometry away from the sampling points. The case study simulations, which were based on the model of fracture genesis in the Yates field, showed a good match by strike and dip, and a reasonable match by intensity, between the numerically generated fracture system and the field fracture data. The results from the case study indicate that the 3D hierarchical geometric-mechanical model has the capability of representing important fracture system characteristics such as fracture orientations related to general stress directions and local geologic structures (folds), and fracture intensity as a function of different material properties.

During the research, presented in the thesis, the 3D fracture system model was implemented in a computer program, GEOFRAC, written in C/C++ and compiled in the MIT UNIX environment Athena. A user manual for the program is included in the Appendix. The main module of GEOFRAC performs generation of fracture sets according to specified fracture orientation PDFs and fracture intensity as it varies in 3D space. Case-specific modules of the program were developed and used during the fracture system simulations for the Yates field.

The major advances of the 3D fracture system model, presented in this thesis, in comparison to previous models, include:

- Geologic classification of the natural geologic settings where rock fracture systems develop, with the intent to find the characteristic geometries created by various geologic mechanisms; in particular, use of the observed relationships between mechanics and geometry in the inference procedure of a geometric-mechanical fracture system model;

- 3D modeling of fracture intensity as cumulative fractured area per unit volume (rather than number of fractures per unit volume), through generation of potential fracture planes and their subdivision into fractured and intact regions;

- Theoretical developments, related to Poisson plane and line processes: fit of Gamma PDF to the distribution of polygons with "good" (i.e. natural fracture-like) shapes, created by a Poisson line tessellation; relation of the modeled fracture intensity to the intensity of the underlying Poisson plane process; relation of the modeled mean fracture size to the intensity of the underlying Poisson line tessellation; Fractal Line Tessellation (modified Poisson line tessellation);

- Development of methods for modeling of 3D variation of fracture intensity, for example, zone marking as a function of polygon distances from specified cluster centers;

- Development of various algorithms for 3D representation of rock fracture systems in a general stress field and in specific geologic settings: e.g. polygon marking, defined by distance to faults, and rotation of polygons in relation to local fold strike and dip.

The 3D model presented in the thesis can be used as a tool for 3D analysis and visualization of rock fracture systems in the geologic environment. Potential applications include: study of flow and contaminant transport through systems of interconnected fractures in rocks; risk assessment and design optimization during underground construction where instability of rock wedges, bounded by fractures, may lead to failure of the walls and roofs of tunnels and mines; study of the potential for secondary recovery of oil and gas from fractured reservoirs (which is the ultimate objective of the Yates field project).

The author considers the fracture system modeling, presented in the thesis, to be a good example of how to model the geologic environment. **Figure 5.1** illustrates this process. When modeling the geologic environment, the phenomena of interest, and the conditions under which they occur, should be identified first. For example, if one is interested in studying flow in rocks, one has to consider that flow occurs along discrete networks of interconnected fractures, and not in a continuous geologic material. Simplifying assumptions should be made only if they do not change the problem itself. For example, fracture systems in rocks, as well as flow through them, are three-dimensional processes and use of 2D models is rarely justified. Also, a model of the geologic environment should account for the uncertainty associated with interpolation of field data among sampling points.

There is always potential for improvement of a model, including a model of fracture systems in rocks. In the future, through geologic and laboratory research, the knowledge about the mechanisms that cause fracturing of rocks in nature will be enhanced. Hence better methods will be designed to better incorporate the geology into the inference procedure of a geometric-mechanical model. With advances in mechanical modeling, it will be possible to develop a geologic fracture system model that represents the true mechanisms of fracture initiation,

propagation, and interaction in 3D space, under stresses that vary in time and space in heterogeneous anisotropic materials, as well as the resulting fracture system geometries. Development of new field testing methods, especially geophysical methods, can enable collection of field data on the true three-dimensional geometry of actual fracture systems, which will reduce the uncertainty that is currently involved in the numerical modeling. Considering all this, one can but conclude that the model, presented in this thesis, although providing good representation of rock fracture systems, reflects only the current status of modeling of the geologic environment.

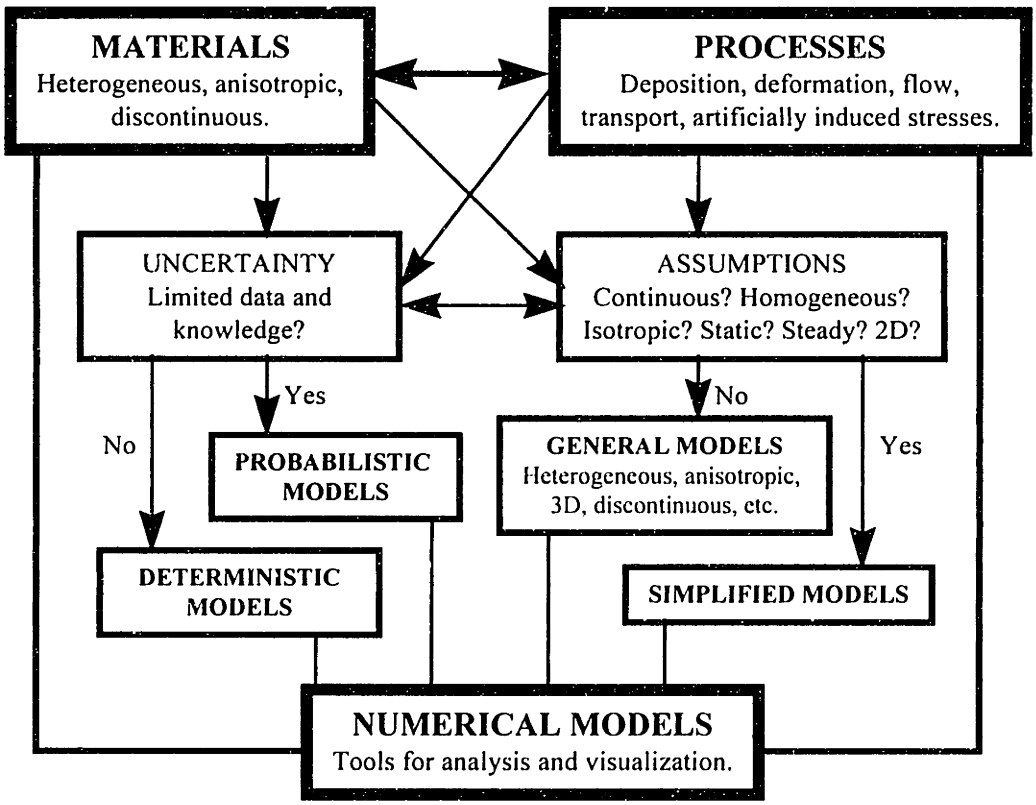


FIGURE 5.1 The process of modeling the geologic environment.

REFERENCES

J.E. Adams & M.L. Rhodes 1960. Dolomitization by Seepage Refluxion. *Am. Assoc. Petrol. Geol. Bull.*, **44**, p. 1912-1920.

E. M. Anderson 1951. *The Dynamics of Faulting and Dike Formation, with Application to Britain.* Oliver and Boyd, Edinburgh.

A. Aydin & J.M. DeGraff 1988. Evolution of Polygonal Fracture Patterns in Lava Flows. *Science*, **239**, p. 471-476.

A. Aydin 1978. Small Faults Formed as Deformation Bands in Sandstone. *Pure & Appl. Geophysics*, **116**, p. 913-930.

A. Aydin & A.M. Johnson 1978. Development of Faults as Zones of Deformation Bands and as Slip Surfaces in Sandstone. *Pure and Appl. Geophysics*, **116**, p. 931-942.

M. P. Billings 1972. *Structural Geology.* Prentice-Hill, Inc., Englewood Cliffs, New Jersey.

A. Bobet 1997. Fracture Coalescence in Rock Materials: Experimental Observations and Numerical Predictions. Sc.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

M. A. Bogdanov 1947. The Intensity of Cleavage as Related to the Thickness of the Bed (Russian text). *Sov. Geol.*, **16**.

S.E. Boyer & D. Elliot 1982. Thrust Systems. *Am. Assoc. Petrol. Geol. Bull.*, **66**, 9, p. 1196-1230.

P.W. Choquette & L.C. Pray 1970. Geologic Nomenclature and Classification of Porosity in Sedimentary Carbonates. *Am. Assoc. Petrol. Geol. Bull.*, **54**, p. 207-250.

E. Cloos 1955. Experimental Analysis of Fracture Patterns. *Geol. Soc. Am. Bull.*, **66**, 3, p. 241-256.

H. Cloos 1922. Tektonik und Magma. *Bd. 1 Abh. D. Preuss. Geol. Land*, **89**.

H. Cloos 1928. Experimente zur inneren Tektonik. *Centralbl. F. Mineralogie*, **B**, 609-671.

D.H. Craig 1990. Yates and Other Guadalupian (Kazanian) Oil Fields, U.S. Permian Basin. *In* J. Brooks (ed.): *Classical Petroleum Provinces.* Geological Society of London Special Publication 50, p. 249-263.

D.H. Craig 1988. Caves and Other Features of Permian Karst in San Andres Dolomite, Yates Field Reservoir, West Texas. *In* P.W. Choquette and N.P. James (eds.): *Paleokarst.* Springer-Verlag, New York, p. 342-363.

D.H. Craig 1963. Regional Geologic Setting of the Yates Field. Technical Memorandum 63-30C, Marathon Oil Company, Littleton, CO.

K.M. Cruikshank, G. Zhao & A.M. Johnson 1991. Duplex Structures Connecting Fault Segments in Entrada Sandstone. *J. Struct. Geol.*, **13**, 10, p. 1185-1196.

P.T. Delaney, D.D. Pollard, J.I. Ziony & E.H. McKee 1986. Field Relations Between Dikes and Joints: Emplacement Processes and Paleostress Analysis. *J. Geophys. Res.*, **91**, B5, p. 4920-4938.

W.S. Dershowitz 1993. Geometric Conceptual Models for Fractured Rock Masses: Implications for Groundwater Flow and Rock Deformation. *In* Ribeiro and Grossman (eds.): *Eurock'93*. Balkema, Rotterdam.

W.S. Dershowitz, G. Lee, J. Geier, S. Hitchcock & P. LaPointe 1994. FRACMAN Interactive Discrete Feature Data Analysis, Geometric Modeling, and Exploration Simulation. User Documentation, Version 2.4, Golder Associates Inc., Seattle, WA.

W.S. Dershowitz 1984. Rock Joint Systems. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

W.S. Dershowitz 1979. Jointed Rock Mass Deformability: a Probabilistic Approach. S.M. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

W.S. Dershowitz & H.H. Herda 1992. Interpretation of Fracture Spacing and Intensity. *In Proceedings*: 32nd US Rock Mechanics Symposium, Santa Fe, New Mexico, p.757-766.

W.S. Dershowitz, P. LaPointe, H.H. Einstein & V. Ivanova 1996, 1997. Fractured Reservoir Discrete Feature Network Technologies. Quarterly progress reports, prepared for contract G4S51728, US Department of Energy, National Oil and Related Programs, BDM-Oklahoma.

P.J. Diggle 1983. *Statistical Analysis of Spatial Point Patterns*. Academic Press, London, UK.

R.J. Dunham 1962. A Classification of Carbonate Rock according to Depositional Texture. *In* W.E. Ham (ed.): *Classification of Carbonate Rocks*. AAPG Memoir 1, p. 108-122.

H.H. Einstein & W.S. Dershowitz 1990. Tensile and Shear Fracturing in Predominantly Compressive Stress Field - a Review. *Eng. Geol.*, **29**, p. 149-172.

T.E. Fitzsimmons, E.E. Wadleigh, B.C. Curran & T.P. Kacir 1997. Flowing Feature Identification for Massive Carbonates: a Real Opportunity. Society of Petroleum Engineers, Middle East Oil Show and Conference, Manama, Bahrain, March 15-18.

J.E.Galley 1958. Oil and Geology in the Permian Basin of Texas and New Mexico. Symposium on Habitat of Oil, AAPG, Tulsa, OK, L.G. Weeks (ed.), p. 395-446.

E.N. Gilbert 1962. Random Subdivision of Space into Crystals. *Annals of Math. Stat.*, **33**, p. 958-972.

W. Hafner 1951. Stress Distributions and Faulting. *Geol. Soc. Am. Bull.*, **62**, p. 373-398.

M.T. Halbouty 1967. *Salt Domes, Gulf Region - U.S. and Mexico*. Gulf Publishing Company, Houston, Texas.

J.F. Harris, G.L. Taylor & J.L. Walper 1960. Relation of Deformational Fractures in Sedimentary Rocks to Regional and Local Structures. *Am. Assoc. Petrol. Geol. Bull.*, **44**, 12, p. 1853.

D.W. Hobbs 1967. The Formation of Tension Joints in Sedimentary Rocks. *Geol. Mag.*, p. 550-556.

D.E. Helgeson & A. Aydin 1991. Characteristics of Joint Propagation across Layer Interfaces in Sedimentary Rocks. *J. Struct. Geol.*, **13**, 8, p. 897-911.

J.M. Hills 1970. Late Paleozoic Structural Directions in Southern Permian Basin, West Texas and Southeastern New Mexico. *Amer. Assoc. Petrol. Geol. Bull.*, **54**, 10, p.1809-1827.

M.K. Hubbert 1951. Mechanical Basis for Certain Familiar Geologic Structures. *Geol. Soc. Am. Bull.*, **62**, p. 1355-1372.

V. Ivanova 1995. Three-Dimensional Stochastic Modeling of Rock Fracture Systems. S.M. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

V. Ivanova, X. Yu, D. Veneziano & H. Einstein 1995. Development of Stochastic Models for Fracture Systems. *In Proceedings*: 35th Symp. Rock Mech., Reno, Nevada. J. Daemen & R. Schultz (eds.), Balkema, Rotterdam, p. 725-730.

M.D. Jackson & D.D. Pollard 1990. Flexure and Faulting of Sedimentary Host Rock during Growth of Igneous Domes, Henry Mountains, Utah. *J. Struct. Geol.*, **12**, p. 185-206.

A.H. Lachenbruch 1962. Mechanics of Thermal Contraction Cracks and Ice-Wedged Polygons in Permafrost. U.S. Geol. Soc. Special Paper 70.

A.H. Lachenbruch 1961. Depth and Spacing of Tension Cracks. *J. Geophys. Res.*, **66**, 12, p. 4273-4289.

F.L. Ladeira & N.J. Price 1980. Relationship between Fracture Spacing and Bed Thickness. *J. Struct. Geol.*, **3**, p. 179-184.

J.-S. Lee 1990. Stochastic and Topological Fracture Geometry Model. Civ. Eng. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

L.-S. Low 1986. Parametric Study of Rock Fracture Geometry. S.M. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

S.J. Martel 1990. Formation of Compound Strike-Slip Fault Zones, Mount Abbot S.J. Quadrangle, California. *J. Struct. Geology*, **12**, 7, p. 869-882.

S.J. Martel & D.D. Pollard 1989. Mechanics of Slip and Fracture Along Small Faults and Simple Strike-Slip Fault Zones in Granitic Rock. *J. Geophys. Res.*, **94**, B7, p. 9417-9428.

S.J. Martel, D.D. Pollard & P. Segall 1988. Development of Simple Strike-Slip Fault Zones, Mount Abbot Quadrangle, Sierra Nevada, California. *Geol. Soc. Amer. Bull.*, **100**, p. 1451-1465.

B.C. McDonald & W.S. Shilts 1973. Interpretation of Faults in Glaciofluvial Sediments. Geological Survey of Canada.

H.E. McKinstry 1953. Shears of the Second Order. *Am. J. Science*, **251**, p. 401-444.

J.L. Meijering 1953. Interface Area, Edge Length and Number of Vertices in Crystal Aggregates with Random Nucleation. Philips research reports, **8**, p. 270-290.

R.H. Merkel 1992. FMS Interpretation on YFU-4007. Marathon Oil Company intercompany correspondence, PTC Littleton, CO.

R.E. Miles 1969. Poisson Flats in Euclidean Space. *Adv. Appl. Prob.*, **1**, p. 211-237.

R.E. Miles 1973. The Various Aggregates of Random Polygons Determined by Random Lines in a Plane. *Adv. in Math.*, **10**, p. 256-290.

J.D. Moody & M.J. Hill 1956. Wrench Fault Tectonics. *Geol. Soc. Am. Bull.*, **67**, p. 1207-1246.

W. Narr & J. Suppe 1991. Joint Spacing in Sedimentary Rock. *J. Struct. Geol.*, **13**, p. 1037-1048.

A.C. Novikova 1947. The Intensity of Cleavage as Related to the Bed Thickness (Russian text), *Sov. Geol.*, **16**.

D.L. Peck & T. Minakami 1968. The Formation of Columnar Joints in the Upper Part of Kilauean Lava Lakes, Hawaii. *Geol. Soc. Amer. Bull.*, **79**, p. 1151-1166.

D.D. Pollard, P. Segall & P.T. Delaney 1982. Formation and Interpretation of Dilatant Echelon Cracks. *Geol. Soc. Amer. Bull.*, **93**, p. 1291-1303.

N.J. Price 1962. Tectonics of the Aberystwyth Grits. *Geol. Mag.*, **99**, p. 542-547.

N.J. Price 1966. *Fault and Joint Development in Brittle and Semi-Brittle Rocks.* Pergamon Press, Oxford, UK.

N.J. Price 1968. A Dynamic Mechanism for the Development of Second Order Faults. *In Proceedings*: Conference on Research in Tectonics. A.J. Baer & D.K. Norris (eds.), Geological Survey of Canada, GSC Paper 68-52.

N.J. Price & Cosgrove 1990. *Analysis of Geological Structures.* Cambridge University Press, Cambridge, UK.

N.J. Price & P. Hancock 1972. Development of Fracture Cleavage. *In Proceedings*: Int'l Geological Congress, Canada, Session 24, Section 3, p. 584-592.

J.G. Ramsay 1967. *Folding and Fracturing of Rocks*. McGraw-Hill Book Company, New York.

J.G. Ramsay & M. I. Huber 1987. *The Techniques of Modern Structural Geology. Volume 2: Folds and Fractures*. Academic Press, Inc. (London) Ltd.

O. Reyes & H.H. Einstein 1991. Failure Mechanisms in Fractured Rocks - a Fracture Coalescence Model. *In Proceedings*: 7th Int'l Congress on Rock Mechanics, W. Wuttke (ed.), Aachen, Germany, p. 333-339.

W. Riedel 1929. Zur Mechanik geologischer Brucherscheinungen. *Zbl. Miner. Geol. Palaeent.*, **B**, 354.

W.W. Sanford 1959. Analytical and Experimental Study of Simple Geologic Structures. *Geol. Soc. Amer. Bull.*, **70**, p. 19-52.

J.F. Sarg and P.J. Lehmann 1986. Lower-Middle Guadalupian Facies and Stratigraphy, San Andres - Grayburg Formations, Permian Basin, Guadalupe Mountains, New Mexico. *In* G.E. Moore and G.L. Wilde (eds.): *Lower and Middle Guadalupian Facies Stratigraphy and Reservoir Geometries, San Andres - Grayburg Formations, Guadalupe Mountains, New Mexico and Texas*. Society of Economic Paleontologists and Mineralogists, Permian Basin Section, Publication No. 86-25, p. 1-36.

R. Sedgewick 1992. *Algorithms in C++*. Addison-Wesley Pub. Co., Reading, MA.

P. Segall 1984. Formation and Growth of Extensional Fracture Sets. *Geol. Soc. Amer. Bull.*, **95**, p. 454-462.

P. Segall & D.D. Pollard 1983a. Joint Formation in Granitic Rock of the Sierra Nevada. *Geol. Soc. Amer. Bull.*, **94**, p. 563-575.

P. Segall & D. D. Pollard 1983b. Nucleation and Growth of Strike Slip Faults in Granite. *J. Geophys. Res.*, **88**, B1, p. 555-568.

P. Segall & D.D. Pollard 1980. Mechanics of Discontinuous Faults. *J. Geophys. Res.*, **85**, B8, p. 4337-4350.

M.M. Singh 1981. Strength of Rock. *In* Y.S. Touloukian, W.R. Judd, and R.F. Roy (eds.): *Physical Properties of Rocks and Minerals*. McGraw-Hill Book Company, New York, NY.

L.U. de Sitter 1956. *Structural Geology*. McGraw - Hill Publishing Company Ltd., London, 552 p.

G.M. Sowers 1973. Theory of Spacing of Extension Fractures. *Eng. Geol. Case Histories*, **9**, p. 27-53.

D.W. Stearns and M. Friedman 1972. Reservoirs in Fractured Rock. *In* R.E. King (ed.): *Stratigraphic Oil and Gas Fields - Classification, Exploration Methods and Case Histories*. AAPG Memoir 16, p.82-106.

D. Stoyan, W.S. Kendall & J. Mecke 1987. *Stochastic Geometry and Its Applications*. Akademie-Verlag Berlin / John Wiley & Sons, UK.

J. Suppe 1985. *Principles of Structural Geology*. Prentice-Hall, Inc. Englewood Cliffs, New Jersey.

J.S. Tchalenko 1968. The Evolution of Kink Bands and the Development of Compression Textures in Sheared Clays. *Tectonophysics*, **6**, p.159-174.

A.L. Thomas & D.D. Pollard 1993. The Geometry of Echelon Fractures in Rock: Implications from Laboratory and Numerical Experiments. *J. Struct. Geol.*, **15**, 3-5, p. 323-334.

S.W. Tinker 1996. Building the 3-D Jigsaw Puzzle: Applications of Sequence Stratigraphy to 3-D Reservoir Characterization, Permian Basin. *Amer. Assoc. Petrol. Geol. Bull.*, **80**, 4, p. 460-485.

S.W. Tinker, J.R. Ehrets & M.D. Brondos 1995. Multiple Karst Events Related to Stratigraphic Cyclicity: San Andres Formation, Yates field, West Texas. *In* D.A. Budd, A.H. Saller, and P.M. Haris (eds.): *Unconformities in Carbonate Strata - Their Recognition and the Significance of Associated Porosity*. AAPG Memoir No. 16, p. 213-237.

S.M. Tinker. and D.H. Mruk 1995. Reservoir Characterization of a Permian Giant: Yates Field, West Texas. *In* E. Stoudt and P.M. Harris (eds.): *Hydrocarbon Reservoir Characterization, Geologic Framework and Flow-Unit Modeling*. SEPM Short Course 34, p.51-128.

M.E.Tucker and V.P. Wright 1990. *Carbonate Sedimentology*. Blackwell Scientific Publications, Oxford, UK.

C.R. Twidale 1973. On the Origin of Sheet Jointing. *Rock Mech.*, **5**, p. 163-187.

G.J.G. Upton & B. Fingleton 1985. *Spatial Data Analysis by Example, Volume I: Point Pattern and Quantitative Data*. John Wiley & Sons, Chichester, UK.

D. Veneziano 1978. Probabilistic Model of Joints in Rocks. Research report, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, MA.

G. Wilson 1946. The Relationship of Slatey Cleavage and Kindred Structures to Tectonics. *Proc. Geol. Assoc.*, **57**, p. 263-302.

H. Wu & D.D. Pollard 1992. Modeling a Fracture Set in a Layered Brittle Material. *Eng. Fract. Mech.*, **42**, 6, p. 1011-1017.

H.-B. Xiao, F.A. Dahlen & J. Suppe 1991. Mechanics of Extensional Wedges. *J. Geophys. Res.*, **96**, B6, p. 10,301-10,318.

X. Yu 1992. Stochastic Modeling of Rock Fracture Geometry. S.M. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

APPENDIX: GEOFRAC USER MANUAL

The Appendix is intended to serve as a user manual for the program GEOFRAC which numerically implements the conceptual 3D model of rock fracture systems, presented in Chapter 3. The Appendix is for use by those involved in the MIT research on the 3D fracture system model. Section A.1 includes a summary of the conceptual model, and essentially repeats the theory of the fracture system model from Chapter 3, without the mathematical details. Other references on the conceptual model are Ivanova (1995) and Ivanova et al. (1995). Section A.2 presents the numerical code GEOFRAC, and includes: (1) an explanation of how objects of the fracture system model (polygon-fractures, outcrops traces, boreholes, fold surfaces, etc.) are represented in 3D by C++ classes (Section A.2.1); (2) the structure of the input and output files for GEOFRAC and its modules; and (3) the C++ source code of the program (Section A.2.3).

A.1 THEORY OF THE 3D FRACTURE SYSTEM MODEL

The conceptual model is a geometric-mechanical model. Simple geometric procedures are used to duplicate the 3D geometry of fracture systems, created by complex mechanical processes, on the basis of inherent relationships between mechanics and geometry.

The model incorporates Poisson plane and line stochastic processes. References on the fundamental theory of Poisson point, line, and plane processes are Veneziano (1978), Diggle (1983), Stoyan et al. (1987), Upton & Fingleton (1885), Miles (1969 and 1973).

Based on field data and geologic history, fractures are grouped into hierarchically related fracture sets. Primary fractures that originate either in intact rock, or without relation to earlier fractures, are combined into independent sets. Fractures, created as secondary, tertiary, etc. in relation to earlier ones with respect to location, orientation, size or other geometric characteristics, are grouped into dependent sets. A fracture system is represented through superposition of hierarchically related fracture sets in 3D space.

A fracture set is generated through a sequence of three stochastic processes (**Figure A.1**; Section A.1.1) that represent relationships between fracture system geometry and mechanics. The primary process (Figure A.1a), a random plane network, models stress field orientation (i.e. potential fracture planes). The secondary process (Figure A.1b) subdivides the planes into intact and fractured areas through a line tessellation and polygon marking. The combined primary and secondary processes model fracture intensity. The tertiary process (Figure A.1c), random polygon translation and rotation, represents the relationship of fracture orientation to local structures (e.g. folds) that modify the general stress field.

Section A.1.1 presents the algorithms for modeling of a fracture set, and Section A.1.2 briefly explains the generation of a fracture system.

A.1.1 Modeling of a fracture set

Geometrically, a fracture set is a collection of fractures with related orientations, sizes, and locations. A fracture set is characterized by two parameters:

- probability density function (PDF) describing the variation of fracture plane orientations;
- intensity of fracturing as it varies in space.

Modeling of stress field orientation: primary stochastic process

Stress field orientation is defined as the orientations of planes of maximum shear and tension that are related to fracture plane orientations. The orientations of the potential fracture planes are represented by the primary stochastic process: a homogeneous, anisotropic, Poisson plane network (Figure A.1a).

Frame of reference and orientations of potential fracture planes

The mean orientation of a fracture set is specified in polar coordinates in terms of an azimuth Θ and a latitude Φ (**Figure A.2**) in the global frame of reference (OXYZ). The axes of the global coordinate system coincide with global directions: usually OX is east, OY is north, and OZ is vertical.

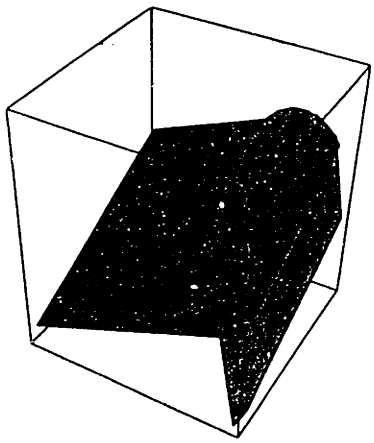
In the frame of reference of a fracture set (Oxyz) a plane is defined as:

$$\mathbf{x}\sin\theta\sin\phi + \mathbf{y}\cos\theta\sin\phi + \mathbf{z}\cos\phi = \mathbf{d} \qquad \text{(A-1)}$$

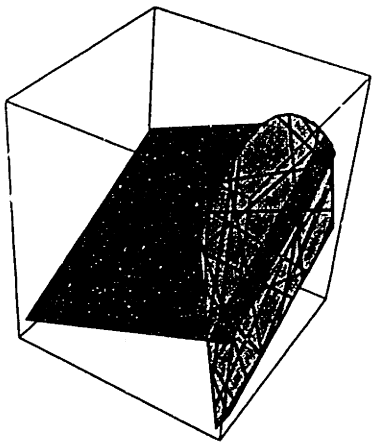
where d is the orthogonal distance from the global origin O to the plane, and θ and ϕ are the azimuth and latitude of the normal vector (pole Oz'), respectively. The orientations of potential fracture planes are randomly generated as pairs of polar coordinates (θ, ϕ) in the frame of reference of the fracture set (Oxyz). The directional pairs (θ, ϕ) follow a spherical PDF, inferred from the measured data. Possible PDFs include uniform or partial uniform, one-parameter or two-parameter Fisher, and Bingham. In some cases all fracture planes may be assumed parallel and assigned the same orientation. The PDFs are reviewed in more detail in Chapter 3, and also by Dershowitz (1979) and Ivanova (1995).

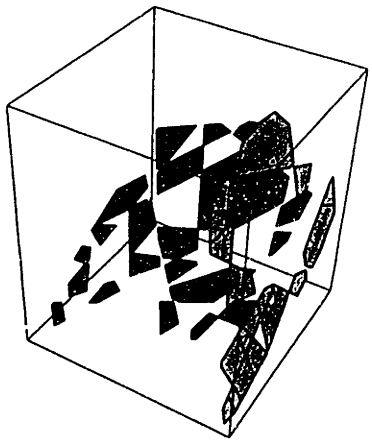
Fractures are represented by convex planar polygons (inset 1 in Figure A.2), randomly oriented and located in 3D space. The pole orientation and the coordinates of the center and the vertices of an individual fracture are determined indirectly when the fracture is generated as a member of a set.

a) PRIMARY PROCESS: POISSON PLANE NETWORK

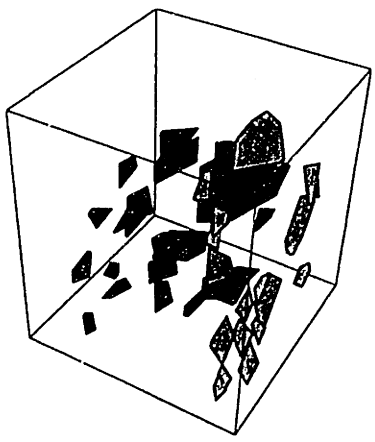


b) SECONDARY PROCESS: POISSON LINE TESSELLATION AND POLYGON MARKING





c) TERTIARY PROCESS: RANDOM POLYGON TRANSLATION / ROTATION



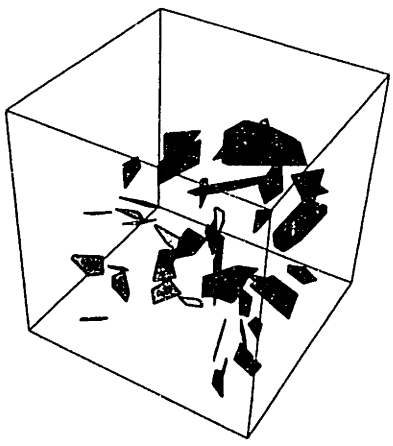


FIGURE A.1 Stochastic processes of the 3D fracture system model (computer generation with program GEOFRAC): a) primary process, Poisson plane network; b) secondary process, subdivision of planes through Poisson line tessellation and polygon marking; c) tertiary process, random rotation and translation of polygons.

b) secondary process, subdivision of planes through Poisson line tessellation and polygon marking; c) tertiary process, random rotation and translation of polygons.

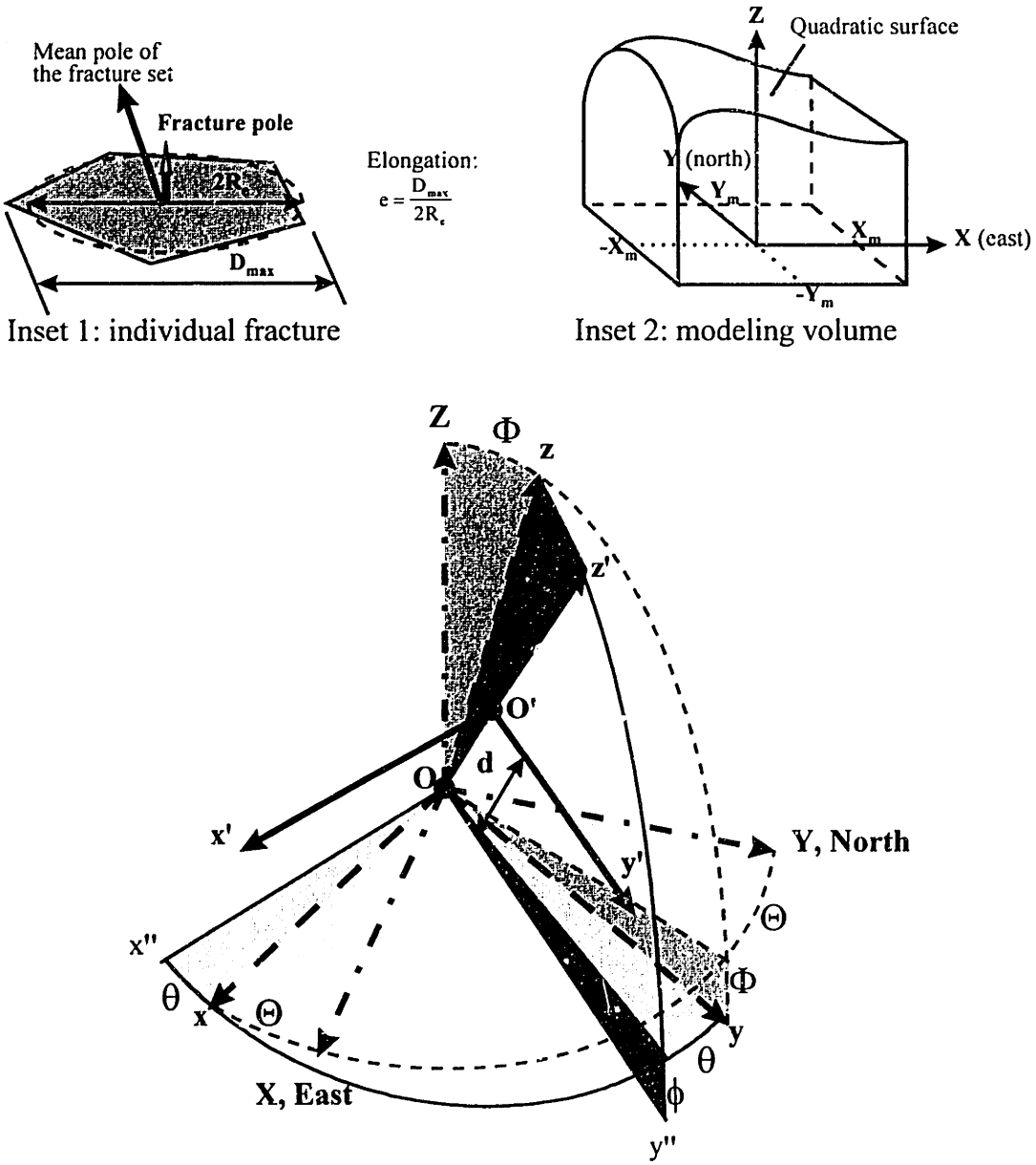


FIGURE A.2 Primary stochastic process: frames of reference of a fracture set. OXYZ: global frame of reference (f.o.r.); Oz: mean pole direction of the fracture set, defined by global azimuth Θ and global latitude Φ; Oxyz: f.o.r. of the fracture set; Oz' : pole (normal vector) of a (potential) fracture plane, defined by local azimuth θ and local latitude ϕ; d = OO' : orthogonal distance of the fracture plane

Modeling volume of a fracture set

The Poisson plane network is generated in a modeling volume that represents the rock mass where the fracture network of interest develops. The modeling volumes of independent sets are defined by the boundaries of geologic structures (such as the topographic surface, bedding planes, fold surfaces, or datum planes) over a certain area. The boundaries of modeling volumes of dependent fracture sets are defined by previously generated fractures. The modeling volume, currently implemented in GEOFRAC (inset 2 in Figure A.2), is enclosed between four vertical planes at $X=X_m$, $X=-X_m$, $Y=Y_m$, $Y=-Y_m$, a horizontal datum plane at $Z=0$, and a quadratic top surface.

Intensity of the Poisson plane network

A homogeneous Poisson plane network of intensity μ corresponds to a Poisson point process in the region:

$$\{(\mathbf{d},\theta,\phi):-\infty < \mathbf{d} < \infty, \mathbf{0} \le \theta \le \pi, \mathbf{0} \le \phi \le \pi\} \tag{A-2}$$

with non-homogeneous intensity function of the type:

$$\mu(\mathbf{d},\theta,\phi) = \mu \mathbf{f}_{\theta,\phi}(\theta,\phi) \tag{A-3}$$

where $f_{\theta,\phi}$ is the joint PDF of θ and ϕ and μ is a positive constant [Miles 1969; Veneziano 1978].

The following algorithm is used for generation of a Poisson plane network of intensity μ in the 3D modeling volume (which usually has irregular shape). Planes are generated in the smallest sphere of radius R, centered at the origin of the global frame of reference, in which the 3D region can be inscribed. The ordered distances from an arbitrary point to the planes of a Poisson network with intensity μ define a Poisson point process of intensity 2μ . The expected number of planes that intersect the sphere is $E[n]=2\mu R$. The i-th random plane is generated so that:

1) the pole orientation (θ_i, ϕ_i) fits to a specified spherical PDF, where $0<\theta_i<2\pi$, and $0<\phi_i<\pi$.
2) the distance d_i (d in Equation A-1) from the global origin O is calculated as:

$$\mathbf{d}_i = -\mathbf{R} + \sum_{m=1}^{i} \delta \mathbf{d}_m \tag{A-4}$$

where $\delta d_m > 0$ are incremental distances, generated according to an exponential distribution with intensity μ. Planes are generated as long as $d_i < R$: thus the number of generated planes is a Poisson number. Some planes produced in this way may not intersect the modeling volume, but only the sphere that inscribes it.

Modeling of fracture intensity: primary and secondary stochastic processes

Fracture intensity is defined as cumulative fractured area per unit rock volume:

$$P_{32} = \frac{\sum_{i=1}^{N} A_{f,i}}{V} \tag{A-5}$$

where N is the total number of fractures and $A_{f,i}$ is the area of an individual fracture inside the volume V [after Dershowitz & Herda 1992]. Fracture intensity is modeled by the combined primary and secondary processes.

A random subdivision of the planes, generated by the primary process, into a fractured region and its complementary region of intact rock constitutes the basis of the secondary process. This subdivision is accomplished by a Poisson line tessellation on every plane and a process of marking the so-created polygons as fractured or intact rock (Figure A.1b). The secondary process produces sets of fractures that have a certain size and shape variation and are arranged in clusters.

Poisson line tessellation

If A is the polygonal intersection between the fracture plane and the modeling volume V, and O'x'y' is the local 2D frame of reference on the plane (**Figure A.3**), a line from the Poisson network is defined as:

$$\mathbf{x'\cos\alpha + y'\sin\alpha = D} \tag{A-6}$$

in terms of an angle α on A, measured counterclockwise from the axis O'x', and a distance D from the origin O' to the line.

A homogeneous Poisson line network with intensity λ corresponds to a Poisson point process in the region:

$$\{(\mathbf{D},\alpha):\mathbf{0} \le \mathbf{D} \le \infty, \mathbf{0} \le \alpha \le \mathbf{2\pi}\} \tag{A-7}$$

with point intensity function defined as:

(A-7)

with point intensity function defined as:

$$\lambda(\mathbf{D},\alpha) = \lambda \mathbf{f}_{\alpha}(\alpha)$$

(A-8)

where λ is a positive constant and $f_\alpha(\alpha)$ is the PDF of α (here assumed uniformly distributed in $[0,2\pi]$) [Miles 1973; Veneziano 1978].

