THE MECHANICS AND KINEMATICS OF

DISTRIBUTED DEFORMATION AND BLOCK ROTATION

IN THREE DIMENSIONS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF GEOPHYSICS

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

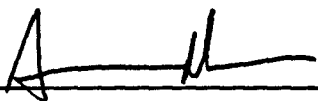FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Oona Scotti

June 1991

ii

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.
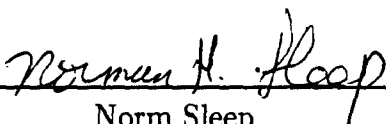
Amos Nur (Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Gary Mavko

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Norm Sleep

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.
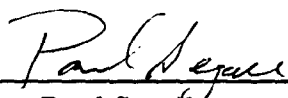
Paul Segall

Approved for the University Committee
on Graduate Studies:

Dean of Graduate studies

Abstract

This study presents three new models for the deformation of the earth's crust, in which two classical assumptions are relaxed: active crustal faults that should (1) be optimally oriented in the tectonic stress field and (2) slip in accordance with laboratory derived friction criteria. The models we propose are based on the well known observation that, in spite of measured stress field directions that are relatively uniform within a given region, brittle deformation in the upper crust is commonly distributed among numerous faults in different orientations. In addition, some faults are short, forming subparallel sets confined to domains (D-faults), which have been shown to undergo rotation. Others are long, throughgoing and usually separate one domain from the next (B-faults).

In the first model, referred to as the faulting model, we estimate in a forward fashion a regionally homogenous stress field that can slip all the D-faults of a region in accordance with friction criteria. The estimated stress fields for three regions in California and Nevada are in good agreement with measured stress directions and laboratory values for the friction coefficient of faults. For B-faults, on the other hand, we found that the faulting model does not yield satisfactory results: estimated friction coefficients are often too low or too high.

Following previous researches, we speculate that appreciable stress field perturbations may arise along B-faults due to the presence of gouge zones. To quantify these perturbations, we model gouge zones as elliptical regions of low elastic moduli. The model results depend on the moduli contrast between the gouge zone and the country rock, the relative principal stress magnitudes, and the orientation of the fault zone in the stress field. For fault zones such as the San Andreas, for example, oriented almost perpendicular to the direction of maximum compression, the model predicts that inside the fault zone the deviatoric stress should be greatly reduced, the mean stress should be increased, and the direction of maximum compression should be less

perpendicular, assuming a gouge with high Poisson's ratio and low bulk modulus.

In the third model, referred to as the three dimensional block rotation model, we track fault slip and block rotation during the deformation of a single domain. Faults in the domain are assumed parallel, the fault bounded blocks are rigid and slip is controlled by friction. Because the Coulomb criteria for rock fracture limits the amount of rotation that can occur along a single set of faults, the model predicts the presence of multiple sets of faults in domains that have undergone great amounts of deformation. A key parameter in the model is the stress ratio $\phi = (\sigma_2 - \sigma_3)/(\sigma_1 - \sigma_3)$, where $\sigma_1 > \sigma_2 > \sigma_3$ are the principal stress magnitudes. When $\phi = 0.1$, the slip history of a fault set can be very complex, typically changing from dip-slip to strike-slip. When $\phi = 0.9$ slip is relatively constant throughout rotation. Because during rotation fault slip can change, the model predicts inclined axis of rotation of blocks and faults. Combining frictional constraints of the block rotation model with paleomagnetic, structural and geological data, we show how only one set of faults, preexisting and rotating in an irrotational strike-slip stress field, can account for the three major phases of deformation observed in the Western Transverse Range domain, Southern California: preexisting north-northeast faults were reactivated as normal faults, rotated and became strike-slip, and subsequent rotations of faults resulted in their present east-west high angle reverse orientation. Therefore it is not necessary to invoke complex regional and local changes in the stress regime or erratic changes in plate motion to account for complex histories of faulting.

In conclusion, assuming that the regional stress field is uniform and that crustal deformation is achieved by domains of D-faults that act as a unit, separated from one another by throughgoing B-faults, this study shows that: (a) most active faults are not optimally oriented in the stress field; (b) the most likely cause for their non-optimal orientation is that within domains deformation is often accommodated by fault slip and block rotation; (c) modeled block and fault rotation can yield complex

slip histories for a single setaults; (d) D-faults can slip in accordance with friction criteria but slip along B-faults appears to be controlled by a different criteria; (e) gouge zones, which are most likely responsible for B-fault behavior, can lead to local stress fields that may be quite different from the regional one. Therefore B-faults should not be used to infer regional stress field directions.

Acknowledgements

The work presented in this thesis was inspired by my advisor Amos Nur. I would like to express sincerest gratitude to him for his continuing support, both financial and moral. His physical insight, vision and understanding of scientific problems have made my stay at Stanford a unique and challenging experience; his ability to focus on the essential details provided a guiding hand when I wondered off the right track; and last but not least, I would like to thank him for his patience in reviewing and editing the many versions of this thesis.

I would like to thank all the Stanford Rock Physics people that patiently listened to my numerous seminars and gave me advice whenever I needed it. Thanks are due in particular to Gary Mavko for his physical insight in rock mechanics, that made difficult problems look simple and for being a supportive friend when times were tough. I would also like to thank Raul Estevez for his help, advice and friendship. During his stay at Stanford he participated actively in my research project and wrote a good part of the three dimensional block rotation program. Thanks also go to Gary Mavko, Greg Beroza, Norm Sleep, and Paul Segall for taking the time to read my thesis, make useful comments, and provide stimulating discussions as members of my dissertation committee.

I would also like to thank the Rock Mechanics people for helping me put my video together. In particular I would like to thank Dave Pollard for allowing me to use not only the video camera and editing system set-up but also the laboratory facilities.

I have also appreciated Stan Ruppert's help and moral support. In particular I would like to thank him for answering computer related questions when he could have instead referred me to the manual. Of course, writing a thesis and doing a Ph.D. is one thing but making sure that all necessary accessories are readily available is another, for this I thank Margaret Muir. Her kindness, patience, and understanding kept me from going off the deep end. I also appreciated Linda Farrel's help, especially

during these last few months.

Finally, I wish to thank the many friends in the Bay area for making these years in California unforgettable. In particular I would like to thank Inge Infante, a friend, a cook, and an artist; thanks to her a lot of the figures in this thesis are neat and tidy. I am very greatful to my husband, Jean-Pierre Panziera, for his friendship, his love, his infinite patience, and his ever present and positive support throughout my thesis. Thanks are also definitely due to Fiona, our two months old daughter, that forced us to have a healthier perspective on life and kept me from taking myself too seriously during these last few months. Lastly, I would like to thank my parents, my sister, my brothers, and the extended family for their continuos encouragement and support throughout.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding the processes by which the upper brittle crust deforms has been the challenge of this century in the field of tectonophysics. In his seminal paper of 1905 and in his memoir of 1942, E. M. Anderson developed the modern mechanical concepts of the origin of faults and emphasized their important role in tectonics. Anderson's theory predicts that faults should form conjugate sets, with planes inclined at acute angles on either side of the maximum principal stress and which include the intermediate principal stress direction. Such faults are referred to as being optimally oriented in the stress field. By applying the condition that near the free surface one of the principal stresses, S_v, be vertical Anderson showed that three major classes of faults result: reverse , normal, and strike-slip when $S_v = \sigma_3, \sigma_1$, and σ_2 respectively $(\sigma_1 > \sigma_2 > \sigma_3)$. Furthermore, assuming the crust to be pervaded with faults of all orientations, only optimally oriented faults should be active and as laboratory experiments on a variety of rock types suggest (Byerlee, 1978), the acute angle between faults and the σ_1 direction should be about 30^0. Although it is generally recognized that these simplifying assumptions have their shortcomings (Scholz, 1990; Nur et al., 1986; McKenzie, 1969; Bott, 1956), for lack of better models it is still customary (Brace and Kohlstedt, 1980; Zoback and Healy, 1984) to assume optimally oriented active faults and Byerlee type friction criteria to infer the tectonics of a region.

The claim of this thesis is that distributed continental deformation is characterized

by numerous active faults with a variety of orientations and slip directions, and that this apparently complex geometry of faults is often organized in domains of faults, that act as a unit, separated by domain-bounding faults. Typically domain faults form a set of short, sub-parallel features whereas domain-bounding faults are throughgoing features with well developed gouge zones. Given that measured stress field directions are often homogeneous over a region of domains, not all active faults can be optimally oriented in the regional stress field. Furthermore, results from paleomagnetic studies obtained during the last two decades suggest that rotations of crustal blocks and faults has occurred within some of these domains.

Rotations about a horizontal axis have been readily recognized in normal faulting domains thanks to the measurable tilt of originally flat lying beds (Ransome, 1910). Vertical axis rotation, on the other hand, could only be inferred in strike-slip faulting domains (Freund, 1970, 1971, 1974). Proof of such rotations has come through relatively recent paleomagnetic studies, which reveal rotations when paleomagnetic declinations within the deforming region are compared with those found in rocks of the same age in stable regions outside the deforming zone. Such rotations were first described in Oregon (Cox, 1957) and have since been recognized in many parts of the world (see Kissel and Laj, 1988).

Two major classes of models for crustal deformation exist in the literature: kinematic models and a mechanical model. The kinematic models (e.g. McKenzie and Jackson, 1986) claim that crustal blocks and faults follow passively the deformation of the underlying, ductile portions of the lithosphere. Tractions are either applied from below or at the boundaries of the deforming domain. The details of each kinematic model are discussed at length in Kissel and Laj (1988). For the purpose of this study, it's important to note that (1) the mechanics of faults is not considered in kinematic models, and (2) kinematic models predict a single sense of rotation of blocks and faults in a given region.

The mechanical model (Nur et al. 1986), on the other hand, supports the idea that active fault slip controls the kinematics of rotating blocks. Because rotation of domains is attributed to domain interaction, both clockwise and counterclockwise rotation of blocks and faults are possible, depending on whether the domain has left-lateral or right-lateral faults. In this model the slip history of D-faults is controlled by friction only, tractions at domain boundaries and at depth are not considered.

The success of kinematic models in describing the long wavelength characteristics of continental deformation arises from the fact that only the very upper layer of the lithosphere deforms in a brittle fashion. Therefore it is reasonable to expect that at a large scale the deformation be controlled by the distributed flow in the lower crust and mantle lithosphere. But, it is clear that deformation is discontinuous in the upper rigid crust and that fault mechanics must play a key role. The objective of this study is to develop new techniques in an attempt to provide a more realistic description of distributed continental deformation that combines the kinematics of block rotation with the mechanics of faulting.

Chapter 2 presents a new faulting model to interpret the complex pattern of faults observed in three regions of distributed deformation: the San Francisco Bay area in northern California, southern California and the Walker Lane between California and Nevada. In the model the orientation of faults is not assumed to be optimal in the stress field. Three useful techniques are developed to: (1) estimate which faults can be reactivated for given material parameters and stress field orientation; (2) distinguish between mechanically different types of faults before estimating the stress field of a region, and (3) calculate the stress field disturbance caused by the presence of a region of low moduli, such as the San Andreas fault zone, by making use of Eshelby's (1957) solution for an elliptical inhomogeneity.

Chapter 3 generalizes to three dimensions the original two-dimensional model of Nur et al. (1986). In the original model it is proposed that active non-optimally

oriented faults, although created in an optimal orientation, rotated into their present orientation by a mechanism that is now known in the literature as block rotation. The new three dimensional model presented here tracks fault slip and block rotation in a three-dimensional stress field. Rotations may occur about any axis: inclined, horizontal, or vertical. Therefore generalized fault slip histories can be simulated. Furthermore, fault slip in the model is controlled by the friction criteria for sliding and faults become locked when the Coulomb fracture limit is reached, predicting multiple sets of faults in highly deformed domains. The model is applied to the West Transverse ranges domain in southern California. This chapter has been accepted for pubblication in the Journal of Geophysical Research (1991).

Chapter 4 is a twenty minutes long video to help visualize some of the three dimensional block rotation models results. The animations show idealized blocks and faults that slip and rotate in a fixed stress field.

Appendix A and B detail the mathematical formulations of chapter 2 and 3. Appendix C describes the computer programs used to calculate the results of chapter 2, and 3: (1) the FAULTSVP program (C-language) estimates slip directions and friction values of a family of faults for given material parameters (friction and cohesion) and stress field; (2) the ESHELBY program (C-language) calculates stress disturbances that result from the presence of an elliptical inclusion with elastic moduli that differ from the matrix; (3) the BLOCKROTATION3D program (Fortran-language) tracks fault slip and block rotation in a three dimensional stress field.

1.1   Discussion

In the following study we attempt to understand the mechanics and kinematics of the complex faulting observed in many regions of distributed continental deformation. In doing so we idealize the crust as being characterized by fault bounded, parallel, rigid blocks that accommodate all the brittle deformation of the upper crust by slip along

4

faults. Slip is assumed to be in accordance with laboratory derived friction criteria. Because blocks and faults appear to be organized in domains, we speculate that domains act as a unit. This is a reasonable assumption because (1) within domains a single fault orientation appears to accommodate the long-term deformation, and (2) domain faults do not cut domain-bounding faults, the latter beeing typically throughgoing features in both geographical extent and in depth.

We have also explored a mechanism for the rotation of blocks and faults in certain domain. Although evidence for rotations has been accumulating for some time, the mechanics of such rotations are still uncertain. We have considered the simplest approach: (1) homogeneous stress boundary conditions throughout the domain, so that the same stress field is present at every point in the domain; (2) friction controls slip and rotation of blocks and faults, ignoring the role of surrounding domains and their possible influence on its deformation history; and (3) at depth blocks are somehow detached from the underlying ductile crust.

Clearly, in order to render the problem tractable we have made simplifying assumptions but in reality: each block does undergo some internal deformation, within a given domain more that one fault orientation may be active, the predominant faults in a domain are not exactly parallel, and faults may slip with different friction coefficients. Furthermore, in some areas, it may be very difficult, and therefore subjective, to distinguish between domain and domain-bounding faults, and lastly, the tractions that are most likely present below and at the boundaries of each domain must also play an important role in the deformation history of domains.

With the above shortcomings in mind, we believe that, to a first order, the faulting model and the block rotation model, as presented in the following study, provide a positive step towards a more realistic description of how, in general, the brittle upper crust may deform. The models are simple and yet the results can be quite complex. Furthermore, we speculate that domains are surrounded by gouge zones

5

but we do not provide a mechanical model to explain how slip occurs along them. Future modeling should integrate both domains and domain bounding faults and may consider the domain as the equivalent to the unit cube in linear elasticity. An important difference between the unit cube and the domain is that the stress-strain relations of a deforming domain are much more complex because they are non-linear. So, the next step is to characterize this new rheology by computing the elastic moduli that relate stresses and strains in a region of domains and then to attempt the study of how two or more domains interact and what the role of domain bounding faults may be.

References

Anderson, E. M., The dynamics of faulting, *Trans. Edinburgh Geol. Soc., 8*, 387-402, 1905.

Anderson, E. M., *The dynamics of faulting*, 1st ed. Edinburgh: Oliver and Boyd, 1942.

Byerlee, J., Friction of rocks. *Pageoph, 116*, 615-626, 1978.

Brace, W. F. and Kohlstedt, D., Limits on lithospheric stress imposed by laboratory experiments, J. Geophys. Res., 85, 6248-6252,1980.

Bott, M.H.P., The mechanics of oblique slip faulting. *Geol. Mag., XCVI*, 2, 1959.

Cox, A., Remanent magnetism of Lower to Middle Eocene basalt flows from Oregon. *Nature, 179*, 685-686, 1957.

Eshelby, J. D., The determination of the elastic field of an ellipsoidal inclusion, and related problems., *Proc. Roy. Soc. London., Ser. A, 241*, 376-396, 1957.

Freund, R., Rotation of strike-slip faults in Sistan, Southeast Iran, *J. Geol. 78*, 188-200, 1970.

Freund, R., The Hope fault, a strike-slip fault in New Zealand, *N. Z. Geol. Surv. Bull., 86*, 49pp., 1971.

Freund, R. , Kinematics of transform and transcurrent faults, *Tectonophysics, 21*, 93-134, 1974.

Kissel, C., C. Laj, *Paleomagnetic rotations and Continental deformation*, edited by C. Kissel and C. Laj, NATO ASI Series, Series C: Mathematical and Physical Sciences, 254, 181-208, 1988.

McKenzie, D., The relation between fault plane solutions for earthquakes and the directions of principal stresses, *Bull. Seismol. Soc. Am., 59*, 2, 591-601, 1969.

McKenzie, D., and Jackson, J. A., A block model of distributed deformation by faulting, *J. Geol. Soc. Lond., 143*, 349, 1896.

Nur, A., H. Ron, and O. Scotti, Fault mechanics and the kinematics of block rotations, *Geology, 14*, 746-749, 1986.

Ransome, F.L., W. H. Emmons, and G. H. Garrey, Geology and ore deposit of the Bullfrog district, Nevada, *U. S. Geol. Surv. Bull., 407*, 1-130, 1910.

Scholz, C. H., *The mechanics of earthquakes and faulting.* Cambridge Univeristy Press, 1990.

Zoback, M.D., and J. H. Healy, Friction, faulting, and << in situ >> stress, *Annales Geophysicae, 2,* 6, 689-698, 1984.

Chapter 2

Strong domain faults and soft domain bounding faults in regions of distributed deformation

Abstract

Regions of distributed deformation are often characterized by short, sub-parallel faults confined to domains, surrounded by longer, throughgoing faults that separate one domain from the next. In this chapter we develop a faulting model to compute stress field directions and relative stress magnitudes in such regions. The fundamental assumption that active faults have to be optimally oriented in the stress field is completely relaxed, one principal stress direction is assumed vertical, and fault slip is assumed consistent with friction criteria. The model is applied to three regions of distributed deformation: the San Francisco Bay Area in northern California, the southern California region, and the Walker Lane belt along the California-Nevada border. Estimated stress field orientations for each region are in general agreement with measured stress directions. Most importantly, calculated friction coefficients for faults within domains range between 0.3 and 0.6, but those for throughgoing domain-bounding faults span a much wider range of values.

Following previous researchers, we speculate that the different rheological behavior of domain-bounding faults may be controlled by the elastic properties of their

characteristically wide gouge zones. We use Eshelby's solution for an elliptical inclusion to model low moduli fault zones and to calculate the resulting disturbed stress field. The results show that, in general, within and close to major bounding faults both the orientation and the magnitude of the stress field may be quite different from the regional stress field. For the case of the San Andreas fault, assuming a high Poisson's ratio and a low bulk modulus gouge, inside the fault zone the model predicts greatly reduced shear stresses, an increased mean stress, and a less perpendicular maximum stress direction zone compared to remote values.

In conclusion, at least two types of faults characterize regions of distributed deformation: (1) strong, domain faults that slip in accordance with friction criteria in a regionally uniform stress field, and (2) soft, domain-bounding faults where slip is controlled by the rheological properties of gouge material and the state of stress may be very different from the regional one.

2.1 The problem

Very commonly the brittle continental upper crust is characterized by distributed deformation with many active faults of varying length, orientation and slip mechanism. It has been traditionally assumed that (1) slip on these faults is governed by frictional sliding as observed in laboratory experiments (Byerlee, 1978), and (2) active faults are always optimally oriented (Figure 2.1) in the regional stress field (Anderson, 1956; Zoback and Healy, 1984; Brace and Kohlstedt, 1980).

The validity of both assumptions is questioned here. The first assumption has been questioned before because of heat flow studies (e.g. Lachenbruch and Saas, 1974, 1991) and recent findings at the Cajon Pass scientific drillhole (Zoback et al., 1987), which indicate that the San Andreas fault has very low shear strength. Because active faults occur at different orientations within a given region, the second assumption that they should all be optimally oriented would require that the state of stress be more

10

spatially heterogeneous than is inferred from breakouts and hydrofractures (Zoback and Zoback, 1980; Zoback et al., 1989).

In this chapter we develop a faulting model that completely relaxes the second assumption. The model is applied to three regions of distributed faulting: (1) the San Francisco Bay Area in northern California, (2) southern California region, and (3) the Walker Lane belt along the Nevada-California border. A simple pattern of active faults is found in these regions: a few, long, throughgoing faults which surround domains with many short faults. The active faults in each domain are typically restricted to one orientation, which is often quite different from neighboring domains. Obviously not all faults can be optimally oriented in a regionally homogeneous stress field. Using accepted friction criteria and the Mohr circle we develop a method to search for a single regional stress field orientation which is consistent with fault orientations and slip directions in all of the domains.

As we show below, our faulting model can reconcile friction criteria and faulting within domains only. Large, domain-bounding faults appear to belong to a separate class of faults. Many researches (Wang, 1984; Lachenbruxch and Saas, 1973, 1991; Byerlee, 1990; Rice, 1990) have speculated on the nature of the rheological properties of mature faults, such as the domain-bounding San Andreas fault in California. Thick fault gouge, characteristic of mature fault zones, is the most likely cause for their special rheology. Consequently, the stress in these zones may be quite different from the normal, surrounding crust. In the second part of this chapter we will therefore study the effects of the elastic properties of fault gouge on the local stress field.

2.2 Regions of active distributed deformation

The San Francisco Bay (Figure 2.2) and southern California (Figure 2.3) are both regions of distributed deformation which are dominated by the active, NW transform boundary between the Pacific and North American plates. However, in both regions,

11
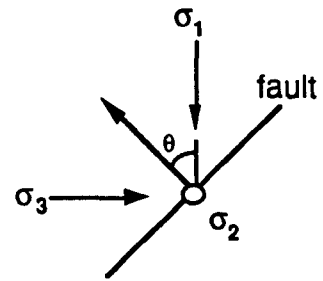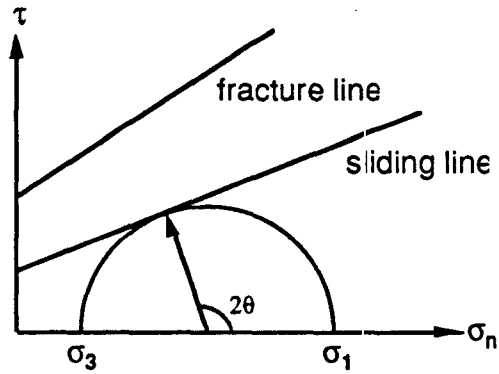
Fig. 2.1 A fault is considered optimally oriented in the stress field if at slip time the state of stress is such that the Mohr circle is tangent to the friction line. Laboratory experiments indicate that friction coefficients of faults should be around 0.6 at pressures > 2 kbars. Therefore, if active faults were always *optimally* oriented in the stress field, their normals should make an angle of roughly 60^0 to the direction of maximum compressive stress σ_1 (equivalent to 120^0 on the Mohr circle representation).

domains of active strike-slip or high-angle reverse faulting are found (Luyendyk et. al 1980; Aydin and Page, 1984; Webb and Kanamori,1985; Nicholson et al., 1986a, 1986b; Jones, 1988; Oppenheimer et al., 1988; Williams et al, 1990). Major strands of the San Andreas fault and other throughgoing structures define the boundaries of these domain. The Walker Lane belt (Figure 2.4), about 700 km long and 200 km wide between the Basin and Range and the Sierra Nevada is also characterized by domains and domain-bounding faults (Stewart, 1988). The first question we wish to address is whether for each region, a single stress field orientation can control slip that is consistent with friction criteria on all the active faults of the region.

2.3   Faulting model

Let us idealize the observed geometry of faults into D-faults, domains of parallel faults, separated by B-faults, domain-bounding, throughgoing fault zones. Measured stress field orientations across a region of domains are relatively homogeneous in direction (Zoback et al., 1987) and deformation in each domain occurs along a single set of sub-parallel faults. Consequently, to find a stress field that is consistent with faulting in all the domains of a region, we must assume that active faults are not necessarily optimally oriented in the stress field. Furthermore, we assume that one principal stress is vertical $S_v = \sigma_1$, σ_2, or σ_3 ($\sigma_3 < \sigma_2 < \sigma_1$) and that faults slip when

$$\tau_f = \mu_f \sigma_n \tag{2.1}$$

where τ_f and σ_n are the shear and normal stresses resolved on the fault, and μ_f is the coefficient of friction of the faults.

Therefore the upper bound to stress magnitudes in the brittle upper crust is not the frictional strength of optimally oriented faults (Zoback and Healy, 1984), but the frictional strength of intact, unfaulted crust (Scotti et al., 1991; Nur et al., 1986):

$$\tau_0 = C_0 + \mu_0 \sigma_n \tag{2.2}$$

13
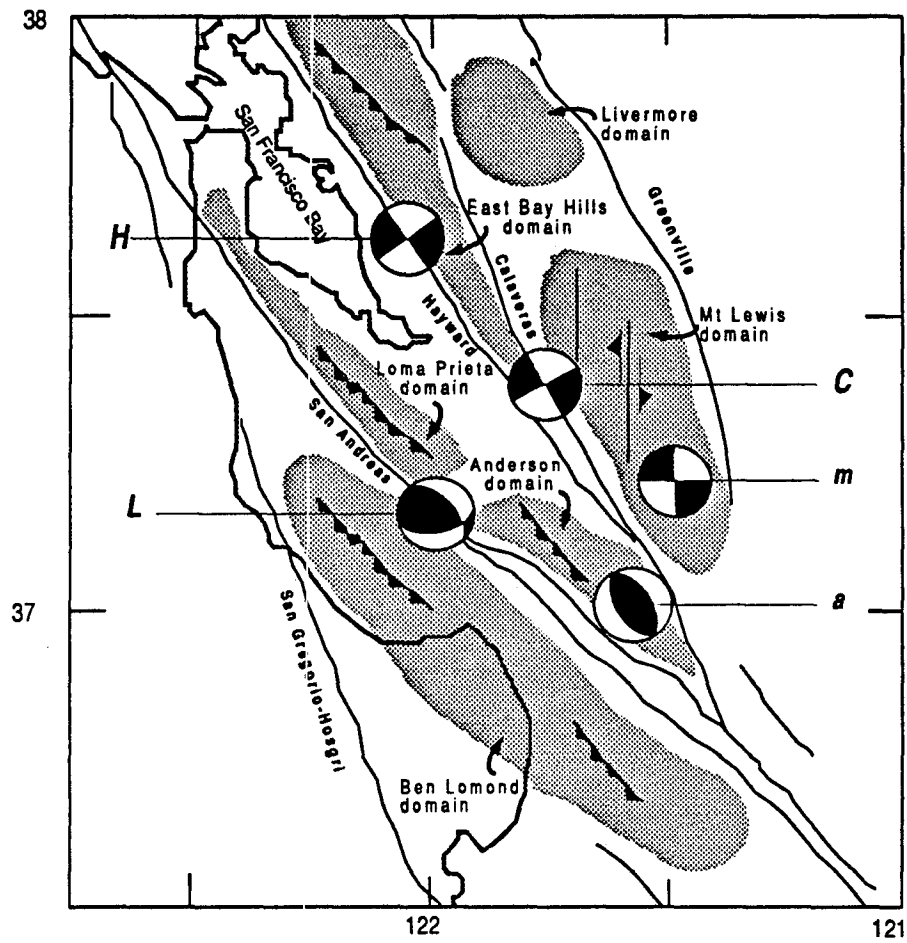
Fig. 2.2 Simplified map of domains and domain-bounding faults in the Bay area, northern California. Base map and domains adapted from Aydin and Page (1984). Representative focal mechanisms are also shown. Capital letters are for domain-bounding faults, small letters for domain faults (see Table 2.1 for fault identification and references).

14
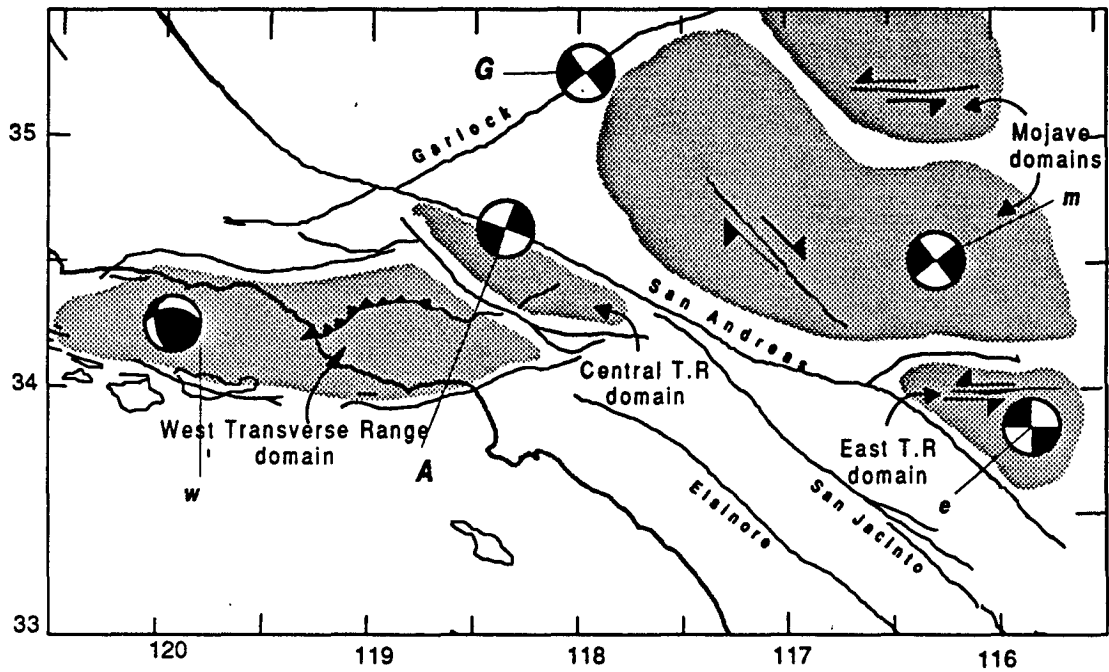
Fig. 2.3 Simplified map of domains and domain-bounding faults in southern California. Base map from Jennings (1975), domains adapted from Lyunedyk et al. (1980) (see also caption in Fig. 2.2).
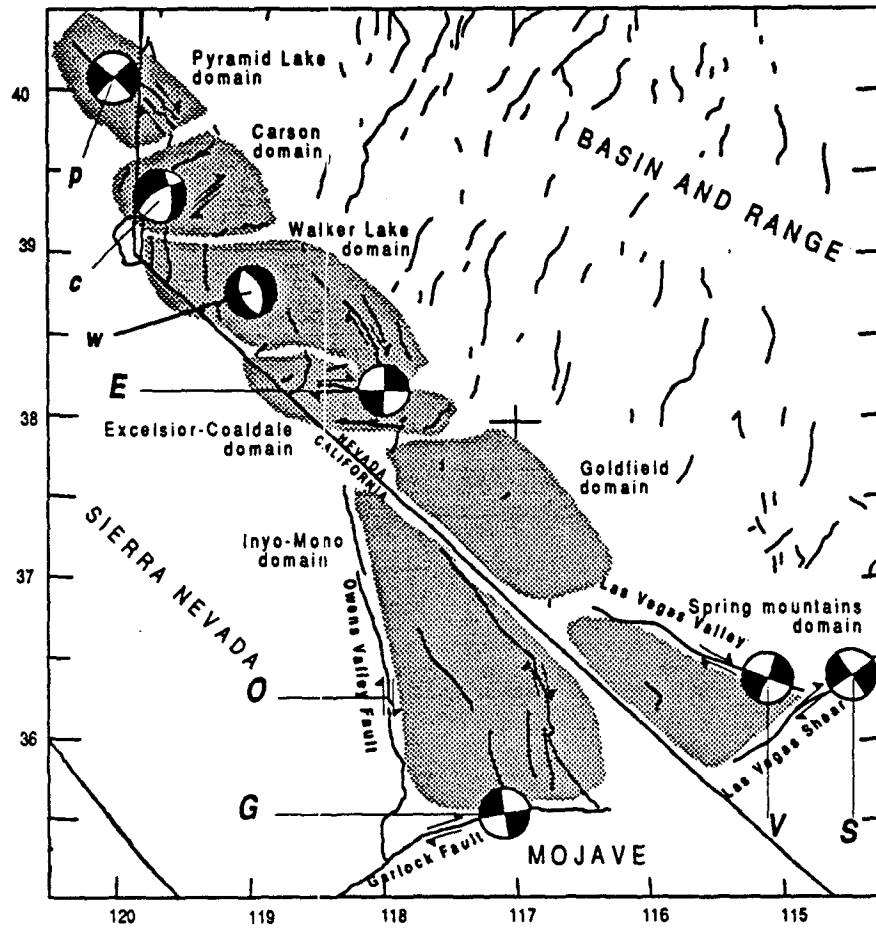
Fig. 2.4 Simplified map of domains and domain-bounding faults in the Walker Lane belt, western Great Basin. Base map and domains from Stewart (1988) (see also caption in Fig. 2.2).

where τ_0 and σ_n are the shear and normal stresses resolved on a fault about to form, μ_0 is the internal friction coefficient, and C_0 is the cohesion of intact crust.

Let us first consider an idealized geometry of faults (Figure 2.5): a north-south set of D-faults slipping in a normal sense, another northeast-southwest set slipping in a strike-slip sense and a third east-west set slipping in a reverse sense. Clearly, to satisfy slip directions along the three fault sets, the principal vertical stress must be intermediate, $S_v = \sigma_2$, with σ_3 parallel to the strike of the reverse fault set and σ_1 parallel to the strike of the normal fault set. Finally, in order to satisfy friction criteria ($0.3 < \mu_f < 0.7$) both the stress ratio $\phi = (\sigma_1 - \sigma_3)/(\sigma_2 - \sigma_3)$ and the σ_3/σ_1 ratio must be low in this example.

We use the 3D Mohr circle representation to visualize the results because friction criteria, stress magnitudes, and fault orientations can all be shown on a single plot. The geographical orientation of the three fault sets and the estimated stress directions are plotted in Figure 2.6a. The lower sloping line represents the criterion for sliding (Equation 2.1), the upper one the criterion for fracturing (Equation 2.2). The three sets of faults represented by their resolved stress state fall between the two lines. This indicates that all three fault sets can slide under the same stress field with a friction coefficient $\mu_f \simeq 0.6$. Our model is only concerned with finding a possible stress solution which is considered good when faults on a 3D Mohr circle plot close to the sliding line so that friction criteria are satisfied, the deviatoric stress, $(\sigma_1 - \sigma_3)/2$, does not exceed the fracture line, and estimated slip directions are consistent with observed ones. The best solution may be found by using a stress inversion technique as proposed by Reches (1987) program.

Next we apply our model to the three regions of distributed deformation (Figure 2.2–2.4). In each case, the vertical principal stress is inferred from the regional tectonics. Stress directions, the ϕ value and the σ_3/σ_1 ratio are found by a trial and error method so that slip directions on all fault sets are satisfied and are consistent

17

Normal + Strike-slip + Reverse ⇒ Strike-slip regime
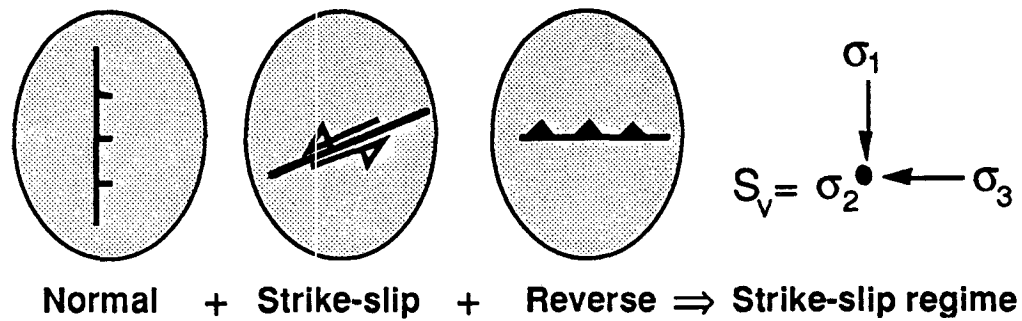
Fig. 2.5 Idealized geometry of faults for three domains. Because in our faulting model we assume that faults within domains are parallel, consider only one D-fault orientation for each domain: a north-south normal fault, a northeast-southwest strike-slip fault, and an east-west reverse fault. To satisfy slip along the three sets of faults, a strike-slip stress regime is necessary with $S_v = \sigma_2$, σ_3 parallel to the reverse fault set, and σ_1 parallel to the normal fault set.

3D Mohr circle   **Wulff projection**
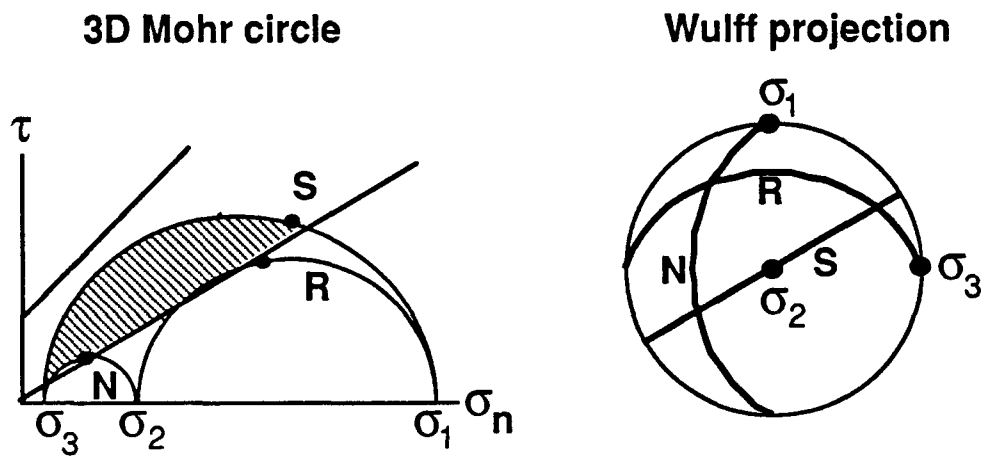
Fig. 2.6 Faults and stress field orientation shown in Fig. 2.5 are plotted here on the 3D Mohr circle and Wulff stereo projection. In order for slip to be consistent with friction criteria $(0.3 < \mu_f < 0.6)$ along all three fault sets, both the σ_3/σ_1 ratio and $\phi = (\sigma_2 - \sigma_3)/(\sigma_1 - \sigma_3)$ must be low. The lower line represents the sliding line, here shown with a slope of 0.6, and the upper line represents the fracture line, with a slope of 1.0 and an intercept on the $\tau - axis$, the cohesion.

with friction criteria.

2.4   Results for domain-faults (D-faults).

We estimate the deformation of each domain from significant earthquakes ($M_L \geq 4.5$), for which a focal mechanism solution is available, and from geological and geophysical constraints. Laboratory experiments on typical rocks (Byerlee, 1978) yield a friction coefficient between $0.5 < \mu_f < 0.7$. However, since crustal faults often contain gouge composed of alteration minerals the friction of well developed crustal faults may be lower $0.3 < \mu_f < 0.5$ (Byerlee, 1978). Earthquakes, fault data, and stress parameters used in Figure 2.7 are listed in Table 2.1.

The San Francisco Bay Area

Deformation in the San Francisco Bay Area is dominated by the strike-slip plate boundary between North America and the Pacific, we therefore assume a strike-slip stress regime with $S_v = \sigma_2$. By trial and error, we found that slip directions in the five domains shown in Figure 2.2 agree with a maximum horizontal principal stress direction of $N48^0E$ and a low ϕ ratio, a solution similar to that of Oppenheimer et al. (Model IV, 1988). In addition our model requires a low σ_3 if faults are to slip in accordance with friction criteria. Specifically, D-faults of this region fall close to the sliding line when $\phi = 0.1$ and $\sigma_3 = 0.3 \cdot \sigma_1$ (Figure 2.7-i, Table 2.1). This implies a friction coefficient for these faults of around $\mu_f = 0.5$.

The direction of σ_1 in this region is tightly constrained (Oppenheimer et al., 1988): the reverse D-faults here are sub-parallel to the San Andreas direction with an almost pure reverse slip. Consequently σ_1 has to be perpendicular to the strike of these faults. In fact, the average breakout orientations in the Bay area (Zoback et al., 1989) is $N50^0 \pm 11^0E$, in good agreement with our estimated σ_1 direction. It is possible, therefore, for all D-faults in this region to slip in a uniform regional

stress orientation, and yet obey normal friction criteria. Most of these D-faults are obviously not optimally oriented in this stress field.

Southern California

Based on the most extreme orientations of D-faults from the five domains shown in Figure 2.3, the best stress solution yields $S_v = \sigma_2$, σ_1 oriented $N20^0E$, and a low ϕ value (see also Scotti et al., 1991). Again σ_3 must be low for D-faults to be able to slip. Assuming $\phi = 0.1$ and $\sigma_3 = 0.1 \cdot \sigma_1$, the estimated best fitting friction coefficient for the D-faults in this region ranges between $0.3 \leq \mu_f \leq 0.4$ (Figure 2.7-ii, Table 2.1).

The orientation of σ_1, the ϕ value and magnitude of σ_3, are all tightly constrained by the extreme orientation and slip direction of D-faults in the West Transverse Ranges and Mojave Desert. This stress solution fits reasonably well the overall deformation pattern (Williams et al., 1990): dipping structures striking E-W show reverse slip, those striking N-S show normal slip, and vertical faults are strike-slip in nature. Consequently the D-faults in this region may obey the normal friction criteria, in a regionally fairly homogeneous stress field, in spite of their unfavorable orientation in this stress field.

Walker Lane

The Walker Lane belt is situated at the transition between the San Andreas strike-slip regime and the normal regime of the Basin and Range. The stress data here including focal mechanisms, fault slip, hydraulic fracturing, breakouts, and volcanic alignments (Vetter and Ryall, 1983; Zoback, 1989; Doser, 1989; Vetter, 1990), indicate some lateral variations in both principal stress orientations and stress magnitude.

According to Zoback (1989) the vertical stress here may be either intermediate or maximum with $\phi \approx 1$. Data for D-faults is only available in three of the seven domains. Our model results for this region (Figure 2.7-iii) yield $.4 \leq \mu_f \leq .6$, for

$S_v = \sigma_1$, $\sigma_3 = 0.3 \cdot \sigma_1$, and σ_2 oriented $N10^0E$. This estimated stress field orientation agrees with the E-W directed extension proposed by Zoback (1989).

2.5 Results for bounding faults (B-faults).

We consider now the stresses resolved on B-faults, domain bounding faults, using the stress field solutions derived in Figure 2.7-(i–iii).

For the B-faults of the Bay Area (Figure 2.7-iv) we find that the horizontal σ_1 direction is perpendicular to the strike of the Hayward (**H**) and Calaveras (**C**). Furthermore, in this stress solution, the 1989 Loma Prieta earthquake (**L**) should have left-lateral instead of the observed right-lateral slip (Romanowicz, 1990).

In southern California (Figure 2.7-v), the San Andreas fault (**A**) is also perpendicular to the computed horizontal σ_1, and therefore must slip with very low friction coefficients in this stress field. The San Jacinto (**J**) and the Garlock (**G**) faults, on the other hand, appear to slip with very high friction coefficients (Table 2.1).

In the Walker Lane region (Figure 2.7-vi), the Las Vegas Shear zone (**S**) and the Excelsior fault (**E**) require very low friction coefficients to be able slip.

Why are D-faults consistent with the assumption of a uniform regional stress field and B-faults are not? One possibility is that whereas the stress field is quite homogeneous on a large scale, it is modified along major fault zones. One cause for this inhomogeneity is the presence of gouge zones characteristic of major faults (Wang, 1984; Lachenbruch and Saas, 1973, 1991; Byerlee, 1990; Rice, 1990).

The soft, gouge-rich B-faults may therefore be zones of very low elastic moduli quite different from the surrounding rocks. This contrast in moduli can cause major perturbations of the local stress field direction and magnitude. In the following section we first discuss evidence for low elastic moduli of fault gouge material and then calculate the magnitude and nature of the stress field perturbation induced by the moduli contrast.
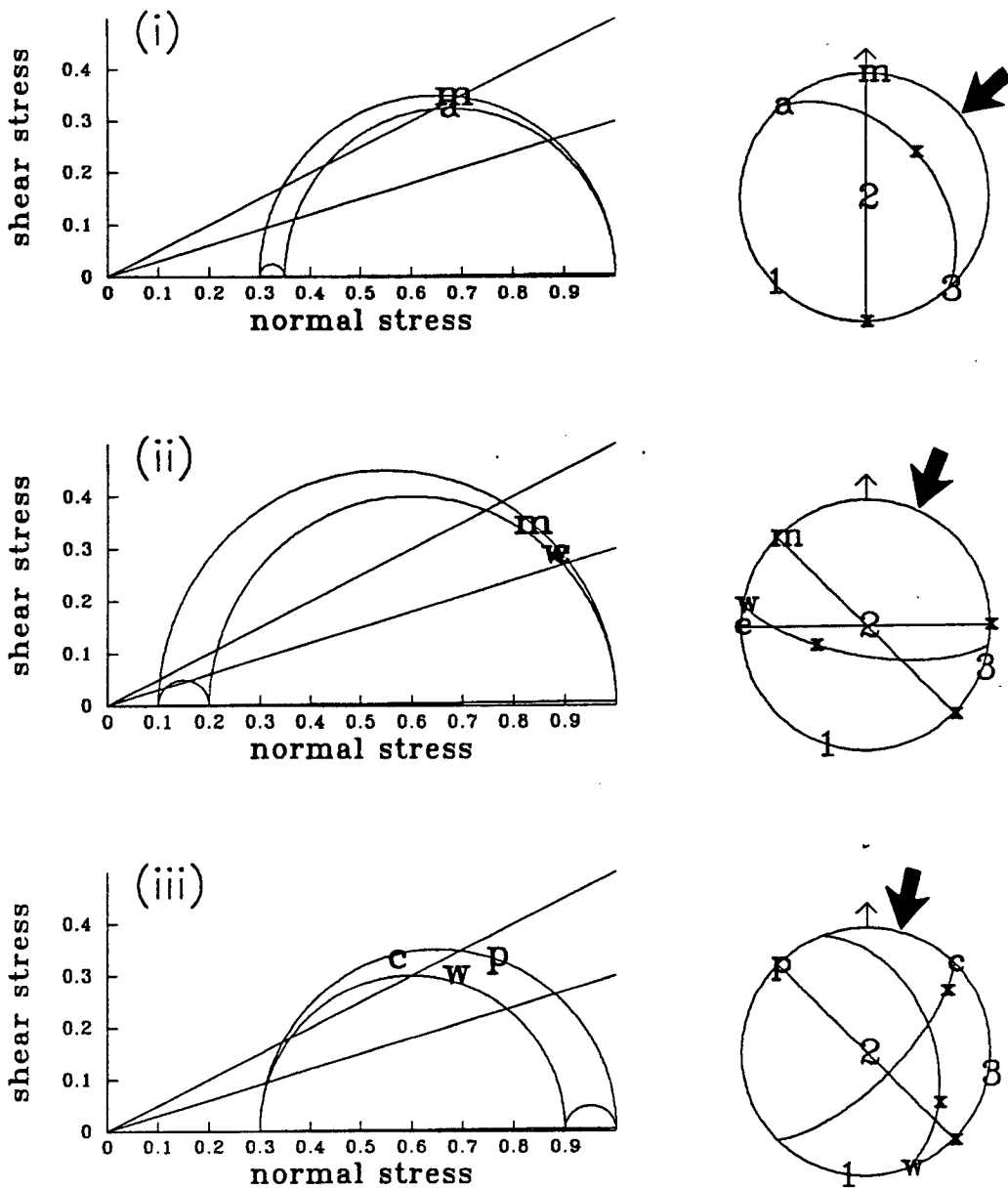
Fig. 2.7 Friction criteria applied to the three regions of distributed deformation: San Francisco Bay (i,iv), southern California (ii,v) and the Walker Lane belt (iii,vi). Results are shown on Mohr diagrams, normalized by σ_1, with two friction lines (slope of $\mu_f = 0.5, 0.3$). The letters (see Table 2.1 for fault identification and references) denote the estimated position of faults on the Mohr circle and in the Wulff stereonet projection. Estimated stress parameters are listed in Table 2.1 and discussed in the text. Domain faults can slip in accordance with friction criteria (i-ii-iii) but most domain boundary faults cannot (iv-v-vi). Arrows on Wulff projections indicate the measured direction of the maximum horizontal compressive stress (from Zoback et al., 1989).
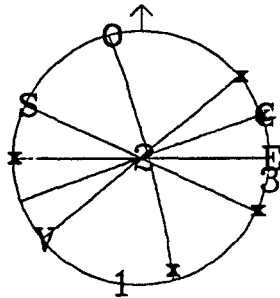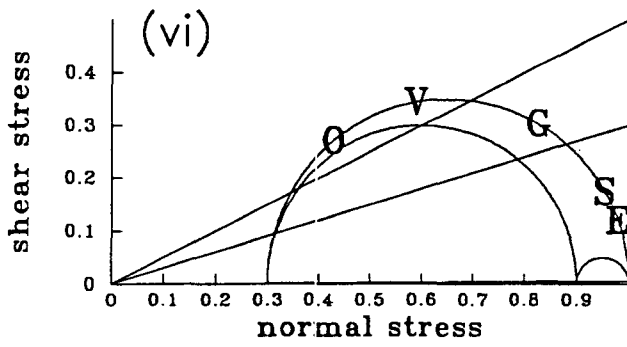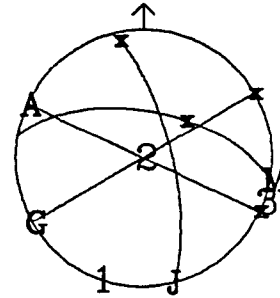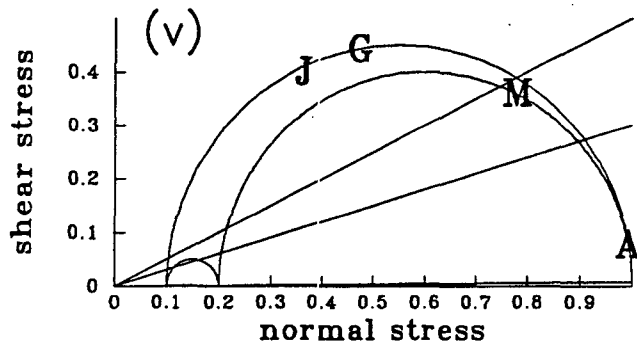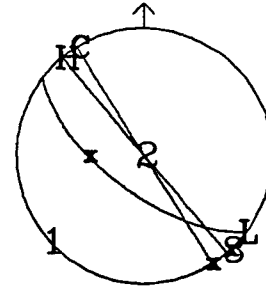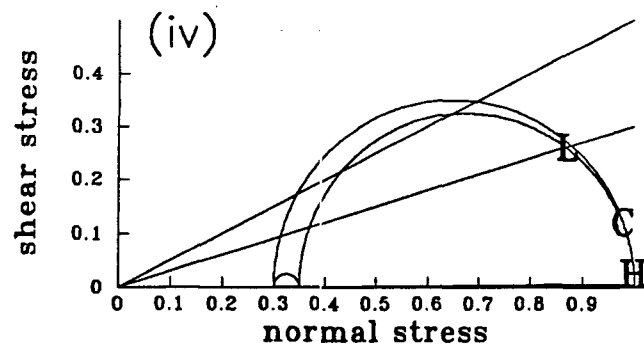
Fig. 2.7 Continued.

