SEISMIC VELOCITY VARIATION IN THE OROZCO FRACTURE ZONE FROM FIRST ARRIVAL TRAVEL TIMES

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE
UNIVERSITY OF HAWAII IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN GEOLOGY AND GEOPHYSICS

AUGUST 1983

By

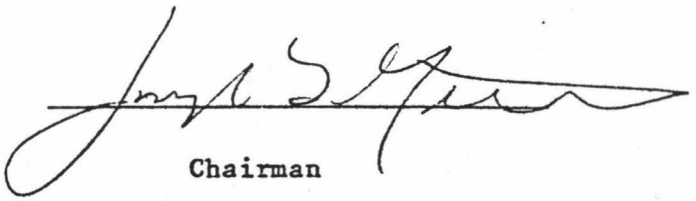
Anthony Edward Schreiner

Thesis Committee:

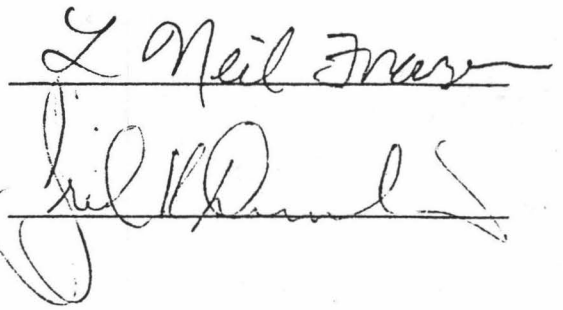
Joseph F. Gettrust, Chairman
Frederick K. Duennebier
L. Neil Frazer

We certify that we have read this thesis and that in our opinion it is satisfactory in scope and quality as a thesis for the degree of Master of Science in Geology and Geophysics.

THESIS COMMITTEE



Chairman



ACKNOWLEDGMENTS

This thesis is dedicated to the memory of my father, Edmund Douglas Schreiner (1921-1983).

In addition to the members of my thesis commmittee, many people have provided assistance and commentary. Special thanks are due to Sharon Latraille for guiding my way through the early stages of data reduction of the the HIG data and retrieval of data from the ROSE data archive. Carlos Mortera, Elizabeth Ambos, and Robert Cessaro have provided good ideas throughout. Dr. Thomas Brocher read an early version of the manuscript. I would also like to thank Dr. LeRoy Dorman for providing his version of the least squares inversion program on which mine is based. Dr. John B. Sinton for permission to use his ray-tracing program, and Kazuo Furukawa who before me, wrote programs to accomplish similar tasks and gave me examples from which to work.

This work was supported by the Office of Naval Research under grant N00014-75-C-0209.

ABSTRACT

Four refraction profiles were carried out in the Orozco Fracture Zone region of the East Pacific Rise during the ROSE experiment of 1979. A large number of ocean bottom receivers were deployed in the area for the purpose of earthquake location so the shots were recorded at a variety of locations. Because a variety of models of seismographs were used and because the crustal structure in the fracture zone is variable, the analysis of the seismic refraction data concentrated on the travel times of the first arrivals. Velocity-depth models were developed in three steps. Least squares inversion of travel time data was used to develop horizontally stratified models for each receiver location. Time terms were examined to determine the lateral variation along each shot profile. Models with two-dimensional structure were constructed from the series of one-dimensional models. The two-dimensional models were verified by a ray-tracing algorithm. The ray-tracing was also used to model post-critical reflections to match the observed amplitude variation in the data and to constrain the total crustal thickness.

The results of the analysis show that the oceanic crust near the active part of the transform fault is anomalous. The crustal structure determined for the exterior of the fracture zone area is comparable to other velocity-depth models developed for crust on the East Pacific Rise. In the interior the crust is not differentiated between upper and lower regions and the average seismic velocities are low. However there is no apparent difference in total crustal thickness between the

interior and the exterior of the fracture zone. The difference in the seismic velocity structure between the center and the exterior of the transform fault region is by a reduction in the magma supply at the intersection of the spreading center and the transform fault.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

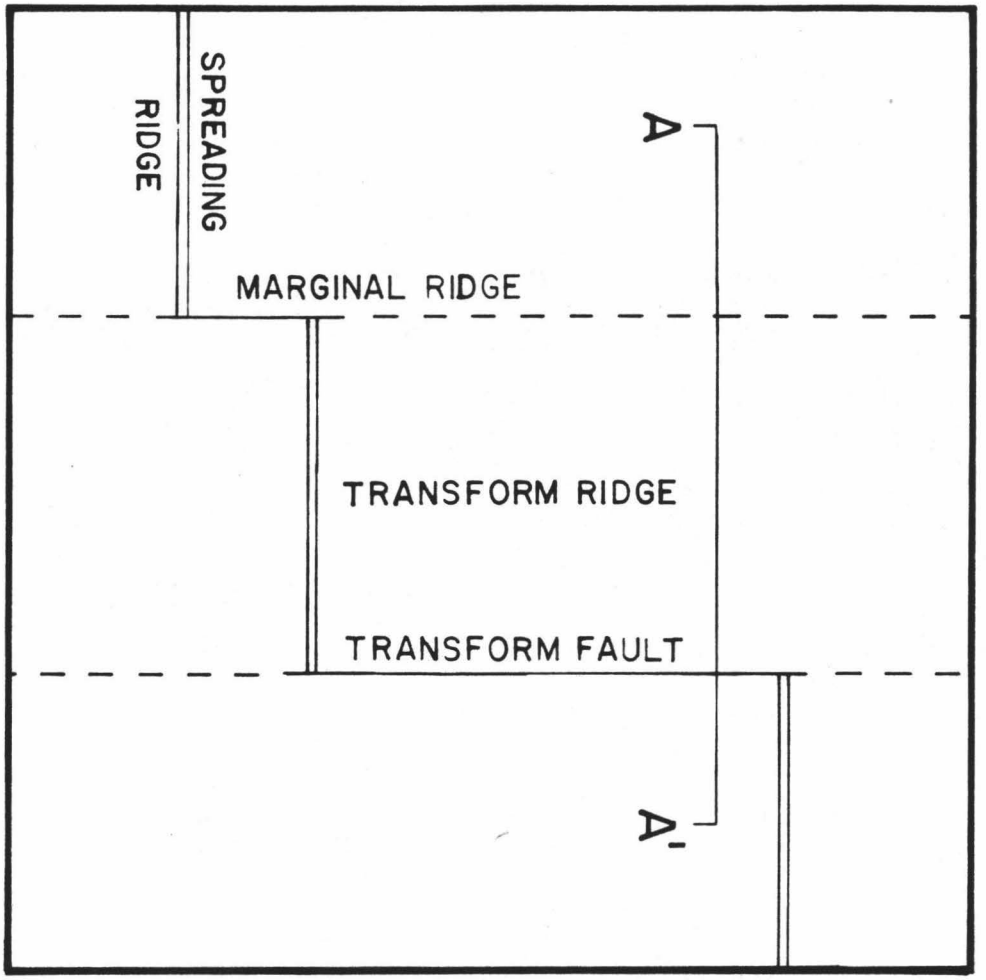
INTRODUCTION

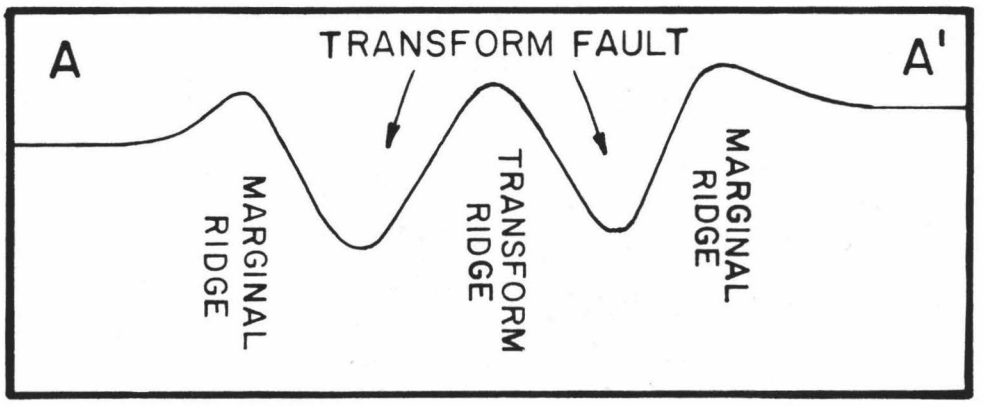
1.1 Historical Background.

Fracture zones have been observed on the ocean floor since the early 1950's [Menard and Dietz, 1952; Menard, 1955]. In the following decade they were recognized as features associated with the great mid-ocean ridge system [Vacquier, 1959; Menard, 1960; Heezen et al., 1964a]. As the theory of plate tectonics became accepted it was recognized that the trend of fracture zones were a record of the plate motion, as they tend to follow small circles about the rotation pole for relative motion between plates [Wilson, 1965; Morgan, 1968; Le Pichon, 1968]. It was established that fracture zones were bathymetric features resulting from strike-slip motion between plates. The active portion of fracture zones, named transform faults are the third type of plate boundary along with spreading ridges and subduction zones [Wilson, 1965; McKenzie and Parker, 1967; Sykes, 1967].

Most of the early exploration of fracture zones was limited to bathymetric studies, starting in the northeastern Pacific [Menard and Fischer, 1958; Menard, 1966; Pitman et al., 1968] and expanding into the Atlantic [Heezen et al., 1964a; Heezen et al., 1964b; Fox et al., 1969; Fleming et al., 1970; Van Andel et al., 1971; Collette et al., 1974] and Indian oceans [Matthews, 1966]. In summary (see also Figure 1) these

Figure 1. Main features in a generalized transform fault system.


SPREADING RIDGE
A
MARGINAL RIDGE
TRANSFORM RIDGE
TRANSFORM FAULT
A'


A
TRANSFORM FAULT
A'
MARGINAL RIDGE
TRANSFORM RIDGE
MARGINAL RIDGE

early bathymetric studies revealed fracture zones to be linear features on the flanks of mid-ocean ridges. They occur every 50 to 100 km along spreading ridges. They may be traced as far as 10000 km away from the ridge in the case of the Mendocino fracture zone. The offset of the opposing ridge segments may be as great as 1200 km, again in the Mendocino fracture zone. Usually there is a single trough which may be filled with sediments. Elongate ridges may flank the trough on either side [Heezen et al., 1964b; Fleming et al., 1970; Thompson and Melson, 1972]. These flanking ridges can rise to heights greater than the nearby spreading ridge [Bonatti, 1978]. Sometimes the fracture zone is marked by a pair of troughs where one or both of them may be the site of ridge offset as in the Vema fracture zone [Heezen et al., 1964b; Van Andel et al., 1971] and the Charlie-Gibbs fracture zone [Fleming et al., 1970; Olivet et al., 1974].

Bottom observation using submersible and deep-towed instruments and sample recovery by dredging greatly increased the available data on transform faults [Thompson and Melson, 1972; Fox et al., 1976; Bonatti and Honnorez, 1976; Schreiber and Fox, 1977; Macdonald et al., 1979]. Samples revealed typical sea-floor tholeitic basalts, greenschist facies metabasalts and metagabbros, and serpentinized ultramafics [Fox et al., 1976]. Ultramafics and gabbroic rocks are commonly recovered from the slopes of the flanking ridges leading some to suggest that these are uplifted blocks of the lower crust [Thompson and Melson, 1972; Bonatti, 1978]. Pictures of the bottom show that the transform valley walls are a result of normal faulting. The displacement is by a series of of

faults with relatively small displacements [Choukroune et al., 1978; Macdonald et al., 1979].

The Tamayo fracture zone at the mouth of the Gulf of California was the site of an extensive survey with with submersibles and deep-towed instruments as part of project RITA in 1978. It is a marginal fracture zone associated with the separation of Baja California from the mainland. It is evident as a pair of transform valleys separated by a ridge. This ridge is not bounded by faults like the transverse ridges of the Atlantic fracture zones, and the uplift is more recent than the sediments overlying it [Macdonald et al., 1979; CYANAMEX and Pastouret, 1981].

Indirect geophysical methods for studying the deeper structure of fracture zones have had more limited use until recently. Cochran [1973] observed that fracture zones in the equatorial Atlantic appear as long, linear, positive gravity anomalies and that only about half of the free-air anomaly can be attributed to uncompensated bathymetry. The magnetic intensity shows peaks which are offset from the peaks of the gravity anomalies. The gravity anomalies were interpreted to indicate a mass excess at shallow depths and the magnetic intensity observations indicate that induced magnetization is more important within the fracture zone than remanent magnetization. Together these imply that fracture zones are sites of intruded ultramafic rocks from the lower crust. Robb and Kane [1975] modeled a mass excess beneath the flanking ridges on the south side of the Vema fracture zone which they postulated was composed of intruded ultramafics. These studies did not consider

the thickness of the crust. Sibuet and Veyrat-Peinet [1980] have shown that a major part of the free-air anomaly over equatorial Atlantic fracture zones is caused by the edge effect of adjacent plate segments with different ages and therefore different thickness and elevation. They do not rule out the presence of bodies of ultramafic material but indicate that they must have density typical of normal upper crustal basalts. Field measurements made near to the source are less sensitive to the deeper structure. In project RITA the gravity and magnetic field was measured by a deeply towed platform. The transform ridge in the Tamayo fracture zone has a strong magnetic intensity anomaly but the gravity anomaly is less than would be expected if the ridge had the same density as the surrounding rock. Kastens et al. [1979] interpreted this result to mean that the ridge is composed of lower density material such as serpentinite.

Potential methods do not provide good constraints on the thickness of the crust. Seismic refraction is the best tool for this. In thirty years of marine refraction work comparatively little has been done in fracture zones. Observations with floating buoys are complicated by the rough topography in these areas. The advent of the ocean bottom seismometer (OBS) facilitated the exploration of axial zones of spreading centers and fracture zones. Fox et al. [1976] deployed OBS's in the Oceanographer fracture zone but did not observe any arrivals so were not able to constrain the thickness of the crust. Ocean bottom recorders and buoys have also been used extensively for

earthquake studies at plate margins but the emphasis has usually been on tectonic processes rather than on crustal structure.

In 1977 a substantial part of the marine geophysical community felt that a more organized and detailed survey of a spreading ridge and fracture zone was necessary and the ROSE project was planned. Seismic exploration of fracture zones has increased greatly since that time as has the understanding of these features. Before describing the ROSE experiment further let us review some of the more recent studies in fracture zones. A survey of the Kane fracture zone revealed that the crust within the fracture zone is anomalously thin, with mantle velocities as shallow as 2 km below the sea-floor [Detrick and Purdy, 1980]. Furthermore this result is caused by an absence of oceanic layer 3 rather than an overall thinning of the crust. Sinha and Louden [1982] made a similar observation in the Oceanographer fracture zone but found that crustal thicknesses of 2 km are not found throughout the fracture zone. In the Vema fracture zone no significant thinning was found [Detrick et al., 1982]. However the crust is anomalous in that the velocity at the surface is very low and the velocity gradient is high and almost constant to a depth of 5 km beneath the sea-floor.

1.2 The ROSE project.

The Rivera Ocean Seismic Experiment (ROSE) was originally conceived as a large scale seismic experiment that would combine natural and explosive sources to study the details of the tectonic behavior in a

fracture zone and the propagation of seismic energy across a spreading center and also accross an ocean-continent boundary [Ewing and Meyer, 1982]. The Rivera fracture zone at 19°N on the East Pacific Rise is well suited to such a project because it is close to a continent, its seisimicity has been studied and the bathymetry in the area is well known. The placement of the ocean bottom receivers and explosive sources could have been planned to optimize the information gathered.

When the government of Mexico did not grant permission for the participating ships to work in its territorial waters, the experiment was moved to the alternate site. The wave propagation experiments and the tectonics studies at sea were physically separated, the first to the East Pacific Rise crest at 12°N, and the second to the Orozco fracture zone at 15°N and the land phase was moved to the Petatlan area.

The institutions taking part in the marine phases of the ROSE project were: the Hawaii Institute of Geophysics, Lamont-Doherty Geological Observatory, Massachusetts Institute of Technology, Naval Ocean Research and Development Activity, Naval Research Laboratory, Naval Undersea Systems Center, Oregon State University, Scripps Institution of Oceanography, University of California at Santa Barbara, University of Texas Marine Sciences Institute, University of Washington, and Instituto Oceanografico at Manzanillo. Ships used were: R/V Robert D. Conrad, Lamont-Doherty; USNS DeSteigeur, NORDA/NAVOCEANO; USNS Hayes, Office of Naval Research; R/V Kana Keoki, Hawaii Institute of Geophysics; R/V Thomas G. Thompson, University of Washington.

The experiment was conducted in 1979 and a special issue of the Journal of Geophysical Research (November, 1982) has appeared with some of the results. Most of the papers covered the first or active phase.

The main objectives of the second or passive phase of ROSE were to determine accurately the location of earthquakes within the transform fault so as to delineate the boundary between the Cocos and Pacific plates, and to study the propagation of earthquake-generated shear waves accross the plate boundary. Accurate hypocenter location requires a known velocity structure and to this end four refraction profiles were shot in the transform fault region.

Results from analysis of the seismicity have been published. A preliminary study [Project ROSE scientists, 1981] used a velocity model derived for 2.9 M.y. old crust on the flank of the EPR near the Siqueiros fracture zone [Orcutt et al., 1975] and 0.4 M.y. old crust of the the Cocos plate south of the Orozco fracture zone [Lewis and Snydsman, 1979]. A more comprehensive study [Trehu, 1982; Trehu and Solomon, 1983] used the same velocity model for the exterior of the transform fault but a different model for the central region. This model was derived from some of the ROSE Phase 2 profiles [Trehu and Purdy, 1982]. Another study [Ouchi et al., 1982] used a separate velocity model derived from refraction data in the central ridge area.

The objective of the present study is to use more of the available data from ROSE Phase 2 to provide a more detailed model for

the Orozco Fracture Zone. The important features will be compared with other fracture zones.

CHAPTER II

PHYSICAL SETTING

The Orozco fracture zone at 15°N, 105°W is one of a series of young fracture zones at the northern end of the East Pacific Rise. It is a complex region that marks a 90 km sinistral offset of the rise crest. The width of the transform fault is comparable to the offset. The transform region consists of two separate faults connected by a relay zone which is roughly perpendicular to the EPR crest. To the east, the fracture zone widens and extends into the Middle America Trench 200 km away. To the west it is visible as a pair of elongate troughs to a distance of 150 km and out to a distance of 400 km by the disruption of the regions magnetic anomaly pattern.

Figure 2 shows the bathymetry an magnetic anomaly pattern for the eastern equatorial Pacific [Klitgord and Mammerickx, 1982]. It is evident from the fan pattern of the anomalies that the current pole of rotation between the Cocos and Pacific plates is very close and that spreading rate increases rapidly to the south. Mammericx and Klitgord [1982] have interpreted this bathymetry and magnetic data as resulting from a series of reorganizations of spreading centers during the last 25 M.y. which is summarized in Figure 3. Of particular noté are the events between 3.5 and 6.5 M.y.b.p. where the spreading center between the Rivera and Orozco fracture zones moved from what is now the Mathematician Ridge to a northern extension of the EPR. The large

Figure 2. Physical features and magnetic anomaly pattern in the eastern equatorial Pacific (from Klitgord and Mammerickx, 1982).

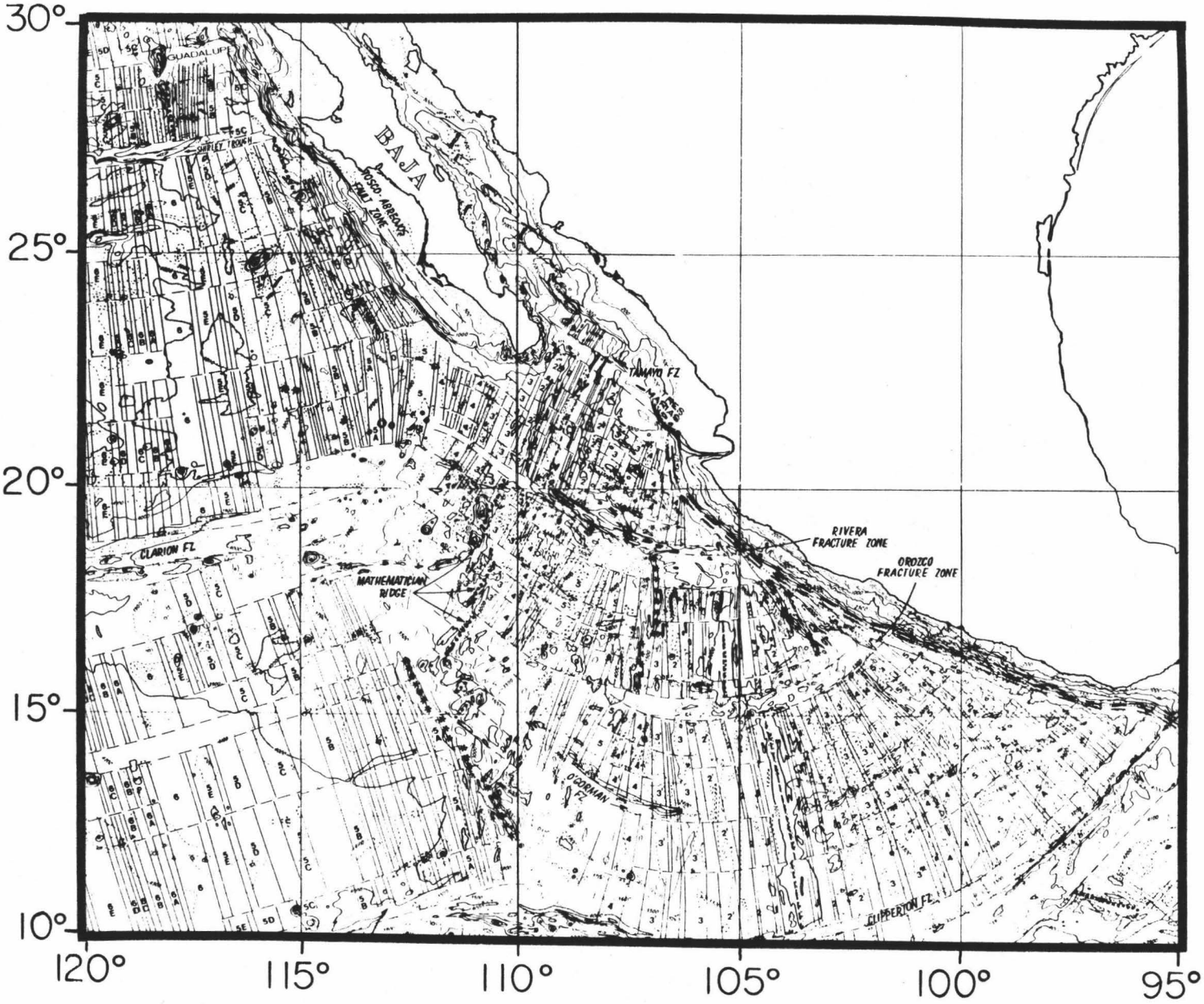
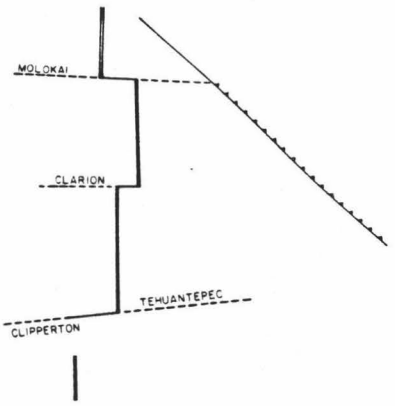
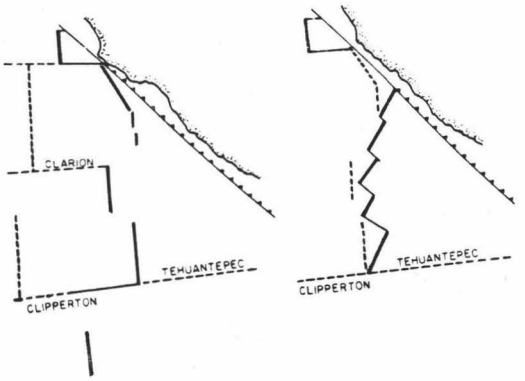

30°
25°
20°
15°
10°
120°
115°
110°
105°
100°
95°
GUADALUPE
BAJA
MOSCO-ARGENTA FAULT ZONE
TAMAYO FZ
CLARION FZ
MATHEMATICIAN RIDGE
RIVERA FRACTURE ZONE
ORozco FRACTURE ZONE
O'GORMAN FZ
CLIPPERTON FZ

Figure 3. Major tectonic re-organizations in the eastern equatorial Pacific during the last 25 M.y. (Mammerickx and Klitgord, 1982). As the northern East Pacific Rise was overridden by the North American plate the spreading centers and poles of spreading were shifted. In the last 6.5 M.y. the present East Pacific Rise has propagated northward in a series of discrete jumps. The last major event was the extension of the rise crest north of the Orozco Fracture Zone 3.5 M.y.B.P.. The western part of the transform fault became inactive as the offset was taken up by the Rivera Fracture Zone further north. It is expected that much of the complicated features in the Orozco fracture zone are remnants of this reorganization.

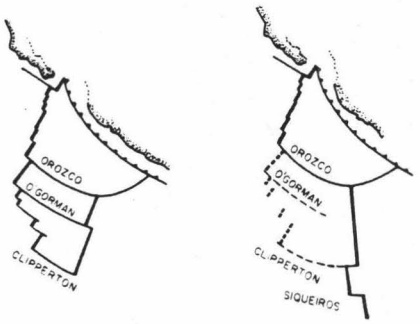
BEFORE 25 M.Y.B.P.



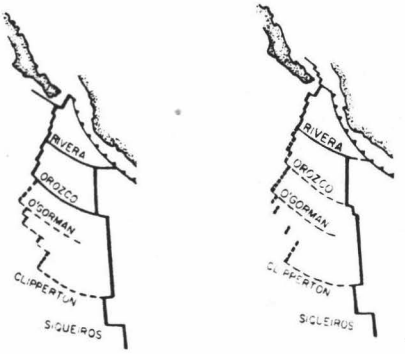
REORGANIZATION AT
12.5 - 11 M.Y.B.P.



REORGANIZATION AT
6.5 M.Y.B.P.



REORGANIZATION AT
3.5 M.Y.B.P.



offset between the two spreading axes moved from the Orozco Fracture Zone to the Rivera Fracture Zone.

The bathymetry in the transform fault region of the Orozco Fracture Zone is shown in Figure 4. The four refraction profiles for ROSE Phase 2 are labeled 1N through 4N. Ocean bottom seismometer locations are shown by triangles. The receivers used in the present study are shown by filled triangles and are numbered.

The area is dominated by ridges and troughs. For the purpose of this study it has been useful to differentiate six physiographic domains within the transform fault region (Figure 5). They are: the East Pacific Rise crest; the north and south transform valleys; the marginal ridges, which parallel the transform fault; the central ridge, which separates the transform valleys; the central transverse trough, which is perpendicular to the transform valleys; and the "normal" oceanic crust of the rise flank. These domains are not necessarily contiguous as in the case of the rise crest or the central ridge. Most of the earthquake activity is confined to the northern transform fault and the central trough. The boundary between the Cocos and the Pacific plates connects between the rise crest via the eastern part of the southern transform valley, the central trough, and the western part of the northern transform valley (Figure 5). First motion studies indicate strike-slip displacement in the northern transform fault [Trehu and Solomon, 1983]. Seismicity in the central trough is more diffuse and varied. This trough is perpendicular to the direction of spreading, N85^{o}E [Minster and Jordan, 1978], so must be a site of extension.

Figure 4. Bathymetry in the Orozco Fracture Zone. The contour interval is 200 m. Depths greater than 3000 m are shaded. Locations of seismic recording instruments deployed on the ocean bottom for ROSE Phase 2 are shown by triangles. Those receivers from which data are used in this paper are indicated by filled triangles and are numbered.

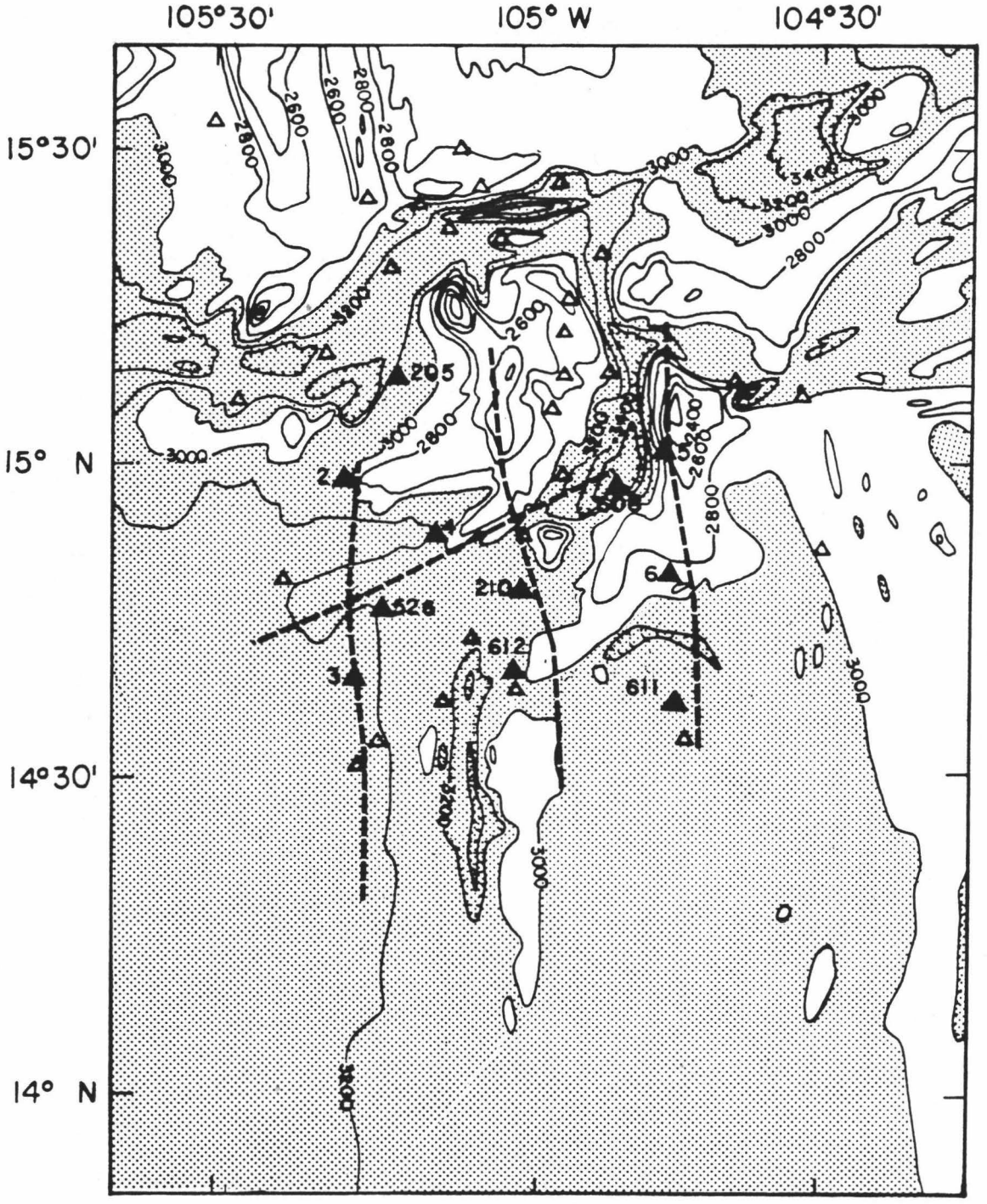
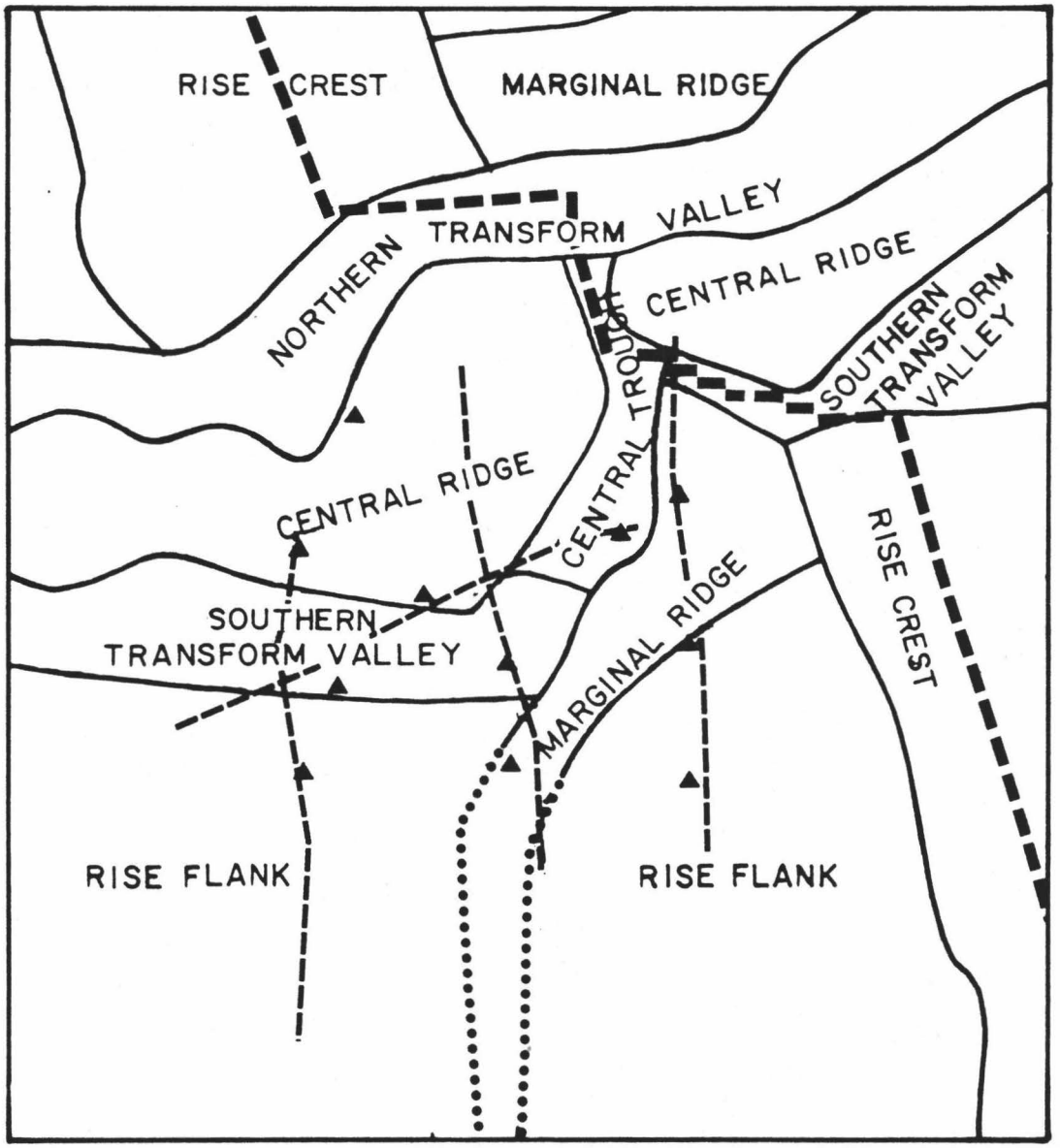

105°30'
105° W
104°30'
15°30'
15° N
14°30'
14° N
2600
2800
3000
3200
3400
205
2
526
210
612
611
6
3

Figure 5. Main bathymetric features in the Orozco Fracture Zone region.


RISE CREST
MARGINAL RIDGE
NORTHERN TRANSFORM VALLEY
CENTRAL TROUGH
CENTRAL RIDGE
SOUTHERN TRANSFORM VALLEY
CENTRAL RIDGE
SOUTHERN TRANSFORM VALLEY
MARGINAL RIDGE
RISE CREST
RISE FLANK
RISE FLANK

CHAPTER III

DATA SET

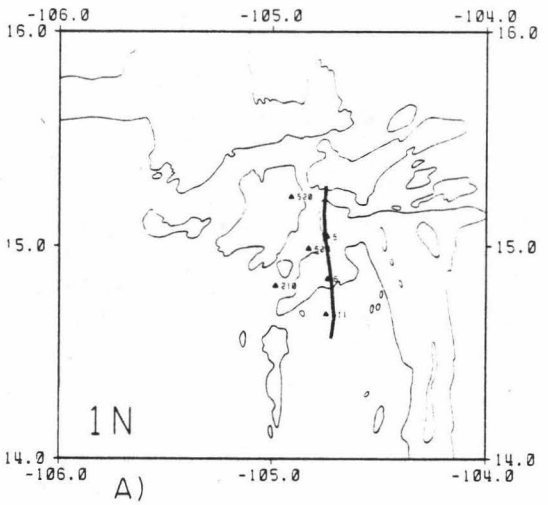
The main emphasis of the passive phase of ROSE was on earthquake location so, despite the large number of instruments deployed, only a relatively small number are located at points suitable for the analysis of the refraction profile data. In addition, because the bathymetry and tectonics of the fracture zone were poorly known in advance of the experiment, the shotlines are not located over the most interesting part of the fracture zone, namely the active transform faults and the relay zone (Figure 5). Profile 1N extends from the southern transform valley, over the ridge south of the fracture zone and onto the rise flank on crust of 0.5 M.y. age. Shotline 2N covers the peak of the transform ridge west of the relay zone, crosses the inactive part of the southern transform fault, and onto the southern tip of the southern ridge. Line 3N samples the western end of the transform ridge and the rise flank of 2 M.y. age. Line 4N is perpendicular to the others and extends along the axis of the fracture zone from the slope between the relay zone and the western transform ridge, along the ridge and along the southern transform fault. The regions of the fracture zone that are sampled by this arrangement of shots and receivers are: the central transform ridge; limited portions of the central transverse trough; the southern transform valley, but mostly in the western, inactive part; and the flank of the EPR.

Profiles 1N and 2N were executed by the R/V Conrad, and profiles 3N and 4N by the R/V Kana Keoki. All shots were 2.7 kg charges of Tovex. At the time of writing, not all of the recorded data have been processed and submitted to the ROSE archive and exchange facility. Data are available for Woods Hole (WHOI) instruments 2, 3, 4, 5 and 6 [Koelsch and Purdy, 1979]; University of Texas Marine Sciences Institute (MSI) instruments 205, 210, 213 and 214 [Latham et al., 1979]; Hawaii Institute of Geophysics (HIG) instruments 520 and 526 [Sutton et al., 1979]; and Oregon State University (OSU) instruments 608, 611 and 612 [Bibee et al., 1979]. Locations of all the receivers are shown in Figure 4. Not all these were used; the receivers used for each shotline are shown in Figure 6.

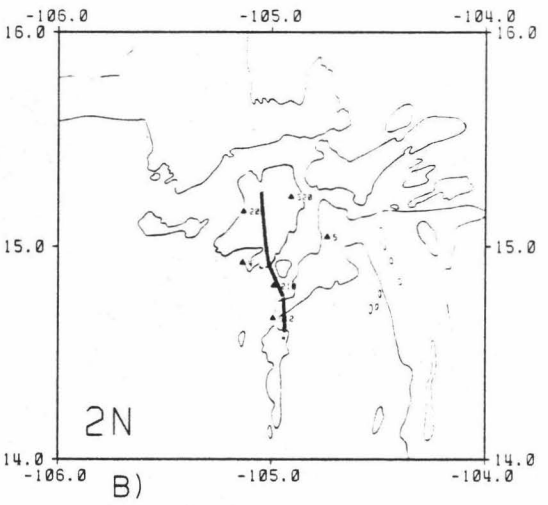
The recorded data are shown in Figures 7 for each shotline/receiver pair. In the plots of the data for instruments at large ranges from the profile, the traces are plotted against range projected on the profile rather than the real range. This does not display apparent phase velocities correctly but may show anomalous travel time delays. Only a single channel is shown in each case although more than one is usually available. The WHOI instruments consist of a hydrophone recorded with two gain settings; the MSI instruments contain a single vertical geophone; and both the HIG and OSU instruments have a hydrophone, a vertical geophone and an unoriented horizontal geophone.

The feature of the data in which I am most interested is the first arrival travel time information. This is mainly because arrival time

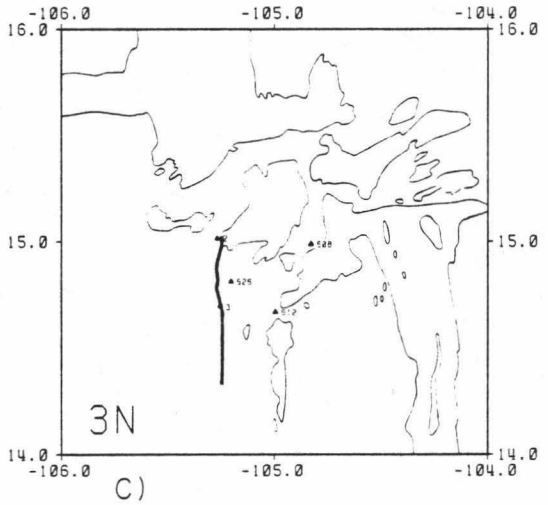
Figure 6. Shot and receiver locations for each of the four shotlines. The 3000 m contour is shown for reference.


-106.0
-105.0
-104.0
16.0
15.0
14.0
1N

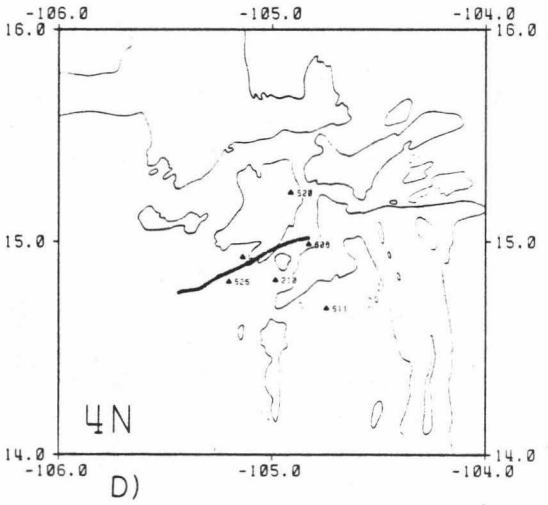
A)


-106.0
-105.0
-104.0
16.0
15.0
14.0
2N

B)


-106.0
-105.0
-104.0
16.0
15.0
14.0
3N

C)


-106.0
-105.0
-104.0
16.0
15.0
14.0
4N

D)

Figure 7. Data used in this thesis. The best data channel is shown for each receiver when more than one is available. Start times are not corrected for bathymetry nor for the water layer. All data is shown unfiltered. For those receivers that are offset by more than 10 km, the shot positions are shown projected onto to the profile of the shotline. The true range is used to calculate the reduced travel times.

a) station 2 (hydrophone), line 3N.
b) station 3 (hydrophone), line 3N.
c) station 4 (hydrophone), line 2N, (offset - 15.1 km).
d) station 4 (hydrophone), line 4N.
e) station 5 (hydrophone), line 1N.
f) station 5 (hydrophone), line 2N, (offset - 30.9 km).
g) station 6 (hydrophone), line 1N.
h) station 205 (vertical), line 2N, (offset - 15.2 km).
i) station 210 (vertical), line 1N, (offset - 29.1 km).
j) station 210 (vertical), line 2N.
k) station 210 (vertical), line 4N, (offset - 14.7 km).
l) station 526 (horizontal), line 3N.
m) station 526 (horizontal), line 4N.
n) station 608 (hydrophone), line 1N.
o) station 608 (hydrophone), line 3N, (offset - 48.9 km).
p) station 608 (hydrophone), line 4N.
q) station 611 (hydrophone), line 1N.
r) station 611 (hydrophone), line 4N, (offset - 39.9 km).
s) station 612 (hydrophone), line 2N.
t) station 612 (hydrophone), line 3N, (offset - 29.0 km).

LINE 3N 08H 002
T - X / 8.0
TIME (S)
RANGE (KM)

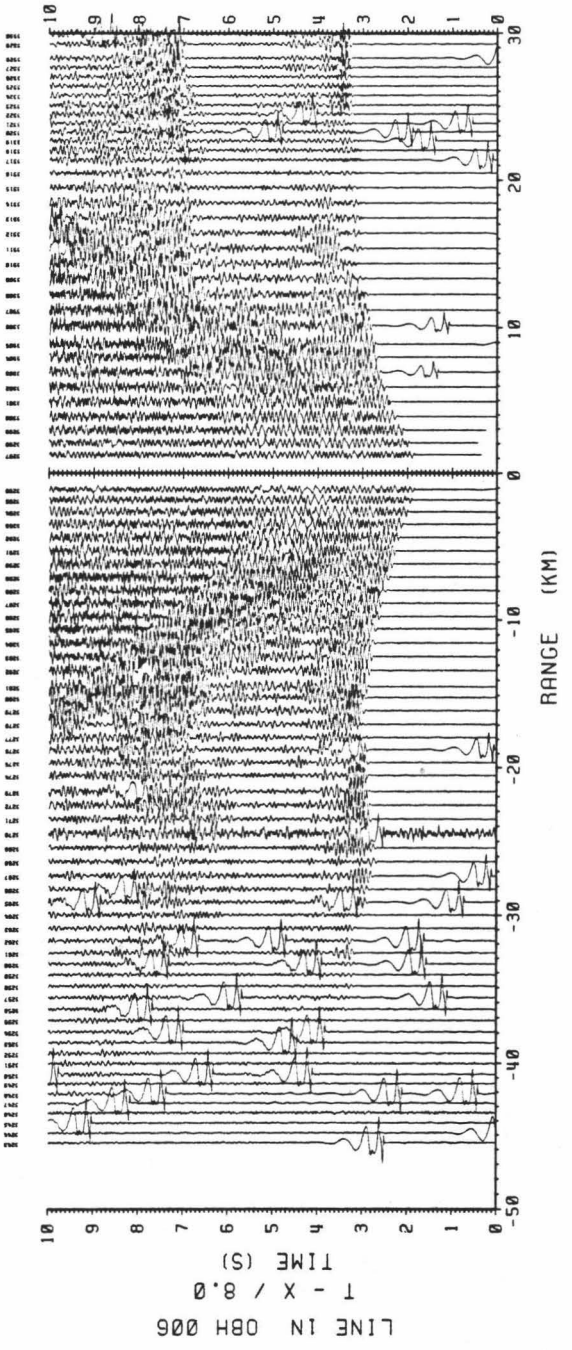
LINE 3N 08H 003
T - X / 8.0
TIME (S)
RANGE (KM)

LINE 2N 08H 00Ч
T - X / 7.7
TIME (S)
RANGE (KM)

LINE 4N 08H 004
T - X / 8.0
TIME (S)
RANGE (KM)

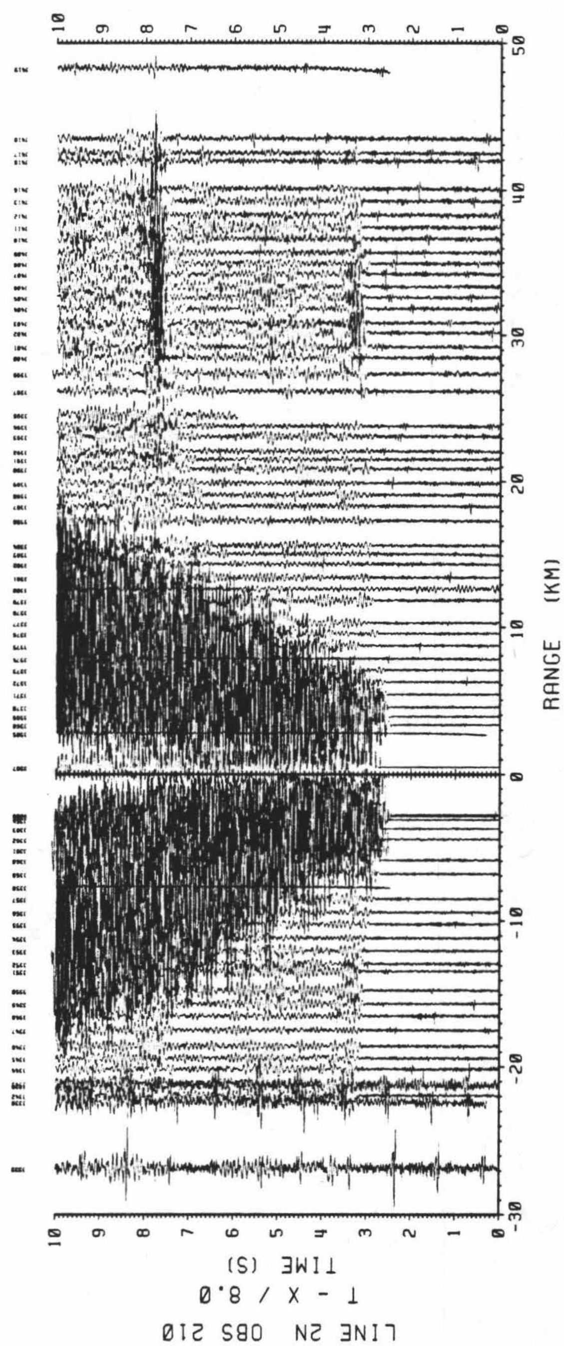
LINE IN 08H 005
T - X / 8.0
TIME (S)
RANGE (KM)

LINE 2N 08H 005
T - X / 7.7
TIME (S)
RANGE (KM)


LINE IN 08H 006
T - X / 8.0
TIME (S)
RANGE (KM)

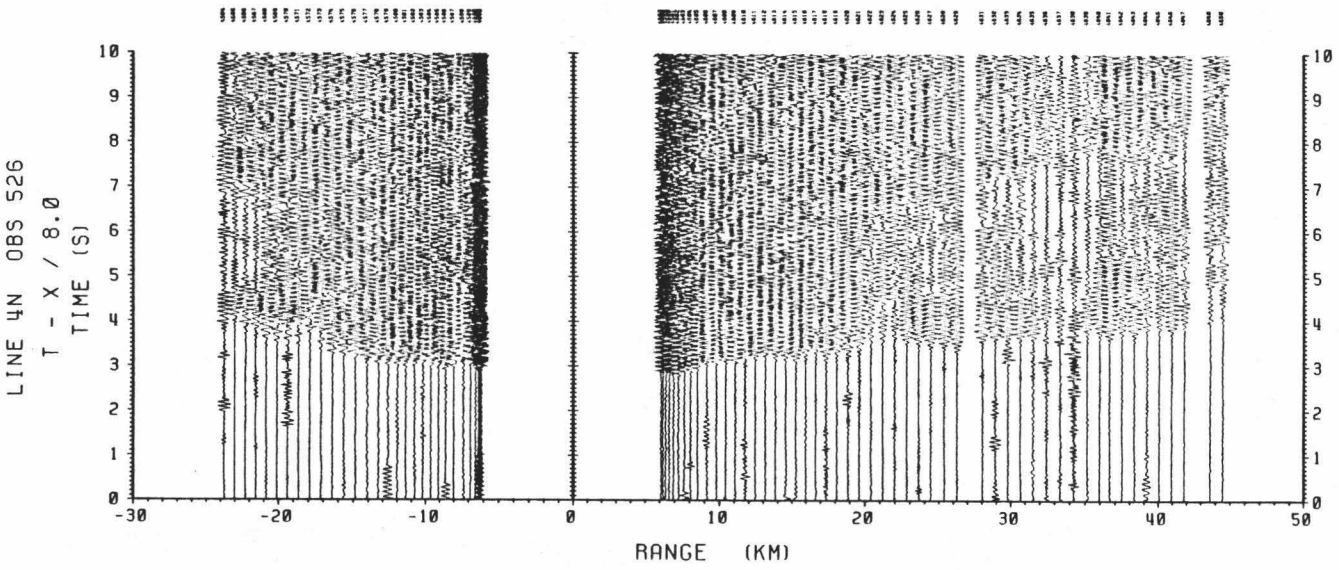
LINE 2N OBS 205
T - X / 7.7
TIME (S)
RANGE (KM)

LINE IN OBS 210
T - X / 7.7
TIME (S)
RANGE (KM)


LINE 2N OBS 210
T - X / 8.0
TIME (S)
RANGE (KM)
-30
-20
-10
0
10
20
30
40
50
0
1
2
3
4
5
6
7
8
9
10

LINE 4N OBS 210
T - X / 7.7
TIME (S)
RANGE (KM)

LINE 3N OBS 526
T - X / 8.0
TIME (S)
RANGE (KM)


LINE 4N OBS 526
T - X / 8.0
TIME (S)
RANGE (KM)
0
1
2
3
4
5
6
7
8
9
10
-30
-20
-10
0
10
20
30
40
50

LINE IN OBS 608
T - X / 7.7
TIME (S)
RANGE (KM)

LINE 3N OBS 608
T - X / 7.7
TIME (S)
RANGE (KM)

LINE 4N OBS 608
T - X / 8.0
TIME (S)
RANGE (KM)

LINE IN OBS 611
T - X / 8.0
TIME (S)
RANGE (KM)

LINE 4N OBS 611
T - X / 7.7
TIME (S)
RANGE (KM)

LINE 2N OBS 612
T - X / 8.0
TIME (S)
RANGE (KM)

LINE 3N OBS 612
T - X / 7.7
TIME (S)
RANGE (KM)

analysis is the easiest method of seismic interpretation. But it is also partly due to the great variety of instruments involved. It would be more difficult to compare amplitudes and wave forms for data recorded by instruments with different characteristic responses.

For the purpose of determining first arrival times the hydrophone data turned out to be the most useful. These hydrophone data typically have superior signal-to-noise ratio (SNR) to geophones. They also do not suffer from difficulties involved with coupling the seismometer to the ocean bottom [Sutton et al., 1980]. Signal to noise ratio is the most important criterion for the data because the particle motion is not desired. In those cases where there is no hydrophone available or where that channel is improperly recorded, one of the geophone channels is used.

The data recorded by the WHOI OBH's are clearly the best of what is currently available (Figures 7a-7g). The SNR for the first arrivals range up to 10:1. The rise times for the first arrivals are short, allowing accurate determination of the arrival time. The signal is typically damped within a second of onset, allowing the observation of the later arrivals. The WHOI receivers have the added benefit of having been located near to the shot profiles.

The MSI data were less informative (Figures 7h-7k). Of the MSI deployments, only OBS 210 was located within 10 km of a shot profile. The SNR of the first arrivals ranges up to about 3:1. The noise component generally has a higher frequency than the earth arrivals so can be filtered. However, even with a low-pass filter the arrivals at

OBS 210 for shots line on line 2N (Figure 7j) are barely discernible in the range of 10 to 24 km. Amplitude variations are sometimes inconsistent making correlation between adjacent traces more difficult.

The HIG data are shown in Figures 71 and 7m. Both figures are of data recorded by OBS 526 for lines 3N and 4N respectively. The only channel that was recorded for this instrument was the horizontal geophone. The first arrival saturates the recorder at the smaller ranges. The resonant signal overwhelms any possible later arrivals. Additionally there is no useful amplitude information. Most of the other HIG OBS's were deployed at greater distances from the shot profiles so they have not been processed.

The data from the hydrophone channel of the OSU OBS's are shown in Figures 7n-7t. The quality is variable. The recording made by OBS 611 (Figue 7q and 7r) compare favorably with the WHOI instruments. The shots on profile 4N are at a large distance from the location of OBS 611 so the arrivals in Figure 7r are weak. When first retrieved from the ROSE data archive and exchange system, some of the OSU data looked unusable. Application of whole second corrections to many of these traces appeared to bring the arrival times to the expected value (OBS 608, Figures 7n, 7m, 7o, and OBS 612, Figures 7s and 7t). However the accuracy of these corrections is difficult to estimate.

The seismic data, whether retrieved from the ROSE archive or digitized from the HIG OBS tapes, are first converted to a format suitable for disk storage and subsequent computer processing. The

signal traces are scanned on a video display terminal and the first arrivals are picked using a cursor that can be positioned to any sample of the trace. The pick is made by first visually correlating phases on adjacent traces of the seismogram and then choosing the closest inflection point. The maximum accuracy of a single sampling interval is not realized for the picks because of noise in the data. The uncertainty is determined qualitatively from the nature of the signal. As a general rule, the picks are assigned accuracies of 5 sampling intervals (0.05 seconds at 100 samples per second) when the signal to noise ratio is greater than 3:1 and the arrival is impulsive. Emergent arrivals or those masked in noise are assigned larger uncertainties which may range up to 15 samples. Shot and receiver locations are determined by satellite fixes with subsequent range accuracies of up to 0.5 km. There is a location in the exchange data format for range error for each shot/receiver pair but it is not used consistently by the groups submitting data. For shots at smaller ranges, where the water wave provides a range estimate, the accuracy can be increased up to 0.1 km.

CHAPTER IV

INVERSION METHODS

Other research in fracture zones has indicated that major changes in crustal structure occur in these areas. The bathymetry in the Orozco Fracture Zone is complex so it is expected that the velocity model will be highly variable. There are no simple methods for developing seismic velocity models in three dimensions so we have to proceed in three steps.

A common simplification made by seismologists is that the material properties in the Earth such as elastic wave velocity and density vary only with depth. This reduces the problem to one dimension. Furthermore it is often assumed that the crust consists of horizontal layers with constant seismic velocity or with constant gradient. In these cases there are analytic expressions for the travel time function and therefore the inverse problem of determining the parameters in the formula given the observed travel times can be solved.

The first stage in the analysis of the ROSE Phase 2 data is to develop a one dimensional model at each station for each shot profile. Two things are apparent in these models: they differ from each other and even individually do not fit the travel times exactly. This can be explained by departures in the crust from strict one-dimensionality. In other words, seismic velocity varies in the horizontal directions.

Lateral variation is difficult to resolve using the seismic refraction method because a typical path for a refracted waves has a large horizontal component. The travel time is affected by the entire length of the path and therefore measures only the spatial average of the velocity. In horizontally stratified media, the phase velocity measured from the travel time curve is equal to the velocity at the turning point depth of the ray. In an unstratified medium this is no longer true. The task becomes that of resolving the variation of velocity with depth from the variation in horizontal direction.

Lateral inhomogenity of small dimension may appear as local residuals between the observed travel times and the least squares fit of the travel time curve. Travel times between the shots and the off-line receivers can be used in this case because it is not necessary to measure the phase velocity from these data. Larger scale variation will affect the overall fit of the travel time curve and the resulting velocity-depth model so cannot be resolved by residuals.

To accomplish the first phase, the effects of topography must be removed from the travel times since it is assumed that the velocity varies only with depth. Following Whitmarsh [1978] a correction is made to move all entry points to an equivalent horizontal datum. This correction requires an input velocity model and the ray parameter of the arrival,

$$\Delta t = \Delta z \, [(1 - v_w^2 p^2)^{1/2} / v_w - (1 - v_s^2 p^2)^{1/2} / v_s];$$

where Δt is the time correction; Δz is the difference in depth between the datum plane and the seafloor beneath the shot; v_w is the water column velocity; v_s is the velocity at the seafloor; and p is the ray parameter of the arrival.

5.1 Reparametrization of Travel Times.

The method of Dorman and Jacobson [1981] is used to calculate the initial models for each OBS/shotline combination. They decomposed the velocity-depth model into horizontal layers of constant velocity gradient dv/dz. Then they formulated a linear relationship between the travel time parameters $\zeta(p) = T + pX$ and $\tau(p) = T - pX$ and the reciprocal of the velocity gradients. The variable p is the ray parameter for a given arrival. The parameters ζ and τ for the j-th observation may be written in the form

$$\sum_j (dz/dv)_j \, f(v_i, p_j)$$

where v_i is the velocity in the i-th layer. In matrix notation, this becomes

$$\underline{y} = \underline{A}\,\underline{x};$$

$$\text{where } \underline{y} = [\, \zeta(p_1), \tau(p_1), \zeta(p_2), \tau(p_2), \ldots]^T \qquad (m \times 1);$$

$$\underline{x} = [(dz/dv)_1, (dz/dv)_2, \ldots]^T \qquad (n \times 1);$$

and $\underline{A}$ is a m x n matrix with coefficients $f(v_i, p_j)$.

Typically, each pair of ζ, τ parameters is used to resolve a single layer, so the matrix is over-determined and may be solved by the method of least squares.

Before describing the mechanics of the parameter inversion, let us note that ζ and τ are considered as functions of p. Since at this point the seismic medium is assumed to be horizontally stratified. the ray parameter is given by the slope of the travel time curve. However the travel time data are a discrete set of points which are subject to error. To determine the slope, the points must be fit by some form of differentiable curve.

There are three requirements for the data smoothing:

1) Random errors in the data should be removed.
2) True changes in the slope and curvature should be preserved.
3) The resulting curve should have negative second derivative everywhere.

The first two requirements are fairly obvious and contradictory. It is not entirely possible to discriminate errors from change in the character of the curve. The second requirement applies to data near critical points where different branches of the travel time curve cross. It is necessary to fit these as sharply as possible, both to constrain

the depth of the interface but also its thickness. However, it should be stressed at this point that first arrivals alone cannot demonstrate the existence of a first order discontinuity. The third requirement is imposed by the inversion routine. It assumes that the model has velocities that increase monotonically with depth.

Typically these requirements are met by least squares fitting of the data with low order polynomials or splines. The use of piece-wise continuous polynomials requires a subjective determination of the cross-over point. Spline functions on the other hand need only be given the degree of smoothing desired. There are several methods for least squares smoothing by splines. The most satisfactory for the present application was to use a number of knots much smaller than the number of observation points. Knots are the points where the adjacent third order polynomials of the cubic spline are forced to be equal in value and in first derivative. In smoothing they are the points that constrain the fitted curve. The error that is minimized by the fixed knot routine is

$$e = \left(\sum_i r_i \, w_i \right)^{1/2};$$

where $r_i = f_i - s(x_i)$;

$w_i = (x_{i+1} - x_{i-1}) \; / \; (x_n - x_1)$;

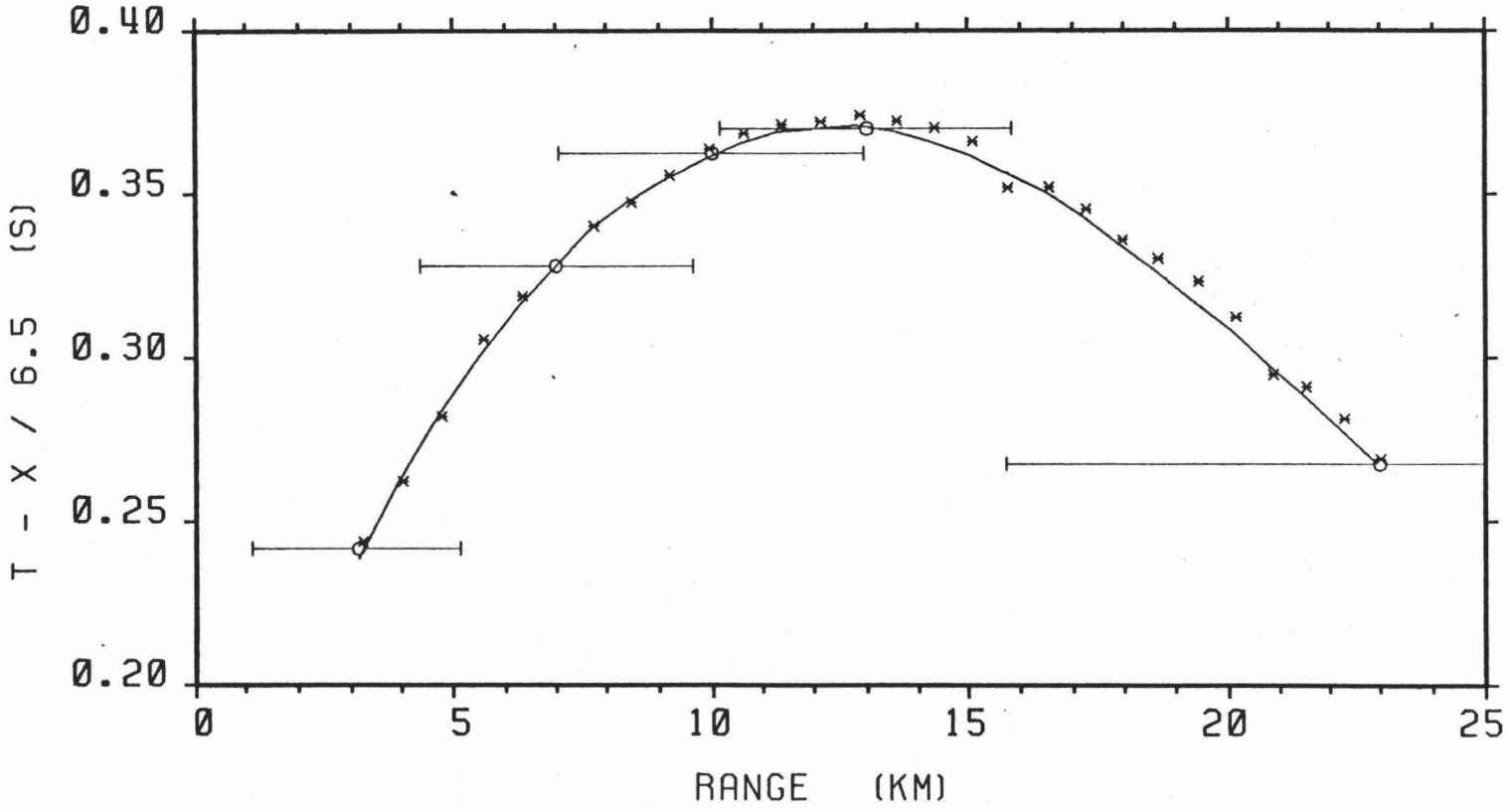
x_i , f_i are the data pairs;

and s is the least squares cubic spline approximation [DeBoor and Rice, 1968; IMSL, 1982].

The more common form of spline fitting is to place a knot at each point and to constrain the fit by a smoothing parameter.In the method of fixed knots the knot positions can be completely specified along the independent variable. Knot locations very close to one another permit tight curvature. This allows better local control of the curvature than a global smoothing constraint.

Dorman and Jacobson [1981] note that smoothing methods usually reduce the number of independent coefficients in the data. They give a general method for calculating the optimal interval between successive data points. The interval is largely dependent on the local curvature of the travel time function. The curvature of the smoothed travel time function is dependent on the knot spacing. This suggests a relationship between the concepts of optimal data spacing and knot spacing. Instead of using Dorman and Jacobson's general treatment to calculate the best distribution of data points, the knot locations of the best fit were used. As a check, the second derivative or the curvature of the spline is determined at the knot locations and from that, the optimal interval is calculated and compared with the spacing between knots. In general the two do not correspond exactly though the difference is rarely large (Figure 8). This usage is sub-optimal in two ways: 1) if the distance

Figure 8. Least squares cubic spline fit of travel time picks. Knots used to generate the curve are shown by open circles. The second derivative of the curve is evaluated at the knots and used to calculate the optimal separation between data points. Most efficient use of the data is made when the separation of the knot locations is equal to the optimal intervals.


