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

[Signatures]
ACKNOWLEDGMENTS

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

In addition to the members of my thesis committee, 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 Bracher 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 raytracing 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.
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

ACKNOWLEDGMENTS ......................................................................................... iii

ABSTRACT ........................................................................................................ iv

LIST OF FIGURES .............................................................................................. viii

CHAPTER I. INTRODUCTION ........................................................................... 1

1.1 Historical Background .................................................................................. 1

1.2 The ROSE project ....................................................................................... 7

CHAPTER II. PHYSICAL SETTING ................................................................... 11

CHAPTER III. DATA SET ................................................................................... 21

CHAPTER IV. INVERSION METHODS .............................................................. 49

4.1 Reparametrization of Travel Times ............................................................. 51

4.2 Inverse problem for the Velocity-depth Function ..................................... 58

4.3 Time Term Analysis .................................................................................... 61

4.4 Verification by Forward Modeling .............................................................. 65

CHAPTER V. INTERPRETATION ................................................................... 67

5.1 Line 1N ........................................................................................................ 68

5.2 Line 2N ........................................................................................................ 77

5.3 Line 3N ........................................................................................................ 85

5.4 Line 4N ........................................................................................................ 91

CHAPTER VI. SUMMARY AND DISCUSSION ............................................... 101

APPENDIX A: COMPUTER PROGRAMS ......................................................... 106

A.1 Program TPICK .......................................................................................... 110

A.2 Program TXCOR ......................................................................................... 119

A.3 Program TXFIT .......................................................................................... 123

vi
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.4 Program VEL0</td>
<td>129</td>
</tr>
<tr>
<td>A.5 Program LSVZ</td>
<td>134</td>
</tr>
<tr>
<td>A.6 Data transmission subroutines</td>
<td>152</td>
</tr>
<tr>
<td>A.7 Text processing subroutines</td>
<td>163</td>
</tr>
<tr>
<td>A.8 Mathematical function subroutines</td>
<td>185</td>
</tr>
<tr>
<td>A.9 Plotting Subroutines</td>
<td>205</td>
</tr>
<tr>
<td>APPENDIX B: FIRST ARRIVAL PICK TRAVEL TIMES</td>
<td>235</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>249</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1. Structure of a generalized fracture zone system. 3
Figure 2. Bathymetric features and magnetic anomaly pattern for the equatorial East Pacific Rise. 13
Figure 3. Tectonic history and major reorganizations of the equatorial East Pacific Rise during the last 25 M.y. 15
Figure 4. Bathymetry of the Orozco Fracture Zone. 18
Figure 5. Major bathymetric features of the Orozco Fracture Zone. 20
Figure 6. Locations of the receivers used for each of the shot lines. 24
Figure 7. Record sections of the seismic data for each instrument. 26
Figure 8. Spline fit of a sample of the travel time data showing the knot locations and the optimal spacing values. 56
Figure 9. Least squares inversion results for a sample data set. 63
Figure 10. Interpreted record sections for the receivers on line 1N. 70
Figure 11. Time term values for the shots along line 1N. 74
Figure 12. Velocity-depth model for the profile of line 1N. 76
Figure 13. Interpreted record sections for the receivers on line 2N. 80
Figure 14. Time term values for the shots along line 2N. 82
Figure 15. Velocity-depth model for the profile of line 2N. 84
Figure 16. Interpreted record sections for the receivers on line 3N. 88
Figure 17. Time term values for the shots along line 3N. 90
Figure 18. Velocity-depth model for the profile of line 3N. 93

viii
Figure 19. Interpreted record sections for the receivers on line 4N. 96

Figure 20. Time term values for the shots along line 4N. 98

Figure 21. Velocity-depth model for the profile of line 4N. 100
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.
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 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 across 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 seismicity 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 Oceanográfico 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 across 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 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 Mummerickx, 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. Mummerickx 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 note 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).
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°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.
Figure 5. Main bathymetric features in the Orozco Fracture Zone region.
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.
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).
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 (Figure 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.
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,

\[ \Delta t = \Delta z \left( \frac{(1 - \frac{v_w^2 p^2}{2})^{1/2}}{v_w} - \frac{(1 - \frac{v_s^2 p^2}{2})^{1/2}}{v_s} \right); \]
where $\Delta t$ is the time correction; $\Delta z$ 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 $\zeta$ and $\tau$ for the $j$-th observation may be written in the form

$$
\sum_{j} \left( \frac{dz}{dv} \right)_j f(v_i, p_j)
$$

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

$$
\mathbf{y} = A \mathbf{x};
$$

where $\mathbf{y} = [\zeta(p_1), \tau(p_1), \zeta(p_2), \tau(p_2), \ldots]^T$ (m x 1);
and $A$ is a $m \times n$ matrix with coefficients $f(v_i, p_j)$.

Typically, each pair of $\zeta$, $\tau$ 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 $\zeta$ and $\tau$ 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 \( r_i = f_i - s(x_i); \)
\[ w_i = \frac{(x_{i+1} - x_{i-1})}{(x_n - x_1)}; \]
\( x_i, f_i \) 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.
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 observation 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 $\varsigma$ and $\tau$, 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 $m \times n$ matrix is

$$A = U S V^T$$

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

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

In practice, the term $S^{-1}$ 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 partitioned as

$$
U = [U_r | U_o] \quad ; \quad S = \begin{bmatrix} S_r & 0 \\ 0 & 0 \end{bmatrix} \quad ; \quad V = [V_r | V_o];
$$

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

$$
x = V_r S_r^{-1} U_r^T y.
$$

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 $U$, $S$, and $V$ provide useful information. The product $E = U_r U_r^T$ has been called the data information density (Wiggins, 1972). It is the orthogonal projection matrix onto the column space of $A$. The estimate of $Ax$ is given by $Ex$. If the matrix $A$ is square and of full rank, then all the singular values can be used and $r = m$ and $E = I$, the $m \times m$ identity matrix. In this case $Ax$ is equivalent to $y$. In the present case, the system of equations is overdetermined so $r < m$. There exists a component $U_o$ of $U$ and the solution is no longer exact. The columns of $E$ are now a weighted average applied to the components of $y$. The
projection onto the orthogonal complement of the column space of $A$ is
given by $U^T U^T$. The residual $y - Ax$ is therefore given by $U^T U^T v$.

The matrix $R = V^T V^T$ is called the resolution matrix of the model
parameters. The rows of $R$ are the averaging coefficients used to
calculate the model. If $r = n$ and there is no $V_o$ space, then again
$V^T V^T = I$, the $n \times n$ 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

$$\langle \Delta x, \Delta x \rangle = \sigma^2 V^T S^{-2} V^T$$

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 equation 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 \( \frac{dz}{dv} \) 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

VELOCITY (KM S\(^{-1}\))

B) 0 1 2 3 4 5 6 7 8

DV/DZ (S\(^{-1}\))

C) 0.0 0.2 0.4 0.6 0.8 1.0

DEPTH (KMI)

0 1 2 3 4 5

63
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; \( \delta_s \) is the time required to travel between the source and the refractor; and \( \delta_r \) 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 accommodated 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 reflected 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 LN. 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.
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 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$^{-1}$ between 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 lN. 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 lN 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 LN. 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.
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.
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.
Figure 14. Time terms for shots on line 2N. See comments for Figure 11. Range is relative to the position of OBS 210.
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.
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 show 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 $4.0 \text{ km s}^{-1}$. 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 a model with a lower gradient in the upper crust and a sharper transition to the $7.2 \text{ km s}^{-1}$ 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.
Figure 17. Time terms for shots along line 3N. See comments for Figure 11. Ranges are relative to the position of OBS 526.
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 introduced 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.
transform valley and the central ridge (Figures 4 and 5). Receivers providing data were WHOI OBH 4, MSI OBS 210, HIC 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 for 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.
Figure 20. Time terms for shots along line 4N. See comments for Figure 11. Ranges are relative to the position of OBH 4.
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.
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 different 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 successfully by the velocity structure for the Samail (Oman) ophiolite [Manghnani et al., 1981; Kempner and Gettrust, 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°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 occurring 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.
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 Institute 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 $\zeta$ and $\tau$ 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 [outfile] [/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.
Program TP1K

************************************************************************

Plot shot records from standard HIG demux files on HP-2647A terminal so arrival times can be picked

************************************************************************

usage: TIMPICK datafile corfile outfile [/FSW]

datafile: HIG demultiplexed OBS data from XDECOM or XRHIG
corfile: time and range corrections from XDECOM or XRHIG
outfile: shot numbers, ranges and time picks with error

options: F filter data before displaying
S save specified number of seconds before and after arrival in separate file
W pick water wave arrivals

************************************************************************

source: 1St2REF*A.PIK
libraries: 1St2ROSE*PLOTLB, 1St2ROSE*CHARLB, 1St2ROSE*MATHLB
1St2REF*WLIB, 1SOOMGG*MRSLIB, *SAUL77, *LIBERY
executable: 1St2REF*TIMPICK

1.0 2/6/82 no options implemented
1.1 2/16/82 filter, save and water arrival options added
1.2 4/6/83 change to new i/o subroutines
1.3 5/1/83 change filter, convert to data to float for processing

************************************************************************

INTEGER DD, CD, OD, SD, HD, POPT, SOPT, WOPT
COMMON /FILES/ DD, CD, OD, SD, HD, POPT, SOPT, WOPT
INTEGER HBUF(112), SHOT, HDRADR, NSAMP
REAL AFTSEC, FORSEC, RV
INTEGER LOCDHD, PROPIK
CALL OPFPIK(DD,CD,OD,SD,HD,POPT,SOPT,WOPT)
HDRADR = LOCDHD(DD,HBUF,NSAMP)
IF (SOPT .EQ. 1) THEN
  CALL REMARK(" enter number of seconds before and after arrival to save")
  READ (0,) FORSEC, AFTSEC
END IF
IF (POPT .EQ. 1) THEN
  CALL REMARK(" enter low cut-off freq, high cut-off freq, and number of poles")
  READ (0,) FL, F2, NPOLES
  CALL GFLTTM(FL,F2,NPOLES,GAIN,DD,HDRADR)
END IF
IF (WOPT .EQ. 1) THEN
  CALL REMARK(" enter water velocity")
  READ (0,) RV
ELSE
  CALL REMARK(" enter reducing velocity or 0")
  READ (0,) RV
END IF
NSAMP = MINO(NSAMP,10000)
CALL INIPLT
CALL ENDFIG(O)
DO
  CALL REMARK(" ahot number?")
  READ (0,) SHOT
UNTIL (PROPIK(SHOT,HDRADR,NSAMP,RV,FORSEC,AFTSEC,FL,F2,NPOLES, & GAIN) .NE.-2)
IF (SOPT .EQ. 1) THEN
  CALL ENDOMX(DD,HD,HBUF,NSAMP)
ELSE
  CALL ENDOMX(DD,HD,HBUF,NSAMP)
END IF
CALL ENDPILT

113
SUBROUTINE OPFPIK(DD, CD, OD, SD, HD, FOPT, SOPT, WOPT)

INTEGER DD, CD, OD, SD, HD, FOPT, SOPT, WOPT
INTEGER ARG(18), LIN(18), TNAME(8), MBUF(112),
       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]")
ENDIF
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]")
ENDIF
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
ENDIF
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)
   ED = FILCR(TNAME)
   IF (ED .EQ. -3) CALL CANT(TNAME)
ENDIF
CALL INIDMX(SD,MBUF)

DO 53: UNTIL (GETLIN(LIN,OD) .EQ. -1)

CALL SETTAB(TABLIST)
RETURN
END
INTEGER CH, RCV, SMPRATE, ADR, SAMPLENO, ARRIV(3), UNC(3), INDEX(3)
INTEGER*6 DST, SBT
REAL RANGE, TIME(3), DELTIME(3), SIGNAL(10000)
REAL RDEP, SDEP, SLANTRANGE
CHARACTER INTERNAL*420 ! for corfile decoding
EQUIVALENCE (DATABUF, INTERNAL)
EQUIVALENCE (FILTBUF, SIGNAL)
INTEGER GETCOR, GETDHD, FILBUF, STSAMP
IF (SHOT .LT. 0) THEN
  CALL REMARK(" end processing")
  PROPIK = -3
  RETURN
END IF
IF (GETCOR(0, SHOT, DATABUF, CD) .EQ. -3) THEN
  CALL REMARK(" shot not found in corfile")
  PROPIK = -2
  RETURN
END IF
IF (GETDBD(DD, HDRADR, SHOT, RCV, CH, SBT, DST, SMPRATE, ADR) .EQ. -3)
  CALL REMARK(" shot not found in data file")
  PROPIK = -3
  RETURN
END IF
IF (FILBUF(DD, DATABUF, NSAMP, ADR) .EQ. -3) THEN
  CALL REMARK(" error during data transfer")
  PROPIK = -3
  RETURN
END IF
IF (WOPT .EQ. 1) THEN
  SLANTRANGE = SQRT(RANGE**2 + (RDEP - SDEP)**2)
  SAMPLENO = STSAMP(DST, SBT, -1.0, SLANTRANGE, RV, SMPRATE)
ELSE
  SAMPLENO = STSAMP(DST, SBT, 0.0, RANGE, RV, SMPRATE)
END IF
IF (SAMPLENO .EQ. '77777777') THEN
  CALL REMARK(" overflow in stsamp ; bad start time in data file")
  PROPIK = -2
  RETURN
ELSE
  SAMPLENO = MAXO(SAMPLENO, 1)
ENDIF
FAIL = VFLOAT(SIGNAL, 1, DATABUF, 1, NSAMP)
IF (FOP .EQ. 1) THEN
  FAIL = FLTTIM(SIGNAL, F1, F2, NPOLES, GAIN, NSAMP)
  CALL SHOPIK(SIGNAL, SAMPLENO, NSAMP, SBT, DST, SMPRATE, ARRIV, UN,
  INDEX, F1, F2, NPOLES)
ELSE
  CALL SHOPIR(SIGNAL, SAMPLENO, NSAMP, SBT, DST, SMPRATE, ARRIV, UN,
  INDEX, F1, F2, NPOLES)
ENDIF
FOR I = 1, 3
  TIME(I) = TIME(1) = FLOAT(ARRIV(I)) / FLOAT(SMPRATE) - FLOAT(SBT-DST) / 1000.0
ENDIF
END FOR
 IF (INDEX(1) .NE. 0 .OR. INDEX(2) .NE. 0 .OR. INDEX(3) .NE. 0)
  & CALL WIPIK(SHOT, RANGE, TIME, DELTIME, INDEX, RCV, OD)
  IF (SOPT .EQ. 1)
    & CALL COMPRS(DATABUF, SHOT, RCV, CH, SBT, DST, SMPRATE, ARRIV,
    & FS, AS, SD, HD)
  PROPIK = -2
  RETURN
END
SUBROUTINE SHOPIK(SIGNAL, SN, N, SHOT, SB, DS, RATE, ARRIV, UNC, INDEX, F1, F2, NPOLES)
  * plot data on BP 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, RD, FOPT, SOPT, WOPT
COMMON /FILES/ DD, CD, OD, SD, RD, FOPT, SOPT, WOPT
INTEGER KEY, NX, SMP, ST
REAL SMAX, SHIN, SMEAN, SCALE, X, XD(20), XINC

CALL SIGPAR(SIGNAL,N,SMAX,SHIN,SMEAN)
SCALE = 2.0 / (SMAX - SHIN)
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) = AMIN1CFLOAT(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 PVECI(SIGNAL(ST),720,0.0,3.9,SHEAN,SCALE,XINC,0.0)
CALL W2PIK1(SHOT,DS,SB,ST,RATE,SHEAN)
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 = MAXO(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(I) = 1
CALL ENDFIG(0)
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
CALL W2PIK4
SUBROUTINE W2PIK1(SHOT, DS, SB, ST, RATE, SMEAN)

* display trace information

INTEGER SHOT, ST, RATE
INTEGER*6 DS, SB
REAL SMEAN

CONST

CALL BOX(1.0, 1.5, 3.3, 1.0, 0.0)
CALL GRSTR(1.1, 2.2, "Shot", 5, 1.0, 0.0)
CALL GRSTR(1.1, 2.0, "Relative start time ", 5, 1.0, 0.0)
RELTIME = REAL(DS-DS) / 1000.0 + FLOAT(ST-1) / FLOAT(RATE)
RELTIME = REAL(DS-DS) / 1000.0 + FLOAT(ST-1) / FLOAT(RATE)

CALL GRNUM(2.2, 2.2, FLOAT(SHOT), 5, -1.0, 0.0)
CALL GRSTR(1.1, 2.0, "Sample number", 13, 0.1, 0.0)
CALL GRNUM(3.2, 1.8, SMEAN, 8, -1.0, 0.0)

RETURN

SUBROUTINE W2PIK2(F1, F2, NPOLES)

* display filter parameters

INTEGER NPOLES
REAL F1, F2
REAL FALLOFF

FALLOFF = 6.0 * NPOLES

CONST

CALL BOX(1.0, 0.5, 3.3, 1.0, 0.0)
CALL GRSTR(1.1, 1.0, "Low frequency", 13, 0.1, 0.0)
CALL GRNUM(3.0, 1.0, F1, 4, -1.0, 0.0)
CALL GRSTR(1.1, 0.8, "High frequency ", 14, 0.1, 0.0)
CALL GRNUM(3.0, 0.8, F2, -1.0, 0.0)
CALL GRSTR(1.1, 0.6, "Fall-off", 8, 0.1, 0.0)
CALL GRNUM(2.8, 0.6, FALLOFF, 4, -1.0, 0.0)

RETURN

SUBROUTINE W2PIK3(WOPT)

* display pick options

INTEGER WOPT

CONST

CALL SETDRM(1)
CALL FILREC(5.05, 0.85, 9.45, 2.45, 1)
CALL BOX(5.0, 0.8, 4.5, 1.7, 0.0)
CALL GRSTR(5.1, 2.3, "Options ", 8, 0.1, 0.0)
CALL GRSTR(5.3, 2.1, "F - single pick", 16, 0.1, 0.0)
IF (WOPT .EQ. 1) THEN
    CALL GRSTR(5.3, 1.9, "D - direct wave", 16, 0.1, 0.0)
    CALL GRSTR(5.3, 1.7, "1 - first reflection", 21, 0.1, 0.0)
    CALL GRSTR(5.3, 1.5, "2 - second reflection", 22, 0.1, 0.0)
ELSE
    RETURN
END

END
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:    RETURN
29:    END
30:    
1:    *-h- W2PIK4 254 12 APR 83 0:38:05
2:    SUBROUTINE W2PIK4
3:    4: * prompt for pick error estimate
5:    6: :HCONST
7:    8: 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
15:  
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:    8: REAL X, Y
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:   RETURN
15:   END
16:  
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 FILCLS(DD, "KEEP")
9:    CALL FILCLS(CD, "KEEP")
10:   CALL FILCLS(OD, "KEEP")
11:   RETURN
12:   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 suppress 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.
***-b- CORMAIN 4095 4 JUN 83 23:16:10
PROGRAM TXCOR

***********************************************************************

Format arrival time information from picker for input to curve fitters

******************************************************'***********'****

usage: TXCOR [.ANW] pickfile corfile outfile [/ANW]
pickfile: arrival times. output from TIMPICK
corfile: shot. receiver corrections file
outfile: output file. corrected times
options: A process all shots
N no sorting by range
W calculate delay in water wave (wv=1.5)

for input to XRMOD

INTEFR.SBOT(200). TAG(200)
REAL(200). IX. T(200.3). DT(200.3)
INTEGER ID. CD. OD. AOPT. NOPT. WOPT
INTEGER REG. FIN. JOB. N. OPT. LDD. PREVSHOT
REAL. SDEP. RDEP. BTC
REAL. STOPO. STOST. STOSX. RTOPO. RTOST. RTOSX. SRTODT. SRTODX
CHARACTER*420 INTERNAL
CHARACTER*100 lOQUAT. lENCE
INTEGER. lPIK. GETCOR
DATA LDD / 200/
CALC OFFCOR(ID. CD. OD. AOPT. NOPT. WOPT)
JOB = 11000
IF (WOPT .EQ. 1) JOB = 11110
IF (AOPT .EQ. 1)
  . BFC = 0
  . FIN = 9999
ELSE
  CALL REMARK(" enter begin shot. end shot")
  READ (0.) REG. FIN
ENDIF
N = lPIK(SHOT. X. T. DT. LDD. JUNK. BEG. FTN. JOB. 200. ID)

phase l. Apply range and time corrections
IF (WOPT . EQ. 0)
  CALL REMARK(" shot topo. rcvr topo. move to datum. move to surf?
&. (4 digit binary)")
  READ(0.) OPT
  . SNC = OPT / 1000. GT. 0
  . BNC = MOD(OPT/100.10). GT. 0
  . MTD = MOD(OPT/10.10). GT. 0
  . MTS = MOD(OPT.10). GT. 0
  . IF (MTD .AND. MTS) CALL ERROR(" cannot move to datum and to surfac
&. e at the same time")
ENDIF
IF (GETCOR(SBOT(I).5.3) .NE. -2)
  CALL PUTDEC(SBOT(I).5.3)
  CALL ERROR(" can't find shot in corfile")
ELSE
  READ (INTERNAL."(T1.10.0.T01.7F10.0.T131.7F10.0.T371.2F10.0)"
&. . 
  SDEP. RDEP. BTC. STOPO. STOST. STOSX. RTOPO. RTOST. RTOSX.
&. 
  SRTODT. SRTODX
ENDIF
PREVSHOT = 0
FOR I = 1.N
  . IF (SHOT(I) . EQ. PREVSHOT)
    . CALL PUTDEC(SHOT(I).5.3)
    . CALL REMARK(" duplicate shot")
    . END IF
  . IF (GETCOR(0.SHOT(I).TAG CD) .NE. -2)
    . CALL PUTDEC(SHOT(I).5.3)
    . CALL ERROR(" can't find shot in corfile")
  . ELSE
    . READ (INTERNAL."(T71.10.0.T101.7F10.0.T131.7F10.0.T371.2F10.0)"
&. 
    . SDEP. RDEP. BTC. STOPO. STOST. STOSX. RTOPO. RTOST. RTOSX.
&. 
    . SRTODT. SRTODX
  . END IF
ENDIF
IF (WOPT . EQ. 1)
calculate delay in water wave arrival with respect to time
reduce by slant range over water velocity

\[
S_X = \text{SORT}(X(I)^2 + (RDEP - SDEP)^2)
\]

\[
T(I,I) = T(I,1) - S_X / 1.51
\]

ELSE

apply corrections

\[
XCOR = 0.0
\]

\[
TCOR = BTC
\]

IF (SRC) TCOR = TCOR + STopo

IF (RRC) TCOR = TCOR + KTOPO

IF (MTD)

\[
TCOR = TCOR + SRTODT
\]

IF (X)

\[
XCOR = XCOR + SRTODX
\]

END IF

IF (MTS)

\[
TCOR = TCOR + STOST
\]

\[
XCOR = XCOR + STOSX
\]

END IF

X(I) = X(I) + XCOR

DX = 0.5

\[
T(I,I) = T(I,1) + TCOR
\]

END IF

PREVSHOT = SHOT(I)

END FOR

phase 2. Sort by range

IF (NOPT .EO. 0)

CALL VINDEX(TAG.1,1.1.1.0.0)

CALL SRTXY(X.TAG.N)

CALL SRTFY(SHOT.TAG.O)

CALL SRTFY(T(1,1).TAG.0)

CALL SRTFY(DT(1,1).TAG.0)

END IF

output

FOR I = 1, N

IF (WOPT .EO. 1)

CALL W2COR(SHOT(I).T(I,1).LDD.3.0)

ELSE

CALL W2COR(SHOT(I).X(I).DX.T(I,1).DT(I,1).OD)

END IF

END FOR

CALL CLPCOR(ID,CD,OD)

END

#-b- OFFCOR 925 4 MAY 83 23:49:10
SUBROUTINE OFFCOR(ID, CD, OD, AOPT, NOPT, WOPT)
open files

INTEGER ID, CD, OD, AOPT, NOPT, WOPT
INTEGER NAME(18)
INTEGER TRC, FILOC, GETARG, GETOPT

CALL PRIMO(11)

IF (GETARG(1).NAME.17) .EO. -1)

CALL ERROR(" usage: TXCOR pickfile corfile outfile [/ANW]")

ELSE

ID = FILOC(NAME)

IF (ID .EO. -3) CALL CANT(NAME)

END IF

IF (GETARG(2).NAME.17) .EO. -1)

CALL ERROR(" usage: TXCOR pickfile corfile outfile [/ANW]")

ELSE

CD = FILOC(NAME)

IF (CD .EO. -3) CALL CANT(NAME)

END IF
25: IF (GETARG(3, NAME, 17) .EQ. -1)
26:   CALL ERROR(" usage: TXCOR pickfile corfile outfile [/ANW]")
27: ELSE
28:   OD = FILOP(NAME)
29:   IF (OD .EQ. -3)
30:     OD = FILCR(NAME)
31:     IF (OD .EQ. -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: *-b- 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 0 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 with 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 \( \zeta(p) = T + px \) (procedure REPARM). Error bounds are also calculated using the formulae given in Dorman and Jacobson [1981] (procedure ERBBND).