Fault	Dipdir	Dip	Rake	$\mu = \tau/\sigma_n$
Northern California				
$S_v = 0.35$; $S_{hmax} = 1.0$; $S_{hmin} = 0.3$; S_{hmax} $N48^0E$				
(m) Mt Lewis[a]	90	90	RL	0.52
(a) Anderson[a]	48	45	R	0.47
(H) Hayward[a]	50	90	RL	0.02
(C) Calaveras[a]	58	90	RL	0.12
(L) Loma Prieta[b]	217	66	RL+R	0.3
Southern California				
$S_v = 0.2$; $S_{hmax} = 1.0$; $S_{hmin} = 0.1$; S_{hmax} $N20^0E$				
(m) Mojave[c]	45	90	RL	0.4
(w) W.T.R.[d]	190	70	R	0.32
(e) E.T.R.[e]	180	90	LL	0.32
(M) S.Monica[f]	10	60	LL+R	0.46
(J) San Jacinto[g]	76	72	RL	1.1
(A) San Andreas	25	90	RL	0.07
(G) Garlock	150	90	LL	0.94
Walker Lane Belt				
$S_v = 1.0$; $S_{hmax} = 0.9$; $S_{hmin} = 0.3$; S_{hmax} $N10^0E$				
(w) Walker Lake[h]	70	45	N	0.43
(c) Carson[h]	135	70	LL	0.57
(p) Pyramid L.[h]	45	90	RL	0.42
(O) Owens V.[h]	75	85	RL+N	0.62
(E) Excelsior[h]	0	90	LL	.12
(G) Garlock[h]	160	90	LL	0.36
(V) L.Vegas S.[h]	140	90	LL	0.57
(S) L.Vegas V.[h]	25	90	RL	0.18

Table 2.1. List of fault parameters used to compute the stress solutions shown in Fig. 2.7. For each region estimated normalized magnitudes of the three principal stresses are listed, one principal stress is assumed vertical and the estimated direction of the maximum horizontal principal stress (S_{hmax}) is given. For each fault, the dip direction, the dip and the approximate known slip direction is listed and the estimated frictional strength $\mu = \tau/\sigma_n$ is given. Capital letters denote domain bounding faults (B-faults), small letters domain faults (D-faults). [(a) Oppenheimer et al., 1988; (b) Romanowicz, 1990; (c) Sauber et al., 1986; (d) Lee et al., 1979; (e) Williams et al., 1990; (f) Hauksson and Saldivar, 1986; (g) Given et al., 1987; (h) Stewart, 1988.]

2.6 Physical properties of B-fault zones

Mooney and Ginzburg (1986) presented an extensive review of the elastic properties within and adjacent to fault zones derived from laboratory, borehole, and seismic refraction and reflection data. The low V_p and V_s velocities often observed along fault zones can be attributed to high fracture density and to the presence of chemically or mechanically altered rocks. The observed decrease in V_s is usually greater than that in V_p, therefore the V_p/V_s ratio or equivalently, the Poisson's ratio, is greater within fault zones compared to the surrounding crust.

Evidence for high Poisson's ratio gouges comes also from the few published measured dynamic and static (Table 2.2) elastic moduli for gouge-like material for pre-consolidated clays at high pressures (Wang et al., 1980), and for dry shales (Jizba, 1991). The dynamic data is from a variety of rock types and fault systems. Figure 2.8 shows sloping lines representing constant values of the fault zone Poisson's ratio, ν_{in}, for different values of the shear moduli ratio, G_{in}/G_{out}, and the bulk moduli ratio, K_{in}/K_{out}, between the fault zone and the country rock. In this plot we have assumed an average country rock bulk modulus value of $K_{out} = 50\ MPa$, a country rock Poisson's ratio $\nu_{out} = 0.25$, and isotropic elastic properties so that:

$$\frac{G_{in}}{G_{out}} = \frac{K_{in}}{K_{out}} \cdot \frac{\frac{1-2\nu_{in}}{1+\nu_{in}}}{\frac{1-2\nu_{out}}{1+\nu_{out}}} \tag{2.3}$$
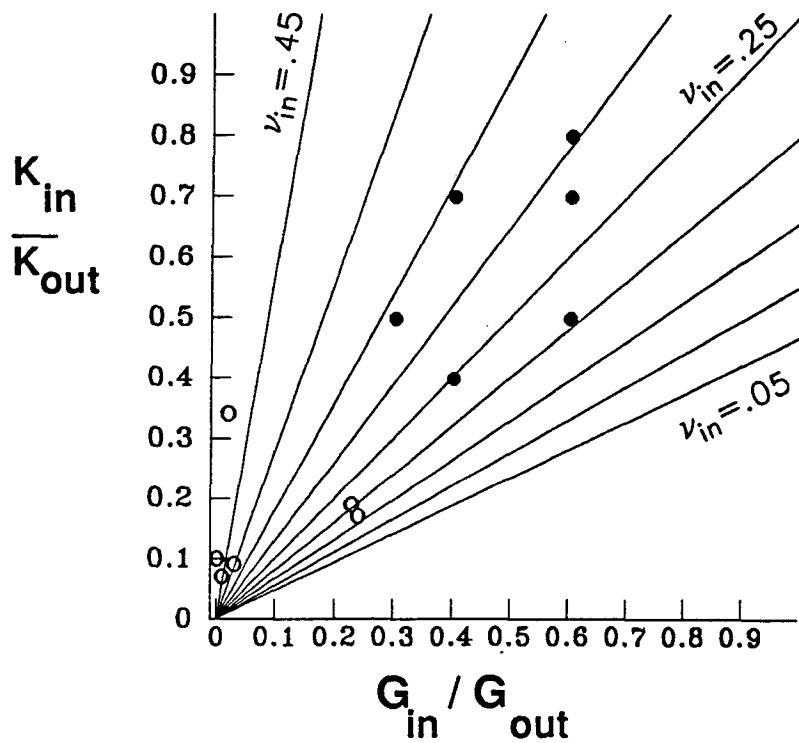
Fig. 2.8 Estimated moduli contrasts listed in Table 2.2 are plotted here on a G_{in}/G_{out} vs. K_{in}/K_{out} plot for a country rock Poisson's ratio of $\nu_{out} = 0.25$. G_{in} and G_{out} are the shear moduli of fault gouge and of country rock respectively. K_{in} and K_{out} are the bulk moduli of fault gouge and country rock respectively. The sloping lines represent constant values of the fault zone Poisson's ratio ν_{in}. The higher moduli ratio are from dynamic ($\bullet$) measurements, the lower ones are from static measurements ($\circ$). In general $\nu_{in} \gg \nu_{out}$.

Table 2.2a. Static moduli of clays.

$K_{in}(GPa)$ [a]	ν_{in} [b]	K_{in}/K_{out} [c]	Reference
3.6	.45	0.07	Wang et al. (Fig. 7,1980)
17	.48	0.34	Wang et al. (Fig. 4, 1980)
5.4	.48	0.1	Wang et al. (Fig. 4, 1980)
4.7	.42	0.09	Wang et al. (Fig. 6, 1980)
9.9	.22	0.19	Jizba (1991)
8.8	.18	0.17	Jizba (1991)

[a] σ_r/ϵ_v
[b] $\epsilon_\theta/\epsilon_z$
[c] Assuming $K_{out} = 50\ GPa$

Table 2.2b. Dynamic moduli of fault zones (in) and country rock (out).

$V_p(in,out)$ (km/s)	$V_s(in,out)$ (km/s)	$\rho(in,out)$ (gm/cc)	$K(in,out)^a$ (GPa)	$G(in,out)^b$ (GPa)	$G_{in/out}$	$K_{in/out}$	Reference
4.2,5.2	2.5,3	2.3, 2.7	21,40	14,24	0.6	0.5	Moos(1988)
4.5,6.5	2.7,4.0	(2.3,2.7) [c]	24,56	16,43	0.4	0.4	Healy,Peake(1975)
5.0,5.8	2.3,3.2	(2.3,2.7)	41,53	12,27	0.4	0.7	Moos(Kent cliff)
4.2,5.7	2.1,3.3	(2.3,2.7)	27,48	10,29	0.3	0.5	Lev(Cajon)
5.0,5.8	2.5,3.1	2.6,2.7	43,56	16,26	0.6	0.7	Lev(Table,unconfined)
5.2,6.0	2.8,3.5	2.6,2.7	43,53	20,33	0.6	0.8	Lev(Table,sonic)

[a] $\rho(V_p^2 - 4V_s^2/3)$
[b] $\rho(V_s^2)$
[c] Estimate

The static measurements indicate that fault zone material may have greatly reduced shear moduli, reduced bulk moduli and a Poisson's ratio that can be as high as $\nu_{in} = 0.48$. Dynamic measurements, on the other hand, show less reduction in moduli and fault material Poisson's ratio in the range of $0.35 \leq \nu_{in} \leq 0.2$. The scatter in moduli contrasts shown in Figure 2.8 is most likely due to a combination of our simplifying comparison of fault gouge to crystalline basement, the wide range of fault gouge properties (i.e. dry vs. saturated), and the well know difference between static and dynamic measurements. It is safe to assume, however, that the Poisson's ratio of fault zone material is greater than the country rock.

Next we use Eshelby's (1957) solution for a soft inclusion to model B-fault zones as elliptical regions of low moduli embedded in an otherwise homogeneous crust. Our aim is to understand the role of gouge zones in disturbing the local magnitude and local orientation of a remotely applied stress field.

2.7   B-faults as zones of low moduli.

Rudnicki (1979) showed how Eshelby's (1957) solution for the elastic field in an ellipsoidal inclusion can be used to determine the local rotation of principal stress axis caused by either slip on a fault or by the presence of a fault with different moduli than the surrounding. The basics of the solution are discussed in Appendix A. Eshelby's most valuable result is that stresses and strains inside an elliptical inclusion are homogeneous given homogeneous remote stress conditions.

Consider here a two dimensional, isotropic, cylindrical ellipsoid infinitely long in the X2 direction, parallel to the intermediate vertical principal stress, $S_v = \sigma_2$. In this geometry, the direction of σ_2 remains undisturbed and the modeled faults are vertical. Consequently the analysis can be restricted to the horizontal X1-X3 plane, which contains σ_1 and σ_3, the largest and least principal stress directions (Figure 2.9). The orientation of the ellipse is kept fixed, parallel to X1, whereas the orientation of

the remote σ_1 ranges from $\omega^\infty = 0^0$ (fault parallel compression) to $\omega^\infty = 90^0$ (fault normal compression).

2.7.1 Results

We discuss results that apply to soft zones that (1) have a small aspect ratio and (2) whose presence disturbs the stress field only in the immediate vicinity of the fault.

Thin fault zone.

The magnitude of the aspect ratio and the moduli contrast together determine whether a fault is effectively equivalent to a thick or a thin fault zone. As Figure 2.10 shows, $R = 1$ defiens a rough threshold between thick and thin fault zones:

$$R = \frac{\alpha}{K_{in}/K_{out}} \tag{2.4}$$

Figure 2.10a is a plot of $\Delta\omega = \omega^\infty - \omega^l$, the difference between the azimuth of the remote and local stress field, and Figure 2.10b is a plot of $\| \Delta\sigma \|$, the norm of the difference between the remote stress tensor σ_{ij}^∞, and the local one σ_{ij}^l, as a function of the distance L_n from the edge of the ellipse, where

$$\| \Delta\sigma \| = \frac{\sqrt{\sum_{i,j=1}^3 (\sigma_{ij}^\infty - \sigma_{ij}^l)^2}}{\sqrt{\sum_{i,j=1}^3 (\sigma_{ij}^\infty)^2}} \tag{2.5}$$

and

$$L_n = \frac{|X3|}{L1} \tag{2.6}$$

When $R \geq 1$, the fault zone is thick, in the sense that it is important to consider the stress field outside the fault zone, where ω^l is significantly different from ω^∞. When $R < 1$, $\omega^l \simeq \omega^\infty$, the fault zone is effectively thin. In this case the region of stress disturbance is localized to the fault zone itself. We shall assume an aspect ratio $\alpha = 0.01$ such that $R < 1$ ($\alpha < K_{in}/K_{out}$) for most of the range of interest (Figure 2.8).
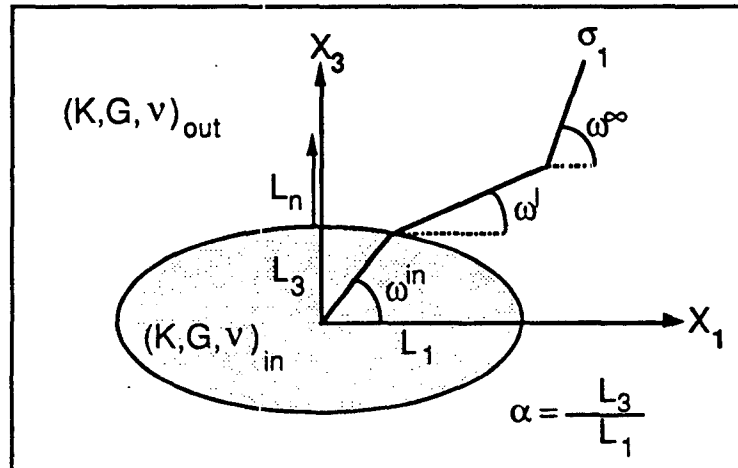
Fig. 2.9 An ellipsoidal inhomogeneity with principal half axes L1, L3, (L2= ∞ points out of the page); aspect ratio $\alpha = L3/L1$; maximum principal stress σ_1, oriented at ω^{in} (inside the ellipse), ω^l (locally outside), and ω^∞ (remotely); L_n is the normalized distance from the edge of the ellipse.
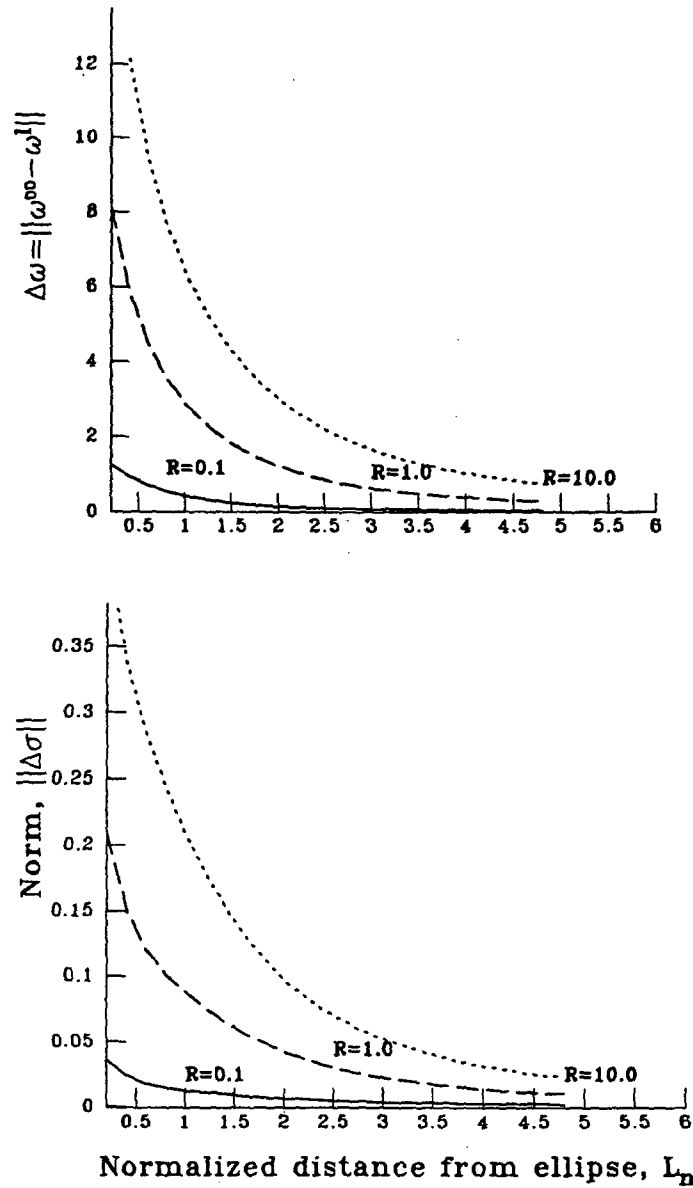
Fig. 2.10 (a) Plot of the norm $\| \Delta\sigma \|$, of the difference between remote and local stress components as defined in the text (Equation 2.4), as a function of L_n, the normal distance from the edge of the ellipse (Equation 2.5). (b) Plot of the angle $\| \Delta\omega \|$, the absolute difference between the orientation of ω locally and remotely. The three lines show that the fall-off with distance of the disturbed stress field depends on R, the ratio of the aspect ratio to the moduli ratio (here K_{in}/K_{out}). Plots are for $\sigma_3/\sigma_1 = 0.3$ and $\omega^\infty = 30^0$. As long as $R < 1$ the stress disturbance is limited to the fault zone, making it effectively a thin fault. For $R > 1$ the disturbance away from the fault zone can be quite extensive, making the fault zone effectively thick.

33

The stress ratio.

To apply the soft fault zone model we need to choose a range of σ_3/σ_1 stress ratios that is consistent with accepted friction values for the crust. In the three examples discussed in section 4 we found that the σ_3/σ_1 ratio that allows D-faults to slip was in the range of $0.1 \leq \sigma_3/\sigma_1 \leq 0.3$ (Table 2.1). In general, the lower limit may be $\sigma_3/\sigma_1 = 0$ and the upper limit is determined by the frictional shear resistance of the optimally oriented fault, which is a function of the friction coefficient, μ_f (Figure 2.11):

$$\frac{\sigma_3}{\sigma_1} = \frac{1 - \sin(\tan^{-1}\mu_f)}{1 + \sin(\tan^{-1}\mu_f)} \tag{2.7}$$

Therefore for $0.3 < \mu_f < 0.6$ the maximum stress ratio must range between $0.3 < \sigma_3/\sigma_1 < 0.55$.

2.7.2 Results for thin, soft fault zones.

In Figure 2.12 we compare the orientation and magnitude of the remote stress field to that inside an elliptical fault zone of aspect ratio $\alpha = 0.01$ and with elastic moduli lower than the surrounding. Note that when $\omega^\infty = 0^0$, the remote orientation of σ_1 is parallel to the strike of the fault, and when $\omega^\infty = 90^0$ it is perpendicular to it. For Poisson's ratio we take $\nu_{out} = 0.25$, and consider three values for the fault zone Poisson's ratio $\nu_{in} = 0.25, 0.35, 0.45$ and four values for the bulk moduli ratios between the fault zone and the country rock of $K_{in}/K_{out} = 0.01, 0.1, 0.4, 0.7$. Notice that as the bulk modulus of the gouge zone approaches that of water, the fault zone becomes effectively drained. For stress ratios we consider three cases $\sigma_3/\sigma_1 = 0.1, 0.3, 0.5$.

Stress field rotation.

Figure 2.12a shows $\Delta\omega$, the change in orientation of the stress field in the fault zone compared to the remote one. Negative $\Delta\omega$ implies a greater angle between the σ_1 direction and the strike of the fault inside the fault zone compared to the remote

34
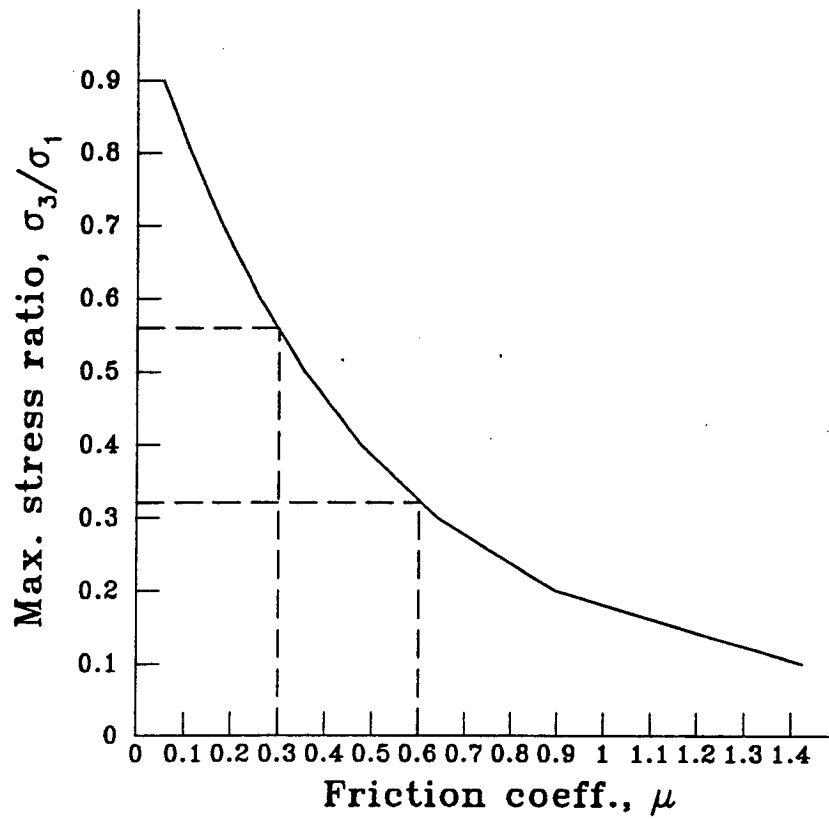
Fig. 2.11 Maximum σ_3/σ_1 required for faulting along optimally oriented faults (as defined in Fig. 2.1) as a function of the friction coefficient μ_f. If $0.3 \leq \mu_f \leq 0.6$ then $0.3 \leq (\sigma_3/\sigma_1)_{max} \leq 0.55$.

value. We notice that as the shear moduli ratio decreases from $G_{in}/G_{out} = 0.7$ to $G_{in}/G_{out} = 0.002$, the stress field rotation for a given ω^∞ increases, and this increase is generally greater for $\sigma_3/\sigma_1 = 0.5$ than $\sigma_3/\sigma_1 = 0.1$.

According to our soft fault zone model, the character of stress field rotation inside fault zones depends primarily on whether the Poisson's ratio is low or high.

(1) For $\nu_{in} = 0.25$ (left column in Figure 2.12a), the curves show always negative stress field rotations, so that σ_1 is always more fault perpendicular inside the fault zone than it is outside. The σ_3/σ_1 curves show a negative maximum in stress field rotation of $\Delta\omega = -10^0$, for $\omega^\infty = 20^0$ and $K_{in}/K_{out} = 0.7$, which increases to $\Delta\omega = -90^0$ for $\omega^\infty = 0^0$, as the bulk moduli ratio decrease to $K_{in}/K_{out} = 0.01$. This means that a 90^0 switch of principal stress directions in the fault zone relative to the remote stress can occur, for very low moduli when σ_1 is almost parallel to the strike of the fault zone.

(2) For $\nu_{in} = 0.45$ (right column in Figure 2.12a), the σ_3/σ_1 curves show a negative maximum in stress field rotation similar to the previous cases and in addition a positive maximum of $\Delta\omega = +20^0$ for $\omega^\infty = 80^0$ and $K_{in}/K_{out} = 0.7$. This implies that the σ_1 direction inside the soft fault zone is less fault perpendicular than the far field direction when the fault is under fault normal compression. Notice that, except for $K_{in}/K_{out} = 0.01$, when $\omega^\infty = 45^0$ there is no stress field rotation, whereas for $\omega^\infty < 45^0$ stress field rotation is negative and for $\omega^\infty > 45^0$ stress field rotation is positive.

Deviatoric stress magnitude perturbations.

Figure 2.12b shows the ratio between the deviatoric stress magnitude inside the fault zone and the remote value, T^{in}/T^∞, where $T = (\sigma_1 - \sigma_3)/2$. Notice that when $\Delta\omega = 90^0$ in Figure 2.12a, the ratio of T^{in}/T^∞ is almost 0 in Figure 2.12b, which implies almost hydrostatic conditions inside the fault zone. According to our soft

fault zone model, the magnitude of the maximum shear stress is disturbed to different extents depending primarely on the elastic moduli contrast and to a lesser extent on the σ_3/σ_1 ratio.

(1) For $K_{in}/K_{out} = 0.01$ (bottom row in Figure 2.12b) stress reduction, which occurs for all ω^∞ values, decreases monotonically as ω^∞ increases. Greater stress reductions occur as the stress ratio is decreased from $\sigma_3/\sigma_1 = 0.5$ to $\sigma_3/\sigma_1 = 0.1$.

(2) For $\nu_{in} = 0.25$ and $K_{in}/K_{out} = 0.7, 0.4, 0.1$ (left column in Figure 2.12b) the stress curves show stress reduction for low values of ω^∞ and stress amplification for large values of ω^∞. Notice how the ratio of T^{in}/T^∞ is very dependent on the value of the stress ratio σ_3/σ_1.

(3) For $\nu_{in} = 0.45$ (right column in Figure 2.12b) stress reduction is almost independent of the stress ratio, with a minimum in reduction at $\omega^\infty = 45^0$. This minimum ranges from $T^{in}/T^\infty = 0.9$ to 0.6 as the bulk moduli ratio decreases from $K_{in}/K_{out} = 0.7$ to 0.1. Stress reduction is $T^{in}/T^\infty = 0.05$ when the remote σ_1 is almost fault parallel and $T^{in}/T^\infty = 0.2$ when it is fault perpendicular.

Mean stress change.

Figure 2.12c shows the ratio between the mean stress in the fault zone and the remote value, $\sigma_0^{in}/\sigma_0^\infty$, where $\sigma_0 = (\sigma_1+\sigma_3)/2$. Notice that the inflexion point for each stress curve occurs at $\omega^\infty = 45^0$. As long as $\omega^\infty < 45^0$, the lower the stress ratio the more reduced is the mean stress inside the fault zone.

For $\omega^\infty > 45^0$, the soft fault zone model predicts different behaviors depending on both the elastic moduli and the stress ratio.

(1) For $K_{in}/K_{out} = 0.01$ (bottom row in Figure 2.12c), the greater the stress ratio the more reduced is the mean stress inside the fault zone.

(2) For $\nu_{in} = 0.25$ (left column in Figure 2.12c), the mean stress usually reduces inside the soft fault zone but for the lowest stress ratio ($\sigma_3/\sigma_1 = 0.1$) when $\omega^\infty > 67^0$.

(3) For $\nu_{in} = 0.45$ (right column in Figure 2.12c), the mean stress usually increases inside the soft fault zone.
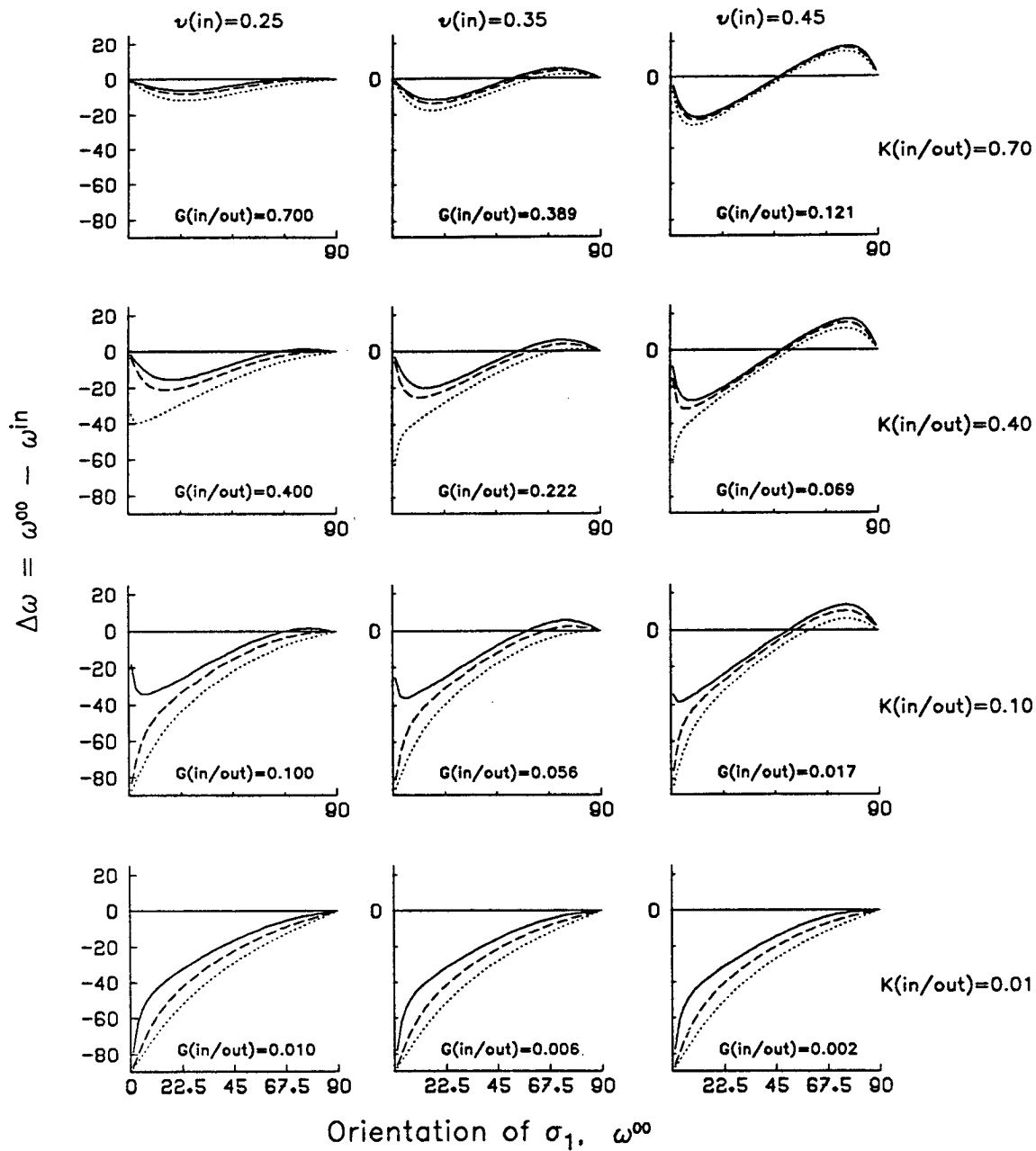
Fig. 2.12.a **Rotation of principal stress axes.** The difference $\Delta\omega = \omega^\infty - \omega^{in}$ between the orientation of σ_1 remotely and in the soft fault zone is plotted against ω^∞. The results are for Poisson's ratio $\nu_{in} = 0.25, 0.35, 0.45$, bulk moduli ratio, $K_{in}/K_{out} = 0.7, 0.4, 0.1, 0.01$ and an aspect ratio $\alpha = 0.01$. The corresponding value of the shear moduli ratio, G_{in}/G_{out} is shown for each case. We assume $\nu_{out} = 0.25$. For each plot we consider $\sigma_3/\sigma_1 = 0.1, 0.3, 0.5$, represented by the continous, dashed and dotted curves respectively. Negative values of $\Delta\omega$ imply a more fault perpendicular orientation of σ_1 within the fault zone compared to the remote orientation. See text for discussion.

39
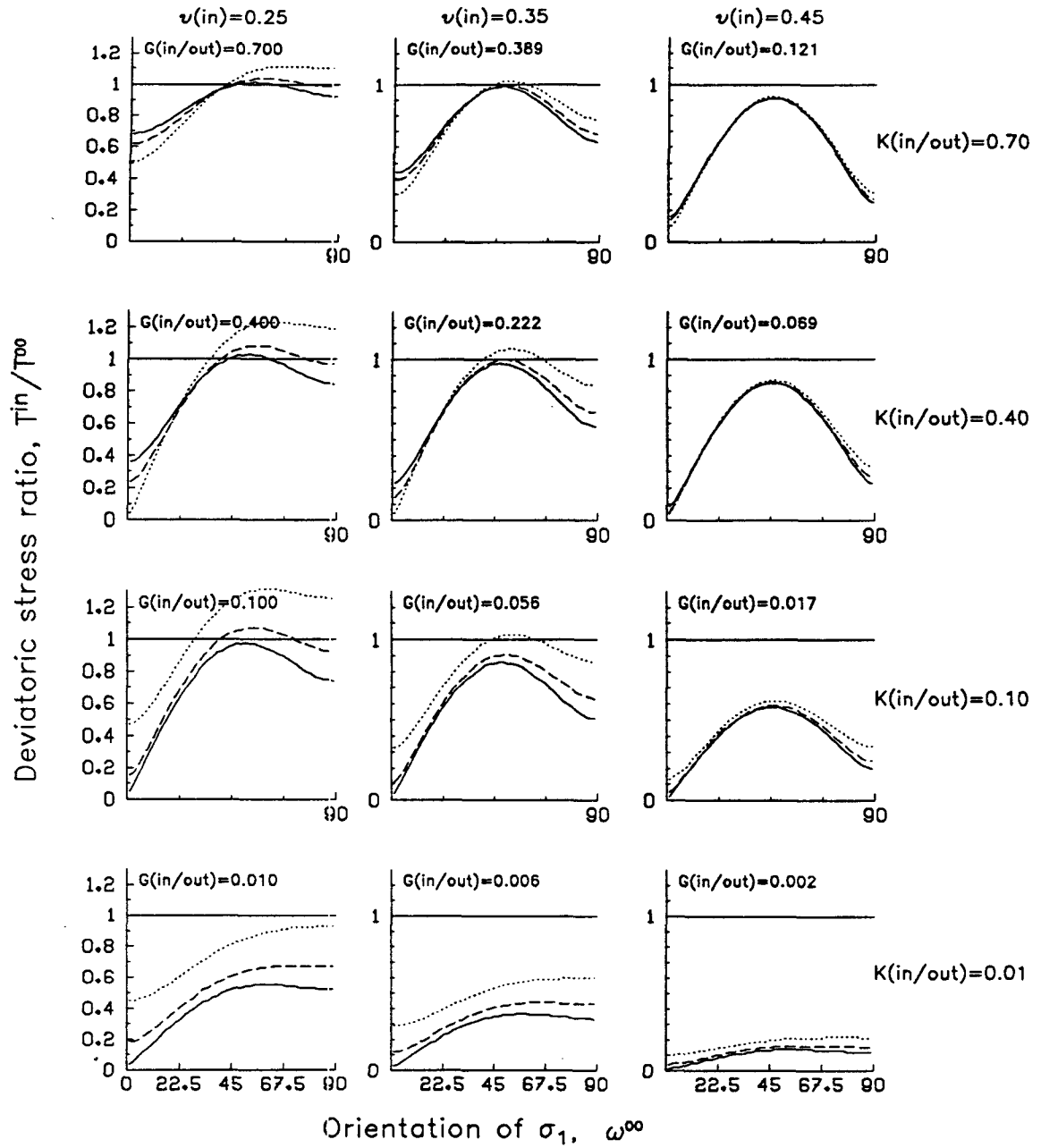
Fig. 2.12.b **Change in deviatoric stress magnitude.** The ratio T^{in}/T^∞ of the deviatoric stress $T = (\sigma_1 - \sigma_3)/2$ in the fault zone and remotely is plotted against ω^∞. When the curves exceed $T^{in}/T^\infty = 1$, shear stress magnitudes are amplified inside the fault zone. The results are for the same parameter values used in Fig. 2.12.a. See text for discussion.
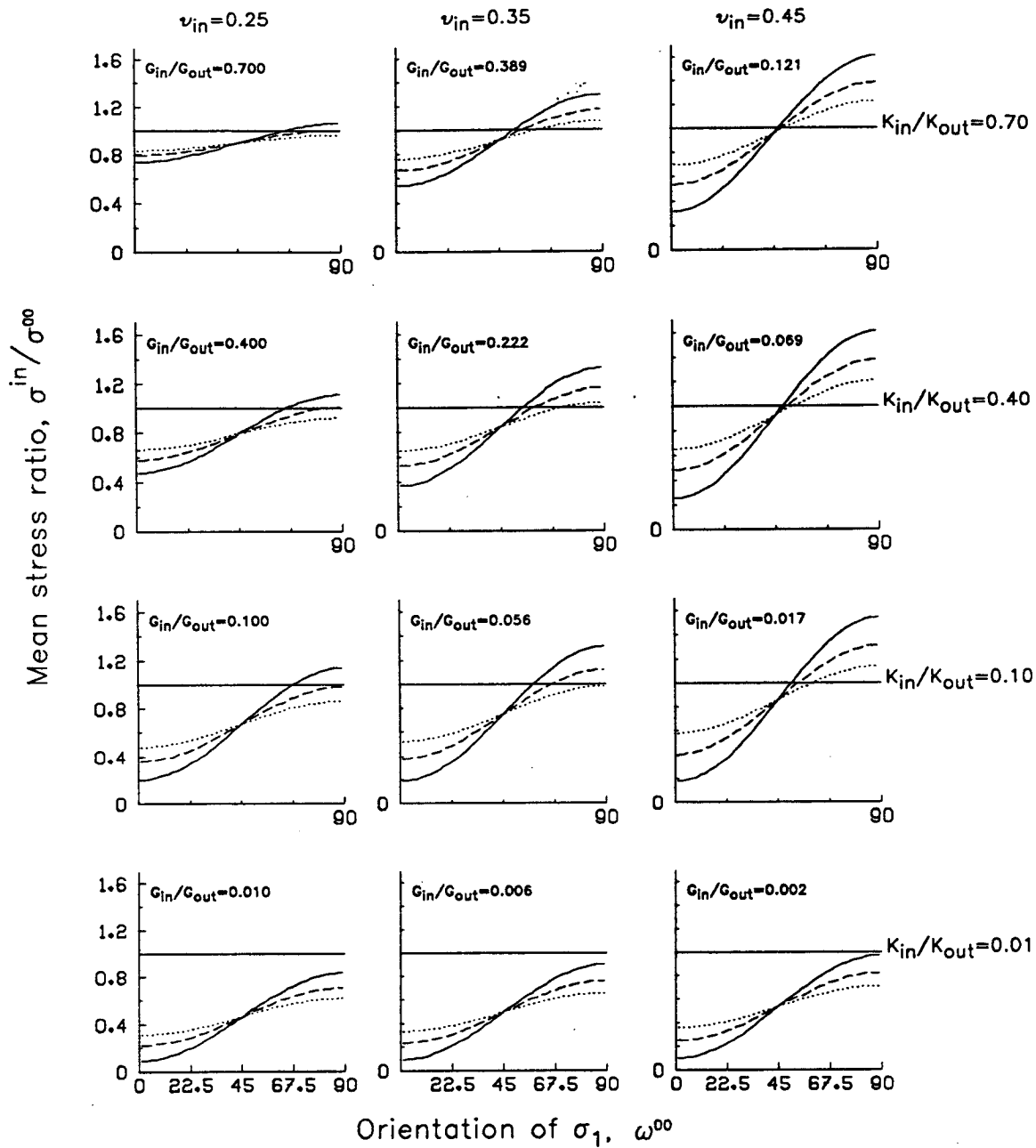
Fig. 2.12.c **Change in mean stress magnitude.** The ratio $\sigma_0^{in}/\sigma_0^\infty$ of the mean stress $\sigma_0 = (\sigma_1 + \sigma_3)/2$ in the fault zone and remotely is plotted against ω^∞. When the curves exceed $\sigma_0^{in}/\sigma_0^\infty = 1$, the mean stress is greater inside the fault zone compared to the remote value. The results are for the same parameter values used in Fig. 2.12.a. See text for discussion.

2.7.3 Soft fault zone model applied to B-faults.

Figure 2.13 shows the observed angles, ω^∞, between the strike of the analyzed B-faults and the regional orientation of σ_1, as estimated by our faulting model (Table 2.1). Estimated B-fault orientations range from $\omega^\infty = 30^0$ to 80^0. Assuming that all B-faults are elastically similar and the Poisson's ratio of fault zone material is high, the model predicts a typically local deviatoric stress reduction and either positive or negative rotation of the local stress field depending on the orientation of the remote σ_1 relative to the fault orientation, ω^∞. Figure 2.14 displays the change in stress field orientation and magnitude within four fault zones for $\nu_{in} = 0.45$ and $1. < K_{in}/K_{out} < 0.01$.

The greatest stress reduction, $T^{in}/T^\infty = 0.2$, is predicted for faults with $\omega^\infty = 80^0$. Lesser reduction, $T^{in}/T^\infty = 0.6$, is found for $\omega^\infty = 30^0$ or 60^0, and only little reduction, $T^{in}/T^\infty = 0.8$, for $\omega^\infty = 45^0$. The greatest stress field rotation $\Delta\omega = -20^0$ should be found for faults with $\omega^\infty = 30^0$, whereas for faults with $\omega^\infty = 45^0$ no stress field rotation is expected. For $\omega^\infty = 60^0$ to 80^0 rotations should be positive, $\Delta\omega = +10^0$. Notice how the mean stress inside the fault is higher than the regional mean stress when stress rotation inside the fault is positive and lower when stress rotation is negative.

2.7.4 Discussion

We have used Ehelby's solution (1957) to model B-faults as soft fault gouge zones assuming an elastic, isotropic, 2D, infinite, cylindrical geometry. We have quantified the resulting stress field disturbances in terms of the elastic properties of gouge zones, the remotely applied stress field and the orientation of the fault zone in that stress field. The results are of particular importance for the San Andreas fault zone (Figure 2.14d), because its orientation is almost perpendicular to the horizontal σ_1 direction: both the direction of the stress field and stress magnitudes may be significantly
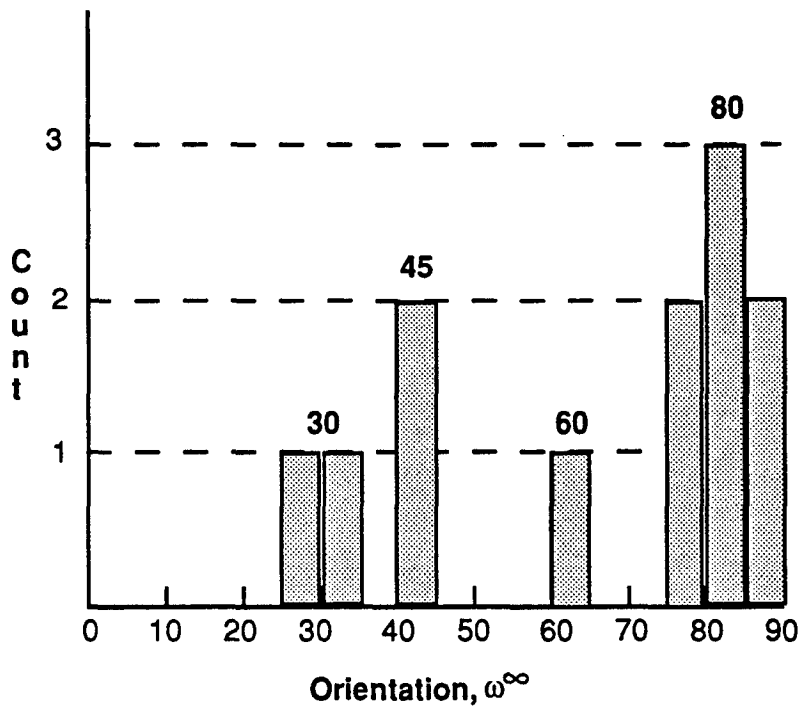
Fig. 2.13 Histogram showing the distribution of B-fault orientations ω^∞ as estimated by our faulting model (Table 2.1). The San Jacinto fault in southern California and the Owens Valley fault in the Walker Lane, fall in the $\omega^\infty = 30^0$ group. The Las Vegas shear zone and Santa Monica fault zone fall in the $\omega^\infty = 45^0$ group. The Garlock falls in the $\omega^\infty = 60^0$ group if subjected to the Southern California stress field and in the $\omega^\infty = 45^0$ group, if subjected to the the Walker Lane stress field. The San Andreas in southern California, the Excelsior and Las Vegas Valley in the Walker Lane, and all the B-faults in the Bay area, fall in the $\omega^\infty = 80^0$ group.
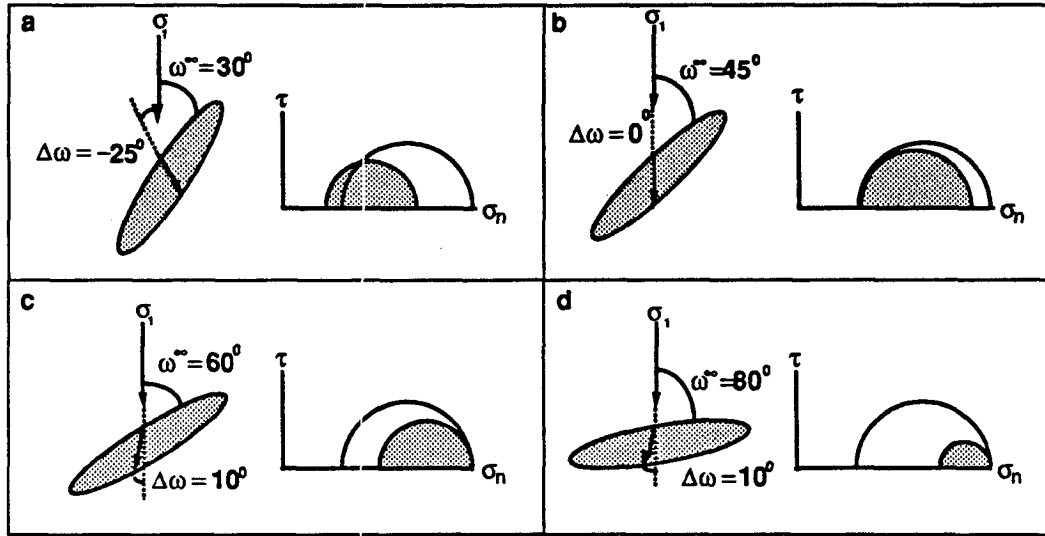
Fig. 2.14 Soft fault zone model results are shown here for four specific fault orientations and for $\nu_{in} = .45$ and $K_{in}/K_{out} = 0.1$. Stress reduction occurs in all cases, but stress rotation and the mean stress change sign across the $\omega^\infty = 45^0$ orientation so that for faults with $\omega^\infty > 45^0$ the stress field is more fault parallel and the mean stress is increased (c, d), whereas for faults with $\omega^\infty < 45^0$ the stress field is more fault perpendicular inside the fault zone and the mean stress is reduced compared to the remote values (a, b).

disturbed in the fault zone.

Based on the soft fault zone model we conclude first of all that the stress field in a soft fault zone, and in the region just outside, may not be a good indicator of the regional stress field. In fact the results suggest that deviatoric stresses in soft fault zones can be so low that earthquakes may completely relieve tectonic shear stresses there. Consequently earthquake slip may be very dependent on the local geometry of fault zones and lead to extreme variability of the local stress distribution. Interestingly, this variability is observed in the aftershock fault plane solutions of the Loma Prieta region (Michael et al., 1990; Oppenheimer, 1990) and along the southern California strand of the San Andreas fault (Jones, 1988). In both cases the fault zone is almost perpendicular to the computed regional orientation of σ_1.

The soft fault zone model considers only the most simple elastic case of a stress field that is locally perturbed due to the presence of low strength material. If relaxation of the stresses is very important along major fault zones, then deformation is better described by a visco-elastic model equivalent to a Poisson's ratio of 0.5 in our formulation. An important parameter that has not been considered in the soft fault zone model is the pore pressure effect. Several models have been recently proposed (Byerlee, 1990; Rice,1990; Lachenbruch and Saas, 1991) to explain how the San Andreas fault may slip in spite of its extremely unfavorable orientation in the stress field due to the presence of extremely high pore pressures. These high pore pressures can only be mantained along fault zones such as the San Andreas because they presumably have a well developed gouge zone. A more realistic model for B-fault should therefore combine both the effects of a low moduli gouge with those of high pore pressure.

2.8 Conclusions

In the first part of this chapter we have looked at faults in three regions of distributed deformation. We found that the geometry of faults is reasonably well described by a simple picture of domain faults (D-faults) separated by throughgoing domain-bounding faults (B-faults). Because the orientation of active D-faults in a given region varies between domains, it is not reasonable to assume that fault orientation in general is optimal in the regional stress field. The aim of our faulting model was to find a best fitting single stress field orientation which is consistent with frictional slip along all active faults of a given region. We first estimated principal stress directions based on the orientation of D-faults only and found that generally they are in agreement with in situ stress measurements. Most importantly, estimated friction coefficients for D-faults range between $0.3 \leq \mu \leq 0.6$, in accordance with laboratory derived values. Although our faulting model can explain the behavior of D-faults as consistent with friction criteria in a single stress field, these estimated stress solutions do not generally satisfy friction along most B-faults. We therefore conclude that there are two types of faults: long, throughgoing B-faults and short D-faults domains.

In the second part of this chapter, we have speculated that the difference between D-faults and B-faults stems primarily from the presence of soft gouge zones within many B-faults. We proposed a soft fault zone model with the aim of analyzing the stress disturbance that results from the presence of an elliptical fault zone with low elastic moduli. The model results depend on the orientation of the fault zone, the moduli contrast between the fault zone and the surrounding, and the value of the stress ratio σ_3/σ_1.

Laboratory and field data show that, in general, the Poisson's ratio within fault zones increases. We can therefore safely conclude that, within most fault zones, the local deviatoric stress should be reduced and the local stress field directions rotated. In the specific cases of high Poisson's ratio $\nu_{in} = .45$ and low bulk moduli

ratio $0.1 < K_{in}/K_{out} < .7$, the soft fault zone model results indicate that (1) stress reduction is a minimum when faults are oriented at $\omega^\infty = 45^0$, and reduction is increased as faults assume a more parallel or more perpendicular orientation relative to the horizontal σ_1, (2) there is no stress field rotation when $\omega^\infty = 45^0$, stress field rotation is negative when $\omega^\infty < 45^0$ and positive when $\omega^\infty > 45^0$, and finally (3) the mean stress is reduced or increased depending on whether the fault is oriented at $\omega^\infty < 45^0$ or $\omega^\infty > 45^0$ respectively. Therefore, for example, because the remote σ_1 for the San Andreas fault is at $\omega^\infty = 80^0$, considerable local deviatoric stress reduction is expected inside the fault zone, the mean stress is expected to increase, and the local direction of σ_1 is expected to be less fault perpendicular compared to the remote one.

In discussing the balance of forces in the brittle upper crust that contains a weak fault, Lachenbruch and Saas (1991) state that four fundamental assumptions of tectonophysics need re-evaluation: (a) one principal stress vertical, (b) hydrostatic fluid pressure, (c) optimally oriented active faults and (d) friction criteria as we know them from laboratory experiments. We believe that by combining the faulting and the soft fault zone models we can reach a better understanding of the tectonic forces and the rheologies that control deformation in regions of distributed deformation. Our model results can accept assumptions (a), (b) and (d) provided that (c) is totally relaxed, namely that active D-faults are not, in general, optimally oriented in the stress field. In addition B-faults must obey a different rheology. Therefore, within domains deviatoric stresses may be high so that D-faults are strong and slip is consistent with known friction values and known regional principal stress directions. On the other hand, the presence of soft gouge zones characteristic of B-faults disturbs the local stress field such that deviatoric stresses may be very low and slip must be controlled by the rheology of gouge material.

References

Anderson, E.M., *The dynamics of faulting.* Oliver and Boyd, Edinburgh, 206pp, 1951.

Atwater, T., Implications of Plate Tectonics for the Cenozoic tectonic evolution of westren North America. *Geol. Soc. Am. Bull., 81*, 3513-3536, 1970.

Aydin, A., and Page, B. M., Diverse Pliocene-Quaternary tectonics in a transform environment, San Francisco Bay region, California. *Geol. Soc. Am. Bull., 95*, 1303-1317, 1984.

Byerlee, J., Friction, overpressure and fault normal compression. *EOS Transactions, AGU, 71*, 43, 1492, 1990.

Byerlee, J., Friction of rocks. *Pageoph, 116*, 615-626, 1978.

Brace W. F. and Kohlstedt D. L., Limits on lithospheric stress imposed by laboratory experiments, *Jour. Geophy. Res., 85*, B11, 6248-6252, 1980.