0.40
0.35
0.30
0.25
0.20
T - X / 6.5 (S)
0
5
10
15
20
25
RANGE (KM)

between knot locations is less than the optimal distance, then there is some dependence between parameters and this means that there are off-diagonal elements in the obervation covariance matrix. 2) If the distance is greater than the optimal distance, then not all of the available data are being used to generate the model. In addition the confidence intervals on the parameters are calculated on the assumption that the data are optimally spaced.

It should be possible to do an iteration where the knot locations are given and checked against optimal data spacing until the knot locations are in optimal positions. This turned out to be impossible in practice for several reasons. The velocity-depth structure in this region generates a travel-time curve that may be divided into a region between 0 and about 10 km where the curvature is large, and a region above 10 km where the slope is almost constant. There may be a change in slope between the two regions. The shot spacing in the ROSE Phase 2 profiles was on the order of 1 km so in general there are less than 10 points used to define the lower part of the curve. Least squares fitting of third order polynomials is unstable under these conditions and small changes in knot locations may cause large changes in local slope and curvature. Additionally it is always necessary to maintain a downward curvature over the entire function. It would be interesting to test this with a data set with much higher shot density.

5.2 Inverse Problem for the Velocity-depth Function.

Once the data has been reparametrized to ζ and τ, the system of equations can be generated and solved. Following Wiggins [1972] and Jackson [1972], the least squares estimate is calculated with the singular value decomposition rather than with the more standard (and faster) QR decomposition or Cholesky decomposition of the normal equations [Lawson and Hanson, 1976]. Much has been written about this method [e.g. Lanczos, 1961; Aki and Richards, 1981] but it is useful to review some of its salient features. The singular value decomposition of a m x n matrix is

$$\underline{A} = \underline{U}\,\underline{S}\,\underline{V}^T$$

where $\underline{U}$ is a m x m matrix of the singular vectors in the data space (column space) of $\underline{A}$; $\underline{S}$ is a n x n diagonal matrix with the singular values; and $\underline{V}$ is a n x n matrix with the singular vectors in the model space (row space) of $\underline{A}$. The solution of $\underline{Ax} = \underline{b}$ is calculated by

$$\underline{x} = \underline{V}\,\underline{S}^{-1}\underline{U}^T\underline{y}.$$

In practice, the term $\underline{S}^{-1}$ causes difficulty, especially when calculated on a computer, because some of the singular values may be

close to zero. However, if there are r non-zero singular values. the matrices may be partitoned as

$$\underline{U} = [\underline{U}_r \mid \underline{U}_o] \; ; \quad \underline{S} = \left|\begin{array}{c|c} \underline{S}_r & 0 \\ \hline 0 & 0 \end{array}\right| \; ; \quad \underline{V} = [\underline{V}_r \mid \underline{V}_o];$$

where the column vectors in the matrix partitions subscripted with r correspond to the non-zero singular values. The generalized inverse solution can then be defined as

$$\underline{x} = \underline{V}_r \underline{S}_r^{-1} \underline{U}_r^T \underline{y}.$$

This effectively removes the small singular values and their associated vectors from the solution.

Different types of generalized inverses can be calculated without the singular value decomposition, but the decomposition products $\underline{U}$, $\underline{S}$, and $\underline{V}$ provide useful information. The product $\underline{E} = \underline{U}_r \underline{U}_r^T$ has been called the data information density (Wiggins, 1972). It is the orthogonal projection matrix onto the column space of A. The estimate of $\underline{Ax}$ is given by $\underline{Ey}$. If the matrix $\underline{A}$ is square and of full rank, then all the singular values can be used and r = m and $\underline{E} = \underline{I}$, the m x m identity matrix. In this case $\underline{Ax}$ is equivalent to $\underline{y}$. In the present case, the system of equations is overdetermined so r < m. There exists a component $\underline{U}_o$ of $\underline{U}$ and the solution is no longer exact. The columns of $\underline{E}$ are a now a weighted average applied to the components of y. The

projection onto the orthogonal complement of the column space of $\underline{A}$ is given by $\underline{U}_o\underline{U}_o^T$. The residual $\underline{y} - \underline{Ax}$ is therefore given by $\underline{U}_o\underline{U}_o^T\underline{y}$.

The matrix $\underline{R} = \underline{V}_r\underline{V}_r^T$ is called the resolution matrix of the model parameters. The rows of $\underline{R}$ are the averaging coefficients used to calculate the model. If r = n and there is no $\underline{V}_o$ space, then again $\underline{V}_r\underline{V}_r^T = \underline{I}$, the n x n identity matrix. In this case the computed model is unique. If r < n then the model is not unique because any component in V_o does not contribute to the model. This generalized inverse solution is the solution with the minimum norm.

The covariance of the model parameters is additionally given by $\langle\underline{\Delta x},\underline{\Delta x}\rangle = \sigma^2\underline{V}_r\underline{S}_r^{-2}\underline{V}_r^T$ if the errors on the observations are independent. From this formula it is evident that the magnitude of the covariance depends strongly on the singular values. Small singular values will result in broad confidence intervals on the computed result.

Examination of these quantities can be of benefit in controlling the nature of the model, and this is the value of the singular value decomposition. If the confidence intervals on the model parameters are unacceptably large, then the solution can be attempted again with fewer singular values. This will reduce the magnitude of the covariance but at the expense of the resolution in both model and data space. The fewer singular values included in the solution, the broader the averaging interval across adjacent parameters, and the smoother the model.

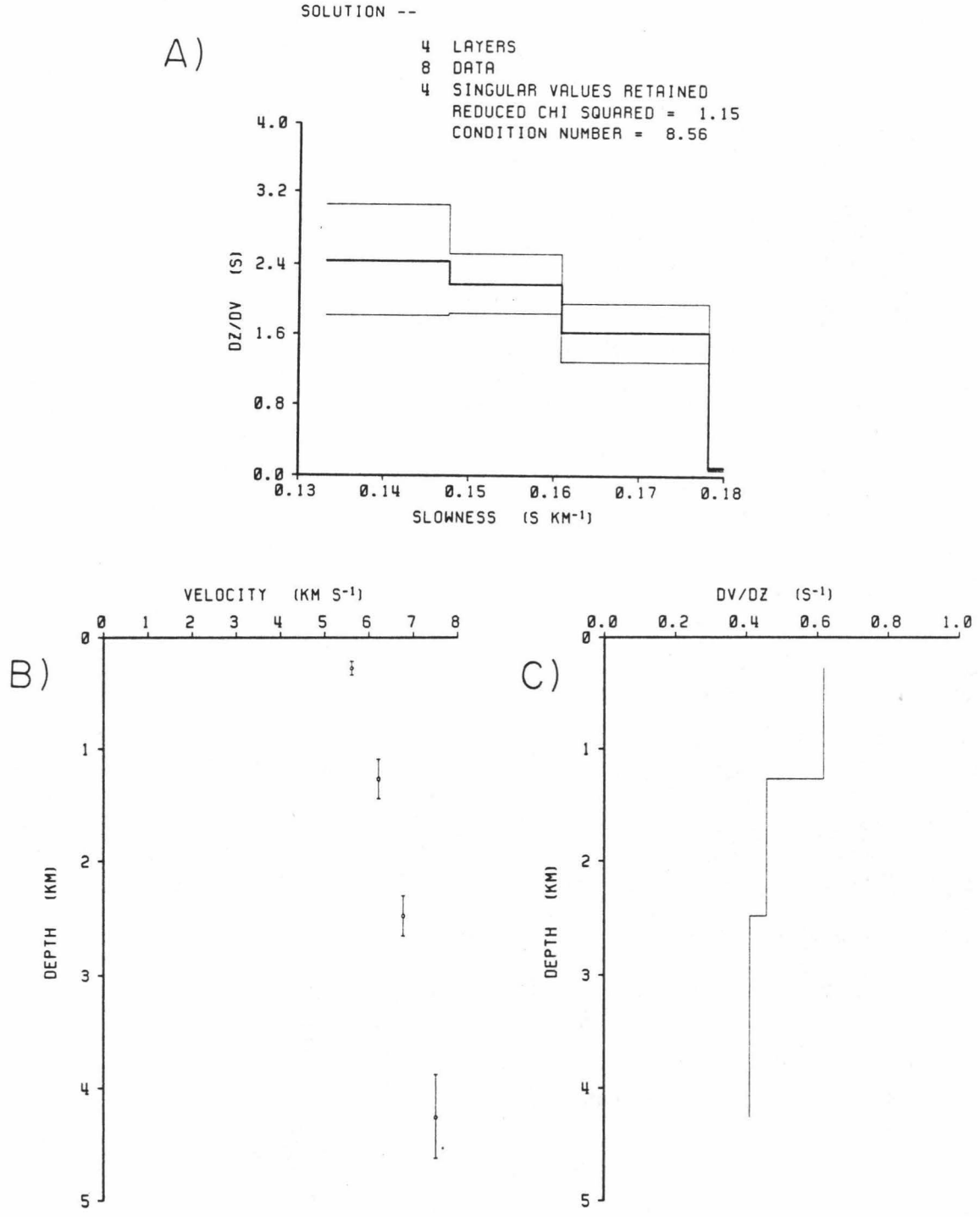
The quantity given by the solution of the least squares eqution is a vector of the inverse velocity gradients dz/dv as a function of ray parameter (Figure 9a). The confidence intervals on these model data are calculated from the covariance matrix. If the input parameters are independent then the error on the model parameters are given by the diagonal elements of the covariance matrix.

To get the ultimate result of depth to turning point as a function of velocity, the inverse gradients are integrated with respect to velocity which is available as the vector of ray parameters (Figure 9c). This assumes that the velocity gradients dv/dz is constant between model points (Figure 9b). The depths are linear combinations of the model parameters so to calculate the confidence intervals on the depths, the covariance matrix is propagated through the result [Rao, 1973, Dorman and Jacobson. 1981].

5.3 Time Term Analysis.

The models computed with this method are generally incomplete. They are laterally homogeneous by assumption and smooth over velocity discontinuities because only first arrivals are used. There are no simple solutions for more complex media so usually one has to resort to modeling. The time term method is used to determine some of the deviation from the flat layer assumption before generating the models.

Figure 9. Solution for a 4-layer, 8-parameter model.
a) Solution of the linear equation, inverse velocity gradient dz/dv as a function of ray parameter p.
b) Integration of the inverse velocity gradients with respect to velocity yields turning point depth as a function of velocity, shown with 95 percent confidence interval.
c) Integration in b) assumes that the velocity gradient is constant between model parameters.


SOLUTION --
A)
4 LAYERS
8 DATA
4 SINGULAR VALUES RETAINED
REDUCED CHI SQUARED = 1.15
CONDITION NUMBER = 8.56
DZ/DV (S)
4.0
3.2
2.4
1.6
0.8
0.0
0.13
0.14
0.15
0.16
0.17
0.18
SLOWNESS (S KM-1)
VELOCITY (KM S-1)
B)
DEPTH (KM)
DV/DZ (S-1)
C)

The time term method is based on the assumption that the refracted ray path and its associated travel time may be decomposed into three parts

$$t = x / v_N + \delta_s + \delta_r;$$

where t is the total travel time; x is the range between source and receiver; v_N is the seismic velocity of the refracting layer; δ_s is the time required to travel between the source and the refractor; and δ_r is the time required to travel from the refractor to the receiver. The latter two terms are called the time terms or delay times at the source and receiver respectively and take the form

$$\sum_{i=1}^{N-1} z_i \, (1 - v_i^2 / v_N^2) / v_i;$$

where z_i and v_i are the thickness and velocity of the overlying layers. The assumptions are that the interfaces do not depart greatly from horizontal and that the horizontal propagation takes place along the interface above the N-th layer. The second assumption conflicts with the premise of the velocity-depth inversion which is that rays are refracted upward by positive velocity gradients. The conflict is not serious because a constant layer model with first order velocity discontinuities is a limiting case of the gradient model. There will be

some error in the velocity of the refracting layer because the phase velocity changes with range.

It is possible to take all the time term data and calculate a model by least squares [Willmore and Bancroft, 1961; Morris, 1969]. The approach taken here is less rigorous. Plots of the reduced travel times are generated for each shotline/receiver combination. By the assumptions made, the reduced travel time is equal to the sum of the source and receiver time terms. By comparing a given shot line for different receivers, any difference in the delay times is due to a combination of effects during the horizontal propagation and the delay at the receiver. It is assumed here initially that there is no variation in horizontal velocity.

The time term calculated for the receiver is then subtracted from the reduced travel time to yield the source time terms for each shot. These are then plotted for all receivers on each shotline to map the variation along the shotline.

5.4 Verification by Forward Modeling.

Under differing assumptions there are techniques to model travel times, amplitudes, and waveforms. One of the most effective waveform modeling method is the reflectivity method [Fuchs and Muller, 1978; Kennett, 1978]. This method assumes horizontal stratification and therefore is useful far away from the center of the transform fault on line 3N. Near the center of the transform fault, the variation in

bathymetry is significant and it is postulated that the deeper structure is also variable. In this area, lateral variation must be accomodated by the modeling scheme. Ray tracing in two dimension can be used to model the travel times. A program was used to model diving rays and post-critical reflections [Sinton and Hussong, 1981]. These are the dominant phases in oceanic crust where there are few velocity discontinuities. The crustal structure is parametrized by surfaces of constant velocity which can vary laterally in depth. From these surfaces, the seismic velocity and velocity gradient can be specified at any point in the medium. The program is based on the shooting method, whereby an initial incidence angle is specified for each ray and it is allowed to propagate until it reaches the surface or exits the model. As the ray propagates, its direction is determined by the local velocity and gradient. The reciprocal of the velocity field is integrated along the ray path to give the travel time. Additionally, the density of rays reaching the surface in the neighborhood of a certain point is a first-order indication of the amplitude of the arrival there.

The most significant departure from lateral homogeneity is at the ocean-crust interface. This is because the sea-floor is the site of the greatest velocity contrast. It is also true that the bathymetry is well known compared to any other structure so careful modeling of its effects is essential. The two-dimensional ray tracing includes bathymetric effects explicitly. In areas of complex sea-floor this is the most satisfactory way of accounting for travel time delays because they are calculated for each arrival individually.

CHAPTER V

INTERPRETATION

Some general observations concerning all data can be made at the outset. Refracted waves can be detected out to a distance of 50 km in most cases. Refractions reflected at the water surface usually are visible to even greater ranges. There are few phases visible between the first arrival and the first water column multiple. Shear waves are seen sporadically. Lewis and Snydsman [1979] have attributed the generation of converted shear waves in some cases to the presence of a sedimentary layer. Indeed the strongest shear waves are seen on the oldest crust at line 3N. The 3.5 kHz depth profile records are inconclusive as to whether sediments are present.

Past the critical angle for the base of the crust, the strongest arrivals are those due to the PmP phase. These are visible on all records and indicated a sharp velocity transition between the crust and mantle. The slope of the asymptote indicates a lower crustal velocity of 7.2 to 7.5 km s^{-1}. Otherwise the velocity variation is smooth over the entire thickness of the crust. The only other phases that are visible are those that are refrected at the water-crust interface such as PP and PmPPmP.

With the possible exception of profile 4N, the P_n arrivals are not visible. It takes a very small velocity gradient to generate a substantial interference head wave [Braile and Smith, 1978] so the

velocity in the uppermost mantle must be very nearly constant. Large shots were detonated throughout the transform region. Using these Trehu and Purdy [1983] have determined an upper mantle seismic velocity of 8.1 km s^{-1}.

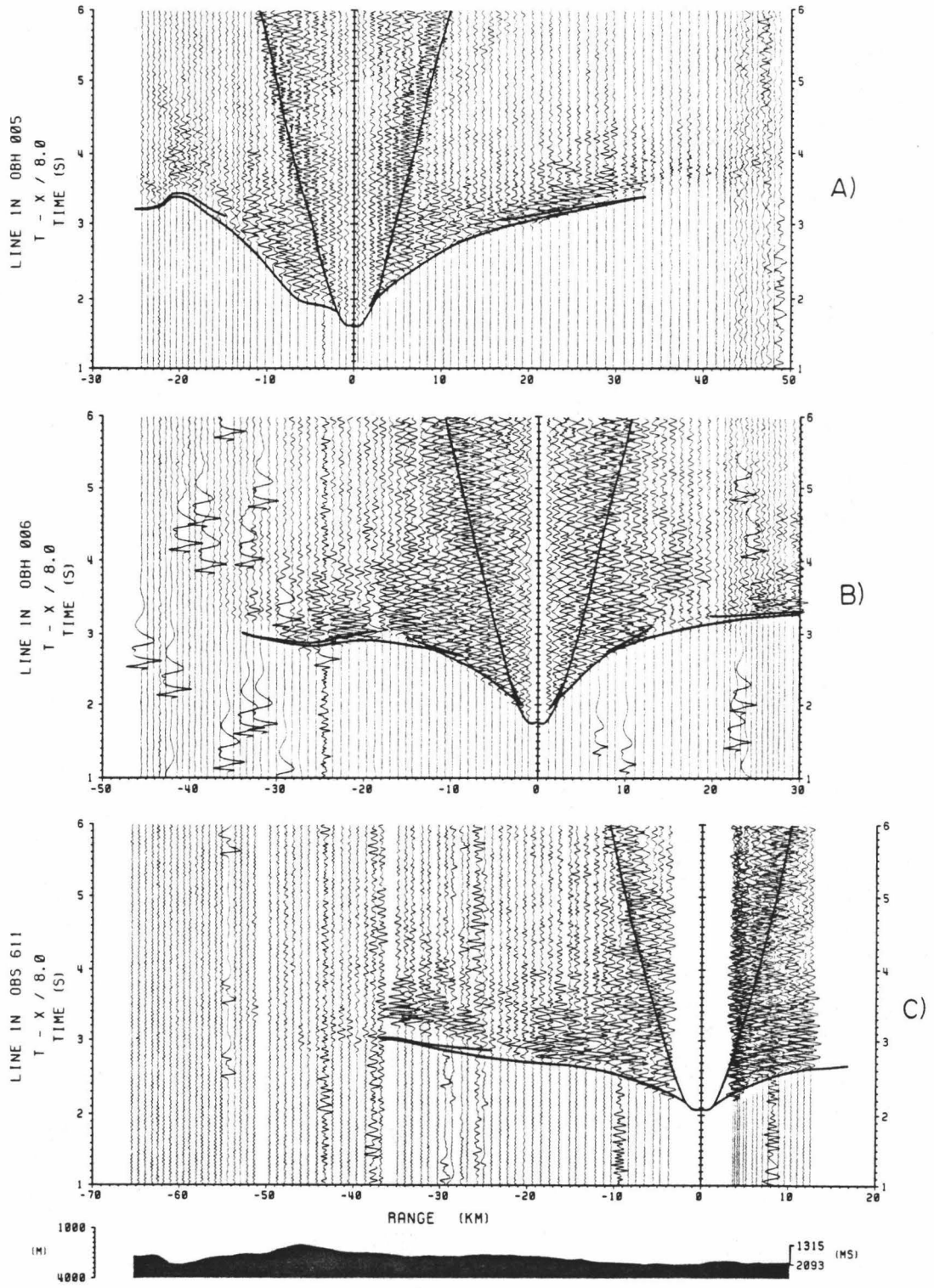
5.1 Line 1N.

Line 1N runs north-south along the east side of the central trough. It extends from the central ridge, crosses the active part of the southern transform fault, runs along the marginal ridge and onto the normal crust south of the fracture zone (Figures 4 and 5). The profile is at an average distance of 25 km from the East Pacific Rise crest so the age of the crust is 0.5 M.y. at most. There are five stations with data available for this line. They are WHOI OBH's 5 and 6, MSI OBS 210, and OSU instruments 608 and 611. The data for the in-line stations 5, 6 and 611 are shown in Figure 10.

The slopes north of OBH 5 are very steep and horizontal datum corrections are not adequate. For this reason, inversion using horizontal layer was not attempted for this region.

Between stations 5 and 6 the amplitudes vary smoothly for all three receivers. The amplitude increasing starting at a range of 17 km south of OBH 5 is due to interference between the refraction at the base of the crust and the PmP phase, the reflection from the Moho. This triplication starts at 19 km north of OBH 6. Normally one would expect that the refraction through the upper mantle, the P_n phase to become the

Figure 10. Record sections for receivers on line 1N. Data is shown unfiltered and uncorrected for bathymetry. The predicted travel times from the model shown in Figure 12a are superimposed.
a) OBH 5.
b) OBH 6.
c) OBS 611.


LINE IN OBH 005
T - X / 8.0
TIME (S)
A)
LINE IN OBH 006
T - X / 8.0
TIME (S)
B)
LINE IN OBS 611
T - X / 8.0
TIME (S)
C)
RANGE (KM)
1000
(M)
4000
1315
2093
(MS)

first arrival a short distance beyond the start of the triplication. The P_n phase is not observed in any of these data but it is expected that it exists and is too weak to observe. Because of this, the arrival time data are used in the least squares inversion process only to a distance of 22 km south of receiver 5 and north of receiver 6. Beyond that range it is likely that the picks were of the PmP and would give erroneous results.

At receiver 5 (Figure 10a) it is evident that the time intercept of the refracted arrivals is delayed from the the direct water arrival. Ewing and Purdy's [1982] formula indicates that this is a result of a very high velocity gradient in the uppermost crust. Extrapolating the velocities to the sea floor at station 5 gives a surface velocity of 2.0 km s^{-1} and a gradient of 5.0 s^{-1} to a depth of 0.8 km. Inversion of these data yields an estimate for the equivalent flat layer model with a virtually constant gradient of 0.6 s^{-1} between 0.8 and 6.0 km depth. The results of this velocity-depth inversion are shown by the circles with error bars in Figure 12b.

Between stations 6 and 611 the amplitudes vary less smoothly. The energy is reduced between 14 and 24 km south of OBH 6. This is also visible as a cut-off of energy at the same location which is 32 km south of OBH 5. This is a constraint on the total thickness of the crust. There is a triplication at 24 km south of receiver 6 and 26 km north of receiver 611. Inversion of travel times to a range of 25 km south of OBH 6 and north of OBH 611 yields a more complicated model. A gradient

of 0.5 s^{-1} betwen 0.8 and 1.2 km depth overlies a gradient of 1.0 s^{-1} between 1.2 and 2.0 km, and a gradient of 0.2 s^{-1} to a depth of 3.0 km. Below 3.0 km there is no information. This model is shown by the circles with error bars in figure 12d.

It is apparent from the results of the two inversions that there is a major change in structure along the length of line 1N. The time terms may resolve some of the finer scale structure. Figure 11 shows the reduced travel times for all the data with a shot to receiver range greater than 25 km. Under the assumption of horizontal layering, the reduced travel time is equal to the combined delay time for the shot and receiver.

Too minimize the mismatch between travel times for the same shot recorded at different receivers it was necessary to introduce a characteristic delay for each receiver. Figure 11 shows that the total time terms at the north of line 1N are .25 to .3 s greater than at the south. This may be due to increasing thickness of the crustal layer, or by decreasing velocity in the crust, or alternatively to the misapplication of the bathymetry corrections in the area.

Ray tracing can provide more information. The post-critical reflection from the crust-mantle interface provides an important constraint on the crustal thickness. Modeling by ray-trace can also be used to gain some understanding of the amplitude variation. Increased amplitudes are usually an indication of a layer of higher velocity gradient. These are evident in the record sections at 18-32 km south of

Figure 11. Time terms for shots along line 1N. Shot positions are plotted as distance along the profile relative to the position of OBH 6 so that a given shot will plot at the same points for all receivers. The time is the reduced travel time which is equivalent to the combined source and receiver time term.

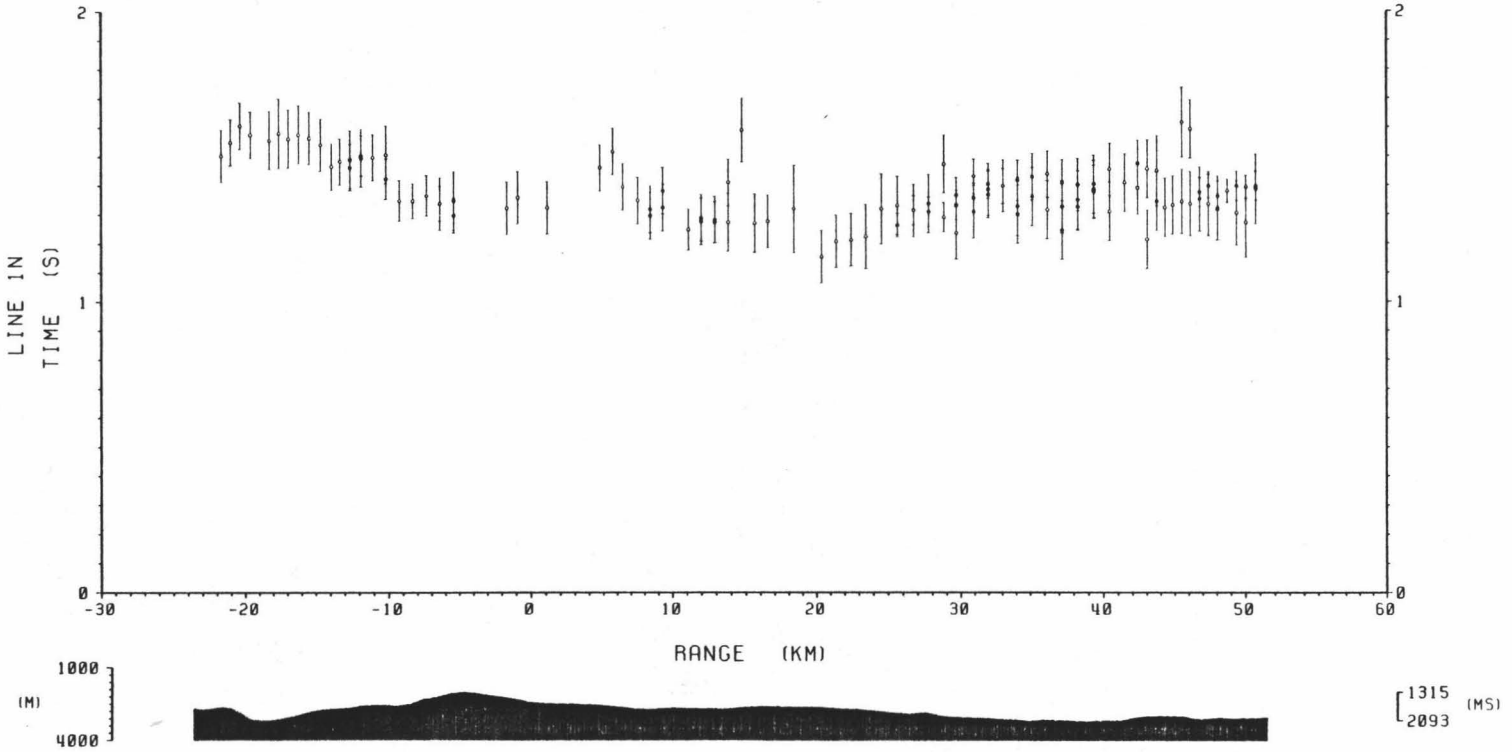
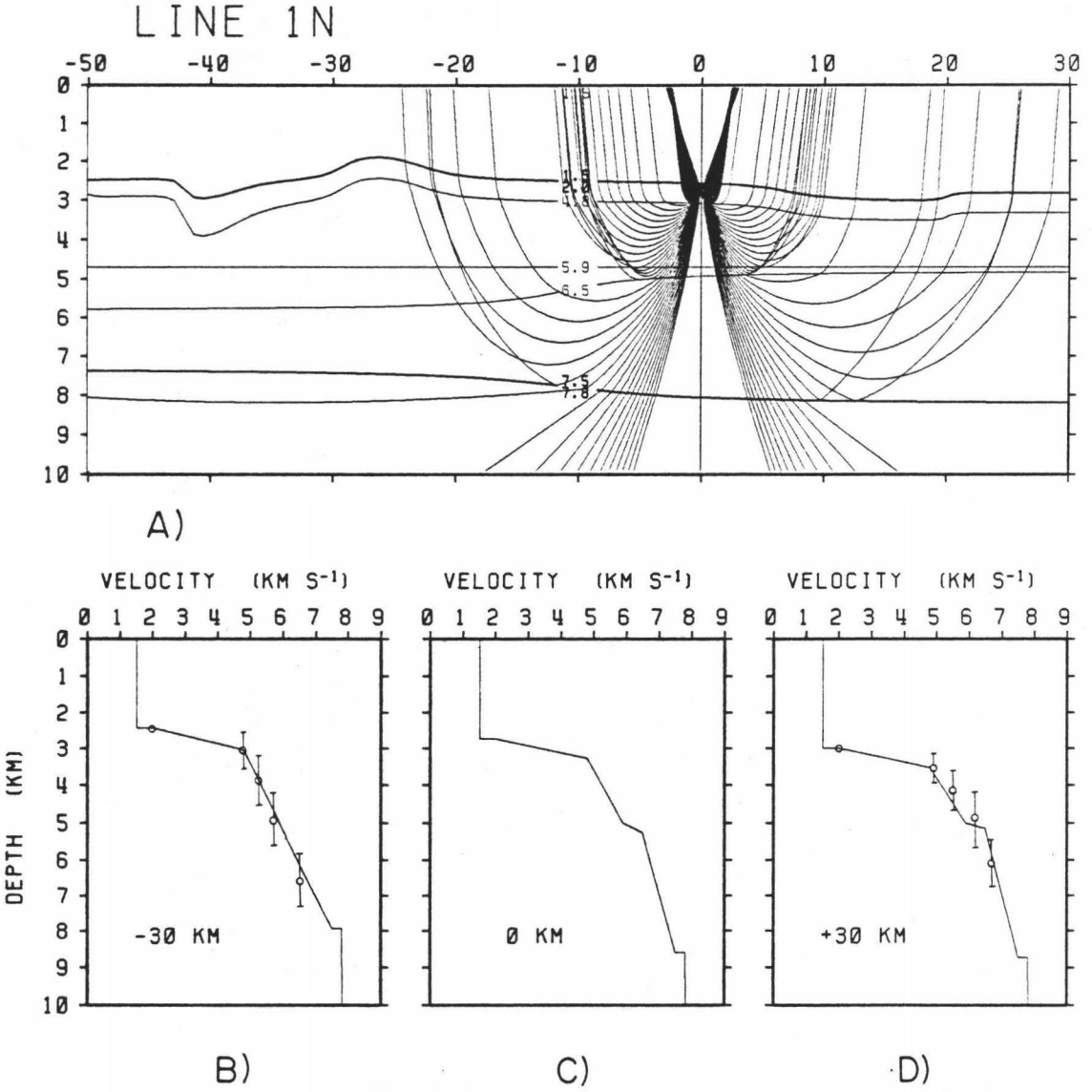

LINE 1N
TIME (S)
2
1
0
-30
-20
-10
0
10
20
30
40
50
60
RANGE (KM)
1000
(M)
4000
1315
2093
(MS)

Figure 12. Velocity-depth model for the profile of line 1N.
a) Isovelocity contours for the model. Bold lines indicate first order discontinuities. Rays originate from the position of OBH 6. Vertical exaggeration is 3 x 1.
b) Velocity-depth function sampled at 30 km north of OBH 6. The results of the inversion of the travel time data at OBH 5 for shots between OBH 5 and OBH 6 are shown by the discrete points with confidence intervals. When building the ray-tracing model, the velocities were constrained to lie within the error bounds of the least squares inversion.
c) Velocity-depth function directly beneath OBH 6.
d) Velocity-depth function sampled at 30 km south of OBH 6. Inversion results are for travel time data at OBH 6 for shots to the south of the receiver.


LINE 1N
-50 -40 -30 -20 -10 0 10 20 30
0 1 2 3 4 5 6 7 8 9 10
5.9
6.5
7.5
7.8
A)
VELOCITY (KM S-1)
0 1 2 3 4 5 6 7 8 9
DEPTH (KM)
-30 KM
0 KM
+30 KM
B)
C)
D)

OBH 5 (Figure 10a), 19-30 km north of OBH 6, 22-30 km south of OBH 6 Figure 10b), and 14-19 and 26-35 km north of OBS 611 (Figure 10c).

Figure 12a shows the final model for the crust at line 1N with some sample rays originating at receiver 6. It is important to note that the lines defining the model are velocity contours and should not necessarily be interpreted as boundaries between different crustal layers (e.g. oceanic layer 2a, 3, etc...). The solid lines in Figures 12b, 12c, and 12d are the velocity-depth relation sampled at -30, 0 and +30 km along the profile relative to the position of station 6.

At the north end the velocity gradient is constant. The model calculated directly from the travel times is not contradicted by the modeling. To the south, the calculated model was not adequate. It was necessary to postulate a high gradient layer at a depth of 2.0 km beneath the sea floor to match the amplitudes at receiver 6 and 611. Additionally, since the beginning of the PmP phase starts at greater range for 611 and to the south of 6 than to the north of 6, it appears that the Moho must be raised by about 1 km to the north of receiver 5. However, since the sea floor is higher in that region, there is no significant reduction in the crustal thickness.

5.2 Line 2N.

Shot line 2N parallels line 1N, 30 km to the west on the opposite side of the central trough. The profile extends from the peak of the central ridge, across the intersection of the central trough and the

western part of the southern transform valley, and onto the continuation of the marginal ridge on the south of the fracture zone (Figures 4 and 5). There were data from five receivers available, WHOI instruments 4 and 5, MSI instruments 205 and 210, and OSU instrument 612.

The data are less informative than for line 1N. Only receivers 210 and 612 were within 10 km of the profile. Figure 13 shows those as well as the data for OBH 4 which was at an offset of 12 km. It was necessary to adjust the start times of many of the shot traces for receiver 612 and it is impossible to gauge the reliability of these corrections. Only those which seem reasonable are included in the plot.

Models were calculated by least squares for the first arrival times north and south of OBS 210. The onsets are difficult to pick in the region between 13 and 26 km north of the instrument. After 26 km the amplitudes increase with the start of the PmP phase.

The velocity was extrapolated to the sea floor resulting in a surface velocity of 2.0 km s^{-1} and an initial gradient of 5.0 s^{-1}. The inversion process gives a constant gradient of 0.5 s^{-1} for the upper 4.5 km to the north (discrete points in Figure 15b). To the south the model differs, with a 0.6 s^{-1} gradient for the upper 2 km and a gradient of 0.3 s^{-1} beneath that (Figure 15c).

The time terms are plotted for the shots line 2N in Figure 14. The relative delay at station 612 is 0.4 s. This is large but is also evident for OBS 612 on line 3N. However, even though OBS 612 is within 5 km of the shot profile, this delay is not visible for the shots for

Figure 13. Record sections for receivers along line 2N. See comments for Figure 10. Predicted travel times are for the model in Figure 15a.
a) OBH 4.
b) OBS 210.
c) OBS 612.

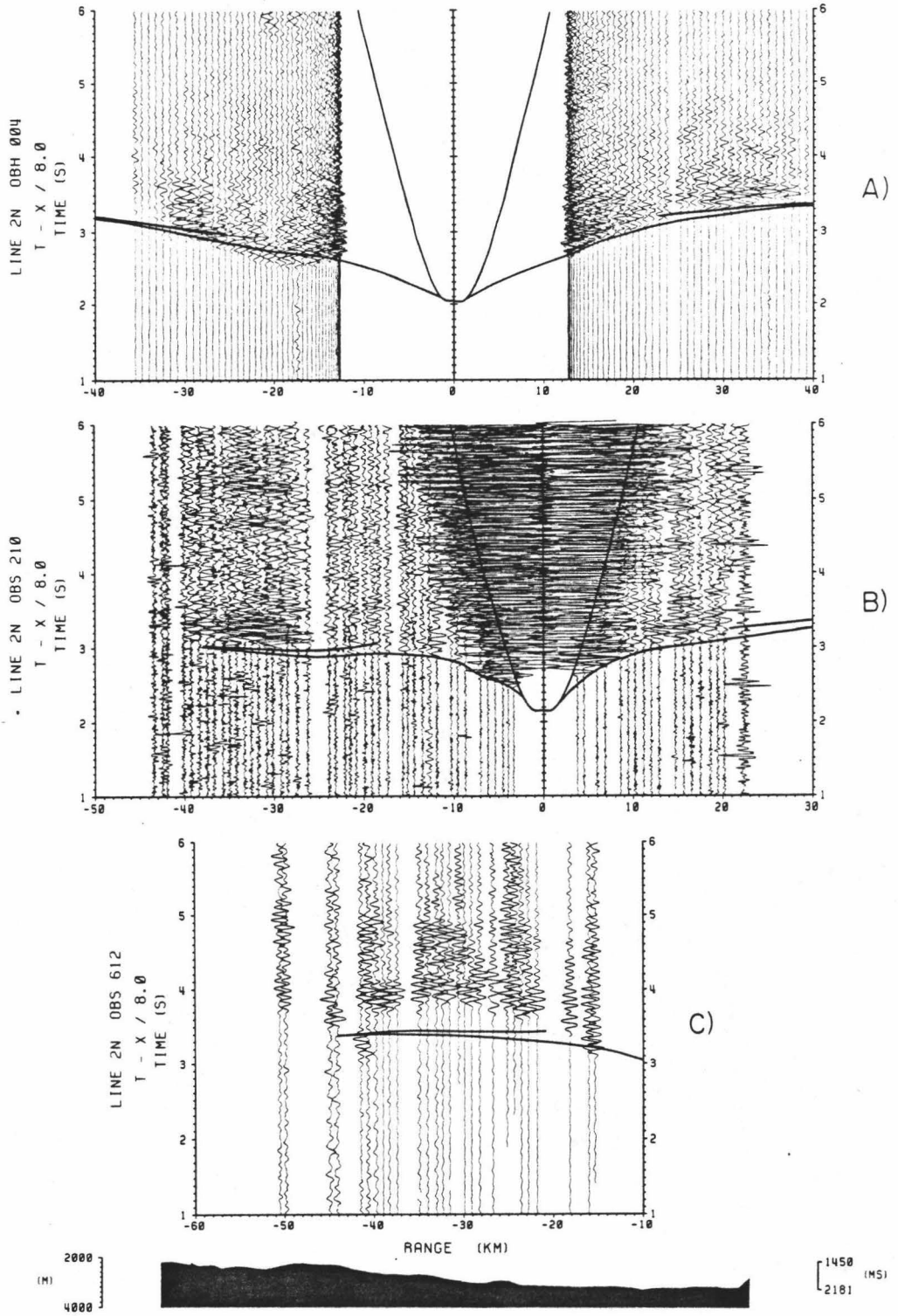

LINE 2N OBH 004
T - X / 8.0
TIME (S)
A)
LINE 2N OBS 210
T - X / 8.0
TIME (S)
B)
LINE 2N OBS 612
T - X / 8.0
TIME (S)
C)
RANGE (KM)
2000
(M)
4000
1450
2181
(MS)

Figure 14. Time terms for shots on line 2N. See comments for Figure 11. Range is relative to the postion of OBS 210.

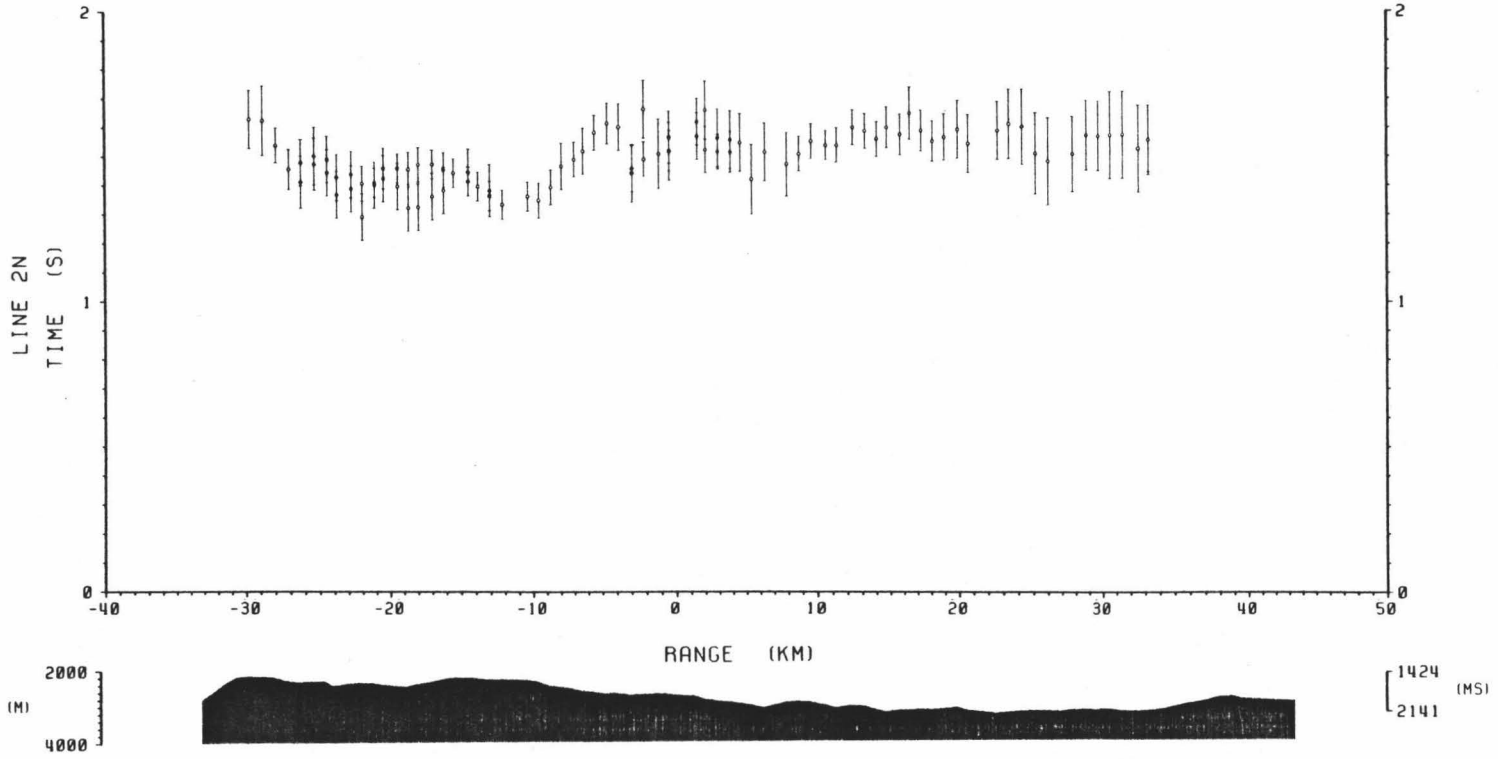
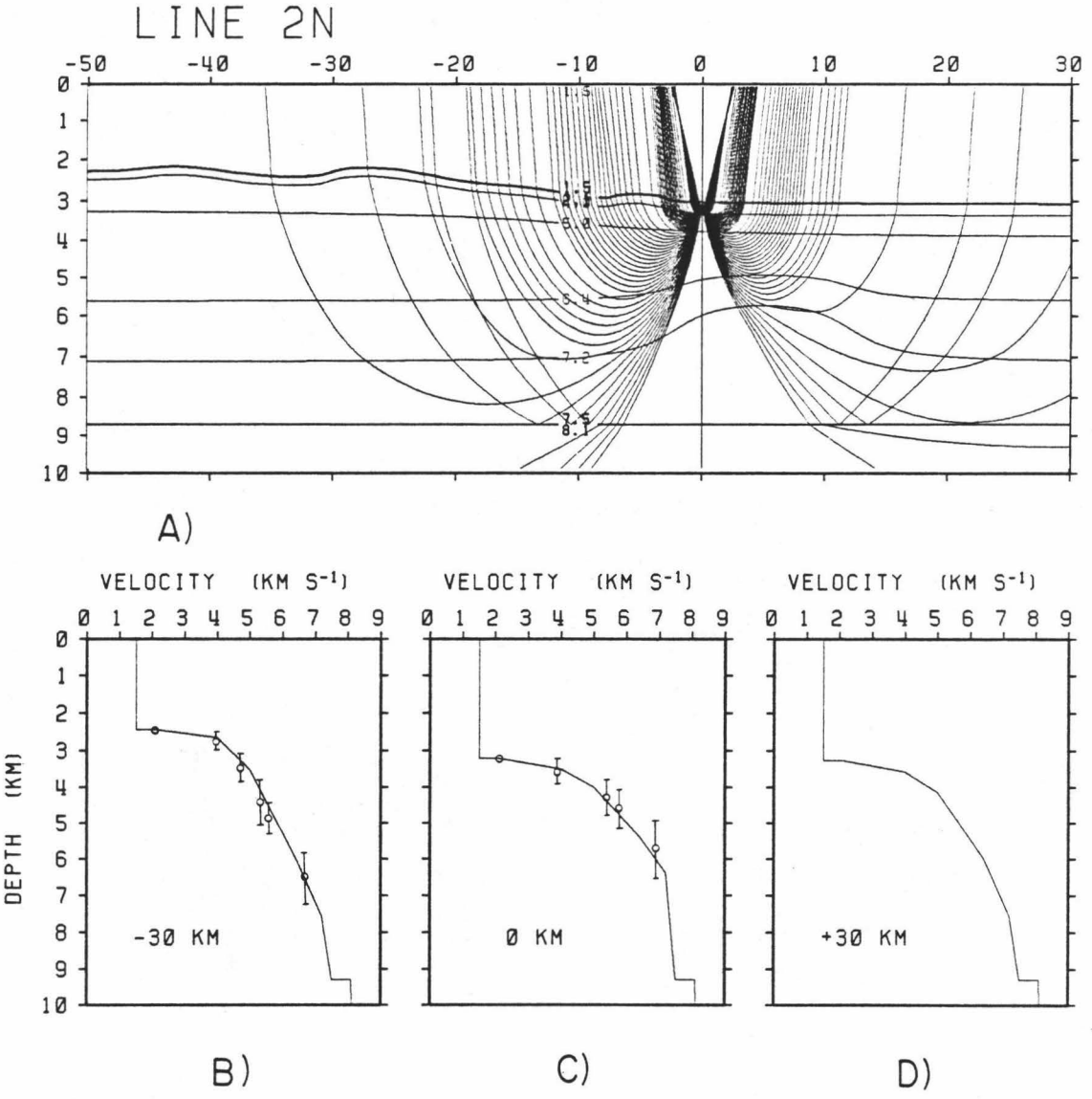

LINE 2N
TIME (S)
2
1
0
-40
-30
-20
-10
0
10
20
30
40
50
RANGE (KM)
(M)
2000
4000
1424
2141
(MS)

Figure 15. Velocity-depth model for the profile of line 2N. See comments for Figure 12.
a) Isovelocity contours for the model. Rays originate from the position of OBS 210.
b) Velocity-depth function sampled at 30 km north of OBS 210. Inversion results are for travel time data at OBS 210 for shots north of the receiver.
c) Velocity-depth function sampled directly beneath the receiver. Inversion results are for travel time data at OBS 210 for shots south of the receiver.
d) Velocity-depth function sampled at 30 km south of OBS 210.


LINE 2N
-50 -40 -30 -20 -10 0 10 20 30
0 1 2 3 4 5 6 7 8 9 10
1.5
5.4
7.2
8.1
A)
VELOCITY (KM S-1)
0 1 2 3 4 5 6 7 8 9
DEPTH (KM)
-30 KM
0 KM
+30 KM
B)
C)
D)

stations 4 and 5 (Figures 13a and b). At ranges -8 to -28 km, arrivals come in early relative to the others by .15 to .2 s. The ranges are relative to the orthogonal projection of the position of station 4 onto the shot line. This is the region of the slope of the central ridge.

Ray tracing is used to determine the depth of the Moho. To the north it was necessary to a 1.5 km thick layer of velocity 7.5 km s^{-1} at the base of the crust to force a triplication at a distance of 27 km. To the south, shots do not extend far enough to shown the Moho triplication, but extending the interface horizontally yields acceptable fits to the travel times. This model fits the arrivals at OBH 4 except for in the range 16 to 24 km north of the receiver. However this deviation is a result of trying to use a two dimensional scheme for such a large offset. It is unlikely that the structure is perpendicular to the shot profile in this region. To match the travel times at OBS 612 it was necessary to perturb the upper interfaces downward to lower the velocity in the upper crust at the south end of 2N. The velocity-depth is consistent with observations for the marginal ridge area under line 1N. The resulting velocity structure is similar to that at the northern extremity of line 2N (Figures 15a and 15d).

5.3 Line 3N.

The shot line furthest from the center of the transform region is 3N. It runs from the crest on the western part of the central ridge, across the inactive part of the southern transform valley, and onto

normal oceanic crust of 2 M.y. age (Figures 4 and 5). There were five instruments available. They were WHOI receivers 2 and 3, HIG receiver 526, and OSU receivers 608 and 612. Stations 2 and 3 were right on the shot profile; station 526 was offset by 6.5 km. The data for these three instruments are shown in Figure 16.

The amplitude increase due to the Moho arrivals starts at 29 km to the south of OBH 2 and 22 km north and south of OBH 3. The signal is saturated for OBS 526 so there are no amplitude constraints. The slopes of the refracted arrival times are nearly tangent to the direct water wave hyperbola. Extrapolation of these velocities to the sea floor yields a surface velocity of 4.0 km s^{-1}. First arrival travel times are used out to a range of 25 km for the least squares inversion. The results of the calculation are shown in Figure 18b for station 2 and Figure 18d for station 3.

The travel times computed from this model do not fit the data at OBS 526 (Figure 16b). Inversion of the travel times for 526 gives an model with a lower gradient in the upper crust and a sharper transition to the 7.2 km s^{-1} velocity observed for the lower crust. The average velocity for the whole crust is lower at station 526.

The combined source plus receiver time terms are shown in Figure 17. The range shown is along the profile and relative to the projection of the location of OBS 526 on the line. With the exception of receiver 612 there is little overlap of time term data. This made the determination of the relative receiver delays somewhat arbitrary.

Figure 16. Record sections for receivers along line 3N. See comments for Figure 10. Predicted travel times are for the model in Figure 18a.
a) OBH 2.
b) OBS 526.
c) OBH 3.

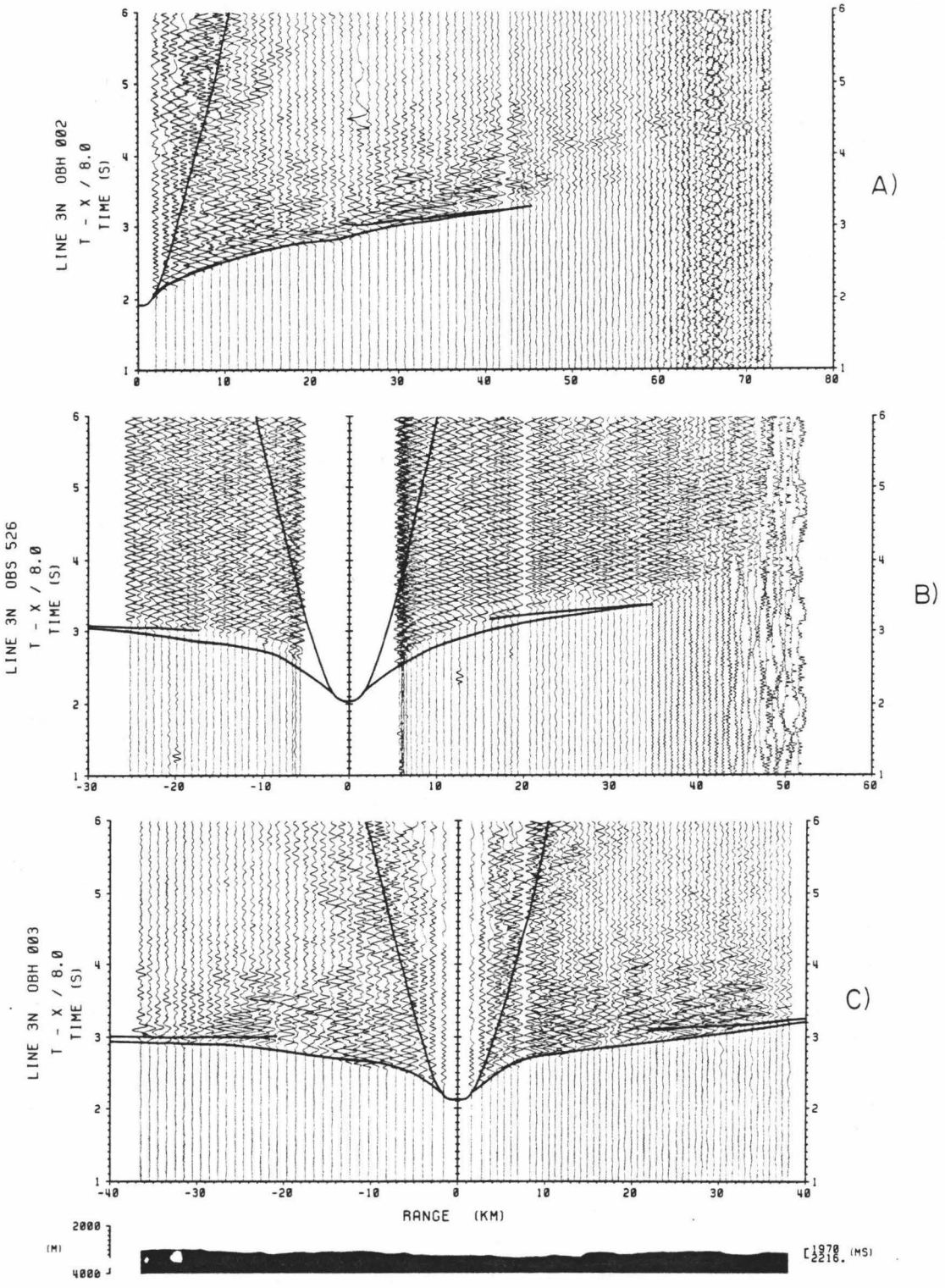
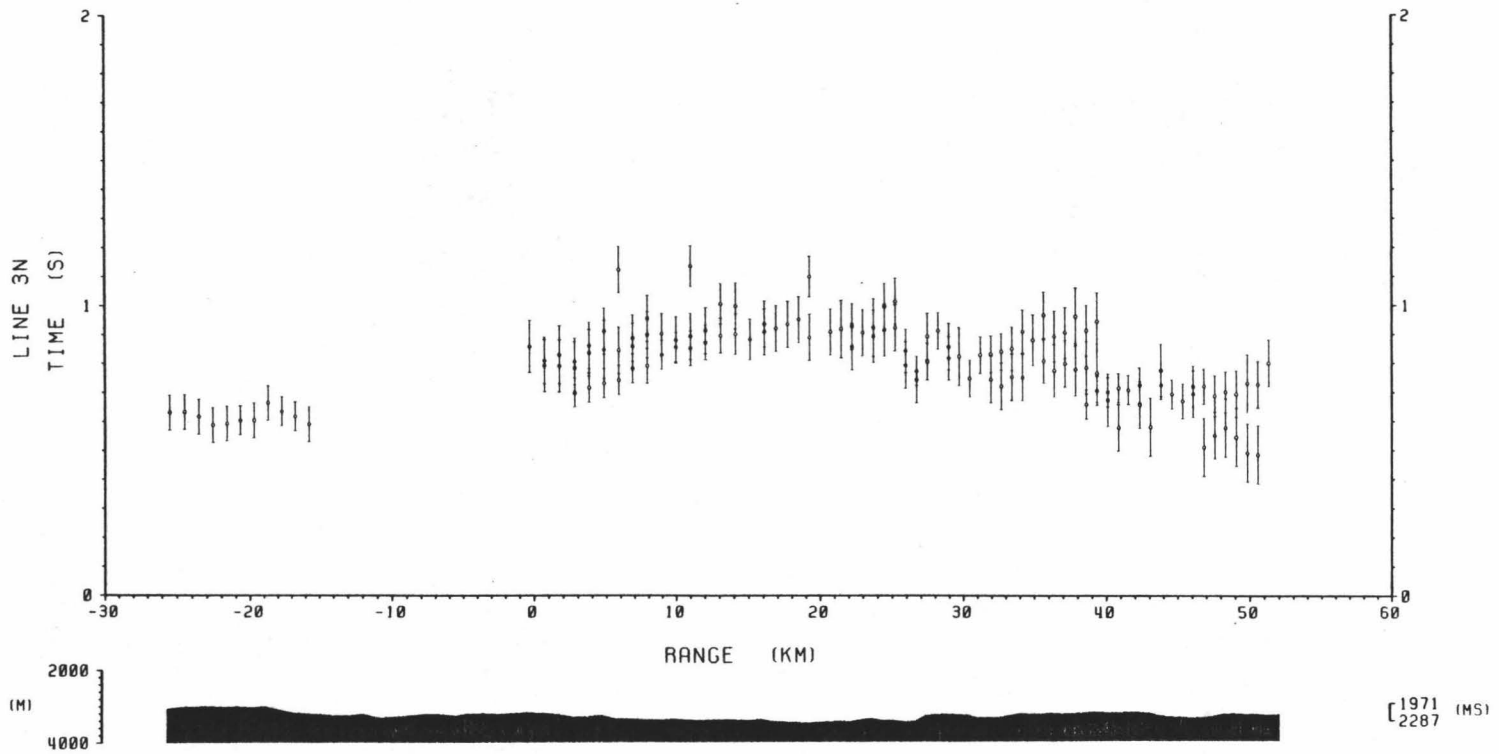

LINE 3N OBH 002
T - X / 8.0
TIME (S)
A)
LINE 3N OBS 526
T - X / 8.0
TIME (S)
B)
LINE 3N OBH 003
T - X / 8.0
TIME (S)
C)
RANGE (KM)
(M)
2000
4000
1970.
2216.
(MS)

Figure 17. Time terms for shots along line 3N. See comments for Figure 11. Ranges are relative to the position of OBS 526.


LINE 3N
TIME (S)
2
1
0
-30
-20
-10
0
10
20
30
40
50
60
RANGE (KM)
(M)
2000
4000
1971
2287
(MS)

The combined time terms show a larger delay for shots in the region of 0 to 20 km south of the origin. This may be related to the lower average velocity determined for OBS 526. However this result is mostly based on the relative delays observed at station 612 and I believe the delays occur in the horizontal propagation.

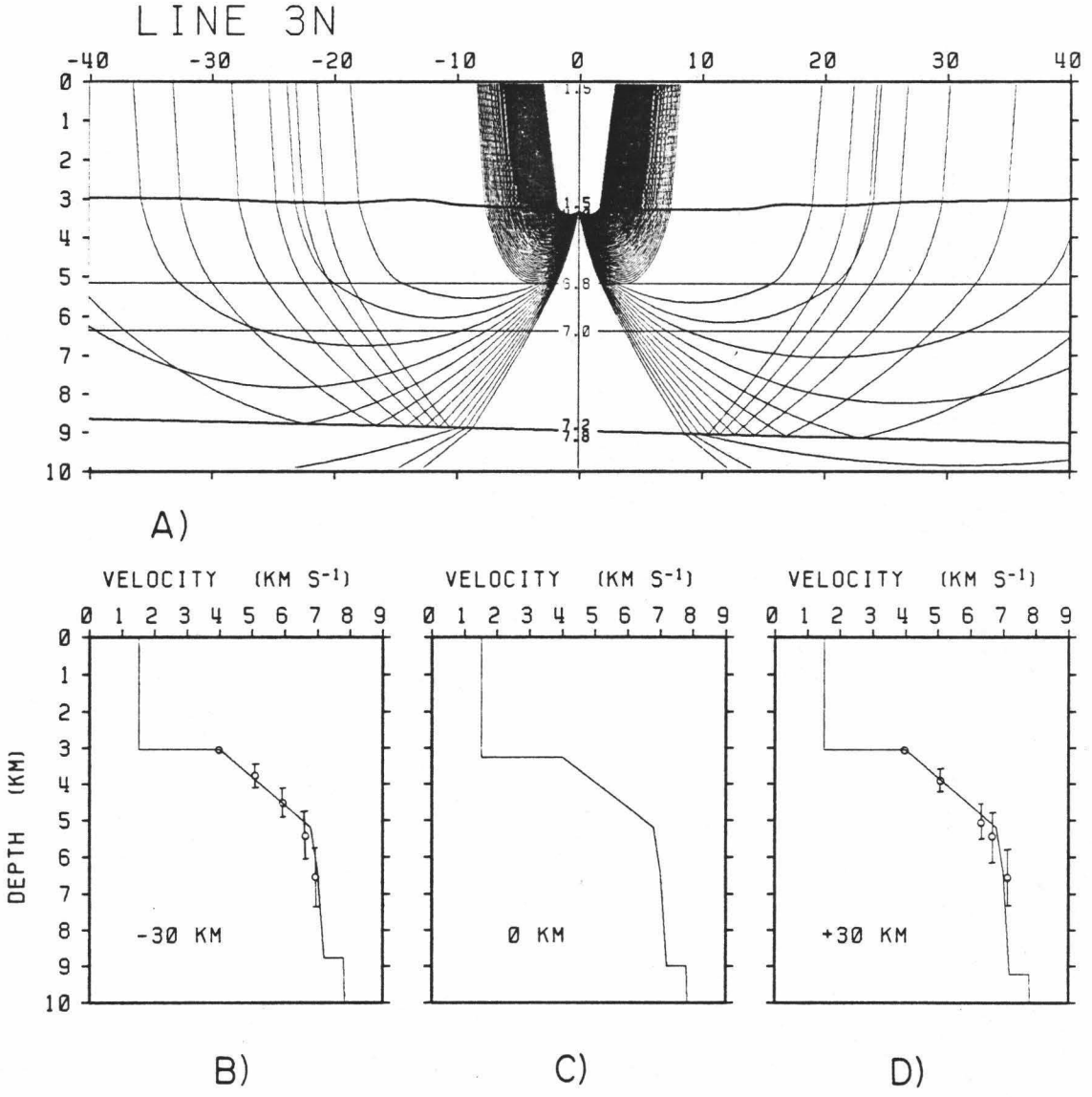
The total model is shown in Figure 18a. A southward slope was introduded to the Moho beneath station 2 to account for the larger distance to the start of the triplication. Travel times calculated from this model fit the data at receivers 2 and 3 well. At receiver 526 they fit at longer ranges. More weight is given to the data at 2 and 3 because of the higher quality. Unexplained delays of -0.08 s occur for both 2 and 3 near the location of 526. However, the sign of the delay is opposite to that expected from the other data. This may be indicative of some heterogeneity local to the vicinity of station 526.

The model is almost identical to the velocities measured by Manghnani et al. [1981] for the Samail (Oman) ophiolite. Reflectivity method synthetics were computed by Kempner and Gettrust [1982] and yield seismograms very similar to the data at receiver 2 (see their Figure 5).

5.4 Line 4N.

The only profile parallel to the transform fault is shotline 4N. This one also provides a tie between the north-south lines 1N through 3N. The shots run from the intersection of the central trough and the southern transform valley, and west along the slope between the

Figure 18. Velocity-depth model for the profile of line 3N. See comments for Figure 12.
a) Isovelocity contours for the model. Rays originate at the position of OBS 526.
b) Velocity-depth function sampled at 30 km north of OBS 526. Inversion results are for travel time data at OBH 2 for shots south of the receiver.
c) Velocity-depth function sampled directly beneath OBS 526.
d) Velocity-depth function sampled at 30 km south of OBH 526. Inversion results are for travel time data at OBH 3 for shots south of the receiver.


LINE 3N
-40 -30 -20 -10 0 10 20 30 40
0 1 2 3 4 5 6 7 8 9 10
1.5
1.5
6.8
7.0
7.2
7.8
A)
VELOCITY (KM S-1)
0 1 2 3 4 5 6 7 8 9
DEPTH (KM)
-30 KM
0 KM
+30 KM
B)
C)
D)

transform valley and the central ridge (Figures 4 and 5). Receivers providing data were WHOI OBH 4, MSI OBS 210, HIG OBS 526, and OSU OBS's 608 and 611. Stations 4, 526 and 608 were in line with the profile. These data are shown in Figure 19.

No inversions of the travel times were computed for this line. For receiver 526 the arrival times are erratic. This observation is unexplained. It was noted for receiver 526 on line 3N that there may be some very local heterogeneity. For 608 it was necessary to correct many of the start times with no estimate of the reliability of the process.

The travel times for the larger ranges are shown in Figure 20. The time terms are relatively constant for the western part of the profile but increase by .15 s to .25 s near the crest of the ridge and the slope into the central trough.

The model developed for line 3N generated travel times that match the data satisfactorily for the east side of station 526 and the west of station 4 but not at station 608. The time terms indicate that there is a change in structure to the east. The thickness of the upper, high gradient layer of the model was increased under the ridge and trough (Figures 21a and 21d). This lowers the gradient in the upper layer and lowers the average velocity fo the entire crust. This combined structure fits the travel times at the western end of the profile and also at close ranges to OBS 608 and to the east of OBH 4.

Figure 19. Record sections for receiver on line 4N. See comments for Figure 10. Predicted travel times are for the model in Figure 21a.
a) OBS 526.
b) OBH 4.
c) OBS 608.