The parameter \( p, \zeta(p), \tau(p), \sigma_\zeta, \) and \( \sigma_\tau \) 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 VELO 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 observation. This file is used by program VELO and others not included here.
**TXFIT**

Reparametrization of travel time data by cubic splines

Usage:

```plaintext
TFIT infile outfile [liatfile] [/O]
```

**Description:**
- `infile`: times, ranges output from TXCOR
- `outfile`: tau, zeta parameters input to LSVZ
- `liatfile`: list of fit parameters, input to TXPLOT, VELO
- `options`: 0 - output parameters at optimal points

**Source:**

- `1512REF*A.FIT`
- `1512ROSE*CHARLB`, `1512ROSE*PLOTLB`, `*IMSL`, `*LIBERY`
- `1512REF*TXFIT`

**Variables:**

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

**Source Code:**

```plaintext
CALL OPFFIT(ID, OD, LD, OOPT)
N = RICOR(SHOT, X, DX, T, DT, 200, ID)
CALL REMARK(" enter reducing velocity")
READ (0, ) RV
CALL INIPLT
CALL MOVE0(1.0, 0.5)
CALL PXIT(X, T, RV, N, XSCALE, TMIN, TSCALE)
LOOP
NP = GETKNT(X, MAXK)
EXIT LOOP IF (NP .EQ. 0)
NK = NP
CALL FIT(X, T, NK, Y, C, LDC, E, S, WORK)
CALL REGETR(1)
CALL PXITSM(X, S, RV, N, XSCALE, TMIN, TSCALE)
CALL W2FIT1(K, Y, C, LDC, E, S, WORK)
END LOOP
CALL REPARCH(SHOT, X, S, NK, Y, C, LDC, NK, OK, F, DP, DX, DT, OOPT)
CALL WFIT(F, X, Y, Z, T, Z, DT, N, OD)
IF (OOPT .EQ. 0) CALL W2FIT2(SHOT, X, T, S, P, OX, N, LD)
CALL ENDPILT
CALL CLFFIT(ID, OD, LD)
END
```
ID = FILOP(FILE)

IF (ID .EQ. -3) CALL CANT(FILE)
ELSE
  CALL ERROR(" usage: TXFIT infile outfile [listfile]")
END IF

IF (GETARG(2,FILE,18) .NE. -1)
  OD • FILOP(FILE)
ENDIF
IF (OD .EQ. -3)
  OD • FILCR(FILE)
ENDIF
ELSE
  OD = 6
ENDIF

OOPT • GETOPT(79)
REnJRN
END

INTEGER FUNCTION GETKNT(K, MAXK)
   *
   query for knot positions
   *
   INTEGER MAXK
   REAL K(1)
   INTEGER I, J, L, LIN(81), N, WRD(10)
   INTEGER GETLIN, GETWRD
   REAL CTOF
   CALL SETDRM(4)
   CALL GRSTR(0.4,0.15,"& 43,0.2,0.0)
   CALL GRSTR(0.4,0.3,"Enter
   "
   L • GETLIN(LIN,0)
   I • 1
   N • 0
   WHILE (GETWRD(LIN,I,WRD) .NE. 0 .AND. N .LT. MAXK)
   N • N + 1
   J • 1
   K(N) • CTOF(WRD,J)
   END WHILE
GETKNT • N
CALL SETDRM(2)
CALL ENDFIG(O)
:NO HCONST
REnJRN
END

SUBROUTINE FIT(X, F, NX, K, NK, Y, C, LDC, RHO, S, WK)
   *
   fit cubic spline to data points using fixed knots
   *
   (calls IMSL routines ICSFKU, ICSEVU)
   *
   INTEGER NX, NK, LDC
   REAL X(1), F(1), K(1), Y(1), C(LDC,1), S(1), WK(1), RHO
   REAL D
   INTEGER IMSLERR
   CALL ICSFKU(X,F,NX,O,K,NK,Y,C,LDC,RHO,WK,IMSLERR)
   CALL ICSEVU(K,Y,NK,C,LDC,X,S,NX,IMSLERR)
   RETURN
END

*-- FIT 603 & MAY 83 23:57:40
SUBROUTINE FIT(X, F, NX, K, NK, Y, C, LDC, RHO, S, WK)
* fit cubic spline to data points using fixed knots
* (calls IMSL routines ICSFKU, ICSEVU)
INTEGER NX, NK, LDC
REAL X(1), F(1), K(1), Y(1), C(LDC,1), S(1), WK(1), RHO
REAL D
INTEGER IMSLERR
CALL ICSFKU(X,F,NX,K,NK,Y,C,LDC,RHO,WK,IMSLERR)
CALL ICSEVU(K,Y,NK,C,LDC,X,S,NX,IMSLERR)
SUBROUTINE REPAllM(X, T, NX, K, Y, C, LDC, NK, OX, P, DP, DZ, DT, OOPT)

reparametrize T(X) data to zeta(P) and tau(P), output

oopt s 0 all data points
oopt • 1 data points separated

calls IMSL routine DCSEVU

INTEGER NX, LDC, NK, OOPT
REAL X(1), T(1), K(1), Y(1),
REAL DP(1), DZ(1), DT(1)
INTEGER I, IMSLERR, N
REAL C(LDC,1), OX(1), P(1)
REAL D

IF (OOPT .EQ. 1)
N • NK
CALL VCOPY(X(2),1,K(2),1,N)
CALL VCOPY(T(2),1,Y(2),1,N)
ELSE
N • NX
END IF

calculate first and second derivatives and then error bounds
CALL DCSEVU(K,Y,NK,C,LDC,X,P,N,OX,N,IHSLERR)
D • X(N) - K(NK-1)
P(N) • (3.0 * C(NK-1,3) * D + 2.0 * C(NK-1,2)) * D + C(NK-1,1)
OX(N) • 6.0 * C(NK-1,3) * D + 2.0 * C(NK-1,2)
CALL ERRBND(DP,OX,DZ,DT,P,OX,NX,N)
tau and zeta are stored in K and Y
FOR I • 1 ,N
  K(I) • T(I) + P(I) * X(I)
  Y(I) • T(I) - P(I) * X(I)
END FOR

NX • N
RETURN
END
SUBROUTINE CLFFIT(ID, OD, LD)

* close file for TXFIT

INTEGER ID, OD, LD

CALL FICL(ID, "KEEP")
CALL FICL(OD, "KEEP")
CALL FICL(LD, "KEEP")

RETURN
END
A.4 Program VELO.

The inversion of the $\xi(p)$, $\eta(p)$ 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. VELO is an interactive program to help estimate the surface velocity. It is invoked by typing:

```
VELO infile water-depth
```

where `infilename` 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 (procedure 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 smoothed 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 satisfied 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.
MARK THE VELOCITY TO SURFACE VELOCITY FROM SLOPE AND INTERCEPT OF NEAREST CRUSTAL ARRIVALS

EXTRAPOLATION TO SURFACE VELOCITY FROM SLOPE AND INTERCEPT OF NEAREST CRUSTAL ARRIVALS


INTEGER PD, ND
REAL X(100), T(100), S(100), OX(100)
REAL RV, XMIN, XSCALE, YMIN,
REAL MAXRANGE, WATERDEPTH
CALL OPFVEL(PD, WATERDEPTH)
ND = R2FIT2(X, T, S, OX, 100, PD)
N = 1
WHILE (X(N) .LT. 10.0 .AND. N .LT. ND)
  N = N + 1
END WHILE
CALL ADWMT(T, S, WATERDEPTH, N)
CALL INIPLT
CALL PXT(X, T, 0.0, N, XSCALE, XMIN, YSCALE, YMIN)
CALL PXT(X, S, 0.0, N, XSCALE, XMIN, YSCALE, YMIN)
CALL HYPPLT(WATERDEPTH, XSCALE, XMIN, YSCALE, YMIN)
CALL ESTIM(WATERDEPTH, XSCALE, XMIN, YSCALE, YMIN)
CALL CLFVEL(PD)
END

INTEGER PD, HW
INTEGER NAME(18)
INTEGER FILOP, GETARG
CALL PRIMIO(11)
IF (GETARG(1, NAME, 18) .EQ. -1)
  CALL ERROR(" usage: SURFVEL datafile water_depth")
ELSE
  PD = FILOP(NAME)
END IF
IF (GETARG(2, NAME, 18) .EQ. -1)
  CALL ERROR(" usage: SURFVEL datafile water_depth")
ELSE
  I = 1
  HW = CTOF(NAME, I)
END IF
RETURN
END

INTEGER N
REAL T(*), S(*), HW
CALL VSMAl(T, 1, L.0, T, 1, HW/1.51, N)
CALL VSMAl(S, 1, L.0, S, 1, HW/1.51, N)
RETURN
END
SUBROUTINE HYPPLT(BW, XSCALE, XMIN, YSCALE, YMIN)

* plot direct water wave hyperbola

REAL BW, XSCALE, XMIN, YSCALE, YMIN
INTEGER I
REAL X, T, XX, YY

CALL MOVEPN(0.0, (BW/1.51-YMIN)*YSCALE, 3, 0)

FOR I = 1, 100
X = (9.0 / XSCALE) * FLOAT(I) / 100.0
T = SQRT(X**2 + BW**2)
EXIT FOR IF (T > (YMIN+4.0/YSCALE))
XX = (X - XMIN) * XSCALE
YY = (T - YMIN) * YSCALE
CALL MOVEPN(XX, YY, 3, 1)
END FOR

RETURN
END

SUBROUTINE ESTIM(HW, XSCALE, XMIN, YSCALE, YMIN)

* estimate slope of closest arrivals and calculate surface velocity

REAL HW, XSCALE, XMIN, YSCALE, YMIN
REAL XO, TO, X1, T1, X, T, K, VO, VM, THETA, XX, YY, Z, U

ASINH(U) = ALOG(U + SQRT(U**2+1.0))

* locate intersection of first refracted arrival with direct wave

CALL SETOO(1)
CALL MOVEPN(1.0, 0.5+(HW/1.51-YMIN)*YSCALE, 1, 0)
CALL GRCURS(1)
CALL MOVEGC(1.0, 0.5+(HW/1.51-YMIN)*YSCALE, 1)
CALL LINTYP(-6, 1)
CALL RBLIN(1)
CALL READGC(XX, YY, KEY)
X1 = (XX + XMIN - 1.0) / XSCALE
T1 = (YY + YMIN - 0.5) / YSCALE
CALL MOVEPN(XX, YY, 1, 1)
CALL LINTYP(-1, 1)
CALL RBLIN(1)
CALL READGC(XX, YY, KEY)
XO = (XX + XMIN - 1.0) / XSCALE
TO = (YY + YMIN - 0.5) / YSCALE
VM = (X1 - XO) / (T1 - TO)
THETA = ASIN(1.51/VM)
X = XO - BW * TAN(THETA)

* guess at gradient and calculate solution

LOOP
CALL SETDRM(4)
HCONST
CALL GRSTR(1.2, 4.0, "enter guess for gradient", 24, 0.1, 0.0)
READ (0, FMT="(F10.0)") U
EXIT LOOP IF (U .LE. 0.0)
K = U
IF ((VM/K) .LT. (X/2.0))
ELSE
THETA = ASIN((X/2.0)*SQRT((VM/K)**2-(X/2.0)**2))
XX = X - 1.0 + (X - XMIN) * XSCALE
YY = (T - YMIN) * YSCALE
CALL MOVEPN(XX, YY, 0.5, 0.0)
END IF
END LOOP

RETURN
END
55: CALL ENDPNT
56: 
57: VO = K * SQRT((VM/K)**2-(X/2.0)**2)
58: Z = (VM - VO) / K
59: 
60: WRITE (3,FMT•”(′ gradient ′,F8.2/′ surface velocity ′,F8.2/
61: & ′ depth to turning point ′,F8.2)′) K, VO, 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

133
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 $\xi(p)$ 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 $\xi$ and $\tau$ are considered as independent observations of a model parameter. With this assumption an overdetermined system of equations can be generated relating $\xi$ and $\tau$ 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 `infil`e is the parameter file output from TXFIT with the addition of the sruface velocity value computed by VELO; `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 $2n \times n$ 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.
**Linear inversion of body wave travel time data**

The relationship between the parameters $\tau(p)$, $\zeta(p)$ and $z(v)$ is posed as a linear problem $Ax = y$ where $y$ are the observed parameters $\tau$, $\zeta$; $x$ are the constant velocity gradient layers and $A$ is the coefficient matrix relating the two.

Let $p$ be the slowness for each arrival and $v$ the layer velocity for $i$: 1,3,5,7,... and $j$: 1,2,3,4,...

The velocity gradients are then integrated to yield depth as a function of velocity.

**Usage:** LSVZ infile [outfile] [/CDSX]

- **infile:** line 1: initial value and plot parameters (8f10.0)
  - surface velocity, max velocity, max gradient, min inv gradient, max inv gradient
  - max depth, min slowness, max slowness,
  - min inv gradient, max inv gradient
  - (surface velocity is required, other quantities will be calculated by program if omitted)
  - line 2: plot title (30al)
  - line 3: plot title (30al)
  - line 4-n: observations (5f10.0) from curve fitter
  - $p$, $\zeta$, $\tau$, sigma($\zeta$), sigma($\tau$)

- **outfile:** list output - optional, if not specified output goes to lfn 6

**Options:**
- **C:** use when running on background (control point)
- **D:** singular value decomposition only, results are written to a binary file
- **S:** no decomposition, singular values and vectors are read from binary file
- **X:** extended list output

**Notes:**
SPECIAL COMMON CARRAY

INTEGER NPARAM, NLAYER, NRT
REAL A(100,100), WEIGHT(100), DZDV(50,2), Z(50,2), DV(50)
REAL Q(50), U(LD,100,100), V(50,50), E(50), WORK(100)
REAL SIGMA(100), WEIGHT(100), DV(50,50), R(50), WORK(100)
REAL GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
COMMON /CARRAY/ A, U, V, COV
INTEGER TITLE1(30), TITLE2(30)
REAL 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 RlFIT(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)
CALL GENSYS(A,LDA,PARAM,NLAYER,F,UP,UPMAX)

IF (XOPT .EQ. 1)
CALL MPRINT(A,LDA,NPARAM,NLAYER,"(12F10.3)",1,LD," coefficient matrix")

CALL DECOMP(A,LDA,NPARAM,NLAYER,SIGMA,WEIGHT,U,LDU,Q,V,LDV,E,
& WORK,11,INFO)
IF (INFO .NE. 0) CALL ERROR(" unsuccessful decomposition")
IF (XOPT .EQ. 1)
CALL MPRINT(U,LDU,NPARAM,"(12F10.3)",1,LD," U")
CALL MPRINT(Q,NLAYER,"(12F10.3)",11,LD," eigenvalues")
CALL MPRINT(V,LDV,NLAYER,"(12F10.3)",1,LD," V")
END IF

CALL BINSAV(WEIGHT,U,LDU,Q,V,LDV,PARAM,NLAYER,1,BD)

CALL REMARK("@N enter number of singular values to retain")
READ (TD,) NRT
WRITE (LD,"(I5,"(15," singular values retained")") NRT

* part II — calculate solution in terms of gradients

IF (DOPT .NE. 1)
CALL BINSAV(WEIGHT,U,LDU,Q,V,LDV,PARAM,NLAYER,2,BD)
CALL MPRINT(Q,NLAYER,"(12F10.3)",11,LD," singular values")
CALL REMARK("@N enter number of singular values to retain")
READ (TD,) NRT
WRITE (LD,"(I5,"(15," singular values retained")") NRT
CALL GISOL(U,LDU,Q,V,LDV,PARS,SIGMA,WEIGHT,NPARAM,NLAYER,NRT,DDV,
& COV,LD,C,11111,A)

IF (XOPT .EQ. 1) CALL MPRINT(RSD,NPARAM,NPARAM,l,"(12Fl0.3)",ll,
& (" model residuals")

WRITE (LD,"("\n• square error",F8.2")") INNERP(RSD,l,RSD,l,NPARAM)

WRITE (LD,"(\nsolution",F8.2")") (DDV(l,j),j=1,NLAYER)

IF (XOPT .EQ. 1) CALL MPRINT(COV,LDC,NLAYER,NLAYER,"(12Fl0.3)",ll,
& (" model covariance")

calculate 2-sigma limits for velocity gradients

CALL VSQRT(DZDV(l,2),1,1.0,COV,LDC+l,0.0,NLAYER)

WRITE (LD,"(/\ndz/dv 95% confidence limit//)")

FOR I=1,NLAYER

WRITE (LD,"(Fl0.4,6X,+-,Fl0.4,••••••••••• ,Fl0.4)") (DZDV(I,J),J=1,2),I=1,NLAYER)

END FOR

END IF

part III — choose plots desired

WRITE (3,"(\nplots desired (6 digit decimal expansion abcdef: y=yes, n=no)")

WRITE (3,"(\nplots desired (6 digit decimal expansion abcdef: y=yes, n=no)")

READ (TD,) PLOTOPT

IF (WVVZ) CALL PLTVVZ(Z,LDZ,UP,VSURF,WORK,NLAYER)

IF (WGVP) CALL PLTGVP(DZDV,LDZ,UP,RSD,Q,NLAYER,NPARAM,NRT)

IF (WTZD) CALL REMARK("tau, zeta plots not implemented")

IF (WRIC) CALL PLTRIC(U,LDU,V,LDV,COV,LDC,NPARAM,NLAYER)

IF (WPRJ) CALL PLTPRJ(Q,U,LDU,V,LDV,PARS,DDV,NPARAM,NLAYER,NRT,UP,WORK)

CALL CLFINV(ID,LD)

END
36:         END IF
17:     18:         IF (GETARG(2, NAME, 18) .EQ. -1) THEN
19:   20:         LD = 6
21: ELSE
22:   23:         IF (LD .EQ. -3) THEN
24:   25:         LD = FILCR(NAME)
26: ELSE
27:   28:         END IF
29: IF (LD .EQ. -3) CALL CANT(NAME)
30:  END IF
31:  END IF
32:  COPT = GETOPT(67) ! run from control point
33: IF (COPT .EQ. 1) THEN
34:   TD = 7
35: ELSE
36:   TD = 0
37: END IF
38:  DOPT = GETOPT(68) ! decompose system but do not solve
39: IF (DOPT .EQ. 1 .OR. SOPT .EQ. 1) THEN
40:   CALL REMARK(" enter binary data storage file name")
41:   CALL GETI.IN(BUF,TD)
42:   I = 1
43:   CALL GETWRD(BUF,1,NAME)
44:   BD = FILOP(NAME)
45: IF (BD .EQ. -3) THEN
46:     IF (DOPT .EQ. 1) BD = FILCRB(NAME)
47: END IF
48: END IF
49: XOPT = GETOPT(88) ! extended output listing
50: CALL VPO7MP("MODEL",1200,"END")
51: CALL INIPLT
52: RETURN
53: END
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, VO, P, TZ, S, D)
3:
4: * list input data to list file
5:
6: INTEGER NP, NO, D
7: REAL VO, P(I), TZ(I), S(I)
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/)") VO
15: WRITE (D,FMT="(/' Parameters', IS,' Observations', IS)") NP, NO
16:
17: FOR J = 1,NO
18: FOR I = 1,D
19: WRITE (D,FMT="((S11.3)") P(J), TZ(I), TZ(IP1), S(I), S(IP1)
20: END FOR
21: END FOR
22: RETURN
23: END

1: *-h- GENSYS 511 24 OCT 82 16:43:24
2: SUBROUTINE GENSYS(A, LDA, NPARAM, NLAYER, P, UP, UPHAX)
3:
4: * generate coefficients for system of equations
5:
6: INTEGER LDA, NPARAM, NLAYER
7: REAL A(LDA,1), P(I), UP(I), UPHAX
8: INTEGER I, IP, J
9: REAL PREVP, TAU, ZETA
10:
11: FOR I = 1,NPARAM,2
12: PREVP = UPHAX
13: J = I
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: RETURN
23: 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)/(VLIM*VLIM))
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

141
SUBROUTINE DECOMP(A,LDA,M,N,S,W,U,LDU,Q,V,LDV,E,WORK,JOB,INFO)

* singular value decomposition of a weighted m by n matrix A

T
A = U \text{diag}(q) V

where A = S A' W

INTEGER LDA, M, N, LDU, LDV, JOB, INFO
REAL A(LDA, *), S(*), W(*), U(LDU,*), Q(*), E(*), WORK(*)
INTEGER I
REAL T
REAL FLOAT, INNERP, SQRT, SVSMAB

CALL VRCP(S,1,1.0,S,1,0.0,M)
FOR I = 1,N
W(I) = SVSMAB(A(I,I),1,M)
END FOR

T = 1.0
FOR I = 1,N
W(I) = 1 / SQRT(AB(W(I)))
T = T * W(I)
END FOR
T = SQRT(FLOAT(N)) * T ** (1.0/N)
CALL VSMAI(T,1.0,T,W,1.0,0.0,N)
S = \text{-1/2 1/2}
calculate S

CALL MSVDC(A,LDA,M,N,Q,E,U,LDU,V,LDV,WORK,JOB,INFO)
RETURN
END
SUBROUTINE BINSAV(W, U, LDU, Q, V, LDV, M, N, RW, D)

storage and retrieval of decomposition of system of equations

INTEGER LDU, LDV, M, N, RW, D
REAL W(l), U(LDU,l), Q(l), V(LDV,l)

IF (RW .EQ. 1)
  WRITE (D) (W(I), I=1,N)
  FOR J = 1, N
    WRITE (D) (U(I,J), I=1,M)
  END FOR
  WRITE (D) (Q(I), I=1,N)
  FOR J = 1, N
    WRITE (D) (V(I,J), I=1,N)
  END FOR
ELSE
  READ (D) (W(I), I=1,N)
  FOR J = 1, N
    READ (D) (U(I,J), I=1,M)
  END FOR
  READ (D) (Q(I), I=1,N)
  FOR J = 1, N
    READ (D) (V(I,J), I=1,N)
  END FOR
END IF

RETURN
END

SUBROUTINE GISOL(U,LDU,Q,V,LDV,Y,S,W,M,N,R,X,COV,LDC,RSD,JOB,WORK)

************************************************************************
* Generalized inverse solution of Ax = y *
************************************************************************

where A is a m by n coefficient matrix (m > n) relating observations y and parameters x
A is given by the decomposed system

T
A = U Q V

so that

-1 T
U = (U ,U ) ; Q = (diag,0) ; V = (V ,V )
 0 r r 0

sharp cut-off of singular values at point r determined by user

************************************************************************

U input: M by M matrix of data eigenvectors
output: if requested M by M information density matrix
LDU input: leading dimension of U in calling program
Q input: N element vector of singular values
V input: M by M matrix of solution eigenvectors
output: if requested N by N resolution matrix
LDV input: leading dimension of V in calling program
T input: M element vector of observations
S input: M element vector of variance on observations
W input: M element vector of weight coefficients
M input: number of parameters
N input: number of observations
R input: number of singular values retained in cut-off
X output: if requested N element vector of model parameters
COV output: if requested N by N model covariance matrix
LDC input: leading dimension of COV
RSD output: M element vector of residuals
* JOB input: integer decimal expansion abcd e
* a > 0 calculate solution
* b > 0 calculate model residual
* c > 0 calculate model covariance
* d > 0 calculate data information density
* e > 0 calculate model resolution
* WORK M*M element work space

**--------------------------------------------------------------------------**

INTEGER LDU, LDV, M, N, R, LDC, JOB
REAL U(LDU,*), Q(*), V(LDV,*), Y(*), S(*),
X(*), COV(LDC,*), RSD(*), WORK(*)
INTEGER I
LOGICAL WSOL, WRSD, WCOV, WINF, WRES

* determine job to be done
WSOL • JOB / 10000 .GT. 0
WRSD • MOD(JOB/1000,10) .GT. 0
WCOV • MOD(JOB/100,10) .GT. 0
WINF • MOD(JOB/10,10) .GT. 0
WRES • MOD(JOB,10) .GT. 0

* generalized inverse solution to decomposed system
IF (WSOL)
* calculate S y store in y
CALL VMUL(Y,1,1.0,S,1.0,Y,1.0,0,M)
* calculate solution x = V diag(q ) U y store in y
r r r
CALL MTPRD(WORK,R,U,LDU,Y,M,M,R,1)
CALL VDIV(WORK,1.0,Q,1.0,0.0,WORK,1.0,0.0)
CALL MPRD(X,N,V,LDV,WORK,R,N,R,1)
* calculate W x store in x
CALL VMUL(X,1,1.0,W,1.0,0.0,X,1.0,0.0,N)
END IF

* calculate residuals rad = y - Ax = P'y -(U)'(U)y
X 0 0
IF (WRSD)
CALL MTPRD(WORK,M-R,U(1,R+1),LDU,Y,M,M,M-R,1)
CALL MPRD(RSD,M,U(1,R+1),LDU,WORK,M-R,M,M-R,1)
END IF

* calculate model covariance matrix <xx> = W V diag(q ) V W
r r r
IF (WCOV)
FOR I• l,R
CALL VSMAl(WORK(N*(I-1)+l),1,1.0/Q(I),V(l,I),1,0.0,N)
END FOR
CALL MPRDT(COV,LDC,WORK,N,WORK,N,N,R,N)
FOR J• l,N
FOR I• l,N
COV(I,J) • W(I) * COV(I,J) * W(J)
END FOR
END FOR
END IF

* calculate information density E = U U
r r r
IF (WINF)
FOR I = 1,R
. CALL VSMAl(WORK(N*(I-1)+1),1,1.0/Q(I),W(I),1,1.0,0.0,N)
END FOR
. CALL MPRDT(COV,LDU,WORK,N,WORK,N,N,N,R)
FOR J• l,N
. FOR I = 1,N
. . . CALL COV(I,J) • W(I) * COV(I,J) * W(J)
. END FOR
END FOR
END IF
127:       . END FOR
128:     END IF
129:
130:      *= calculate model resolution \( R = \frac{V}{V_r} \)
131:      *=
132:     *=
133:
134:     IF (VRES)
135:        CALL MPEDT(WORK,N,V,LDV,V,LDV,N,R,N)
136:        FOR I = 1,N
137:           CALL VCOPY(V(I),I,WORK(N*(I-1)+1),N)
138:        END FOR
139:      END IF
140:   RETURN
141: END