Doser, D.I., Source parameters of earthquakes in the Nevada seismic zone, 1915-1943. *Jour. Geophy. Res., 93*, 15, 001-15, 015, 1988.

Eshelby, J. D., The determination of the elastic field of an ellipsoidal inclusion, and related problems., *Proc. Roy. Soc. London., Ser. A, 241*, 376-396, 1957.

Freund, R., Rotation of strike-slip faults in Sistan, Southeast Iran. *Journ. Geol., 78*, 188-200, 1970.

Given, D. D., Jones, L. M., and Hutton, L. K., The southern California network bulletin, July-December, 1986 *U. S. Geol. Surv. Open File Rep., 87-488*, 40pp. 1987

Handin, J., On the Coulomb-Mohr failure criterion. *Jour. Geophy. Res., 74*, 22, 1969.

Hauksson, E., and Saldivar, G. V., The 1930 Santa MOnica and the 1979 Malibu, California, earthquakes. *Bull. Seism. Soc. Am., 76*, 6, 1542-1559, 1986.

Healy, J.H. and Peake, L. G., Seismic velocity structure along a section of the San Andreas fault near Bear Valley, California. *Bull. Seis. Soc. Am., 65*, 1177-1197, 1975.

Jennings, C.W., Fault Map of California, *Calif. Div. of Mines and Geology*, 1975.

Jizba, D., Mechanical and acoustical properties of sandstones and shales, *Ph.D. Thesis*, Stanford univeristy, 1991.

Jones, L.M., Focal mechanisms and the state of stress on the San Andreas fault in Southern California. *Jour. Geophy. Res., 93*, B8, 8869-8891, 1988.

Kanamori, H., The state of stress in the earth's lithosphere. in *Physics of the earth's interior*, LXXVII Corso, Soc. Italiana di Fisica, Bologna, Italy, 1980.

Lachenbruch, A. H. and Saas, J. H., Heat flow from Cajon Pass, fault strength, and tectonic implications. *Jour. Geophy. Res.*, in press, 1991.

Lachenbruch, A. H. and Saas, J. H., Thermo-mechanical aspects of the San Andreas faults system, in *Proceedings of the conference on the tectonic problems of the San Andreas Fault system.* 192-205, Stanford University Press, Palo Alto, Calif., 1973.

Lee, W.H.K., Yerkes, R.F. and Simirenko, M., Recent earthquake activity and focal mechanisms in the Western Transverse Ranges, California. in *U.S.Geological Survey*, Map MF-1032, 1979.

Luyendyk, B.P., Kamerling, M.J. and Terres, R.R., Geometric model for Neogene crustal rotations in Southern California. *GSA Bull, 91*, 211-217, 1980.

Michael, A. J., Ellsworth, W. L., and Oppenheimer, D. H., Coseismic stress changes induced by the 1989 Loma Prieta, California earthquake. *Geophys. Res. Lett., 17*, 9, 1441-1444, 1990.

Michelini, A. and McEvilly, T.V., Seismological studies at Parkfield: I. Simultaneous inversion for velocity structure and hypocenters using cubic B-splines parametrization. *Bull. Seism. Soc.*, in press, 1991.

Moos, D., Personal communication., 1991.

Moos, D., The effects of mylonitization and fractures on elastic wave velocities in crystalline rock, example from the Cajon Pass scientific drillhole. *Geoph. Res. Lett., 15*, 9, 1053-1056, 1988.

Mooney, W. D., and Ginzburg, A., Seismic measurements of the internal properties of fault zones. *Pageoph., 124*, 1/2, 141-157, 1986.

Mura, T., *Micromechanics of defects in solids*, Martinus Nijhoff, Boston, 1987.

Nicholson, C., Seeber, L., Williams, P., and Sykes, L.R., Seismicity and fault kinematics through the Eastern Transverse Ranges, California: Block rotation, strike-slip faulting and shallow-angle thrusts. *Jour. Geophy. Res., 91*, 4891-4908, 1986a.

Nicholson, C., Seeber, L., Williams, P., and Sykes, L.R., Seismic deformation along

the San Andreas fault, California: Implications for conjugate slip and rotational block tectonics. *Tectonics, 5*, 629-648, 1986b.

Nur, A., Ron, H. and Scotti, O., Fault mechanics and the kinematics of block rotations. *Geology, 14*, 746-749, 1986.

Oppenheimer, D. H., Aftershock sip behavior of the Loma Prieta, California earthquake. *Geophys. Res. Lett., 17*, 8, 1199-1202, 1990.

Oppenheimer, D. H., Reasenberg, P. A., and Simpson, R. W., Fault plane solutions for the 1984 Morgan Hill, California, earthquake sequence: evidence for the state fo stress on the Calaveras fault. *Jour. Geophy. Res., 93*, B8, 9007-9026, 1988.

Reches, Z., Determination of the tectonic stress tensor from slip along faults with Coulomb yield condition. *Tectonics, 6*, 849-861, 1987.

Rice, J., Fault stress state, pore pressure distribution, and the weakness of the San Andreas fault. *EOS Transactions, AGU, 71*, 43, 1652, 1990.

Romanowicz, B., The Loma Prieta earthquake of October 18, 1989: teleseimic analysis using global digital data. *EOS Transactions, AGU, 71*, 17, 554, 1990.

Ron, H., Nur, A., and Aydin, A., Stress field rotation or block rotation: an example from the Lake Mead fault system, submitted to *Tectonics*, 1989.

Rudnicki, J. W., Rotation of principal stress axes caused by faulting. *Geoph. Res. Lett., 6*, 3, 135-138, 1979.

Sauber, J.W., Tatcher, W. and Solomon, S.C., Geodetic measurements of deformation in the central Mojave Desert, California. *J.Geophy.Res., 91*, 12.683-12.693, 1986.

Scotti, O., Nur, A., and Estevez, R., Distributed deformation and block rotation in 3D, in press, *Jour. Geophy. Res.*, 1991.

Shamir, G., Crustal stress orientation profile to a depth of 3.5 km near the San Andreas fault at Cajon Pass, California. *Ph.D. Thesis*, Stanford University, 1991.

Stewart, J., H., Tectonics of the Walker Lane belt, western Great Basin: Mesozoic and Cenozoic deformation in a zone of shear. in *Metamorphism and Crustal Evolution of the western United States*. Rubey, VII, edt. W. G., Ernst, 683-713, Prentice-Hall, Englewood Cliffs, N.J., 1988.

Vetter, U. R., and Ryall, A. S., Systematic change of focal mechanism with depth in the western Great Basin. *Jour. Geophy. Res., 88*, B10, 8237-8250, 1983.

Vetter, U. R., Variation of the regional stress tensor at the western Great Basin boundary from the inversion of earthquake focal mechanisms. *Tectonics, 9*, 1, 63-79, 1990.

Vernik, L., Personal communication, 1990.

Wang, C., Mao, N., and Wu F., Mechanical properties of clays at high pressure. *J.G.R, 85*, B3, 1462-1468, 1980.

Wang, C., On the constitution of the San Andreas fault zone in Central California.*Jour. Geophy. Res., 89*, B7, 5858-5866, 1984.

Webb, T. H., and Kanamori, H., Earthquake focal mechanisms in the Transverse Ranges and San Emigdio Mountains, southern California and evidence for a regional decollement. *Bull. Seism. Soc. Am., 75*, 735-757, 1985.

Wesnousky, S. G., Seismicity as a function of cumulative geologic offset: some observations from southern California. *Bull. Seism. Soc., 80*, 5, 1374-1381, 1990.

Williams, P. L., Sykes, L. R., Nicholson, C., and Seeber, L., Seismotectonics of the easternmost Transverse Ranges, California: relevance for seismic potential of the southern San Andreas fault. *Tectonics, 9*, 1, 185-204, 1990.

Ziony and Jones, Map showing late Quaternary faults and 1978-1984 seismicity of the Los Angels region, California. *U. S. Geol. Surv. Series*, Map MF-1964, 1989.

Zoback, M.D. and Healy, J.H., Friction, faulting, and << in situ >> stress, *Annales Geophysicae, 2*, 6, 689-698, 1984.

Zoback, M.D., Zoback, M.L., Mount, Van S., Suppe, J., Eaton J.P., Healy, J.H., Oppenheimer, D., Reasenberg, P., Jones, L., Raleigh, C.B., Wong, I.G., Scotti, O., and Wentworth, C., New evidence on the state of stress of the San Andreas fault system, *Science, 238*, 1105-1111, 1987.

Zoback, M.L., State of stress and modern deformation of the northern Basin and Range province. *Jour. Geophy. Res., 94*, B6, 7105-7128, 1989.

Zoback, M.L. and Zoback, M., State of stress in the conterminous United States. *Jour. Geophy. Res., 85*, B11, 611, 1980.

Zoback, M. L, Zoback, M. D., Adams, J., Assumpĉao, M., Bell, S., Bergman, E. A, Bl'umling, Brereton, N. R., Denham, D., Fing, J., Fuchs, K., Gay, N., Gregersen, S., Gupta, H. K., Gvishiani, A., Jacob, K., Klein, R., Knoll, P., Magee, M., Mercier, J. L., Müller, B. C., Paquin, C., Rajendran, K., Stephansson, O., Suarez,

G., Suter, M., Udias, A., Xu, Z. H., and Zhizhin, M., Global patterns of tectonic stress, *Nature, 341*, 291-298, 1989.

Chapter 3

Distributed deformation and block rotation in three dimensions

Abstract

The focus of this chapter is to understand distributed deformation, in particular the relationship between fault slip and rotation of faults and blocks in a three dimensional stress field. Regions of distributed deformation, such as Southern California, are organized in complex arrays of contemporaneously active block-faulted domains. We believe that the present day orientation of faults in many domains is due to the contemporaneous slip and rotation of the faults and of the blocks they bound. Traditional friction models cannot explain active unfavorably oriented faults and do not consider how faults become unfavorably oriented.

To solve this problem we propose a three dimensional block rotation model that tracks the orientation of blocks and their bounding faults during rotation. Mechanically, we consider Coulomb criteria for rock fracture as an upper bound, and slippage on a frictional surface as a lower bound. The key parameter in our model is the value of $\phi = (\sigma_2 - \sigma_3)/(\sigma_1 - \sigma_3)$, where $\sigma_1 > \sigma_2 > \sigma_3$ are the principal stress axis. When ϕ is low slip directions may change throughout rotation, whereas when ϕ is high they remain constant at their initial value. Principal stress directions are assumed irrotational through time. This model predicts up to 75^0 of vertical axis rotation along a

single set of faults. During rotation, fault slip may change, sometimes dramatically, giving rise to mixed vertical as well as horizontal axis of rotation of blocks and faults. For very unfavorably oriented faults the model predicts rotations about a vertical axis in both the normal and reverse stress regimes, and about a horizontal axis in the strike-slip stress regime. Therefore paleomagnetically inferred rotations may not always be directly related back to a specific stress regime.

Combining frictional constraints of the block rotation model with paleomagnetic, structural and geological data, we show how only one set of faults, preexisting and rotating in an irrotational strike-slip stress field, can account for the three major phases of deformation observed in the Western Transverse Range domain, Southern California: preexisting north-northeast faults were reactivated as normal faults, rotated and became strike-slip, and subsequent rotations of faults resulted in their present east-west high angle reverse orientation. This example demonstrates that it is not necessary to invoke complex regional and local changes in the stress regime or erratic changes in plate motion to account for alternate periods of compression and extension.

3.1 Crustal deformation by block rotation

3.1.1 The problem

Deformation at plate boundaries is commonly distributed across a wide zone characterized by domains of faults (Freund, 1970; Freund, 1971; Garfunkel, 1974; Luyendyk et al., 1980; Ron et al., 1984; and others). Through-going faults, such as the San Andreas and Garlock faults in Southern California (Figure 3.1), often mark domain boundaries. Within each domain faults form a set having similar orientations and fault slips. Adjacent fault sets may have different orientations and may have different fault slips as well. During the course of this chapter we will use terms such as rake, strike, and dip to refer to fault slip and orientation, as shown in Figure 3.2.

54
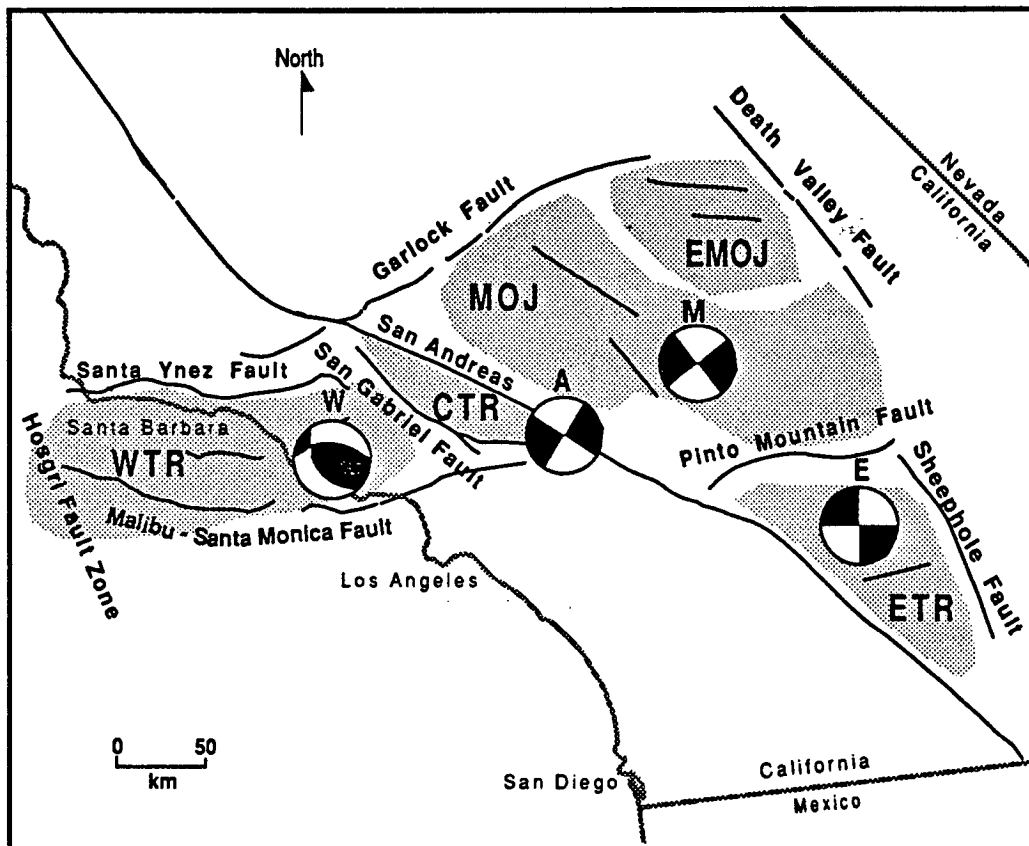
Fig. 3.1 Southern California domains. The shaded domains define subparallel sets of faults (Quaternary in age) of different orientations where rotation of block and faults has been inferred: Mojave (MOJ) and East Mojave (EMOJ), West (WTR), Central (CTR) and East Transverse Ranges (ETR). Domains are separated by through-going boundary faults such as the San Andreas, Santa Ynez, and Garlock faults (base map adapted from *Jennings* [1975]). Only the three most extreme fault plane solutions are shown (W, E, and M), and the San Andreas fault (A), one of the most prominent bounding faults in this region.
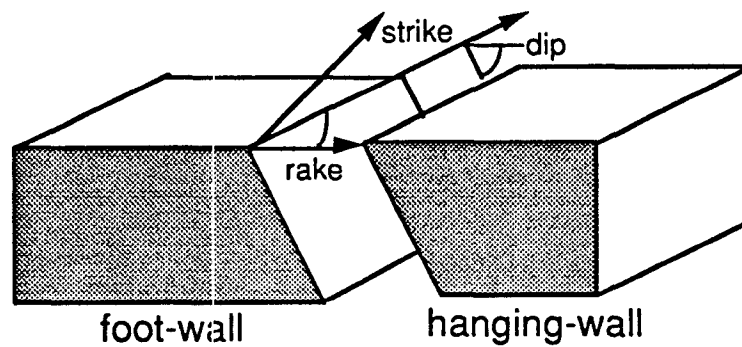
Fig. 3.2 The orientation of a fault set is given by the strike, or the azimuth of the fault with respect to a reference direction, and the dip, or the angle between the fault plane and the horizontal plane (0^0 = horizontal; 90^0 = vertical). Slip directions are determined by the rake, or the angle between the direction of fault slip and the horizontal. For plotting purposes, we define rake = $+90^0, 0^0, -90^0$ as pure normal, pure strike-slip, and pure reverse, respectively.

The immediate problem that arises is how can fault sets in such varying orientations and with such differing rakes accommodate deformation contemporaneously? At least three solutions can be suggested:

(1) The stress field changes in orientation and magnitude from one domain to the next.

(2) The stress field is irrotational and homogeneous. That is, principal stress directions and magnitudes are uniform, but in regions of distributed deformation fault sets behave in accordance with unknown friction criteria.

(3) The stress field is irrotational. Fault sets slip in accordance with friction criteria but in some domains blocks and faults rotate (and in others they don't). Only principal stress magnitudes vary from one domain to the next.

The answer probably lies in a combination of these three extreme cases. Fault behavior depends on slip rate and slip history; the earth's crust is not really homogeneous, thus the stress field must be rotational and inhomogeneous to some extent; rotation of blocks and faults must occur when sub-parallel fault sets move contemporaneously (Freund, 1970).

We restrict our attention to the third case: the rotation of faults and of the blocks between them. Accumulating paleomagnetic and structural evidence shows that in regions of distributed deformation, many structural domains have rotated in the past, and some are rotating today (see Nur et al., 1986). Different block rotation mechanisms have been proposed (see Molnar, 1988, and Kissel and Laj, 1988). Here, we propose a three dimensional block rotation mechanism based on the two dimensions model of Nur et al. (1986). For this reason, we first give a brief summary of the kinematics and mechanics of the two dimensional version.

3.1.2 Kinematics of block rotation in two dimensions

The kinematics of block rotation in strike-slip stress regimes were originally proposed by Freund (1970; 1974) on the basis of structural data alone. He observed two sets

of contemporaneous strike-slip faults at a high angle to each other. He attributed this angular spread to the tendency of strike-slip faults to rotate about a vertical axis away from the maximum compressional direction. In fact, rotation of faults as a mechanism for accommodating extension, was first recognized along normal faults (Ransome et al., 1910; see references in Jackson and White, 1989). Block rotation was also modeled kinematically in two dimensions by Garfunkel (1988) and Ron et al. (1984).

The kinematics of block and fault rotation are illustrated schematically in Figure 3.3. In this simple case, the vertical stress, $S_v = \sigma_1 = gravity$, is the only stress acting. As the books slide on the shelf they rotate away from σ_1, the direction of maximum compression. Therefore—whether strike-slip or dip-slip— right-lateral fault motion leads to counterclockwise sense of rotation of blocks and faults, and left-lateral movement leads to clockwise sense of rotation.

Block rotation models assume that the rotating blocks are rigid. Luyendyk et al. (1985), Ron et al. (1984), Carter et al. (1987) and Terres and Luyendyk (1985) have validated this assumption. In the domains they studied, two significant correlations were found. The first among the measured sense of slip, the expected sense of rotation and the paleomagnetically inferred axis of rotation (as explained in Figure 3.3); the second one among the known amount of total displacement across a fault set, the measured average spacing between sub-parallel faults, and the amount of rotation inferred paleomagnetically for a domain.

3.1.3   Mechanics of block rotation in two dimensions

In Freund's model (1970), faults and the blocks they bound slip and rotate away from the direction of maximum compression. Nur et al. (1986) added two mechanical constraints to this kinematic model for the case of vertical strike-slip faults rotating in a strike-slip stress regime. The first constraint is given by the Coulomb criterion for sliding, and the second one by the Coulomb criterion for fracturing (Equation 1 and
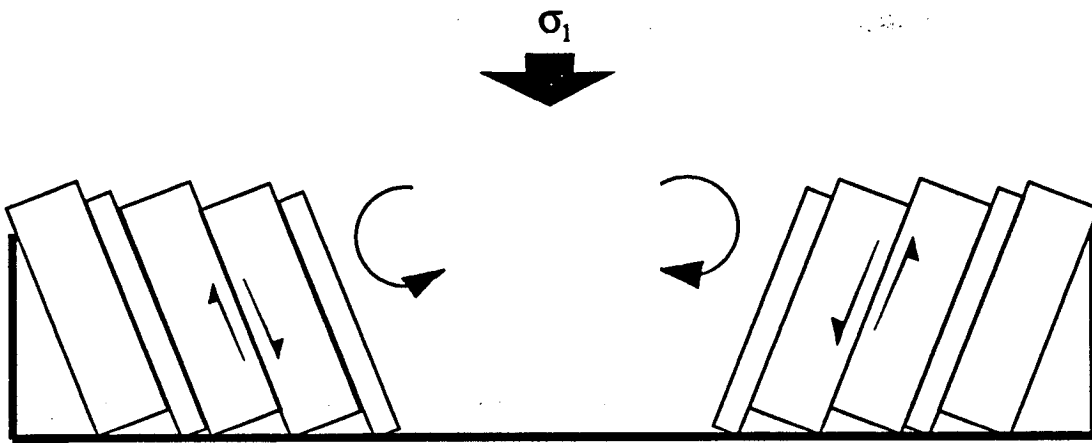
Fig. 3.3 Kinematics of block rotation. Blocks (books) of different orientations abut against boundary faults (shelf and book-holders). The orientation of the blocks with respect to the direction of maximum compression (in this case gravity) controls the sense of slip and therefore the sense of rotation of blocks.

2 in appendix B). In Nur's two dimensional block rotation (BR) model, faults are created in an optimal orientation and rotate away from σ_1 to less favorable orientations within an irrotational stress field. A fault is considered optimally oriented when the intermediate stress, σ_2, is contained in the plane of the fault and the shear stress required for slip along the fault is such that the $\sigma_1 - \sigma_3$ Mohr circle is just tangent to the sliding line (Figure 3.4).

In regions of distributed deformation, each domain is characterized by one predominant fault set orientation. Therefore, the first Coulomb constraint implies that fault sets may remain active even if unfavorably oriented in the stress field (if other, more favorably oriented faults existed in the domain they would slip instead), and the second Coulomb constraint sets a limit on how unfavorably oriented the fault set may become. For blocks and faults to continue to slip and rotate in an irrotational stress field, the magnitude of the maximum shear stress must change in a domain (Figure 3.5). When the magnitude of the maximum shear stress reaches the strength of the intact crust (the fracture line in Figure 3.5), a new set of more optimally oriented faults forms, and the old set becomes locked. Thus, a discrete range of fault orientations is predicted by the two dimensional BR model. Only when sufficiently large rotations occur in a domain will cross-cutting generations of fault sets give rise to a complex pattern of faulting, such as those observed in regions of distributed deformation (Angelier et al., 1985; Ron et al, 1990).
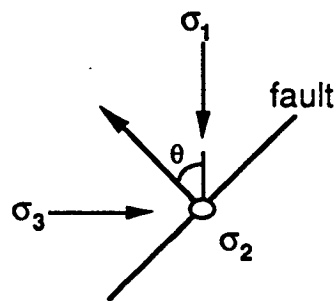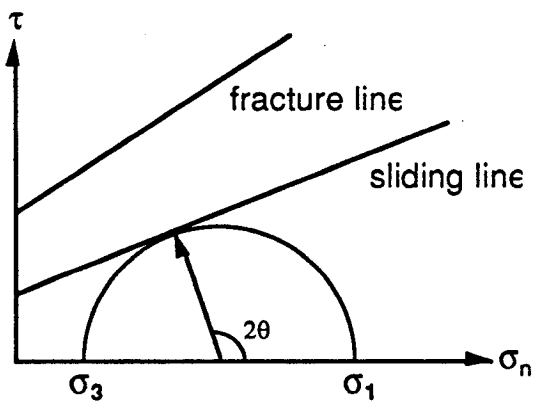
Fig. 3.4 Optimally oriented fault. According to traditional friction models [*Zoback and Healy*, 1984], active faults are generally optimally oriented in the stress field. Therefore, when plotted on the Mohr circle, optimally oriented faults should be represented by a $\sigma_1 - \sigma_3$ circle tangent to the sliding line, with the intermediate stress, σ_2, contained in the plane of the fault. The lower line is the Coulomb criterion for sliding along preexisting faults, and the upper one is the Coulomb criterion for fracturing an intact rock mass.
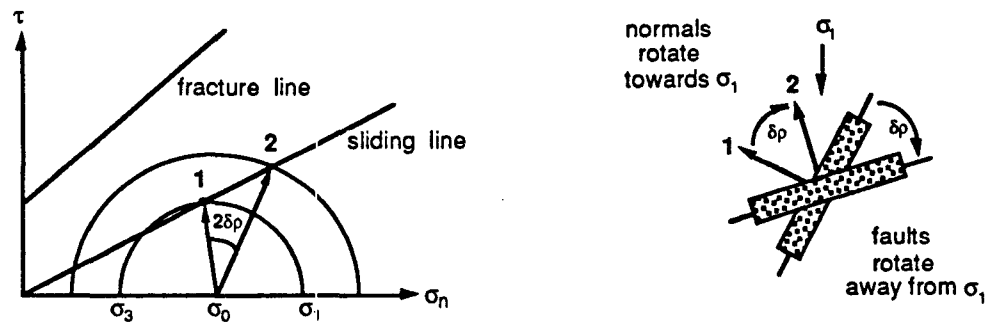
Fig. 3.5 Mechanics of block rotation. The sliding line is the locus of σ_n and τ values required for the fault to slip, before (point 1) and after rotation (point 2). As illustrated in the sketch to the right, faults rotate away from σ_1. Therefore, normals to faults will rotate towards σ_1 on a Mohr circle.

The mechanical constraints of Nur's two dimensional BR model provide an important step toward quantitative estimates of the contribution of block rotation to distributed deformation in the Earth's crust. However, the two dimensional model is limited in its application since σ_2 must remain in the fault plane. This limitation prevents any change in rake during rotation. A three dimensional formulation is necessary to model the more common case of rotation along reactivated faults, where σ_2 is not necessarily in the plane of the fault. Then, we can identify the conditions that allow a fault set to change its rake as it rotates. The fundamental rock fracture and fault friction criteria proposed in the two dimensional model are preserved. In addition, the three dimensional model considers any possible starting orientation of modeled faults.

3.2 The state of stress

3.2.1 One principal stress assumed vertical

Following Anderson (1951), the vertical stress, S_v, is assumed to be a principal stress. The magnitude of S_v determines the stress regime: normal if $S_v = \sigma_1$, reverse if $S_v = \sigma_3$, and strike-slip if $S_v = \sigma_2$ ($\sigma_1 > \sigma_2 > \sigma_3$ and compressive stresses are positive). As we will show in section 3.3, beyond Anderson's view, a given stress regime does not necessitate a specific rake, particularly in regions of distributed deformation.

3.2.2 Stress models

As shown in Figure 3.5 for the two dimensional case, stress magnitudes must change in a domain if we assume that faults and blocks progressively rotate within an irrotational stress field. We don't know how this change takes place, but we know it must if the friction criteria for sliding is to be satisfied. One assumption, used in Figure 3.5, is to keep the volume of the crust being deformed constant so that the mean normal

stress is constant

$$\frac{\sigma_1 + \sigma_2 + \sigma_3}{3} = constant$$

As it turns out, additional constraints (such as change in the stress regime) are required before applying this stress model.

Second and simplest assumption, used in the three dimensional BR model, is that the stress ratio ϕ remains unchanged

$$\phi = \frac{\sigma_2 - \sigma_3}{\sigma_1 - \sigma_3} = constant$$

The dimensionless stress ratio ϕ is often used to describe the three dimensional state of stress in the Earth's crust. It may vary from $\phi = 0$ when $\sigma_2 = \sigma_3$ to $\phi = 1$ when $\sigma_2 = \sigma_1$. There is no physical reason to choose this second assumption other than it does not require additional constraints.

3.2.3 Three dimensional fault geometry representations

Independently from the above assumptions, we need a way of tracking fault plane orientations during rotation in a three dimensional space. We use two representations: the Mohr circle, which plots the orientation of faults in a $\sigma_n - \tau$ plane, and the Wulff projection stereonet which plots the orientation of fault planes in the principal stress axis reference frame.

The three dimensional Mohr circle representation.

In two dimensional space, a single angle defines fault orientation. In three dimensions space, two angles are required to define the normal to a fault plane. The term pole is often used to refer to the normal to a fault plane in three dimensions. Figure 3.6a shows how the angles that define the direction of a pole in three dimensional space are represented in the Mohr diagram. For a more complete treatment of the three dimensions Mohr representation we refer the reader to Jaeger and Cook (Chp 2.6, 1969). Let us consider here only three extreme fault orientations:

64

(1) pole P_2 which falls exactly on the $\sigma_1 - \sigma_3$ circle: this is equivalent to the two dimensional representation in which σ_2 is in the plane of the fault.

(2) pole P_1 which falls exactly on the $\sigma_3 - \sigma_2$ circle: in this case σ_1 is in the plane of the fault and

(3) pole P_3 which falls exactly on the $\sigma_2 - \sigma_1$ circle: in this case σ_3 is in the plane of the fault.

When a pole (P) falls in the shaded region in Figure 3.6a, then all three principal stresses are off the plane of the fault. In this case α_1 and α_3, the angles that the pole makes with σ_1 and σ_3 respectively, are calculated by drawing two circles concentric with the two small $\sigma_1 - \sigma_2$ and $\sigma_2 - \sigma_3$ circles and passing through the pole. Then the intersection of the $\sigma_1 - \sigma_2$ concentric expansion with the $\sigma_2 - \sigma_3$ circle defines α_3, and the intersection of the $\sigma_2 - \sigma_3$ concentric expansion with the $\sigma_1 - \sigma_2$ circle defines α_1, as shown in Figure 3.6a. The Mohr circle representation is powerful because it represents stress magnitudes, friction criteria and the geometry of faults in the stress field, all in one graph.

The Wulff projection.

The same fault normals shown in Figure 3.6a are plotted on a Wulff lower hemisphere projection in Figure 3.6b: P_2 plots on the line joining the $\sigma_1 - \sigma_3$ directions, P_1 plots on the line joining the $\sigma_2 - \sigma_3$ directions and P_3 plots on the line joining the $\sigma_1 - \sigma_2$ directions. Depending on the stress regime, either σ_1, σ_2, or σ_3 is in the down orientation throughout the chapter. This type of representation is often used by structural geologists because it makes it easier to visualize the geometry of faults in the stress field.
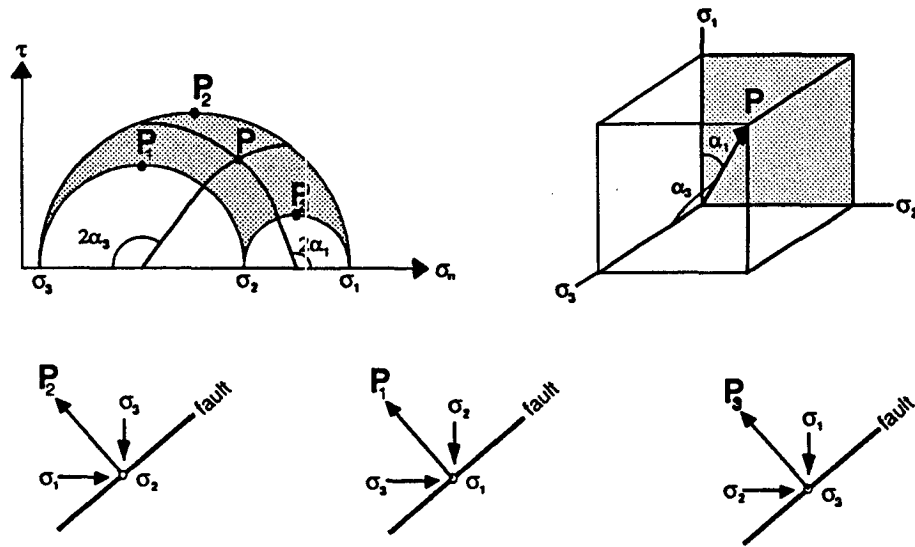
Fig. 3.6.a The 3D Mohr circle is a three-dimensional construction on a two-dimensional $\sigma_n - \tau$ plane. Any point in the shaded region represents the orientation of a pole (fault plane normal) in space. For example, α_1 and α_3 are the direction cosines of a pole (P) relative to the σ_1 and σ_3 directions, respectively. Three specific fault orientations are shown: pole (P_2) plotting on the $\sigma_1 - \sigma_3$ circle with the σ_2 direction contained in its fault plane, identical to the 2D case; pole (P_1) plotting on the $\sigma_2 - \sigma_3$ circle, with the σ_1 direction in its fault plane, and pole (P_3) plotting on the $\sigma_1 - \sigma_2$ circle, with the σ_3 direction in the fault plane.
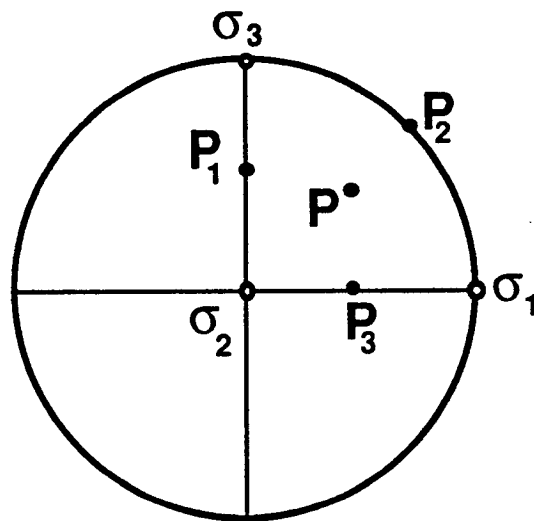
Fig. 3.6.b The lower hemisphere Wulff stereo projection is another way of plotting the orientation of poles in 3D space. Poles P, P_1, P_2, and P_3 are the same as in Figure 6a.

3.2.4 Choice of material parameters

Finally, we need to assume some values for the mechanical constraints of the BR model. The relationship between material parameters and block rotation is discussed in detail in Nur et al. (1986). As we will see for the case of the Western Transverse Range (section 3.6), the specific values do not affect the qualitative aspect of the results. The more important result is the change in rake as faults rotate. In the model, we assume the following values for the material parameters (Handin, 1969): 1.0 for the coefficient of friction of intact rock and 0.6 for the coefficient of friction of preexisting fault sets; 1000 bars for the cohesion of intact rock and 50 bars for the cohesion of preexisting fault sets. Given these values, the stress limits (Coulomb criteria) employed in the model become:

(1) The upper limit, representing the strength of the crust (the fracture line)

$$\tau_0 = 1000 + 1.0\sigma_n \quad (in\ bars)$$

(2) The lower limit, representing the strength of preexisting faults (the sliding line)

$$\tau_f = 50 + 0.6\sigma_n \quad (in\ bars)$$

(3) The tensile limit, assuming the crust cannot withstand any tensile stress

$$\sigma_3 \geq 0 \quad (\sigma_3 < \sigma_2 < \sigma_1)$$

The relative magnitude of these values controls the limits of fault and block rotation, their absolute values are not critical.

3.3 Three dimensional stress field: the key to distributed deformation

In a three-dimensional situation, the value of the intermediate stress, plays a key role in determining rakes of faults. This contrasts with the two dimensional geometry where the rake is constant because σ_2 is always in the plane of the fault, a condition

which restricts the fault normal to always fall on the $\sigma_1 - \sigma_3$ circle in a Mohr's diagram (Bott, 1959; McKenzie, 1969; and others). Here we include σ_2 specifically to allow us to understand how active faults become unfavorably oriented during three dimensional block rotation. Before considering block and fault rotation, let us first utilize the Mohr diagram to see which fault orientations are favorable for reactivation—and what this means for regionally distributed deformation in Southern California.

3.3.1 Friction criteria and the 3D Mohr diagram

Clearly, reactivation of preexisting faults is constrained by the strength of the crust, the strength of active faults, their orientation in the stress field and the relative magnitudes of the three principal stresses. Jaeger and Rosengren (1969) discuss the influence of these parameters in more detail. Let us consider material parameter values as discussed in the previous section and a maximum shear stress magnitude exceeding the sliding line but not the fracture line, as shown in Figure 3.7.

Only three sets of preexisting faults are considered: (a) set 1 which contains the σ_1 direction, (b) set 2 which contains the σ_2 direction and (c) set 3 which contains the σ_3 direction. The rakes expected along each fault set depend on the stress regime and the fault set orientation. Faults can slip when their poles plot within the shaded region of the Mohr diagram. The size of the shaded region depends on the magnitude of the maximum shear stress and on the ϕ value. Assuming a sufficiently high maximum shear stress, at ϕ values less than 0.4, all three fault sets will be re-activated. In the normal stress regime, set 1 will be reactivated as strike-slip faults, while set 2 and set 3 as normal faults. In the reverse stress regime set 1 and set 2 will be reactivated as reverse faults and set 3 as strike-slip faults. In the strike-slip stress regime, set 1 will be reactivated as normal faults, set 2 as strike-slip faults and set 3 as reverse faults. At ϕ values exceeding 0.4, fault set 3, which contains σ_3 in the fault plane, will be locked, while the other two fault sets can be reactivated.

These fault orientations produce the most extreme rakes that can be expected in the three stress regimes. Most domains are characterized by obliquely slipping fault sets. Of course, oblique slip would occur if poles plotted within the shaded region of the Mohr circle. The rake will be a function of ϕ and of the stress regime. A summary table for the limiting cases is provided in Table 3.1.

3.3.2 Southern California domains: example of distributed deformation

Southern California is characterized by domains of faults separated by through-going boundary faults (Figure 3.1). The shaded regions in Figure 3.1 represent domains where block rotation has been inferred (Luyendyk et al., 1985) and therefore where we expect faults to be unfavorably oriented in the stress field. Rakes range from high-angle reverse-oblique along east-west trending faults in the Western Transverse Range (WTR) domain (Lee et al., 1979; Yerkes and Lee, 1979a,b), to right-lateral strike-slip along northwest-southeast trending vertical faults in the Mojave (MOJ) domain (Sauber et al., 1986), and left-lateral strike-slip along E-W trending vertical faults in the East Transverse Range (ETR) domain (Jones, 1988; Powell, 1982). Fault orientations and rakes in the remaining domains are not considered here because they fall in between these extreme cases. The question we would like to answer is: can this range of rakes and fault orientations be the result of a regionally uniform and timewise irrotational stress field—in accordance with friction criteria? We will assume that the regional stress field is strike-slip because the San Andreas strike-slip system dominates the region's tectonics.

As we have seen in Table 3.1, reverse and strike-slip faulting may coexist within a strike-slip stress regime only if ϕ is low and the maximum shear stress is sufficiently high. In addition we use a forward method of trial and error to estimate a direction of σ_1 that allows faults to slip with a reasonable coefficient of friction, μ_f. For the case of Southern California domains, the σ_1 direction is tightly constrained by the
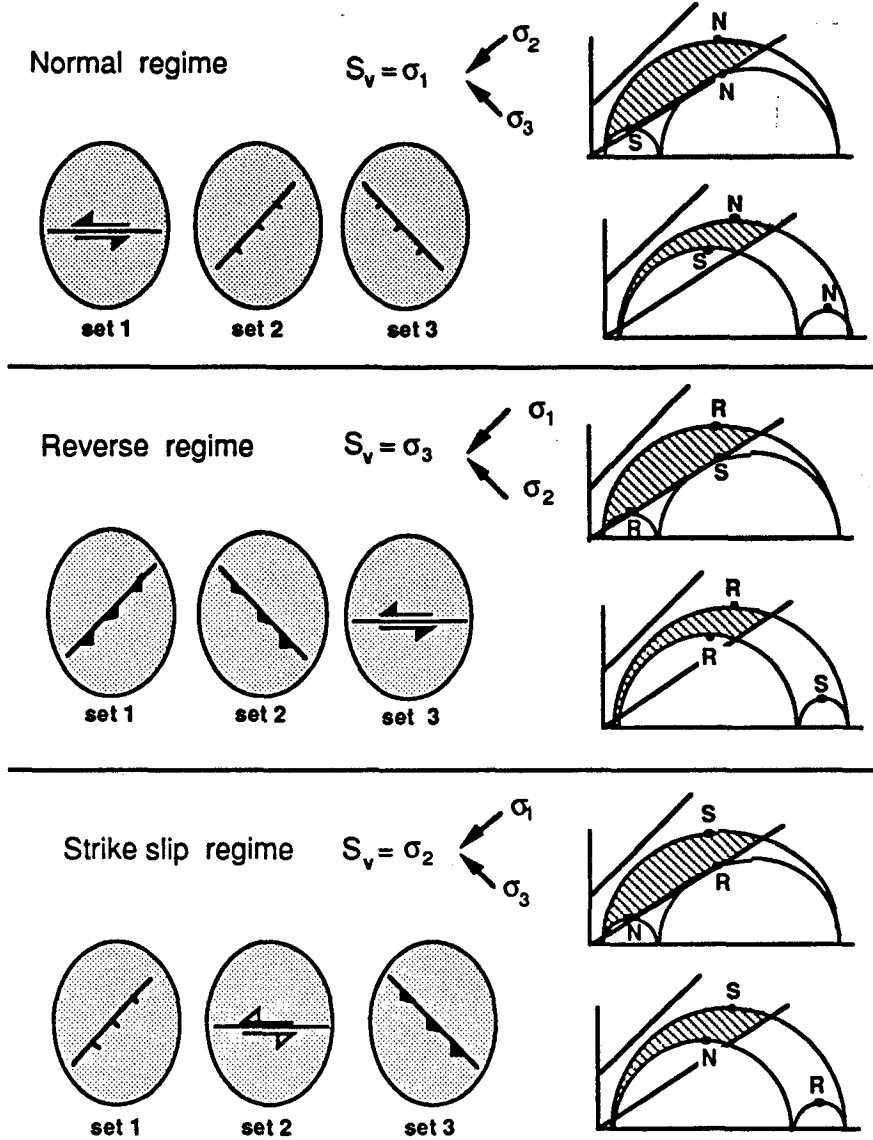
Fig. 3.7 Mohr circle, friction criteria, and rakes. Different rakes may be observed (N = pure normal, R = pure reverse, S = pure strike-slip) in a region of distributed deformation. Rakes depend on the orientation of fault sets, the magnitude of the vertical stress, S_v, the value of ϕ, and material parameter values. Here we consider three sets of faults in map view, in the three stress regimes, for the case of low and high ϕ values (assuming 1.0 and 0.6 for the fracture and sliding coefficients, respectively): set 2, which plots on the $\sigma_1 - \sigma_3$ Mohr circle because it contains the σ_2 direction in its plane (the 2D case), set 1, which plots on the $\sigma_2 - \sigma_3$ circle because it contains the σ_1 direction in its plane, and set 3, which plots on the $\sigma_1 - \sigma_2$ circle because it contains the σ_3 direction in its plane. See also Table 1 for explanations.

71

Stress Ratio	Stress regime			Faults
	normal $S_v = \sigma_1$	*reverse* $S_v = \sigma_3$	*strike-slip* $S_v = \sigma_2$	
low $0. < \phi < 0.4$	normal	reverse	strike-slip	set 2
	strike-slip	reverse	normal	set 1
	normal	strike-slip	reverse	set 3
high $0.4 < \phi < 1.0$	normal	reverse	strike-slip	set 2
	strike-slip	reverse	normal	set 1
	locked	locked	locked	set 3

Table 3.1. Limiting cases of rakes expected in the three stress regimes as a function of $\phi = (\sigma_2 - \sigma_3)/(\sigma_1 - \sigma_3)$, for three fault set orientations, each aligned with one of the principal stresses in their planes, and for a high maximum shear stress.

most extreme orientations and rakes of strike-slip faults in the MOJ domain and reverse faults in the WTR domain. Figure 3.8 shows a possible stress solution with σ_1 trending $N20^0E$, for $\mu_f = 0.4$. and $\phi = 0.1$. If we were to choose a higher friction coefficient or a differently oriented stress field, friction criteria and the irrotational stress field assumptions would not be simultaneously satisfied for these most extreme fault orientations and rakes. According to this simple model of distributed deformation, very different fault set orientations and rakes may be the product of a regionally irrotational strike-slip stress regime, in accordance with friction criteria.

Notice that this model cannot account for slip along domain bounding faults such as the San Andreas. Indeed, it predicts that very little shear stress is resolved on the San Andreas fault, the major through-going boundary fault in this region (SAF in Figure 3.8). This result, which agrees with recent borehole findings at Cajon Pass (Zoback et al., 1987), is a major problem in tectonophysics, because the mechanics of such weak faults are not readily explained by existing theories (see Lachenbruch and Sass, 1991). We shall restrict our attention to the mechanics of faults within domains.

In summary, we see discrete fault orientations that separate into domains. According to our model, in Southern California even the most unfavorably oriented fault sets can, at present, slip in a strike-slip stress regime provided that σ_1 is $N20^0E$ directed, $\phi = 0.1$ and $\mu_f = 0.4$. We will return to these values in the final part of the chapter (section 3.6) when discussing the faulting history of the WTR domain.

A fundamental question must be considered at this point: how do faults that presumably form in a favorable orientation become unfavorably oriented? As mentioned previously, extensive paleomagnetic studies (references in Luyendyk et al., 1985 and Dokka, 1989) indicate that these domains have undergone a complex history of deformation including rotations about a vertical axis. In the following sections, we hope to demonstrate that three dimensional block rotation is a simple mechanism to explain
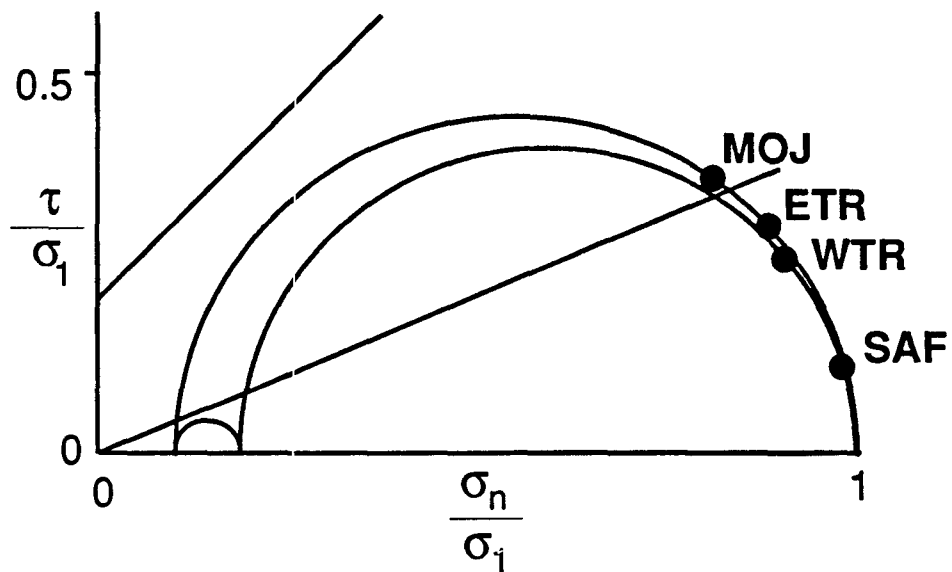
Fig. 3.8 Representative fault planes shown in Figure 1 are plotted on a 3D Mohr circle (normalized by σ_1). A slope of 0.4 for the sliding line, a value of $\phi = 0.1$, and a regional strike-slip stress regime with σ_1 oriented N20^{0}E are required to slip faults in the domains with the most extreme fault orientations. Notice that the San Andreas fault (SAF), a major throughgoing boundary fault in this region, cannot slip in accordance with friction criteria assumed in this model. See text for discussion.

the paleomagnetically inferred rotations in accordance with friction criteria.

3.4 Three dimensional block rotation

In the previous section, we defined conditions in a three dimensional stress field under which reactivation of unfavorably oriented faults can occur (Table 3.1). We showed that domains of unfavorably oriented faults can be active in a regionally irrotational stress field if the limits for the magnitude of the maximum shear stress in the Earth's crust lie between the upper bound (rock strength) and the lower bound (fault strength).

We now investigate a process which allows faults to pass from optimal to unfavorable orientations: the rotation of blocks and their associated faults. The model allows us to determine both the maximum rotation possible for a given fault set, and the conditions needed to induce a change of rake and orientation.

3.4.1 The model

Following previous authors (Garfunkel, 1974; Ron et al., 1984; Luyendyk et. al, 1985; and others) we assume that rotating blocks are rigid. Faults in a set are typically sub-parallel, so it is reasonable to consider the rotation history of one fault to be representative of the entire set. We also assume that all faults in a given set are active simultaneously. We assume that faulting in the upper 15 kms of the earth's crust occurs by brittle processes and that a detachment at depth decouples upper crustal rotations from the lower ductile crust.

In the brittle upper crust, we have four criteria for the three dimensional BR model. The first, a mechanical constraint, assumes that the Coulomb criterion controls sliding of faults in the set. The second assumes that fault slip and rotation are directed along the maximum resolved shear stress on the fault (see appendix B, Figure B1). The third, a kinematic constraint, assumes that both blocks and fault planes rotate away from the σ_1 direction. The fourth and final assumption is the

irrotational aspect of the principal stress directions in both space and time. Thus, once fault sets have slipped and blocks rotated, stress magnitudes need to change to allow further slip and rotation.

To facilitate the analysis, we assume that the magnitude of the principal vertical stress, S_v, remains constant throughout rotation. The other two stress magnitudes vary to conserve the stress ratio ϕ (see appendix B for discussion). Here, we present results for a constant ϕ stress model only.

Given a stress model and an irrotational stress field, we would like to keep track of the fault orientation and rake during rotation. Two new concepts are therefore introduced: the stress path and the rotation path of a rotating fault. These concepts form the basis of the three dimensional BR model.