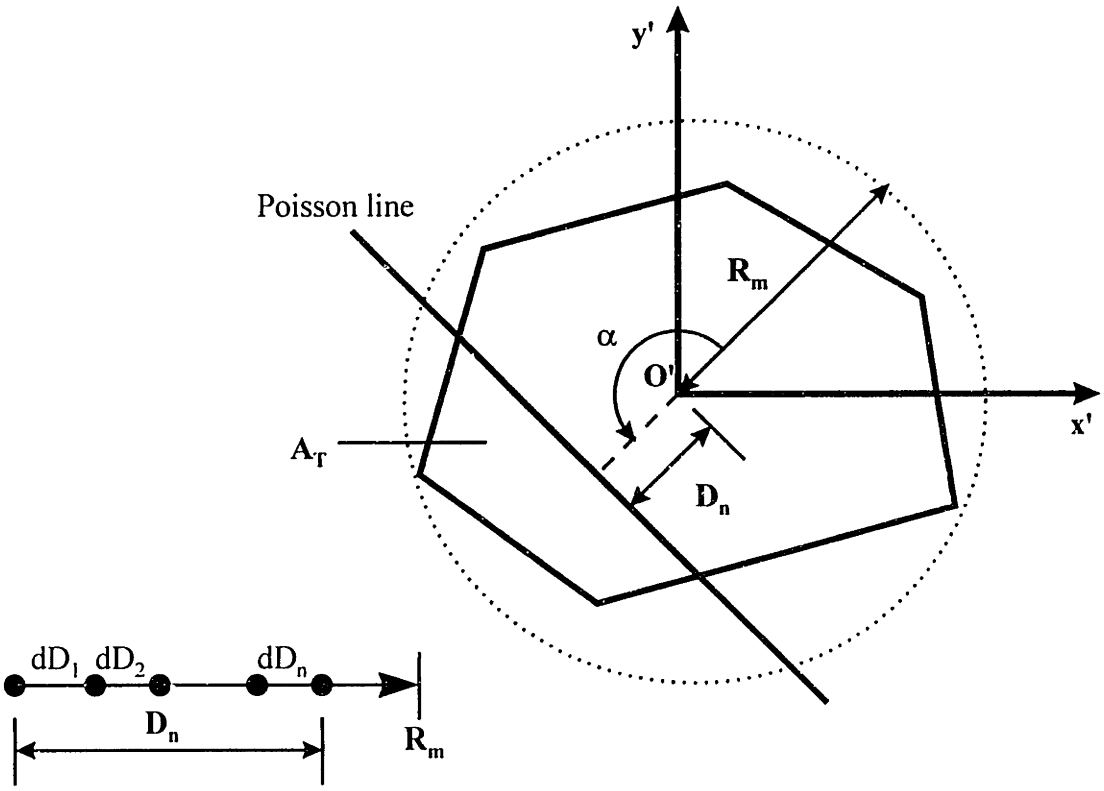


FIGURE A.3 Secondary stochastic process: generation of a Poisson line.

Figure A.3 illustrates the algorithm for generation of Poisson line tessellation of intensity λ in a finite polygonal region, defined by the intersection of a random plane with the 3D modeling volume. The intersection of the plane with the curved top surface is approximated by several line segments. The center of the polygonal region (assumed to coincide with the local origin O') is chosen as a starting point. The maximum possible distance from the origin to a line that intersects the region is equal to the radius R_m of the smallest circle that inscribes the region. The

ordered distances D_i from the origin to the Poisson lines form a Poisson point process. In a Poisson point process, the incremental distances dD_i follow an exponential distribution. The n-th Poisson line is defined by an angle α uniformly generated between zero and 2π, and a distance D_i that is the sum of n distances dD_i, generated according to an exponential distribution with intensity λ. Starting from $D_0=-R_m$, lines are generated until D_i exceeds R_m. The number of lines generated by this method has Poisson distribution with mean $2\lambda R_m$.

The points of intersection of Poisson lines determine the vertices of polygonal tiles. The mean area E[A] of the polygons is a function of the intensity λ of the line tessellation:

$$\mathbf{E[A]} = \frac{\pi}{\lambda^2} \tag{A-9}$$

The standard deviation of all polygon areas is $\sigma_A = 1.98$ E[A].

Polygon marking procedure and distribution of fracture sizes and shapes

The process of dividing each plane into a fractured region and its complementary region of intact rock is homogeneous in a statistical sense. The probability of marking a polygon as fractured, P_f, is calculated individually for every polygon:

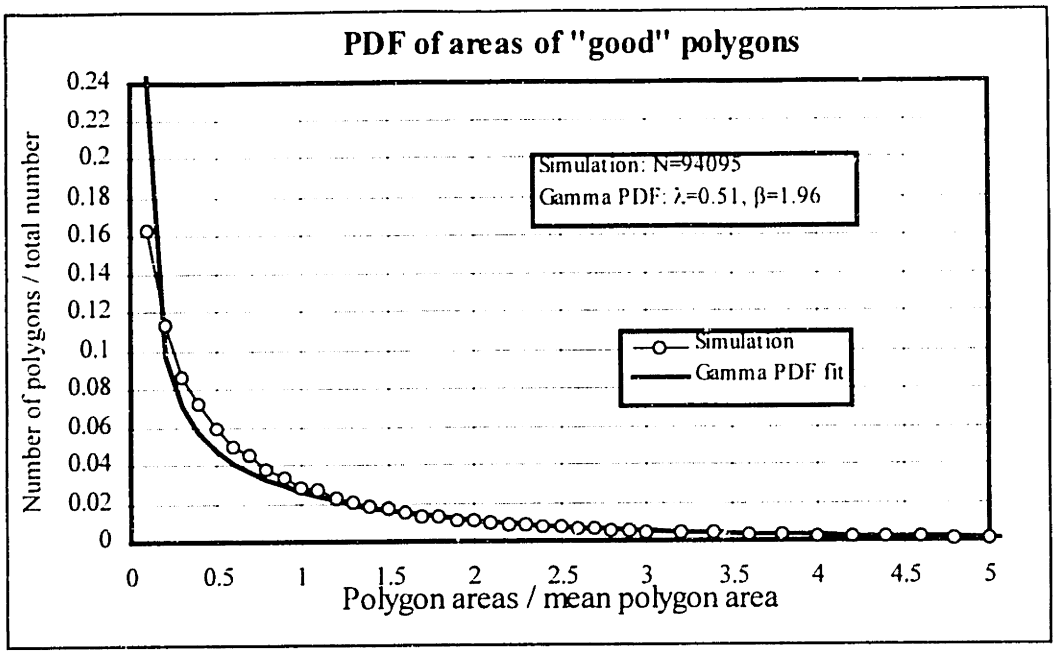
$$\mathbf{P_f = P_f(size, shape, location)} \tag{A-10}$$

Only polygons that have shapes similar to that of natural fractures are retained from the population of polygons created by a Poisson line tessellation. The shape of a "good" is that of a typical natural fracture, here defined as:

(1) the polygon has at least four vertices;
(2) all angles are at least $\pi/3$ (60 degrees);
(3) the polygon elongation (*e* in Figure A.2) is not more than 1.6 (modeling of elongated shapes is discussed by Ivanova 1995).

Figure A.4 a and b illustrate the PDF and cumulative PDF of sizes (areas) of "good" polygons (i.e. potential fractures) generated through a Poisson line tessellation. In the numerical simulation, the expected area of all polygons E[A] was given as input, and the intensity of the Poisson line tessellation was back-calculated from Equation A-9. The tessellation was performed in a finite region with total area A_T according to the algorithm described above. Figure A.4 shows the portion of the PDF in the range where the areas of "good" polygons are smaller than five times the expected area, and includes more than 95% of all "good" polygons.

a)



b)

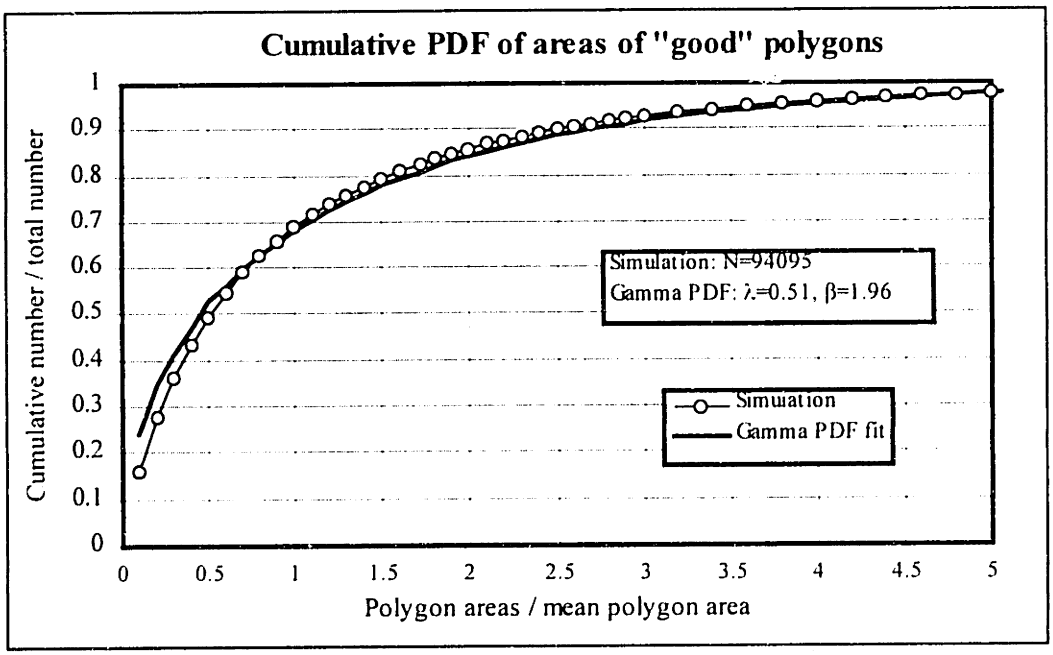


FIGURE A.4 Distribution of sizes (areas) of polygons with "good" fracture-like shapes obtained through a Poisson line tessellation: a) PDF of polygon areas; b) cumulative PDF of polygon areas.

In Figure A.4, the size intervals on the abscissa are given in terms of fracture areas normalized by the mean area of "good" polygons E[A']. For example, the interval (0.4-0.5) on the abscissa corresponds to fracture areas (0.4-0.5) E[A']. On the ordinate, the number of "good" polygons is normalized by the total number of "good" polygons in the simulation. For example, in Figure A.4a, the number 0.06 on the ordinate, plotted above value 0.5 on the abscissa, indicates that the number of "good" polygons that have sizes (0.4-0.5) E[A'], is equal to 6% of the total number of "good" polygons produced by a Poisson line tessellation. In Figure A.4 b, the number 0.5 on the ordinate plotted above value 0.5 on the abscissa indicates that 50% of the "good" polygons have areas smaller than half the mean area of "good" polygons (i.e. the median of the size PDF is at 0.5 E[A']). The fit of Gamma PDF (α=0.51, β=1.96) to the PDF of fracture sizes is also shown in Figure A.4. **Table A.1** summarizes the statistics of the PDF of "good" polygons produced by a Poisson line tessellation with intensity λ.

STATISTICAL PROPERTY	VALUE BASED ON SIMULATIONS
Mean fracture area, $E[A']$	$E[A'] = C_A \frac{\pi}{\lambda^2}$, where C_A=2.2
Standard deviation of fracture area, $\sigma_{A'}$	$\sigma_{A'} = 1.4E[A']$
Median of fracture area, $M_{A'}$	$M_{A'} = 0.5E[A']$
Cumulative fracture area, A'_T	$A'_T = \gamma A_T$, where γ=0.4

TABLE A.1 Statistics of the distribution of areas of marked polygons (i.e. "good" polygons with fracture-like shapes), produced by a Poisson line tessellation of intensity λ in a region with total area A_T.

Two correction coefficients best express the relationship of the areas of "good" polygons to the intensity of the underlying Poisson line tessellation:

(1) the ratio C_A of the mean area of "good" polygons to the mean area of all polygons produced by a tessellation with intensity λ;
(2) the ratio $\gamma = A'_T/A_T$, where A'_T is the sum of the areas of all "good" polygons, created within a finite region of total area A_T by a Poisson line tessellation of intensity λ.

The coefficients C_A and γ do not depend on the intensity λ of the Poisson line network but only on the rule by which polygons are marked as "good". **Figure A.5** illustrates an explanation of this important property of the population of

The coefficients C_A and γ do not depend on the intensity λ of the Poisson line network but only on the rule by which polygons are marked as "good". **Figure A.5** illustrates an explanation of this important property of the population of polygons created by a Poisson line tessellation. The sizes of polygons are defined by the intensity λ of the line network, whereas the shapes are determined by the PDF of line orientations. The ordered distances from an arbitrary point to the lines of a Poisson line process of intensity λ form a point process of intensity 2λ. If the intensity of the line process is increased (or decreased) n times, the intensity of the point process of the distances is correspondingly increased (or decreased) n times. If the PDF of line orientations is kept the same, then increasing (or decreasing) the line intensity λ means that in statistical sense the lines are simply moved closer (or further away) from the point of reference. Therefore, the distribution of polygon shapes, and hence the ratio of polygons with "good" shapes versus all polygons, is the same for any intensity of the line process. Once the rule of marking polygons according to their shape is established, the correction coefficients C_A and γ can be uniquely calculated. Therefore, C_A=2.2 and γ=0.4, given in Table A.1, are valid for any intensity λ of the Poisson lines, as long as "good" polygons are marked according to the three rules listed above.

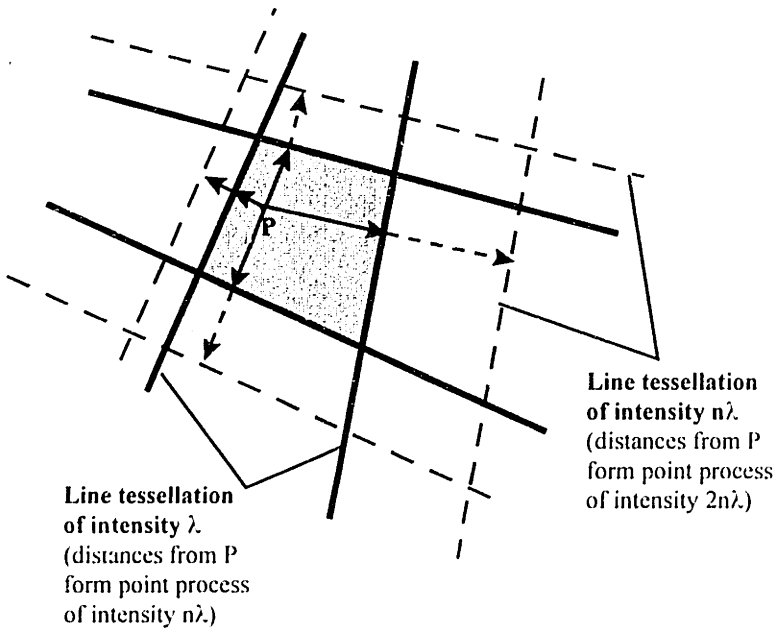


FIGURE A.5 Diagram illustrating the similarity in shape of polygons created by Poisson line tessellations that have different intensity, but the same PDF of line orientations.

The distribution of polygon sizes, illustrated in Figure A.4 and described in Table A.1, is representative of natural fracture systems in the sense that they usually include few large fractures and many small fractures. However, in many cases one

may want to represent a smaller variation of fracture sizes (i.e. a smaller standard deviation $\sigma_{A'}$ of sizes of "good" polygons). In order to do so, the model currently implements a simple procedure of marking "good' polygons not only by shape, but also according to their equivalent radii relative to the mean equivalent radius of "good" polygons $E[R'_e]$, and their areas relative to the mean polygon area E[A] of the underlying Poisson line tessellation. **Table A.2** summarizes statistics of the PDF of areas of "good" polygons obtained through a Poisson line tessellation and different shape-and-relative-size marking processes (for comparison, the characteristic values for the marking process only by shape are also shown). As Table A.2 shows, the correction coefficients C_A and g are different for every marking rule. Theoretically, as long as the marking is only according to shape and relative size, the coefficients C_A and γ do not depend on the intensity λ of the Poisson line tessellation.

MARKING RULE	$C_A = \frac{E[A']}{E[A]}$	$\gamma = \frac{\sum_{i=1}^{N} A'_i}{A_T}$	$\frac{\sigma_{A'}}{E[A']}$	$\frac{M_{A'}}{E[A']}$
1) ***All with good shape***	2.2	0.4	1.4	0.5
2) $A'_i > E[A]$	3.6	0.36	0.93	0.7
3) $A'_i > 2E[A]$	5.0	0.30	0.74	0.75
4) $R'_{e,i} < 3E[R'_e]$	1.8	0.38	1.26	0.6
5) $R'_{e,i} < 2E[R'_e]$	1.4	0.23	0.96	0.7
6) $A'_i > E[A]$, $R'_{e,i} < 3E[R'_e]$	3.6	0.36	0.90	0.8
7) $A'_i > E[A]$, $R'_{e,i} < 2E[R'_e]$	3.4	0.36	0.72	0.75

TABLE A.2 Statistics of "good" polygons, marked by shape and relative size, obtained through a Poisson line tesseliation of intensity λ in a finite region with total area A_T. $E[A]=\pi/\lambda^2$: theoretical mean area of all polygons; N: total number of "good" polygons; $E[A']$, $\sigma_{A'}$, and $M_{A'}$: expected value, standard deviation, and median, respectively, of the areas of "good" polygons; A'_i and $R'_{e,i}$: area and equivalent radius (radius of a circle with area equal to A'_i) of the i-th "good" polygon; $E[R'_e]$: expected value of the equivalent radius of "good" polygons.

Applying a marking rule according to relative polygon size leads to truncation of the size PDF, hence to decrease in the spread of sizes of remaining polygons (i.e. a smaller $\sigma_{A'}$). **Figure A.6** shows the PDFs of areas of "good" polygons, obtained through a Poisson line tessellation, and three of the marking processes in

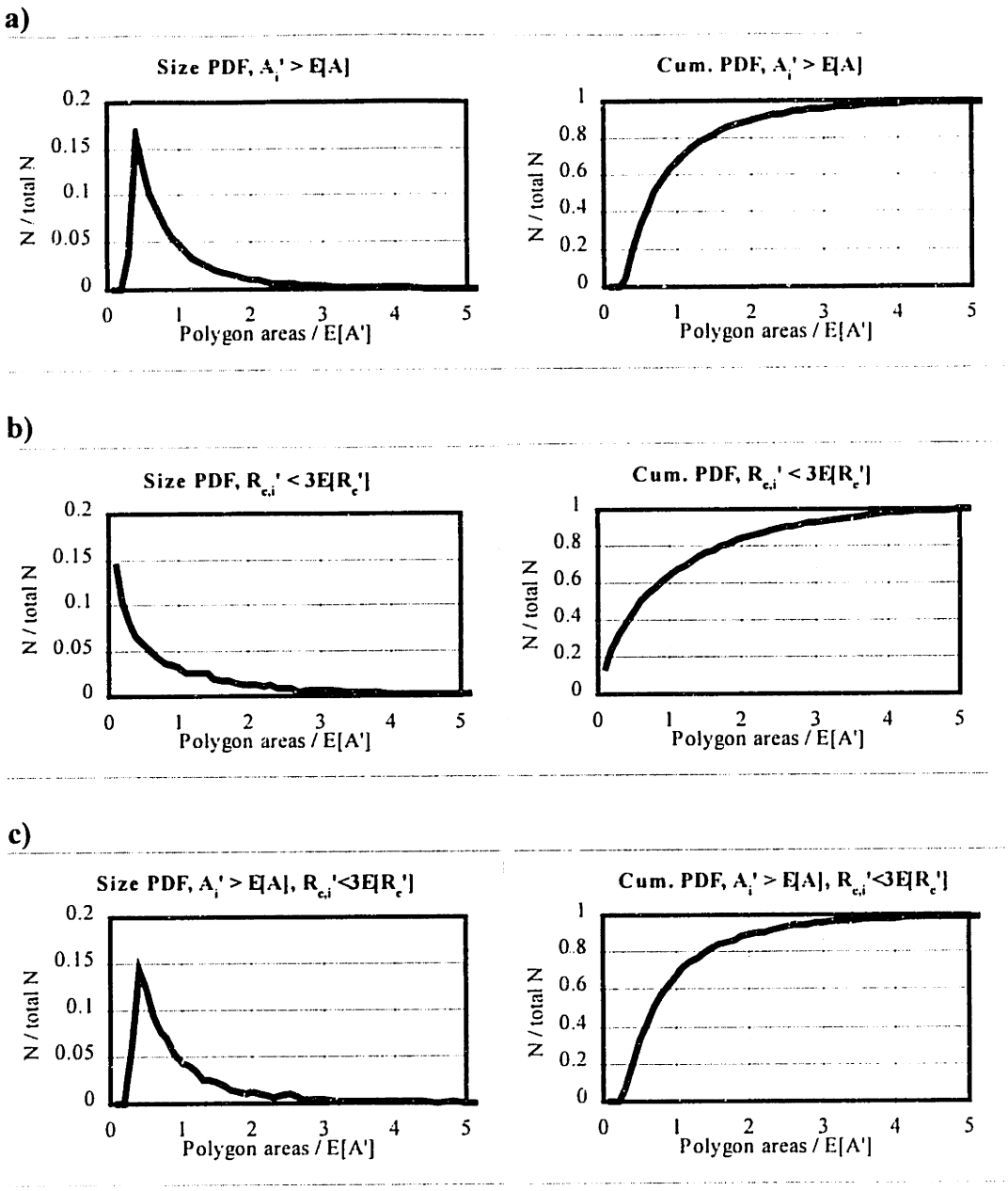


FIGURE A.6 Distribution of sizes (areas) of "good" polygons, obtained through a Poisson line tessellation of intensity λ, and marked according to shape and relative size. Size marking rules: a) $A'_i > E[A]$ (rule 2 in Table A.2); b) $R'_{e,i} < 3E[R'_e]$ (rule 4 in Table A.2); c) $A'_i > E[A]$, $R'_{e,i} < 3E[R'_e]$ (rule 6 in Table A.2). A'_i and $R'_{e,i}$ are the area and the equivalent radius of the i-th "good" polygon; $E[A]=\pi/\lambda^2$ is the expected area of all polygons; $E[R'_e]$ is the expected equivalent radius of "good" polygons; N is the number of "good" polygons in a size interval, and total N is the total number of "good" polygons in the simulation.

Poisson plane network intensity and total fracture area in the modeling volume

The expected cumulative area of potential fractures per unit volume is:

$$E\left[\frac{A_{T,f}}{V}\right] = P_{32} = \gamma\mu \qquad \text{(A-11)}$$

where μ is the intensity of the Poisson plane network, and $\gamma \leq 0.4$ is the fractured portion of a Poisson plane. The expected cumulative area of "good" polygons per unit volume is independent of the shape and size of the modeling volume in which the Poisson plane process is generated. The modeled P_{32} only depends on the intensity μ of the Poisson line network and on the rules for marking of polygons as potential fractures. Truncations of fractures by the boundaries of the modeling volume do not affect significantly this property, as long as the desired fracture size is much smaller than the size of the entire region of interest.

Zone marking of polygons and varying fracture intensity in space

To obtain different fracture intensity of a fracture set in different portions V_1, V_2, etc. of the modeling volume V, one has to calculate the different probabilities P_1, P_2, P_i, etc. for polygon marking in the regions V_1, V_2, V_i, etc. Volumes V_i do not have to be continuous; they can be defined as a function of any continuous rock property. For example, volumes V_i may be defined in terms of distances from the fractures of a primary set or from a fault face. In this case, a zone is defined by the maximum and minimum value of the distance from the face of a polygon-fracture to the face of another polygonal feature (**Figure A.7**). Within every zone j the zone probability P_j is constant, but that constant is different in different zones. Polygons in zone j are retained with probability P_j and discarded with probability $1 - P_j$, where $0 < P_i \leq 1$. Also, V_i can be defined as "the regions where the rock is dolomite" or "the regions where the porosity of the rock is not more than n%", etc. It is only the intensity of fractures, and not the orientation, that can be controlled by the zone marking process. Any marking according to polygon orientation would affect the function $f_{0,\phi}(\theta,\phi)$ in Equation (A-3) and the constant μ itself.

Summary of fracture intensity modeling

In summary, desired fracture intensity is modeled in the following order:

1) Mean fracture size $E[A']$, fracture intensity $P_{32,i}$ in various regions V_i and the extent of those regions are given as input.
2) The intensity of the Poisson plane process is calculated as $\mu = P_{32,max}/\gamma$.
3) Poisson planes are generated in the total volume V.
4) The expected mean polygon area of the Poisson line network is calculated as $E[A] = E[A']/C_A$.

3) Poisson planes are generated in the total volume V.
4) The expected mean polygon area of the Poisson line network is calculated as $E[A]= E[A']/C_A$.
5) Poisson line tessellation with intensity $\lambda=(\pi/E[A])^{1/2}$ is generated on the planes.
6) The probability of marking "good polygons" volume V_i is calculated as $P_i=P_{32,i}/P_{32,max}$.
7) Polygons with bad shapes are discarded and polygons with good shapes are marked as fractures (with probability P_i in region V_i).

The above algorithm ensures that the expected value of the intensity of the fracture set will be the desired intensity P_{32} (or $P_{32,i}$ in regions V_i, respectively).

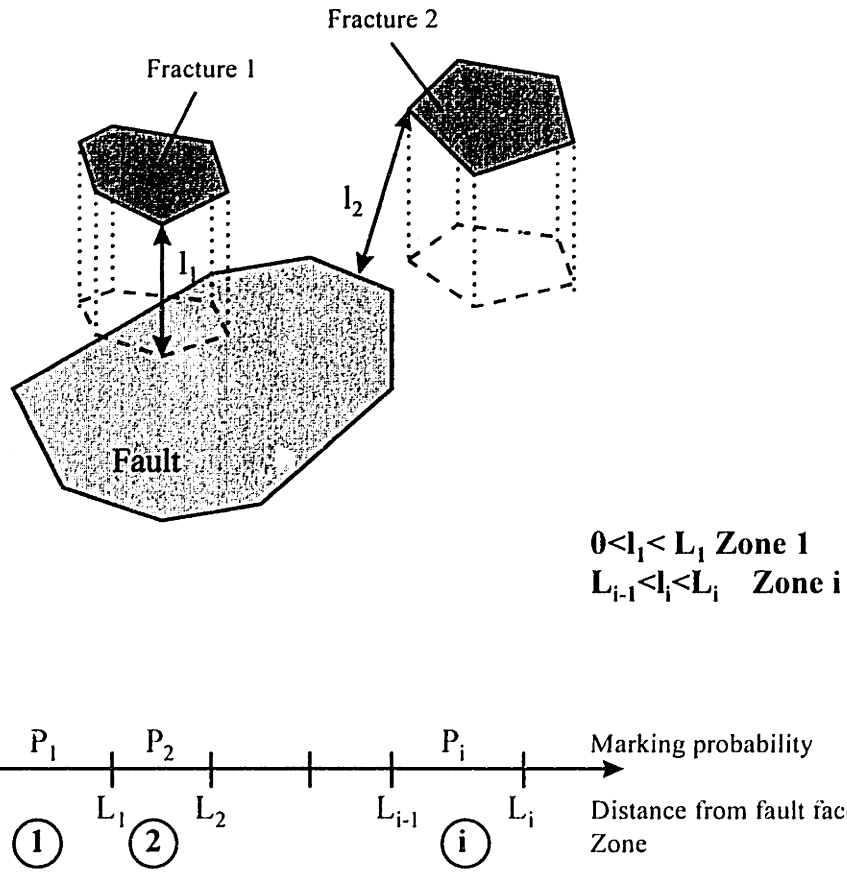


FIGURE A.7 Secondary stochastic process: definition of fracture zones and zone marking probability.

Fractal Line Tessellation

A variation of the Poisson line tessellation, called Fractal Line Tessellation (FLT), can be used as another possible method for subdivision of the planes into polygons. In the FLT, first, the original plane is divided into large polygons by a small number of Poisson lines, and some of the polygons are marked as fracture zones. Next, a Poisson line tessellation and a marking process is performed only in the fracture zones. This procedure is repeated in a number of iterations. In cases of low intensity, the FLT can enable much faster subdivision of planes into fractured and intact areas, compared to the general Poisson line tessellation. The FLT can also model various fracture size variations, for example, clustering of small fractures around big fractures. The FLT algorithms can be described in terms of three parameters: number of iterations, N; probability of marking "good" polygons as fractured at every iteration, P; and fractal dimension, F, which is the ratio of expected areas of "good" polygons in two consecutive iterations. Suggestions for development of the fractal line tessellation can be found in Chapter 3 (in the discussion of the model in Section 3.4).

Modeling of stress field variation: tertiary stochastic process

The translation procedure of the tertiary stochastic process (Figure A.1c) controls the ratio of coplanar fractures and fractures that are parallel but not coplanar. The rotation algorithm (Figure A.1c) represents possible deviations of fracture orientations from the regional directions due to variations of the stress field near local geologic structures.

Random polygon translation

In the tertiary stochastic process translation is performed in the frame of reference of the fracture plane ($O'x'y'z'$ in Figure A.2). Translation is accomplished by assigning a non-zero coordinate z'_i to the center and the vertices of a polygon and hence to the entire polygon. A polygon is translated at a maximum distance:

$$dz'_{max} = C\frac{E[R'_e]}{R'_e}E[R'_e] \qquad \text{(A-12)}$$

where R'_e is the equivalent radius of a polygon, and $E[R'_e]$ is the expected value of that radius. Thus larger polygons are shifted closer to their original positions than smaller polygons. C in Equation (A-12) is a coefficient of fracture coplanarity: the smaller C, the more coplanar fractures can be expected.

Random polygon rotation

The random rotation of polygons in the tertiary process is used for cases when the local conditions modify the stress field so much that the most likely fracture orientation deviates from the most likely orientation defined by the general stress

field. Fracture rotation is primarily used to relate fracture orientations to curved surfaces, i.e. folds. In the current version of GEOFRAC, the 3D geometry of a fold is described by a cubic function. The coefficients of the cubic equation can be derived through polynomial fit to the elevations of the contacts of folded strata in boreholes or on mapped outcrop exposures.

The orientations of fractures in folded strata are related to the varying strike and dip of the fold surface (**Figure A.8**). The strike and dip of a fold surface at point P can be calculated as:

$$\mathbf{strike} = \theta_f - \frac{\pi}{2} = \mathbf{Arc\,tan}\left(\frac{n_Y}{n_X}\right) - \frac{\pi}{2}$$

$$\mathbf{dip} = \phi_f = \mathbf{Arc\,cos}\left(n_Z\right)$$

(A-13)

where $n_f=(n_X, n_Y, n_Z)$ is the unit normal vector to the fold surface at point P (directional first derivatives of the function F(X,Y,Z)=0 that describes the fold), and θ_f and ϕ_f are the azimuth and latitude of n_f. To compare the orientation of a fracture-polygon to that of a fold, one can calculate the angle α between n_f and the unit normal vector of the fracture plane, $n_p = (n_{X,p}, n_{Y,p}, n_{Z,p})$. α can be calculated from the vector dot product:

$$\vec{n}_p \bullet \vec{n}_f = \left|\vec{n}_p\right|\left|\vec{n}_f\right|\cos\alpha = (1)(1)\cos\alpha = n_{X,p}n_X + n_{Y,p}n_Y + n_{Z,p}n_Z$$

$$\alpha = \mathbf{Arc\,cos}\left(n_{X,p}n_X + n_{Y,p}n_Y + n_{Z,p}n_Z\right)$$

(A-14)

To compare the strike of a fracture-polygon to the local strike of a fold, one can calculate the difference α_{strike} between the azimuth of the polygon, θ_p, and the azimuth of the fold surface at the polygon center:

$$\alpha_{strike} = \left|\theta_p - \theta_f\right|$$

(A-15)

To compare the dip of a fracture-polygon to the local dip of a fold, one can calculate the difference α_{dip} between the latitude of the polygon, ϕ_p, and the latitude of the fold surface at the polygon center:

$$\alpha_{dip} = \left|\phi_p - \phi_f\right|$$

(A-16)

Angles α, α_{strike}, and α_{dip} are checked against specified relationships of the fractures to the fold. For example, a small angle α indicates that the fracture is

subparallel to the fold surface. An angle α_{strike} close to 90° means that the fracture strikes approximately orthogonal to the local fold strike (i.e. parallel to the local slope of the fold surface). An angle α_{strike} close to zero or 180° means that the fracture strikes approximately parallel to the local fold strike (i.e. orthogonal to the local slope of the fold surface). A small angle α_{dip} indicates that the fracture has approximately the same dip as the fold at that location.

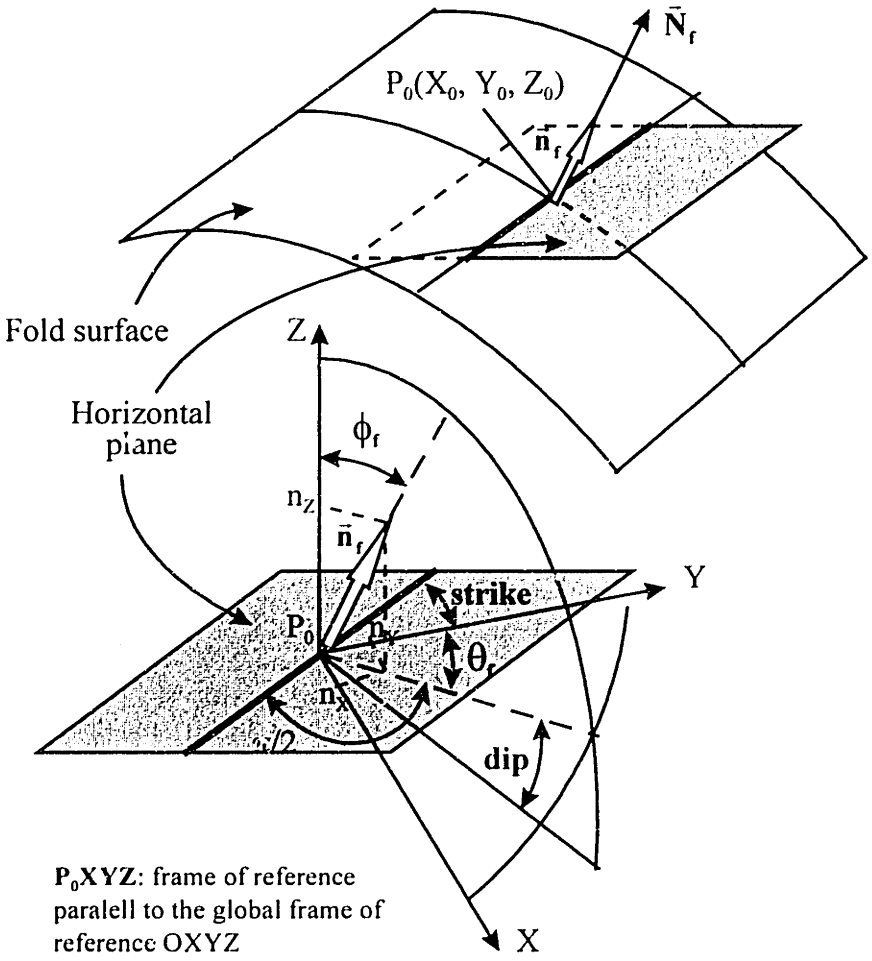


FIGURE A.8 Tertiary process: relationship by dip and strike of fracture orientation to the local orientation of a fold.

If the orientation of a fracture-polygon does not conform to the specified relationships between fractures and fold, the polygon is rotated. For example, rotation by dip (so that the new fracture dip is subparallel to the local fold dip) is performed by assigning a new latitude angle ϕ_{new} to the polygon:

$$\phi_{new} \propto \mathbf{U}\left[\phi_f - \delta\alpha_{dip}, \phi_f + \delta\alpha_{dip}\right] \tag{A-17}$$

where $\delta\alpha_{dip}$ is a small angle of allowed deviation of the fracture dip from the local dip of the fold. Rotation by strike is performed by assigning a new azimuth angle θ_{new} to the polygon:

$$\theta_{new} \propto \mathbf{U}\left[\theta_f - \delta\alpha_{strike}, \theta_f + \delta\alpha_{strike}\right] + \mathbf{C}_s \frac{\pi}{2} \tag{A-18}$$

where $\delta\alpha_{strike}$ is a small angle of allowed deviation of the fracture strike from the local strike of the fold, or from the horizontal direction orthogonal to the local strike of the fold. The coefficient C_s is either C_s=0 or C_s=1.0, depending on whether the fracture is rotated to be concentric or radial to the fold, respectively. If necessary, a completely new orientation of the fracture can be generated (in a frame of reference where the local normal vector to the fold is assumed to be the mean polar direction). After calculating the new dip or/and strike, the fracture-polygon is rotated. Rotation of a polygon means that new 3D coordinates are calculated for every vertex (formulas for rotation in the global frame of reference of the model are given in Ivanova 1995).

A.1.2 Modeling of a fracture system

A fracture system is reproduced by superposition of independent and dependent fracture sets. An independent set is not related to any other sets; however, it is modeled with the stochastic sequence, described in the previous section, and there is a correlation between the geometric characteristics of its members. A dependent fracture set is modeled by the same stochastic processes. Dependence on previously generated sets can be obtained in many ways; for example, the zone probability can be defined as a function of the distance to primary fractures.

Characteristic geometries of rock fracture systems in different geologic settings are reviewed in Chapter 2 of this thesis. Specific algorithms for representation of rock fracture systems in the major geologic fracture-producing environments (folds, crustal faults, remote tension, thermal contraction, and central structures) are discussed in Chapter 3 (Sections 3.3.2-3.3.6).

A.2 PROGRAM GEOFRAC

This section of the Appendix is a summary of the numerical code GEOFRAC which implements the 3D fracture system model, presented in Section A.1 and Chapter 3. The program is not a commercial software and a user-friendly interface has not

been developed yet. Section A.2.1 describes the GEOFRAC C++ classes that represent geometric objects: polygons, surfaces, etc. Section A.2.2 explains the structure of input and output files of the general program and the modules of GEOFRAC. The source files are appended in Section A.2.3.

GEOFRAC is written in C/C++ for UNIX. The program has been compiled in the MIT computing environment (Athena release 8.1) on Sun SPARCstations (running Solaris 2.5/SunOS 5.5) and on SGI INDY workstations (running Irix). The source files are in directory **/mit/1.381_dev/GEOFRAC**. The directory **/mit/1.381_dev** is a "locker" that is served by Project Athena. Only users that are members of the list **1.381_dev** can have access to the locker directory /mit/1.381_dev. Only the administrator can add new members to the list. To find the current administrator of the list, Athena users can run the **listmaint** program:

athena% listmaint

Members of the list 1.381_dev can access the locker by typing:

athena% add 1.381_dev
athena% cd /mit/1.381_dev

Non-Athena users, interested in the 3D fracture system model and in the computer program GEOFRAC, should contact Prof. Herbert H. Einstein (einstein@mit.edu; Room 1-342, MIT).

A.2.1 C++ classes in GEOFRAC

Table A.3 summarizes the C++ classes in GEOFRAC, including the names of the header files where the classes are defined, and the names of the source files where the functions of the classes are defined. Brief description of all classes and the contents of the corresponding files follows.

Classes representing coordinates

Class **Cartesian** is defined in header file cartesian.h. The class includes three Cartesian coordinates in 3D: **X, Y, Z**. Class Cartesian also includes functions for rotation and translation of coordinates in a Cartesian frame of reference. The source code of these functions is in file cartesian.C.

Class **Polar** is defined in header file polar.h. The class includes two Polar coordinates in 3D: **phi, theta**. Class Polar also includes functions for 3D transformation between polar and Cartesian coordinates. The source code of these functions is in file polar.C. The source code of several functions for random generation of polar coordinates in 3D is in file random.C.