1: *-b- 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: PLAST = PMAX
11: ZLAST = 0.0
12: FOR I = 1,N
13:    DV(I) = 1.0 / P(I) - 1.0 / PLAST
14:    Z(I) = ZLAST + DZDV(I) * DV(I)
15:    PLAST = P(I)
16:    ZLAST = Z(I)
17: END FOR
18: RETURN
19: END

1: *-b- 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, SOFT
13: REAL HEIGHT, KLEN, EXMIN, EXMAK, T,YLEN, MAX, MIN, YSCALE
14: DATA PENDOWN /1/, PENUP /0/
15: DATA XLEN /8.0/, YLEN /8.0/, HEIGHT /0.15/
16: DATA XLEN /5.0/, YLEN /8.0/, HEIGHT /0.15/
17: CALL REMARK(" Plot velocity vs depth")
18: HCONST
19: CALL VERP(WORK,1.0,1.0,P,1.0,0,N)
20: CALL NEWPEN(3)
21: CALL GRST(1.0,1.0,0.0,TITLE1,30,0.2,0.0)
22: CALL MOVE00(1.0,0.0)
23:   IF (VMAX .EQ. 0.0)
24:      . SOFT = 1
25:     ELSE
26:      . SOFT = 0
27:     END IF
28:     XMAX = VMAX
29:     END IF
30:     XMIN = 0.0
31:   NAX = INT(XMAX)
32:   CALL PAXIS(0.0,KLEN,0.0,0.0,XMIN,XMAX,4,-1,HEIGHT,4,90.0,SOFT,
33: & NAX,1)
34:   XSCALE = KLEN / XMAX
35:   CALL AXLAB(0.0,KLEN,0.0,0.0,"VELOCITY (KM S^{-1} )","
36: & HEIGHT,3.5*HEIGHT)
IF (ZMAX .EQ. 0.0)
  .SOFT = 1
  YMAX = SVMAX(Z(1,1),1,JUNK,N)
ELSE
  .SOFT = 0
  YMAX = ZMAX
END IF

YMIN = 0.0
NAY = INT(YMAX)
CALL PAXIS(O.0,O.0,O.0,-YLEN,YMIN,YMAX,4,-1,HEIGHT,2,180.0,SOFT,
& NAY,1)
YSCALE = YLEN / YMAX
CALL AXLAB(O.0,O.0,-YLEN,0.0,"DEPTH (km)",HEIGHT,4.5*HEIGHT)

XX = VSURF * XSCALE
YY = 0.0
FOR I = 1,N
  XX = WORK(I) * XSCALE
  YY = -Z(I,1) * YSCALE
  CALL PNTCNF(XX,YY,0.0,Z(I,2)*YSCALE,0.05)
END FOR
CALL ENDFIG(1)
RETURN
END

SUBROUTINE PLTGVZ(GRD, Z, WORK, N)
* plot velocity gradient vs. depth
INTEGER N
REAL GRD(*), Z(*), WORK(*)
INTEGER TITLE1(30), TITLE2(30)
REAL VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
COMMON /CPLT/ VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX,
& TITLE1, TITLE2
INTEGER I, JUNK, NAX, NAY, PEN, PENDOWN, PENUP, SOFT
REAL LENGTH, XLEN, XMIN, XSCALE, YLEN, YMAX, YMIN, YSCALE
REAL SVMAX
DATA XLEN /5.0/, YLEN /8.0/, HEIGHT /0.15/
DATA PENDOWN /1/, PENUP /0/
CALL REMARK(" Plot gradient vs depth")
HCONST
CALL VRCP(WORK,l,l.O,GRD,l,O.O,N)
CALL MOVE00(l.0,9.0)
IF (GMAX .EQ. 0.0)
  SOFT = 1
  XMAX = SVMAX(WORK,1,JUNK,N)
ELSE
  .SOFT = 0
  XMAX = GMAX
END IF
XMIN = 0.0
CALL PAXIS(O.0,XLEN,O.0,O.0,XMIN,XMAX,4,1,HEIGHT,4.90.0,SOFT,
& & NAX,1)
XSCALE = XLEN / XMAX
CALL AXLAB(O.0,XLEN,0.0,0.0,\"DV/DZ (Sn-lo)\",HEIGHT,3.5*HEIGHT)
IF (ZMAX .EQ. 0.0)
  SOFT = 1
  YMAX = SVMAX(WORK,1,JUNK,N)
ELSE
  .SOFT = 0
  YMAX = ZMAX
END IF
YMIN = 0.0
CALL PAXIS(O.0,O.0,O.0,-YLEN,YMIN,YMAX,4,-1,HEIGHT,2,180.0,SOFT,
& & NAY,1)
YSCALE = YLEN / YMAX
RETURN
END
CALL AXLAB(0.0,0.0,-YLEN,0.0,"DEPTH (km)",HEIGHT,4.5*HEIGHT)

52: :NOT HCONST
53: IF (GMAX .EQ. 0.0)
54:   J = 1
55: ELSE
56:   J = 2
57: END IF
58: PEN = PENUP
59: FOR I = J,N
60: XX = (WORK(I) - XMIN) * XSCALE
61: CALL MOVEPN(XX,YY,3,PEN)
62: PEN = PENDOWN
63: YY = -Z(I) * YSCALE
64: CALL MOVEPN(XX,YY,3,PEN)
65: END FOR
66: CALL ENDFIG(l)
67: RETURN
68: END

1: *---* PLTGVP 2641 A 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: INTEGER LDG, M, N, R
6: REAL GRD(LDG,2), P(*), RSD(*), Q(*)
7: INTEGER TITLE1(30), TITLE2(30)
8: REAL VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
9: COMMON /CPLOT/ VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX,
10: & TITLE1, TITLE2
11: INTEGER I, J, PEN, PENWIDTH(-1:1)
12: INTEGER NAX, NAY, PENDOWN, PENUP, SOFT
13: REAL HEIGHT, XLEN, XSCALE, YLEN, YMAX, YSCALE, XX, YY
14: REAL XMAX, XMIN
15: INTEGER ITOC, FTOC
16: REAL FLOAT, AINT, INNERP, SVMAX, SVMIN
17: DATA PENWIDTH /1,4,1/, XLEN /6.0/, YLEN /5.0/, HEIGHT /0.15/
18: DATA PENDOWN /1/, PENUP /0/
19: CALL :BCONST
20: CALL :CONST
21: CALL NEWPEN(3)
22: CALL GESTR(1.0,8.5,"SOLUTION — ",12,HEIGHT,0.0)
23: CALL GNUM(2.7,8.0,FLOAT(N),3,-1,HEIGHT,0.0)
24: CALL GESTR(3.0,8.0," LAYERS",7,HEIGHT,0.0)
25: CALL GNUM(2.7,7.7,FLOAT(M),3,-1,HEIGHT,0.0)
26: CALL GESTR(3.0,7.7," DATA",5,HEIGHT,0.0)
27: CALL GNUM(2.7,7.4,FLOAT(R),3,-1,HEIGHT,0.0)
28: CALL GESTR(3.0,7.4," SINGULAR VALUES RETAINED",25,HEIGHT,0.0)
29: CALL GNUM(6.6,7.1,REO,5.2,HEIGHT,0.0)
30: CALL GESTR(3.0,6.8," CONDITION NUMBER =",19,HEIGHT,0.0)
31: CALL GNUM(6.15,6.8,Q(1)/Q(R),5.2,HEIGHT,0.0)
32: CALL MOVEOO(1.0,2.0)
33: IF (PMIN .EQ. 0.0)
34:   SOFT = 1
35:   XMIN = SVMIN(P,1,JUNK,N)
36: ELSE
37:   SOFT = 0
38:   XMIN = PMIN
39:   XMAX = PMAX
40: END IF
41: NAX = 5
CALL PAXIS(0.0,XLEN,0.0,0.0,XMIN,XMAX,4,2,HEIGHT,3,-90.0,SOFT,
&MAX,1)
XSCALE = XLEN / (XMAX - XMIN)
CALL AXLAB(0.0,XLEN,0.0,0.0,"SLOWNESS (S km-1lo)",
&HEIGHT,-4.5*HEIGHT)
IF (IGMIN .EQ. 0.0 .AND. IGMAX .EQ. 0.0)
SOPT = 1
YMIN = SVMIN(GRD(l,1),1,JUNK,N)
YMAX = SVMAX(GRD(l,1),1,JUNK,N)
ELSE
SOPT = 0
YMIN = IGMIN
YMAX = IGMAX
END IF
NAY = 5
CALL PAXIS(O.O,0.0,0.0,YLEN,YMIN,YMAX,5,l,REIGHT,2,180.0,SOPT,
&NAY,l)
YSCALE = YLEN / (YMAX - YMIN)
CALL AXLAB(O.O,0.0,0.0,YLEN,"DZ/DV (S). " ,HEIGHT , 5.5'*BEIGHT )
NOT HCONST
FOR J = -1,1
CALL NEWPEN (PENWIDTH(J))
PEN = PENUP
XX = (XMAX - XMIN) * XSCALE
FOR I = 1,N
YY = (GRD(I,1) + J * GRD(I,2) - YMIN) *
YSCALE
CALL MOVEPN(XX,YY,3,PEN)
PEN = PENDOWN
XX = (P(I) - XMIN) * XSCALE
CALL MOVEPN(XX,YY,3,PEN)
END FOR
END FOR
CALL ENDFIG(l)
RETURN
END

SUBROUTINE PLTRIC(U, LDU, V, LDV, COV, LDC, M, N)
* plot resolution VV , information density UU and model covariance
INTEGER LDU, LDV, M, N
REAL U(LDU,*), V(LDV,*), COV(LDC,*)
REAL XLEN, XOFF, YLEN, YOFF
INTEGER DF
DATA XLEN /3.0/, XOFF /3.9/, YLEN /3.0/, YOFF /5.0/
CALL REMARK(" Plot resolution,information,covariance matrices")
IF (N .LE. 25)
DF = 5
ELSE
DF = 10
END IF
CALL MOVE00(0.6,1.5)
CALL NEWPE N ( 2 )
CALL AXLAB ( 0.0,XOFF+XLEN , 0.0,0.0,"INFORMATION DENSITY.",0.15,-1.0)
CALL PMAT(U,LDU,N,2,N,2,XLEN,YLEN,0.8,0.15,DF,DF)
CALL AXLAB(O.O,XLEN,O.O,O.O, "ZETA OBSERVATIONS.",0.12,-0.6)
CALL MOVE00(-XOFF,YOFF)
CALL PMAT(U(2,2),LDU,N,2,N,2,XLEN,YOFF,0.8,0.15,DF,DF)
CALL AXLAB(O.O,XLEN,0.0,0.0,"TAU OBSERVATIONS.",0.12,-0.6)
CALL MOVE00(-XOFF,YOFF)
148
CALL NEWPEN(2)
CALL PMAT(V,LDV,N,1,1,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 MOVEOU(XOFF,0.0)
CALL NEWPEN(2)
CALL PMAT(COV,LDC,N,1,1,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

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(3a), TITLE2(3a)
REAL VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX
COMM /CPLOT/ VMAX, GMAX, ZMAX, PMIN, PMAX, IGMIN, IGMAX,
& TITLE1, TITLE2
INTEGER DF, NAX, NAY, PEN, PENDOWN, PENUP, SOFT
REAL XI.EN, XMIN, XOFF, XSCALE,
XX
REAL
YLEll, YMIN, YOFF, YSCALE,
YY
REAL SVMAX, SVMIN
DATA PENDOWN /1/, PENUP /a/
DATA XLEN /3.a/, XOFF /3.9/, YLEN /1.5/, YOFF /3.a/

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

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

CALL MOVEOU(3.5,1.8)
CALL NEWPEN(2)
CALL VRCP(WK,1,1.0,Q,1,a.O,N)
CALL PAXIS(a.a,XLEN,a.a,a.a,a.0,FLOAT(N),4,-l,0.12,3,-9a.a,
& a,N,DF)
XX = FLOAT(R) * XSCALE
YY = a.o
CALL MOVEPN(XX,YY,3,PENUP)
YY = YLEN
CALL MOVEPN(XX,YY,3,PENDOWN)
XX = XX - 0.2
YY = YY + a.16
CALL GRSTR(XX, YY, "SHARP" ,5 ,O.a8,a .0)

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

CALL MOVEOU(3.5,1.8)
CALL NEWPEN(2)
CALL VRCP(WK,1,1.0,Q,1,a.O,N)
CALL PAXIS(a.a,XLEN,a.a,a.a,a.0,FLOAT(N),4,-l,0.12,3,-9a.a,
& a,N,DF)
XX = FLOAT(R) * XSCALE
YY = a.o
CALL MOVEPN(XX,YY,3,PENUP)
YY = YLEN
CALL MOVEPN(XX,YY,3,PENDOWN)
XX = XX - 0.2
YY = YY + a.16
CALL GRSTR(XX, YY, "SHARP" ,5 ,O.a8,a .0)

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

CALL MOVEOU(3.5,1.8)
CALL NEWPEN(2)
CALL VRCP(WK,1,1.0,Q,1,a.O,N)
CALL PAXIS(a.a,XLEN,a.a,a.a,a.0,FLOAT(N),4,-l,0.12,3,-9a.a,
& a,N,DF)
XX = FLOAT(R) * XSCALE
YY = a.o
CALL MOVEPN(XX,YY,3,PENUP)
YY = YLEN
CALL MOVEPN(XX,YY,3,PENDOWN)
XX = XX - 0.2
YY = YY + a.16
CALL GRSTR(XX, YY, "SHARP" ,5 ,O.a8,a .0)

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

CALL MOVEOU(3.5,1.8)
CALL NEWPEN(2)
CALL VRCP(WK,1,1.0,Q,1,a.O,N)
CALL PAXIS(a.a,XLEN,a.a,a.a,a.0,FLOAT(N),4,-l,0.12,3,-9a.a,
& a,N,DF)
XX = FLOAT(R) * XSCALE
YY = a.o
CALL MOVEPN(XX,YY,3,PENUP)
YY = YLEN
CALL MOVEPN(XX,YY,3,PENDOWN)
XX = XX - 0.2
YY = YY + a.16
CALL GRSTR(XX, YY, "SHARP" ,5 ,O.a8,a .0)
CALL GRSTR(XX, YY, "CUT-OFF", 7, 0.08, 0.0)

* projection onto column (observation) space of matrix

IF (PMIN .EQ. 0.0)
   SOPT • 1
   XMIN • SVMIN(P, l, JUNK, N)
   XMAX • SVMAX(P, l, JUNK, N)
ELSE
   SOPT • 0
   XMIN • PMIN
   XMAX • PMAX
END IF

IF (PMm .EQ. 0.0)
   SOPT • 1
   XMIN • SVMIN(P, l, JUNK, N)
   XMAX • SVMAX(P, l, JUNK, N)
ELSE
   SOPT • 0
   XMIN • PMm
   XMAX • PMAX
END IF

CALL MOVE00(-2.0, YOFF)
CALL MPRD(WK, M, U, LDU, Y, M, M, 1)
NAX • 5
CALL PAXIS(0.0, XLEN, 0.0, 0.0, XMIN, XMAX, 4, 0.12, 3, -90.0, SOFT, & NAX, 1)
XSCALE • XLEN / (XMAX - XMIN)
CALL AXLAB(0.0, XLEN, 0.0, 0.0, "(S KM-1o).", 0.15, -0.6)
CALL AXLAB(0.0, XLEN, 0.0, 0.0, "PROJECTION ON COLUMN SPACE.", & 0.15, -0.9)
YMm • 0.0
YMAX • SVMAX(WK, 2, JUNK, N)
NAY • 3
CALL PAXIS(XOFF, 0.0, 0.0, 0.0, YMIN, YMAX, 4, 0.12, 3, 180.0, 1, NAX, 1)
CALL AXLAB(0.0, XLEN, 0.0, 0.0, "TAU", 0.15, 0.6)
YSCALE • YLEN / (YMAX - YMIN)
CALL GRSTR(2.0, 1.0, "ZETA", 4, 0.1, 0.0)
PEN • PENUP
FOR I • 1, N
   J • 2 * I - 1
   XX • (P(I) - XMIN) * XSCALE
   YY • (WK(J) - YMm) * YSCALE
   CALL MOVEPN(XX, YY, 3, PEN)
PEN • PENDOWN
END FOR

FOR I • 1, N
   J • 2 * I - 1
   XX • (P(I) - XMIN) * XSCALE
   YY • (Y(J) - YMm) * YSCALE
   CALL GRSYMB(XX, YY, 11, 0.05, 0.0)
END FOR

* projection onto row (model) space

CALL MOVE00(-1.9, YOFF)
CALL MPRD(WK, N, V, LDV, X, N, N, 1)
CALL PAXIS(XOFF, 0.0, XLEN, 0.0, 0.0, XMIN, XMAX, 4, 2, 0.12, 3, -90.0, MAX, 1)
CALL AXLAB(0.0, XLEN, 0.0, 0.0, "(S KM-1o).", 0.15, -0.6)
CALL AXLAB(0.0, XLEN, 0.0, 0.0, "PROJECTION ON ROW SPACE.", & 0.15, -0.9)
YMIN = 0.0
YMAX = SYMAX(WK, 1, JUNK, N)
NAY = 3
CALL PAXIS(0.0, 0.0, 0.0, YLEN, YMIN, YMAX, 4, 0, 12, 2, 180.0, 0.1, NAY, 1)
CALL AXLAB(0.0, 0.0, 0.0, YLEN, "(S)", "", 0.15, 0.6)
YSCALE = YLEN / (YMAX - YMIN)
PEN = PENUM
FOR I = 1, N
  XX = (P(I) - XMIN) * XSCALE
  YY = (WK(I) - YMIN) * YSCALE
  CALL MOVEPN(XX, YY, 3, PEN)
PEN = PENDOWN
END FOR
FOR I = 1, N
  XX = (P(I) - XMIN) * XSCALE
  YY = (X(I) - YMIN) * YSCALE
  CALL GRSYM(XX, YY, 11, 0.05, 0.0)
END FOR
END FOR
NOT BCONST
CALL ENDFIG(1)
RETURN
END

* Close files and terminate plotting
INTEGER ID, LD
CALL FILCL(ID, "KEEP")
CALL FILCL(LD, "KEEP")
CALL ENDPLT
RETURN
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 transmission 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 WlPIK is used for output in TPIK, but RlPIK 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 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.
**COMPRS**

SUBROUTINE COMPRS(BUF, SHOT, RCV, CH, SB, DS, RATE, SAMP, FS, AS, HD)

* output shortened buffer to save file

INTEGER BUF(*), SHOT, RCV, CH, RATE, SAMP, SD, HD
INTEGER*6 SB, DS
REAL FS, AS
INTEGER HBUF(18), STSAMP, NSAMP, SW, ADR
INTEGER*6 SB, DS
REAL TIME
CHARACTER INTERNAL*54
EQUIVALENCE (INTERNAL, HBUF)

* calculate start time, end time
TIME = FLOAT(SAMP) / FLOAT(RATE) - FLOAT(SB-DS) / 1000.0
STSAMP = SAMP - INT((TIME/FS)#1000.0)
NSAMP = INT((AS+FS)#FLOAT(RATE))

* encode header and output to temporary file
CALL DSTAT(SD, S6W, ADR)
WRITE (INTERNAL, F'MT•'(2I6.I2.2I6.I3.2X.A3)') RCV, SHOT, CH, SB, DS,
RATE, ADR
BUFFF. OUT (HD, HBUF, 18, SW)
CALL STATUS(HD)

* output data
CALL BUFOUT(SD, BUF(STSAMP).NSAMP, SW)
RETURN
END

**ENDMX**

SUBROUTINE ENDMX(DD, HD, HBUF, NSAMP)

* append headers to end of data file and close
INTEGER DD, HD, NSAMP, HBUF(112)
INTEGER BDRADR, SW
INTEGER*6 S6W
CALL DSTAT(DD, S6W, HDRADR)
ENDFILE HD
REWIND HD
CALL DPOS(DD, 2)
CALL BUFIN(DD, HBUF, 112, SW)
HBUF(111) = NSAMP
HBUF(112) = HDRADR
CALL DPOS(DD, HDRADR)
CALL DPOS(DD, 2)
CALL BUFOUT(DD, HBUF, 112, SW)

LOOP
BUFFER IN (HD, HBUF, B, 18, SW)
CALL STATUS(HD)
EXIT LOOP IF (SW .GE. 3)
CALL BUFOUT(DD, HBUF, 18, SW)
END LOOP
ENDFILE DD
CLOSE (UNIT=DD)
CLOSE (UNIT=HD, STATUS=‘DELETE’)
RETURN
END

**FILBUF**

INTEGER FUNCTION FILBUF(D, BUF, N, ADR)

* fill buffer with shot data from standard HIG demultiplex file
INTEGER D, BUF(*), N, ADR

154
8: CALL DPOS(D,ADR)
9: CALL BUFIN(D,BUF,N,SW)
11: IF (SW .GE. 3)
12: FILBUF = -3
13: ELSE
14: FILBUF = -2
15: END IF
17: RETURN
18: END

1: *-h- FLTTIM 301 2 MAY 83 22:49:19
2: SUBROUTINE FLTTIM(X, FL, F2, NPOLES, GAIN, N)
4: * apply n-pole filter to real array x
5: INTEGER NPOLES, N
6: REAL X(*), FL, F2, GAIN
8: INTEGER ORDER
10: ORDER = NPOLES / 2
12: CALL FILT(X, ORDER, 1)
13: T = 1.0 / GAIN
14: CALL VSMAX(X, 1.0, X, 1.0, 1.0, 0, N)
16: RETURN
17: END

1: *-h- GETCOR 1278 26 MAY 83 22:21:30
2: INTEGER FUNCTION GETCOR(R, S, BUF, D)
4: * find corfile buffer containing data for shot s
5: INTEGER R, S, BUF(*), D
6: INTEGER I, IOSW, M, N, STR(13)
7: INTEGER CTOI
8: BUFFER IN (D, BUF, B, 140, IOSW)
9: CALL STATUS(D)
10: IF (IOSW .GE. 3)
11: REWIND
12: BUFFER IN (D, BUF, B, 140, IOSW)
13: CALL STATUS(D)
14: IF (IOSW .GE. 3)
15: GETCOR = -3
16: RETURN
17: END IF
18: END IF
19: CALL UNPACK(BUF, STR, 4)
20: STR(13) = -2
21: IF ((M .EQ. R .AND. N .EQ. S) .OR. (R .EQ. 0 .AND. N .EQ. S))
22: GETCOR = -2
23: RETURN
24: END IF
25: CALL UNPACK(BUF, STR, 4)
26: STR(13) = -2
28: I = 1
29: M = CTOI(STR, 1)
30: N = CTOI(STR, 1)
31: 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: IF ((M .EQ. R .AND. N .EQ. S))
37: & REWIND D
38: LOOUP
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

155
I • 1
M • CTO1 STR • 1
N • CTO1 STR • 1
EXIT LOOP IF ((M . EQ . R . AND. N . EQ. S) . OR.
&.
(R . EQ . 0 . AND. N . EQ. S))
~ND
LOOP
GETCOR • -2
RETURN
END

1: *-h- GETDHD 592 30 JUN 82 10:16:50
2: INTEGER FUNCTION GETDHD(D. HAD. SHOT. RCV, CH. SB, OS. 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(l8). SW
9: CHARACTER*54 INTERNAL
10:
11:
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:
25:
26:
27:
28:
29:
30:
31:
32:
33:
34:
35:
36:
37:
38:
39:
40:
41:
42:
43:
44:
45:
46:
47:
48:
49:
50:
51:
52:
53:
54:
55:
56:
57:
58:
59:
60:
61:
62:
63:
64:
65:
66:
67:
68:
69:
70:
71:
72:
73:
74:
75:
76:
77:
78:
79:
80:
81:
82:
83:
84:
85:
86:
87:
88:
89:
90:
91:
92:
93:
94:
95:
96:
97:
98:
99:
100:
101:
102:
103:
104:
105:
106:
107:
108:
109:
110:
111:
112:
113:
114:
115:
116:
117:
118:
119:
120:
121:
122:
123:
124:
125:
126:
127:
128:
129:
130:
131:
132:
133:
134:
135:
136:
137:
138:
139:
140:
141:
142:
143:
144:
145:
146:
147:
148:
149:
150:
151:
152:
153:
154:
155:
156:
157:
158:
159:
160:
161:
162:
163:
164:
165:
166:
167:
168:
169:
170:
171:
172:
173:
174:
175:
176:
177:
178:
179:
180:
181:
182:
183:
184:
185:
186:
187:
188:
189:
190:
191:
192:
193:
194:
195:
196:
197:
198:
199:
200:
201:
202:
203:
204:
205:
206:
207:
208:
209:
210:
211:
212:
213:
214:
215:
216:
217:
218:
219:
220:
221:
222:
223:
224:
225:
226:
227:
228:
229:
230:
231:
232:
233:
234:
235:
236:
237:
238:
239:
240:
241:
242:
243:
244:
245:
246:
247:
248:
249:
250:
251:
252:
253:
254:
255:
256:
257:
258:
259:
260:
261:
262:
263:
264:
265:
266:
267:
268:
269:
270:
271:
272:
273:
274:
275:
276:
277:
278:
279:
280:
281:
282:
283:
284:
285:
286:
287:
288:
289:
290:
291:
292:
293:
294:
295:
296:
297:
298:
299:
300:
301:
302:
303:
304:
305:
306:
307:
308:
309:
310:
311:
312:
313:
314:
315:
316:
317:
318:
319:
320:
321:
322:
323:
324:
325:
326:
327:
328:
329:
330:
331:
332:
333:
334:
335:
336:
337:
338:
339:
340:
341:
342:
343:
344:
345:
346:
347:
348:
349:
350:
351:
352:
353:
354:
355:
356:
357:
358:
359:
360:
361:
362:
363:
364:
365:
366:
367:
368:
369:
370:
371:
372:
373:
374:
375:
376:
377:
378:
379:
380:
381:
382:
383:
384:
385:
386:
387:
388:
389:
390:
391:
392:
393:
394:
395:
396:
397:
398:
399:
400:
401:
402:
403:
404:
405:
406:
407:
408:
409:
410:
411:
412:
413:
414:
415:
416:
417:
418:
419:
420:
421:
422:
423:
424:
425:
426:
427:
428:
429:
430:
431:
432:
433:
434:
435:
436:
437:
438:
439:
440:
441:
442:
443:
444:
445:
446:
447:
448:
449:
500:
501:
502:
503:
504:
505:
506:
507:
508:
156
GAINMAX = SVMAX(GAIN,1.JUNK.NYQFREQ)
RETURN
END