3.4.2 Two dimensional stress path

For a given fault orientation, stress model, and initial state of stress, we calculate the stress path of a fault, that is, the subsequent values of $[\sigma_n, \tau]$ resolved on the fault as the state of stress changes. For example, consider a domain with a preexisting fault set with a pole oriented at an angle α_1 to the σ_1 direction (Figure 3.9a, point 1), an initial state of stress given by the values of $[\sigma_1, \sigma_3]$, and a stress model that keeps the mean stress, σ_o, constant. The aim is to drive the fault into motion. Thus, as the state of stress changes to $[\sigma'_1, \sigma'_3]$, the $[\sigma_n, \tau]$ values resolved on the fault plot along a line that goes from point 1 to point 2 in Figure 3.9a. This line represents the stress path for a given fault orientation and stress model. The stress path concept is discussed in considerable detail in Estevez et al. (1987) and is summarized in appendix B here. For the purpose of this chapter, we are interested in the intersection point of the stress path with the sliding line. At the intersection point (2 in Figure 3.9a,b) the fault slips and rotates. The new fault orientation will produce a new stress path and therefore a new intersection point with the sliding line. The sequence of intersection points represents the rotation path of the rotating fault.
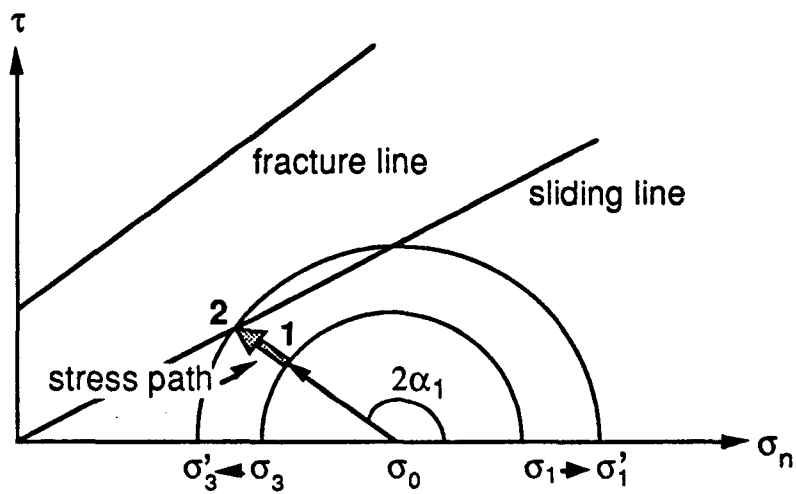
Fig. 3.9.a The stress path. Assume a set of faults exists in a domain with a pole oriented at α_1 to the σ_1 direction. The σ_2 direction is contained in the fault plane. Stress magnitudes must change $(1 \rightarrow 2)$ to allow sliding on the fault set. Assume that this change takes place by keeping the mean stress σ_0 constant. The line from point 1 to point 2 is the stress path of this fault set for a stress model with $\sigma_0 =$const.
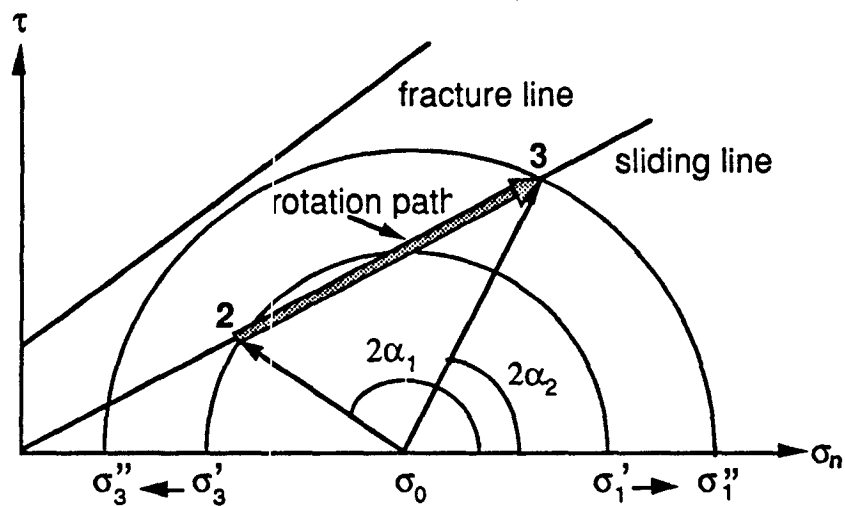
Fig. 3.9.b The rotation path. At point 2 the stress path intersects the sliding line, therefore there is sufficient shear stress along the faults of the set to overcome friction. Faults and blocks slip, and rotation is assumed to occur. The Mohr circle plots poles (see Figure 5). Thus, as rotation proceeds, stress magnitudes change and the intersection points plot closer and closer to σ_1, tracing a rotation path from intersection point 2 to intersection point 3 along the sliding line.
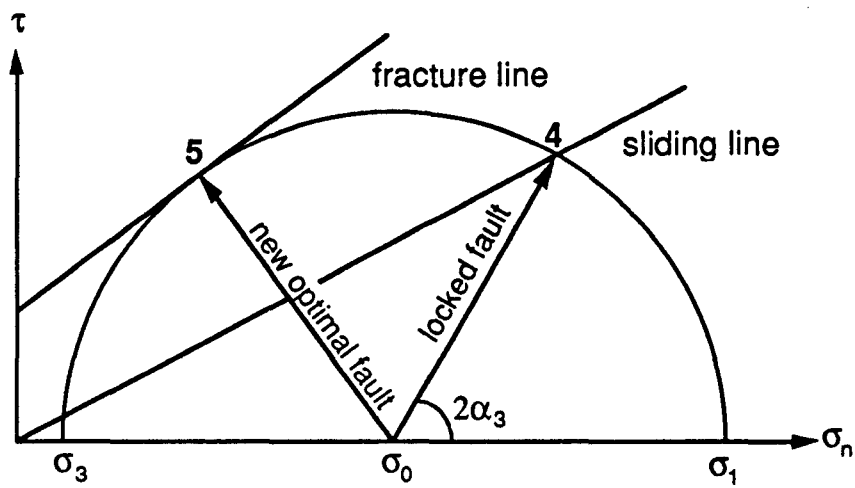
Fig. 3.9.c At intersection point 4, rotation along this set of faults ceases because the maximum shear stress magnitude reached the fracture line. Therefore a new, more optimally oriented set of faults forms, point 5, and the old set remains locked into an orientation α_3 relative to σ_1. In 2D, only fault orientations with the σ_2 direction in their plane can be modeled.

3.4.3 Two dimensional rotation path

Figure 3.9b,c describe schematically the sequence of intersection points of the stress paths with the sliding line: after many slip and rotation episodes, the intersection point moves from 2 to 3. As shown earlier in Figure 3.5, the angles α_1 and α_2 represent the orientation of a pole, thus as the fault rotates away from σ_1, the pole of the fault approaches the σ_1 point along the σ_n axis. Eventually, at intersection point 4 (Figure 3.9c) the Coulomb fracture criterion comes into play—the fault set becomes locked in a position α_3, and a new set of faults will be created (point 5 in Figure 3.9c). The path followed from intersection point 2 to 4 is the rotation path for the two dimensional case where σ_2 is in the plane of the fault.

3.4.4 Three dimensional rotation paths

Many rotation paths are possible in three dimensions. Here we analyze only three initial orientations of fault sets, as shown in Figure 3.10a. The initial orientation of faults in the stress field is critical in determining the rake and therefore the rotation path of faults. Case A represents the two dimensional case of a rotating fault. The rake is constant because σ_2 is contained in the plane of the fault. The rotation path in this case follows the $\sigma_1 - \sigma_3$ circle. Once σ_2 is off the plane of the fault small differences in the initial fault orientation, as between case B and C, may give very different results. In case B, rake and fault rotation are controlled by σ_1, σ_2, and σ_3. Rotation is truly three dimensional and faults display rakes that vary considerably along the rotation path. On the other hand, in case C only σ_2 and σ_3 control rake and block rotation. σ_1 is almost contained in the fault plane and plays a minor role. Therefore along this rotation path rakes remain relatively unchanged. The calculations involved in rotating a fault in three dimensions are detailed in appendix B.

Except for the two dimensional case it is clear that changes in rake occur in most

cases as faults rotate in an irrotational stress field. Because we assume a constant ϕ stress model, we can summarize the three cases of Figure 3.10a into Figure 3.10b where rotation paths are plotted on a three dimensional Mohr diagram (only rotation paths are shown) and on a Wulff projection (only the sequence of fault orientations are plotted). Rotation paths plot as curved lines on normalized Mohr diagrams because the absolute magnitude of the principal stresses varies with fault rotation.

In the following section, we present the results of three dimensional BR modeling for each stress regime. We have assumed irrotational stress directions and constant ϕ during fault rotation. Separately, we discuss modeled changes in rake and fault orientation (strike and dip) for the same three initial fault orientations described in Figure 3.10. Rakes, strikes and dips will be very different, of course, depending on the value of ϕ, the stress regime considered, and the initial fault orientation.
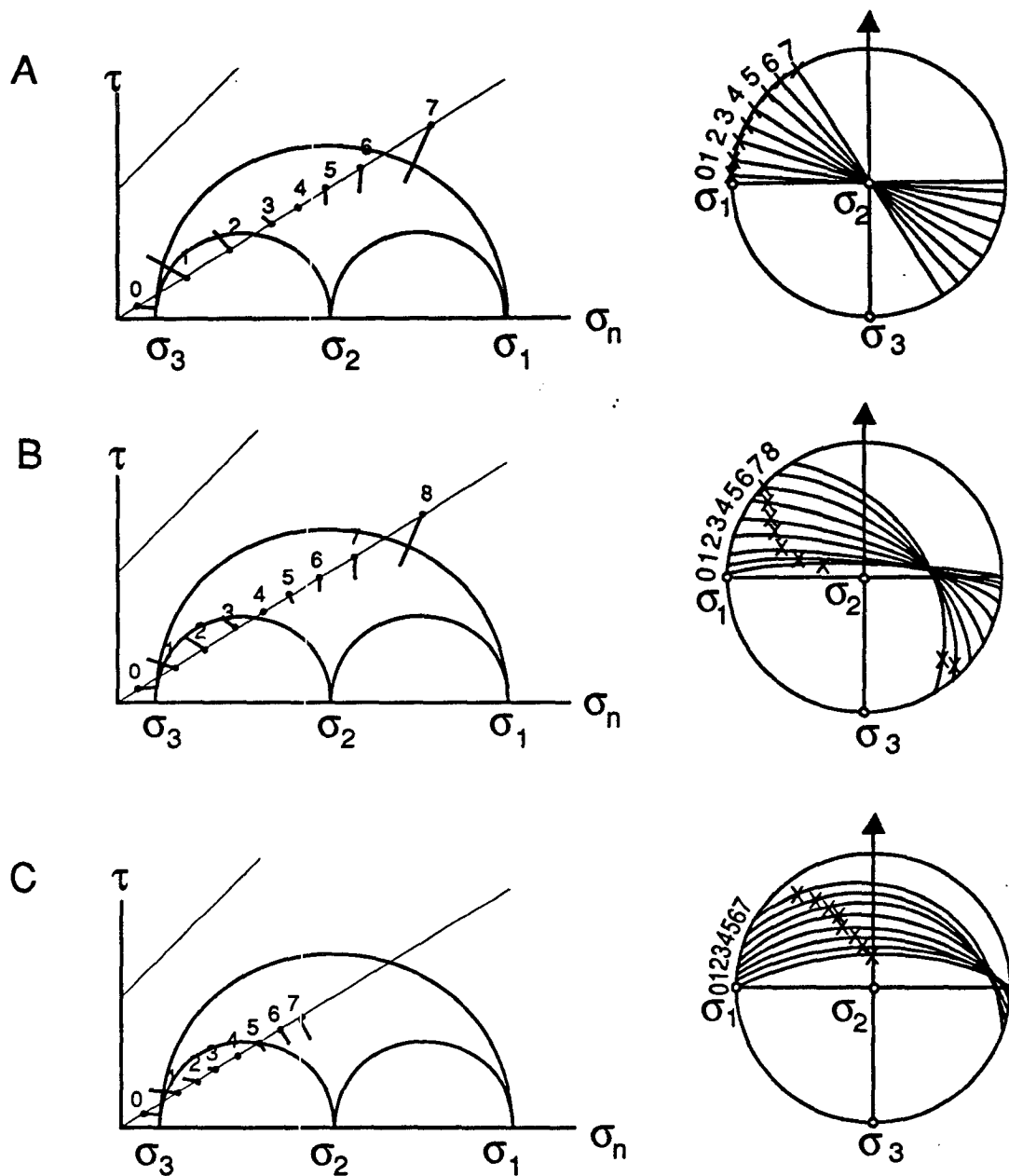
Fig. 3.10.a In 3D there are many rotation paths; only three are analyzed in this paper. Although their initial position is hard to make out because they all plot close to the σ_3 direction, the initial orientation of the fault set is close to the $\sigma_1 - \sigma_3$ circle for case A, close to the $\sigma_2 - \sigma_3$ circle for case C, and at an intermediate position between the two for case B. Only the magnitude of the initial Mohr circle is shown. The numbers on the Coulomb sliding line refer to the successive positions of fault sets at slip time, solid lines represent their stress paths. The corresponding fault plane orientations and rakes (crosses) are plotted in Wulff projections in the right column.
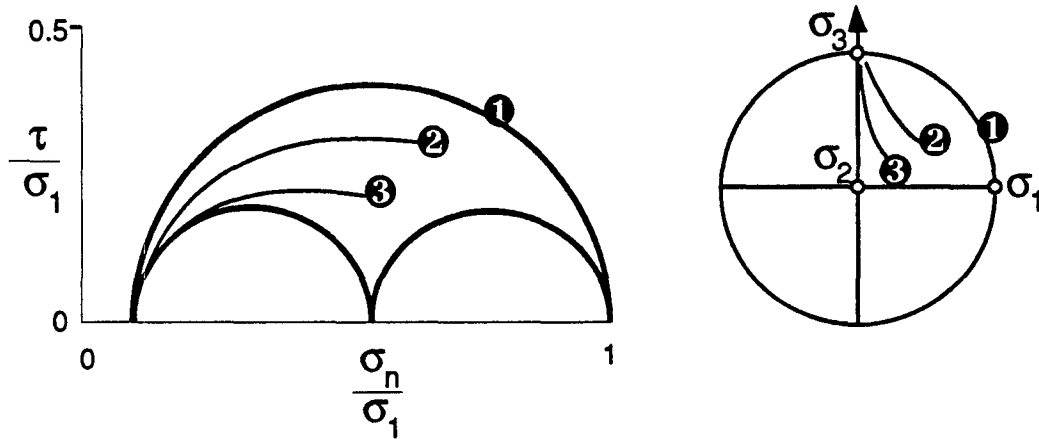
Fig. 3.10.b Cases A, B, and C in Figure 10*a* correspond here to rotation paths 1, 2, and 3. Rotation paths shown in this figure and Figure 11 are plotted on graphs normalized by the magnitude of σ_1, for a $\phi = $ const stress model. Thus rotation paths will appear as curved lines. Symbols are at the end of rotation paths.

rotation path of the three dimensions cases (paths #2 and #3 in Fig 10a) depends on the ϕ value and on the initial fault orientation. The higher the ϕ value the more constant is the rake throughout rotation, as illustrated by the Mohr diagrams in Figure 3.11. As $\phi \to 1$, the $\sigma_2 - \sigma_3$ circle grows at the expense of the $\sigma_1 - \sigma_2$ circle, and rotation paths stop farther away from the $\sigma_1 - \sigma_2$ circle. At lower ϕ values, however, where rotation paths reach all the way to the $\sigma_1 - \sigma_2$ circle, the model yields surprising rake histories for the three dimensional cases.

In the normal stress regime, an initially strike-slip fault may rotate into a pure normal fault (paths #2 and #3). In the reverse stress regime, an initially oblique-slip fault with a large component of strike-slip motion may rotate into a reverse slip fault (path #2) or a strike-slip fault (path #3). The same rotation paths for the $\phi = 0.1$ case, in the strike-slip stress regime, show the most interesting, and perhaps the most surprising results of our model. They suggest that it is possible for a fault (path #3) to be reactivated with a pure normal rake, rotate into an orientation that produces a strike-slip rake, and subsequently into one that produces reverse-oblique slip rake— all within a strike-slip stress regime. Even at intermediate ϕ values in the strike-slip stress regime, dramatic changes take place in the rakes of paths #2 and #3.

3.5.2 Changing fault orientation—Strike and Dip

Fault orientations and rakes are plotted in Figure 3.12 for the same rotation paths shown in Figure 3.11. These results re-emphasize the difference between the three dimensional BR model and classical (i.e. two dimensional) models of rotation usually assumed in paleomagnetic interpretations. The columns represent the three ϕ cases of Figure 3.11, but Figure 3.12a,b,c show the normal, reverse and strike-slip stress regime separately. Three dimensional motion is shown by plotting *rake/strike, rake/dip*, and *strike/dip* along each rotation path.

Rotation paths are plotted on Wulff projections at the bottom of Figure 3.12, to help visualize the changing orientation of faults during fault and block rotation.
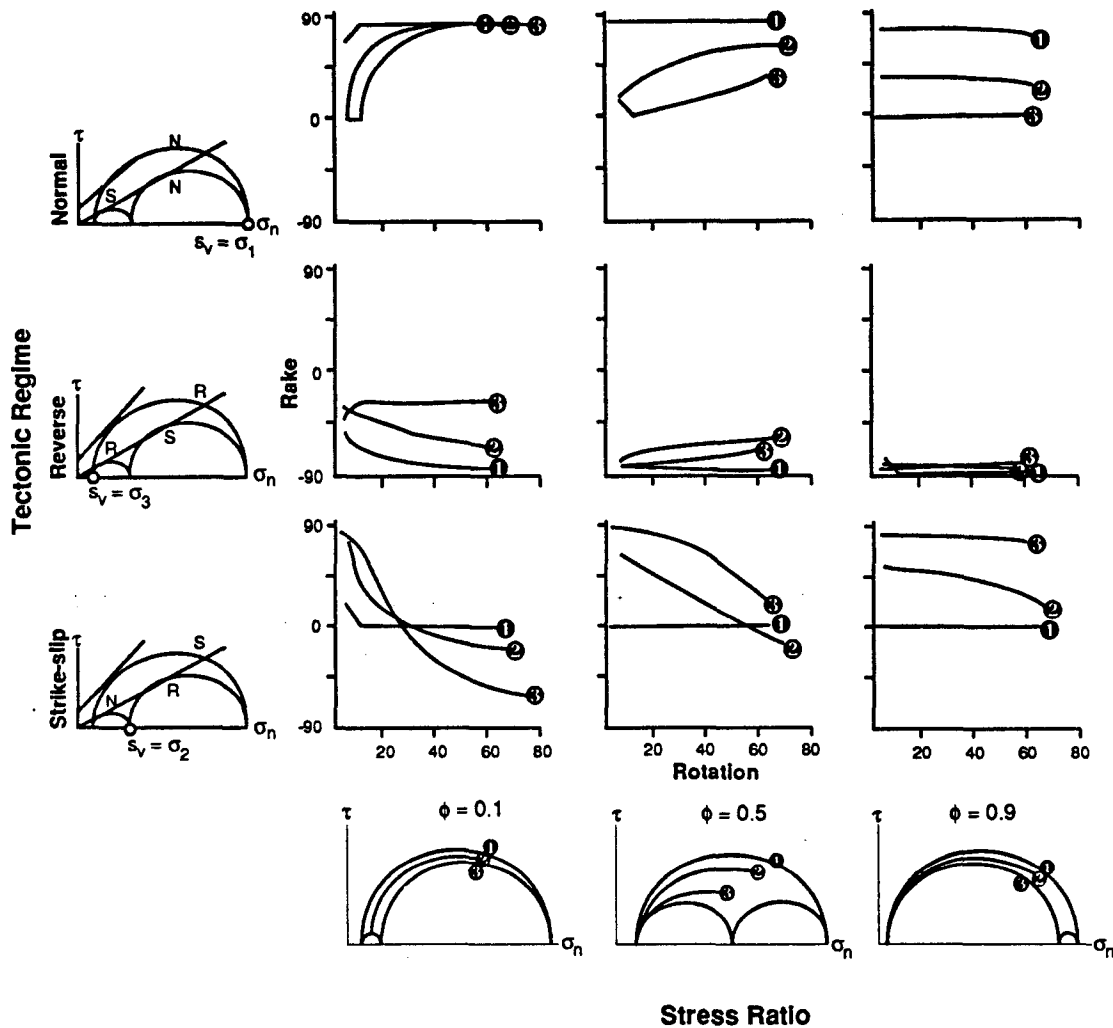
Fig. 3.11 Cross-plot of rake versus rotation for different stress regimes and ϕ values. Strike-slip fault motion (**S**) is at 0^0, normal motion (**N**) is at 90^0, and reverse motion (**R**) is at -90^0. Variations in rake with rotation are controlled by σ_2. At low ϕ, large changes in modeled rakes are predicted during rotation, while at high ϕ, rakes remain largely unchanged. Symbols are at the end of rotation paths.

The vertical principal stress for each stress regime is plotted in the down direction. Notice, again, the stabilizing effect of high ϕ values displayed by the shape of the rotation paths. The dominant σ_1 at $\phi = 0.1$, drives all three paths very close to its direction. Some curvature of the paths can be noticed at $\phi = 0.5$, while straight paths characterize the $\phi = 0.9$ case— where σ_1 and σ_2 are nearly equal in magnitude.

The model results for the case of the normal stress regime are detailed in Figure 3.12a. For the low ϕ case, fault sets show mostly a flattening of the dip. A horizontal axis of rotation of faults and blocks is predicted, similar to two dimensional models. But at intermediate ϕ values increased changes in strike are predicted during rotation, and at high ϕ values the model predicts greater changes in the strike than in the dip of faults, particularly for the most extreme case (path #3). Thus the results for the normal stress regime are surprising because, depending on fault orientation and ϕ ratio, both horizontal as well as vertical-axis rotation of faults and blocks are mechanically possible.

The model results for the reverse stress regime case are shown in Figure 3.12b. For the low ϕ case, the model predicts orientation and rake changes throughout rotation for both fault sets (paths #2 and #3). Again, the results are surprising because greater changes in the strike than in the dip of faults are predicted. Thus vertical-axis rotation of faults and blocks is mechanically possible in the reverse stress regime as well. Increasing ϕ values induce predominantly steepening of the fault planes during rotation, resulting in the classic, two dimensional horizontal-axis rotations of faults and blocks.
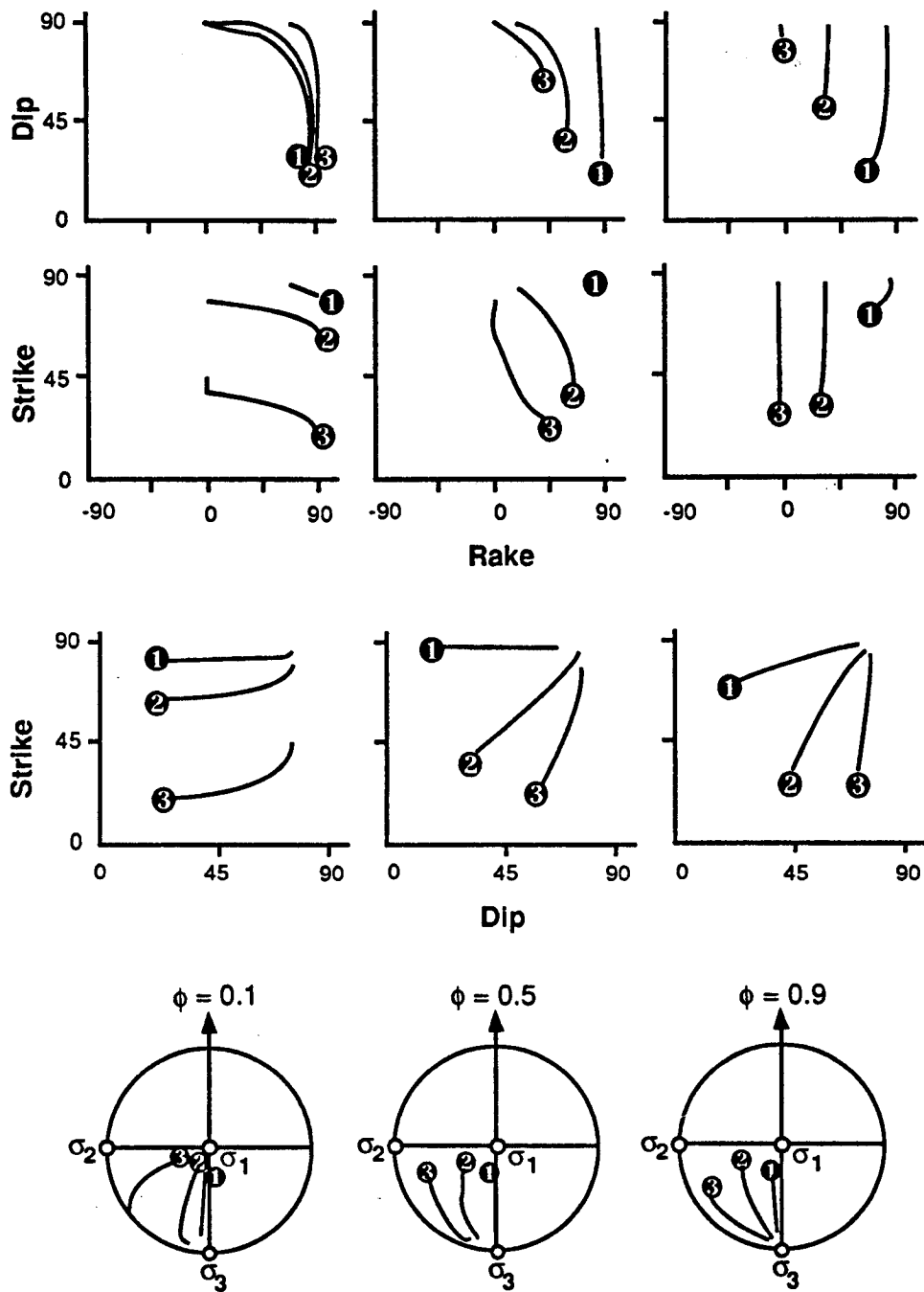
Fig. 3.12.a Rake versus dip, rake versus strike, and dip versus strike are plotted for the three rotation paths at different values of ϕ. Wulff stereoprojections for the three rotation paths are shown, for reference, at the bottom of each figure.

Normal stress regime, $S_v = \sigma_1$. The flattening of the dip decreases with increasing ϕ. The rake changes from strike-slip to normal at $\phi = 0.1$, but remains constant at the initial value at higher ϕ values. This is accompanied by an increase in vertical axis rotation with increasing ϕ (see paths 2 and 3): the strike is relatively constant at $\phi = 0.1$ but changes considerably at higher ϕ values (σ_3 is at strike= 0^0).

86
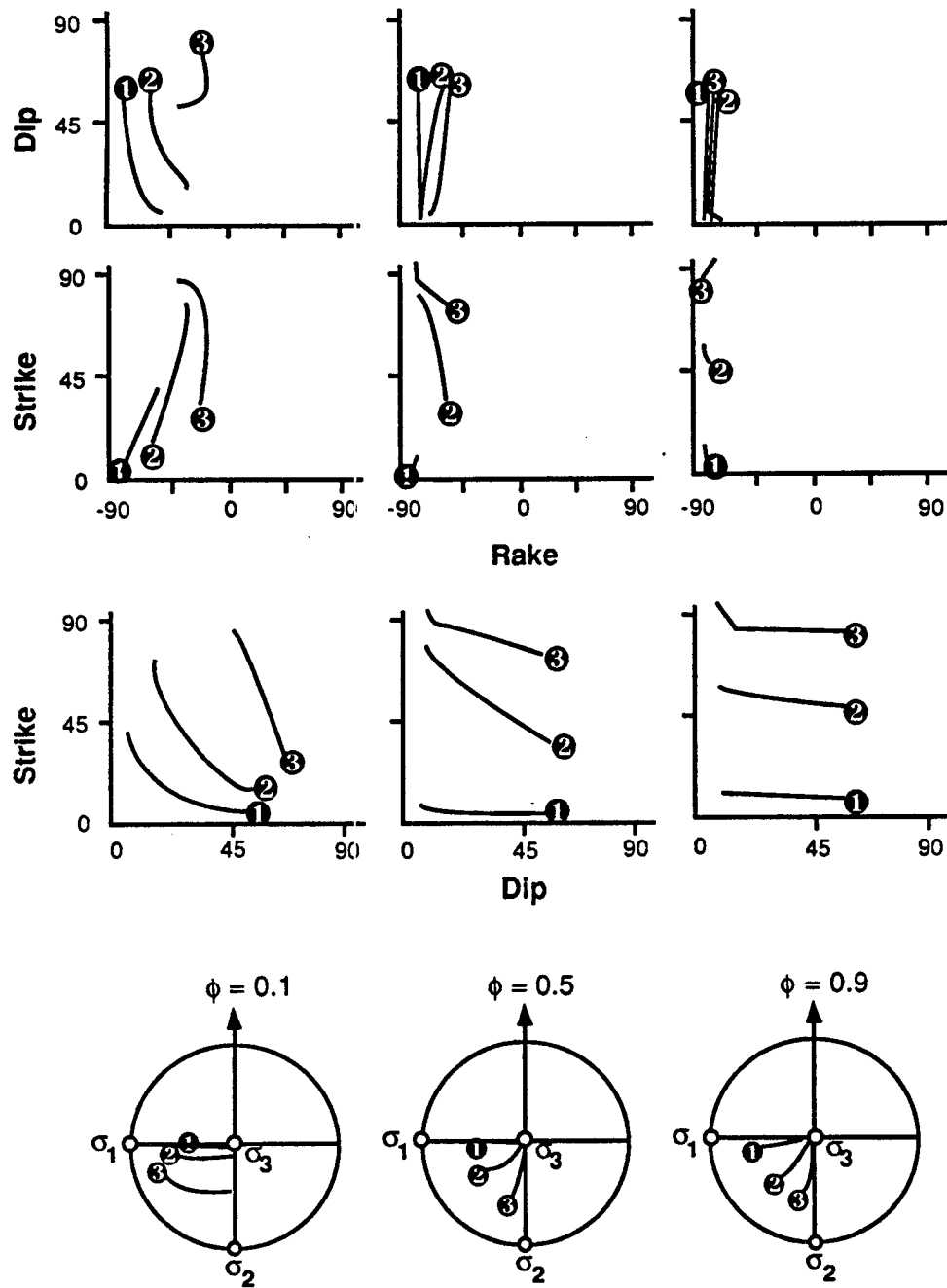
Fig. 3.12.b Reverse stress regime, $S_v = \sigma_3$. The dip steepens throughout rotation for all fault planes at all ϕ values. The rake changes from oblique-strike-slip to oblique-reverse-slip (path 2) or to almost pure strike-slip (path 3) at $\phi = 0.1$. These paths show therefore some vertical axis rotation because the strike of faults changes considerably. At intermediate and high ϕ values the rake is reverse along all rotation paths, and rotations occur predominantly about a horizontal axis (σ_2 is at strike= 0^0).
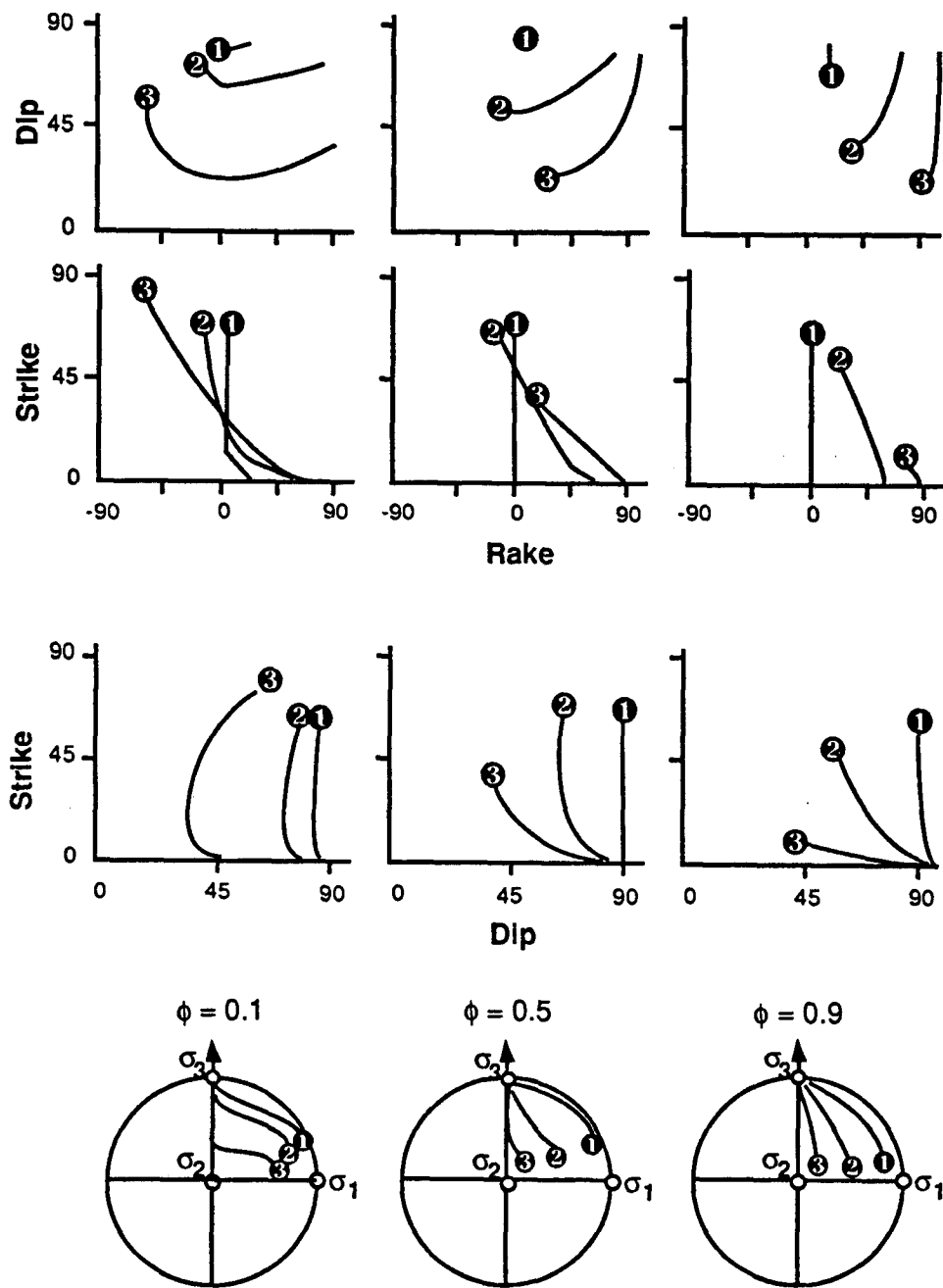
Fig. 3.12.c Strike-slip stress regime, $S_v = \sigma_2$. As the rake goes from normal, through strike-slip and then to reverse, the dip of the fault plane flattens at first and then steepens at $\phi = 0.1$. At higher ϕ values, rotation stops before the reverse field is reached. The strike change is greater than the dip change, producing predominant vertical axis rotations in most cases (σ_1 is at strike= 0). An exception to this is rotation path 3 at $\phi = 0.9$: the strike does not change, only the dip and therefore a purely horizontal axis rotation can occur in a strike-slip stress regime.

88

The results for the strike-slip stress regime (Figure 3.12c), are the most notable ones. Interestingly, for the low ϕ case the model predicts a decrease in dip during normal slip, followed by a steepening of the dip when the rake becomes reverse. In the strike-slip stress regime, the model predicts that most fault sets will follow a mixed axis of rotation, where both dips and strikes of faults change during rotation. One notable exception, though, is the rotation path of the most extreme fault set (path #3) for the $\phi = 0.9$ case, where faults and blocks rotate about a horizontal axis in the strike-slip stress regime.

3.5.3 Discussion

To summarize the results of our model, three specific initial orientations of faults were studied in nine different combinations of ϕ values ($\phi = 0.1, 0.5, 0.9$) and stress regimes ($S_v = \sigma_1, \sigma_2, \sigma_3$). In each case the predictions of the model were presented using the rotation path. Along this path we analyzed fault rake and orientation as faults rotated in an irrotational, $\phi = constant$ stress field.

By considering extreme fault orientations, we can explain rotations of up to 75^0 about a vertical axis, much greater than those of Nur et al. (1986) for two dimensions. We have also found cases where rake, dip and strike of faults change considerably during the rotation path of a single fault set.

The most important conclusions are: (1) for most cases of fault rotation the results predict a mixed axis of rotation, (2) as faults change their orientation in the stress field due to rotation, the rake may change as well—sometimes dramatically, (3) it is mechanically possible for blocks and faults of a domain to rotate about a vertical axis in normal and reverse stress regimes, and about a horizontal axis in strike-slip stress regimes, given an initial fault set orientation close to containing the σ_1 principal stress direction. Therefore (4) paleomagnetically inferred rotations may not always be directly related back to a specific stress regime. Indeed, inferred rotations of blocks about vertical axis, while usually found in strike-slip stress regimes (Hornafius et al.,

1986; Kamerling and Luyendyk, 1985; and others), have been documented in normal stress regimes as well (Westaway, 1990; Pavlides et al., 1988; Hudson and Geissman, 1987; Kissel et al. 1986; Brown and Golombek, 1986).

Paleomagnetism is an invaluable aid to decipher complex histories of rotation in regions of distributed deformation (Ron et al., 1984). Our model provides a framework within which paleomagnetic, structural and stress data can be combined to understand better complex rotation histories. We analyze such a case in the following section, to demonstrate the application of the three dimensional BR model to complex tectonic problems.

3.6 The Western Transverse Ranges: a 3D block rotation model

The shear motion between the Pacific and North American plates in Southern California is distributed across a 200 km wide zone which consists of a complex array of block-faulted domains (Figure 3.1; Luyendyk et al., 1985). In section 3.3 (Figure 3.8) of this chapter, we showed how the diverse orientations and rakes observed today in these domains can be explained in an irrotational strike-slip stress regime. In this section, we apply the results of the three dimensional BR model to the rotation history of one of these domains, the Western Transverse Ranges (WTR).

3.6.1 Tectonic history

The WTR domain is a region limited to the north by the Santa Ynez fault, to the south by the Malibu fault system, to the west by the Hosgri fault and to the east by the San Gabriel fault (see WTR in Figure 3.1). The right-lateral shear between the Pacific and North American plates since Oligo-Miocene time should have dominated the tectonic history of this domain. From the geological record alone, however, the tectonic history seems much more complex. This is documented by detailed stratigraphic and structural studies (Yerkes et al.,1981; Luyendyk et al., 1985; Yeats, 1987;

90

Namson and Davis, 1988 and others). The following discussion is concerned with the most prominent deformational phases documented for this region.

Phase I: in late Oligocene to early Miocene time, there was deposition of sediments in elongate almost north-south striking normal-fault controlled basins. Today these basins are oriented east-northeast (Terres and Luyendyk, 1985).

Phase II: a mixture of strike-slip and normal faulting characterized the extensional tectonics of Miocene times (Yeats, 1987).

Phase III: a significant period of compression followed at the beginning of the Quaternary, overprinting all previous events. Based on structural evidence Yerkes and Lee (1979a) suggest that "at the present rates all the measured compressive deformation within the WTR could have occurred during the last 0.5 to 1 m.y." Namson and Davis (1988) describe in detail this last phase of north-south convergence.

This sequence of phases is shown schematically in Figure 3.13: a rake of 90^0, representing the normal faulting period—phase I, is followed by a rake of 0^0, representing the strike-slip faulting period—phase II, leading to a rake of -65^0, representing the present day period of reverse oblique faulting—phase III.

With the advent of paleomagnetic studies carried out in many structural domains across Southern California (Luyendyk et al., 1980; Terres and Luyendyk, 1985; Hornafius et al., 1986: Carter et al., 1987), it appears that block rotation has been the predominant mechanism of deformation in this region. Paleomagnetic interpretations indicate that since mid-Miocene time, the Transverse Ranges have experienced significant rotation of blocks about a vertical axis, particularly in the WTR domain where up to 90^0 of rotation is estimated.

According to Luyendyk and his co-workers, the WTR clockwise rotations were associated with left-lateral slip on a set of vertical, strike-slip faults originally north-northeast trending. The faults defined blocks about 100-200 kms long and 10 kms wide that rotated as rigid bodies away from the direction of compression.
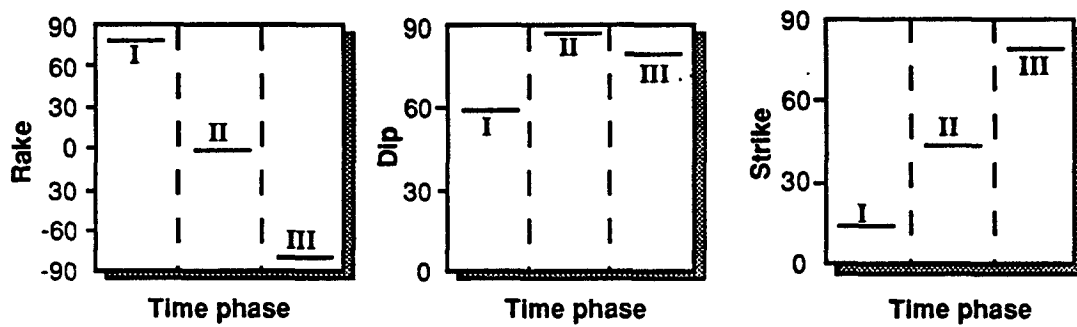
Fig. 3.13 Observed and inferred rake, dip, and strike of the WTR fault set for the three phases of deformation, as discussed in the text. σ_1, assumed at N20^{0}E, is at strike= 0^0 on this plot. Based on structural, geological and paleomagnetic data, **I** was a predominantly normal faulting phase, **II** was a predominantly strike-slip faulting phase, and **III** is the present-day phase.

Terres and Luyendyk (1985) suggest that the structural history for the WTR, can be explained by three separate set of faults formed under three different stress regimes (Figure 3.13). A set of north-northeast striking normal faults formed during extension, phase I, another set of strike-slip faults, which rotated with the blocks, formed during shearing, phase II, and yet another set of east-west striking high angle reverse faults formed during the present-day compression, phase III. By combining the frictional constraints of the three dimensional BR model with paleomagnetic, structural and geologic data, we show how only one set of faults, preexisting and rotating in an irrotational strike-slip stress field, can account for all three deformational phases.

3.6.2   The three dimensional BR model for the WTR

How can we use the results of Figure 3.11 and 3.12 to help us understand the history of rotation of the WTR? Let us start with the simplest assumption: the stress directions remained irrotational, in the present day orientation, throughout the post-Oligocene deformation history of the domain. According to our estimates from section 3.3 (Figure 3.8), we know that the predominant east-northeast faults in the WTR may slip in an oblique reverse motion in a strike-slip stress regime with σ_1 $N20^\circ E$ directed, if $\mu_f = 0.4$ and $\phi = 0.1$. Let's therefore consider rotation path #3 in Figure 3.11, which ends with an oblique reverse rake similar to that of the WTR, for the $\phi = 0.1$, strike-slip stress regime case. Lowering the coefficient of frictional sliding from $\mu_f = 0.6$ (Figure 3.12c) to $\mu_f = 0.4$ (Figure 3.14) allows faults to rotate further away from σ_1. Note that the shape of the rotation path remains the same for both friction values. The only difference is that faults with a lower friction coefficient may rotate further, in the case of the WTR until a steeper dip is achieved. What does this rotation path tell us about previous fault behavior and previous geometries of faulting? Figure 3.14 displays the structural history predicted by the model for this rotation path. The rake starts at an initial value near pure normal slip, a time of "extension" similar to phase I. Following fault and block rotation, the rake becomes

pure strike-slip, a time of predominantly vertical axis rotation, similar to phase II, before reaching the present day "compressive" period of reverse-oblique faulting—phase III. The corresponding values for the dip and the strike of the rotating fault set are also shown. As the strike of the fault set rotates away from σ_1 (I → II in the plot), the dip flattens at first and then steepens as soon as the reverse slip field is reached (II → III).

3.6.3   Agreement with the observations

Our goal was to demonstrate how deformation in the WTR could have occurred along a single set of faults, reactivated in middle Miocene time as block rotation began. In agreement with the observations (Figure 3.13), the results of the three dimensional BR model shown in Figure 3.15 display the salient features of the structural history of the WTR fault set. The original orientation of the preexisting fault set in the WTR domain may have been north-northeast striking, dipping $55^0 - 60^0$ to the west and slipping in a normal sense. Subsequently, the same fault set rotated away from the direction of maximum compression σ_1 and became a predominantly strike-slip fault set with a dip between 40^0 and 50^0. Finally, the fault set rotated into the present-day east-west, high angle orientation displaying a reverse-oblique sense of motion. The results of our modeling show that along the rotation path (Figure 3.15), a single set of faults changed from normal to strike-slip and then to predominantly reverse rakes during the 75^0 clockwise vertical axis rotation of faults and blocks, while stress directions remained irrotational with σ_1 oriented $N20^0E$, $S_v = \sigma_2$, and a constant $\phi = 0.1$ (see also Chapter 4).

Two of the most critical simplifying assumptions we made are not necessary. The first one is that of an irrotational stress field throughout the rotation history of the WTR. Studies of plate motion (Cox and Engebretson, 1985) indicate a clockwise change in relative plate motion ∼ 5 My ago, implying a rotation of the stress field. Incorporating this change would introduce an additional degree of freedom and allow

94
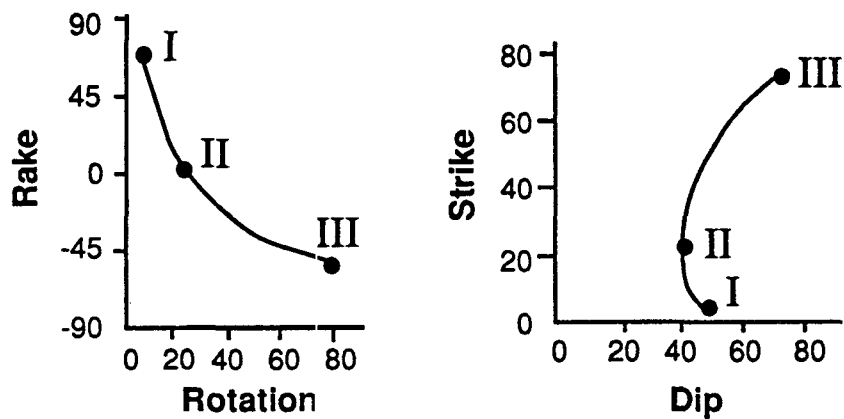
Fig. 3.14 Modeling 3D fault rotation in the West Transverse Ranges. According to the 3D BR model, a single set of rotating faults may have accommodated the deformation of the WTR domain. The model predicts, in accordance with the observations (Figure 13) and friction criteria, three different structural phases since rotation began: normal-oblique faulting (**I**) → strike-slip faulting (**II**) → high angle reverse-oblique faulting (**III**).
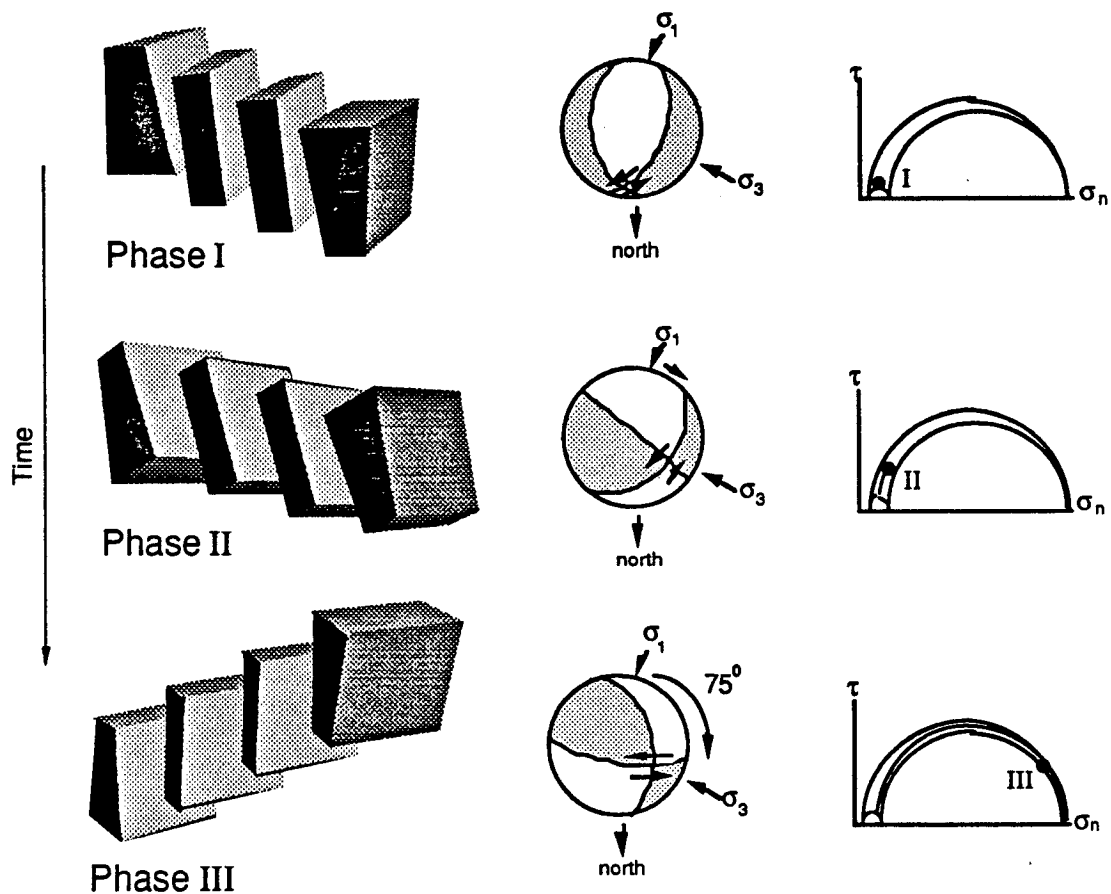
Fig. 3.15 The evolution of rake, dip, and strike of faults of the WTR through time. Each phase may be better visualized with idealized block diagrams and focal plane solutions (lower hemisphere projections). The corresponding rotation path predicted by the 3D BR model is shown on normalized Mohr diagrams to the right. When the strike of faults is almost parallel to σ_1 (phase I) faults move in a normal-oblique (rake= $+70^0$) sense. When none of the principal stresses are parallel to the strike of faults (phase II) faults move in a strike-slip sense (rake= 0^0). Finally, when the strike of faults becomes parallel to the σ_3 direction (phase III) faults move in a reverse-oblique sense (rake= -65^0).

a less restrained solution. The original orientation of faults could be closer to a north-south strike, and the amount of total rotation that could be achieved by one single set of faults would increase by an amount comparable to the change in plate motion.

The second constraining assumption, that of a constant ϕ stress model, maintains a strike-slip stress regime during the rotation history of the faults. This need not be the case. A change of stress regime might occur during the history of rotation of a fault set, and it could reverse the sense of rotation of blocks and faults or allow them to rotate in another direction.

With our simple three dimensional BR model assuming an irrotational stress field and a $\phi = 0.1$, we have been able to predict a sequence of structural phases consistent with the known structural history of the WTR. This example demonstrates that it is not necessary to invoke complex regional and local changes in the stress regime or erratic changes in plate motions to account for alternate periods of compression and extension. The geometry of each domain of fault sets and the values of ϕ determine the rake that will occur there. The regional stress regime may remain irrotational.

3.7 Conclusions

According to the existing two dimensional BR model, rotation of blocks and faults are caused by contemporaneous slip on fault sets in adjacent domains. Based on fracture mechanics and friction criteria, the orientation of faults sets with respect to the direction of maximum compression, σ_1, determines the sense of fault slip and thus the sense of rotation of the blocks in each domain. Consequently, faults that are initially in optimal orientations must rotate away from this optimal orientation in domains undergoing rotation. We generalized these concepts into a three dimensional model of block rotation and distributed deformation.

The need for a three dimensional model is dictated by the following observations: (a) faults are reactivated even if unfavorably oriented relative to the stress field, (b)

obliquely slipping faults are found in many domains that have undergone rotation, (c) these same faults change their rake through time (i.e West Transverse Ranges) and (d) block and fault rotation is not clearly related to the regional tectonic setting. We show that mechanical constraints of friction and strength combined with the three dimensional Mohr circle and the kinematics of block rotation can explain these observations.

The main conclusions from our three dimensional model results are:

(1) Block rotation is the cause of unfavorably oriented faults in many domains.

(2) Spatial variations in rake: with a simple model of distributed deformation, we are able to determine the range of fault orientations that may slip as well as their rakes. The value of ϕ plays a key role. At low ϕ values, where σ_2 is close in magnitude to σ_3, faults may be reactivated even if very unfavorably oriented. Thus, in different domains at the same point in time a range of rakes may be observed. In the normal stress regime, both pure strike-slip and pure normal rakes are possible, in the reverse stress regime, both pure strike-slip and pure reverse rakes are possible, and in the strike-slip stress regime, pure strike-slip, pure normal and pure reverse rakes can all occur. At high ϕ values, where σ_2 is closer to σ_1 in magnitude, rakes are limited to strike-slip and normal in both the normal and strike-slip stress regimes and only to reverse in the reverse stress regime.