CLASS	HEADER FILES	SOURCE FILES
Cartesian	cartesian.h	cartesian.C
Polar	polar.h	polar.C
Point	point.h	point.C
Plane	plane.h	plane.C, random.C
Surface	surface.h	surface.C
Cubic	cubic.h	cubic.C
Volume	volume.h	volume.C
Line	line.h	line.C
Polygon	polygon.h	polygon.C, initial.C, divide.C, rotation.C, cell.C, intersections.C
Node, ListPolygons	listpol.h	listpol.C, rotation.C, cell.C, zones.C, intersections.C
Node_line, ListLines	listline.h	listline.C
Stat	stat.h	-
Borehole	borehole.h	borehole.C
Cell, Column	cell.h	cell.C

TABLE A.3 C++ classes in GEOFRAC, header files where the classes are defined, and source files that include the class functions. All files are in directory /mit/1.381_dev/GEOFRAC. Classes Cell and Column are defined for a GEOFRAC module specifically developed for the Yates field case study (Chapter 4).

Class **Point** is defined in header file point.h. The class includes three Cartesian coordinates (by inheritance of class Cartesian): **X**, **Y**, **Z**. Class Point also includes a **pointer to a Point** which enables one to create linked lists of points. The structure of class Point is illustrated in **Figure A.9**. Class Point also includes a function for graphical output, included in file point.C, which prints a MATHEMATICA code (details on graphical output follow in Section A.2.2).

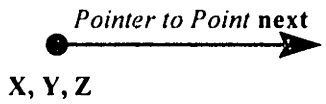


FIGURE A.9 Structure of GEOFRAC class Point.

Classes representing surfaces and volumes

Class **Plane** is defined in header file plane.h. Class Plane includes the following variables: Polar **MeanPole** representing the mean pole of the parent set of planes (Oz in Figure A.2); Polar **abs_polar** and **rel_polar** representing the polar coordinates of the plane normal vector (Oz' in Figure A.2) in the global frame of reference (OXYZ), and in the local frame of reference of the set of planes (Oxyz), respectively; Cartesian **abs_cart** and **rel_cart** representing the Cartesian coordinates of the normal vector in the global and local frame of reference, respectively; floating values **A**, **B**, **C**, and **D** of the plane equation:

$$\mathbf{AX+BY+CZ=D} \tag{A-19}$$

where A, B, and C are the global Cartesian coordinates of the normal vector to the plane, and D is the distance from the global origin to the plane (d in Figure A.2).

Constructors for class Plane are included in file plane.C. Class Plane can be constructed by various methods, for example, through a point and orthogonal to a vector, etc. The Plane constructor that is most commonly used in GEOFRAC involves random generation of **rel_polar** according to a specified spherical PDF, and generation of the distance D according to the algorithm, described in Section A.1 (p. 3-4). The functions for generation of class Polar according to uniform, Fisher, etc. PDFs are included in file random.C.

Class **Surface** is defined in header file surface.h. Class Surface represents a quadratic surface (for example, the topographic surface). The class includes six floating values, **A**, **B**, **C**, **D**, **E**, and **F**, of the quadratic equation:

$$\mathbf{Z = AX^2 + BXY + CY^2 + DX + EY + F} \tag{A-20}$$

File surface.C includes functions of class Surface for the following calculations: coordinates of points on a quadratic surface; normal vector, dip, and strike, at a point on a quadratic surface; volume, enclosed under a quadratic surface over a rectangular area, etc. A function to find the point of intersection of a line and a surface is included in file intersections.C.

Class **Cubic** is defined in header file cubic.h. Class Cubic represents a cubic surface (for example, a fold). The class includes ten floating values, **A**, **B**, **C**, **D**, **E**, **F**, **G**, **H**, **I**, and **J** of the cubic equation:

$$\mathbf{Z = AX^3 + BX^2Y + CXY^2 + DY^3 + EX^2 + FXY + GY^2 + HX + IY + J} \tag{A-21}$$

By varying the coefficients, one can represent various fold shapes. For example, if all coefficients except J and E are zero, Equation A-21 represents a cylindrical fold with vertical axial plane parallel to the global axis OY (a syncline if E>0, or

an anticline if E<0). File cubic.C includes functions of class Cubic for calculation of normal vector, dip, and strike at a point on a cubic surface.

Class **Volume** is defined in header file volume.h. Class volume is constructed from a top quadratic surface over area defined by X_m and Y_m (see inset 2 in Figure A.2). Class Volume includes: Surface **top**; four objects of class Points, representing the top four corners of the volume; two floating values: **volume**, and **Zmax**. File volume.C includes functions for generation of a random point inside the volume.

Classes representing fractures and fracture traces

Class **Line**, defined in header file line.h, is used for representation of a single linear object in 3D (e.g. a fracture trace on a plane, or a line of intersection between two polygonal fractures). Class Line includes two objects of class Point: **end1, end2**. Class Line also includes an object **vector** of class Cartesian that represents the unit vector of a line, and a floating value **length** equal to the length of a line. The structure of class Line is illustrated in **Figure A.10**. Class Line also includes functions for graphical output of lines (MATHEMATICA code, discussed in Section A.2.2), calculation of intersections with lines and planes (including two algorithms by Sedgwick, 1992), and rotation of lines in a 3D Cartesian frame of reference. The functions are included in file line.C.

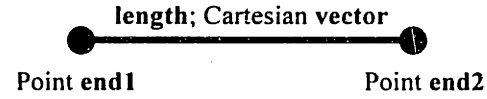


FIGURE A.10 Structure of GEOFRAC class Line.

Class **Polygon**, defined in header file polygon.h, is used for modeling of an individual fracture or another 2D polygonal object in 3D space (e.g. a major fault). In GEOFRAC, a polygon is represented as a linked list of points, as illustrated in **Figure A.11**. The first Point object in the linked list is declared by a pointer to a Point in class Polygon: **head**. The head Point points to the second Point, etc. Class Polygon also includes: Point **center**, floating values **area**, **radius, dip, strike**, an integer **noP** (the number of points in the polygon), and two Polar objects: **Pole**, **setPole**. Pole represents the polar orientation of the polygon in the global frame of reference in terms of azimuth and latitude (Pole.theta, Pole.phi). SetPole represents the mean orientation of the fracture set to which a polygon belongs. If the polygon (e.g. a major fault) does not belong to a set, SetPole.theta=0, and SetPole.phi=0 (i.e. SetPole coincides with the global Z axis).

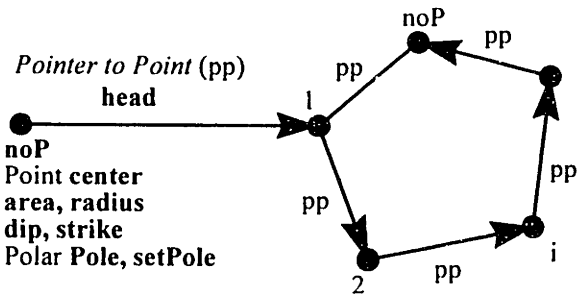


FIGURE A.11 Structure of GEOFRAC class Polygon.

Class Polygon includes all functions that involve operations on an individual polygonal object in 3D space. Basic functions, defined in file `polygon.C`, include: graphical output of a polygon (MATHEMATICA code, see Section A.2.2); calculation of distances to other polygons; rotation and translation of a polygon in a 3D Cartesian frame of reference; marking of a polygon by shape or size; calculation of the area and equivalent radius of a polygon, etc. Functions for calculation of the polygon intersections with lines, other polygons, planes, and surfaces, are defined in file `intersections.C`. File `initial.C` includes functions for calculation of the initial polygon, cut by a plane (class Plane) from the modeling volume (class Volume). File `divide.C` includes functions for tessellation of a polygon by lines (Poisson line tessellation).

Class **Node** and class **Node_line** are auxiliary classes that enable one to create linked lists of polygons and lines, respectively. Class Node, defined in header file `listpol.h`, includes a pointer to another Node and a pointer to an object of class Polygon: **content**. Class Node_line, defined in header file `listline.h`, includes a pointer to another Node_line and a pointer to an object of class Line: **content**. The structures of the two classes are illustrated in **Figure A.12**.

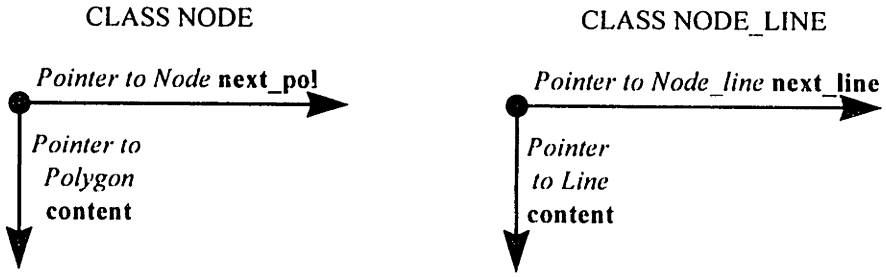


FIGURE A.12 Structure of GEOFRAC classes Node and Node_line.

Class **Stat** is an auxiliary class, defined in header file `stat.h`. Class Stat includes three floating values: **Mean**, **SD**, and **total**. Functions that produce objects of

class Stat by calc 'ating statistics of polygon areas and statistics of line lengths are included in file listpol.C and file listline.C, respectively.

Class **ListLines**, defined in header file listline.h, is used for modeling of a set of fracture traces or other linear objects in 3D (e.g. intersections of fractures). In GEOFRAC, a trace set is represented as a linked list of objects of class Node_line, as illustrated in **Figure A.13**. The first Node_line in the linked list is declared by a pointer to a Node_line in class ListLines: **head_line**. A pointer of the head Node_line points to the second Node_line in the linked list; a pointer of the second Node_line points to the third Node_line, etc. Every Node_line in the linked list has a pointer that points to an object of class Line, representing a fracture trace or another linear object. Class ListLines also includes an integer, indicating the number of lines in the linked list: **Nline**. File listline.C includes the source code of functions for graphical output of a list of lines (MATHEMATICA code), addition of new lines to an existing list, and calculation of statistics (mean, standard deviation, and total trace length) of the lengths of lines in a list of lines.

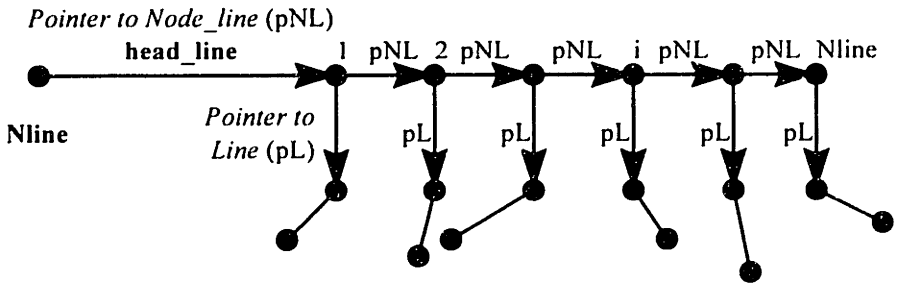


FIGURE A.13 Structure of GEOFRAC class ListLines.

Class **ListPolygons**, defined in header file listpol.h, is used for modeling of a fracture set or another set of 2D polygonal objects in 3D (e.g. a set of major faults). In GEOFRAC, a fracture set is represented as a linked list of objects of class Node, as illustrated in **Figure A.14**. The first Node in the linked list is declared by a pointer to a Node in class ListPolygons: **head_pol**. A pointer of the head Node points to the second Node; a pointer of the second Node points to the third Node, etc. Every Node in the linked list has a pointer that points to an object of class Polygon, representing a fracture or another 2D polygonal object in 3D. Class ListPolygons also includes an integer, indicating the number of polygons in the linked list: **Npol**.

Class ListPolygons includes all functions that involve operations on a set of polygonal objects in 3D space. Basic functions, defined in file listpol.C, include: graphical output of a list of polygons (MATHEMATICA code); calculation of

distances to other polygons; translation of a list of polygons in a 3D Cartesian frame of reference; marking of a list of polygons by shape or size; calculation of the areas and equivalent radii of a list of polygons, including statistics (mean, standard deviation, and total polygon area), etc. File listpol.C includes also a function for addition of a list of polygons, used for modeling of superposition of fracture sets into a fracture system. Functions for calculation of the intersections of a list of polygons with other polygons, planes, and surfaces, are defined in file intersections.C. Functions for marking and rotation of polygons in relation to a curved surface (quadratic or cubic) are included in file rotation.C. File divide.C includes functions for tessellation of a list of polygons by lines (Fractal line tessellation). File zones.C includes functions for marking of a list of polygons (e.g. a secondary fracture set) in terms of their distances to another list of polygons (e.g. a primary fracture set or a major fault set). A function to find the intersections of a list of polygons with a borehole line is included in file borehole.C (see definition of class Borehole in header file borehole.h).

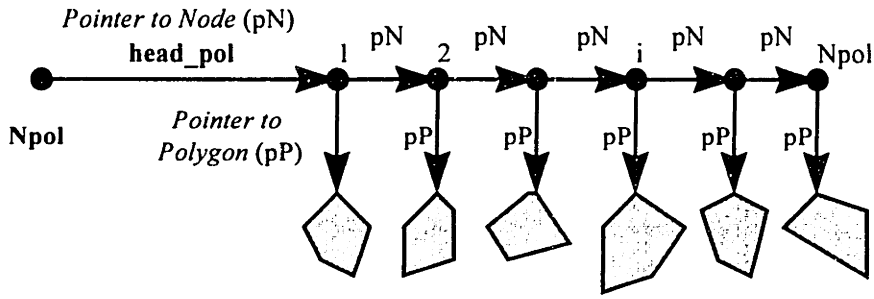


FIGURE A.14 Structure of GEOFRAC class ListPolygons.

Class **Node_frac** is an auxiliary class defined in header file borehole.h. Class Node_frac includes four floating points: **X**, **Y**, **Z**, representing the global Cartesian coordinates of the intersection of a fracture with a borehole, and **t**, representing the relative location of that intersection along the borehole (distance from the end of the borehole assumed to be the beginning end). Class Node_frac also includes a pointer to another Node_frac, **next_frac**, and a pointer to an object of class Polygon, **fracture**. The structure of class Node_frac is illustrated in **Figure A.15**.

Class **Borehole** is defined in header file borehole.h. Class Borehole represents the intersections of a fracture set (class ListPolygons) with a borehole. Class borehole includes an object of class Line, representing a borehole: **log**. Class Borehole also includes an integer, **Nfrac**, indicating the number of fractures intersected by the borehole, and a pointer to class Node_frac, **head_frac**, pointing to the first fracture intersection. Functions to find the intersections of a list of

polygons with a borehole line and to calculate the statistics of the spacing between intersections are included in file borehole.C. The structure of class Borehole is illustrated in Figure A.15.

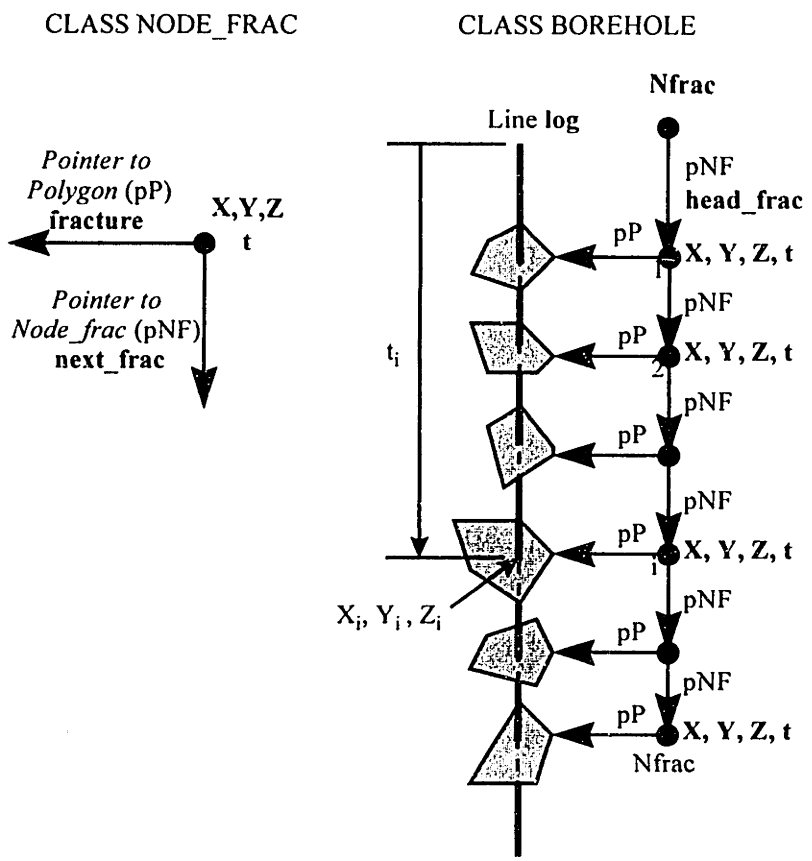


FIGURE A.15 Structure of GEOFRAC classes Node_frac and Borehole.

Classes **Cell** and **Column** are defined in header file cell.h. These two classes are designed specifically for reading Stratamodel data for the Yates case study (Chapter 4) and may not be applicable to other cases. **Figure A.16** illustrates the arrangement of Stratamodel geocells and columns (stacks of geocells). Class Cell represents a Stratamodel geocell, and includes the following floating values: elevation of the geocell center **Z**, porosity reading, **value**, and gammaray response **GR**. Class Cell also includes a pointer to another Cell: **next_cell**. Class Column represents a stack of geocells, and includes: coordinates of the axis of the column, **X** and **Y**, an integer indicating the number of geocells in the column, **Ncells**, and a pointer to an object of class Cell, pointing to the first geocell in the column, **head_cell**. Functions for calculation of the intersections of Stratamodel geocells by a polygon, and marking of a polygon and a list of polygons according to porosity and shale content (based on GR) are included in file cell.C.

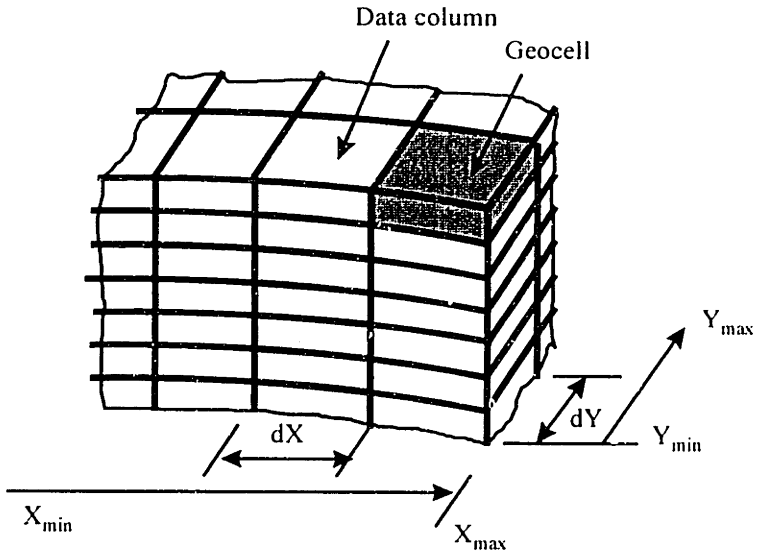


FIGURE A.16 Arrangement of Stratamodel geocells and columns (stacks of geocells) in the Yates field case study (Chapter 4). See Table 4.3.1 for details of the Stratamodel in the case study area.

A.2.2 Input and output files for GEOFRAC modules

The computer program that implements the 3D model, presented in Section A.1 and in Chapter 3, currently has three modules: 1) the general program GEOFRAC; 2) a module FRACTAL that is used for testing the Fractal Line Tessellation (FLT); and 3) program modules for generation of the fracture system around wells in the Yates field case study (Chapter 4): YU1711, YU1755, YU17D5, YU2511 for Tract 17, and YU4007, YU4903, YU5127, YU3728, and YU4959 for Tract 49.

The program modules of GEOFRAC are compiled with the **cygnus** g++ compiler. To use this compiler on Athena, one needs to add the **cygnus** locker and specify the directory path:

```
athena% add -f cygnus
athena% setenv LD_LIBRARY_PATH `athdir /afs/athena/software/cygnus lib`
```

The directory path may change if the file system is reorganized by Project Athena.

The rest of Section A.2.2 presents the structure of input and output files of the three program modules: 1) GEOFRAC, starting on page 399; 2) FRACTAL, starting on page 403; and 3) YU1711, an example of the Yates field case study programs, starting on page 405.

Program GEOFRAC

The following files in directory /mit/1.381_dev/GEOFRAC contain the source code of program GEOFRAC:

cartesian.h	polar.h	point.h	plane.h	surface.h
cubic.h	volume.h	polygon.h	line.h	listpol.h
listline.h	borehole.h	stat.h		
cartesian.C	polar.C	point.C	plane.C	surface.C
cubic.C	volume.C	polygon.C	line.C	listpol.C
listline.C	borehole.C	random.C	divide.C	intial.C
rotation.C	zones.C	intersections.C		(A)

The main routine and the makefile of GEOFRAC are in files `main.C` and `makefile`, respectively. The program can be compiled by typing at the Athena prompt in directory /mit/1.381_dev/GEOFRAC:

athena% make

The command compiles the source files listed in (A) and the `main.C` file into the ready-to-use GEOFRAC program. To use the program, one needs to specify input parameters of the 3D model into the following files:

INPUT
ZONES (only if fracture zones in terms of distances to faults are desired)
FOLD (only if relationship of fracture orientations to a fold are desired)
BOREHOLES (only if borehole intersections of the fracture system are desired)

File INPUT (page 494 in Section A.2.3) has the following structure.

Line 1 and line 2 indicate the dimensions X_m and Y_m of the modeling volume, respectively (see inset 2 in Figure A.2). Line 3 gives the elevation of the datum (Z=0) of the modeling volume.

Lines 4-9 list the coefficients A, B, C, D, E, and F of the quadratic surface that defines the top of the modeling volume. If A=B=C=D=E=0, then F indicates the elevation of a plane that defines the top of the modeling volume. The top surface (defined by Equation A-20) has to be above the datum plane at all points; otherwise the program will produce an error.

Lines 10-11 indicate the coefficients that define the type of polygon marking according to shape and relative size (see Table A.2). The values in line 10 indicate the minimum ratio a_{min} of polygon areas vs. expected polygon area E[A] of the Poisson line tessellation, and the maximum ratio r_{max} of polygon equivalent radii vs. the expected equivalent radius $E[R'_e]$ of "good" polygons; polygons with areas smaller than $a_{min}E[A]$ or radii larger than $r_{max}E[R'_e]$ are discarded. If the

polygon size distribution is not truncated at the small area end, the value of a_{min} is zero. If the polygon size distribution is not truncated at the large area sizes, then a large value (e.g. 100) is entered for r_{max}. The values in line 11 show the corresponding coefficients C_A and γ for the given type of shape-and-relative-size marking (from Table A.2).

Line 12 gives the number of fracture sets.

Line 13 indicates the number of distances, N_d, that define $N_{zones}=N_d+1$ zones of different fracture intensity as a function of fracture distance from faults or primary fractures (see Figure A.8). If N_d is a positive number, file ZONES with fault coordinates must exist in directory /mit/1.381_dev/GEOFRAC. The first row in file ZONES (p. 495 in Section A.2.3) is a list of N_d distances from the faults that define the intensity zones. Zone 1 is from distance zero to the first distance in the row; zone 2 is between the first and the second distances, etc.; zone N_{zones} is at distances larger than the N_d-th distance (see Figure A.8). The second row in file ZONES lists the probabilities P_i (i=1, 2, ..., N_{zones}) for marking of "good" polygons in the zones. The third row in file ZONES gives the number of faults N_{faults} around which the zones are defined. The lines after that specify the coordinates of the faults which are represented by polygons. For every fault, the number of points of the polygon is given first, and then the 3D coordinates (X, Y, Z) of all vertices are listed. The vertices must be ordered, i.e. the coordinates of every vertex must be preceded and followed by the coordinates of vertices to which it is connected; otherwise the program will produce an error. File ZONES on page 495 specifies marking of polygons with probability 1.0 (i.e. all polygons) if they are within 50 m, and with probability 0.05 if they are at a distance more than 50 m, from a fault that is orthogonal to the global axis X and defined by four (ordered) vertices. If file ZONES lists coordinates of primary fractures, produced by GEOFRAC, it essentially defines zones for clustering of secondary fractures.

Line 14 in file INPUT defines whether there is a relationship to a fold. If the character on line 14 is 'y' or 'Y', then file FOLD (see example on p. 495) must exist in directory /mit/1.381_dev/GEOFRAC. The file contains the coefficients A, B, C, D, E, F, G, H, and I of the fold cubic surface, expressed by Equation A-21.

Line 15 defines the type of desired graphical output in this order: 3D fracture system, horizontal outcrop, vertical outcrop. If the first, second, and/or third character is 'y' or 'Y', GEOFRAC will produce a graphical output of the 3D fracture system, a horizontal outcrop, and/or a vertical outcrop, written as MATHEMATICA code in output files FRACTURES.m, PLAN.m, and PROFILE.m, respectively. If these files exist prior to the simulation, their contents are replaced by the new graphical output. To plot the graphics on the screen one should first add the MATHEMATICA locker on Athena, and then run the software, by typing the following lines at the Athena prompt:

athena% add math
athena% math

To plot the contents of a file, type at the MATHEMATICA prompt:

In[1]:= <<*filename.m*

where *filename.m* is, for example, FRACTURES.m or another graphical output file, created by GEOFRAC. In order to perform other operations with the graphical output from GEOFRAC, for example, to save the displayed graphics into PostScript files, and to print them, use another version of MATHEMATICA by typing at the Athena prompt the command:

athena% mathematica

(after adding the MATHEMATICA locker) and following the options in the menu.

Line 16 in file INPUT specifies the type of desired non-graphical output. If the first, second, or/and third character is 'y' or 'Y', GEOFRAC prints coordinates of fractures, horizontal outcrop traces, or/and vertical outcrop traces, in file FRACTURES.txt, PLAN.txt, and PROFILE.txt, respectively. Coordinates of a line include the (X, Y, Z) global coordinates of the two ends. Coordinates of a polygon include the number of points of the polygon, followed by the (X, Y, Z) global coordinates of the ordered vertices of the polygon.

Lines 17 and 18 in file INPUT specify the coefficients A, B, C, and D (Equation A-19) of the vertical (PROFILE) and horizontal (PLAN) outcrop planes, respectively. The planes do not have to be actually vertical and horizontal; the program will produce outcrops on any plane that intersects the modeling volume. If no trace output is specified in lines 15 and 16, GEOFRAC still reads lines 17 and 18, and creates two planes, but does not calculate outcrops on them.

Line 19 specifies the number of desired boreholes, N_{bore}. If N_{bore} is a positive number, file BOREHOLES (see example on p. 495) has to exist in directory /mit/1.381_dev/GEOFRAC. File BOREHOLES contains N_{bore} rows, every one of which defines a boreline in the following order: global X, Y, Z of the first end, X, Y, Z of the second end. Boreholes can have any orientations. When boreholes are specified, GEOFRAC calculates the fracture intersections (ordered from the first end) for every one of them. Two output files are created: BOREHOLES.m and BOREHOLES.txt for graphical and non-graphical output, respectively. If the output files exist, their contents are replaced by the new output. Output file BOREHOLES.m contains MATHEMATICA code for fracture intersections by the specified boreholes. If the file is run at the MATHEMATICA prompt:

In[1]:= <<BOREHOLES.m

a graphical window is created for every borehole to show the intersected fractures. In file BOREHOLES.txt, output for every borehole begins with expression "Next borehole", the coordinates of the two ends, and the length of the borehole, followed by data about every fracture intersection: number from the beginning of the borehole; global X, Y, and Z coordinates of the intersection with the borehole; distance t from the beginning of the borehole; strike, azimuth, dip, area, equivalent radius, and center coordinates X_c, Y_c, and Z_c of the intersected fracture. At the end of every borehole, GEOFRAC prints the mean and standard deviation of the spacing of fracture intersections in that borehole.

Line 20 in file INPUT specifies the number of points for approximation of the curve of intersection between the top quadratic surface of the modeling volume and a plane from the Poisson plane network. If the number is N_p, the curve is approximated by N_p+1 linear segments.

Line 21 specifies the maximum latitude angle ϕ_{max} for partial uniform PDF of plane orientations around the mean pole of Poisson planes. If another type of spherical PDF is specified (see below), this number is entered, but not used by GEOFRAC.

Line 22 specifies the Fisher constant for one-parameter Fisher PDF of plane orientations around the mean pole of Poisson planes. If another type of spherical PDF is specified (see below), this number is entered, but not used by GEOFRAC.

Lines 23 and 24 specify the minimum acceptable angles and maximum acceptable elongation of a "good" polygon (see definition of a "good" polygon in Section A.1). The coefficients C_A and γ for various marking by relative size (Table A.2) are calculated for minimum angle of 1.0472 rad (60°) and maximum elongation of 1.6; these are the values that must be entered in lines 23 and 24 during fracture system simulations. If another angle and elongation are entered, GEOFRAC can be used for calculation of the corresponding coefficients C_A and γ.

Line 25 specifies the relation of fractures to the strike of the fold surface (see Equations A-15-18). If 'c' or 'C' is entered, fractures are rotated to be predominantly concentric to the fold. If 'r' or 'R' is entered, fractures are rotated to be predominantly radially striking from the fold. Line 26 specifies the maximum acceptable angle α_{strike}-$\pi/2$ (α_{strike} is defined in Equation A-15) for radial relationship, and the percent of fractures that must conform to the radial strike requirement. Line 27 specifies the maximal acceptable angle α_{strike} (see Equation A-15) for concentric relationship to the fold strike, and the percent of fractures that must conform to the concentric strike requirement. Line 28 gives the angle $\delta\alpha_{strike}$ of maximal deviation from the strike of the fold for concentric relationship (or from the slope of the fold for radial relationship) when a fracture is rotated according to Equation A-18.

The command compiles the source files listed in (A) and the fractal.C file into the ready-to-use FRACTAL program. To use the program, one needs to specify FLT input parameters in the input file FractalInput (p. 502).

Lines 1 and 2 in file FractalInput specify the half-dimensions X_m and Y_m of the rectangular region where FLT is performed. The origin of the local 2D frame of reference is at the center of the rectangular region which has a total area of $(2X_m)x(2Y_m)=4X_mY_m$.

Line 3 specifies whether marking of "good" polygons according to shape and relative size is performed. If the character is 'y' or 'Y', in every FLT iteration polygons are marked according to shape and relative size. Line 4 specifies the minimum acceptable ratio of polygon areas vs. mean polygon area of the Poisson line tessellation, and the maximum acceptable ratio of polygon equivalent radii vs. mean equivalent radius of "good" polygons (same as line 10 in file INPUT for GEOFRAC; see also Table A.2). Line 5 indicates the coefficient C_A (usually assumed $C_A=1.0$ during test FLT simulations).

Line 6 indicates the type of FLT to be performed: 1, 2, or 3. Type 1 means that in every iteration all polygons are tessellated. Type 2 or type 3 means that only polygons bigger or smaller than a certain size are tessellated. Line 7 indicates the cut-off ratio a_t of areas of polygons to be tessellated vs. the mean polygon area in the iteration. In FLT type 2 only polygons with areas smaller than $a_tE[A']$ are tessellated in every iteration. This algorithms produces at the end numerous small polygons, clustered around few large polygons. In FLT type 3 only polygons with areas larger than $a_tE[A']$ are tessellated in every iteration. This algorithms produces at the end polygons with very similar sizes. For example, lines 6 and 7 in file FractalInput (p.502) indicate that in every iteration only polygons with areas smaller than twice the mean polygon area are tessellated.

Line 8 shows the number of FLT iterations, N_{FLT}, the fractal dimension F, and the probability P of marking polygons as fractured at every iteration. Fractal dimension F is the ratio of expected polygon areas in two consecutive iterations.

Lines 9 and 10 specify the minimum acceptable angles and maximum acceptable elongation of a "good" polygon (see definition of a "good" polygon in Section A.1). The test FLT simulations are performed for minimum angles 1.0472 rad (60°) and maximum elongation of 1.6.

Line 11 indicates the number of size intervals, N_{size}, for calculation of the size PDF of generated polygons. After this value, starting on line 12, N_{size} values indicate the upper limits of the size intervals, as ratios of the mean polygon area in the simulation. FRACTAL creates the output file FractalSize.txt in which the size PDF is listed as pairs of upper interval limit, and number of polygons in the size interval defined by this and the preceding upper interval limit.

Line 29 specifies the relation of fractures to the dip of the fold surface. If 'd' or 'D' is entered, fractures are rotated to be predominantly subparallel to the fold dip. Line 30 specifies the maximal acceptable angle α_{dip} (defined in Equation A-16), and the percent of fractures that must conform to the dip requirement. Line 31 gives the angle $\delta\alpha_{dip}$ of maximal deviation from the dip of the fold when a fracture is rotated according to Equation A-17.

Line 32 indicates the number of size intervals, N_{size}, for calculation of the size PDF of generated polygons. After this value, starting on line 33, N_{size} values indicate the upper limits of the size intervals, as ratios of the mean polygon area in the simulation. GEOFRAC creates the output file `Size.txt` in which the size PDF is listed as pairs of (1) upper interval limit, and (2) number of polygons in the size interval defined by this and the preceding upper interval limit.

Lines 34-38 give parameters for the first fracture set (the number of fracture sets is given in line 12). Line 34 indicates the fracture intensity P_{32} of the set. Line 35 gives the global polar orientation of the mean pole of fracture planes in the set: azimuth, and latitude. Line 36 indicates the type of desired PDF of fracture plane orientations: 'f' for Fisher, 'c' for constant, 'u' for 'uniform, 'p' for partial uniform. Line 37 gives the coefficient C for translation of polygons (Equation A-12). Line 38 gives the expected value of the fracture (polygon) equivalent radius for the first set.

If more than one fracture set is modeled, for every set five lines, similar to lines 34-38, have to be entered. GEOFRAC reads as many groups of five lines as the number entered in line 12; if the number of five-line groups of fracture set data is smaller than the number in line 12, the program will produce an error.

Besides the specific output files described above, GEOFRAC also appends a summary of every simulation to file `Frac.txt`. The summary includes the number of fracture sets, and for every set: input parameters, Poisson plane intensity, Poisson line intensity, calculated P_{32} (to be compared to the input P_{32}), calculated mean fracture area, etc.

Program FRACTAL

Program FRACTAL is used for test simulations of the Fractal Line Tessellation (discussed in Section A.1 and in Section 3.4). The main routine and the makefile of FRACTAL are in files `fractal.C` and `fractal.mak`, respectively. The program can be compiled by typing at the `Athena` prompt in directory /mit/1.381_dev/GEOFRAC:

```
athena% make -f fractal.mak
```

The last line in file FractalInput gives the mean equivalent radius of polygons $E[R_e]$ created by the first FLT iteration (i.e. by a regular Poisson line tessellation). The equivalent radius in every subsequent iteration decreases, and is printed in the output file FractalFrac.txt (see next paragraph).

Program FRACTAL creates the graphical output file Fractal.m (MATHEMATICA code) with the coordinates of all fractures created by FLT. The fracture system can be displayed with MATHEMATICA (following the instructions for displaying GEOFRAC graphics). FRACTAL also creates the output file FractalFrac.txt and prints a summary of the FLT simulation in it. The summary includes mean and standard deviation of polygon areas, mean of polygon equivalent radii, number of polygons and their cumulative area in every iteration.

Example program for the Yates field: YU1711

Modules of program GEOFRAC were created for simulation of the fracture system around logged wells in Tract 17 and Tract 49 in the Yates field. Every module program has the name of the corresponding well: YU1711, YU1755, YU17D5, and YU2511 for Tract 17; YU4007, YU4903, YU5127, YU3728, and YU4959 for Tract 49. This section explains the structure of input and output files for one of the programs: YU1711. The files of the other module programs are similar; to find their names, simply substitute "1711" with the number of another logged well. The Yates field programs have been compiled on the Sun SPARCstations and on the SGI workstations in the MIT Athena environment. However, it is recommended to compile the programs on SGIs since they can run more extensive simulations.

The main routine and the makefile of YU1711 are in files 1711main.C and YU1711.mak, respectively. Besides the files of the general GEOFRAC program, listed in (A), two other files must exist in directory /mit/1.381_dev/GEOFRAC: cell.h and cell.C. Program YU1711 can be compiled by typing at the Athena prompt in directory /mit/1.381_dev/GEOFRAC:

athena% make -f YU1711.mak

The command compiles the source files listed in (A), files cell.h and cell.C, and the 1711main.C file into the ready-to-use YU1711 program. To use the program, one needs to specify input parameters of the 3D model into the following files:

1711input.dat
1711SGM.dat (only for wells YU1711, YU1755, YU17D5, YU4007, YU4903, and YU5127 where Stratamodel data are available)

File 1711input.dat (p. 517-518) has the following structure.

Line 1 and line 2 indicate the half-dimensions X_m and Y_m of the modeling volume, respectively (see inset 2 in Figure A.2). For all vertical wells $X_m=Y_m=300$ ft, i.e. simulations are performed in a 600 ft x 600 ft area centered at the logged well. For the 1550 ft long, north-trending, horizontal well YU17D5, $X_m=200$ ft, and $Y_m=800$ ft, i.e. simulations are performed in an area of 400 ft x 1600 ft. Line 3 gives the elevation of the datum (Z=0) of the modeling volume. For all vertical wells the datum is at elevation 700 ft; for the horizontal well YU17D5 the datum is at elevation 800 ft (the lower boundary of the Yates Field Unit).

Lines 4-9 list the coefficients A, B, C, D, E, and F of the quadratic surface that defines the top of the modeling volume (Equation A-20). The coefficients of the quadratic surface were obtained though polynomial fit with MATLAB to observed elevations of the San Andres top in wells in Tract 17 and Tract 49. The top surface of the modeling volume is assumed at 100 ft above the top of the San Andres formation for vertical wells, and at 100 ft below the San Andres top for the horizontal well YU17D5.

Lines 10-19 contain the coefficients A, B, C, D, E, F, G, H, and I of the fold cubic surface (Equation A-21) to which fractures in the Yates field are related by strike. The coefficients were obtained through polynomial fit with MATLAB to observed elevations of the Seven Rivers M-horizon in wells in Tract 17 and Tract 49. The implemented cubic surface of the type $Z=F(X^3, X^2, X, Y^3, Y^2, Y)$ defines a parallel fold, i.e. the fold curvature at all elevations at a given (X, Y) location is the same.

Lines 20-21 indicate the coefficients that define the type of polygon marking according to shape and relative size (see Table A.2). The values in line 20 indicate the minimum ratio a_{min} of polygon areas vs. the expected polygon area E[A] of the Poisson line tessellation, and the maximum ratio r_{max} of polygon equivalent radii vs. the expected equivalent radius of "good" polygons $E[R'_e]$; polygons with areas smaller than $a_{min}E[A]$ or equivalent radii larger than $r_{max}E[R'_e]$ are discarded. The values in line 21 show the corresponding coefficients C_A and γ for the given type of shape-and-relative-size marking (from Table A.2). For wells in Tract 17, $a_{min}=0$, $r_{max}=3$, $C_A=1.8$, and $\gamma=0.38$. For wells in Tract 49, $a_{min}=1$, $r_{max}=3$, $C_A=3.6$, and $\gamma=0.36$.

Line 22 gives the number of fracture sets, defined by regional stresses. For all wells in the Yates field, there are two fracture sets.