*--* INDMX 266 30 JUN 82 10:16:53
SUBROUTINE INDMX(F, BUF)
* write 3 blank records to set up seismic data file
INTEGER F, BUF(112)
INTEGER SW
CALL BUFOUT(F,BUF,112,SW)
CALL BUFOUT(F,BUF,112,SW)
CALL BUFOUT(F,BUF,112,SW)
ENDFILE F
RETURN
END

*--* LOCDHD 275 30 JUN 82 10:16:52
INTEGER FUNCTION LOCDHD(D, BUF, N)
* find start address of headers in standard HIG seismic data file
INTEGER D, BUF(112), N
INTEGER SW
CALL DPOS(D,2)
CALL BUFIN(D,BUF,112,SW)
N = BUF(lll)
LOCDHD = BUF(ll2)
RETURN
END

*--* PXT 1144 6 JAN 83 15:21:38
SUBROUTINE PXT(X, Y, RV, N, XMIN, XSCALE, YMIN, YSCALE)
*
plot data points, calculate plot parameters
INTEGER N
REAL X(*), Y(*), RV, XSCALE, YSCALE
REAL XLEN, XMAX, XMIN, YLEN, YMAX
INTEGER JUNK, ML, M2, NN
REAL SVMAX, SVMIN
DATA XI.EN /8.0/, YLEN /4.0/
NN = IABS(N)
XMIN = X(l)
XMAX = X(NN)
IF (RV .NE. 0,0) CALL VSMA2(Y,l.l.JUNK,NN)
YHAX = SVMAX(Y, l.JUNK.NN)
YMIN = SVMIN(Y,l.JUNK.NN)
IF (N .GT. 0) ML = NINT(XMAX)
M2 = 4
CALL PAXIS(0.0.XLEN.0.0.0.0.XMIN.XMAX,3.-1.0.15.3.-90.0.1.ML,5)
CALL PAXIS(0.0.0.0.0.0.YLEN.YMIN.YMAX,5.2.0.15.2.180.0.1.M2,1)
CALL PAXIS(XLEN.XLEN.0.0.YLEN.YMIN.YMAX,0.0.0.15.0.0.0.M2.1)
END IF
XSCALE = XLEN / (XMAX - XMIN)
YSCALE = YLEN / (YMAX - YMIN)
FOR I = 1,NN
XX = XSCALE * (X(I) - XMIN)
YY = YSCALE * (Y(I) - YMIN)
CALL SYMBOL(XX,YY,0.1.42,0.0.-1)
END FOR
40: IF (RV .NE. 0.0) CALL VSMA2(Y,1.1.0,Y,1.1.0/RV,X,1.0.0,NN)
41: RETURN
42: END

1: *-h- PXTSM 393 6 JAN 83 15:21:40
2: SUBROUTINE PXTSM(X, Y, RV, N, XSCALE, YMIN, YSCALE)
3: * plot fitted curve on frame previously drawn
4: INTEGER N
5: REAL X(*), Y(*), RV, XSCALE, XMIN, YSCALE, YMIN
6: IF (RV .NE. 0.0) CALL VSMA2(Y,1.1.0,Y,1.1.0/RV,X,1.0.0,N)
7: CALL PVF2(X, Y, N, 0.0, XMIN, XSCALE, YMIN, YSCALE)
8: IF (RV .NE. 0.0) CALL VSMA2(Y,1.1.0,Y,1.1.0/RV,X,1.0.0,N)
9: RETURN
10: END

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

1: *-h- R2FITI 783 19 MAY 83 23:49:54
2: INTEGER FUNCTION R2FITI(K, Y, C, LDC, MAXDATA, D)
3: * read spline parameters of fitted curve (beginning marked by "@")
4: INTEGER LDC, MAXDATA, D
5: REAL K(*), Y(*), C(LDC, *)
6: INTEGER STR(2), PAT(81), LIN(82), N
7: INTEGER GETLIN, GETPAT, MATCH
8: DATA STR /64,-2/
9: IF (GETPAT(STR, PAT) .EQ. -3) CALL ERROR(" illegal pattern")
10: WHILE (GETLIN(LIN.D) .NE. -1)
11: EXIT WHILE IF (MATCH(LIN.PAT) .EQ. 1)
12: END WHILE
13: BACKSPACE D
14: N = 1
15: LOOP
16: IF (N .GT. MAXDATA)
17: CALL ERROR(" r2fiti: too many data")
18: READ (D.FMT ."(SF11.0)") K(N), Y(N), (C(N I).I .3)
19: EXIT LOOP IF (K(N) .EQ. 0.0 .AND. Y(N) .EQ. 0.0 .AND.
20: & C(N I) .EQ. 0.0)
21: N = N + 1
22: END LOOP
23: BACKSPACE D
24: N = 1
25: LOOP
26: IF (N .GT. MAXDATA)
27: CALL ERROR(" r2fiti: too many data")
28: READ (D.FMT ."(5F11.0)") K(N), Y(N), (C(N I).I .3)
29: EXIT LOOP IF (K(N) .EQ. 0.0 .AND. Y(N) .EQ. 0.0 .AND.
30: & C(N I) .EQ. 0.0)
31: N = N + 1
32: BACKSPACE D
33: EXIT LOOP
34: END

158
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")
I = 1
S(N) = CTOI(LIN(I))
IF (S(N) .GE. SS .AND. S(N) .LE. ES)
   IF (Cl .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 = 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
   END LOOP
   CONTINUE
RETURN
END
SUBROUTINE SIGPAR(DATA, N, SMAX, SMIN, SMEAN)

* calculate max, min and mean of vector

INTEGER N
REAL DATA(*), SMAX, SMIN, SMEAN
INTEGER I
SMAX = DATA(1)
SMIN = DATA(1)
SMEAN = DATA(1) / FLOAT(N)
FOR I = 2, N
SMAX = AMAX1(SMAX, DATA(I))
SMIN = AMIN1(SMIN, DATA(I))
SMEAN = SMEAN + DATA(I) / FLOAT(N)
END FOR
RETURN
END

INTEGER FUNCTION STSAMP(DS, SB, ST, RANGE, RV, RATE)

* calculate sample number to start plotting. if rv=0 then start at sb

INTEGER RATE
INTEGER*6 DS, SB
REAL ST, RANGE, RV
REAL RT
IF (IABS(SB-DS) .GT. 8388607D) ST = 77777777
RETURN
END IF
IF (RV .EQ. 0.0) RT = 0.0 ELSE RT = RANGE / RV
RETURN
END

SUBROUTINE WICOR(S, X, DX, T, DT, D)

INTEGER S, D
REAL X, DX, T, DT
CALL PUTDEC(S)
CALL PUTFLT(X, 10.3)
CALL PUTFLT(DX, 8.3)
CALL PUTFLT(T, 10.3)
CALL PUTFLT(DT, 8.3)
CALL PUTCH
RETURN
END
1: *-h- WlFIT 448 12 MAY 83 15:06:05
2: SUBROUTINE WlFIT(F, Z, T, DZ, DT, N, D)
3: *
4: * output new parameters for LSVZ
5: *
6: INTEGER N, D
7: REAL F(*), 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(F(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: RETURN
21: 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: WRITE (D, FMT•“(’ ’,80(’ ’))”)
11: WRITE (D, FMT•“(’+’ ,80(’ ’)/)”)
12: WRITE (D, FMT•“(#)’”)”)
13: FOR I = 1, N
14: WRITE (D, FMT•“(I5 , 6F10.3)”) K(I), Y(I), C(I, 1)
15: END FOR
16: WRITE (D, FMT•“(’Rms error’ ,F8.3/)”) E
17: RETURN
18: 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: WRITE (D, FMT•“(’ ’,80(’ ’))”)
11: & ‘OX’, 11X, ’)’”)”)
12: WRITE (D, FMT•“(’+’ ,80(’ ’)/)”)
13: WRITE (D, FMT•“(’ ’,80(’ ’))”)
14: FOR I = 1, N
15: WRITE (D, FMT•“(”’”,80(’ ‘))”)
16: & ’P(I), Q(I)
17: END FOR
18: RETURN
19: END

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

161
CALL PUTDFC(S.5.D)
CALL PUTFLT(X,10.3.D)
FOR I = 1,3
IF (INDEX(I) .EQ. 1)
    CALL PUTFLT(T(I),10.3.D)
    CALL PUTFLT(DT(I),8.3.D)
ELSE
    CALL PUTTAB(D)
END IF
END FOR
CALL PUTDEC(R.5.D)
CALL PUTCH(10.D)
RETURN
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 successful, 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, usually 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.
INTEGER FUNCTION ADDSET(C, SET, J, MAXSIZ)

* put c in set(j) if it fits, increment j

IF (J .GT. MAXSIZ)
   ADDSET = 0
ELSE
   SET(J) = C
   J = J + 1
END IF
RETURN
END

INTEGER FUNCTION AMATCH(LIN, FROM, PAT)

* look for match starting at lin(from)

INTEGER LIN(82), PAT(81)
INTEGER AMATCH, PATSIZ
INTEGER FROM, I, J, OFFSET, STACK
STACK = 0
OFFSET = FROM
J = 1
WHILE (PAT(J) .NE. -2)
   I = OFFSET
   WHILE (LIN(I) .NE. -2)
      EXIT WHILE IF (AMATCH(LIN(I), PAT, J) .EQ. 0)
      PAT(STACK+l) = I - OFFSET
      PAT(STACK+3) = OFFSET
      OFFSET = I
      EXIT WHILE IF (STACK .GT. 0)
      STACK = PAT(STACK+2)
      END WHILE
   END IF
   AMATCH = 0
   RETURN
   I
END IF
J = STACK + 4
OFFSET = PAT(STACK+3) + PAT(STACK+l)
END WHILE
AMATCH = OFFSET
RETURN
END

SUBROUTINE CANT(NAME)

* output message: can't open

INTEGER NAME(18)
CALL PUTCH(32, 3)
CALL PUTLIN(NAME, 3)
CALL REMARK("can't open")
STOP
END
REAL FUNCTION CTOF(STR, I)

* convert character string to real number

INTEGER STR(I)
INTEGER D, DIGITS(11), I, S
REAL VAL, POWER
INTEGER INDEX
EXTERNAL INDEX
DATA DIGITS /48,49,50,51,52,53,54,55,56,57,-2/

* skip white space
WHILE (STR(I) .EQ. 32 .OR. STR(I) .EQ. 9 .AND. STR(I) .NE. -2) I = I + 1
END WHILE

* determine sign; skip over
S = 1
IF (STR(I) .EQ. 43) I = I + 1
ELSE IF (STR(I) .EQ. 45) S = -1
I = I + 1
END IF

* whole part of number
VAL = 0
WHILE (STR(I) .LE. 57 .AND. D = INDEX(DIGITS, STR(I))) VAL = 10 * VAL + D - 1
I = I + 1
END WHILE
CTOF = S * VAL / POWER
RETURN
END

INTEGER FUNCTION CTOI(IN, I)

* convert string at in(i) to integer, increment i
INTEGER IN(81), DIGITS(11)
INTEGER D, I, S, BLANK, EOS
INTEGER INDEX
EXTERNAL INDEX
DATA BLANK/32/, EOS/-2/
DATA DIGITS /48,49,50,51,52,53,54,55,56,57,-2/
WHILE (IN(I) .EQ. BLANK) I = I + 1
END WHILE
S = 1
IF (IN(I) .EQ. 43) I = I + 1
ELSE IF (IN(I) .EQ. 45) S = -1
I = I + 1
END IF
26:     CTOI = 0
27:     WHILE (IN(1) .NE. EOS)
28:       .  D = INDEX(DIGITS, IN(1))
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:     RETURN
37:     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: INTEGER IN(I), I, DIGITS(IN)
7: INTEGER B, D, S
8: INTEGER INDEX, LENGTH
9: EXTERNAL INDEX
10: B = LENGTH(DIGITS)
11: WHILE (IN(I) .EQ. 32)
12:     I = I + 1
13: END WHILE
14: S = 1
15: IF (IN(I) .EQ. 43)
16:     I = I + 1
17: ELSE IF (IN(I) .EQ. 45)
18:     S = -1
19: END IF
20: CTOI2 = 0
21: WHILE (IN(I) .NE. -2)
22:     D = INDEX(DIGITS, IN(1))
23:     IF (D .EQ. 0)
24:        EXIT WHILE
25:     END IF
26:     CTOI2 = B * CTOI2 + D - 1
27:     I = I + 1
28: END WHILE
29: CTOI2 = S * CTOI2
30: RETURN
31: 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+l) into set(j)... from valid
5: INTEGER ESC
6: INTEGER ADDSET, INDEX
7: EXTERNAL INDEX
8: INTEGER I, J, JUNK, LIMIT, MAXSET
9: INTEGER ARRAY(I), SET(MAXSET), VALID(I)
10: LIMIT = INDEX(VALID, ESC(ARRAY, I))
11: K = INDEX(VALID, SET(J))
12: WHILE (K .LE. LIMIT)
13:     JUNK = ADDSET(VALID, K .SET.J, MAXSET)
14:     K = K + 1
15: END WHILE
16: RETURN
17: END
INTEGER FUNCTION EQUAL(STR1, STR2)

*compare 'str1' to 'str2'; return YES if equal, NO if not

INTEGER STR1(100), STR2(100), I
DATA EOS / -2/

I = 1
WHILE (STR1(I) .EQ. STR2(I))
   . IF (STR1(I) .EQ. EOS)
      . . EQUAL = 1
      . . RETURN
   . END IF
   . I = I + 1
END WHILE
EQUAL = 0
RETURN
END

SUBROUTINE ERROR(MSG)

* write error message and stop

CHARACTER MSG*(*)
CALL REMARK(MSG)
STOP
END

INTEGER FUNCTION ESC (ARRAY, I)

*map array(i) into escaped character if possible

INTEGER ARRAY(*), I
INTEGER DIGITS(9)
INTEGER CTOI2, TYPE
DATA DIGITS /48,49,50,51,52,53,54,55,-2/

IF (ARRAY(I) .NE. 64)
   ESC = ARRAY(I)
ELSE IF (ARRAY(I+1) = -2)
   ! @ special at end
   ESC = 64
ELSE
   I = I + 1
   . IF (TYPE(ARRAY(I)) .EQ. 0)
      ! octal ASCII character code
      ESC = CTOI2(ARRAY(I), DIGITS)
   . I = I + 1
   ELSE IF (ARRAY(I) = 66)
      ESC = 8
   ELSE IF (ARRAY(I) = 78)
      ESC = 10
   ELSE IF (ARRAY(I) = 84)
      ESC = 9
   ELSE
      ESC = ARRAY(I)
   END IF
END IF
RETURN
END

INTEGER FUNCTION FILCR(NAME)

* create file NAME, open to successive 1fn

INTEGER NAME(*)
INTEGER UNIT, CLINE(82), IBP, IBUF(82), ISP, ISTK(300)
COMMON /CIN/ UNIT, CLINE, IBP, IBUF, ISP, ISTK

168
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
END IF
CONTINUE
   FILCR = -3
   RETURN
END IF

END

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)
   CALL PUTDEC(UNIT,5,3)
   CALL REMARK(" system reserved lfn")
   FILCRB = -3
   RETURN
END IF

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

END

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")
   RETURN
END IF

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

END

1: *-h- FILOP 830 17 APR 83 23:54:06
2: INTEGER FUNCTION FILOP(NAME)
3:
4: * open file NAME 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: INTEGER TERM(2)
11: CHARACTER*16 FILENM
12: INTEGER EQUAL. OPICC
13: DATA TERM /42.-2/
14:
15: IF (UNIT .LE. 10 .OR. UNIT .GE. 200)
16:   CALL PUTDEC(UNIT,5,3)
17:   CALL REMARK(" system reserved lfn")
18:   RETURN
19:
20: IF (OPICC(NAME.FILENM) .EQ. -3)
21:   FILCRB = -3
22:   RETURN
23: ELSE
24:   OPEN (UNIT,FILE•FILENM,STATUS='NEW',ACCESS='SEQUENTIAL',
25:      FORM='UNFORMATTED'.ERR=1)
26:   UNIT = UNIT + 1
27:   FILCRB = UNIT - 1
28:   RETURN
29:   CONTINUE
30:   FILCRB = -3
31:   RETURN
32:   END IF
33:
34:  END
18:      . FILOP = -3
19:      . RETURN
20:      END IF
21:      IF (EQUAL(NAME,TERM) .EQ. 1)
22:      . FILOP = 3
23:      . RETURN
24:      ELSE IF (OPICC(NAME.FILENM) .EQ. -3)
25:      . FILOP = -3
26:      . RETURN
27:      ELSE
28:      OPEN (UNIT,FILE•FILENM.STATUS•'OLD',ACCESS•'SEQUENTIAL',ERR•l)
29:      UNIT • UNIT + 1
30:      FILOP • UNIT - l
31:      RETURN
32:      CONTINUE
33:      . FILOP = -3
34:      . RETURN
35:      END IF
36:      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(I), DELIM, SET(MAXSET)
11: INTEGER DIGITS(l1), LOWALF(27), UPALF(27)
12: DATA DIGITS /48.49,50.51.52.53.54.55,56,57,-2/
14: DATA UPALF /65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,-2/
15: WHILE (ARRAY(I) .NE. DELIM .AND. ARRAY(I) .NE. -2)
16:      . JUNK • ADDSET(ESC(ARRAY.I),SET,J,MAXSET)
17:      ELSE IF (ARRAY(I) .NE. 45)
18:      . JUNK • ADDSET(ARRAY(I),SET,J,MAXSET)
19:      ELSE
20:      . JUNK • ADDSET(45,SET,J,MAXSET)
21:      ELSE IF (INDEX(DIGITS.SET(J-1)) .GT. 0)
22:      . CALL DODASH(DIGITS.ARRAY.I,SET,J,MAXSET)
23:      ELSE IF (INDEX(LOWALF,SET(J-1)) .GT. 0)
24:      . CALL DODASH(LOWALF,ARRAY,I,SET,J,MAXSET)
25:      ELSE IF (INDEX(UPALF,SET(J-1)) .GT. 0)
26:      . CALL DODASH(UPALF,ARRAY,I,SET,J,MAXSET)
27:      ELSE
28:      . JUNK • ADDSET(45,SET,J,MAXSET)
29:      END IF
30:      I • I + 1
31:      END WHILE
32:      RETURN
33:      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(I), FIELD, DEC
7: REAL FLT
8: INTEGER D, FP, I, J, K, L, WP
9: INTEGER DIGITS(l1)
10: REAL AF, ZERO
11: DATA ZERO /0.0/, DIGITS /48.49,50.51,52.53,54.55,56,57,-2/
12: AF • ABS(FLT)
13: STR(I) • -2
14: I • I + 1
15: RETURN
16: END
IF (AF .EQ. 0.0)  ! check if insufficient room
  L = 2 + DEC
ELSE
  L = INT(DIM ALOG10(AF), ZERO) + 1 + DEC
END IF
IF (L .GT. FIELD)
  WHILE (I .LE. FIELD+1)
    STR(I) = 42
    I = I + 1
  END WHILE
ELSE
  IF (DEC .GT. 0)
    fractional part
    FP = NINT(AF - AINT(AF)) * (10.0 ** DEC)
    FOR J = 1,DEC
      D = MOD(FP, 10)
      STR(I) = DIGITS(D+1)
      FP = FP / 10
      I = I + 1
    END FOR
END IF
IF (DEC .GE. 0)
  STR(I) = 46
  I = I + 1
END IF
pi = 3.1416
WP = INT(AF)  ! whole part
IF (WP .EQ. 0) AND (I .LE. FIELD+1)
  STR(I) = 48
  I = I + 1
ELSE
  DO
    D = MOD(WP, 10)
    STR(I) = DIGITS(D+1)
    WP = WP / 10
    I = I + 1
  UNTIL (WP .EQ. 0)
END IF
IF (FLT .LT. 0.0) sign
  STR(I) = 45
  I = I + 1
ENDIF
FTOC = I - 1  ! reverse order
J = 1
WHILE (J .LT. I)
  K = STR(I)
  STR(I) = STR(J)
  STR(J) = K
  I = I - 1
  J = J + 1
END WHILE
RETURN
END
INTEGER FUNCTION GETCCL(ARG(I), PAT(J))

* expand char class at arg(i) into pat(j)

INTEGER ARG(81), PAT(81)
INTEGER ADDSET
INTEGER I, J, JSTART, JUNK

10: I = I + 1  

IF (ARG(I) .EQ. 126) 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: RETURN
27: END
*-h* GETCH 628 17 APR 83 23:54:15
1: INTEGER FUNCTION GETCH(C, D)
2: * get next (possibly pushed back) character from input
3: INTEGER C, D
4: INTEGER UNIT, CLINE(82), IBP, IBUF(82), ISP, ISTK(300)
5: COMMON /CIN/ UNIT, CLINE, IBP, IBUF, ISP, ISTK
6: INTEGER GETLIN
7: DATA IBP /82/, ISP /0/
8: IF (ISP .GT. 0) THEN
9: C = ISTK(ISP)
10: ISP = ISP - 1
11: ELSE
12: IF (IBP .GT. 81) THEN
13: IF (GETLIN(IBUF,D) .F.Q. -1) THEN
14: C = -1
15: GETCH = -1
16: RETURN
17: END IF
18: IBP = 1
19: END IF
20: END IF
21: C = IBUF(IBP)
22: IF (C .EQ. 10) IBP = 81
23: IBP = IBP + 1
24: END IF
25: GETCH = C
26: RETURN
27: END

*-h* GETLIN 599 17 APR 83 23:54:55
1: INTEGER FUNCTION GETLIN(LIN, D)
2: * read next line from input. strip trailing blanks
3: INTEGER LIN(82), D
4: INTEGER I, LEN
5: CHARACTER BUF(80)
6: READ (D,"(80A1)"!, END=1) (BUF( I ). I=1.80)
7: LEN = 81
8: WHILE (LEN .GT. 1 .AND. BUF(LEN-1) .EQ. ' ') LEN = LEN - 1
9: END WHILE
10: LIN(LEN) = 10 1 end-of-line
11: LIN(LEN+1) = -2 1 end-of-string
12: I = LEN - 1
13: WHILE (I .GT. 0)
14: LIN(I) = ICHAR(BUF(I))
15: I = I - 1
16: END WHILE
17: GETLIN = LEN
18: RETURN
19: 1 CONTINUE
20: GETLIN = -1
21: RETURN
22: END

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

173
L = GETWRD(CLINE, I, ARG)
J = 1
WHILE (ARG(J) .NE. -2)
  EXIT WHILE IF (ARG(J) .EQ. 46) ! ","
  J = J + 1
END WHILE
IF (ARG(J) .EQ. 46)
  J = J + 1
END IF
WHILE (ARG(J) .NE. -2)
  IF (ARG(J) .EQ. OPT)
    GETOPT = 1
  RETURN
  J = J + 1
END WHILE
END IF
K = 1
WHILE (GETWRD(CLINE, I, ARG) .NE. 0)
  IF (ARG(I) .EQ. 47) /* / */
    J = 2
  END IF
  GETOPT = 1
  RETURN
  J = J + 1
END WHILE
END IF
K = K + 1
END WHILE
GETOPT = 0
RETURN
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: RETURN
11: 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(1) .EQ. 32 OR. IN(1) .EQ. 9)
12:  I = I + 1
13: END WHILE
14:
15: J = 1
16: IF (IN(1) .EQ. 34 OR. IN(1) .EQ. 39)
17:  QUOTE = IN(1)
18:  I = I + 1
19:  WHILE (IN(1) .NE. QUOTE .AND. IN(1) .NE. 10 .AND. IN(1) .NE. 9)
20:    OUT(J) = IN(1)
21:    I = I + 1
22:    J = J + 1
23:    END WHILE
24:    I = I + 1
25: ELSE
26:  WHILE (IN(1) .NE. 10 .AND. IN(1) .NE. -2 .AND. IN(1) .NE. 32
27:    & .AND. IN(1) .NE. 9)
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 (SIR(INDEX) .EQ. C)
12: RETURN
13: END IF
14: INDEX = INDEX + 1
15: END WHILE
16: *
17: INDEX = 0
18: RETURN
19: 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
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))+.5) + NINT(.5*(1-ISIGN(1,INT)))
25: END IF
26: *
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: ITOC = I - 1
52: J = 1
53: WHILE (J .LT. I)
54:   K = STR(I)
55:   STR(I) = STR(J)
56:   STR(J) = K
57:   I = I - 1
58:   J = J + 1
59: END WHILE
60: RETURN
61: DE
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: RETURN
13: 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: *
8: INTEGER I, OFFSET
9: *
10: size of class is at offset, characters follow
11: I = OFFSET + PAT(OFFSET)
12: WHILE (I .GT. OFFSET)
13:   IF (C .EQ. PAT(I))
14:     LOCATE = 1
15:   END IF
16:   I = I - 1
17: END WHILE
18: LOCATE = 0
19: RETURN
20: 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
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)