(3) When applied to the distributed deformation of Southern California, our model can explain the diverse rakes and fault orientations observed in the different domains by assuming an homogeneous regional strike-slip stress regime with σ_1 oriented $N20^0E$, $\phi = 0.1$, and $\mu_f = 0.4$.

(4) With the three dimensional BR we can also determine the maximum amount of rotation blocks and faults may experience before a new set of faults may take over the deformation. Thus, consistent with field observations, the model predicts only a discrete number of fault sets in each domain and up to 75^0 of rotation along a single

set of faults.

(5) Temporal variations in rake: the results show that rakes may vary during fault and block rotation at low ϕ values, but remain relatively constant at high ϕ values, demonstrating once again the key role of the intermediate principal stress.

(6) The three dimensional BR model can also explain the observations that paleo-magnetically inferred rotations are not always clearly related to the regional stress regime: initially unfavorably oriented faults may rotate about a vertical axis, at low ϕ values in the reverse stress regime, and at high ϕ values in the normal stress regime, whereas faults may rotate about a horizontal axis, at high ϕ values in the strike-slip stress regime. Therefore, contrary to classical two dimensional models of rotations, for the general case of initially unfavorably oriented fault sets, mixed axis of rotation are predicted by the three dimensional BR model in all three stress regimes.

(7) The best model results indicate that for the West Transverse Ranges domain, a single set of originally north-northeast striking faults could have accommodated the deformation observed there. Pre-existing faults were reactivated as normal faults at the beginning of the rotation period. Subsequently, the same faults rotated, and became left-lateral strike-slip until they finally rotated into the present day "compressive" period of east-west striking reverse faults. In spite of its simplifying assumptions, our model provides clues to earlier geometries of faults for the WTR that are consistent with the geological, geophysical and paleomagnetic records.

We believe that block rotation is a fundamental process in regions of distributed deformation where unfavorably oriented faults are widespread. When applied to actual geologic situations, the model provides a powerful, yet simple tool to model the complex faulting histories and complex fault geometries that characterize regions

of distributed deformation.

Acknowledgements

References

Anderson, E. M., The dynamics of faulting, Oliver and Boyd, Edinburgh, 206pp, 1951.

Angelier, J., B. Colletta, and R. E. Anderson, Neogene paleostress changes in the Basin and Range: a case study at Hoover Dam, Nevada-Arizona, *Geol. Soc. Am. Bull., 96*, 347-361, 1985.

Bott, M.H.P., The mechanics of oblique slip faulting. *Geol. Mag., XCVI*, 2, 1959.

Brown, L., and P. Golombek, Block rotations in the Rio Grande Rift, New Mexico, *Tectonics, 5*, 3, 423-438, 1986.

Carter, J. N., B. P. Luyendyk, and R. R. Terres, Neogene clockwise tectonic rotation of the eastern Transverse Ranges, California, suggested by paleomagnetic vectors, *Geol. Soc. Am. Bull., 98*, 199-206, 1987

Cox, A., and D. Engebretson, Change in plate motion of Pacific plate at 5 Myr BP, *Nature, 313*, 472-474, 1985.

Dokka, R. K., The Mojave extensional belt of Southern California, *Tectonics, 8*, 2, 363-390, 1989

Estevez, R., O. Scotti, and A. Nur, Block rotation in 3D, *Stanford Rock Physics Project*, Unpublished manuscript, 1987

Freund, R., Rotation of strike-slip faults in Sistan, Southeast Iran, *J. Geol. 78*, 188-200, 1970.

Freund, R., The Hope fault, a strike-slip fault in New Zealand, *N. Z. Geol. Surv. Bull., 86*, 49pp., 1971.

Freund, R. , Kinematics of transform and transcurrent faults, *Tectonophysics, 21*, 93-134, 1974.

Garfunkel, Z., Model for the late Cenozoic tectonic history of the Mojave desert, California, and for its relation to adjacent regions, *Geol. Soc. Am. Bull., 85*, 1931-1944, 1974.

Garfunkel, Z., 1988, Regional deformation by block translation and rotation, in *Paleomagnetic rotations and Continental deformation*, edited by C. Kissel and C. Laj, NATO ASI Series, Series C: Mathematical and Physical Sciences, 254, 181-208, 1988.

Handin, J., On the Coulomb-Mohr failure criterion, *J. Geophys. Res., 74*, 22, 1969.

Hornafius, J. S., B. P. Luyendyk, R. R. Terres, and M. J. Kamerling, Timing and extent of Neogene tectonic rotation in the western Transverse Ranges, California, *Geol. Soc. Am. Bull., 97*, 1476-1487, 1986.

Hudson, M.R., and W. J. Geissman, W.J., Paleomagnetic and structural evidence for Middle Tertiary counterclockwise block rotation in the Dixie valley region, west-central Nevada, *Geology, 15*, 638-642, 1987.

Jackson, J.A., and N. J. White, Normal faulting in the upper continental crust: observations from regions of active extension, *J. Struct. Geol., 11*, 1/2, 15-36, 1989.

Jaeger, J. C., and N. G. W. Cook, Fundamentals of rock mechanics, Methuen and Co. Ltd, London, 515pp, 1969.

Jaeger, J. C., and K. J. Rosengren, Friction and sliding of Joints, *Proc. Aust. Inst. Min. Met., 229*, 93-104, 1969.

Jennings, C.W, Fault Map of California, *Calif. Div. Mines and Geology*, 1975.

Jones, L.M., Focal mechanisms and the state of stress on the San Andreas fault in Southern California, *J. Geophys. Res., 93*, B8, 8869-8891, 1988.

Kamerling, M. J., and B. P. Luyendyk, Paleomagnetism and Neogene tectonics of the Northern Channel Islands, California, *J. Geophys. Res., 90*, B14, 12,485-12,502, 1985.

Kissel, C., C. Laj, *Paleomagnetic rotations and Continental deformation*, edited by C. Kissel and C. Laj, NATO ASI Series, Series C: Mathematical and Physical Sciences, 254, 181-208, 1988.

Kissel, C., C. Laj, and A. Mazaud, First paleomagnetic results from Neogene formations in Evia, Skyros and the Volvos region and the deformation of central Aegea, *Geophys. Res. Lett., 13*, 13, 1446-1449, 1986.

Lachenbruch, A. H., and J. H. Sass, Heat flow from Cajon Pass, Fault strength, and tectonic implications, *J. Geophys. Res.* in press, 1991.

Lee, W.H.K., R. F. Yerkes, and M. Simirenko, Recent earthquake activity and focal mechanisms in the Western Transverse Ranges, California, in *U. S. Geol. Surv.*, Map MF-1032, 1979.

Luyendyk, B.P., M. J. Kamerling, and R. R. Terres, Geometric model for Neogene crustal rotations in Southern California, *Geol. Soc. Am. Bull., 91*, 211-217, 1980.

Luyendyk, B.P., M. J. Kamerling, R. R. Terres, and J. S. Hornafius, Simple shear of Southern California during Neogene times suggested by paleomagnetic declinations, *J. Geophys. Res., 90*, B14, 12, 454-466, 1985.

McKenzie, D.P., The relation between fault plane solutions for earthquakes and the directions of principal stresses, *Bull. Seismol. Soc. Am., 59*, 2, 591-601, 1969.

Molnar, P., Continental tectonics in the aftermath of plate tectonics, *Nature, 335*, 131-137, 1988.

Namson, J., and T. Davis, Structural transect of the western Transverse Ranges, California: Implications for lithospheric kinematics and seismic risk evaluation, *Geology, 16*, 675-679, 1988.

Nur, A., H. Ron, and O. Scotti, Fault mechanics and the kinematics of block rotations, *Geology, 14*, 746-749, 1986.

Pavlides, S.B., D. P. Kondopoulou, A. A. Kilias, and M. Westphal, Complex rotational deformations in the Serbo-Macedonian massif (north Greece): structural and paleomagnetic evidence, *Tectonophysics, 145*, 329-335, 1988.

Powell, R.E., Crystalline basement terranes in the South-Eastern Transverse ranges , Southern California, in *Geological excursion in the Transverse Ranges*, edited by GSA Cordillera section, 109-151, 1982.

Ransome, F.L., W. H. Emmons, and G. H. Garrey, Geology and ore deposit of the Bullfrog district, Nevada, *U. S. Geol. Surv. Bull., 407*, 1-130, 1910.

Ron, H., R. Freund, Z. Garfunkel, and A. Nur, Block-rotation by strike-slip faulting: structural and paleomagnetic evidence, *J. Geophy. Res., 89*, 6256-6270, 1984.

Ron, H., A. Nur, and Y. Eyal, Multiple strike slip fault sets: a case study from the Dead sea transform, *Tectonics, in press*, 1990.

Sauber, J. W., W. Tatcher W., and S. C. Solomon, Geodetic measurements of deformation in the central Mojave Desert, California, *J. Geophys. Res., 91*, 12,683-12,693, 1986.

Terres, R. R., and B. P. Luyendyk, Neogene tectonic rotation of the San Gabriel region, California suggested by paleomagnetic vectors, *J. Geophys. Res., 90*, B14, 12, 467-12, 484, 1985.

Westaway, R., Block rotation in western Turkey: 1. Observational evidence, *EOS, 71*, 16, 408, 1990.

Yeats, R.S., Changing tectonic styles in Cenozoic basins of Southern California, in *Cenozoic basin development of coastal California*, edited by R. V. Ingersoll and W. G. Ernst, Rubey Vol VI, 259, 1987.

Yerkes, R.F., and W. H. Lee, Late quaternary deformation in the West Transverse ranges, California, *U. S. Geol. Surv. Circ., 799-B*, 27-37, 1979a.

Yerkes, R.F. and W. H. Lee, Map showing faults and fault activity and epicenters, focal depths and focal mechanisms for 1970-1975 earthquakes, Western transverse Ranges, California, *U. S. Geol. Surv. Misc. Field Studies*, Map MF-1032, 1979b.

Yerkes, R.F., H. G. Greene, J. C. Tinsley, and K. R. Lajoie, Seismotectonic setting of the Santa Barbara channel area, Southern California, *U. S. Geol. Surv. Misc. Field Studies*, Map MF-1169, 1981.

Zoback, M.D., and J. H. Healy, Friction, faulting, and << in situ >> stress, *Annales Geophysicae, 2*, 6, 689-698, 1984.

Zoback, M.D., M. L. Zoback, V. S. Mount, J. Suppe, J. P. Eaton, J. H. Healy, D. H. Oppenheimer, P. A. Reasenberg, L. M. Jones, C. B. Raleigh, I. G. Wong, O. Scotti, and C. M. Wentworth, New evidence on the state of stress of the San Andreas fault system, *Science, 238*, 1105-1111, 1987.

Chapter 4

Visualizing block rotation in three dimensions

Abstract

The 20 minutes long video provided with this thesis, summarizes the main points of this study. It is composed of four parts: the first part is an introduction to the problem of distributed deformation, the second and third part explain visually the kinematics and mechanics of the block rotation model with the help of computer simulations of rotating rigid blocks. It also shows the formation of multiple sets of faults. Finally, the fourth part provides a short tutorial on the three dimensional Mohr circle and shows results of computer simulations of rigid blocks slipping and rotating in a three dimensional stress field with applications to the West Transverse Ranges domain, southern California. (Copies of the video tape [VHS-format] may be obtained by writing to Prof. Amos Nur, Geophysics Dpt., Stanford University, Stanford, CA 94305).

Appendix A

Eshelby's inclusion problem

The following is a brief summary of Eshelby's (1957) derivation as presented in Mura (1987). Expressions for the solution are different for interior points (points inside the inclusion) and exterior points (points outside the inclusion). Eshelby's most valuable result is that the strain and stress fields become uniform for the interior points of an elliptical inclusion. In this appendix we discuss the solution for the interior points only. When, as in our case, the elastic moduli of the ellipsoidal subdomain differ from the remainder (matrix), the subdomain is called an ellipsoidal inhomogeneity. A material containing inhomogeneities is free from any stress field unless a load is applied.

The equivalent inclusion method. Eshelby (1957) first pointed out that the stress disturbance in an applied stress field due to the presence of an inhomogeneity can be simulated by an eigenstress caused by an inclusion when the eigenstrain is chosen properly. "Eigenstrain" is a generic name introduced by Mura (1987) for such nonelastic strains as thermal expansion, phase transformation and misfit strains. Equivalently, "eigenstress" is a name introduced for self-equilibrated internal stresses caused by one or several of these eigenstrains which are free from any other external force and surface constraint.

Given a domain Ω with elastic moduli C^*_{ijkl}, and a matrix $D - \Omega$ with elastic

moduli C_{ijkl}, Hooke's law is written as:

$$\sigma_{ij}^0 + \sigma_{ij} = C_{ijkl}^* \left(u_{k,l}^0 + u_{k,l}\right) \quad in \ \Omega \tag{A.1}$$

$$\sigma_{ij}^0 + \sigma_{ij} = C_{ijkl} \left(u_{k,l}^0 + u_{k,l}\right) \quad in \ D - \Omega \tag{A.2}$$

where σ_{ij} and u_i are the stress and displacement disturbance, σ_{ij}^0 and u_i^0 are the applied stress and corresponding displacement at infinity, and the summation convention for repeated indices is employed.

The equivalent inclusion method is used to simulate the stress disturbance using the eigenstress resulting from an inclusion which occupies the space Ω. Let's introduce an arbitrary eigenstrain ϵ_{ij}^* in order to simulate the inhomogeneity problem. For a material that is homogenous everywhere but contains an eigenstrain ϵ_{ij}^* in a subdomain, Hooke's law yields:

$$\sigma_{ij}^0 + \sigma_{ij} = C_{ijkl} \left(u_{k,l}^0 + u_{k,l} - \epsilon_{kl}^*\right) \quad in \ \Omega \tag{A.3}$$

$$\sigma_{ij}^0 + \sigma_{ij} = C_{ijkl} \left(u_{k,l}^0 + u_{k,l}\right) \quad in \ D - \Omega \tag{A.4}$$

where $\sigma_{ij}^0 = C_{ijkl} u_{k,l}^0$.

The necessary and sufficient condition for the equivalency of the stresses and strains in the above two problems of inhomogeneity and inclusion is

$$C_{ijkl} \left(u_{k,l}^0 + u_{k,l}\right) = C_{ijkl} \left(u_{k,l}^0 + u_{k,l} - \epsilon_{kl}^*\right) \quad in \ \Omega \tag{A.5}$$

or

$$C_{ijkl} \left(u_{k,l}^0 + u_{k,l}\right) = C_{ijkl} \left(\epsilon_{kl}^0 + \epsilon_{kl} - \epsilon_{kl}^*\right) \quad in \ \Omega \tag{A.6}$$

ϵ_{kl} in the above equation can be obtained as a known function of ϵ_{kl}^* when the eigenstrain problem in the homogeneous material is solved.

$$\epsilon_{kl} = \frac{1}{2}(u_{k,l} + u_{l,k}) = S_{klmn}\epsilon_{nm}^* \tag{A.7}$$

where S_{klmn} are factors that depend on the Poisson's ratio of the matrix and the geometry of the ellipsoid. They are tabulated in Mura (1987, p. 80). Substitution of (A.7) into (A.6) leads to

$$C^*_{ijkl}\left(\epsilon^0_{kl} + S_{klmn}\epsilon^*_{nm}\right) = C_{ijkl}\left(\epsilon^0_{kl} + S_{klmn}\epsilon^*_{nm} - \epsilon^*_{kl}\right) \tag{A.8}$$

from which the six unknowns, ϵ^*_{ij}, are determined for a given ϵ^0_{kl}. The stress $\sigma^0_{ij} + \sigma_{ij}$ can then be found from (A.1), (A.2) or (A.3), (A.4).

A printed version of the Eshelby program can be found in appendix C.

References

Eshelby, J. D., The determination of the elastic field of an ellipsoidal inclusion, and related problems., Proc. Roy. Soc. London., Ser. A, **241**, 376-396, 1957.

Mura, T., *Micromechanics of defects in solids*, Martinus Nijhoff, Boston, 1987.

Appendix B

Block rotation equations

B.1 Stress limits

According to the Coulomb criterion for sliding, a fault can slip if:

$$\tau_f = C_f + \mu_f \sigma_n \tag{B.1}$$

where τ_f is the shear stress on the fault plane, σ_n the normal stress, C_f the roughness or cohesion and μ_f the friction coefficient. Prior to slip and rotation, it is necessary to check that the Coulomb fracture line is not exceeded:

$$\tau_0 = C_0 + \mu_0 \sigma_n \tag{B.2}$$

where τ_0 is the shear strength of the intact crust, C_0 the cohesion and μ_0 the friction coefficient. This line defines the upper limit to the magnitude of the maximum shear stress, T_{max}, in the Earth's intact crust:

$$T_{max} = \frac{\sigma_1 - \sigma_3}{2} \leq \frac{1}{\sqrt{1 + \mu_0^2}}[C_0 + \frac{\mu_0}{2}(\sigma_1 + \sigma_3)] \tag{B.3}$$

Furthermore, it is necessary to ensure that:

$$\sigma_3 \geq 0 \tag{B.4}$$

because the brittle crust cannot support tension.

B.2   Stress models and stress paths

The normal $[\sigma_n]$ and shear $[\tau]$ stresses across a plane are computed according to the following relations (Jaeger and Cook, 1969):

$$1 = \alpha_1^2 + \alpha_2^2 + \alpha_3^2 \tag{B.5}$$

$$\sigma_n = \sigma_1 + \alpha_2^2(\sigma_2 - \sigma_1) + \alpha_3^2(\sigma_3 - \sigma_1) \tag{B.6}$$

$$\tau^2 = \sigma_1^2 + \alpha_2^2(\sigma_2^2 - \sigma_1^2) + \alpha_3^2(\sigma_3^2 - \sigma_1^2) - \sigma_n^2 \tag{B.7}$$

where $\alpha_1, \alpha_2, \alpha_3$ are the direction cosines of the pole (fault normal) with respect to the principal stress axis, σ_1, σ_2, and σ_3.

As principal stress magnitudes change to satisfy equation (B.1), the initial value $[\sigma_n, \tau]$ on a given fault plane will change to a new value $[\sigma_n', \tau']$. The sequence of $[\sigma_n', \tau']$ described by the pole is computed with $\alpha_1, \alpha_2, \alpha_3 = constant$, because during these stress changes and before any rotation event occurs, the fault plane itself remains fixed in stress space.

This sequence of $[\sigma_n', \tau']$ will be referred to as the stress path of a fault. It depends on the stress history chosen and on the initial conditions (σ_n, τ, σ_1, σ_2, σ_3, α_1, α_2, α_3).

As explained in the text, this paper considers three stress regimes in which one of the principal stress axis is always assumed vertical (S_v) and stresses are assumed to change according to a $\phi = (\sigma_2' - \sigma_3')/(\sigma_1' - \sigma_3') = constant$ model. For each stress regime, the stress path of a fault is computed as follows:

NORMAL STRESS REGIME ($S_v = \sigma_1$)

$$\sigma_1' = \sigma_1 = constant \tag{B.8}$$

$$\sigma_2' = \sigma_2 - (1 - \phi)\Delta\sigma \tag{B.9}$$

$$\sigma_3' = \sigma_3 - \Delta\sigma \tag{B.10}$$

REVERSE STRESS REGIME ($S_v = \sigma_3$)

$$\sigma_1' = \sigma_1 + \Delta\sigma \tag{B.11}$$

$$\sigma_2' = \sigma_2 + \phi\Delta\sigma \tag{B.12}$$

$$\sigma_3' = \sigma_3 = constant \tag{B.13}$$

STRIKE-SLIP STRESS REGIME ($S_v = \sigma_2$)

$$\sigma_1' = \sigma_1 + \Delta\sigma \tag{B.14}$$

$$\sigma_2' = \sigma_2 = constant \tag{B.15}$$

$$\sigma_3' = \sigma_3 - \frac{\phi}{1-\phi}\Delta\sigma \tag{B.16}$$

Substituting the values for σ_i' into equations (B.6) and (B.7), the stress paths that result from the above stress models are given by the following equations with $\Delta\sigma$ as a parameter:

$$\sigma_n' = \sigma_n + a\Delta\sigma \tag{B.17}$$

$$\tau'^2 = \tau^2 + b\Delta\sigma + c(\Delta\sigma)^2 \tag{B.18}$$

where $\sigma_1, \sigma_2, \sigma_3, \tau, \sigma_n$ are known initial values and $\alpha_1, \alpha_2, \alpha_3$ are the constant direction cosines of the fault normal. The coefficients a, b and c assume different values depending on the stress regime considered (see Table B1).

Replacing $\Delta\sigma$ from (B.17) into (B.18), we get an explicit equation for the stress path

$$a^2\tau'^2 = A\sigma_n'^2 + B\sigma_n' + C, \tag{B.19}$$

where

$$A = c \tag{B.20}$$

$$B = ab - 2c\sigma_n \tag{B.21}$$

$$C = a^2\tau^2 - ab\sigma_n + c\sigma_n^2 \tag{B.22}$$

111

By combining equation (B.19) and equation (B.1), we obtain the coordinates $[\sigma'_n, \tau']$ of the intersection between the pole's stress path and the Coulomb sliding line. At this intersection point the fault plane can slip and rotate.

B.3   Rotation of fault planes

Consider a pole with direction cosines α_1, α_2, α_3 with respect to the principal stress axis. Once the shear stress, τ, across the fault plane satisfies equation (B.1), the fault plane will slip and rotate in the direction of the resolved shear stress, represented by the unitary vector $\tilde{\tau}$. [A tilde denotes a unitary vector]. The nature of this rotation is better understood in Figure B1, where the horizontal plane represents the fault before rotation and $\tilde{r}$ is its unit normal vector (same direction as $\vec{\sigma}_n$). Given the principal components of stress $[\vec{\sigma}]_i$ acting on the fault plane

$$[\vec{\sigma}]_1 = \alpha_1\sigma_1 \tag{B.23}$$

$$[\vec{\sigma}]_2 = \alpha_2\sigma_2 \tag{B.24}$$

$$[\vec{\sigma}]_3 = \alpha_3\sigma_3 \tag{B.25}$$

we can compute the coordinates of the rotation vector $\tilde{n}$, the normal to the fault plane $\tilde{r}$ and $\tilde{\tau}$. In order to compute $\tilde{n}$ we first compute vector $\vec{n} = \tilde{r} \times \vec{\sigma}$ with components of shear stress:

$$n_1 = \alpha_2\alpha_3(\sigma_3 - \sigma_2) \tag{B.26}$$

$$n_2 = \alpha_1\alpha_3(\sigma_1 - \sigma_3) \tag{B.27}$$

$$n_3 = \alpha_1\alpha_2(\sigma_2 - \sigma_1) \tag{B.28}$$

Vector $\tilde{n}$ is then simply $\vec{n}/|\vec{n}|$. Similarly, $\vec{\tau} = \vec{n} \times \tilde{r}$ and has components:

$$\tau_1 = \alpha_1\{\alpha_3^2(\sigma_1 - \sigma_3) - \alpha_2^2(\sigma_2 - \sigma_1)\} \tag{B.29}$$

$$\tau_2 = \alpha_2\{\alpha_1^2(\sigma_2 - \sigma_1) - \alpha_3^2(\sigma_3 - \sigma_2)\} \tag{B.30}$$

$$\tau_3 = \alpha_3\{\alpha_2^2(\sigma_3 - \sigma_2) - \alpha_1^2(\sigma_1 - \sigma_3)\} \tag{B.31}$$

		Coefficients	
S_v	a	b	c
σ_1	$-(\alpha_3^2$ $+(1-\phi)\alpha_2^2)$	$-2[(1-\phi)\alpha_2^2(\sigma_2-\sigma_n)$ $+\alpha_3^2(\sigma_3-\sigma_n)]$	$(\phi\alpha_2\alpha_3)^2$ $+(\alpha_3^2+(1-\phi)^2\alpha_2^2)\alpha_1^2$
σ_3	$\alpha_1^2+\phi\alpha_2^2$	$2[\alpha_1^2(\sigma_1-\sigma_n)+\phi\alpha_2^2(\sigma_2-\sigma_n)]$	$(1-\phi)^2\alpha_1^2\alpha_2^2$ $+(\alpha_1^2+\phi^2\alpha_2^2)\alpha_3^2$
σ_2	$\alpha_1^2-\frac{\phi}{1-\phi}\alpha_3^2$	$2[\alpha_1^2(\sigma_1-\sigma_n)-\frac{\phi}{1-\phi}\alpha_3^2(\sigma_3-\sigma_n)]$	$\alpha_1^2+(\frac{\phi}{1-\phi})^2\alpha_3^2-a^2$

Table B.1. a, b and c are coefficients required to calculate the stress path of a fault. They depend on the orientation of the fault (α_1, α_2, α_3), the magnitude of the vertical stress, S_v, the assumed stress model (in this case ϕ=constant), and the initial values of σ_1, σ_2 and σ_3.
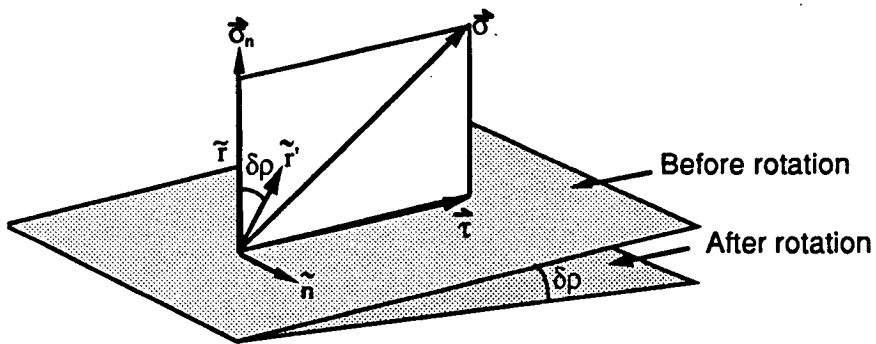
Fig. B.1 Rotation of fault plane. The horizontal plane, with its normal vector $\tilde{\mathbf{r}}$, represents the initial orientation of the fault. The dipping plane, with its normal vector $\tilde{\mathbf{r}}'$, represents the final orientation of the fault, after a rotation $\delta\rho$. Rotation is imposed by rotating the normal $\tilde{\mathbf{r}}$ through an angle $\delta\rho$ within the plane with normal $\tilde{\mathbf{n}}$, that contains $\vec{\sigma}_n$, $\tilde{\mathbf{r}}$, $\vec{\sigma}, \vec{\tau}$, where $\vec{\sigma}_n$ is the normal stress and $\vec{\tau}$ is the shear stress resolved on the initial orientation of the fault.

The direction cosines of the plane after a rotation $\delta\rho$ will be the components of the vector $\tilde{r}'$. Therefore, they must satisfy the following 3 equations:

$$\tilde{r}' \cdot \tilde{n} = 0 \implies \alpha_1' n_1 + \alpha_2' n_2 + \alpha_3' n_3 = 0 \tag{B.32}$$

$$\tilde{r}' \cdot \tilde{r} =: \cos \delta\rho \implies \alpha_1' \alpha_1 + \alpha_2' \alpha_2 + \alpha_3' \alpha_3 = \cos \delta\rho \tag{B.33}$$

$$\tilde{r}' \cdot \tilde{\tau} = \tilde{r}' \cdot \vec{\tau} / |\vec{\tau}| =: \sin \delta\rho \implies \alpha_1' \tau_1 + \alpha_2' \tau_2 + \alpha_3' \tau_3 = \sin \delta\rho \cdot |\vec{\tau}| \tag{B.34}$$

After rotation has occurred, principal stresses will drop and Mohr circles should "shrink". The newly computed values of the rotated fault plane direction cosines $(\alpha_1', \alpha_2', \alpha_3')$ define a new stress path, along which the pole must move as the stresses change (after each rotation we can choose a different stress model). No direct relationship exists yet between the amount of stress drop and the amount of rotation. Nonetheless, irrespective of the stress drop, the new pole must move along its stress path. Since the rotation path is fully defined by the intersection between the stress path and the sliding line, it is not necessary to know the stress drop (that is, how far the pole will move along its new path) to estimate the next rotation event.

A printed version of the block rotation program can be found in appendix C.

References

Jaeger, J. C., and N. G. W. Cook, *Fundamentals of rock mechanics*, Methuen and Co. Ltd, London, 515pp, 1969.

Appendix C

Computer programs

C.1 Faultsvp

The FAULTSVP program (FaultSlipVisualizationPlot) is a useful tool to visualize on a Mohr circle whether a family of faults can be reactivated in a given stress field for given material parameters (such as friction and cohesion values of intact crust and preexisting faults). This is not a stress inversion program, it is simply a plotting program (written in the C-language) aimed at visualizing stress and faulting in a 3D stress field. It may be used in conjuction with stress inversion programs.

Given a stress field orientation and dip and strike of faults, the program computes the normal and shear stress components resolved on each fault. On a Mohr circle, only faults that plot between the upper line (strength of the crust) and the lower line (strength of preexisting faults) can slip, otherwise they are locked. The slip direction (rake) for each fault is computed. A mapping between Mohr circle and Wulff projection is also provided. The program assumes that one principal stress orientation is vertical.

```
------------------------------------------------------------------
                  /*  Beginning of program FaultSVP  */
------------------------------------------------------------------
```

/*This program plots faults [strike,dip==> sn,tau] on 3D Mohr-circle
  and on a Wulff stereo projection. It calculates the rake value as
  well.

VARIABLES:

  anggh(i)   - Angles between pole and Wulff axes in geographic
                  space.i=0,1,2 correspond to the x,y and depth
                  direction respectively.
  angstr(i)   - Angles between pole and Stress Wulff Projection
                  axes.
  cosang(i)   - Auxiliary variable. Contains pole's direction
                  cosines
                  in relation to Stress and Space Wulff axes.
  ddirgh,dipgh - Pole's Dip Direction and Dip in Space Projection.
  ddirst,dipst - Pole's Dip Direction and Dip in Stress Projection.
  sigmadir(i,j)- Matrix with principal stresses's direction cosines.
                  i,j =0,1,2.
  csangsq(i)   - squares of direction cosines. i=0,1,2.
  angshr(i)   - angles corresponding to direction cosines of shear
  csangsh(i)   - direction cosines of shear direction.i=0,1,2.
  sig(i)   - principal stress magnitudes. i=0,1,2.
  signor,tau   - normal and shear stress resolved on a fault.
  sx,sy,sz   - S_hmin,S_Hmax and S_vert assumed to be
                  principal stress axes.
  cl,cm,cn   - plane direction cosines
  trend,plunge - trend and plunge of shear direction on fault plane

INPUT FILES:

        in=faultfile

        dipdirection dip
        dipdirection dip

        (maximum of faults = 10)

        par=parfile

        sigmavert=1800   vertical stress
        sigmahmin=600   minimum horizontal stress

                              117

```
         sigmaHmax=3000    maximum horizontal stress
          azimuthhmin=90    azimuth from NORTH of min horizontal stress
        rockfriction=1.0
        faultfriction=0.6
        rockcohesion=1000
        faultcohesion=50
```

OUTPUT FILES:

```
    filenameroot.mohr  - contains sig,tau points to be plotted
                        on a Mohr circle.
    filenameroot.onnet - contains dipdirection,dip of poles to
                        be plotted on Wulff-projection.
    filenameroot.rake  - contains rakes of faults.
  filenameroot.print - prints values (for stresses ,dips,
                          dipdir, and material parameters) used
                          in the program.
```

REQUIRED SUBROUTINES:

```
    draw_mohrcrc(sig,&signormohr,&sigtaumohr)
    do_geotostr(ipole,sigmadir,&ddirst,&dipst,ddirgh,dipgh,angstr,
              0,1,2,&irev);
    Calculate_rake(cl,cm,cn,sx,sy,sz,&newparameter->rake[i]);
    Calculate_shear(sig,csangsh,cosang,csangsq);
    do_wlfcord(iwucrd,angshr,csangsh[1],trend,0,1,2,irev1);
    do_strtogeo(ipole,sigmadir,ddirst,dipst,ddirgh,dipgh,angstr,
              anggh,0,1,2,irev);
    Calculate_sigtau(sig,alfa1,alfa2,alfa3,signor,tau);
    Print_parameters(sx,sy,sz,newparameter,fp1,fp2,fp3,fp4);
```

```c
char *documentation[] = {
" NAME",
"        fsv",
"",
" syntax",
"        fsv [inputfiles] ",
"",
"    inputfiles:",
"            in=filename      filename is the fault's input file",
"            par=filename     filename is the parameter file",
"            [out=filename]   filename is the file to /tmp/output",
"",
"",
```

```
""
};
int doclength = { sizeof documentation/sizeof documentation[0] };


/*   DIMENSION AND INITIALIZE VARIABLES */

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <sys/ioctl.h>
#include <sgtty.h>
#include "defs.h"
#include "parameter.h"

char string[80],filenameroot[80],string1[10];
char string2[10],string3[10],string4[10];
FILE *controltty;
FILE *out_file;
FILE * fp1;
FILE * fp2;
FILE * fp3;
FILE * fp4;
struct sgttyb ttystat;
static char *name="fsv";

/* for getpar */
int xargc;
char **xargv;

main(argc,argv)
int argc;
char *argv[];
{
  Parameter * newparameter;
  FILE *temp;
  register char *cptr;
  char c;
  int i,piped_in, piped_out, controlfd;

/*
 * If no arguments, and not in a pipeline, self document
 */
piped_in = ioctl ((fileno (stdin)), TIOCGETP, &ttystat);
piped_out = ioctl ((fileno (stdout)), TIOCGETP, &ttystat);
```

```
if (argc == 1 && !piped_in)
{
for( i=0; i<doclength; i++)
printf("%s\n",documentation[i]);
exit (0);
}

/*
 * determine whether commands are coming from the terminal
 *     (Unix only)
 */
controlfd = open("/dev/tty",0);
controltty = fdopen(controlfd,"r");

xargc = argc;
xargv = argv;

/*
 * process in= inputfiles
 */
if(getpar("in","s",string))
{
sscanf(string,"%s",filenameroot);
if((temp=fopen(filenameroot,"r")) != NULL){

                newparameter = read_parameter(temp);
        }
else
   {
fprintf(stderr,"%s: cannot open %s",name,filenameroot);
                exit(1);
  }
        }
/*
 * next process out= outputfiles
 */
if(getpar("out","s",string))
{
sscanf(string,"%s",filenameroot);
        }

/* OPEN OUTPUT FILES WITH NAME OF OUTPUT FILE OR NAME OF
   INPUT FILE IF OUTPUT FILE NOT GIVEN
 */
```

120

```c
        strcpy(filenameroot,string);
        strcpy(string1,".onnet\0");
        strcpy(string2,".mohr\0");
        strcpy(string3,".rake\0");
        strcpy(string4,".print\0");

        strcat(filenameroot,string1);
        fp1 = fopen(filenameroot,"w");
        strcpy(filenameroot,string);
        strcat(filenameroot,string2);
        fp2 = fopen(filenameroot,"w");
        strcpy(filenameroot,string);
        strcat(filenameroot,string3);
        fp3 = fopen(filenameroot,"w");
        strcpy(filenameroot,string);
        strcat(filenameroot,string4);
        fp4 = fopen(filenameroot,"w");

        getpar("sigmavert","g",&(newparameter->sigma_[2]));
        getpar("sigmahmin","g",&(newparameter-> sigma_[0]));
        getpar("sigmaHmax","g",&(newparameter-> sigma_[1]));
        getpar("azimuthhmin","g",&(newparameter-> azimuth_[0]));
        getpar("rockfriction","g",&(newparameter-> rock_friction));
        getpar("faultfriction","g",&(newparameter-> fault_friction));
        getpar("rockcohesion","g",&(newparameter-> rock_cohesion));
        getpar("faultcohesion","g",&(newparameter-> fault_cohesion));

                if(newparameter->sigma_[0] > newparameter->sigma_[1])
                {
                  fprintf(stderr,"\n sigmahmin(%f) > sigmaHmax(%f),
                                  program %s stops\n",
                                  newparameter->sigma_[0],
                                  newparameter->sigma_[1],name);
                  exit(1);
                }

            do_faults(newparameter);

}

do_faults(newparameter)
Parameter * newparameter;
{

  double cosang[3],angstr[3],anggh[3],sigmadir[3][3];
```

```c
      double csangsq[3],angshr[3],csangsh[3],sig[3];
      double sigma_vert,sigma_hmin,sigma_Hmax,azimuth_hmin;
      double sx,sy,sz,cl,cm,cn,ddirgh,dipgh,dipst,ddirst;
      double trend,plunge,signor,tau;
      int nfault,l,m,k;

      sx = newparameter-> sigma_[0];
      sy = newparameter-> sigma_[1];
      sz = newparameter-> sigma_[2];


/*------------------------------------------------------------------*/

      /* CALCULATE DIRECTION COSINES OF PRINCIPAL STRESS AXIS */

      sigmadir[2][0]= 0.;
      sigmadir[2][1]= 0.;
      sigmadir[2][2]= -1.;
      sigmadir[0][0]= cos(newparameter->azimuth_[0]*RAD);
      sigmadir[0][1]= sin(newparameter->azimuth_[0]*RAD);
      sigmadir[0][2]= 0. ;

      sigmadir[1][0]= sigmadir[2][1]*sigmadir[0][2] -
                      sigmadir[2][2]*sigmadir[0][1] ;
      sigmadir[1][1]= sigmadir[2][2]*sigmadir[0][0] -
                      sigmadir[2][0]*sigmadir[0][2] ;
      sigmadir[1][2]= sigmadir[2][0]*sigmadir[0][1] -
                      sigmadir[2][1]*sigmadir[0][0] ;

      newparameter->azimuth_[1] = newparameter->azimuth_[0] + 90. ;
      newparameter->azimuth_[2] = 0.;
      newparameter->dip_[0] = 0.;
      newparameter->dip_[1] = 0.;
      newparameter->dip_[2] = 90.;

/*------------------------------------------------------------------*/
/* FOR EACH FAULT  */
   for (nfault=0; nfault<newparameter->numfaults; nfault++){

      fprintf(stderr,"\n doing fault # %d \n",nfault);

/*------------------------------------------------------------------*/

/* (1) CHANGE FROM GEOGRAPHIC TO PRINCIPAL STRESS COORDINATES */
```

```c
      ddirgh = newparameter->dipdir_fault[nfault];
      dipgh = newparameter->dip_fault[nfault];

      do_geotostr(sigmadir,ddirgh,dipgh,angstr);

/*------------------------------------------------------------------*/

/*  (2) CALCULATE SN,TAU */

      for (k=0; k<3 ; k++)     cosang[k] = cos(angstr[k]);

      csangsq[0] = cosang[0]*cosang[0];
      csangsq[1]= cosang[1]*cosang[1];
      csangsq[2]= cosang[2]*cosang[2];

      for(k=0; k<3; k++) sig[k]= newparameter-> sigma_[k];

      Calculate_sigtau(sig,csangsq[0],csangsq[1],csangsq[2],
                    &signor,&tau);

      newparameter->signor_fault[nfault] = signor;
      newparameter->sigtau_fault[nfault] = tau;

/*------------------------------------------------------------------*/

/*  (3) CALCULATE THE RAKE VALUE */

        for (m=0 ; m<3; m++){
if ( sx IS newparameter-> sigma_[m]) cl = cos(angstr[m]);
if ( sy IS newparameter-> sigma_[m]) cm = cos(angstr[m]);
if ( sz IS newparameter-> sigma_[m]) cn = cos(angstr[m]);
        }

      Calculate_rake(cl,cm,cn,sx,sy,sz,
                    &newparameter->rake[nfault]);
/*------------------------------------------------------------------*/

/*  (4) CALCULATE THE DIRECTION OF FAULT SLIP */


      if (csangsq[0] IS 1 OR csangsq[1] IS 1 OR csangsq[2] IS 1){
        fprintf(stderr,"\n fault %d has no shear resolved on it \n",
                    nfault);
        fprintf(stderr,"\n program continues to next fault \n");
        continue;
```

```
    }

    Calculate_shear(sig,csangsh,cosang,csangsq);
/*-------------------------------------------------------------*/

/* (5) COMPUTE ANGLES CORRESPONDING TO SHEAR DIRECTION COSINES */

    for(m=0; m<3; m++){
      angshr[m]= acos(csangsh[m]);
    }

/* (6) COMPUTE SHEAR STRESS DIRECTION'S IN STRESS WULFF COORD:
       trend & plunge*/

    do_wlfcord(angshr,csangsh,&trend,&plunge);

/*-------------------------------------------------------------*/

/* (7) CHANGE BACK FROM STRESS TO GEOGRAPHIC COORDINATES  */
/*      TO PLOT SHEAR DIRECTION ON A STEREOPROJECTION       */

    ddirst = trend ;
    dipst = plunge ;

    do_strtogeo(sigmadir,ddirst,dipst,
&newparameter->dipdir_shear[nfault],
&newparameter->dip_shear[nfault]);

/*-------------------------------------------------------------*/
  }

/* PRINT FILES: a. filenameroot.mohr
                b. filenameroot.rake
c. filenameroot.onnet
                d. filenameroot.print
*/

  print_parameters(sx,sy,sz,newparameter,fp1,fp2,fp3,fp4);
}

Calculate_sigtau(sig,alfa1,alfa2,alfa3,signor,tau)
double sig[3],alfa1,alfa2,alfa3;
double *signor,*tau;
{
  double sigsq,tausq;
```

124

```
*signor = (sig[0]*alfa1) + (sig[1]*alfa2) + (sig[2]*alfa3) ;
sigsq  = (*signor)*(*signor);
tausq = (alfa1*sig[0]*sig[0])+
        (alfa2*sig[1]*sig[1])+(alfa3*sig[2]*sig[2]) ;
tausq = tausq - sigsq;
*tau = sqrt(ABS(tausq)) ;

return;
}
```

```
------------------------------------------------------------------
                /* Beginning of subroutine READ_PARAMETER */
------------------------------------------------------------------

/* This subroutine reads the input file, containing the following
   information:

        dipdir  dip  } for every fault

     and defaults for material parameters

        sigma_vert=1800      vertical stress
        sigma_hmin=600       minimum horizontal stress
        sigma_Hmax=3000    maximum horizontal stress
        azimuth_hmin=90       azimuth of minimum horizontal stress
        rock_friction=1.0
        fault_friction=0.6
        rock_cohesion=1000
        fault_cohesion=50

VARIABLES: this    - is a pointer to a structure contained in
                     parameter.h

----------------------------------------------------------------*/
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "defs.h"
#include "parameter.h"

extern char * malloc() ;

Parameter * read_parameter(fd)
FILE * fd ;
{
   char dipdir[20], dip[20] ;
   char line[512] ;
   int nfault=0;
   Parameter * this ;

   this = DefaultParameter() ;

   while( fgets(line,sizeof(line),fd) != nil ) {

      if( line[0] == '#' ) continue ;
      if( line[0] == '\n' ) continue ;
```

126

```
        sscanf(line,"%s %s",dipdir,dip) ;
    this-> dipdir_fault[nfault] = atof(dipdir) ;
    this-> dip_fault[nfault] = atof(dip) ;
    nfault++;
        }

    this->numfaults = nfault;

    return this ;
 }


Parameter * DefaultParameter()
{
    int i ;
    Parameter * this = (Parameter *)malloc(sizeof(Parameter));

    this->sigma_[0]= SIGMA_3    ;
    this->sigma_[1]= SIGMA_1    ;
    this->sigma_[2]= SIGMA_2    ;

    this->rock_friction = ROCK_FRICTION ;
    this->fault_friction = FAULT_FRICTION ;
    this->rock_cohesion = ROCK_COHESION  ;
    this->fault_cohesion =  FAULT_COHESION ;

    return this ;
}


---------------------------------------------------------------------
```

```
------------------------------------------------------------------
                /* Beginning of subroutine GEOTOSTRESS */
------------------------------------------------------------------
/* This program maps poles  between a Wulff Projection with
   Principal Stresses along its axes and another Wulff Projection
   with geographic N-E-Z axes.

                               VARIABLES
                         ================
 VARIABLES:

   anggh(i)      - Angles between pole and Wulff axes in geographic
                   space.
                   i=1,2,3 correspond to the north, east and depth
                   direction respectively.
   angstr(i)     - Angles between pole and Stress Wulff Projection
               axes.
   cosang(i)     - Auxiliary variable. Contains pole's direction
                   cosines in relation to Stress and Space Wulff
                   axes, depending on iflag.
   ddirgh,dipgh - Pole's Dip Direction and Dip in the Usual Space
                   Projection.
   0,1,2         - Principal Stress oriented in the north, east and
                   depth directions in a Stress Wulff stereographic
                   projection.
   sigmadir(i,j)- Matrix with principal stresses's direction
                   cosines. i,j = 1,2,3.

  INPUT VARIABLES: sigmadir ddirgh,dipgh

  OUTPUT VARIABLES: angstr,anggh AND ddirgh,dipgh

  REQUIRED SUBROUTINES: axesrot(iflag,x,sigmadir)

-----------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include "defs.h"
#include "parameter.h"

do_geotostr(sigmadir,ddirgh,dipgh,angstr)
double angstr[3],sigmadir[3][3];
double ddirgh,dipgh;
{
  double cosang[3],theta,sin3,sinz,anggh[3];
```

128

```c
   int i;
/* COMPUTE DIRECTION COSINES AND CORRESPONDING ANGLES */
/*  MAPPING POLES */

   theta = (ddirgh - 180.) * RAD;
   anggh[2] = dipgh * RAD ;


/* COMPUTE ANGLES AND DIRECTION COSINES */

   sin3 = sin(anggh[2]);
   cosang[2] = cos(anggh[2]);
   cosang[0] = sin3 * cos(theta);
   cosang[1] = sin3 * sin(theta);
   anggh[0] = acos(cosang[0]) ;
   anggh[1] = acos(cosang[1]) ;


/*  ROTATE AXES ACCORDING TO SIGMADIR . COMPUTE DIRECTION COSINES
    IN STRESS COORDINATES
*/

  axesrot(1,cosang,sigmadir) ;


/* IF THE NEW POLE'S RADIUS VECTOR POINTS UP, TAKE THE OPPOSITE
   TO THE SAME   PLANE (SINCE WE ARE WORKING WITH LOWER
   HEMISPHERE WULFF PROJECTION)
*/

   if(cosang[2] < -0.001){
     for(i=0 ; i<3; i++){
       cosang[i] = -cosang[i];
     }
   }
/*  COMPUTE ANGLES IN STRESS COORDINATES   */

   for (i=0; i< 3 ;i++)   angstr[i] = acos(cosang[i]);
   return;
 }
```

```
-----------------------------------------------------------------
                  /* Beginning of subroutine AXESROT */
-----------------------------------------------------------------
/* This subroutine computes coordinates's transformations during
   pure rotations

  VARIABLES:

  alpha(i,j) - transformation matrix with the direction cosines of
               the new axis
               i=1,2,3 corresponds to new axis X',Y',Z'respectively,
               and j=1,2,3 corresponds to its direction cosines.
  iflag - = +1 for direct and =-1 for inverse transformations
  x(i) - i=1,2,3 x,y,z coordinates for a single point. On input
               contains the original coordinates, on output - the
               transformed ones.
  s(i) - Auxiliary (temporary) variable. Same as x(i).

  INPUT VARIABLES: iflag,alpha,x

  OUTPUT VARIABLES: x
-----------------------------------------------------------------
*/

#include <stdio.h>
#include <math.h>
#include "parameter.h"
#include "defs.h"

axesrot(iflag,x,alpha)
int iflag;
double x[3],alpha[3][3];
{

  int i,j;
  double s[3];

/*  DIRECT OR INVERSE TRANSFORMATION (ROTATION) ? */

  if(iflag IS_NOT 1 AND iflag IS_NOT -1){
   fprintf(stderr,"\n IFLAG has improper value in subroutine axesrot \n");
   exit(1);
  }

  if(iflag IS 1){
```

130

```
c                          DO INVERSE TRANSFORMATION
c----------------------------------------------------------------------c
   do 20 i=1,3
         s(i) = 0.
   do 20 j=1,3
s(i) = s(i) + alpha(j,i) * x(j)
20    continue
endif
c----------------------------------------------------------------------c
c              PUT RESULTS (TRANSFORMED COORDINATES) IN VECTOR x(i)
c----------------------------------------------------------------------c
do 30 i = 1,3
      x(i) = s(i)
30 continue
return
end
c----------------------------------------------------------------------c
```

```c
/*  DO DIRECT TRANSFORMATION */

    for(i=0; i<3 ; i++){
      s[i] = 0.;
      for(j=0; j<3 ; j++){
        s[i] = s[i] + alpha[i][j] * x[j];
      }
    }
  } else {

/*  DO INVERSE TRANSFORMATION */

    for(i=0; i<3 ; i++){
      s[i] = 0.;
      for(j=0; j<3 ; j++){
s[i] = s[i] + alpha[j][i] * x[j];
      }
    }
  }

/*  PUT RESULTS (TRANSFORMED COORDINATES) IN VECTOR x(i) */


    for(i=0; i< 3 ; i++) x[i] = s[i] ;
    return;
}
```

```
------------------------------------------------------------------------
                /* Beginning of subroutine CALCULATE_RAKE */
------------------------------------------------------------------------
/*  This subroutine calculates the rake, or the slip direction of
    a fault using Bott's equation (1959).

VARABLES:

cl,cm,cn   direction cosines of the plane;
sx,sy,sz   S_hmin,S_Hmax and S-vert are the magnitudes of the
           horizontal  and vertical stress axis (here coincident
           with the principal stress axis).
rake       slip direction of fault;

INPUT VARIABLES: cl,cm,cn,sx,sy,sz;

OUTPUT VARAIBLES: rake

REFERENCES: Bott, M.H.P., 1959, Geol.Mag.,vol XCVI no. 2
------------------------------------------------------------------------
*/

#include <stdio.h>
#include <math.h>
#include "parameter.h"
#include "defs.h"

Calculate_rake(cl,cm,cn,sx,sy,sz,rake)
double cl,cm,cn,sx,sy,sz,*rake;
{
  char * sense[1];

  strcpy(sense,"l");
  if (cl > 0 AND cm > 0)  strcpy(sense,"r");
  if (cl < 0 AND cm < 0)  strcpy(sense,"r");

  cl = ABS(cl);
  cm = ABS(cm);
  cn = ABS(cn);

 if(ABS(cl)IS 1 OR ABS(cm)IS 1 OR ABS(cn) IS 1){
    *rake= 9999.;
  } else if( sz > sy AND sy IS sx){
    *rake = 90.;
  } else if( sx > sz AND sy IS sx){
```

```
    *rake = -90.;
  } else if(ABS(cl) < 0.000001){
    if(sz > sy) *rake = 90.;
    if(sz < sy) *rake = -90.;
  } else if(ABS(cm) < 0.000001){
    if(sz > sx) *rake = 90.;
    if(sz < sx) *rake = -90.;
  } else {
    *rake  = (1. - cn*cn)*(sz - sx)/(sy - sx);
    *rake = cn*(cm*cm - *rake)/(cl*cm);
    *rake = atan(*rake);
    *rake = -((*rake) * DEG);
  }

/* CONVENTION:
   0 is left-lateral , 180 is right-lateral, 90 is normal,
   -90 is reverse
*/
  if (!strcmp(sense,"r") AND (*rake IS_NOT 9999.)){
       if ((*rake) > 0) (*rake) = 180. - (*rake);
       if ((*rake) < 0) (*rake) = -(180. + (*rake));
       if ((*rake) < -180.) (*rake) = (*rake) + 180.;
       if ((*rake) > 180.) (*rake) = (*rake) - 180.;
  }


  return;
}
-----------------------------------------------------------------
```

```
-----------------------------------------------------------------------
                    /* Beginning of subroutine SHEAR */
-----------------------------------------------------------------------
/*  This subroutine computes the coordinates [csangsh(1,2,3)-
    direction cosines in relation to principal stresses] of the
    maximum shear stress vector on a plane defined by its pole's n
    directiocosines [cosang(1,2,3)].


                              VARIABLES
                         =================


    sig1,2,3      - Principal stresses modulii
    cosang(i)     - Pole's direction cosines in relation to Principal
                    Stresses  (cosines of alpha) (i=1,2,3)
    csangsh(i,j)  - Direction cosines of the maximum shear stress on
                    the fracture's plane in relation to principal
                    stresses axes. (cosines of angshr) (i=1,2,3;
                    j=1,irot)
    csangsq(i)    - Cosines of alpha squared (cosang**2)


 INPUT VARIABLES: cosang, csangsq, sigma1, sigam2, sigma3


 OUTPUT VARIABLES: csangsh


 -----------------------------------------------------------------------*/


#include <stdio.h>
#include <math.h>
#include "parameter.h"
#include "defs.h"


Calculate_shear(sig,csangsh,cosang,csangsq)
double sig[3],csangsh[3],cosang[3],csangsq[3];
{

  double amodshr;
  int i;

/*  COMPUTE VECTOR PROPORTIONAL TO THE MAXIMUM SHEAR STRESS
*/

  csangsh[0] = cosang[0] * (csangsq[2]*(sig[0]-sig[2]) -
    csangsq[1]*(sig[1]-sig[0]));

  csangsh[1] = cosang[1] * (csangsq[0]*(sig[1]-sig[0]) -
```

134

```
        csangsq[2]*(sig[2]-sig[1]));

   csangsh[2] = cosang[2] * (csangsq[1]*(sig[2]-sig[1]) -
      csangsq[0]*(sig[0]-sig[2]));

/*   COMPUTE UNITARY VECTOR (DIRECTION COSINES) IN THIS DIRECTION */

 amodshr = sqrt(csangsh[0]*csangsh[0] + csangsh[1]*csangsh[1] +
 csangsh[2]*csangsh[2]);

 for(i=0 ; i <3 ; i++) csangsh[i] = csangsh[i]/amodshr;

   return;
}
------------------------------------------------------------------
```

```
----------------------------------------------------------------------
                 /* Beginning of subroutine WLFCORD   */
----------------------------------------------------------------------
/*  This subroutine computes Stress-Wulff coordinates "trend
    and plunge", given a vector's direction cosines.


                            VARIABLES
                          ================


    alpha(i)        - Angles between pole's radius vector and Principal
                      Axes of  Stresses (i=1,2,3)
    cosang(i)       - Pole's direction cosines in relation to Principal
                      Stresses (cosines of alpha) (i=1,2,3)
    ddir            - Pole's dip direction in Wulff's Projection
    dip             - Pole's dip in Wulff's Projection
    0,1,2           - Principal Stresses oriented in the north, east and
                      depth directions in a Stress Wulff stereographic
                      projection.
    plunge          - Line's (vector) plunge in Stress-Wulff Projection.
    sinz            - Sinus of alpha(istz).
    trend           - Line's (vector) trend in Stress-Wulff Projection.

  INPUT VARIABLES: alpha, cosang.

  OUTPUT VARIABLES: trend and plunge.


----------------------------------------------------------------------*/


#include <stdio.h>
#include <math.h>
#include "parameter.h"
#include "defs.h"

do_wlfcord(alpha,cosang,trend,plunge)
double alpha[3],cosang[3],*trend,*plunge;
{
  double theta,sinz,ddir,dip;
  int i;



/* IF THE VECTOR (POLE) IS POINTING TOWARDS THE UPPER HEMISPHERE,
   REVERSE IT.
*/

  if(cosang[2] < -0.0001){
```

```c
   for(i=0 ; i<3 ; i++){
     cosang[i] = -cosang[i];
     alpha[i] = acos(cosang[i]);
   }
 }

/* COMPUTE PLUNGE... */

  *plunge = 90. - alpha[2]* DEG;

/*  CHECK FOR POSSIBLE SINGULARITIES */

 sinz = sin(alpha[2]);

 if(ABS(sinz)<0.01 AND ABS(cosang[0])<0.01){

/* IF sinz = 0, trend AND ddir ARE UNDEFINED. WE ARBITRARILY SET
   THEN TO 0..
*/

  *trend = 0.;
  ddir = 0. ;

  } else {

  /*  ...OTHERWISE, COMPUTE THETA */

  if( (cosang[0]- sinz)*(cosang[0]- sinz) < 0.0001){
    theta = 0;
  } else {
    theta = atan2(cosang[1],cosang[0]);
    theta *= DEG;
  }
 }

/* COMPUTE TREND. IF ANGLES ARE BIGGER THAN 360, SUBSTRACT 360
*/
   *trend = theta ;
   if(*trend > 359.9) *trend = *trend - 359.9;

 return;
}
------------------------------------------------------------------
```

```
-----------------------------------------------------------------
                 /* Beginning of subroutine   STRESS-> GEO */
-----------------------------------------------------------------
```

/* This program maps lines between a Wulff Projection with Principal
   Stresses along its axes and another Wulff Projection with
   geographic N-E-Z axes.