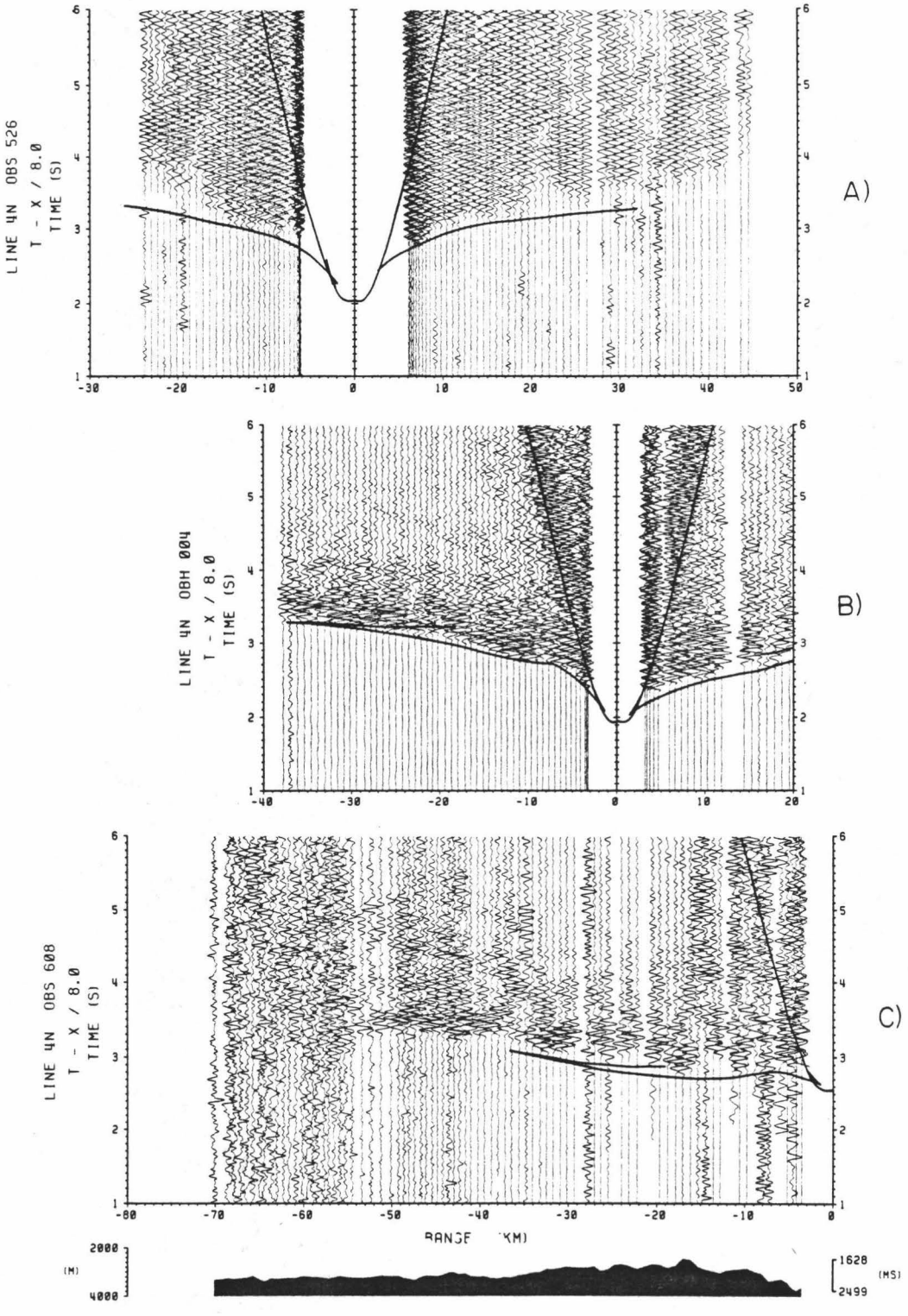

LINE 4N OBS 526
T - X / 8.0
TIME (S)
A)
LINE 4N OBH 004
T - X / 8.0
TIME (S)
B)
LINE 4N OBS 608
T - X / 8.0
TIME (S)
C)
RANGE (KM)
2000
(M)
4000
1628
(MS)
2499

Figure 20. Time terms for shots along line 4N. See comments for Figure 11. Ranges are relative to the position of OBH 4.

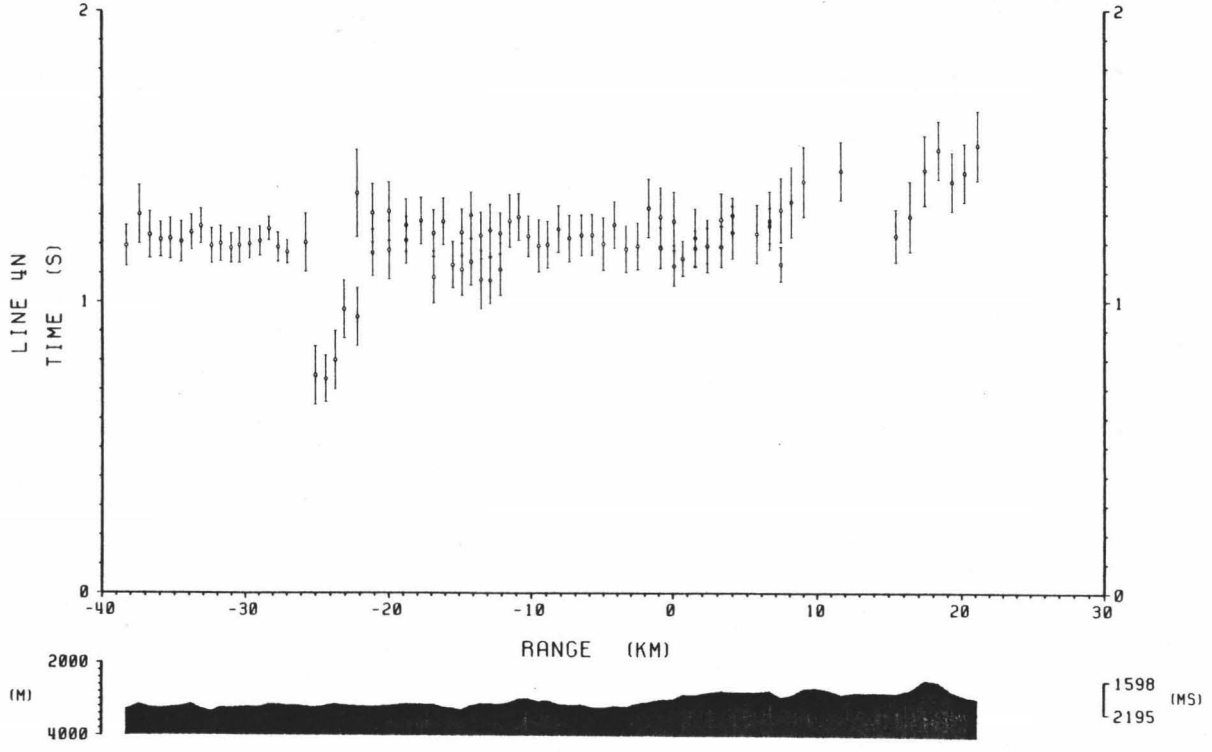
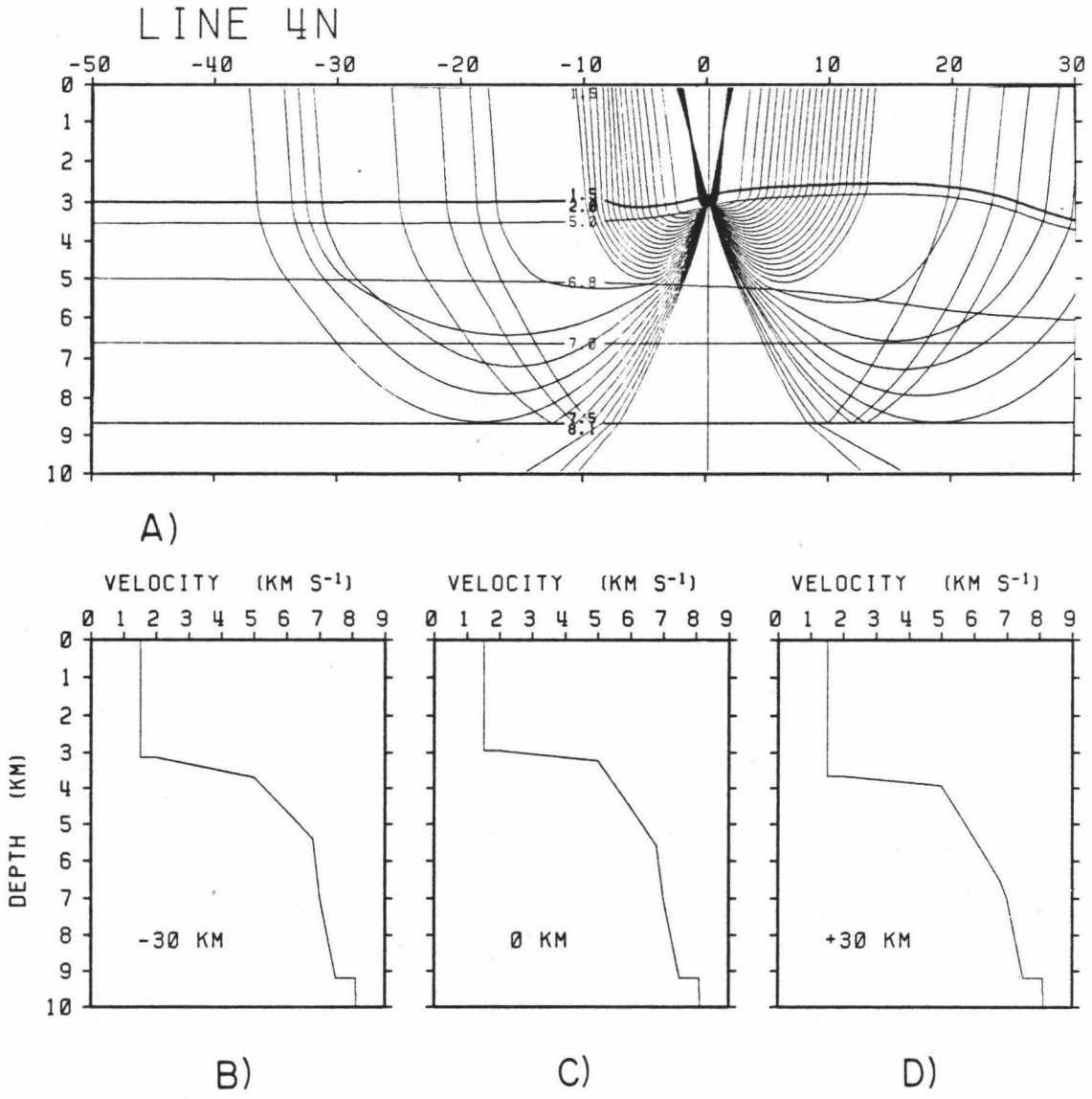

LINE 4N
TIME (S)
2
1
0
-40
-30
-20
-10
0
10
20
30
RANGE (KM)
2000
4000
(M)
1598
2195
(MS)

Figure 21. Velocity-depth model for the profile of line 4N. See comments for Figure 12.
a) Isovelocity contours for the model. Rays originate from the position of OBH 4.
b) Velocity-depth function sampled at 30 km west of OBH 4.
c) Velocity-depth function sampled directly beneath OBH 4.
d) Velocity-depth function sampled at 30 km east of OBH 4.


LINE 4N
-50 -40 -30 -20 -10 0 10 20 30
0 1 2 3 4 5 6 7 8 9 10
1.8
5.0
6.8
7.0
7.5
8.1
A)
VELOCITY (KM S-1)
0 1 2 3 4 5 6 7 8 9
DEPTH (KM)
-30 KM
0 KM
+30 KM
B)
C)
D)

CHAPTER VI

SUMMARY AND CONCLUSION

The models developed for the refraction profiles show substantial lateral variation of velocity structure within the Orozco Transform Fault region. However the dufferent types of structure are not easily correlated with the surface features described in chapter 2.

The models developed for the exterior of the fracture zone which was classified as rise flank match standard models for young oceanic crust at fast-spreading ridges [Orcutt et al., 1975; Lewis and Snydsman, 1978; Gettrust et al., 1982]. These seismic profiles are modeled succesfully by the velocity structure for the Samail (Oman) ophiolite [Manghnani et al., 1981; Kempner and Getrust, 1982]. It consists essentially of a 2 km thick upper layer with seismic velocity ranging from 4.0 to 6.8 km s^{-1} and a gradient of 0.7 s^{-1}, overlying a 3 to 4 km thickness with a maximum velocity of 7.8 km s^{-1} and a gradient of 0.1 s^{-1}, and an upper mantle velocity of 8.1 km s^{-1}. This type of structure is observed for the southern part of line 1N, all of line 3N, and the western part of line 4N.

There is not a great deal of variation in total crustal thickness. The depth to the mantle ranges between 5.5 and 6.5 km. Other observations on fracture zones indicate that significant thinning of the crust can occur in transform faults. Most of the crust in the transform valleys of the Orozco Fracture Zone is not well sampled by the

data available. The area just south of the ridge and west of the central trough is deep and is roughly an extension of the transform fault connecting the southern limb of the EPR so it has been classified as a transform valley. This region is right in the area of the explosion profiles so is thoroughly covered. There is no evidence of any crustal thinning. The time terms at the shots along lines 2N and 3N do not show any anomalous delays in the valley. Line 4N which runs along the valley yields a model not significantly different from those in other regions. This negative result may mean that the region was incorrectly classified and indeed has no history of transform faulting. It is however the site of a sharp transition between the crustal structure typical of the exterior and that of the central transform region.

The velocity structure in the crust beneath the ridges is very different from that elsewhere. Whereas normal oceanic crust can be differentiated by seismic velocity into upper and lower layers, the velocity in the upper crust of the ridges is depressed so that the velocity gradient is virtually constant over the entire thickness of the crust. This distribution of velocities is also found in the southern part of the central trough. Models for this type of crust are generated by the data at the northern ends of lines 1N and 2N as well as the eastern end of line 4N. The central ridge extends out beyond 105^{o}W and line 3N but that profile indicates normal exterior type crust. The anomalous crust is confined to the immediate vicinity of the active part of the transform fault. This result is in agreement with those of Trehu

[1982] and Trehu and Purdy [1983] who used a subset of the data available here. Detrick et al. [1983] observe a similar structure along the Vema Fracture Zone and its associated transform ridges. Both of these studies attribute the depressed surface velocities to intense fracturing and brecciation caused by tectonic activity. Because of the geographic distribution of the available data the present study does not indicate whether this type of crust is found along the other transform faults and ridges in the Orozco Fracture Zone.

The central trough itself is a feature that remains unexplained. At its northern end it connects two active transform faults. The trough is perpendicular to the spreading direction so must be a site of extension. Trehu [1982] and Trehu and Solomon [1983] suggest evidence for a low velocity body at the intersection of the central trough with the eastern part of the southern transform fault. The velocities measured there are within the range of a magma body, so there may be active volcanism occuring in the trough.

If crust is being generated, the thermal regime of the central trough is different than on the axes of the spreading ridges. The East Pacific Rise does not have a deep rift valley at its crest. One model for the rift valley is that the ascending magma loses energy by friction with the conduit and reaches equilibrium at a lower level. The energy is conserved by the emplaced material at the edge of the rift which lifts up [Sleep, 1969; Lachenbruch, 1973]. At fast spreading ridges the magma production is greater and comparatively less interaction with the conduit occurs resulting in a smaller loss of hydraulic head.

Within the transform region the production of magma is probably seriously reduced. The reason is that the transform fault offset introduces a third wall into the magma chamber and causes greater cooling [Sleep and Biehler, 1970]. Alternatively it is possible that the transform faults are a result and not a cause of decreased magma flow on the ridge [Collette et al., 1974; Schouten and White, 1980]. The reduced generation of crustal material at these sites is presumably part of the explanation for the thin crust observed in fracture zones [Detrick and Purdy, 1980; Sinton and Hussong, 1982; Louden and Sinha, 1983]. In the central trough of the Orozco Fracture Zone the transform ridge is being pulled apart by the relative motion of the Cocos and Pacific plates and there is insufficient generation of new crust to maintain the normal elevation of the rise crest. The present seismic data do not illuminate the deeper structure in this part of the trough.

Reduced magma production will also have an effect on the seismic velocity structure of the crust. The massive gabbros of the lower crust are postulated to be formed by solidification on the walls of the magma chamber at the base of the crust [Macdonald, 1982]. The upper crust is built of surface eruptions and their feeder dikes. A reduced magma supply will result in a diminished magma chamber and a correspondingly greater proportion of the crustal column being formed as upper crust. The seismic velocity structure will not be controlled by compositional variations as much as fracturing and fluid circulation. As a result it should be more homogeneous, which is what is observed in the interior of the transform fault region of the Orozco Fracture Zone.

In summary, the ROSE Phase 2 data for the Orozco Fracture Zone indicate some similarities and some differences with other fracture zones. The velocity structure in the central transform region is clearly different from oceanic crust. The velocities of the upper crust are depressed so that the velocity gradient is nearly constant over the 5 km thickness. There is no evidence of major thinning of the crustal layer. Typical mantle velocities are found at depths of about 6 km at all locations sampled.

The present analysis can be extended further. Only data from receivers in the area covered by the explosive profiles were analyzed. There are a great deal more receivers to be looked at. These studies will have to make use of more sophisticated three-dimensional analysis techniques. The only such tool available in this study was the time term method. The velocity structure is variable enough so that most of the assumptions of the method are invalid and only limited amounts of information could be gained.

APPENDIX A

COMPUTER PROGRAMS

The ROSE project generated tremendous amounts of data. One of the goals of this project was to develop a semi-automated interpretation system to aid the generation of velocity-depth models from the travel time data. Barring complete automation, the next best thing is to have an interactive system where the reduction and analysis of data can proceed with occasional direction by the human interpreter. The Hawaii Instiute of Geophysics has fairly extensive computing facilities. The main computer is a Harris H800 with floating point hardware and 500K 24 bit word virtual memory, operating under the VOS 1 operating system. Of primary importance to this project are a Hewlett-Packard HP2647A graphics terminal, a Versatec Model 1200 electrostatic plotter, and a 300 Mbyte disk devoted to ROSE project participants.

The data reduction and analysis described in chapter 5 can be summarized as follows:

1) if (HIG data)
   convert analog OBS tapes to digital form
   demultiplex and compress 4-channel digital data
   else if (ROSE archive and exchange facility data)
   retrieve digital data from archive
2) plot data in record section format (signal trace at distance from receiver.

3) if (shot or receiver relocation is necessary)

apply corrections

go to step 2.

4) pick the first arrival for each shot/receiver pair

5) correct travel times for water column and bathymetry

6) fit a differentiable curve to the data and reparametrize to $\zeta(p)$ and $\tau(p)$.

7) extrapolate seismic velocity to surface.

8) invert ζ and τ parameters to yield velocity-depth models for each receiver/shotline combination.

9) combine models for different receivers along the same shotline.

10) use ray-tracing to determine response for combined 2-dimensional models.

11) if (travel times and amplitudes do not fit the data)

go to step 9.

else

process is completed.

This thesis concerns steps 4 through 11. I have written programs for steps 4 through 8 and they are included in the appendix. The data analysis consists of several steps and decisions must be made along the path. A separate program handles each step. Data exchange between the programs must be done in a consistent manner so path changes can be made with a minimum of programming effort. The program listings are for TPIK

for first arrival picking; TXCOR to apply the corrections; TXFIT to smooth the travel time data and reparametrize; VEL0 to extrapolate to the surface; and LSVZ to calculate the velocity vs. depth models.

The programs are written in Harris Fortran 77 which is a superset of ANSI Fortran 77 [ANS X3.9, 1978]. The major incompatibility with ANSI Fortran is the use of block structures. These structures contribute much to the clarity of the programs and can easily be translated to standard Fortran. There are certain other operating system dependencies which are usually confined to lower level subroutines and noted in the code.

An interactive interpreting system must be easy to use. It must free the user from having to remember or look up values used internally by the programs. The most obvious example of this is in file assignments. Programs should make their own logical connections to data files, the user should have to supply only the file name. Another example is to recognize the end of the input. The user should not have to inform the program of how many data points there are in a file. The program can recognize the end. Additionally the program should ask a minimum of questions and print only data that is useful. Extra output gets in the way and discourages its examination. These guidelines have directed the style of most of the programs.

Along the way, I have collected and developed many general purpose subroutines and extensive use is made of them by the processing programs. Developing tools for computation makes the programming task easier and also imposes some discipline on the form of the programs [Kernighan and Plauger, 1979]. Listings are provided in the appendix for the

subroutines that are used. They can be divided into four categories: 1) data transmission; 2) text processing; 3) mathematical functions; and 4) plotting.

A.1 Program TPIK.

My requirement was a facility that would permit the entire length of each data buffer to be examined on a video display and that would allow any point on the trace to be specified with a cursor. Program TPIK is executed by typing:

TPIK datafile corfile [outfil] [/FSW]

where datafile is a standard HIG demultiplexed OBS file (created in step 2); corfile is a file containing certain auxiliary information about each shot/receiver pair (created in step 2); and outfile is the output containing the shot number, its range from the receiver, and the computed travel time for the arrival picks. Options are F to apply a zero-phase Butterworth filter to the data before displaying; S to save a specified amount of data before and after the pick in a separate file; and W to inform the program that water arrivals are to be picked.

The program prompts for a reduction velocity. If a positive velocity is entered, the trace will be displayed starting at time slightly in advance of the expected arrival, otherwise it will be displayed from the start of the data buffer. The program then enters a loop where it prompts for the shot number to be displayed. If a negative number is entered, the program stops. If a valid shot number is entered, the signal is buffered in from the data file, and the range is read from the corfile (procedure PROPIK). The starting sample is calculated by

$$samp = ((t_{shot\ break} - t_{data\ start}) + range / velocity) * samplerate$$

(procedure STSAMP). The reducing velocity should be chosen as the velocity of most of the phases to be picked. This will minimize the amount that the cursor must be moved between each shot.

The data is filtered if required (procedure FLTTM) and then displayed to the video screen of the HP2647A in blocks of 720 samples (procedure SHOPIK). The graphics cursor is turned on. This may be positioned with terminal keys and the following options are displayed:

A advance to next block.

B back up to previous block.

P pick a single arrival.

F pick the first arrival.

S pick the second arrival.

T pick the third arrival.

X exit plotting.

If the characters P, F, S or T is typed, the position of the cursor on the screen is determined and converted to travel time. The program prompts for an uncertainty for the pick given in number of samples. When this is entered the program writes out the shot number, range, times and uncertainties to outfile, goes to the top of the loop and prompts for a new shot.

Fortran does not have the facility for reading from the terminal before a carriage return is given, so the cursor keys must be read with

assembly language and the Harris VOS I/O service. The code that accomplishes this is in subroutine INHP.

```
1: *-h- PIKMAIN 3561  2 MAY 83  22:48:45
2:       PROGRAM TPIK
3:
4: ************************************************************************
5: *                                                                      *
6: *  Plot shot records from standard HIG demux files on HP-2647A         *
7: *   terminal so arrival times can be picked                            *
8: *                                                                      *
9: ************************************************************************
10: *                                                                      *
11: *   usage:  TIMPICK  datafile corfile outfile [/FSW]                   *
12: *                                                                      *
13: *     datafile: HIG demultiplexed OBS data from XDECOM or XRHIG        *
14: *     corfile:  time and range corrections from XDECOM or XRHIG        *
15: *     outfile:  shot numbers, ranges and time picks with error         *
16: *     options:  F  filter data before displaying                       *
17: *               S  save specified number of seconds before and after   *
18: *                   arrival in separate file                           *
19: *               W  pick water wave arrivals                            *
20: *                                                                      *
21: ************************************************************************
22: *                                                                      *
23: *  source: 1512REF*A.PIK                                               *
24: *  libraries: 1512ROSE*PLOTLB, 1512ROSE*CHARLB, 1512ROSE*MATHLB        *
25: *              1512REF*WLIB, 1500MGG*MRSLIB, *SAUL77, *LIBERY          *
26: *  executable: 1512REF*TIMPICK                                         *
27: *                                                                      *
28: ************************************************************************
29: *   1.0   2/6/82     no options implemented                           *
30: *   1.1   2/16/82    filter, save and water arrival options added     *
31: *   1.2   4/6/83     change to new i/o subroutines                    *
32: *                     introduce error recovery for bad data times     *
33: *   1.3   5/1/83     change filter, convert to data to float for      *
34: *                     processing                                       *
35: ************************************************************************
36:
37:       INTEGER DD, CD, OD, SD, HD, FOPT, SOPT, WOPT
38:       COMMON /FILES/ DD, CD, OD, SD, HD, FOPT, SOPT, WOPT
39:       INTEGER HBUF(112), SHOT, HDRADR, NSAMP
40:       REAL AFTSEC, FORSEC, RV
41:       INTEGER LOCDHD, PROPIK
42:
43:
44:       CALL OPFPIK(DD,CD,OD,SD,HD,FOPT,SOPT,WOPT)
45:       HDRADR = LOCDHD(DD,HBUF,NSAMP)
46:
47:       IF (SOPT .EQ. 1) THEN
48:       .  CALL REMARK(" enter number of seconds before and after arrival to
49:      &.  save")
50:       .  READ (0,) FORSEC, AFTSEC
51:       END IF
52:
53:       IF (FOPT .EQ. 1) THEN
54:       .  CALL REMARK(" enter low cut-off freq, high cut-off freq, and numbe
55:      &.  r of poles")
56:       .  READ (0,) F1, F2, NPOLES
57:       .  CALL GFLTTM(F1,F2,NPOLES,GAIN,DD,HDRADR)
58:       END IF
59:
60:       IF (WOPT .EQ. 1) THEN
61:       .  CALL REMARK(" enter water velocity")
62:       .  READ (0,) RV
63:       ELSE
64:       .  CALL REMARK(" enter reducing velocity  or 0")
65:       .  READ (0,) RV
66:       END IF
67:
68:       NSAMP = MIN0(NSAMP,10000)
69:       CALL INIPLT
70:       CALL ENDFIG(0)
71:
72:       DO
73:       .  CALL REMARK(" shot number?")
74:       .  READ (0,) SHOT
75:       UNTIL (PROPIK(SHOT,HDRADR,NSAMP,RV,FORSEC,AFTSEC,F1,F2,NPOLES,
76:      & GAIN) .NE.-2)
77:
78:       IF (SOPT .EQ. 1)
79:      &  CALL ENDDMX(SD,HD,MBUF,NSAMP)
80:       CALL ENDPLT
```

```
      CALL CLFPIK(DD,CD,OD)

      END


*-h- OPFPIK 1488 16 MAY 83   0:12:05
      SUBROUTINE OPFPIK(DD, CD, OD, SD, HD, FOPT, SOPT, WOPT)

* read command line, resolve options, open files

      INTEGER DD, CD, OD, SD, HD, FOPT, SOPT, WOPT
      INTEGER ARG(18), LIN(18), TNAME(8), MBUF(112), TABLIST(6)
      INTEGER FILCR, FILOP, GETARG, GETLIN, GETOPT
      DATA TNAME /72,68,82,84,69,77,80,-2/
      DATA TABLIST /6,16,34,52,70,0/

      CALL PRIMIO(11)
      IF (GETARG(1,ARG,17) .NE. -1) THEN
      .  DD = FILOP(ARG)
      .  IF (DD .EQ. -3) CALL CANT(ARG)
      ELSE
      .  CALL ERROR(" usage: TIMPICK datafile corfile outfile [/FSW]")
      END IF

      IF (GETARG(2,ARG,17) .NE. -1) THEN
      .  CD = FILOP(ARG)
      .  IF (CD .EQ. -3) CALL CANT(ARG)
      ELSE
      .  CALL ERROR(" usage: TIMPICK datafile corfile outfile [/FSW]")
      END IF

      IF (GETARG(3,ARG,17) .NE. -1) THEN
      .  OD = FILOP(ARG)
      .  IF (OD .EQ. -3) OD = FILCR(ARG)
      .  IF (OD .EQ. -3) CALL CANT(ARG)
      ELSE
      .  OD = 6
      END IF

      FOPT = GETOPT(70)
      SOPT = GETOPT(83)
      WOPT = GETOPT(87)

      IF (SOPT .EQ. 1)
      .  CALL REMARK(" enter compressed output file name")
      .  JUNK = GETLIN(LIN,0)
      .  I = 1
      .  CALL GETWRD(LIN,I,ARG)
      .  SD = FILOP(ARG)
      .  IF (SD .EQ. -3) CALL CANT(ARG)
      .  HD = FILCR(TNAME)
      .  IF (HD .EQ. -3) CALL CANT(TNAME)
      .  CALL INIDMX(SD,MBUF)
      END IF

*  advance to end of output file

      DO
      UNTIL (GETLIN(LIN,OD) .EQ. -1)

*  set tab stops for output

      CALL SETTAB(TABLIST)

      RETURN
      END


*-h- PROPIK 2735 26 MAY 83  23:21:36
      INTEGER FUNCTION PROPIK(SHOT, HDRADR, NSAMP, RV, FS, AS, F1, F2,
     &  NPOLES, GAIN)

* get time and range information for shot, display, and pick arrivals

      INTEGER SHOT, NSAMP, HDRADR, NPOLES
      REAL RV, FS, AS, F1, F2, GAIN
      INTEGER DD, CD, OD, SD, HD, FOPT, SOPT, WOPT
      COMMON /FILES/ DD, CD, OD, SD, HD, FOPT, SOPT, WOPT
      INTEGER DATABUF(20000), FILTBUF(20000)
      COMMON /DATA/ DATABUF, FILTBUF
```

```
13:       INTEGER CH,RCV,SMPRATE,ADR,SAMPLENO,ARRIV(3),UNC(3),INDEX(3)
14:       INTEGER*6 DST, SBT
15:       REAL RANGE, TIME(3), DELTIME(3), SIGNAL(10000)
16:       REAL RDEP, SDEP, SLANTRANGE
17:       CHARACTER INTERNAL*420                    ! for corfile decoding
18:       EQUIVALENCE (DATABUF,INTERNAL)
19:       EQUIVALENCE (FILTBUF,SIGNAL)
20:       INTEGER GETCOR, GETDHD, FILBUF, STSAMP
21:
22:       IF (SHOT .LT. 0) THEN
23:       .  CALL REMARK(" end processing")
24:       .  PROPIK = -3
25:       .  RETURN
26:       END IF
27:       IF (GETCOR(0,SHOT,DATABUF,CD) .EQ. -3) THEN
28:       .  CALL REMARK(" shot not found in corfile")
29:       .  PROPIK = -2
30:       .  RETURN
31:       ELSE
32:       .  READ (INTERNAL,"(I6,T71,F10.0,T111,2F10.0)")
33:      &.    RCV, SDEP, RANGE, RDEP
34:       END IF
35:       IF (GETDHD(DD,HDRADR,SHOT,RCV,CH,SBT,DST,SMPRATE,ADR) .EQ. -3)
36:       .  CALL REMARK(" shot not found in data file")
37:       .  PROPIK = -3
38:       .  RETURN
39:       END IF
40:       IF (FILBUF(DD,DATABUF,NSAMP,ADR) .EQ. -3) THEN
41:       .  CALL REMARK(" error during data transfer")
42:       .  PROPIK = -3
43:       END IF
44:
45:       IF (WOPT .EQ. 1) THEN
46:       .  SLANTRANGE = SQRT(RANGE**2+(RDEP-SDEP)**2)
47:       .  SAMPLENO = STSAMP(DST,SBT,-1.0,SLANTRANGE,RV,SMPRATE)
48:       ELSE
49:       .  SAMPLENO = STSAMP(DST,SBT,0.0,RANGE,RV,SMPRATE)
50:       END IF
51:       IF (SAMPLENO .EQ. '77777777) THEN
52:       .  CALL REMARK(" overflow in stsamp; bad start time in data file")
53:       .  PROPIK = -2
54:       .  RETURN
55:       ELSE
56:       .  SAMPLENO = MAX0(SAMPLENO,1)
57:       END IF
58:
59:       CALL VFLOAT(SIGNAL,1,DATABUF,1,NSAMP)
60:       IF (FOPT .EQ. 1) THEN
61:       .  CALL FLTTIM(SIGNAL,F1,F2,NPOLES,GAIN,NSAMP)
62:       .  CALL SHOPIK(SIGNAL,SAMPLENO,NSAMP,SHOT,SBT,DST,SMPRATE,ARRIV,UNC,
63:      &.    INDEX,F1,F2,NPOLES)
64:       ELSE
65:       .  CALL SHOPIK(SIGNAL,SAMPLENO,NSAMP,SHOT,SBT,DST,SMPRATE,ARRIV,UNC,
66:      &.    INDEX,F1,F2,NPOLES)
67:       END IF
68:
69:       FOR I = 1,3
70:       .  IF (INDEX(I) .EQ. 1) THEN
71:       .  .  TIME(I) = FLOAT(ARRIV(I))/FLOAT(SMPRATE) - FLOAT(SBT-DST)/1000.0
72:       .  .  DELTIME(I) = FLOAT(UNC(I)) / FLOAT(SMPRATE)
73:       .  ELSE
74:       .  .  TIME(I) = 0.0
75:       .  .  DELTIME(I) = 0.0
76:       .  END IF
77:       END FOR
78:
79:       IF (INDEX(1) .NE. 0 .OR. INDEX(2) .NE. 0 .OR. INDEX(3) .NE. 0)
80:      &  CALL WIPIK(SHOT,RANGE,TIME,DELTIME,INDEX,RCV,OD)
81:
82:       IF (SOPT .EQ. 1)
83:      &  CALL COMPRS(DATABUF,SHOT,RCV,CH,SBT,DST,SMPRATE,ARRIV,
84:      &  FS,AS,SD,HD)
85:
86:       PROPIK = -2
87:       RETURN
88:       END
```

```
*-h- SHOPIK 1989  2 MAY 83  16:21:25
      SUBROUTINE SHOPIK(SIGNAL, SN, N, SHOT, SB, DS, RATE, ARRIV, UNC,
     &  INDEX, F1, F2, NPOLES)

* plot data on HP screen and turn on cursor for pick

      INTEGER SN, N, SHOT, RATE, ARRIV(3), UNC(3), INDEX(3)
      INTEGER NPOLES
      INTEGER*6 SB, DS
      REAL SIGNAL(*), F1, F2
      INTEGER DD, CD, OD, SD, HD, FOPT, SOPT, WOPT
      COMMON /FILES/ DD, CD, OD, SD, HD, FOPT, SOPT, WOPT
      INTEGER KEY, NX, SMP, ST
      REAL SMAX, SMIN, SMEAN, SCALE, X, XD(20), XINC

      CALL SIGPAR(SIGNAL,N,SMAX,SMIN,SMEAN)
      SCALE = 2.0 / (SMAX - SMIN)
      XINC = 1.0 / 72.0
      FOR I = 1,3
      .  INDEX(I) = 0
      END FOR
      ST = SN

      LOOP
      .  CALL RESTORE(0)
      .
      .  X = 0.0
      .  NX = 1
      .  WHILE (X .LT. 9.99)
      .  .  XD(NX) = AMIN1(FLOAT(RATE)/72.0,10.0)
      .  .  X = X + XD(NX)
      .  .  NX = NX + 1
      .  END WHILE
      .  NX = NX - 1
      .  XD(NX) = XD(NX) - 9.99 + X
      .  CALL GRID(0.0,2.9,1000+NX,XD,1,2.0,-7)
      .
      .  CALL LINTYP(-1,1)
      .  CALL PVEC1(SIGNAL(ST),720,0.0,3.9,SMEAN,SCALE,XINC,0.0)
      .  CALL W2PIK1(SHOT,DS,SB,ST,RATE,SMEAN)
      .  IF (FOPT .EQ. 1) CALL W2PIK2(F1,F2,NPOLES)
      .
      .  LOOP
      .  .  CALL W2PIK3(WOPT)
      .  .  CALL PICK(ST,SMP,KEY)
      .  .
      .  .  IF (KEY .EQ. 65) THEN
      .  .  .  ST = ST + 700
      .  .  .  CALL ENDFIG(0)
      .  .  .  EXIT LOOP
      .  .  ELSE IF (KEY .EQ. 66) THEN
      .  .  .  ST = MAX0(ST-700,1)
      .  .  .  CALL ENDFIG(0)
      .  .  .  EXIT LOOP
      .  .  ELSE IF (KEY .EQ. 88) THEN
      .  .  .  CALL ENDFIG(0)
      .  .  .  RETURN
      .  .  ELSE IF (KEY .EQ. 80) THEN
      .  .  .  INDEX(1) = 1
      .  .  .  ARRIV(1) = SMP
      .  .  .  CALL W2PIK4
      .  .  .  READ (0,) UNC(1)
      .  .  .  CALL ENDFIG(0)
      .  .  .  RETURN
      .  .  ELSE
      .  .  .  IF (KEY .EQ. 68 .OR. KEY .EQ. 70) THEN
      .  .  .  .  I = 1
      .  .  .  ELSE IF (KEY .EQ. 49 .OR. KEY .EQ. 83) THEN
      .  .  .  .  I = 2
      .  .  .  ELSE IF (KEY .EQ. 50 .OR. KEY .EQ. 84) THEN
      .  .  .  .  I = 3
      .  .  .  ELSE
      .  .  .  .  I = 0
      .  .  .  END IF
      .  .  .  IF (I .NE. 0) THEN
      .  .  .  .  INDEX(I) = 1
      .  .  .  .  ARRIV(I) = SMP
      .  .  .  .  CALL W2PIK4
```

```
79:       . . . . READ (0,) UNC(I)
80:       . . . END IF
81:       . . END IF
82:       . END LOOP
83:       END LOOP
84:
85:       RETURN
86:       END
```

```
 1: *-h- W2PIK1 697 19 APR 83  14:17:21
 2:       SUBROUTINE W2PIK1(SHOT, DS, SB, ST, RATE, SMEAN)
 3:
 4: * display trace information
 5:
 6:       INTEGER SHOT, ST, RATE
 7:       INTEGER*6 DS, SB
 8:       REAL SMEAN
 9:
10: :HCONST
11:       CALL BOX(1.0,1.5,3.3,1.0,0.0)
12:       CALL GRSTR(1.1,2.2,"Shot",4,0.1,0.0)
13:       CALL GRNUM(2.2,2.2,FLOAT(SHOT),5,-1,0.1,0.0)
14:       CALL GRSTR(1.1,2.0,"Relative start time ",20,0.1,0.0)
15:       RELTIME = REAL(DS-SB) / 1000.0 + FLOAT(ST-1) / FLOAT(RATE)
16:       CALL GRNUM(3.2,2.0,RELTIME,8,3,0.1,0.0)
17:       CALL GRSTR(1.1,1.8,"Sample number",13,0.1,0.0)
18:       CALL GRNUM(3.2,1.8,FLOAT(ST),5,-1,0.1,0.0)
19:       CALL GRSTR(1.1,1.6,"Signal mean",11,0.1,0.0)
20:       CALL GRNUM(3.2,1.6,SMEAN,8,-1,0.1,0.0)
21: :NO HCONST
22:
23:       RETURN
24:       END
```

```
 1: *-h- W2PIK2 567 18 APR 83  14:21:38
 2:       SUBROUTINE W2PIK2(F1, F2, NPOLES)
 3:
 4: * display filter parameters
 5:
 6:       INTEGER NPOLES
 7:       REAL F1, F2
 8:       REAL FALLOFF
 9:
10:       FALLOFF = 6.0 * NPOLES
11: :HCONST
12:       CALL BOX(1.0,0.5,3.3,0.8,0.0)
13:       CALL GRSTR(1.1,1.0,"Low frequency",13,0.1,0.0)
14:       CALL GRNUM(3.0,1.0,F1,4,-1,0.1,0.0)
15:       CALL GRSTR(1.1,0.8,"High frequency",14,0.1,0.0)
16:       CALL GRNUM(3.0,0.8,F2,4,-1,0.1,0.0)
17:       CALL GRSTR(1.1,0.6,"Fall-off",8,0.1,0.0)
18:       CALL GRNUM(2.8,0.6,FALLOFF,4,-1,0.1,0.0)
19:       CALL GRSTR(3.1,0.6,"dB/oct",6,0.1,0.0)
20: :NO HCONST
21:
22:       RETURN
23:       END
```

```
 1: *-h- W2PIK3 932 12 APR 83   0:38:04
 2:       SUBROUTINE W2PIK3(WOPT)
 3:
 4: * display pick options
 5:
 6:       INTEGER WOPT
 7:
 8: :HCONST
 9:       CALL SETDRM(1)
10:       CALL FILREC(5.05,0.85,9.45,2.45,1)
11:       CALL SETDRM(4)
12:       CALL BOX(5.0,0.8,4.5,1.7,0.0)
13:       CALL GRSTR(5.1,2.3,"Options:",8,0.1,0.0)
14:       CALL GRSTR(5.3,2.1,"P -  single pick",16,0.1,0.0)
15:       IF (WOPT .EQ. 1)
16:       .  CALL GRSTR(5.3,1.9,"D -  direct wave",16,0.1,0.0)
17:       .  CALL GRSTR(5.3,1.7,"1 -  first reflection",21,0.1,0.0)
18:       .  CALL GRSTR(5.3,1.5,"2 -  second reflection",22,0.1,0.0)
19:       ELSE
```

```
20:       . CALL GRSTR(5.3,1.9,"F -  first arrival",18,0.1,0.0)
21:       . CALL GRSTR(5.3,1.7,"S -  second arrival",19,0.1,0.0)
22:       . CALL GRSTR(5.3,1.5,"T -  third arrival",18,0.1,0.0)
23:       END IF
24:       CALL GRSTR(5.3,1.3,"A -  advance to next block",26,0.1,0.0)
25:       CALL GRSTR(5.3,1.1,"B -  back to previous block",27,0.1,0.0)
26:       CALL GRSTR(5.3,0.9,"X -  exit plotting",18,0.1,0.0)
27: :NO HCONST
28:
29:       RETURN
30:       END
```

```
 1: *-h- W2PIK4 254 12 APR 83   0:38:05
 2:       SUBROUTINE W2PIK4
 3:
 4: * prompt for pick error estimate
 5:
 6: :HCONST
 7:       CALL SETDRM(1)
 8:       CALL FILREC(5.05,0.85,9.45,2.45,1)
 9:       CALL SETDRM(4)
10:       CALL GRSTR(5.3,2.0,"enter uncertainty in samples",28,0.1,0.0)
11: :NOT HCONST
12:
13:       RETURN
14:       END
```

```
 1: *-h- PICK 271 12 APR 83   0:38:05
 2:       SUBROUTINE PICK(ST, SP, KEY)
 3:
 4: * pick arrival time by reading cursor position
 5:
 6:       INTEGER ST, SP, KEY
 7:       REAL X, Y
 8:
 9:       CALL GRCURS(1)
10:       CALL READGC(X,Y,KEY)
11:       CALL GRCURS(0)
12:       SP = ST + INT(X*72.0)
13:       CALL ZOOM(X,Y,1)
14:
15:       RETURN
16:       END
```

```
 1: *-h- CLFPIK 186 12 APR 83   0:38:06
 2:       SUBROUTINE CLFPIK(DD, CD, OD)
 3:
 4: * close files
 5:
 6:       INTEGER DD, CD, OD
 7:
 8:       CALL FILCL(DD,"KEEP")
 9:       CALL FILCL(CD,"KEEP")
10:       CALL FILCL(OD,"KEEP")
11:
12:       RETURN
13:       END
```

A.2 Program TXCOR.

After the arrivals are picked by TPIK, the data must be formatted for use by subsequent programs. Program TXCOR is invoked by typing:

```
TXCOR pickfile corfile outfile [/ANW]
```

where pickfile is the output file from TPIK; corfile contains the auxiliary data for each shot/receiver pair; and outfile is the output from TXCOR. Options are A to process all shots in pickfile; N to supress sorting by range; and W to calculate delays with respect to theoretical waterwave.

If option A is not specified, TXCOR prompts for the start shot and the end shot to read from pick file. If the subsequent operation is to be curve fitting and reparametrization, then only data from one side of the receiver must be used and it must be sorted by range. The program then asks what type of correction is desired. The user may specify any combination of shot topographic correction, receiver topographic correction, and move to common datum or move to sea surface. Inversion using Dorman and Jacobson's [1981] method requires the first three corrections. A range error of 0.5 km is assigned to each shot and the shot number, range, range error, time, and time error are output to outfile.

```
1: *-h- CORMAIN 4095  4 JUN 83  23:16:10
2:       PROGRAM TXCOR
3:
4: ************************************************************************
5: *                                                                      *
6: * Format arrival time information from picker for input to curve       *
7: *   fitters                                                            *
8: *                                                                      *
9: ************************************************************************
10: *                                                                     *
11: *    usage:  TXCOR[.ANW] pickfile corfile outfile [/ANW]              *
12: *                                                                     *
13: *            pickfile:  arrival times, output from TIMPICK            *
14: *            corfile:   shot, receiver corrections file               *
15: *            outfile:   output file, corrected times                  *
16: *            options:   A  process all shots                          *
17: *                       N  no sorting by range                        *
18: *                       W  calculate delay in water wave  (wv=1.51)   *
19: *                            for input to XRMOD                       *
20: *                                                                     *
21: ***********************************************************************
22:
23:       INTEGER SHOT(200), TAG(200)
24:       REAL X(200), DX, T(200,3), DT(200,3)
25:       INTEGER ID, CD, OD, AOPT, NOPT, WOPT
26:       INTEGER BEG, FIN, JOB, N, OPT, LDD, PREVSHOT
27:       REAL SDEP, RDEP, BTC
28:       REAL STOPO, STOST, STOSX, RTOPO, RTOST, RTOSX, SRTODT, SRTODX
29:       LOGICAL SBC, RBC, MTD, MTS
30:       CHARACTER*420 INTERNAL              ! for decoding corfile
31:       EQUIVALENCE (TAG,INTERNAL)          ! use tag i/o buffer
32:       INTEGER RPIK, GETCOR
33:       DATA LDD /200/
34:
35:       CALL OPFCOR(ID, CD, OD, AOPT, NOPT, WOPT)
36:
37:       JOB = 11000
38:       IF (WOPT .EQ. 1) JOB = 11110
39:       IF (AOPT .EQ. 1)
40:       .  BEG = 0
41:       .  FIN = 9999
42:       ELSE
43:       .  CALL REMARK(" enter begin shot, end shot")
44:       .  READ (0,) BEG, FIN
45:       END IF
46:
47:       N = RPIK(SHOT,X,T,DT,LDD,JUNK,BEG,FIN,JOB,200,ID)
48:
49: *  phase 1.  Apply range and time corrections
50:
51:       IF (WOPT .EQ. 0)
52:       .  CALL REMARK(" shot topo, rcvr topo, move to datum, move to surf?
53:      &.  (4 digit binary)")
54:       .  READ (0,) OPT
55:       .  SBC = OPT / 1000 .GT. 0
56:       .  RBC = MOD(OPT/100,10) .GT. 0
57:       .  MTD = MOD(OPT/10,10) .GT. 0
58:       .  MTS = MOD(OPT,10) .GT. 0
59:       .  IF (MTD .AND. MTS) CALL ERROR(" cannot move to datum and to surfac
60:      &.  e at the same time")
61:       END IF
62:
63:       PREVSHOT = 0
64:       FOR I = 1,N
65:       .  IF (SHOT(I) .EQ. PREVSHOT)
66:       .  .  CALL PUTDEC(SHOT(I),5,3)
67:       .  .  CALL REMARK(" duplicate shot")
68:       .  END IF
69:       .  IF (GETCOR(0,SHOT(I),TAG,CD) .NE. -2)
70:       .  .  CALL PUTDEC(SHOT(I),5,3)
71:       .  .  CALL ERROR(" : can't find shot in corfile")
72:       .  ELSE
73:       .  .  READ (INTERNAL,"(T71,F10.0,T101,F10.0,T131,7F10.0,T371,2F10.0)")
74:      &.  .    SDEP, RDEP, BTC, STOPO, STOST, STOSX, RTOPO, RTOST, RTOSX,
75:      &.  .    SRTODT, SRTODX
76:       .  END IF
77:       .
78:       .  IF (WOPT .EQ. 1)
79:       .
```

```
 80: *  calculate delay in water wave arrival with respect to time
 81: *   reduce by slant range over water velocity
 82:        .
 83:        .  .  SX = SQRT(X(I)**2+(RDEP-SDEP)**2)
 84:        .  .  T(I,1) = T(I,1) - SX / 1.51
 85:        .  .
 86:        .  ELSE
 87:        .
 88: *  apply corrections
 89:        .
 90:        .  .  XCOR = 0.0
 91:        .  .  TCOR = BTC
 92:        .  .  IF (SBC) TCOR = TCOR + STOPO
 93:        .  .  IF (RBC) TCOR = TCOR+ RTOPO
 94:        .  .  IF (MTD)
 95:        .  .  .  TCOR = TCOR + SRTODT
 96:        .  .  .  XCOR = XCOR + SRTODX
 97:        .  .  END IF
 98:        .  .  IF (MTS)
 99:        .  .  .  TCOR = TCOR + STOST
100:        .  .  .  XCOR = XCOR + STOSX
101:        .  .  END IF
102:        .  .
103:        .  .  X(I) = X(I) + XCOR
104:        .  .  DX = 0.5
105:        .  .  T(I,1) = T(I,1) + TCOR
106:        .  .
107:        .  END IF
108:        .  PREVSHOT = SHOT(I)
109:        END FOR
110:
111: *  phase 2.  Sort by range
112:
113:        IF (NOPT .EQ. 0)
114:        .  CALL VINDEX(TAG.1.1.1.1.0.N)
115:        .  CALL SRTRT(X.TAG.N)
116:        .  CALL SRTPI(SHOT.TAG.0.N)
117:        .  CALL SRTPR(T(1,1).TAG.0.N)
118:        .  CALL SRTPR(DT(1,1).TAG.0.N)
119:        END IF
120:
121: *  output
122:
123:        FOR I = 1.N
124:        .  IF (WOPT .EQ. 1)
125:        .  .  CALL W2COR(SHOT(I).T(I,1).LDD.3.OD)
126:        .  ELSE
127:        .  .  CALL W1COR(SHOT(I).X(I).DX.T(I,1).DT(I,1).OD)
128:        .  END IF
129:        END FOR
130:
131:        CALL CLFCOR(ID.CD.OD)
132:
133:        END
```

```
 1: *-h- OPFCOR 925  4 MAY 83  23:49:10
 2:        SUBROUTINE OPFCOR(ID. CD. OD. AOPT. NOPT. WOPT)
 3:
 4: * open files
 5:
 6:        INTEGER ID. CD. OD. AOPT. NOPT. WOPT
 7:        INTEGER NAME(18)
 8:        INTEGER FILCR. FILOP. GETARG. GETOPT
 9:
10:        CALL PRIMIO(11)
11:        IF (GETARG(1.NAME.17) .EQ. -1)
12:        .  CALL ERROR(" usage: TXCOR  pickfile corfile outfile [/ANW]")
13:        ELSE
14:        .  ID = FILOP(NAME)
15:        .  IF (ID .EQ. -3) CALL CANT(NAME)
16:        END IF
17:
18:        IF (GETARG(2.NAME.17) .EQ. -1)
19:        .  CALL ERROR(" usage: TXCOR  pickfile corfile outfile [/ANW]")
20:        ELSE
21:        .  CD = FILOP(NAME)
22:        .  IF (CD .EQ. -3) CALL CANT(NAME)
23:        END IF
24:
```

```
25:       IF (GETARG(3.NAME.17) .EO. -1)
26:       .  CALL ERROR(" usage: TXCOR  pickfile corfile outfile [/ANW]")
27:       ELSE
28:       .  OD = FILOP(NAME)
29:       .  IF (OD .EO. -3)
30:       .  .  OD = FILCR(NAME)
31:       .  .  IF (OD .EO. -3) CALL CANT(NAME)
32:       .  END IF
33:       END IF
34:
35:       AOPT = GETOPT(65)
36:       NOPT = GETOPT(78)
37:       WOPT = GETOPT(87)
38:
39:       RETURN
40:       END
```

```
 1: *-h- CLFCOR 186  4 MAY 83  23:49:11
 2:       SUBROUTINE CLFCOR(ID. CD. OD)
 3:
 4: * close files
 5:
 6:       INTEGER ID. CD. OD
 7:
 8:       CALL FILCL(ID."KEEP")
 9:       CALL FILCL(CD."KEEP")
10:       CALL FILCL(OD."KEEP")
11:
12:       RETURN
13:       END
```

A.3 Program TXFIT.

Some of the processes and problems involved with fitting curves to travel time data are covered in chapter 5. A more detailed explanation of the algorithm follows. Program TXFIT may only be run on the HP2647A terminal and is invoked by typing:

```
TXFIT infile outfile listfile [/O]
```

where infile is the output from TXCOR; outfile is the reparametrized data to be used by LSVZ; and listfile is a listing of the spline fitting parameters and the fitted data. Option O specifies that the parameters will only be calculated at the knot positions.

The program queries for a reducing velocity which should be chosen to emphasise the greatest curvature. The reduced travel times for the data points are plotted to the screen (procedure PXT) and the program queries: "enter knot locations". Up to 20 points may be specified along the range axis (procedure GETKNT). As explained in chapter 5, several points are required where the curvature needs to be high, and very few where the the travel time function is close to straight. It has been my experience that 5 to 6 points suffice out to a distance of 25 km. Knots must be specified at range 0 and at a range greater than any observation. The spline is calculated with routine ICSFKU,and evaluated wit routine ICSEVU [IMSL, 1982].

If the user is satisfied with the fit, a carriage return is entered, otherwise a new set of knots is entered. If the fit is acceptable, the program proceeds to reparametrize the data. First routine DCSEVU is used to calculate the first and second derivatives at the observation points, or the knot locations if option 0 is used. The first derivative gives the ray parameter p and this is now used as the independent variable. The dependent variables are given by $\tau(p) = T - pX$ and $\zeta(p) = T + pX$ (procedute REPARM). Error bounds are also calculated using the formulae given in Dorman and Jacobson [1981] (procedure ERRBND).

The parameter p, $\zeta(p)$, $\tau(p)$, σ_ζ and σ_τ are written to outfile. Three blank lines are generated at the beginning of the file. The user must edit this file and insert the surface velocity determined by VEL0 and two titles for use by LSVZ. The knot locations and spline coefficients are output to listfile, followed by a list of the original data, the fitted data, the ray parameter and optimal interval for each observaion. This file is used by program VEL0 and others not included here.

```
*-h- FITMAIN 2801 12 MAY 83  23:18:17
      PROGRAM TXFIT

C*********************************************************************
C*                                                                   *
C*  Reparametrization of travel time data by cubic splines           *
C*                                                                   *
C*********************************************************************
C*                                                                   *
C*      usage:  TXFIT  infile outfile [listfile] [/O]                *
C*                                                                   *
C*              infile:   times, ranges output from TXCOR            *
C*              outfile:  tau, zeta parameters input to LSVZ         *
C*              listfile: list of fit parameters, input to TXPLOT, VELO *
C*              options:  O - output parameters at optimal points    *
C*                                                                   *
C*********************************************************************
C*                                                                   *
C*     source:     1512REF*A.FIT                                         *
C*     libraries:  1512REF*REFRLB,                                   *
C*                 1512ROSE*CHARLB, 1512ROSE*MATHLB,                 *
C*                 1512ROSE*PLOTLB, 1512ROSE*HPPLB,                  *
C*                 *IMSL, *LIBERY                                    *
C*     executable: 1512REF*TXFIT                                     *
C*                                                                   *
C*********************************************************************

      INTEGER SHOT(100)
      REAL X(100), DX(100), T(100), DT(100)
      REAL K(100), Y(100), C(20,3), S(100), P(100)
      REAL WORK(2600), DP(100), OX(100), DZ(100)
      COMMON /CDATA/ X, T, DT, S, P, Q, DP, OX, DZ, WORK
      REAL E
      INTEGER LDC, MAXK, N, NK, NP
      REAL RV, XSCALE, XMIN, YSCALE, YMIN
      INTEGER ID, OD, LD, OOPT
      INTEGER GETKNT, R1COR
      DATA LDC /20/, MAXK /20/

      CALL OPFFIT(ID,OD,LD,OOPT)
      N = R1COR(SHOT,X,DX,T,DT,200,ID)

      CALL REMARK(" enter reducing velocity")
      READ (0,) RV
      CALL INIPLT
      CALL MOVEOO(1.0,0.5)
      CALL PXT(X,T,RV,N,XMIN,XSCALE,TMIN,TSCALE)

      LOOP
      .  NP = GETKNT(K, MAXK)
      .  EXIT LOOP IF (NP .EQ. 0)
      .  NK = NP
      .  CALL FIT(X,T,N,K,NK,Y,C,LDC,E,S,WORK)
      .  CALL RESTOR(1)
      .  CALL PXTSM(X,S,RV,N,XMIN,XSCALE,TMIN,TSCALE)
      .  CALL W2FIT1(K,Y,C,LDC,E,NK,LD)
      END LOOP

      CALL REPARM(X,S,N,K,Y,C,LDC,NK,OX,P,DP,DZ,DT,OOPT)
      CALL W1FIT(P,K,Y,DZ,DT,N,OD)
      IF (OOPT .EQ. 0) CALL W2FIT2(SHOT,X,T,S,P,OX,N,LD)

      CALL ENDPLT
      CALL CLFFIT(ID,OD,LD)

      END


*-h- OPFFIT 894  4 MAY 83  23:57:38
      SUBROUTINE OPFFIT(ID, OD, LD, OOPT)

* read command line and open files for TXFIT

      INTEGER ID, OD, LD, OOPT
      INTEGER FILE(18)
      INTEGER FILCR, FILOP, GETARG, GETOPT

      WRITE (3,)
      CALL PRIMIO(11)
      IF (GETARG(1,FILE,18) .NE. -1)
```

```
13:       .  ID = FILOP(FILE)
14:       .  IF (ID .EQ. -3) CALL CANT(FILE)
15:       ELSE
16:       .  CALL ERROR(" usage:  TXFIT infile outfile [listfile]")
17:       END IF
18:
19:       IF (GETARG(2,FILE,18) .NE. -1)
20:       .  OD = FILOP(FILE)
21:       .  IF (OD .EQ. -3)
22:       .  .  OD = FILCR(FILE)
23:       .  .  IF (OD .EQ. -3) CALL CANT(FILE)
24:       .  END IF
25:       ELSE
26:       .  CALL ERROR(" usage:  TXFIT infile outfile [listfile]")
27:       END IF
28:
29:       IF (GETARG(3,FILE,18) .NE. -1)
30:       .  LD = FILOP(FILE)
31:       .  IF (LD .EQ. -3)
32:       .  .  LD = FILCR(FILE)
33:       .  .  IF (LD .EQ. -3) CALL CANT(FILE)
34:       .  END IF
35:       ELSE
36:       .  LD = 6
37:       END IF
38:
39:       OOPT = GETOPT(79)
40:
41:       RETURN
42:       END
```

```
 1: *-h- GETKNT 645  4 MAY 83  23:57:39
 2:       INTEGER FUNCTION GETKNT(K, MAXK)
 3:
 4: *  query for knot positions
 5:
 6:       INTEGER MAXK
 7:       REAL K(1)
 8:       INTEGER I, J, L, LIN(81), N, WRD(10)
 9:       INTEGER GETLIN, GETWRD
10:       REAL CTOF
11:
12: :HCONST
13:       CALL SETDRM(4)
14:       CALL GRSTR(0.4,0.15,"                                                ",
15:      &  43,0.2,0.0)
16:       CALL GRSTR(0.4,0.3,"Enter knot positions",20,0.2,0.0)
17:
18:       L = GETLIN(LIN,0)
19:       I = 1
20:       N = 0
21:       WHILE (GETWRD(LIN,I,WRD) .NE. 0 .AND. N .LT. MAXK)
22:       .  N = N + 1
23:       .  J = 1
24:       .  K(N) = CTOF(WRD,J)
25:       END WHILE
26:
27:       GETKNT = N
28:       CALL SETDRM(2)
29:       CALL ENDFIG(0)
30: :NO HCONST
31:
32:       RETURN
33:       END
```

```
 1: *-h- FIT 603  4 MAY 83  23:57:40
 2:       SUBROUTINE FIT(X, F, NX, K, NK, Y, C, LDC, RHO, S, WK)
 3:
 4: *  fit cubic spline to data points using fixed knots
 5: *     (calls IMSL routines ICSFKU, ICSEVU)
 6:
 7:       INTEGER NX, NK, LDC
 8:       REAL X(1), F(1), K(1), Y(1), C(LDC,1), S(1), WK(1), RHO
 9:       REAL D
10:       INTEGER IMSLERR
11:
12:       CALL ICSFKU(X,F,NX,0,K,NK,Y,C,LDC,RHO,WK,IMSLERR)
13:       CALL ICSEVU(K,Y,NK,C,LDC,X,S,NX,IMSLERR)
```

```
14:
15: *  last point must be calculated explicitly
16:
17:       CALL ICSEVU(K,Y,NK,C,LDC,K(NK),Y(NK),1,IMSLERR)
18:       D = X(NX) - K(NK-1)
19:       S(NX) = ((C(NK-1,3)*D + C(NK-1,2))*D + C(NK-1,1))*D + Y(NK-1)
20:
21:       RETURN
22:       END
```

```
 1: *-h- REPARM 1046 19 MAY 83  22:32:43
 2:       SUBROUTINE REPARM(X, T, NX, K, Y, C, LDC, NK, OX, P, DP, DZ, DT,
 3:      &  OOPT)
 4:
 5: *  reparametrize T(X) data to zeta(P) and tau(P), output
 6: *    oopt = 0   all data points
 7: *    oopt = 1   data points separated by optimal distances
 8: *   calls IMSL routine DCSEVU
 9:
10:       INTEGER NX, LDC, NK, OOPT
11:       REAL X(1), T(1), K(1), Y(1), C(LDC,1), OX(1), P(1)
12:       REAL DP(1), DZ(1), DT(1)
13:       INTEGER I, IMSLERR, N
14:       REAL D
15:
16:       IF (OOPT .EQ. 1)
17:       .  N = NK
18:       .  CALL VCOPY(X(2),1,K(2),1,N)
19:       .  CALL VCOPY(T(2),1,Y(2),1,N)
20:       ELSE
21:       .  N = NX
22:       END IF
23:
24: *  calculate first and second derivatives and then error bounds
25:
26:       CALL DCSEVU(K,Y,NK,C,LDC,X,P,N,OX,N,IMSLERR)
27:       D = X(N) - K(NK-1)
28:       P(N) = (3.0 * C(NK-1,3) * D + 2.0 * C(NK-1,2)) * D + C(NK-1,1)
29:       OX(N) = 6.0 * C(NK-1,3) * D + 2.0 * C(NK-1,2)
30:       CALL ERRBND(DP,OX,DZ,DT,P,OX,NX,N)
31:
32: *  tau and zeta are stored in K and Y
33:
34:       FOR I = 1,N
35:       .  K(I) = T(I) + P(I) * X(I)
36:       .  Y(I) = T(I) - P(I) * X(I)
37:       END FOR
38:
39:       NX = N
40:
41:       RETURN
42:       END
```

```
 1: *-h- ERRBND 500  4 MAY 83  23:57:43
 2:       SUBROUTINE ERRBND(SP, SX, SZ, ST, P, Q, ND, N)
 3:
 4: *  optimal interval between data points and 2-sigma bounds on parameters
 5: *   given that interval.  see Dorman & Jacobson (1981) Geophysics
 6:
 7:       INTEGER ND, N
 8:       REAL SP(1), SX(1), SZ(1), ST(1), P(1), Q(1)
 9:       REAL XN
10:
11:       XN = SQRT(1.0/FLOAT(ND))
12:       FOR I = 1,N
13:       .  Q(I) = 0.5 * ABS(Q(I))
14:       .  SP(I) = SQRT(ST(I)*Q(I)*XN)
15:       .  SX(I) = SQRT(2.0*ST(I)*(1.0/Q(I))*XN)
16:       .  SZ(I) = P(I) * SX(I)
17:       END FOR
18:
19:       RETURN
20:       END
```

```
 1: *-h- CLFFIT 195  4 MAY 83  23:57:44
 2:       SUBROUTINE CLFFIT(ID, OD, LD)
 3:
 4: * close file for TXFIT
 5:
 6:       INTEGER ID, OD, LD
 7:
 8:       CALL FILCL(ID,"KEEP")
 9:       CALL FILCL(OD,"KEEP")
10:       CALL FILCL(LD,"KEEP")
11:
12:       RETURN
13:       END
```

A.4 Program VEL0.

The inversion of the $\zeta(p)$, $\tau(p)$ parameters additionally requires a value for the seismic velocity at the surface (the ocean bottom in this case). For typical OBS experiments where the sources are on the ocean surface, the ray geometry means that the data contain no fundamental information about the uppermost crust [Ewing and Purdy, 1982]. By assuming that the top layer has a constant gradient, it is possible to extrapolate the observed velocities to the surface. VEL0 is an interactive program to help estimate the surface velocity. It is invoked by typing:

VEL0 infile water-depth

where infile is the list output from TXFIT; and water-depth is the depth of the receiver in km.