Lines 23 and 24 specify the coefficients A, B, C, and D (Equation A-19) of a vertical and a horizontal outcrop plane, respectively. The planes do not have to be actually vertical and horizontal; the program will produce outcrops on any plane that intersects the modeling volume. Fracture trace outcrops in the Yates field can be generated only to illustrate the modeled relation of fractures to the drape fold. Since there are no mapped trace outcrops, the simulated outcrops cannot be compared to actual ones. The calculation of outcrop traces is usually disabled

during simulations of the fracture system in the Yates reservoir (by commenting code lines in the files with the main routines, i.e. YU1711main.C, etc.). Also, the code for output (graphical or not) of the coordinates of all fractures around the wells is usually commented; because of the large number of simulated fractures, usually only statistics of the generated fracture populations are printed in output files.

Line 25 defines the desired borehole in the following order: global X, Y, Z of the first end, X, Y, Z of the second end, where Z is the elevation above the datum of the modeling volume. For vertical wells, X=0 and Y=0 for both ends, i.e. the simulated borehole coincides with the axis of the modeling volume which is along the actual field well. For example, the values in line 25 in file 1711input.dat (p. 518) indicate that the simulated well is at the center of the modeling volume, from 100 ft to 350 ft above datum of 700 ft , i.e. from elevation 800 ft to 1050 ft above the sea level. Thus the simulated well coincides with the actual well YU1711. Two output files with simulated borehole data are created: 1711bore.m and 1711bore.txt for graphical and non-graphical output, respectively. Output file 1711bore.m contains MATHEMATICA code for fracture intersections by the specified boreholes. If the file exists, then it is overwritten by the graphical output from a new simulation. A plot with intersected fractures can be shown on the screen by typing at the MATHEMATICA prompt:

In[1]= <<1711bore.m

In file 1711bore.txt, the coordinates of the two ends, and the length of the borehole are written first, followed by data about every fracture intersection: number from the beginning of the borehole; global X, Y, and Z coordinates of the intersection with the borehole; distance t from the beginning of the borehole; strike, azimuth, dip, area, equivalent radius, and center coordinates X_c, Y_c, and Z_c of the intersected fracture. At the end of every borehole, YU1711 prints the mean and standard deviation of the spacing of fracture intersections in the borehole. The results from a new simulation are appended to an existing file 1711bore.txt.

Line 26 in file 1711input.dat specifies the number of points for approximation of the curve of intersection between the top quadratic surface of the modeling volume and a plane from the Poisson plane network. If the number is N_p, the curve is approximated by N_p+1 linear segments.

Line 27 specifies the maximum latitude angle ϕ_{max} for partial uniform PDF of plane orientations around the mean pole of Poisson planes. If another type of spherical PDF is specified (see below), this number is entered, but not used by YU1711.

Line 28 specifies the Fisher constant for one-parameter Fisher PDF of plane orientations around the mean pole of Poisson planes. If another type of spherical PDF is specified (see below), this number is entered, but not used by YU1711.

Lines 29 and 30 specify the minimum acceptable angles and maximum acceptable elongation of a "good" polygon (see definition of a "good" polygon in Section A.1). The coefficients C_A and γ for various marking by relative size (Table A.2) are calculated for minimum angle of 1.0472 rad (60°) and maximum elongation of 1.6; these values must be entered in lines 29 and 30.

Line 31 specifies the relation of fractures to the strike of the fold surface (see equations A-15-17). If 'c' or 'C' is entered, fractures are rotated to be predominantly concentric to the fold. If 'r' or 'R' is entered, fractures are rotated to be predominantly radially striking from the fold. Line 32 specifies the maximum acceptable angle α_{strike}-$\pi/2$ (α_{strike} is defined in Equation A-15) for radial relationship, and the percent of fractures that must conform to the radial strike requirement. Line 33 specifies the maximum acceptable angle α_{strike} (see Equation A-15) for concentric relationship to the fold strike, and the percent of fractures that must conform to the concentric strike requirement. Line 34 gives the angle $\delta\alpha_{strike}$ of maximum deviation from the strike of the fold for concentric relationship (or from the slope of the fold for radial relationship) when a fracture is rotated according to Equation A-18.

Line 35 indicates the number of size intervals, N_{size}, for calculation of the size PDF of generated polygons. After this value, starting on line 36, N_{size} values indicate the upper limits of the size intervals, as ratios of the mean polygon area in the simulation. YU1711 creates the output file 1711size.txt in which the size PDF is listed as pairs of upper interval limit, and number of polygons in the size interval defined by this and the preceding upper interval limit.

Lines 37-41 specify how to incorporate Stratamodel data into the simulations of the fracture system in the vicinity of the logged wells. Character 'y' or 'Y' on line 37 indicates that Stratamodel data should be considered in the simulation. Line 38 shows how many columns of Stratamodel geocells are read for the well (56 for well YU1711; see Figure A.16). Line 39 gives the cut-off value of porosity: 0.2 (20%). The fracture intensity is assumed to be lower in dolomite with average porosity n>0.2. Line 40 gives the cut-off value of gammaray response: 40 API (shale content indicator, GR=40 API corresponds to 10% shale). Line 40 exists only in the input files for Tract 17 wells where fracture intensity is assumed to be lower in dolomite with average shale content higher than 10% (GR=40 API). Line 41 indicates the marking probability for good polygons in zones of high porosity or high shale content. For example, the value of 0.5 on line 41 in file 1711input.dat (p. 518) indicates that fractures ("good" polygons) are marked with probability P=0.5 (i.e. 50% of them are discarded) if the rock matrix around them has porosity n>0.2 or GR>40 API.

If line 37 in file 1711input.dat reads 'y' or 'Y', then file 1711SGM.dat must exist in the directory /mit/1.381_dev/GEOFRAC. File 1711SGM.dat (first page is shown on p. 519) includes the Stratamodel porosity and shale readings, arranged in columns (Figure A.16). Every column begins with X and Y coordinates of the column axis (measured in feet from the center of the modeling volume for the well). After that the geocells in the column are listed as a list of three values: elevation (in feet), porosity, and GR (for wells in Tract 49, only elevation and porosity are given).

Lines 42-46 give parameters for the first fracture set. Line 42 indicates the fracture intensity P_{32} of the set. Line 43 gives the global polar orientation of the mean pole of fracture planes in the set: azimuth N40°E, and latitude 90° (p.518). Line 44 indicates the type of desired PDF of fracture plane orientations: 'f' for Fisher. Line 45 gives the coefficient C for translation of polygons (Equation A-12). Line 46 gives the expected value of the fracture (polygon) equivalent radius for the first set: 30 ft. Lines 47-51 give the same information (as lines 42-46) for the second fracture set.

Besides the specific output files described above, YU1711 also appends a summary of every simulation to file 1711frac.txt. For each fracture set, the summary includes: input parameters (P_{32}, expected radius and area of fractures, etc.), Poisson plane and line intensities, calculated P_{32} (to be compared to the input P_{32}), calculated mean fracture area and equivalent radius, etc.

A.3 GEOFRAC **SOURCE CODE**

The source files of GEOFRAC and its modules are appended in the following order:

- Header files (pages 411-425):
  - cartesian.h
  - polar.h
  - point.h
  - plane.h
  - surface.h
  - cubic.h
  - volume.h
  - polygon.h
  - line.h
  - listpol.h
  - listline.h
  - borehole.h
  - stat.h
- Source files (pages 426-482):
  - cartesian.C

polar.C
point.C
plane.C
surface.C
cubic.C
volume.C
polygon.C
line.C
listpol.C
listline.C
borehole.C
random.C
divide.C
initial.C
rotation.C
zones.C
intersections.C

- GEOFRAC files (pages 483-495):
  makefile
  main.C
  INPUT
  ZONES
  FOLD
  BOREHOLES
- FRACTAL files (pages 496-502):
  fractal.mak
  fractal.C
  FractalInput
- Files of the Yates field case study programs (pages 503-530):
  cell.h
  cell.C
  YU1711.mak
  YU1711main.C
  1711input.dat
  1711SGM.dat (only first page of the file)
  1755input.dat
  17D5input.dat
  2511input.dat
  4007input.dat
  4903input.dat
  5127input.dat
  3728input.dat
  4959input.dat

File **cartesian.h**

```
#ifndef _CARTESIAN_H
#define _CARTESIAN_H

#include <math.h>
#include <iostream.h>
#include <fstream.h>
#define UNIT_VECTOR Cartesian;
#define VECTOR Cartesian
#define FALSE 0
#define TRUE 1
#define boolian int
#include "polar.h"

class Cartesian
{ public:
   double X, Y, Z;
   Cartesian (void)
      {X=0.0; Y=0.0; Z=0.0;};
   Cartesian (double x,double y,double z)
      {X = x; Y = y; Z = z;};
   friend class Polar;
  Cartesian (Polar& pole)
   { X=sin(pole.phi)*sin(pole.theta);
     Y=sin(pole.phi)*cos(pole.theta);
     Z=cos(pole.phi);   };

 Cartesian& operator = (Cartesian  c)
     {X = c.X;  Y = c.Y; Z = c.Z;   return (*this); };
  boolian operator== (Cartesian& c)
    {   if (X==c.X && Y==c.Y && Z==c.Z)
        return (TRUE);
        return (FALSE);};
  double operator* (Cartesian& c)      // Dot product
    {double product = X*c.X+Y*c.Y+Z*c.Z;
     return product; };

    Cartesian cross (Cartesian& c)     // Cross product: LEFT handed system
    {double xx = Y*c.Z - Z*c.Y;
     double yy = Z*c.X - X*c.Z;
     double zz = X*c.Y - Y*c.X;
     Cartesian* C = new Cartesian (xx, yy, zz);
     return *C;};

 friend ostream& operator<< (ostream& o, Cartesian& c)
  {return o <<c.X<<" "<<c.Y<<" "<<c.Z<<endl;};

 Polar convert_to_polar ();
 Cartesian local_coordinates (Polar&);
 Cartesian global_coordinates (Polar&);};

#endif _CARTESIAN_H
```

File **polar.h**

```
#ifndef _POLAR_H
#define _POLAR_H

#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)

#include <iostream.h>
#include <fstream.h>
#define FALSE 0
#define TRUE 1
#define boolian int

class Cartesian;

class Polar
{
 public:
   double theta, phi;
   Polar (void)                  // Default constructor
     {theta=0.0;
      phi=0.0; };
   Polar (double angle1, double angle2)   // Constructor from two angles
     {theta=angle1;
      phi=angle2;
     };

boolian operator== (Polar& p)
 {
  if (theta == p.theta && phi == p.phi)
    return TRUE;
    return FALSE;
 };

friend ostream& operator<< (ostream& o ,Polar& p)
 { return o << p.theta <<" "<< p.phi<<endl;};

Polar& operator= (Polar p)
      {
       theta = p.theta;
       phi = p.phi;
       return (*this);
      };
~Polar (void) {};              // Default destructor
Cartesian convert_to_cartesian ();
 };

#endif _POLAR_H
```

File **point.h**

```
#ifndef _POINT_H
#define _POINT_H
#include <math.h>
#include <iostream.h>
#include <fstream.h>
#include "cartesian.h"

#define TRUE 1
#define FALSE 0
#define boolian int
#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)
extern double Datum;

class Point: public Cartesian
{
  public:  Point* next;

Point (void)              //default constructor
 { X=0.0; Y=0.0; Z=0.0;  next = 0; };

Point (double x, double y, double z)   //constructor for 3d point
  {X=x; Y=y; Z=z;
    if (X <=0.000001 && X>=-0.000001)
          X=0.;
    if (Y <=0.000001 && Y>=-0.000001)
              Y=0.;
    if (Z <=0.000001 && Z>=-0.000001)
              Z=0.;
      next = 0;};

~Point () {};       // default destructor

Point& operator= (Point p)
 {X = p.X; Y = p.Y; Z = p.Z;  return (*this);  };
boolian operator==(Point& p)
 { if (X == p.X && Y == p.Y && Z == p.Z)
        return (TRUE);
        return (FALSE); };
void operator+ (Cartesian& c)     // Addition of a vector: translation
   {X+=c.X; Y+=c.Y; Z+=c.Z;  return; };
void operator- (Cartesian& c)     // Substraction of a vector:
   {X-=c.X; Y-=c.Y; Z-=c.Z;  return; };
friend ostream& operator<< (ostream& o , Point& p)
 {o <<p.X<<" "<<p.Y<<" "<<p.Z+Datum<<endl; return o;};

void print();
};

#endif _POINT_H
```

File **plane.h**

```
#ifndef _PLANE_H
#define _PLANE_H

#include "polar.h"
#include "cartesian.h"
#include "point.h"
#include <math.h>
#include <iostream.h>

#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)

// Class Plane is used for the planes of tension and shear generated according to a spherical PDF.
// A plane in 3D is described by the equation Ax+By+Cz=D. This plane is  perpendicular to vector
// (A, B, C) and passes through point (X,Y,Z),  where A*X+B*Y+C*Z=D.

class Plane
{  public:
   double A, B, C, D, strike, dip;   Polar MeanPole;
    Polar rel_polar, abs_polar;          // Normal vector in polar coordinates
    Cartesian rel_cart, abs_cart;        // Normal vector in cartesian coord.

    Plane (void)                  // default constructor: xy plane
    {MeanPole.theta=0.; MeanPole.phi=0.;
     rel_polar.theta = 0.0; rel_polar.phi = 0.0; abs_polar.theta = 0.; abs_polar.phi = 0.;
     rel_cart.X = 0.0;  rel_cart.Y = 0.0;  rel_cart.Z = 1.0;
     abs_cart.X = 0.0;  abs_cart.Y = 0.0;  abs_cart.Z = 1.0;
     A=0.; B=0.; C=1.; D=0.;   strike =0.; dip = 0.; };

Plane (double, double, double, double);      //constructor from A, B, C, D
Plane (Cartesian& , Point&, Polar&);      // abs_cart is given by vector N (A,B,C).
                                          // Constructor from N and point P (D=A*X+B*Y+C*Z)
Plane (Polar&, Polar&, Point&);             // MeanPole is given by first Polar;
                                  // rel_cart is given by second Polar; Point is in absolute f.o.r.
Plane (char, Polar&, double );       //constructor of plane according to an orientation PDF (char)

~Plane (void) {};            //Default destructor

Cartesian find_binormal();          // Calculates a vector which is perpendicular to both strike and
                                    // dip of the plane and points upward.
friend ostream& operator<< (ostream&, Plane&);
Plane& operator= (Plane p)
 {[  A = p.A; B = p.B; C = p.C; D = p.D;
     abs_cart = p.abs_cart;  abs_polar = p.abs_polar;
     rel_cart = p.rel_cart;  rel_polar = p.rel_polar;
     MeanPole = p.MeanPole; strike = p.strike; dip = p.dip;
      return (*this);  };
};

#endif _PLANE_H
```

File **surface.h**

```
#ifndef _SURFACE_H
#define _SURFACE_H
#include <math.h>
#include <iostream.h>
#include <fstream.h>
#include "polar.h"
#include "cartesian.h"
#include "point.h"
#include "line.h"

#define TRUE 1
#define FALSE 0
#define boolian int

class Line;

// Class of surface is defined by quadratic equation  z = Ax^2 + Bxy + Cy^2 + Dx + Ey + F
// This class is used for topographic surface and for internal surfaces for example shale layers

class Surface
{ public:
  double A, B, C, D, E, F;

Surface()                 //Default constructor
  { A=0.; B=0.; C=0.; D=0.; E=0.; F=0.;};

Surface (double a, double b, double c, double d, double e, double f)
  { A=a; B=b; C=c; D=d; E=e; F=f;};

friend ostream& operator<< (ostream& o , Surface& s)
 { o <<s.A<<" "<<s.B<<" "<<s.C<<" "<<s.D<<" "<<s.E<<" "<<s.F<<endl;  return o; };

Surface& operator= (Surface s)
 { A=s.A; B=s.B; C=s.C; D=s.D; E=s.E; F=s.F;  return (*this); };

~Surface () {};               //Default destructor

boolian is_point_on_surface(Point&);
boolian is_line_on_surface (Line&);
Cartesian normal_at_point(Point&);
Polar polar_normal_at_point (Point&);
double find_Z_on_surface (double, double);
Polar strike_dip_at_point (Point&);
double find_enclosed_volume (double, double);
double Zmax_over_XY(double, double);
int is_point_above_below (Point&);
int how_line_intersect (Line& L);
Point intersection_by_line (Line&);
};

#endif _SURFACE_H
```

File **cubic.h**

```
#ifndef _CUBIC_H
#define _CUBIC_H
#include <math.h>
#include <iostream.h>
#include <fstream.h>
#include "polar.h"
#include "cartesian.h"
#include "point.h"

#define TRUE 1
#define FALSE 0
#define boolian int


// Class of cubic surface is defined by equation
// z = Ax^3 + Bx^2y + Cxy^2 + Dy^3 + Ex^2 + Fxy + Gy^2 + Hx + Iy +J;
// This class is used for fold surfaces.

class Cubic
{
public:
double A, B, C, D, E, F, G, H, I, J;

Cubic()                 //Default constructor
  { A=0.; B=0.; C=0.; D=0.; E=0.; F=0.; G=0.; H=0.; I=0.; J=0.;};

Cubic (double a, double b, double c, double d, double e, double f, double g, double h, double i, \
double j)
  { A=a; B=b; C=c; D=d; E=e; F=f; G=g; H=h; I=i; J=j;};

friend ostream& operator<< (ostream& o , Cubic& c)
 {
 o<<c.A<<" "<<c.B<<" "<<c.C<<" "<<c.D<<" "<<c.E<<" "<<c.F<<" "<<c.G<<" "<<c.H<<\
 " "<<c.I<<" "<<c.J<<endl;
return o;  };

Cubic& operator= (Cubic c)
 { A=c.A; B=c.B; C=c.C; D=c.D; E=c.E; F=c.F; G=c.G; H=c.H; I=c.I; J=c.J;
  return (*this);  };

~Cubic () {};               //Default destructor

Cartesian normal_at_point(Point&);
Polar polar_normal_at_point (Point&);
Polar strike_dip_at_point (Point&);

};

#endif _CUBIC_H
```

File **volume.h**

```
#ifndef _VOLUME_H
#define _VOLUME_H
#include <math.h>
#include <iostream.h>
#include <fstream.h>
#include "polar.h"
#include "cartesian.h"
#include "point.h"
#include "surface.h"

#define TRUE 1
#define FALSE 0
#define boolian int


// Class Volume is used for calculation of the modeling volume for fracture sets enclosed
// between the topographic surface and five planes (horizontal and four vertical). The horizontal
// bottom plane z=0 is at elevation Datim. The vertical planes are at x=Xm, x=-Xm, y=Ym,
// and y=-Ym. The origin (0,0,0) of the coordinate system is set to to the center of the bottom
plane.
// The four points P[i] are the points in the corners of the volume on the top surface.

class Surface;

class Volume
{ public:
  Point P[4];
  double volume;
  Surface top;
  double Zmax;

Volume (Surface& ground);  // Constructor

~Volume () {};       //Default destructor

friend ostream& operator<< (ostream& o, Volume& v)
  { int i;
    for (i=0; i<4; ++i)
      o << v.P[i];
      o <<"volume: "<<v.volume<<" Zmax: "<<v.Zmax<<endl;
      o <<"top surface: "<<v.top<<endl;  };

Point random_point();
boolian is_point_inside(Point&);

};

#endif _VOLUME_H
```

File **polygon.h**
(page 1 / 2)

```
#ifndef _POLYGON_H
#define _POLYGON_H

#include <iostream.h>
#include <fstream.h>
#include <math.h>
#include "point.h"
#include "line.h"
#include "plane.h"
#include "volume.h"
#include "surface.h"
#include "cell.h"

#define boolian int
#define TRUE 1
#define FALSE 0


class Polygon
{
 public :
    Polar setPole;          // in global f.o.r
    Polar Pole;             // in global f.o.r.
    double strike, dip;
    Point* head;
    int noP;
    Point center;
    double radius, area;

    Polygon () {                    // Default constructor
     setPole.phi =0.; setPole.theta=0.;
     Pole.theta=0.; Pole.phi=0.;
     head=0; noP=0;
     strike =0.; dip =0.;
     center.X = 0.; center.Y = 0.; center.Z =0.;
     radius =0.; area = 0.;};

Polygon (Plane&);       // Constructor for polygon parallel to

friend ostream& operator<< (ostream& o , Polygon& pol)
 { Point* marker = pol.head; int i;
    for (i=0; i < pol.noP; ++i)
    {o << (*marker);
     marker = marker -> next; };
 o<<pol.area<<" "<<pol.radius<<" "<<pol.center;
 return o; };

void operator+ (Cartesian &);       // Translation
void operator- (Cartesian &);       // Translation back
```

File **polygon.h**
(page 2 / 2)

```
boolian is_it_member (Point& );
void add_point (Point&);
void find_area_radius_2d ();              // for polygon in 2d
void area_radius_3d ();                   // for 3d polygon
void find_center ();                      // for 3d polygon
void make_polygon_2d ();
void make_polygon_3d ();
void sort_points_2d ();                   // for polygon in 2d
void print();
void add_points_on_surface (Plane&, Surface& , int, double Zm);
boolian if_good_shape (double, double);
double minR_inscribe ();
int how_line_intersects (Line&);          // in 2d
boolian if_line_intersects(Line& );       // in 2d
Polygon divide_by_line (Line&);           // in 2d
double find_elongation ();
boolian if_intersects_plane(Plane& );
Line intersection_with_plane (Plane& );
void translate_2d (double, double);
boolian mark_parallel_to_strike (double, double, double);
boolian mark_orthogonal_to_strike (double, double, double);
boolian mark_by_dip (double, double, double);
void rotate_by_strike (double);
void rotate_by_dip (double);
int above_or_below_surface (Surface& );
void cut_by_surface (Surface&, char);
double distance_from_point (Point& );     // in 3d
double distance_from_polygon (Polygon& ); // in 3d
boolian if_3d_line_intersects (Line& );   // in 3d
Point intersection_with_line (Line&);     // in 3d
boolian if_polygon_intersects (Polygon& );
double find_average_porosity (Column&);
double find_average_GR (Column& C);
boolian mark_by_porosity (Column*, int, double, double);
boolian mark_by_porosity_and_GR (Column*, int, double, double, double);
int find_closest_column (Column* , int);
void include_in_size_pdf (int, int*, double*, double);

~Polygon () {};

};

#endif _POLYGON_H
```

File **line.h**
(page 1 / 2)

```
#ifndef _LINE_H
#define _LINE_H

#include <iostream.h>
#include <fstream.h>
#include <math.h>
#include "cartesian.h"
#include "point.h"
#include "plane.h"

#define TRUE 1
#define FALSE 0

// Class Line represents a LINE SEGMENT between 2 3D points P1(x1, y1, z1)  and P2(x2, y2,
z2).
// The line vector is (x2-x1, y2-y1, z2-z1).  Parametric equation is
// t=(X-x1)/(x2-x1)=(Y-y1)/(y2-y1)=(Z-z1)/(z2-z1) for point P(X,Y,Z) on the line.
// If t=[0,1] P is on the segment P1P2 (for t=0, P=P1; for t=1 P=P2).
// If t=(-infinity, +infinity), point (X, Y, Z) is on the infinite 3D line.

class Line
{
 public:
  Point end1, end2;
  Cartesian vector;
  double length;

Line (void)                 //default constructor
  {
    end1.X=0.0;              //line segment from (0,0,0) to (1,1,1)
    end1.Y=0.0;
    end1.Z=0.0;
    end2.X=1.0;
    end2.Y=1.0;
    end2.Z=1.0;
    vector.X =1.0; vector.Y =1.0; vector.Z =1.0;
    length = 0.; };

Line (Point& p1, Point& p2)          //constructor from 2 points;
  {
    end1 = p1;
    end2 = p2;
    vector.X = end2.X - end1.X;
    vector.Y = end2.Y - end1.Y;
    vector.Z = end2.Z - end1.Z;
    length = sqrt(vector.X*vector.X+vector.Y*vector.Y+vector.Z*vector.Z);
    vector.X/=length; vector.Y/=length; vector.Z/=length;
  };
```

File **line.h**
(page 2 / 2)

```
Line (Point& p, Cartesian& c, double l)
 { end1.X=p.X;              // constructor for line trough point P
   end1.Y=p.Y;              // parallel to a vector c and with length l
   end1.Z=p.Z;
   double cl = sqrt (c.X*c.X + c.Y*c.Y + c.Z*c.Z);
   if (c.X || c.Y || c.Z)
    {vector.X=c.X/cl; vector.Y=c.Y/cl; vector.Z=c.Z/cl;};
   end2.X=end1.X+l*vector.X;
   end2.Y=end1.Y+l*vector.Y;
   end2.Z=end1.Z+l*vector.Z;
   length = l;  };

Line (double angle, double D, double R, Point& C);
                // Constructor for a 2D line with equation
                // X*cos(angle) + Y*sin(angle) = D intersecting
                // the  circle with radius R and center C

~Line () {};                    //destructor

boolian intersect (Line& );  //if a 2D line intersects another 2D line
Point intersection_with_line(Line& );
boolian is_point_on_line (Point&);

double angle_with_line (Line& L)            // angle between two lines in 3D
 {double angle = acos (vector*L.vector);
   return angle;};

boolian if_intersects_plane (Plane&);
Point intersection_with_plane (Plane&);
void global_coordinates (Polar&);
void local_coordinates (Polar&);

friend ostream& operator<< (ostream& o , Line& l)
{ o <<l.end1.X<<" "<<l.end1.Y<<" "<<l.end1.Z<<endl;
  o <<l.end2.X<<" "<<l.end2.Y<<" "<<l.end2.Z<<endl;
  o <<"vector "<<l.vector<<" length "<<l.length<<endl;
  return o;  };

void print();

boolian operator==(Line& l)
 {   if ((end1 == l.end1 && end2==l.end2)||(end2==l.end1 && end1==l.end2))
       return (TRUE);
       return (FALSE);   };
};

#endif _LINE_H
```

File **listpol.h**
(page 1 / 2)

```
#ifndef _LISTPOL_H
#define _LISTPOL_H
#include <iostream.h>
#include <fstream.h>
#include <math.h>
#include "polygon.h"
#include "line.h"
#include "listline.h"
#include "surface.h"
#include "cubic.h"
#include "stat.h"
#include "borehole.h"
#include "cell.h"

class Node
{ public:
  Node* next_pol;
  Polygon* content; };

class ListPolygons
{ public:
  Node* head_pol;
  int Npol;
  ListPolygons ()        // default constructor
  { head_pol = 0; Npol = 0; };

void add_polygon  (Polygon&);
friend ostream& operator<< (ostream& , ListPolygons&);
void print();
void intersect_by_line (Line& );
void make_listpol_2d ();
void make_listpol_3d();
void find_area_radius_2d ();
void mark_good_shape (double, double);
void mark_good_shape_and_P (double, double, double );
double find_mean_area();
double find_SD_area ();
void add_listpol (ListPolygons& );
void fractal_tessellation (double, double);
void fractal_big_and_small (double, double);
void fractal_similar (double, double);
ListLines traces_on_plane (Plane& );
Stat find_mean_sd (double, double);
Stat find_mean_sd ();
void translate_2d (double, double);
void mark_parallel_to_strike (Surface&, double, double);
void mark_orthogonal_to_strike (Surface&, double, double);
void mark_by_strike (Surface&, double, double, double, char, double, double);
void mark_by_strike (Cubic&, double, double, double, char, double, double);
void mark_by_dip (Surface&, double, double, double);
```

File **listpol.h**
(page 1 / 2)

```
void mark_by_dip (Cubic&, double, double,  double);void cut_by_surface (Surface&, char);
void cut_between_two_surfaces (Surface&, Surface&);
void cut_outside_two_surfaces (Surface&, Surface&);
double shortest_distance_from_polygon (Polygon& );
boolian if_zone_mark_polygon (Polygon&, int, double*, double*);
void mark_by_zones (ListPolygons& , int, double* , double*);
Borehole intersections_with_logline (Line&);
void ListPolygons :: mark_to_surface (Surface& , double , char );
void discard_high_porosity (Column*, int, double, double);
void discard_shale (Column*, int, double, double, double);
void size_distribution (int, int*, double*, double);
~ListPolygons () {};    // default destructor
};

#endif _LISTPOL_H
```

File **listline.h**

```
#ifndef _LISTLINE_H
#define _LISTLINE_H
#include <iostream.h>
#include <fstream.h>
#include <math.h>
#include "polygon.h"
#include "line.h"
#include "stat.h"

class Node_line
{ public:
 Node_line* next_line;
 Line* content; };

class ListLines
{ public:
 Node_line* head_line;
  int Nline;

ListLines ()          // default constructor
  { head_line=0;  Nline = 0; };
~ListLines () {};    //default destructor

void add_line  (Line&);
friend ostream& operator<< (ostream& , ListLines&);
void print();
Stat find_mean_sd_length ();
};

#endif _LISTLINE_H
```

File **borehole.h**

```
#ifndef _BOREHOLE_H
#define _BOREHOLE_H
#include <iostream.h>
#include <fstream.h>
#include <math.h>
#include "line.h"
#include "point.h"
#include "stat.h"
#include "polygon.h"
extern ofstream out;

class Node_frac
{ public:
  double X, Y, Z;
  double t;            // from parametric equation of a line for a line segment: 0<= t <=length
Polygon* fracture;
Node_frac* next_frac;

friend ostream& operator<< (ostream& o, Node_frac& nf)
{ o<<nf.X<<" "<<nf.Y<<" "<<nf.Z<<" "<<nf.t<<" "<<nf.fracture->Pole.theta*180./PI-90.<< \
  " "<<nf.fracture->Pole.theta*180./PI<<" "<<nf.fracture->dip*180./PI<<" "<<*nf.fracture; };
};


class Borehole
{ public:
 Line log;
 Node_frac* head_frac;
 int Nfrac;

Borehole (Line& L)                    // default constructor
  { log = L;
    head_frac = 0;
    Nfrac = 0; };

void if_exists_add_intersection (Polygon& );
friend ostream& operator<< (ostream& , Borehole&);
void print();
Stat find_mean_sd_spacing ();

~Borehole () {};                      // default destructor

};

#endif _BOREHOLE_H
```

File **stat.h**

```
#ifndef _STAT_H
#define _STAT_H
#include <iostream.h>
#include <fstream.h>
#include <math.h>


class Stat {

public:
  double Mean;
  double SD;
  double total;
Stat ()
  { Mean = 0.; SD =0.; total = 0.;};

Stat (double M, double sd, double t)
  {
Mean = M;
SD = sd;
total = t;
};

friend ostream& operator<< (ostream& o , Stat s)
 {
  o <<s.Mean<<" "<<s.SD<<" "<<s.total;
  return o;
};

};


#endif _STAT_H
```

File **cartesian.C**
(page 1 / 2)

```
#include "cartesian.h"
#include "polar.h"
#include <math.h>

// Procedure to transform the cartesian coordinates of a direction into polar. Not to be used
// for points because polar coordinates are only two angles theta and phi, there is no distance.

Polar Cartesian::convert_to_polar ()
      {
double theta, phi;
double l3d = sqrt(X*X+Y*Y+Z*Z);
double l2d = sqrt(X*X+Y*Y);

if (X==0. && Y==0. && Z>=0.)
  theta = phi =0.;
else if (X==0. && Y==0. && Z<0.)
  theta = phi =PI;
else
 {
    phi = acos(Z/l3d);
   theta = acos(Y/l2d);
   if (X<0.)
    theta = -theta;  };

Polar* polar_normal = new Polar(theta, phi);
return (*polar_normal);
};



// Function to find the coordinates of a vector into the LOCAL coordinate  system.
// The Polar MeanPole (theta, phi) gives the orientation of the mean vector in the global plane
// of reference. The direction of this vector is the axis Z of the local coordinate system.

Cartesian Cartesian::local_coordinates (Polar& MP)  {
double ct = cos(MP.theta);
double st = sin(MP.theta);
double cp = cos(MP.phi);
double sp = sin(MP.phi);

double xx = X*ct - Y*st;
double yy = X*st*cp + Y*ct*cp - Z*sp;
double zz = X*st*sp + Y*ct*sp + Z*cp;
Cartesian local (xx, yy, zz);
return local;
 };
```

File **cartesian.C**
(page 2 / 2)

```
// Function to find the coordinates of a vector into the GLOBAL coordinate system.
// The Polar MeanPole (theta, phi) gives the orientation of the mean vector in the global
// plane of reference. The direction of this vector is the axis Z of the local coordinate system.

Cartesian Cartesian::global_coordinates (Polar& MP) {
double ct = cos(MP.theta);
double st = sin(MP.theta);
double cp = cos(MP.phi);
double sp = sin(MP.phi);

double xx = X*ct + Y*st*cp + Z*st*sp;
double yy = -X*st + Y*ct*cp + Z*sp*ct;
double zz = -Y*sp + Z*cp;

Cartesian global (xx, yy, zz);
return global;
 };
```

File **polar.C**

```
#include "polar.h"
#include "cartesian.h"

Cartesian Polar::convert_to_cartesian()
   {
Cartesian* c = new Cartesian ();
     c->X=sin(phi)*sin(theta);
     c->Y=sin(phi)*cos(theta);
     c->Z=cos(phi);
     return (*c);
};
```

File **point.C**

```
#include "point.h"

extern ofstream out;
extern double Datum;

void Point::print()
{ out<<"{"<<X<<","<<Y<<","<<Z+Datum<<"}"; };
```

File **plane.C**
(page 1 / 4)

```
#include "polar.h"
#include "plane.h"
#include "cartesian.h"
#include "point.h"
extern double bFk1, bFk2, Fk, MaxPhi;

Polar ran_uniform_orientation(void);
Polar uniform_max_phi_orient(double);
Polar constant_orientation(void);
Polar Fisher_orientation (double);
Polar bivariate_Fisher_orientation (double, double);

#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)

ostream& operator<< (ostream& o, Plane& p)
 {  o <<p.A<<" "<<p.B<<" "<<p.C<<" "<<p.D<<endl;
    o <<"rel polar "<<p.rel_polar<<" abs polar "<<p.abs_polar<<endl;
    o << "strike "<<p.strike<<" dip "<<p.dip<<endl;
    return o;  };

// Constructor for the plane when the A, B, C, and D in the equation
// Ax+By+Cz=D are given

Plane :: Plane (double a, double b, double c, double d) {

A=a; B=b; C=c; D=d;
MeanPole.theta=0.; MeanPole.phi=0.;
rel_cart.X=A; rel_cart.Y=B; rel_cart.Z=C;
abs_cart.X=A; abs_cart.Y=B; abs_cart.Z=C;
rel_polar = rel_cart.convert_to_polar();
abs_polar.theta = rel_polar.theta;
abs_polar.phi = rel_polar.phi;

strike = abs_polar.theta - HalfPI;
if (strike < -PI)
  strike += TwoPI;

dip = abs_polar.phi;
if (dip > HalfPI)
    {dip = PI-dip;
 if (strike > 0.)
   strike-=PI;
 else
   strike +=PI;
    };
};
```

```
// Constructor for a plane from a normal vector v(A, B, C) and a point p(X,Y,Z) in the global f.o.r.
// The plane is defined by equation Ax+By+Cz=D where D=AX+BY+CZ. The plane belongs to a
// set with mean pole orientation given by the class Polar MeanP. The coordinates of the pole in
// the local frame of reference (Z=MeanPole) are calculated correspondingly.

Plane::Plane (Cartesian& v, Point& p, Polar& MeanP) {
D=v.X*p.X+v.Y*p.Y+v.Z*p.Z;
A=v.X; B=v.Y; C=v.Z;  MeanPole = MeanP;
abs_cart.X=A; abs_cart.Y=B; abs_cart.Z=C;
abs_polar = abs_cart.convert_to_polar();
rel_cart = abs_cart.local_coordinates(MeanPole);
rel_polar = rel_cart.convert_to_polar();

strike = abs_polar.theta - HalfPI;
if (strike < -PI)
  strike += TwoPI;

dip = abs_polar.phi;
if (dip > HalfPI)
    {dip = PI-dip;
 if (strike > 0.)
   strike-=PI;
 else
   strike +=PI;  };
};

// Constructor to create a plane through a given point P(X,Y,Z) in space. The MeanPole (Z in
//  relative f.o.r.) is given by first class Polar. The orientation of the plane in the RELATIVE f.o.r.
//  is given by the second class Polar. The coordinates of the point are in ABSOLUTE f.o.r.

Plane::Plane (Polar& MeanP, Polar& planePole, Point& p3d)
{ MeanPole = MeanP;  rel_polar = planePole;
  rel_cart =  rel_polar.convert_to_cartesian();
  abs_cart = rel_cart.global_coordinates (MeanPole);
  abs_polar = abs_cart.convert_to_polar();
  D = abs_cart.X*p3d.X+abs_cart.Y*p3d.Y*abs_cart.Z*p3d.Z;
  A = abs_cart.X; B = abs_cart.Y; C = abs_cart.Z;

strike = abs_polar.theta - HalfPI;
if (strike < -PI)
  strike += TwoPI;
dip = abs_polar.phi;
if (dip > HalfPI)
    {dip = PI-dip;
 if (strike > 0.)
   strike-=PI;
 else
   strike +=PI; };
};
```

File **plane.C**
(page 3 / 4)

```
// Procedure to create a plane with a given orientation through a given  point in 3D. The relative
//  polar orientation is generated in the LOCAL  f.o.r. stochastically according to a PDF specified
// by the character option. The relative cartesian, and the absolute pole orientations are calculated
//  next. The equation of the palne (A, B, C, D) is calculated so that the plane passes through the
// point p3d (X,Y,Z) given in the GLOBAL f.o.r. The Polar Mean gives the orientation of the local
// f.o.r. which is the mean pole orientation for the set of planes to which the new plane belongs.

Plane:: Plane (char option, Polar& MeanP, double Distance)
{
switch (option)
  {
  case 'u':                    // uniform distribution on the hemisphere
rel_polar = ran_uniform_orientation();
  break;
  case 'p':                    // partial uniform  on the hemisphere
rel_polar =  uniform_max_phi_orient(MaxPhi);
  break;
  case 'c':                     // constant orientation = Mean
rel_polar =  constant_orientation();
  break;
  case 'f':                      // Fisher
rel_polar = Fisher_orientation (Fk);
  break;
  case 'b':                        // Bivariate Fisher
rel_polar = bivariate_Fisher_orientation (bFk1, bFk2);
  break;
    default:
    cout<<"Unknown distribution."<<endl;
    break;
  };

MeanPole = MeanP;
rel_cart =  rel_polar.convert_to_cartesian();
abs_cart = rel_cart.global_coordinates (MeanPole);
if (option == 'c')
   abs_polar = MeanPole;
else
   abs_polar = abs_cart.convert_to_polar();

if (abs_polar.phi > HalfPI)          // make pole always point upward
     {abs_polar.phi = PI - abs_polar.phi;
      abs_polar.theta -= PI;
      if (abs_polar.theta < -PI)
            abs_polar.theta += TwoPI; };

A = abs_cart.X; B = abs_cart.Y; C = abs_cart.Z;
D = Distance;
```

```
strike = abs_polar.theta - HalfPI;
if (strike < -PI)
  strike += TwoPI;
dip = abs_polar.phi;
if (dip > HalfPI)
    {dip = PI-dip;
 if (strike > 0.)
   strike-=PI;
 else
   strike +=PI;
    };
};


// Function to calculates a vector of unit length which is perpendicular to the pole of the
// plane, lies in the plane and points upward along the dip of the plane.

Cartesian Plane :: find_binormal () {
Cartesian* bc = new Cartesian;

if (C>0.)
  {bc->X = cos(dip) * sin(abs_polar.theta+PI);
   bc->Y = cos(dip) * cos(abs_polar.theta+PI);
   bc->Z = sin(dip);};
if (C<0.)
  {bc->X = cos(dip)*sin(abs_polar.theta);
   bc->Y = cos(dip)*cos(abs_polar.theta);
   bc->Z = sin(dip);};
if (C==0.)
  {bc->Y=0.; bc->Y=0.; bc->Z=1.;};

return (*bc);
};
```