                           VARIABLES
                     ================

VARIABLES:

    anggh(i)   - Angles between pole and Wulff axes in geographic
                    space. i=1,2,3 correspond to the north, east and
                    depth direction respectively.
    angstr(i)   - Angles between pole and Stress Wulff Projection
                    axes.
    cosang(i)   - Auxiliary variable. Contains pole's direction
                    cosines in relation to Stress and Space Wulff axes,
                    depending on iflag.
    ddirgh,dipgh - Pole's Dip Direction and Dip in the Usual Space
                    Projection.
    ddirst,dipst - Pole's Dip Direction and Dip in the Stress
                    Projection.
    0,1,2   - Principal Stresss oriented in the north, east and
                    depth directions in a Stress Wulff stereographic
                    projection.
    sigmadir(i,j)- Matrix with principal stresses's direction cosines.
                    i,j = 1,2,3.

  INPUT VARIABLES: sigmadir,0,1,2 AND ddirst,dipst
                   (iflag=1) OR ddirgh,dipgh (iflag=-1).

  OUTPUT VARIABLES: angstr,anggh AND ddirgh,dipgh.

  REQUIRED SUBROUTINES: axesrot(iflag,x,sigmadir)


```
-----------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include "defs.h"
#include "parameter.h"

do_strtogeo(sigmadir,ddirst,dipst,ddirgh,dipgh)
double sigmadir[3][3], ddirst,dipst,*ddirgh,*dipgh;
{
```

```c
    double cosang[3],theta,sin3,sinz,angstr[3],anggh[3];
    int i;

/* COMPUTE DIRECTION COSINES AND CORRESPONDING ANGLES */

/* MAPPING LINES   */

        theta = ddirst * RAD ;
        angstr[2] = (90. - dipst) * RAD;

/* COMPUTE ANGLES AND DIRECTION COSINES */

    sinz = sin(angstr[2]);

    cosang[2] = cos(angstr[2]);
    cosang[0] = sinz * cos(theta);
    cosang[1] = sinz * sin(theta);

    angstr[0] = acos(cosang[0]) ;
    angstr[2] = acos(cosang[2]) ;

/*  ROTATE AXES ACCORDING TO SIGMADIR. COMPUTE DIRECTION COSINES
    IN GEOGRAPHIC COORDINATES
*/

    axesrot(-1,cosang,sigmadir) ;

/* IF THE NEW POLE'S RADIUS VECTOR POINTS UP, TAKE THE OPPOSITE TO
    THE SAME PLANE (SINCE WE ARE WORKING WITH LOWER HEMISPHERE
    WULFF PROJECTION)
*/

    if(cosang[2] < -0.001){

      for(i=0 ; i<3; i++){
        cosang[i] = -cosang[i];
      }
    }

/*  COMPUTE ANGLES IN GEOGRAPHIC COORDINATES   */

    for(i=0; i< 3 ; i++)  anggh[i] = acos(cosang[i]);

/*  COMPUTE DIP FOR LINES AND POLES   */
```

```
      *dipgh = 90. - (anggh[2])* DEG ;

/* CHECK FOR POSSIBLE SINGULARITIES */

  sin3 = sin(anggh[2]);

  if(ABS(sin3)< 0.001 AND ABS(cosang[0])<0.001){

/*  IF sin3=0 THEN ddirgh IS UNDEFINED. WE ARBITRARILY SET
    IT TO 0 */

    *ddirgh = 0. ;
  }else {
    theta = atan2(cosang[1],cosang[0]);
    theta *= DEG;

/* COMPUTE DIP DIRECTION (eqn. 80) FOR THE POLE OR THE LINE. IF
   ddir > 360,  SUBSTRACT 360
*/
    *ddirgh = theta;

    if(*ddirgh > 359.9) *ddirgh = *ddirgh - 359.9;
  }

  return;
 }
-------------------------------------------------------------------
```

```
------------------------------------------------------------------
                /* Beginning of subroutine DRAW_MOHR */
------------------------------------------------------------------
/* This subroutine computes the X,Y coordinates of the 3-D Mohr
   Circles (c32, c21, c31) and writes them in an output file
   (filenameroot.mohr) according to the format required by system
   subroutine GRAPH(L) (number of points in the plot, followed by
   X,Y coordinates of each point).
------------------------------------------------------------------


                            VARIABLES
                        ===============


    dphi        - angle sampling interval for the plots.
    nfile       - number of the file 'mohrin', opened in the main
                  program.
    np          - number of points for the circles's plot
                  (maximum : 500 !!).
                  dimension of x21,y21,x31,y31,x32,y32.
    o21,o31,o32- mohr circle centers.
    phi         - variable angle for the plots.
    r21,r31,r32- mohr circle radius.
    sigma1,2,3 - principal stresses magnitudes.
    x21,y21     - mohr circle x,y coordinates.

 INPUT VARIABLES: sigma1,2,3,nfile

 OUTPUT: Writes circles's X,Y coordinates in file
         'filenameroot.mohr'


-----------------------------------------------------------------*/


#include <stdio.h>
#include <math.h>
#include "parameter.h"
#include "defs.h"
#define SIZE 200

draw_mohr(sig,fd)
double sig[3];
FILE * fd;
{

/*  DIMENSION AND INITIALIZE VARIABLES */
```

```c
    double r32,o32,o21,r21,r31,o31,cphi,sphi;
    double x32[500],y32[500],x21[500],y21[500],x31[500],y31[500];
    double dphi = PI/(double)SIZE;
    double phi = PI + dphi;
    int i,l,m;

/*  COMPUTE ORIGINS (O) AND RADIUS (R) OF EACH CIRCLE  */

r32 = (sig[1] - sig[2])/2. ;
o32 = (sig[1] + sig[2])/2. ;
o21 = (sig[0] + sig[1])/2. ;
r21 = (sig[0] - sig[1])/2. ;
o31 = (sig[0] + sig[2])/2. ;
r31 = (sig[0] - sig[2])/2. ;

r32 = ABS(r32);
o32 = ABS(o32);
r21 = ABS(r21);
o21 = ABS(o21);
r31 = ABS(r31);
o31 = ABS(o31);

/*  COMPUTE X,Y COORDINATES FOR EACH CIRCLE */

  for(i=0; i<SIZE ; i++){
    phi = phi - dphi ;
    cphi = cos(phi) ;
    sphi = sin(phi) ;

    x32[i] = o32 + r32 * cphi ;
    y32[i] = r32 * sphi ;
    x21[i] = o21 + r21 * cphi ;
    y21[i] = r21 * sphi ;
    x31[i] = o31 + r31 * cphi ;
    y31[i] = r31 * sphi ;
  }

/* WRITE COORDINATES IN FILE 'mohr.dat' */

  fprintf(fd,"%d \n",SIZE*3);

  for(i=0; i<SIZE ; i++){
    fprintf(fd,"%f %f \n",x32[i],y32[i]);
  }
```

```
for(l=i; l<(i+SIZE) ; l++){
    fprintf(fd,"%f %f \n", x21[l-i],y21[l-i]);
}

for(m=l; m<(l+SIZE) ; m++){
    fprintf(fd,"%f %f \n",x31[m-l],y31[m-l]);
}

return ;
}
```

```
for(l=i; l<(i+SIZE) ; l++){
    fprintf(fd,"%f %f \n", x21[l-i],y21[l-i]);
```

```
-----------------------------------------------------------------
                /* Beginning of subroutine PRINT_PARAMETERS */
-----------------------------------------------------------------
/*  This subroutine prints output to files:
filenameroot.onnet (fp1)
filenameroot.mohr (fp2)
filenameroot.rake (fp3)
filenameroot.print (fp4)
*/


#include <stdio.h>
#include <math.h>
#include "defs.h"
#include "parameter.h"

print_parameters(sx,sy,sz,parameters,fp1,fp2,fp3,fp4)
FILE *fp1,*fp2,*fp3,*fp4;
double sx,sy,sz;
Parameter* parameters;
{
  int i,signum[3];
  double maxtau_rock,maxtau_fault,sigmax,sigmin;

/*-----------------file to onnet-----------------------------*/

/* STRESS DIRECTIONS    */

  if(sz > sx AND sz > sy){
    signum[0] = 3;
    signum[1] = 2;
    signum[2] = 1;
    sigmax= sz;
    sigmin= sx;
  } else if(sz > sx AND sz < sy){
    signum[0] = 3;
    signum[1] = 1;
    signum[2] = 2;
    sigmax= sy;
    sigmin= sx;
  } else if(sz < sx AND sz < sy) {
    signum[0] = 2;
    signum[1] = 1;
    signum[2] = 3;
    sigmax= sy;
```

```c
    sigmin= sz;
  }

  fprintf(fp1,"    Wulff\n");
  fprintf(fp1,"size title 10\n");
  fprintf(fp1,"fatness title 5\n");

  for (i=0; i<3; i++){
    print_stress(signum[i],parameters->azimuth_[i],
                 parameters->dip_[i],fp1);
  }

/* FAULTS ORIENTATIONS */

  for(i=0; i<parameters->numfaults ; i++){
    print_fault(i,parameters->dipdir_fault[i],
                parameters->dip_fault[i],
parameters->dipdir_shear[i],parameters->dip_shear[i],fp1);
  }

/*--------------file to Mohr circle------------------------*/

/*  CALCULATE COULOMB LINE OF FRACTURE AND FRICTION  */

  maxtau_fault =  parameters-> fault_cohesion +
      (sigmax * parameters-> fault_friction);
  maxtau_rock =  parameters-> rock_cohesion +
    (sigmin* parameters-> rock_friction);

  fprintf(fp2,"2\n");
  fprintf(fp2,"0.0 %f \n",parameters->fault_cohesion);
  fprintf(fp2,"%f %f \n",sigmax,maxtau_fault);
  fprintf(fp2,"2\n");
  fprintf(fp2,"0.0 %f \n",parameters->rock_cohesion);
  fprintf(fp2,"%f %f \n",sigmin,maxtau_rock);

/*  CONSTRUCT A 3D MOHR-CIRCLE  */

  draw_mohr(parameters->sigma_,fp2);

  for(i=0; i<parameters->numfaults; i++){
    fprintf(fp2,"1 \n");
    fprintf(fp2,"%f %f \n",parameters-> signor_fault[i],
    parameters-> sigtau_fault[i]);
  }
```

145

```c
/* ------------file to rake---------------------------------*/

  fprintf(fp3,"%d \n",parameters->numfaults);
  for(i=0; i<parameters->numfaults; i++){
    fprintf(fp3,"%d %f \n",i,parameters-> rake[i]);
  }

/* -----------file to print----------------------------*/

  fprintf(fp4," S_hmin = %.2f S_Hmax = %.2f S_vert = %.2f \n",
             sx,sy,sz);
  fprintf(fp4," hmin_azimuth = %.2f \n",parameters->azimuth_[0]);
  fprintf(fp4," rock: friction= %.2f   cohesion = %.2f \n",
  parameters-> rock_friction, parameters-> rock_cohesion);
  fprintf(fp4," fault: friction= %.2f   cohesion = %.2f \n",
  parameters-> fault_friction, parameters-> fault_cohesion);

  fprintf(fp4," rake convention: 0 LL/ 180 RL/ 90 N/ -90 R \n");

  fprintf(fp4,"                 dipdir    dip        rake \n");
  for(i=0; i<parameters->numfaults; i++){
    fprintf(fp4," fault-%d   %.2f       %.2f      %.2f \n",
    i,parameters->dipdir_fault[i],parameters->dip_fault[i],
    parameters->rake[i]);
  }
  fclose(fp1);
  fclose(fp2);
  fclose(fp3);
  fclose(fp4);
}

print_stress(stressnum,dipdir,dip,fd)
int stressnum;
double dipdir,dip;
FILE * fd;
{
  fprintf(fd,"size line 10 \n");
  fprintf(fd,"fatness line 2 \n");
  fprintf(fd,"symbol line %d \n",stressnum);
  fprintf(fd,"line %f %f\n",dipdir,dip);
}

print_fault(faultnum,dipdir,dip,shdipdir,shdip,fd)
int faultnum;
```

146

```c
double dipdir,dip,shdipdir,shdip;
FILE * fd;
{
  double strike;

  strike = dipdir - 90. ;
  if (strike < 0 ) strike = strike + 180.;
  fprintf(fd,"fatness line 4 \n");
  fprintf(fd,"size line 6 \n");
  fprintf(fd,"symbol line %d \n",faultnum);
  fprintf(fd,"line %f 0 \n",strike);
  fprintf(fd,"plane %f %f \n",dipdir,dip);
  fprintf(fd,"size line 6 \n");
  fprintf(fd,"fatness line 4 \n");
  fprintf(fd,"symbol line x \n");
  fprintf(fd,"line %f %f \n",shdipdir,shdip);
}
```

```
-----------------------------------------------------------------
             /* Precompiler files */
-----------------------------------------------------------------
parameter.h
-----------------------------------------------------------------
#ifndef parameter_h
#define parameter_h
/* default parameters */
#define NUMFAULTS_MAX 10
#define AZIMUTH_1 0
#define AZIMUTH_2 0
#define AZIMUTH_3 90
#define DIP_1 90
#define DIP_2 0
#define DIP_3 0
#define SIGMA_1 3000
#define SIGMA_2 1200
#define SIGMA_3 600
#define SIGMA_VERT      1
#define SIGMA_NORTH     2
#define SIGMA_EAST      3
#define ROCK_FRICTION   1.0
#define FAULT_FRICTION  0.6
#define ROCK_COHESION   1000
#define FAULT_COHESION  50


/*---------------class Parameters ----------------------------*/
/* parameters controlling fault motion                    * */
/*                                                         * */
                                                         /* */
typedef struct parameter_ {                              /* */
      double sigma_[3]    ;
    int sigma_vert    ;
    int sigma_north   ;
    int sigma_east    ;
     double azimuth_[3]    ;
    double dip_[3]   ;
    double rock_friction  ;
    double fault_friction ;
    double rock_cohesion  ;
    double fault_cohesion ;
    double dipdir_fault[NUMFAULTS_MAX];
    double dip_fault[NUMFAULTS_MAX];
    int numfaults;
    double rake[NUMFAULTS_MAX];
```

```c
        double dipdir_shear[NUMFAULTS_MAX];
        double dip_shear[NUMFAULTS_MAX];
        double signor_fault[NUMFAULTS_MAX];
        double sigtau_fault[NUMFAULTS_MAX];
} Parameter ;                                                   /* */
extern double cosang[3],angstr[3],anggh[3],sigmadir[3][3];
extern double csangsq[3],angshr[3],csangsh[3],sig[3];
extern int ipole,irev,irev1,iwucrd;
extern double sx,sy,sz,cl,cm,cn,ddirgh,dipgh,dipst,ddirst;
extern double ddir,dip,trend,plunge;
extern double *signor_fault,*sigtau_fault;
extern Parameter * read_parameter() ;                           /* */
extern Parameter * DefaultParameter() ;                         /* */
/*--------------class Parameter ----------------------------*/

#endif
------------------------------------------------------------------
defs.h
------------------------------------------------------------------
#define nil 0
#define false 0
#define true  1
/* que j'aime a faire connaitre ce nombre utiles aux sages */
#define RAD 3.1415926635/180.
#define DEG 180./3.1415926635
#define PI 3.1415926635
#define IS ==
#define IS_NOT !=
#define AND &&
#define OR ||
#define FNX(x, y, theta) ((x) * cos(theta) - (y) * sin(theta))
#define FNY(x, y, theta) ((x) * sin(theta) + (y) * cos(theta))
#define MIN(a, b)( ((a) < (b)) ? (a) : (b) )
#define MAX(a, b)( ((a) > (b)) ? (a) : (b) )
#define ABS(a)( ((a) < (0)) ? (-a) : (a) )
------------------------------------------------------------------
```

C.2 Eshelby

The following is a brief description of three programs I've written to determine the elastic field of an ellipsoidal inclusion with different elastic moduli embedded in an otherwise homogeneous matrix. The method employed here to solve this problem is referred to in the literature as the equivalent inclusion method (Mura, 1987), and is described in the appendix at the end of chapter 1. The equations described in the following programs are derived in Eshelby (1957) and Mura (1987).

The programs are written in the C-language. The first program, called MAIN, is the core of the calculation. Here we solve for the total stress and strain fields inside the inclusion, given remote stress conditions and moduli contrasts between the inclusion and the matrix. The geometry of the problem is shown in Figure 1: an infinite elliptical cylinder is assumed, with the [0] axes parallel to the x-axis, the [1] axes parallel to the y-axis, and the [2] axes infinite and parallel to the z-axes (out of the page). The orientation of the remote stress field is determined by θ, the angle between the x-axis and the direction of maximum compressive stress.

The second program, called JUSTOUT, can be used in conjunction with MAIN to determine the total stress and strain fields just outside the inclusion. As Eshelby shows (1957, Eq. 2.13), stresses and strains just outside the inclusion can be found from their values at an adjacent point just inside without having to solve the exterior problem at all.

The third program, called MURA, can be used in conjunction with MAIN to calculate the stress and strain field anywhere inside, just outside or at a great distance from the inclusion. The bulk of the equations described in this program are derived in Mura (p.84-88, 1987).

The programs compute: (1) the applied remote stress and strain matrix, (2) the constrained strain which is related to the stress via Hooke's law, and (3) the transformation strain or eigenstrain which is the stress-free strain due to the fact

that the inclusion has different elastic properties than the matrix.

```
--------------------------------------------------------------------------
                /*Beginning of program MAIN*/
--------------------------------------------------------------------------
/* The following program calculates stresses and strains
INSIDE an elliptical isotropic inclusion with elastic moduli
different from the matrix. */

/* INPUT
(1) Remote stress conditions S_MAX,S_INT,S_MIN;
(2) Matrix: Shear modulus(G), Poisson's ratio(nu);
(3) Inclusion: Bulk modulus(K_I), Poisson's ratio(nu_I);
*/

/* OUTPUT
(1) Total stress field in inclusion (sigma_I) ;
(2) Total strain field in inclusion (strain_I);
*/

#include <stdio.h>
#include <math.h>

#define nil 0
#define false 0
#define true  1
#define RAD 3.1415926635/180.
#define DEG 180./3.1415926635
#define PI 3.1415926635
#define IS ==
#define IS_NOT !=
#define AND &&
#define OR ||

#define G 0.631E+11
#define K_FACTOR .1
#define NU 0.25
#define NU_I 0.25
#define S_MAX 1.0
#define S_INT 0.5
#define S_MIN 0.2
#define DEG_FROM_X_AXIS 30
#define AA1 1
#define AA2 1

/*Elastic moduli*/
double nu,nu_I; /* Poisson's ratios*/
```

```
double G,G_I; /* Shear moduli*/
double E,E_I; /*Young's moduli*/
double lambda,lambda_I; /*Lame's constants*/
double K,K_I; /* Bulk moduli*/

/*Eshelby tensor variables*/
double S1111,S2222,S3333,S1122,S1133,S2233,S1212,S1313,S2323;
double S3322,S2211,S3311;

main()
{
    double Dlambda,Dg; /*Moduli contrast*/
    double sigmap[3],sigmap_I[3]; /*Principal stress magnitudes*/
    double theta; /*Orientation of S_MAX from horizontal*/
    double strain[3][3],sigma[3][3];/*Remote stress,strain comp.*/
    double a1,a2; /*Dimensions of ellipse*/
    double ell1,ell2,den; /*Geometric variables*/


    /*Variables to calculate normal strain components*/
    double A1,A2,A3;
    double B11,B12,B13,B21,B22,B23,B31,B32,B33;
    double C1,C2,C3,Det_B;
    double Det_B1,Det_B2,Det_B3;

    /*Disturbed stress and strain field due to presence of inclusion*/
    double strain_D[3][3],sigma_D[3][3];
    double strainT,strain_IT;

    /*Principal stress magnitudes inside the inclusion*/
    double sigmap_I[3];
    /*Total stress and strain field inside inclusion*/
    double strain_I[3][3],sigma_I[3][3];
    double theta_I; /*Principal stress directions in inclusion*/
    double etheta_I;/*Principal strain directions in inclusion*/

    double num1,den1,num2,den2;

/*Specify elastic moduli of matrix*/

    lambda = 2.*G*nu/(1. - 2.*nu) ;
    E = 2.* (1. + nu)*G;
    K = lambda + 2.*G/3. ;

/*Specify elastic moduli of inclusion*/
```

```
      K_I = K*K_FACTOR;
      G_I = 3.*K_I*(1.-2.*nu_I)/(2.*(1+nu_I));
      E_I = 2.* (1. + nu_I)*mu_I;
      lambda_I = K_I - 2.*mu_I/3. ;


/*Calculate moduli contrats*/

      DG = G_I - G;
      Dlambda = lambda_I - lambda;


/*Specify remote stress conditions (homogeneous) in terms of
principal stress magnitudes.
The program is written for the case of an elliptical cylinder with
the [2] direction parallel to the infinite axis of the cylinder!! */

      sigmap[0] = S_MAX ;
      sigmap[1] = S_MIN;
      sigmap[2] = S_INT ;


/*Specify the orientation of S_MAX from horizontal axis*/

      theta= DEG_FROM_X_AXIS;


/*Calculate remote stress components*/

      sigma[0][0] = 0.5*(sigmap[0] + sigmap[1]) +
                    0.5*(sigmap[0] - sigmap[1])*cos(2.*theta*RAD);
      sigma[1][1] = 0.5*(sigmap[0] + sigmap[1]) -
                    0.5*(sigmap[0] - sigmap[1])*cos(2.*theta*RAD);
      sigma[0][1] = -0.5*(sigmap[0] - sigmap[1])*sin(2.*theta*RAD);
      sigma[0][2] = 0. ;
      sigma[1][2] = 0. ;
      /* If problem is plain strain ezz=0 ==> szz is different from 0 */
      sigma[2][2] = nu*(sigma[0][0] + sigma[1][1]) ;
      /* If problem is plain stress szz=0 ==> ezz is different from 0
      sigma[2][2] = 0; */


/*Calculate remote strain components*/

      strain[0][1] = sigma[0][1]/(2.*G);
      strain[1][2] = sigma[1][2]/(2.*G);
      strain[0][2] = sigma[0][2]/(2.*G);
      strain[0][0] = (sigma[0][0] - nu*(sigma[1][1] + sigma[2][2]))/E ;
      strain[1][1] = (sigma[1][1] - nu*(sigma[0][0] + sigma[2][2]))/E ;
```

```
        strain[2][2] = (sigma[2][2] - nu*(sigma[1][1] + sigma[0][0]))/E ;

/* Compute equivalent eigenstrains for inhomogeneous elliptic
cylindrical inclusion (a3 is infinite) */

/*Specify aspect ratio*/

        a1=AA1;
        a2=AA2;

        den = 2.*(1. - nu);
        ell1 = a1/(a1 + a2);
        ell2 = a2/(a1 + a2);

/*Calculate the Eshelby tensor (Eq. 11.22, Mura, 1957)*/

        S1111 = ((a2*a2 + 2*a1*a2)/
                ((a1 + a2)*(a1 + a2))+(1. - 2.*nu)*ell2)/den;
        S2222 = ((a1*a1 + 2*a1*a2)/
                ((a1 + a2)*(a1 + a2))+(1. - 2.*nu)*ell1)/den;
        S3333 = 0;
        S1122 = (ell2*ell2 - (1. - 2.*nu)*ell2)/den ;
        S2233 = (2.*nu*ell1)/den ;
        S3311 = 0 ;
        S1133 = (2.*nu*ell2)/den ;
        S2211 = (ell1*ell1 - (1. - 2.*nu)*ell1)/den ;
        S3322 = 0 ;
        S1313 = ell2/2;
        S2323 = ell1/2;
        S1212 = ((a1*a1 + a2*a2)/
                (2.*(a1 + a2)*(a1 + a2)) + (1. - 2.*nu)/2.)/den;

/*Calculate disturbed strain field inside the inclusion.
                                (Eq. 4.5, Eshelby, 1957)*/

        /*  shear strains are independent of each other*/

        strain_D[0][2]= -DG*strain[0][2]/(2.*DG*S1313 + G);
        strain_D[1][2]= -DG*strain[1][2]/(2.*DG*S2323 + G);
        strain_D[0][1]= -DG*strain[0][1]/(2.*DG*S1212 + G);

        /*  normal strains are not independent of each other
            ==>require solving a system of 3 linear equations*/

        A1 = S1111 + S2211 + S3311;
```

```
A2 = S1122 + S2222 + S3322;
A3 = S1133 + S2233 + S3333;


B11 = Dlambda*A1 + 2.*DG*S1111 + lambda + 2.*G ;
B12 = Dlambda*A2 + 2.*DG*S1122 + lambda ;
B13 = Dlambda*A3 + 2.*DG*S1133 + lambda ;
B21 = Dlambda*A1 + 2.*DG*S2211 + lambda ;
B22 = Dlambda*A2 + 2.*DG*S2222 + lambda + 2.*G ;
B23 = Dlambda*A3 + 2.*DG*S2233 + lambda ;
B31 = Dlambda*A1 + 2.*DG*S3311 + lambda ;
B32 = Dlambda*A2 + 2.*DG*S3322 + lambda ;
B33 = Dlambda*A3 + 2.*DG*S3333 + lambda + 2.*G ;


strainT = strain[0][0] + strain[1][1] + strain[2][2] ;


C1 = -(Dlambda*strainT + 2.*DG*strain[0][0]);
C2 = -(Dlambda*strainT + 2.*DG*strain[1][1]);
C3 = -(Dlambda*strainT + 2.*DG*strain[2][2]);


Det_B =    B11*(B22*B33 - B23*B32)
         - B12*(B21*B33 - B23*B31)
           + B13*(B21*B32 - B22*B31) ;


if (Det_B IS nil) fprintf(stderr,"Det_B = %e\n",Det_B);


strain_D[0][0] = (C1*(B22*B33 - B23*B32) -
                  B12*(C2*B33 - C3*B23)  +
                  B13*(C2*B32 - C3*B22)      )/Det_B ;
strain_D[1][1] = (B11*(C2*B33 - B23*C3) -
                  C1*(B21*B33 - B31*B23)  +
                  B13*(C3*B21 - C2*B31)      )/Det_B ;
strain_D[2][2] = (B11*(B22*C3 - C2*B32) -
                  B12*(C3*B21 - C2*B31)  +
                  C1*(B21*B32 - B31*B22)      )/Det_B ;

/* Calculate the disturbed stress field inside the inclusion.
                              (Eq. 11.22.1, Mura, 1987)*/
        den = G/(1. - nu);

sigma_D[0][0] =( (-2 + (1 + 2*a1/a2)*ell2*ell2 +ell2)
                  *strain_D[0][0]
                  + (ell2*ell2 - ell2)*strain_D[1][1]
                  - (2*nu*ell1)*strain_D[2][2]          )*den;
sigma_D[1][1] =( (-2 + (1 + 2*a2/a1)*ell1*ell1 +ell1)
                  *strain_D[1][1]
```

```
                            + (ell1*ell1 - ell1)*strain_D[0][0]
                            - (2*nu*ell2)*strain_D[2][2]        )*den;
            sigma_D[2][2] =(- (2*nu*ell1)*strain_D[0][0]
                            - (2*nu*ell2)*strain_D[1][1]
                            - 2*strain_D[2][2]        )*den;
            sigma_D[0][1] = -  2*(a1/a2)*ell2*ell2*strain_D[0][1]*den;
            sigma_D[1][2] = -  2*ell2*G*strain_D[1][2];
            sigma_D[0][2] = -  2*ell1*G*strain_D[0][2];


----------------------------------------------------------------
/* Programs JUSTOUT or MURA may be called here */
        /* In which case the following calculations are for */
        /* stress and strain fields at specified (x,y) points */
        /* and not necessarily in the inclusion             */
----------------------------------------------------------------
/*   Calculate total stress in the inclusion */

        for (i=0; i< 3; i++)
            {
            for(j=0; j<3 ; j++)
            {
               sigma_I[i][j] = sigma[i][j] + sigma_D[i][j];
            }
        }

/*   Calculate total strain in the inclusion*/

        /*normal strains are not independent of each other
          ==>require solving a system of 3 linear equations */

        B11 = -Dlambda - 2*DG ;
        B12 = -Dlambda ;
        B13 = -Dlambda ;
        B21 = -Dlambda ;
        B22 = -Dlambda - 2*DG ;
        B23 = -Dlambda ;
        B31 = -Dlambda ;
        B32 = -Dlambda ;
        B33 = -Dlambda - 2*DG ;

        strain_IT = strain_D[0][0] + strain_D[1][1] + strain_D[2][2] ;

        C1 = lambda*strain_IT + 2*G*strain_D[0][0];
        C2 = lambda*strain_IT + 2*G*strain_D[1][1];
        C3 = lambda*strain_IT + 2*G*strain_D[2][2];
```

```
Det_B = B11*(B22*B33 - B23*B32) -B12*(B21*B33 - B23*B31)
        +B13*(B21*B32 - B22*B31) ;

if (Det_B IS nil) fprintf(stderr,"Det_B**** = %e\n",Det_B);

strain_I[0][0] = (C1*(B22*B33 - B23*B32) -
                  B12*(C2*B33 - C3*B23)  +
                  B13*(C2*B32 - C3*B22)     )/Det_B ;
strain_I[1][1] = (B11*(C2*B33 - B23*C3) -
                  C1*(B21*B33 - B31*B23)  +
                  B13*(C3*B21 - C2*B31)     )/Det_B ;
strain_I[2][2] = (B11*(B22*C3 - C2*B32) -
                  B12*(C3*B21 - C2*B31)  +
                  C1*(B21*B32 - B31*B22)     )/Det_B ;

/*Shear strains are independent of each other*/

for (i=0;i<3;i++)
{
    for (j=0;j<3;j++)
    {
        if (i IS_NOT j)
        strain_I[i][j] = strain_D[i][j]*G/(G - G_I);
    }
}

/* Calculate principal stress directions in inclusion */

num2 = 2*sigma_I[0][1];
den2 = sigma_I[0][0] - sigma_I[1][1];
theta_I = 0.5*atan2(num2,den2);

/* Calculate principal strains directions in inclusions */

etheta_I = 0.5*atan2((2*strain_I[0][1]),
           (strain_I[0][0] - strain_I[1][1]));

/* Calculate principal stress magnitudes in inclusion */

sigmap_I[0]= sigma_I[0][0]*cos(theta_I)*cos(theta_I)
           + 2*sigma_I[0][1]*cos(theta_I)*sin(theta_I)
           + sigma_I[1][1]*sin(theta_I)*sin(theta_I) ;
sigmap_I[1]= sigma_I[0][0]*sin(theta_I)*sin(theta_I)
           - 2*sigma_I[0][1]*cos(theta_I)*sin(theta_I)
```

```
                + sigma_I[1][1]*cos(theta_I)*cos(theta_I) ;
        sigmap_I[2]= sigma_I[2][2];

------------------------------------------------------------------
                        /* End of program MAIN*/
------------------------------------------------------------------
}
```

```
--------------------------------------------------------------------
                    /* Beginning of subroutine JUSTOUT*/
--------------------------------------------------------------------
/*The following subroutine calculates the stress and strain field at
any point just outside the inclusion, asssuming it is called from the
main program described previously. Equation numbers refer to Eshelby's
paper (1957)*/

/*INPUT (from MAIN)*/
/* Strain disturbance due to presence of inclusion: strain_D*/

/*OUTPUT*/
/* Stress and strain disturbance just outside the inclusion */

#define X_COORD 0

void justout(strain_D)
double strain_D[3][3];
{
        int i,l;
        double x[2];    /*x,y location of point just outside inclusion*/
        double h,aa[2],n[3];

        double dil,dim,dlm,dij; /* Kronecker delta */

        /* Variables for calculating Eshelby's eqs 2.13    */
        double X1,X,Z1,Z,primo,secondo,terzo;
        double quarto,quinto,sesto,settimo;

        /*Constrained strain inside inclusion*/
        double Cstrain_I[3][3];
        double Cstrain_IT,strain_DT,strain_T;

        /*Stress and strain disturbance just outside inclusion*/
        double sigma_O[3][3];
        double strain_O[3][3];

/* Specify dimensions of ellipse*/

        aa[0]=AA1;
        aa[1]=AA2;

/*    Calculate constrained strain (Eq. 3.6, Eshelby, 1957)*/

        Cstrain_I[0][0]= S1111*strain_D[0][0]+S1122*strain_D[1][1]
```

160

```
                      + S1133*strain_D[2][2];
       Cstrain_I[1][1]= S2211*strain_D[0][0]+S2222*strain_D[1][1]
                      + S2233*strain_D[2][2];
       Cstrain_I[2][2]= S3311*strain_D[0][0]+S3322*strain_D[1][1]
                      + S3333*strain_D[2][2];
       Cstrain_I[0][1]= 2.*S1212*strain_D[0][1];
       Cstrain_I[0][2]= 2.*S1313*strain_D[0][2];
       Cstrain_I[1][2]= 2.*S2323*strain_D[1][2];

       Cstrain_I[1][0]= Cstrain_I[0][1];
       Cstrain_I[2][1]= Cstrain_I[1][2];
       Cstrain_I[2][0]= Cstrain_I[0][2];

/* Specify x,y coordinates of point just outside inclusion*/

       x[0]=X_COORD;

       x[1] = aa[1]*sqrt(1. - x[0]*x[0]/(aa[0]*aa[0]));
       h = sqrt(x[0]*x[0]/(aa[0]*aa[0]*aa[0]*aa[0])
           + x[1]*x[1]/(aa[1]*aa[1]*aa[1]*aa[1]));

/* Calculate normal vector*/

       n[0] = x[0]/(aa[0]*aa[0]*h);
       n[1] = x[1]/(aa[1]*aa[1]*h);
       n[2] = 0.;

/*Calculate constrained strain just outside the inclusion.
                           (Eq. 2.13, Eshelby, 1957) */

       strain_DT= strain_D[0][0] + strain_D[1][1] + strain_D[2][2];
       Cstrain_IT= Cstrain_I[0][0]+Cstrain_I[1][1]+Cstrain_I[2][2];

       X = (1. + nu)/(3. - 3.*nu);
       X1 = 1./(1.-nu);
       Z = (1.-2.*nu)/(1.-nu);
       Z1 = (1.-2.*nu)/(3.-3.*nu);

       factor=0;
       for (i=0;i<3;i++)
       {
              for (l=0;l<3;l++)
              {
              if(i IS l) {dil = 1.;} else {dil=0.;}
              factor+=(strain_D[i][l]-(strain_DT*dil/3.))*n[i]*n[l];
```

161

```
            }
    }

    for (i=0;i<3;i++)
    {
            for (l=0;l<3;l++)
            {
            if(i IS l) {dil = 1.;} else {dil=0.;}
            primo = Cstrain_I[i][l] - (Cstrain_IT*dil/3.);
            secondo = factor * n[i]*n[l] ;
            terzo=0;
            quarto=0;
                    for (m=0;m<3;m++)
                    {
                    if(i IS m) {dim = 1.;} else {dim=0.;}
                    if(l IS m) {dlm = 1.;} else {dlm=0.;}
                    terzo+=(strain_D[i][m]-(strain_DT*dim/3.))*n[m];
                    quarto+=(strain_D[l][m]-(strain_DT*dlm/3.))*n[m];
                    }
            terzo *=n[l];
            quarto *= quarto*n[i];
            quinto = factor*dil;
            sesto = strain_DT*(n[i]*n[l] - (dil/3.));
            settimo = Cstrain_IT - (X*strain_DT) - (Z*factor);
            strain_O[i][l] = primo + (X1*secondo) - terzo - quarto
                    + (Z1*quinto) - (X*sesto) + (settimo*dil/3.);
            }
    }

    strainT= strain_O[0][0] + strain_O[1][1] + strain_O[2][2];

/* Calculate stress disturbance just outside inclusion */

    sigma_O[0][0] = (2.*G*strain_O[0][0]) + (lambda*strainT);
    sigma_O[1][1] = (2.*G*strain_O[1][1]) + (lambda*strainT);
    sigma_O[2][2] = (2.*G*strain_O[2][2]) + (lambda*strainT);
    sigma_O[0][1] = 2.*G*strain_O[0][1] ;
    sigma_O[0][2] = 2.*G*strain_O[0][2] ;
    sigma_O[1][2] = 2.*G*strain_O[1][2] ;
    sigma_O[1][0]= sigma_O[0][1];
    sigma_O[2][1]= sigma_O[1][2];
    sigma_O[2][0]= sigma_O[0][2];

/* To calculate total stress/strain fields return{sigma_O,strain_O}
to main program*/
```

```
-----------------------------------------------------------------------
                     /*End of subroutine JUSTOUT*/
-----------------------------------------------------------------------
}
```

```
----------------------------------------------------------------------
                    /* Begin subroutine MURA */
----------------------------------------------------------------------
/* The following program solve Eshelby's problem for infinite cylinder
at any point inside or outside the inclusion.*/

/* INPUT (from MAIN)
Eigenstrain tensor strain_D */

/* OUTPUT
Total stress field sigma_O*/

/* The following subroutines are defined at the end of the program*/
double Delta_b();
double KD();
double DlDelt_l();

void mura(strain_D)
double strain_D[3][3];
{
        int step,i,j,k,ki,l,m,ck,cki;
        /* Temporary variables that assume either the value within
           the inclusion or the value of the matrix
           (depends on (x,y))*/
        double G_f,nu_f,lambda_f,E_f;
        /*Position (x,y,z) of the point of interest*/
        double x[3];
        /* Variables to compute lower limit of integration*/
        double p,q,ll_Dis,ll_1,ll_2;
        double dlower_a[3],dldlw_l[3];
        double DD_s,D_s,D3_s,d2_lambda[3];
        double Db,DDb,sign;
        double Dp[3],DDp[3][3],Dq[3],DDq[3][3],DDl[3][3];
        /* I-integrals and their derivatives*/
        double I[3],ID[3][3];
        double I_a[3][3],ID_a[3][3][3],I_ab[3][3][3],ID_ab[3][3][3][3];
        double SE[3][3][3][3]; /*Eshelby tensor*/
        double D[3][3][3][3]; /*Tensor that relates constrained
                                 strain to eigenstrain*/

/* Calculate lower limit to integration ( Eq. 11.37,Mura, 1987) */

        p=aa[0]*aa[0] + aa[1]*aa[1] - x[0]*x[0] - x[1]*x[1];
        q=aa[0]*aa[0]*aa[1]*aa[1] - x[0]*x[0]*aa[1]*aa[1]
                                  - x[1]*x[1]*aa[0]*aa[0];
```

164

```c
        ll_Dis= sqrt(p*p/4. - q);
        ll_1 = ll_Dis;
        ll_2= -p/2. - ll_1;
        ll_1= -p/2. + ll_1;
        llower=MAX(ll_1,ll_2);

        sign = 1.;
        if(llower IS ll_2 ) sign= -sign;
        if ((x[0]*x[0]/(aa[0]*aa[0])+x[1]*x[1]/(aa[1]*aa[1])) < 1.)
            llower=0;
        Db =Delta_b(llower);
        DDb=DlDelt_l(llower);

/* Calculate first derivatives of p and q */

        for(i=0; i<3; i++)
        {
        Dp[i]= -2.*x[i];
        Dq[i]= -2.*x[i]*(aa[1]*aa[1]*KD(i,0) + aa[0]*aa[0]*KD(i,1));
        }
        Dp[2] = 0.;
        Dq[2] = 0.;

/* Calculate second derivatives of p and q */

        for(i=0; i<3; i++)
        {
                for(j=0; j<3; j++)
                {
                if(i IS 2 OR j IS 2)
                 {
                 DDp[i][j] = 0.;
                 DDq[i][j] = 0.;
                 } else {
        DDp[i][j]= (-2.*KD(i,j));
        DDq[i][j]=(-2.*KD(i,j)*(aa[1]*aa[1]*KD(i,0)
                +aa[0]*aa[0]*KD(i,1)));
                 }
                }
        }
        if( ( x[0]*x[0]/aa[0]*aa[0] + x[1]*x[1]/aa[1]*aa[1]) < 1.)
            llower=0.;

/* Define the derivative of llower with respect to X(j) */
```

```
for(i=0; i<2;i++)
{
dlower_a[i] = -0.5*Dp[i]+sign*0.5*(0.5*p*Dp[i]-Dq[i])/ll_Dis;
}
dlower_a[2]=0;

for(i=0; i<3;i++)
{
   for(k=0; k<3;k++)
   {
    if(i IS 2 OR k IS 2)
      {
        DD1[i][k] = 0.;
      } else {
      DD1[i][k] = -0.5*DDp[i][k]-sign*0.25*(0.5*p*Dp[k]-Dq[k])
                  *(0.5*p*Dp[i]-Dq[i])/(ll_Dis*ll_Dis*ll_Dis)
                  +sign*0.25*(p*DDp[i][k]+Dp[i]*Dp[k]
                  -2.*DDq[i][k])/ll_Dis;
      }
   }
}


/*Define the six functions Ii--Ii,j--Ii,jk--Iij--Iij,k--Iij,kl*/

     if (aa[0] IS aa[1] )
     {


/* Solve the problem of a circular infinite cylinder */

     /* Definition of Ii */

     I[0]=2.*PI*aa[0]*aa[0]/(aa[0]*aa[0]+llower);
     I[1] =2.*PI*aa[0]*aa[0]/(aa[0]*aa[0]+llower);
     I[2] = 0.;

     /* Definition of Ii,j */
     for (i=0; i<3; i++)
     {
          for (j=0; j<3; j++)
          {
           if(i IS 2 OR j IS 2 OR llower IS 0)
                { I_a[i][j]=0.;
                } else {
                  I_a[i][j]= -dlower_a[j]*2.*PI*aa[0]*aa[1]/
                          ((aa[i]*aa[i] + llower)*(Db));
```

166

```
                    }
            }
    }

/* Definition of Ii,jk */
for(i=0;i<3;i++)
        {
        for(j=0;j<3;j++)
                {
                for(k=0;k<3;k++)
                 {
                if(i IS 2 OR j IS 2 OR k IS 2 OR llower IS 0)
                 {
                 I_ab[i][j][k] = 0.;
                 } else {
                 I_ab[i][j][k]= 2.*PI*aa[0]*aa[1]*
                                (dlower_a[j]
   *dlower_a[k]/((aa[i]*aa[i]+llower)*(aa[i]*aa[i]+llower)*Db)
+dlower_a[j] * DDb* dlower_a[k]/((aa[i]*aa[i]+llower) *Db *Db)
        -DDl[j][k]/((aa[i]*aa[i]+llower) *Db )
                                );
                }
                }
        }
}

/* Definition of Iij */
ID[0][0] = PI*aa[0]*aa[0]/((aa[0]*aa[0]+llower)
           *(aa[0]*aa[0]+llower));
ID[0][1] = PI*aa[0]*aa[0]/((aa[0]*aa[0]+llower)
           *(aa[0]*aa[0]+llower));
ID[1][1] = PI*aa[0]*aa[0]/((aa[0]*aa[0]+llower)
           *(aa[0]*aa[0]+llower));
ID[1][0]= PI*aa[0]*aa[0]/((aa[0]*aa[0]+llower)
           *(aa[0]*aa[0]+llower));
ID[2][2] =0.;
ID[0][2]=0.;
ID[2][0]=0.;
ID[1][2]=0.;
ID[2][1]=0.;

/* Definition of Iij,k */
for (i=0;i<3;i++)
{
        for (j=0;j<3;j++)
```

```
            {
                    for(k=0;k<3;k++)
                    {
                    if(i IS 2 OR j IS 2 OR k IS 2 OR llower IS 0)
                     {
                     ID_a[i][j][k] = 0;
                     } else {
                      ID_a[i][j][k] = - dlower_a[k]*2.*PI
                                     *aa[1]*aa[0]
            /((aa[i]*aa[i]+llower) *(aa[j]*aa[j]+llower)*Db);
                     }
                    }
            }
    }


/* Definition of Iij,kl */
for (i=0;i<3;i++)
{
        for (j=i;j<3;j++)
        {
                for(k=0;k<3;k++)
                {
                  for(l=0;l<3;l++)
                  {
        if(i IS 2 OR j IS 2 OR k IS 2 OR l IS 2 OR llower IS 0)
                  {
                  ID_ab[i][j][k][l] = 0;
                  } else {
                  ID_ab[i][j][k][l] =
            2.*PI*aa[0]*aa[1]*dlower_a[l]*
                        (
            dlower_a[k]/((aa[i]*aa[i]+llower)*
                        (aa[i]*aa[i]+llower)*
                        (aa[j]*aa[j]+llower)*Db)
            +dlower_a[k]/((aa[i]*aa[i]+llower)*
                        (aa[j]*aa[j]+llower)*
                        (aa[j]*aa[j]+llower)*Db)
            +dlower_a[k]*DDb/((aa[i]*aa[i]+llower)*
                        (aa[j]*aa[j]+llower) *Db *Db)
                    )
            -2.*PI*aa[0]*aa[1]*DDl[k][l]/
             ((aa[i]*aa[i]+llower)*(aa[j]*aa[j]+llower)*Db);
                  }
                }
              }
```

```
                }
        }
        } else {

/* Solve problem of inifinet elliptical cylinder */

/*Define the six functions Ii--Ii,j--Ii,jk--Iij--Iij,k--Iij,kl */

        /* Definition of Ii  (Eq. 12.20, Mura, 1957)*/
        I[0] = ( 4.*PI*aa[0]*aa[1]/(aa[1]*aa[1]-aa[0]*aa[0]))
                * (sqrt(aa[1]*aa[1] + llower)/sqrt(aa[0]*aa[0]
                                                + llower) - 1.);
        I[1] = ( 4.*PI*aa[0]*aa[1]/(aa[0]*aa[0]-aa[1]*aa[1]))
                * (sqrt(aa[0]*aa[0] + llower)/sqrt(aa[1]*aa[1]
                                                + llower) - 1.);
        I[2] = 0.;

        /* Definition of Ii,j */
        for (i=0; i<3; i++)
        {
                for (j=0; j<3; j++)
                {
                 if(i IS 2 OR j IS 2 OR llower IS 0)
                        { I_a[i][j]=0.;
                        } else {
                         I_a[i][j]= -dlower_a[j]*2.*PI*aa[0]*aa[1]/
                                    ((aa[i]*aa[i] + llower)*(Db));
                        }
                }
        }

        /* Definition of Ii,jk */
        for(i=0;i<3;i++)
                {
                for(j=0;j<3;j++)
                        {
                        for(k=0;k<3;k++)
                         {
                         if(i IS 2 OR j IS 2 OR k IS 2 OR llower IS 0)
                         {
                         I_ab[i][j][k] = 0.;
                         } else {
                         I_ab[i][j][k]= 2.*PI*aa[0]*aa[1]*
                                       (
                         dlower_a[j]*dlower_a[k]/((aa[i]*aa[i]+llower)
```

```
                                        *(aa[i]*aa[i]+llower) *Db)
                 +dlower_a[j]*DDb*dlower_a[k]
                  /((aa[i]*aa[i]+llower) *Db *Db)
                --DD1[j][k]/((aa[i]*aa[i]+llower) *Db )
                            );
                }
            }
        }
    }


/* Definition of Iij */
ID[0][1] = (I[1] - I[0])/(aa[0]*aa[0]-aa[1]*aa[1]);
ID[0][0] = 4.*PI*aa[0]*aa[1]/((aa[0]*aa[0] + llower)*Db)
              -ID[0][1] ;
ID[0][0] = ID[0][0]/3.;
ID[1][1] = 4.*PI*aa[1]*aa[0]/((aa[1]*aa[1] + llower)*Db)
              -ID[0][1] ;
ID[1][1] = ID[1][1]/3.;
ID[1][0]= ID[0][1];
ID[2][2] =0.;
ID[0][2]=0.;
ID[2][0]=0.;
ID[1][2]=0.;
ID[2][1]=0.;


/* Definition of Iij,k */
for (i=0;i<3;i++)
{
        for (j=0;j<3;j++)
        {
                for(k=0;k<3;k++)
                {
                if(i IS 2 OR j IS 2 OR k IS 2 OR llower IS 0)
                 {
                 ID_a[i][j][k] = 0;
                 } else {
                 ID_a[i][j][k] = - dlower_a[k]*2.*PI*aa[1]*aa[0]
                /((aa[i]*aa[i]+llower) *(aa[j]*aa[j]+llower)*Db);
                 }
                }
        }
}


/* Definition of Iij,kl */
for (i=0;i<3;i++)
```

```
        {
            for (j=i;j<3;j++)
            {
                for(k=0;k<3;k++)
                {
                    for(l=0;l<3;l++)
                    {
            if(i IS 2 OR j IS 2 OR k IS 2 OR l IS 2 OR llower IS 0)
                        {
                        ID_ab[i][j][k][l] = 0;
                        } else {
                        ID_ab[i][j][k][l] =
                    2.*PI*aa[0]*aa[1]*dlower_a[l]*
                        (
                    dlower_a[k]/((aa[i]*aa[i]+llower)
                            *(aa[i]*aa[i]+llower)
                            * (aa[j]*aa[j]+llower)*Db)
                    +dlower_a[k]/((aa[i]*aa[i]+llower)
                            *(aa[j]*aa[j]+llower)
                            *(aa[j]*aa[j]+llower)*Db)
                    +dlower_a[k]*DDb/((aa[i]*aa[i]+llower)
                            *(aa[j]*aa[j]+llower) *Db *Db)
                        )
                    -2.*PI*aa[0]*aa[1]*DDl[k][l]/(
                        (aa[i]*aa[i]+llower)
                        *(aa[j]*aa[j]+llower)*Db);
                    }
                }
                }
            }
        }


    }


/* Calculate SE, Eshelby's tensor  (Eq. 11.42, Mura, 1987) */

    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
        {
            for(k=0;k<3;k++)
            {
                for(l=0;l<3;l++)
```