The program plots the first arrival times and smoothed fit to a range of 10 km on the HP2647A video screen. The time of propagation through the water column has to be added to these values because they were removed by TXCOR prior to smoothing (procedute ADDWT). The program plots the hyperbola representing the direct water wave to the receiver (procedure HYPPLT) and places the graphics cursor at the hyperbola's intercept with the time axis. The HP2647A has a feature called the "rubber band line" where a drawn line follows the motion of the cursor to show the appearance of a line before it is actually executed by the

computer. The interpreter positions the cursor somewhere on the smooothed travel time curve (usually near the third or fourth point) and hits any key. The program connects the original cursor position with the new point. If this line is not tangent to the direct wave hyperbola then there is an indication that there is a strong velocity gradient near the surface. Next the user moves the cursor to where he or she estimates the travel time curve intercepts the hyperbola and again hits any key. The program prompts the user to enter an estimate for the gradient. Using the formulae given in Ewing and Purdy [1982] the true travel time for that gradient is calculated and displayed by an arrow on the screen. If the arrow is below the intercept or the travel time curve and the hyperbola, then the gradient needs to be increased and vice versa. When the interpreter is satisified with the estimate for the gradient, a carriage return is entered. and the program prints the estimated gradient, surface velocity, and depth to turning point for the range where the travel time curve intercepts the direct wave (procedure ESTIM). The user must edit the output file from TXFIT and enter this surface velocity value in the first line before running LSVZ.

```
 1: *-h- VELMAIN 1308 19 MAY 83  23:55:42
 2:       PROGRAM VEL0
 3:
 4: ************************************************************************
 5: *                                                                      *
 6: *  Extrapolate to surface velocity from slope and intercept of nearest *
 7: *   crustal arrivals                                                   *
 8: *    see:  Ewing, J.I. and G.M. Purdy (1982) Jour. Geophys. Res.       *
 9: *                                                                      *
10: ************************************************************************
11:
12:       INTEGER PD, ND
13:       REAL X(100), T(100), S(100), OX(100)
14:       REAL RV, XMIN, XSCALE, YMIN, YSCALE
15:       REAL MAXRANGE, WATERDEPTH
16:
17:       CALL OPFVEL(PD,WATERDEPTH)
18:
19:       ND = R2FIT2(X,T,S,OX,100,PD)
20:       N = 1
21:       WHILE (X(N) .LT. 10.0 .AND. N .LT. ND)
22:       .  N = N + 1
23:       END WHILE
24:       CALL ADDWT(T,S,WATERDEPTH,N)
25:
26:       CALL INIPLT
27:       CALL PXT(X,T,0.0,N,XSCALE,XMIN,YSCALE,YMIN)
28:       CALL PXTSM(X,S,0.0,N,XSCALE,XMIN,YSCALE,YMIN)
29:
30:       CALL HYPPLT(WATERDEPTH,XSCALE,XMIN,YSCALE,YMIN)
31:
32:       CALL ESTIM(WATERDEPTH,XSCALE,XMIN,YSCALE,YMIN)
33:
34:       CALL CLFVEL(PD)
35:
36:       END
```

```
 1: *-h- OPFVEL 542  5 MAY 83   0:06:42
 2:       SUBROUTINE OPFVEL(PD, HW)
 3:
 4: * get command line arguments and open file
 5:
 6:       INTEGER PD
 7:       REAL HW
 8:       INTEGER NAME(18)
 9:       INTEGER FILOP, GETARG
10:
11:       CALL PRIMIO(11)
12:       IF (GETARG(1,NAME,18) .EQ. -1)
13:       .  CALL ERROR(" usage:  SURFVEL datafile water_depth")
14:       ELSE
15:       .  PD = FILOP(NAME)
16:       .  IF (PD .EQ. -3) CALL CANT(NAME)
17:       END IF
18:
19:       IF (GETARG(2,NAME,18) .EQ. -1)
20:       .  CALL ERROR(" usage:  SURFVEL datafile water_depth")
21:       ELSE
22:       .  I = 1
23:       .  HW = CTOF(NAME,I)
24:       END IF
25:
26:       RETURN
27:       END
```

```
 1: *-h- ADDWT 226  5 MAY 83   0:06:44
 2:       SUBROUTINE ADDWT(T, S, HW, N)
 3:
 4: * add water wave travel time to arrivals
 5:
 6:       INTEGER N
 7:       REAL T(*), S(*), HW
 8:
 9:       CALL VSMA1(T,1,1.0,T,1,HW/1.51,N)
10:       CALL VSMA1(S,1,1.0,S,1,HW/1.51,N)
11:
12:       RETURN
13:       END
```

```
 1: *-h- HYPPLT 498  5 MAY 83   0:06:45
 2:       SUBROUTINE HYPPLT(HW, XSCALE, XMIN, YSCALE, YMIN)
 3:
 4: * plot direct water wave hyperbola
 5:
 6:       REAL HW, XSCALE, XMIN, YSCALE, YMIN
 7:       INTEGER I
 8:       REAL X, T, XX, YY
 9:
10:       CALL MOVEPN(0.0,(HW/1.51-YMIN)*YSCALE,3,0)
11:
12:       FOR I = 1,100
13:       .  X = (9.0 / XSCALE) * FLOAT(I) / 100.0
14:       .  T = SQRT(X**2+HW**2) / 1.51
15:       .  EXIT FOR IF (T .GT. (YMIN+4.0/YSCALE))
16:       .  XX = (X - XMIN) * XSCALE
17:       .  YY = (T - YMIN) * YSCALE
18:       .  CALL MOVEPN(XX,YY,3,1)
19:       END FOR
20:
21:       RETURN
22:       END
```

```
 1: *-h- ESTIM 1735  5 MAY 83   0:06:46
 2:       SUBROUTINE ESTIM(HW, XSCALE, XMIN, YSCALE, YMIN)
 3:
 4: * estimate slope of closest arrivals and calculate surface velocity
 5:
 6:       REAL HW, XSCALE, XMIN, YSCALE, YMIN
 7:       REAL X0, T0, X1, T1, X, T, K, V0, VM, THETA, XX, YY, Z, U
 8:
 9:       ASINH(U) = ALOG(U + SQRT(U**2+1.0))
10:
11: *  locate intersection of first refracted arrival with direct wave
12:
13:       CALL SET00(1)
14:       CALL MOVEPN(1.0,0.5+(HW/1.51-YMIN)*YSCALE,1,0)
15:       CALL GRCURS(1)
16:       CALL MOVEGC(1.0,0.5+(HW/1.51-YMIN)*YSCALE,1)
17:       CALL LINTYP(-6,1)
18:       CALL RBLIN(1)
19:       CALL READGC(XX,YY,KEY)
20:       X1 = (XX + XMIN - 1.0) / XSCALE
21:       T1 = (YY + YMIN - 0.5) / YSCALE
22:       CALL MOVEPN(XX,YY,1,1)
23:       CALL LINTYP(-1,1)
24:
25: *  turn on rubber band line to estimate slope
26:
27:       CALL RBLIN(1)
28:       CALL READGC(XX,YY,KEY)
29:       X0 = (XX + XMIN - 1.0) / XSCALE
30:       T0 = (YY + YMIN - 0.5) / YSCALE
31:       VM = (X1 - X0) / (T1 - T0)
32:       THETA = ASIN(1.51/VM)
33:       X = X0 - HW * TAN(THETA)
34:
35: *  guess at gradient and calculate solution
36:
37:       LOOP
38:       .  CALL SETDRM(4)
39: :HCONST
40:       .  CALL GRSTR(1.2,4.0,"enter guess for gradient",24,0.1,0.0)
41:       .  READ (0,FMT="(F10.0)") U
42:       .  EXIT LOOP IF (U .LE. 0.0)
43:       .  K = U
44:       .  IF ((VM/K) .LT. (X/2.0))
45: :NO HCONST
46:       .  .  CALL REMARK(" gradient too high.")
47:       .  ELSE
48:       .  .  T = (2.0 / K) * ASINH((X/2.0)*SQRT(1.0/((VM/K)**2-(X/2.0)**2)))
49:       .  .  XX = 1.0 + (X0 - XMIN) * XSCALE
50:       .  .  YY = 0.5 + (T - YMIN + HW / (1.51 * COS(THETA))) * YSCALE
51:       .  .  CALL ARROW(XX,YY,0.5,0.0)
52:       .  END IF
53:       END LOOP
54:
```

```
55:       CALL ENDPLT
56:
57:       V0 = K * SQRT((VM/K)**2-(X/2.0)**2)
58:       Z = (VM - V0) / K
59:
60:       WRITE (3,FMT="(' gradient ',F8.2/' surface velocity ',F8.2/
61:      &  ' depth to turning point ',F8.2)") K, V0, Z
62:
63:       RETURN
64:       END
```

```
 1: *-h- CLFVEL 114  5 MAY 83   0:06:48
 2:       SUBROUTINE CLFVEL(PD)
 3:
 4: * close files
 5:
 6:       INTEGER PD
 7:
 8:       CALL FILCL(PD,"KEEP")
 9:
10:       RETURN
11:       END
```

A.5 Program LSVZ.

This is the program that performs the actual inversion, the calculation of a model given the observations. The T(X) travel time have been reparametrized by TXFIT to the form $\zeta(p)$ and $\tau(p)$ because the latter offer advantages in terms of mutual statistical independence to the effects of errors in the computation of p. The parameters ζ and τ are considered as independent observations of a model parameter. With this assumption an overdetermined system of equations can be generated relating and and the reciprocal velocity gradients. Because the coefficient matrix can be near singular, the least squares inverse is calculated with the signular value decomposition (SVD) instead of the faster QR decomposition. The SVD products give valuable information on how to partition the matrix by discarding small singular values and their associated vectors. This procedure reduces the condition of the matrix and results in a more stable inverse.

The program is invoked by typing

```
LSVZ infile [listfile] [/CDSX]
```

where infile is the parameter file output from TXFIT with the addition of the sruface velocity value computed by VEL0; listfile is an otional file for the listing of the decomposition products and the solution. The options are C to indicate that the program is to be run as a batch job (certain file assignments are changed); D specifies that only the

singular value decomposition is calculated and the products are stored in a temporary file; S specifies that the decomposition products are to be read from a temporary file and the solution calculated from them; and X means that an extended listing will be output. The program generates up to six plots which must be handled by the VERSAPLOT phase 2 processor and copied to the electrostatic plotter.

The program reads the parameters from infile. If the S option is not specified it generates 2n x n matrix of the coefficients where n is the number of observations (procedures GENSYS and PRMKER). The matrix is premultiplied by the diagonal matrix of the error bounds on the observations, and postmultiplied by a diagonal matrix of weight coefficients calculated to make the norm of the errors for each row approximately equal (procedure DECOMP). The singular value decomposition of the matrix is calculated (procedure MSVDC). yielding the singular values and the right and left singular vectors.

If the D option is not specified. the singular values are printed to the display. The user chooses a cut-off value beyond which the singular values are considered too small to be used in the solution. A useful value is a number r such that the ratio of the first singular value and the r-th singular value is about ten. The generalized inverse solution is computed with the remaining singular values and vectors. The solution is in the form of inverse velocity gradient as a function of ray parameter. Also calculated are the model residual, the data information density matrix, the model resolution matrix, and the model covariance matrix (procedure GISOL). The desired model is turning point depth as a

function of velocity (inverse of ray parameter). The vector of gradients is numerically integrated with respect to velocity to yield the depths (procedure INTGRD).

There are six plots available. They are: 1) velocity vs. depth; 2) gradient vs. depth; 3) the solution, inverse gradient vs. ray parameter; 4) the and kernels; 5) the information density, model resolution, and model covariance matrices; and 6) the eigenvalues and the orthogonal projections onto different subspaces of the data matrix. The user enters a six digit number where each digit specifies the corresponding plot and a one means yes and zero means no.

```
1: *-h- LSMAIN 10802  4 MAY 83  22:42:32
2:       PROGRAM LSVZ
3:
4: ************************************************************************
5: *                                                                      *
6: *  Linear inversion of body wave travel time data                      *
7: *                                                                      *
8: *    The relationship between the parameters tau(p), zeta(p) and z(v)  *
9: *   is posed as a linear problem  Ax = y  where y are the observed     *
10: *   parameters tau, zeta; x are the constant velocity gradient layers  *
11: *   and A is the coefficient matrix relating the two.                  *
12: *                                                                      *
13: *   let p be the slowness for each arrival and v  the layer velocity   *
14: *                                               i                      *
15: *   and let v     = min(1/p,v )                                        *
16: *           lim            i                                          *
17: *                                      2 2    1/2                      *
18: *                    v    (1 + (1 - p v     )   )                    *
19: *                     lim               j-1                          *
20: *    A   = 2 ( log ( ------------------------   )                     *
21: *     ij                               2 2    1/2                     *
22: *                    v    (1 + (1 - p v     )   )                    *
23: *                     j-1               lim                          *
24: *                                                                     *
25: *                       2 2    1/2          2 2    1/2               *
26: *              - (1 - p v     )     + (1 - p v     )      )           *
27: *                        lim                 j-1                     *
28: *                                                                     *
29: *  for i: 1,3,5,7,...     and j: 1,2,3,4,...                         *
30: *  and                                                                *
31: *                                                                     *
32: *                                      2 2    1/2                     *
33: *                    v    (1 + (1 - p v     )   )                    *
34: *                     lim               j-1                          *
35: *    A   = 2 ( log ( ------------------------   )                     *
36: *     ij                               2 2    1/2                     *
37: *                    v    (1 + (1 - p v     )   )                    *
38: *                     j-1               lim                          *
39: *                                                                     *
40: *                     2 2     1/2           2 2    1/2               *
41: *             + (1 - p v      )      - (1 - p v     )                 *
42: *                       lim                 j-1                     *
43: *                                                                     *
44: *   for i: 2,4,6,8,...     and j: 1,2,3,4,                           *
45: *                                                                     *
46: *   The velocity gradients are then integrated to yield depth as a    *
47: *   function of velocity.                                             *
48: *                                                                     *
49: *    see  Dorman LM, Jacobson RS (1981)  Geophysics v46,2 p.138-151   *
50: *           Linear inversion of body wave data - Part 1: velocity     *
51: *            structure from travel times and ranges                   *
52: *                                                                     *
53: ************************************************************************
54: *                                                                     *
55: *   usage:  LSVZ  infile [outfile] [/CDSX]                            *
56: *                                                                     *
57: *      infile: line 1: initial value and plot parameters  (8f10.0)    *
58: *               surface velocity, max velocity, max gradient,         *
59: *               max depth, min slowness, max slowness,                *
60: *               min inv gradient, max inv gradient                    *
61: *                (surface velocity is required, other quantities will *
62: *                 be calculated by program if omitted)                *
63: *              line 2: plot title  (30a1)                             *
64: *              line 3: plot title  (30a1)                             *
65: *              line 4-n: observations  (5f10.0)  from curve fitter    *
66: *               p, zeta, tau, sigma(zeta), sigma(tau)                 *
67: *                                                                     *
68: *      outfile: list output - optional, if not specified output goes  *
69: *                    to lfn 6                                         *
70: *                                                                     *
71: *     options:  /                                                     *
72: *       C:      use when running on background (control point)        *
73: *       D:      singular value decomposition only, results are written *
74: *               to a binary file                                      *
75: *       S:      no decomposition, singular values and vectors are read *
76: *               from binary file                                      *
77: *       X:      extended list output                                  *
78: *                                                                     *
79: ************************************************************************
80: *                                                                      *
```

```
*   source file: 1512REF*LSMAIN,  subroutines in 1512REF*A.INVERT      *
*   libraries: (for Versatec graphics)                                 *
*               1512REF*REFRLB, 1512ROSE*MATHLB, 1512ROSE*CHARLB,      *
*               1512ROSE*PLOTLB, 1512ROSE*VPLB, *SAUVPL                *
*               *LIBERY                                                *
*             (for HP graphics)                                        *
*               1512REF*REFRLB,1512ROSE*MATHLB, 1512ROSE*CHARLB        *
*               1512ROSE*PLOTLB, 1512ROSE*HPPLB,                       *
*               *LIBERY                                                *
*   executable:  1512REF*LSVZ                                          *
*                                                                      *
************************************************************************

      SPECIAL COMMON CARRAY

      INTEGER NPARAM, NLAYER, NRT
      REAL P(100), UP(50), PARS(100), SIGMA(100)
      REAL A(100,100), WEIGHT(100), DZDV(50,2), Z(50,2), DV(50)
      REAL Q(50), U(100,100), V(50,50), E(50), WORK(100)
      REAL COV(50,50), RSD(100)
      COMMON /CARRAY/ A, U, V, COV
      INTEGER TITLE1(30), TITLE2(30)
      REAL VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
      COMMON /CPLOT/ VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX,
     &  TITLE1, TITLE2
      INTEGER ID, OD, BD, TD, COPT, DOPT, SOPT, XOPT
      INTEGER LDA, LDU, LDV, LDC, LDZ, PLOTOPT
      REAL T, VSURF, UPMAX, UPMIN, RHO
      LOGICAL WVVZ, WGVZ, WGVP, WTZD, WRIC, WPRJ
      REAL INNERP, SQRT
      DATA LDA /100/, LDU /100/, LDV /50/, LDC /50/, LDZ /50/

      CALL OPFINV(ID,LD,BD,TD,COPT,DOPT,SOPT,XOPT)
      CALL R1FIT(VSURF,P,UP,PARS,SIGMA,NPARAM,NLAYER,ID)

      UPMAX = 1.0 / VSURF
      UPMIN = P(1)

      CALL LSTDAT(NPARAM,NLAYER,VSURF,UP,PARS,SIGMA,LD)

*  part I -- form linear system of equations and decompose

      IF (SOPT .NE. 1)

*  form observational equations

      .  CALL GENSYS(A,LDA,NPARAM,NLAYER,P,UP,UPMAX)
      .
      .  IF (XOPT .EQ. 1) CALL MPRINT(A,LDA,NPARAM,NLAYER,"(12F10.3)",
     &.    1,LD," coefficient matrix")
      .
*  solve system for velocity gradients
      .
      .  CALL DECOMP(A,LDA,NPARAM,NLAYER,SIGMA,WEIGHT,U,LDU,Q,V,LDV,E,
     &.    WORK,11,INFO)
      .  IF (INFO .NE. 0) CALL ERROR(" unsuccesful decomposition")
      .
      .  IF (XOPT .EQ. 1)
      .  .  CALL MPRINT(U,LDU,NPARAM,NPARAM,"(12F10.3)",1,LD," U")
      .  .  CALL MPRINT(Q,NLAYER,NLAYER,1,"(12F10.3)",11,LD," eigenvalues")
      .  .  CALL MPRINT(V,LDV,NLAYER,NLAYER,"(12F10.3)",1,LD," V")
      .  END IF
      .
      .  IF (DOPT .EQ. 1)
     &.    CALL BINSAV(WEIGHT,U,LDU,Q,V,LDV,NPARAM,NLAYER,1,BD)
      .
      END IF

*  part II -- calculate solution in terms of gradients

      IF (DOPT .NE. 1)

      .  IF (SOPT .EQ. 1)
     &.    CALL BINSAV(WEIGHT,U,LDU,Q,V,LDV,NPARAM,NLAYER,2,BD)
      .  CALL MPRINT(Q,NLAYER,NLAYER,1,"(5F10.3)",11,3," singular values")
      .  CALL REMARK("@N enter number of singular values to retain")
      .  READ (TD,) NRT
      .  WRITE (LD,"(I5,' singular values retained')") NRT
      .
```

```
160:       .  CALL GISOL(U,LDU,Q,V,LDV,PARS,SIGMA,WEIGHT,NPARAM,NLAYER,NRT,DZDV,
161:      &.    COV,LDC,RSD,11111,A)
162:       .
163:       .  IF (XOPT .EQ. 1) CALL MPRINT(RSD,NPARAM,NPARAM,1,"(12F10.3)",11,
164:      &.    LD," model residuals")
165:       .  RHO = INNERP(RSD,1,RSD,1,NPARAM)
166:       .  WRITE (LD,"(' sum square error',F8.2)") INNERP(RSD,1,RSD,1,NPARAM)
167:       .  WRITE (LD,"(' condition number',F8.2)") Q(1) / Q(NRT)
168:       .  IF (XOPT .EQ. 1) CALL MPRINT(COV,LDC,NLAYER,NLAYER,"(12F10.3)",1,
169:      &.    LD," model covariance")
170:       .
171: *  calculate 2-sigma limits for velocity gradients
172:       .
173:       .  CALL VSQRT(DZDV(1,2),1,1.0,COV,LDC+1,0.0,NLAYER)
174:       .  WRITE (LD,"(/'     dz/dv     95% confidence limit'/)")
175:       .  WRITE(LD,"(F10.4,6X,'+-',F10.4)") ((DZDV(I,J),J=1,2),I=1,NLAYER)
176:       .
177: *  calculate depths and 2-sigma limits
178:       .
179:       .  CALL INTGRD(Z,DV,DZDV,UP,UPMAX,NLAYER)
180:       .  FOR I = 1,NLAYER
181:       .  .  T = 0.0
182:       .  .  FOR K = 1,I
183:       .  .  .  FOR J = 1,I
184:       .  .  .  .  T = T + DV(J) * COV(J,K) * DV(K)
185:       .  .  .  END FOR
186:       .  .  END FOR
187:       .  .  Z(I,2) = SQRT(T)
188:       .  END FOR
189:       .  WRITE(LD,"(/'     z(v)      95% confidence limit              v'
190:      &.    /)")
191:       .  FOR I = 1,NLAYER
192:       .  .  WRITE (LD,"(F10.4,6X,'+-',F10.4,'    ..........',F10.4)") Z(I,1),
193:      &.  .    Z(I,2),1.0/UP(I)
194:       .  END FOR
195:       .
196:       END IF
197:
198: *  part III -- choose plots desired
199:
200:       WRITE (3,"(/' plots desired  (6 digit decimal expansion  abcdef:
201:      &yes=1, no=0)')")
202:       WRITE (3,"(' < V v. Z >< dV/dZ v. Z >< dZ/dV v. p >< zeta,tau >< r
203:      &esolution >< projections >')")
204:       READ (TD,) PLOTOPT
205:       WVVZ = (PLOTOPT / 100000) .GT. 0
206:       WGVZ = MOD(PLOTOPT/10000,10) .GT. 0
207:       WGVP = MOD(PLOTOPT/1000,10) .GT. 0
208:       WTZD = MOD(PLOTOPT/100,10) .GT. 0
209:       WRIC = MOD(PLOTOPT/10,10) .GT. 0
210:       WPRJ = MOD(PLOTOPT,10) .GT. 0
211:
212:       IF (WVVZ) CALL PLTVVZ(Z,LDZ,UP,VSURF,WORK,NLAYER)
213:       IF (WGVZ) CALL PLTGVZ(DZDV,Z,WORK,NLAYER)
214:       IF (WGVP) CALL PLTGVP(DZDV,LDZ,UP,RSD,Q,NLAYER,NPARAM,NRT)
215:       IF (WTZD) CALL REMARK(" tau, zeta plots not implemented")
216:       IF (WRIC) CALL PLTRIC(U,LDU,V,LDV,COV,LDC,NPARAM,NLAYER)
217:       IF (WPRJ)
218:      &  CALL PLTPRJ(Q,U,LDU,V,LDV,PARS,DZDV,NPARAM,NLAYER,NRT,UP,WORK)
219:
220:       CALL CLFINV(ID,LD)
221:
222:       END


  1: *-h- OPFINV 1385  4 MAY 83  22:30:38
  2:       SUBROUTINE OPFINV(ID, LD, BD, TD, COPT, DOPT, SOPT, XOPT)
  3:
  4: * open input & output files and initialize plotting
  5:
  6:       INTEGER ID, LD, BD, TD, COPT, DOPT, SOPT, XOPT
  7:       INTEGER NAME(18), BUF(81), I
  8:       INTEGER FILCR, FILCRB, FILOP, GETARG, GETOPT
  9:
 10:       CALL PRIMIO(11)
 11:       IF (GETARG(1,NAME,18) .EQ. -1) THEN
 12:       .  CALL ERROR(" usage:  LSVZ  infile [listfile] [/CDSX]")
 13:       ELSE
 14:       .  ID = FILOP(NAME)
 15:       .  IF (ID .EQ. -3) CALL CANT(NAME)
```

```
16:       END IF
17:
18:       IF (GETARG(2,NAME,18) .EQ. -1) THEN
19:       .  LD = 6
20:       ELSE
21:       .  LD = FILOP(NAME)
22:       .  IF (LD .EQ. -3) THEN
23:       .  .  LD = FILCR(NAME)
24:       .  .  IF (LD .EQ. -3) CALL CANT(NAME)
25:       .  END IF
26:       END IF
27:
28:       COPT = GETOPT(67)     !  run from control point
29:       IF (COPT .EQ. 1) THEN
30:       .  TD = 7
31:       ELSE
32:       .  TD = 0
33:       END IF
34:
35:       DOPT = GETOPT(68)     !  decompose system but do not solve
36:       SOPT = GETOPT(83)     !  solve decomposed system
37:       IF (DOPT .EQ. 1 .OR. SOPT .EQ. 1) THEN
38:       .  CALL REMARK(" enter binary data storage file name")
39:       .  CALL GETLIN(BUF,TD)
40:       .  I = 1
41:       .  CALL GETWRD(BUF,I,NAME)
42:       .  BD = FILOP(NAME)
43:       .  IF (BD .EQ. -3) THEN
44:       .  .  IF (DOPT .EQ. 1) BD = FILCRB(NAME)
45:       .  .  IF (BD .EQ. -3) CALL CANT(NAME)
46:       .  END IF
47:       END IF
48:
49:       XOPT = GETOPT(88)     !  extended output listing
50:
51:       CALL VP07MP("MODEL",1200,"END")
52:       CALL INIPLT
53:
54:       RETURN
55:       END


 1: *-h- R1FIT 1079  4 MAY 83  22:30:40
 2:       SUBROUTINE R1FIT(VSURF, P, UP, PARS, SIGMA, NPARAM, NLAYER, D)
 3:
 4: * read in p, zeta, tau, sigma zeta, sigma tau
 5:
 6:       INTEGER NLAYER, NPARAM, D
 7:       REAL VSURF, P(*), UP(*), PARS(*), SIGMA(*)
 8:       INTEGER TITLE1(30), TITLE2(30)
 9:       REAL VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
10:       COMMON /CPLOT/ VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX,
11:      &  TITLE1, TITLE2
12:       INTEGER I, IP1, J
13:       REAL BUF(5)
14:
15:       READ (D,"(8F10.0)") VSURF,VMAX,GMAX,ZMAX,PMIN,PMAX,IGMIN,IGMAX
16:       READ (D,"(30A3)") TITLE1
17:       READ (D,"(30A3)") TITLE2
18:
19:       I = -1
20:       J = 0
21:       LOOP
22:       .  READ (D,"(5F10.0)",END=10) BUF
23:       .  I = I + 2
24:       .  IP1 = I + 1
25:       .  J = J + 1
26:       .  IF (J .GE. 50)
27:      &.  CALL ERROR(" too many input data; max=50")
28:       .  P(I) = BUF(1)              ! p
29:       .  P(IP1) = BUF(1)
30:       .  UP(J) = BUF(1)
31:       .  PARS(I) = BUF(2)           ! zeta
32:       .  PARS(IP1) = BUF(3)         ! tau
33:       .  SIGMA(I) = BUF(4)          ! sigma zeta
34:       .  SIGMA(IP1) = BUF(5)        ! sigma tau
35:       END LOOP
36:    10 CONTINUE
37:
```

```
38:       NPARAM = IP1
39:       NLAYER = J
40:
41:       RETURN
42:       END
```

```
 1: *-h- LSTDAT 747 24 OCT 82  16:43:23
 2:       SUBROUTINE LSTDAT(NP, NO, V0, P, TZ, S, D)
 3:
 4: *  list input data to list file
 5:
 6:       INTEGER NP, NO, D
 7:       REAL V0, P(1), TZ(1), S(1)
 8:       INTEGER TITLE1(30), TITLE2(30)
 9:       REAL VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
10:       COMMON /CPLOT/ VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX,
11:      &  TITLE1, TITLE2
12:
13:       WRITE (D,FMT="(30A3)") TITLE1
14:       WRITE (D,FMT="(/' Velocity at surface',F8.2/)") V0
15:       WRITE (D,FMT="(' Parameters',I5,'     Observations',I5)") NP, NO
16:       WRITE (D,FMT="(/7X,'P',9X,'Zeta',8X,'Tau',5X,'Sigma(Zeta)',1X,
17:      &  'Sigma(Tau)'/)")
18:       I = 1
19:       FOR J = 1,NO
20:       .  IP1 = I + 1
21:       .  WRITE (D,FMT="(5F11.3)") P(J), TZ(I), TZ(IP1), S(I), S(IP1)
22:       .  I = I + 2
23:       END FOR
24:
25:       RETURN
26:       END
```

```
 1: *-h- GENSYS 511 24 OCT 82  16:43:24
 2:       SUBROUTINE GENSYS(A, LDA, NPARAM, NLAYER, P, UP, UPMAX)
 3:
 4: *   generate coefficients for system of equations
 5:
 6:       INTEGER LDA, NPARAM, NLAYER
 7:       REAL A(LDA,1), P(1), UP(1), UPMAX
 8:       INTEGER I, IP, J
 9:       REAL PREVP, TAU, ZETA
10:
11:       FOR I = 1,NPARAM,2
12:       .  PREVP = UPMAX
13:       .  J = 1
14:       .  WHILE (P(I) .LT. PREVP)
15:       .  .  CALL PRMKER(P(I),1.0/PREVP,1.0/UP(J),ZETA,TAU)
16:       .  .  A(I,J) = ZETA
17:       .  .  A(I+1,J) = TAU
18:       .  .  PREVP = UP(J)
19:       .  .  J = J + 1
20:       .  END WHILE
21:       END FOR
22:
23:       RETURN
24:       END
```

```
 1: *-h- PRMKER 511 15 NOV 81  14:01:21
 2:       SUBROUTINE PRMKER(P, V1, V2, ZETA, TAU)
 3:
 4: * calculate zeta, tau as functions of the layer parameters.
 5: *  factor of dz/dv not included  (see Dorman   eq (32) )
 6:
 7:       REAL P, V1, V2, ZETA, TAU
 8:       REAL T, TA, TB, VLIM, X
 9:       REAL ALOG, AMIN1, SQRT
10:
11:       VLIM = AMIN1(1.0/P,V2)
12:       TA = SQRT(1.0-(P*P)*(V1*V1))
13:       TB = SQRT(1.0-(P*P)*(VLIM*VLIM))
14:       X = (2.0/P) * (TA - TB)
15:       T = 2.0 * ALOG((VLIM/V1)*((1.0+TA)/(1.0+TB)))
16:       ZETA = T + P * X
17:       TAU = T - P * X
18:
19:       RETURN
20:       END
```

```
1: *-h- DECOMP 3202 28 FEB 83  17:01:01
2:       SUBROUTINE DECOMP(A,LDA,M,N,S,W,U,LDU,Q,V,LDV,E,WORK,JOB,INFO)
3:
4: ************************************************************************
5: *                                                                      *
6: *   singular value decomposition of a weighted m by n matrix A         *
7: *                                                                      *
8: *                                      T                               *
9: *                         A = U diag(q) V                              *
10: *                                                                     *
11: *                                -1/2     1/2                         *
12: *                      where A = S     A' W                          *
13: *                                                                     *
14: ************************************************************************
15: *                                                                     *
16: *     A      input:  M by N matrix of coefficients; contents destroyed *
17: *     LDA    input:  leading dimension of A in calling program        *
18: *     M      input:  number of parameters                             *
19: *     N      input:  number of observations                           *
20: *     S      input:  M element vector of variances on observations    *
21: *     W      output: M element vector of weight coefficients          *
22: *     U      output: M by N matrix of data eigenvectors               *
23: *     LDU    input:  leading dimension of U in calling program        *
24: *     Q      output: N element vector of singular values              *
25: *     V      output: N by N matrix of solution eigenvectors           *
26: *     LDV    input:  leading dimension of V in calling program        *
27: *     E      output: N element vector of superdiagonals               *
28: *     WORK   input:  work area, must be greater than or equal to M    *
29: *     INFO   output: singular values Q(info+1),...,Q(n) are correct   *
30: *     JOB    input:  decimal expansion ab                             *
31: *                                                                     *
32: ************************************************************************
33:
34:       INTEGER LDA, M, N, LDU, LDV, JOB, INFO
35:       REAL A(LDA,*), S(*), W(*), U(LDU,*), Q(*), V(LDV,*)
36:       REAL E(*), WORK(*)
37:       INTEGER I
38:       REAL T
39:       REAL FLOAT, INNERP, SQRT, SVSMAB
40:
41: *                     -1/2
42: *  store diagonal of S     in s
43:
44:       CALL VRCP(S,1,1.0,S,1,0.0,M)
45:
46: *                      -1/2
47: *   put column sums of S    A  in A
48:
49:       FOR I = 1,N
50:       .  CALL VMUL(A(1,I),1,1.0,A(1,I),1,0.0,S,1,0.0,M)
51:       .  W(I) = SVSMAB(A(1,I),1,M)
52:       END FOR
53:
54: *             -1/2
55: *  calculate W     store in w
56:
57:       T = 1.0
58:       FOR I = 1,N
59:       .  W(I) = 1 / SQRT(ABS(W(I)))
60:       .  T = T * W(I)
61:       END FOR
62:       T = SQRT(FLOAT(N)) * T ** (1.0/N)
63:       CALL VSMA1(W,1,1.0/T,W,1,0.0,N)
64:
65: *             -1/2  1/2
66: *  calculate S    AW
67:
68:       FOR I = 1,N
69:       .  CALL VSMA1(A(1,I),1,W(I),A(1,I),1,0.0,M)
70:       END FOR
71:
72: *                                                 T
73: *  singular value decomposition  A = U diag(q) V
74: *
75:
76:       CALL MSVDC(A,LDA,M,N,Q,E,U,LDU,V,LDV,WORK,JOB,INFO)
77:
78:       RETURN
79:       END
```

```
 1: *-h- BINSAV 645 23 SEP 82  20:12:06
 2:       SUBROUTINE BINSAV(W, U, LDU, Q, V, LDV, M, N, RW, D)
 3:
 4: * storage and retrieval of decomposition of system of equations
 5:
 6:       INTEGER LDU, LDV, M, N, RW, D
 7:       REAL W(1), U(LDU,1), Q(1), V(LDV,1)
 8:
 9:       IF (RW .EQ. 1)
10:       .  WRITE (D) (W(I), I=1,N)
11:       .  FOR J = 1,N
12:       .  .  WRITE (D) (U(I,J), I=1,M)
13:       .  END FOR
14:       .  WRITE (D) (Q(I), I=1,N)
15:       .  FOR J = 1,N
16:       .  .  WRITE (D) (V(I,J), I=1,N)
17:       .  END FOR
18:       ELSE
19:       .  READ (D) (W(I), I=1,N)
20:       .  FOR J = 1,N
21:       .  .  READ (D) (U(I,J), I=1,M)
22:       .  END FOR
23:       .  READ (D) (Q(I), I=1,N)
24:       .  FOR J = 1,N
25:       .  .  READ (D) (V(I,J), I=1,N)
26:       .  END FOR
27:       END IF
28:
29:       RETURN
30:       END
```

```
 1: *-h- GISOL 6161 28 FEB 83  17:01:05
 2:       SUBROUTINE GISOL(U,LDU,Q,V,LDV,Y,S,W,M,N,R,X,COV,LDC,RSD,JOB,WORK)
 3:
 4: ************************************************************************
 5: *                                                                      *
 6: * Generalized inverse solution of Ax = y                               *
 7: *                                                                      *
 8: *          -1/2    1/2         -1/2          -1/2                      *
 9: *     A = S    A' W    ;  x = S    x';  y = W    y'                     *
10: *                                                                      *
11: *      where A is a m by n coefficient matrix (m > n) relating         *
12: *       observations y and parameters x                                *
13: *      A is given by the decomposed system                             *
14: *                             T                                        *
15: *                      A = U Q V                                       *
16: *       where                                                          *
17: *                                                                      *
18: *           U = (U ,U ) ;   Q = (diag(q) ,0) ;   V = (V ,V )           *
19: *                r  0                  r             r  0            *
20: *                                                                      *
21: *       so that                                                        *
22: *                                -1   T                                *
23: *                    x = V diag(q  ) U y                               *
24: *                         r         r r                                *
25: *                                                                      *
26: *     sharp cut-off of singular values at point r determined by user   *
27: *                                                                      *
28: ************************************************************************
29: *                                                                      *
30: *     U       input:  M by M matrix of data eigenvectors               *
31: *             output: if requested M by M information density matrix   *
32: *     LDU     input:  leading dimension of U in calling program        *
33: *     Q       input:  N element vector of singular values              *
34: *     V       input:  N by N matrix of solution eigenvectors           *
35: *             output: if requested N by N resolution matrix            *
36: *     LDV     input:  leading dimension of V in calling program        *
37: *     Y       input:  M element vector of observations                 *
38: *     S       input:  M element vector of variance on observations     *
39: *     W       input:  M element vector of weight coefficients          *
40: *     M       input:  number of parameters                             *
41: *     N       input:  number of observations                           *
42: *     R       input:  number of singular values retained in cut-off    *
43: *     X       output: if requested N element vector of model parameters *
44: *     COV     output: if requested N by N model covariance matrix      *
45: *     LDC     input:  leading dimension of COV                         *
46: *     RSD     output: M element vector of residuals                    *
```

```
47: *     JOB    input:  integer decimal expansion abcde                  *
48: *                      a > 0  calculate solution                      *
49: *                      b > 0  calculate model residual                *
50: *                      c > 0  calculate model covariance              *
51: *                      d > 0  calculate data information density      *
52: *                      e > 0  calculate model resolution              *
53: *      WORK          M*M element work space                           *
54: *                                                                     *
55: ***********************************************************************
56:
57:       INTEGER LDU, LDV, M, N, R, LDC, JOB
58:       REAL U(LDU,*), Q(*), V(LDV,*), Y(*), S(*), W(*)
59:       REAL X(*), COV(LDC,*), RSD(*), WORK(*)
60:       INTEGER I
61:       LOGICAL WSOL, WRSD, WCOV, WINF, WRES
62:
63: *  determine job to be done
64:
65:       WSOL = JOB / 10000 .GT. 0
66:       WRSD = MOD(JOB/1000,10) .GT. 0
67:       WCOV = MOD(JOB/100,10) .GT. 0
68:       WINF = MOD(JOB/10,10) .GT. 0
69:       WRES = MOD(JOB,10) .GT. 0
70:
71: *  generalized inverse solution to decomposed system
72:
73:       IF (WSOL)
74:
75: *            -1/2
76: *  calculate S    y  store in y
77:
78:       .  CALL VMUL(Y,1,1.0,S,1,0.0,Y,1,0.0,M)
79:       .
80: *                                  -1   T
81: *  calculate solution  x = V diag(q  ) U y   store in y
82: *                           r         r r
83:       .
84:       .  CALL MTPRD(WORK,R,U,LDU,Y,M,M,R,1)
85:       .  CALL VDIV(WORK,1,1.0,Q,1,0.0,WORK,1,0.0,R)
86:       .  CALL MPRD(X,N,V,LDV,WORK,R,N,R,1)
87:       .
88: *            -1/2
89: *  calculate W    x  store in x
90:       .
91:       .  CALL VMUL(X,1,1.0,W,1,0.0,X,1,0.0,N)
92:       END IF
93:
94: *                                                    T
95: *  calculate residuals rsd = y - Ax  = P'y = (U )(U ) y
96: *                                     X      0   0
97:
98:       IF (WRSD)
99:       .  CALL MTPRD(WORK,M-R,U(1,R+1),LDU,Y,M,M,M-R,1)
100:      .  CALL MPRD(RSD,M,U(1,R+1),LDU,WORK,M-R,M,M-R,1)
101:      END IF
102:
103: *                                       T     T        -2   T
104: *  calculate model covariance matrix  <xx > = W V diag(q  ) V W
105: *                                            r         r r
106:
107:      IF (WCOV)
108:      .  FOR I = 1,R
109:      .  .  CALL VSMA1(WORK(N*(I-1)+1),1,1.0/Q(I),V(1,I),1,0.0,N)
110:      .  END FOR
111:      .  CALL MPRDT(COV,LDC,WORK,N,WORK,N,N,R,N)
112:      .  FOR J = 1,N
113:      .  .  FOR I = 1,N
114:      .  .  .  COV(I,J) = W(I) * COV(I,J) * W(J)
115:      .  .  END FOR
116:      .  END FOR
117:      END IF
118:
119: *                                    T
120: *  calculate information density  E = U U
121: *                                   r r
122:
123:      IF (WINF)
124:      .  CALL MPRDT(WORK,M,U,LDU,U,LDU,M,R,M)
125:      .  FOR I = 1,M
126:      .  .  CALL VCOPY(U(1,I),1,WORK(M*(I-1)+1),1,M)
```

```
127:       . END FOR
128:       END IF
129:
130: *                                      T
131: *  calculate model resolution  R = V V
132: *                                     r r
133:
134:       IF (WRES)
135:       . CALL MPRDT(WORK,N,V,LDV,V,LDV,N,R,N)
136:       . FOR I = 1,N
137:       . . CALL VCOPY(V(1,I),1,WORK(N*(I-1)+1),1,N)
138:       . END FOR
139:       END IF
140:
141:       RETURN
142:       END
```

```
  1: *-h- INTGRD 411  4 MAY 83  22:42:45
  2:       SUBROUTINE INTGRD(Z,DV,DZDV,P,PMAX,N)
  3:
  4: *  integrate velocity gradient wrt velocity to get depth
  5:
  6:       INTEGER N
  7:       REAL Z(*), DZDV(*), DV(*), P(*), PMAX
  8:       INTEGER I
  9:       REAL PLAST, ZLAST
 10:
 11:       PLAST = PMAX
 12:       ZLAST = 0.0
 13:       FOR I = 1,N
 14:       . DV(I) = 1.0 / P(I) - 1.0 / PLAST
 15:       . Z(I) = ZLAST + DZDV(I) * DV(I)
 16:       . PLAST = P(I)
 17:       . ZLAST = Z(I)
 18:       END FOR
 19:
 20:       RETURN
 21:       END
```

```
  1: *-h- PLTVVZ 1689  4 MAY 83  22:50:49
  2:       SUBROUTINE PLTVVZ(Z, LDZ, P, VSURF, WORK, N)
  3:
  4: *  plot velocity and 2-sigma limits vs depth
  5:
  6:       INTEGER LDZ, N
  7:       REAL Z(LDZ,2), P(*), VSURF, WORK(*)
  8:       INTEGER TITLE1(30), TITLE2(30)
  9:       REAL VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
 10:       COMMON /CPLOT/ VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX,
 11:      &  TITLE1, TITLE2
 12:       INTEGER I, J, NAX, NAY, PEN, PENDOWN, PENUP, SOPT
 13:       REAL HEIGHT,XLEN,XMAX,XMIN,XSCALE,YY,YLEN,YMAX,YMIN,YSCALE
 14:       DATA PENDOWN /1/, PENUP /0/
 15:       DATA XLEN /5.0/, YLEN /8.0/, HEIGHT /0.15/
 16:
 17:       CALL REMARK(" Plot velocity vs depth")
 18: :HCONST
 19:       CALL VRCP(WORK,1,1.0,P,1,0.0,N)
 20:
 21:       CALL NEWPEN(3)
 22:       CALL GRSTR(1.0,10.0,TITLE1,30,0.2,0.0)
 23:
 24:       CALL MOVEOO(1.0,9.0)
 25:
 26:       IF (VMAX .EQ. 0.0)
 27:       . SOPT = 1
 28:       . XMAX = SVMAX(WORK,1,JUNK,N)
 29:       ELSE
 30:       . SOPT = 0
 31:       . XMAX = VMAX
 32:       END IF
 33:       XMIN = 0.0
 34:       NAX = INT(XMAX)
 35:       CALL PAXIS(0.0,XLEN,0.0,0.0,XMIN,XMAX,4,-1,HEIGHT,4,90.0,SOPT,
 36:      &  NAX,1)
 37:       XSCALE = XLEN / XMAX
 38:       CALL AXLAB(0.0,XLEN,0.0,0.0,"VELOCITY  (KM Sn-lo).",
 39:      &  HEIGHT,3.5*HEIGHT)
```

```
40:
41:       IF (ZMAX .EQ. 0.0)
42:       .  SOPT = 1
43:       .  YMAX = SVMAX(Z(1,1),1,JUNK,N)
44:       ELSE
45:       .  SOPT = 0
46:       .  YMAX = ZMAX
47:       END IF
48:       YMIN = 0.0
49:       NAY = INT(YMAX)
50:       CALL PAXIS(0.0,0.0,0.0,-YLEN,YMIN,YMAX,4,-1,HEIGHT,2,180.0,SOPT,
51:      &  NAY,1)
52:       YSCALE = YLEN / YMAX
53:       CALL AXLAB(0.0,0.0,-YLEN,0.0,"DEPTH  (KM).",HEIGHT,4.5*HEIGHT)
54: :NO HCONST
55:
56:       XX = VSURF * XSCALE
57:       YY = 0.0
58:       FOR I = 1,N
59:       .  XX = WORK(I) * XSCALE
60:       .  YY = -Z(I,1) * YSCALE
61:       .  CALL PNTCNF(XX,YY,0.0,Z(I,2)*YSCALE,0.05)
62:       END FOR
63:
64:       CALL ENDFIG(1)
65:
66:       RETURN
67:       END


 1: *-h- PLTGVZ 1732  4 MAY 83  22:30:49
 2:       SUBROUTINE PLTGVZ(GRD, Z, WORK, N)
 3:
 4: * plot velocity gradient vs. depth
 5:
 6:       INTEGER N
 7:       REAL GRD(*), Z(*), WORK(*)
 8:       INTEGER TITLE1(30), TITLE2(30)
 9:       REAL VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
10:       COMMON /CPLOT/ VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX,
11:      &  TITLE1, TITLE2
12:       INTEGER I, JUNK, NAX, NAY, PEN, PENDOWN, PENUP, SOPT
13:       REAL HEIGHT, XX, YY
14:       REAL XLEN, XMIN, XSCALE, YLEN, YMAX, YMIN, YSCALE
15:       REAL SVMAX
16:       DATA XLEN /5.0/, YLEN /8.0/, HEIGHT /0.15/
17:       DATA PENDOWN /1/, PENUP /0/
18:
19:       CALL REMARK(" Plot gradient vs depth")
20: :HCONST
21:       CALL VRCP(WORK,1,1.0,GRD,1,0.0,N)
22:
23:       CALL MOVE00(1.0,9.0)
24:
25:       IF (GMAX .EQ. 0.0)
26:       .  SOPT = 1
27:       .  XMAX = SVMAX(WORK,1,JUNK,N)
28:       ELSE
29:       .  SOPT = 0
30:       .  XMAX = GMAX
31:       END IF
32:       XMIN = 0.0
33:       NAX = 5
34:       CALL PAXIS(0.0,XLEN,0.0,0.0,XMIN,XMAX,4,1,HEIGHT,4,90.0,SOPT,
35:      &  NAX,1)
36:       XSCALE = XLEN / XMAX
37:       CALL AXLAB(0.0,XLEN,0.0,0.0,"DV/DZ  (Sn-1o).",HEIGHT,3.5*HEIGHT)
38:
39:       IF (ZMAX .EQ. 0.0)
40:       .  SOPT = 1
41:       .  YMAX = SVMAX(Z,1,JUNK,N)
42:       ELSE
43:       .  SOPT = 0
44:       .  YMAX = ZMAX
45:       END IF
46:       YMIN = 0.0
47:       NAY = INT(YMAX)
48:       CALL PAXIS(0.0,0.0,0.0,-YLEN,YMIN,YMAX,4,-1,HEIGHT,2,180.0,SOPT,
49:      &  NAY,1)
50:       YSCALE = YLEN / YMAX
```

```
51:       CALL AXLAB(0.0,0.0,-YLEN,0.0,"DEPTH  (KM).",HEIGHT,4.5*HEIGHT)
52: :NOT HCONST
53:
54:       IF (GMAX .EQ. 0.0)
55:       . J = 1
56:       . YY = 0.0
57:       ELSE
58:       . J = 2
59:       . YY = -Z(1) * YSCALE
60:       END IF
61:
62:       PEN = PENUP
63:       FOR I = J,N
64:       . XX = (WORK(I) - XMIN) * XSCALE
65:       . CALL MOVEPN(XX,YY,3,PEN)
66:       . PEN = PENDOWN
67:       . YY = -Z(I) * YSCALE
68:       . CALL MOVEPN(XX,YY,3,PEN)
69:       END FOR
70:
71:       CALL ENDFIG(1)
72:
73:       RETURN
74:       END
```

```
 1: *-h- PLTGVP 2641  4 MAY 83  22:54:19
 2:       SUBROUTINE PLTGVP(GRD, LDG, P, RSD, Q, N, M, R)
 3:
 4: *  plot dz/dv and 2-sigma limits versus p
 5:
 6:       INTEGER LDG, M, N, R
 7:       REAL GRD(LDG,2), P(*), RSD(*), Q(*)
 8:       INTEGER TITLE1(30), TITLE2(30)
 9:       REAL VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
10:       COMMON /CPLOT/ VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX,
11:      &  TITLE1, TITLE2
12:       INTEGER I, J, PEN, PENWIDTH(-1:1)
13:       INTEGER NAX, NAY, PENDOWN, PENUP, SOPT
14:       REAL HEIGHT, XLEN, XSCALE, YLEN, YMIN, YMAX, YSCALE, XX, YY
15:       REAL XMAX, XMIN
16:       INTEGER ITOC, FTOC
17:       REAL FLOAT, AINT, INNERP, SVMAX, SVMIN
18:       DATA PENWIDTH /1,4,1/, XLEN /6.0/, YLEN /5.0/, HEIGHT /0.15/
19:       DATA PENDOWN /1/, PENUP /0/
20:
21:       CALL REMARK(" Plot gradient vs slowness")
22: :HCONST
23:       CALL NEWPEN(3)
24:       CALL GRSTR(1.0,8.5,"SOLUTION -- ",12,HEIGHT,0.0)
25:
26:       CALL GRNUM(2.7,8.0,FLOAT(N),3,-1,HEIGHT,0.0)
27:       CALL GRSTR(3.0,8.0," LAYERS",7,HEIGHT,0.0)
28:
29:       CALL GRNUM(2.7,7.7,FLOAT(M),3,-1,HEIGHT,0.0)
30:       CALL GRSTR(3.0,7.7," DATA",5,HEIGHT,0.0)
31:
32:       CALL GRNUM(2.7,7.4,FLOAT(R),3,-1,HEIGHT,0.0)
33:       CALL GRSTR(3.0,7.4," SINGULAR VALUES RETAINED",25,HEIGHT,0.0)
34:
35:       RHO = SQRT(INNERP(RSD,1,RSD,1,M)/FLOAT(M-R))
36:       CALL GRSTR(3.0,7.1," REDUCED CHI SQUARED =",22,HEIGHT,0.0)
37:       CALL GRNUM(6.6,7.1,RHO,5,2,HEIGHT,0.0)
38:
39:       CALL GRSTR(3.0,6.8," CONDITION NUMBER =",19,HEIGHT,0.0)
40:       CALL GRNUM(6.15,6.8,Q(1)/Q(R),5,2,HEIGHT,0.0)
41:
42:       CALL MOVEOO(1.0,2.0)
43:
44:       IF (PMIN .EQ. 0.0)
45:       .  SOPT = 1
46:       .  XMIN = SVMIN(P,1,JUNK,N)
47:       .  XMAX = SVMAX(P,1,JUNK,N)
48:       ELSE
49:       .  SOPT = 0
50:       .  XMIN = PMIN
51:       .  XMAX = PMAX
52:       END IF
53:       NAX = 5
```

```
54:       CALL PAXIS(0.0,XLEN,0.0,0.0,XMIN,XMAX,4,2,HEIGHT,3,-90.0,SOPT,
55:      &NAX,1)
56:       XSCALE = XLEN / (XMAX - XMIN)
57:       CALL AXLAB(0.0,XLEN,0.0,0.0,"SLOWNESS  (S KMn-1o).",
58:      &  HEIGHT,-4.5*HEIGHT)
59:
60:       IF (IGMIN .EQ. 0.0 .AND. IGMAX .EQ. 0.0)
61:       .  SOPT = 1
62:       .  YMIN = SVMIN(GRD(1,1),1,JUNK,N)
63:       .  YMAX = SVMAX(GRD(1,1),1,JUNK,N)
64:       ELSE
65:       .  SOPT = 0
66:       .  YMIN = IGMIN
67:       .  YMAX = IGMAX
68:       END IF
69:       NAY = 5
70:       CALL PAXIS(0.0,0.0,0.0,YLEN,YMIN,YMAX,5,1,HEIGHT,2,180.0,SOPT,
71:      &  NAY,1)
72:       YSCALE = YLEN / (YMAX - YMIN)
73:       CALL AXLAB(0.0,0.0,0.0,YLEN,"DZ/DV  (S).",HEIGHT,5.5*HEIGHT)
74: :NOT HCONST
75:
76:       FOR J = -1,1
77:       .  CALL NEWPEN(PENWIDTH(J))
78:       .  PEN = PENUP
79:       .  XX = (XMAX - XMIN) * XSCALE
80:       .  FOR I = 1,N
81:       .  .  YY = (GRD(I,1) + J * GRD(I,2) - YMIN) * YSCALE
82:       .  .  CALL MOVEPN(XX,YY,3,PEN)
83:       .  .  PEN = PENDOWN
84:       .  .  XX = (P(I) - XMIN) * XSCALE
85:       .  .  CALL MOVEPN(XX,YY,3,PEN)
86:       .  END FOR
87:       END FOR
88:
89:       CALL ENDFIG(1)
90:
91:       RETURN
92:       END
```

```
 1: *-h- PLTRIC 1441  4 MAY 83  22:31:19
 2:       SUBROUTINE PLTRIC(U, LDU, V, LDV, COV, LDC, M, N)
 3:
 4: *                   T                          T
 5: * plot resolution VV  , information density UU   and model covariance
 6:
 7:       INTEGER LDU, LDV, M, N
 8:       REAL U(LDU,*), V(LDV,*), COV(LDC,*)
 9:       REAL XLEN, XOFF, YLEN, YOFF
10:       INTEGER DF
11:       DATA XLEN /3.0/, XOFF /3.9/, YLEN /3.0/, YOFF /5.0/
12:
13:       CALL REMARK(" Plot resolution,information,covariance matrices")
14: :HCONST
15:       IF (N .LE. 25)
16:       .  DF = 5
17:       ELSE
18:       .  DF = 10
19:       END IF
20:
21: *  information density matrix
22: *     zeta observations
23:
24:       CALL MOVE00(0.6,1.5)
25:       CALL NEWPEN(2)
26:       CALL AXLAB(0.0,XOFF+XLEN,0.0,0.0,"INFORMATION DENSITY.",0.15,-1.0)
27:       CALL PMAT(U,LDU,N,2,N,2,XLEN,YLEN,0.8,0.15,DF,DF)
28:       CALL AXLAB(0.0,XLEN,0.0,0.0,"ZETA OBSERVATIONS.",0.12,-0.6)
29:
30: *     tau observations
31:
32:       CALL MOVE00(XOFF,0.0)
33:       CALL NEWPEN(2)
34:       CALL PMAT(U(2,2),LDU,N,2,N,2,XLEN,YLEN,0.8,0.15,DF,DF)
35:       CALL AXLAB(0.0,XLEN,0.0,0.0,"TAU OBSERVATIONS.",0.12,-0.6)
36:
37: *  resolution matrix
38:
39:       CALL MOVE00(-XOFF,YOFF)
```

```
      CALL NEWPEN(2)
      CALL PMAT(V,LDV,N,1,N,1,XLEN,YLEN,0.8,0.15,DF,DF)
      CALL AXLAB(0.0,XLEN,0.0,0.0,"RESOLUTION.",0.15,-0.6)

*  covariance matrix

      CALL MOVEOO(XOFF,0.0)
      CALL NEWPEN(2)
      CALL PMAT(COV,LDC,N,1,N,1,XLEN,YLEN,0.8,0.15,DF,DF)
      CALL AXLAB(0.0,XLEN,0.0,0.0,"COVARIANCE.",0.15,-0.6)

      CALL ENDFIG(1)
:NO HCONST

      RETURN
      END
```

```
*-h- PLTPRJ 4791  4 MAY 83  22:31:24
      SUBROUTINE PLTPRJ(Q, U, LDU, V, LDV, Y, X, M, N, R, P, WK)

*  plot data and model eigenvectors and eigenvalues

      INTEGER LDU, LDV, M, N, R
      REAL Q(*), U(LDU,*), V(LDV,*), Y(*), X(*), P(*), WK(*)
      INTEGER TITLE1(30), TITLE2(30)
      REAL VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
      COMMON /CPLOT/ VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX,
     &  TITLE1, TITLE2
      INTEGER DF, NAX, NAY, PEN, PENDOWN, PENUP, SOPT
      REAL XLEN, XMIN, XOFF, XSCALE, XX
      REAL YLEN, YMIN, YOFF, YSCALE, YY
      REAL SVMAX, SVMIN
      DATA PENDOWN /1/, PENUP /0/
      DATA XLEN /3.0/, XOFF /3.9/, YLEN /1.5/, YOFF /3.0/

      CALL REMARK(" Plot eigenvalues and projections onto subspaces")
:HCONST

*  reciprocal of eigenvalues and cut-off

      IF (N .LE. 25)
      .  DF = 5
      ELSE
      .  DF = 10
      END IF

      CALL MOVEOO(3.5,1.8)
      CALL NEWPEN(2)
      CALL VRCP(WK,1,1.0,Q,1,0.0,N)
      CALL PAXIS(0.0,XLEN,0.0,0.0,0.0,FLOAT(N),4,-1,0.12,3,-90.0,
     &  0,N,DF)
      XSCALE = XLEN / FLOAT(N)
      CALL AXLAB(0.0,XLEN,0.0,0.0,"INVERSE.",0.15,-0.6)
      CALL AXLAB(0.0,XLEN,0.0,0.0,"EIGENVALUES.",0.15,-0.8)
      YMIN = 0.0
      YMAX = SVMAX(WK,1,JUNK,N)
      NAY = 3
      CALL PAXIS(0.0,0.0,0.0,YLEN,YMIN,YMAX,3,0,0.12,2,180.0,1,NAY,1)
      YSCALE = YLEN / (2.0 * (YMAX - YMIN))
      CALL PAXIS(XLEN,XLEN,0.0,YLEN,0.0,0.0,0,0,0.12,0,0.0,0,NAY,1)
      CALL NEWPEN(2)
      PEN = PENUP
      FOR I = 1,N
      .  XX = FLOAT(I) * XSCALE
      .  YY = (WK(I) - YMIN) * YSCALE
      .  CALL MOVEPN(XX,YY,3,PEN)
      .  PEN = PENDOWN
      END FOR
      CALL NEWPEN(1)
      XX = FLOAT(R) * XSCALE
      YY = 0.0
      CALL MOVEPN(XX,YY,3,PENUP)
      YY = YLEN
      CALL MOVEPN(XX,YY,3,PENDOWN)
      XX = XX - 0.2
      YY = YY + 0.16
      CALL GRSTR(XX,YY,"SHARP",5,0.08,0.0)
      XX = XX - 0.08
      YY = YY - 0.10
```

```
63:       CALL GRSTR(XX,YY,"CUT-OFF",7,0.08,0.0)
64:
65: *  projection onto column (observation) space of matrix
66: *   zeta observations
67:
68:       IF (PMIN .EQ. 0.0)
69:       .  SOPT = 1
70:       .  XMIN = SVMIN(P,1,JUNK,N)
71:       .  XMAX = SVMAX(P,1,JUNK,N)
72:       ELSE
73:       .  SOPT = 0
74:       .  XMIN = PMIN
75:       .  XMAX = PMAX
76:       END IF
77:
78:       CALL MOVEOO(-2.0,YOFF)
79:       CALL MPRD(WK,M,U,LDU,Y,M,M,M,1)
80:       NAX = 5
81:       CALL PAXIS(0.0,XLEN,0.0,0.0,XMIN,XMAX,4,2,0.12,3,-90.0,SOPT,
82:      &  NAX,1)
83:       XSCALE = XLEN / (XMAX - XMIN)
84:       CALL AXLAB(0.0,XLEN,0.0,0.0,"(S KMn-1o).",0.15,-0.6)
85:       CALL AXLAB(0.0,XOFF+XLEN,0.0,0.0,"PROJECTION ON COLUMN SPACE.",
86:      &  0.15,-0.9)
87:       YMIN = 0.0
88:       YMAX = SVMAX(WK,2,JUNK,N)
89:       NAY = 3
90:       CALL PAXIS(0.0,0.0,0.0,YLEN,YMIN,YMAX,4,0,0.12,2,180.0,1,NAY,1)
91:       CALL AXLAB(0.0,0.0,0.0,YLEN,"(S).",0.15,0.6)
92:       YSCALE = YLEN / (YMAX - YMIN)
93:       CALL GRSTR(2.0,1.0,"ZETA",4,0.1,0.0)
94:       PEN = PENUP
95:       FOR I = 1,N
96:       .  J = 2 * I - 1
97:       .  XX = (P(I) - XMIN) * XSCALE
98:       .  YY = (WK(J) - YMIN) * YSCALE
99:       .  CALL MOVEPN(XX,YY,3,PEN)
100:      .  PEN = PENDOWN
101:      END FOR
102:      FOR I = 1,N
103:      .  J = 2 * I - 1
104:      .  XX = (P(I) - XMIN) * XSCALE
105:      .  YY = (Y(J) - YMIN) * YSCALE
106:      .  CALL GRSYMB(XX,YY,11,0.05,0.0)
107:      END FOR
108:
109: *  tau observations
110:
111:      CALL MOVEOO(XOFF,0.0)
112:      CALL PAXIS(0.0,XLEN,0.0,0.0,XMIN,XMAX,4,2,0.12,3,-90.0,0,NAX,1)
113:      CALL AXLAB(0.0,XLEN,0.0,0.0,"(S KMn-1o).",0.15,-0.6)
114:      YMIN = 0.0
115:      YMAX = SVMAX(WK(2),2,JUNK,N)
116:      NAY = 3
117:      CALL PAXIS(0.0,0.0,0.0,YLEN,YMIN,YMAX,4,0,0.12,2,180.0,1,NAY,1)
118:      CALL AXLAB(0.0,0.0,0.0,YLEN,"(S).",0.15,0.6)
119:      YSCALE = YLEN / (YMAX - YMIN)
120:      CALL GRSTR(2.0,1.0,"TAU",3,0.1,0.0)
121:      PEN = PENUP
122:      FOR I = 1,N
123:      .  J = 2 * I
124:      .  XX = (P(I) - XMIN) * XSCALE
125:      .  YY = (WK(J) - YMIN) * YSCALE
126:      .  CALL MOVEPN(XX,YY,3,PEN)
127:      .  PEN = PENDOWN
128:      END FOR
129:      FOR I = 1,N
130:      .  J = 2 * I
131:      .  XX = (P(I) - XMIN) * XSCALE
132:      .  YY = (Y(J) - YMIN) * YSCALE
133:      .  CALL GRSYMB(XX,YY,11,0.05,0.0)
134:      END FOR
135:
136: *  projection onto row (model) space
137:
138:      CALL MOVEOO(-1.9,YOFF)
139:      CALL MPRD(WK,N,V,LDV,X,N,N,N,1)
140:      CALL PAXIS(0.0,XLEN,0.0,0.0,XMIN,XMAX,4,2,0.12,3,-90.0,0,NAX,1)
141:      CALL AXLAB(0.0,XLEN,0.0,0.0,"(S KMn-1o).",0.15,-0.6)
142:      CALL AXLAB(0.0,XLEN,0.0,0.0,"PROJECTION ON ROW SPACE.",0.15,-0.9)
```

```
143:       YMIN = 0.0
144:       YMAX = SVMAX(WK,1,JUNK,N)
145:       NAY = 3
146:       CALL PAXIS(0.0,0.0,0.0,YLEN,YMIN,YMAX,4,0,0.12,2,180.0,1,NAY,1)
147:       CALL AXLAB(0.0,0.0,0.0,YLEN,"(S).",0.15,0.6)
148:       YSCALE = YLEN / (YMAX - YMIN)
149:       PEN = PENUP
150:       FOR I = 1,N
151:       .  XX = (P(I) - XMIN) * XSCALE
152:       .  YY = (WK(I) - YMIN) * YSCALE
153:       .  CALL MOVEPN(XX,YY,3,PEN)
154:       .  PEN = PENDOWN
155:       END FOR
156:       FOR I = 1,N
157:       .  XX = (P(I) - XMIN) * XSCALE
158:       .  YY = (X(I) - YMIN) * YSCALE
159:       .  CALL GRSYMB(XX,YY,11,0.05,0.0)
160:       END FOR
161: :NOT HCONST
162:
163:       CALL ENDFIG(1)
164:
165:       RETURN
166:       END
```

```
  1: *-h- CLFINV 191  4 MAY 83  22:51:03
  2:       SUBROUTINE CLFINV(ID, LD)
  3:
  4: * close files and terminate plotting
  5:
  6:       INTEGER ID, LD
  7:
  8:       CALL FILCL(ID,"KEEP")
  9:       CALL FILCL(LD,"KEEP")
 10:       CALL ENDPLT
 11:
 12:       RETURN
 13:       END
```

A.5 Data transmission subroutines.

The procedures in this chapter are concerned with the transfer of data to and from disk files. Three out of the four processing programs described in this appendix have output which is to be used by one or more other programs. It is important that data trasmission occur in a regular way. The programs are TPIK. TXCOR and TXFIT and are identified by the three character sequences PIK, COR and FIT. Subroutines that write data are prefixed with W and subroutines that read are prefixed with R. It is important to note that the identification is associated with the program generating the data so W1PIK is used for output in TPIK, but R1PIK is a routine in TXCOR that is used to read the output from TPIK. If there is a primary and a secondary output, these are identified by the first number. If one of the outputs occurs in two stages, these stages are identified by a second number. For example W2FIT1 executes the first stage of the secondary output from TXFIT. The reason for this system is that the input and output activity may be readily identified in the programs. If the format of the output from one of the programs is changed, then only a single input routine needs to be changed.

Another group of procedures in this section concerns the input and output from the H.I.G. format seismic data files. These files have three sections: a file header, the seismic data, and sequence of headers for each data block. The start address of the data for a given shot is stored in its header in the last section. The beginning of the data header section is returned by routine LOCDHD. The location of the

seismic data is retrieved from the appropriate header by routine GETDHD. Routine FILBUF is used to transfer the data from the disk to the program memory. A new data file requires the creation of a file header which is accomplished by INIDMX, the transfer of the data to the file, and the inclusion of the data headers which is accomplished by ENDDMX. Routine COMPRS saves a time window of data from an old file in a new file.