File **surface.C**
(page 1 / 4)

```
#include "surface.h"
#include "line.h"

boolian Surface::is_point_on_surface (Point& p)
{ double zz=A*p.X*p.X+B*p.X*p.Y+C*p.Y*p.Y+D*p.X+E*p.Y+F;
  double dif = p.Z-zz;
  if (dif>=-0.000001 && dif<=0.000001)
    {return TRUE;}
  else
    {return FALSE;};
  };

// Procedure to calculate if a point P(X,Y,Z) is above or below a surface.
// If Z<z on the surface at (X,Y), the point is below and the function returns -1.
// If Z>z on the surface at (X,Y), the point is above and the function returns 1.

int Surface :: is_point_above_below (Point& P)
{  if (is_point_on_surface (P))
    return 0;
double zz = find_Z_on_surface (P.X, P.Y);
if (P.Z>zz)
  {return 1; };    // The point is above the surface
if (P.Z<zz)
  {return -1; };    // The point is below the surface
};


boolian Surface::is_line_on_surface (Line& l)
{ if ((is_point_on_surface(l.end1)) && (is_point_on_surface(l.end2)))
  return TRUE;
else
  return FALSE;
};


// Procedure to find out if a line lies above or below a surface, or if the line intersects the
// surface. 1 is returned if both ends of the line are above the surafce, -1 if they are below the
// surface. If one end is above and the other one is below the surface, 0 is returned to inticate
// that the line intersects the surface.

int Surface :: how_line_intersect (Line& L)
{ int a1 = is_point_above_below (L.end1);
  int a2 = is_point_above_below (L.end2);
   if (a1*a2 < 0)              // The line intersects the surface
    return 0;
  else if (a1+a2 > 0)          // Both ends are on the same side of the surface
    return 1;                 // The line is above the surface
  else
    return -1;                // The line is below the surface
 };
```

File **surface.C**
(page 2/ 4)

```
// This function calculates the coordinates of the normal vector  o a quadratic surface at a point
// (X,Y, Z) on the surface. The normal vector is defined by the first derivative of the
// surface equation f(x,y,z)=0, i.e. n=(df/dx,df/dy,df/dz). For quadratic surface (see definition
// of class Surface) the normal vector is (-2*A*X-B*Y-D, -2*C*Y-B*X-E, 1)

Cartesian Surface::normal_at_point(Point& p)
{
double xx=2*A*p.X+B*p.Y+D;
double yy=2*C*p.Y+B*p.X+E;

Cartesian* vector = new Cartesian(-xx, -yy, 1.);
return (*vector);
};


// This function finds the Z coordinate of an (X, Y) point such that
// point (X,Y,Z) is on the given surface

double Surface::find_Z_on_surface (double x, double y) {
double Z=A*x*x+B*x*y+C*y*y+D*x+E*y+F;
return Z;
};


// This function finds the azimuth and latitude of a surface at a point.
// Latitude is calculated as the angle from north (axis X). Positive values
// are to the east, negative values are to the west. Axis Y is east.
// Azimuth is calculated as the angle between the normal vector at the point
// and the vertical axis Z. A Polar (latitude, azimuth) is returned.
// The coordinate system (X,Y,Z) is left-handed.

Polar Surface::polar_normal_at_point (Point& p) {
Cartesian N = normal_at_point(p);
double theta, phi;
double l3d = sqrt(N.X*N.X+N.Y*N.Y+N.Z*N.Z);
double l2d = sqrt(N.X*N.X+N.Y*N.Y);

phi = asin(l2d/l3d);
theta = acos(N.Y/l2d);
if (N.X<0.)
   theta*=-1.;

Polar* polar_normal = new Polar(theta, phi);
return (*polar_normal);
};
```

File **surface.C**
(page 3 / 4)

```
// This function returns the strike and dip of a surface at a point, calculated as the strike and dip
// of a tangent plane. Dip is between zero and pi/2 measured from the horizontal to the slope
// in a vertical plane. Strike is between -pi to pi, negative values are west from north,
// positive values are east from north. The system NTB is left-handed, where N is the normal
// vector, T is the strike tangent vector, and B=NxT is binormal pointing upward along the dip.

Polar Surface::strike_dip_at_point (Point& p)
{ Polar SD = polar_normal_at_point(p);
  SD.theta-=HalfPI;
  if (SD.theta < -PI)
     SD.theta+=TwoPI;
  return SD;  };

// This function calculates the volume enclosed under the surface over the
// rectangular area (x, y) where x=[-xm, xm] and y=[-ym, ym]

double Surface::find_enclosed_volume (double xm, double ym)
{ double volume = xm*ym*(4./3.*(A*xm*xm+C*ym*ym)+B*xm*ym+2.*(D*xm+E*ym)+4.*F);
  return volume;  };


// Function to find the maximum Z value of a surface above a rectangular area bounded by
// X, -X, Y, and -Y. Zmax is chosen to be the maximum value of: 1) the values at the four end
// points; 2) the local maximum of the function along the edges, if such a maximum exists;
// 3) the local maximum inside the area (X,Y), if such a maximum exists. A local maximum
// of f(x) exists at the point where the df/dx is zero, and d^2f/dx^2 is negative. For f(x, y) the
// function has a local maximum if df/dx=df/dy=0, and d^2f/dx^2*d^2f/dy^2-(d^2f/dxdy)^2<0.
// This procedure ignores some cases for which the test is inconclusive (see calculus references).
// For an upward convex surface such cases are unlikely.

double Surface::Zmax_over_XY (double X, double Y)
{ double Zmax, zz, x, y;
  Zmax = find_Z_on_surface (X, Y);
   zz = find_Z_on_surface (-X, Y);
     if (zz>Zmax)
        Zmax=zz;
   zz = find_Z_on_surface (-X, -Y);
     if (zz>Zmax)
        Zmax=zz;
   zz = find_Z_on_surface (X, -Y);
     if (zz>Zmax)
        Zmax=zz;

double f = 4*A*C-B*B;
if (f<0.)                        // possible maximum inside (X, Y)
    { x=(B*E-2*D*C)/f;
    y=(B*D-2*A*E)/f;             // point where 1st derivatives are 0.
zz = find_Z_on_surface(x, y);
    if (zz>Zmax)
      Zmax=zz;  };
```

File **surface.C**
(page 4 / 4)

```
if (C<0.)                    // possible maximum along x=X or x=-X
 {
  y=(-E-B*X)/(2*C);          // checking max along x=X
    if (y>-Y && y<Y)
      zz = find_Z_on_surface(X, y);
    if (zz>Zmax)
      Zmax=zz;
  y=(-E+B*X)/(2*C);          // checking max along x=-X
    if (y>-Y && y<Y)
      zz = find_Z_on_surface(-X, y);
    if (zz>Zmax)
      Zmax=zz;
 };


if (A<0.)                    // possible maximum along y=Y or y=-Y
 {
  x=(-D-B*Y)/(2*A);          // checking max along y=Y
    if (x>-X && x<X)
      zz = find_Z_on_surface(x, Y);
    if (zz>Zmax)
      Zmax=zz;
  x=(-D+B*Y)/(2*A);          // checking max along y=-Y
    if (x>-X && x<X)
      zz = find_Z_on_surface(x,-Y);
    if (zz>Zmax)
      Zmax=zz;
 };

return Zmax;
};
```

File **cubic.C**

```
#include "cubic.h"

// This function calculates the coordinates of the normal vector
// to a cubic surface at a point p(X,Y,Z) on the surface.
// The normal vector is defined by the first derivative of the
// surface equation f(x,y,z)=0, i.e. n=(df/dx,df/dy,df/dz).
// For cubic surface (see definiton of class Cubic)
// the normal vector is (-3AXX-2BXY-CYY-2EX-FY-H, -BXX-2CXY-3DYY-FX-2GY-I, 1)

Cartesian Cubic::normal_at_point(Point& p)
{
double xx = 3*A*p.X*p.X + 2*B*p.X*p.Y + C*p.Y*p.Y + 2*E*p.X + F*p.Y + H;
double yy = B*p.X*p.X + 2*C*p.X*p.Y + 3*D*p.Y*p.Y + F*p.X + 2*G*p.Y + I;
Cartesian* vector = new Cartesian(-xx, -yy, 1.);
return (*vector);
};

// This function finds the azimuth and latitude of a cubic surface at a point.
// Latitude is calculated as the angle from north (axis X). Positive values
// are to the east, negative values are to the west. Axis Y is east.
// Azimuth is calculated as the angle between the normal vector at the point
// and the vertical axis Z. A Polar (latitude, azimuth) is returned.
// The coordinate system (X,Y,Z) is left-handed.

Polar Cubic :: polar_normal_at_point (Point& p)
{
Cartesian N = normal_at_point(p);
Polar* polar_normal = new Polar;
*polar_normal = N.convert_to_polar ();
return (*polar_normal);
};


// This function returns the strike and dip of a cubic surface at a point,
// calculated as the strike and dip of a tangent plane. Dip is between zero
// and pi/2 measured from the horizontal to the slope in a vertical plane.
// Strike is between -pi to pi, negative values are west from north,
// positive values are east from north. The system NTB is left-handed,
// where N is the normal vector, T is the strike tangent vector, and
// B=NxT is binormal pointing upward along the dip.

Polar Cubic :: strike_dip_at_point (Point& p)
{
Polar SD = polar_normal_at_point(p);
SD.theta -= HalfPI;
if (SD.theta < -PI)
   SD.theta += TwoPI;
return SD;
};
```

File **volume.C**

```
#include "volume.h"

double Random01 ();
double Random0a (double a);
double  RandomBC (double  b, double c);
extern double Xm, Ym;


// Constructor for the total volume enclosed under a quadratic surface above
// a rectangular area defined by the vertical planes x=Xm, x=-Xm, y=Ym,
// and y=-Ym. The origin of the coordinate system is assumed to be
// in the middle of the rectangular area, on the horizontal plane bounding
// the modeling volume from below.

Volume :: Volume (Surface& ground)
 {
  top=ground;
  P[0].X=Xm; P[0].Y=Ym; P[0].Z = top.find_Z_on_surface(Xm, Ym);
  P[1].X=-Xm; P[1].Y=Ym; P[1].Z = top.find_Z_on_surface(-Xm, Ym);
  P[2].X=-Xm; P[2].Y=-Ym; P[2].Z = top.find_Z_on_surface(-Xm, -Ym);
  P[3].X=Xm; P[3].Y=-Ym; P[3].Z = top.find_Z_on_surface(Xm, -Ym);
  volume=top.find_enclosed_volume(Xm, Ym);
  Zmax=top.Zmax_over_XY(Xm, Ym);
 };


// Procedure to generate a random point inside the modeling volume.
// The coordinates of the point are calculated uniformly as x=U(-Xm, Xm),
// y=U(-Ym, Ym), z=U(0, Zmax). After that it is calculated if it is inside
// the modeling volume (if Z of the point is below the top surface);
// if not, a new point is generated.

Point Volume :: random_point() {
Point* p = new Point;
do {
p->X = RandomBC (-Xm, Xm);
p->Y = RandomBC (-Ym, Ym);
p->Z = Random0a (Zmax);
}
while (!is_point_inside (*p));
return (*p);
};

boolian Volume :: is_point_inside (Point& p) {
if (p.X>=-Xm && p.X<=Xm && p.Y>=-Ym && p.Y<=Ym)
 {double zz = top.find_Z_on_surface(p.X, p.Y);
  if ( p.Z<=zz )
     return TRUE;};
return FALSE;
};
```

File **polygon.C**
(page 1 / 9)

```
#include "cartesian.h"
#include "point.h"
#include "polygon.h"
extern ofstream out;
extern double Xm, Ym;
extern double MeanArea;
extern double ratioMA;

double  RandomBC (double, double);


// Procedure to print a polygon for graphics output. The function creates
// a string which can be plotted directly by running Mathematica.

void Polygon :: print()
{
Point* marker = head;
int i;
out<<"Graphics3D [Polygon [{";
for (i=0; i< (noP-1); i++)
  {
   marker->print(); out<<",";
   marker = marker->next;
  };
marker->print();
out << "}], PlotRange->All, Axes->True, Boxed->True]";
return;
};



// A constructor to create a "polygon" with one vertex coinciding with
// the point P given in the global f.o.r., and parallel to the plane p.
// The plane does not have to pass through the point.

Polygon :: Polygon (Plane& p)
{
setPole = p.MeanPole;
Pole = p.abs_polar;
strike = p.strike;
dip = p.dip;
head = 0;
noP =0;
center.X = 0.;  center.Y = 0.;  center.Z = 0.;
area = 0.; radius = 0.;
};
```

```
// Procedure to find the coordinates of the center of a polygon.

void Polygon :: find_center ()
{
Point* current = head;
double xx=0.; double yy=0.; double zz=0.;
int i;
 for (i=0; i<noP; ++i)
  {
xx+=current->X; yy+=current->Y; zz+=current->Z;
current = current->next;
};
xx /= noP; yy /= noP; zz /= noP;
center.X = xx; center.Y = yy; center.Z = zz;
return;
};


// Procedure to calculate the area of a 2d polygon (all Z coordinates of
// vertices must be the same!! The formula used is
// A = 1/2(XiYi+1 + ... - XiYi-1 - ...) for i = 0 ... noP-1

void Polygon::find_area_radius_2d() {
area =0.; radius = 0.;
Point* current = head;
int i;
double xx, yy;

for (i=0; i<noP; ++i)
  {
xx = current->X; yy = current->next->Y;
area += (xx*yy);
current = current->next;
};
for (i=0; i<noP; ++i)
  {
yy = current->Y; xx = current->next->X;
area -= (xx*yy);
current = current->next;
};
area /= 2;
if (area<0.)
    area*=(-1.);
radius = sqrt (area/PI);
return;
};
```

```
// Procedure to calculate the coordinates of a polygon in its own plane,
// i.e. the polygon becomes essentially two-D, all Z coordinates are the same.

void Polygon :: make_polygon_2d ()
{
Point* current = head; int i;
Cartesian temp;
for (i=0; i<noP; ++i)
{ temp.X = current->X; temp.Y = current->Y; temp.Z = current->Z;
temp = temp.local_coordinates (Pole);
Point p(temp.X, temp.Y, temp.Z);
current->X = p.X; current->Y = p.Y; current->Z = p.Z;
current = current->next; };
find_center ();
if (!area && !radius && noP>2)
  { sort_points_2d();
    find_area_radius_2d(); };
return;
};


// Procedure to sort the vertices in a 2D polygon in order: according to
// the angle, defined by a line through a vertex and the center, and the
// local axis X. All Z coordinates are the same, no need to calculate them.

void Polygon :: sort_points_2d ()
{
Point* current = head; Point* nextP;
int i, j; double xxI, yyI, xxJ, yyJ, xx, yy, zz, angleI, angleJ;
for (i=1; i < noP; ++i)
  {   nextP = current->next;
     for (j=i; j < noP; ++j)
      { xxI = current->X - center.X; yyI = current->Y -  center.Y;
        angleI = acos (xxI / sqrt(xxI*xxI + yyI*yyI));
          if (yyI<0.)
            angleI = TwoPI - angleI;
xxJ = nextP->X - center.X; yyJ = nextP->Y -  center.Y;
angleJ = acos (xxJ / sqrt(xxJ*xxJ + yyJ*yyJ));
          if (yyJ<0.)
            angleJ = TwoPI - angleJ;
     if (angleI > angleJ)
          {   xx = current->X; yy = current->Y; zz = current->Z;
              current->X = nextP->X; current->Y = nextP->Y; current->Z = nextP->Z;
               nextP->X = xx; nextP->Y = yy; nextP->Z = zz;  };
      nextP = nextP->next;
     };
     current = current->next;
};
return;
};
```

```
// Procedure to calculate the coordinates of a 2d polygon from its local
// f.o.r. back into 3d in the global f.o.r.

void Polygon :: make_polygon_3d ()
{
Point* current = head;
int i;
Cartesian temp;
for (i=0; i < noP; ++i)
{ temp.X= current->X; temp.Y= current->Y; temp.Z= current->Z;
temp = temp.global_coordinates (Pole);
Point p(temp.X, temp.Y, temp.Z);
current->X = p.X; current->Y = p.Y; current->Z = p.Z;
current = current->next; };
find_center ();
return;
};

// Procedure to find the area and the equivalent radius of a 3d polygon.
// The coordinates of the vertices are transformed into 2d f.o.r. The
// vertices are sorted in order. Then the area of the 2d polygon is
// calculated and the radius is found as the radius of a circle with the
// same area. The coordinates of the vertices are then back calculated in 3d.

void Polygon :: area_radius_3d () {
make_polygon_2d();
sort_points_2d ();
find_area_radius_2d ();
//cout<<" 2D POLYGON "<<*this<<endl;
make_polygon_3d();
//cout<<"POLYGON "<<*this<<endl;
return;
};


// Function to find out if a point belongs to a polygon. Returns TRUE if the
// point has the same coordinates as one of the vertices of the polygon.

boolian Polygon :: is_it_member (Point& P)
{
Point* marker = head;
int i;
for (i=0; i<noP; i++)
  {   if (marker->X == P.X && marker->Y == P.Y && marker->Z == P.Z)
            return TRUE;
        marker = marker->next;    };
    return FALSE;
};
```

```
// Function to add a point to a polygon. The new point is appended
// at the head  of the list of vertices (Points) which consitute the
// polygon. The last point points to the new point which is the new
// head point.

void Polygon :: add_point (Point& P)
{
Point* temp = new Point;
++noP;
temp->X = P.X;
temp->Y = P.Y;
temp->Z = P.Z;
 temp->next = head;
head = temp;
Point* current = head;
int i;
for (i=1; i < noP; ++i)
     current = current->next;
current->next = head;
if (noP==3 && !Pole.theta && !Pole.phi)
  {
double a1 = head->next->X - head->X;
double b1 = head->next->Y - head->Y;
double c1 = head->next->Z - head->Z;
double a2 = head->next->next->X - head->X;
double b2 = head->next->next->Y - head->Y;
double c2 = head->next->next->Z - head->Z;

Cartesian N ((b1*c2-b2*c1), (a2*c1-c2*a1), (a1*b2-b1*a2));
Pole = N.convert_to_polar();
  };
return;
};



// Procedure to find out if the shape of the polygon is good.
// For every vertex,  two values are calculated: 1) the angle between the
// two adjacent sides is calculated and compared to a specified minimum
// allowed angle MinAngle; and 2) the ratio of the distance to the center
// versus the equivalent radius of the polygon is compared to a specified
// maximum allowed elongation.

boolian Polygon :: if_good_shape (double MinAngle, double MaxE) {
if (noP<4)
 return FALSE;

if (ratioMA)
  {   if (area < ratioMA*MeanArea)
       return FALSE   };
```

```
double el = find_elongation ();
 if (el/(2*radius) > MaxE)
  return FALSE;
int i;
double ANGLE;
Point* current = head;
Point* nextP = current->next;
Point* thirdP = current->next->next;
for (i=0; i<noP; ++i)
  {
Line l1 (*nextP, *current);
Line l2 (*nextP, *thirdP);
ANGLE = l1.angle_with_line(l2);
if (ANGLE < MinAngle)
   return FALSE;
current = current->next;
nextP = nextP->next;
thirdP = thirdP->next;
  };
return TRUE;
};


// Procedure to find the elongation of a polygon, defined as the maximum
// distance between two vertices in the polygon.

double Polygon :: find_elongation ()
{
Point* current=head;
Point* nextP;
int i=0, j=2;
double elongation = 0.;
for (i=0; i<noP-2; i++)
  {
nextP = current->next->next;
for (j=i+2; j<noP; j++)
  {
Line l(*current, *nextP);
if (l.length > elongation)
 elongation = l.length;
nextP = nextP->next;
  };
current=current->next;
  };
return elongation;
};
```

```
// Procedure to find the radius R of the smallest sphere in which a polygon
// can be inscribed. This is assumed to be the largest of the distances
// between the center and one of the vertices.

double Polygon :: minR_inscribe () {
Point* current = head;
double R=0., temp, dx, dy, dz;
int i;
for (i=0; i<noP; ++i)
  {
dx=current->X - center.X;
dy=current->Y - center.Y;
dz=current->Z - center.Z;
temp = sqrt(dx*dx + dy*dy + dz*dz);
if (temp > R)
  R=temp;
current=current->next;
  };
return R;
};


// Two functions for translation of polygons in 3D

void Polygon :: operator+ (Cartesian& vector)
{
Point* current = head;
int i;
for (i=0; i<noP; i++)
  { *current + vector;
  current = current->next; };
center+vector;
return;
};


void Polygon :: operator- (Cartesian& vector)
{
Point* current = head;
int i:
for (i=0; i<noP; i++)
  { *current - vector;
  current = current->next; };
center-vector;
return;
};
```

```
// Function to translate a polygon perpendicular to its plane.    It is written in such a way that
// bigger polygons ar eless likely to be translated far form their original plane. The polygon
// has to be 2D!! Translation is accomplished by changing the Z coordinate.

void Polygon :: translate_2d (double ratio, double MeanR)
{ if (ratio)
  { double maxDZ = ratio*MeanR*MeanR/radius;
    double translation = RandomBC(-maxDZ, maxDZ);
    Point* current= head;
     int i;
for (i=0; i<noP; ++i)
  {  (current->Z) += translation;
      current=current->next;  };
center.Z += translation;  };
return;
};


// Procedure to find the distance from a point P to a polygon. The calculations  are in the
// FRAME OF REFERENCE of the polygon. The orthogonal progection  P1 of the point
// is found and if it is within the polygon, the distance PP1 is the shortest distance. If not,
// the point of interseciton P2 of the line between P1 and the center of the polygon is found.
// PP2 is then the shortest distance which is returned by the function. The 3d polygon and point
// MUST FIRST BE TRANSFORMED INTO LOCAL COORDINATES OF THE POLYGON.

double Polygon :: distance_from_point (Point& P)
{
Point P1 (P.X, P.Y, center.Z);
double distance;
Point* current = head;
int i;
Line l1 (center, P1);
for (i=0; i<noP; ++i)
  {
Line l2(*current, *current->next);
if (l1.intersect (l2))
  {Point P2 = l1.intersection_with_line (l2);

distance = sqrt((P.X-P2.X)*(P.X-P2.X)+(P.Y-P2.Y)*(P.Y-P2.Y)+(P.Z-P2.Z)*(P.Z-P2.Z));
return distance;}
else
  current = current->next;
  };
distance = P.Z - center.Z;
if (distance<0.)
    distance *= -1.;
return distance;
};
```

File **polygon.C**
(page 9 / 9)

```
// Procedure to find the shortest distance from a polygon pol (usually a new  fracture) to
// another polygon (usually a major feature such as fault). The shortest distance from a verex
// of pol to the major polygon is calculated using the function above. The initial and final
// coordinates  are in the global f.o.r.

double Polygon :: distance_from_polygon (Polygon& pol)
{
int LINE = 0;
Line L;
Cartesian N (Pole);
Plane p (N, *head, setPole);
if (pol.if_intersects_plane (p))
{ L = pol.intersection_with_plane (p);
 LINE = 1;};
Polar POLE = pol.Pole;
pol.Pole = Pole;
make_polygon_2d ();
pol.make_polygon_2d();
double distance = distance_from_point(*pol.head);
if (LINE)
 {L.local_coordinates (Pole);
 int HOW = how_line_intersects (L);
 if (HOW)
   distance = 0.;
 else {
   double distance1 = distance_from_point(L.end1);
   double distance2 = distance_from_point(L.end2);
   if (distance1 < distance)
       distance = distance1;
   if (distance2 < distance)
       distance = distance2;
 }; };

if (distance)
 { int i;
   double  temp;
   Point* current = pol.head->next;

for (i=1; i<pol.noP; ++i)
 { temp = distance_from_point (*current);
   if (temp < distance);
        distance = temp;
   current = current->next;
 }; };
make_polygon_3d();
pol.make_polygon_3d();
pol.Pole = POLE;

return distance;
};
```

File **line.C**
(page 1 / 3)

```
#include "point.h"
#include "line.h"
extern ofstream out;

// Function to print a line for graphics output (code in mathematica)

void Line::print()
{
out<<"Graphics3D [Line [{";
 end1.print();
 out<<", ";
 end2.print();
 out << "}], PlotRange->All, Axes->False, Boxed->False, ViewPoint->{0., 0., 100}]";
    return;
};


// Function to find if a Point P(X,Y,Z) is on a line segment L1L2.
// The function uses the parametric equation for a line (see "line.h")

boolian Line ::  is_point_on_line (Point& P)
{
double xx=0., yy=0., zz=0., t1, t2;
if (vector.X<-0.000001 || vector.X > 0.000001)
    xx = (P.X - end1.X) / vector.X;
if (vector.Y<-0.000001 || vector.Y>0.000001)
    yy = (P.Y - end1.Y) / vector.Y;
if (vector.Z<-0.000001 || vector.Z>0.000001)
    zz = (P.Z - end1.Z) / vector.Z;

if (xx && yy && zz)
  {
t1 = xx-yy, t2 = xx-zz;
if ((t1<0.00001) && (t1>-0.00001) && (t2<0.00001) && (t2>-0.00001) && (xx<=length) &&
(xx>0.))
  return TRUE;
  return FALSE;}
else if (xx && yy)
  {t1 = xx-yy;
if (t1<0.00001 && t1>-0.00001 && xx <= length && xx >= 0.)
  return TRUE;
  return FALSE;}
else if (xx && zz)
  {t1 = xx-zz;
if (t1<0.00001 && t1>-0.00001 && xx <= length && xx >= 0.)
  return TRUE;
  return FALSE;}
else if (yy && zz)
  {t1 = yy-zz;
if (t1<0.00001 && t1>-0.00001 && yy <= length && yy >= 0.)
```

```
 return TRUE;
 return FALSE;}
else if (xx)
 {if (xx <= length && xx >= 0.)
 return TRUE;}
else if (yy)
 {if (yy <= length && yy >= 0.)
 return TRUE;
 return FALSE;}
else if (zz)
 {if (zz <= length && zz >= 0.)
  return TRUE;
  return FALSE;}
else
   return FALSE;
};


// Two functions necessary to determine if two lines in (X,Y) plane
// intersect or cross
// From "Algorithms in C++" by Sedgwick, Chapter 24

int ccw (Point& p0, Point& p1, Point& p2)
{ double dx1, dx2, dy1, dy2;
   dx1= p1.X - p0.X; dy1 = p1.Y - p0.Y;
   dx2= p2.X - p0.X; dy2 = p2.Y - p0.Y;
if (dx1*dy2 > dy1*dx2) return 1;
if (dx1*dy2 < dy1*dx2) return -1;
if ((dx1*dx2 < 0) || (dy1*dy2 < 0)) return -1;
if ((dx1*dx1+dy1*dy1) < (dx2*dx2+dy2*dy2)) return 1;
return 0; };

boolian Line::intersect (Line& l)
{ if (((ccw(end1, end2, l.end1) * ccw(end1, end2, l.end2)) <=0 ) && ((ccw(l.end1, l.end2, end1) \
* ccw(l.end1, l.end2, end2)) <=0))
return TRUE;
return FALSE; };


// Function to find the point of intersection of a 2D line with
// another 2D line  in a plane
// FIRST USE THE TWO FUNCTIONS ABOVE TO FIND OUT IF THE TWO LINES
// INTERSECT AT ALL

Point Line::intersection_with_line (Line& l)
{ double xx, yy, slope1, slope2;
   if (end1.X==end2.X)
  { xx = end1.X;
     slope2 = (l.end2.Y-l.end1.Y)/(l.end2.X-l.end1.X);
      yy = l.end1.Y + slope2*(xx-l.end1.X);}
```

```
else if (l.end1.X==l.end2.X)
   { xx = l.end1.X;
     slope1 = (end2.Y-end1.Y)/(end2.X-end1.X);
      yy = end1.Y + slope1*(xx-end1.X);}
else {
slope1 = (end2.Y-end1.Y)/(end2.X-end1.X);
slope2 = (l.end2.Y-l.end1.Y)/(l.end2.X-l.end1.X);
xx = (end1.Y-slope1*end1.X-l.end1.Y+slope2*l.end1.X)/(slope2-slope1);
yy = (slope2*end1.Y-slope2*slope1*end1.X-slope1*l.end1.Y+slope1*slope2*l.end1.X)/(slope2-
slope1); };
Point* intersection = new Point;
*intersection = Point (xx, yy, end1.Z);
return *intersection;
};


// Procedure to find the local coordinates of a line in a frame of reference
// with Z axis given by Pole in the global f.o.r.

void Line :: local_coordinates (Polar& p)
{ Cartesian temp1, temp2;
   temp1.X = end1.X; temp1.Y = end1.Y; temp1.Z = end1.Z;
    temp1 = temp1.local_coordinates (p);
    end1.X = temp1.X; end1.Y = temp1.Y; end1.Z = temp1.Z;
temp2.X = end2.X; temp2.Y = end2.Y; temp2.Z = end2.Z;
temp2 = temp2.local_coordinates (p);
end2.X = temp2.X; end2.Y = temp2.Y; end2.Z = temp2.Z;
vector.X = end2.X - end1.X;
vector.Y = end2.Y - end1.Y;
vector.Z = end2.Z - end1.Z;
return;
};


// Procedure to find the global coordinates of a line.

void Line :: global_coordinates (Polar& p)
{ Cartesian temp1, temp2;
temp1.X = end1.X; temp1.Y = end1.Y; temp1.Z = end1.Z;
temp1 = temp1.global_coordinates (p);
end1.X = temp1.X; end1.Y = temp1.Y; end1.Z = temp1.Z;
temp2.X = end2.X; temp2.Y = end2.Y; temp2.Z = end2.Z;
temp2 = temp2.global_coordinates (p);
end2.X = temp2.X; end2.Y = temp2.Y; end2.Z = temp2.Z;
vector.X = end2.X - end1.X;
vector.Y = end2.Y - end1.Y;
vector.Z = end2.Z - end1.Z;
return;
};
```

File **listpol.C**
(page 1 / 7)

```
#include "listpol.h"
#include "polygon.h"
#include "surface.h"
#include "stat.h"
#include <math.h>
#include <iostream.h>

#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)
extern ofstream out;

double Random01 ();

// Function to add a new polygon to a list of polygons as the  new head polygon of the list.

void ListPolygons :: add_polygon (Polygon& pol)
{ Node* new_head = new Node;
  new_head->content = &pol;
   new_head->next_pol = head_pol;
   head_pol = new_head;
   ++Npol;
    return;  };

// Function to print a list of polygons for non graphical output.

ostream& operator<< (ostream& o, ListPolygons& lp)
{ o<<"N POl "<<lp.Npol<<endl;
  Node* current = lp.head_pol;
   int i;
for (i=0; i<lp.Npol; ++i)
{  o<<(*current->content) <<endl;
   current=current->next_pol;  };
return  o;  };

// Function to print a list of polygons for graphics output as mathematica code

void ListPolygons :: print()
{
int i;
int j=0;
Node* current = head_pol;
if (current->content)
  {   for (i=0; i<Npol-1; ++i)
    {  current->content->print();
       out<<", ";};
        j++;
  current=current->next_pol;};
current->content->print();  };
return; };
```

```
// Function to make all polygons in a list of polygons 2d

void ListPolygons :: make_listpol_2d ()
{ Node* current = head_pol;
  int i;
for (i=0; i<Npol; i++)
  {current->content->make_polygon_2d();
  current->content->find_center();
  current=current->next_pol; };
return; };

// Function to make all polygons in a 2d list of polygons 3d

void ListPolygons :: make_listpol_3d ()
{ Node* current = head_pol;
  int i;
for (i=0; i<Npol; i++)
  {current->content->make_polygon_3d();
 current->content->find_center();
 current=current->next_pol; };
return;   };

// Function to find the areas and radii of a list of 2d polygons

void ListPolygons :: find_area_radius_2d ()
{  Node* current = head_pol;
   int i;
for (i=0; i<Npol; ++i)
  {current->content->find_area_radius_2d();
  current=current->next_pol; };
return; };

// Function to discard the polygons with bad shape, i.e. only polygons with angles
// larger than MinAngle and elongation less than MaxE are retained in the list

void ListPolygons :: mark_good_shape (double MinAngle, double MaxE)
{  Node* current = head_pol;
do  {
if ((current->next_pol) && !(current->next_pol->content->if_good_shape (MinAngle, MaxE)))
    { if (current->next_pol->next_pol)
            delete current->next_pol->content;
     current->next_pol = current->next_pol->next_pol;
     --Npol; }
else
  current=current->next_pol;  }
while (current);
if(!(head_pol->content->if_good_shape (MinAngle, MaxE)))
{  head_pol = head_pol->next_pol;
  --Npol; };
return;  };
```

```
// Function to discard all polygons with bad shape, i.e. only polygons with  angles larger than
//  MinAngle and elongation less than MaxE are retained  in the list. The function also marks
// the good polygons with probability P, only P% of the good polygons are kept

void ListPolygons :: mark_good_shape_and_P (double MinAngle, double MaxE, double P)
{ Node* current = head_pol;
do {
if ((current->next_pol) && (!(current->next_pol->content->if_good_shape (MinAngle, MaxE)) ||
Random01 () > P))
  { if (current->next_pol->next_pol)
            delete current->next_pol->content;
     current->next_pol = current->next_pol->next_pol;
     --Npol; }
else
  current=current->next_pol; }
while (current);
if(!(head_pol->content->if_good_shape (MinAngle, MaxE)))
{  head_pol = head_pol->next_pol;
 --Npol; };
return; };


// Function to mark the polygons below (B) or above (A) a surface with probability P.
// Such marking decreases P32 below or above the surface to P of P32 initial. 0<P<1.

void ListPolygons :: mark_to_surface (Surface& S, double P, char how)
{ int HOW;
 Node* current = head_pol;
 do  {

if (current->next_pol)
  {  HOW =  S.is_point_above_below (current->next_pol->content->center);
     if ((how == 'A' && HOW > 0 || how == 'B' && HOW < 0) && Random01 () > P)
    { current->next_pol = current->next_pol->next_pol;
    --Npol; }
else
 current=current->next_pol; }
else
 current=current->next_pol; }
while (current);

HOW =  S.is_point_above_below (head_pol->content->center);
if ((how == 'A' && HOW > 0 || how == 'B' && HOW < 0) && Random01 () > P)
{head_pol = head_pol->next_pol;
 --Npol;
};
return;
};
```

```
// Function to add a list of polygons to a  list of polygons. The main list  (which calls the function)
//  now contains the two lists. The argument list is added at the head of the main list.

void ListPolygons :: add_listpol (ListPolygons& lp1)
{
if (lp1.Npol)
  {
if (!Npol)
  {head_pol = lp1.head_pol;}
else
  {
int i;
Node* current = lp1.head_pol;

for (i=0; i<lp1.Npol-1; ++i)
   current=current->next_pol;
current->next_pol = head_pol;
head_pol = lp1.head_pol;
  };
Npol += lp1.Npol;
  };
return;
};


// Function to find mean and standard deviation of the areas of a list
// of polygons.

Stat ListPolygons :: find_mean_sd ()
{ int i ;
double total=0., sd=0., mean =0.;
Node* current = head_pol;
double area;
if (Npol)
  { for (i=0; i<Npol; ++i)
  {  area = current->content->area;
     total += area;
     sd += area*area;
 current=current->next_pol; };

mean = total / Npol;
if (Npol > 1)
 sd = sqrt ((sd - Npol*mean*mean) / (Npol -1));
 };
Stat meanSD (mean, sd, total);
return meanSD;
};
```

```
// Function to find mean and standard deviation of the areas of a list of polygons.
// At this point polygons that are too large are discarded from  the list o ffractures.

Stat ListPolygons :: find_mean_sd (double meanR, double maxRatio)
{ int i=0, j=0 ;
 double total=0., sd=0., mean =0.;
 Node* current = head_pol;
 double area;
 cout<<" calculate statistics of "<<Npol<<" polygons "<<endl;
while (current->next_pol)
 { if (current->next_pol && (current->next_pol->content->radius < meanR*maxRatio))
  { area = current->next_pol->content->area;
    total += area;
    sd += area*area;
 current=current->next_pol; }
else  {
current->next_pol = current->next_pol->next_pol;
  -- Npol; };
};
if (head_pol->content->radius < meanR*maxRatio)
  {  area = head_pol->content->area;
     total += area;
     sd += area*area;   }
else {
 head_pol = head_pol->next_pol;
  -- Npol; };
mean = total / Npol;
if (Npol > 1)
 sd = sqrt ((sd - Npol*mean*mean) / (Npol -1));
Stat meanSD (mean, sd, total);
return meanSD;
};

// Procedure to find the shortest distance from a polygon pol in 3d to a member
// of a list of polygons.

double ListPolygons :: shortest_distance_from_polygon (Polygon& pol)
{ double distance = head_pol->content->distance_from_polygon (pol);
  Node* current = head_pol->next_pol;
  int i;
  double temp;
for (i=1; i < Npol; ++i)
  {
temp = current->content->distance_from_polygon (pol);
if (temp < distance)
     distance = temp;
current = current->next_pol;
};
return distance;
};
```

```
// Procedure to translate the polygons in a list of polygons in the vicinity of their original
// position. Bigger polygons are translated closer to the the original plane than smaller polygons.

void ListPolygons :: translate_2d (double meanR, double ratio)
{  if(ratio)
  { Node* current = head_pol;
    int i;
for (i=0; i<Npol; ++i)
  { current->content->translate_2d (meanR, ratio);
current = current->next_pol;  };  };
return; };

// Procedure to cut a list of polygons above of below a surface. Cuts  the polygons only if they
// are intersected by the surface at all.  Retains the portion above the surface if the character
// argument is 'A' or 'a', or the portion below the surface if the character is 'B' or 'b'

void ListPolygons :: cut_by_surface (Surface& s, char how)
{Node* current = head_pol;
int i;
for (i=0; i<Npol; ++i)
  { if (!current->content->above_or_below_surface (s))
     current->content->cut_by_surface (s, how);
current = current->next_pol; };
return; };

// Two procedure to cut polygons by two surfaces. The first Surface sA has to be above the
// second Surface sB thorughout the extent of the modeled region i.e. for -Xm<x<Xm
// and -Ym<y<Ym. The location of the center of every polygon is calculated to find out
// if it is below or above the surfaces. The first function retains only the portion of polygons
// which is between the two  surfaces, if the center is located between the surfaces.
// Otherwise the  polygons are discarded.

void ListPolygons :: cut_between_two_surfaces  (Surface& sA, Surface& sB)
{ Node* current=head_pol;
int i, cA, cB;
while (current->next_pol)
  { cA = sA.is_point_above_below (current->next_pol->content->center);
    cB = sB.is_point_above_below (current->next_pol->content->center);
if (cA<0 && cB>0)              // the center is between the surfaces
  {                            // retain only portion  below sA
if (!current->next_pol->content->above_or_below_surface (sA))
 current->next_pol->content->cut_by_surface (sA, 'B');
                    // retain only portion above sB
if (!current->next_pol->content->above_or_below_surface (sB))
 current->next_pol->content->cut_by_surface (sB, 'A');
current = current->next_pol;  }
else
{ current ->next_pol = current->next_pol->next_pol;
--Npol; }; };
```

```
cA = sA.is_point_above_below (head_pol->content->center);
cB = sB.is_point_above_below (head_pol->content->center);
if (cA<0 && cB>0)              // the center is between the surfaces
  {                            // retain only portion  below sA
if (!head_pol->content->above_or_below_surface (sA))
 head_pol->content->cut_by_surface (sA, 'B');
                        // retain only portion above sB
if (!head_pol->content->above_or_below_surface (sB))
 head_pol->content->cut_by_surface (sB, 'A'); }
else  {
head_pol = head_pol->next_pol;
--Npol;};
return;
};


// The second procedure discards polygons with centers between the two
// surfaces. For all others, only the portions below sB or above sA are
// retained. sA has to be above sB throughout the modeling volume.

void ListPolygons :: cut_outside_two_surfaces  (Surface& sA, Surface& sB)
{ Node* current=head_pol;
  int i, cA, cB;
while (current->next_pol)
  { cA = sA.is_point_above_below (current->next_pol->content->center);
    cB = sB.is_point_above_below (current->next_pol->content->center);
if (cA<0 && cB>0)              // the center is between the surfaces
  {current->next_pol = current->next_pol->next_pol;
  -- Npol; }
else  {                        // retain only portion  above sA
if (cA>0 && !current->next_pol->content->above_or_below_surface (sA))
 current->next_pol->content->cut_by_surface (sA, 'A');
                              // OR retain only portion below sB
if (cB<0 && !current->next_pol->content->above_or_below_surface (sB))
 current->next_pol->content->cut_by_surface (sB, 'B');
current = current->next_pol;
 }; };
cA = sA.is_point_above_below (head_pol->content->center);
cB = sB.is_point_above_below (head_pol->content->center);
if (cA<0 && cB>0)              // the center is between the surfaces
  {head_pol = head_pol->next_pol;
  -- Npol; }
else  {                        // retain only portion  above sA
if (cA>0 && !head_pol->content->above_or_below_surface (sA))
 head_pol->content->cut_by_surface (sA, 'A');
                        // OR retain only portion below sB
if (cB<0 && !head_pol->content->above_or_below_surface (sB))
 head_pol->content->cut_by_surface (sB, 'B'); };
return;
};
```