176
**-h- MATCH 331 2 OCT 81 16:10:21
1: INTEGER FUNCTION MATCH(LIN, PAT)
2: * find match anywhere on line
3: 4: INTEGER LIN(82), PAT(81)
5: 6: INTEGER AMATCH
7: 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: RETURN
20: END
21: END

**-h- MPRINT 2492 4 AUG 82 12:36:13
1: SUBROUTINE MPRINT(X, LDX, M, N, F, JOB, D, TITLE)
2: ************************************************************************
3: *print a matrix stored in full mode*
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: b=0 print transpose of matrix
15: b=1 rectangular matrix
16: b=2 square symmetric matrix. only upper triangle is required
17: b=3 square upper triangular matrix. only upper triangle is required
18: TITLE input: character string appearing above matrix
19: 20: END
21: END
22: (7/82)
23: 24: ***********************************************************************
25:
INTEGER LDX, M, N, JOB, D
REAL X(LDX, 1)
CHARACTER *(*) F, TITLE
INTEGER I, J, K
REAL ZERO
LOGICAL TRAN, RECT, SYMM, TRIA
DATA ZERO /0.0/
TRAN = JOB / 10 .GT. 0
RECT = MOD(JOB, 10) .EQ. 1
SYMM = MOD(JOB, 10) .EQ. 2
TRIA = MOD(JOB, 10) .EQ. 3
WRITE (D,) TITLE
IF (RECT)
  IF (TRAN)
    FOR J = 1, N
      WRITE (D, FMT•F) (X(I, J), I = 1, M)
    END FOR
  ELSE
    FOR I = 1, M
      WRITE (D, FMT•F) (X(I, J), J = 1, N)
    END FOR
  END IF
ELSE IF (SYMM)
  FOR K = 1, N
    WRITE (D, FMT•F) (X(I, K), I = 1, K), (X(K, J), J = K+1, N)
  END FOR
ELSE IF (TRIA)
  IF (TRAN)
    FOR J = 1, N
      WRITE (D, FMT•F) (X(I, J), I = 1, J), (ZERO, J = J+1, N)
    END FOR
  ELSE
    FOR I = 1, N
      WRITE (D, FMT•F) (ZERO, J = 1, I-1), (X(I, J), J = I, N)
    END FOR
  END IF
END IF
END IF
RETURN
END

*-- omatch 933 17 APR 83 23:55:08
INTEGER FUNCTION OMATCH(LIN, I, PAT, J)
* try to match a single pattern at pat(j)
INTEGER LIN(82), PAT(81)
INTEGER LOCATE
INTEGER BUMP, I, J
OMATCH = 0
IF (LIN(I) .EQ. -2)
  RETURN
BUMP = -1
IF (PAT(J) .EQ. 97) THEN
  IF (LIN(I) = PAT(J+1))
    BUMP = 1
ELSE IF (PAT(J) .EQ. 37) THEN
  IF (LIN(I) .NE. 10)
    BUMP = 1
ELSE IF (PAT(J) .EQ. 36) THEN
  IF (LIN(I) .EQ. 10)
    BUMP = 0
ELSE IF (PAT(J) .EQ. 63) THEN
  IF (LIN(I) .NE. 10)
    BUMP = 1
ELSE IF (PAT(J) .EQ. 6) THEN
  IF (LOCATE(LIN(I), PAT, J+1) .EQ. 1)
    BUMP = 1
ELSE IF (PAT(J) .EQ. 110) THEN
  IF (LIN(I) .NE. 10 .AND. LOCATE(LIN(I), PAT, J+1) .EQ. 0)
    BUMP = 1
ELSE
  CALL ERROR(" in omatch: can't happen")
END IF
RETURN
IF (BUMP .GE. 0) THEN
  I = I + BUMP
  OMATCH = 1
END IF
RETURN
END

INTEGER FUNCTION OPICC(NAME, FILE)

* convert internally mapped area name to HARRIS form for OPEN

INTEGER NAME(l)
CHARACTER*16 FILE
INTEGER l, J
CHARACTER*4 QUALC.
CHARACTER*8 AREA
INTEGER TYPE

* initialize
QUALN = ' ' QUALC = ' ' AREA = ' '
I = 1 J = 1

* name beginning with STAR or DIGIT means there is a qualifier
IF (TYPE(NAME(l)) .EQ. 0) ! system qualifier
  QUALN = '0000'
  QUALC = 'SYST'
  J = 2
ELSE IF (TYPE(NAME(l)) .EQ. 42)
  I = J
  WHILE (TYPE(NAME(I)) .NE. 65)
    WHILE (NAME(I) .EQ. -2 .AND. I .GT. 5)
      OPICC = -3
      RETURN
    END IF
    I = I + 1
  END WHILE
  WHILE (NAME(I) .NE. 42)
    QUALC(I-J+l:I-J+l) • CHAR(NAME(I))
    I = I + 1
  END WHILE
  J = I
ELSE IF (TYPE(NAME(l)) .EQ. -2 .AND. I .GT. 5)
  I = 1
  WHILE (NAME(I) .EQ. -2 .OR. I .GT. 9)
    OPICC = -3
    RETURN
  END IF
  I = I - 1
  OPICC = -3
  RETURN
END IF

* copy area name
WHILE (NAME(I) .NE. -2)
  AREA(I-J+l:I-J+l) • CHAR(NAME(I))
  I = I + 1
END WHILE
FILE = QUALN // QUALC // AREA
**SUBROUTINE PACK(BYTES, WORDS)**

1: * pack 3 characters into HARRIS 24-bit words

6: INTEGER BYTES(81), WORDS(27), PW, PB
7: PB = 1
8: WHILE (PB .LE. 81)
   9:   EXIT WHILE IF (BYTES(PB) .EQ. -2)
10:   PW = (PB + 2) / 3
11:   WORDS(PW) = BYTES(PB) .SHIFT. 16
12:   PB = PB + 1
13:   EXIT WHILE IF (BYTES(PB) .EQ. -2)
14:   WORDS(PW) = WORDS(PW) .OR. (BYTES(PB) .SHIFT. 8)
15:   PB = PB + 1
16:   EXIT WHILE IF (BYTES(PB) .EQ. -2)
17:   WORDS(PW) = WORDS(PW) .OR. BYTES(PB)
18:   PB = PB + 1
19: END WHILE
20: RETURN
21: END

**FUNCTION PATSIZ(PAT, N)**

6: INTEGER PAT(81)
7: INTEGER N
8: IF (PAT(N) .EQ. 97)
9:   PATSIZ = 2
10: ELSE IF (PAT(N) .EQ. 37 .OR. PAT(N) .EQ. 36 .OR. PAT(N) .EQ. 63)
11:   PATSIZ = 1
12: ELSE IF (PAT(N) .EQ. 91 .OR. PAT(N) .EQ. 110)
13:   PATSIZ = PAT(N+1) + 2
14: ELSE IF (PAT(N) .EQ. 42)
15:   PATSIZ = 4
16: ELSE
17:   CALL ERROR(" in patsiz: can't happen")
18: END IF
19: RETURN
20: END

**SUBROUTINE PRIMIO(D)**

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: LEVEL = 1
12: UNIT = D
13: BACKSPACE 0
14: CALL GETLIN(CLINE,0)
15: * read command line
16: RETURN
17: END

180
SUBROUTINE PUTCH(C, D)

INTEGER C, D
INTEGER OBUF(BO), OBP, TAJIS(BO)
DATA NL /10. -2/

IF (C .EQ. 10 .OR. OBP .GE. 80) THEN
   CALL PUTLIN(NL, D)
ELSE
   OBP = OBP + 1
   OBUF(OBP) = C
END IF
RETURN
END

SUBROUTINE PUTDEC(N, W, F)

INTEGER CHARS(8)
INTEGER ITOC
INTEGER F, I, N, ND, W
ND = ITOC(N, CHARS, 8)
I = ND + 1
WHILE (I .LE. W)
   CALL PUTCB(32, F)
   I = I + 1
END WHILE
I = 1
WHILE (I .LE. ND)
   CALL PUTCH (CHARS(I), F)
   I = I + 1
END WHILE
RETURN
END

SUBROUTINE PUTFLT(FLT, FIELD, DEC, D)

INTEGER FIELD, DEC, D
REAL FLT
INTEGER I, ND, STRC16
INTEGER FTOC
ND = FTOC(FLT, STRC16, DEC)
FOR I = ND+1, FIELD
   CALL PUTCB(32, D)
END FOR
FOR I = 1, ND
   CALL PUTCH(STRC16(I), D)
END FOR
RETURN
END

SUBROUTINE PUTLIN(LIN, D)

INTEGER LIN(*), D
INTEGER OBUF(BO), OBP, TAJIS(BO)
COMMON /COUT/ OBUF, OBP, TAJIS
INTEGER I, J
CHARACTER BUF(BO)

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.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: RETURN
29: END

1: -*b- 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: CALL PUTCH(32.3)
10: DO
11: CALL PUTCH(32.3)
12: UNTIL (TABS(OBP+1) .EQ. 1 .OR. OBP .EQ. 0)
13: RETURN
14: END

1: -*b- 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: L = LEN(MSG)
10: FOR I = 1, L
11: BUF(I) = ICHAR(MSG(I:1))
12: END FOR
13: BUF(L+1) = -2
14: I = 1
15: WHILE (I .LE. L)
16: IF (BUF(I) .EQ. 64) THEN
17: CALL PUTCH(ESC(BUF(I),3)
18: ELSE
19: CALL PUTCH(BUF(I),3)
20: END IF
21: I = I + 1
22: END WHILE
23: CALL PUTCH(32.3)
24: FOR I = 1, 3
25: CALL PUTCH(42, 3)
26: CALL PUTCH(10, 3)
27: RETURN
28: END

1: -*b- 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: 182
INTEGER FROM(100), TO(100)
INTEGER I, J, K1, K2
K2 = J
K1 = I
WHILE (FROM(K1) .NE. -2)
  TO(K2) = FROM(K1)
  K2 = K2 + 1
  K1 = K1 + 1
END WHILE
TO(K2) = -2
RETURN
END

SUBROUTINE SETTAB(LIST)
INTEGER LIST(*)
INTEGER OBUF(80), OBP, TABS(80)
COMMON /COUT/ OBUF, OBP, TABS
INTEGER I, J
IF (LIST(2) .EQ. 43) THEN
  I = 1
  WHILE (I .LT. LIST(1))
    TABS(I) = 0
    I = I + 1
  END WHILE
  TABS(I) = 1
  I = I + 1
  WHILE (I .LT. 80)
    IF (MOD(I - LIST(1), LIST(3)) .EQ. 0) THEN
      TABS(I) = 1
    ELSE
      TABS(I) = 0
    END IF
    I = I + 1
  END WHILE
ELSE
  J = 1
  I = 1
  WHILE (I .LT. 80)
    IF (I .EQ. LIST(J)) THEN
      TABS(I) = 1
      J = J + 1
    ELSE
      TABS(I) = 0
    END IF
    I = I + 1
  END WHILE
END IF
RETURN
END

INTEGER FUNCTION STCLOS(PAT, J, LASTJ, LASTCL)
INTEGER PAT(81)
INTEGER ADDSET
INTEGER J, JP, JT, JUNK, LASTCL, LASTJ
JP = J - 1
WHILE (JP .GE. LASTJ) ! make a hole
  JT = JP + 4
  JUNK = ADDSET(PAT(JP), PAT(JT, 81))
  JP = JP - 1
END WHILE
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.
* REAL FUNCTION INNERP(U, INCU, V, INCV, N)
  2: ! n
  3: L
  4: REAL U(*), V(*)
  5: INTEGER N, IU, INCU, IV, INCV
  6: REAL B
  7: B = 0.0
  8: IF (N .EQ. 0)
  9:  INNERP = B
10:  RETURN
11: END IF
12: IF (INCU .EQ. 1 .AND. INCV .EQ. 1)
13:  ! contiguous data
14:  FOR I=1,N
15:    B = B + (U(I) * V(I))
16:  END FOR
17: ELSE
18:  ! non-contiguous data
19:  IU = 1
20:  IV = 1
21:  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
22:  IF (INCV .LT. 0) IV = (-N+1) * INCV + 1
23:  FOR I=1,N
24:    AX = ABS(X(I))
25:    IF (CASE .EQ. 1)
26:      IF ( IX .LE. NMAX)
27:        SUM = SUM + AX
28:        XMAX = MAX(XMAX, AX)
29:      END IF
30:  END FOR
31:  END IF
32:  INNERP = B
33: RETURN
34: END

* REAL FUNCTION L2NORM(X, INCX, N)
  2: INTEGER INCX, N
  3: REAL X(*)
  4: INTEGER CASE, IX, NMAX
  5: REAL AX, HICUT, HITES!, LOCUT /4.44E-16/, SUM, XMAX
  6: DATA LOCUT /4.44E-16/, HICUT /1.304E+19/
  7: IF (N .LE. 0)
  8:  L2NORM = 0.0
  9:  RETURN
10: END IF
11: SUM = 0.0
12: XMAX = 0.0
13: CASE = 1
14: IX = 1
15: NMAX = N * INCX
16: WHILE (IX .LE. NMAX)
17:  AX = ABS(X(IX))
18:  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 + (X(IX) / XMAX) ** 2
46: ELSE
47: ELSE
48: SUM = SUM + (X(IX) / XMAX) ** 2
49: END IF
50: IX = IX + INCX
51: END IF
52: END IF
53: END IF
54: END IF
55: ELSE IF (CASE .EQ. 3)
56: FOR J = IX,NMAX,INCX
57: AX = ABS(X(J))
58: IF (AX .GE. HITEST)
59: CASE = 4
60: ELSE
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: IF (CASE .EQ. 3)
69: L2NORM = SQRT(SUM)
70: RETURN
71: END IF
72: ELSE IF (CASE .EQ. 4)
73: IF (AX .GT. XMAX)
74: SUM = 1.0 + SUM * (XMAX / X(IX)) ** 2
75: ELSE
76: SUM = SUM + (X(IX) / XMAX) ** 2
77: END IF
78: IX = IX + INCX
79: END IF
80: END WHILE
81: L2NORM = XMAX * SQRT(SUM)
82: RETURN
83: 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: *
6: INTEGER LDY, LDU, LDV, L, M, N
7: REAL Y(LDY,*), U(LDU,*), V(LDV,*)
8: INTEGER I, J
9: FOR J = 1,N
10: CALL VCOPY(Y(1,J),1,0.0,0,L)
11: FOR I = 1,M
12: CALL VSMA2(Y(1,J),1,0.0,0,L)
13: CALL VSMAX(Y(1,J),1,0.0,Y(1,J),1,V(1,J),1,U(1,I),1,0.0,L)
14: END FOR
15: END FOR
16: RETURN
17: END
18: RETURN
19: END
SUBROUTINE MPRDT(Y, LDY, U, LDU, V, LDV, L, M, N)

* matrix inner product Y = UV
* column oriented algorithm

INTEGER LDY, LDU, LDV, L, M, N
REAL Y(LDY,*) , U(LDU,*) , V(LDV,*)
INTEGER I, J
FOR J = 1, N
  CALL VCOPY(Y(1,J),1,0.0,0,L)
FOR I = 1, M
  CALL VSMA2(Y(1,J),1,1.0,Y(1,J),1, V(J,I), U(1,I),1,0.0,L)
END FOR
END FOR
RETURN
END

SUBROUTINE MSVDC(X,LDX,N,P,E,U,LDU,V,LDV,WORK,JOBL,INFO)

* Reduce a real n by n matrix X by orthogonal transformations U, V
* to diagonal form.  the diagonal elements s(i) are the singular
* values of X; the columns of U are the left singular vectors; and
* the columns of V are the right singular vectors

INTEGER LDX, N, P, LDU, LDV, JOB, INFO
REAL X(LDX,*) , E(*), U(LDU,*) , V(LDV,*)
INTEGER CASE, ITER, J, JOBU, K, KK, L, LL, LMS, LNP, LNP1, LRT, LRT1
INTEGER LU, M, MAXI, MM, MM1, MP1, NCT, NCTP1, NCU, NRT, NRTP1
INTEGER ZERO

REAL T, T1, TEST, ZTEST
LOGICAL WANTU, WANTV
INTEGER MAXO, MINO, MOD
REAL ABS, AMAX!, INNERP, SCALE, SHIFT, SI, SL, SM, SMM
DATA ZERO / 0/
MAXIT = 30  ! maximum number of iterations

* decide what to do
JOBU = MOD(JOB,100) / 10
NCU = N

 ***************
59: IF (JOBU .GT. 1) NCU = MINO(N,P)
60: WANTU = JOBU .NE. 0
61: WANTV = MOD(JOB,10) .NE. 0
62: * reduce X to bidiagonal form storing diagonal element in s and
63: super-diagonal element in e
64: INFO = 0
65: NCT = MINO(N-1,P)
66: NRT = MAXO(ZERO,MINO(P-2,N))
67: LU = MAXO(NCT,NRT)
68: IF (LU .GE. 1)
69: FOR L = 1,LU
70: LPI = L + 1
71: IF (L .LE. NCT)
72: compute transformation for 1th column and place 1th diagonal in s(l)
73: 74: S(L) = L2NORM(X(L,L),1,N-L+1)
75: IF (S(L) .NE. 0.0)
76: IF (X(L,L) .NE. 0.0) S(L) = SIGN(S(L),X(L,L))
77: CALL VSCALE(X(L,L),1,1.0/S(L),N-L+1)
78: X(L,L) = 1.0 + X(L,L)
79: END IF
80: S(L) = -S(L)
81: END IF
82: IF (P .GE. LPI)
83: FOR J = LPI,P
84: apply transformation
85: T = -INNERP(X(L,L),1,X(L,J),1,N-L+1) / X(L,L)
86: CALL VSMA2(X(L,J),1,T,X(L,L),1,0.0,X(L,J),1,0.0,N-L+1)
87: END IF
88: END FOR
89: place 1th row of X into e for subsequent calculation of new transform
90: E(J) = X(L,J)
91: END FOR
92: END IF
93: IF (WANTU .AND. L .LE. NCT)
94: place transformation in U for subsequent back multiplication
95: CALL VCOPY(U(L,L),1,X(L,L),1,N-L+1)
96: END IF
97: END IF
98: IF (WANTV)
99: place transformation in V for subsequent back multiplication
100: CALL VCOPY(V(LPI,L),1,E(LPI),1,P-L)
101: END IF
102: END IF
103: compute the 1th row transformation and place the 1th superdiagonal
104: in e(l)
105: E(L) = L2NORM(E(LPI),1,P-L)
106: IF (E(L) .NE. 0.0)
107: IF (E(LPI) .NE. 0.0) E(L) = SIGN(E(L),E(LPI))
108: CALL VSMA2(E(LPI),1,1.0/E(L),P-L)
109: E(LPI) = 1.0 + E(LPI)
110: END IF
111: E(L) = -E(L)
112: IF (LPI .LE. N .AND. E(L) .NE. 0.0)
113: apply transformation
114: CALL VCOPY(WORK(LPI),1,0.0,0,N-L)
115: FOR J = LPI,P
116: CALL VSMA2(WORK(LPI),1,0.0,WORK(LPI),1,E(J),X(LPI,J),1,0.0,N-L)
117: END FOR
118: FOR J = LPI,P
119: CALL VSMA2(X(LPI,J),1,0.0,X(LPI,J),1,-E(J)/E(LPI),WORK(LPI),1,
120: 0.0,N-L)
121: END FOR
122: END IF
123: IF (WANTV)
124: place the transformation in V for subsequent back multiplication
125: CALL VCOPY(V(LPI,L),1,E(LPI),1,P-L)
126: END IF
127: END IF
128: END FOR
129: END IF
130: set up final bi-diagonal matrix of order m
131: M = MINO(P,N-1)
132: NCPI = NCT + 1
133: NRTPI = NRT + 1
134: IF (NCPI .LT. P) S(NCPI) = X(NCPI,NCPI)
135: IF (NRTPI .LT. M) S(NRTPI) = 0.0
136: IF (NRTPI .LT. M) E(NRTPI) = X(NRTPI,M)
137: M = MINO(P,N-1)
138: NCPI = NCT + 1
139: NRTPI = NRT + 1
140: IF (NCPI .LT. P) S(NCPI) = X(NCPI,NCPI)
141: IF (NRTPI .LT. M) S(NRTPI) = 0.0
142: IF (NRTPI .LT. M) E(NRTPI) = X(NRTPI,M)
143: 189
E(M) = 0.0

* generate U

IF (WANTU)
  FOR J = NCTP1, NCU
    CALL VCOPY(U(1,J),1,0.0,0,N)
  END FOR
ENDIF

IF (NCT .GE. NCTP1)
  FOR J = NCTP1, NCU
    U(J,J) = 1.0
  END FOR
ENDIF

IF (NCT .GE. 1)
  FOR LL = 1, NCT
    L = NCT - LL + 1
    IF (S(L) .NE. 0.0)
      LPl = L + 1
      IF (NCU .GE. LPl)
        FOR J = LPl, 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),1,0.0,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)
      LMl = L - 1
      IF (LMl .GT. 0) CALL VCOPY(U(l,L),1,0.0,0,LMl)
    ELSE
      CALL VCOPY(U(l,L),1,0.0,0,N)
      U(L,L) = 1.0
    END IF
  END FOR
ENDIF
ENDIF

* generate V

IF (WANTV)
  FOR LL = 1,P
    L = P - LL + 1
    IF (L .LE. NRT )
      IF (S(L) .NE. 0.0)
        FOR J = LPl, P
          T = -INNERP(V(LPl,L),1,V(LPl,J),1,P-L) / V(LPl,L)
          CALL VSMA2(V(LP1,J),1,T,V(LP1,L),1,0.0,V(LP1,J),1,0.0,P-L)
        END FOR
      END IF
    END IF
  END FOR
ENDIF

main iteration loop for singular values

MM = M
ITER = 0
WHILE (M .GT. 0)
  m = 0 means all singular values have been found
  inspect for negligible elements in s and e, on completion variables
  case and 1 are set as follows
    case=1 s(m) and e(1-1) are negligible & 1<m
    case=2 s(1) is negligible & 1=m
    case=3 e(1-1) is negligible & 1<m & s(1)...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

190
WHILE (L .LE. 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
        EXIT FOR IF (LS .NE. L+1)
        TEST = TEST + ABS(S(LS))
        IF (TEST .EQ. TEST)
            S(LS) = 0.0
            EXIT FOR
        END IF
    END FOR
    IF (LS .EQ. H)
        CASE = 3
    ELSE IF (LS .EQ. H-1)
        CASE = 2
    ELSE
        CASE = 1
    END IF
    L = L + 1
END IF

* perform task indicated by case

IF (CASE .EQ. 1)
    F = E(L-1)
    E(L-1) = 0.0
    FOR KK = L,H
        T1 = S(K)
        CALL SROTG(T1,F,CS,SI)
        S(K) = T1
        F = -SI * E(K)
        E(K) = CS * E(K)
    END FOR
    IF (WANTU) CALL VROT(U(1,K),1,U(1,L-1),1,CS,SI,N)
END IF

ELSE IF (CASE .EQ. 2)

ELSE IF (CASE .EQ. 3)

* deflate negligible s(m)

MM1 = M - 1
F = E(M-1)
E(M-1) = 0.0
FOR KK = L,MM1
    K = MM1 - KK + L
    TI = S(K)
    CALL SROTG(T1,F,CS,SI)
    S(K) = T1
    F = -SI * E(K-1)
    E(K-1) = CS * E(K-1)
END FOR
IF (WANTY) CALL VROT(V(1,K),1,V(1,H),1,CS,SI,P)
END IF
ELSE IF (CASE .EQ. 2)

ELSE IF (CASE .EQ. 3)

* split at negligible s(l)

F = E(L-1)
E(L-1) = 0.0
FOR K = L,M
    TI = S(K)
    CALL SROTG(T1,F,CS,SI)
    S(K) = TI
    F = -SI * E(K)
    E(K) = CS * E(K)
END FOR
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

SCALE = AMAX1(ABS(S(M)),ABS(S(M-1)),ABS(E(M-1)),ABS(S(L)),
          ABS(S(L+1))),

& SCALE = SM / SCALE

SM = S(M) / SCALE

SM1 = S(M-1) / SCALE

191
EMDL = E(M-1) / SCALE
SL = S(L) / SCALE
EL = E(L) / SCALE

B = ((EMDL + SM) * (EMDL - SM) + EMDL ** 2) / 2.0
C = (SM * EMDL) ** 2
SHIFT = 0.0

IF (B .NE. 0.0 .OR. C .NE. 0.0)
SHIFT = SQRT(B**2+C)
IF (B .LT. 0.) SHIFT = -SHIFT
SHIFT = C / (B + SHIFT)
END IF

F = (SL + SM) * (SL - SM) - SHIFT
G = SL*EL

chase zeroes
MM = M - 1
FOR K = L,MM
CALL SROTG(F,G,CS,SI)
IF (K .NE. L) E(K-1) = F
F = CS * S(K) + SI * E(K)
E(K) = -SI * S(K) + CS * E(K)
G = SI * S(K+1)
END FOR
IF
F = (WANTV) CALL VROT(V(1,K),1,V(1,K+1),1,CS,SI,F)
END IF