171

```
                          {
                            if(i IS 2)
                             {
                            SE[i][j][k][l]=KD(i,j)*KD(k,l)
                    *(2.*nu*I[i]-I[k]+I[k])
                    +(KD(i,k)*KD(j,l) + KD(j,k)*KD(i,l))
                    *(I[j]-I[j]+ (1.-nu) *(I[k]+I[l]));
                             } else {
                            SE[i][j][k][l]=KD(i,j)*KD(k,l)
                    *(2.*nu*I[i]-I[k]+aa[i]*aa[i]*ID[k][i])
                    +(KD(i,k)*KD(j,l) + KD(j,k)*KD(i,l))
                    *(aa[i]*aa[i]*ID[i][j]-I[j]+ (1.-nu)
                    *(I[k]+I[l]));
                             }
                           }
                         }
                  }
          }


/* Calculate Dijkl of eps(ij)=Dijkl*eps*(kl)
   (Eq. 11.41, Mura, 1987)*/


        for (i=0;i<3;i++)
        {
                for (j=0;j<3;j++)
                {
                        for(k=0;k<3;k++)
                        {
                          for(l=0;l<3;l++)
                          {
                          if (i IS 2)
                          {
                          D[i][j][k][l] = SE[i][j][k][l]
                                + 2.*nu*KD(k,l)*x[i]*I_a[i][j]
                                +(1.-nu)
                                 *(KD(i,l)*x[k]*I_a[k][j]
                                +KD(j,l)*x[k]*I_a[k][i]
                                +KD(i,k)*x[l]*I_a[l][j]
                                +KD(j,k)*x[l]*I_a[l][i]);
                          } else {
                          D[i][j][k][l] = SE[i][j][k][l]
                                + 2.*nu*KD(k,l)*x[i]*I_a[i][j]
                                +(1.-nu)
                                *(KD(i,l)*x[k]*I_a[k][j]
                                +KD(j,l)*x[k]*I_a[k][i]
```

```
                        +KD(i,k)*x[l]*I_a[l][j]
                        +KD(j,k)*x[l]*I_a[l][i])
                        -KD(i,j)*x[k]
                        *(I_a[k][l]-aa[i]*aa[i]
                            *ID_a[k][i][l])
                        -(KD(i,k)*x[j]+KD(j,k)*x[i])
                        *(I_a[j][l]-aa[i]*aa[i]
                            *ID_a[i][j][l])
                        -(KD(i,l)*x[j]+KD(j,l)*x[i])
                        *(I_a[j][k]-aa[i]*aa[i]
                            *ID_a[i][j][k]) -x[i]*x[j]
                        *(I_ab[j][l][k]-aa[i]*aa[i]
                            *ID_ab[i][j][l][k]);
                        D[i][j][k][l] /= (8.*PI*(1.-nu));
                    }
                }
            }
        }
```

/* Calculate constrained strain (Eq. 11.33, Mura, 1987).
The following result holds for both exterior and interior points.
For interior points, llower=0 and all derivatives of Ii and Iij
vanish. Then D[i][j][k][l] becomes S[i][j][k][l] for (llower=0),
which can be found to be equal to (Eq. 11.16, Mura, 1987).*/

```
        for (i=0;i<3;i++)
        {
             for (j=i;j<3;j++)
            {
            strain_F[i][j]=0.;
                for (k=0;k<3;k++)
                {
                    for (l=0;l<3;l++)
                    {
                    strain_F[i][j]+=D[i][j][k][l]*strain_D[k][l];
                    }
                }
            }
        }
```

/* Calculate total stress */

```
        if (llower IS 0)
        {
```

```c
      printf("Inside inclusion %f %f \n",x[0],x[1]);
      G_f = G_I;
      lambda_f= lambda_I;
      nu_f = nu_I;
      E_f = E_I;
    } else {
      G_f = G;
      lambda_f= lambda;
      nu_f = nu;
      E_f = E;
    }
    strainT= strain_F[0][0] + strain_F[1][1] + strain_F[2][2];
    RstrainT= strain[0][0] + strain[1][1] + strain[2][2];
    Osigma[0][0] = (2.*G_f*(strain_F[0][0]+strain[0][0]))
                   + (lambda_f* (RstrainT+strainT));
    Osigma[1][1] = (2.*G_f*(strain_F[1][1]+strain[1][1]))
                  + (lambda_f* (RstrainT+strainT));
    Osigma[2][2] = (2.*G_f*(strain_F[2][2]+strain[2][2]))
                  + (lambda_f* (RstrainT+strainT));
    Osigma[0][1] = 2.*G_f*(strain_F[0][1]+strain[0][1] );
    Osigma[0][2] = 2.*G_f*(strain_F[0][2]+strain[0][2] );
    G_f[1][2] = 2.*G_f*(strain_F[1][2]+strain[1][2] );
    Osigma[1][0]= Osigma[0][1];
    Osigma[2][1]= Osigma[1][2];
    Osigma[2][0]= Osigma[0][2];



 -----------------------------------------------------------------
              /* End of subroutine MURA */
 -----------------------------------------------------------------


}


double KD(i,j)
int i,j;
{
      /* Function that returns the value of the delta Kronecker */
      double r;

      if (i IS j)
      {
       r=1;
      } else {
       r=0;
      }
      return(r);
```

174

```
}

double Delta_b(x)
double x;
{
        /* Define Delta_bar of s
           D(s)=sqrt((aa[0]*aa[0 +s)(aa[1]*aa[1]+s))*/
        double r;

        r=sqrt(aa[0]*aa[0] + x)*sqrt(aa[1]*aa[1] + x);
        return(r);
}

double DlDelt_l(x)
double x;
{
        /* Derivative with respect to lambda of delta_bar of lambda */
        double r;
        int i;

        r=0.;
         for(i=0; i<2; i++)
        {
         r= r + (aa[i]*aa[i] + x);
        }
        r=r*0.5/Delta_b(x) ;
        return(r);
}
```

C.3 Blockrotation3d

The following main program and subroutines were used to compute the slip and rotation of faults described in chapter 3. Two types of reppresentations are used: the Mohr circle and Wulff stereonet projections. The necessary mapping relations between them are also given. Three different models are presented for the evolution of the stress field in a domain undergoing block rotation: a strike-slip, a normal and a reverse faulting model. The most important assumption contained in the present version of this program is that the stress ratio $\phi = (\sigma_2 - \sigma_3)/(\sigma_1 - \sigma_3)$ and principal stress directions are kept constant. Both assumptions can be easily changed if the necessary information is available. In order to change the first one both the MAIN program and the INTERSE subroutine need editing. Another important assumption that can be changed (subroutine SHEAR) is that fault slip occurs along the direction of maximum resolved shear stress.

The structure of the program

1. READ INPUT DATA: "sigma.in" and "pole.in"

2. COMPUTE POLE'S INITIAL ORIENTATION: dip, dipdirection

3. COMPUTE SLIP DIRECTION: rake

4. DEFINE POLE'S STRESS TRAJECTORY AND ITS INTERSECTION WITH THE FRACTURE LINE

5. IF INTERSECTION DOES NOT OCCUR, STOP, OTHERWISE

6. DEFINE NEW STRESS CONFIGURATION

7. IF FRACTURE AND TENSION LIMITS ARE EXCEEDED, STOP, OTHERWISE

8. ROTATE POLE, GO BACK TO 3.

176

```
c-------------------------------------------------------------c
c                 Beginning of program BLOCKROTATION3D
c-------------------------------------------------------------c
c   This is the main program to calculate Block Rotations in 3-D.
c The program reads a stress configuration from file 'sigma.in', and
c fault's stress coordinates (normal and shear stresses) and rock's
c parameters (friction, strength) from file 'pole.in'.
c If the given fault set can rotate, the program computes rotation
c paths within the corresponding Mohr Diagram and in stereographic
c projections (Wulff lower hemisphere). Three different models for
c stress variation are considered in the case of strike-slip, normal
c and thrust faulting, respectively.
c Slip and stress changes (history) are also estimated and  plotted
c along rotation paths. Many stress-history models may be considered:
c here, only a phi(=(s2-s3)/(s1-s3)) value = constant is considered.
c-------------------------------------------------------------c
c-------------------------------------------------------------c
c                              VARIABLES
c-------------------------------------------------------------c
c
c ROTATION
c   ipath        - Number of steps in the rotation path.
c                  Dimension of sigpath and taupath.
c                  Maximum value = 200 !! (ipath=2*irot)
c   irot         - Number of rotation episodes. Dimension of ddrpath,
c                  dippath, radius, drcoul, dpcoul. Maximum value
c                  is =100 (if taking 'dteta' very small, you may
c                  have to increase this number and  'ipath=2*irot'
c                  in the 'dimension'  statements)
c   dteta        - Rotation angle sampling interval
c-------------------------------------------------------------c
c
c STRESS
c    dsigma       - Principal Stresses increment
c   istn,iste,istz- Principal Stress (1,2 or 3) oriented in the
c                   north, east and depth directions in a Stress
c                   Wulff stereographic projection.
c                   Can take only different values among 1, 2 or 3.
c   model        - Model for stress variation. Strike-slip faulting
c                  corresponds to model=1; thrust to model=2 and
c                  normal faulting to model=3
c   r31          - Mohr circles's external radius
c                  [(sigma1-sigma3)/2]
c                  (also equal to the system maximum shear stress)
c   radius(i)    - Succesive values of r31 as stresses change
```

```
c                       (i=1,irot).
c  rmax                - Value of r31 when it reaches the Coulomb fracture
c                        line
c  sigma1,2,3          - Principal stresses magnitudes
c   sig1,2,3in         - PrincipalStresses initial values
c   sigmadir(i,j)      - Angles between principal stresses and
c                        geographical  coordinates (i,j=1,2,3)
c   sigprim1,2,3       - Principal Stresses after stress increments
c                        (sigma' in Estevez's report). Used by the program
c                        as temporary variables
c-----------------------------------------------------------------------c
c
c FAULT GEOMETRY
c    alpha(i)          - Angles between pole's radius vector and Principal
c                        Axes of Stresses (i=1,2,3)
c    angshr(i)         - Angles between shear stress on the fracture and
c                        Principal Axes of Stresses (i=1,2,3)
c    cosang(i)         - Pole's direction cosines in relation to Principal
c                        Stresses (cosines of alpha) (i=1,2,3)
c    csangsh(i,j)      - Direction cosines of the shear stress on the
c                        fracture's plane in relation to principal
c                        stresses. axes (cosines of angshr)
c                        (i=1,2,3; j=1,irot)
c    csangsq(i)        - Cosines of alpha squared (cosang**2)
c    ddir              - Pole's dip direction in Wulff's Projection
c    dip               - Pole's dip in Wulff's Projection
c    ddrpath(i),       - Rotation path's coordinates in Stereographic
c                        Projections
c    dippath(i)        (i=1,irot)
c    plnshr(i)         - Plunge of the shear stress direction on the
c                        fault (i=1,irot).
c    sig               - Fault set pole's normal stress
c    sigint,tauint     - Coordinates of the intersection between pole's
c                        Mohr path and the Coulomb Sliding line
c    sigpath(i),       - Rotation path's coordinates in Mohr's
c                        Configuration
c    tau               - Pole's shear stress
c    taupath(i)        (i=1,ipath; ipath=2*irot, maximum value=200)
c    trnshr(i)         - Trend of the shear stress direction on the fault
c                        (i=1,irot).


c-----------------------------------------------------------------------c
c
c MATERIAL PROPERTIES
c    miuvir,svir       - Virgin rock friction and strength
```

```
c   miufrac,sfrac - Fractured rock friction and strength
c   drcoul(i)     - Coulomb sliding line coordinates (ddir,dip) in
c                   Wulff's space
c   dpcoul(i)       (i=1,irot)
c-----------------------------------------------------------------c
c
c AUXILIARY
c   nfile         - File number
c   rotmtrx(i,j)  - Matrix of the linear equation system for finding
c                   new values of alpha after rotation.(i,j=1,2,3)
c   rotvect(i)    - Right hand side vector of the rotation equation
c                   system (i=1,2,3)
c-----------------------------------------------------------------c
c-----------------------------------------------------------------c
c                         INPUT FILES
c-----------------------------------------------------------------c
c
c 'sigma.in':          model  (1-strike-slip; 2-thrust; 3-normal)
c                      sigma1 sigma2 sigma3
c                      istn iste istz
c                      sigmadir(1,1) sigmadir(1,2) sigmadir(1,3)
c                      sigmadir(2,1) sigmadir(2,2) sigmadir(2,3)
c                      sigmadir(3,1) sigmadir(3,2) sigmadir(3,3)    ,
c
c       where sigmadir(i,j)- angle in degrees between principal
c       stress sigma(i) and axis Xj (X1- north, X2- east, X3- depth)
c
c       Example:
c                      3
c                      1400. 800. 400.
c                      3 1 2
c                      90. 90. 0.
c                      135. 45. 90.
c                      135. 135. 90.
c-----------------------------------------------------------------c
c 'pole.in':           sigma    tau
c                      miuvir svir
c                      miufrac sfrac
c                      dteta (degrees)
c
c       Example:
c                      600. 250.
c                      0.8 500.
c                      0.3 100.
c                      2.
```

179

```
c-------------------------------------------------------------------c
c-------------------------------------------------------------------c
c                        BLOCK SUBROUTINES
c-------------------------------------------------------------------c
c
c   coulomb.f   - This subroutine maps the  Coulomb sliding line
c                 from the Mohr's Space into a Wulff-Stress
c                 Stereoprojection.
c   datainpt.f  - Reads main program's input data from files '
c                 'sigma.in and 'pole.in'
c   interse.f   - Computes the intersection between a pole's stress
c                 trajectory and the Coulomb sliding line as the
c                 system stresses grow.
c                 Outputs intersection's stress coordinates and
c                 system stress's increment.
c   mhrdiag.f   - Computes and writes a file to plot a Mohr's Diagram
c   rotate.f    - Rotates a fault set (pole) and computes its new
c                 direction cosines (alpha) and its Mohr's (sig,tau)
c                 and Wulff's (ddir, dip) coordinates.
c   shear.f     - Computes the direction cosines [csangsh(1,2,3)] of
c                 the shear stress vector on a fault plane.
c   stress.f    - Produces a file to plot normal, shear and system
c                 stress changes and increments during rotation.
c   wlfgeo.f    - Computes the file to plot the rotation path in a
c                 Wulff's Stereonet with geographical coordinates on
c                 its axes.
c-------------------------------------------------------------------c
c-------------------------------------------------------------------c
c                       GENERAL SUBROUTINES
c-------------------------------------------------------------------c
c
c   axesrot     - Computes coordinates's transformations during pure
c                 rotations.
c   lineqn.f    - Solves simultaneous linear equations by
c                 gauss-jordan elimination.
c   mhrstmap.f  - Maps poles between a 3-D Mohr's diagram and a
c                 Wulff stereographic projection with the Principal
c                 Stresses along its axes.
c   mohrcrc.f   - Computes the X,Y coordinates of the 3-D Mohr's
c                 Circles(C32,  C21, C31).
c   wlfcord.f   - Computes Wulff's coordinates "trend and plunge",
c                 given a vector's direction cosines or Wulf's
c                 coordinates "ddir,dip", given a pole's direction
c                 cosines.
c   wulfmap.f   - Maps poles between a Wulff projection with
```

180

```
c              Principal Stresses along its axes and another Wulff
c              projection with standard  N-E-Z geographic axes.
c-------------------------------------------------------------------c
c-------------------------------------------------------------------c
c                              BUGS
c-------------------------------------------------------------------c
c The program does not allowe s1=s2 or s2=s3. Dips greater than 89 e
c degrees ar set =90 degrees, rakes greater than 89 degrees are
c set =90 degrees and rakes smaller than 1 degree are set=0 degrees.
c-------------------------------------------------------------------c
c-------------------------------------------------------------------c
c                            REFERENCES
c-------------------------------------------------------------------c
c
c AUTHOR: Raul Estevez, Geophysics Department, Stanford University.
c                     Sept 1987.
c updated version : Oona Scotti, Stanford University. Sept 1988.
c
c Estevez,R., 1987, "Block Rotations in 3-D". Report, Rock Physics
c              Group, Geophysics Department, Stanford University.
c
c Jaeger,J.C., Cook,N.G., 1976, "Fundamentals of Rock Mechanics".
c                             John Wiley & Sons, Inc., New York.
c
c Nur,A., Ron,H.,Scotti,O., 1986, "Fault Mechanics and the Kinematics
c                 of Block Rotations". GEOLOGY, v.14,p.746-749.
c-------------------------------------------------------------------c
c                        DIMENSION VARIABLES
c-------------------------------------------------------------------c
character*20 title
dimension sigmadir(3,3),alpha(3),cosang(3),csangsq(3)
dimension ddrpath(100),dippath(100),radius(100),csangsh(3,100)
dimension sigpath(200),taupath(200),drcoul(100),dpcoul(100)
 dimension trnshr(100),plnshr(100),strainv(100),strainh(100)
 dimension sigmamax(100),sigmamin(100), rake(100),sradiu(100)
 dimension sn(100),taus(100),borig(100),bradiu(100),sorig(100)
real miuvir,miufrac
c-------------------------------------------------------------------c
c                           OPEN FILES
c-------------------------------------------------------------------c
c
 open (1, file = 'sigma.in', status = 'old')
 open (2, file = 'pole.in', status = 'old')
c-------------------------------------------------------------------c
c                         READ INPUT DATA
```

```
c----------------------------------------------------------------c
c
 nfile1 = 1
 nfile2 = 2
call datainpt(nfile1,model,sigma1,sigma2,sigma3,istn,iste,istz,
     +                    sigmadir,nfile2,sig,tau,miuvir,svir,miufrac,
     +                    sfrac,dteta,movie)
c----------------------------------------------------------------c
c DEFINE CONSTANTS TO CONVERT DEGREES TO RADIANS AND VICE-VERSA.
c ASSIGN INITIAL VALUES
c----------------------------------------------------------------c
c
rad = 3.14159/180.
deg = 180./3.14159
sig1in = sigma1
sig2in = sigma2
sig3in = sigma3
 phiin = (sigma2 - sigma3)/(sigma1 - sigma3)
 radius(1) = 0.5*(sigma1 - sigma3)
 bradiu(1) = 0.5*(sigma1 -sigma3)
 sradiu(1) = 0.5*(sigma2 -sigma3)
 borig(1) = 0.5*(sigma1 +sigma3)
 sorig(1) = 0.5*(sigma3 +sigma2)
 close(1)
close(2)
c----------------------------------------------------------------c
c             INCLUDE POLE'S INITIAL LOCATION INTO MOHR PATH ARRAYS
c----------------------------------------------------------------c
 irot=1
ipath = 1
 sigpath(ipath) = sig
 taupath(ipath) = tau
 sn(irot) = sig
 taus(irot) = tau
c----------------------------------------------------------------c
c                COMPUTE POLE'S INITIAL WULFF COORDINATES
c       COMPUTE POLE'S WULFF-STRESS INITIAL COORDINATES AND DIRECTION
c       COSINES.
c----------------------------------------------------------------c
iflag = -1
call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,ddir,dip,alpha,
     +                    istn,iste,istz,irev)
c
do 15 j=1,3
    cosang(j) = cos(alpha(j))
```

182

```
      csangsq(j) = cosang(j)**2
 15 continue
c----------------------------------------------------------------c
c             COMPUTE DIRECTION OF SHEAR STRESS ON THE FAULT
c----------------------------------------------------------------c
 20 call shear(sigma1,sigma2,sigma3,csangsh(1,irot),cosang,csangsq)
c----------------------------------------------------------------c
c DEFINE POLE'S STRESS TRAJECTORY AND ITS INTERSECTION WITH COULOMB
c SLIDING LINE COMPUTE INTERSECTION COORDINATES sigint, tauint AND
c STRESS INCREMENT dsigma.
c----------------------------------------------------------------c
   call interse(model,sigma1,sigma2,sigma3,sig,tau,miufrac,sfrac,
     +                cosang,csangsq,sigint,tauint,dsigma,iflag)
c
c----------------------------------------------------------------c
c      IF LINES DO NOT INTERSECT, STOP ROTATIONS AND PLOT RESULTS
c----------------------------------------------------------------c
   if(iflag .eq. 1) go to 100
c----------------------------------------------------------------c
c                        FILL MOHR PATH ARRAYS
c----------------------------------------------------------------c
ipath = ipath + 1
sigpath(ipath) = sigint
taupath(ipath) = tauint
 sn(irot) = sigint
 taus(irot) = tauint
 if (model .eq. 1) then
         sx= sigma3
         sy= sigma1
         sz= sigma2
         cl= cosang(3)
         cm= cosang(1)
         cn= cosang(2)
   endif
   if (model .eq. 2) then
         sx= sigma2
         sy= sigma1
         sz= sigma3
         cl= cosang(2)
         cm= cosang(1)
         cn= cosang(3)
   endif
   if (model .eq. 3) then
         sx= sigma3
         sy= sigma2
```

```fortran
      sz= sigmal
      cl= cosang(3)
      cm= cosang(2)
      cn= cosang(1)
   endif
c-----------------------------------------------------------------------c
                 CALCULATE SLIP DIRECTION (RAKE) ON FAULT SURFACE
c-----------------------------------------------------------------------c
      call calcrake(cl,cm,cn,sx,sy,sz,rake(irot))
c-----------------------------------------------------------------------c
c                        DEFINE NEW STRESS CONFIGURATION
c!!!!!!!!!!!!!!!!!!!!!!!!!!!!change here and in interse.f!!!!!!!!!!!!!!!!
c!!!!!!!!!!!!!!!!!!!!!!!when dealing with stress history!!!!!!!!!!!!!!!!!!!!
c-----------------------------------------------------------------------c
if(model .eq. 1) then
      sigprim1 = sigmal + dsigma
      sigprim2 = sigma2
      rt = (sigma2 - sigma3)/(sigmal - sigma2)
      sigprim3 = sigma3 - rt*dsigma
      sigmamax(irot) = sigprim1
      sigmamin(irot) = sigprim3
elseif(model .eq. 2) then
      rt = (sigma2 - sigma3)/(sigmal - sigma3)
      sigprim1 = sigmal + dsigma
      sigprim2 = sigma2 + rt*dsigma
     sigprim3 = sigma3
elseif(model .eq. 3) then
      rt = (sigmal - sigma2)/(sigmal - sigma3)
      sigprim1 = sigmal
       sigprim2 = sigma2 - rt*dsigma
      sigprim3 = sigma3 - dsigma
endif
c-----------------------------------------------------------------------c
c                             CAN FAULT ROTATE ?
c                    COMPUTE EXTERNAL MOHR CIRCLE'S RADIUS.
c-----------------------------------------------------------------------c
r31 = .5*(sigprim1 - sigprim3)
radius(irot) = r31
c-----------------------------------------------------------------------c
c  CHECK IF THIS CIRCLE HAS NOT REACHED THE COULOMB FRACTURE LINE,
cIN WHICH CASE, STOP ROTATIONS.(Estevez, 1987; Relations 18 and 19).
c-----------------------------------------------------------------------c
rmax=.5*(2.*svir+miuvir*(sigprim1+sigprim3))/sqrt(1.+miuvir**2)
  if(r31 .ge. rmax) go to 100
c-----------------------------------------------------------------------c
```

```
c                         COMPUTE TENSION LIMIT.
c-----------------------------------------------------------------c
 if (sigprim3.lt.0.001.or.sigprim2.lt.0.0001.or.sigprim1.lt.0.001)
 +     go to 100
c-----------------------------------------------------------------c
c             COMPUTE NEW PRINCIPAL STRESS MAGNITUDES
c-----------------------------------------------------------------c
sigma1 = sigprim1
sigma2 = sigprim2
sigma3 = sigprim3
 phiin = (sigma2 - sigma3)/(sigma1 - sigma3)
 bradiu(irot) = 0.5*(sigma1 -sigma3)
 sradiu(irot) = 0.5*(sigma2 -sigma3)
 borig(irot) = 0.5*(sigma1 +sigma3)
 sorig(irot) = 0.5*(sigma3 +sigma2)
c-----------------------------------------------------------------c
c                           ROTATE POLE
c     COMPUTE POLE'S NEW DIRECTION COSINES (alpha) AND MOHR (sig,tau)
c             AND STRESS-WULFF (ddir,dip) COORDINATES.
c-----------------------------------------------------------------c
call rotate(sigma1,sigma2,sigma3,alpha,cosang,csangsq,
 +              dteta,csangsh(1,irot),sig,tau,ddir,dip,
 +              istn,iste,istz)
c-----------------------------------------------------------------c
c                       END OF ROTATION EVENT
c         CONTINUE FILLING POLE'S MOHR AND STRESS-WULFF ARRAYS
c-----------------------------------------------------------------c
irot = irot + 1
 ipath = ipath + 1
sigpath(ipath) = sig
taupath(ipath) = tau
 ddrpath(irot) = ddir
 dippath(irot) = dip
 if (model .eq. 1) then
         sx= sigma3
         sy= sigma1
         sz= sigma2
         cl= cosang(3)
         cm= cosang(1)
         cn= cosang(2)
 endif
 if (model .eq. 2) then
         sx= sigma2
         sy= sigma1
         sz= sigma3
```

```
              cl= cosang(2)
              cm= cosang(1)
              cn= cosang(3)
        endif
        if (model .eq. 3) then
              sx= sigma3
              sy= sigma2
              sz= sigma1
              cl= cosang(3)
              cm= cosang(2)
              cn= cosang(1)
         endif
         call calcrake(cl,cm,cn,sx,sy,sz,rake(irot))
c------------------------------------------------------------------------c
c                       GO BACK TO COMPUTE A NEW ROTATION EVENT
c------------------------------------------------------------------------c
go to 20
 100 continue
c------------------------------------------------------------------------c
c                       PLOTTING SUBROUTINES
c------------------------------------------------------------------------c
c                       MOHR   CIRCLE
c------------------------------------------------------------------------c
c
   npoints = 200
   linevir=0
   linefrac=0
   call mhrdiag(npoints,sig1in,sig2in,sig3in,linevir,miuvir,svir
       +               ,linefrac,miufrac,sfrac,ipath,sigpath,taupath)
   do 80 i=1,irot
       dip = dippath(i)
       ddir = ddrpath(i)
       iflag = 1
       call mhrstmap(iflag,sig1in,sig2in,sig3in,sig,tau,
       +               ddir,dip,alpha,istn,iste,istz,irev)
c------------------------------------------------------------------------c
c                       INSERT FILE TO PLOT sig,tau
c------------------------------------------------------------------------c
80       continue
linevir = 1
linefrac = 1
c------------------------------------------------------------------------c
c                       INITIAL VALUES
c------------------------------------------------------------------------c
   call mhrdiag(npoints,sig1in,sig2in,sig3in,linevir,miuvir,svir,
```

```
      +                      linefrac,miufrac,sfrac,ipath,sigpath,taupath)
linevir = 1
linefrac = 1
c-------------------------------------------------------------------c
c                          FINAL VALUES
c-------------------------------------------------------------------c
 call mhrdiag(npoints,sigma1,sigma2,sigma3,linevir,miuvir,svir,
      +                      linefrac,miufrac,sfrac,ipath,sigpath,taupath)
c-------------------------------------------------------------------c
c INCLUDE COULOMB SLIDING LINE FOR INITIAL VALUES OF PRINCIPAL
c STRESSES
c-------------------------------------------------------------------c
npcoul = 0
call coulomb (npcoul,drcoul,dpcoul,sig1in,sig2in,sig3in,
    +        miufrac,sfrac,istn,iste,istz)
c-------------------------------------------------------------------c
c INCLUDE COULOMB SLIDING LINE FOR FINAL VALUES OF PRINCIPAL STRESSES
c-------------------------------------------------------------------c
 npcoul = 0
call coulomb(npcoul,drcoul,dpcoul,sigma1,sigma2,sigma3,
    +                 miufrac,sfrac,istn,iste,istz)
c-------------------------------------------------------------------c
c COMPUTE ROTATION PATH IN WULFF-GEOGRAPHIC COORDINATES.
c-------------------------------------------------------------------c
npcoul = 0
call wlfgeo(sigmadir,irot,ddrpath,dippath,csangsh,
    +             npcoul,drcoul,dpcoul,istn,iste,istz,trnshr,
    +             plnshr,strainv,strainh,rake,sn,taus,borig,bradiu,
    +             sorig,sradiu)
c-------------------------------------------------------------------c
c NORMAL, SHEAR AND SYSTEM STRESSES VARIATIONS AND INCREMENTS d).
c-------------------------------------------------------------------c
call stress(irot,ipath,dteta,sigpath,taupath,radius,strainv,
    +             strainh,sig1in,sigmamax,sigmamin)
stop
end
c-------------------------------------------------------------------c
```

```
c---------------------------------------------------------------c
c                              DATAINPT
c                 THIS SUBROUTINE READS DATA FROM INPUT FILES
c---------------------------------------------------------------c
c---------------------------------------------------------------c
c                              VARIABLES
c---------------------------------------------------------------c
c
c INPUT
c    nfile1        - File number for variables  model; sigma1,2,3;
c                    istn,iste,istz; sigmadir.
c    nfile2        - File number for variables  sig, tau, miuvir,
c                    svir,miufrac, sfrac, dteta, movie.
c---------------------------------------------------------------c
c
c OUTPUT
c    dteta    - Rotation angle sampling interval
c    istn,iste,istz- Principal Stress (1,2 or 3) oriented in the
c                    north, east and depth directions in a Stress
c                    Wulff stereographic projection.
c                    Can take only different values among 1, 2 or 3.
c    miufrac,sfrac - Fractured rock friction and strength
c    miuvir,svir   - Virgin rock friction and strength
c    model         - Model for stress variation (1,2 or 3)
c    sig,tau       - Normal and shear stresses on the initial fault
c                    set.
c    sigma1,2,3    - Principal stresses magnitudes
c    sigmadir(i,j) - Principal stresses's direction cosines in
c                    relation to geographical coordinates (i,j=1,2,3)
c---------------------------------------------------------------c
subroutine datainpt(nfile1,model,sigma1,sigma2,sigma3,istn,iste,
     +                    istz,sigmadir,nfile2,sig,tau,miuvir,svir,
     +                    miufrac,sfrac,dteta,movie)
c---------------------------------------------------------------c
c                 DECLARE AND DIMENSION VARIABLES
c---------------------------------------------------------------c
dimension sigmadir(3,3)
real miuvir,miufrac
c---------------------------------------------------------------c
c  DEFINE CONSTANTS TO CONVERT DEGREES TO RADIANS AND  VICE-VERSA.
c---------------------------------------------------------------c
rad = 3.14159/180.
deg = 180./3.14159
c---------------------------------------------------------------c
c READ PRINCIPAL STRESSES AND ITS N-E-Z ANGLES. CONVERT TO DIRECTION
```

```
c COSINES.
c----------------------------------------------------------------c
read(nfile1,*) model
read(nfile1,*) sigma1, sigma2, sigma3
read(nfile1,*) istn,iste,istz
do 10 i=1,3
     read(nfile1,*) sigmadir(i,1), sigmadir(i,2), sigmadir(i,3)
do 10 j=1,3
   sigmadir(i,j) = cos(sigmadir(i,j) * rad)
10 continue
c----------------------------------------------------------------c
c READ POLE'S INITIAL COORDINATES sigma, tau, FRACTURE/SLIDING
c PARAMETERS, ROTATION ANGLE dteta AND FLAG movie.
c----------------------------------------------------------------c
read(nfile2,*) sig,tau
        read(nfile2,*) miuvir,svir
        read(nfile2,*) miufrac,sfrac
read(nfile2,*) dteta
dteta = dteta * rad
read(nfile2,*) movie
c----------------------------------------------------------------c
c                         DATA VALIDATION
c            COMPUTE MOHR CIRCLES'S RADII AND CENTERS.
c----------------------------------------------------------------c
R32 = (sigma2 - sigma3)/2.
O32 = (sigma2 + sigma3)/2.
O21 = (sigma1 + sigma2)/2.
R21 = (sigma1 - sigma2)/2.
O31 = (sigma1 + sigma3)/2.
R31 = (sigma1 - sigma3)/2.
c----------------------------------------------------------------c
c COMPUTE DISTANCES BETWEEN INITIAL POLE LOCATION AND CIRCLES'S
c CENTERS
c----------------------------------------------------------------c
 d32p = sqrt(tau**2 + (sig - O32)**2)
 d31p = sqrt(tau**2 + (sig - O31)**2)
 d21p = sqrt(tau**2 + (sig - O21)**2)
c----------------------------------------------------------------c
c         VALIDATE INPUT DATA (INITIAL POLE LOCATION).
c----------------------------------------------------------------c
  if(d32p .lt. R32 .or. d31p .gt. R31 .or. d21p .lt. R21) then
         print*
         print*, "Sorry baby, your pole is not defined by your"
         print*, "initial state of stress. The pole is not within"
         print*, "the Mohr circles valid sigma-tau region. "
```

```fortran
          print*, "Try again! Program stops here!!!"
          stop
     endif
     if( tau .gt. (miufrac*sig + sfrac)) then
          print*
          print*, "Sorry baby, your pole is above the Coulomb "
          print*, "fracture line. Try again!."
          print*, "Program stops here!!!"
          stop
     endif
return
end
c----------------------------------------------------------------c
```

```
c---------------------------------------------------------------c
c                            MHRSTMAP
c---------------------------------------------------------------c
c    This subroutine maps poles between a 3D Mohr diagram and a
c Wulff Stereographic Projection with the Principal Stresses along
cits  axes. That is, given a poles's coordinates sigma,tau in the
c Mohr's diagram, the subroutine computes its dip direction (ddir)
c and dip in a Stress-Wulff Projection, and the other way around.
c---------------------------------------------------------------c
c---------------------------------------------------------------c
c                           VARIABLES
c---------------------------------------------------------------c
c   alpha(i)      - Angles between poles and principal axes of
c                   stress. i=1,2,3 correspond to angles with
c                   sigma1,2,3 respectively. Angles are given in
c                   radians.
c   cosang(3)     - Auxiliary variable. Contains pole's direction
c                   cosines.
c   ddir,dip      - Poles's coordinates in Wulff Projection
c                   (dip direction and dip)
c   iflag         - +1 maps from Wulff to Mohr, -1 from Mohr to
c                   Wulff.
c   istn,iste,istz- Principal Stress (1,2 or 3) oriented in the
c                   north, east and depth directions in a Stress
c                   Wulff stereographic projection. Can take only
c                   different values among 1, 2 or 3.
c   irev          - Flag. irev=1 implies that the sense of the
c                   pole's radius vector was reversed during
c                   computations.
c   sigma1,2,3    - Principal stresses magnitudes
c   sig,tau       - Poles's coordinates in Mohr diagram (normal and
c                   shear stresses).
c---------------------------------------------------------------c
c
c INPUT : iflag,sigma1,sigma2,sigma3,istn,iste,istz AND sig,
c                tau (iflag=-1) OR ddir,dip (iflag=1).
c---------------------------------------------------------------c
c
c OUTPUT: alpha; ddir,dip (iflag=-1) OR sig,tau (iflag=1).
c---------------------------------------------------------------c
subroutine mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,ddir,
     +       dip,alpha,istn,iste,istz,irev)
c---------------------------------------------------------------c
c                      DIMENSION VARIABLES
c---------------------------------------------------------------c
```

```
      dimension alpha(3),cosang(3)
c----------------------------------------------------------------c
c         DEFINE CONSTANTS TO CONVERT DEGREES TO RADIANS AND VICE-VERSA
deg = 180./3.14159
 rad = 3.14159/180.
if(iflag .eq. 1) then
c----------------------------------------------------------------c
c                              WULFF --> MOHR
c----------------------------------------------------------------c
c COMPUTE DIRECTION COSINES AND CORRESPONDING ANGLES (eqns. 72 - 75)
c----------------------------------------------------------------c
   theta = (ddir - 180.) * rad
   alpha(istz) = dip * rad
   sinz = sin(alpha(istz))
   cosang(istz) = cos(alpha(istz))
   cosang(iste) = sinz * sin(theta)
   cosang(istn) = sinz * cos(theta)
   alpha(istn) = acos(cosang(istn))
   alpha(iste) = acos(cosang(iste))
c----------------------------------------------------------------c
c COMPUTE NORMAL AND SHEAR STRESSES (eqns. 63,64)
c----------------------------------------------------------------c
    a1 = cosang(1)**2
   a2 = cosang(2)**2
   a3 = cosang(3)**2
  sig = a1*sigma1 + a2*sigma2 + a3*sigma3
  delta = a1*sigma1**2+a2*sigma2**2+a3*sigma3**2 - sig**2
  if(delta .lt. 0.) then
      tau = 0.
  else
     tau = sqrt(delta)
  endif
else
c----------------------------------------------------------------c
c                              MOHR --> WULFF
c----------------------------------------------------------------c
c COMPUTE DIRECTION COSINES (eqns. 65-67)
c----------------------------------------------------------------c
   s12 = sigma1 - sigma2
   s13 = sigma1 - sigma3
   s23 = sigma2 - sigma3
   s1n = sigma1 - sig
   s2n = sigma2 - sig
   s3n = sigma3 - sig
   cosang(1) = sqrt(abs((s2n*s3n + tau**2)/(s12*s13)))
```

```fortran
      cosang(2) = sqrt(abs((s1n*s3n + tau**2)/(-s12*s23)))
      cosang(3) = sqrt(abs((s2n*s1n + tau**2)/(s13*s23)))
c---------------------------------------------------------------------c
cIF THE POLE'S RADIUS VECTOR POINTS UP, TAKE THE OPPOSITE TO THE SAME
c PLANE (SINCE WE ARE WORKING WITH LOWER HEMISPHERE WULFF PROJECTION)
c---------------------------------------------------------------------c
  if(cosang(istz) .lt. -0.01) then
        irev = 1
     do 10 i=1,3
        cosang(i) = -cosang(i)
10    continue
c
  endif
c---------------------------------------------------------------------c
c COMPUTE THE ANGLES CORRESPONDING TO THE DIRECTION COSINES
c---------------------------------------------------------------------c
   do 20 i=1,3
        if (cosang(i) .gt. 0.9999) then
            alpha(i) = 0.0
        else
           alpha(i) = acos(cosang(i))
        endif
20    continue
c---------------------------------------------------------------------c
c COMPUTE DIP (eqn. 71)
c---------------------------------------------------------------------c
           dip = alpha(istz) * deg
c---------------------------------------------------------------------c
c CHECK FOR POSSIBLE SINGULARITIES (eqn. 69)
c---------------------------------------------------------------------c
   sinz = sin(alpha(istz))
    if(abs(sinz).lt.0.01 .and. abs(cosang(istn)).lt.0.01) then
c---------------------------------------------------------------------c
c IF sinz=0 THEN ddir IS UNDEFINED. WE ARBITRARILY SET IT TO 0
c---------------------------------------------------------------------c
       ddir = 0.
     else
c---------------------------------------------------------------------c
c iIF cosang(istn)/sinz > 1.0 CANNOT TAKE  acos
c---------------------------------------------------------------------c
        if (cosang(istn) .gt. sinz ) then
                   ddir= 180.
                   goto 22
        else
c---------------------------------------------------------------------c
```