The shot/receiver range and other auxiliary information is stored in a separate file. Routine GETCOR searches this file for a the record containing the data for a given shot. The data will then be decoded from the buffer using the Fortran internal file I/O feature. The plotting of seismic data requires that the starting sample be determined given various parameters, such as reducing velocity and sample rate. Routine STSAMP calculates the starting sample. The seismic traces are scaled so the signal maximum, minimum and mean are required; these are returned by SIGPAR. Travel time picks may be plotted against range with routine PXT which will rescale the data and draw axes if desired. Smoothed travel times may be plotted with PXTSM which will draw a line through the points.

```
 1: *-h- COMPRS 873 18 APR 83  12:29:45
 2:       SUBROUTINE COMPRS(BUF.SHOT.RCV,CH.SB,DS.RATE.SAMP.FS.AS.SD.HD)
 3:
 4: * output shortened buffer to save file
 5:
 6:       INTEGER BUF(*), SHOT. RCV, CH. RATE. SAMP. SD. HD
 7:       INTEGER*6 SB, DS
 8:       REAL FS. AS
 9:       INTEGER HBUF(18), STSAMP. NSAMP. SW, ADR
10:       INTEGER*6 S6W
11:       REAL TIME
12:       CHARACTER INTERNAL*54
13:       EQUIVALENCE (INTERNAL,HBUF)
14:
15: * calculate start time. end time
16:
17:       TIME = FLOAT(SAMP) / FLOAT(RATE) - FLOAT(SB-DS) / 1000.0
18:       DS = DS + INT((TIME-FS)*1000.0)
19:       STSAMP = SAMP - INT(FS*FLOAT(RATE))
20:       NSAMP = INT((AS+FS)*FLOAT(RATE))
21:
22: * encode header and output to temporary file
23:
24:       CALL DSTAT(SD.S6W,ADR)
25:       WRITE (INTERNAL,FMT='(2I6.I2.2I16.I3.2X,A3)') RCV,SHOT.CH.SB.DS.
26:      &  RATE.ADR
27:       BUFFER OUT (HD.HBUF.B,18,SW)
28:       CALL STATUS(HD)
29:
30: * output data
31:
32:       CALL BUFOUT(SD.BUF(STSAMP).NSAMP.SW)
33:
34:       RETURN
35:       END
```

```
 1: *-h- ENDDMX 681 30 JUN 82  10:16:48
 2:       SUBROUTINE ENDDMX(DD, HD. HBUF. NSAMP)
 3:
 4: * append headers to end of data file and close
 5:
 6:       INTEGER DD. HD. NSAMP. HBUF(112)
 7:       INTEGER HDRADR. SW
 8:       INTEGER*6 S6W
 9:
10:       CALL DSTAT(DD.S6W,HDRADR)
11:
12:       ENDFILE HD
13:       REWIND HD
14:
15:       CALL DPOS(DD.2)
16:       CALL BUFIN(DD.HBUF,112.SW)
17:       HBUF(111) = NSAMP
18:       HBUF(112) = HDRADR
19:       CALL DPOS(DD.2)
20:       CALL BUFOUT(DD.HBUF.112.SW)
21:
22:       CALL DPOS(DD,HDRADR)
23:       LOOP
24:       .  BUFFER IN (HD.HBUF.B,18,SW)
25:       .  CALL STATUS(HD)
26:       .  EXIT LOOP IF (SW .GE. 3)
27:       .  CALL BUFOUT(DD,HBUF.18.SW)
28:       END LOOP
29:       ENDFILE DD
30:
31:       CLOSE (UNIT=DD)
32:       CLOSE (UNIT=HD.STATUS='DELETE')
33:       RETURN
34:       END
```

```
 1: *-h- FILBUF 319  4 AUG 82  23:38:47
 2:       INTEGER FUNCTION FILBUF(D. BUF. N. ADR)
 3:
 4: * fill buffer with shot data from standard HIG demultiplex file
 5:
 6:       INTEGER D. BUF(*), N. ADR
 7:       INTEGER SW
```

```
 8:
 9:       CALL DPOS(D,ADR)
10:       CALL BUFIN(D,BUF,N,SW)
11:       IF (SW .GE. 3)
12:       .  FILBUF = -3
13:       ELSE
14:       .  FILBUF = -2
15:       END IF
16:
17:       RETURN
18:       END
```

```
 1: *-h- FLTTIM 301  2 MAY 83  22:49:19
 2:       SUBROUTINE FLTTIM(X, F1, F2, NPOLES, GAIN, N)
 3:
 4: * apply n-pole filter to real array x
 5:
 6:       INTEGER NPOLES, N
 7:       REAL X(*), F1, F2, GAIN
 8:       INTEGER ORDER
 9:
10:       ORDER = NPOLES / 2
11:
12:       CALL FILT(X,N,ORDER,1)
13:       T = 1.0 / GAIN
14:       CALL VSMA1(X,1,T,X,1,0.0,N)
15:
16:       RETURN
17:       END
```

```
 1: *-h- GETCOR 1278 26 MAY 83  22:21:30
 2:       INTEGER FUNCTION GETCOR(R, S, BUF, D)
 3:
 4: * find corfile buffer containing data for shot s
 5: *
 6: *     R    input  - integer receiver number (not matched if r=0)
 7: *     S    input  - integer shot number
 8: *     BUF  output - buffer containing corfile data
 9: *     D    input  - logical unit of corfile
10:
11:       INTEGER R, S, BUF(*), D
12:       INTEGER I, IOSW, M, N, STR(13)
13:       INTEGER CTOI
14:
15:       BUFFER IN (D, BUF, B, 140, IOSW)
16:       CALL STATUS(D)
17:       IF (IOSW .GE. 3)
18:       .  REWIND D
19:       .  BUFFER IN (D, BUF, B, 140, IOSW)
20:       .  CALL STATUS(D)
21:       .  IF (IOSW .GE. 3)
22:       .  .  GETCOR = -3
23:       .  .  RETURN
24:       .  END IF
25:       END IF
26:       CALL UNPACK(BUF,STR,4)
27:       STR(13) = -2
28:
29:       I = 1
30:       M = CTOI(STR,I)
31:       N = CTOI(STR,I)
32:       IF ((M .EQ. R .AND. N .EQ. S) .OR. (R .EQ. 0 .AND. N .EQ. S))
33:       .  GETCOR = -2
34:       .  RETURN
35:       END IF
36:
37:       IF (M .GE. R .AND. N .GT. S)
38:      &  REWIND D
39:
40:       LOOP
41:       .  BUFFER IN (D, BUF, B, 140, IOSW)
42:       .  CALL STATUS(D)
43:       .  IF (IOSW .GE. 3)
44:       .  .  GETCOR = -3
45:       .  .  RETURN
46:       .  END IF
47:       .  CALL UNPACK(BUF,STR,4)
48:       .  STR(13) = -2
```

```
49:       .  I = 1
50:       .  M = CTOI(STR.I)
51:       .  N = CTOI(STR.I)
52:       .  EXIT LOOP IF ((M .EQ. R .AND. N .EO. S) .OR.
53:      &.    (R .EQ. 0 .AND. N .EO. S))
54:       END LOOP
55:
56:       GETCOR = -2
57:       RETURN
58:       END


 1: *-h- GETDHD 592 30 JUN 82  10:16:50
 2:       INTEGER FUNCTION GETDHD(D. HAD. SHOT. RCV, CH. SB, DS. RATE. DAD)
 3:
 4: * return address of data for required shot
 5:
 6:       INTEGER D. HAD. SHOT. RCV, CH. RATE. DAD
 7:       INTEGER*6 SB, DS
 8:       INTEGER BUF(18). SW
 9:       CHARACTER*54 INTERNAL
10:       EOUIVALENCE (INTERNAL,BUF)
11:
12:       CALL DPOS(D,HAD)
13:
14:       DO
15:       .  CALL BUFIN(D,BUF,18.SW)
16:       .  IF (SW .GE. 3)
17:       .  .  GETDHD = -3
18:       .  .  RETURN
19:       .  END IF
20:       .  READ (INTERNAL,FMT='(6X,I6)') N
21:       UNTIL (N .EO. SHOT)
22:       READ (INTERNAL,FMT='(I6.6X,I2.2I16.I3.2X,A3)') RCV,CH.SB.DS.RATE.
23:      & DAD
24:       GETDHD = -2
25:
26:       RETURN
27:       END


 1: *-h- GFLTTM 1148  2 MAY 83  22:49:25
 2:       SUBROUTINE GFLTTM(F1. F2. NPOLES. GAINMAX. D. HDRADR)
 3:
 4: * generate time domain filter
 5:
 6:       INTEGER NPOLES. D. HDRADR
 7:       REAL F1. F2. GAINMAX
 8:       REAL A0. A1(30). A2(30). B1(30). B2(30)
 9:       COMMON /BUTT/ A1. A2. B1. B2
10:       INTEGER NYOFREO. ORDER. SMPRATE. HBUF(18). SW, JUNK
11:       REAL GAIN(100). PHASE(100). F1C, F2C, FCC
12:       COMPLEX ALPHA(30). BETA(30)
13:       CHARACTER*54 INTERNAL
14:       EOUIVALENCE (HBUF.INTERNAL)
15:
16: *  get sample rate from data file headers
17:
18:       CALL DPOS(D,HDRADR)
19:       CALL BUFIN(D,HBUF.18,SW)
20:       READ (INTERNAL.FMT='(T47,I3)') SMPRATE
21:
22:       F1C = F1 / FLOAT(SMPRATE)
23:       F2C = F2 / FLOAT(SMPRATE)
24:       FCC = F2C - F1C
25:       ORDER = NPOLES / 2
26:       NYQFREO = MIN0(SMPRATE/2.100)
27:
28: * compute prototype low pass filter
29:
30:       CALL BUTTER(ORDER.FCC.ALPHA.BETA)
31:
32: * modify to band-pass if necessary
33:
34:       IF (F1 .GT. 0.0) CALL BPASS(ORDER.F1C,F2C,ALPHA.BETA)
35:
36: * compute serial realization in terms of 2-pole. 2-zero filters
37:
38:       CALL COEFF(ORDER/2.ALPHA.BETA.A1.A2.B1.B2.A0)
39:       CALL FREO(ORDER/2.NYQFREO.A1.A2.B1.B2.A0.GAIN.PHASE)
```

```
40:
41:       GAINMAX = SVMAX(GAIN,1,JUNK,NYQFREQ)
42:
43:       RETURN
44:       END
```

```
 1: *-h- INIDMX 266 30 JUN 82  10:16:53
 2:       SUBROUTINE INIDMX(F, BUF)
 3:
 4: * write 3 blank records to set up seismic data file
 5:
 6:       INTEGER F, BUF(112)
 7:       INTEGER SW
 8:
 9:       CALL BUFOUT(F,BUF,112,SW)
10:       CALL BUFOUT(F,BUF,112,SW)
11:       CALL BUFOUT(F,BUF,112,SW)
12:       ENDFILE F
13:
14:       RETURN
15:       END
```

```
 1: *-h- LOCDHD 275 30 JUN 82  10:16:52
 2:       INTEGER FUNCTION LOCDHD(D, BUF, N)
 3:
 4: * find start address of headers in standard HIG seismic data file
 5:
 6:       INTEGER D, BUF(112), N
 7:       INTEGER SW
 8:
 9:       CALL DPOS(D,2)
10:       CALL BUFIN(D,BUF,112,SW)
11:       N = BUF(111)
12:
13:       LOCDHD = BUF(112)
14:       RETURN
15:       END
```

```
 1: *-h- PXT 1144  6 JAN 83  15:21:38
 2:       SUBROUTINE PXT(X, Y, RV, N, XMIN, XSCALE, YMIN, YSCALE)
 3:
 4: *  plot data points, calculate plot parameters
 5:
 6:       INTEGER N
 7:       REAL X(*), Y(*), RV, XSCALE, YSCALE
 8:       REAL XLEN, XMAX, XMIN, YLEN, YMAX, YMIN
 9:       INTEGER JUNK, M1, M2, NN
10:       REAL SVMAX, SVMIN
11:       DATA XLEN /8.0/, YLEN /4.0/
12:
13:
14:       NN = IABS(N)
15:       XMIN = X(1)
16:       XMAX = X(NN)
17:
18:       IF (RV .NE. 0.0) CALL VSMA2(Y,1,1.0,Y,1,-1.0/RV,X,1,0,NN)
19:       YMAX = SVMAX(Y,1,JUNK,NN)
20:       YMIN = SVMIN(Y,1,JUNK,NN)
21:
22:       IF (N .GT. 0)
23:       .  M1 = NINT(XMAX)
24:       .  M2 = 4
25:       .  CALL PAXIS(0.0,XLEN,0.0,0.0,XMIN,XMAX,3,-1.0,15,3.-90.0,1,M1,5)
26:       .  CALL PAXIS(0.0,0.0,0.0,YLEN,YMIN,YMAX,5,2.0,15,2.180.0,1,M2,1)
27:       .  CALL PAXIS(0.0,XLEN,YLEN,YLEN,XMIN,XMAX,0.0,0.15,0.90,0.0,M1,5)
28:       .  CALL PAXIS(XLEN,XLEN,0.0,YLEN,YMIN,YMAX,0.0,0.15,0.0,0.0,M2,1)
29:       END IF
30:
31:       XSCALE = XLEN / (XMAX - XMIN)
32:       YSCALE = YLEN / (YMAX - YMIN)
33:
34:       FOR I = 1,NN
35:       .  XX = XSCALE * (X(I) - XMIN)
36:       .  YY = YSCALE * (Y(I) - YMIN)
37:       .  CALL SYMBOL(XX,YY,0,1,42,0.0,-1)
38:       END FOR
39:
```

```
40:       IF (RV .NE. 0.0) CALL VSMA2(Y,1,1.0,Y,1,1.0/RV,X,1,0.0,NN)
41:
42:       RETURN
43:       END


 1: *-h- PXTSM 393  6 JAN 83  15:21:40
 2:       SUBROUTINE PXTSM(X, Y, RV, N, XMIN, XSCALE, YMIN, YSCALE)
 3:
 4: *  plot fitted curve on frame previously drawn
 5:
 6:       INTEGER N
 7:       REAL X(*), Y(*), RV, XSCALE, XMIN, YSCALE, YMIN
 8:
 9:       IF (RV .NE. 0.0) CALL VSMA2(Y,1,1.0,Y,1,-1.0/RV,X,1,0.0,N)
10:       CALL PVEC2(X,Y,N,0.0,0.0,XMIN,XSCALE,YMIN,YSCALE)
11:       IF (RV .NE. 0.0) CALL VSMA2(Y,1,1.0,Y,1,1.0/RV,X,1,0.0,N)
12:
13:       RETURN
14:       END


 1: *-h- R1COR 546 12 MAY 83  15:05:47
 2:       INTEGER FUNCTION R1COR(S, X, DX, T, DT, MAXDAT, D)
 3:
 4: * read data output by TXCOR
 5:
 6:       INTEGER S(*), MAXDAT, D
 7:       REAL X(*), DX(*), T(*), DT(*)
 8:       INTEGER I, LIN(82), N
 9:       INTEGER CTOI, GETLIN
10:       REAL CTOF
11:
12:       N = 0
13:       WHILE (GETLIN(LIN,D) .NE. -1)
14:       .  N = N + 1
15:       .  IF (N .GT. MAXDAT) CALL ERROR(" r1cor: too many data")
16:       .  I = 1
17:       .  S(N) = CTOI(LIN,I)
18:       .  X(N) = CTOF(LIN,I)
19:       .  DX(N) = CTOF(LIN,I)
20:       .  T(N) = CTOF(LIN,I)
21:       .  DT(N) = CTOF(LIN,I)
22:       END WHILE
23:
24:       R1COR = N
25:       RETURN
26:
27:       END


 1: *-h- R2FIT1 783 19 MAY 83  23:49:54
 2:       INTEGER FUNCTION R2FIT1(K, Y, C, LDC, MAXDATA, D)
 3:
 4: * read spline parameters of fitted curve (beginning marked by '@')
 5:
 6:       INTEGER LDC, MAXDATA, D
 7:       REAL K(*), Y(*), C(LDC,*)
 8:       INTEGER STR(2), PAT(81), LIN(82), N
 9:       INTEGER GETLIN, GETPAT, MATCH
10:       DATA STR /64,-2/
11:
12:       IF (GETPAT(STR,PAT) .EQ. -3) CALL ERROR(" illegal pattern")
13:       WHILE (GETLIN(LIN,D) .NE. -1)
14:       .  EXIT WHILE IF (MATCH(LIN,PAT) .EQ. 1)
15:       END WHILE
16:       BACKSPACE D
17:
18:       N = 1
19:       LOOP
20:       .  IF (N .GT. MAXDATA)
21:      &.     CALL ERROR(" r2fit1:  too many data")
22:       .  READ (D,FMT='(5F11.0)') K(N), Y(N), (C(N,I),I=1,3)
23:       .  EXIT LOOP IF (K(N) .EQ. 0.0 .AND. Y(N) .EQ. 0.0 .AND.
24:      &.     C(N,1) .EQ. 0.0)
25:       .  N = N + 1
26:       END LOOP
27:
```

```
      R2FIT1 = N - 1
      RETURN
      END
```

```
*-h- R2FIT2 712 19 MAY 83  23:49:47
      INTEGER FUNCTION R2FIT2(X, T, S, OX, MAXDATA, D)

* read values of fitted curve (beginning of data marked by '#')

      INTEGER MAXDATA, D
      REAL X(*), T(*), S(*), OX(*)
      INTEGER STR(2), PAT(81), LIN(82), N
      INTEGER GETLIN, GETPAT, MATCH
      DATA STR /35,-2/

      IF (GETPAT(STR,PAT) .EQ. -3) CALL ERROR(" illegal pattern")
      WHILE (GETLIN(LIN,D) .NE. -1)
      .  EXIT WHILE IF (MATCH(LIN,PAT) .EQ. 1)
      END WHILE

      N = 1
      LOOP
      .  IF (N .GT. MAXDATA)
     &.     CALL ERROR(" r2fit2: too many data")
      .  READ (D,FMT='(T6,3F11.0,22X,F11.0)',END=1) X(N), T(N), S(N),
     &.     OX(N)
      .  N = N + 1
      END LOOP
    1 CONTINUE

      R2FIT2 = N - 1
      RETURN
      END
```

```
*-h- RPIK 1353 26 MAY 83  22:09:36
      INTEGER FUNCTION RPIK(S, X, T, DT, LD, R, SS, ES, JOB, MAXDAT, D)

* read data output by TPIK
*  JOB controls which data is to be read

      INTEGER S(*), R(*), SS, ES, JOB, MAXDAT, D
      REAL X(*), T(LD,*), DT(LD,*)
      INTEGER C1, C2, C3, C4, C5, I, LIN(82), N
      INTEGER CTOI, GETLIN
      REAL CTOF

*                                        ! shot number is alway read
      C1 = JOB / 10000                   ! range
      C2 = MOD(JOB/1000,10)              ! first pick and error
      C3 = MOD(JOB/100,10)               ! second pick and error
      C4 = MOD(JOB/10,10)                ! third pick and error
      C5 = MOD(JOB,10)                   ! receiver number

      N = 1
      WHILE (GETLIN(LIN,D) .NE. -1)
      .  IF (N .GT. MAXDAT) CALL ERROR(" rpik: too many data")
      .  I = 1
      .  S(N) = CTOI(LIN,I)
      .  IF (S(N) .GE. SS .AND. S(N) .LE. ES)
      .  .  IF (C1 .EQ. 1) X(N) = CTOF(LIN,I)
      .  .  I = 16
      .  .  IF (C2 .EQ. 1)
      .  .  .  T(N,1) = CTOF(LIN,I)
      .  .  .  DT(N,1) = CTOF(LIN,I)
      .  .  END IF
      .  .  I = 34
      .  .  IF (C3 .EQ. 1)
      .  .  .  T(N,2) = CTOF(LIN,I)
      .  .  .  DT(N,2) = CTOF(LIN,I)
      .  .  END IF
      .  .  I = 52
      .  .  IF (C4 .EQ. 1)
      .  .  .  T(N,3) = CTOF(LIN,I)
      .  .  .  DT(N,3) = CTOF(LIN,I)
      .  .  END IF
      .  .  I = 70
      .  .  IF (C5 .EQ. 1) R(N) = CTOI(LIN,I)
      .  .  N = N + 1
```

```
45:       . END IF
46:       END WHILE
47:
48:       RPIK = N - 1
49:       RETURN
50:
51:       END
```

```
 1: *-h- SIGPAR 404 19 APR 83  14:16:10
 2:       SUBROUTINE SIGPAR(DATA. N. SMAX. SMIN. SMEAN)
 3:
 4: * calculate max, min and mean of vector
 5:
 6:       INTEGER N
 7:       REAL DATA(*). SMAX. SMIN. SMEAN
 8:       INTEGER I
 9:
10:       SMAX = DATA(1)
11:       SMIN = DATA(1)
12:       SMEAN = DATA(1) / FLOAT(N)
13:
14:       FOR I = 2.N
15:       .  SMAX = AMAX1(SMAX,DATA(I))
16:       .  SMIN = AMIN1(SMIN,DATA(I))
17:       .  SMEAN = SMEAN + DATA(I) / FLOAT(N)
18:       END FOR
19:
20:       RETURN
21:       END
```

```
 1: *-h- STSAMP 504 17 MAR 83  13:46:31
 2:       INTEGER FUNCTION STSAMP(DS. SB, ST. RANGE. RV, RATE)
 3:
 4: *  calculate sample number to start plotting. if rv=0 then start at sb
 5:
 6:       INTEGER RATE
 7:       INTEGER*6 DS. SB
 8:       REAL ST. RANGE. RV
 9:       REAL RT
10:
11: *  check for bad unacceptable times
12:
13:       IF (IABS(SB-DS) .GT. 8388607D)
14:       .  STSAMP = '77777777
15:       .  RETURN
16:       END IF
17:
18:       IF (RV .EO. 0.0)
19:       .  RT = 0.0
20:       ELSE
21:       .  RT = RANGE / RV
22:       END IF
23:
24:       STSAMP = NINT(((REAL(SB-DS)/1000.0)+ST+RT)*RATE)
25:       RETURN
26:
27:       END
```

```
 1: *-h- W1COR 300 13 MAR 83  22:17:54
 2:       SUBROUTINE W1COR(S. X. DX. T. DT. D)
 3:
 4: * output routine for TXCOR
 5:
 6:       INTEGER S. D
 7:       REAL X. DX. T. DT
 8:
 9:       CALL PUTDEC(S.5.D)
10:       CALL PUTFLT(X,10.3.D)
11:       CALL PUTFLT(DX,8,3.D)
12:       CALL PUTFLT(T,10 3.D)
13:       CALL PUTFLT(DT,8.3.D)
14:       CALL PUTCH(10,D)
15:
16:       RETURN
17:       END
```

```
 1: *-h- W1FIT 448 12 MAY 83  15:06:05
 2:       SUBROUTINE W1FIT(P, Z, T. DZ, DT, N. D)
 3:
 4: * output new parameters for LSVZ
 5:
 6:       INTEGER N. D
 7:       REAL P(*), Z(*), T(*), DZ(*), DT(*)
 8:
 9:       CALL PUTCH(10.D)
10:       CALL PUTCH(10.D)
11:       CALL PUTCH(10.D)
12:       FOR I = 1.N
13:       .  CALL PUTFLT(P(I),10.5.D)
14:       .  CALL PUTFLT(Z(I),10.5.D)
15:       .  CALL PUTFLT(T(I),10.5.D)
16:       .  CALL PUTFLT(DZ(I),10.5.D)
17:       .  CALL PUTFLT(DT(I),10.5.D)
18:       .  CALL PUTCH(10.D)
19:       END FOR
20:
21:       RETURN
22:       END


 1: *-h- W2FIT1 513 19 MAY 83  23:49:52
 2:       SUBROUTINE W2FIT1(K, Y. C. LDC. E. N. D)
 3:
 4: *  list knot locations and spline coefficients
 5:
 6:       INTEGER LDC, N. D
 7:       REAL K(1). Y(1). C(LDC,1). E
 8:       INTEGER I
 9:
10:       WRITE (D,"(' '.80(' '))")
11:       WRITE (D,"(' [',6X,'K',9X,'Y',14X,'C[1]',7X,'C[2]',7X,
12:      &  'C[3]',21X,']')")
13:
14:       WRITE (D,"('+',80(' '))")
15:       WRITE (D,"('@')")
16:       FOR I = 1.N
17:       .  WRITE (D,"(F11.2.F11.3.5X.3F11.3)") K(I).Y(I).(C(I.J).J=1.3)
18:       END FOR
19:       WRITE (D,"(/' Rms error'.F8.3/)") E
20:
21:       RETURN
22:       END


 1: *-h- W2FIT2 511  1 MAR 83  21:23:35
 2:       SUBROUTINE W2FIT2(SHOT. X, Y. S. P. Q, N. D)
 3:
 4: *  list data and smoothed fit
 5:
 6:       INTEGER SHOT(1). N. D
 7:       REAL X(1). Y(1), S(1). P(1). Q(1)
 8:       INTEGER I
 9:
10:       WRITE (D,FMT="(' '.80(' '))")
11:       WRITE (D,FMT="(' [',11X,'X',10X,'T'.10X,'S'.9X,'ER'.10X,'P'.9X,
12:      &  'OX'.11X,']')")
13:       WRITE (D,FMT="('+'.80(' ')/)")
14:
15:       WRITE (D,FMT="('#')")
16:       FOR I = 1.N
17:       .  WRITE (D,FMT="(I5.6F11.3)") SHOT(I). X(I). Y(I). S(I). Y(I)-S(I).
18:      &.    P(I). Q(I)
19:       END FOR
20:
21:       RETURN
22:       END


 1: *-h- W1PIK 443 20 APR 83  21:51:16
 2:       SUBROUTINE W1PIK(S. X. T. DT. INDEX. R. D)
 3:
 4: * output routine for TPIK
 5:
 6:       INTEGER S. INDEX(3). R. D
 7:       REAL X. T(3). DT(3)
 8:       INTEGER I
```

```
 9:
10:       CALL PUTDEC(S.5.D)
11:       CALL PUTFLT(X,10.3.D)
12:       FOR I = 1.3
13:       .  IF (INDEX(I) .EQ. 1)
14:       .  .  CALL PUTFLT(T(I),10.3.D)
15:       .  .  CALL PUTFLT(DT(I),8.3.D)
16:       .  ELSE
17:       .  .  CALL PUTTAB(D)
18:       .  END IF
19:       END FOR
20:       CALL PUTDEC(R.5.D)
21:       CALL PUTCH(10.D)
22:
23:       RETURN
24:       END
```

A.6 Text processing subroutine package.

Almost all the routines in this chapter are copied. adapted from or suggested by Kernighan and Plauger [1978]. Those requiring information on them beyond what is supplied here are advised to refer to the book.

Fortran does not handle textual information gracefully. Character variables have been introduced with the 77 standard [ANS X3.9, 1978] but their use is still very limiting. The routines in Kernighan and Plauger store characters and strings in integer arrays as their ASCII equivalents. This is wasteful of storage because two out of three bytes in each integer element are unused, but it allows very flexible manipulation.

Character data is read into the programs with routines GETLIN for a line of text or GETCH for a single character. These routines convert the data to internal form for use by the other routines. Output is handled ultimately by routines PUTLIN and PUTCH which convert back to external format. PUTCH and PUTLIN are more useful than the WRITE statement of Fortran in that characters and strings can be copied into the output buffer at any point in the program but are not output to the external device until the special "newline" character is transmitted. Routines REMARK and ERROR are used for diagnostic messages, ERROR causes the program to stop. Both take quoted strings as arguments. They convert the character strings to internal format then use PUTCH to output the message to interactive terminal. Numeric data are output by PUTFLT

and PUTDEC for real data and integer data respectively. Conversion between numeric data and character data is handled by CTOF. CTOI, CTOI2, FTOC and ITOC.

As stated previously, data connections are made internally. Routines GETARG and GETOPT interpret the command line. GETARG separates out the individual arguments which are typically file names and converts them to internal format. GETOPT searches for arguments that are flagged by the character "/" and interprets each subsequent letter as an option. Existing files are connected by routine FILOP. This routine returns an integer which is the logical connection between the file and the program. It then increments an internal variable so that the next file connection is made with a different number. If the operation is not succesful, FILOP returns an error code. New files can be created and connected with FILCR for symbolic data and FILCRB for binary data. They are analogous to FILOP. The Harris VOS operating system uses logical units between 0 and 10 and above 200 so these are restricted from the program. Routine PRIMIO is used to start the numbering system, ususally at 11. Routine FILCL breaks the logical connection to a file.

The only other routines that are called directly by the processing programs are MPRINT which prints the elements of a matrix in a format supplied by the user; and MATCH which searches the input file for a given text pattern. MATCH is used by R2FIT1 and R2FIT2 to advance the input to the required data. The other routines in this package are lower level routines that are called by those just described. They will not be described and are included for completeness.

```
 1: *-h- ADDSET 287  2 OCT 81  14:58:21
 2:       INTEGER FUNCTION ADDSET(C, SET, J, MAXSIZ)
 3:
 4: * put c in set(j) if it fits, increment j
 5:
 6:       INTEGER J, MAXSIZ
 7:       INTEGER C, SET(MAXSIZ)
 8:
 9:       IF (J .GT. MAXSIZ)
10:       .  ADDSET = 0
11:       ELSE
12:       .  SET(J) = C
13:       .  J = J + 1
14:       .  ADDSET = 1
15:       END IF
16:
17:       RETURN
18:       END
```

```
 1: *-h- AMATCH 1291  2 OCT 81  14:58:22
 2:       INTEGER FUNCTION AMATCH(LIN, FROM, PAT)
 3:
 4: * look for match starting at lin(from)
 5:
 6:       INTEGER LIN(82), PAT(81)
 7:       INTEGER OMATCH, PATSIZ
 8:       INTEGER FROM, I, J, OFFSET, STACK
 9:
10:       STACK = 0
11:       OFFSET = FROM                    ! next unexamined input character
12:       J = 1
13:       WHILE (PAT(J) .NE. -2)
14:       .  IF (PAT(J) .EQ. 42)             ! a closure entry
15:       .  .  STACK = J
16:       .  .  J = J + 4                       ! step over CLOSURE
17:       .  .  I = OFFSET
18:       .  .  WHILE (LIN(I) .NE. -2)          ! match as many as possible
19:       .  .  .  EXIT WHILE IF (OMATCH(LIN,I,PAT,J) .EQ. 0)
20:       .  .  END WHILE
21:       .  .  PAT(STACK+1) = I - OFFSET
22:       .  .  PAT(STACK+3) = OFFSET
23:       .  .  OFFSET = I                        ! character that caused failure
24:       .  ELSE IF (OMATCH(LIN,OFFSET,PAT,J) .EQ. 0)   ! non-closure
25:       .  .  WHILE (STACK .GT. 0)
26:       .  .  .  EXIT WHILE IF (PAT(STACK+1) .GT. 0)
27:       .  .  .  STACK = PAT(STACK+2)
28:       .  .  END WHILE
29:       .  .  IF (STACK .LE. 0)                ! stack is empty
30:       .  .  .  AMATCH = 0
31:       .  .  .  RETURN                           ! unsuccesfully
32:       .  .  END IF
33:       .  .  PAT(STACK+1) = PAT(STACK+1) - 1
34:       .  .  J = STACK + 4
35:       .  .  OFFSET = PAT(STACK+3) + PAT(STACK+1)
36:       .  END IF
37:       .  J = J + PATSIZ(PAT,J)
38:       END WHILE
39:       AMATCH = OFFSET
40:
41:       RETURN                             ! succesfully
42:       END
```

```
 1: *-h- CANT 188 17 APR 83  23:53:55
 2:       SUBROUTINE CANT(NAME)
 3:
 4: * output message: can't open
 5:
 6:       INTEGER NAME(18)
 7:
 8:       CALL PUTCH(32,3)
 9:       CALL PUTLIN(NAME,3)
10:       CALL REMARK(": can't open")
11:
12:       STOP
13:       END
```

```
 1: *-h- CTOF 1006 25 JUN 82  12:13:38
 2:       REAL FUNCTION CTOF(STR, I)
 3:
 4: * convert character string to real number
 5:
 6:       INTEGER STR(1)
 7:       INTEGER D, DIGITS(11), I, S
 8:       REAL VAL, POWER
 9:       INTEGER INDEX
10:       EXTERNAL INDEX
11:       DATA DIGITS /48,49,50,51,52,53,54,55,56,57,-2/
12:
13: *  skip white space
14:
15:       WHILE (STR(I) .EQ. 32 .OR. STR(I) .EQ. 9 .AND. STR(I) .NE. -2)
16:       .  I = I + 1
17:       END WHILE
18:
19: *  determine sign; skip over
20:
21:       S = 1
22:       IF (STR(I) .EQ. 43)
23:       .  I = I + 1
24:       ELSE IF (STR(I) .EQ. 45)
25:       .  S = -1
26:       .  I = I + 1
27:       END IF
28:
29: *  whole part of number
30:
31:       VAL = 0
32:       WHILE (STR(I) .LE. 57 .AND. STR(I) .GE. 48)
33:       .  D = INDEX(DIGITS,STR(I))
34:       .  VAL = 10 * VAL + D - 1
35:       .  I = I + 1
36:       END WHILE
37:
38:       IF (STR(I) .EQ. 46) I = I + 1
39:
40: *  fractional part
41:
42:       POWER = 1
43:       WHILE (STR(I) .LE. 57 .AND. STR(I) .GE. 48)
44:       .  D = INDEX(DIGITS,STR(I))
45:       .  VAL = 10 * VAL + D - 1
46:       .  POWER = POWER * 10
47:       .  I = I + 1
48:       END WHILE
49:
50:       CTOF = S * VAL / POWER
51:
52:       RETURN
53:       END
```

```
 1: *-h- CTOI 704  5 FEB 82  10:34:30
 2:       INTEGER FUNCTION CTOI(IN, I)
 3:
 4: * convert string at in(i) to integer, increment i
 5:
 6:       INTEGER IN(81), DIGITS(11)
 7:       INTEGER D, I, S, BLANK, EOS
 8:       INTEGER INDEX
 9:       EXTERNAL INDEX
10:
11:       DATA BLANK/32/, EOS/-2/
12:       DATA DIGITS /48,49,50,51,52,53,54,55,56,57,-2/
13:
14:       WHILE (IN(I) .EQ. BLANK)
15:       .  I = I + 1
16:       END WHILE
17:
18:       S = 1
19:       IF (IN(I) .EQ. 43)
20:       .  I = I + 1
21:       ELSE IF (IN(I) .EQ. 45)
22:       .  S = -1
23:       .  I = I + 1
24:       END IF
25:
```

```
26:       CTOI = 0
27:       WHILE (IN(I) .NE. EOS)
28:       .  D = INDEX(DIGITS, IN(I))
29:       .  IF (D .EQ. 0)
30:       .  .  EXIT WHILE
31:       .  END IF
32:       .  CTOI = 10 * CTOI + D - 1
33:       .  I = I + 1
34:       END WHILE
35:       CTOI = S * CTOI
36:
37:       RETURN
38:       END
```

```
 1: *-h- CTOI2 686 10 FEB 83  18:01:49
 2:       INTEGER FUNCTION CTOI2(IN, I, DIGITS)
 3:
 4: * convert string at in(i) to integer, increment i
 5: *  version 2 - arbitrary digit set
 6:
 7:       INTEGER IN(1), I, DIGITS(1)
 8:       INTEGER B, D, S
 9:       INTEGER INDEX, LENGTH
10:       EXTERNAL INDEX
11:
12:       B = LENGTH(DIGITS)
13:       WHILE (IN(I) .EQ. 32)
14:       .  I = I + 1
15:       END WHILE
16:
17:       S = 1
18:       IF (IN(I) .EQ. 43)
19:       .  I = I + 1
20:       ELSE IF (IN(I) .EQ. 45)
21:       .  S = -1
22:       .  I = I + 1
23:       END IF
24:
25:       CTOI2 = 0
26:       WHILE (IN(I) .NE. -2)
27:       .  D = INDEX(DIGITS, IN(I))
28:       .  IF (D .EQ. 0)
29:       .  .  EXIT WHILE
30:       .  END IF
31:       .  CTOI2 = B * CTOI2 + D - 1
32:       .  I = I + 1
33:       END WHILE
34:       CTOI2 = S * CTOI2
35:
36:       RETURN
37:       END
```

```
 1: *-h- DODASH 502  2 OCT 81  15:02:43
 2:       SUBROUTINE DODASH(VALID, ARRAY, I, SET, J, MAXSET)
 3:
 4: * expand array(i-1)-array(i+1) into set(j)... from valid
 5:
 6:       INTEGER ESC
 7:       INTEGER ADDSET, INDEX
 8:       EXTERNAL INDEX
 9:       INTEGER I, J, JUNK, K, LIMIT, MAXSET
10:       INTEGER ARRAY(1), SET(MAXSET), VALID(1)
11:
12:       I = I + 1
13:       J = J - 1
14:       LIMIT = INDEX(VALID, ESC(ARRAY,I))
15:       K = INDEX(VALID,SET(J))
16:       WHILE (K .LE. LIMIT)
17:       .  JUNK = ADDSET(VALID(K),SET,J,MAXSET)
18:       .  K = K + 1
19:       END WHILE
20:
21:       RETURN
22:       END
```

```
 1: *-h- EQUAL 368  2 OCT 81  15:19:29
 2:       INTEGER FUNCTION EQUAL(STR1, STR2)
 3:
 4: * compare 'str1' to 'str2'; return YES if equal, NO if not
 5:
 6:       INTEGER STR1(100), STR2(100), I
 7:       INTEGER EOS
 8:       DATA EOS /-2/
 9:
10:       I = 1
11:       WHILE (STR1(I) .EQ. STR2(I))
12:       .  IF (STR1(I) .EQ. EOS)
13:       .  .  EQUAL = 1
14:       .  .  RETURN
15:       .  END IF
16:       .  I = I + 1
17:       END WHILE
18:
19:       EQUAL = 0
20:       RETURN
21:       END
```

```
 1: *-h- ERROR 131 11 APR 83  12:00:53
 2:       SUBROUTINE ERROR(MSG)
 3:
 4: * write error message and stop
 5:
 6:       CHARACTER MSG*(*)
 7:
 8:       CALL REMARK(MSG)
 9:       STOP
10:
11:       END
```

```
 1: *-h- ESC 707 19 FEB 83  17:34:46
 2:       INTEGER FUNCTION ESC(ARRAY, I)
 3:
 4: * map array(i) into escaped character if possible
 5:
 6:       INTEGER ARRAY(1), I
 7:       INTEGER DIGITS(9)
 8:       INTEGER CTOI2, TYPE
 9:       DATA DIGITS /48,49,50,51,52,53,54,55,-2/
10:
11:       IF (ARRAY(I) .NE. 64)
12:       .  ESC = ARRAY(I)
13:       ELSE IF (ARRAY(I+1) .EQ. -2)     !  @ not special at end
14:       .  ESC = 64
15:       ELSE
16:       .  I = I + 1
17:       .  IF (TYPE(ARRAY(I)) .EQ. 0)    !  octal ASCII character code
18:       .  .  ESC = CTOI2(ARRAY,I,DIGITS)
19:       .  .  I = I - 1
20:       .  ELSE IF (ARRAY(I) .EQ. 66)
21:       .  .  ESC = 8
22:       .  ELSE IF (ARRAY(I) .EQ. 78)
23:       .  .  ESC = 10
24:       .  ELSE IF (ARRAY(I) .EQ. 84)
25:       .  .  ESC = 9
26:       .  ELSE
27:       .  .  ESC = ARRAY(I)
28:       .  END IF
29:       END IF
30:
31:       RETURN
32:       END
```

```
 1: *-h- FILCR 733 17 APR 83  23:54:02
 2:       INTEGER FUNCTION FILCR(NAME)
 3:
 4: * create file NAME, open to successive lfn
 5: *  system dependent
 6:
 7:       INTEGER NAME(*)
 8:       INTEGER UNIT, CLINE(82), IBP, IBUF(82), ISP, ISTK(300)
 9:       COMMON /CIN/ UNIT, CLINE, IBP, IBUF, ISP, ISTK
10:       CHARACTER*16 FILENM
```

```
      INTEGER EQUAL. OPICC

      IF (UNIT .LE. 10 .OR. UNIT .GE. 200) THEN
      .  CALL PUTDEC(UNIT,5,3)
      .  CALL REMARK(" system reserved lfn")
      .  FILCR = -3
      .  RETURN
      END IF

      IF (OPICC(NAME.FILENM) .EQ. -3) THEN
      .  FILCR = -3
      .  RETURN
      ELSE
      .  OPEN (UNIT.FILE=FILENM.STATUS='NEW'.ACCESS='SEQUENTIAL'.ERR=1)
      .  UNIT = UNIT + 1
      .  FILCR = UNIT - 1
      .  RETURN
      .
    1 .  CONTINUE
      .  FILCR = -3
      .  RETURN
      END IF

      END
```

```
*-h- FILCRB 788 17 APR 83  23:54:04
      INTEGER FUNCTION FILCRB(NAME)

* create file NAME for unformatted input. open to successive lfn
*  system dependent

      INTEGER NAME(*)
      INTEGER UNIT. CLINE(82). IBP. IBUF(82). ISP. ISTK(300)
      COMMON /CIN/ UNIT. CLINE. IBP. IBUF. ISP. ISTK
      CHARACTER*16 FILENM
      INTEGER EQUAL. OPICC

      IF (UNIT .LE. 10 .OR. UNIT .GE. 200) THEN
      .  CALL PUTDEC(UNIT.5.3)
      .  CALL REMARK(" system reserved lfn")
      .  FILCRB = -3
      .  RETURN
      END IF

      IF (OPICC(NAME.FILENM) .EQ. -3) THEN
      .  FILCRB = -3
      .  RETURN
      ELSE
      .  OPEN (UNIT.FILE=FILENM.STATUS='NEW'.ACCESS='SEQUENTIAL'.
     &.    FORM='UNFORMATTED'.ERR=1)
      .  UNIT = UNIT + 1
      .  FILCRB = UNIT - 1
      .  RETURN
      .
    1 .  CONTINUE
      .  FILCRB = -3
      .  RETURN
      END IF

      END
```

```
*-h- FILOP 830 17 APR 83  23:54:06
      INTEGER FUNCTION FILOP(NAME)

* open file NAME to successive lfn
*  system dependent

      INTEGER NAME(*)
      INTEGER UNIT. CLINE(82), IBP, IBUF(82), ISP. ISTK(300)
      COMMON /CIN/ UNIT. CLINE. IBP, IBUF, ISP. ISTK
      INTEGER TERM(2)
      CHARACTER*16 FILENM
      INTEGER EQUAL. OPICC
      DATA TERM /42.-2/

      IF (UNIT .LE. 10 .OR. UNIT .GE. 200)
      .  CALL PUTDEC(UNIT.5.3)
      .  CALL REMARK(" system reserved lfn")
```

```
18:       . FILOP = -3
19:       . RETURN
20:       END IF
21:
22:       IF (EQUAL(NAME,TERM) .EQ. 1)
23:       . FILOP = 3
24:       . RETURN
25:       ELSE IF (OPICC(NAME,FILENM) .EQ. -3)
26:       . FILOP = -3
27:       . RETURN
28:       ELSE
29:       . OPEN (UNIT,FILE=FILENM,STATUS='OLD',ACCESS='SEQUENTIAL',ERR=1)
30:       . UNIT = UNIT + 1
31:       . FILOP = UNIT - 1
32:       . RETURN
33:       .
34:     1 . CONTINUE
35:       . FILOP = -3
36:       . RETURN
37:       END IF
38:
39:       END
```

```
 1: *-h- FILSET 1320  2 OCT 81  15:27:03
 2:       SUBROUTINE FILSET(DELIM, ARRAY, I, SET, J, MAXSET)
 3:
 4: * expand set at array(i) into set(j), stop at delim
 5:
 6:       INTEGER ESC
 7:       INTEGER ADDSET, INDEX
 8:       EXTERNAL INDEX
 9:       INTEGER I, J, JUNK, MAXSET
10:       INTEGER ARRAY(1), DELIM, SET(MAXSET)
11:       INTEGER DIGITS(11), LOWALF(27), UPALF(27)
12:       DATA DIGITS /48,49,50,51,52,53,54,55,56,57,-2/
13:       DATA LOWALF /97,98,99,100,101,102,103,104,105,106,107,108,109,
14:      +  110,111,112,113,114,115,116,117,118,119,120,121,122,-2/
15:       DATA UPALF /65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,
16:      +  82,83,84,85,86,87,88,89,90,-2/
17:
18:       WHILE (ARRAY(I) .NE. DELIM .AND. ARRAY(I) .NE. -2)
19:       . IF (ARRAY(I) .EQ. 64)
20:       . . JUNK = ADDSET(ESC(ARRAY,I),SET,J,MAXSET)
21:       . ELSE IF (ARRAY(I) .NE. 45)
22:       . . JUNK = ADDSET(ARRAY(I),SET,J,MAXSET)
23:       . ELSE IF (J .LE. 1 .OR. ARRAY(I+1) .EQ. -2)     ! literal -
24:       . . JUNK = ADDSET(45,SET,J,MAXSET)
25:       . ELSE IF (INDEX(DIGITS,SET(J-1)) .GT. 0)
26:       . . CALL DODASH(DIGITS,ARRAY,I,SET,J,MAXSET)
27:       . ELSE IF (INDEX(LOWALF,SET(J-1)) .GT. 0)
28:       . . CALL DODASH(LOWALF,ARRAY,I,SET,J,MAXSET)
29:       . ELSE IF (INDEX(UPALF,SET(J-1)) .GT. 0)
30:       . . CALL DODASH(UPALF,ARRAY,I,SET,J,MAXSET)
31:       . ELSE
32:       . . JUNK = ADDSET(45,SET,J,MAXSET)
33:       . END IF
34:       . I = I + 1
35:       END WHILE
36:
37:       RETURN
38:       END
```

```
 1: *-h- FTOC 1520 28 SEP 82  22:06:02
 2:       INTEGER FUNCTION FTOC(FLT, STR, FIELD, DEC)
 3:
 4: * convert real number to string
 5:
 6:       INTEGER STR(1), FIELD, DEC
 7:       REAL FLT
 8:       INTEGER D, FP, I, J, K, L, WP
 9:       INTEGER DIGITS(11)
10:       REAL AF, ZERO
11:       DATA ZERO /0.0/, DIGITS /48,49,50,51,52,53,54,55,56,57,-2/
12:
13:       AF = ABS(FLT)
14:       STR(1) = -2
15:       I = 2
16:
```

```
17:       IF (AF .EQ. 0.0)    !  check if insufficient room
18:       . L = 2 + DEC
19:       ELSE
20:       . L = INT(DIM(ALOG10(AF),ZERO)+1.0) + 1 + DEC
21:      &.    + INT(0.5*(1.0-SIGN(1.0,FLT)))
22:       END IF
23:       IF (L .GT. FIELD)
24:       . WHILE (I .LE. FIELD+1)
25:       . . STR(I) = 42
26:       . . I = I + 1
27:       . END WHILE
28:       ELSE
29:
30:       . IF (DEC .GT. 0)     !  fractional part
31:       . . FP = NINT((AF - AINT(AF)) * (10.0 ** DEC))
32:       . . FOR J = 1,DEC
33:       . . . D = MOD(FP,10)
34:       . . . STR(I) = DIGITS(D+1)
35:       . . . FP = FP / 10
36:       . . . I = I + 1
37:       . . END FOR
38:       . END IF
39:       .
40:       . IF (DEC .GE. 0)     !  decimal point
41:       . . STR(I) = 46
42:       . . I = I + 1
43:       . END IF
44:       .
45:       . WP = INT(AF)        !  whole part
46:       . IF (WP .EQ. 0 .AND. I .LE. FIELD+1)
47:       . . STR(I) = 48
48:       . . I = I + 1
49:       . ELSE
50:       . . DO
51:       . . . D = MOD(WP,10)
52:       . . . STR(I) = DIGITS(D+1)
53:       . . . WP = WP / 10
54:       . . . I = I + 1
55:       . . UNTIL (WP .EQ. 0)
56:       . END IF
57:       .
58:       . IF (FLT .LT. 0.0)      !  sign
59:       . . STR(I) = 45
60:       . . I = I + 1
61:       . END IF
62:       END IF
63:       I = I - 1
64:
65:       FTOC = I - 1       !  reverse order
66:       J = 1
67:       WHILE (J .LT. I)
68:       . K = STR(I)
69:       . STR(I) = STR(J)
70:       . STR(J) = K
71:       . I = I - 1
72:       . J = J + 1
73:       END WHILE
74:
75:       RETURN
76:       END


 1: *-h- GETARG 1770 17 APR 83  23:54:12
 2:       INTEGER FUNCTION GETARG(N, ARG, MAXSIZ)
 3:
 4: * put nth argument on command line in ARG
 5:
 6:       INTEGER N, ARG(1), MAXSIZ
 7:       INTEGER UNIT, CLINE(82), IBP, IBUF(82), ISP, ISTK(300)
 8:       COMMON /CIN/ UNIT, CLINE, IBP, IBUF, ISP, ISTK
 9:       INTEGER ARGNO, I, L, QUOTE
10:       INTEGER GETLIN
11:
12: *  argeument line is read by PRIMIO
13:
14:       I = 1
15:       ARGNO = -1              !  program name is argument 0
16:       WHILE (ARGNO .LT. N)
17:       . WHILE (CLINE(I) .EQ. 32 .OR. CLINE(I) .EQ. 9)  !  skip blanks
18:       . . I = I + 1
```

```
19:       .  END WHILE
20:       .
21:       .  IF (CLINE(I) .EQ. 10 .OR. CLINE(I) .EQ. -2 .OR. I .GE. 81)
22:       .  .  GETARG = -1              !  argument not found;  return EOF
23:       .  .  RETURN
24:       .  ELSE IF (CLINE(I) .EQ. 34 .OR. CLINE(I) .EQ. 39)  !  quoted string
25:       .  .  QUOTE = CLINE(I)
26:       .  .  I = I + 1
27:       .  .  ARGNO = ARGNO + 1
28:       .  .  IF (ARGNO .EQ. N)     !  copy argument
29:       .  .  .  J = 1
30:       .  .  .  WHILE (CLINE(I).NE.QUOTE .AND. CLINE(I).NE.10 .AND. J .LT. MAXSIZ)
31:       .  .  .  .  ARG(J) = CLINE(I)
32:       .  .  .  .  J = J + 1
33:       .  .  .  .  I = I + 1
34:       .  .  .  END WHILE
35:       .  .  ELSE                  !  skip argument
36:       .  .  .  WHILE (CLINE(I).NE.QUOTE .AND. CLINE(I).NE.10)
37:       .  .  .  .  I = I + 1
38:       .  .  .  END WHILE
39:       .  .  END IF
40:       .  .  I = I + 1
41:       .  ELSE                             !  unquoted string
42:       .  .  IF (CLINE(I) .NE. 47) ARGNO = ARGNO + 1 !  options are not counted
43:       .  .  IF (ARGNO .EQ. N)  !  copy argument
44:       .  .  .  J = 1
45:       .  .  .  WHILE (CLINE(I).NE.32 .AND. CLINE(I).NE.9 .AND. CLINE(I).NE.10
46:      &.  .  .  .   .AND. J.LT.MAXSIZ)
47:       .  .  .  .  ARG(J) = CLINE(I)
48:       .  .  .  .  I = I + 1
49:       .  .  .  .  J = J + 1
50:       .  .  .  END WHILE

51:       .  .  ELSE               !  skip argument
52:       .  .  .  WHILE (CLINE(I).NE.32 .AND. CLINE(I).NE.9 .AND. CLINE(I).NE.10)
53:       .  .  .  .  I = I + 1
54:       .  .  .  END WHILE
55:       .  .  END IF
56:       .  END IF
57:       END WHILE
58:
59:       ARG(J) = -2
60:       GETARG = J - 1
61:
62:       RETURN
63:       END


 1: *-h- GETCCL 590  2 OCT 81  15:45:50
 2:       INTEGER FUNCTION GETCCL(ARG, I, PAT, J)
 3:
 4: * expand char class at arg(i) into pat(j)
 5:
 6:       INTEGER ARG(81), PAT(81)
 7:       INTEGER ADDSET
 8:       INTEGER I, J, JSTART, JUNK
 9:
10:       I = I + 1                     ! skip over [
11:       IF (ARG(I) .EQ. 126)
12:       .  JUNK = ADDSET(110,PAT,J,81)
13:       .  I = I + 1
14:       ELSE
15:       .  JUNK = ADDSET(91,PAT,J,81)
16:       END IF
17:       JSTART = J
18:       JUNK = ADDSET(0,PAT,J,81)
19:       CALL FILSET(93,ARG,I,PAT,J,81)
20:       PAT(JSTART) = J - JSTART - 1
21:       IF (ARG(I) .EQ. 93)
22:       .  GETCCL = -2
23:       ELSE
24:       .  GETCCL = -3
25:       END IF
26:
27:       RETURN
28:       END
```

```
 1: *-h- GETCH 628 17 APR 83  23:54:15
 2:       INTEGER FUNCTION GETCH(C, D)
 3:
 4: * get next (possibly pushed back) character from input
 5:
 6:       INTEGER C, D
 7:       INTEGER UNIT, CLINE(82), IBP, IBUF(82), ISP, ISTK(300)
 8:       COMMON /CIN/ UNIT, CLINE, IBP, IBUF, ISP, ISTK
 9:       INTEGER GETLIN
10:       DATA IBP /82/, ISP /0/
11:
12:       IF (ISP .GT. 0) THEN
13:       .  C = ISTK(ISP)
14:       .  ISP = ISP - 1
15:       ELSE
16:       .  IF (IBP .GT. 81) THEN
17:       .  .  IF (GETLIN(IBUF,D) .EQ. -1) THEN
18:       .  .  .  C = -1
19:       .  .  .  GETCH = -1
20:       .  .  .  RETURN
21:       .  .  END IF
22:       .  .  IBP = 1
23:       .  END IF
24:       .  C = IBUF(IBP)
25:       .  IF (C .EQ. 10) IBP = 81
26:       .  IBP = IBP + 1
27:       END IF
28:
29:       GETCH = C
30:       RETURN
31:
32:       END
```

```
 1: *-h- GETLIN 599 17 APR 83  23:54:55
 2:       INTEGER FUNCTION GETLIN(LIN, D)
 3:
 4: * read next line from input, strip trailing blanks
 5:
 6:       INTEGER LIN(82), D
 7:       INTEGER I, LEN
 8:       CHARACTER BUF(80)
 9:
10:       READ (D,"(80A1)",END=1) (BUF(I),I=1,80)
11:
12:       LEN = 81
13:       WHILE (LEN .GT. 1 .AND. BUF(LEN-1) .EQ. ' ')
14:       .  LEN = LEN - 1
15:       END WHILE
16:       LIN(LEN) = 10                ! end-of-line
17:       LIN(LEN+1) = -2              ! end-of-string
18:       I = LEN - 1
19:       WHILE (I .GT. 0)
20:       .  LIN(I) = ICHAR(BUF(I))
21:       .  I = I - 1
22:       END WHILE
23:       GETLIN = LEN
24:       RETURN
25:
26:     1 CONTINUE
27:       GETLIN = -1
28:       RETURN
29:
30:       END
```

```
 1: *-h- GETOPT 1037 17 APR 83  23:54:57
 2:       INTEGER FUNCTION GETOPT(OPT)
 3:
 4: * get character option from command line
 5:
 6:       INTEGER OPT
 7:       INTEGER UNIT, CLINE(82), IBP, IBUF(82), ISP, ISTK(300)
 8:       COMMON /CIN/ UNIT, CLINE, IBP, IBUF, ISP, ISTK
 9:       INTEGER ARG(20), I, J, K, L
10:       INTEGER GETWRD
11:
12: *  option on processor name (HARRIS system)
13:
14:       I = 1
```

```
15:       L = GETWRD(CLINE,I.ARG)
16:       J = 1
17:       WHILE (ARG(J) .NE. -2)
18:       .  EXIT WHILE IF (ARG(J) .EQ. 46)    ! "."
19:       .  J = J + 1
20:       END WHILE
21:       IF (ARG(J) .EQ. 46)
22:       .  J = J + 1
23:       .  WHILE (ARG(J) .NE. -2)
24:       .  .  IF (ARG(J) .EQ. OPT)
25:       .  .  .  GETOPT = 1
26:       .  .  .  RETURN
27:       .  .  END IF
28:       .  .  J = J + 1
29:       .  END WHILE
30:       END IF
31:
32: *  option character flagged by slash (general)
33:
34:       K = 1
35:       WHILE (GETWRD(CLINE.I.ARG) .NE. 0)
36:       .  IF (ARG(1) .EQ. 47)                ! "/"
37:       .  .  J = 2
38:       .  .  WHILE (ARG(J) .NE. -2)
39:       .  .  .  IF (ARG(J) .EQ. OPT)
40:       .  .  .  .  GETOPT = 1
41:       .  .  .  .  RETURN
42:       .  .  .  END IF
43:       .  .  .  J = J + 1
44:       .  .  END WHILE
45:       .  END IF
46:       .  K = K + 1
47:       END WHILE
48:
49:       GETOPT = 0
50:       RETURN
51:       END
```

```
 1: *-h- GETPAT 187  2 OCT 81  15:50:11
 2:       INTEGER FUNCTION GETPAT(ARG. PAT)
 3:
 4: * convert argument into pattern
 5:
 6:       INTEGER ARG(81). PAT(81)
 7:       INTEGER MAKPAT
 8:
 9:       GETPAT = MAKPAT(ARG.1.-2.PAT)
10:
11:       RETURN
12:       END
```

```
 1: *-h- GETWRD 768 21 SEP 82  11:25:14
 2:       INTEGER FUNCTION GETWRD(IN. I. OUT)
 3:
 4: * starting at IN[i], place first non-blank or quoted  string in OUT
 5:
 6:       INTEGER IN(1), I, OUT(1)
 7:       INTEGER J, QUOTE
 8:
 9: *  skip blanks and tabs
10:
11:       WHILE (IN(I) .EQ. 32 .OR. IN(I) .EQ. 9)
12:       .  I = I + 1
13:       END WHILE
14:
15:       J = 1
16:       IF (IN(I) .EQ. 34 .OR. IN(I) .EQ. 39)
17:       .  QUOTE = IN(I)
18:       .  I = I + 1
19:       .  WHILE (IN(I) .NE. QUOTE .AND. IN(I) .NE. 10 .AND. IN(I) .NE. 9)
20:       .  .  OUT(J) = IN(I)
21:       .  .  I = I + 1
22:       .  .  J = J + 1
23:       .  END WHILE
24:       .  I = I + 1
25:       ELSE
26:       .  WHILE (IN(I) .NE. 10 .AND. IN(I) .NE. -2 .AND. IN(I) .NE. 32
27:      &.   .AND. IN(I) .NE. 9)
```

```
28:       .  .  OUT(J) = IN(I)
29:       .  .  I = I + 1
30:       .  .  J = J + 1
31:       .  END WHILE
32:       END IF
33:
34:       OUT(J) = -2
35:       GETWRD = J - 1
36:       RETURN
37:       END
```

```
 1: *-h- INDEX 309  2 OCT 81  15:57:30
 2:       INTEGER FUNCTION INDEX(STR, C)
 3:
 4: * find character c in string str
 5:
 6:       INTEGER STR(81), C, EOS
 7:       DATA EOS /-2/
 8:
 9:       INDEX = 1
10:       WHILE (STR(INDEX) .NE. EOS)
11:       .  IF (STR(INDEX) .EQ. C)
12:       .  .  RETURN
13:       .  END IF
14:       .  INDEX = INDEX + 1
15:       END WHILE
16:
17:       INDEX = 0
18:
19:       RETURN
20:       END
```

```
 1: *-h- ITOC 1132 11 FEB 83  14:41:16
 2:       INTEGER FUNCTION ITOC(INT, STR, SIZE)
 3:
 4: * convert integer int to string str
 5:
 6:       INTEGER SIZE
 7:       INTEGER INT, STR(SIZE)
 8:       INTEGER D, I, INTVAL, J, K, L
 9:       INTEGER DIGITS(11)
10:       REAL ZERO
11:       INTEGER IABS, ISIGN, MOD
12:       REAL ALOG10
13:       DATA DIGITS /48,49,50,51,52,53,54,55,56,57,-2/, ZERO /0.0/
14:
15:       INTVAL = IABS(INT)
16:       STR(1) = -2
17:       I = 1
18:
19: *  check for insufficient space
20:
21:       IF (INTVAL .EQ. 0)
22:       .  L = 1
23:       ELSE
24:       .  L = NINT(ALOG10(FLOAT(INTVAL))+1.0) +
25:      &.    NINT(0.5*(1-ISIGN(1,INT)))
26:       END IF
27:       IF (L .GT. SIZE)
28:       .  FOR I = 2.SIZE+1
29:       .  .  STR(I) = 42
30:       .  END FOR
31:       ELSE
32:
33: *  generate digits in reverse order
34:
35:       .  DO
36:       .  .  I = I + 1
37:       .  .  D = MOD(INTVAL,10)
38:       .  .  STR(I) = DIGITS(D+1)
39:       .  .  INTVAL = INTVAL / 10
40:       .  UNTIL (INTVAL .EQ. 0 .OR. I .GT. SIZE)
41:       .
42: *  sign
43:       .
44:       .  IF (INT .LT. 0 .AND. I .LE. SIZE)
45:       .  .  I = I + 1
46:       .  .  STR(I) = 45
```

```
47:       .  END IF
48:       END IF
49:
50: *  reverse

51:
52:       ITOC = I - 1
53:       J = 1
54:       WHILE (J .LT. I)
55:       .  K = STR(I)
56:       .  STR(I) = STR(J)
57:       .  STR(J) = K
58:       .  I = I - 1
59:       .  J = J + 1
60:       END WHILE
61:
62:       RETURN
63:       END
```

```
 1: *-h- LENGTH 207  2 OCT 81  16:01:42
 2:       INTEGER FUNCTION LENGTH(STR)
 3:
 4: * compute length of string
 5:
 6:       INTEGER STR(100)
 7:
 8:       LENGTH = 0
 9:       WHILE (STR(LENGTH+1) .NE. -2)
10:       .  LENGTH = LENGTH + 1
11:       END WHILE
12:
13:       RETURN
14:       END
```

```
 1: *-h- LOCATE 390  2 OCT 81  16:01:43
 2:       INTEGER FUNCTION LOCATE(C, PAT, OFFSET)
 3:
 4: * look for c in char class at pat(offset)
 5:
 6:       INTEGER C, PAT(81)
 7:       INTEGER I, OFFSET
 8:
 9: *  size of class is at offset, characters follow
10:
11:       I = OFFSET + PAT(OFFSET)
12:       WHILE (I .GT. OFFSET)
13:       .  IF (C .EQ. PAT(I))
14:       .  .  LOCATE = 1
15:       .  .  RETURN
16:       .  END IF
17:       .  I = I - 1
18:       END WHILE
19:       LOCATE = 0
20:
21:       RETURN
22:       END
```

```
 1: *-h- MAKPAT 1316  2 OCT 81  16:10:17
 2:       INTEGER FUNCTION MAKPAT(ARG,FROM,DELIM,PAT)
 3:
 4: * make pattern from arg(from), terminate at delim
 5:
 6:       INTEGER ESC
 7:       INTEGER ARG(81), DELIM, PAT(81)
 8:       INTEGER ADDSET, GETCCL, STCLOS
 9:       INTEGER FROM, I, J, JUNK, LASTCL, LASTJ, LJ
10:
11:       J = 1                                  ! pat index
12:       LASTJ = 1
13:       LASTCL = 0
14:       I = FROM
15:       WHILE (ARG(I) .NE. DELIM .AND. ARG(I) .NE. -2)
16:       .  LJ = J
17:       .  IF (ARG(I) .EQ. 63)
18:       .  .  JUNK = ADDSET(63,PAT,J,81)
19:       .  ELSE IF (ARG(I) .EQ. 37 .AND. I .EQ. FROM)
20:       .  .  JUNK = ADDSET(37,PAT,J,81)
```

```
21:       .  ELSE IF (ARG(I) .EQ. 36 .AND. ARG(I+1) .EQ. DELIM)
22:       .  .  JUNK = ADDSET(36,PAT,J,81)
23:       .  ELSE IF (ARG(I) .EQ. 91)
24:       .  .  IF (GETCCL(ARG,I,PAT,J) .EQ. -3)
25:       .  .  .  EXIT WHILE
26:       .  .  END IF
27:       .  ELSE IF (ARG(I) .EQ. 42 .AND. I .GT. FROM)
28:       .  .  LJ = LASTJ
29:       .  .  IF (PAT(LJ) .EQ. 37 .OR. PAT(LJ) .EQ. 36 .OR. PAT(LJ) .EQ. 42)
30:       .  .  .  EXIT WHILE
31:       .  .  END IF
32:       .  .  LASTCL = STCLOS(PAT,J,LASTJ,LASTCL)
33:       .  ELSE
34:       .  .  JUNK = ADDSET(97,PAT,J,81)
35:       .  .  JUNK = ADDSET(ESC(ARG,I),PAT,J,81)
36:       .  END IF
37:       .  LASTJ = LJ
38:       .  I = I + 1
39:       END WHILE
40:       IF (ARG(I) .NE. DELIM)              ! unexpected end
41:       .  MAKPAT = -3
42:       ELSE IF (ADDSET(-2,PAT,J,81) .EQ. 0)        ! no room
43:       .  MAKPAT = -3
44:       ELSE
45:       .  MAKPAT = I
46:       END IF
47:
48:       RETURN
49:       END
```

```
 1: *-h- MATCH 331  2 OCT 81  16:10:21
 2:       INTEGER FUNCTION MATCH(LIN, PAT)
 3:
 4: * find match anywhere on line
 5:
 6:       INTEGER LIN(82), PAT(81)
 7:       INTEGER AMATCH
 8:       INTEGER I
 9:
10:       I = 1
11:       WHILE (LIN(I) .NE. -2)
12:       .  IF (AMATCH(LIN,I,PAT) .GT. 0)
13:       .  .  MATCH = 1
14:       .  .  RETURN
15:       .  END IF
16:       .  I = I + 1
17:       END WHILE
18:       MATCH = 0
19:
20:       RETURN
21:       END
```

```
 1: *-h- MPRINT 2492  4 AUG 82  12:36:13
 2:       SUBROUTINE MPRINT(X, LDX, M, N, F, JOB, D, TITLE)
 3:
 4: ************************************************************************
 5: *                                                                      *
 6: * print a matrix stored in full mode                                   *
 7: *                                                                      *
 8: *     X       input:  ldx by n matrix to be printed                    *
 9: *     LDX     input:  leading dimension of X                           *
10: *     M       input:  number of rows in X                              *
11: *     N       input:  number of columns in X                           *
12: *     F       input:  character string specifying print format         *
13: *     JOB     input:  decimal expansion specifying type of matrix      *
14: *                      a=0  print transpose of matrix                  *
15: *                      b=1   rectangular matrix                        *
16: *                      b=2   square symmetric matrix. only upper       *
17: *                            triangle is required                      *
18: *                      b=3   square upper triangular matrix. only      *
19: *                            upper triangle is required                *
20: *     TITLE   input:  character string appearing above matrix          *
21: *                                                                      *
22: *   (7/82)                                                             *
23: *                                                                      *
24: ************************************************************************
25:
```

```
26:       INTEGER LDX, M, N, JOB, D
27:       REAL X(LDX,1)
28:       CHARACTER *(*) F, TITLE
29:       INTEGER I, J, K
30:       REAL ZERO
31:       LOGICAL TRAN, RECT, SYMM, TRIA
32:       DATA ZERO /0.0/
33:
34:       TRAN = JOB / 10 .GT. 0
35:       RECT = MOD(JOB,10) .EQ. 1
36:       SYMM = MOD(JOB,10) .EQ. 2
37:       TRIA = MOD(JOB,10) .EQ. 3
38:
39:       WRITE (D,) TITLE
40:
41:       IF (RECT)
42:       .  IF (TRAN)
43:       .  .  FOR J = 1,N
44:       .  .  .  WRITE (D,FMT=F) (X(I,J),I=1,M)
45:       .  .  END FOR
46:       .  ELSE
47:       .  .  FOR I = 1,M
48:       .  .  .  WRITE (D,FMT=F) (X(I,J),J=1,N)
49:       .  .  END FOR
50:       .  END IF
51:       ELSE IF (SYMM)
52:       .  FOR K = 1,N
53:       .  .  WRITE (D,FMT=F) (X(I,K),I=1,K), (X(K,J),J=K+1,N)
54:       .  END FOR
55:       ELSE IF (TRIA)
56:       .  IF (TRAN)
57:       .  .  FOR J = 1,N
58:       .  .  .  WRITE (D,FMT=F) (X(I,J),I=1,J), (ZERO,I=J+1,N)
59:       .  .  END FOR
60:       .  ELSE
61:       .  .  FOR I = 1,N
62:       .  .  .  WRITE (D,FMT=F) (ZERO,J=1,I-1), (X(I,J),J=I,N)
63:       .  .  END FOR
64:       .  END IF
65:       .
66:       END IF
67:
68:       RETURN
69:       END
```

```
 1: *-h- OMATCH 933 17 APR 83  23:55:08
 2:       INTEGER FUNCTION OMATCH(LIN, I, PAT, J)
 3:
 4: * try to match a single pattern at pat(j)
 5:
 6:       INTEGER LIN(82), PAT(81)
 7:       INTEGER LOCATE
 8:       INTEGER BUMP, I, J
 9:
10:       OMATCH = 0
11:       IF (LIN(I) .EQ. -2)
12:      &  RETURN
13:       BUMP = -1
14:       IF (PAT(J) .EQ. 97) THEN
15:       .  IF (LIN(I) .EQ. PAT(J+1))
16:      &.    BUMP = 1
17:       ELSE IF (PAT(J) .EQ. 37) THEN
18:       .  IF (I .EQ. 1)
19:      &.    BUMP = 0
20:       ELSE IF (PAT(J) .EQ. 63) THEN
21:       .  IF (LIN(I) .NE. 10)
22:      &.    BUMP = 1
23:       ELSE IF (PAT(J) .EQ. 36) THEN
24:       .  IF (LIN(I) .EQ. 10)
25:      &.    BUMP = 0
26:       ELSE IF (PAT(J) .EQ. 91) THEN
27:       .  IF (LOCATE(LIN(I),PAT,J+1) .EQ. 1)
28:      &.    BUMP = 1
29:       ELSE IF (PAT(J) .EQ. 110) THEN
30:       .  IF (LIN(I) .NE. 10 .AND. LOCATE(LIN(I),PAT,J+1) .EQ. 0)
31:      &.    BUMP = 1
32:       ELSE
33:       .  CALL ERROR(" in omatch: can't happen")
34:       END IF
```

```
35:       IF (BUMP .GE. 0) THEN
36:       .  I = I + BUMP
37:       .  OMATCH = 1
38:       END IF
39:
40:       RETURN
41:       END


 1: *-h- OPICC 1704  5 MAY 82  13:54:28
 2:       INTEGER FUNCTION OPICC(NAME, FILE)
 3:
 4: * convert internally mapped area name to HARRIS form for OPEN
 5:
 6:       INTEGER NAME(1)
 7:       CHARACTER*16 FILE
 8:       INTEGER I, J
 9:       CHARACTER*4 QUALC, QUALN
10:       CHARACTER*8 AREA
11:       INTEGER TYPE
12:
13: *  intialize
14:
15:       QUALN = '    '
16:       QUALC = '    '
17:       AREA = '        '
18:       I = 1
19:       J = 1
20:
21: *  name beginning with STAR or DIGIT means there is a qualifier
22:
23:       IF (TYPE(NAME(1)) .EQ. 42)            ! system qualifier
24:       .  QUALN = '0000'
25:       .  QUALC = 'SYST'
26:       .  J = 2
27:       .  I = J
28:       ELSE IF (TYPE(NAME(1)) .EQ. 0)        ! area name with qualifier
29:       .  I = 1
30:       .  WHILE (TYPE(NAME(I)) .NE. 65)         ! find end of numeric field
31:       .  .  I = I + 1
32:       .  .  IF (NAME(I) .EQ. -2 .AND. I .GT. 5)
33:       .  .  .  OPICC = -3
34:       .  .  .  RETURN
35:       .  .  END IF
36:       .  END WHILE
37:       .  J = I
38:       .  I = I - 1
39:       .  WHILE (I .GT. 0)                      ! copy numeric field
40:       .  .  QUALN(5-J+I:5-J+I) = CHAR(NAME(I))    !    backward
41:       .  .  I = I - 1
42:       .  END WHILE
43:       .  WHILE (I .GT. J-5)                    ! fill with zeroes if
44:       .  .  QUALN(5-J+I:5-J+I) = '0'              !    necessary
45:       .  .  I = I - 1
46:       .  END WHILE
47:       .  I = J
48:       .  WHILE (NAME(I) .NE. 42)               ! copy character field
49:       .  .  QUALC(I-J+1:I-J+1) = CHAR(NAME(I))
50:       .  .  I = I + 1
51:       .  .  IF (NAME(I) .EQ. -2 .OR. I .GT. 9)
52:       .  .  .  OPICC = -3
53:       .  .  .  RETURN
54:       .  .  END IF
55:       .  END WHILE
56:       .  I = I + 1
57:       .  J = I
58:       END IF
59:
60: *  copy area name
61:
62:       WHILE (NAME(I) .NE. -2)
63:       .  AREA(I-J+1:I-J+1) = CHAR(NAME(I))
64:       .  I = I + 1
65:       .  IF (I .GT. 18)
66:       .  .  OPICC = -3
67:       .  .  RETURN
68:       .  END IF
69:       END WHILE
70:
71:       FILE = QUALN // QUALC // AREA
```

```
72:       OPICC = -2
73:
74:       RETURN
75:       END
```

```
 1: *-h- PACK 540  2 OCT 81  19:56:36
 2:       SUBROUTINE PACK(BYTES, WORDS)
 3:
 4: * pack 3 characters into HARRIS 24-bit words
 5:
 6:       INTEGER BYTES(81), WORDS(27), PW, PB
 7:
 8:       PB = 1
 9:       WHILE (PB .LE. 81)
10:       .  EXIT WHILE IF (BYTES(PB) .EQ. -2)
11:       .  PW = (PB + 2) / 3
12:       .  WORDS(PW) = BYTES(PB) .SHIFT. 16
13:       .  PB = PB + 1
14:       .  EXIT WHILE IF (BYTES(PB) .EQ. -2)
15:       .  WORDS(PW) = WORDS(PW) .OR. (BYTES(PB) .SHIFT. 8)
16:       .  PB = PB + 1
17:       .  EXIT WHILE IF (BYTES(PB) .EQ. -2)
18:       .  WORDS(PW) = WORDS(PW) .OR. BYTES(PB)
19:       .  PB = PB + 1
20:       END WHILE
21:
22:       RETURN
23:       END
```

```
 1: *-h- PATSIZ 472 17 APR 83  23:55:12
 2:       INTEGER FUNCTION PATSIZ(PAT, N)
 3:
 4: * return size of pattern entry at pat(n)
 5:
 6:       INTEGER PAT(81)
 7:       INTEGER N
 8:
 9:       IF (PAT(N) .EQ. 97)
10:       .  PATSIZ = 2
11:       ELSE IF (PAT(N) .EQ. 37 .OR. PAT(N) .EQ. 36 .OR. PAT(N) .EQ. 63)
12:       .  PATSIZ = 1
13:       ELSE IF (PAT(N) .EQ. 91 .OR. PAT(N) .EQ. 110)
14:       .  PATSIZ = PAT(N+1) + 2
15:       ELSE IF (PAT(N) .EQ. 42)
16:       .  PATSIZ = 4
17:       ELSE
18:       .  CALL ERROR(" in patsiz: can't happen")
19:       END IF
20:
21:       RETURN
22:       END
```

```
 1: *-h- PRIMIO 410 17 APR 83  23:55:14
 2:       SUBROUTINE PRIMIO(D)
 3:
 4: * prime logical unit count for FILCR, FILCRB, FILCRR, FILOP
 5:
 6:       INTEGER D
 7:       INTEGER UNIT, CLINE(82), IBP, IBUF(82), ISP, ISTK(300)
 8:       COMMON /CIN/ UNIT, CLINE, IBP, IBUF, ISP, ISTK
 9:       INTEGER LEVEL, INFILE(5)
10:       COMMON /CLEVEL/ LEVEL, INFILE
11:
12:       LEVEL = 1
13:       UNIT = D
14:
15: *  read command line
16:
17:       BACKSPACE 0
18:       CALL GETLIN(CLINE,0)
19:
20:       RETURN
21:       END
```

```
 1: *-h- PUTCH 337 11 APR 83  12:01:22
 2:       SUBROUTINE PUTCH(C, D)
 3:
 4: * output character
 5:
 6:       INTEGER C, D
 7:       INTEGER OBUF(80), OBP, TABS(80)
 8:       COMMON /COUT/ OBUF, OBP, TABS
 9:       INTEGER NL(2)
10:       DATA NL /10, -2/
11:
12:       IF (C .EQ. 10 .OR. OBP .GE. 80)
13:       .  CALL PUTLIN(NL,D)
14:       ELSE
15:       .  OBP = OBP + 1
16:       .  OBUF(OBP) = C
17:       END IF
18:
19:       RETURN
20:       END
```

```
 1: *-h- PUTDEC 396 16 JUN 82  14:31:24
 2:       SUBROUTINE PUTDEC(N, W, F)
 3:
 4: * put decimal integer n in field width >= w
 5:
 6:       INTEGER CHARS(8)
 7:       INTEGER ITOC
 8:       INTEGER F, I, N, ND, W
 9:
10:       ND = ITOC(N, CHARS, 8)
11:       I = ND + 1
12:       WHILE (I .LE. W)
13:       .  CALL PUTCH(32,F)
14:       .  I = I + 1
15:       END WHILE
16:       I = 1
17:       WHILE (I .LE. ND)
18:       .  CALL PUTCH(CHARS(I),F)
19:       .  I = I + 1
20:       END WHILE
21:
22:       RETURN
23:       END
```

```
 1: *-h- PUTFLT 369  2 MAR 83  22:55:04
 2:       SUBROUTINE PUTFLT(FLT, FIELD, DEC, D)
 3:
 4: * output floating point number with format field.dec
 5:
 6:       INTEGER FIELD, DEC, D
 7:       REAL FLT
 8:       INTEGER I, ND, STR(16)
 9:       INTEGER FTOC
10:
11:       ND = FTOC(FLT,STR,15,DEC)
12:       FOR I = ND+1,FIELD
13:       .  CALL PUTCH(32,D)
14:       END FOR
15:       FOR I = 1,ND
16:       .  CALL PUTCH(STR(I),D)
17:       END FOR
18:
19:       RETURN
20:       END
```

```
 1: *-h- PUTLIN 611 11 APR 83  12:01:26
 2:       SUBROUTINE PUTLIN(LIN, D)
 3:
 4: * write string to output (actual output takes place after newline)
 5:
 6:       INTEGER LIN(*), D
 7:       INTEGER OBUF(80), OBP, TABS(80)
 8:       COMMON /COUT/ OBUF, OBP, TABS
 9:       INTEGER I, J
10:       CHARACTER BUF(80)
11:
```

```
12:       I = 1
13:       WHILE (LIN(I) .NE. -2)
14:       .  IF (LIN(I) .EQ. 10 .OR. OBP .GE. 80) THEN
15:       .  .  J = 1
16:       .  .  WHILE (J .LE. OBP)
17:       .  .  .  BUF(J) = CHAR(OBUF(J))
18:       .  .  .  J = J + 1
19:       .  .  END WHILE
20:       .  .  WRITE (D,"(80A1)") (BUF(J),J=1,OBP)
21:       .  .  OBP = 0
22:       .  ELSE
23:       .  .  OBP = OBP + 1
24:       .  .  OBUF(OBP) = LIN(I)
25:       .  END IF
26:       .  I = I + 1
27:       END WHILE
28:
29:       RETURN
30:       END


 1: *-h- PUTTAB 291 17 APR 83  23:55:17
 2:       SUBROUTINE PUTTAB(D)
 3:
 4: * space forward on output to next tab stop
 5:
 6:       INTEGER D
 7:       INTEGER OBUF(80), OBP, TABS(80)
 8:       COMMON /COUT/ OBUF, OBP, TABS
 9:
10:       CALL PUTCH(32,3)
11:       DO
12:       .  CALL PUTCH(32,3)
13:       UNTIL (TABS(OBP+1) .EQ. 1 .OR. OBP .EQ. 0)
14:
15:       RETURN
16:       END


 1: *-h- REMARK 537 11 APR 83  12:01:28
 2:       SUBROUTINE REMARK(MSG)
 3:
 4: * write message to ERROUT
 5:
 6:       CHARACTER MSG*(*)
 7:       INTEGER BUF(81), I, L
 8:       INTEGER ESC
 9:
10:       L = LEN(MSG)
11:       FOR I = 1,L
12:       .  BUF(I) = ICHAR(MSG(I:I))
13:       END FOR
14:       BUF(L+1) = -2
15:
16:       I = 1
17:       WHILE (I .LE. L)
18:       .  IF (BUF(I) .EQ. 64) THEN
19:       .  .  CALL PUTCH(ESC(BUF,I),3)
20:       .  ELSE
21:       .  .  CALL PUTCH(BUF(I),3)
22:       .  END IF
23:       .  I = I + 1
24:       END WHILE
25:
26:       CALL PUTCH(32,3)
27:       FOR I = 1,3
28:       .  CALL PUTCH(42,3)
29:       END FOR
30:       CALL PUTCH(10,3)
31:
32:       RETURN
33:       END


 1: *-h- SCOPY 311  2 OCT 81  20:15:05
 2:       SUBROUTINE SCOPY(FROM, I, TO, J)
 3:
 4: * copy string at from(i) to to(j)
 5:
```

```
 6:       INTEGER FROM(100), TO(100)
 7:       INTEGER I, J. K1. K2
 8:
 9:       K2 = J
10:       K1 = I
11:       WHILE (FROM(K1) .NE. -2)
12:       .  TO(K2) = FROM(K1)
13:       .  K2 = K2 + 1
14:       .  K1 = K1 + 1
15:       END WHILE
16:       TO(K2) = -2
17:
18:       RETURN
19:       END
```

```
 1: *-h- SETTAB 718 29 MAY 83  14:59:51
 2:       SUBROUTINE SETTAB(LIST)
 3:
 4: * set tab stops
 5:
 6:       INTEGER LIST(*)
 7:       INTEGER OBUF(80). OBP. TABS(80)
 8:       COMMON /COUT/ OBUF. OBP, TABS
 9:       INTEGER I. J
10:
11:       IF (LIST(2) .EQ. 43) THEN
12:       .  I = 1
13:       .  WHILE (I .LT. LIST(1))
14:       .  .  TABS(I) = 0
15:       .  .  I = I + 1
16:       .  END WHILE
17:       .  TABS(I) = 1
18:       .  I = I + 1
19:       .  WHILE (I .LT. 80)
20:       .  .  IF (MOD(I-LIST(1),LIST(3)) .EQ. 0) THEN
21:       .  .  .  TABS(I) = 1
22:       .  .  ELSE
23:       .  .  .  TABS(I) = 0
24:       .  .  END IF
25:       .  .  I = I + 1
26:       .  END WHILE
27:       ELSE
28:       .  J = 1
29:       .  I = 1
30:       .  WHILE (I .LT. 80)
31:       .  .  IF (I .EQ. LIST(J)) THEN
32:       .  .  .  TABS(I) = 1
33:       .  .  .  J = J + 1
34:       .  .  ELSE
35:       .  .  .  TABS(I) = 0
36:       .  .  END IF
37:       .  .  I = I + 1
38:       .  END WHILE
39:       END IF
40:
41:       RETURN
42:       END
```

```
 1: *-h- STCLOS 614  2 OCT 81  20:15:05
 2:       INTEGER FUNCTION STCLOS(PAT. J , LASTJ, LASTCL)
 3:
 4: * insert closure entry at pat(j)
 5:
 6:       INTEGER PAT(81)
 7:       INTEGER ADDSET
 8:       INTEGER J, JP. JT. JUNK, LASTCL.  LASTJ
 9:
10:       JP = J - 1
11:       WHILE (JP .GE. LASTJ)                ! make a hole
12:       .  JT = JP + 4
13:       .  JUNK = ADDSET(PAT(JP).PAT.JT.81)
14:       .  JP = JP - 1
15:       END WHILE
```

```
16:       J = J + 4
17:       STCLOS = LASTJ
18:       JUNK = ADDSET(42,PAT,LASTJ,81)    ! put closure in it
19:       JUNK = ADDSET(0,PAT,LASTJ,81)     ! COUNT
20:       JUNK = ADDSET(LASTCL,PAT,LASTJ,81)  ! PREVCL
21:       JUNK = ADDSET(0,PAT,LASTJ,81)       ! START
22:
23:       RETURN
24:       END
```

```
 1: *-h- TYPE 335  2 OCT 81  20:20:51
 2:       INTEGER FUNCTION TYPE(C)
 3:
 4: * return 'letter', 'digit' or other (ASCII alphabet)
 5:
 6:       INTEGER C
 7:
 8:       IF (C .GE. 48 .AND. C .LE. 57)
 9:       .  TYPE = 0
10:       ELSE IF (C .GE. 65 .AND. C .LE. 90)
11:       .  TYPE = 65
12:       ELSE IF (C .GE. 97 .AND. C .LE. 122)
13:       .  TYPE = 65
14:       ELSE
15:       .  TYPE = C
16:       END IF
17:
18:       RETURN
19:       END
```

```
 1: *-h- UNPACK 368  3 OCT 81  22:97:02
 2:       SUBROUTINE UNPACK(WORDS, BYTES, NWORDS)
 3:
 4: * split string into one right justified character per word
 5:
 6:       INTEGER WORDS(27), BYTES(81), NWORDS
 7:       INTEGER PW
 8:
 9:       FOR PW = 1,NWORDS
10:       .  BYTES(PW*3-2) = WORDS(PW) .SHIFT. -16
11:       .  BYTES(PW*3-1) = 255 .AND. (WORDS(PW) .SHIFT. -8)
12:       .  BYTES(PW*3) = 255 .AND. WORDS(PW)
13:       END FOR
14:
15:       RETURN
16:       END
```