* convergence
make the singular value positive and order
IF (S(L) .LT. 0.0)
S(L) = -S(L)
IF (WANTV) CALL VSCALE(V(l,L),1,-1.0,P)
END IF
WHILE (L .NE. MM)
T = S(L)
S(L) = S(L+1)
S(L+1) = T
IF (WANTV .AND. L .LT. P) CALL VSWAP(V(l,L),1,V(l,L+1),1,P)
IF (WANTU .AND. L .LT. N) CALL VSWAP(U(l,L),1,U(l,L+1),1,N)
L = L + 1
END WHILE
ITER = 0
M = M - 1
END WHILE
INFO = M
RETURN
END
SUBROUTINE SROTG(A, B, C, S)

* construct Givens plane rotation

REAL A, B, C, S
REAL ROE, SCALE, R, Z

ROE = B
IF (ABS(A) .GT. ABS(B)) ROE = A
SCALE = ABS(A) + ABS(B)
IF (SCALE .EQ. 0.0) SCALE = SQRT(ABS(A)**2 + ABS(B)**2)
C = A / R
S = B / R
R = SCALE * SIGN(1.0, ROE) * R
C = A / R
S = B / R
Z = 1.0
IF (ABS(A) .GT. ABS(B)) Z = S
IF (ABS(B) .GT. ABS(A) .AND. C .NE. 0.0) Z = 1.0 / C
A = R
B = Z
RETURN
END

SUBROUTINE SRTRT(X, T, N)

* sort elements of real vector in increasing algebraic order
* record permutations in T

C.A.R. Hoare's Quicksort algorithm

INTEGER N, T(*)
REAL X(*)
INTEGER I, J, K, LV(20), P, UV(20)
REAL PIVOT, S

LV(1) = 1
UV(1) = N
P = 1
WHILE (P .GT. 0)
IF (LV(P) .GE. UV(P)) P = P - 1
ELSE
I = LV(P) - 1
J = UV(P)
PIVOT = X(J)
WHILE (I .LT. J)
I = I + 1
WHILE (X(I) .LT. PIVOT)
I = I + 1
END WHILE
J = J - 1
WHILE (J .GT. I)
EXIT WHILE IF (X(J) .LE. PIVOT)
J = J - 1
END WHILE
IF (I .GT. J) S = X(J)
X(J) = X(I)
X(I) = S
K = T(J)
T(J) = T(I)
T(I) = K
END IF
END WHILE
J = UV(P)
S = X(J)
X(J) = X(I)

RETURN
END

17: RETURN
18: 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) SCALE = SQRT(ABS(A)**2 + ABS(B)**2)
13: C = A / R
14: S = B / R
15: R = SCALE * SIGN(1.0, ROE) * R
16: C = A / R
17: S = B / R
18: Z = 1.0
19: IF (ABS(A) .GT. ABS(B)) Z = S
20: IF (ABS(B) .GT. ABS(A) .AND. C .NE. 0.0) Z = 1.0 / C
21: A = R
22: B = Z
23: RETURN
24: 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)) P = P - 1
18: ELSE
19: I = LV(P) - 1
20: J = UV(P)
21: PIVOT = X(J)
22: WHILE (I .LT. J)
23: I = I + 1
24: WHILE (X(I) .LT. PIVOT)
25: I = I + 1
26: END WHILE
27: EXIT WHILE IF (X(J) .LE. PIVOT)
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 .GT. J) S = X(J)
34: X(J) = X(I)
35: X(I) = S
36: K = T(J)
37: T(J) = T(I)
38: T(I) = K
39: END IF
40: END WHILE
41: J = UV(P)
42: S = X(J)
43: X(J) = X(I)
44: RETURN
45: END

193
SUBROUTINE SRTPI (A, T, JOB, N)

INTEGER A(*), T(*), JOB, N

IF (N .EQ. 1)
  RETURN

FOR I = 1, N

IF (T(I) .LT. 0)
  T(I) = -T(I)
  J = T(I)
  WHILE (J .NE. I)
    S = A(J)
    A(J) = A(I)
    A(I) = S
    T(J) = -T(J)
    J = T(J)
  END WHILE
END IF
END FOR

END IF
END FOR
RETURN
END
SUBROUTINE SRTPR(A, T, JOB, N)
   * rearrange elements of real A by permutation vector T generated by
   
   * job == 0  A[T[k]] moved to A[k] :gather
   * job == 0  A[k] moved to A[T[k]] :scatter

   INTEGER T(*), JOB, N
   REAL A(*)

   INTEGER I, J, K
   REAL S

   IF (N .EQ. 1) RETURN

   FOR I = 1, N
      T(I) = -T(I)
   END FOR

   IF (JOB .EQ. 0)
      forward permutation
      FOR I = 1, N
         IF (T(I) .LT. 0)
            J = I
            T(J) = -T(J)
            K = T(J)
            WHILE (T(K) .LT. 0)
               S = A(J)
               A(J) = A(K)
               A(K) = S
               T(K) = -T(K)
               J = K
               K = T(K)
         END WHILE
      END IF
   END FOR

   ELSE
      backward permutation
      FOR I = 1, N
         IF (T(I) .LT. 0)
            T(I) = -T(I)
            J = T(I)
            WHILE (J .NE. I)
               S = A(I)
               A(I) = A(J)
               A(J) = S
               T(J) = -T(J)
               J = T(J)
            END WHILE
      END IF
   END IF

   RETURN
END

REAL FUNCTION SVMAX(U, INCU, INDEX, N)
   * return value and index of maximum in U

   INTEGER INCU, INDEX, N
   REAL U(*)

   INTEGER I, IU
   REAL A

   IF (N .LE. 0) SVMAX = 0.0
   RETURN
END
21: * contiguous data
22:  FOR I = 2,N
23:   IF (U(I) .GT. A)
24:     A = U(I)
25:     INDEX = I
26:   END IF
27:  END FOR
28: ELSE
29:  non-contiguous data
30:  IF (U(1) .GT. A)
31:    A = U(1)
32:    INDEX = 1
33:  END IF
34:  IF (INCU .EQ. 1)
35:    FOR I = 2,N
36:      IF (U(I) .LT. A)
37:        A = U(I)
38:        INDEX = I
39:      END IF
40:    END FOR
41:  END IF
42:  SVMIN = A
43:  RETURN
44:  END

1: * SVMIN 642 30 Aug 82 13:02:37
2: REAL FUNCTION SVMIN(U, INCU, INDEX, N)
3: * return value and index of minimum in U
4: INTEGER INCU, INDEX, N
5: REAL U(*), A
6: IF (N .LE. 0)
7: SVMIN = 0
8: RETURN
9: END IF
10: A = U(1)
11: INDEX = 1
12: IF (INCU .EQ. 1)
13: SVMIN = 0
14: RETURN
15: END IF
16: SVMIN = A
17: RETURN
18: END
REAL FUNCTION SVSHAB(U, INCU, N)

* sum(U[k])

INTEGER INCU, N
REAL U(*)
INTEGER I, IU
REAL C

IF (INCU .EQ. 1)
* contiguous data

C = 0.0
FOR I = 1,N
C = C + ABS(U(I))
END FOR

ELSE
* non-contiguous data

IU = 1
C = 0.0
FOR I = 1,N
C = C + ABS(U(IU))
IU = IU + INCU
END FOR

ENDIF

SVSHAB = C
RETURN

END

REAL FUNCTION SVSUM(U, INCU, N)

* sum(U[k])

INTEGER INCU, N
REAL U(*)
INTEGER I, IU
REAL C

IF (INCU .EQ. 1)
* contiguous data

C = 0.0
FOR I = 1,N
C = C + U(I)
END FOR

ELSE
* non-contiguous data

IU = 1
C = 0.0
FOR I = 1,N
C = C + U(IU)
IU = IU + INCU
END FOR

ENDIF

SVSUM = C
RETURN

END
SUBROUTINE VCOPY(Y, INCY, U, INCU, N)

INTEGER INCY, INCU, N
REAL Y(*), U(*)
INTEGER I, IY, IU

IF (INCY .EQ. 1 .AND. INCU .EQ. 1)
  CONTIGUOUS DATA
  FOR I = 1, N
    Y(I) = U(I)
  END FOR
ELSE
  NON-CONTIGUOUS DATA
  IY = 1
  IU = 1
  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
  FOR I = 1, N
    Y(IY) = U(IU)
    IY = IY + INCY
    IU = IU + INCU
  END FOR
END IF
RETURN
END

SUBROUTINE VDIV(Y, INCY, A, U, INCU, B, V, INCV, C, N)

INTEGER INCY, INCU, INCV, N
REAL Y(*), U(*), V(*), A, B, C
INTEGER I, IY, IU, IV

IF (N .LE. 0) RETURN
IF (INCY .EQ. 1 .AND. INCU .EQ. 1 .AND. INCV .EQ. 1)
  CONTIGUOUS DATA
  FOR I = 1, N
    Y(I) = (A / (U(I) + B)) * (V(I) + C)
  END FOR
ELSE
  NON-CONTIGUOUS DATA
  IY = 1
  IU = 1
  IV = 1
  IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
  IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
  IF (INCV .LT. 0) IV = (-N+1) * INCV + 1
  FOR I = 1, N
    Y(IY) = (A / (U(IU) + B)) * (V(IV) + C)
    IY = IY + INCY
    IU = IU + INCU
    IV = IV + INCV
  END FOR
END IF
RETURN
END
SUBROUTINE VFIX(Y, INCY, U, INCU, N)

* convert vector to integer
INTEGER Y(*), INCY, INCU, N
REAL U(*)
INTEGER I
INTEGER INT

IF (N .LE. 0) RETURN
IF (INCY .EQ. 1 .AND. INCU .EQ. 1) * contiguous data
FOR I = 1, N
Y(I) = INT(U(I))
END FOR
ELSE * non-contiguous data
IY = 1
IU = 1
IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
FOR I = 1, N
Y(IY) = INT(U(IU))
IY = IY + INCY
IU = IU + INCU
END FOR
END IF
RETURN
END

SUBROUTINE VFLOAT(Y, INCY, U, INCU, N)

* convert vector to real
REAL Y(*)
INTEGER U(*), REAL FLOAT
INTEGER I
INTEGER INCY, INCU, N

IF (N .LE. 0) RETURN
IF (INCY .EQ. 1 .AND. INCU .EQ. 1) * contiguous data
FOR I = 1, N
Y(I) = FLOAT(U(I))
END FOR
ELSE * non-contiguous data
IY = 1
IU = 1
IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
FOR I = 1, N
Y(IY) = FLOAT(U(IU))
IY = IY + INCY
IU = IU + INCU
END FOR
END IF
RETURN
END
SUBROUTINE VINDEX(Y, INCY, A, B, U, INCU, N)

* linear weighting of vector U (integer version)

Y[k] = (a + bk) * U[k]

INTEGER Y(*), INCY, A, B, U(*), INCU, N

IF (N .LE. 0) RETURN

IF (INCY .EQ. 1 .AND. INCU .EQ. 1)

FOR I = 1, N
    Y(I) = (A + (I - 1) * B) * U(I)
END FOR

ELSE

IY = 1
IU = 1
IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
FOR I = 1, N
    Y(IY) = (A + (I - 1) * B) * U(IU)
    IU = IU + INCU
END FOR

END IF

RETURN
END

SUBROUTINE VMUL(Y, INCY, A, U, INCU, B, V, INCV, C, N)

* Y[k] = a * (b+U[k]) * (c+V[k])

REAL Y(*), U(*), V(*), A, B, C
INTEGER I, INCY, INCU, INCV, IY, IU, IV, N

IF (N .LE. 0) RETURN

IF (INCY .EQ. 1 .AND. INCU .EQ. 1 .AND. INCV .EQ. 1)

FOR I = 1, N
    Y(I) = A * (B + U(I)) * (C + V(I))
END FOR

ELSE

IY = 1
IU = 1
IV = 1
IF (INCY .LT. 0) IY = (-N+1) * INCY + 1
IF (INCU .LT. 0) IU = (-N+1) * INCU + 1
IF (INCV .LT. 0) IV = (-N+1) * INCV + 1
FOR I = 1, N
    Y(IY) = A * (U(IU) + B) * (V(IV) + C)
    IU = IU + INCU
    IV = IV + INCV
END FOR

END IF

RETURN
END
1: *---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) END
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:   : FOR I = 1, N
29:   : Y(IY) = A / (U(IU) + B)
30:   : IY = IY + INCY
31:   : IU = IU + INCU
32:   : END FOR
33: 34:   ELSE
35: 36:  RETURN
37: END
38: END

1: *---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) END
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:   : IX = 1
27:   : IT = 1
28:   : IF (INCY .LT. 0) IX = (-N+1) * INCX + 1
29:   : IF (INCU .LT. 0) IT = (-N+1) * INCU + 1
30:   : FOR I = 1, N
31:   : STEMP = C * X(IX) + S * Y(IY)
32:   : Y(IY) = C * Y(IY) - S * X(IX)
33:   : X(IX) = STEMP
34:   : IX = IX + INCX
35:   : Y(IY) = Y(IY) - IX
36:   : IX = IX + INCX
37:   : IT = IT + INCU
38:   : END FOR
39: 40:  END IF
41: 42: END

201
SUBROUTINE VSCALE(Y, INCY, A, N)

Y[k] = a * Y[k]

INTEGER INCY, N
REAL Y(*), A
CALL VSMA1(Y, INCY, A, Y, INCY, 0.0, N)
RETURN
END

SUBROUTINE VSMA1(Y, INCY, A, U, INCU, B, N)

Y[k] = a * U[k] + b

INTEGER INCY, INCU, N
REAL Y(*), U(*), A, B
INTEGER I, IY, IU
IF (N .LE. 0) RETURN
IF (INCY .EQ. 1 .AND. INCU .EQ. 1)

FOR I = 1, N
  Y(I) = A * U(I)
END FOR
ELSE

IY = 1
IU = 1
IF (INCY .LT. 0) IY = (-N+1) * INCY
IF (INCU .LT. 0) IU = (-N+1) * INCU
FOR I = 1, N
  Y(IY) = A * U(IU)
  IY = IY + INCY
  IU = IU + INCU
END FOR
END IF
RETURN
END

SUBROUTINE VSMA2(Y, INCY, A, U, INCU, B, V, INCV, C, N)

Y[k] = a * U[k] + b * V[k] + c

REAL Y(*), U(*), V(*), A, B, C
INTEGER I, INCY, INCU, INCV, IY, IU, IV, N
IF (N .LE. 0) RETURN
IF (INCY .EQ. 1 .AND. INCU .EQ. 1 .AND. INCV .EQ. 1)

FOR I = 1, N
  Y(I) = A * U(I) + B * V(I) + C
END FOR
ELSE

IY = 1
IU = 1
IV = 1
IF (INCY .LT. 0) IY = (-N+1) * INCY
IF (INCU .LT. 0) IU = (-N+1) * INCU
IF (INCV .LT. 0) IV = (-N+1) * INCV
FOR I = 1, N
  Y(IY) = A * U(IU) + B * V(IV) + C
  IY = IY + INCY
  IU = IU + INCU
  IV = IV + INCV
END FOR
END IF
RETURN
END
21: * non-contiguous data
22: 23: IY = 1
24: IY = 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: RETURN
37: END
38: 39: SUBROUTINE VSQRT(Y, INCY, A, U, INCU, B, N)
40: 41: * Y[k] = a * sqrt(U[k]+b)
42: INTEGER INCY, INCU, N
43: REAL Y(*), U(*), A, B
44: INTEGER I, IY, IU
45: 46: IF (INCY .EQ. 1 .AND. INCU .EQ. 1)
47: FOR I = 1, N
48: Y(I) = A * SQRT(U(I)+B)
49: END FOR
50: ELSE
51: 52: FOR I = 1,N
53: Y(IY) = A * SQRT(U(IU)+B)
54: IY = IY + INCY
55: IU = IU + INCU
56: END FOR
57: END IF
58: RETURN
59: 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: REAL X(*), Y(*), TEMP
6: INTEGER I, INCX, INCY, IX, IY, N
7: 8: IF (N .LE. 0) RETURN
9: 10: IF (INCX .EQ. 1 .AND. INCY .EQ. 1)
11: 12: * contiguous data
13: FOR I = 1,N
14: TEMP = X(I)
15: X(I) = Y(I)
16: Y(I) = TEMP
17: END FOR
18: RETURN
19: END

203
ELSE

* non-contiguous data

IX = 1

IF (INCY .LT. 0) IY = (-N+1) * INCY + 1

FOR I = 1,N
    TEMP = X(IX)
    X(IX) = Y(IY)
    Y(IY) = TEMP
    IX = IX + INCX
    IY = IY + INCY
END FOR

RETURN

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 transmitted 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 PVECl 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 longitudes 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.
**ARROW**

```plaintext
* draw arrow with point at <x,y>

routines required: movepn, (hpplb or vplb)
wrot (mathlb)

draw arrow with point at <x, y>

routines required: movepn, (hpplb or vplb)
wrot (mathlb)

CALL MOVEPN(X, Y, 00, PENUP)

XX = 0.4 * SIZE
YY = 0.1 * SIZE
CALL VROT(XX, YY, CA, SA, 1)
CALL MOVEPN(X+XX, Y+YY, 00, PENUP)

XX = 0.4 * SIZE
YY = -0.1 * SIZE
CALL VROT(XX, YY, CA, SA, 1)
CALL MOVEPN(X+XX, Y+YY, 00, PENUP)

CALL MOVEPN(X, Y, 00, PENUP)

RETURN

END
```

**AXLAB**

```plaintext
* draw label for axis

XMIN - input real
XMAX - input real
YMIN - input real
YMAX - input real
LAB - input integer character string terminated by default delimiter
HEIGHT - input real character height
OFF - input real distance of label from axis

find length of string

I = 1
J = 1
LEN = 0
WHILE (I .LT. 81)
  IF (IN(I) .EQ. 64 )
    IN(J) = ESC(IN(I)) ! escaped character
```

211
ELSE IF (IN(J) .EQ. 110 .OR. IN(J) .EQ. 111) I super or subscript
LEN = LEN - 1
IN(J) = IN(I)
END IF
IF (IN(J) .EQ. DELIM)
IN(J) = -2
EXIT WHILE
END IF
I = I + 1
J = J + 1
LEN = LEN + 1
END WHILE
ANG = ATAN2(YMAX-YMIN,XMAX-XMIN)
CA = COS(ANG)
SA = SIN(ANG)
ANG = 57.29578 * ANG
X = (XMIN+XMAX)/2.0-CA*FLOAT(LEN)*HEIGHT/2.0-SA*OFFSET
Y = (YMIN+YMAX)/2.0-SA*FLOAT(LEN)*HEIGHT/2.0+CA*OFFSET
* draw characters
CALL NEWPEN(3)
CALL PACK(IN,OUT)
CALL GRSTR(X,Y,OUT,J-1,HEIGHT,ANG)
CALL NEWPEN(1)
RETURN
END

SUBROUTINE BOX(XLO, YLO, BASE, HEIGHT, ANG)
REAL XLO, YLO, BASE, HEIGHT, ANG
REAL CA, DR, SA, XX, YY
INTEGER OO
DATA DR /0,017453/
CA = COS(ANG*DR)
SA = SIN(ANG*DR)
OO = INT(RDMTAB(17.2))
CALL MOVEPN(XLO,YLO,00,0)
XX = BASE
YY = 0.0
CALL VROT(XX,Y,YY,C,SA,1)
CALL MOVEPN(XLO+XX,YLO+YY,00,1)
XX = BASE
YY = HEIGHT
CALL VROT(XX,Y,YY,C,SA,1)
CALL MOVEPN(XLO+XX,YLO+YY,00,1)
XX = 0.0
YY = HEIGHT
CALL VROT(XX,Y,YY,C,SA,1)
CALL MOVEPN(XLO+XX,YLO+YY,00,1)
XX = BASE
YY = 0.0
CALL VROT(XX,Y,YY,C,SA,1)
CALL MOVEPN(XLO+XX,YLO+YY,00,1)
RETURN
END

SUBROUTINE CIRCLE(X, Y, R)
REAL X, Y, R
REAL XX, YY, ANG, DANG, PIT2, PINCFI
INTEGER OO
DATA PIT2 /6.2831853/
* draw a circle at <x,y> with radius r
XX = BASE
YY = 0.0
CALL VROT(XX,Y,YY,C,SA,1)
CALL MOVEPN(XLO+XX,YLO+YY,00,1)
XX = BASE
YY = HEIGHT
CALL VROT(XX,Y,YY,C,SA,1)
CALL MOVEPN(XLO+XX,YLO+YY,00,1)
XX = 0.0
YY = HEIGHT
CALL VROT(XX,Y,YY,C,SA,1)
CALL MOVEPN(XLO+XX,YLO+YY,00,1)
XX = BASE
YY = 0.0
CALL VROT(XX,Y,YY,C,SA,1)
CALL MOVEPN(XLO+XX,YLO+YY,00,1)
RETURN
END
**ENDFIG 544 1 AUG 82 14:23:48**

```
1: SUBROUTINE ENDFIG(WAIT)

2: INTEGER WAIT
3: REAL PMODE(25,2)
4: COM 110t: /CP1'rrAB / PMODE
5: INTEGER PS, STR(5)
6: REAL X, Y
7: IF (WAIT .EQ. 1)
8: PMODE07 .2) • 1.0 I set coordinate to absolute
9: CALL GRSYMB(0.3.0.3.63 . 0.25,0.0)
10: READ ( 0, )
11: END IF
12: STR(1) • 100
13: STR(2) • 100
14: STR(3) • 101
15: STR(4) • 104
16: STR(5) • -2
17: PS • 4
18: CALL OUTPPP(STR, PS)
19: CALL FLUSH
20: RETURN
21: END
```

**ENDFIG 149 11 JUN 82 14:41:24**

```
1: SUBROUTINE ENDFIG(WAIT)
2: INTEGER WAIT
3: REAL PMODE
4: CALL PLOT(0.0.0.0.-999)
5: RETURN
6: END
```

**ENDPLT 281 16 FEB 82 21:07:28**

```
1: SUBROUTINE ENDPLT
2: INTEGER PS, STR(5)
3: STR(1) • 100
4: STR(2) • 97
5: STR(3) • 100
6: STR(4) • 101
7: STR(5) • -2
8: PS • 4
9: CALL OUTPPP(STR, PS)
10: RETURN
11: END
```

---

213
15: CALL FLUSH
16: RETURN
17: 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(0.0.0.0.0.999)
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 /CPMTR/ 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(l.l)*PMODE(l5,l)*PMODE(l5,2))
17: CXY(2) = INT(YLO*PMODE(l.2)*PMODE(l5,l)*PMODE(l5,2))
18: CXY(3) = INT(XHI*PMODE(l.l)*PMODE(l5,1)*PMODE(l5,2))
19: CXY(4) = INT(YHI*PMODE(l.2)*PMODE(l5,l)*PMODE(l5,2))
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: * set coordinate type
28: 29: IF (PMODE(l7,l) .EQ. 1.0)
30: . STR(PS) = 101
31: ELSE IF (PMODE(l7,l) .EQ. 3.0)
32: . STR(PS) = 102
33: ELSE
34: . CALL ERROR(" illegal coordinate type")
35: END IF
36: 37: STR(PS+1) = -2
38: 39: CALL OUTRPP(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 1, 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•'(lX.27A3)') PBUF
214
SUBROUTINE GRCURS(CURS)

INTEGER CURS
REAL PMODE(25,2)
COMMON /CPMTAB/ PMODE
INTEGER STR(3), PS

STR(1) = 100
IF (CURS .EQ. 1) . STR(2) = 107
ELSE IF (CURS .EQ. 0) . STR(2) = 108
END IF
STR(3) = -2
PS = 2
CALL OUTHEPP(STR, PS)

CALL FLUSH temporary

* update mode table
PMODE(16,2) = FLOAT(CURS)

RETURN
END

SUBROUTINE GRID(X, Y, NX, XD, NY, YD, MASK)

INTEGER NX, NY, MASK
REAL X, XD(1), Y, YD(1)
REAL PMODE(25,2)
COMMON /CPMTAB/ PMODE
INTEGER 00
REAL XLEN, XX, YLEN, YY
LOGICAL VARSPX, VARSPY

00 = INT(PMODE(17,2))
VARSPX = NX .GT. 1000
VARSPY = NY .GT. 1000

IF (VARSPX)
XLEN = SUM(XD, 1, MOD(NX, 1000))
ELSE
XLEN = NX * XD(1)
END IF
IF (VARSPY)
YLEN = SUM(YD, 1, MOD(NY, 1000))
ELSE
YLEN = NY * YD(1)
END IF
CALL LINTPP(MASK, 1)
CALL MOVEP(X, Y, 0.0)
XX = X
YY = Y
CALL MOVEP(XX+XLEN, YY, 0.0)
FOR I = 1, MOD(NY, 1000)
. IF (VARSPY)
. YY = YY + YD(1)
. ELSE
. YY = YY + YD(1)
. END IF
. CALL MOVEP(XX, YY, 0.0)
. END FOR

RETURN
END
CALL MOVEPN(X,Y,0.0,0.0)
XX = X
YY = Y
CALL MOVEPN(XX,YY+YLEN,0.0,0.1)
FOR I = 1, MOD(NX,1000)
IF (VARSPX)
   XX = XX + XD(I)
ELSE
   XX = XX + XD(I)
END IF
CALL MOVEPN(XX,YY,0.0,0.0)
CALL MOVEPN(XX,YY+YLEN,0.0,0.1)
END FOR
CALL MOVEPN(X,Y,0.0,0.0)
RETURN
END