File **listline.C**
(page 1 / 2)

```
#include "listline.h"
#include "line.h"
#include <math.h>
#include <iostream.h>
extern ofstream out;

#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)

// Function to add a new line at the head of a list of lines.

void ListLines :: add_line (Line& l)
{ Node_line* new_head = new Node_line;
new_head->content = &l;
new_head->next_line = head_line;
head_line = new_head;
++Nline;
return; };


// Function to print a list of lines for non graphical output.

ostream& operator<< (ostream& o, ListLines& ll)
 { o<<"N LINE "<<ll.Nline<<endl;
   Node_line* current = ll.head_line;
    int i;
for (i=0; i<ll.Nline; ++i)
{  o<<(*current->content) <<endl;
   current=current->next_line; };
return  o; }

// Function to print a list of lines for graphics output as mathematica code

void ListLines :: print()
{ int i;
int j=0;
Node_line* current = head_line;
if (current->content)
  { for (i=0; i<Nline-1; ++i)
  {if (!j)
    {   current->content->print();
       out<<",  ";};
  j++;
  if (j==1)
     j=0;
  current=current->next_line;};
current->content->print();  }:
return; };
```

```
// Function to find mean and standard deviation of the lengths of a list of lines

Stat ListLines :: find_mean_sd_length ()
{
int i ;
double total=0., sd=0., mean =0.;
Node_line* current = head_line;
double L;

if (Nline)
  {
for (i=0; i<Nline; ++i)
  {
   L = current->content->length;
   total += L;
   sd += L*L;
 current=current->next_line;
};
mean = total / Nline;
if (Nline > 1)
 sd = sqrt ((sd - Nline*mean*mean) / (Nline -1));
};

Stat meanSD (mean, sd, total);
return meanSD;
};
```

File **borehole.C**
(page 1 / 2)

```
#include "borehole.h"
#include "listpol.h"
#include "polygon.h"
#include "stat.h"
#include <math.h>
#include <iostream.h>

#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)
extern ofstream out;

ostream& operator<< (ostream& o, Borehole& bh)
{ o<<"Logline "<<endl<<bh.log<<"N intersections: "<<bh.Nfrac<<endl<<endl;
int i;   Node_frac* current = bh.head_frac;
o<<"Intersection with borehole "<<endl<<"Number X Y Z t STRIKE AZIMUTH DIP AREA \
RADIUS Xc Yc Zc"<<endl<<endl;
for (i=0; i<bh.Nfrac; ++i)
  {o<<i+1<<" "<<*current;   current = current->next_frac;};
o<<endl;  return o; };

// Procedure to add the intersection of a log line with a fracture (Polygon)  to the list of
//  intersections in a Borehole. the function first checks if the line and the polygon intersect at all.

void Borehole :: if_exists_add_intersection (Polygon& pol)
{ if (pol.if_3d_line_intersects (log))
  { Point P = pol.intersection_with_line (log);
    Node_frac* new_nf = new Node_frac;
    new_nf->X = P.X; new_nf->Y = P.Y; new_nf->Z = P.Z;
    new_nf->fracture = &pol;
if (log.vector.Z)
   new_nf->t = (new_nf->Z - log.end1.Z) / log.vector.Z;
else if (log.vector.Y)
   new_nf->t = (new_nf->Y - log.end1.Y) / log.vector.Y;
else
   new_nf->t = (new_nf->X - log.end1.X) / log.vector.X;
if (!Nfrac)
  { head_frac = new_nf;
    new_nf ->next_frac = 0; }
else if (head_frac->t >= new_nf->t)
    {      new_nf->next_frac = head_frac;
           head_frac = new_nf;}
else   {
Node_frac* current = head_frac;
while (current->next_frac && (current->next_frac->t < new_nf->t))
    current = current->next_frac;
new_nf->next_frac = current->next_frac;
current->next_frac = new_nf; };
Nfrac++; };
return; };
```

```
// Procedure to find the intersections of a log line with a list of  polygon-fractures. Returns
// an ordered log, i.e. the intersections with polygons form end1 to end2 of the given line.

Borehole ListPolygons :: intersections_with_logline (Line& L)
{ Node* current = head_pol;
  int i = Npol;
  Borehole* new_bh = new Borehole (L);
for (i=0; i<Npol; ++i)
  {new_bh -> if_exists_add_intersection (*current->content);
  current = current->next_pol;};
return *new_bh; };

// Procedure to find the mean and standard deviation of the spacing between the fracture
// intersections in a borehole.

Stat Borehole :: find_mean_sd_spacing ()
{ Node_frac* current = head_frac;
  int i;
double total = log.length;
double meanS=0., sdS=0.;
double spacing;
if (Nfrac>1)
  { for (i=1; i<Nfrac; ++i)
  { spacing = current->next_frac->t - current->t;
meanS += spacing;
sdS += spacing*spacing;
current = current->next_frac; };
meanS /= (Nfrac -1);
if (Nfrac == 2)
 sdS = 0.;
else
 sdS = sqrt ((sdS - (Nfrac-1)*meanS*meanS) / (Nfrac -2)); };
Stat spacingMSD (meanS, sdS, total);
return spacingMSD; };

// Procedure to print the polygons-fractures of a borehole for graphical output

void Borehole :: print()
{ int i;
Node_frac* current = head_frac;
if (current->fracture)
  {
for (i=0; i<Nfrac-1; ++i)
  {current->fracture->print();
  out<<", ";
  current=current->next_frac;};
current->fracture->print(); };
return;
};
```

File **random.C**
(page 1 / 2)

```
#include <math.h>
#include <iostream.h>
#include <stdlib.h>

#include "polar.h"
class Polar;

int rand(void);

// Procedure to generate a random number between 0 and 1

double Random01 ()
   { double  random_number =  (float) rand() / (float) 32767;
     return  random_number; };


// Procedure to generate a random number between 0 and a specified value

double Random0a (double a)
   { double random_number = a * (float) rand() / (float) 32767;
     return  random_number; };

// Procedure to generate a random number between two numbers b and c

double  RandomBC (double  b, double c)
    {  if (c<b)
       {float temp=c;
        c=b;
        b=temp;}
    double  random_number = b+(c-b) * (float) rand() / (float) 32767;
   return random_number; };

// Procedure to generate an orientation in spherical coordinates (phi,theta)
// according to a uniform distribution on a unit hemisphere

Polar ran_uniform_orientation(void)
   {  double fromX = RandomBC (-PI,PI);
      double fromZ = Random0a (HalfPI);
      Polar orientation (fromX, fromZ);
return orientation; };

// Procedure to generate an orientation in spherical coordinates (phi,theta)
// according to a uniform distribution on a unit hemisphere where theta
// varies between -PI and PI, and phi between 0 and PhiMax < HalfPI

Polar uniform_max_phi_orient(double PhiM)
 { double fromX = RandomBC (-PI, PI);
   double fromZ = Random0a (PhiM);
   Polar orientation (fromX, fromZ);
   return orientation;  };
```

```
// Procedure to generate constant orientation equal to the mean orientation
// in spherical coordinates (MeanTheta, MeanPhi)

Polar constant_orientation(void)
   { Polar orientation;
    return orientation; };

// Procedure to generate an orientation according to a univariate Fisher
// distribution on a unit hemisphere

Polar Fisher_orientation (double k)
   {double fromX = RandomBC (-PI, PI);
    double fromZ = acos(1/k*log(exp(k)-(Random01())*(exp(k)-1)));
    Polar orientation (fromX, fromZ);
    return orientation;  };

// Procedure to generate an orientation according to a bivariate Fisher PDF on a unit hemisphere

Polar bivariate_Fisher_orientation (double k1, double k2)
    {double fromX = RandomBC (-PI, PI);
     double K = k1*sin(fromX)*sin(fromX) + k2*cos(fromX)*cos(fromX);
     double fromZ = acos(1/K*log(exp(K)-(Random01())*(exp(K)-1)));
     Polar orientation (fromX, fromZ);
 return orientation; };

// Procedure to generate a value (distance, area or other continuous variable) according to
// an exponential distribution with density lambda. A  random number is generated uniformly
// between 0 and 1, and then the cumulative exponential distribution is used to back-calculate
// an exponentially distributed value. exp. pdf(rv)=lambda*exp(-lambda*rv);
// cum. PDF(rv)=1-exp(-lambda*rv);

double exp_value (double lambda)
{ double y = Random01 ();
  double RV = -(log(1-y))/lambda;
   return RV; };

// Procedure to calculate a random Poisson number with expected value N=lambda*A, where
// lambda is the density of the process (mean occurrence per unit area (or per unit distance,
// etc.), and A is the total area (or distance, etc.). This procedure is used mainly to calculate
// the random number of lines which will produce a line tessellation of intensity
// lambda over a polygon with area A.

int PoissonN (double lambda, double Area)
{ int PN = 0;
double sumA = 0.;
do {
PN++;
sumA += exp_value (lambda); }
while (sumA < Area);
return (PN-1); };
```

File **divide.C**
(page 1 / 5)

```
#include "line.h"
#include "polygon.h"
#include "listpol.h"

#define TRUE 1
#define FALSE 0
#define boolian int
#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)

#define x1(a, b, f)  (-b+sqrt(f))/(2*a)
#define x2(a, b, f)  (-b-sqrt(f))/(2*a)

extern double MeanArea;
double Random0a (double a);
double exp_value (double lambda);
int PoissonN (double lambda, double Area);
extern double AT;

// Procedure to create a 2D line from an angle alpha, distance D and a  circle radius R  and center
// given by a Point C.  The line equation is X*cos(alpha) + Y*sin(alpha) = D. The two end
// points are calculated to lie on the circle. Z coordinates are ZERO.

Line :: Line (double angle, double D, double R, Point& C)
{ end1.Y = C.Y + R;
end2.Y = C.Y - R;
end1.X = C.X + (D-R*sin(angle))/cos(angle);
end2.X = C.X + (D+R*sin(angle))/cos(angle);
end1.Z = end2.Z = C.Z;
length = 2*(sqrt(R*R - D*D));
vector.X = -sin(angle);
vector.Y = cos(angle);
vector.Z = 0.; };


// Procedure to find out if a line intersects a polygon in 2d.

boolian Polygon :: if_line_intersects (Line& l)
  { int i;
Point* current = head;
for (i=0; i<noP; ++i)
  { Line temp (*current, *current->next);
if (temp.intersect (l))
    return TRUE;
current=current->next;
  };
return FALSE;
  };
```

File **divide.C**
(page 2 / 5)

```
// Procedure to find out how a line intersects a polygon. Returns 0 if they don't intersect,
// 1 if one side of the polygon is intersected by the line, 2 if two sides are intersected, and 3
// if the line is inside the polygon. The line and the polygon are 2D and lie in the SAME
// PLANE. It is essential  to have calculated the center of the polygon correctly.

int Polygon :: how_line_intersects (Line& l)
  { int i=0, c1=0, c2=0, c=0;
Point* current = head;
Line line1 (center, l.end1);
Line line2 (center, l.end2);
for (i=0; i<noP; i++)
  { Line temp (*current, *current->next);
    if (temp.intersect (line1))
     c1++;
if (temp.intersect (line2))
     c2++;
if (temp.intersect (l))
     c++;
current=current->next; };
if (!c1 && !c2)
  return 3;          // the line lies inside the polygon
if ((!c1 && c2) || (!c2 && c1))
  return 1;       // one end of the line is inside the polygon,
                  // the other end is outside
if (c1 && c2)
  { if (c)
    return 2;        // the line intersects two sides of the polygon
    else
     return 0;};      // the line does not intersect the polygon
  };

// Procedure to divide a polygon into two polygons by a line that intersects
// it (first make sure the line intersects two sides of the polygon using
// the function above. One of the two new polygons is returned, the other new
// polygon has replaced the original polygon. POLYGON AND LINE ARE IN 2D.

Polygon Polygon :: divide_by_line (Line& L)
  { Point* new_points = new (Point[4]);
Point* current = head;
int i, j=0;
for (i=0; i<noP && j<2; ++i)
  { Line a_line (*current,*current->next);
    if (L.intersect(a_line))
  {    new_points[j] = L.intersection_with_line (a_line);
       new_points[j].next = current->next;
       current->next = new_points+ j;
       j++;
   current=(current->next)->next;}
else
  current=current->next; };
```

```
noP+=4;
new_points[2] = new_points[0];
new_points[3] = new_points[1];
Polygon* new_pol = new Polygon ();
new_pol->head = new_points+2;
new_pol->head->next = new_points[0].next;
new_points[3].next = new_points[1].next;
new_points[1].next = new_pol->head;
new_points[0].next = new_points+3;

current=new_pol->head;
new_pol->noP = 0;
do {
new_pol->noP++;
current=current->next; }
while (!(current==new_pol->head));
noP -= new_pol->noP;
new_pol->setPole = setPole;
new_pol->Pole = Pole;
new_pol->strike = strike;
new_pol->dip = dip;
return (*new_pol); };


// Procedure to tessellate a 2D polygon with Poisson lines. The following  property of a line
//  process is used: if the line intensity is lambda (i.e.  the expected number of Poisson lines
//  is lambda*the area), then the ordered distances from an arbitrary point to the lines forms
//  a Poisson process with intensity 2*lambda (the distance increments are exponential). A Poisson
//  line is produced by generating an angle alpha uniformly between 0  and TwoPI, and  a distance
// as a sum of exponential distances with point intensity 2*lambda until the cumulative distance
// becomes larger than  the radius of the surcumscribed circle. Thus the number of lines is Poisson.


ListPolygons Poisson_lines_on_polygon (Polygon& initial, double lambda) {
ListPolygons* lp = new ListPolygons;
Node* current;
int i, j, lineInt;
double R = initial.minR_inscribe();
double expD = exp_value (lambda);
double cumD = -R + expD;
double angle;
lp->add_polygon (initial);
double xx = initial.center.X;
double yy = initial.center.Y;
double zz = initial.center.Z;
Point C (xx, yy, zz);
int k =0;
do {
++k;
angle = Random0a (TwoPI);
```

File **divide.C**
(page 4 / 5)

```
Line L (angle, cumD, R, C);
current = lp->head_pol;
j = lp->Npol;
for (i=0; i<j; ++i)
  {
if ( current->content->if_line_intersects (L))
     {Polygon* new_pol = new Polygon;
      *new_pol = current->content->divide_by_line (L);
      new_pol->find_area_radius_2d ();
      new_pol->find_center();
      current->content->find_area_radius_2d ();
      current->content->find_center();
      lp->add_polygon (*new_pol);     };
      current = current->next_pol;};
expD = exp_value (lambda);
cumD += expD;
}
while (cumD < R);
return *lp;
};


// Procedure to perform a fractal tessellation to a polygon which is divided  into polygons already.
// The same line tessellation which has been done to the original polygon is performed into all
// of the polygons of the list until the polygons become too small to be affected by a line
// network with the given intensity.

void ListPolygons :: fractal_tessellation (double lambda, double fractal)
{
Node* current = head_pol;
int i, j;
int N = Npol; cout<<"beginning "<<N<<endl;
ListPolygons* new_list = new ListPolygons;
for (i=0; i<Npol; ++i)
  {
    ListPolygons* new_lp = new ListPolygons;
*new_lp = Poisson_lines_on_polygon (*current->content, fractal*lambda);
new_list->add_listpol (*new_lp);
 current = current->next_pol;
 };
*this = *new_list;
  int N1 = Npol; cout<<" end "<<N1<<" new "<<N1-N<<endl;
return;
};
```

```
// Procedure to perform a fractal tessellation to a polygon which is divided  into polygons already.
//  The same line tessellation which has been done to the original polygon is performed into all of
// the polygons of the list until the polygons become too small to be affected by a line  network
// with the given intensity.

void ListPolygons :: fractal_big_and_small (double lambda, double fractal)
{
Node* current = head_pol;
int i, j;
int N = Npol; cout<<"beginning "<<N<<endl;
ListPolygons* new_list = new ListPolygons;
for (i=0; i<Npol; ++i)
  {
if (current->content->area < AT*MeanArea)  {
ListPolygons* new_lp = new ListPolygons;
*new_lp = Poisson_lines_on_polygon (*current->content, fractal*lambda);
new_list->add_listpol (*new_lp); }
else  {
Polygon* new_pol = current->content;
new_list->add_polygon (*new_pol); };
current = current->next_pol;
  };
*this = *new_list;
   int N1 = Npol; cout<<" end "<<N1<<" new "<<N1-N<<endl;
return;
};

// Procedure to perform a fractal tessellation to a polygon which is divided  into polygons already.
//  The same line tessellation which has been done to the original polygon is performed into all of
// the polygons of the list until the polygons become too small to be affected by a line network
// with the given intensity.

void ListPolygons :: fractal_similar (double lambda, double fractal)
{
Node* current = head_pol;
int i, j;
int N = Npol; cout<<"beginning "<<N<<endl;
ListPolygons* new_list = new ListPolygons;
for (i=0; i<Npol; ++i)
  { if (current->content->area> AT*MeanArea)
 { ListPolygons* new_lp = new ListPolygons;
*new_lp = Poisson_lines_on_polygon (*current->content, fractal*lambda);
new_list->add_listpol (*new_lp); }
else  {
Polygon* new_pol = current->content;
new_list->add_polygon (*new_pol); };
current = current->next_pol; };
*this = *new_list;
   int N1 = Npol; cout<<" end "<<N1<<" new "<<N1-N<<endl;
return; };
```

File **initial.C**
(page 1 / 3)

```
#include "point.h"
#include "polygon.h"
#include "plane.h"
#include "volume.h"

#include <iostream.h>
#include <fstream.h>
extern ofstream out;
extern double Xm, Ym;
extern int Np;

#define x1(a, b, f)  (-b+sqrt(f))/(2*a)
#define x2(a, b, f)  (-b-sqrt(f))/(2*a)


// A function to create the polygon which a plane P cuts from the modeling  volume V.
// If p intersects the top quadratic surface of V, then the curve of intersection is approximated by
// N straight line segments between points N+1 on the surface. Xm and Ym
// (also -Xm and -Ym) are the lateral boundaries of the volume.

Polygon make_initial (Plane& p, Volume& v) {
Polygon initial (p);
double A = p.A, B=p.B, C=p.C, D=p.D; int i;
double xx, yy, zz, a, b, c, d, e, f, aa, bb, BB, CC, ff;
a=v.top.A; b=v.top.B; c=v.top.C; d=v.top.D; e=v.top.E; f=v.top.F;
                          // First calculate points of intersections,
                          // if any, with the lateral planes at elevation 0.
if (B)                    // Possible points of intersection
  { for (i=0; i<2; ++i)          // with x=Xm and x=-Xm at z=0.
    { if (!i)
        xx=Xm;
      else
        xx=-Xm;
yy=(D-A*xx)/B;
    if (yy>=-Ym && yy<=Ym)
          {Point* P = new Point (xx, yy, 0.);
        initial.add_point (*P);
            if (!p.C)                 // if the plane is vertical
              {double zz1=v.top.find_Z_on_surface (xx, yy);
               Point* P1=new Point (xx, yy, zz1);
               initial.add_point (*P1);};    }; };};
if (A)                          // Possible points of intersection
  { for (i=0; i<2; ++i)         // with y=Ym and y=-Ym
    {if (!i)
        yy=Ym;
      else
        yy=-Ym;
xx=(D-B*yy)/A;
    if (xx>=-Xm && xx<=Xm)
          {Point* P=new Point (xx, yy, 0.);
```

File **initial.C**
(page 2 / 3)

```
      initial.add_point (*P);
          if (!p.C)                              // if the plane is vertical
            {double zz1=v.top.find_Z_on_surface (xx, yy);
             Point* P1=new Point (xx, yy, zz1);
             initial.add_point (*P1);};    }; };};
if (C)                                  // if the plane is not vertical
  { for (i=0; i<4; ++i)                 // checking if it intersects the vertical edges
   {xx=v.P[i].X; yy=v.P[i].Y;
    zz=(D-A*xx-B*yy)/C;
    if (zz<=v.P[i].Z && zz>=-0.)
     {Point* P=new Point (xx, yy, zz);
      initial.add_point (*P);}; }; };
if (C&&A || C&&B)
  {    { for (i=0; i<2; ++i)             // Possible ntersections with
     {if (!i)                           // the top surface of the volume
          xx=Xm;                        // at x=Xm and x=-Xm
     else
       xx=-Xm;
aa=(D-A*xx)/C; bb=B/C;
BB=(bb+b*xx+e); CC=(a*xx*xx+d*xx-aa+f);
if (c)
  {   ff=BB*BB-4*CC*c;
      if (ff>=0.)
     {yy=x1(c, BB, ff);  zz=aa-bb*yy;
      if (yy>=-Ym && yy<=Ym)
           {Point* P = new Point (xx, yy, zz);
       initial.add_point (*P);}; };
   if (ff>0.)
     {yy=x2(c, BB, ff);  zz=aa-bb*yy;
      if (yy>=-Ym && yy<=Ym)
           {Point* P = new Point (xx, yy, zz);
       initial.add_point (*P);}; }; }
else if (BB)
  {yy=-CC/BB;  zz=aa-bb*yy;
   if (yy>=-Ym && yy<=Ym)
           {Point* P = new Point (xx, yy, zz);
           initial.add_point (*P);}; }; };};

   { for (i=0; i<2; ++i)                // Possible intersections with
     {if (!i)                          // the top surface of the volume
          yy=Ym;                       // at y=Ym and y=-Ym
   else
      yy=-Ym;
aa=(D-B*yy)/C; bb=A/C;
BB=(bb+b*yy+d); CC=(c*yy*yy+e*yy-aa+f);
if (a)
  {   ff=BB*BB-4*CC*a;
      if (ff>=0.)
     {xx=x1(a, BB, ff);  zz=aa-bb*xx;
      if (xx>=-Xm && xx<=Xm)
```

```
            {Point* P = new Point (xx, yy, zz);
          initial.add_point (*P);}; };
    if (ff>0.)
      {xx=x2(a, BB, ff);  zz=aa-bb*xx;
       if (xx>=-Xm && xx<=Xm)
            {Point* P = new Point (xx, yy, zz);
            initial.add_point (*P);}; }; }
else if (BB)
  {xx=-CC/BB;  zz=aa-bb*xx;
   if (xx>=-Xm && xx<=Xm)
            {Point* P = new Point (xx, yy, zz);
            initial.add_point (*P);}; }; };};
  };
initial.find_center();
return initial; };


// Function to add N points to a side of a polygon which is not linear but lies on a surface.
// The curved side is approximated by N+1 linear segments.

void Polygon::add_points_on_surface (Plane& pl, Surface& s, int N, double Zm) {
double a=s.A, b=s.B, c=s.C, d=s.D, e=s.E, f=s.E;
if (a || b || c)
  { Point* current = head;
    int i, j;
double xx, yy, zz, x, y, z, ff;
double A=pl.A, B=pl.B, C=pl.C, D=pl.D;
Cartesian CB = pl.find_binormal();
for (i=0; i<noP; i++)
  { if (s.is_point_on_surface(*current) && s.is_point_on_surface(*current->next))
{ xx =(current->next->X - current->X) / ((double)N+1.);
 yy = (current->next->Y - current->Y)  / ((double)N+1.);
 zz = (current->next->Z - current->Z)  / ((double)N+1.);
x = current->X; y=current->Y; z=current->Z;
for (j=0; j<N; ++j)
   {x+=xx; y+=yy; z+=zz;
    Point p (x, y, z);
    Line l (p, CB, Zm);
    Point *P = new Point;
    *P = s.intersection_by_line (l);
    P->next = current->next;
    current->next = P;
    noP ++;
    current=current->next;
   };
  return; }
else
    current=current->next;
  }; };
return; };
```

File **rotation.C**
(page 1 / 5)

```
#include "cartesian.h"
#include "point.h"
#include "polygon.h"
#include "listpol.h"
extern ofstream out;
extern double Xm, Ym;
extern double MeanArea;
double Random01 ();

double  RandomBC (double, double);

// Procedure to mark the polygon so that it is subparallel to  the strike of a surface at its center.
// The strike of a surface  is calculated with function of class Surface. A ration Ps of polygons
// is marked as fractures if the angle between the strike of the polygon and the strike of the surface
// does not exceed the specified angle. Angle between the strike is calculated using the dot
// product between two unit vectors: cos angle = a1*a2 + b1*b2 + c1*c2.

boolian Polygon :: mark_parallel_to_strike (double Sstrike, double angle, double Ps)
{ double cosD = cos(Sstrike)*cos(strike)+sin(Sstrike)*sin(strike);
if ((cosD >= cos(angle) && cosD <=1 || cosD<=-cos(angle) && cosD>=-1.) \
&& Random01 () < Ps)
 return TRUE;
else
 return FALSE; };

// Procedure to mark the polygon so that it is sub orthogonal  to the strike of a surface at its center.
// The strike of a surface is calculated with function of class Surface. A ration Po of polygons
// is marked as fractures if the angle between the strike of the polygon and the strike of the surface
// deviates from PI/2 not more that a specified  angle. Angle between the strike is calculated using
// the dot product between two unit vectors: cos angle = a1*a2 + b1*b2 + c1*c2.

boolian Polygon :: mark_orthogonal_to_strike (double Sstrike, double angle, double Po)
{ double cosD = cos(Sstrike)*cos(strike)+sin(Sstrike)*sin(strike);
if ( cosD >= -sin(angle) && cosD <= sin(angle) && Random01() < Po)
 return TRUE;
else
 return FALSE; };

// Procedure to mark the polygon accoding to its dip compared to the dip of a surface. Both dips
// are between zero and PI/2. Return True if the dip difference is smaller than teh specified angle.

boolian Polygon :: mark_by_dip (double Sdip, double angle, double P)
{
double cosD = cos(Sdip)*cos(dip)+sin(Sdip)*sin(dip);
if ( cosD >= -sin(angle) && cosD <= sin(angle) && Random01() < P)
 return TRUE;
else
 return FALSE;
};
```

```
// Procedure to rotate a polygon. The arguments are the new latitude and azimuth (theta and phi)
// of the polygon pole in the absolute frame of reference.

void Polygon :: rotate_by_strike (double NewStrike)
{ Cartesian V(center.X, center.Y, center.Z);
(*this)-V;
make_polygon_2d();
if (RandomBC(-1., 1.)>= 0.)
   { Pole.theta = NewStrike + HalfPI;
     if (Pole.theta > PI)
    Pole.theta -= TwoPI; }
else
{ Pole.theta = NewStrike - HalfPI;
if (Pole.theta < PI)
    Pole.theta += TwoPI; };
make_polygon_3d();
strike = NewStrike;
(*this)+V;
return; };

// Procedure to rotate a polygon. The arguments are the new latitude and
// azimuth (theta and phi) of the polygon pole in the absolute frame of reference.

void Polygon :: rotate_by_dip (double NewDip)
{ Cartesian V(center.X, center.Y, center.Z);
(*this)-V;
make_polygon_2d();
Pole.phi = NewDip;
make_polygon_3d();
(*this)+V;
return; };

// Procedure to mark the polygons in a list according to their strike compared to the strike of a
// surface at their centers. Polygons are retained in the list if the difference in strike does not
// exceed some specified angle.

void ListPolygons :: mark_parallel_to_strike (Surface& S, double angle, double Pc)
{ Node* current = head_pol;
int i=0;
Polar SD;
for (i=0; i<Npol; ++i)
{  SD = S.strike_dip_at_point (current->content->center);
if (!(current->content->mark_parallel_to_strike (SD.theta, angle, Pc)))
  {
double angle1 = RandomBC(-angle, angle) + SD.theta;
current->content->rotate_by_strike (angle1);  };
 current = current->next_pol;
};
return;
};
```

File **rotation.C**
(page 3 / 5)

```
// Procedure to mark the polygons in a list according to their strike compared to the strike
// of a surface at their centers. Polygons are  retained in the list if the difference in strike does not
// exceed some specified abgle.


void ListPolygons :: mark_orthogonal_to_strike (Surface& S, double angle, double Pr)
{ Node* current = head_pol;
int i=0;
Polar SD;
for (i=0; i<Npol; ++i)
{  SD = S.strike_dip_at_point (current->content->center);
    if (!(current->content->mark_orthogonal_to_strike (SD.theta, angle, Pr)))
  {   double angle1 = RandomBC(-angle, angle) + SD.theta + HalfPI;
current->content->rotate_by_strike (angle1); };
 current = current->next_pol; };
return; };

// Procedure to mark the polygons according to the strike of a surface.
// If the polygon is either sub parallel or sub orhtogonal to the strike
// of the surface, its orientation is kept. Otherwise, the polygon is
// rotated to be sub parallel if the character option is C, or sub orthogonal
// if the character option is R



void ListPolygons :: mark_by_strike (Surface& S, double angleR, double angleC, \
double angle2, char option, double Pr, double Pc)
{
Node* current = head_pol;
int i=0;
Polar SD;
for (i=0; i<Npol; ++i)
{  SD = S.strike_dip_at_point (current->content->center);
if (!(current->content->mark_orthogonal_to_strike (SD.theta, angleR, Pr)) && !(current-
>content->mark_parallel_to_strike (SD.theta, angleC, Pc)))
  {
double new_strike;
if (option == 'c')
new_strike = RandomBC(-angle2, angle2) + SD.theta;
if (option == 'r')
new_strike = RandomBC(-angle2, angle2) + SD.theta + HalfPI;
if (new_strike > PI)
  new_strike -= TwoPI;
if (new_strike < -PI)
  new_strike += TwoPI;
current->content->rotate_by_strike (new_strike);  };
 current = current->next_pol; };
return;
};
```

```
// Procedures to mark the polygons in alist according to their dips. If the dips are close to that
// of the surface, they are kept. Otherwise the polygons are rotated

void ListPolygons :: mark_by_dip (Surface& S, double angle1, double angle2, double P)
{
Node* current = head_pol;
int i=0;
Polar SD;
for (i=0; i<Npol; ++i)
{  SD = S.strike_dip_at_point (current->content->center);
if (!(current->content->mark_by_dip (SD.phi, angle1, P) ))
  {    double new_dip = RandomBC(-angle2, angle2) + PI/2 - SD.phi;
current->content->rotate_by_dip (new_dip); };
 current = current->next_pol;
};
return;
 };


void ListPolygons :: mark_by_dip (Cubic& C, double angle1, double angle2, double P)
{
 Node* current = head_pol;
 int i=0;
Polar SD;
for (i=0; i<Npol; ++i)
{
SD = C.strike_dip_at_point (current->content->center);
double Ctheta = SD.theta + HalfPI;
  if (Ctheta > PI)
      Ctheta -= TwoPI;

if (!(current->content->mark_by_dip (SD.phi, angle1, P) ))
  {
   double new_dip = RandomBC(-angle2, angle2) - SD.phi + HalfPI;
 if ( (current->content->Pole.theta)*Ctheta > 0. )
  {   if ( current->content->Pole.theta >= 0.)
        current->content->Pole.theta -= PI;
     else
        current->content->Pole.theta += PI;   };
current->content->rotate_by_dip (new_dip); };
 current = current->next_pol; };
return;
};
```

File **rotation.C**
(page 5 / 5)

```
// Procedure to mark the polygons according to the strike of a CUBIC surface.
// If the polygon is either sub parallel or sub orhtogonal to the strike
// of the surface, its orientation is kept. Otherwise, the polygon is
// rotated to be sub parallel if the character option is C, or sub orthogonal
// if the character option is R


void ListPolygons :: mark_by_strike (Cubic& C, double angleR, double angleC, \
double angle2, char option, double Pr, double Pc)
{
Node* current = head_pol;
int i=0;
Polar SD;
for (i=0; i<Npol; ++i)
{
SD = C.strike_dip_at_point (current->content->center);
if (!(current->content->mark_orthogonal_to_strike (SD.theta, angleR, Pr)) && !(current-
>content->mark_parallel_to_strike (SD.theta, angleC, Pc)))
  {
double new_strike;
if (option == 'c')
new_strike = RandomBC(-angle2, angle2) + SD.theta;

if (option == 'r')
new_strike = RandomBC(-angle2, angle2) + SD.theta + HalfPI;

if (new_strike > PI)
  new_strike -= TwoPI;
if (new_strike < -PI)
  new_strike += TwoPI;

current->content->rotate_by_strike (new_strike); };
 current = current->next_pol;
};
return;
};
```

File **zones.C**
(page 1 / 2)

```
#include "polygon.h"
#include "listpol.h"

#include <math.h>
#include <iostream.h>
#include <stdlib.h>
#define FALSE 0
#define TRUE 1

double Random01 ();

// Procedure to mark a polygon with a zone probability Pi. The shortest  distance D to a list
// of polygons is found. Then this distance is compared to distances defined by the array
// zones to find out in which zone i  is teh polygon. Then the polygon is marked as fracture with a
// marking probability Pi for zone i, defined by the array marks. The function returns TRUE
// if a generated random number is smaller than Pi, or FALSE if the number is larger than Pi.
// Pi and the random number are between 0. and 1.

boolian ListPolygons :: if_zone_mark_polygon (Polygon& pol, int Nzones, double* zoneDmax,
double* zoneP)
{
double D = shortest_distance_from_polygon (pol);
double mark = *(zoneP+Nzones-1);
int MARKED = FALSE;
int i;
for (i=0; i<Nzones-1 && !MARKED; ++i)
  {
if (D < *(zoneDmax+i))
  { mark = *(zoneP+i);
  MARKED = TRUE; }; };

double RN = Random01();
if (RN <= mark)
 return TRUE;
else
  return FALSE;
};


// Procedure to mark the polygons in a list of polygons with zone
// probabilities according to the distances to another list of polygons.
// The argument ListPolygons are teh polygons (usually major faults) the
// distances from which define the zones. The list of polygons that calls
// the function (usually new fractures) are the ones that are being marked.

void ListPolygons :: mark_by_zones (ListPolygons& lp, int Nzones, double* zoneDmax, \
double* zoneP)
{
Node* current = head_pol;
while (current -> next_pol)
```

```
 { if (!(lp.if_zone_mark_polygon (*current->next_pol->content, Nzones, zoneDmax, zoneP)))
 { if (current->next_pol->next_pol)
            delete current->next_pol->content;
 current->next_pol = current->next_pol->next_pol;
 -- Npol; }
else
 current = current->next_pol; };
if (!(lp.if_zone_mark_polygon (*head_pol->content, Nzones, zoneDmax, zoneP)))
 {head_pol = head_pol->next_pol;
 -- Npol; };
return; };


// Function to consider the size of a polygon in the pdf of fracture sizes.
// The function finds the specified "zone" by polygon size and increases the
// number of polygons that have such size specified by Amin < Apol < Amax.

void Polygon :: include_in_size_pdf (int Nzones, int* Nin_zones, double* maxAratio, double
MeanA)
{ int i;
for (i=0; i<Nzones; ++i)
 { if ((area/MeanA) < *(maxAratio+i))
 { ++ *(Nin_zones+i);
  return;}; };
++ *(Nin_zones + Nzones);
return; };

// Function to find the size distribution of a list of polygons

void ListPolygons:: size_distribution (int Nzones, int* Nin_zones, double* maxAratio, double
MeanA)
{
int i;
for (i=0; i<Nzones; ++i)
 Nin_zones[i] = 0;
Node* current = head_pol;

while (current)
 { current->content->include_in_size_pdf (Nzones, Nin_zones, maxAratio, MeanA);
current = current->next_pol; };
return;
};
```

File **intersections.C**
(page 1 / 5)

```
#include "point.h"
#include "line.h"
#include "surface.h"
#include "plane.h"
#include "polygon.h"
#include "listpol.h"
#include "listline.h"

#define x1(a, b, f)  (-b+sqrt(f))/(2*a)
#define x2(a, b, f)  (-b-sqrt(f))/(2*a)

// Procedure to calculate coordinates of a point where a line intersects a surface. First use the
// procedures in "surface.C" to find out if the line intersects the surface at all. Calculations in 3D.

Point Surface :: intersection_by_line (Line& L) {
Point* P = new Point;
double t, aa, bb, cc, ff;
double a = L.vector.X, b = L.vector.Y, c = L.vector.Z;
double X1 = L.end1.X, Y1 = L.end1.Y, Z1=L.end1.Z;
aa = A*a*a + B*a*b + C*b*b;
bb = 2*A*a*X1 + B*a*Y1 +B*b*X1 + 2*C*b*Y1 + D*a + E*b - c;
cc = A*X1*X1 + B*X1*Y1 + C*Y1*Y1 + D*X1 + E*Y1 + F - Z1;
if (!aa)
  {t = -cc/bb;  P->X = a*t+X1; P->Y = b*t+Y1; P->Z = c*t+Z1;
   return *P; }
else   {     ff = bb*bb - 4*aa*cc;
if (ff >= 0.)
  {  t = x1(aa, bb, ff);
 if(t>=0. && t<=L.length)
   {P->X = a*t+X1; P->Y = b*t+Y1; P->Z = c*t+Z1;  return *P; }
 else  {   t = x2(aa, bb, ff);
  P->X = a*t+X1; P->Y = b*t+Y1; P->Z = c*t+Z1;
  return *P; }; }; }; };

// Function to calculate if a plane intersects a line segment. The parametric  equation of a line
// is used: x = X + at, y = Y + bt, z = Z + ct, where P(X, Y, Z) is a point on the line, (a, b, c) is
// the vector parallel to the line. If P is the first end of the segment, then the line intersects
// the plane if t is between 0 and the length.

boolian Line :: if_intersects_plane (Plane& p)
{ double a=vector.X, b=vector.Y, c=vector.Z;
double abc = p.A*a + p.B*b + p.C*c;
if (abc>= -0.00001 && abc<=0.00001)
  return FALSE;                  // the line is parallel to the plane
else
   { double t = (p.D - p.A*end1.X - p.B*end1.Y - p.C*end1.Z)/abc;
if (t>=0. && t<=length)
    return TRUE;
else
    return FALSE; }; };
```