193

```
c NOTICE THAT acos HAS 2 POSSIBLE SOLUTIONS (theta,360-theta, eqn.69)
c THE SIGN OF cosang(iste) DECIDES WHICH TO TAKE
c------------------------------------------------------------------c
          theta = acos(cosang(istn)/sinz)*deg
           if(cosang(iste) .lt. -0.0001) theta = 360. - theta
c------------------------------------------------------------------c
c COMPUTE DIP DIRECTION (eqn. 70). IF ddir > 360, SUBSTRACT 360
c------------------------------------------------------------------c
          ddir = theta + 180.
         if(ddir .ge. 359.9) ddir = ddir - 359.9
        endif
    endif
c------------------------------------------------------------------c
  endif
22     continue
   return
   end
```

```
c------------------------------------------------------------------c
c                              ROTATE
c------------------------------------------------------------------c
c This subroutine rotates a fault in the direction of the shear
c stress applied on it and computes its pole's new Mohr (sig,tau)
c and Wulff (ddir,dip) coordinates.
c------------------------------------------------------------------c
c------------------------------------------------------------------c
c                             VARIABLES
c------------------------------------------------------------------c
c   alpha(i)      - Angles between pole's radius vector and Principal
c                   Axes of Stresses (i=1,2,3)
c   cosang(i)     - Pole's direction cosines in relation to Principal
c                   Stresses (cosines of alpha) (i=1,2,3)
c   csangsh(i)    - Direction cosines of the shear stress on the
c                   fracture's plane in relation to principal
c                   stress axes. (cosines of angshr)
c                   (i=1,2,3; j=1,irot)
c   csangsq(i)    - Cosines of alpha squared (cosang**2)
c   ddir          - Pole's dip direction in Wulff's Projection
c   dip           - Pole's dip in Wulff's Projection
c   dteta    - Rotation angle sampling interval
c   istn,iste,istz- Principal Stress (1,2 or 3) oriented in the
c                   north, east and depth directions in a Stress
c                   Wulff stereographic projection. Can take only
c                   different values among 1, 2 or 3.
c   sigma1,2,3    - Principal stresses magnitudes
c   sig           - Fault set pole's normal stress
c   rotmtrx(i,j)  - Matrix of the linear equation system for finding
c                   new values of alpha after rotation.(i,j=1,2,3)
c   rotvect(i)    - Right hand side vector of the rotation equation
c                   system (i=1,2,3)
c   tau           - Pole's shear stress
c   vn            - Vector perpendicular to the pole and the total
c                   stress vector
c------------------------------------------------------------------c
c
c INPUT VARIABLES: cosang, csangsh, dteta, sigma1, sigma2, sigma3
c------------------------------------------------------------------c
c
c OUTPUT VARIABLES: alpha, cosang, csangsq, ddir, dip, sig, tau
c------------------------------------------------------------------c
c                        REQUIRED SUBROUTINES
c------------------------------------------------------------------c
c   lineqn.f     - Solves simultaneous linear equations by
```

```
c                    gauss-jordan elimination.
c    wlfcord.f    - Computes Wulff's coordinates "trend and plunge",
c                    given a vector's direction cosines or Wulf's
c                    coordinates "ddir,dip", given a pole's direction
c                    cosines.
c-----------------------------------------------------------------------c
   subroutine rotate(sigma1,sigma2,sigma3,alpha,cosang,csangsq,
  +                  dteta,csangsh,sig,tau,ddir,dip,istn,iste,istz)
c-----------------------------------------------------------------------c
c                    DIMENSION VARIABLES
c-----------------------------------------------------------------------c
dimension alpha(3),cosang(3),csangsq(3),csangsh(3)
dimension rotmtrx(3,3),vn(3),rotvect(3)
c-----------------------------------------------------------------------c
c COMPUTE VECTOR PERPENDICULAR TO THE POLE AND THE TOTAL STRESS
c VECTORS              (Eqns. 54,55,56 in Estevez's report)
c-----------------------------------------------------------------------c
vn(1) = cosang(2) * cosang(3) * (sigma3-sigma2)
vn(2) = cosang(1) * cosang(3) * (sigma1-sigma3)
vn(3) = cosang(2) * cosang(1) * (sigma2-sigma1)
c-----------------------------------------------------------------------c
c  COMPUTE UNITARY VECTOR (DIRECTION COSINES) IN THIS DIRECTION (vn)
c-----------------------------------------------------------------------c
amodn = sqrt(vn(1)**2 + vn(2)**2 + vn(3)**2)
do 10 i=1,3
     vn(i) = vn(i)/amodn
10 continue
c-----------------------------------------------------------------------c
c    FILL MATRIX AND VECTOR FOR SOLVING THE ROTATION EQUATION SYSTEM
c                  (Eqns. 60, 61, 62)
c-----------------------------------------------------------------------c
do 20 i=1,3
   rotmtrx(1,i) = cosang(i)
   rotmtrx(2,i) = csangsh(i)
   rotmtrx(3,i) = vn(i)
20 continue
rotvect(1) = cos(dteta)
rotvect(2) = sin(dteta)
rotvect(3) = 0.
c-----------------------------------------------------------------------c
c COMPUTE NEW DIRECTION COSINES FOR THE ROTATED POLE
c-----------------------------------------------------------------------c
call lineqn(rotmtrx,rotvect,3,3,ierr)
do 40 j=1,3
     cosang(j) = rotvect(j)
```

```fortran
      csangsq(j) = cosang(j)**2
      alpha(j) = acos(cosang(j))
40 continue
c-------------------------------------------------------------c
c COMPUTE WULFF(ddir, dip) AND MOHR(sig,tau) COORDINATES.
c NOTE THAT WE ARE ASSUMING THAT AFTER ROTATION PRINCIPAL STRESSES
c MAGNITUDES DROP TO THE VALUE THEY HAD BEFORE. BE AWARE THAT THIS
c IS THE PLACE  WHERE YOU MAY INCLUDE BETTER ASUMPTIONS ABOUT STRESS
c DROP.
c-------------------------------------------------------------c
ipole = 1
call wlfcord(ipole,alpha,cosang,trend,plunge,ddir,dip,
     +                istn,iste,istz,irev)
sig = csangsq(1)*sigma1 + csangsq(2)*sigma2 +
     +          csangsq(3)*sigma3
tau = sqrt(csangsq(1)*sigma1**2 + csangsq(2)*sigma2**2 +
     +              csangsq(3)*sigma3**2 - sig**2)
return
end
c-------------------------------------------------------------c
```

```
c----------------------------------------------------------------c
c                              LINEQN
c----------------------------------------------------------------c
c This subroutine solve simultaneous equns by gauss-jordan
c elimination
c----------------------------------------------------------------c
c----------------------------------------------------------------c
c    arguments: a.......m cols x n rows matrix of coefficients
c of the system of equations  [a}x[x]=[b].....(1)
c b.......n rows matrix of [b] coefficients of equation (1)
c n.......number of equations
c m.......number of unknowns
c the matrix should be set up as illustraled below:
c ! a11 a12 a13.....a1m !   ! x1 !   ! b1 !
c ! a21 a22 a23.....a2m !   ! x2 !   ! b2 !
c ! a31 a32 a33.....a3m ! x ! x3 ! = ! b3 !
c !  .   .   ...... .   !   ! . !   ! . !
c !  .   .   ...... .   !   ! . !   ! . !
c ! an1 an2 an3.....anm !   ! xn !   ! bn !
c ier.....error return code:  1 = normal
c -1 = m # n (equations solved
c -2 = singular matrix
c----------------------------------------------------------------c
subroutine lineqn(a,b,n,m,ier)
 real a(n,m),b(n)
real aij,akj,swap
real piv
c----------------------------------------------------------------c
c initialize number of solutions to zero
c----------------------------------------------------------------c
npiv=0
c----------------------------------------------------------------c
c loop to reduce the equations
c----------------------------------------------------------------c
do 10 j=1,m
if(npiv.ge.n) goto 70
piv=0.
ipiv=0
i0=npiv+1
c----------------------------------------------------------------c
c loop to locate maximum pivot for column j
c----------------------------------------------------------------c
do 20 i=i0,n
absaij=abs(a(i,j))
if(absaij.lt.piv) goto 20
```

```
piv=absaij
ipiv=i
20 continue
if(ipiv.eq.0) goto 10
i=ipiv
aij=a(i,j)
c------------------------------------------------------------c
c divide pivot row by pivot matrix element a(i,j)
c------------------------------------------------------------c
do 30 l=1,m
a(i,l)=a(i,l)/aij
30 continue
b(i)=b(i)/aij
c------------------------------------------------------------c
c perform pivoting operation
c------------------------------------------------------------c
do 40 k=1,n
if(k.eq.i) goto 40
akj=a(k,j)
do 50 l=1,m
a(k,l)=a(k,l)-a(i,l)*akj
50 continue
b(k)=b(k)-b(i)*akj
40 continue
c------------------------------------------------------------c
c evaluate the number of pivots located
c------------------------------------------------------------c
npiv=npiv+1
c------------------------------------------------------------c
c interchange rows if necessary
c------------------------------------------------------------c
if(i.eq.j) goto 10
do 60 l=1,m
swap=a(i,l)
a(i,l)=a(j,l)
a(j,l)=swap
60 continue
swap=b(i)
b(i)=b(j)
b(j)=swap
10 continue
c------------------------------------------------------------c
c evaluate error return code
c------------------------------------------------------------c
70 continue
```

```
      ier=1
      if(m.ne.n) ier=-1
      if(npiv.ne.n) ier=-2
      return
      end
c--------------------------------------------------------------------c
```

```
c---------------------------------------------------------------c
c                            WLFCORD
c---------------------------------------------------------------c
c    This subroutine allows to compute Stress-Wulff coordinates "trend
c    and  plunge", given a vector's direction cosines (ipole=0) or
c    Stress-Wulff c coordinates "ddir,dip", given a pole's direction
c    cosines (ipole=1). irev=1 implies that the direction of the
c    vector (pole) was reversed so that the end pointed
c    down (lower hemisphere projection). Otherwise, c irev=0.
c---------------------------------------------------------------c
c---------------------------------------------------------------c
c                            VARIABLES
c---------------------------------------------------------------c
c    alpha(i)      - Angles between pole's radius vector and Principal
c                    Axes of Stresses (i=1,2,3)
c    cosang(i)     - Pole's direction cosines in relation to Principal
c                    Stresse  (cosines of alpha) (i=1,2,3)
c    ddir          - Pole's dip direction in Wulff's Projection
c    dip           - Pole's dip in Wulff's Projection
c    ipole         - Flag. If ipole=0 computes coordinates "trend &
c                    plunge" for a vector. If ipole=1, computes "ddir
c                    and dip" for a  pole.
c    irev          - Flag. irev=1 implies that the sense of the vector
c                    (pole) was reversed during computations.
c    istn,iste,istz- Principal Stress (1,2 or 3) oriented in the
c                    north, east and depth directions in a Stress
c                    Wulff stereographic  projection.
c                    Can take only different values among 1, 2 or 3.
c    plunge        - Line's (vector) plunge in Stress-Wulff
c                    Projection.
c    sinz          - Sinus of alpha(istz).
c    trend         - Line's (vector) trend in Stress-Wulff Projection.
c---------------------------------------------------------------c
c INPUT : alpha, cosang, ipole, istn, iste, istz.
c---------------------------------------------------------------c
c OUTPUT : irev and ddir & dip (ipole=1) or trend & plunge (ipole=0).
c---------------------------------------------------------------c
c AUXILIARY VARIABLES: sinz
c---------------------------------------------------------------c
subroutine wlfcord(ipole,alpha,cosang,trend,plunge,ddir,dip,
     +                    istn,iste,istz,irev)
c---------------------------------------------------------------c
c             DIMENSION AND INITIALIZE VARIABLES
c---------------------------------------------------------------c
dimension alpha(3),cosang(3)
```

```
      irev = 0
c-------------------------------------------------------------c
c              DEFINE CONSTANT TO CONVERT RADIANS TO DEGREES
c-------------------------------------------------------------c
      deg = 180./3.14159
c-------------------------------------------------------------c
c              IF THE VECTOR (POLE) IS POINTING TOWARDS THE UPPER
c              HEMISPHERE,REVERSE IT.
c-------------------------------------------------------------c
      if(cosang(istz) .lt. -0.0001) then
        irev = 1
        do 10 i=1,3
             cosang(i) = -cosang(i)
             alpha(i) = acos(cosang(i))
10      continue
      endif
c-------------------------------------------------------------c
c                       COMPUTE PLUNGE...
c-------------------------------------------------------------c
      if(ipole .eq. 0) then
          plunge = 90. - alpha(istz)*deg
      else
c-------------------------------------------------------------c
c ...OR DIP (eqn. 71 in Estevez's report)
c-------------------------------------------------------------c
              dip = alpha(istz) * deg
      endif
c-------------------------------------------------------------c
c        CHECK FOR POSSIBLE SINGULARITIES (in relation to eqn. 69)
c-------------------------------------------------------------c
      sinz = sin(alpha(istz))
       if(abs(sinz).lt.0.01 .and. abs(cosang(istn)).lt.0.01) then
c-------------------------------------------------------------c
c              IF sinz = 0, trend AND ddir ARE UNDEFINED.
c                   WE ARBITRARILY SET THEN TO 0...
c-------------------------------------------------------------c
        trend = 0.
         ddir = 0.
      else
c-------------------------------------------------------------c
c              ...OTHERWISE, COMPUTE THETA (eqn. 69)
c-------------------------------------------------------------c
          if (cosang(istn) .gt. sinz) cosang(istn) = sinz
         theta = acos(cosang(istn)/sinz)*deg
c-------------------------------------------------------------c
```

```
c             ACOS HAS 2 POSSIBLE SOLUTIONS (theta & 360-theta).
c             THE SIGN OF cosang(iste) DECIDES WHICH TO TAKE
c-----------------------------------------------------------------------c
      if(cosang(iste) .lt. -0.0001) theta = 360. - theta
c-----------------------------------------------------------------------c
c COMPUTE TREND OR DDIR (eqn. 70). IF ANGLES ARE BIGGER THAN 360,
c                     SUBSTRACT 360
c-----------------------------------------------------------------------c
      if(ipole .eq. 0) then

         trend = theta
            if(trend .ge. 359.9) trend = trend - 359.9
      else
           ddir = theta + 180.
            if(ddir .ge. 359.9) ddir = ddir - 359.9
      endif
   endif
   return
   end
c-----------------------------------------------------------------------c
```

```
c--------------------------------------------------------------------c
c                              INTERSE
c--------------------------------------------------------------------c
c    This subroutnie computes the intersection between a pole's stress
c trajectory and the Coulomb sliding line as the system stresses
c grow. It outputs intersection's stress coordinates and system
c stress's increment.
c--------------------------------------------------------------------c
c--------------------------------------------------------------------c
c                              VARIABLES
c--------------------------------------------------------------------c
c
c INPUT:
c    cosang(i)      - Pole's direction cosines in relation to Principal
c                     Stresses (cosines of alpha) (i=1,2,3)
c    csangsq(i)     - Cosines of alpha squared (cosang**2)
c    miufrac,sfrac  - Fractured rock friction and strength
c    model          - Model for stress variation (1,2 or 3)
c    sig, tau       - Fault set pole's normal and shear stresses
c    sigma1,2,3     - Principal stresses magnitudes
c--------------------------------------------------------------------c
c
c OUTPUT:
c    dsigma         - Principal Stresses increment
c    iflag          - Flag. When equal 1 means that there is not
c                     intersection and rotation can't occur. Otherwise
c                     its value is set to 0
c    sigint,tauint  - Coordinates of the intersection between pole's
c                     Mohr path and the Coulomb Sliding line
c--------------------------------------------------------------------c
      subroutine interse(model,sigma1,sigma2,sigma3,sig,tau,miufrac,
     +                   sfrac,cosang,csangsq,sigint,tauint,dsigma,
     +                   iflag)
c--------------------------------------------------------------------c
c                DECLARE AND DIMENSION VARIABLES
c--------------------------------------------------------------------c
      dimension cosang(3),csangsq(3)
      real miufrac, infinite
      iflag = 0
      infinite = 1.0e+10
c--------------------------------------------------------------------c
c            DEFINE POLE'S STRESS TRAJECTORY IN MOHR SPACE
c            COMPUTE POLE TRAJECTORY EQUATIONS'S COEFFICIENTS
c--------------------------------------------------------------------c
c MODEL 1: PURE SHEAR ( SIGMA2- FIXED ).
```

```
c (Relations 11,12,13, pag. 8 in Estevez's Report)
c-----------------------------------------------------------------c
if(model .eq. 1) then
           r = (sigma2 - sigma3)/(sigma1 - sigma2)
           at = csangsq(1) - r*csangsq(3)
           bt = 2.*(csangsq(1)*(sigma1 - sig) -
     +          r*csangsq(3)*(sigma3 - sig))
           ct = csangsq(1) + (r**2)*csangsq(3) - at**2
c-----------------------------------------------------------------c
c MODEL 2: THRUST FAULTING ( SIGMA3- FIXED ).
c (Relations 34,36,37,38 in Estevez's Report)
c-----------------------------------------------------------------c
elseif(model .eq. 2) then
     r = (sigma2 - sigma3)/(sigma1 - sigma3)
     at = csangsq(1) + r*csangsq(2)
         bt = 2.*((sigma1 - sig)*csangsq(1) +
     +          r*(sigma2 - sig)*csangsq(2))
     ct = ((1.-r)**2)*csangsq(1)*csangsq(2) +
     +          csangsq(3)*(csangsq(1) + csangsq(2)*(r**2))
c-----------------------------------------------------------------c
c MODEL 3: NORMAL FAULTING ( SIGMA1- FIXED ).
c (Relations 44,47,49,50)
c-----------------------------------------------------------------c
elseif(model .eq. 3) then
     r = (sigma1 - sigma2)/(sigma1 - sigma3)
     at = - csangsq(3) - r*csangsq(2)
         bt = - 2.*((sigma3 - sig)*csangsq(3) +
     +          r*(sigma2 - sig)*csangsq(2))
     ct = ((1.-r)**2)*csangsq(3)*csangsq(2) +
     +          csangsq(1)*(csangsq(3) + csangsq(2)*(r**2))
endif
c-----------------------------------------------------------------c
c CHECK FOR POSSIBLE SINGULARITIES (at =0. or cosang(1)=cosang(3))
c-----------------------------------------------------------------c
if(abs(at). gt. .0001) then
c-----------------------------------------------------------------c
c  COMPUTE MINIMUM OF THE POLE'S TRAJECTORY (sigmin, Eqns. 16,19)
c-----------------------------------------------------------------c
   at2 = at**2
   B = at*bt - 2.*ct*sig
   sigmin = -B/(2.*ct)
c-----------------------------------------------------------------c
c          COMPUTE INTERSECTION'S COORDINATES AND STRESS INCREMENT
c-----------------------------------------------------------------c
c COMPUTE INTERSECTION EQUATION'S COEFFICIENTS
```

```fortran
c ( Equations 15, 16, 17, 20)
c----------------------------------------------------------------c
      A = ct - at2*miufrac**2
      B = B - 2.*at2*miufrac*sfrac
      C = at2*tau**2 - at*bt*sig + ct*sig**2 - at2*sfrac**2
c----------------------------------------------------------------c
c COMPUTE EQUATION'S RADICAL AND CHECK IF INTERSECTION OCCURS
c----------------------------------------------------------------c
      delta = B**2 - 4.*A*C
      if(delta .lt. 0. .or. abs(A) .lt. .0001) then
c----------------------------------------------------------------c
c TRAJECTORY DOES NOT INTERSECT SLIDING LINE
c----------------------------------------------------------------c
         iflag = 1
         return
      else
c----------------------------------------------------------------c
c COMPUTE EQUATION'S TWO SOLUTIONS (TWO INTERSECTIONS)...
c----------------------------------------------------------------c
         sigfrac0 = -sfrac/miufrac
         sd = sqrt(delta)
         sig1 = (-B + sd)/(2.*A)
          if(sig1 .le. sigfrac0) sig1 = infinite
          sig2 = (-B - sd)/(2.*A)
          if(sig2 .le. sigfrac0) sig2 = infinite
c----------------------------------------------------------------c
c...AND DECIDE WHICH ONE TO KEEP DEPENDIG WETHER THE POLE IS AT THE
c RIGTH OR LEFT OF THE TRAJECTORY'S MINIMUM
c (See discussion on pag.9 of Estevez's rep.)
c----------------------------------------------------------------c
         if(sig .ge. sigmin) then
                 sigint = amax1(sig1,sig2)
         else
                 sigint = amin1(sig1,sig2)
         endif
c----------------------------------------------------------------c
c COMPUTE INTERSECTION'S SHEAR STRESS AND SYSTEM STRESS INCREMENT
c ( Eqns. 18 and 21)
c----------------------------------------------------------------c
         tauint = miufrac * sigint + sfrac
         dsigma = (sigint - sig)/at
      endif
else
c----------------------------------------------------------------c
c IF at OR ct HAVE INDEED SINGULAR VALUES ( at=0 and/or ct=0 )
```

```
c ( Equations 25,26 ...)
c------------------------------------------------------------------c
      sigint = sig
      tauint = miufrac * sigint + sfrac
      dtausqr = tauint**2 - tau**2
c------------------------------------------------------------------c
c (...and equation 10)
c------------------------------------------------------------------c
      if(abs(ct) .lt. 0.0001) then
         dsigma = dtausqr/bt
      else
         delta = bt**2 + 4.*ct*dtausqr
         dsigma = (-bt + sqrt(delta))/(2.*ct)
      endif
   endif
   return
   end
c------------------------------------------------------------------c
```

```
c------------------------------------------------------------------c
c                           COULOMB
c------------------------------------------------------------------c
c      This subroutine maps the "sliding" Coulomb line i from the
c Mohr's space into a Wulff stress projection. Only the segment
c inside the Mohr circles is projected.In order to compute the
c segment we have to find the intersection between the Coulomb line
c and the external Mohr circle. The sigmac coordinates of this
c intersection satisfy the second degree equation:
c                         As**2 + Bs + C = 0,
c where
c            A = 1 + miufrac**2
c            B = 2miufrac*sfrac - sigma1 - sigma3
c            C = sigma1*sigma3 + sfrac**2
c    Notice that the shape of this line in the Wulff projection highly
c depends on the stress configuration. Thus, although in Mohr space
c the line is a unique straight line, its projection in Wulff space
c the depends on principal stresses magnitudes at the moment when the
c projection is performed.
c------------------------------------------------------------------c
c------------------------------------------------------------------c
c                           VARIABLES
c------------------------------------------------------------------c
c INPUT:
c   istn,iste,istz- Principal Stress (1,2 or 3) oriented in the
c                   north, east and depth directions in a Stress
c                   Wulff stereographic projection.
c                   Can take only different values among 1, 2 or 3.
c   miufrac,sfrac - Fractured rock friction and strength
c   npoint        - Number of coordinate points to be considered.
c                   Arrays'sdimension.
c   sigma1,2,3    - Principal stresses magnitudes
c------------------------------------------------------------------c
c
c OUTPUT:
c   ddir(i),dip(i)- Coulomb line's coordinates dip direction (ddir)
c                   and dip in a stress-Wulff stereographic
c                   projection (i=1,npoint).
c------------------------------------------------------------------c
c
c AUXILIARY:
c   a, b, c       - Coefficients of the second degree intersection
c                   equation
c   alpha(i)      - If we interpret Coulomb line's coordinate points
c                   as "poles", these are the angles between a
```

```
c                        pole's (i=1,2,3) radius vector and the Stress
c                        Principal Axes
c   delta             - Discriminant of the equation
c    dsig             - sampling interval to plot the Coulomb line
c    irev             - Flag. irev=1 implies that the sense of the
c                        pole's radius vector was reversed during
c                        computations.
c  sig,tau            - Coulomb line's coordinates in Mohr space (normal
c                        and shear stresses).
c   sig1, sig2        - Roots (intersection coordinates) of the
c                        intersection eqn.
c-----------------------------------------------------------------c
c                        REQUIRED SUBROUTINES
c-----------------------------------------------------------------c
c
c   mhrstmap.f  - Maps poles between a 3-D Mohr's diagram and a
c                 Wulff stereographic projection with the Principal
c                 Stresses along its axes.
c-----------------------------------------------------------------c
subroutine coulomb(npoint,ddir,dip,sigma1,sigma2,sigma3,
     +                    miufrac,sfrac,istn,iste,istz)
c-----------------------------------------------------------------c
c                        DIMENSION VARIABLES
c-----------------------------------------------------------------c
dimension alpha(3)
dimension ddir(npoint),dip(npoint)
real miufrac
c-----------------------------------------------------------------c
c FIND INTERSECTION BETWEEN COULOMB LINE AND MOHR CIRCLES
c (SOLVE SECOND DEGREE EQUATION, MENTIONED AT THE BEGINNING)
c-----------------------------------------------------------------c
a = 1. + miufrac**2
b = 2.*miufrac*sfrac - sigma1 - sigma3
b12 = 2.*miufrac*sfrac - sigma1 - sigma2
b23 = 2.*miufrac*sfrac - sigma3 - sigma2
c = sigma1*sigma3 + sfrac**2
c12 = sigma1*sigma2 + sfrac**2
c23 = sigma3*sigma2 + sfrac**2
delta = b**2 - 4.*a*c
delta12 = b12**2 - 4.*a*c12
delta23 = b23**2 - 4.*a*c23
c-----------------------------------------------------------------c
c IF THERE IS NO INTERSECTION, SET OUTPUT TO ZERO AND RETURN TO MAIN
c PROGRAM
c-----------------------------------------------------------------c
```

```fortran
      if (delta .le. 0.) then
c-------------------------------------------------------------c
c                       NO INTERSECTION
c-------------------------------------------------------------c
      do 5 i=1,npoint
         ddir(i) = 90.
        dip(i) = 90.
5 continue
        return
      else
c-------------------------------------------------------------c
c         OTHERWISE, FIND THE ROOTS (INTERSECTION COORDINATES) OF THE
c         EQUATION
c-------------------------------------------------------------c
      sdelta = sqrt(delta)
      sig1 = (-b + sdelta)/(2.*a)
      sig2 = (-b - sdelta)/(2.*a)
      endif
c-------------------------------------------------------------c
c                COMPUTE COULOMB LINE'S WULFF COORDINATES
c-------------------------------------------------------------c
      dsig = (sig1 - sig2)/float(npoint - 1)
      sig = sig2 - dsig
      if (delta12 .le. 0. .and. delta23 .le. 0.) then
c-------------------------------------------------------------c
c 12 and 23 CIRCLES DON"T INTERCEPT FRACTURE LINE
c-------------------------------------------------------------c
        do 10 i=1,npoint
           sig = sig + dsig
          tau = sfrac + miufrac*sig
          iflag = -1
          call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                        ddir(i),dip(i),alpha,istn,iste,istz,irev)
10      continue
        endif
        if ( delta12 .le. 0. .and. delta23 .gt. 0.) then
c-------------------------------------------------------------c
c ONLY 23 circle INTERCEPTS
c-------------------------------------------------------------c
      sdelta = sqrt(delta23)
      sig31 = (-b23 + sdelta)/(2.*a)
      sig32 = (-b23 - sdelta)/(2.*a)
      do 20 i=1,npoint
        sig = sig + dsig
         if (sig .gt. sig32 .and. sig .lt. sig31) then
```

```
                  sig = sig32
                   mpoint = i
                  tau = sfrac + miufrac*sig
                  iflag = -1
                  call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                          ddir(i),dip(i),alpha,istn,iste,istz,irev)
                   sig = sig31
                  tau = sfrac + miufrac*sig
                  iflag = -1
                  call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                          ddir(i),dip(i),alpha,istn,iste,istz,irev)
                   dsig = (sig1 -sig31)/float(npoint - mpoint)
                   goto 20
             endif
           tau = sfrac + miufrac*sig
           iflag = -1
           call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                          ddir(i),dip(i),alpha,istn,iste,istz,irev)
20    continue
      endif
      if ( delta12 .gt. 0. .and. delta23 .le. 0.) then
c-----------------------------------------------------------------c
c ONLY 12 CIRCLE INTERCEPTS
c-----------------------------------------------------------------c
      sdelta = sqrt(delta12)
      sig21 = (-b12 + sdelta)/(2.*a)
      sig22 = (-b12 - sdelta)/(2.*a)
       do 30 i=1,npoint
         sig = sig + dsig
          if (sig .gt. sig22 .and. sig .lt. sig21 ) then
                mpoint = i
                sig = sig22
              tau = sfrac + miufrac*sig
              iflag = -1
              call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                          ddir(i),dip(i),alpha,istn,iste,istz,irev)
               sig = sig21
              tau = sfrac + miufrac*sig
              iflag = -1
              call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                          ddir(i),dip(i),alpha,istn,iste,istz,irev)
               dsig = (sig1 - sig21)/float(npoint - mpoint)
               goto 30
          endif
       tau = sfrac + miufrac*sig
```

```fortran
      iflag = -1
      call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                        ddir(i),dip(i),alpha,istn,iste,istz,irev)
30    continue
      endif
      if ( delta12 .gt. 0. .and. delta23 .gt. 0.) then
c-------------------------------------------------------------------c
c 12 and 23 CIRCLES INTERCEPTS
c-------------------------------------------------------------------c
      sdelta = sqrt(delta23)
      sig31 = (-b23 + sdelta)/(2.*a)
      sig32 = (-b23 - sdelta)/(2.*a)
      sdelta = sqrt(delta12)
      sig21 = (-b12 + sdelta)/(2.*a)
      sig22 = (-b12 - sdelta)/(2.*a)
      do 40 i=1,npoint
      sig = sig + dsig
       if (sig .gt. sig32 .and. sig .lt. sig31) then
              sig = sig32
              mpoint = i
           tau = sfrac + miufrac*sig
           iflag = -1
           call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                     ddir(i),dip(i),alpha,istn,iste,istz,irev)
            sig = sig31
           tau = sfrac + miufrac*sig
           iflag = -1
           call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                     ddir(i),dip(i),alpha,istn,iste,istz,irev)
             dsig = (sig1 - sig31)/float(npoint - mpoint)
      endif
      if (sig .gt. sig22 .and. sig .lt. sig21) then
              sig = sig22
              mpoint = i
           tau = sfrac + miufrac*sig
           iflag = -1
           call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                     ddir(i),dip(i),alpha,istn,iste,istz,irev)
             sig = sig 21
           tau = sfrac + miufrac*sig
           iflag = -1
           call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                     ddir(i),dip(i),alpha,istn,iste,istz,irev)
               dsig = (sig1 - sig21)/float(npoint - mpoint)
      endif
```

```
      tau = sfrac + miufrac*sig
      iflag = -1
     call mhrstmap(iflag,sigma1,sigma2,sigma3,sig,tau,
     +                  ddir(i),dip(i),alpha,istn,iste,istz,irev)
40 continue
   endif
return
end
c-------------------------------------------------------------c
```

213

```
c------------------------------------------------------------------c
c                               SHEAR
c------------------------------------------------------------------c
c     This subroutine computes the coordinates [csangsh(1,2,3)-
c direction cosines in relation to principal stresses] of the maximum
c shear stress vector on a plane defined by its pole's direction
c cosines [cosang(1,2,3)].
c------------------------------------------------------------------c
c------------------------------------------------------------------c
c                             VARIABLES
c------------------------------------------------------------------c
c   sigma1,2,3    - Principal stresses modulii
c   cosang(i)     - Pole's direction cosines in relation to Principal
c                   Stresses (cosines of alpha) (i=1,2,3)
c   csangsh(i,j)  - Direction cosines of the maximum shear stress on
c                   the fracture's plane in relation to principal
c                   stresses axes. (cosines of angshr)
c                   (i=1,2,3; j=1,irot)
c   csangsq(i)    - Cosines of alpha squared (cosang**2)
c------------------------------------------------------------------c
c
c INPUT: cosang, csangsq, sigma1, sigam2, sigma3
c------------------------------------------------------------------c
c
c OUTPUTS: csangsh
c------------------------------------------------------------------c
subroutine shear(sigma1,sigma2,sigma3,csangsh,cosang,csangsq)
c------------------------------------------------------------------c
c                        DIMENSION VARIABLES
c------------------------------------------------------------------c
dimension csangsh(3),cosang(3),csangsq(3)
c------------------------------------------------------------------c
c          COMPUTE VECTOR PROPORTIONAL TO THE MAXIMUM SHEAR STRESS
c (Equations 57, 58, 59 in Estevez's report)
c------------------------------------------------------------------c
csangsh(1) = cosang(1) * (csangsq(3)*(sigma1-sigma3) -
     +                csangsq(2)*(sigma2-sigma1))
csangsh(2) = cosang(2) * (csangsq(1)*(sigma2-sigma1) -
     +                csangsq(3)*(sigma3-sigma2))
csangsh(3) = cosang(3) * (csangsq(2)*(sigma3-sigma2) -
     +                csangsq(1)*(sigma1-sigma3))
c------------------------------------------------------------------c
c COMPUTE UNITARY VECTOR (DIRECTION COSINES) IN THIS DIRECTION
c------------------------------------------------------------------c
amodshr = sqrt(csangsh(1)**2 + csangsh(2)**2 + csangsh(3)**2)
```

```
do 10 i=1,3
    csangsh(i) = csangsh(i)/amodshr
10 continue
return
end
c-----------------------------------------------------------------------c
```

```
c-------------------------------------------------------------c
c                         CALCRAKE
c-------------------------------------------------------------c
c   This subroutine calculates the slip direction on a fault using
c    Bott's equation.
c-------------------------------------------------------------c
c-------------------------------------------------------------c
c                         VARIABLES
c-------------------------------------------------------------c
c
c INPUT:   cl,cm,cn:  direction cosines of the fault's normal
c          sx,sy,sz:  horizontal and vertical components of stress
c-------------------------------------------------------------c
c
c OUTPUT   rake:   slip direction on the fault
c-------------------------------------------------------------c
      subroutine calcrake(cl,cm,cn,sx,sy,sz,rake)
      PI = 3.1415927
      if ( sz .gt. sy .and. sy .eq. sx) then
         rake = 90.
      elseif(  sx .gt. sz .and. sy .eq. sx) then
         rake = -90.
      elseif(cl .eq. 0) then
         if(sz .gt. sy) rake = 90.
         if(sz .lt. sy) rake = -90.
      elseif(cm .eq. 0) then
         if(sz .gt. sx) rake = 90.
         if(sz .lt. sx) rake = -90.
      elseif(cl .eq. 1 .or. cm .eq. 1 .or. cn .eq. 1) then
         rake= 9999.
      else
       rake = (1. - cn*cn)*(sz - sx)/(sy - sx)
       rake = cn*(cm*cm - rake)/(cl*cm)
       rake = atan(rake)
       rake = -rake * 180./PI
      endif
   return
   end
c-------------------------------------------------------------c
```

```
c---------------------------------------------------------------c
c                            MHRDIAG
c---------------------------------------------------------------c
c   This subroutine can be used to write a file to plot
c a 3-D Mohr's Diagram. It has the option to include the rotation
c path and the Coulomb fracture and sliding lines.
c---------------------------------------------------------------c
c                            VARIABLES
c---------------------------------------------------------------c
c INPUT:
c    linefrac      - Flag. Takes values 0 or 1. When equal 1, the
c                    Coulomb sliding line is included in the plot
c    linevir       - Flag. When equal 1, the Coulomb fracture line is
c                    included in the plot
c    miufrac,sfrac - Fractured rock friction and strength
c    miuvir,svir   - Virgin rock friction and strength
c    nfile         - File number for the plot (rotmhrin.d)
c    npoints       - Number of coordinates points in the Mohr circles
c    sigma1,2,3    - Principal stresses magnitudes
c    sigpath(i),   - Rotation path's coordinates in Mohr's
c                    Configuration
c    taupath(i)    (i=1,ipath; ipath=2*irot, maximum value=200)
c---------------------------------------------------------------c
c                      REQUIRED SUBROUTINES
c---------------------------------------------------------------c
c    mohrcrc.f     - Computes the X,Y coordinates of the 3-D Mohr's
c                    Circles .
c---------------------------------------------------------------c
      subroutine mhrdiag(npoints,sigma1,sigma2,sigma3,linevir,miuvir,
     +                   svir,linefrac,miufrac,sfrac,ipath,sigpath,
     +                   taupath,nfile)
c---------------------------------------------------------------c
c                DECLARE AND DIMENSION VARIABLES
c---------------------------------------------------------------c
      dimension sigpath(ipath),taupath(ipath)
      real miuvir,miufrac
c---------------------------------------------------------------c
c WRITE MOHR CIRCLES'S COORDINATE POINTS
c---------------------------------------------------------------c
      call mohrcrc(npoints,sigma1,sigma2,sigma3,r21,r31,r32,nfile)
c---------------------------------------------------------------c
c              CAN ALSO INCLUDE POLE'S ROTATION PATH
c---------------------------------------------------------------c
      return
      end
```

c--c

```
c----------------------------------------------------------------c
c                            SUBROUTINE   MOHRCRC
c----------------------------------------------------------------c
c    This subroutine computes the X,Y coordinates of the 3-D Mohr
c Circles (C32, C21, C31).
c----------------------------------------------------------------c
c----------------------------------------------------------------c
c                                VARIABLES
c----------------------------------------------------------------c
c    DPHI        - Angle sampling interval for the plots.
c    nfile       - Number of the file 'mohrin'.
c    np          - Number of points for the circles's plot
c                  (maximum 500). Dimension of X21,Y21,X31,Y31,X32,Y32.
c    O21,O31,O32- Mohr circle centers.
c    PHI         - Variable angle for the plots.
c    R21,R31,R32- Mohr circle radius.
c    sigma1,2,3 - Principal stresses magnitudes.
c    X21,Y21     - Mohr circle X,Y coordinates.
c----------------------------------------------------------------c
c
c INPUT : np,sigma1,2,3
c----------------------------------------------------------------c
c
c OUTPUT:  X,Y coordinates
c----------------------------------------------------------------c
subroutine mohrcrc(np,sigma1,sigma2,sigma3,R21,R31,R32)
c----------------------------------------------------------------c
c                  DIMENSION AND INITIALIZE VARIABLES
c----------------------------------------------------------------c
dimension X32(500),Y32(500),X21(500),Y21(500),X31(500),Y31(500)
DPHI = 3.14159/float(np)
PHI = 3.14159 + DPHI
c----------------------------------------------------------------c
c COMPUTE ORIGINS (O) AND RADIUS (R) OF EACH CIRCLE
c----------------------------------------------------------------c
R32 = (sigma2 - sigma3)/2.
O32 = (sigma2 + sigma3)/2.
O21 = (sigma1 + sigma2)/2.
R21 = (sigma1 - sigma2)/2.
O31 = (sigma1 + sigma3)/2.
R31 = (sigma1 - sigma3)/2.
c----------------------------------------------------------------c
c COMPUTE X,Y COORDINATES FOR EACH CIRCLE
c----------------------------------------------------------------c
do 10 I=1,np
```

```
PHI = PHI - DPHI
CPHI = COS(PHI)
SPHI = SIN(PHI)
X32(I) = O32 + R32 * CPHI
Y32(I) = R32 * SPHI
X21(I) = O21 + R21 * CPHI
Y21(I) = R21 * SPHI
X31(I) = O31 + R31 * CPHI
Y31(I) = R31 * SPHIc
10      continue
return
end
c--------------------------------------------------------------------c
```

```
c----------------------------------------------------------------c
c                            WLFGEO
c----------------------------------------------------------------c
c  This subroutine writes the file to plot the rotation path in a
c  Wulff Stereonet with geographical coordinates on its axes. It also
c  plots the planes corresponding to the initial and final position of
c  the pole along its path c and the direction of the shear stress on
c  these planes (direction of rotation). For each fault orientation it
c  is possible to construct a focal mechanism by assigning the
c  correct value to 'inum' and 'inumf'.  Optionally, (npcoul > 1) it
c  can include, in the same projection, the Coulomb  sliding line
c----------------------------------------------------------------c
c----------------------------------------------------------------c
c                            VARIABLES
c----------------------------------------------------------------c
c ROTATION:
c
c    alpha(i)      - Angles between pole's radius vector and Principal
c                    Axes of Stresses (i=1,2,3)
c    anggh(i)      - Angles between a given direction and N E Z axes
c                    in geographic coordinates. (i =1,2,3).
c    angshr(i)     - Angles between shear stress on the fracture and
c                    Principal Axes of Stresses (i=1,2,3)
c    csangsh(i,j)  - Direction cosines of the shear stress on the
c                    fracture's plane in relation to principal
c                    stresses axes.
c                    (cosines of angshr) (i=1,2,3; j=1,irot)
c    ddir          - Pole's dip direction in Wulff's Projection
c    dip           - Pole's dip in Wulff's Projection
c    ddrpath(i),   - Rotation path's coordinates in Stereographic
c                    Projections
c    dippath(i)      (i=1,irot)
c    drcoul(i),    - Coulomb sliding line coordinates (ddir,dip) in
c                    Wulff's space
c    dpcoul(i)       (i=1,npcoul)
c    iflag         - Flag. In subr. 'wulfmap' iflag=1 implies mapping
c                    from Stress Geographic coordinates.
c     ipole        - In subrs 'wlfcord' and 'wulfmap' ipole =0 implies
c                    that we are mapping lines (direction vectors)
c                    instead of fault poles
c    irot          - Number of rotation episodes. Dimension of
c                    ddrpath, dippath, radius, drcoul,dpcoul.
c                    Maximum value = 100 !!!
c                    (if taking 'dteta' very small, you may
c                    have to increase this numberand  'ipath=2*irot'
```

221

```
c                        in the  'dimension'  statements)
c   istn,iste,istz-  Principal Stress (1,2 or 3) oriented in the
c                        north, east and  depth directions in a Stress
c                        Wulff stereographic projection.Can take only
c                        different values among 1, 2 or 3.
c   np               - Number of poles to be ploted in subr. 'wlfpole'
c   npcoul           - Number of points in the Coulomb sliding line plot.
c                        Dimension of drcoul, dpcoul. Also acts as a flag :
c                        when npcoul = 0, the plot is omitted from the
c                        projection.
c   plunge           - Plunge coordinate of a given direction in a
c                        Stereographic Projections.
c   rake(i)          - the slip direction of the fault
c   sigmtrn(i)       - Dip directions of the 3 Principal Stresses. i =
c                        1,2,3.
c   sigmpln(i)       - Dip of the 3 Principal Stresses. i = 1,2,3.
c
c   sigmpln(i)       - Dip of the 3 Principal Stresses. i = 1,2,3.
c   sigmadir(i,j)    - Angles between principal stresses and
c                        geographical coordinates (i,j=1,2,3)
c   trend            - Trend coordinate of a given direction in a
c                        Stereographic Projections.
c   trnshr(i)        - Trend of the shear stress direction on the fault
c                        (i=1,irot).
c-----------------------------------------------------------------------c
c INPUT : csangsh, ddrpath, dippath, dpcoul, drcoul, irot,
c                  istr., iste,istz, movie, nfile,npcoul, sigmadir.
c-----------------------------------------------------------------------c
c OUTPUT : trnshr, plnshr,rake,dip,strike
c-----------------------------------------------------------------------c
c AUXILIARY : answer, ddir, dip, trnshr, plnshr, irev, irev1, irev2,
c               plunge, sigmtrn, sigmpln, title, trend.
c-----------------------------------------------------------------------c
c                         REQUIRED SUBROUTINES
c-----------------------------------------------------------------------c
c   wlfcord.f    - Computes Wulff's coordinates "trend and plunge",
c                    given a vector's direction cosines or Wulf's
c                    coordinates "ddir,dip", given a pole's direction
c                    cosines.
c   wulfmap.f    - Maps poles between a Wulff projection with
c                    Principal Stresses along its axes and another Wulff
c                    projection  with standard  N-E-Z geographic axes.
c-----------------------------------------------------------------------c
   subroutine wlfgeo(nfile,nfileb,sigmadir,irot,ddrpath,dippath,
      +                 csangsh,movie,npcoul,drcoul,dpcoul,istn,
```

```
     +                         iste, istz,trnshr,plnshr,strainv,strainh,
     +                         rake,sn,tau,borig,bradiu,sorig,sradiu)
c-------------------------------------------------------------------c
c                         DIMENSION VARIABLES
c-------------------------------------------------------------------c
character*20 title
logical answer
dimension sigmadir(3,3),sigmtrn(3),sigmpln(3),alpha(3),anggh(3)
dimension ddrpath(irot),dippath(irot),angshr(3),csangsh(3,irot)
dimension trnshr(100),plnshr(100)
dimension drcoul(npcoul),dpcoul(npcoul)
 dimension rake(100),strainv(irot),strainh(irot),rake(irot)
 dimension sn(100),tau(100),borig(100),bradiu(100)
 dimension sorig(100),sradiu(100)
c-------------------------------------------------------------------c
c          MAP ROTATION PATH AND SHEAR STRESS DIRECTION
c              FROM STRESS TO GEOGRAPHIC COORDINATES
c-------------------------------------------------------------------c
    pi= 3.1415927/180.
    do 10 i = 1,irot
c-------------------------------------------------------------------c
c COMPUTE ROTATION PATH COORDINATES (ddir & dip) IN GEOGRAPHIC WULFF.
c  NOTICE THAT WE ARE USING THE SAME ARRAYS (ddrpath & dippath) TO
c  PUT BOTH THE STRESS AND GEOGRAPHIC COORD.
c-------------------------------------------------------------------c
    iflag = 1
    ipole = 1
    call wulfmap(iflag,ipole,sigmadir,ddrpath(i),dippath(i),
     +                    ddir,dip,alpha,anggh,istn,iste,istz,irev)
    ddrpath(i) = ddir
    dippath(i) = dip
c-------------------------------------------------------------------c
c COMPUTE ANGLES CORRESPONDING TO DIRECTION COSINES OF SHEAR STRESS
c DIRECTION
c-------------------------------------------------------------------c
 do 15 j=1,3
angshr(j) = acos(csangsh(j,i))
15       continue
c-------------------------------------------------------------------c
c COMPUTE SHEAR STRESS DIRECTION'S STRESS-WULFF COORDINATES
c trend & plunge.
c-------------------------------------------------------------------c
    ipole = 0
    call wlfcord(ipole,angshr,csangsh(1,i),trend,plunge,
     +     ddir,dip,istn,iste,istz,irev1)
```

223

```
c----------------------------------------------------------------c
c MAP SHEAR STRESS VECTOR FROM STRESS TO GEOGRAPHIC WULFF COORDINATES
c----------------------------------------------------------------c
      iflag = 1
      ipole = 0
     call wulfmap(iflag,ipole,sigmadir,trend,plunge,trnshr(i),
     +            plnshr(i),angshr,anggh,istn,iste,istz,irev2)
c----------------------------------------------------------------c
c MAP PRINCIPAL STRESSES COORDINATES FROM STRESS TO GEOGRAPHIC-WULFF
c                 AND PUT THEM IN THE SAME ARRAYS
c----------------------------------------------------------------c
     do 30 i=1,3
       call wulfmap(1,1,sigmadir,sigmtrn(i),sigmpln(i),ddir,dip,
     +                    alpha,anggh,istn,iste,istz,irev)
       sigmtrn(i) = ddir
       sigmpln(i) = dip
30     continue
c----------------------------------------------------------------c
c       SPECIFY HERE FOR WHICH EVENT (inum) DO YOU WANT FAULT PLANE
c                              SOLUTION
c----------------------------------------------------------------c
       inum=1
       trend1 = trnshr(inum) -180.
       if (trend1 .lt. 0.) trend1 = 360. - abs(trend1)
       dip1 = 90. - plnshr(inum)
c----------------------------------------------------------------c
c     TO CONSTRUCT  FAULT PLANE SOLUTIONS
c----------------------------------------------------------------c
       trend2 = trnshr(inumm) - 180.
       if (trend2 .lt. 0.) trend2 = 360. - abs(trend2)
       dip2 = 90. - plnshr(inumm)
c----------------------------------------------------------------c
c IF DESIRED, INCLUDED ALSO COULOMB SLIDING LINE
c----------------------------------------------------------------c
if(npcoul .ge. 1) then
c----------------------------------------------------------------c
c         MAP THE LINE FROM STRESS-WULFF TO GEOGRAPHIC-WULFF
c----------------------------------------------------------------c
     do 20 i=1,npcoul
         iflag = 1
         ipole = 1
         call wulfmap(iflag,ipole,sigmadir,drcoul(i),dpcoul(i),
     +                    dr,dp,alpha,anggh,istn,iste,istz,irev)
       drcoul(i) = dr
       dpcoul(i) = dp
```

```
20      continue
 endif
 return
end
```
c--c

```
c-----------------------------------------------------------------c
c                              WULFMAP
c-----------------------------------------------------------------c
c    This subroutine maps poles (ipole=1) and lines (ipole=0) between
c    a Wulff Projection with Principal Stresses along its axes and
c    another Wulff Projection with geographic N-E-Z axes.
c-----------------------------------------------------------------c
c-----------------------------------------------------------------c
c                              VARIABLES
c-----------------------------------------------------------------c
c  anggh(i)      - Angles between pole and Wulff axes in geographic
c                  space.i=1,2,3 correspond.to the north, east and
c                  depth direction respectively.
c  angstr(i)     - Angles between pole and Stress Wulff Projection
c                  axes.
c  cosang(i)     - Auxiliary variable. Contains pole's direction
c                  cosines in relation to Stress and Space Wulff axes,
c                  depending on  iflag.
c  ddirgh,dipgh - Pole's Dip Direction and Dip in the Usual Space
c                  Projection.
c  ddirst,dipst - Pole's Dip Direction and Dip in the Stress
c                  Projection.
c  iflag         - +1 maps from stresses to geographic coordinates,
c                  -1 the otherway round.
c  irev          - Flag. irev=1 implies that the sense of the pole's
c                  vector was reversed during computations.
c istn,iste,istz- Principal Stress (1,2 or 3) oriented in the north,
c                  east and depth directions in a Stress Wulff
c                  stereographic  projection.
c                  Can take only different values among 1, 2 or 3.
c  sigmadir(i,j)- Matrix with principal stresses's direction cosines.
c                  i,j = 1,2,3.
c-----------------------------------------------------------------c
c INPUT : iflag, ipole, sigmadir,istn,iste,istz AND ddirst,dipst
c                  (iflag=1) OR ddirgh,dipgh (iflag=-1).
c-----------------------------------------------------------------c
c OUTPUT: irev,angstr,anggh AND ddirgh,dipgh (iflag=1) OR
c                  ddirst,dipst (iflag=-1).
c-----------------------------------------------------------------c
c REQUIRED SUBROUTINES: axesrot(iflag,x,sigmadir)
c-----------------------------------------------------------------c
subroutine wulfmap(iflag,ipole,sigmadir,ddirst,dipst,ddirgh,
     +                    dipgh,angstr,anggh,istn,iste,istz,irev)
c-----------------------------------------------------------------c
c             DIMENSION AND INITIALIZE VARIABLES
```

```
c-----------------------------------------------------------------c
dimension cosang(3),angstr(3),anggh(3),sigmadir(3,3)
irev = 0
c-----------------------------------------------------------------c
c    DEFINE CONSTANTS TO CONVERT DEGREES TO RADIANS AND BACK
c-----------------------------------------------------------------c
 rad = 3.14159/180.
deg = 180./3.14159
if(iflag .eq. 1) then
c-----------------------------------------------------------------c
c                       FROM STRESS TO GEOGRAPHIC COORDINATES
c-----------------------------------------------------------------c
c COMPUTE DIRECTION COSINES AND CORRESPONDING ANGLES (eqns. 72-75)
c-----------------------------------------------------------------c
   if(ipole .eq. 0) then
c-----------------------------------------------------------------c
c IF YOU ARE MAPPING LINES...
c-----------------------------------------------------------------c
      theta = ddirst * rad
      angstr(istz) = (90. - dipst) * rad
   else
c-----------------------------------------------------------------c
c          IF YOU ARE MAPPING POLES (eqns. 72,73)...
c-----------------------------------------------------------------c
      theta = (ddirst - 180.) * rad
      angstr(istz) = dipst * rad
   endif
   sinz = sin(angstr(istz))
c-----------------------------------------------------------------c
c     COMPUTE ANGLES AND DIRECTION COSINES (eqns. 74,75)
c-----------------------------------------------------------------c
   cosang(istz) = cos(angstr(istz))
   cosang(istn) = sinz * cos(theta)
   cosang(iste) = sinz * sin(theta)
   angstr(istn) = acos(cosang(istn))
   angstr(iste) = acos(cosang(iste))
c-----------------------------------------------------------------c
c     ROTATE AXES ACCORDING TO SIGMADIR (eqn. 79). COMPUTE DIRECTION
c                  COSINES IN GEOGRAPHIC COORDINATES
c-----------------------------------------------------------------c
   call axesrot(-1,cosang,sigmadir)
c-----------------------------------------------------------------c
c IF THE NEW POLE'S RADIUS VECTOR POINTS UP, TAKE THE OPPOSITE TO THE
c SAME PLANE (SINCE WE ARE WORKING WITH LOWER HEMISPHERE WULFF
c PROJECTION)
```

```
c-------------------------------------------------------------c
   if(cosang(3) .lt. -0.001) then
       irev = 1
       do 5 i=1,3
         cosang(i) = -cosang(i)
5      continue
   endif
c-------------------------------------------------------------c
c COMPUTE ANGLES IN GEOGRAPHIC COORDINATES
c-------------------------------------------------------------c
   do 10 i = 1,3
                anggh(i) = acos(cosang(i))
10     continue
c-------------------------------------------------------------c
c COMPUTE DIP FOR LINES AND POLES (eqn. 81)
c-------------------------------------------------------------c
   if(ipole .eq. 0) then
       dipgh = 90. - anggh(3)*deg
   else
       dipgh = anggh(3) * deg
   endif
c-------------------------------------------------------------c
c CHECK FOR POSSIBLE SINGULARITIES (eqn. 69 or 80)
c-------------------------------------------------------------c
   sin3 = sin(anggh(3))
    if(abs(sin3).lt.0.001 .and. abs(cosang(1)).lt.0.001) then
c-------------------------------------------------------------c
c IF sin3=0 THEN ddirgh IS UNDEFINED. WE ARBITRARILY SET IT TO 0
c-------------------------------------------------------------c
            ddirgh = 0.
          else
            if (abs(cosang(1)) .gt. abs(sin3) )   then
            theta = 0.0
             goto 11
            endif
        theta = acos(cosang(1)/sin3)*deg
11     continue
c-------------------------------------------------------------c
c NOTICE THAT acos HAS 2 POSSIBLE SOLUTIONS (theta,360-theta, eqn.69)
c THE SIGN OF cosang(2) DECIDES WHICH TO TAKE
c-------------------------------------------------------------c
            if(cosang(2) .lt. -0.01) theta = 360. - theta
c-------------------------------------------------------------c
c COMPUTE DIP DIRECTION (eqn. 80) FOR THE POLE OR THE LINE.
c IF ddir > 360, SUBSTRACT 360
```

```
c--------------------------------------------------------------c
      if(ipole .eq. 0) then
        ddirgh = theta
      else
        ddirgh = theta + 180.
      endif
      if(ddirgh .ge. 359.9) ddirgh = ddirgh - 359.9
   endif
else
c--------------------------------------------------------------c
c                    FROM GEOGRAPHIC TO STRESS COORDINATES
c--------------------------------------------------------------c
c COMPUTE DIRECTION COSINES AND CORRESPONDING ANGLES (eqns. 72-75)
c--------------------------------------------------------------c
   if(ipole .eq. 0) then
c--------------------------------------------------------------c
c IF YOU ARE MAPPING LINES...
      theta = ddirgh * rad
      anggh(3) = (90. - dipgh) * rad
   else
c--------------------------------------------------------------c
c IF YOU ARE MAPPING POLES (eqns. 72,73)...
c--------------------------------------------------------------c
      theta = (ddirgh - 180.) * rad
      anggh(3) = dipgh * rad
   endif
c--------------------------------------------------------------c
c COMPUTE ANGLES AND DIRECTION COSINES (eqns. 74,75)
c--------------------------------------------------------------c
   sin3 = sin(anggh(3))
   cosang(3) = cos(anggh(3))
   cosang(1) = sin3 * cos(theta)
   cosang(2) = sin3 * sin(theta)
   anggh(1) = acos(cosang(1))
   anggh(2) = acos(cosang(2))
c--------------------------------------------------------------c
c ROTATE AXES ACCORDING TO SIGMADIR (eqn. 79). COMPUTE DIRECTION
c COSINES IN STRESS COORDINATES
c--------------------------------------------------------------c
   call axesrot(1,cosang,sigmadir)
c--------------------------------------------------------------c
c IF THE NEW POLE'S RADIUS VECTOR POINTS UP, TAKE THE OPPOSITE TO THE
c SAME PLANE (SINCE WE ARE WORKING WITH LOWER HEMISPHERE WULFF
c PROJECTION)
c--------------------------------------------------------------c
```

```fortran
      if(cosang(istz) .lt. -0.001) then
         irev = 1
         do 15 i=1,3
           cosang(i) = -cosang(i)
15        continue
      endif
c-----------------------------------------------------------------c
c COMPUTE ANGLES IN STRESS COORDINATES
c-----------------------------------------------------------------c
      do 20 i = 1,3
           angstr(i) = acos(cosang(i))
20     continue
c-----------------------------------------------------------------c
c COMPUTE DIP FOR LINES AND POLES (eqn. 81)
c-----------------------------------------------------------------c
      if(ipole .eq. 0) then
         dipst = 90. - angstr(istz)*deg
      else
          dipst = angstr(istz) * deg
      endif
c-----------------------------------------------------------------c
c CHECK FOR POSSIBLE SINGULARITIES (eqn. 69 or 80)
c-----------------------------------------------------------------c
      sinz = sin(angstr(istz))
       if(abs(sinz).lt.0.001 .and. abs(cosang(istn)).lt.0.001) then
c-----------------------------------------------------------------c
c IF sinz=0 THEN ddirst IS UNDEFINED. WE ARBITRARILY SET IT TO 0
c-----------------------------------------------------------------c
          ddirst = 0.
      else
        theta = acos(cosang(istn)/sinz)*deg
c-----------------------------------------------------------------c
c NOTICE THAT acos HAS 2 POSSIBLE SOLUTIONS (theta & 360-theta,eqn.
c      69): THE c SIGN OF cosang(iste) DECIDES WHICH TO TAKE
c-----------------------------------------------------------------c
          if(cosang(iste) .lt. -0.01) theta = 360. - theta
c-----------------------------------------------------------------c
c COMPUTE DIP DIRECTION (eqn. 80) FOR THE POLE OR THE LINE.
c IF ddir > 360, SUBSTRACT 360
c-----------------------------------------------------------------c
          if(ipole .eq. 0) then
           ddirst = theta
          else
            ddirst = theta + 180.
          endif
```

```
         if(ddirst .ge. 359.9) ddirst = ddirst - 359.9
      endif
endif
return
end
c-----------------------------------------------------------------c
```

```
c-----------------------------------------------------------c
                    SUBROUTINE  AXESROT
c-----------------------------------------------------------c
c   This subroutine computes coordinates's transformations during
c   pure rotations
c-----------------------------------------------------------c
c-----------------------------------------------------------c
c                       VARIABLES
c-----------------------------------------------------------c
c  alpha(i,j) - transformation matrix with the direction cosines of
c               the new axis
c               i=1,2,3 correspond to new axis X',Y',Z' respectively,
c               j=1,2,3 correspond to its direction cosines.
c
c iflag - = +1 for direct transformations and =-1 for inverse transf.
c
c  x(i) - i=1,2,3 x,y,z coordinates for a single point.
c  '        On input contains the original coordinates,
c           on output - the transformed ones.
c  s(i) - Auxiliary (temporary) variable. Same as x(i).
c-----------------------------------------------------------c
c INPUT : iflag,alpha,x
c-----------------------------------------------------------c
c OUTPUT: x
c-----------------------------------------------------------c
subroutine axesrot(iflag,x,alpha)
dimension x(3),s(3),alpha(3,3)
c-----------------------------------------------------------c
c          DIRECT OR INVERSE TRANSFORMATION (ROTATION) ?
c-----------------------------------------------------------c
if(iflag .ne. 1 .and. iflag .ne. -1) then
   print *, "'iflag' has improper value in subroutine axesrot"
   stop
endif
if(iflag .eq. 1) then
c-----------------------------------------------------------c
c                    DO DIRECT TRANSFORMATION
c-----------------------------------------------------------c
   do 10 i=1,3
    s(i) = 0.
    do 10 j=1,3
        s(i) = s(i) + alpha(i,j) * x(j)
10    continue
else
c-----------------------------------------------------------c
```

232