INTEGER N, FIELD, DEC
REAL X, Y, FLT, HEIGHT, ANG
REAL PMODE(2,2)
cnw. r. N
/CPITAB/ PMODE
INTEGER 00
INTEGER STR(20), PS, L, ORIENT, SIZE
INTEGER FTOC
move pen to <x,y>
00 = INT(PMODE(17,2))
CALL MOVEPN(X,Y,0.0,0.0)
* move pen to <x,y>
set up size and orientation
SIZE = INT(HEIGHT/0.25) + 1
IF (ANG .GE. 0.0)
   ORIENT = MOD (NINT(ANG / 90.0),4) + 1
ELSE
   ORIENT = MOD(NINT((360.0+ANG)/90.0),4)
END IF
STR(1) = 109
STR(2) = 48 + SIZE
STR(3) = 109
STR(4) = 48 + ORIENT
STR(5) = 110
STR(6) = 49
STR(7) = 113
STR(8) = -2
PS = 7
CALL OUTHPP(STR,PS)
* convert floating point number to string and set up for output
STR(1) = 100
STR(2) = 83
PS = 3
L = FTOC(FLT,STR(PS),FIELD,DEC)
PS = PS + L
CALL OUTHPT(STR,PS)
* turn off graphics text mode
STR(1) = 100
STR(2) = 116
STR(3) = -2
PS = 2
CALL OUTHPP(STR,PS)
RETURN
END
*-h- GRNUH 345 3 AUG 82 11:01:43
SUBROUTINE GRNUH(X, Y, FLT, FIELD, DEC, HEIGHT, ANG)
* plot floating point number from <x,y>
* interface to VPLOT07
INTEGER FIELD, DEC
RF. AL X, Y, FLT, HEIGHT, ANG
RF. AL PHODE(25,2)
COMMON /CPHTAJ3/ PHODE
CALL NUMBER(X,Y,HEIGHT,FLT,ANG,DEC)
PMODE(6.1) = X
PMODE(6.2) = Y
RETURN
END

*-h- GRSTR 1394 2 MAR 83 12:49:18
SUBROUTINE GRSTR(X, Y, BUF, N, HEIGHT, ANG)
* graphics text output of character string to HP 2647A
* if n > 0 string is packed and n is the number of characters
* if n <= 0 string is unpacked and terminated by EOS
INTEGER BUF(27), N
RF. AL X, Y, HEIGHT, ANG
RF. AL PHODE(25,2)
COMMON /CPHTAJ3/ PMODE
INTEGER 00, ORIENT, PS, LENGTH
INTEGER STR(81), UBUF(81)

move pen to <x,y>
00 • INT(PMODE(17.2))
CALL MOVEPN(X,Y,00.0)

set up size and orientation
SIZE • INT(HEIGHT/0.25) + 1
IF (ANG .GE. 0.0)
ORIENT• MOD(NINT(ANG/90.0),4) + 1
ELSE
ORIENT• MOD(NINT((360.0+ANG)/90.0),4) + 1
END IF

STR(1) • 109
STR(2) • 48 + SIZE
STR(3) • 109
STR(4) • 48 + ORIENT
STR(5) • 110
STR(6) • 49
STR(7) • 113
STR(6 ) • -2
PS • 7
CALL OUTBPP(STR.PS)

set up string for output
STR(1) • 100
STR(2) = 83
PS = 2
IF (N .GT. 0)
CALL UNPACK(BUF,UBUF,N/3+1)
FOR PU = 1,N
PS = PS + 1
STR(PS) = UBUF(1)
FOR TU = 2,N
STR(PS+1) = -2
ELSE
CALL SCOPY(BUF,1.STR.PS+1)
PS = PS + LENGTH(BUF) + 1
END IF
CALL OUTRPT(STR.PS)
59: * turn off graphics text mode
60:
61: STR(1) = 100
62: STR(2) = 116
63: STR(3) = -2
64: PS = 2
65: CALL OUTFPP(STR, PS)
66:
67: RETURN
68: END

1: *-h- GRSTR 604 2 MAR 83 12:57:15
2: SUBROUTINE GRSTR(X, Y, BUF, N, HEIGHT, ANG)
3: * plot string from <x, y>
4: * interface to VPLOT07
5: * if n > 0 string is packed and n is the number of characters
6: * if n <= 0 string is terminated by EOS
7:
8: INTEGER BUF(l), N
9: REAL X, Y, HEIGHT, ANG
10: REAL PHODE(25, 2)
11: COMMON /CPMTAB/ PHODE
12: INTEGER LENGTH
13: INTEGER LEN, PBUF(27)
14:
15: IF (N .GT. 0)
16: CALL SYMBOL(X, Y, HEIGHT, BUF, ANG, N)
17: ELSE
18: LEN = LENGTH(BUF)
19: CALL PACK(BUF, PBUF)
20: CALL SYMBOL(X, Y, HEIGHT, PBUF, ANG, LEN)
21:
22: END IF
23:
24: PMODE(6.1) = X
25: PMODE(6.2) = Y
26:
27: RETURN
28:
29: END

1: *-h- GRSYM 1044 21 FEB 83 15:25:23
2: SUBROUTINE GRSYM(X, Y, ASC, HEIGHT, ANG)
3: * write ASCII symbol at <x, y>
4:
5: REAL X, Y, HEIGHT, ANG
6: INTEGER ASC
7: REAL PHODE(25, 2)
8: COMMON /CPMTAB/ PHODE
9: INTEGER OTHER
10: INTEGER STR(l5), PS, ORIENT, SIZE
11:
12: * move pen to <x, y>
13: OO = INT(PMODE(17, 2))
14: CALL MOVEPH(X, Y, OO, 0)
15:
16: * set text parameters
17: SIZE = INT(HEIGHT/0.25) + 1
18: IF (ANG .GE. 0.0)
19: . ORIENT = MOD(NINT(ANG/90.0), 4) + 1
20: ELSE
21: . ORIENT = MOD(NINT((360.0+ANG)/90.0), 4) + 1
22: END IF
23:
24: STR(1) = 109
25: STR(2) = 48 + SIZE
26: STR(3) = 109
27: STR(4) = 48 + ORIENT
28: STR(5) = 110
29: STR(6) = 53
30: STR(7) = 113
31: STR(8) = -2
32: PS = 7
33:
34: CALL OUTFPP(STR, PS)
38: * draw character
39: STR(1) = 100
40: STR(2) = 83
41: STR(3) = 116
42: STR(4) = -2
43: RETURN
44: CALL OUTHPT(STR PS)
45: * turn off graphics text (I don't know why this is necessary AES 1/81)
46:
47: STR(1) = 100
48: STR(2) = 116
49: STR(3) = -2
50: PS = 2
51: CALL OUTHPT(STR PS)
52: RETURN
53: END

1: *-h- GRSYMB 311 21 FEB 83 14:13:52
2: SUBROUTINE GRSYMB(X, Y, ASC, HEIGHT, ANG)
3: *plot single symbol at <x,y>
4: * interface to VSPLOT
5: INTEGER ASC
6: REAL X, Y, HEIGHT, ANG
7: CALL SYMBOL(X, Y, HEIGHT, ASC, ANG, -1)
8: RETURN
9: END

1: *-h- INBP 901 22 JUN 82 21:50:03
2: SUBROUTINE INBP(BUF, PB)
3: * receive buffer from HP 2647A terminal. reformat for use
4: * by plot package
5: INTEGER BUF(1), PB
6: COUNT
7: STADD
8: K
9: WRITE (3.FMT=’(/)’)
10: transfer word count to COUNT
11: transfer start address of BUF to STADD
12: transfer long operand at label to K
13: call I/O routine. wait for completion
14: turn on data program counter
15: turn on instruction program counter
16: END
*-h- INIPLT 3998 31 MAY 83 22:58:54

SUBROUTINE INIPLT

* prepare HP 2647A graphics terminal for plotting

INTEGER PLTBUF(81), PB
COMMON /CHPOUT/ PLTBUF, PB
REAL PMODE(25,2)
COMMON /CPMTAB/ PMODE
INTEGER PS, STR(5)

*************** mode table for graphics ***************

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x scale factor</td>
<td>y scale factor</td>
</tr>
<tr>
<td>2</td>
<td>x coordinate base</td>
<td>y coordinate base</td>
</tr>
<tr>
<td>3</td>
<td>x max dimension</td>
<td>y max dimension</td>
</tr>
<tr>
<td>4</td>
<td>x window min</td>
<td>y window min</td>
</tr>
<tr>
<td>5</td>
<td>x window max</td>
<td>y window max</td>
</tr>
<tr>
<td>6</td>
<td>x current pen pos</td>
<td>y current pen pos</td>
</tr>
<tr>
<td>7</td>
<td>x current origin pos</td>
<td>y current origin pos</td>
</tr>
<tr>
<td>8</td>
<td>x cursor pos</td>
<td>y cursor pos</td>
</tr>
<tr>
<td>9</td>
<td>x zoom pos</td>
<td>y zoom pos</td>
</tr>
<tr>
<td>10</td>
<td>x previous pen pos</td>
<td>y previous pen pos</td>
</tr>
<tr>
<td>11</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>12</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>13</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

32: * x, y scale factor
33: * x, y scale factor
34: * x, y scale factor
35: * x, y scale factor
36: * x, y scale factor
37: * x, y scale factor
38: * x, y scale factor
39: * x, y scale factor
40: * x, y scale factor
41: * x, y scale factor
42: * x, y scale factor
43: * x, y scale factor

44: * x, y scale factor
45: * x, y scale factor
46: * x, y scale factor
47: * x, y scale factor
48: * x, y scale factor
49: * x, y scale factor
50: * x, y scale factor
51: * x, y scale factor
52: * x, y scale factor
53: * x, y scale factor
54: * x, y scale factor
55: * x, y scale factor
56: * x, y scale factor
57: * x, y scale factor
58: * x, y scale factor
59: * x, y scale factor
60: * x, y scale factor
61: * x, y scale factor
62: * x, y scale factor
63: * x, y scale factor
64: * x, y scale factor
65: * x, y scale factor
66: * x, y scale factor
67: * x, y scale factor
68: * x, y scale factor
69: * x, y scale factor
70: * x, y scale factor
71: * x, y scale factor
72: * x, y scale factor
73: * x, y scale factor
74: * x, y scale factor
75: * x, y scale factor
76: * x, y scale factor
77: * x, y scale factor
78: * x, y scale factor
79: * x, y scale factor
80: * x, y scale factor

* HP 2647A defaults

PMODE(1,1) = 1.0
PMODE(1,2) = 1.0
PMODE(2,1) = 0.0
PMODE(2,2) = 0.6
PMODE(3,1) = 10.0
PMODE(3,2) = 0.0
PMODE(4,1) = 0.0
PMODE(4,2) = 0.0
PMODE(5,1) = 0.0
PMODE(5,2) = 0.0
PMODE(6,1) = 0.0
PMODE(6,2) = 0.0
PMODE(7,1) = 0.0
PMODE(7,2) = 0.0
PMODE(8,1) = 0.0
PMODE(8,2) = 0.0
PMODE(9,1) = 0.0
PMODE(9,2) = 0.0
PMODE(10,1) = 0.0
PMODE(10,2) = 0.0
PMODE(11,1) = 0.0
PMODE(11,2) = 0.0
PMODE(12,1) = 0.0
PMODE(12,2) = 0.0
PMODE(13,1) = 0.0
PMODE(13,2) = 0.0
PMODE(14,1) = 0.0
PMODE(14,2) = 0.0
PMODE(15,1) = 1.0
PMODE(15,2) = 72.0
PMODE(16,1) = 0.0
PMODE(16,2) = 0.0
PMODE(17,1) = 0.0
PMODE(17,2) = 1.0
PMODE(18,1) = 4.0
PMODE(18,2) = 0.0
PMODE(19,1) = 0.0
PMODE(19,2) = 46.0

220
81: * initialize output buffer
82:   PLTBUF(1) = 27
83:   PLTBUF(2) = 42
84:   PLTBUF(3) = 109
85:   PB = 3
86: * clear screen and turn on graphics mode
87:   STR(1) = 109
88:   STR(2) = 114
89:   STR(3) = -2
90:   STR(4) = -2
91:   PB = 3
92:   CALL OUTFPP(STR, PS)
93: * prepare VERSATEC 1200 plotter for plotting
94: REAL PMODE(25, 2)
95: COMMON /CPMTAB/ PMODE
96: 1: *-h- INIPLT 3348 21 FEB 83 15:34:06
97: 2: SUBROUTINE INIPLT
98: 3:
99: 4: 5
100: 6:   REAL PMODE(25, 2)
101:   COMMON /CPMTAB/ PMODE
102:   PMODE(l, l) = 1.0
103:   PMODE(l, 2) = -1.0
104:   PMODE(2, l) = 0.0
105:   PMODE(2, 2) = 0.0
106: END
SUBROUTINE LINTYP(MASK, SCALE)

* define dot pattern for drawing vectors or filling areas

INTEGER MASK, SCALE
INTEGER STR(10), PS, L, M
INTEGER ITOC

STR(1) = 109
PS = 2

IF (MASK .GT. 0)

* user defined mask

STR(PS) = 32
PS = PS
L = ITOC(MASK . STR(PS), 4)
PS = PS + L
STR(PS) = 32
PS = PS + 1
L = ITOC(SCALE, STR(PS), 3)
PS = PS + L
STR(PS) = 99
PS = PS + 1
M = -2
ELSE
M = MASK
END IF

STR(PS) = 32
PS = PS + 1
L = ITOC(-M, STR(PS), 3)
PS = PS + L
STR(PS) = 98
STR(PS+1) = -2
CALL OUTHPP(STR, PS)
RETURN
END

SUBROUTINE MOVEOO(X, Y)

* move relocatable origin to X, Y

REAL X, Y
INTEGER STR(15), PS, L, CX, CY
REAL PMODE(25, 2)
COMMON /CPMTAB/ PMODE

STR(1) = 109
PS = 2

CALL PLOTS(0, 0, 0)
RETURN
END
14: * scale coordinates
15:  CX = INT(X*PMODE(1,1)*PMODE(15,1)*PMODE(15,2))
16:  CY = INT(Y*PMODE(1,1)*PMODE(15,1)*PMODE(15,2))
17:  L = ITOC(CX,STR(PS),5)
18:  PS = PS + L
19:  STR(PS) = 32
20:  PS = PS + 1
21:  L = ITOC(CY,STR(PS),5)
22:  PS = PS + L
23:  STR(PS) = 106
24:  STR(PS+1) = -2
25:  CALL OUTBPP(STR,PS)
26:  * update mode table
27:  PMODE(7,1) • X
28:  PMODE(7,2) • Y
29:  PMODE(17,2) • 3.0
30:  RETURN
31:  END

scale coordinates

---

**FUNCTION MOVEO**

3 AUG 82 11:01:49

SUBROUTINE MOVEO(X, Y)

* move relocatable origin to <x,y>*

4: * interface to VPLOT07

5: REAL X, Y
6: REAL PMODE(25,2)
7: COMMON /CPMTAB/ PMODE
8: CALL PLOT(X,Y,-3)
9: PMODE(7,1) • PMODE(7,1) + X
10: PMODE(7,2) • PMODE(7,2) + Y
11: RETURN
12: END

**FUNCTION MOVEPI**

28 MAY 83 12:34:48

SUBROUTINE MOVEPI(X, Y, 00, PEN)

* move pen to <x,y> in absolute screen coordinates*

4: INTEGER X, Y, 00, PEN
5: REAL PMODE(25,2)
6: COMMON /CPMTAB/ PMODE
7: INTEGER STR(15), PS, L, BYTE(3), I, M, P
8: STR(1) • 112
9: PS • 2
10: IF (PEN .EQ. 0)
11: • STR(PS) • 97
12: ELSE IF (PEN .EQ. 1)
13: • STR(PS) • 98
14: ELSE IF (00 .EQ. 3)
15: • STR(PS) • 106
16: ELSE
17: • CALL ERROR(" illegal coordinate type")
18: END IF
19: PS • PS + 1
20: RETURN
21: END

**FUNCTION MOVE**

276 3 AUG 82 11:01:49

SUBROUTINE MOVEO(X, Y)

* move relocatable origin to <x,y>*

4: * interface to VPLOT07

5: REAL X, Y
6: REAL PMODE(25,2)
7: COMMON /CPMTAB/ PMODE
8: CALL PLOT(X,Y,-3)
9: PMODE(7,1) • PMODE(7,1) + X
10: PMODE(7,2) • PMODE(7,2) + Y
11: RETURN
12: END

**FUNCTION MOVEPI**

1673 28 MAY 83 12:34:48

SUBROUTINE MOVEPI(X, Y, 00, PEN)

* move pen to <x,y> in absolute screen coordinates*

4: INTEGER X, Y, 00, PEN
5: REAL PMODE(25,2)
6: COMMON /CPMTAB/ PMODE
7: INTEGER STR(15), PS, L, BYTE(3), I, M, P
8: STR(1) • 112
9: PS • 2
10: IF (PEN .EQ. 0)
11: • STR(PS) • 97
12: ELSE IF (PEN .EQ. 1)
13: • STR(PS) • 98
14: ELSE IF (00 .EQ. 3)
15: • STR(PS) • 106
16: ELSE
17: • CALL ERROR(" illegal coordinate type")
18: END IF
19: PS • PS + 1
20: RETURN
21: END

---

223
36: * load coordinates into successive bytes. 2 bytes are required for
37: * absolute coordinates, 3 for incremental and relative coordinates
38: * (to accommodate negative numbers). each byte stores 5 bits of
39: * coordinate. negative numbers are in two's complement
40: * (compatible with HARRIS)
41: *
42: M = 1
43: I = 0
44: IF (OO .GT. 1)  
45: I = I + 1
46: BYTE(I) = (X .SHIFT. -10) .AND. '37
47: END IF
48: I = I + 1
49: BYTE(I) = (X .SHIFT. -5) .AND. '37
50: I = I + 1
51: BYTE(I) = X .AND. '37
52: M = I
53: FOR I = 1,M
54: . STR(PS) = BYTE(I) .OR. '40
55: . PS = PS + 1
56: END FOR
57: I = 0
58: IF (OO .GT. 1)  
59: I = I + 1
60: BYTE(I) = (Y .SHIFT. -10) .AND. '37
61: END IF
62: I = I + 1
63: BYTE(I) = (Y .SHIFT. -5) .AND. '37
64: I = I + 1
65: BYTE(I) = Y .AND. '37
66: FOR I = 1,M
67: . STR(PS) = BYTE(I) .OR. '40
68: . PS = PS + 1
69: END FOR
70: STR(PS) = -2
71: CALL OUTHPP(STR,PS-1)
72: RETURN
73: END
74: *
75: RETURN
76: END
77: **
78: SUBROUTINE MOVEP(X,Y,OO,PEN)
79: *
80: move pen to <x,y>
81: REAL X, Y
82: INTEGER OO, PEN
83: REAL FMODE(25,2)
84: COMMON /CPMAB/ FMODE
85: INTEGER CX, CT, F
86: *
87: update current pen position
88: FMODE(6,1) = X
89: FMODE(6,2) = Y
90: *
91: if clip mode is on calculate new drawing coordinates
92: IF (FMODE(19,1) .EQ. 1)
93: CALL FCLIP(X,Y,OO,PEN)
94: ELSE
95: CX = NINT(X*FMODE(1,1)*FMODE(15,1)*FMODE(15,2))
96: CY = NINT(Y*FMODE(1,2)*FMODE(15,1)*FMODE(15,2))
97: CALL MOVEPI(CX,CT,OO,PEN)
98: END IF
99: *
100: update previous pen position
101: FMODE(10,1) = X
102: FMODE(10,2) = Y
103: RETURN
104: END
SUBROUTINE MOVEPN(X, Y, COOR, PEN)

INTEGER COOR, PEN
REAL PMODE(25,2)
COMMON /CPMTAB/ PMODE

PMODE(6,1) = X
PMODE(6,2) = Y
PMODE(17,1) = FLOAT(PEN)

* recalculate coordinates if clip mode is on
IF (PMODE(19,1) .EQ. 1.0)
  CALL PCLIP(X,Y,COOR,PEN)
ELSE
  CALL PLOT(X,Y,3-PEN)
END IF

store <x,y> coordinates for next call
PMODE(10,1) = X
PMODE(10,2) = Y
RETURN

SUBROUTINE NEWPEN(PEN)

INTEGER PEN
RETURN

SUBROUTINE OUTHPP(STR, PS)

INTEGER STR(l), PS
INTEGER PLTBUF(81), PB
COMMON /CHPOUT/ PLTBUF, PB

if device control mode is different, send buffer to terminal
IF (STR(1) .NE. PLTBUF(3))
  IF (PB .GT. 3) CALL FLUSH
  PLTBUF(3) = STR(1)
END IF

if insufficient room in buffer, send to terminal
IF (PB+PS+2 .GE. 81)
  IF (PB .GT. 3) CALL FLUSH
  PLTBUF(3) = STR(1)
END IF

* copy command string to output buffer
PB = PB + 1
CALL SCOPY(STR,2,PLTBUF,PB)
PB = PB + PS - 2
RETURN
END
SUBROUTINE OUTPT (STR, PS)

INTEGER STR(1), PS
INTEGER PLTBUF(81), PB
COMMON /CHPOUT/ PLTBUF, PB
INTEGER I, PBUF(27)

IF (STR(1) .NE. PLTBUF(3)) THEN
   CALL FLUSH
   END IF

CALL SCOPY(STR(1), PLTBUF, PB)
PB = PB + PS
PLTBUF(PB) = 13
FOR I = 1,27
   PBUF(I) = 0
END FOR
CALL PACK(PLTBUF, PB)
WRITE (3, FMT='(1X,27A3)') PBUF
PB = 3
RETURN
END

SUBROUTINE PAXIS (XHIN, XMAX, YMIN, YMAX, SHIN, SMAX, FIELD, DEC, HEIGHT, AOPT, ANG, SOPT, N, DF)

INTEGER FIELD, DEC, AOPT, SOPT, N, DF
REAL XMIN, XMAX, YMIN, YMAX, SHIN, SMAX
INTEGER I, H, 00, PENDOWN, PENUP
REAL XX, YY
DATA PENUP /0/. PENDOWN /1/
00 = INT(RDMTAB(17.2))

CALL NEWPEN(5)
CALL MOVEP(XMIN, YMIN, 00, PENUP)
CALL MOVEP(XMAX, YMAX, 00, PENDOWN)
CALL NEWPEN(1)

IF (N .GT. 0) THEN
   N = N / DF
   CALL SCALE1(SMIN, SMAX, N, SMIN, SMAX, DIST)
   N = NINT((SMAX - SMIN) / DIST) * DF
   ELSE IF (SOPT .EQ. 2) THEN
   N = N / DF
   CALL SCALE2(SMIN, SMAX, N, SMIN, SMAX, DIST)
   N = NINT((SMAX - SMIN) / DIST) * DF
   END IF

OO = INT(RDMTAB(17.2))
FOR I = 0, N
XX = XMIN + (XMAX - XMIN) * FLOAT(I) / FLOAT(N)
YY = YMIN + (YMAX - YMIN) * FLOAT(I) / FLOAT(N)
IF (MOD(I, DF) .EQ. 0)
S = SMIN + (SMAX - SMIN) * FLOAT(I) / FLOAT(N)
CALL TIC(XX, YY, S, FIELD, DEC, WEIGHT, ANG)
ELSE
CALL TIC(XX, YY, 0, 0, 0, 0, 6*HEIGHT, 0, ANG)
ENDIF
END FOR
ENDIF
CALL MOVEPN(XMIN, YMIN, 0, 0, PENUP)
RETURN
END

SUBROUTINE PLOTS(A1, A2, A3)
* initialize plotting
* emulate Versaplot-07 calls on HP 2647A terminal
INTEGER A1, A2, A3
CALL INIPLT
RETURN
END

SUBROUTINE PMAT(A, LDA, M, INCN, N, INCN, XLEN, YLEN, SLEN, HEIGHT, MDF, NDF)
plot coefficients of matrix in rectangle
INTEGER LDA, M, INCN, N, INCN, MDF, NDF
REAL A(LDA,1), XLEN, YLEN, SLEN, HEIGHT
INTEGER I, J, IN, IM, PEN, PENDOWN.
REAL XDEL, YDEL, YMAX, YMIN, YSCALE
REAL SVMAX, SVMIN
DATA PENDOWN / 1 /, PENUP / 0 /
XDEL = XLEN / FLOAT(M)
YDEL = YLEN / FLOAT(N)
00 = INT(RDMTAB(17.2))
IF (MDF .GT. 0 .AND. NDF .GT. 0)
PENUP
CALL PAXIS(0, 0, XLEN, 0, 0, YLEN, 0, 0, FLOAT(N), 0, 0, FLOAT(M), 4, -1, 0, 15, 0, 0, 0, ANG)
CALL PAXIS(0, 0, YLEN, 0, 0, FLOAT(N), 0, 0, FLOAT(M), 0, 0, 0, 15, 0, 0, 0, 0, ANG)
CALL PAXIS(XLEN, XLEN, 0, 0, YLEN, 0, 0, FLOAT(N), 0, 0, 0, 0, 0, 0, ANG)
ENDIF
YMAX = -1.0E37
YMIN = 1.0E37
IN = 1
FOR I = 1, N
YMAX = AMAX1(YMAX, SVMAX(A(I, IN), INCN, JUNK, M))
YMIN = AMIN1(YMIN, SVMIN(A(I, IN), INCN, JUNK, M))
IN = IN + INCN
END FOR
YSCALE = SLEN / (YMAX - YMIN)
FOR J = 1, N
PEN = PENUP
IM = 1
FOR I = 1, M
XX = I * XDEL
YY = A(I, IM) * YSCALE + J * YDEL
CALL MOVEPN(XX, YY, 0, 0, 0, PEN)
PEN = PENDOWN
IM = IM + INCN
END FOR
IN = IN + INCN
END IF
ENDIF
CALL MOVEPN(XMIN, YMIN, 0, 0, PENUP)
RETURN
END