File **intersections.C**
(page 2 / 5)

```
// Procedure to find the point of intersection of a line segment and a plane.
// First use the procedure to find out if they intersect at all.

Point Line :: intersection_with_plane (Plane& p)
{ double t = (p.D-p.A*end1.X-p.B*end1.Y-
p.C*end1.Z)/(p.A*vector.X+p.B*vector.Y+p.C*vector.Z);
double xx = end1.X + vector.X*t;
double yy = end1.Y + vector.Y*t;
double zz = end1.Z + vector.Z*t;
Point* intersection = new Point(xx, yy, zz);
return *intersection; };

// Procedure to find out if a line intersects a polygon. Temporarily creates a plane through the
// polygon. If the line intersects the plane, the point of intersection is calculated.
// If the point is inside the polygon, the function returns TRUE.

boolian Polygon :: if_3d_line_intersects (Line& L)
{ Cartesian N (Pole);
Plane p (N, *head, setPole);
if (!(L.if_intersects_plane (p)))
  return FALSE;
else {
int how = TRUE;
Point P = L.intersection_with_plane (p);
Cartesian temp (P.X, P.Y, P.Z);
temp = temp.local_coordinates (Pole);
P.X = temp.X; P.Y=temp.Y; P.Z = temp.Z;
make_polygon_2d();
Line a_line (P, center);
if (if_line_intersects(a_line))
    how = FALSE;
make_polygon_3d();
return how; }; };

// Procedure to find the point of intersection of a line with a polygon.
// First use the procedure above to find out if the line intersects the polygon
// at all. If the line intersects the polygon, then there is only one
// point where they intersect, and it is the point where the line segment
// intersects the plane of the polygon.

Point Polygon :: intersection_with_line (Line& L)
{
Cartesian N (Pole);
Plane p (N, *head, setPole);
Point* new_point = new Point;
*new_point = L.intersection_with_plane (p);
return *new_point;
};
```

```
// Procedure to find out if a polygon intersects another polygon in 3D.

boolian Polygon :: if_polygon_intersects (Polygon& pol)
{ Cartesian N (Pole);
Plane p (N, *head, setPole);
if (pol.if_intersects_plane(p))
  { Line L = pol.intersection_with_plane (p);
make_polygon_2d();
L.local_coordinates(Pole);
if (if_line_intersects (L))
  { make_polygon_3d();
  return TRUE; }
else
  make_polygon_3d();};
return FALSE; };

// Procedure to find out if a plane intersects a polygon. Returns TRUE if
// any of the sides of the polygon intersects the plane.

boolian Polygon :: if_intersects_plane(Plane& p)
{ Point* current = head;
int i;
for (i=0; i<noP; ++i)
  { Line temp (*current, *current->next);
    if (temp.if_intersects_plane (p))
        return TRUE;
else
   current = current->next; };
return FALSE; };

// Procedure to calculate the line of intersection of a polygon with a plane.
// First use the procedure above to find out if they intersect at all.

Line Polygon :: intersection_with_plane (Plane& p)
{ Point* current = head;
   int i, j=0;
   Point first, second;
for (i=0; i<noP && j<2; ++i)
  { Line l(*current, *current->next);
if (l.if_intersects_piane (p))
{ if(!j)
   first = l.intersection_with_plane (p);
  else
   second = l.intersection_with_plane (p);
++j; };
current = current->next; };
Line* trace = new Line(first, second);
return *trace;
};
```

```
// Procedure to calculate the list of lines which are the intersections of
// the list of polygons with a plane.

ListLines ListPolygons :: traces_on_plane (Plane& p)
{
ListLines* traces = new ListLines;
int i;
Node* current = head_pol;
for (i=0; i<Npol; ++i)
  {
if (current->content->if_intersects_plane (p))
   { Line* new_line = new Line;
     *new_line = current->content->intersection_with_plane(p);
     traces->add_line (*new_line);};
current = current->next_pol;  };
return *traces;
};


// Procedure to find where a polygon is located relative to a surface.
// The function returns 1 if the polygon (all of its vertices) is entirely
// above the surface, -1 if it is entirely below the surface, or 0 if
// it intersects the surface.

int Polygon :: above_or_below_surface (Surface& s)
{
  Point* current = head;
  int i, counter =0;
  int how;
for (i=0; i<noP; ++i)
  { how = s.is_point_above_below (*current);
counter += how;
current = current->next; };

if (counter == noP )
  return 1;              // All vertices are above the surface
if (counter == -noP)
  return -1;             // All vertices are below the surface
else
  return 0;              // The surface intersects the polygon
};
```

```
// Procedure to cut a polygon by a surface. The new polygon is the portion
// above the surface if the character argument is 'A' or 'a', or the portion
// below the surface if the character constant is 'B' or 'b'. Use only after the function
// above has returned 0 for the polygon, i.e. if the polygon is intersected for sure by the surface.
// After returning from the function make sure to use the function for calculating area, center,
// and radius of the new polygon.

void Polygon :: cut_by_surface (Surface& s, char how)

{
int how_head = s.is_point_above_below (*head);
Point* current = head;
int i, j=0;
Point* new_points = new (Point[2]);
for (i=0; i<noP && j<2; ++i)
 {
Line a_line (*current,*current->next);
if (!s.how_line_intersect (a_line))
 {
  new_points[j] = s.intersection_by_line (a_line);
  new_points[j].next = current->next;
  current->next = new_points+j;
  j++;
  current=current->next->next;  }
else
  current=current->next; };

if ((how_head <=0 && (how == 'B' || how=='b')) || (how_head>=0&&(how == 'A' || how=='a')))
 { head = &new_points[0];
  head->next = &new_points[1];  }
else
 {   head = &new_points[1];
  head->next = &new_points[0]; };

current = head;
noP = 0;
do {
noP++;
current=current->next; }
while (!(current==head));
area_radius_3d();
return;
};
```

File **makefile**

```
LFLAGS =  -lm
CC = g++

geofrac:  cartesian.o point.o polar.o line.o surface.o cubic.o random.o plane.o volume.o \
polygon.o  initial.o intersections.o listpol.o divide.o listline.o rotation.o zones.o borehole.o \
cell.o main.o
        $(CC) -o geofrac cartesian.o point.o polar.o line.o surface.o cubic.o random.o\
plane.o volume.o polygon.o initial.o intersections.o listpol.o divide.o listline.o rotation.o \
zones.o borehole.o cell.o main.o ${LFLAGS}

cartesian.o: cartesian.C cartesian.h
        $(CC)  -c  $*.C
point.o: point.C point.h
        $(CC)  -c  $*.C
polar.o: polar.C polar.h
        $(CC)  -c  $*.C
line.o: line.C line.h
        $(CC)  -c  $*.C
surface.o: surface.C surface.h
        $(CC)  -c  $*.C
cubic.o: cubic.C cubic.h
        $(CC)  -c  $*.C
random.o: random.C
        $(CC)  -c  $*.C
plane.o: plane.C plane.h
        $(CC)  -c  $*.C
volume.o: volume.C volume.h
        $(CC)  -c  $*.C
polygon.o: polygon.C polygon.h
        $(CC)  -c  $*.C
initial.o: initial.C
        $(CC)  -c  $*.C
intersections.o: intersections.C
        $(CC)  -c  $*.C
listpol.o: listpol.C listpol.h stat.h
        $(CC)  -c  $*.C
divide.o: divide.C
        $(CC)  -c  $*.C
listline.o: listline.C listline.h
        $(CC)  -c  $*.C
rotation.o: rotation.C polygon.h
        $(CC)  -c  $*.C
zones.o: zones.C polygon.h listpol.h
        $(CC)  -c  $*.C
borehole.o: borehole.h borehole.C
        $(CC)  -c  $*.C
cell.o: cell.C
        $(CC)  -c  $*.C
main.o: main.C
        $(CC)  -c  $*.C
```

File **main.C**
(page 1 / 10)

```
#include <math.h>
#include <stdio.h>
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <sys/types.h>
#include <time.h>
#include <unistd.h>

#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)
#define max(a,b) ((a) > (b) ? (a) : (b))
#define min(a,b) ((a) < (b) ? (a) : (b))
double Xm, Ym;                  // Lateral boundaries of the volume
int Nsets;                      // No. of fracture sets
double A, B, C, D, E, F;        // Top surface of modeling volume
double mA, mB, mC, mD, mE, mF, mG, mH, mI, mJ;  // Cubic surface of the fold
double oAv, oBv, oCv, oDv;
double oAh, oBh, oCh, oDh;
double bX1, bX2, bY1, bY2, bZ1, bZ2;
double bFk1, bFk2, Fk, MaxPhi;
double TRatio;
double AngleMin, elongation;

double angleR, angleC, angleNew;
double ratioR, ratioC;

double MeanArea, MeanA;
ofstream out;
char strike, dip;
double angleD, angleDip, ratioD;
double ratioMA, CA, gama;


#define x1(a, b, f)  (-b+sqrt(f))/(2*a)
#define x2(a, b, f)  (-b-sqrt(f))/(2*a)

#include "polar.h"
#include "cartesian.h"
#include "point.h"
#include "line.h"
#include "surface.h"
#include "volume.h"
#include "plane.h"
#include "polygon.h"
#include "listpol.h"
#include "listline.h"
#include "borehole.h"
#include "stat.h"
```

```
time_t time(time_t *tloc);
double Random01();
double Random0a(double);
double  RandomBC(double , double);
double exp_value (double);
int PoissonN (double, double);
Polar ran_uniform_orientation();
Polar uniform_max_phi_orient(double);
Polar constant_orientation();
Polar Fisher_orientation (double);
Polygon make_initial (Plane&, Volume&);
ListPolygons Poisson_lines_on_polygon (Polygon& , double);
int Np;
int sizeNpdf;
double maxR;
double Datum;
char folding;
int Nzones, Nfaults, Nboreholes;
char gF, gH, gV, cF, cH, cV;

main ()
{
srand(time(0) * getpid());
int i;

/*************************************************************************
*          INPUT OF MODELING VOLUME FROM FILE "INPUT"
*************************************************************************/
ifstream in ("INPUT");
if (!in)
 cout<<"cannot open file"<<endl;
in>>Xm>>Ym;              // Extent of rectangular area
in>>Datum;               // Datum (Z=0) is bottom surface of modeling volume
in>>A>>B>>C>>D>>E>>F;     // Top surface

// INPUT OF TYPE OF MODEL MARKING
in>>ratioMA>>maxR;
in>>CA>>gama;     // Coefficients of the model plane, line,  and marking processes
in>>maxR;
in>>Nsets;                  // Number of fracture sets
in>>Nzones;
in>>folding;
in>>gF>>gH>>gV;          // Graphical output: gF for 3D system; gH for horizontal
                         // outcrop plane; gV for vertical outcrop plane
in>>cF>>cH>>cV;          // Coordinate output: cF for 3D system; cH for horizontal
                         // traces; gV for vertical traces
in>>oAv>>oBv>>oCv>>oDv;          // profile outcrop plane
in>>oAh>>oBh>>oCh>>oDh;          // plan view outcrop plane

in>>Nboreholes;             // how many boreholes are desired?
```

File **main.C**
(page 3 / 10)

```
in>>Np;                 // Np is the number of points on the top surface
in>>MaxPhi;             // Maximum azimuth, if uniform orientation
in>>Fk;                 // Fisher constant, if Fisher orientation
in>>AngleMin;           // Minimum allowed angle for polygon shapes
in>>elongation;         // Maximum allowed elongation of polygons
in>>strike;             // Relationship to strike: C(oncentric), R(adial)
in>>angleR>>ratioR;       // maximum angle and ratio between strike of
                          // polygon and slope of surface
in>>angleC>>ratioC;       // maximum angle and ratio between strike of
                           // polygon and strike of surface
in>>angleNew;             // new maximum angle between strike of polygon
                          // and strike (or slope) of surface
in>>dip;
in>>angleD>>ratioD;
in>>angleDip;
in>>sizeNpdf;
int Nsize_int [sizeNpdf+1];
double sizeMax [sizeNpdf];

for (i=0; i<sizeNpdf; ++i)
  {Nsize_int[i] = 0;
  in>>sizeMax[i];};
Nsize_int[sizeNpdf] = 0;
Polar MeanP[Nsets];
double FracInt[Nsets];
char pdf[Nsets];
double MeanR[Nsets];
double Ca[Nsets];
double TRatio[Nsets];

// INPUT OF PARAMETERS FOR INDIVIDUAL FRACTURE SETS
for (i=0; i<Nsets; ++i)
  {
in>>FracInt[i];           // Fracture intensity: cum frac area / volume
in>>MeanP[i].theta>>MeanP[i].phi;   // Mean oorientation of the set
in>>pdf[i];               // Spherical orientation PDF
in>>TRatio[i];             // Translation (non-coplanarity)
in>>MeanR[i];  };
in.close();

/*************************************************************************
*                CREATING THE MODELING VOLUME
*************************************************************************/
Surface ground (A, B, C, D, E, F);      // Topographic surface
Volume rock (ground);                  // Modeling volume over area Xm, Ym
double VOLUME = ground.find_enclosed_volume (Xm, Ym);

double Rmax = sqrt(Xm*Xm + Ym*Ym + rock.Zmax*rock.Zmax);
double Zm;
cout<<"Number of fracture sets "<<Nsets<<endl;
```

```
// CREATING FOLD IF ANY
Cubic fold;
if (folding == 'y' || folding == 'Y')
  {
in.open ("FOLD");
in>>fold.A>>fold.B>>fold.C>>fold.D>>fold.E>>fold.F>>fold.G>>fold.H>>fold.I>>fold.J;
                    // Input of the cubic surface of a fold
in.close();
  };

/*************************************************************************
*         INPUT OF FRACTURE ZONES (if any)  FROM FILE "ZONES"
Zones are difined according to distance from faults (given as polygons)
*************************************************************************/
ListPolygons faults;
double Distances[Nzones-1];
double Marks[Nzones];

if (Nzones>0)
  { int j;
in.open ("ZONES");
if (!in)
 cout<<"cannot open file"<<endl;

for (i=0; i<Nzones-1; ++i)
  in>>Distances[i];              // Distances that define the zones
for (i=0; i<Nzones; ++i)
  in>>Marks[i];                 // Mark probability: ratio of intensity
                                // in the zone to the greatest  intensity of the fracture sets
in>>Nfaults;
double xx, yy, zz;

for (i=0; i<Nfaults; ++i)
  {
Polygon pol;
int N;
in>>N;                          // Number of vertices defining a fault

for (j=0; j<N; ++j)             // Coordinates of fault vertices
  {in>>xx>>yy>>zz;
   Point* P = new Point (xx, yy, zz);
   pol.add_point(*P);
    };
Polygon* new_pol = new Polygon;
*new_pol = pol;
faults.add_polygon(*new_pol);      // Adding the fault to a list of faults
 };
in.close();
  };
```

```
/*************************************************************************
*                CREATING OUTCROP PLANES
*************************************************************************/
// VERTICAL OUTCROP PLANE
Plane profile (oAv, oBv, oCv, oDv);
Polygon outPolV = make_initial(profile, rock);
outPolV.make_polygon_2d();
outPolV.sort_points_2d();
outPolV.make_polygon_3d();
Zm =  2*(rock.Zmax / sin(profile.dip));
outPolV.add_points_on_surface (profile, rock.top, Np, Zm);

// HORIZONTAL OUTCROP PLANE
Plane plan (oAh, oBh, oCh, oDh);
Polygon outPolH = make_initial(plan, rock);
outPolH.make_polygon_2d();
outPolH.sort_points_2d();
outPolH.make_polygon_3d();

// 3D VIEW OF THE OUTCROP PLANES
out.open("outcrops.m");
out<<"Show [ ";
outPolV.print();
out<<", ";
outPolH.print();
out<<"]  ";
out.close();

/*************************************************************************
*                GENERATION OF FRACTURE SETS
*************************************************************************/
ListPolygons FracSet[i];
ListPolygons FracSys;
Stat meanSD;
double P32;
out.open ("Frac.txt");
out<<endl<<"mark > "<<ratioMA<<" and < "<<maxR<<" gama "<<gama<<" CA "<<CA<<endl;
out<<"angleR ratioR angleC ratioC strike angleNew "<<endl;
out<<angleR<<" "<<ratioR<<" "<<angleC<<" "<<ratioC<<" "<<strike<<" "<<angleNew<<endl;
out<<"ratioMA CA gama "<<ratioMA<<" "<<CA<<" "<<gama<<endl;
for (i=0; i<Nsets; ++i)
  { int counter = 1;
double Mu = FracInt[i]/gama;        // Intensity of Poisson plane network
double expD = exp_value(Mu);
double planeD = -Rmax+expD;
MeanA = PI*MeanR[i]*MeanR[i];
cout<<MeanR[i]<<" "<<MeanA<<endl;
MeanArea = MeanA/CA;
double intensity = sqrt(PI/(MeanArea));
cout<<"Fracture set "<<i+1<<endl;
```

```
out<<endl<<"EXPECTED FRACTURE INTENSITY "<<FracInt[i]<<endl;
out<<"PLANE INTENSITY "<<Mu<<endl;
out<<"EXPECTED RADIUS AREA "<<MeanR[i]<<" "<<MeanA<<endl;
out <<"LINE INTENSITY "<<intensity<<endl;
out.close();

// GENERATION OF THE FRACTURE PLANES OF A FRACTURE SET
while (planeD < Rmax)
 { ListPolygons* FracPlane = new ListPolygons;
cout<<" set  "<<i+1<<" plane "<<counter<<endl;
Plane p (pdf[i], MeanP[i], planeD);
Polygon temp =  make_initial(p, rock);
if (temp.noP >2)
  { Polygon* pol = new Polygon;     *pol = temp;
pol->make_polygon_2d();
pol->sort_points_2d();
pol->make_polygon_3d();
Zm =  2*(rock.Zmax / sin(p.dip));
pol->add_points_on_surface (p, rock.top, Np, Zm);
pol->area_radius_3d ();

// PLOT PRIMARY PLANE PROCESS
//out.open ("Primary.m", ios::app);
//out<<"Show [ "; pol->print();
//out<<"]  "<<endl<<endl;;
//out.close();

pol->make_polygon_2d();
double InitialA = pol->area;

// TESSELLATION OF A FRACTURE PLANE INTO POLYGONS
*FracPlane = Poisson_lines_on_polygon (*pol, intensity);

// PLOT SECONDARY PROCESS: TESSELLATION
//FracPlane->make_listpol_3d();
//out.open ("Tessellation.m", ios::app);
//out<<"Show [ ";  FracPlane->print();
//out<<"]  "<<endl;
//out.close();
//FracPlane->make_listpol_2d();

// MARKING OF POLYGONS WITH GOOD SHAPE: ~40% of the TOTAL AREA
FracPlane->mark_good_shape(AngleMin, elongation);
cout<<"after mark "<<FracPlane->Npol<<endl;
//FracPlane->make_listpol_3d();
//out.open ("Marking.m", ios::app);
//out<<"Show [ "; FracPlane->print();
//out<<"]  "<<endl;
//out.close();
//FracPlane->make_listpol_2d();
```

```
// ZONE MARKING OF POLYGONS FOR DIFFERENT ZONE FRACTURE INTENSITY
if (FracPlane->Npol && Nzones && Nfaults)
  { FracPlane->mark_by_zones(faults, Nzones, Distances, Marks);
    cout<<"after marking by zones"<<FracPlane->Npol<<" fractures"<<endl; };

// TRANSLATION
FracPlane->translate_2d (MeanR[i], TRatio[i]);
FracPlane->make_listpol_3d();

if (FracPlane->Npol)
   FracSet[i].add_listpol(*FracPlane);
else
   delete FracPlane;
cout<<"total "<<(FracSet[i].Npol+FracSys.Npol)<<endl; };
expD = exp_value (Mu);
planeD += expD;
++ counter;
  };            // End of line tessellation for one plane


FracSys.add_listpol (FracSet[i]);
};                  // End of generation of one fracture set

out.open ("Frac.txt", ios::app);
double Radius = sqrt (MeanA/PI);
meanSD = FracSys.find_mean_sd(Radius, maxR);
P32 = meanSD.total / VOLUME;
out<<"Mean SD Total N P32 "<<meanSD<<" "<<FracSys.Npol<<" "<<P32<<endl;
out<<"Mean/expected mean SD/Mean "<<meanSD.Mean/MeanA<<" ";
out<<meanSD.SD / meanSD.Mean<<endl;
cout<<"Finished calculating statistics "<<endl;
out.close ();

ListLines tracesH;
ListLines tracesV;

// GRAPHICAL OUTPUT BEFORE ROTATION
//out.open ("Fractures.m", ios::app);
//out<<"Show [ ";
//FracSys.print();
//out<<"]  "<<endl;
//out.close();

// VERTICAL OUTCROP in a file "Profile.m"
//tracesV = FracSys.traces_on_plane(profile);
//out.open ("Profile.m");
//out<<"Show [ ";  outPolV.print();  out<<",  ";
//tracesV.print();
//out<<"]  "<<endl<<endl;;
//out.close();
```

```
//HORIZONTAL OUTCROP in a file "Plan.m"
//tracesH = FracSys.traces_on_plane(plan);
//out.open ("Plan.m");
//out<<"Show [ ";
//outPolH.print();
//out<<",  ";
//tracesH.print();
//out<<"]  "<<endl<<endl;;
//out.close();


/*************************************************************************
*    ROTATION OF FRACTURES TO BETTER FIT SURFACE STRIKE AND DIP
*************************************************************************/
// RADIAL AND CONCENTRIC FRACTURES WITH ONE OF THE ORIENTATIONS
// PREFERRED
if (folding == 'y' || folding == 'Y')
  { if ( strike == 'r' || strike == 'c')
FracSys.mark_by_strike (fold, angleR, angleC, angleNew, strike, ratioR, ratioC);
if (dip == 'd')
FracSys.mark_by_dip (fold, angleD, angleDip, ratioD); };

/*************************************************************************
*                  SIZE  DISTRIBUTION
*************************************************************************/
out.open ("Size.txt");
out<<"Mean SD Total N P32 "<<meanSD<<" "<<FracSys.Npol<<" "<<P32<<endl;
out<<"Mean/expected mean SD/Mean "<<meanSD.Mean/MeanA<<" "<<meanSD.SD /
meanSD.Mean<<endl<<endl;
int j;
FracSys.size_distribution (sizeNpdf, Nsize_int, sizeMax, meanSD.Mean);
for (j=0; j<sizeNpdf; ++j)
  out<<sizeMax[j]<<"  "<<Nsize_int[j]<<endl;
out<<Nsize_int[sizeNpdf]<<endl<<endl;;
out.close();

/*************************************************************************
 *      INTERSECTION WITH BOREHOLE(S) IF DESIRED
*************************************************************************/
if (Nboreholes)
  {  in.open ("BOREHOLE");
for (i=0; i<Nboreholes; ++i)
  { in>>bX1>>bX2>>bY1>>bY2>>bZ1>>bZ2;    // borehole

// INTERSECTIONS WITH A BOREHOLE
Point P1 (bX1, bY1, bZ1);
Point P2 (bX2, bY2, bZ2);
Line boreL (P1, P2);
Borehole BoreLine = FracSys.intersections_with_logline (boreL);
Stat boreMSD = BoreLine.find_mean_sd_spacing();
```

```
if (BoreLine.Nfrac)
  { if (!i)
  {out.open ("Bore.txt");}
else
   out.open ("Bore.txt", ios::app);
out<<endl<<endl<<"Next borehole "<<endl;
out<<BoreLine;
out<<"SPACING mean sd boreline length "<<boreMSD<<endl;
out.close();
cout<<"Borehole calculated "<<endl;
out.open ("Frac.txt", ios::app);
out<<"SPACING mean sd boreline length "<<boreMSD<<endl;
out.close();

if (!i)
  {
out.open ("Bore.m");}
else
   out.open ("Bore.m", ios::app);
out<<"Show [ ";
BoreLine.print();
out<<"]  ";
out.close(); };  };
in.close();
};

/*********************************************************************
*     OUTPUT FOR FRACTURE SYSTEM AND TRACES ON OUTCROP PLANES
*********************************************************************/
//ALL FRACTURES in a file "Fractures.m"
if (gF == 'Y' || gF == 'y' || cF == 'y' || cF == 'Y')
  {  if (gF == 'Y' || gF == 'y')
  {  out.open ("FRACTURES.m");
out<<"Show [ ";
FracSys.print();
out<<"]  ";
out.close(); };
if (cF == 'Y' || cF == 'y')
  { out.open ("FRACTURES.txt");
            out<<FracSys<<endl;
out.close(); }; };

// VERTICAL OUTCROP in a file "Profile.m"
if (gV == 'Y' || gV == 'y' || cV == 'y' || cV == 'Y')
  {
tracesV = FracSys.traces_on_plane(profile);
Stat outVmsd = tracesV.find_mean_sd_length();
out.open ("Frac.txt", ios::app);
out<<"profile length mean sd total P21 "<<outVmsd<<" "<<outVmsd.total/outPolV.area<<endl;
out.close();
```

```
if (gV == 'Y' || gV == 'y')
  {
out.open ("PROFILE.m", ios::app);
out<<"Show [ ";
outPolV.print();
out<<",  ";
tracesV.print();
out<<"]  ";
out.close(); };

if (cV == 'Y' || cV == 'y')
  { out.open ("PROFILE.txt");
            out<<tracesV<<endl;
out.close(); }; };

//HORIZONTAL OUTCROP in a file "Plan.m"
if (gH == 'Y' || gH == 'y' || cH == 'y' || cH == 'Y')
  { tracesH = FracSys.traces_on_plane(plan);
cout<<"After changes there are "<<tracesH.Nline<<" plan traces"<<endl;
Stat outHmsd = tracesH.find_mean_sd_length();
out.open ("Frac.txt", ios::app);
out<<"plan length mean sd total P21 "<<outHmsd<<" "<<outHmsd.total/outPolH.area<<endl;
out.close();

if (gH == 'Y' || gH == 'y')
  { out.open ("PLAN.m");
out<<"Show [ ";
outPolH.print();
out<<",  ";
tracesH.print();
out<<"]  ";
out.close(); };

if (cH == 'Y' || cH == 'y')
  { out.open ("PLAN.txt");
            out<<tracesH<<endl;
out.close(); };  };
};
```

File **INPUT**

```
100.
100.
0.

-0.001
0.
0.
0.
0.
150.

0. 100.
2.2 .4
1
1
Y
y y y
y y y
0. 1. 0. 10.
0. 0. 1. 50.
1

8
0.3
2.
1.04720
1.6
c
0.2618 1.
0.2618 0.
0.5236
n
.17453 1.
.17453

60
 .1  .2  .3  .4  .5  .6  .7  .8  .9  1. 1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.
2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9  3.0  3.2  3.4  3.6  3.8  4.  4.2  4.4  4.6  4.8  5.
5.2  5.4  5.6  5.8  6.0  6.5  7.  7.5  8.  8.5  9.  9.5  10.  10.5  11.  12.  13.  14.  15.  16.

.005
1.5708 1.5708
f
1.
10.
```

File **ZONES**

```
50.

1. 0.05

1

4
-5. 50. 0.
-5. 50. 100.
-5. -50. 100.
-5. -50. 0.
```

File **FOLD**

```
0.
0.
0.
0.
-0.001
0.
0.
0.
0.
150.
```

File **BOREHOLES**

```
0.    0.    0.    0.    50.    150.
```

File **fractal.mak**

```
LFLAGS =  -lm
CC = g++

geoFractal:  cartesian.o point.o polar.o line.o surface.o cubic.o random.o plane.o volume.o \
polygon.o initial.o intersections.o listpol.o divide.o listline.o rotation.o zones.o borehole.o \
cell.o fractal.o
        $(CC) -o geoFractal cartesian.o point.o polar.o line.o surface.o cubic.o random.o \
plane.o volume.o polygon.o initial.o intersections.o listpol.o divide.o listline.o rotation.o \
zones.o borehole.o cell.o fractal.o ${LFLAGS}

cartesian.o: cartesian.C cartesian.h
        $(CC)  -c  $*.C
point.o: point.C point.h
        $(CC)  -c  $*.C
polar.o: polar.C polar.h
        $(CC)  -c  $*.C
line.o: line.C line.h
        $(CC)  -c  $*.C
surface.o: surface.C surface.h
        $(CC)  -c  $*.C
cubic.o: cubic.C cubic.h
        $(CC)  -c  $*.C
random.o: random.C
        $(CC)  -c  $*.C
plane.o: plane.C plane.h
        $(CC)  -c  $*.C
volume.o: volume.C volume.h
        $(CC)  -c  $*.C
polygon.o: polygon.C polygon.h
        $(CC)  -c  $*.C
initial.o: initial.C
        $(CC)  -c  $*.C
intersections.o: intersections.C
        $(CC)  -c  $*.C
listpol.o: listpol.C listpol.h stat.h
        $(CC)  -c  $*.C
divide.o: divide.C
        $(CC)  -c  $*.C
listline.o: listline.C listline.h
        $(CC)  -c  $*.C
rotation.o: rotation.C polygon.h
        $(CC)  -c  $*.C
zones.o: zones.C polygon.h listpol.h
        $(CC)  -c  $*.C
borehole.o: borehole.h borehole.C
        $(CC)  -c  $*.C
cell.o: cell.C
        $(CC)  -c  $*.C
fractal.o: fractal.C
        $(CC)  -c  $*.C
```

File **fractal.C**
(page 1 / 5)

```
#include <math.h>
#include <stdio.h>
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <sys/types.h>
#include <time.h>
#include <unistd.h>

#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)
#define max(a,b) ((a) > (b) ? (a) : (b))
#define min(a,b) ((a) < (b) ? (a) : (b))
double Xm, Ym;                  // Lateral boundaries of the volume
double TRatio;
double AngleMin, elongation;
double MeanArea, MeanA;
ofstream out;
double ratioMA, CA, gama;
double Datum;
double MaxPhi;
double Fk, bFk2, bFk1;
int sizeNpdf;
double maxR;
int Type;
int FLTiterations;
double FLT;
double P;
char mark;
double AT;

#define x1(a, b, f)  (-b+sqrt(f))/(2*a)
#define x2(a, b, f)  (-b-sqrt(f))/(2*a)

#include "polar.h"
#include "cartesian.h"
#include "point.h"
#include "line.h"
#include "surface.h"
#include "volume.h"
#include "plane.h"
#include "polygon.h"
#include "listpol.h"
#include "listline.h"
#include "borehole.h"
#include "stat.h"

time_t time(time_t *tloc);
double Random01();
```

```
double Random0a(double);
double  RandomBC(double , double);
double exp_value (double);
int PoissonN (double, double);
Polar ran_uniform_orientation();
Polar uniform_max_phi_orient(double);
Polar constant_orientation();
Polar Fisher_orientation (double);
Polygon make_initial (Plane&, Volume&);
ListPolygons Poisson_lines_on_polygon (Polygon& , double);

main ()
{
srand(time(0) * getpid());
int i;

/*************************************************************************
*          INPUT OF PARAMETERS FROM FILE "FractalInput.dat"
*************************************************************************/
ifstream in ("FractalInput.dat");
if (!in)
 cout<<"cannot open file"<<endl;
Datum = 0.;
in>>Xm>>Ym;              // Extent of rectangular area (tract)

// INPUT OF TYPE OF MODEL MARKING
in>>mark;
in>>ratioMA>>maxR;     // Coefficients of the model plane, line,  and marking processes
in>>CA;
in>>Type;                  // Type of fractal line tessellation
in>>FLTiterations>>FLT>>P;       // parameters of Fractal Line Tessellation
in>>AngleMin;            // Minimum allowed angle for polygon shapes
in>>elongation;          // Maximum allowed elongation of polygons
in>>sizeNpdf;
int Nsize_int [sizeNpdf+1];
double sizeMax [sizeNpdf];

for (i=0; i<sizeNpdf; ++i)
  {Nsize_int[i] = 0;
  in>>sizeMax[i];};
Nsize_int[sizeNpdf] = 0;

double FracInt;
double MeanR;
double TRatio;

// INPUT OF PARAMETERS FOR INDIVIDUAL FRACTURE SETS
in>>TRatio;              // Translation (non-coplanarity)
in>>MeanR;
in.close();
```

File **fractal.C**
(page 3 / 5)

```
/*************************************************************************
*                GENERATION OF FRACTURE PLANE
*************************************************************************/
Stat meanSD;
out.open ("FractalFrac.txt");
out<<endl<<"mark > "<<ratioMA<<" and < "<<maxR<<endl;
out<<"ratioMA CA "<<ratioMA<<" "<<CA<<endl;
MeanA = PI*MeanR*MeanR;
cout<<MeanR<<" "<<MeanA<<endl;
MeanArea = MeanA/CA;
double intensity = sqrt(PI/(MeanArea));
out<<"EXPECTED RADIUS AREA "<<MeanR<<" "<<MeanA<<endl;
out <<"LINE INTENSITY "<<intensity<<endl;

// GENERATION OF THE FRACTURE PLANE
ListPolygons* FracPlane = new ListPolygons;
Polygon initial;
Point p1 (Xm, Ym, 0.);
initial.add_point (p1);
Point p2 (-Xm, Ym, 0.);
initial.add_point (p2);
Point p3 (-Xm, -Ym, 0.);
initial.add_point (p3);
Point p4 (Xm, -Ym, 0.);
initial.add_point (p4);
initial.find_area_radius_2d();
Line l1 (p1, p2);
Line l2 (p2, p3);
Line l3 (p3, p4);
Line l4 (p4, p1);
ListLines LL;
LL.add_line(l1);
LL.add_line(l2);
LL.add_line(l3);
LL.add_line(l4);
double InitialA = initial.area;

// TESSELLATION OF A FRACTURE PLANE INTO POLYGONS
*FracPlane = Poisson_lines_on_polygon (initial, intensity);
cout<<"PLT N polygons "<<FracPlane->Npol<<endl;

//ALL FRACTURES in a file "Fractal.m"
out.open ("Fractal.m");
out<<"Show [ ";
LL.print();
out<<" ,";
FracPlane->print();
out<<"] "<<endl<<endl;
out.close();
```

File **fractal.C**
(page 4 / 5)

```
meanSD = FracPlane->find_mean_sd();
out.open ("FractalFrac.txt", ios::app);
out<<"All polygons by Poisson LT "<<endl;
out<<"Mean SD Total N "<<meanSD<<" "<<FracPlane->Npol<<" "<<endl;
out<<"Mean/expected mean SD/Mean "<<meanSD.Mean/MeanA<<" "<<meanSD.SD /
meanSD.Mean<<endl;
out.close ();

//SIZE OF ALL POLYGONS IN FILE "FractalSize.txt"
out.open ("FractalSize.txt");
out<<"ALL POLYGONS by Poisson line tessellation "<<endl;
out<<"Mean SD Total N "<<meanSD<<" "<<FracPlane->Npol<<" "<<endl;
out<<"Mean/expected mean SD/Mean "<<meanSD.Mean/MeanA<<" "<<meanSD.SD /
meanSD.Mean<<endl<<endl;
int j;
FracPlane->size_distribution (sizeNpdf, Nsize_int, sizeMax, MeanA);
for (j=0; j<sizeNpdf; ++j)
 out<<sizeMax[j]<<"  "<<Nsize_int[j]<<endl;
out<<Nsize_int[sizeNpdf]<<endl<<endl;
out.close();

// MARKING OF POLYGONS WITH GOOD SHAPE:
if (mark == 'y' || mark == 'Y')
 { FracPlane->mark_good_shape_and_P(AngleMin, elongation, P);
   cout<<"after mark "<<FracPlane->Npol<<endl;

//ALL FRACTURES in a file "Fractal.m"
out.open ("Fractal.m", ios::app);
out<<"Show [ ";
LL.print();  out<<" ,";
FracPlane->print();   out<<"]  "<<endl<<endl;
out.close();
meanSD = FracPlane->find_mean_sd(MeanR, maxR);
out.open ("FractalFrac.txt", ios::app);
out<<"Marked polygons by Poisson LT "<<endl;
out<<"Mean SD Total N "<<meanSD<<" "<<FracPlane->Npol<<" "<<endl;
out<<"Mean/expected mean SD/Mean "<<meanSD.Mean/MeanA<<" "<<meanSD.SD /
meanSD.Mean<<endl;
out.close ();

//SIZE OF MARKED POLYGONS IN FILE "FractalSize.txt"
out.open ("FractalSize.txt", ios::app);
out<<"Marked polygons by Poisson LT "<<endl;
out<<"Mean SD Total N "<<meanSD<<" "<<FracPlane->Npol<<" "<<endl;
out<<"Mean/expected mean SD/Mean "<<meanSD.Mean/MeanA<<" "<<meanSD.SD /
meanSD.Mean<<endl<<endl;
int j;
FracPlane->size_distribution (sizeNpdf, Nsize_int, sizeMax, MeanA);
for (j=0; j<sizeNpdf; ++j)
 out<<sizeMax[j]<<"  "<<Nsize_int[j]<<endl;
```

File **fractal.C**
(page (5 / 5)

```
out<<Nsize_int[sizeNpdf]<<endl<<endl;;
out.close(); };

// FRACTAL LINE TESSELLATION if specified in INPUT

if (FLTiterations)
  { int m;
for (m=0; m<FLTiterations; ++m)
  { if (Type == 2)
FracPlane->fractal_big_and_small (intensity, FLT, AT);
else if (Type == 3)
FracPlane->fractal_similar(intensity, FLT, AT);
else
FracPlane->fractal_tessellation (intensity, FLT);
cout<<"before mark "<<FracPlane->Npol<<endl;

// MARK FRACTAL POLYGONS
if ( (mark == 'y' || mark == 'Y') && m == FLTiterations-1)
  { FracPlane->mark_good_shape_and_P(AngleMin, elongation, P);
    cout<<"after mark "<<FracPlane->Npol<<endl; };

//ALL FRACTURES in a file "Fractal.m"
if (m >= FLTiterations-2)
  { out.open ("Fractal.m", ios::app);
out<<"Show [ ";
LL.print();
out<<" ,";
FracPlane->print();
out<<"]  "<<endl<<endl;
out.close(); };
meanSD = FracPlane->find_mean_sd(MeanR, maxR);
out.open ("FractalFrac.txt", ios::app);
out<<"All polygons by Fractal LT iteration "<<m<<endl;
out<<"Mean SD Total N "<<meanSD<<" "<<FracPlane->Npol<<" "<<endl;
out<<"Mean/expected mean SD/Mean gama "<<meanSD.Mean/MeanA<<" "<<meanSD.SD /
meanSD.Mean<<" "<<meanSD.total/InitialA<<endl;
out.close ();

//SIZE OF POLYGONS IN FILE "FractalSize.txt"
out.open ("FractalSize.txt", ios::app);
out<<"FLT POLYGONS after marking iteration "<<m<<endl;
out<<"Mean SD Total N "<<meanSD<<" "<<FracPlane->Npol<<" "<<endl;
out<<"Mean/expected mean SD/Mean gama"<<meanSD.Mean/MeanA<<" "<<meanSD.SD /
meanSD.Mean<<" "<<meanSD.total/InitialA<<endl<<endl;
int j;
FracPlane->size_distribution (sizeNpdf, Nsize_int, sizeMax, MeanA);
for (j=0; j<sizeNpdf; ++j)
  out<<sizeMax[j]<<"  "<<Nsize_int[j]<<endl;
out<<Nsize_int[sizeNpdf]<<endl<<endl;;
out.close(); }; };  };
```

File **FractalInput**

```
250.
250.

y
0 100.
1.

2
2.
3  1.  1.


1.0472
1.6

60
 .1  .2  .3  .4  .5  .6  .7  .8  .9  1.
1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.
2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9  3.0
3.2  3.4  3.6  3.8  4.
4.5 5. 5.5 6. 6.5 7. 7.5 8. 8.5 9. 9.5 10.
11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 25. 30. 40.

1.
```

File **cell.h**

```
#ifndef _CELL_H
#define _CELL_H
#include <iostream.h>
#include <fstream.h>
#include <math.h>

class Cell
{
 public:

Cell* next_cell;
double Z;
double value;
double GR;

Cell (double z, double V) {
Z = z;
value = V;
next_cell = 0; };

Cell (double z, double V, double gamaray) {
Z = z;
value = V;
GR = gamaray;
next_cell = 0; };
};


class Column
{
 public:
double X, Y;
Cell* head_cell;
int Ncells;
Column ()  // default constructor
  {
head_cell = 0;
Ncells = 0;
};
void add_cell (Cell);
friend ostream& operator<< (ostream& , Column&);
};

#endif _CELL_H
```