A.7 Mathematical function subroutine package.

Most the routines in this chapter are for performing repetitive calculations on arrays of data. Many are adapted from the SNAP library of array processing routines [CSPI, 1978] although they are written in Fortran for the present application. Routines that take vector inputs and give vector outputs are prefixed by the letter 'V'. Routines that take vector input and give a scalar output are prefixed by the letters 'SV'. One exception to this rule is the real function INNERP which calculates the inner product of two vectors. Their function is generally very simple and is described adequately by the comments in the programs themselves so are not described further.

There are routines that do not fall into the above category. Routine MSVDC calculates the singular value decomposition of a rectangular matrix. It is adapted from SSVDC in Dongarra et al. [1980] to use the available vector functions. This routine is used by program LSVZ to calculate the generalized inverse of the data matrix. Routine LSVDF of the IMSL package was not entirely satisfactory for my purposes. Routines MPRD, MPRDT and MTPRD calculate matrix inner products. Routine SRTRT sorts a real array into increasing order and keeps track of the permutations in an integer array. Routines SRTPI and SRTPR use the information stored by SRTRT to apply the permutations to other integer and real arrays respectively. The latter five subroutines are also available in some form in the IMSL package. I wrote them before that became available at HIG.

```
 1: *-h- INNERP 676 30 AUG 82  13:02:17
 2:       REAL FUNCTION INNERP(U, INCU, V, INCV, N)
 3:
 4: *     n
 5: * sum   (U[k] * V[k])
 6: *    k=1
 7:
 8:       REAL U(*), V(*)
 9:       INTEGER N, IU, INCU, IV, INCV
10:       REAL B
11:
12:       B = 0.0
13:       IF (N .EQ. 0)
14:       .  INNERP = B
15:       .  RETURN
16:       END IF
17:
18:       IF (INCU .EQ. 1 .AND. INCV .EQ. 1)
19:
20: *  contigous data
21:
22:       .  FOR I = 1,N
23:       .  .  B = B + (U(I) * V(I))
24:       .  END FOR
25:       .
26:       ELSE
27:
28: *  non-contiguous data
29:
30:       .  IU = 1
31:       .  IV = 1
32:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
33:       .  IF (INCV .LT. 0) IV = (-N+1) * INCV + 1
34:       .
35:       .  FOR I = 1,N
36:       .  .  B = B + (U(IU) * V(IV))
37:       .  .  IU = IU + INCU
38:       .  .  IV = IV + INCV
39:       .  END FOR
40:       .
41:       END IF
42:
43:       INNERP = B
44:
45:       RETURN
46:       END
```

```
 1: *-h- L2NORM 1879 28 JAN 83  12:27:09
 2:       REAL FUNCTION L2NORM(X, INCX, N)
 3:
 4: * calculate the L  norm of a real vector
 5: *                2
 6: *  proceeds in 4 parts to avoid overflow and underflow
 7: *   phase 1:  scan until non-zero element is detected
 8: *   phase 2:  sum numbers smaller than lower limit
 9: *   phase 3:  sum numbers between lower and upper limit
10: *   phase 4:  sum numbers greater than upper limit
11:
12:       INTEGER INCX, N
13:       REAL X(*)
14:       INTEGER CASE, IX, NMAX
15:       REAL AX, HICUT, HITEST, LOCUT, SUM, XMAX
16:       DATA LOCUT /4.441E-16/, HICUT /1.304E+19/
17:
18:       IF (N .LE. 0)
19:       .  L2NORM = 0.0
20:       .  RETURN
21:       END IF
22:
23:       SUM = 0.0
24:       XMAX = 0.0
25:       HITEST = HICUT / FLOAT(N)
26:       IX = 1
27:       NMAX = N * INCX
28:       CASE = 1
29:
30:       WHILE (IX .LE. NMAX)
31:       .  AX = ABS(X(IX))
32:       .  IF (CASE .EQ. 1)
```

```
33:       . .  IF (X(IX) .EQ. 0.0)
34:       . .  .  IX = IX + INCX
35:       . .  ELSE IF (AX .GT. LOCUT)
36:       . .  .  CASE = 3
37:       . .  ELSE
38:       . .  .  CASE = 2
39:       . .  .  XMAX = AX
40:       . .  .  SUM = SUM + (X(IX) / XMAX) ** 2
41:       . .  .  IX = IX + INCX
42:       . .  END IF
43:       .  ELSE IF (CASE .EQ. 2)
44:       . .  IF (AX .GE. LOCUT)
45:       . .  .  SUM = SUM * XMAX ** 2
46:       . .  .  CASE = 3
47:       . .  ELSE
48:       . .  .  IF (AX .GT. XMAX)
49:       . .  .  .  SUM = 1.0 + SUM * (XMAX / X(IX)) ** 2
50:       . .  .  .  XMAX = AX
51:       . .  .  ELSE
52:       . .  .  .  SUM = SUM + (X(IX) / XMAX) ** 2
53:       . .  .  END IF
54:       . .  .  IX = IX + INCX
55:       . .  END IF
56:       .  ELSE IF (CASE .EQ. 3)
57:       . .  FOR J = IX,NMAX,INCX
58:       . .  .  AX = ABS(X(J))
59:       . .  .  IF (AX .GE. HITEST)
60:       . .  .  .  CASE = 4
61:       . .  .  .  SUM = SUM / X(J) ** 2
62:       . .  .  .  XMAX = AX
63:       . .  .  .  IX = J
64:       . .  .  .  EXIT FOR
65:       . .  .  END IF
66:       . .  .  SUM = SUM + X(J) ** 2
67:       . .  END FOR
68:       . .
69:       . .  IF (CASE .EQ. 3)
70:       . .  .  L2NORM = SQRT(SUM)
71:       . .  .  RETURN
72:       . .  END IF
73:       . .
74:       .  ELSE IF (CASE .EQ. 4)
75:       . .  IF (AX .GT. XMAX)
76:       . .  .  SUM = 1.0 + SUM * (XMAX / X(IX)) ** 2
77:       . .  .  XMAX = AX
78:       . .  ELSE
79:       . .  .  SUM = SUM + (X(IX) / XMAX) ** 2
80:       . .  END IF
81:       . .  IX = IX + INCX
82:       .  END IF
83:       END WHILE
84:
85:       L2NORM = XMAX * SQRT(SUM)
86:       RETURN
87:
88:       END
```

```
 1: *-h- MPRD 398  6 SEP 82  15:27:59
 2:       SUBROUTINE MPRD(Y, LDY, U, LDU, V, LDV, L, M, N)
 3:
 4: * matrix inner product Y = UV
 5: *  column oriented algorithm
 6:
 7:       INTEGER LDY, LDU, LDV, L, M, N
 8:       REAL Y(LDY,*), U(LDU,*), V(LDV,*)
 9:       INTEGER I, J
10:
11:       FOR J = 1,N
12:       .  CALL VCOPY(Y(1,J),1,0.0,0,L)
13:       .  FOR I = 1,M
14:       .  .  CALL VSMA2(Y(1,J),1,1.0,Y(1,J),1,V(I,J),U(1,I),1,0.0,L)
15:       .  END FOR
16:       END FOR
17:
18:       RETURN
19:       END
```

```
 1: *-h- MPRDT 430  6 SEP 82  15:28:00
 2:       SUBROUTINE MPRDT(Y, LDY, U, LDU, V, LDV, L, M, N)
 3:
 4: *                          T
 5: * matrix inner product Y = UV
 6: *  column oriented algorithm
 7:
 8:       INTEGER LDY, LDU, LDV, L, M, N
 9:       REAL Y(LDY,*), U(LDU,*), V(LDV,*)
10:       INTEGER I, J
11:
12:       FOR J = 1,N
13:       .  CALL VCOPY(Y(1,J),1,0.0,0,L)
14:       .  FOR I = 1,M
15:       .  .  CALL VSMA2(Y(1,J),1,1.0,Y(1,J),1,V(J,I),U(1,I),1,0.0,L)
16:       .  END FOR
17:       END FOR
18:
19:       RETURN
20:       END
```

```
 1: *-h- MSVDC 10493 11 APR 83  23:07:32
 2:       SUBROUTINE MSVDC(X,LDX,N,P,S,E,U,LDU,V,LDV,WORK,JOB,INFO)
 3:
 4: ************************************************************************
 5: *                                                                      *
 6: * Reduce a real n by p matrix X by orthogonal transformations U, V     *
 7: *  to diagonal form.  the diagonal elements s(i) are the singular      *
 8: *  values of X; the columns of U are the left singular vectors; and    *
 9: *  the columns of V are the right singular vectors                      *
10: *                                                                      *
11: *       X     input:  ldx by p matrix whose singular value decom-      *
12: *                       position is to be computed                     *
13: *       LDX   input:  leading dimension of X                           *
14: *       N     input:  number of columns of X                           *
15: *       P     input:  number of rows of X                              *
16: *       S     output: first min(n+1,p) entries of s contain the        *
17: *                       singular values of X in descending order       *
18: *                       of magnitude                                   *
19: *       E     output: min(n,p) values usually contain zeroes. if       *
20: *                       INFO=0 then it contains the superdiagonal of   *
21: *                                                 T                    *
22: *                       the bidiagonal matrix B = UXV                  *
23: *       U     output: ldu by k matrix of left singular vectors         *
24: *       LDU   input:  leading dimension of U                           *
25: *       V     output: ldv by k matrix of right singular vectors        *
26: *       LDV   input:  leading dimension of V                           *
27: *       WORK          work vector                                      *
28: *       JOB   input:  decimal digit ab                                 *
29: *                       (a=0)  do not compute left singular vectors    *
30: *                       (a=1)  return n left singular vectors          *
31: *                       (a=2)  return 1st min(p,n) left vectors        *
32: *                       (b=0)  do not return right singular vectors    *
33: *                       (b=1)  return the right singular vectors       *
34: *       INFO  output: the singular values and corresponding singular   *
35: *                       vectors s(info+1),s(info+2)...s(m) are correct *
36: *                       if info=0 all are correct.                     *
37: *     (Linpack)                                                        *
38: *                                                                      *
39: ************************************************************************
40:
41:       INTEGER LDX, N, P, LDU, LDV, JOB, INFO
42:       REAL X(LDX,*), S(*), E(*), U(LDU,*), V(LDV,*), WORK(*)
43:       INTEGER CASE, ITER, J, JOBU, K, KK, L, LL, LLS, LM1, LP1, LS
44:       INTEGER LU, M, MAXIT, MM, MM1, MP1, NCT, NCTP1, NCU, NRT, NRTP1
45:       INTEGER ZERO
46:       REAL B, C, CS, EL, EMM1, F, G, R, SCALE, SHIFT, SI, SL, SM, SMM1
47:       REAL T, T1, TEST, ZTEST
48:       LOGICAL WANTU, WANTV
49:       INTEGER MAX0, MIN0, MOD
50:       REAL ABS, AMAX1, INNERP, L2NORM, SQRT
51:       DATA ZERO /0/
52:
53:       MAXIT = 30      ! maximum number of iterations
54:
55: *  decide what to do
56:
57:       JOBU = MOD(JOB,100) / 10
58:       NCU = N
```

```
      IF (JOBU .GT. 1) NCU = MIN0(N,P)
      WANTU = JOBU .NE. 0
      WANTV = MOD(JOB,10) .NE. 0

*  reduce X to bidiagonal form storing diagonal element in s and
*   super-diagonal element in e

      INFO = 0
      NCT = MIN0(N-1,P)
      NRT = MAX0(ZERO,MIN0(P-2,N))
      LU = MAX0(NCT,NRT)
      IF (LU .GE. 1)
     .  FOR L = 1,LU
     .  .  LP1 = L + 1
     .  .  IF (L .LE. NCT)
     .  .
*  compute transformation for lth column and place lth diagonal in s(1)
     .  .
     .  .  .  S(L) = L2NORM(X(L,L),1,N-L+1)
     .  .  .  IF (S(L) .NE. 0.0)
     .  .  .  .  IF (X(L,L) .NE. 0.0) S(L) = SIGN(S(L),X(L,L))
     .  .  .  .  CALL VSCALE(X(L,L),1,1.0/S(L),N-L+1)
     .  .  .  .  X(L,L) = 1.0 + X(L,L)
     .  .  .  END IF
     .  .  .  S(L) = -S(L)
     .  .  END IF
     .  .  IF (P .GE. LP1)
     .  .  .  FOR J = LP1,P
     .  .  .  .  IF (L .LE. NCT)
     .  .  .  .  .  IF (S(L) .NE. 0.0)
*  apply transformation
     .  .  .  .  .  .  T = -INNERP(X(L,L),1,X(L,J),1,N-L+1) / X(L,L)
     .  .  .  .  .  .  CALL VSMA2(X(L,J),1,T,X(L,L),1,1.0,X(L,J),1,0.0,N-L+1)
     .  .  .  .  .  END IF
     .  .  .  .  END IF
*  place lth row of X into e for subsequent calculation of new transform
     .  .  .  .  E(J) = X(L,J)
     .  .  .  END FOR
     .  .  END IF
     .  .  IF (WANTU .AND. L .LE. NCT)
*  place transformation in U for subsequent back multiplication
     .  .  .  CALL VCOPY(U(L,L),1,X(L,L),1,N-L+1)
     .  .  END IF
     .  .  IF (L .LE. NRT)
*  compute the lth row transformation and place the lth superdiagonal
*    in e(1)
     .  .  .  E(L) = L2NORM(E(LP1),1,P-L)
     .  .  .  IF (E(L) .NE. 0.0)
     .  .  .  .  IF (E(LP1) .NE. 0.0) E(L) = SIGN(E(L),E(LP1))
     .  .  .  .  CALL VSCALE(E(LP1),1,1.0/E(L),P-L)
     .  .  .  .  E(LP1) = 1.0 + E(LP1)
     .  .  .  END IF
     .  .  .  E(L) = -E(L)
     .  .  .  IF (LP1 .LE. N .AND. E(L) .NE. 0.0)
*  apply transformation
     .  .  .  .  CALL VCOPY(WORK(LP1),1,0.0,0,N-L)
     .  .  .  .  FOR J = LP1,P
     .  .  .  .  .  CALL VSMA2(WORK(LP1),1,1.0,WORK(LP1),1,E(J),X(LP1,J),1,0.0,N-L)
     .  .  .  .  END FOR
     .  .  .  .  FOR J = LP1,P
     .  .  .  .  .  CALL VSMA2(X(LP1,J),1,1.0,X(LP1,J),1,-E(J)/E(LP1),WORK(LP1),1,
    &.  .  .  .  .     0.0,N-L)
     .  .  .  .  END FOR
     .  .  .  END IF
     .  .  .  IF (WANTV)
*  place the transformation in V for subsequent back-multiplication
     .  .  .  .  CALL VCOPY(V(LP1,L),1,E(LP1),1,P-L)
     .  .  .  END IF
     .  .  END IF
     .  END FOR
      END IF

*  set up final bi-diagonal matrix of order m

      M = MIN0(P,N+1)
      NCTP1 = NCT + 1
      NRTP1 = NRT + 1
      IF (NCT .LT. P) S(NCTP1) = X(NCTP1,NCTP1)
      IF (N .LT. M) S(M) = 0.0
      IF (NRTP1 .LT. M) E(NRTP1) = X(NRTP1,M)
```

```
      E(M) = 0.0

      IF (WANTU)

*  generate U

      . IF (NCU .GE. NCTP1)
      . . FOR J = NCTP1,NCU
      . . . CALL VCOPY(U(1,J),1,0.0,0,N)
      . . . U(J,J) = 1.0
      . . END FOR
      . END IF
      . IF (NCT .GE. 1)
      . . FOR LL = 1,NCT
      . . . L = NCT - LL + 1
      . . . IF (S(L) .NE. 0.0)
      . . . . LP1 = L + 1
      . . . . IF (NCU .GE. LP1)
      . . . . . FOR J = LP1,NCU
      . . . . . . T = -INNERP(U(L,L),1,U(L,J),1,N-L+1) / U(L,L)
      . . . . . . CALL VSMA2(U(L,J),1,T,U(L,L),1,1.0,U(L,J),1,0.0,N-L+1)
      . . . . . END FOR
      . . . . END IF
      . . . . CALL VSCALE(U(L,L),1,-1.0,N-L+1)
      . . . . U(L,L) = 1.0 + U(L,L)
      . . . . LM1 = L - 1
      . . . . IF (LM1 .GT. 0) CALL VCOPY(U(1,L),1,0.0,0,LM1)
      . . . ELSE
      . . . . CALL VCOPY(U(1,L),1,0.0,0,N)
      . . . . U(L,L) = 1.0
      . . . END IF
      . . END FOR
      . END IF
      END IF

      IF (WANTV)

*  generate V

      . FOR LL = 1,P
      . . L = P - LL + 1
      . . LP1 = L + 1
      . . IF (L .LE. NRT)
      . . . IF (E(L) .NE. 0.0)
      . . . . FOR J = LP1,P
      . . . . . T = -INNERP(V(LP1,L),1,V(LP1,J),1,P-L) / V(LP1,L)
      . . . . . CALL VSMA2(V(LP1,J),1,T,V(LP1,L),1,1.0,V(LP1,J),1,0.0,P-L)
      . . . . END FOR
      . . . END IF
      . . END IF
      . . CALL VCOPY(V(1,L),1,0.0,0,P)
      . . V(L,L) = 1.0
      . END FOR
      END IF


*
*  main iteration loop for singular values
*

      MM = M
      ITER = 0
      WHILE (M .GT. 0)

*  m = 0 means all singular values have ben found
*
*  inspect for negligible elements in s and e, on completion variables
*   case and l are set as follows
*        case=1  s(m) and e(l-1) are negligible & l<m
*        case=2  s(l) is negligible & l<m
*        case=3  e(l-1) is negligible & l<m & s(l)...s(m) are not
*                  negligible  (QR step)
*        case=4  e(m-1) is negligible  (convergence)

      . IF (ITER .EQ. MAXIT)
      . . INFO = M
      . . EXIT WHILE
      . END IF
      . L = M - 1
```

```
      .  WHILE (L .NE. 0)
      .  .  TEST = ABS(S(L)) + ABS(S(L+1))
      .  .  ZTEST = TEST + ABS(E(L))
      .  .  IF (ZTEST .EQ. TEST)
      .  .  .  E(L) = 0.0
      .  .  .  EXIT WHILE
      .  .  END IF
      .  .  L = L - 1
      .  END WHILE
      .  IF (L .EQ. M-1)
      .  .  CASE = 4
      .  ELSE
      .  .  LP1 = L + 1
      .  .  MP1 = M + 1
      .  .  FOR LLS = LP1,MP1
      .  .  .  LS = M - LLS + LP1
      .  .  .  EXIT FOR IF (LS .EQ. L)
      .  .  .  TEST = 0.0
      .  .  .  IF (LS .NE. M) TEST = TEST + ABS(E(LS))
      .  .  .  IF (LS .NE. L+1) TEST = TEST + ABS(E(LS-1))
      .  .  .  ZTEST = TEST + ABS(S(LS))
      .  .  .  IF (ZTEST .EQ. TEST)
      .  .  .  .  S(LS) = 0.0
      .  .  .  .  EXIT FOR
      .  .  .  END IF
      .  .  END FOR
      .  .  IF (LS .EQ. L)
      .  .  .  CASE = 3
      .  .  ELSE IF (LS .EQ. M)
      .  .  .  CASE = 1
      .  .  ELSE
      .  .  .  CASE = 2
      .  .  .  L = LS
      .  .  END IF
      .  END IF
      .  L = L + 1
      .
*     perform task indicated by case
      .
      .  IF (CASE .EQ. 1)
      .
*     deflate negligible s(m)
      .
      .  .  MM1 = M - 1
      .  .  F = E(M-1)
      .  .  E(M-1) = 0.0
      .  .  FOR KK = L,MM1
      .  .  .  K = MM1 - KK + L
      .  .  .  T1 = S(K)
      .  .  .  CALL SROTG(T1,F,CS,SI)
      .  .  .  S(K) = T1
      .  .  .  IF (K .NE. L)
      .  .  .  .  F = -SI * E(K-1)
      .  .  .  .  E(K-1) = CS * E(K-1)
      .  .  .  END IF
      .  .  .  IF (WANTV)  CALL VROT(V(1,K),1,V(1,M),1,CS,SI,P)
      .  .  END FOR
      .  ELSE IF (CASE .EQ. 2)
      .
*     split at negligible s(1)
      .
      .  .  F = E(L-1)
      .  .  E(L-1) = 0.0
      .  .  FOR K = L,M
      .  .  .  T1 = S(K)
      .  .  .  CALL SROTG(T1,F,CS,SI)
      .  .  .  S(K) = T1
      .  .  .  F = -SI * E(K)
      .  .  .  E(K) = CS * E(K)
      .  .  .  IF (WANTU) CALL VROT(U(1,K),1,U(1,L-1),1,CS,SI,N)
      .  .  END FOR
      .  ELSE IF (CASE .EQ. 3)
      .
*     perform one QR step
*       calculate shift
      .
      .  .  SCALE = AMAX1(ABS(S(M)),ABS(S(M-1)),ABS(E(M-1)),ABS(S(L)),
     &.  .    ABS(E(L)))
      .  .  SM = S(M) / SCALE
      .  .  SMM1 = S(M-1) / SCALE
```

```
298:       . .  EMM1 = E(M-1) / SCALE
299:       . .  SL = S(L) / SCALE
300:       . .  EL = E(L) / SCALE
301:       . .  B = ((SMM1 + SM) * (SMM1 - SM) + EMM1 ** 2) / 2.0
302:       . .  C = (SM * EMM1) ** 2
303:       . .  SHIFT = 0.0
304:       . .  IF (B .NE. 0.0 .OR. C .NE. 0.0)
305:       . . .  SHIFT = SQRT(B**2+C)
306:       . . .  IF (B .LT. 0.) SHIFT = -SHIFT
307:       . . .  SHIFT = C / (B + SHIFT)
308:       . .  END IF
309:       . .  F = (SL + SM) * (SL - SM) - SHIFT
310:       . .  G = SL * EL
311:       . .
312: *  chase zeroes
313:       . .
314:       . .  MM1 = M - 1
315:       . .  FOR K = L,MM1
316:       . . .  CALL SROTG(F,G,CS,SI)
317:       . . .  IF (K .NE. L) E(K-1) = F
318:       . . .  F = CS * S(K) + SI * E(K)
319:       . . .  E(K) = -SI * S(K) + CS * E(K)
320:       . . .  G = SI * S(K+1)
321:       . . .  S(K+1) = CS * S(K+1)
322:       . . .  IF (WANTV) CALL VROT(V(1,K),1,V(1,K+1),1,CS,SI,P)
323:       . . .
324:       . . .  CALL SROTG(F,G,CS,SI)
325:       . . .  S(K) = F
326:       . . .  F = CS * E(K) + SI * S(K+1)
327:       . . .  S(K+1) = -SI * E(K) + CS * S(K+1)
328:       . . .  G = SI * E(K+1)
329:       . . .  E(K+1) = CS * E(K+1)
330:       . . .  IF (WANTU .AND. K .LT. N) CALL VROT(U(1,K),1,U(1,K+1),1,CS,SI,N)
331:       . .  END FOR
332:       . .  E(M-1) = F
333:       . .  ITER = ITER + 1
334:       .  ELSE
335:       .
336: *  convergence
337: *    make the singular value positive and order
338:       .
339:       . .  IF (S(L) .LT. 0.0)
340:       . . .  S(L) = -S(L)
341:       . . .  IF (WANTV) CALL VSCALE(V(1,L),1,-1.0,P)
342:       . .  END IF
343:       . .  WHILE (L .NE. MM)
344:       . . .  EXIT WHILE IF (S(L) .GE. S(L+1))
345:       . . .  T = S(L)
346:       . . .  S(L) = S(L+1)
347:       . . .  S(L+1) = T
348:       . . .  IF (WANTV .AND. L .LT. P) CALL VSWAP(V(1,L),1,V(1,L+1),1,P)
349:       . . .  IF (WANTU .AND. L .LT. N) CALL VSWAP(U(1,L),1,U(1,L+1),1,N)
350:       . . .  L = L + 1
351:       . .  END WHILE
352:       . .  ITER = 0
353:       . .  M = M - 1
354:       .  END IF
355:       END WHILE
356:       INFO = M
357:
358:       RETURN
359:       END
```

```
 1: *-h- MTPRD 371  8 OCT 81  13:46:49
 2:       SUBROUTINE MTPRD(Y, LDY, U, LDU, V, LDV, L, M, N)
 3:
 4: *                                T
 5: *   matrix inner product  Y = U V
 6:
 7:       INTEGER LDY, LDU, LDV, L, M, N
 8:       REAL Y(LDY,*), U(LDU,*), V(LDV,*)
 9:       INTEGER I, J
10:       REAL INNERP
11:
12:       FOR J = 1,N
13:       .  FOR I = 1,M
14:       .  .  Y(I,J) = INNERP(U(1,I),1,V(1,J),1,L)
15:       .  END FOR
16:       END FOR
```

```
17:
18:       RETURN
19:       END
```

```
 1: *-h- SROTG 562 15 JUN 82  21:51:51
 2:       SUBROUTINE SROTG(A,B,C,S)
 3:
 4: * construct Givens plane rotation
 5:
 6:       REAL A, B, C, S
 7:       REAL ROE, SCALE, R, Z
 8:
 9:       ROE = B
10:       IF (ABS(A) .GT. ABS(B)) ROE = A
11:       SCALE = ABS(A) + ABS(B)
12:       IF (SCALE .EQ. 0.0)
13:       .  C = 1.0
14:       .  S = 0.0
15:       .  R = 0.0
16:       ELSE
17:       .  R = SCALE * SQRT((A/SCALE)**2 + (B/SCALE)**2)
18:       .  R = SIGN(1.0,ROE) * R
19:       .  C = A / R
20:       .  S = B / R
21:       END IF
22:       Z = 1.0
23:       IF (ABS(A) .GT. ABS(B)) Z = S
24:       IF (ABS(B) .GE. ABS(A) .AND. C .NE. 0.0) Z = 1.0 / C
25:       A = R
26:       B = Z
27:
28:       RETURN
29:       END
```

```
 1: *-h- SRTRT 1422 13 MAR 83  21:23:07
 2:       SUBROUTINE SRTRT(X, T, N)
 3:
 4: * sort elements of real vector in increasing algebraic order
 5: * record permutations in T
 6: *  C.A.R. Hoare's Quicksort algorithm
 7:
 8:       INTEGER N, T(*)
 9:       REAL X(*)
10:       INTEGER I, J, K, LV(20), P, UV(20)
11:       REAL PIVOT, S
12:
13:       LV(1) = 1
14:       UV(1) = N
15:       P = 1
16:       WHILE (P .GT. 0)
17:       .  IF (LV(P) .GE. UV(P))          ! only one element in this subset
18:       .  .  P = P - 1                      ! pop stack
19:       .  ELSE
20:       .  .  I = LV(P) - 1
21:       .  .  J = UV(P)
22:       .  .  PIVOT = X(J)
23:       .  .  WHILE (I .LT. J)
24:       .  .  .  I = I + 1
25:       .  .  .  WHILE (X(I) .LT. PIVOT)
26:       .  .  .  .  I = I + 1
27:       .  .  .  END WHILE
28:       .  .  .  J = J - 1
29:       .  .  .  WHILE (J .GT. I)
30:       .  .  .  .  EXIT WHILE IF (X(J) .LE. PIVOT)
31:       .  .  .  .  J = J - 1
32:       .  .  .  END WHILE
33:       .  .  .  IF (I .LT. J)                        ! out of order pair
34:       .  .  .  .  S = X(J)
35:       .  .  .  .  X(J) = X(I)
36:       .  .  .  .  X(I) = S
37:       .  .  .  .  K = T(J)
38:       .  .  .  .  T(J) = T(I) I
39:       .  .  .  .  T(I) = K
40:       .  .  .  END IF
41:       .  .  END WHILE
42:       .  .  J = UV(P)                           ! move pivot into position i
43:       .  .  S = X(J)
44:       .  .  X(J) = X(I)
```

```
45:       . .  X(I) = S
46:       . .  K = T(J)
47:       . .  T(J) = T(I)
48:       . .  T(I) = K
49:       . .  IF (I - LV(P) .LT. UV(P) - I)        ! stack so shorter done first
50:       . . .  LV(P+1) = LV(P)
51:       . . .  UV(P+1) = I - 1
52:       . . .  LV(P) = I + 1
53:       . .  ELSE
54:       . . .  LV(P+1) = I + 1
55:       . . .  UV(P+1) = UV(P)
56:       . . .  UV(P) = I - 1
57:       . .  END IF
58:       . .  P = P + 1                            ! push onto stack
59:       .  END IF
60:       END WHILE
61:
62:       RETURN
63:       END


 1: *-h- SRTPI 942 14 MAR 83  11:44:37
 2:       SUBROUTINE SRTPI(A, T, JOB, N)
 3:
 4: * rearrange elements of integer A by permutation vector T generated by
 5: *     job == 0  A[T[k]] moved to A[k]    :gather
 6: *     job ~= 0  A[k] moved to A[T[k]]    :scatter
 7:
 8:       INTEGER A(*), T(*), JOB, N
 9:       INTEGER I, J, K, S
10:
11:       IF (N .EQ. 1) RETURN
12:
13:       FOR I = 1,N
14:       .  T(I) = -T(I)
15:       END FOR
16:
17:       IF (JOB .EQ. 0)
18:
19: *  forward permutation
20:
21:       .  FOR I = 1,N
22:       . .  IF (T(I) .LT. 0)
23:       . . .  J = I
24:       . . .  T(J) = -T(J)
25:       . . .  K = T(J)
26:       . . .  WHILE (T(K) .LT. 0)
27:       . . . .  S = A(J)
28:       . . . .  A(J) = A(K)
29:       . . . .  A(K) = S
30:       . . . .  T(K) = -T(K)
31:       . . . .  J = K
32:       . . . .  K = T(K)
33:       . . .  END WHILE
34:       . .  END IF
35:       .  END FOR
36:       ELSE
37:
38: *  backward permutation
39:
40:       .  FOR I = 1,N
41:       . .  IF (T(I) .LT. 0)
42:       . . .  T(I) = -T(I)
43:       . . .  J = T(I)
44:       . . .  WHILE (J .NE. I)
45:       . . . .  S = A(I)
46:       . . . .  A(I) = A(J)
47:       . . . .  A(J) = S
48:       . . . .  T(J) = -T(J)
49:       . . . .  J = T(J)
50:       . . .  END WHILE
51:       . .  END IF
52:       .  END FOR
53:       END IF
54:
55:       RETURN
56:       END
```

```
 1: *-h- SRTPR 958 14 MAR 83  11:44:39
 2:       SUBROUTINE SRTPR(A, T, JOB, N)
 3:
 4: * rearrange elements of real A by permutation vector T generated by
 5: *     job == 0  A[T[k]] moved to A[k]    :gather
 6: *     job  = 0  A[k] moved to A[T[k]]    :scatter
 7:
 8:       INTEGER T(*), JOB, N
 9:       REAL A(*)
10:       INTEGER I, J, K
11:       REAL S
12:
13:       IF (N .EQ. 1) RETURN
14:
15:       FOR I = 1,N
16:       .  T(I) = -T(I)
17:       END FOR
18:
19:       IF (JOB .EQ. 0)
20:
21: *  forward permutation
22:
23:       .  FOR I = 1,N
24:       .  .  IF (T(I) .LT. 0)
25:       .  .  .  J = I
26:       .  .  .  T(J) = -T(J)
27:       .  .  .  K = T(J)
28:       .  .  .  WHILE (T(K) .LT. 0)
29:       .  .  .  .  S = A(J)
30:       .  .  .  .  A(J) = A(K)
31:       .  .  .  .  A(K) = S
32:       .  .  .  .  T(K) = -T(K)
33:       .  .  .  .  J = K
34:       .  .  .  .  K = T(K)
35:       .  .  .  END WHILE
36:       .  .  END IF
37:       .  END FOR
38:       ELSE
39:
40: *  backward permutation
41:
42:       .  FOR I = 1,N
43:       .  .  IF (T(I) .LT. 0)
44:       .  .  .  T(I) = -T(I)
45:       .  .  .  J = T(I)
46:       .  .  .  WHILE (J .NE. I)
47:       .  .  .  .  S = A(I)
48:       .  .  .  .  A(I) = A(J)
49:       .  .  .  .  A(J) = S
50:       .  .  .  .  T(J) = -T(J)
51:       .  .  .  .  J = T(J)
52:       .  .  .  END WHILE
53:       .  .  END IF
54:       .  END FOR
55:       END IF
56:
57:       RETURN
58:       END
```

```
 1: *-h- SVMAX 641 30 AUG 82  13:02:32
 2:       REAL FUNCTION SVMAX(U, INCU, INDEX, N)
 3:
 4: * return value and index of maximum in U
 5:
 6:       INTEGER INCU, INDEX, N
 7:       REAL U(*)
 8:       INTEGER I, IU
 9:       REAL A
10:
11:       IF (N .LE. 0)
12:       .  SVMAX = 0.0
13:       .  RETURN
14:       END IF
15:
16:       A = U(1)
17:       INDEX = 1
18:
19:       IF (INCU .EQ. 1)
20:
```

```
21: *  contiguous data
22:
23:       .  FOR I = 2,N
24:       .  .  IF (U(I) .GT. A)
25:       .  .  .  A = U(I)
26:       .  .  .  INDEX = I
27:       .  .  END IF
28:       .  END FOR
29:       .
30:       ELSE
31:
32: *  non-contiguous data
33:
34:       .  IU = 1 + INCU
35:       .  FOR I = 2,N
36:       .  .  IF (U(IU) .GT. A)
37:       .  .  .  A = U(IU)
38:       .  .  .  INDEX = IU
39:       .  .  END IF
40:       .  .  IU = IU + INCU
41:       .  END FOR
42:       .
43:       END IF
44:
45:       SVMAX = A
46:       RETURN
47:       END
```

```
 1: *-h- SVMIN 642 30 AUG 82  13:02:37
 2:       REAL FUNCTION SVMIN(U, INCU, INDEX, N)
 3:
 4: *  return value and index of minimum in U
 5:
 6:       INTEGER INCU, INDEX, N
 7:       REAL U(*)
 8:       INTEGER I, IU
 9:       REAL A
10:
11:       IF (N .LE. 0)
12:       .  SVMIN = 0.0
13:       .  RETURN
14:       END IF
15:
16:       A = U(1)
17:       INDEX = 1
18:
19:       IF (INCU .EQ. 1)
20:
21: *  contiguous data
22:
23:       .  FOR I = 2,N
24:       .  .  IF (U(I) .LT. A)
25:       .  .  .  A = U(I)
26:       .  .  .  INDEX = I
27:       .  .  END IF
28:       .  END FOR
29:       .
30:       ELSE
31:
32: *  non-contiguous data
33:
34:       .  IU = 1 + INCU
35:       .  FOR I = 2,N
36:       .  .  IF (U(IU) .LT. A)
37:       .  .  .  A = U(IU)
38:       .  .  .  INDEX = IU
39:       .  .  END IF
40:       .  .  IU = IU + INCU
41:       .  END FOR
42:       .
43:       END IF
44:
45:       SVMIN = A
46:       RETURN
47:       END
```

```
 1: *-h- SVSMAB 439 11 APR 83  23:07:53
 2:       REAL FUNCTION SVSMAB(U, INCU, N)
 3:
 4: * sum(|U[k]|)
 5:
 6:       INTEGER INCU, N
 7:       REAL U(*)
 8:       INTEGER I, IU
 9:       REAL C
10:
11:       IF (INCU .EQ. 1)
12:
13: *  contiguous data
14:
15:       .  C = 0.0
16:       .  FOR I = 1,N
17:       .  .  C = C + ABS(U(I))
18:       .  END FOR
19:       .
20:       ELSE
21:
22: *  non-continguous data
23:
24:       .  IU = 1
25:       .  C = 0.0
26:       .  FOR I = 1,N
27:       .  .  C = C + ABS(U(IU))
28:       .  .  IU = IU + INCU
29:       .  END FOR
30:       .
31:       END IF
32:
33:       SVSMAB = C
34:       RETURN
35:       END
```

```
 1: *-h- SVSUM 424 11 APR 83  23:07:55
 2:       REAL FUNCTION SVSUM(U, INCU, N)
 3:
 4: * sum(U[k])
 5:
 6:       INTEGER INCU, N
 7:       REAL U(*)
 8:       INTEGER I, IU
 9:       REAL C
10:
11:       IF (INCU .EQ. 1)
12:
13: *  contiguous data
14:
15:       .  C = 0.0
16:       .  FOR I = 1,N
17:       .  .  C = C + U(I)
18:       .  END FOR
19:       .
20:       ELSE
21:
22: *  non-contiguous data
23:
24:       .  IU = 1
25:       .  C = 0.0
26:       .  FOR I = 1,N
27:       .  .  C = C + U(IU)
28:       .  .  IU = IU + INCU
29:       .  END FOR
30:       .
31:       END IF
32:
33:       SVSUM = C
34:       RETURN
35:       END
```

```
 1: *-h- VCOPY 534 12 APR 83  12:52:29
 2:       SUBROUTINE VCOPY(Y, INCY, U, INCU, N)
 3:
 4: * Y[k] = U[k]
 5:
 6:       INTEGER INCY, INCU, N
 7:       REAL Y(*), U(*)
 8:       INTEGER I, IY, IU
 9:
10:       IF (INCY .EQ. 1 .AND. INCU .EQ. 1)
11:
12: *  contiguous data
13:
14:       .  FOR I = 1,N
15:       .  .  Y(I) = U(I)
16:       .  END FOR
17:       .
18:       ELSE
19:
20: * non-contiguous data
21:
22:       .  IY = 1
23:       .  IU = 1
24:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
25:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
26:       .
27:       .  FOR I = 1,N
28:       .  .  Y(IY) = U(IU)
29:       .  .  IY = IY + INCY
30:       .  .  IU = IU + INCU
31:       .  END FOR
32:       .
33:       END IF
34:
35:       RETURN
36:       END
```

```
 1: *-h- VDIV 765 12 APR 83  12:52:57
 2:       SUBROUTINE VDIV(Y,INCY,A,U,INCU,B,V,INCV,C,N)
 3:
 4: * Y[k] = (a / (b+U[k])) * (c+V[k])
 5:
 6:       INTEGER INCY, INCU, INCV, N
 7:       REAL Y(*), U(*), V(*), A, B, C
 8:       INTEGER I, IY, IU, IV
 9:
10:       IF (N .LE. 0) RETURN
11:
12:       IF (INCY .EQ. 1 .AND. INCU .EQ. 1 .AND. INCV .EQ. 1)
13:
14: *  contiguous data
15:
16:       .  FOR I = 1,N
17:       .  .  Y(I) = (A / (U(I) + B)) * (V(I) + C)
18:       .  END FOR
19:       .
20:       ELSE
21:
22: * non-contiguous data
23:
24:       .  IY = 1
25:       .  IU = 1
26:       .  IV = 1
27:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
28:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
29:       .  IF (INCV .LT. 0) IV = (-N+1) * INCV + 1
30:       .
31:       .  FOR I = 1,N
32:       .  .  Y(IY) = (A / (U(IU) + B)) * (V(IV) + C)
33:       .  .  IY = IY + INCY
34:       .  .  IU = IU + INCU
35:       .  .  IV = IV + INCV
36:       .  END FOR
37:       .
38:       END IF
39:
40:       RETURN
41:       END
```

```
 1: *-h- VFIX 595 12 APR 83  12:53:28
 2:       SUBROUTINE VFIX(Y, INCY, U, INCU, N)
 3:
 4: * convert vector to integer
 5:
 6:       INTEGER Y(*), INCY, INCU, N
 7:       REAL U(*)
 8:       INTEGER I
 9:       INTEGER INT
10:
11:       IF (N .LE. 0) RETURN
12:
13:       IF (INCY .EQ. 1 .AND. INCU .EQ. 1)
14:
15: *  coniguous data
16:
17:       .  FOR I = 1,N
18:       .  .  Y(I) = INT(U(I))
19:       .  END FOR
20:       .
21:       ELSE
22:
23: *  non-contiguous data
24:
25:       .  IY = 1
26:       .  IU = 1
27:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
28:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
29:       .
30:       .  FOR I = 1,N
31:       .  .  Y(IY) = INT(U(IU))
32:       .  .  IY = IY + INCY
33:       .  .  IU = IU + INCU
34:       .  END FOR
35:       .
36:       END IF
37:
38:       RETURN
39:       END
```

```
 1: *-h- VFLOAT 598 11 APR 83  23:34:18
 2:       SUBROUTINE VFLOAT(Y, INCY, U, INCU, N)
 3:
 4: * convert vector to real
 5:
 6:       REAL Y(*)
 7:       INTEGER U(*), INCY, INCU, N
 8:       REAL FLOAT
 9:       INTEGER I
10:
11:       IF (N .LE. 0) RETURN
12:
13:       IF (INCY .EQ. 1 .AND. INCU .EQ. 1)
14:
15: *  contiguous data
16:
17:       .  FOR I = 1,N
18:       .  .  Y(I) = FLOAT(U(I))
19:       .  END FOR
20:       .
21:       ELSE
22:
23: *  non-contiguous data
24:
25:       .  IY = 1
26:       .  IU = 1
27:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
28:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
29:       .
30:       .  FOR I = 1,N
31:       .  .  Y(IY) = FLOAT(U(IU))
32:       .  .  IY = IY + INCY
33:       .  .  IU = IU + INCU
34:       .  END FOR
35:       .
36:       END IF
37:
38:       RETURN
39:       END
```

```
 1: *-h- VINDEX 666 20 MAY 83  13:44:50
 2:       SUBROUTINE VINDEX(Y, INCY, A, B, U, INCU, N)
 3:
 4: * linear weighting of vector U  (integer version)
 5: * Y[k] = (a + bk) * U[k]
 6:
 7:       INTEGER Y(*), INCY, A, B, U(*), INCU, N
 8:       INTEGER I, IU, IY
 9:
10:       IF (N .LE. 0) RETURN
11:
12:       IF (INCY .EQ. 1 .AND. INCU .EQ. 1)
13:
14: *  contiguous data
15:
16:       .  FOR I = 1,N
17:       .  .  Y(I) = (A + (I- 1) * B) * U(I)
18:       .  END FOR
19:       .
20:       ELSE
21:
22: *  non-contiguous data
23:
24:       .  IY = 1
25:       .  IU = 1
26:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
27:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
28:       .
29:       .  FOR I = 1,N
30:       .  .  Y(IY) = (A + (I - 1) * B) * U(IU)
31:       .  .  IY = IY + INCY
32:       .  .  IU = IU + INCU
33:       .  END FOR
34:       .
35:       END IF
36:
37:       RETURN
38:       END
```

```
 1: *-h- VMUL 751 11 APR 83  23:34:23
 2:       SUBROUTINE VMUL(Y, INCY, A, U, INCU, B, V, INCV, C, N)
 3:
 4: * Y[k] = a * (b+U[k]) * (c+V[k])
 5:
 6:       REAL Y(*),U(*),V(*),A,B,C
 7:       INTEGER I, INCY, INCU, INCV, IY, IU, IV, N
 8:
 9:       IF (N .LE. 0) RETURN
10:
11:       IF (INCY .EQ. 1 .AND. INCU .EQ. 1 .AND. INCV .EQ. 1)
12:
13: *  contiguous data
14:
15:       .  FOR I = 1,N
16:       .  .  Y(I) = A * (B + U(I)) * (C + V(I))
17:       .  END FOR
18:       .
19:       ELSE
20:
21: *  non-contiguous data
22:
23:       .  IY = 1
24:       .  IU = 1
25:       .  IV = 1
26:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
27:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
28:       .  IF (INCV .LT. 0) IV = (-N+1) * INCV + 1
29:       .
30:       .  FOR I = 1,N
31:       .  .  Y(IY) = A * (U(IU) + B) * (V(IV) + C)
32:       .  .  IY = IY + INCY
33:       .  .  IU = IU + INCU
34:       .  .  IV = IV + INCV
35:       .  END FOR
36:       .
37:       END IF
38:
39:       RETURN
40:       END
```

```
 1: *-h- VRCP 601 11 APR 83  23:34:26
 2:       SUBROUTINE VRCP(Y, INCY, A, U, INCU, B, N)
 3:
 4: * Y[k] = a / (U[k] + b)
 5:
 6:       INTEGER INCY, INCU, N
 7:       REAL Y(*), U(*), A
 8:       INTEGER IY, IU, I
 9:
10:       IF (N .LE. 0) RETURN
11:
12:       IF (INCY .EQ. 1 .AND. INCU .EQ. 1)
13:
14: *  contiguous data
15:
16:       .  FOR I = 1,N
17:       .  .  Y(I) = A / (U(I) + B)
18:       .  END FOR
19:       .
20:       ELSE
21:
22: *  non-contiguous data
23:
24:       .  IY = 1
25:       .  IU = 1
26:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
27:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
28:       .
29:       .  FOR I = 1,N
30:       .  .  Y(IY) = A / (U(IU) + B)
31:       .  .  IY = IY + INCY
32:       .  .  IU = IU + INCU
33:       .  END FOR
34:       .
35:       END IF
36:
37:       RETURN
38:       END
```

```
 1: *-h- VROT 746 20 MAY 83  13:45:05
 2:       SUBROUTINE VROT(X, INCX, Y, INCY, C, S, N)
 3:
 4: * apply a plane rotation
 5:
 6:       INTEGER INCX, INCY, N
 7:       REAL X(*), Y(*), C, S
 8:       REAL STEMP
 9:       INTEGER I, IX, IY
10:
11:       IF (N .LE. 0) RETURN
12:
13:       IF (INCX .EQ. 1 .AND. INCY .EQ. 1)
14:
15: *  contiguous data
16:
17:       .  FOR I = 1,N
18:       .  .  STEMP = C * X(I) + S * Y(I)
19:       .  .  Y(I) = C * Y(I) - S * X(I)
20:       .  .  X(I) = STEMP
21:       .  END FOR
22:       .
23:       ELSE
24:
25: *  non-contiguous data
26:
27:       .  IX = 1
28:       .  IY = 1
29:       .  IF (INCX .LT. 0) IX = (-N + 1) * INCX + 1
30:       .  IF (INCY .LT. 0) IY = (-N + 1) * INCY + 1
31:       .
32:       .  FOR I = 1,N
33:       .  .  STEMP = C * X(IX) + S * Y(IY)
34:       .  .  Y(IY) = C * Y(IY) - S * X(IX)
35:       .  .  X(IX) = STEMP
36:       .  .  IX = IX + INCX
37:       .  .  IY = IY + INCY
38:       .  END FOR
39:       .
40:       END IF
```

```
41:
42:       RETURN
43:       END


 1: *-h- VSCALE 163 17 DEC 82  20:50:39
 2:       SUBROUTINE VSCALE(Y, INCY, A, N)
 3:
 4: * Y[k] = a*Y[k]
 5:
 6:       INTEGER INCY, N
 7:       REAL Y(*), A
 8:
 9:       CALL VSMA1(Y,INCY,A,Y,INCY,0.0,N)
10:
11:       RETURN
12:       END


 1: *-h- VSMA1 588 30 AUG 82  13:03:00
 2:       SUBROUTINE VSMA1(Y,INCY,A,U,INCU,B,N)
 3:
 4: * Y[k] = a*U[k] + b
 5:
 6:       INTEGER INCY, INCU, N
 7:       REAL Y(*),U(*),A,B
 8:       INTEGER I, IY, IU
 9:
10:       IF (N .LE. 0) RETURN
11:
12:       IF (INCY .EQ. 1 .AND. INCU .EQ. 1)
13:
14: *  contiguous data
15:
16:       .  FOR I = 1,N
17:       .  .  Y(I) = A * U(I) + B
18:       .  END FOR
19:       .
20:       ELSE
21:
22: *  non-contiguous data
23:
24:       .  IY = 1
25:       .  IU = 1
26:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
27:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
28:       .
29:       .  FOR I = 1,N
30:       .  .  Y(IY) = A * U(IU) + B
31:       .  .  IY = IY + INCY
32:       .  .  IU = IU + INCU
33:       .  END FOR
34:       .
35:       END IF
36:
37:       RETURN
38:       END


 1: *-h- VSMA2 732 30 AUG 82  13:03:01
 2:       SUBROUTINE VSMA2(Y,INCY,A,U,INCU,B,V,INCV,C,N)
 3:
 4: * Y[k] = a*U[k] + b*V[k] + c
 5:
 6:       REAL Y(*),U(*),V(*),A,B,C
 7:       INTEGER I, INCY, INCU, INCV, IY, IU, IV, N
 8:
 9:       IF (N .LE. 0) RETURN
10:
11:       IF (INCY .EQ. 1 .AND. INCU .EQ. 1 .AND. INCV .EQ. 1)
12:
13: *  contiguous data
14:
15:       .  FOR I = 1,N
16:       .  .  Y(I) =  A * U(I) + B * V(I) + C
17:       .  END FOR
18:       .
19:       ELSE
20:
```

```
21: *  non-contiguous data
22:
23:       .  IY = 1
24:       .  IU = 1
25:       .  IV = 1
26:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
27:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
28:       .  IF (INCV .LT. 0) IV = (-N+1) * INCV + 1
29:       .
30:       .  FOR I = 1,N
31:       .  .  Y(IY) = A * U(IU) + B * V(IV) + C
32:       .  .  IY = IY + INCY
33:       .  .  IU = IU + INCU
34:       .  .  IV = IV + INCV
35:       .  END FOR
36:       .
37:       END IF
38:
39:       RETURN
40:       END


 1: *-h- VSQRT 583 11 APR 83  23:08:03
 2:       SUBROUTINE VSQRT(Y, INCY, A, U, INCU, B, N)
 3:
 4: * Y[k] = a * sqrt(U[k]+b)
 5:
 6:       INTEGER INCY, INCU, N
 7:       REAL Y(*), U(*), A, B
 8:       INTEGER I, IY, IU
 9:
10:       IF (INCY .EQ. 1 .AND. INCU .EQ. 1)
11:
12: *  contiguous data
13:
14:       .  FOR I = 1,N
15:       .  .  Y(I) = A * SQRT(U(I)+B)
16:       .  END FOR
17:       .
18:       ELSE
19:
20: *  non-contiguous data
21:
22:       .  IY = 1
23:       .  IU = 1
24:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
25:       .  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
26:       .
27:       .  FOR I = 1,N
28:       .  .  Y(IY) = A * SQRT(U(IU)+B)
29:       .  .  IY = IY + INCY
30:       .  .  IU = IU + INCU
31:       .  END FOR
32:       .
33:       END IF
34:
35:       RETURN
36:       END


 1: *-h- VSWAP 636 11 APR 83  23:08:04
 2:       SUBROUTINE VSWAP(X,INCX,Y,INCY,N)
 3:
 4: * interchange 2 vectors
 5:
 6:       REAL X(*), Y(*), TEMP
 7:       INTEGER I, INCX, INCY, IX, IY, N
 8:
 9:       IF (N .LE. 0) RETURN
10:
11:       IF (INCX .EQ. 1 .AND. INCY .EQ. 1)
12:
13: *  contiguous data
14:
15:       .  FOR I = 1,N
16:       .  .  TEMP = X(I)
17:       .  .  X(I) = Y(I)
18:       .  .  Y(I) = TEMP
19:       .  END FOR
20:       .
```

```
21:       ELSE
22:
23: *  non-contiguous data
24:
25:       .  IX = 1
26:       .  IY = 1
27:       .  IF (INCX .LT. 0) IX = (-N+1) * INCX + 1
28:       .  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
29:       .
30:       .  FOR I = 1,N
31:       .  .  TEMP = X(IX)
32:       .  .  X(IX) = Y(IY)
33:       .  .  Y(IY) = TEMP
34:       .  .  IX = IX + INCX
35:       .  .  IY = IY + INCY
36:       .  END FOR
37:       .
38:       END IF
39:
40:       RETURN
41:       END
```

A.8 Plotting subroutine package.

The heart of an interactive system is the graphic display facilities. The HP2647A graphics terminal allows the rapid display of data and enables decisions to be made regarding the processing as it is being done. We had no package of Fortran callable procedures for graphics on this terminal. In hindsight it might have been better to inquire with Hewlett-Packard about such a graphics package but instead I developed a library of routines for my own use. There are certain benefits to this approach. First of all I have learned a great deal about computer graphics. but more important to the present work, the routines are tailored to my needs. As a result of my amateur effort, there are a few shortcomings to the graphics package but they are not serious.

The HP2647A terminal interprets certain character sequences as graphics commands. The terminal has some fairly advanced features such as electronic cursor, rubber band line, masking and zoom. The programming task was therefore to map higher level plotting functions into character strings and buffer them to the terminal. Since Fortran I/O is a serious bottleneck in terms of speed. characters are stored in memory until the maximum number may be trasmitted at once. Hewlett-Packard has classified commands into a limited number of types. Plotting commands must specify the class and the member of the class as well as the coordinates if those are required. Plotting commands of the same type may be given with only a single reference to the class so it is

efficient to group operations of a given type. Commands of a type different than the previous one must specify all information (see the HP2647A manual for more information [Hewlett-Packard, 1978]).

At the same time as I was trying to develop a system for plotting on the video display I was frustrated with the inflexibility of the routines in the Calcomp-compatible Versaplot library for the Versatec electrostatic plotter [Versatec, 1981]. All routines developed for the CRT are duplicated for the Versatec if the operations are possible on a hard copy device. Some operations that are excluded are zooming, cursor and interaction.

The plotting package may be divided into two levels: the upper level of device independent procedures, and the lower level of routines particular to either the HP terminal or the Versatec plotter. The upper level routines are described first.

Certain geometrical figures are drawn frequently. Routines ARROW, BOX and CIRCLE draw such figures at given coordinates, with given size and given orientation. Data usually is stored in programs as vectors or matrices. Routine PVEC1 draws a line representing a vector of data as a function of an incremental independent variable. The calling program must specify the starting coordinates, the scale, the reference value for the data, and the direction the plot is to take. PVEC2 draws a vector of data as a function of a second vector of data. Again the starting coordinates, the scale, and the reference value must be specified but not the direction. Routine PNTCNF draws a single datum with its error bars.

Routine PMAT is used to represent the coefficients of a matrix. The columns of the matrix are drawn as continuous lines.

For data plots to be interpreted, there must be some indication of scale. Routine PAXIS draws on axis. This routine has fourteen arguments in the calling sequence because I desired total control over the location, orientation and annotation of the axis. The physical limits of the axis are specified by the variables xmin, xmax, ymin and ymax. The annotation limits are specified by smin and smax. The format of the annotation values are specified by field and dec for the number of digits and height, ang and aopt for the direction. The variable aopt is an option controlling the direction the digits are read and is described further for routine TIC. The annotation limits may be optionally rescaled by PAXIS. Graphs are best displayed with annotation increments in multiples of 2, 5 or 10. The option sopt controls the rescaling, with sopt=0 being no rescaling, sopt=1 meaning recalculating plot limits so that approximately n annotations are provided. and sopt=2 meaning that exactly n annotations are provided. The best results are given by sopt=1; the plot is more completely filled. Routine TIC is called by PAXIS to draw the tic marks on the axis and write the numbers. An x,y coordinate is specified for the tic mark as well as an angle which is measured from horizontal. The variable aopt controls the direction of the numbers relative to the tic mark. If aopt<1 or aopt>4 then no number will be drawn, if aopt=1 the most significant digit start at the tic mark and extends away from the axis. if aopt=2 the least significant digit is near the tic mark, if aopt=3 the number is plotted at right angles to the

tic with the top of the number near the axis. and if aopt=4 the number is plotted at right angles with the base of the number near the axis. As an example, the latitudes and longtitudes in Figure 6 are plotted with aopt=1 for the east, aopt=2 for the west, aopt=3 for the south. and aopt=4 for the north.Routines SCALE1 and SCALE2 are called by PAXIS to calculate the new scale values for sopt=1 and sopt=2 respectively. These two routines are adapted from Lewart [1973].

The lower level routines are stored in separate libraries so the user can link the applications program to the appropriate library depending on whether video or hardcopy output is desired. The video plotting routines are described. An asterisk on the routine name signifies that an equivalent routine is present in the library for hardcopy plotting.

The first plotting operation must be an initialization with INIPLT* which erases the screen. converts the terminal to graphics mode, and sets some global variables. A single figure is terminated by ENDFIG* which may be called with an argument of 0 or 1. This causes the output buffer to be transmitted to the display and then clears the screen. If ENDFIG is called with 1 then the terminal will wait for the user to enter a carriage return before clearing the screen. A call to RESTOR returns the terminal to graphics mode after a call to ENDFIG. If RESTOR is called with 1 it erases the display memory otherwise it restores the old picture. All plotting must be terminated by ENDPLT*

which restores alphanumeric mode on the terminal and clears the graphics memory.

Pen movement is controlled by MOVEPN*. This moves the pen position to the given coordinate (given in inches) with the pen up pen=0 or down pen=1. Plotting limits may be specified and lines drawn outside those limits truncated. This feature is not used by any of the included processing routines so the subroutines are not included. MOVEPN calls MOVEPI to send the line coordinates to the output buffer. LINTYP sets a mask for dashed and dotted lines. SETDRM sets the drawing mode. FILREC colors a rectangular area.

Numbers are plotted with GRNUM*, symbols with GRSYMB*, and character strings with GRSTR*. Note that the character strings in the calling program are integer equivalents (Hollerith data) rather than character variables so the Fortran compiler must be informed. hence the ":HCONST" construct in some of the calling programs.

Other miscellaneous routines are RBLIN to turn on the rubber band line which follows the motion of the graphics cursor; RDMTAB to retrieve certain information from the plot mode table; and GRID to draw a grid. GRID is a high level plotting function but is placed here so as not to preempt a compatible call in the Versaplot library.

Certain calls in the Versaplot library are included in the library as dummy routines to ensure that a program that was originally written for a Versatec electrostatic plotter can be relinked to execute plots on the Hewlett-Packard without any program changes. These dummy routines

are VP07MP, PLOTS and NEWPEN. Routine PLOT does the pen movements for the Versaplot programs but it is not used in any of the programs running on the video terminal in this text.

Still lower level routines are called by the above routines to execute the data transfer between the computer and the terminal. Routine OUTHPP stores plotting coordinates and command types in an output buffer until the buffer is full or the command type is changed. When either one of these events occurs FLUSH is called to transfer the data to the terminal and to reset the pointers. It may be additionally called if an immediate response is required from the graphics. Routine OUTHPT is used to output textual information which is handled slightly differently. Routines that require information from the terminal call INHP to transfer data from the terminal to the computer. INHP requires an assembly language reference to the VOS I/O service because it needs to interpret a single key stroke rather than a whole line buffer and Fortran does not provide that facility.

```
 1: *-h- ARROW 868 24 SEP 82  16:10:33
 2:       SUBROUTINE ARROW(X, Y, SIZE, ANG)
 3:
 4: * draw arrow with point at <x,y>
 5: *    routines required:  movepn,  (hpplb or vplb)
 6: *                        vrot     (mathlb)
 7:
 8:       REAL X, Y, SIZE, ANG
 9:       REAL CA, SA, XX, YY
10:       REAL DR
11:       INTEGER OO, PENDOWN, PENUP
12:       DATA DR /0.01745329/, PENDOWN /1/, PENUP /0/
13:
14:       CA = COS(ANG*DR)
15:       SA = SIN(ANG*DR)
16:       OO = INT(RDMTAB(17,2))
17:
18:       CALL MOVEPN(X,Y,OO,PENUP)
19:
20:       XX = 0.4 * SIZE
21:       YY = 0.1 * SIZE
22:       CALL VROT(XX,1.YY,1.CA,SA,1)
23:       CALL MOVEPN(X+XX,Y+YY,OO,PENDOWN)
24:
25:       XX = 0.4 * SIZE
26:       YY = -0.1 * SIZE
27:       CALL VROT(XX,1.YY,1.CA,SA,1)
28:       CALL MOVEPN(X+XX,Y+YY,OO,PENDOWN)
29:
30:       CALL MOVEPN(X,Y,OO,PENDOWN)
31:
32:       XX = SIZE
33:       YY = 0.0
34:       CALL VROT(XX,1.YY,1.CA,SA,1)
35:       CALL MOVEPN(X+XX,Y+YY,OO,PENDOWN)
36:
37:       CALL MOVEPN(X,Y,OO,PENUP)
38:
39:       RETURN
40:       END
```

```
 1: *-h- AXLAB 2228  6 MAY 83  16:05:50
 2:       SUBROUTINE AXLAB(XMIN,XMAX,YMIN,YMAX,LAB,HEIGHT,OFFSET)
 3:
 4: ************************************************************************
 5: *                                                                      *
 6: * draw label for axis                                                  *
 7: *                                                                      *
 8: *     XMIN   - input real  }                                           *
 9: *     XMAX   - input real  } minimum and maximum coordinates of        *
10: *     YMIN   - input real  }  the associated axis                      *
11: *     YMAX   - input real  }                                           *
12: *     LAB    - input integer character string terminated by default    *
13: *                    delimiter                                         *
14: *     HEIGHT - input real character height                             *
15: *     OFF    - input real distance of label from axis                  *
16: *                                                                      *
17: ************************************************************************
18:
19:       INTEGER LAB(*)
20:       REAL XMIN, XMAX, YMIN, YMAX, HEIGHT, OFFSET
21:       REAL PMODE(25,2)
22:       COMMON /CPMTAB/ PMODE
23:       INTEGER DELIM, I, IN(81), J, LEN, OUT(27)
24:       REAL ANG, CA, SA, X, Y
25:       INTEGER ESC
26:       REAL RDMTAB
27:
28:       DELIM = INT(RDMTAB(19,2))
29:       CALL UNPACK(LAB,IN.27)
30:
31: *  find length of string
32:
33:       I = 1
34:       J = 1
35:       LEN = 0
36:       WHILE (I .LT. 81)
37:       .  IF (IN(I) .EQ. 64)       ! escaped character
38:       .  .  IN(J) = ESC(IN.I)
```

```
39:       .  ELSE IF (IN(I) .EQ. 110 .OR. IN(I) .EQ. 111) ! super or subscript
40:       .  .  LEN = LEN - 1
41:       .  .  IN(J) = IN(I)
42:       .  ELSE
43:       .  .  IN(J) = IN(I)
44:       .  END IF
45:       .  IF (IN(J) .EQ. DELIM)
46:       .  .  IN(J) = -2
47:       .  .  EXIT WHILE
48:       .  END IF
49:       .  I = I + 1
50:       .  J = J + 1
51:       .  LEN = LEN + 1
52:       END WHILE
53:
54:       ANG = ATAN2(YMAX-YMIN,XMAX-XMIN)
55:       CA = COS(ANG)
56:       SA = SIN(ANG)
57:       ANG = 57.29578 * ANG
58:
59:       X = (XMIN+XMAX)/2.0-CA*FLOAT(LEN)*HEIGHT/2.0-SA*OFFSET
60:       Y = (YMIN+YMAX)/2.0-SA*FLOAT(LEN)*HEIGHT/2.0+CA*OFFSET
61:
62: *  draw characters
63:
64:       CALL NEWPEN(3)
65:       CALL PACK(IN,OUT)
66:       CALL GRSTR(X,Y,OUT,J-1,HEIGHT,ANG)
67:       CALL NEWPEN(1)
68:
69:       RETURN
70:       END


 1: *-h- BOX 693 24 SEP 82  16:10:37
 2:       SUBROUTINE BOX(XLO, YLO, BASE, HEIGHT, ANG)
 3:
 4: * draw a rectangle with corner at <xlo,ylo>
 5:
 6:       REAL XLO, YLO, BASE, HEIGHT, ANG
 7:       REAL CA, DR, SA, XX, YY
 8:       INTEGER OO
 9:       DATA DR /0.017453/
10:
11:       CA = COS(ANG*DR)
12:       SA = SIN(ANG*DR)
13:       OO = INT(RDMTAB(17,2))
14:
15:       CALL MOVEPN(XLO,YLO,OO,0)
16:
17:       XX = BASE
18:       YY = 0.0
19:       CALL VROT(XX,1,YY,1,CA,SA,1)
20:       CALL MOVEPN(XLO+XX,YLO+YY,OO,1)
21:
22:       XX = BASE
23:       YY = HEIGHT
24:       CALL VROT(XX,1,YY,1,CA,SA,1)
25:       CALL MOVEPN(XLO+XX,YLO+YY,OO,1)
26:
27:       XX = 0.0
28:       YY = HEIGHT
29:       CALL VROT(XX,1,YY,1,CA,SA,1)
30:       CALL MOVEPN(XLO+XX,YLO+YY,OO,1)
31:
32:       CALL MOVEPN(XLO,YLO,OO,1)
33:
34:       RETURN
35:       END


 1: *-h- CIRCLE 528 24 SEP 82  16:10:38
 2:       SUBROUTINE CIRCLE(X, Y, R)
 3:
 4: * draw a circle at <x,y> with radius r
 5:
 6:       REAL X, Y, R
 7:       REAL XX, YY, ANG, DANG, PIT2, PINCPI
 8:       INTEGER OO
 9:       DATA PIT2 /6.2831853/
```

```
10:
11:       PINCPI = RDMTAB(15,2)
12:       OO = INT(RDMTAB(17,2))
13:       DANG = PIT2 / (PINCPI * R)
14:
15:       ANG = DANG
16:       CALL MOVEPN(X+R.Y,OO.0)
17:
18:       WHILE (ANG .LE. PIT2)
19:       .  XX = X + R * COS(ANG)
20:       .  YY = Y + R * SIN(ANG)
21:       .  CALL MOVEPN(XX,YY,OO,1)
22:       .  ANG = ANG + DANG
23:       END WHILE
24:       CALL MOVEPN(X+R.Y,OO.1)
25:
26:       RETURN
27:       END
```

```
 1: *-h- ENDFIG 544  1 AUG 82  14:23:48
 2:       SUBROUTINE ENDFIG(WAIT)
 3:
 4: * end single plot on HP 2647A terminal, wait for <CR> if wait=1
 5:
 6:       INTEGER WAIT
 7:       REAL PMODE(25,2)
 8:       COMMON /CPMTAB/ PMODE
 9:       INTEGER PS, STR(5)
10:       REAL X, Y
11:
12:       IF (WAIT .EQ. 1)
13:       .  PMODE(17,2) = 1.0                ! set coordinate to absolute
14:       .  CALL GRSYMB(0.3.0.3.63.0.25,0.0)
15:       .  READ (0,)
16:       END IF
17:
18:       STR(1) = 100
19:       STR(2) = 100
20:       STR(3) = 101
21:       STR(4) = 104
22:       STR(5) = -2
23:       PS = 4
24:
25:       CALL OUTHPP(STR.PS)
26:
27:       CALL FLUSH
28:
29:       RETURN
30:       END
```

```
 1: *-h- ENDFIG 149 11 JUN 82  14:41:24
 2:       SUBROUTINE ENDFIG(WAIT)
 3:
 4: * end current plot
 5: *  interface to VPLOT07
 6:
 7:       INTEGER WAIT
 8:
 9:       CALL PLOT(0.0,0.0,-999)
10:
11:       RETURN
12:       END
```

```
 1: *-h- ENDPLT 281 16 FEB 82  21:07:28
 2:       SUBROUTINE ENDPLT
 3:
 4: * return HP 2647A terminal to normal alphanumeric mode
 5:
 6:       INTEGER PS, STR(5)
 7:
 8:       STR(1) = 100
 9:       STR(2) = 97
10:       STR(3) = 100
11:       STR(4) = 101
12:       STR(5) = -2
13:       PS = 4
14:       CALL OUTHPP(STR.PS)
```

```
15:
16:       CALL FLUSH
17:
18:       RETURN
19:       END
```

```
 1: *-h- ENDPLT 126 11 JUN 82  14:41:24
 2:        SUBROUTINE ENDPLT
 3:
 4: * end all plotting
 5: *  interface to VPLOT07
 6:
 7:        CALL PLOT(0.0,0.0,999)
 8:
 9:        RETURN
10:        END
```

```
 1: *-h- FILREC 925  1 AUG 82  14:23:50
 2:       SUBROUTINE FILREC(XLO, YLO, XHI, YHI)
 3:
 4: * fill rectangle with pattern defined by procedure ARETYP
 5:
 6:       INTEGER COOR
 7:       REAL XHI, XLO, YHI, YLO
 8:       REAL PMODE(25,2)
 9:       COMMON /CPMTAB/ PMODE
10:       INTEGER STR(25), PS, L, CXY(4)
11:
12:       STR(1) = 109
13:       STR(2) = 32
14:       PS = 3
15:
16:       CXY(1) = INT(XLO*PMODE(1,1)*PMODE(15,1)*PMODE(15,2))
17:       CXY(2) = INT(YLO*PMODE(1,2)*PMODE(15,1)*PMODE(15,2))
18:       CXY(3) = INT(XHI*PMODE(1,1)*PMODE(15,1)*PMODE(15,2))
19:       CXY(4) = INT(YHI*PMODE(1,2)*PMODE(15,1)*PMODE(15,2))
20:
21:       FOR I = 1,4
22:       .  L = ITOC(CXY(I),STR(PS),4)
23:       .  PS = PS + L
24:       .  STR(PS) = 32
25:       .  PS = PS + 1
26:       END FOR
27:
28: *  set coordinate type
29:
30:       IF (PMODE(17,2) .EQ. 1.0)
31:       .  STR(PS) = 101
32:       ELSE IF (PMODE(17,2) .EQ. 3.0)
33:       .  STR(PS) = 102
34:       ELSE
35:       .  CALL ERROR(" illegal coordinate type")
36:       END IF
37:       STR(PS+1) = -2
38:
39:       CALL OUTHPP(STR,PS)
40:
41:       RETURN
42:       END
```

```
 1: *-h- FLUSH 378  1 AUG 82  15:02:22
 2:       SUBROUTINE FLUSH
 3:
 4: * clear output buffer for HP plotting library
 5:
 6:       INTEGER PLTBUF(81), PB
 7:       COMMON /CHPOUT/ PLTBUF, PB
 8:       INTEGER I, PBUF(27)
 9:
10:       PB = PB + 1
11:       PLTBUF(PB) = 90
12:       PLTBUF(PB+1) = -2
13:       FOR I = 1,27
14:       .  PBUF(I) = 0
15:       END FOR
16:       CALL PACK(PLTBUF,PBUF)
17:       WRITE (3,FMT='(1X,27A3)') PBUF
```

```
18:       PB = 3
19:
20:       RETURN
21:       END
```

```
 1: *-h- GRCURS 463  4 OCT 82  15:07:30
 2:       SUBROUTINE GRCURS(CURS)
 3:
 4: * graphics cursor: 1=on, 0=off
 5:
 6:       INTEGER CURS
 7:       REAL PMODE(25,2)
 8:       COMMON /CPMTAB/ PMODE
 9:       INTEGER STR(3), PS
10:
11:       STR(1) = 100
12:       IF (CURS .EQ. 1)
13:       .  STR(2) = 107
14:       ELSE IF (CURS .EQ. 0)
15:       .  STR(2) = 108
16:       END IF
17:       STR(3) = -2
18:       PS = 2
19:
20:       CALL OUTHPP(STR.PS)
21:       CALL FLUSH                    !  temporary
22:
23: *  update mode table
24:
25:       PMODE(16.2) = FLOAT(CURS)
26:
27:       RETURN
28:       END
```

```
 1: *-h- GRID 1236  1 AUG 82  14:23:52
 2:       SUBROUTINE GRID(X, Y, NX, XD, NY, YD, MASK)
 3:
 4: * draw rectangular grid with origin at <x,y> (relative)
 5:
 6:       INTEGER NX, NY, MASK
 7:       REAL X, XD(1), Y, YD(1)
 8:       REAL PMODE(25,2)
 9:       COMMON /CPMTAB/ PMODE
10:       INTEGER OO
11:       REAL XLEN, XX, YLEN, YY
12:       LOGICAL VARSPX, VARSPY
13:
14:       OO = INT(PMODE(17,2))
15:       VARSPX = NX .GT. 1000
16:       VARSPY = NY .GT. 1000
17:
18:       IF (VARSPX)
19:       .  XLEN = SVSUM(XD,1.MOD(NX,1000))
20:       ELSE
21:       .  XLEN = NX * XD(1)
22:       END IF
23:       IF (VARSPY)
24:       .  YLEN = SVSUM(YD,1.MOD(NY,1000))
25:       ELSE
26:       .  YLEN = NY * YD(1)
27:       END IF
28:
29:       CALL LINTYP(MASK,1)
30:
31:       CALL MOVEPN(X,Y,OO.0)
32:       XX = X
33:       YY = Y
34:
35:       CALL MOVEPN(XX+XLEN.YY,OO.1)
36:       FOR I = 1,MOD(NY,1000)
37:       .  IF (VARSPY)
38:       .  .  YY = YY + YD(I)
39:       .  ELSE
40:       .  .  YY = YY + YD(1)
41:       .  END IF
42:       .  CALL MOVEPN(XX,YY,OO.0)
43:       .  CALL MOVEPN(XX+XLEN.YY,OO.1)
44:       END FOR
```

```
45:
46:       CALL MOVEPN(X,Y,00.0)
47:       XX = X
48:       YY = Y
49:
50:       CALL MOVEPN(XX,YY+YLEN.00.1)
51:       FOR I = 1,MOD(NX,1000)
52:       .  IF (VARSPX)
53:       .  .  XX = XX + XD(I)
54:       .  ELSE
55:       .  .  XX = XX + XD(1)
56:       .  END IF
57:       .  CALL MOVEPN(XX,YY,00.0)
58:       .  CALL MOVEPN(XX,YY+YLEN.00.1)
59:       END FOR
60:
61:       CALL MOVEPN(X,Y,00.0)
62:
63:       RETURN
64:       END


 1: *-h- GRNUM 1144  1 AUG 82  14:24:12
 2:       SUBROUTINE GRNUM(X, Y, FLT, FIELD, DEC, HEIGHT. ANG)
 3:
 4: * graphics text output of packed string to HP 2647A
 5:
 6:       INTEGER N, FIELD, DEC
 7:       REAL X, Y, FLT, HEIGHT. ANG
 8:       REAL PMODE(25,2)
 9:       COMMON /CPMTAB/ PMODE
10:       INTEGER OO
11:       INTEGER STR(20). PS. L, ORIENT. SIZE
12:       INTEGER FTOC
13:
14: *  move pen to <x,y>
15:
16:       OO = INT(PMODE(17,2))
17:       CALL MOVEPN(X,Y,00.0)
18:
19: *  set up size and orientation
20:
21:       SIZE = INT(HEIGHT/0.25) + 1
22:       IF (ANG .GE. 0.0)
23:       .  ORIENT = MOD(NINT(ANG/90.0),4) + 1
24:       ELSE
25:       .  ORIENT = MOD(NINT((360.0+ANG)/90.0),4) + 1
26:       END IF
27:
28:       STR(1) = 109
29:       STR(2) = 48 + SIZE
30:       STR(3) = 109
31:       STR(4) = 48 + ORIENT
32:       STR(5) = 110
33:       STR(6) = 49
34:       STR(7) = 113
35:       STR(8) = -2
36:       PS = 7
37:       CALL OUTHPP(STR,PS)
38:
39: *  convert floating point number to string and set up for output
40:
41:       STR(1) = 100
42:       STR(2) = 83
43:       PS = 3
44:       L = FTOC(FLT,STR(PS),FIELD,DEC)
45:       PS = PS + L
46:       CALL OUTHPT(STR.PS)
47:
48: *  turn off graphics text mode
49:
50:       STR(1) = 100
51:       STR(2) = 116
52:       STR(3) = -2
53:       PS = 2
54:       CALL OUTHPP(STR.PS)
55:
56:       RETURN
57:       END
```

```
 1: *-h- GRNUM 345  3 AUG 82  11:01:43
 2:       SUBROUTINE GRNUM(X, Y, FLT, FIELD, DEC, HEIGHT, ANG)
 3:
 4: * plot floating point number from <x,y>
 5: *  interface to VPLOT07
 6:
 7:       INTEGER FIELD, DEC
 8:       REAL X, Y, FLT, HEIGHT. ANG
 9:       REAL PMODE(25,2)
10:       COMMON /CPMTAB/ PMODE
11:
12:       CALL NUMBER(X,Y,HEIGHT,FLT,ANG,DEC)
13:
14:       PMODE(6.1) = X
15:       PMODE(6.2) = Y
16:
17:       RETURN
18:       END
```

```
 1: *-h- GRSTR 1394  2 MAR 83  12:49:18
 2:       SUBROUTINE GRSTR(X, Y, BUF, N, HEIGHT. ANG)
 3:
 4: * graphics text output of character string to HP 2647A
 5: *  if n > 0 string is paked and n is the number of characters
 6: *  if n <= 0 string is unpacked and terminated by EOS
 7:
 8:       INTEGER BUF(27), N
 9:       REAL X, Y, HEIGHT. ANG
10:       REAL PMODE(25,2)
11:       COMMON /CPMTAB/ PMODE
12:       INTEGER OO. ORIENT. PS. PU. SIZE. STR(81), UBUF(81)
13:       INTEGER LENGTH
14:
15: *  move pen to <x,y>
16:
17:       OO = INT(PMODE(17.2))
18:       CALL MOVEPN(X,Y,OO.0)
19:
20: *  set up size and orientation
21:
22:       SIZE = INT(HEIGHT/0.25) + 1
23:       IF (ANG .GE. 0.0)
24:       .  ORIENT = MOD(NINT(ANG/90.0),4) + 1
25:       ELSE
26:       .  ORIENT = MOD(NINT((360.0+ANG)/90.0),4) + 1
27:       END IF
28:
29:       STR(1) = 109
30:       STR(2) = 48 + SIZE
31:       STR(3) = 109
32:       STR(4) = 48 + ORIENT
33:       STR(5) = 110
34:       STR(6) = 49
35:       STR(7) = 113
36:       STR(8) = -2
37:       PS = 7
38:       CALL OUTHPP(STR.PS)
39:
40: *  set up string for output
41:
42:       STR(1) = 100
43:       STR(2) = 83
44:       PS = 2
45:       IF (N .GT. 0)
46:       .  CALL UNPACK(BUF.UBUF.N/3+1)
47:       .  FOR PU = 1.N
48:       .  .  PS = PS + 1
49:       .  .  STR(PS) = UBUF(PU)
50:       .  END FOR
51:       .  STR(PS+1) = -2
52:       ELSE
53:       .  CALL SCOPY(BUF,1.STR.PS+1)
54:       .  PS = PS + LENGTH(BUF) + 1
55:       END IF
56:
57:       CALL OUTHPT(STR.PS)
58:
```

```
59: *  turn off graphics text mode
60:
61:       STR(1) = 100
62:       STR(2) = 116
63:       STR(3) = -2
64:       PS = 2
65:       CALL OUTHPP(STR.PS)
66:
67:       RETURN
68:       END
```

```
 1: *-h- GRSTR 604  2 MAR 83  12:57:15
 2:
 3:       SUBROUTINE GRSTR(X, Y, BUF, N, HEIGHT, ANG)
 4:
 5: * plot string from <x,y>
 6: *  interface to VPLOT07
 7: *  if n > 0 string is packed and n is the number of characters
 8: *  if n <= 0 string is terminated by EOS
 9:
10:       INTEGER BUF(1), N
11:       REAL X, Y, HEIGHT, ANG
12:       REAL PMODE(25,2)
13:       COMMON /CPMTAB/ PMODE
14:       INTEGER LEN, PBUF(27)
15:       INTEGER LENGTH
16:
17:       IF (N .GT. 0)
18:       .  CALL SYMBOL(X,Y,HEIGHT,BUF,ANG,N)
19:       ELSE
20:       .  LEN = LENGTH(BUF)
21:       .  CALL PACK(BUF,PBUF)
22:       .  CALL SYMBOL(X,Y,HEIGHT,PBUF,ANG,LEN)
23:       END IF
24:
25:       PMODE(6,1) = X
26:       PMODE(6,2) = Y
27:
28:       RETURN
29:       END
```

```
 1: *-h- GRSYMB 1044 21 FEB 83  15:25:23
 2:       SUBROUTINE GRSYMB(X, Y, ASC, HEIGHT, ANG)
 3:
 4: * write ASCII symbol at <x,y>
 5:
 6:       REAL X, Y, HEIGHT, ANG
 7:       INTEGER ASC
 8:       REAL PMODE(25,2)
 9:       COMMON /CPMTAB/ PMODE
10:       INTEGER OO
11:       INTEGER STR(15), PS, ORIENT, SIZE
12:
13: *  move pen to <x,y>
14:
15:       OO = INT(PMODE(17,2))
16:       CALL MOVEPN(X,Y,OO,0)
17:
18: *  set text parameters
19:
20:       SIZE = INT(HEIGHT/0.25) + 1
21:       IF (ANG .GE. 0.0)
22:       .  ORIENT = MOD(NINT(ANG/90.0),4) + 1
23:       ELSE
24:       .  ORIENT = MOD(NINT((360.0+ANG)/90.0),4) + 1
25:       END IF
26:
27:       STR(1) = 109
28:       STR(2) = 48 + SIZE
29:       STR(3) = 109
30:       STR(4) = 48 + ORIENT
31:       STR(5) = 110
32:       STR(6) = 53
33:       STR(7) = 113
34:       STR(8) = -2
35:       PS = 7
36:
37:       CALL OUTHPP(STR.PS)
```

```
38:
39: *  draw character
40:
41:       STR(1) = 100
42:       STR(2) = 83
43:       STR(3) = ASC
44:       STR(4) = -2
45:       PS = 3
46:
47:       CALL OUTHPT(STR.PS)
48:
49: *  turn off graphics text (I don't know why this is necessary AES 1/81)
50:
51:       STR(1) = 100
52:       STR(2) = 116
53:       STR(3) = -2
54:       PS = 2
55:
56:       CALL OUTHPP(STR.PS)
57:
58:       RETURN
59:       END
```

```
 1: *-h- GRSYMB 311 21 FEB 83  14:13:52
 2:       SUBROUTINE GRSYMB(X, Y, ASC, HEIGHT, ANG)
 3:
 4: * plot single symbol at <x,y>
 5: *  interface to VPLOT07
 6:
 7:       INTEGER ASC
 8:       REAL X, Y, HEIGHT, ANG
 9:       REAL PMODE(25,2)
10:       COMMON /CPMTAB/ PMODE
11:
12:       CALL SYMBOL(X,Y,HEIGHT,ASC,ANG,-1)
13:
14:       PMODE(6.1) = X
15:       PMODE(6.2) = Y
16:
17:       RETURN
18:       END
```

```
 1: *-h- INHP 901 22 JUN 82  21:50:03
 2:       SUBROUTINE INHP(BUF, PB)
 3:
 4: * receive buffer from HP 2647A terminal, reformat for use
 5: *  by plot package
 6:
 7:       INTEGER BUF(1), PB
 8:
 9: *  special read
10:
11: :ASSE
12:         TMA*  PB
13:         TAM   COUNT          transfer word count to COUNT
14:         TMA   BUF
15:         TAM   STADD          transfer start address of BUF to STADD
16:         TLO   PARLST         transfer long operand at label to K
17:         BLU   $IOW           call I/O routine, wait for completion
18: :END
19:
20:       WRITE (3.FMT='(/)')
21:
22: *  replace '.' in buffer by EOS
23:
24:       BUF(7) = -2
25:       BUF(14) = -2
26:       BUF(PB+1) = -2
27:
28:       RETURN
29: :ASSE
30:         PORG  *              turn on data program counter
31: PARLST  DATA  '0305          special read from lfn 3
32: COUNT   ***                  save space for word count
33: STADD   ***                  save space for buffer address
34:         RORG  *              turn on instruction program counter
35: :END
36:       END
```

```
1: *-h- INIPLT 3998 31 MAY 83  22:58:54
2:       SUBROUTINE INIPLT
3:
4: * prepare HP 2647A graphics terminal for plotting
5:
6:       INTEGER PLTBUF(81), PB
7:       COMMON /CHPOUT/ PLTBUF, PB
8:       REAL PMODE(25,2)
9:       COMMON /CPMTAB/ PMODE
10:      INTEGER PS, STR(5)
11:
12: ************************************************************************
13: *                                                                      *
14: *                      mode table for graphics                         *
15: *                                                                      *
16: *                    1                          2                     *
17: *          ---------------------------------------------------         *
18: *     1   |     x scale factor        |     y scale factor        |      *
19: *     2   |     x coordinate base     |     y coordinate base     |      *
20: *     3   |     x max dimension       |     y max dimension       |      *
21: *     4   |     x window min          |     y window min          |      *
22: *     5   |     x window max          |     y window max          |      *
23: *     6   |     x current pen pos     |     y current pen pos     |      *
24: *     7   |     x current origin pos  |     y current origin pos  |      *
25: *     8   |     x cursor pos          |     y cursor pos          |      *
26: *     9   |     x zoom pos            |     y zoom pos            |      *
27: *    10   |     x previous pen pos    |     y previous pen pos    |      *
28: *    11   |       ---                 |       ---                 |      *
29: *    12   |       ---                 |       ---                 |      *
30: *    13   |       ---                 |       ---                 |      *
31: *    14   |       ---                 |       ---                 |      *
32: *    15   |     x,y scale factor      |     plot incs per inch    |      *
33: *    16   |     zoom size             |     graphics cursor stat  |      *
34: *    17   |     pen status            |     origin status         |      *
35: *    18   |     drawing mode          |     text mode             |      *
36: *    19   |     window status         |     string terminator     |      *
37: *    20   |       ---                 |       ---                 |      *
38: *    21   |       ***                 |       ***                 |      *
39: *    22   |       ***                 |       ***                 |      *
40: *    23   |       ***                 |       ***                 |      *
41: *    24   |       ***                 |       ***                 |      *
42: *    25   |       ***                 |       ***                 |      *
43: *          ---------------------------------------------------         *
44: *                                                                      *
45: *          --- reserved for future features                            *
46: *          *** reserved for user use                                   *
47: *                                                                      *
48: ************************************************************************
49:
50: *  HP 2647A defaults
51:
52:       PMODE(1,1) = 1.0
53:       PMODE(1,2) = 1.0
54:       PMODE(2,1) = 0.0
55:       PMODE(2,2) = 0.0
56:       PMODE(3,1) = 10.0
57:       PMODE(3,2) = 5.0
58:       PMODE(4,1) = 0.0
59:       PMODE(4,2) = 0.0
60:       PMODE(5,1) = 0.0
61:       PMODE(5,2) = 0.0
62:       PMODE(6,1) = 0.0
63:       PMODE(6,2) = 0.0
64:       PMODE(7,1) = 0.0
65:       PMODE(7,2) = 0.0
66:       PMODE(8,1) = 0.0
67:       PMODE(8,2) = 0.0
68:       PMODE(9,1) = 0.0
69:       PMODE(9,2) = 0.0
70:       PMODE(15,1) = 1.0
71:       PMODE(15,2) = 72.0
72:       PMODE(16,1) = 0.0
73:       PMODE(16,2) = 0.0
74:       PMODE(17,1) = 0.0
75:       PMODE(17,2) = 1.0
76:       PMODE(18,1) = 4.0
77:       PMODE(18,2) = 0.0
78:       PMODE(19,1) = 0.0
79:       PMODE(19,2) = 46.0
80:
```

```
 81: *  initialize output buffer
 82:
 83:       PLTBUF(1) = 27
 84:       PLTBUF(2) = 42
 85:       PLTBUF(3) = 109
 86:       PB = 3
 87:
 88: *  clear screen and turn on graphics mode
 89:
 90:       STR(1) = 109
 91:       STR(2) = 114
 92:       STR(3) = -2
 93:       PS = 2
 94:       CALL OUTHPP(STR,PS)
 95:
 96:       STR(1) = 100
 97:       STR(2) = 97
 98:       STR(3) = 102
 99:       STR(4) = -2
100:       PS = 3
101:       CALL OUTHPP(STR,PS)
102:
103:       CALL FLUSH
104:
105:       RETURN
106:       END
```

```
  1: *-h- INIPLT 3348 21 FEB 83  15:34:06
  2:       SUBROUTINE INIPLT
  3:
  4: * prepare VERSATEC 1200 plotter for plotting
  5:
  6:       REAL PMODE(25,2)
  7:       COMMON /CPMTAB/ PMODE
  8:
  9: ************************************************************************
 10: *                                                                      *
 11: *                      mode table for graphics                         *
 12: *                                                                      *
 13: *                     1                          2                     *
 14: *       ---------------------------------------------------------      *
 15: *     1 |     x scale factor        |     y scale factor        |      *
 16: *     2 |     x coordinate base     |     y coordinate base     |      *
 17: *     3 |     x max dimension       |     y max dimension       |      *
 18: *     4 |     x window min          |     y window min          |      *
 19: *     5 |     x window max          |     y window max          |      *
 20: *     6 |     x current pen pos     |     y current pen pos     |      *
 21: *     7 |     x current origin pos  |     y current origin pos  |      *
 22: *     8 |     x cursor pos          |     y cursor pos          |      *
 23: *     9 |     x zoom pos            |     y zoom pos            |      *
 24: *    10 |     x previous pen pos    |     y previous pen pos    |      *
 25: *    11 |       ---                 |       ---                 |      *
 26: *    12 |       ---                 |       ---                 |      *
 27: *    13 |       ---                 |       ---                 |      *
 28: *    14 |       ---                 |       ---                 |      *
 29: *    15 |     x,y scale factor      |     plot incs per inch    |      *
 30: *    16 |     zoom size             |     graphics cursor stat  |      *
 31: *    17 |     pen status            |     origin status         |      *
 32: *    18 |     drawing mode          |     text mode             |      *
 33: *    19 |     window status         |     string terminator     |      *
 34: *    20 |       ---                 |       ---                 |      *
 35: *    21 |       ***                 |       ***                 |      *
 36: *    22 |       ***                 |       ***                 |      *
 37: *    23 |       ***                 |       ***                 |      *
 38: *    24 |       ***                 |       ***                 |      *
 39: *    25 |       ***                 |       ***                 |      *
 40: *       ---------------------------------------------------------      *
 41: *                                                                      *
 42: *        --- reserved for future features                              *
 43: *        *** reserved for user use                                     *
 44: *                                                                      *
 45: ************************************************************************
 46:
 47: *  VERSATEC 1200 defaults
 48:
 49:       PMODE(1,1) = 1.0
 50:       PMODE(1,2) = 1.0
 51:       PMODE(2,1) = 0.0
 52:       PMODE(2,2) = 0.0
```

```
53:       PMODE(3,1) = 20.0
54:       PMODE(3,2) = 10.9
55:       PMODE(4,1) = 0.0
56:       PMODE(4,2) = 0.0
57:       PMODE(5,1) = 0.0
58:       PMODE(5,2) = 0.0
59:       PMODE(6,1) = 0.0
60:       PMODE(6,2) = 0.0
61:       PMODE(7,1) = 0.0
62:       PMODE(7,2) = 0.0
63:       PMODE(15,1) = 1.0
64:       PMODE(15,2) = 200.0
65:       PMODE(17,1) = 3.0
66:       PMODE(19,1) = 0.0
67:       PMODE(19,2) = 46.0
68:
69:       CALL PLOTS(0,0,0)
70:
71:       RETURN
72:       END
```

```
 1: *-h- LINTYP 692 16 FEB 82  21:08:14
 2:       SUBROUTINE LINTYP(MASK. SCALE)
 3:
 4: * define dot pattern for drawing vectors or filling areas
 5:
 6:       INTEGER MASK, SCALE
 7:       INTEGER STR(10), PS, L, M
 8:       INTEGER ITOC
 9:
10:       STR(1) = 109
11:       PS = 2
12:
13:       IF (MASK .GT. 0)
14:
15: *  user defined mask
16:
17:       .  STR(PS) = 32
18:       .  PS = PS + 1
19:       .  L = ITOC(MASK.STR(PS),4)
20:       .  PS = PS + L
21:       .  STR(PS) = 32
22:       .  PS = PS + 1
23:       .  L = ITOC(SCALE,STR(PS),3)
24:       .  PS = PS + L
25:       .  STR(PS) = 99
26:       .  PS = PS + 1
27:       .  M = -2
28:       ELSE
29:       .  M = MASK
30:       END IF
31:
32:       STR(PS) = 32
33:       PS = PS + 1
34:       L = ITOC(-M,STR(PS),3)
35:       PS = PS + L
36:       STR(PS) = 98
37:       STR(PS+1) = -2
38:
39:       CALL OUTHPP(STR,PS)
40:
41:       RETURN
42:       END
```

```
 1: *-h- MOVE00 646  4 DEC 82  17:27:20
 2:       SUBROUTINE MOVE00(X, Y)
 3:
 4: * move relocatable origin to <x,y>
 5:
 6:       REAL X, Y
 7:       INTEGER STR(15), PS, L, CX, CY
 8:       REAL PMODE(25,2)
 9:       COMMON /CPMTAB/ PMODE
10:
11:       STR(1) = 109
12:       PS = 2
13:
```

```
14: *  scale coordinates
15:
16:       CX = INT(X*PMODE(1,1)*PMODE(15,1)*PMODE(15,2))
17:       CY = INT(Y*PMODE(1,2)*PMODE(15,1)*PMODE(15,2))
18:
19:       L = ITOC(CX,STR(PS),5)
20:       PS = PS + L
21:       STR(PS) = 32
22:       PS = PS + 1
23:       L = ITOC(CY,STR(PS),5)
24:       PS = PS + L
25:       STR(PS) = 106
26:       STR(PS+1) = -2
27:
28:       CALL OUTHPP(STR,PS)
29:
30: *  update mode table
31:
32:       PMODE(7,1) = X
33:       PMODE(7,2) = Y
34:       PMODE(17,2) = 3.0
35:
36:       RETURN
37:       END
```

```
 1: *-h- MOVE00 276  3 AUG 82  11:01:49
 2:       SUBROUTINE MOVE00(X, Y)
 3:
 4: * move relocatable origin to <x,y>
 5: *  interface to VPLOT07
 6:
 7:       REAL X, Y
 8:       REAL PMODE(25,2)
 9:       COMMON /CPMTAB/ PMODE
10:
11:       CALL PLOT(X,Y,-3)
12:
13:       PMODE(7,1) = PMODE(7,1) + X
14:       PMODE(7,2) = PMODE(7,2) + Y
15:
16:       RETURN
17:       END
```

```
 1: *-h- MOVEPI 1673 28 MAY 83  12:34:48
 2:       SUBROUTINE MOVEPI(X, Y, OO, PEN)
 3:
 4: * move pen to <x,y> in absolute screen coordinates
 5:
 6:       INTEGER X, Y, OO, PEN
 7:       REAL PMODE(25,2)
 8:       COMMON /CPMTAB/ PMODE
 9:       INTEGER STR(15), PS, L, BYTE(3), I, M, P
10:
11:       STR(1) = 112
12:       PS = 2
13:
14: *  lift or lower pen as necessary
15:
16:       IF (PEN .EQ. 0)
17:       .  STR(PS) = 97
18:       .  PS = PS + 1
19:       ELSE IF (PEN .EQ. 1)
20:       .  STR(PS) = 98
21:       .  PS = PS + 1
22:       END IF
23:
24: *  choose from absolute, incremental, or relative coordinates
25:
26:       IF (OO .EQ. 1)
27:       .  STR(PS) = 105
28:       ELSE IF (OO .EQ. 2)
29:       .  STR(PS) = 107
30:       ELSE IF (OO .EQ. 3)
31:       .  STR(PS) = 108
32:       ELSE
33:       .  CALL ERROR(" illegal coordinate type")
34:       END IF
35:       PS = PS + 1
```

```
36:
37: *  load coordinates into successive bytes.  2 bytes are required for
38: *   absolute coordinates, 3 for incremental and relative coordinates
39: *   (to accomodate negative numbers).  each byte stores 5 bits of
40: *   coordinate.  negative numbers are in two's complement
41: *   (compatible with HARRIS)
42:
43:       M = 1
44:       I = 0
45:       IF (OO .GT. 1)
46:       .  I = I + 1
47:       .  BYTE(I) = (X .SHIFT. -10) .AND. '37
48:       END IF
49:       I = I + 1
50:       BYTE(I) = (X .SHIFT. -5) .AND. '37
51:       I = I + 1
52:       BYTE(I) = X .AND. '37
53:       M = I
54:       FOR I = 1,M
55:       .  STR(PS) = BYTE(I) .OR. '40
56:       .  PS = PS + 1
57:       END FOR
58:
59:       I = 0
60:       IF (OO .GT. 1)
61:       .  I = I + 1
62:       .  BYTE(I) = (Y .SHIFT. -10) .AND. '37
63:       END IF
64:       I = I + 1
65:       BYTE(I) = (Y .SHIFT. -5) .AND. '37
66:       I = I + 1
67:       BYTE(I) = Y .AND. '37
68:       FOR I = 1,M
69:       .  STR(PS) = BYTE(I) .OR. '40
70:       .  PS = PS + 1
71:       END FOR
72:       STR(PS) = -2
73:
74:       CALL OUTHPP(STR,PS-1)
75:
76:       RETURN
77:       END
```

```
 1: *-h- MOVEPN 632 27 MAY 83  12:59:17
 2:       SUBROUTINE MOVEPN(X, Y, OO, PEN)
 3:
 4: * move pen to <x,y>
 5:
 6:       REAL X, Y
 7:       INTEGER OO, PEN
 8:       REAL PMODE(25,2)
 9:       COMMON /CPMTAB/ PMODE
10:       INTEGER CX, CY. P
11:
12: *  update current pen position
13:
14:       PMODE(6,1) = X
15:       PMODE(6,2) = Y
16:
17: *  if clip mode is on calculate new drawing coordinates
18:
19:       IF (PMODE(19,1) .EQ. 1)
20:       .  CALL PCLIP(X,Y,OO,PEN)
21:       ELSE
22:       .  CX = NINT(X*PMODE(1,1)*PMODE(15,1)*PMODE(15,2))
23:       .  CY = NINT(Y*PMODE(1,2)*PMODE(15,1)*PMODE(15,2))
24:       .  CALL MOVEPI(CX,CY,OO,PEN)
25:       END IF
26:
27: *  update previous pen position
28:
29:       PMODE(10,1) = X
30:       PMODE(10,2) = Y
31:
32:       RETURN
33:       END
```

```
 1: *-h- MOVEPN 510 28 MAY 83  11:15:44
 2:       SUBROUTINE MOVEPN(X, Y, COOR. PEN)
 3:
 4: * move pen to <x,y>
 5: *  interface to VPLOT07
 6:
 7:       INTEGER COOR, PEN
 8:       REAL PMODE(25,2)
 9:       COMMON /CPMTAB/ PMODE
10:
11:       PMODE(6,1) = X
12:       PMODE(6,2) = Y
13:       PMODE(17,1) = FLOAT(PEN)
14:
15: *  recalculate coordinates if clip mode is on
16:
17:       IF (PMODE(19,1) .EQ. 1.0)
18:       .  CALL PCLIP(X,Y,COOR.PEN)
19:       ELSE
20:       .  CALL PLOT(X,Y,3-PEN)
21:       END IF
22:
23:
24: *  store <x,y> coordinates for next call
25:
26:       PMODE(10,1) = X
27:       PMODE(10,2) = Y
28:
29:       RETURN
30:       END
```

```
 1: *-h- NEWPEN 110  8 JAN 82  12:24:50
 2:       SUBROUTINE NEWPEN(PEN)
 3:
 4: * dummy routine for HP 2647A library
 5:
 6:       INTEGER PEN
 7:
 8:       RETURN
 9:       END
```

```
 1: *-h- OUTHPP 635  1 AUG 82  15:02:32
 2:       SUBROUTINE OUTHPP(STR, PS)
 3:
 4: * output routine used by HP 2647A plotting routines
 5:
 6:       INTEGER STR(1), PS
 7:       INTEGER PLTBUF(81), PB
 8:       COMMON /CHPOUT/ PLTBUF, PB
 9:
10: *  if device control mode is different, send buffer to terminal
11:
12:       IF (STR(1) .NE. PLTBUF(3))
13:       .  IF (PB .GT. 3) CALL FLUSH
14:       .  PLTBUF(3) = STR(1)
15:       END IF
16:
17: *  if insufficient room in buffer, send to terminal
18:
19:       IF (PB+PS+2 .GE. 81)
20:       .  IF (PB .GT. 3) CALL FLUSH
21:       .  PLTBUF(3) = STR(1)
22:       END IF
23:
24: *  copy command string to output buffer
25:
26:       PB = PB + 1
27:       CALL SCOPY(STR,2,PLTBUF,PB)
28:       PB = PB + PS - 2
29:
30:       RETURN
31:       END
```

```
 1: *-h- OUTHPT 522  1 AUG 82  15:02:33
 2:       SUBROUTINE OUTHPT(STR.PS)
 3:
 4: * output routine for graphics text for HP 2647A terminal
 5:
 6:       INTEGER STR(1), PS
 7:       INTEGER PLTBUF(81), PB
 8:       COMMON /CHPOUT/ PLTBUF, PB
 9:       INTEGER I, PBUF(27)
10:
11:       IF (STR(1) .NE. PLTBUF(3))
12:       .  CALL FLUSH
13:       END IF
14:
15:       CALL SCOPY(STR,1,PLTBUF,PB)
16:       PB = PB + PS
17:       PLTBUF(PB) = 13
18:       PLTBUF(PB+1) = -2
19:       FOR I = 1,27
20:       .  PBUF(I) = 0
21:       END FOR
22:       CALL PACK(PLTBUF,PBUF)
23:       WRITE (3,FMT='(1X,27A3)') PBUF
24:       PB = 3
25:
26:       RETURN
27:       END
```

```
 1: *-h- PAXIS 2751  4 NOV 82  12:04:06
 2:       SUBROUTINE PAXIS(XMIN,XMAX,YMIN,YMAX,SMIN,SMAX,FIELD,DEC,HEIGHT,
 3:      &AOPT,ANG,SOPT,N,DF)
 4:
 5: ************************************************************************
 6: *                                                                      *
 7: *  plot one axis, annotate optionally, recalculate scale optionally    *
 8: *                                                                      *
 9: *     XMIN, YMIN:   start coordinates of axis                          *
10: *     XMAX, YMAX:   end coordinate of axis                             *
11: *     SMIN      :   lower bound of scale range                         *
12: *     SMAX      :   upper bound of scale range                         *
13: *     FIELD, DEC:   width and decimal of annotation                    *
14: *     HEIGHT    :   height of digits                                   *
15: *     AOPT      :   annotation option (see subroutine TICK)            *
16: *     ANG       :   angle of tick mark                                 *
17: *     SOPT      :   scaling option                                     *
18: *                       0- no rescaling                                *
19: *                       1- calculate new optimal scale bounds for      *
20: *                           approximately n divisions                  *
21: *                       2- calculate new optimal scale bounds for      *
22: *                           exactly n divisions                        *
23: *     N         :   number of tick marks                               *
24: *     DF        :   decimation factor for annotation                   *
25: *                                                                      *
26: ************************************************************************
27:
28:       INTEGER FIELD, DEC, AOPT, SOPT, N, DF
29:       REAL XMIN, XMAX, YMIN, YMAX, SMIN, SMAX, ANG
30:       INTEGER I, M, OO, PENDOWN, PENUP
31:       REAL XX, YY
32:       DATA PENUP /0/, PENDOWN /1/
33:
34:       OO = INT(RDMTAB(17,2))
35:
36:       CALL NEWPEN(5)
37:       CALL MOVEPN(XMIN,YMIN,OO,PENUP)
38:       CALL MOVEPN(XMAX,YMAX,OO,PENDOWN)
39:       CALL NEWPEN(1)
40:
41:       IF (N .GT. 0)
42:       .  IF (SOPT .EQ. 1)
43:       .  .  N = N / DF
44:       .  .  CALL SCALE1(SMIN,SMAX,N,SMIN,SMAX,DIST)
45:       .  .  N = NINT((SMAX-SMIN)/DIST) * DF
46:       .  ELSE IF (SOPT .EQ. 2)
47:       .  .  N = N / DF
48:       .  .  CALL SCALE2(SMIN,SMAX,N,SMIN,SMAX,DIST)
49:       .  .  N = NINT((SMAX-SMIN)/DIST) * DF
50:       .  END IF
51:       .
```

```
52:       . FOR I = 0.N
53:       . . XX = XMIN + (XMAX - XMIN) * FLOAT(I) / FLOAT(N)
54:       . . YY = YMIN + (YMAX - YMIN) * FLOAT(I) / FLOAT(N)
55:       . . IF (MOD(I,DF) .EQ. 0)
56:       . . . S = SMIN + (SMAX - SMIN) * FLOAT(I) / FLOAT(N)
57:       . . . CALL TIC(XX,YY,S,FIELD,DEC,HEIGHT,AOPT,ANG)
58:       . . ELSE
59:       . . . CALL TIC(XX,YY,0.0,0.0,0.0,0.6*HEIGHT,0.ANG)
60:       . . END IF
61:       . END FOR
62:       END IF
63:
64:       CALL MOVEPN(XMIN,YMIN,00,PENUP)
65:
66:       RETURN
67:       END
```

```
 1: *-h- PLOTS 178 12 JAN 82  15:21:34
 2:       SUBROUTINE PLOTS(A1, A2, A3)
 3:
 4: * initialize plotting
 5: *  emulate Versaplot-07 calls on HP 2647A terminal
 6:
 7:       INTEGER A1, A2, A3
 8:
 9:       CALL INIPLT
10:
11:       RETURN
12:       END
```

```
 1: *-h- PMAT 1306 24 JAN 83  11:19:20
 2:       SUBROUTINE PMAT(A, LDA, M, INCM, N, INCN, XLEN, YLEN, SLEN,
 3:      &  HEIGHT, MDF, NDF)
 4:
 5: *  plot coefficients of matrix in rectangle
 6:
 7:       INTEGER LDA, M, INCM, N, INCN, MDF, NDF
 8:       REAL A(LDA,1), XLEN, YLEN, SLEN, HEIGHT
 9:       INTEGER I, J, IM, IN, 00, PEN, PENDOWN, PENUP
10:       REAL XDEL, YDEL, YMAX, YMIN, YSCALE
11:       REAL SVMAX, SVMIN
12:       DATA PENDOWN /1/, PENUP /0/
13:
14:       XDEL = XLEN / FLOAT(M)
15:       YDEL = YLEN / FLOAT(N)
16:       00 = INT(RDMTAB(17.2))
17:
18:       IF (MDF .GT. 0 .AND. NDF .GT. 0)
19:       . CALL PAXIS(0.0,XLEN,0.0,0.0,0.0,0.0,FLOAT(M),4.,-1.0,15,3.,-90.0,0.
20:      &.    M,MDF)
21:       . CALL PAXIS(0.0,0.0,0.0,0.0,YLEN,0.0,FLOAT(N),4.,-1.0,15,2.,180.0,0.
22:      &.    N,NDF)
23:       . CALL PAXIS(XLEN,XLEN,0.0,0.YLEN,0.0,0.FLOAT(N),0.0,0.15,0.0,0.0,0.
24:      &.    N,NDF)
25:       END IF
26:
27:       YMAX = -1.0E37
28:       YMIN = 1.0E37
29:       IN = 1
30:       FOR I = 1.N
31:       . YMAX = AMAX1(YMAX,SVMAX(A(1,IN),INCM,JUNK,M))
32:       . YMIN = AMIN1(YMIN,SVMIN(A(1,IN),INCM,JUNK,M))
33:       . IN = IN + INCN
34:       END FOR
35:       YSCALE = SLEN / (YMAX - YMIN)
36:
37:       IN = 1
38:       FOR J = 1.N
39:       . PEN = PENUP
40:       . IM = 1
41:       . FOR I = 1.M
42:       . . XX = I * XDEL
43:       . . YY = A(IM,IN) * YSCALE + J * YDEL
44:       . . CALL MOVEPN(XX,YY,00,PEN)
45:       . . PEN = PENDOWN
46:       . . IM = IM + INCM
47:       . END FOR
48:       . IN = IN + INCN
```

```
49:       END FOR
50:
51:       RETURN
52:       END


 2:       SUBROUTINE PVEC1(Y, N. XO. YO. YREF. YSCALE. XINC. ANG)
 3:
 4: * plot a vector of data against an incremental ordinate
 5:
 6:       INTEGER N
 7:       REAL Y(1). XO. YO. YREF. YSCALE. XINC. ANG
 8:       INTEGER I. OO. PEN
 9:       REAL CA. DR. SA. XX. YY
10:       REAL RDMTAB
11:       DATA DR /0.01745329/
12:
13:       OO = INT(RDMTAB(17.2))
14:
15:       CA = COS(DR*ANG)
16:       SA = SIN(DR*ANG)
17:
18:       PEN = 0
19:       FOR I = 1.N
20:       .  XX = XINC * FLOAT(I-1)
21:       .  YY = (Y(I) - YREF) * YSCALE
22:       .  CALL VROT(XX,1.YY,1.CA,SA,1)
23:       .  CALL MOVEPN(XO+XX,YO+YY,OO.PEN)
24:       .  PEN = 1
25:       END FOR
26:
27:       RETURN
28:       END


 1: *-h- PVEC2 493 20 DEC 82  11:25:43
 2:       SUBROUTINE PVEC2(X. Y. N. XO. YO. XREF. XSCALE. YREF. YSCALE)
 3:
 4: * plot array of points <x . y >
 5: *                        i   i
 6:
 7:       INTEGER N
 8:       REAL X(1). Y(1). XO. YO. XREF. XSCALE. YREF. YSCALE
 9:       INTEGER I. OO. PEN
10:       REAL XX. YY
11:       REAL RDMTAB
12:
13:       OO = INT(RDMTAB(17.2))
14:
15:       PEN = 0
16:       FOR I = 1.N
17:       .  XX = XO + (X(I) - XREF) * XSCALE
18:       .  YY = YO + (Y(I) - YREF) * YSCALE
19:       .  CALL MOVEPN(XX,YY.OO.PEN)
20:       .  PEN = 1
21:       END FOR
22:
23:       RETURN
24:       END


 1: *-h- PNTCNF 1276 23 SEP 82   9:58:56
 2:       SUBROUTINE PNTCNF(X. Y. XERR. YERR. SIZE)
 3:
 4: * plot datum point plus error bars
 5:
 6:       REAL X. Y. XERR. YERR. SIZE
 7:       INTEGER OO
 8:       REAL SIZED2. XX, YY
 9:
10:       OO = INT(RDMTAB(17.2))  !  get origin type
11:       SIZED2 = SIZE / 2.0
12:       CALL CIRCLE(X,Y,SIZED2)
13:
```

```
14: *  x error bars
15:
16:       IF (XERR .GT. 0.0)
17:       .  XX = X + SIZED2
18:       .  CALL MOVEPN(XX,Y,00.0)
19:       .  XX = X + XERR
20:       .  CALL MOVEPN(XX,Y,00.1)
21:       .  YY = Y - SIZED2
22:       .  CALL MOVEPN(XX,YY,00.0)
23:       .  YY = Y + SIZED2
24:       .  CALL MOVEPN(XX,YY,00.1)
25:       .  XX = X - SIZED2
26:       .  CALL MOVEPN(XX,Y,00.0)
27:       .  XX = X - XERR
28:       .  CALL MOVEPN(XX,Y,00.1)
29:       .  YY = Y - SIZED2
30:       .  CALL MOVEPN(XX,YY,00.0)
31:       .  YY = Y + SIZED2
32:       .  CALL MOVEPN(XX,YY,00.1)
33:       .  CALL MOVEPN(X,Y,00.0)
34:       END IF
35:
36: *  y error bars
37:
38:       IF (YERR .GT. 0.0)
39:       .  YY = Y + SIZED2
40:       .  CALL MOVEPN(X,YY,00.0)
41:       .  YY = Y + YERR
42:       .  CALL MOVEPN(X,YY,00.1)
43:       .  XX = X - SIZED2
44:       .  CALL MOVEPN(XX,YY,00.0)
45:       .  XX = X + SIZED2
46:       .  CALL MOVEPN(XX,YY,00.1)
47:       .  YY = Y - SIZED2
48:       .  CALL MOVEPN(X,YY,00.0)
49:       .  YY = Y - YERR
50:       .  CALL MOVEPN(X,YY,00.1)
51:       .  XX = X - SIZED2
52:       .  CALL MOVEPN(XX,YY,00.0)
53:       .  XX = X + SIZED2
54:       .  CALL MOVEPN(XX,YY,00.1)
55:       .  CALL MOVEPN(X,Y,00.0)
56:       END IF
57:
58:       RETURN
59:       END
```

```
 1: *-h- RBLIN 393 24 OCT 82  22:03:22
 2:       SUBROUTINE RBLIN(LIN)
 3:
 4: * turn rubber band line on (lin=1) or off (lin=0)
 5:
 6:       INTEGER LIN
 7:       REAL PMODE(25,2)
 8:       COMMON /CPMTAB/ PMODE
 9:       INTEGER STR(3), PS
10:
11:       STR(1) = 100
12:       IF (LIN .EQ. 1)
13:       .  STR(2) = 109
14:       ELSE IF (LIN .EQ. 0)
15:       .  STR(2) = 110
16:       END IF
17:       STR(3) = -2
18:       PS = 2
19:
20:       CALL OUTHPP(STR,PS)
21:       CALL FLUSH
22:
23:       RETURN
24:       END
```

```
 1: *-h- RDMTAB 197  3 AUG 82  16:25:48
 2:       REAL FUNCTION RDMTAB(I, J)
 3:
 4: * read i,j-th element of plot mode table
 5:
 6:       INTEGER I, J
 7:       REAL PMODE(25,2)
 8:       COMMON /CPMTAB/ PMODE
 9:
10:       RDMTAB = PMODE(I,J)
11:
12:       RETURN
13:       END
```

```
 1: *-h- READGC 757  4 DEC 82  14:26:34
 2:       SUBROUTINE READGC(X, Y, KEY)
 3:
 4: * read graphics cursor position on screen when key is entered
 5:
 6:       INTEGER KEY
 7:       REAL X, Y
 8:       REAL PMODE(25,2)
 9:       COMMON /CPMTAB/ PMODE
10:       INTEGER BUF(18), PB, CX, CY, STR(4), PS
11:       INTEGER CTOI
12:
13:       STR(1) = 115
14:       STR(2) = 52
15:       STR(3) = 94
16:       STR(4) = -2
17:       PS = 3
18:       CALL OUTHPT(STR,PS)
19:
20:       PB = 17
21:       CALL INHP(BUF,PB)
22:
23:       PB = 1
24:       CX = CTOI(BUF,PB)
25:       X = FLOAT(CX) / (PMODE(1,1) * PMODE(15,1) * PMODE(15,2))
26:
27:       PB = PB + 1
28:       CY = CTOI(BUF,PB)
29:       Y = FLOAT(CY) / (PMODE(1,2) * PMODE(15,1) * PMODE(15,2))
30:
31:       PB = PB + 1
32:       KEY = CTOI(BUF,PB)
33:
34: *  update mode table
35:
36:       PMODE(8,1) = FLOAT(CX)
37:       PMODE(8,2) = FLOAT(CY)
38:
39:       RETURN
40:       END
```

```
 1: *-h- RESTOR 386 16 FEB 82  21:09:00
 2:       SUBROUTINE RESTOR(CLEAR)
 3:
 4: * restore terminal plot settings after ENDFIG, optionally clear screen
 5:
 6:       INTEGER CLEAR
 7:       INTEGER STR(5), PS
 8:
 9:       STR(1) = 100
10:       PS = 2
11:       IF (CLEAR .EQ. 0)
12:       .  STR(PS) = 97
13:       .  PS = PS + 1
14:       END IF
15:       STR(PS) = 99
16:       PS  = PS + 1
17:       STR(PS) = 102
18:       STR(PS+1) = -2
19:
20:       CALL OUTHPP(STR,PS)
21:
22:       RETURN
23:       END
```

```
 1: *-h- SCALE1 1410 28 MAY 83  12:35:01
 2:       SUBROUTINE SCALE1(XMIN, XMAX, N, XMINP, XMAXP, DIST)
 3:
 4: *  calculate range (xminp-xmaxp) divisible into approximately n linear
 5: *    intervals of size dist (1.2.5  integer power of 10) given input
 6: *    range (xmin-xmax)
 7:
 8:       INTEGER N
 9:       REAL XMIN, XMAX, XMINP, XMAXP, DIST
10:       INTEGER NAL, M
11:       REAL A, EPS, FN, FM, SQR(3), VINT(4)
12:       DATA SQR /1.414214,3.162278,7.071068/
13:       DATA VINT /1.0.2.0.5.0.10.0/
14:
15:       IF (XMIN .GT. XMAX .OR. N .LT. 0)
16:       .  CALL REMARK(" improper input to SCALE1")
17:       .  RETURN
18:       END IF
19:
20:       EPS = 0.00002
21:       FN = FLOAT(N)
22:
23: *  find approximate interval of size a
24:
25:       A = (XMAX - XMIN) / FN
26:       NAL = INT(ALOG10(A))
27:       IF (A .LT. 1.0) NAL = NAL - 1
28:
29: *  scale a between 1 and 10
30:
31:       A = A / 10.0 ** NAL
32:
33: *  find closest possible value for a
34:
35:       I = 1
36:       WHILE (A .GE. SQR(I))
37:       .  I = I + 1
38:       .  EXIT WHILE IF (I .GT. 3)
39:       END WHILE
40:
41: *  compute interval size
42:
43:       DIST = VINT(I) * 10.0 ** NAL
44:
45: *  find new min and max limits
46:
47:       FM = XMIN / DIST
48:       M = INT(FM)
49:       IF (FM .LT. 0.0) M = M - 1
50:       IF (ABS(FLOAT(M)+1.0-FM) .LT. EPS) M = M + 1
51:       XMINP = DIST * FLOAT(M)
52:       FM = XMAX / DIST
53:       M = INT(FM) + 1
54:       IF (FM .LT. -1.0) M = M - 1
55:       IF (ABS(FM+1.0-FLOAT(M)) .LT. EPS) M = M - 1
56:       XMAXP = DIST * FLOAT(M)
57:
58:       XMINP = AMIN1(XMINP,XMIN)
59:       XMAXP = AMAX1(XMAXP,XMAX)
60:
61:       RETURN
62:       END
```

```
 1: *-h- SCALE2 1603 28 MAY 83  12:35:05
 2:       SUBROUTINE SCALE2(XMIN, XMAX, N, XMINP, XMAXP, DIST)
 3:
 4: *  find new range (xminp-xmaxp) divisible into exactly n linear
 5: *    intervals of size dist (1.2.5 * integer power of 10) given input
 6: *    range (xmin-xmax)
 7:
 8:       INTEGER N
 9:       REAL XMIN, XMAX, XMINP, XMAXP, DIST
10:       INTEGER I, M1, M2, NAL, NP, NX
11:       REAL A, EPS, FN, FM, VINT(5)
12:       DATA VINT /1.0.2.0.5.0.10.0.20.0/
13:
14:       IF (XMIN .GT. XMAX .OR. N .LE. 1)
15:       .  CALL REMARK(" improper input supplied to SCALE2")
16:       .  RETURN
17:       END IF
```

```
18:
19:       EPS = 0.00002
20:       FN = FLOAT(N)
21:
22:
23: * find approximate interval of size a
24:
25:       A  = (XMAX - XMIN) / FN
26:       NAL = INT(ALOG10(A))
27:       IF (A .LT. 1.0) NAL = NAL - 1
28:
29: *  scale a between 1 and 10
30:
31:       A = A / 10.0 ** NAL
32:
33: *  find closest permissible value for b
34:
35:       I = 1
36:       WHILE (A .GE. (VINT(I)+EPS))
37:       .  I = I + 1
38:       .  EXIT WHILE IF (I .GT. 3)
39:       END WHILE
40:
41:       DO
42: *  compute interval size. (may need second pass)
43:
44:       .  DIST = VINT(I) * 10.0 ** NAL
45:       .
46: *  find new min and max limits
47:       .
48:       .  FM = XMIN / DIST
49:       .  M1 = INT(FM)
50:       .  IF (FM .LT. 0.0) M1 = M1 - 1
51:       .  IF (ABS(FLOAT(M1)+1.0-FM) .LT. EPS) M1 = M1 + 1
52:       .  XMINP = DIST * FLOAT(M1)
53:       .  FM = XMAX / DIST
54:       .  M2 = INT(FM) + 1.0
55:       .  IF (FM .LT. -1.0) M2 = M2 - 1
56:       .  IF (ABS(FM+1.0-FLOAT(M2)) .LT. EPS) M2 = M2 - 1
57:       .  XMAXP = DIST * FLOAT(M2)
58:       .  NP = M2 - M1
59:       .  I = I + 1
60:       UNTIL (NP .LE. N)
61:
62:       NX = (N - NP) / 2
63:       XMINP = XMINP - FLOAT(NX) * DIST
64:       XMAXP = XMINP + FLOAT(N) * DIST
65:
66:       XMINP = AMIN1(XMINP,XMIN)
67:       XMAXP = AMAX1(XMAXP,XMAX)
68:
69:       RETURN
70:       END


 1: *-h- SETDRM 383  1 AUG 82  14:25:09
 2:       SUBROUTINE SETDRM(MODE)
 3:
 4: * set drawing mode: 1=clear, 2=set, 3=complement, 4=jam
 5:
 6:       INTEGER MODE
 7:       REAL PMODE(25,2)
 8:       COMMON /CPMTAB/ PMODE
 9:       INTEGER STR(4), PS
10:
11:       STR(1) = 109
12:       STR(2) = 48 + MODE
13:       STR(3) = 97
14:       STR(4) = -2
15:       PS = 3
16:
17:       CALL OUTHPP(STR,PS)
18:
19: *  update mode table
20:
21:       PMODE(18,1) = FLOAT(MODE)
22:
23:       RETURN
24:       END
```

```
 1: *-h- TIC 2723  9 MAR 83  12:05:16
 2:       SUBROUTINE TIC(X, Y, A, FIELD, DEC, HEIGHT, OPT, ANG)
 3:
 4: ************************************************************************
 5: *                                                                      *
 6: * draw tic mark at (x,y)                                               *
 7: *   if opt = {1-4} annotate with a                                     *
 8: *                                                                      *
 9: *     X, Y       :   start coordinates of tic mark                     *
10: *     A          :   real number associated with tic mark              *
11: *     FIELD, DEC:    width and decimal of annotation                   *
12: *     HEIGHT     :   height of digits                                  *
13: *     OPT        :   annotation option                                 *
14: *                     opt = 1  digits start at tic and parallel        *
15: *                     opt = 2  digits end at tic and parallel          *
16: *                     opt = 3  digits below tic and perpendicular      *
17: *                     opt = 4  digits above tic and perpendicular      *
18: *     ANG        :   angle of tic mark                                 *
19: *                                                                      *
20: ************************************************************************
21:
22:       INTEGER FIELD, DEC, OPT
23:       REAL X, Y, A, HEIGHT, ANG
24:       INTEGER LEN, OO, PENDOWN, PENUP, STR(10)
25:       REAL CA, DR, SA, XX, YY
26:       INTEGER FTOC
27:       REAL COS, SIN, RDMTAB
28:       DATA PENDOWN /1/, PENUP /0/, DR /0.01745329/
29:
30:       OO = INT(RDMTAB(17.2))
31:       CA = COS(DR*ANG)
32:       SA = SIN(DR*ANG)
33:       LEN = FTOC(A,STR,FIELD,DEC)
34:
35:       CALL NEWPEN(3)
36:
37:       XX = X
38:       YY = Y
39:       CALL MOVEPN(X,Y,OO,PENUP)
40:       XX = XX + 0.5 * HEIGHT * CA
41:       YY = YY + 0.5 * HEIGHT * SA
42:       CALL MOVEPN(XX,YY,OO,PENDOWN)
43:
44:       IF (OPT .EO. 1)
45:      .  XX = XX + 0.5 * HEIGHT * CA + 0.5 * HEIGHT * SA
46:      .  YY = YY + 0.5 * HEIGHT * SA - 0.5 * HEIGHT * CA
47:      .  CALL GRNUM(XX,YY,A,FIELD,DEC,HEIGHT,ANG)
48:       ELSE IF (OPT .EO. 2)
49:      .  XX = XX + (FLOAT(LEN) + 0.5) * HEIGHT * CA - 0.5 * HEIGHT * SA
50:      .  YY = YY + (FLOAT(LEN) + 0.5) * HEIGHT * SA + 0.5 * HEIGHT * CA
51:      .  CALL GRNUM(XX,YY,A,FIELD,DEC,HEIGHT,ANG+180.0)
52:       ELSE IF (OPT .EO. 3)
53:      .  XX = XX + 1.5 * HEIGHT * CA + 0.5 * FLOAT(LEN) * HEIGHT * SA
54:      .  YY = YY + 1.5 * HEIGHT * SA - 0.5 * FLOAT(LEN) * HEIGHT * CA
55:      .  CALL GRNUM(XX,YY,A,FIELD,DEC,HEIGHT,ANG+90.0)
56:       ELSE IF (OPT .EO. 4)
57:      .  XX = XX + 0.5 * HEIGHT * CA - 0.5 * FLOAT(LEN) * HEIGHT * SA
58:      .  YY = YY + 0.5 * HEIGHT * SA + 0.5 * FLOAT(LEN) * 0.5 * HEIGHT * CA
59:      .  CALL GRNUM(XX,YY,A,FIELD,DEC,HEIGHT,ANG+270.0)
60:       END IF
61:
62:       CALL NEWPEN(1)
63:
64:       RETURN
65:       END
```

```
 1: *-h- VP07MP 132 17 AUG 82  11:36:49
 2:       SUBROUTINE VP07MP(A1, A2, A3)
 3:
 4: * dummy routine to emulate VERSAPLOT 07 call
 5:
 6:       INTEGER A1, A2, A3
 7:
 8:       RETURN
 9:       END
```

```
 1: *-h- ZOOM 867  1 AUG 82  14:25:10
 2:       SUBROUTINE ZOOM(X, Y, N)
 3:
 4: * set zoom factor (1-8) at <x,y>
 5:
 6:       INTEGER N
 7:       REAL X, Y
 8:       REAL PMODE(25,2)
 9:       COMMON /CPMTAB/ PMODE
10:       INTEGER STR(16), PS, CX, CY, L, OO
11:       INTEGER ITOC
12:
13:       STR(1) = 100
14:       STR(2) = 32
15:
16: *  set zoom size
17:
18:       PS = 3
19:       L = ITOC(N,STR(PS),2)
20:       PS = PS + L
21:       STR(PS) = 105
22:       PS = PS + 1
23:
24: *  set zoom coordinates
25:
26:       CX = INT(X*PMODE(1,1)*PMODE(15,1)*PMODE(15,2))
27:       CY = INT(Y*PMODE(1,2)*PMODE(15,1)*PMODE(15,2))
28:
29:       L = ITOC(CX,STR(PS),4)
30:       PS = PS + L
31:       STR(PS) = 32
32:       PS = PS + 1
33:       L = ITOC(CY,STR(PS),4)
34:       PS = PS + L
35:       STR(PS) = 106
36:       PS = PS + 1
37:       STR(PS) = 103
38:       STR(PS+1) = -2
39:
40:       CALL OUTHPP(STR,PS)
41:
42: *  update mode table
43:
44:       PMODE(9,1) = FLOAT(CX)
45:       PMODE(9,2) = FLOAT(CY)
46:       PMODE(16,1) = FLOAT(N)
47:
48:       RETURN
49:       END
```