File **cell.C**
(page 1 / 4)

```
#include "cell.h"
#include "polygon.h"
#include "listpol.h"

#include <math.h>
#include <iostream.h>

#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)
extern ofstream out;
double Random01 ();

ostream& operator<< (ostream& o, Column& c)
{ int i;
o<<" X "<<c.X<<" Y "<<c.Y<<endl;
Cell* current = c.head_cell;
for (i=0; i<c.Ncells; ++i)
  {o<<i+1<<" Z "<<current->Z<<" V "<<current->value<<endl;
  if (current->GR)
    o<<" GR "<<current->GR<<endl;
  current = current->next_cell;};  return o; };

// Function to add a new cell (e.g. porosity at elevation Z) to a column of data

void Column :: add_cell (Cell c)
{ Cell* newCell = new Cell (c.Z, c.value, c.GR);
newCell->next_cell = head_cell;
head_cell = newCell; ++Ncells; return; };

// Function to find the average porosity of a column where it is intersected by a polygon.
// Approximate calculation: X and Y of the column have been checked to be the closest to
// the center of the polygon X and Y. Then the minimum and maximum Z are calculated as
// Zmin=Zc-Re, and Zmax=Zc+Re, where Re is the equivalent radius of the polygon. All values
// of porosity for cells which are at elevation between Zmin and Zmax are sumemd, and
// the sum (divided by the number of intersected cells) is returned.

double Polygon :: find_average_porosity (Column& C)
{ int N=0, i;
double Zmax = center.Z + radius + 700.;
double Zmin = center.Z - radius + 700.;
double porosity = 0.;
Cell* current = C.head_cell;
for (i=0; i<C.Ncells; ++i)
  {if (current->Z > Zmin && current->Z < Zmax)
{porosity += current->value; N++;};
current = current->next_cell;  };
if (N)
 porosity /= N;
return porosity; };
```

```
// Function to find the average GR of a column where it is intersected by a polygon.
// Approximate calculation: X and Y of the column have been checked to be the closest to
// the center of the polygon X and Y. Then the minimum and maximum Z are calculated as
// Zmin=Zc-Re, and Zmax=Zc+Re, where Re is the equivalent radius of the polygon. All values
// of porosity for cells which are at elevation between Zmin and Zmax are sumemd, and
// the sum (divided by the number of intersected cells) is returned.

double Polygon :: find_average_GR (Column& C)
{ int N=0, i;
double Zmax = center.Z + radius + 700.;
double Zmin = center.Z - radius + 700.;
double gamaray = 0.;

Cell* current = C.head_cell;
for (i=0; i<C.Ncells; ++i)
  { if (current->Z > Zmin && current->Z < Zmax)
{gamaray += current->GR;
N++;};
current = current->next_cell; };

if (N)
 gamaray /= N;
return gamaray; };

// Function to mark a polygon according to the averaged porosity of the surrounding rock.
// This function is specifically written to interpret the Yates field Stratamodel porosity
// data. Above a certain elevation (Zmax) there are no porosity readings available, but it
// it known that the rock is very porous (unconformity) therefore the polygons are marked
// with the same probability as in the strata where teh averaged porosity is higher than   maxPor.

boolian Polygon :: mark_by_porosity (Column* sgm, int Ncol, double maxPor, double ratioP)
{ int N = find_closest_column (sgm, Ncol);
//cout<<"center "<<center;
//cout<<"closest column "<<N<<" "<<sgm[N].X<<" "<<sgm[N].Y<<endl;

double ave_por = find_average_porosity (sgm[N]);

if ((ave_por>maxPor || !ave_por) && Random01() > ratioP)
  {return FALSE;}
else
  { return TRUE;}; };


// Function to mark a polygon according to the averaged porosity and GR of the surrounding rock.
// This function is specifically written to interpret the Yates field Stratamodel porosity & GR
// data. Above a certain elevation (Zmax) there are no porosity readings available, but it
// it known that the rock is very porous (unconformity) therefore the polygons are marked
// with the same probability as in the strata where the rock is shale.
```

```
boolian Polygon :: mark_by_porosity_and_GR (Column* sgm, int Ncol, double minPor, double
maxGR, double ratioP)
{ int N = find_closest_column (sgm, Ncol);
double ave_por = find_average_porosity (sgm[N]);
double ave_GR = find_average_GR (sgm[N]);
cout<<" P GR "<<ave_por<<" "<<ave_GR;
if ((!ave_por || ave_por<minPor && ave_GR>maxGR ) && Random01() > ratioP)
  {cout<<" No "<<endl; return FALSE;}
else
  {cout<<" Yes "<<endl;  return TRUE;}; };

// Procedure to find and return the number of the column in the Stratamodel in which a polygon
// is located. The function finds the column for which the horizontal distance between the
// center of the polygon and the axis (center) of the column is shortest.

int Polygon :: find_closest_column (Column* cp, int Ncol)
{ int i, colN=0;
double minD, temp;
double xx = cp->X - center.X;
double yy = cp->Y - center.Y;
minD = sqrt (xx*xx+yy*yy);
for (i=1; i<Ncol; ++i)
  { xx = (cp+i)->X - center.X;
yy = (cp+i)->Y - center.Y;
temp = sqrt (xx*xx+yy*yy);
if (temp < minD)
  {minD = temp;
  colN = i;}; };
return colN; };

// Function to mark polygons-fractures by porosity, discarding those which are in very
// porous rock. This function is specific for the Yates field where highly porous rock is
// too ductile to farture.

void ListPolygons :: discard_high_porosity (Column* sgm, int Ncol, double maxPor, \
double ratioP)
{ Node* current = head_pol;
while (current->next_pol)
  { if (!current->next_pol->content->mark_by_porosity (sgm, Ncol, maxPor, ratioP))
  { current->next_pol = current->next_pol->next_pol;
  --Npol;}
else
  current = current->next_pol; };
if (!head_pol->content->mark_by_porosity (sgm, Ncol, maxPor, ratioP))
  {  head_pol = head_pol->next_pol;
  --Npol;};
return; };
```

```
// Function to mark polygons-fractures by porosity and GR, discarding those which are in
// shale (low porosity, high GR) . This function is specific for TRACT17.

void ListPolygons :: discard_shale (Column* sgm, int Ncol, double minPor, double maxGR,
double ratioP)
{ Node* current = head_pol;
while (current->next_pol)
  { if (!current->next_pol->content->mark_by_porosity_and_GR \
(sgm, Ncol, minPor, maxGR, ratioP))
  {   current->next_pol = current->next_pol->next_pol;
  --Npol;}
else
  current = current->next_pol;   };
if (!head_pol->content->mark_by_porosity_and_GR (sgm, Ncol, minPor, maxGR, ratioP))
  {
 head_pol = head_pol->next_pol;
  --Npol;};
return;
};
```

File **YU1711.mak**

```
LFLAGS =  -lm
CC = g++

YU1711:  cartesian.o point.o polar.o line.o surface.o cubic.o random.o plane.o \
volume.o polygon.o initial.o intersections.o listpol.o divide.o listline.o rotation.o \
zones.o borehole.o cell.o 1711main.o
        $(CC) -o YU1711 cartesian.o point.o polar.o line.o surface.o cubic.o random.o \
plane.o volume.o polygon.o initial.o intersections.o listpol.o divide.o listline.o rotation.o \
zones.o borehole.o cell.o 1711main.o ${LFLAGS}

cartesian.o: cartesian.C cartesian.h
        $(CC)  -c  $*.C
point.o: point.C point.h
        $(CC)  -c  $*.C
polar.o: polar.C polar.h
        $(CC)  -c  $*.C
line.o: line.C line.h
        $(CC)  -c  $*.C
surface.o: surface.C surface.h
        $(CC)  -c  $*.C
cubic.o: cubic.C cubic.h
        $(CC)  -c  $*.C
random.o: random.C
        $(CC)  -c  $*.C
plane.o: plane.C plane.h
        $(CC)  -c  $*.C
volume.o: volume.C volume.h
        $(CC)  -c  $*.C
polygon.o: polygon.C polygon.h
        $(CC)  -c  $*.C
initial.o: initial.C
        $(CC)  -c  $*.C
intersections.o: intersections.C
        $(CC)  -c  $*.C
listpol.o: listpol.C listpol.h
        $(CC)  -c  $*.C
divide.o: divide.C
        $(CC)  -c  $*.C
listline.o: listline.C listline.h
        $(CC)  -c  $*.C
rotation.o: rotation.C polygon.h
        $(CC)  -c  $*.C
zones.o: zones.C polygon.h listpol.h
        $(CC)  -c  $*.C
borehole.o: borehole.h borehole.C
        $(CC)  -c  $*.C
cell.o: cell.C cell.h
        $(CC)  -c  $*.C
1711main.o: 1711main.C
        $(CC)  -c  $*.C
```

File **1711main.C**
(page 1 / 9)

```
#include <math.h>
#include <stdio.h>
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <sys/types.h>
#include <time.h>
#include <unistd.h>

#define PI M_PI
#define HalfPI (M_PI/2)
#define TwoPI (M_PI*2)
#define max(a,b) ((a) > (b) ? (a) : (b))
#define min(a,b) ((a) < (b) ? (a) : (b))
double Xm, Ym;                  // Lateral boundaries of the volume
int Nsets;                      // No. of fracture sets
double A, B, C, D, E, F;        // Top surface of modeling volume
double mA, mB, mC, mD, mE, mF, mG, mH, mI, mJ;  // Cubic surface of the fold
double oAv, oBv, oCv, oDv;
double oAh, oBh, oCh, oDh;
double bX1, bX2, bY1, bY2, bZ1, bZ2;
double bFk1, bFk2, Fk, MaxPhi;
double TRatio;
double AngleMin, elongation;
double angleR, angleC, angleNew;
double ratioR, ratioC;
double MeanArea, MeanA;
ofstream out;
char strike;
double ratioMA, CA, gama, maxR;
double Datum;

#define x1(a, b, f)  (-b+sqrt(f))/(2*a)
#define x2(a, b, f)  (-b-sqrt(f))/(2*a)

#include "polar.h"
#include "cartesian.h"
#include "point.h"
#include "line.h"
#include "surface.h"
#include "volume.h"
#include "plane.h"
#include "polygon.h"
#include "listpol.h"
#include "listline.h"
#include "borehole.h"
#include "stat.h"

time_t time(time_t *tloc);
double Random01();
```

```
double Random0a(double);
double  RandomBC(double , double);
double exp_value (double);
int PoissonN (double, double);
Polar ran_uniform_orientation();
Polar uniform_max_phi_orient(double);
Polar constant_orientation();
Polar Fisher_orientation (double);
Polygon make_initial (Plane&, Volume&);
ListPolygons Poisson_lines_on_polygon (Polygon& , double);
int Np;
int sizeNpdf;

char porosity;
int Pcol;
double Pmin;
double PPmax;
double GRmax;

main ()
{
srand(time(0) * getpid());
int i;

/*************************************************************************
*       INPUT OF YU1711 MODELING VOLUME FROM FILE "YU1711input.dat"
*************************************************************************/
ifstream in ("1711input.dat");
if (!in)
 cout<<"cannot open file"<<endl;
in>>Xm>>Ym;              // Extent of rectangular area (tract): origin
                         // (0,0) of the global f.o.r. is at (113,827 vara,
                         // 109041 vara); Xm and Ym are in FEET
in>>Datum;               // Datum Z=0 is bottom surface at elevation 700 ft above sea level
in>>A>>B>>C>>D>>E>>F;    // Top surface: 100 FEET above San Andres
in>>mA>>mB>>mC>>mD>>mE>>mF>>mG>>mH>>mI>>mJ;
                         // Input of the cubic surface of the fold
                         // (derived from Seven Rivers M horizon)

// INPUT OF TYPE OF MODEL MARKING
in>>ratioMA>>maxR;
in>>CA>>gama;    // Coefficients of the model plane, line,  and marking processes
cout<<ratioMA<<" "<<CA<<" "<<gama<<" "<<maxR<<endl;
in>>Nsets;           // Number of fracture sets
in>>oAv>>oBv>>oCv>>oDv;   // profile outcrop plane
in>>oAh>>oBh>>oCh>>oDh;   // plan view outcrop plane
in>>bX1>>bX2>>bY1>>bY2>>bZ1>>bZ2;
in>>Np;              // Np is the number of points on the top surfac
in>>MaxPhi;          // Maximum azimuth, if uniform orientation
in>>Fk;              // Fisher constant, if Fisher orientation
```

```
in>>AngleMin;              // Minimum allowed angle for polygon shapes
in>>elongation;            // Maximum allowed elongation of polygons
in>>strike;                // Relationship to strike: C(oncentric), R(adial)
in>>angleR>>ratioR;        // maximum angle and ratio between strike of
                           // polygon and slope of surface
in>>angleC>>ratioC;        // maximum angle and ratio between strike of
                           // polygon and strike of surface
in>>angleNew;              // new maximum angle between strike of polygon
                           // and strike (or slope) of surface
in>>sizeNpdf;
int Nsize_int [sizeNpdf+1];
double sizeMax [sizeNpdf];

for (i=0; i<sizeNpdf; ++i)
  {Nsize_int[i] = 0;
  in>>sizeMax[i];};
Nsize_int[sizeNpdf] = 0;

Polar MeanP[Nsets];
double FracInt[Nsets];
char pdf[Nsets];
double MeanR[Nsets];
double Ca[Nsets];
double TRatio[Nsets];


// INPUT OF GENERAL STRATAMODEL DATA
in>>porosity;          // 'Y' or 'y' to consider porosity effect on intensity
in>>Pcol;             // Number of columns with Stratamodel porosity cells
in>>Pmin;              // Minimum porosity value: below it rock is shale
in>>GRmax;               // Maximum GR for non-shale
in>>PPmax;             // Ratio of polygons that are marked as fractured if
                       // the rock is mostly shale

// INPUT OF PARAMETERS FOR INDIVIDUAL FRACTURE SETS
for (i=0; i<Nsets; ++i)
  {
in>>FracInt[i];              // Fracture intensity: cum frac area / volume
in>>MeanP[i].theta>>MeanP[i].phi;   // Mean oorientation of the set
in>>pdf[i];                 // Spherical orientation PDF
in>>TRatio[i];               // Translation (non-coplanarity) ration
in>>MeanR[i];
  };
in.close();

// INPUT OF DETAILED STRATAMODEL DATA FROM FILE "1711SGM.dat"
Column SGMcol[Pcol];
in.open("1711SGM.dat");
double XZ, YP, GR;
in>>SGMcol[0].X>>SGMcol[0].Y;
```

```
for (i=0; i<Pcol; ++i)
  { in>>XZ>>YP;
while (YP<1. && YP>0.)
     { in>>GR;
Cell* new_cell = new Cell (XZ, YP, GR);
SGMcol[i].add_cell (*new_cell);
in>>XZ>>YP;
     };
if (i<Pcol-1)
{ SGMcol[i+1].X = XZ;
SGMcol[i+1].Y = YP; };
  };
in.close();

/*************************************************************************
*                CREATING THE MODELING VOLUME
*************************************************************************/
Cubic fold (mA, mB, mC, mD, mE, mF, mG, mH, mI, mJ);    // Folding surface
Surface ground (A, B, C, D, E, F);       // Topographic surface
Volume rock (ground);                    // Modeling volume over area Xm, Ym
double VOLUME = ground.find_enclosed_volume (Xm, Ym);
double Rmax = sqrt(Xm*Xm + Ym*Ym + rock.Zmax*rock.Zmax);
double Zm;
cout<<"Number of fracture sets "<<Nsets<<endl;

/*************************************************************************
*                CREATING OUTCROP PLANES

// VERTICAL OUTCROP PLANE
Plane profile (oAv, oBv, oCv, oDv);
Polygon outPolV = make_initial(profile, rock);
outPolV make_polygon_2d();
outPolV.sort_points_2d();
outPolV.make_polygon_3d();
Zm =  2*(rock.Zmax / sin(profile.dip));
outPolV.add_points_on_surface (profile, rock.top, Np, Zm);

// HORIZONTAL OUTCROP PLANE
Plane plan (oAh, oBh, oCh, oDh);
Polygon outPolH = make_initial(plan, rock);
outPolH.make_polygon_2d();
outPolH.sort_points_2d();
outPolH.make_polygon_3d();

// 3D VIEW OF THE OUTCROP PLANES
//out.open("1711out.plot");
//out<<"Show [ ";   outPolV.print(); out<<", ";
//outPolH.print(); out<<"]  ";
//out.close();
*************************************************************************/
```

```
/*************************************************************************
*                GENERATION OF FRACTURE SETS
*** ********************************************************************/
ListPolygons FracSet[i];
ListPolygons FracSys;
Stat meanSD;
double P32;
out.open ("1711frac.txt", ios::app);
out<<endl<<"mark > "<<ratioMA<<" and < "<<maxR<<" gama "<<gama<<" CA "<<CA<<endl;
out<<"angleR ratioR angleC ratioC strike angleNew "<<endl;
out<<angleR<<" "<<ratioR<<" "<<angleC<<" "<<ratioC<<" "<<strike<<" "<<angleNew<<endl;
out<<"ratioMA CA gama "<<ratioMA<<" "<<CA<<" "<<gama<<endl;
out<<"porosity Ncol Pmin GRmax PPmax "<<porosity<<" "<<Pcol<<" "<<Pmin<<"
"<<GRmax<<" "<<PPmax<<endl<<endl;

for (i=0; i<Nsets; ++i)
  {
int counter = 1;
double TwoMu = FracInt[i]/gama;        // Intensity of plane distances on a line
double expD = exp_value(TwoMu);
double planeD = -Rmax+expD;
MeanA = PI*MeanR[i]*MeanR[i];
cout<<MeanR[i]<<" "<<MeanA<<endl;
MeanArea = MeanA/CA;
double intensity = sqrt(PI/(MeanArea));
cout<<"Fracture set "<<i+1<<endl;
out<<"Fracture set "<<i+1<<endl;
out<<"EXPECTED FRACTURE INTENSITY "<<FracInt[i]<<endl;
out<<"PLANE INTENSITY "<<TwoMu<<endl;
out<<"EXPECTED RADIUS AREA "<<MeanR[i]<<" "<<MeanA<<endl;
out <<"LINE INTENSITY "<<intensity<<endl<<endl;

// GENERATION OF THE FRACTURE PLANES OF A FRACTURE SET
while (planeD < Rmax)
  { ListPolygons* FracPlane = new ListPolygons;
cout<<" set  "<<i+1<<" plane "<<counter<<endl;
Plane p (pdf[i], MeanP[i], planeD);
Polygon temp =  make_initial(p, rock);
if (temp.noP >2)
  { Polygon* pol = new Polygon;
    *pol = temp;
pol->make_polygon_2d();
pol->sort_points_2d();
pol->make_polygon_3d();
Zm =  2*(rock.Zmax / sin(p.dip));
pol->add_points_on_surface (p, rock.top, Np, Zm);
pol->area_radius_3d ();
pol->make_polygon_2d();
double InitialA = pol->area;
//out<<"Initial area IA:  "<<InitialA<<endl;
```

```
// TESSELLATION OF A FRACTURE PLANE INTO POLYGONS
*FracPlane = Poisson_lines_on_polygon (*pol, intensity);

// MARKING OF POLYGONS WITH GOOD SHAPE: ~40% of the TOTAL AREA
FracPlane->mark_good_shape(AngleMin, elongation);
cout<<FracPlane->Npol<<" fractures "<<endl;

// TRANSLATION
FracPlane->translate_2d (MeanR[i], TRatio[i]);
FracPlane->make_listpol_3d();
if (FracPlane->Npol)
  FracSet[i].add_listpol(*FracPlane);
else
  delete FracPlane;
cout<<"total "<<(FracSet[i].Npol+FracSys.Npol)<<endl;
};

expD = exp_value (TwoMu);
planeD += expD;
++ counter;
 };            // End of line tessellation for one plane


// CALCULATING STATISTICS OF THE FRACTURE SET
//meanSD = FracSet[i].find_mean_sd();
//P32 = meanSD.total / VOLUME;
//out<<"Mean SD Total N P32 "<<meanSD<<" "<<FracSet[i].Npol<<" "<<P32<<endl;
//out<<"Mean/expected mean SD/Mean "<<meanSD.Mean/MeanA<<" "<< \
meanSD.SD / meanSD.Mean<<endl<<endl;
//cout<<"Finished calculating statistics "<<endl;
FracSys.add_listpol (FracSet[i]);
 };              // End of generation of one fracture set

double Radius = sqrt (MeanA/PI);
meanSD = FracSys.find_mean_sd(Radius, maxR);
P32 = meanSD.total / VOLUME;
out<<"Mean SD Total N P32 "<<meanSD<<" "<<FracSys.Npol<<" "<<P32<<endl;
out<<"Mean/expected mean SD/Mean "<<meanSD.Mean/MeanA<<" "<<meanSD.SD /
meanSD.Mean<<endl;
cout<<"Finished calculating statistics "<<endl;
out.close ();

/*************************************************************************
*    ROTATION OF FRACTURES TO BETTER FIT SURFACE STRIKE
*************************************************************************/
// RADIAL AND CONCENTRIC FRACTURES WITH ONE OF THE ORIENTATIONS
// PREFERRED
if (strike == 'r' || strike == 'c')
FracSys.mark_by_strike (fold, angleR, angleC, angleNew, strike, ratioR, ratioC);
```

```
/*************************************************************************
 *      INTERSECTION WITH A BOREHOLE
 *************************************************************************/
 // INTERSECTIONS WITH A BOREHOLE
Point P1 (bX1, bY1, bZ1);
Point P2 (bX2, bY2, bZ2);
Line boreL (P1, P2);
Borehole BoreLine = FracSys.intersections_with_logline (boreL);
Stat boreMSD = BoreLine.find_mean_sd_spacing();

if (BoreLine.Nfrac)
{ out.open ("1711bore.txt", ios::app);
out<<endl<<endl<<"Next borehole "<<endl;
out<<BoreLine;
out<<"SPACING mean sd boreline length "<<endl<<boreMSD<<endl<<endl;
out.close();
cout<<"Borehole calculated "<<endl;
out.open ("1711frac.txt", ios::app);
out<<"Borehole N "<<BoreLine.Nfrac<<endl;
out<<"SPACING mean sd boreline length "<<boreMSD<<endl<<endl;
out.close();

out.open ("1711bore.plot");
out<<"Show [ ";
BoreLine.print();
out<<"]  ";
out.close();   }
else
 cout<<"No borehole intersections"<<endl;

/*************************************************************************
*                SIZE  DISTRIBUTION
*************************************************************************/
//out.open ("1711size.txt", ios::app);
//out<<"Mean SD Total N P32 "<<meanSD<<" "<<FracSys.Npol<<" "<<P32<<endl;
//out<<"Mean/expected mean SD/Mean "<<meanSD.Mean/MeanA<<" "<< \
meanSD.SD / meanSD.Mean<<endl<<endl;
//int j;
//FracSys.size_distribution (sizeNpdf, Nsize_int, sizeMax, meanSD.Mean);
//for (j=0; j<sizeNpdf; ++j)
//  out<<sizeMax[j]<<"  "<<Nsize_int[j]<<endl;<<Nsize_int[sizeNpdf]<<endl<<endl;;
//out.close();

/*************************************************************************
 *     MARKING OF POLYGONS BY POROSITY AND GR
 *************************************************************************/
if (porosity == 'y' || porosity == 'Y')
  {
FracSys.discard_high_porosity (SGMcol, Pcol, Pmin, PPmax);
FracSys.discard_shale (SGMcol, Pcol, Pmin, GRmax, PPmax);
```

```
out.open ("1711frac.txt", ios::app);
meanSD = FracSys.find_mean_sd();
P32 = meanSD.total / VOLUME;
out<<"Mean SD Total N P32 after shale marking"<<endl<<meanSD<<" "<<FracSys.Npol<<"
"<<P32<<endl;
out<<"Mean/expected mean SD/Mean "<<meanSD.Mean/MeanA<<" "<<meanSD.SD /
meanSD.Mean<<endl;
out.close();
cout<<"Finished calculating statistics after shale marking"<<endl;

// BOREHOLE INTERSECTION
BoreLine = FracSys.intersections_with_logline (boreL);
boreMSD = BoreLine.find_mean_sd_spacing();

if (BoreLine.Nfrac)
 {
out.open ("1711bore.txt", ios::app);
out<<endl<<endl<<"Borehole after shale marking"<<endl;
out<<BoreLine;
out<<"SPACING mean sd boreline length after shale marking "<<endl<<boreMSD<<endl;
out.close();
cout<<"Borehole calculated after shale marking"<<endl;
out.open ("1711frac.txt", ios::app);
out<<"Borehole N after shale marking "<<BoreLine.Nfrac<<endl;
out<<"SPACING mean sd boreline length after shale marking
"<<endl<<boreMSD<<endl<<endl<<endl;
out.close();

out.open ("1711bore.plot", ios::app);
out<<"Show [ ";
BoreLine.print();
out<<"] ";
out.close(); }; };
};

/*********************************************************************
*    GRAPHICAL OUTPUT FOR MODIFIED FRACTURE SYSTEM

// ALL FRACTURES in a file "5127frac"
// out.open ("1711frac.plot");
// out<<"Show [ ";   FracSys.print();   out<<"] ";
// out.close();

// VERTICAL OUTCROP in a file "tr49profile"
ListLines tracesV = FracSys.traces_on_plane(profile);
cout<<"After changes there are "<<tracesV.Nline<<" profile traces "<<endl;
Stat outVmsd = tracesV.find_mean_sd_length();
out.open ("1711frac.txt", ios::app);
out<<"profile length mean sd total P21 "<<outVmsd<<" "<<outVmsd.total/outPolV.area<<endl;
out.close();
```

File **1711main.C**
(page 9 / 9)

```
out.open ("1711profile.plot");
out<<"Show [ "; outPolV.print();
out<<",  "; tracesV.print(); out<<"]  ";
out.close();

//HORIZONTAL OUTCROP in a file "tr49plan"
ListLines tracesH = FracSys.traces_on_plane(plan);
cout<<"After changes there are "<<tracesH.Nline<<" plan traces"<<endl;
Stat outHmsd = tracesH.find_mean_sd_length();
out.open ("1711frac.txt", ios::app);
out<<"plan length mean sd total P21 "<<outHmsd<<" "<<outHmsd.total/outPolH.area<<endl;
out.close();

out.open ("1711plan.plot");
out<<"Show [ "; outPolH.print();
out<<",  "; tracesH.print(); out<<"]  ";
out.close();
**********************************************************************/
```

File **1711input.dat**
(page 1/ 2)

```
300.
300.
700.

 -0.000001997078643
  -0.000003605472801
  -0.000004268155186
   0.002740574961320
  -0.018972135148053
522.389583405467000

 -0.000000000178968
  -0.000000000350171
  -0.000000000716380
  -0.000000000563307
  -0.000002199139463
  -0.000005788472293
  -0.000007070338889
   0.005271449674254
  -0.017227881147111
1450.988755965268900
```

File 1711input.dat
(page 2 / 2)

```
0.   3.
1.8  .38

2
-0.087  .996  0.  10.
0. 0. 1. 10.
0. 0. 0. 0. 100. 350.

8
0.5236
20.
1.04720
1.6
c
.69813 .9
0.69813 0.6
0.5236

44
 .1  .2  .3  .4  .5  .6  .7  .8  .9  1.
1.1  1.2  1.3  1.4  1.5   1.6  1.7  1.8  1.9  2.
2.1  2.2  2.3  2.4  2.5   2.6  2.7  2.8  2.9  3.0
3.5 4. 4.5 5. 5.5 6. 6.5 7. 7.5 8. 8.5 9. 9.5 10.

y
56
.2
40.
.5

0.6
0.6981503 1.57079
f
10.
30.

.2
-0.8726646 1.57079
f
10.
30.
```

File **1711SGM.dat**
(only first page of file)

-367.4123277 -295.2073351

1237.33398	0.08	70
1232.33398	0.09421	81.06692
1227.33398	0.09173	54.14182
1222.33398	0.08551	36.5926
1218.35101	0.06071	34.52285
1216.855895	0.13986	39.43117
1216.70093	0.12288	28.00477
1216.2995	0.11461	22.81259
1215.708005	0.08947	24.56693
1214.37512	0.0854	34.51685
1212.71594	0.09808	36.87631
1211.983215	0.09975	27.33395
1211.118225	0.11516	19.42662
1209.53552	0.1547	13.29077
1207.952695	0.16462	12.88014
1206.36975	0.14792	15.40197
1204.786925	0.13229	16.77268
1203.20404	0.11756	22.32459
1201.4126	0.10343	43.04713
1199.4126	0.14154	53.74398
1197.726685	0.15535	33.29139
1195.773925	0.18364	17.41536
1193.240295	0.19095	10.50372
1190.706725	0.18598	10.08062
1188.173095	0.18828	10.37848
1185.639405	0.17277	13.79442
1183.105775	0.16996	13.58084
1180.572085	0.19738	11.76708
1178.03833	0.19141	11.63857
1175.50476	0.20119	10.51227
1172.971195	0.17705	11.34673
1170.70435	0.13921	16.69334
1169.40204	0.14682	17.83624
1168.494385	0.16461	15.83556
1167.28363	0.15082	16.38573
1166.072875	0.15	16.54417
1164.862185	0.16296	14.71723
1163.65149	0.15384	15.99253
1162.440735	0.13614	17.96447
1160.83539	0.11909	22.60125
1159.02777	0.1116	25.85746
1157.841125	0.11652	24.45264
1157.08313	0.12816	23.20313
1156.32501	0.13447	24.3166

.....

File **1755input.dat**
(page 1 / 2)

```
300.
300.
700.

-0.000001997078643
 -0.000003605472801
 -0.000004268155186
 -0.012358108090774
 -0.035134659455770
493.318386916569100

 -0.000000000178968
 -0.000000000350171
 -0.000000000716380
 -0.000000000563307
 -0.000004164239328
 -0.000008819860443
 -0.000010278745111
 -0.019531548446880
 -0.049997557532956
1415.605351335590500

0. 3.
1.8 0.38

2
.996  -0.087 0.  10.
0. 0. 1. 10.
0. 0. 0. 0. 100. 400.

8
0.5236
20.
1.04720
1.6
r
.7854 0.6
0.7854 1.
0.5236

44
 .1  .2  .3  .4  .5  .6  .7  .8  .9  1. 1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.
2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9  3.0  3.5 4. 4.5 5. 5.5 6. 6.5 7. 7.5 8. 8.5 9. 9.5 10.

n
56
.2
40.
.5
```

File **1755input.dat**
(page 2 / 2)

```
0.02
0.5236 1.57079
f
1.
30.

.06
-0.5236 1.57079
f
1.
30.
```

File **17d5input.dat**
(page 1 / 2)

```
200.
800.
800.

 -0.000000849681775
 -0.000004158799743
 -0.000006507978129
  0.007188944094383
 -0.008396895931382
411.816395521319900

 0.000000001052789
 -0.000000001340217
 -0.000000001577960
 -0.000000001175702
  0.000000556219013
 -0.000006669332017
 -0.000006787429663
  0.008519193573786
 -0.005660626384491
1439.423483158074100

0. 3.
1.8 0.38

2
-0.087 .996 0. 10.
0. 0. 1. 10.
0. 0. -750. 750. 200. 200.
```

File **17d5input.dat**
(page 2 / 2)

```
8
0.5236
20.
1.04720
1.6
c
.5236 .5
0.69813 0.8
0.5236

44
 .  .2 .3 .4 .5 .6 .7 .8 .9 1. 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.
2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.5 4. 4.5 5. 5.5 6. 6.5 7. 7.5 8. 8.5 9. 9.5 10.

n
54
.2
40.
.5

0.06
0.6981503 1.57079
f
1.
30.

0.02
-0.8726646 1.57079
f
1.
30.
```

File **2511input.dat**
(page 1 / 2)

```
300.
300.
700.

-0.000001997078643
 -0.000003605472801
 -0.000004268155186
  0.016639967528111
  0.016308414001760
532.051250996195700
```

File **2511input.dat**
(page 2 / 2)

```
-0.000000000178968
 -0.000000000350171
 -0.000000000716380
 -0.000000000563307
 -0.000000909607526
  0.000000094525068
 -0.000000086962461
  0.016265068981885
  0.012424607637665
1445.804913203465500

0.   3.
1.8   .38
2
.996  -0.087 0.  10.
0. 0. 1. 10.
0. 0. 0. 0. 100. 500.

8
0.5236
10.
1.04720
1.6
r
.69813 1.
0.69813 0.8
0.5236

60
 .1  .2  .3  .4  .5  .6  .7  .8  .9  1. 1.1  1.2  1.3  1.4  1.5   1.6  1.7  1.8  1.9  2.
2.1  2.2  2.3  2.4  2.5   2.6  2.7  2.8  2.9  3.0   3.2  3.4  3.6  3.8   4.  4.2  4.4  4.6  4.8  5.
5.2  5.4  5.6  5.8   6.0 6.5  7.   7.5  8.   8.5   9.   9.5  10.  10.5  11.  12.  13.  14.  15. 16.

n
81
.16
30.
.25

.4
0.5236 1.57079
f
3.
30.

1.
-0.5236 1.57079
f
3.
30.
```

File **4007input.dat**
(page 1 / 2)

```
300.
300.
700.

-0.000002776821280
  0.000002874802670
 -0.000004363820585
 -0.026632838598135
 -0.021354683372863
595.012795908047792

 -0.000000002131057
  -0.000000002690133
   0.000000000006221
   0.000000000545625
  -0.000005091213805
   0.000003758923729
  -0.000004811661193
  -0.009106515975567
  -0.015577945335875
1505.590721527707700

1. 3.
3.6 0.36

2
-0.087  .996  0.  10.
0. 0. 1. 200.
0. 0. 0. 0. 100. 250.

8
0.5236
30.
1.04720
1.6
c
0.69813 0.4
0.2618  0.4
0.5236

51
 .1  .2  .3  .4  .5  .6  .7  .8  .9  1. 1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.
2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9  3.0  3.2  3.4  3.6  3.8  4.  4.2  4.4  4.6  4.8  5.
5.2  5.4  5.6  5.8  6.0 6.5  7.  7.5  8.  8.5
9.

n
72
.2
.5
```

File **4007input.dat**
(page 2 / 2)

```
.3
0.6981503 1.57079
f
1.
30.

.3
-0.8726646 1.57079
f
1.
30.
```

File **4903input.dat**
(page 1 / 2)

```
300.
300.
700.

 -0.000002776821280
  0.000002874802670
 -0.000004363820585
 -0.031093083649278
  0.010631093441253
655.058372739859124

 0.000000002310403
  0.000000000423665
  0.000000000880546
  0.000000000150196
  0.000001582329892
  0.000001714595706
 -0.000005658621600
 -0.045849219784381
  0.014157850392298
1539.192142367417546

1. 3.
3.6 .36

2
-0.087  .996  0.  10.
0. 0. 1. 200.
0. 0. 0. 0. 120. 320.
```

File **4903input.dat**
(page 2 / 2)

```
8
0.5236
30.
1.04720
1.6
c
0.69813 0.6
0.2618  0.6
0.5236

51
.1  .2  .3  .4  .5  .6  .7  .8  .9  1. 1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.
2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9  3.0  3.2  3.4  3.6  3.8  4.  4.2  4.4  4.6  4.8  5.
5.2  5.4  5.6  5.8  6.0 6.5  7.  7.5  8.  8.5  9.

n
81
.2
.5

.3
0.6981503 1 57079
f
1.
30.

.3
-0.8726646 1.57079
f
1.
30.
```

File **5127input.dat**
(page 1 / 2)

```
300.
300.
700.

-0.000002776821280
 0.000002874802670
-0.000004363820585
-0.009830140857970
-0.028399848131740
658.005980018862829
```

File **5127input.dat**
(page 2 / 2)

```
 -0.000000014952633
  0.000000002662401
  0.000000013007986
  0.000000006271154
 -0.000022542677996
  0.000000090124917
  0.000000978844408
  0.002855853736420
 -0.034564090240907
1558.712233719843900

1. 3.
3.6 .36

2
-0.087 .996 0. 10.
0. 0. 1. 200.
0. 0. 0. 0. 100. 300.

8
0.5236
30.
1.04720
1.6
c
0.69813 0.3
0.2618 0.4
0.5236

60
 .1 .2 .3 .4 .5 .6 .7 .8 .9 1. 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.
2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.0 3.2 3.4 3.6 3.8 4. 4.2 4.4 4.6 4.8 5.
5.2 5.4 5.6 5.8 6.0 6.5 7. 7.5 8. 8.5 9. 9.5 10. 10.5 11. 12. 13. 14. 15. 16.

y
81
.2
.5

.03
0.6981503 1.57079
f
1.
30.

.03
-0.8726646 1.57079
f
1.
30.
```

File **3728input.dat**
(page 1 / 2)

```
300.
300.
700.

 -0.000000360047806
  0.000001035175917
 -0.000004363820585
 -0.001788501844326
  -0.037097553435224
733.258264105697663

 0.000000000107871
  0.000000000054933
  0.000000000317072
  0.000000000150196
 -0.000001565148758
  0.000002924414993
 -0.000005518551548
  0.000387801873314
  -0.049758973800385
1528.133977779659290

1.   3.
3.6 .36

2
-0.087  .996  0.  10.
0. 0. 1. 200.
0. 0. 0. 0. 100. 500.

8
0.5236
20.
1.04720
1.6
c
0.69813 0.4
0.2618  0.4
0.5236

60
 .1  .2  .3  .4  .5  .6  .7  .8  .9  1. 1.1  1.2  1.3  1.4  1.5   1.6  1.7  1.8  1.9  2.
2.1  2.2  2.3  2.4  2.5   2.6  2.7  2.8  2.9  3.0   3.2  3.4  3.6  3.8   4.  4.2  4.4  4.6  4.8  5.
5.2  5.4  5.6  5.8   6.0 6.5  7.   7.5  8.   8.5  9.   9.5  10.  10.5  11.  12.  13.  14.  15. 16.

y
81
.2
.5
```

File **3728input.dat**
(page 2 / 2)

```
.3
0.6981503 1.57079
f
3.
30.

.3
-0.8726646 1.57079
f
3.
30.
```

File **4959input.dat**
(page 1 / 2)

```
300.
300.
700.

-0.000002776821280
  0.000002874802670
 -0.000004363820585
 -0.024869287753803
  0.016084690353363
787.765776298391529

 0.000000002310403
  0.000000000423665
  0.000000000880546
  0.000000000150196
 -0.000010992411295
 -0.000001870942692
 -0.000007731579827
 -0.029372840577478
  0.030339763710972
1586.105803479595124

1.   3.
3.6 .36

2
-0.087  .996  0.  10.
0. 0. 1. 200.
0. 0. 0. 0. 100. 500.
```

File **4959input.dat**
(page 2 / 2)

```
8
0.5236
20.
1.04720
1.6
c
0.69813 0.4
0.2618  0.4
0.5236

60
 .1  .2  .3  .4  .5  .6  .7  .8  .9  1. 1.1  1.2  1.3  1.4  1.5   1.6  1.7  1.8  1.9  2.
2.1  2.2  2.3  2.4  2.5   2.6  2.7  2.8  2.9  3.0   3.2  3.4  3.6  3.8   4.  4.2  4.4  4.6  4.8  5.
5.2  5.4  5.6  5.8   6.0 6.5  7.   7.5  8.   8.5   9.   9.5  10.  10.5  11.  12.  13.  14.  15. 16.

n
81
.2
.5

.3
0.6981503 1.57079
f
3.
30.

.3
-0.8726646 1.57079
f
3.
30.
```