APPENDIX B.

FIRST ARRIVAL PICK TRAVEL TIMES.

Receiver 2 Shot line 3N

Shot	Range	Time	Err
4496	57.595	10.657	0.120
4497	56.827	10.577	0.100
4498	56.101	10.477	0.100
4499	55.346	10.367	0.100
4500	54.632	10.248	0.100
4501	53.889	10.128	0.100
4502	53.196	10.198	0.100
4503	52.478	10.068	0.100
4504	51.758	9.928	0.100
4505	51.020	9.827	0.100
4506	50.321	9.617	0.100
4507	49.583	9.468	0.090
4508	48.829	9.468	0.100
4509	48.100	9.367	0.090
4510	47.365	9.197	0.080
4511	46.605	9.077	0.080
4512	45.873	8.897	0.100
4513	45.150	8.798	0.080
4514	44.430	8.747	0.080
4515	43.724	8.647	0.100
4516	43.020	8.547	0.080
4518	41.580	8.357	0.080
4519	40.875	8.317	0.080
4520	40.109	8.197	0.080
4521	39.327	8.068	0.080
4522	38.541	7.937	0.080
4523	37.623	7.798	0.070
4524	36.650	7.687	0.070
4525	35.711	7.557	0.060
4526	34.707	7.409	0.060
4527	33.707	7.258	0.060
4528	32.713	7.118	0.050
4529	31.734	6.978	0.050
4530	30.732	6.808	0.060
4531	29.713	6.658	0.050
4532	28.730	6.488	0.050
4533	27.685	6.298	0.050
4534	26.665	6.168	0.050
4535	25.639	6.018	0.050
4536	24.666	5.848	0.050
4537	23.599	5.678	0.050
4538	22.594	5.548	0.050
4539	21.511	5.428	0.040
4540	20.468	5.258	0.040
4541	19.485	5.139	0.040
4542	18.488	5.008	0.050
4543	17.469	4.888	0.040
4544	16.453	4.699	0.060
4545	15.458	4.539	0.050
4546	14.451	4.418	0.040
4547	13.500	4.298	0.040
4548	12.523	4.128	0.040
4549	11.505	3.958	0.040
4550	10.469	3.799	0.040
4551	9.468	3.618	0.030
4552	8.432	3.438	0.030
4553	7.422	3.268	0.040
4554	6.384	3.089	0.030
4555	5.326	2.879	0.030
4556	4.274	2.679	0.030
4557	3.193	2.469	0.030
4558	2.027	2.240	0.030
4563	1.827	2.209	0.030

Receiver 3 Shot line 3N

Shot	Range	Time	Err
4475	38.058	7.904	0.080
4476	37.313	7.724	0.080
4477	36.617	7.633	0.100
4478	35.863	7.493	0.080
4479	35.187	7.414	0.070
4480	34.479	7.334	0.070
4481	33.760	7.283	0.060
4482	33.034	7.194	0.070
4483	32.351	7.043	0.060
4484	31.645	6.973	0.050
4485	30.903	6.895	0.050
4487	29.475	6.674	0.060
4488	28.716	6.555	0.050
4489	28.073	6.484	0.050
4490	27.358	6.374	0.050
4491	26.600	6.274	0.050
4492	25.892	6.144	0.050
4493	25.171	6.064	0.040
4494	24.280	5.894	0.060
4495	23.569	5.834	0.050
4496	22.852	5.735	0.050

4497	22.094	5.655	0.050
4498	21.406	5.555	0.040
4499	20.674	5.465	0.050
4500	19.980	5.356	0.050
4501	19.226	5.206	0.050
4502	18.526	5.126	0.040
4503	17.808	5.016	0.050
4504	17.089	4.856	0.060
4505	16.329	4.775	0.050
4506	15.652	4.725	0.040
4507	14.897	4.626	0.040
4508	14.142	4.516	0.040
4509	13.429	4.426	0.040
4510	12.694	4.286	0.050
4511	11.945	4.156	0.040
4512	11.196	4.066	0.050
4513	10.499	3.987	0.030
4514	9.752	3.896	0.040
4515	9.046	3.776	0.040
4516	8.314	3.666	0.040
4518	6.892	3.406	0.030
4519	6.139	3.286	0.040
4520	5.343	3.136	0.030
4521	4.587	3.007	0.030
4522	3.788	2.827	0.030
4523	2.775	2.688	0.030
4524	1.595	2.457	0.030
4529	1.634	2.507	0.030
4530	2.922	2.717	0.030
4531	3.991	2.897	0.030
4532	5.063	3.067	0.040
4533	6.043	3.238	0.030
4534	7.112	3.368	0.040
4535	8.166	3.508	0.050
4536	9.193	3.688	0.030
4537	10.201	3.808	0.030
4538	11.255	3.978	0.030
4539	12.340	4.128	0.030
4540	13.351	4.278	0.040
4541	14.409	4.449	0.050
4542	15.456	4.608	0.050
4543	16.507	4.748	0.040
4544	17.513	4.939	0.050
4545	18.611	5.089	0.040
4546	19.621	5.259	0.050
4547	20.737	5.389	0.050
4548	21.669	5.529	0.040
4549	22.622	5.659	0.060
4550	23.617	5.790	0.050
4551	24.601	5.959	0.060
4552	25.539	6.039	0.060
4553	26.581	6.159	0.060
4554	27.547	6.299	0.050
4555	28.514	6.449	0.050
4556	29.521	6.569	0.060
4557	30.505	6.650	0.060
4558	31.490	6.771	0.050
4559	32.448	6.881	0.060
4560	33.488	7.010	0.060
4561	34.478	7.171	0.060
4562	35.434	7.310	0.060
4563	36.452	7.470	0.060

Receiver 4 Shot line 2N

Shot	Range	Time	Err
3336	40.928	8.180	0.090
3337	40.098	8.121	0.080
3338	39.252	8.071	0.070
3339	38.424	8.041	0.090
3340	37.560	7.887	0.060
3341	36.717	7.809	0.070
3342	35.888	7.721	0.070
3343	35.027	7.668	0.070
3344	34.195	7.551	0.060
3345	33.406	7.442	0.060
3346	32.555	7.328	0.060
3347	31.724	7.220	0.050
3348	30.854	7.080	0.050
3349	30.007	6.974	0.050
3350	29.160	6.858	0.050
3351	28.308	6.732	0.040
3352	27.493	6.639	0.050
3353	26.702	6.539	0.040
3354	25.962	6.419	0.060
3355	25.201	6.338	0.040
3357	23.663	6.081	0.050
3358	22.854	5.916	0.050
3359	22.049	5.813	0.040
3360	21.302	5.662	0.040
3361	20.585	5.540	0.040
3362	19.896	5.433	0.040
3363	19.192	5.325	0.040
3364	18.519	5.230	0.040

3365	17.849	5.095	0.040
3366	17.212	4.944	0.040
3367	16.613	4.914	0.040
3368	16.005	4.877	0.040
3369	15.467	4.733	0.040
3370	14.949	4.656	0.040
3371	14.471	4.518	0.040
3372	14.016	4.427	0.040
3373	13.651	4.375	0.040
3374	13.312	4.302	0.040
3375	13.083	4.265	0.040
3376	12.883	4.234	0.040
3377	12.750	4.242	0.050
3378	12.676	4.173	0.040
3379	12.649	4.196	0.050
3380	12.677	4.194	0.040
3381	12.769	4.207	0.040
3382	12.839	4.175	0.040
3383	12.968	4.162	0.040
3384	13.135	4.203	0.040
3385	13.360	4.203	0.040
3386	13.601	4.245	0.040
3387	13.886	4.267	0.060
3388	14.237	4.324	0.050
3389	14.603	4.347	0.040
3390	14.982	4.404	0.040
3391	15.354	4.383	0.040
3392	15.823	4.424	0.040
3393	16.320	4.502	0.040
3394	16.838	4.576	0.040
3395	17.351	4.633	0.080
3396	17.919	4.696	0.080
3397	18.441	4.754	0.050
3398	19.047	4.816	0.050
3399	19.631	4.891	0.040
3400	20.254	4.979	0.050
3401	20.840	5.075	0.040
3402	21.447	5.202	0.050
3403	22.089	5.295	0.050
3404	22.750	5.402	0.060
3405	23.413	5.542	0.040
3406	24.045	5.683	0.040
3407	24.753	5.743	0.040
3408	25.437	5.899	0.040
3409	26.122	5.980	0.050
3410	26.835	6.124	0.060
3411	27.561	6.261	0.050
3412	28.295	6.342	0.050
3413	29.041	6.506	0.050
3414	29.830	6.624	0.060
3415	30.845	6.732	0.060
3416	31.630	6.802	0.080
3417	32.423	6.872	0.080
3418	33.212	7.230	0.100
3419	33.974	7.488	0.100
3420	34.779	7.588	0.100
3421	35.518	7.516	0.150

Receiver 4 Shot line 4N

Shot	Range	Time	Err
4564	37.779	7.942	0.070
4565	36.985	7.883	0.100
4566	36.160	7.742	0.080
4567	35.368	7.633	0.060
4568	34.602	7.523	0.070
4569	33.835	7.402	0.070
4570	33.026	7.302	0.060
4571	32.301	7.282	0.060
4572	31.563	7.153	0.060
4573	30.842	7.022	0.060
4574	30.188	6.923	0.050
4575	29.453	6.831	0.060
4576	28.506	6.711	0.050
4577	27.817	6.631	0.050
4578	27.169	6.561	0.040
4579	26.452	6.411	0.050
4580	25.738	6.311	0.040
4581	25.005	6.211	0.050
4582	24.270	6.111	0.060
4583	23.562	6.061	0.050
4584	22.823	5.932	0.040
4585	22.132	5.801	0.050
4586	21.445	5.721	0.050
4587	20.856	5.690	0.040
4588	20.235	5.600	0.040
4589	19.397	5.450	0.040
4590	18.698	5.330	0.040
4591	17.978	5.240	0.040
4592	17.163	5.081	0.030
4593	16.504	5.011	0.050
4594	15.833	4.921	0.040
4595	15.104	4.800	0.040
4596	14.460	4.691	0.040

4597	13.723	4.540	0.030
4598	13.072	4.439	0.040
4599	12.405	4.349	0.030
4600	11.673	4.189	0.030
4601	11.003	4.079	0.050
4602	10.334	3.999	0.040
4603	9.675	3.909	0.040
4604	8.998	3.820	0.040
4605	8.367	3.769	0.050
4606	7.679	3.630	0.040
4607	7.012	3.530	0.030
4608	6.364	3.439	0.050
4609	5.732	3.328	0.050
4610	5.166	3.258	0.040
4611	4.589	3.179	0.040
4612	4.102	2.969	0.040
4613	3.567	2.878	0.030
4614	3.389	2.789	0.030
4615	3.229	2.778	0.030
4616	3.206	2.748	0.030
4617	3.414	2.768	0.030
4618	3.747	2.808	0.030
4619	4.191	2.839	0.040
4620	4.686	2.899	0.030
4622	5.928	3.138	0.040
4623	6.578	3.218	0.040
4624	7.302	3.347	0.050
4625	8.021	3.518	0.030
4626	8.786	3.567	0.040
4627	9.552	3.657	0.040
4628	10.288	3.757	0.040
4629	11.107	3.918	0.040
4630	11.896	4.108	0.040
4633	14.429	4.456	0.050
4634	15.294	4.626	0.040
4635	16.110	4.576	0.100
4636	16.989	4.657	0.080
4637	17.860	4.846	0.060
4638	18.699	5.136	0.060
4639	19.533	5.296	0.050

Receiver 5 Shot line 1N

Shot	Range	Time	Err
3243	24.368	6.408	0.080
3244	23.691	6.202	0.080
3245	23.026	6.026	0.070
3246	22.362	5.946	0.090
3247	21.711	5.918	0.060
3248	21.053	5.942	0.060
3249	20.364	5.996	0.060
3250	19.637	5.893	0.060
3251	18.965	5.786	0.060
3252	18.283	5.614	0.050
3253	17.597	5.395	0.060
3254	16.921	5.244	0.050
3255	16.212	5.142	0.080
3256	15.446	4.874	0.060
3257	14.659	4.736	0.060
3258	13.893	4.597	0.050
3259	13.323	4.431	0.050
3260	12.622	4.312	0.050
3261	11.873	4.200	0.050
3262	11.080	4.042	0.040
3263	10.171	3.803	0.040
3264	9.257	3.531	0.040
3265	8.316	3.288	0.040
3266	7.356	3.082	0.040
3267	6.394	2.810	0.050
3268	5.388	2.610	0.040
3269	4.431	2.482	0.040
3270	3.470	2.407	0.050
3271	2.522	2.207	0.030
3272	1.625	2.033	0.030
3273	0.870	1.833	0.030
3274	0.489	1.752	0.030
3275	1.131	1.897	0.030
3276	2.053	2.209	0.030
3277	2.950	2.389	0.030
3278	3.841	2.589	0.040
3279	4.780	2.784	0.040
3280	5.695	3.034	0.040
3281	6.417	3.154	0.050
3283	8.430	3.569	0.040
3284	9.328	3.736	0.040
3285	10.285	3.914	0.060

3286	11.125	4.031	0.040
3287	12.008	4.185	0.050
3288	12.920	4.340	0.040
3289	13.825	4.520	0.040
3290	14.741	4.650	0.040
3291	15.623	4.755	0.050
3292	16.532	4.926	0.040
3293	17.435	5.097	0.040
3294	18.370	5.225	0.040
3295	19.286	5.343	0.040
3296	20.386	5.467	0.040
3297	21.421	5.659	0.040
3298	22.480	5.786	0.050
3299	23.496	5.936	0.040
3300	24.571	6.130	0.040
3301	25.646	6.317	0.040
3302	26.740	6.489	0.050
3303	27.773	6.673	0.050
3304	28.828	6.811	0.050
3305	29.701	6.978	0.050
3306	30.947	7.165	0.050
3307	32.005	7.363	0.050
3308	33.067	7.503	0.060
3309	34.130	7.684	0.070
3310	35.164	7.815	0.080
3311	36.232	7.971	0.080
3312	37.243	8.089	0.080
3313	38.300	8.167	0.100
3314	39.374	8.352	0.100
3315	40.448	8.539	0.090
3317	42.376	8.735	0.080
3318	43.038	8.808	0.100
3319	43.715	8.886	0.120
3322	45.520	9.327	0.120
3323	46.128	9.398	0.100

Receiver 5 Shot line 2N

Shot	Range	Time	Err
3343	51.246	9.927	0.120
3344	50.555	9.810	0.150
3345	49.896	9.771	0.150
3346	49.207	9.666	0.150
3347	48.519	9.578	0.120
3348	47.807	9.488	0.120
3349	47.091	9.332	0.130
3351	45.730	9.140	0.150
3352	45.043	9.078	0.140
3353	44.374	9.088	0.130
3354	43.727	9.027	0.120
3355	43.078	8.926	0.100
3356	42.430	8.809	0.080
3357	41.777	8.679	0.100
3358	41.103	8.604	0.100
3359	40.439	8.511	0.080
3360	39.834	8.430	0.070
3361	39.271	8.388	0.070
3362	38.784	8.391	0.090
3363	38.337	8.262	0.070
3364	37.813	8.238	0.070
3365	37.300	8.103	0.060
3366	36.823	8.022	0.060
3367	36.315	7.962	0.060
3368	35.803	7.865	0.060
3369	35.265	7.761	0.050
3370	34.734	7.674	0.060
3371	34.229	7.555	0.060
3374	32.675	7.429	0.100
3377	31.665	7.230	0.070
3378	31.372	7.181	0.060
3379	31.083	7.123	0.080
3380	30.809	7.151	0.080
3381	30.617	7.004	0.070
3384	30.167	6.920	0.060
3385	30.071	6.880	0.080
3386	29.999	6.992	0.080
3387	29.970	7.004	0.070
3388	29.942	6.951	0.060
3389	29.940	6.874	0.080
3390	29.966	6.821	0.060
3391	29.987	6.780	0.080
3392	30.047	6.701	0.060
3393	30.093	6.609	0.060
3394	30.156	6.613	0.050
3396	30.348	6.602	0.050
3397	30.495	6.650	0.050
3398	30.656	6.682	0.050
3399	30.825	6.718	0.050
3400	31.019	6.776	0.050
3401	31.427	6.862	0.060
3402	31.738	6.949	0.050
3403	32.032	7.012	0.060
3404	32.320	7.058	0.060
3405	32.679	7.098	0.050

3406	32.982	7.119	0.070
3407	33.351	7.099	0.050
3408	33.686	7.135	0.060
3409	34.037	7.226	0.080
3410	34.368	7.281	0.080
3411	34.761	7.338	0.080
3412	35.160	7.408	0.100
3413	35.557	7.442	0.080
3414	35.884	7.440	0.070
3415	36.300	7.538	0.060

Receiver 6 Shot line 1N

Shot	Range	Time	Err
3252	39.357	8.371	0.100
3253	38.643	8.272	0.120
3254	37.935	8.121	0.100
3255	37.187	7.989	0.100
3256	36.446	7.841	0.090
3257	35.674	7.703	0.090
3258	34.900	7.514	0.080
3259	34.134	7.418	0.080
3260	33.395	7.269	0.080
3261	32.626	7.198	0.070
3262	31.781	7.080	0.080
3263	30.905	6.901	0.070
3264	29.985	6.679	0.070
3265	29.095	6.476	0.060
3266	28.211	6.340	0.070
3267	27.298	6.138	0.060
3268	26.349	5.938	0.060
3269	25.429	5.811	0.070
3271	23.528	5.666	0.060
3272	22.587	5.612	0.050
3273	21.668	5.512	0.050
3274	20.551	5.401	0.050
3275	19.640	5.256	0.060
3276	18.727	5.178	0.050
3277	17.875	5.049	0.050
3278	16.962	5.009	0.070
3279	16.035	4.894	0.040
3280	15.139	4.754	0.050
3281	14.422	4.634	0.040
3282	13.357	4.442	0.040
3283	12.441	4.269	0.040
3284	11.543	4.097	0.050
3285	10.595	3.965	0.040
3286	9.761	3.762	0.050
3287	8.853	3.627	0.040
3288	7.960	3.381	0.050
3289	7.046	3.222	0.040
3290	6.134	3.021	0.040
3291	5.240	2.866	0.030
3292	4.323	2.647	0.030
3293	3.438	2.479	0.030
3294	2.602	2.288	0.030
3295	1.802	2.086	0.030
3296	1.107	1.950	0.030
3297	1.247	1.972	0.030
3298	2.051	2.179	0.040
3299	2.928	2.429	0.030
3300	3.876	2.663	0.030
3301	4.909	2.910	0.050
3302	5.932	3.172	0.040
3303	6.968	3.427	0.050
3304	8.008	3.595	0.040
3305	8.885	3.782	0.040
3306	10.140	3.989	0.040
3307	11.174	4.187	0.050
3308	12.226	4.347	0.050
3309	13.296	4.498	0.050
3310	14.310	4.640	0.080
3311	15.352	4.766	0.050
3312	16.400	5.024	0.060
3313	17.433	5.102	0.060
3314	18.448	5.217	0.060
3315	19.499	5.334	0.070
3316	20.490	5.484	0.060
3315	19.499	5.344	0.080
3316	20.490	5.484	0.060
3317	21.375	5.620	0.060
3318	22.027	5.763	0.050
3319	22.657	5.862	0.060
3320	23.265	6.004	0.060
3321	23.845	6.090	0.060
3322	24.471	6.213	0.060
3323	25.076	6.304	0.040
3324	25.736	6.398	0.050
3325	26.372	6.489	0.040
3326	27.007	6.558	0.050
3327	27.644	6.650	0.040
3328	28.278	6.750	0.050
3329	29.245	6.869	0.040
3330	29.911	6.949	0.050

Receiver 205 Shot line 2N

Shot	Range	Time	Err
3372	35.124	7.577	0.110
3375	32.686	7.235	0.120
3376	31.892	7.239	0.100
3377	30.923	7.108	0.100
3378	30.431	7.039	0.100
3379	29.771	7.021	0.100
3380	29.017	6.800	0.080
3381	28.448	6.701	0.090
3382	27.666	6.540	0.100
3383	26.945	6.441	0.120
3384	26.393	6.535	0.100
3385	25.926	6.257	0.100
3386	24.981	6.134	0.080
3387	24.305	6.039	0.080
3388	23.583	5.862	0.090
3389	22.971	5.775	0.080
3390	22.584	5.637	0.100
3391	21.638	5.523	0.080
3392	21.160	5.454	0.090
3393	20.648	5.337	0.090
3394	20.049	5.260	0.080
3395	19.528	5.235	0.080
3396	18.809	5.184	0.080
3397	18.302	4.989	0.080
3398	17.791	4.928	0.100
3399	17.426	4.849	0.100
3400	16.780	4.800	0.100
3401	16.262	4.690	0.090
3402	15.939	4.704	0.080
3403	15.449	4.663	0.100
3404	15.096	4.480	0.070
3405	14.770	4.610	0.080
3406	14.530	4.604	0.080
3407	14.274	4.574	0.090
3408	14.157	4.558	0.070
3409	14.030	4.514	0.060
3410	13.859	4.490	0.100
3411	13.737	4.544	0.100
3412	13.779	4.422	0.110
3413	13.974	4.442	0.080
3414	14.158	4.467	0.090
3415	14.252	4.453	0.080
3416	14.548	4.474	0.090
3417	14.819	4.555	0.090
3418	15.121	4.553	0.090
3419	15.551	4.683	0.100
3420	15.889	4.870	0.100
3420	15.889	4.870	0.100
3421	16.341	4.964	0.120

Receiver 210 Shot line 1N

Shot	Range	Time	Err
3283	30.855	6.939	0.080
3284	30.361	6.949	0.080
3287	29.492	6.754	0.080
3288	29.306	6.726	0.080
3289	29.016	6.792	0.081
3290	28.807	6.927	0.110
3291	28.681	6.586	0.100
3292	28.554	6.580	0.090
3294	28.363	6.602	0.150
3296	28.170	6.434	0.090
3297	28.170	6.499	0.090
3298	28.298	6.531	0.090
3299	28.341	6.580	0.110
3300	28.341	6.705	0.120
3301	28.488	6.755	0.100
3302	28.511	6.717	0.090
3303	28.744	6.833	0.100
3304	28.870	7.009	0.100
3305	29.120	6.947	0.060
3306	29.309	7.038	0.060
3307	29.696	7.039	0.080
3308	30.040	7.124	0.090
3309	30.361	7.094	0.100
3310	30.855	7.201	0.100
3312	31.839	7.241	0.100
3313	32.179	7.363	0.080
3314	33.014	7.522	0.100

Receiver 210 Shot line 2N

Shot	Range	Time	Err
3342	21.949	5.904	0.070
3344	20.169	5.720	0.090
3345	19.424	5.620	0.100
3346	18.587	5.407	0.080
3347	17.483	5.251	0.080
3348	16.489	5.139	0.090
3349	15.635	4.986	0.080
3350	14.696	4.876	0.070
3351	13.375	4.791	0.081
3352	12.871	4.615	0.090
3353	11.989	4.489	0.070
3354	11.157	4.415	0.090
3355	10.228	4.211	0.080
3356	9.442	4.068	0.080
3357	8.554	3.908	0.080
3359	6.864	3.562	0.060
3360	5.942	3.404	0.040
3362	4.507	3.230	0.040
3363	3.766	3.072	0.040
3364	3.150	2.915	0.040
3365	2.907	2.812	0.040
3366	2.805	2.781	0.040
3368	3.345	2.849	0.050
3369	3.904	2.933	0.060
3370	4.525	3.056	0.060
3371	5.372	3.190	0.060
3372	6.212	3.306	0.060
3373	7.016	3.427	0.080
3376	9.564	3.928	0.080
3377	10.316	4.059	0.100
3379	11.898	4.310	0.080
3380	12.711	4.421	0.110
3381	13.506	4.593	0.090
3382	14.435	4.690	0.110
3383	15.130	4.772	0.120
3384	15.692	4.860	0.090
3386	17.353	5.052	0.090
3387	18.333	5.222	0.100
3388	19.080	5.280	0.100
3390	20.870	5.518	0.080
3391	21.531	5.589	0.100
3392	22.109	5.654	0.100
3393	23.150	5.740	0.090
3394	23.862	5.885	0.100
3397	26.261	6.249	0.090
3399	27.436	6.440	0.081
3401	29.217	6.635	0.080
3402	30.159	6.763	0.080
3403	30.821	6.838	0.080
3404	31.788	6.987	0.080
3405	32.557	7.151	0.080
3406	33.308	7.258	0.080
3407	34.196	7.334	0.080
3408	34.946	7.315	0.080
3409	35.710	7.526	0.080
3410	36.686	7.653	0.080
3411	37.474	7.776	0.080
3412	38.324	7.922	0.090
3413	39.272	7.988	0.090
3416	41.923	8.480	0.120
3417	42.446	8.548	0.100

Receiver 210 Shot line 4N

Shot	Range	Time	Err
4587	32.875	7.577	0.150
4588	32.294	7.439	0.100
4589	31.693	7.349	0.100
4590	30.820	7.185	0.090
4592	29.970	6.989	0.090
4594	28.893	6.896	0.080
4595	28.404	6.802	0.090
4596	27.940	6.749	0.080
4597	27.143	6.595	0.100
4598	26.418	6.491	0.080
4599	25.811	6.448	0.090
4600	25.171	6.277	0.070
4607	20.654	5.702	0.100
4609	19.229	5.563	0.100
4610	18.616	5.423	0.080
4613	17.135	5.024	0.060
4616	15.732	4.683	0.060
4617	15.288	4.663	0.060
4618	14.981	4.596	0.060
4619	14.631	4.646	0.070
4620	14.298	4.605	0.080
4621	14.058	4.517	0.060
4622	13.820	4.460	0.100

4623	13.642	4.499	0.060
4624	13.613	4.512	0.080
4626	13.563	4.394	0.090
4628	13.748	4.511	0.100
4629	13.818	4.513	0.090
4631	14.309	4.653	0.100
4633	14.845	4.775	0.100
4634	15.162	4.802	0.081
4635	15.479	4.614	0.080
4638	16.771	5.010	0.060
4639	17.258	5.176	0.060
4642	18.655	5.391	0.060
4643	19.390	5.513	0.060
4645	22.212	5.843	0.050
4646	22.972	5.915	0.050
4647	23.811	6.007	0.060
4649	24.796	6.357	0.060
4650	25.634	6.496	0.060
4651	26.246	6.661	0.060
4652	26.890	6.785	0.060
4653	27.638	7.019	0.080

Receiver 526 Shot line 3N

Shot	Range	Time	Err
4491	39.163	8.210	0.100
4492	38.418	8.085	0.088
4493	37.676	8.035	0.100
4494	36.947	7.885	0.088
4495	36.220	7.785	0.088
4496	35.504	7.598	0.075
4497	34.766	7.573	0.088
4498	34.040	7.522	0.075
4499	33.303	7.372	0.075
4500	32.579	7.272	0.063
4501	31.855	7.173	0.063
4502	31.120	7.060	0.063
4503	30.385	6.885	0.063
4504	29.656	6.860	0.100
4505	28.913	6.760	0.075
4506	28.182	6.760	0.063
4507	27.434	6.585	0.063
4508	26.688	6.460	0.050
4509	25.941	6.423	0.075
4510	25.204	6.298	0.063
4511	24.453	6.173	0.063
4512	23.708	6.073	0.063
4513	22.965	5.973	0.063
4514	22.227	5.860	0.063
4515	21.507	5.785	0.075
4516	20.766	5.685	0.050
4518	19.345	5.460	0.063
4519	18.607	5.360	0.075
4520	17.866	5.223	0.050
4521	17.104	5.147	0.050
4522	16.350	4.998	0.063
4523	15.430	4.860	0.063
4524	14.491	4.710	0.050
4525	13.580	4.585	0.050
4526	12.666	4.423	0.063
4527	11.756	4.198	0.063
4528	10.869	4.035	0.050
4529	10.013	3.897	0.050
4530	9.182	3.772	0.050
4531	8.383	3.660	0.050
4532	7.706	3.510	0.050
4533	7.090	3.360	0.050
4534	6.593	3.247	0.050
4535	6.243	3.173	0.050
4536	6.044	3.122	0.050
4537	6.038	3.035	0.050
4538	5.768	3.085	0.063
4539	5.632	3.072	0.050
4540	6.161	3.135	0.063
4541	6.506	3.197	0.063
4542	7.200	3.335	0.063
4543	7.968	3.397	0.050
4544	8.803	3.610	0.075
4545	9.689	3.760	0.063
4546	10.595	3.947	0.075
4547	11.536	4.160	0.075
4548	12.303	4.260	0.063
4549	13.078	4.372	0.050
4550	13.901	4.473	0.063
4551	14.716	4.560	0.063
4552	15.556	4.698	0.075
4553	16.394	4.798	0.063
4554	17.247	4.973	0.075
4555	18.083	5.022	0.063
4556	18.950	5.147	0.063
4557	19.816	5.235	0.063
4558	20.691	5.310	0.075
4559	21.556	5.485	0.088
4560	22.446	5.535	0.075

4561	23.328	5.723	0.088
4562	24.223	5.835	0.075
4563	25.161	5.960	0.088

Receiver 526 Shot line 4N

Shot	Range	Time	Err
4564	23.759	6.460	0.100
4565	23.018	6.310	0.088
4566	22.280	6.298	0.088
4567	21.567	6.110	0.100
4568	20.841	5.897	0.075
4569	20.113	5.772	0.100
4571	18.655	5.598	0.075
4572	17.899	5.448	0.075
4573	17.161	5.298	0.075
4574	16.402	5.147	0.075
4575	15.648	5.022	0.063
4576	14.873	4.860	0.075
4577	14.088	4.685	0.088
4578	13.315	4.585	0.063
4579	12.653	4.485	0.063
4580	12.002	4.385	0.063
4581	11.409	4.272	0.075
4582	10.790	4.198	0.063
4583	10.211	4.160	0.075
4584	9.639	4.022	0.063
4585	9.092	3.947	0.063
4586	8.592	3.860	0.063
4587	8.020	3.772	0.063
4588	7.353	3.735	0.075
4589	6.864	3.635	0.063
4590	6.535	3.560	0.063
4591	6.337	3.535	0.063
4592	6.252	3.497	0.075
4593	6.218	3.510	0.063
4594	6.216	3.522	0.063
4595	6.221	3.497	0.075
4596	6.156	3.485	0.063
4597	6.090	3.472	0.050
4598	6.215	3.472	0.063
4599	6.402	3.497	0.063
4600	6.634	3.535	0.063
4601	6.923	3.535	0.063
4602	7.245	3.548	0.063
4603	7.636	3.610	0.075
4604	8.066	3.647	0.063
4605	8.527	3.760	0.063
4606	9.138	3.972	0.063
4607	9.774	4.097	0.063
4608	10.413	4.185	0.075
4609	11.061	4.260	0.063
4610	11.735	4.298	0.075
4611	12.431	4.535	0.075
4612	13.128	4.635	0.063
4613	13.837	4.635	0.075
4614	14.554	4.748	0.063
4615	15.278	4.835	0.075
4616	15.963	4.848	0.075
4617	16.666	4.948	0.075
4618	17.374	5.060	0.075
4619	18.112	5.185	0.075
4620	18.863	5.285	0.063
4621	19.622	5.372	0.100
4622	20.394	5.473	0.088
4623	21.179	5.760	0.075
4624	21.968	5.760	0.100
4625	22.779	5.960	0.100
4626	23.592	6.073	0.075
4627	24.423	6.085	0.075
4628	25.356	6.260	0.088
4629	26.255	6.385	0.075
4631	28.038	6.698	0.088
4632	28.929	6.785	0.075
4634	30.672	7.010	0.088
4636	32.450	7.348	0.113
4637	33.357	7.473	0.075
4639	35.144	7.723	0.075
4640	35.977	7.910	0.075
4641	36.638	7.935	0.075
4642	37.472	8.110	0.075
4643	38.324	8.160	0.075
4644	39.186	8.385	0.100
4645	40.073	8.497	0.075
4646	40.947	8.473	0.088
4647	41.840	8.660	0.100
4649	43.646	9.110	0.125
4650	44.495	9.198	0.125

Receiver 608 Shot line 1N

Shot	Range	Time	Err
3242	32.885	7.260	0.100
3247	29.321	6.903	0.090
3248	28.575	6.948	0.080
3249	27.799	7.013	0.080
3250	27.024	6.902	0.080
3253	24.981	6.478	0.080
3254	24.209	6.359	0.080
3256	22.810	6.031	0.080
3257	22.079	5.893	0.080
3258	21.313	5.776	0.070
3259	20.692	5.611	0.070
3260	19.930	5.504	0.060
3261	19.211	5.404	0.060
3262	18.440	5.298	0.060
3263	17.634	5.170	0.060
3264	16.885	4.918	0.070
3265	16.151	4.717	0.070
3266	15.380	4.532	0.060
3267	14.679	4.352	0.060
3268	13.940	4.152	0.060
3269	13.195	4.075	0.060
3270	12.502	4.143	0.080
3271	11.909	3.973	0.060
3272	11.202	3.900	0.060
3273	10.612	3.882	0.060
3274	10.154	3.852	0.060
3275	9.701	3.778	0.060
3276	9.382	3.810	0.060
3277	9.240	3.853	0.070
3281	9.141	3.812	0.060
3282	9.474	3.832	0.060
3283	9.737	3.830	0.060
3285	10.596	3.908	0.080
3286	11.033	4.046	0.070
3288	12.227	4.198	0.070
3289	12.971	4.329	0.070
3290	13.547	4.399	0.070
3291	14.219	4.476	0.100
3292	15.069	4.618	0.080
3293	15.785	4.779	0.090
3294	16.518	4.900	0.080
3295	17.353	4.969	0.080
3298	20.014	5.326	0.080
3299	20.996	5.518	0.080
3300	22.030	5.653	0.080
3305	26.570	6.447	0.090
3306	27.470	6.636	0.090
3307	28.523	6.865	0.090
3309	30.497	7.098	0.100
3311	32.735	7.367	0.100
3312	33.808	7.536	0.090
3313	34.534	7.705	0.090
3314	35.858	7.843	0.090
3315	36.938	7.910	0.100
3316	37.670	8.051	0.100
3317	38.264	8.080	0.090
3318	39.071	8.014	0.100
3319	39.564	8.204	0.100
3320	40.484	8.306	0.100
3321	41.088	8.404	0.100
3322	41.801	8.538	0.110
3323	42.517	8.641	0.110
3324	43.021	8.705	0.110
3325	43.741	8.767	0.110
3326	44.464	8.868	0.110
3328	45.914	9.033	0.110
3329	46.643	9.091	0.120
3330	47.373	9.292	0.120

Receiver 608 Shot line 3N

Shot	Range	Time	Err
4526	56.353	10.424	0.080
4527	55.872	10.339	0.080
4528	55.472	10.259	0.080
4530	54.812	10.205	0.080
4531	54.463	10.138	0.080
4532	54.131	10.051	0.080
4533	53.911	9.982	0.100
4534	53.562	9.942	0.080
4535	53.285	9.877	0.080
4536	53.028	9.838	0.100
4537	52.788	9.773	0.080

Receiver 608 Shot line 4N

Shot	Range	Time	Err
4582	57.687	10.542	0.100
4583	57.163	10.028	0.100
4584	56.539	9.919	0.080
4585	55.874	9.875	0.100
4586	55.252	9.977	0.100
4587	54.488	9.866	0.100
4588	53.384	9.949	0.080
4589	52.342	9.811	0.100
4590	51.239	9.679	0.080
4591	50.099	9.603	0.080
4592	49.239	9.448	0.080
4593	48.619	9.441	0.080
4595	47.445	9.284	0.080
4596	46.699	9.198	0.080
4597	46.118	9.028	0.080
4598	45.436	8.961	0.090
4599	44.817	8.873	0.070
4600	44.199	8.817	0.090
4601	43.517	8.696	0.080
4602	42.936	8.550	0.070
4603	42.319	8.470	0.090
4604	41.636	8.376	0.080
4605	41.056	8.396	0.080
4606	40.236	8.270	0.080
4607	39.517	8.180	0.070
4608	38.698	8.103	0.070
4609	37.980	7.990	0.090
4610	37.161	7.928	0.080
4611	36.443	7.759	0.080
4612	35.590	7.615	0.080
4614	34.055	7.362	0.070
4615	33.204	7.185	0.070
4616	32.489	7.065	0.060
4617	31.742	7.003	0.060
4618	30.926	6.883	0.060
4619	30.183	6.762	0.070
4620	29.338	6.769	0.060
4623	27.049	6.425	0.060
4624	26.342	6.272	0.060
4625	25.507	6.156	0.080
4627	23.972	5.949	0.060
4628	23.143	5.890	0.070
4629	22.213	5.819	0.060
4631	20.470	5.367	0.080
4632	19.657	5.429	0.060
4633	18.760	5.226	0.060
4634	17.850	4.993	0.050
4635	17.065	4.859	0.060
4636	16.168	4.812	0.060
4639	13.536	4.715	0.050
4640	12.783	4.570	0.050
4642	11.298	4.235	0.080
4643	10.574	4.043	0.050
4644	9.759	3.930	0.060
4649	5.901	3.649	0.080
4650	5.207	3.543	0.040
4652	4.182	3.518	0.050

Receiver 611 Shot line 1N

Shot	Range	Time	Err
3260	52.169	9.784	0.100
3261	51.391	9.675	0.100
3263	49.609	9.463	0.100
3267	46.044	8.618	0.090
3268	45.155	8.478	0.100
3272	41.371	8.090	0.090
3273	40.481	8.063	0.090
3275	38.588	7.810	0.090
3279	34.915	7.514	0.080
3280	34.025	7.476	0.080
3281	33.247	7.279	0.080
3282	32.138	7.099	0.080
3283	31.361	6.958	0.080
3284	30.251	6.807	0.080
3286	28.590	6.526	0.070
3287	27.486	6.414	0.080
3288	26.823	6.329	0.070
3289	25.718	6.161	0.100
3290	24.946	6.002	0.080
3291	24.065	5.820	0.070
3292	23.081	5.683	0.060
3293	22.203	5.525	0.070
3294	21.326	5.366	0.070
3295	20.340	5.226	0.070
3296	19.356	5.094	0.060
3297	18.388	4.966	0.060
3298	17.411	4.845	0.060

3299	16.328	4.718	0.050
3300	15.122	4.563	0.050
3301	14.375	4.463	0.050
3302	13.284	4.295	0.050
3303	12.113	4.133	0.050
3304	11.142	4.032	0.050
3305	10.313	3.901	0.050
3306	9.357	3.811	0.090
3307	8.345	3.491	0.050
3308	7.465	3.311	0.040
3309	6.472	3.125	0.040
3310	5.494	2.958	0.050
3311	4.716	2.836	0.050
3313	3.733	2.655	0.040
3314	3.658	2.594	0.050
3315	3.895	2.652	0.040
3316	4.163	2.744	0.040
3317	4.399	2.853	0.050
3318	4.871	2.958	0.050
3319	5.195	3.058	0.050
3320	5.921	3.171	0.050
3321	6.428	3.260	0.050
3322	7.064	3.355	0.060
3323	7.727	3.469	0.060
3325	8.901	3.646	0.050
3326	9.613	3.778	0.060
3327	10.335	3.861	0.060
3328	11.066	3.964	0.060
3329	11.803	4.083	0.060
3330	12.547	4.175	0.060

Receiver 611 Shot line 4N

Shot	Range	Time	Err
4613	47.928	9.320	0.100
4614	47.526	9.208	0.100
4615	47.094	9.131	0.100
4617	46.393	8.931	0.100
4618	45.936	8.822	0.090
4619	45.641	8.852	0.090
4620	45.268	8.770	0.090
4621	44.933	8.730	0.080
4622	44.586	8.677	0.100
4623	44.340	8.677	0.100
4624	44.036	8.745	0.110
4625	43.730	8.710	0.120
4626	43.449	8.667	0.120
4629	42.586	8.656	0.100
4631	42.074	8.655	0.120
4632	41.882	8.328	0.100
4633	41.554	8.276	0.090
4634	41.328	8.283	0.120
4635	41.104	8.290	0.120
4636	40.919	8.354	0.100
4637	40.665	8.327	0.100
4638	40.432	8.382	0.100
4639	40.310	8.490	0.120
4640	40.083	8.346	0.100
4641	40.020	8.351	0.120
4642	39.861	8.339	0.120
4643	39.775	8.298	0.100
4644	39.656	8.415	0.100
4646	39.335	8.309	0.100
4647	39.179	8.413	0.100
4649	39.031	8.479	0.080
4650	38.831	8.453	0.100
4652	38.625	8.519	0.100
4654	38.460	8.562	0.100

Receiver 612 Shot line 2N

Shot	Range	Time	Err
3360	15.368	4.985	0.050
3361	16.051	5.023	0.060
3364	18.168	5.476	0.060
3369	21.807	6.291	0.060
3370	22.797	6.395	0.060
3371	23.573	6.447	0.070
3372	24.355	6.677	0.080
3373	25.141	6.756	0.080
3375	26.729	6.946	0.070
3377	28.206	7.185	0.070
3378	29.118	7.307	0.070
3379	29.902	7.480	0.080
3380	30.576	7.777	0.100
3381	31.582	7.672	0.080
3382	32.367	7.751	0.080
3383	33.041	7.817	0.080
3384	34.047	7.941	0.080
3385	34.953	7.990	0.080
3388	37.418	8.323	0.080
3389	38.314	8.396	0.080

3390	38.989	8.423	0.080
3391	39.774	8.512	0.080
3392	40.669	8.615	0.100
3393	41.455	8.803	0.100
3396	44.131	8.987	0.100
3397	44.917	9.057	0.100
3403	49.828	9.872	0.100
3404	50.502	9.939	0.100

Receiver 612 Shot line 3N

Shot	Range	Time	Err
4476	45.253	9.005	0.100
4477	44.633	8.925	0.100
4478	44.019	8.895	0.100
4479	43.497	8.862	0.100
4480	42.892	8.783	0.080
4481	42.309	8.676	0.100
4482	41.715	8.786	0.080
4485	40.051	8.625	0.090
4486	39.481	8.328	0.100
4487	38.919	8.325	0.080
4489	37.895	8.117	0.080
4490	37.356	8.136	0.090
4491	36.821	8.154	0.100
4492	36.296	8.114	0.090
4493	35.781	8.048	0.090
4494	35.346	8.015	0.080
4495	34.933	7.931	0.090
4496	34.516	8.078	0.080
4498	33.488	7.725	0.080
4499	33.097	7.698	0.080
4500	32.653	7.629	0.080
4501	32.222	7.596	0.080
4505	30.740	7.479	0.080
4507	30.027	7.441	0.080
4508	29.741	7.321	0.080
4509	29.424	7.334	0.080
4510	29.123	7.403	0.080
4511	28.880	7.362	0.090
4512	28.613	7.293	0.090
4514	28.168	7.235	0.080
4518	27.570	7.418	0.070
4522	27.069	7.145	0.080
4524	27.002	7.201	0.080
4525	27.239	7.237	0.070
4527	27.833	7.447	0.070
4529	28.663	7.320	0.070
4530	29.071	7.422	0.080
4531	29.510	7.374	0.080
4532	29.977	7.697	0.080
4533	30.574	7.517	0.080
4534	31.130	7.556	0.080
4535	31.677	7.550	0.090
4536	32.247	7.601	0.090
4537	32.839	7.665	0.090
4538	33.063	7.749	0.090

REFERENCES

Aki, K. and P. G. Richards, Quantitative Seismology: Theory and Methods, vol. 2, ch. 12, W. H. Freeman and Co., San Francisco, CA, 1980.

ANS X3.9-1978, Programming Language Fortran 77, American National Standards Institute. 1978.

Bonatti, E., Vertical tectonism in oceanic fracture zones, Earth and Planet. Sci. Lett., 37, 369-379, 1978.

Braile, L. W. and R. B. Smith, Guide to the interpretation of refraction profiles, Geophys. J. R. Astron. Soc., 40, 145-176, 1975.

Choukroune, P., J. Franchetau and X. Le Pichon, In situ structural observations along Transform Fault A in the FAMOUS area, Mid-Atlantic Ridge, Geol. Soc. Am. Bull., 89, 1013-1029, 1978.

Cochran, J. R., Gravity and magnetic investigations in the Guyana basin, western equatorial Atlantic. Geol. Soc. Am. Bull., 84, 3249-3268, 1973.

Collette, B. J., H. Schouten, K. Rutter and A. P. Slootweg. Structure of Mid-Atlantic Ridge province between 12°N and 18°N, Mar. Geophys. Res., 2, 143-179, 1974.

CSPI. Simple Notation for Array Processing. Version II, Reference Manual, CSP Inc., Billerica, MA, 1978.

CYANAMEX scientific team and L. Pastouret, Submersible structural study of the Tamayo transform fault: East Pacific Rise, 23°N (Project RITA). Mar. Geophys. Res., 4, 381-401, 1981.

DeBoor, C. and J. R. Rice, Cubic spline approximation I - fixed knots. Computer Science Department TR20, Purdue University, 1968.

Detrick, R. S., M. H. Cormier, R. A. Prince, D. W. Forsyth and E. L. Ambos, Seismic constraints on the crustal structure within the Vema Fracture Zone. J. Geophys. Res., 87, 10599-10612, 1982.

Detrick, R. S. and G. M. Purdy, The crustal structure of the Kane fracture zone from seismic refraction studies. J. Geophys. Res., 85, 3759-3777, 1980.

Dongarra, J. J., C. B. Moler, J. R. Bunch, and G. W. Stewart, LINPACK Users Guide, SIAM. Philadelphia, PA, 1979.

Dorman, L. M. and R. S. Jacobson. Linear inversion of body wave data - Part 1: Velocity structure from travel times and ranges, Geophysics, 46, 138-151, 1981.

Ewing, J. I. and R. P. Meyer, Rivera Ocean Seismic Experiment (ROSE) overview, J. Geophys. Res., 87, 8354-8357, 1982.

Fleming, H. S., N. Z. Cherkis and J. B. Heirtzler, The Gibbs Fracture Zone: a double fracture zone at 52°30′N in the Atlantic Ocean, Mar. Geophys. Res., 1, 37-45, 1970.

Fox, P. J., A. Lowrie and B. C. Heezen, Oceanographer Fracture Zone, Deep Sea Res., 16, 59-66, 1969.

Fox, P. J., W. C. Pitman III and F. Shepard. Crustal plates in the central Atlantic: Evidence for at least two poles of rotation, Science, 165, 487-489, 1969.

Fox, P. J., E. Schreiber, H. Rowlett and K. McCamy, The geology of the Oceanographer Fracture Zone: A model for fracture zones, J. Geophys. Res., 81, 4117-4128, 1976.

Fuchs, K. and G. Muller, Computation of synthetic seismograms with the reflectivity method and comparison with observations. Geophys. J. Roy. Astron. Soc., 23, 417-433, 1971.

Gettrust, J. F., K. Furukawa and W. C. Kempner, Variations in young oceanic crust and upper mantle structure, J. Geophys. Res., 87, 8435-8445, 1982.

Heezen, B. C., R. D. Gerard and M. Tharp, The Vema Fracture Zone in the equatorial Atlantic, J. Geophys. Res., 69, 733-739, 1964a.

Heezen, B. C., E. T. Bunce, J. B. Hersey and M. Tharp, Chain and Romanche Fracture Zones, Deep Sea Res., 11, 11-33, 1964b.

Hewlett-Packard, HP 2647A Intelligent Graphics. Reference Manual, Hewlett-Packard Co., Cupertino, CA, 1979.

IMSL, Routine ICSFKU, International Mathematical and Statistical Libraries Inc., Houston, TX. 1980.

Jackson, D. D., Interpretation of inaccurate, insufficient and inconsistent data, Geophys. J. R. Astron. Soc., 28, 97-109, 1972.

Johnson, S. H., M. D. Cranford. B. T. Brown, J. E. Bowers and R. E. McAllister, A free-fall direct recording ocean bottom seismograph, Mar. Geophys. Res., 3, 103-117, 1977.

Kastens, K. A., K. C. MacDonald, K. Becker and K. Crane, The Tamayo Fracture Zone at the mouth of the Gulf of California, Mar. Geophys. Res., 4, 129-151, 1979.

Kempner, W. C. and J. F. Gettrust, Ophiolites, synthetic seismograms and oceanic structure. Part II: A comparison of synthetic seismograms of the Samail ophiolite, Oman and the ROSE refraction data from the East Pacific Rise, J. Geophys. Res., 87, 8463-8476, 1982.

Kennett, B. L. N., Theoretical seismogram calculation for laterally varying crustal models, Geophys. J. Roy. Astron. Soc., 42, 579-589, 1975.

Kernighan, B. W. and P. J. Plauger, Software Tools, Addison-Wesley, Reading, MA, 1976.

Koelsch, D. E. and G. M. Purdy, An ocean bottom hydrophone for seismic refraction experiments in the deep ocean, Mar. Geophys. Res., 4, 115-125, 1979.

Lanczos, C., Linear Differential Operators, Chapter 3, Van Nostrand. London. 1961

Latham, G., P. Donoho, K. Griffiths. A. Roberts and A. K. Ibrahim, The Texas ocean bottom seismograph, presented at the Offshore Technology Conference, Soc. of Explor. Geophys., Houston TX. 1978.

Latraille, S. L., J. F. Gettrust and M. E. Simpson, The ROSE seismic data storage and exchange facility, J. Geophys. Res., 87, 8359-8363, 1982.

Lawson, C. L. and R. J. Hanson, Solving Least Squares Problems, Prentice-Hall, Englewood Cliffs, NJ, 1974.

Le Pichon, X., Sea-floor spreading and continental drift, J. Geophys. Res., 73, 3661-3697, 1968.

Lewart, C. R., Algorithm 436, Algorithms SCALE1, SCALE2, and SCALE3 for determination of scales on computer generated plots, Comm. A. C. M., 16, 639-640, 1973.

Lewis, B. T. R. and J. McClain, Converted shear waves as seen by ocean bottom seismometers and surface buoys. Bull. Seis. Soc. Am., 67, 1291-1302, 1977.

Lewis, B. T. R. and. W. E. Snydsman, Fine structure of the lower oceanic crust on the Cocos plate, Tectonophysics. 55, 87-105, 1979

Lynn, W. S. and B. T. R. Lewis, Tectonic evolution of the northern Cocos plate, Geology, 4, 718-722, 1976

Macdonald, K. C., Mid-ocean ridges: fine scale tectonic, volcanic and hydrothermal processes within the plate boundary zone, Ann. Rev. Earth Planet. Sci., 10, 155-190, 1982.

MacDonald, K. C., K. A. Kastens, F. N. Spiess and S. P. Miller, Deep tow survey of the Tamayo Transform Fault, Mar. Geophys. Res., 4, 37-70, 1979.

Mammericx, J., The bathymetry of the Orozco Fracture Zone, unpublished map, Scripps Institution of Oceanography, La Jolla, CA, 1980.

Mammerickx, J. and K. D. Klitgord, East Pacific Rise: Evolution from 25 M.y.B.P. to present, J. Geophys. Res., 87, 6751-6759, 1982.

Matthews, D. H., The Owen Fracture Zone and the northern Carlsberg Ridge, Phil. Trans. Roy. Soc. London, 252, 172-186, 1966.

McKenzie, D. P. and R. L. Parker, The North Pacific: An example of tectonics on a sphere, Nature, 216, 1276-1280, 1967.

Menard, H. W., Deformation of the northeastern Pacific basin and the west coast of North America, Geol. Soc. Am. Bull., 66, 1149-????. 1955.

Menard, H. W., The East Pacific Rise, Science, 132, 1737-1746, 1960.

Menard, H. W., Fracture zones and offsets of the East Pacific Rise, J. Geophys. Res., 71, 682-685, 1966.

Menard, H. W. and R. S. Dietz, Mendocino submarine escarpment, J. Geol., 60, 266-278, 1952

Menard, H. W. and R. L. Fisher, Clipperton fracture zone in the northeast equatorial Pacific. J. Geol., 66, 239-253, 1958.

Minster, J. B. and T. H. Jordan, Present-day plate motions, J. Geophys. Res., 83, 5331-5354, 1978.

Morgan, W. J., Rises, trenches, great faults and crustal blocks, J. Geophys. Res., 73, 1959-1982, 1968.

Morris, G. B., Delay-time-function method and its application to the Lake Superior refraction data, J. Geophys. Res., 77, 297-314, 1972.

Olivet, J. L., X. Le Pichon, S. Monti and B. Sichler, Charlie Gibbs Fracture Zone. J. Geophys. Res., 79, 2059-2072, 1974.

Orcutt, J. A., B. L. N. Kennett and L. M. Dorman, Structure of the East Pacific Rise from an ocean bottom seismometer survey, Geophys. J. R. Astron. Soc., 45, 205-220, 1976.

Pitman, W. C. III, E. M. Herron and J. R. Heirtzler, Magnetic anomalies in the Pacific and sea floor spreading. J. Geophys. Res., 73, 2069-2085, 1968.

Project ROSE Scientists, Microearthquake activity on the Orozco Fracture Zone: Preliminary report from project ROSE, J. Geophys. Res., 86, 3783-3790, 1981.

Purdy, G. M., The correction for the travel time effects of the seafloor topography in the interpretation of marine seismic data, J. Geophys. Res., 87, 8389-8396, 1982.

Rao, C. R., Linear Statistical Inference and its Applications, Chapter 4, Wiley, New York, NY, 1973.

Robb, J. M. and M. F. Kane, Structure of the Vema Fracture Zone from gravity and magnetic intensity profiles. J. Geophys. Res., 80, 4441-4445, 1975.

Schouten, H. and R. S. White Zero-offset fracture zones, Geology, 8, 175-179, 1980.

Sibuet. J. C., and B. Veyrat-Peinet, Gravimetric model of the Atlantic equatorial fracture zones. J. Geophys. Res., 85, 943-954, 1980.

Sinton, J. B. and D. M. Hussong. Ray tracing in laterally inhomogeneous media, Indiana Geol. Survey Occ. Paper, Geophys. Comp. Program, accepted for publication, 1981.

Sinton, J. B. and D. M. Hussong. Crustal structure of a short length transform fault in the central Mariana Trough, Geophys. Monogr. Ser., 27, Amer. Geophys. Union, 1983.

Sleep. N. H., Formation of oceanic crust: Some thermal constraints, J. Geophys. Res., 80, 4037-4042, 1975

Sleep. N. H. and S. Biehler, Topography and tectonics at the intersection of fracture zones with central rifts, J. Geophys. Res., 75, 2748-2752, 1970.

Sutton, G. H., J. Kasahara, W. N. Ichinose and D. A. Byrne, Ocean bottom seismograph development at Hawaii Institute of Geophysics, Mar. Geophys. Res., 3, 153-177. 1977.

Sutton, G. H., F. K. Duennebier and B. Iwatake, Coupling of ocean bottom seismometers to soft bottom, Mar. Geophys. Res., 5, 35-51, 1981.

Sykes, L. R., Mechanism of earthquakes and nature of faulting on the mid-ocean ridges, J. Geophys. Res., 72, 2131-2153, 1967.

Trehu, A. M., Seismicity and structure of the Orozco Fracture Zone from ocean bottom seismic observations, Ph.D. Thesis, Massachusetts Institute of Technology/Woods Hole Oceanographic Instiution, WHOI 82-13, 1982.

Trehu, A. M. and G. M. Purdy, Crustal structure in the Orozco Fracture Zone, J. Geophys. Res., submitted 1983.

Trehu, A. M. and S. C. Solomon, Earthquakes and the Orozco Fracture Zone: Seismicity, source mechanism and tectonics, J. Geophys. Res., submitted 1983.

Vacquier, V., Measurement of horizontal displacement along faults in the ocean floor, Nature, 183, 452-453, 1959.

van Andel, Tj. H., R. P. Von Herzen and J. D. Phillips, The Vema Fracture Zone and the tectonics of transverse shear zones in oceanic crustal plates, Mar. Geophys. Res., 1, 261-283, 1971.

Versatec, Versaplot-07, Graphics Programming Manual, edition 2, Versatec Inc., Santa Clara, CA, 1978.

Whitmarsh, R. B., Axial intrusion zone beneath the median valley of the Mid-Atlantic Ridge at 37°N detected by explosion seismology, Geophys. J. R. Astron. Soc., 42, 189-215, 1975.

Wiggins, R. A., The general linear inverse problem: Implications of surface waves and free oscillations for Earth structure, Rev. Geophys. and Space Phys., 10, 251-285, 1972

Willmore, P. L. and A. M. Bancroft, A time term approach to refraction seismology, Geophys. J. R. Astron. Soc., 3, 419-432, 1960

Wilson, J. T., A new class of faults and their bearing on continental drift, Nature, 207, 907-910, 1965.