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

1: *-h- PMAT 1306 24 JAN 83 11:19:20
2: SUBROUTINE PMAT(A, LDA, M, INCN, N, INCN, XLEN, YLEN, SLEN, HEIGHT, MDF, NDF)
3: & HEI
4: 
5: * plot coefficients of matrix in rectangle
6: 
7: INTEGER LDA, M, INCN, N, INCN, MDF, NDF
8: REAL A(LDA,1), XLEN, YLEN, SLEN, HEIGHT
9: INTEGER I, J, IN, IM, PEN, PENDOWN.
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: PENUP
20: 
21: REAL XMAX, YMAX
22: 6. MDF
23: CALL PAXIS(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ANG)
24: 6. N NDF
25: CALL PAXIS(XLEN, XLEN, 0, 0, YLEN, 0, 0, FLOAT(N), 0, 0, 0, 0, 0, 0, 0, 0, ANG)
26: 
27: 
28: IF (MDF .GT. 0 .AND. NDF .GT. 0)
29: PENUP
30: 
31: REAL XMAX, YMAX
32: 6. MDF
33: CALL PAXIS(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ANG)
34: 6. N NDF
35: CALL PAXIS(XLEN, XLEN, 0, 0, YLEN, 0, 0, FLOAT(N), 0, 0, 0, 0, 0, 0, 0, 0, ANG)
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(I, IM) * YSCALE + J * YDEL
44: CALL MOVEPN(XX, YY, 0, 0, 0, PEN)
45: PEN = PENDOWN
46: IM = IM + INCN
47: END FOR
48: IN = IN + INCN

227
SUBROUTINE PVECl(Y, N, XO, Y0, YREF, YSCALE, XINC, ANG)

* plot a vector of data against an incremental ordinate

INTEGER N
REAL Y(l), XO, YO, YREF, YSCALE, XINC, ANG

INTEGER I, 00, PEN
REAL CA, SA, XX, YY
REAL RDMTAB

DATA DR /0.01745329/

00 • INT(RDMTAB(l7 .2))
CA • COS(DR*ANG)
SA • SIN(DR*ANG)
PEN • 0

FOR I • 1, N
XX • XINC * FLOAT(I-1)
YY • (Y(I) - YREF) * YSCALE
CALL VROT(XX,1.YY,1.CA,SA,1)
CALL MOVEPN(XO+XX,YO+YY.00.PEN)
PEN • 1
END FOR
RETURN
END

SUBROUTINE PVEC2(X, Y, N, XO, Y0, XREF, XSCALE, YREF, YSCALE)

* plot array of points <x, y>

INTEGER N
REAL X(l), Y(l), XO, YO, XREF, XSCALE, YREF, YSCALE

INTEGER I, 00, PEN
REAL XX, YY
REAL RDMTAB

00 • INT(RDMTAB(l7.2))
PEN • 0

FOR I • 1, N
XX • XO + (X(I) - XREF) * XSCALE
YY • YO + (Y(l) - YREF) * YSCALE
CALL MOVEPN(XX.YY.00.PEN)
PEN • 1
END FOR
RETURN
END

SUBROUTINE PNTCNF(X, Y, XERR, YERR, SIZE)

* plot datum point plus error bars

INTEGER 00
REAL SIZED2, XX, YY

00 • INT(RDMTAB(l7.2))
SIZED2 • SIZE / 2.0
CALL CIRCLE(X.Y, SIZED.D2)
get origin type

CALL CIRCLE(X.Y, SIZED2)
14: * x error bars
15: 16: IF (XERR .GT. 0.0) 
17:  . XX = X + SIZE2 
18:  . CALL MOVEP(XX,Y,00.0) 
19:  . XX = X + XERR 
20:  . CALL MOVEP(XX,Y,00.1) 
21:  . YY = Y - SIZE2 
22:  . CALL MOVEP(XX,YY,00.0) 
23:  . YY = Y + SIZE2 
24:  . CALL MOVEP(XX,YY,00.1) 
25:  . XX = X - SIZE2 
26:  . CALL MOVEP(XX,Y,00.0) 
27:  . XX = X - XERR 
28:  . CALL MOVEP(XX,Y,00.1) 
29:  . YY = Y - SIZE2 
30:  . CALL MOVEP(XX,YY,00.0) 
31:  . YY = Y - SIZE2 
32:  . CALL MOVEP(XX,YY,00.1) 
33:  END IF 
34: 35: 36: * y error bars 
37: 38: IF (YERR .GT. 0.0) 
39:  . YY = Y + SIZE2 
40:  . CALL MOVEP(X,YY,00.0) 
41:  . YY = Y + YERR 
42:  . CALL MOVEP(X,YY,00.1) 
43:  . XX = X - SIZE2 
44:  . CALL MOVEP(XX,Y,00.0) 
45:  . XX = X - YERR 
46:  . CALL MOVEP(XX,Y,00.1) 
47:  . YY = Y - SIZE2 
48:  . CALL MOVEP(X,YY,00.0) 
49:  . YY = Y - YERR 
50:  . CALL MOVEP(X,YY,00.1) 
51:  . XX = X - SIZE2 
52:  . CALL MOVEP(XX,Y,00.0) 
53:  . XX = X - SIZE2 
54:  . CALL MOVEP(XX,Y,00.1) 
55:  . CALL MOVEP(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: 7: REAL PMODE(25.2)
8: 8: COMMON /CPMTAB/ PMODE 
9: 9: INTEGER STR(3), PS 
10: 11: STR(1) = 100 
12: 12: IF (LIN .EQ. 1) 
13: . STR(2) = 109 
14: ELSE IF (LIN .EQ. 0) 
15: . STR(2) = 110 
16: END IF 
17: 17: STR(3) = -2 
18: 18: PS = 2 
19: 19: CALL CUTHPP(STR,PS) 
20: 20: CALL FLUSH 
21: 21: RETURN 
22: 22: END 

229
1: *-h- RDMTAR 197 3 AUG 82 16:25:48
2: REAL FUNCTION RDMTAR(I, J)
3: * read i,j-th element of plot mode table
4: INTEGER I, J
5: REAL PMODE(25,2)
6: COMMON /CPMTAR/ PMODE
7: RDMTAR = PMODE(I,J)
8: RETURN
9: END

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

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

* calculate range (xminp-xmaxp) divisible into approximately n linear
* intervals of size dist (1.2.5 integer power of 10) given input
* range (xmin-xmax)

INTEGER N
REAL XMIN, XMAX, XMINP, XMAXP, DIST

INTEGER NAL, M
REAL A, EPS, FN, FM, SQR(3), VINT(4)

DATA SQR /1.414214,3.162278,7.071068/
DATA VINT /1.0 . 2.0, 5.0, 10.0 /

IF (XMIN .GT. XMAX .OR. N .LT. 0) CALL REMARK(" improper input to SCALE")
RETURN
END

IF (EPS .EQ. 0.00002) FN = FLOAT(N)

* find approximate interval of size a
A = (XMAX - XMIN) / FN
NAL = INT(ALOG10(A))
IF (A .LT. 1.0) NAL = NAL - 1

* scale a between 1 and 10
A = A * 10.0 ** NAL

* find closest possible value for a
I = 1
WHILE (A .GE. SQR(I)) I = I + 1
EXIT WHILE IF (I .GT. 3) END WHILE

* compute interval size
DIST = VINT(I) * 10.0 ** NAL

* find new min and max limits
FM = XMIN / DIST
M = INT(FM)
IF (FM .LT. 0.0) M = M - 1
IF (ABS(FLOAT(M)+1.0-FM) .LT. EPS) M = M + 1
XMINP = DIST * FLOAT(M)
FM = XMAX / DIST
M = INT(FM) + 1
IF (FM .LT. -1.0) M = M - 1
IF (ABS(FM+1.0-FLOAT(M)) .LT. EPS) M = M - 1
XMAXP = DIST * FLOAT(M)

XMINP = AMIN1(XMINP, XMIN)
XMAXP = AMAX1(XMAXP, XMAX)
RETURN
END

SUBROUTINE SCALE2(XMIN, XMAX, N, XMINP, XMAXP, DIST)

* find new range (xminp-xmaxp) divisible into exactly n linear
* intervals of size dist (1.2.5 integer power of 10) given input
* range (xmin-xmax)

INTEGER N
REAL XMIN, XMAX, XMINP, XMAXP, DIST

INTEGER I, M1, M2, NAL, NP, NX
REAL A, EPS, FN, FM, VINT(5)

DATA VINT /1.0, 2.0, 5.0, 10.0, 20.0 /

IF (XMIN .GT. XMAX .OR. N .LE. 1) CALL REMARK(" improper input supplied to SCALE2")
RETURN
END

RETURN
END IF

231
18:  EPS = 0.000002
19:  FN = FLOAT(N)
20:  
21:  * find approximate interval of size a
22: 23:  A = (XMAX - XMIN) / FN
24:  NAL = INT(ALOG10(A))
25:  IF (A .LT. 1.0) NAL = NAL - 1
26:  
27:  * scale a between 1 and 10
28: 29:  A = A / 10.0 ** NAL
30:  
31:  * find closest permissible value for b
32: 33:  I = 1
34:  WHILE (A .GE. (VINT(I)+EPS))
35:  I = I + 1
36:  EXIT WHILE IF (I .GT. 3)
37:  END WHILE
38:  
39:  DO
40:  compute interval size. (may need second pass)
41: 42:  DIST = VINT(I) * 10.0 ** NAL
43: 44:  * find new min and max limits
45: 46:  FM = XMIN / DIST
47: 48:  M1 = INT(FM)
49:  IF (FM .LT. 0.0) M1 = M1 - 1
50:  IF (ABS(FLOAT(M1)+1.0-FM) .LT. EPS) M1 = M1 + 1
51:  XMINF = DIST * FLOAT(M1)
52:  FM = XMAX / DIST
53:  M2 = INT(FM) + 1.0
54:  IF (FM .LT. -1.0) M2 = M2 - 1
55:  IF (ABS(FLOAT(M2)+1.0-FM) .LT. EPS) M2 = M2 - 1
56:  XMAFP = DIST * FLOAT(M2)
57:  NP = M2 - M1
58:  I = I + 1
59:  UNTIL (NP .LE. N)
60:  
61:  NX = (N - NP) / 2
62:  XMINF = XMINF - FLOAT(NX) * DIST
63:  XMAFP = XMINF + FLOAT(N) * DIST
64:  
65:  XMINF = AMIN1(XMINF,XMIN)
66:  XMAFP = AMAX1(XMAFP,XMAX)
67:  
68:  RETURN
69:  
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 OOTHPP(STR,PS)
18:  
19: * update mode table
20:  
21: PMODE(18,1) = FLOAT(MODE)
22:  
23: RETURN
24:  
END
SUBROUTINE TIC(X, Y, A, FIELD, DEC, HEIGHT, OPT, ANG)

* draw tic mark at (x,y) *

* if opt • (1-4) annotate with a *

X, Y : start coordinates of tic mark *

A, FIELD, DEC: real number associated with tic mark

HEIGHT, opt : width and decimal of annotation

HEIGHT : height of digits *

OPT : annotation option *

opt = 1 digits start at tic and parallel *

opt = 2 digits end at tic and parallel *

opt = 3 digits below tic and perpendicular *

opt = 4 digits above tic and perpendicular *

ANG : angle of tic mark *

INTEGER FIELD, DEC, OPT
REAL X, Y, A, HEIGHT, ANG
INTEGER LEN, DD, PENDOWN, PENUP, STR(10)
REAL CA, DR, SA, XX, YY
INTEGER FTOC
REAL COS, SIN, RDMTAB
DATA PENDOWN /1/, PENUP /0/, DR /0.01745329/
DO = INT(RDMTAB(17.2))
CA = COS(DR*ANG)
SA = SIN(DR*ANG)
LEN = FTOC(A,STR,FIELD,DEC)
CALL NEWPEN(3)
XX = X
YY = Y
CALL MOVEPN(X,Y,00,PENDOWN)
XX = XX + 0.5 * HEIGHT * CA
YY = YY + 0.5 * HEIGHT * SA
CALL MOVEPN(XX,YY,00,PENUP)
IF (OPT .EQ. 1) THEN
   XX = XX + 0.5 * HEIGHT * CA + 0.5 * HEIGHT * SA
   YY = YY + 0.5 * HEIGHT * SA - 0.5 * HEIGHT * CA
   CALL GRNUM(XX,YY,A,FIELD,DEC,HEIGHT,ANG)
ELSE IF (OPT .EQ. 2) THEN
   XX = XX + (FLOAT(LEN) + 0.5) * HEIGHT * CA - 0.5 * HEIGHT * SA
   YY = YY + (FLOAT(LEN) + 0.5) * HEIGHT * SA + 0.5 * HEIGHT * CA
   CALL GRNUM(XX,YY,A,FIELD,DEC,HEIGHT,ANG+180.0)
ELSE IF (OPT .EQ. 3) THEN
   XX = XX + 1.5 * HEIGHT * CA + 0.5 * FLOAT(LEN) * HEIGHT * SA
   YY = YY + 1.5 * HEIGHT * SA - 0.5 * FLOAT(LEN) * HEIGHT * CA
   CALL GRNUM(XX,YY,A,FIELD,DEC,HEIGHT,ANG+90.0)
ELSE IF (OPT .EQ. 4) THEN
   XX = XX + 0.5 * HEIGHT * CA - 0.5 * FLOAT(LEN) * HEIGHT * SA
   YY = YY + 0.5 * HEIGHT * SA + 0.5 * FLOAT(LEN) * 0.5 * HEIGHT * CA
   CALL GRNUM(XX,YY,A,FIELD,DEC,HEIGHT,ANG+270.0)
END IF
CALL NEWPEN(1)
RETURN
END

SUBROUTINE VP07MF(Al, A2, A3)

* dummy routine to emulate VERSAPLOT 07 call *

INTEGER Al, A2, A3
RETURN
END
SUBROUTINE ZOOM(X, Y, N)

* set zoom factor (1-8) at <x,y>

INTEGER N
REAL X, Y
REAL PH:DE(25.2)
COMMON /CPMTAB/ PMODE
INTEGER STR(16), PS, CX, CY, L, 00
INTEGER ITOC

STR(1) = 100
STR(2) = 32

* set zoom size
PS = 3
L = ITOC(N, STR(PS), 2)
PS = PS + 1
STR(PS) = 105
PS = PS + 1

* set zoom coordinates
CX = INT(X*PH:DE(1.1)*PH:DE(15.1)*PH:DE(15.2))
CY = INT(Y*PH:DE(1.2)*PH:DE(15.1)*PH:DE(15.2))
L = ITOC(CX, STR(PS), 4)
PS = PS + 1
STR(PS) = 106
PS = PS + 1
STR(PS+1) = -2

CALL OUTHPP(STR, PS)

* update mode table
PMODE(9,1) = FLOAT(CX)
PMODE(9,2) = FLOAT(CY)
PMODE(16,1) = FLOAT(N)
RETURN
END
## APPENDIX B.

### FIRST ARRIVAL PICK TRAVEL TIMES.

<table>
<thead>
<tr>
<th>Receiver 2 Shot line 3N</th>
<th>Shot Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>4496</td>
<td>57.595</td>
<td>10.657</td>
<td>0.120</td>
</tr>
<tr>
<td>4497</td>
<td>56.827</td>
<td>10.577</td>
<td>0.100</td>
</tr>
<tr>
<td>4498</td>
<td>56.101</td>
<td>10.477</td>
<td>0.100</td>
</tr>
<tr>
<td>4499</td>
<td>55.346</td>
<td>10.367</td>
<td>0.100</td>
</tr>
<tr>
<td>4500</td>
<td>54.632</td>
<td>10.248</td>
<td>0.100</td>
</tr>
<tr>
<td>4501</td>
<td>53.889</td>
<td>10.128</td>
<td>0.100</td>
</tr>
<tr>
<td>4502</td>
<td>53.196</td>
<td>10.198</td>
<td>0.100</td>
</tr>
<tr>
<td>4503</td>
<td>52.478</td>
<td>10.068</td>
<td>0.100</td>
</tr>
<tr>
<td>4504</td>
<td>51.758</td>
<td>9.928</td>
<td>0.100</td>
</tr>
<tr>
<td>4505</td>
<td>51.020</td>
<td>9.827</td>
<td>0.100</td>
</tr>
<tr>
<td>4506</td>
<td>50.321</td>
<td>9.617</td>
<td>0.100</td>
</tr>
<tr>
<td>4507</td>
<td>49.583</td>
<td>9.468</td>
<td>0.090</td>
</tr>
<tr>
<td>4508</td>
<td>48.829</td>
<td>9.468</td>
<td>0.100</td>
</tr>
<tr>
<td>4509</td>
<td>48.100</td>
<td>9.367</td>
<td>0.090</td>
</tr>
<tr>
<td>4510</td>
<td>47.365</td>
<td>9.197</td>
<td>0.080</td>
</tr>
<tr>
<td>4511</td>
<td>46.605</td>
<td>9.077</td>
<td>0.080</td>
</tr>
<tr>
<td>4512</td>
<td>45.873</td>
<td>8.897</td>
<td>0.100</td>
</tr>
<tr>
<td>4513</td>
<td>45.150</td>
<td>8.798</td>
<td>0.080</td>
</tr>
<tr>
<td>4514</td>
<td>44.430</td>
<td>8.647</td>
<td>0.100</td>
</tr>
<tr>
<td>4515</td>
<td>43.724</td>
<td>8.647</td>
<td>0.100</td>
</tr>
<tr>
<td>4516</td>
<td>43.020</td>
<td>8.547</td>
<td>0.080</td>
</tr>
<tr>
<td>4517</td>
<td>41.580</td>
<td>8.357</td>
<td>0.080</td>
</tr>
<tr>
<td>4518</td>
<td>40.875</td>
<td>8.317</td>
<td>0.080</td>
</tr>
<tr>
<td>4519</td>
<td>40.109</td>
<td>8.197</td>
<td>0.080</td>
</tr>
<tr>
<td>4520</td>
<td>39.327</td>
<td>8.068</td>
<td>0.080</td>
</tr>
<tr>
<td>4521</td>
<td>38.541</td>
<td>7.937</td>
<td>0.080</td>
</tr>
<tr>
<td>4522</td>
<td>37.623</td>
<td>7.798</td>
<td>0.070</td>
</tr>
<tr>
<td>4523</td>
<td>36.650</td>
<td>7.687</td>
<td>0.070</td>
</tr>
<tr>
<td>4524</td>
<td>35.711</td>
<td>7.557</td>
<td>0.060</td>
</tr>
<tr>
<td>4525</td>
<td>34.707</td>
<td>7.409</td>
<td>0.060</td>
</tr>
<tr>
<td>4526</td>
<td>33.707</td>
<td>7.258</td>
<td>0.060</td>
</tr>
<tr>
<td>4527</td>
<td>32.713</td>
<td>7.118</td>
<td>0.050</td>
</tr>
<tr>
<td>4528</td>
<td>31.734</td>
<td>6.978</td>
<td>0.050</td>
</tr>
<tr>
<td>4529</td>
<td>30.732</td>
<td>6.808</td>
<td>0.060</td>
</tr>
<tr>
<td>4530</td>
<td>29.713</td>
<td>6.658</td>
<td>0.050</td>
</tr>
<tr>
<td>4531</td>
<td>28.730</td>
<td>6.488</td>
<td>0.050</td>
</tr>
<tr>
<td>4532</td>
<td>27.685</td>
<td>6.298</td>
<td>0.050</td>
</tr>
<tr>
<td>4533</td>
<td>26.665</td>
<td>6.168</td>
<td>0.050</td>
</tr>
<tr>
<td>4534</td>
<td>25.639</td>
<td>6.018</td>
<td>0.050</td>
</tr>
<tr>
<td>4535</td>
<td>24.666</td>
<td>5.848</td>
<td>0.050</td>
</tr>
<tr>
<td>4536</td>
<td>23.599</td>
<td>5.678</td>
<td>0.050</td>
</tr>
<tr>
<td>4537</td>
<td>22.594</td>
<td>5.548</td>
<td>0.050</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Receiver 3 Shot line 3N</th>
<th>Shot Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>4539</td>
<td>21.511</td>
<td>5.428</td>
<td>0.040</td>
</tr>
<tr>
<td>4540</td>
<td>20.468</td>
<td>5.258</td>
<td>0.040</td>
</tr>
<tr>
<td>4541</td>
<td>19.485</td>
<td>5.139</td>
<td>0.040</td>
</tr>
<tr>
<td>4542</td>
<td>18.488</td>
<td>5.008</td>
<td>0.050</td>
</tr>
<tr>
<td>4543</td>
<td>17.469</td>
<td>4.888</td>
<td>0.040</td>
</tr>
<tr>
<td>4544</td>
<td>16.453</td>
<td>4.699</td>
<td>0.060</td>
</tr>
<tr>
<td>4545</td>
<td>15.458</td>
<td>4.539</td>
<td>0.050</td>
</tr>
<tr>
<td>4546</td>
<td>14.451</td>
<td>4.418</td>
<td>0.040</td>
</tr>
<tr>
<td>4547</td>
<td>13.500</td>
<td>4.298</td>
<td>0.040</td>
</tr>
<tr>
<td>4548</td>
<td>12.523</td>
<td>4.128</td>
<td>0.040</td>
</tr>
<tr>
<td>4549</td>
<td>11.505</td>
<td>3.958</td>
<td>0.040</td>
</tr>
<tr>
<td>4550</td>
<td>10.469</td>
<td>3.799</td>
<td>0.040</td>
</tr>
<tr>
<td>4551</td>
<td>9.468</td>
<td>3.618</td>
<td>0.030</td>
</tr>
<tr>
<td>4552</td>
<td>8.432</td>
<td>3.438</td>
<td>0.030</td>
</tr>
<tr>
<td>4553</td>
<td>7.422</td>
<td>3.268</td>
<td>0.040</td>
</tr>
<tr>
<td>4554</td>
<td>6.384</td>
<td>3.089</td>
<td>0.030</td>
</tr>
<tr>
<td>4555</td>
<td>5.326</td>
<td>2.879</td>
<td>0.030</td>
</tr>
<tr>
<td>4556</td>
<td>4.274</td>
<td>2.679</td>
<td>0.030</td>
</tr>
<tr>
<td>4557</td>
<td>3.193</td>
<td>2.469</td>
<td>0.030</td>
</tr>
<tr>
<td>4558</td>
<td>2.027</td>
<td>2.240</td>
<td>0.030</td>
</tr>
<tr>
<td>4563</td>
<td>1.827</td>
<td>2.209</td>
<td>0.030</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shot Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

235
<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>4497</td>
<td>22.094</td>
<td>5.655</td>
<td>0.050</td>
</tr>
<tr>
<td>4498</td>
<td>21.406</td>
<td>5.555</td>
<td>0.040</td>
</tr>
<tr>
<td>4499</td>
<td>20.674</td>
<td>5.465</td>
<td>0.050</td>
</tr>
<tr>
<td>4500</td>
<td>19.980</td>
<td>5.356</td>
<td>0.050</td>
</tr>
<tr>
<td>4501</td>
<td>19.226</td>
<td>5.206</td>
<td>0.050</td>
</tr>
<tr>
<td>4502</td>
<td>18.526</td>
<td>5.126</td>
<td>0.040</td>
</tr>
<tr>
<td>4503</td>
<td>17.808</td>
<td>5.016</td>
<td>0.050</td>
</tr>
<tr>
<td>4504</td>
<td>17.089</td>
<td>4.856</td>
<td>0.060</td>
</tr>
<tr>
<td>4505</td>
<td>16.329</td>
<td>4.775</td>
<td>0.050</td>
</tr>
<tr>
<td>4506</td>
<td>15.652</td>
<td>4.725</td>
<td>0.040</td>
</tr>
<tr>
<td>4507</td>
<td>14.897</td>
<td>4.626</td>
<td>0.040</td>
</tr>
<tr>
<td>4508</td>
<td>14.142</td>
<td>4.516</td>
<td>0.040</td>
</tr>
<tr>
<td>4509</td>
<td>13.429</td>
<td>4.426</td>
<td>0.040</td>
</tr>
<tr>
<td>4510</td>
<td>12.694</td>
<td>4.286</td>
<td>0.050</td>
</tr>
<tr>
<td>4511</td>
<td>11.945</td>
<td>4.156</td>
<td>0.040</td>
</tr>
<tr>
<td>4512</td>
<td>11.196</td>
<td>4.066</td>
<td>0.050</td>
</tr>
<tr>
<td>4513</td>
<td>10.499</td>
<td>3.987</td>
<td>0.030</td>
</tr>
<tr>
<td>4514</td>
<td>9.752</td>
<td>3.896</td>
<td>0.040</td>
</tr>
<tr>
<td>4515</td>
<td>9.046</td>
<td>3.776</td>
<td>0.040</td>
</tr>
<tr>
<td>4516</td>
<td>8.314</td>
<td>3.666</td>
<td>0.040</td>
</tr>
<tr>
<td>4517</td>
<td>7.634</td>
<td>3.556</td>
<td>0.040</td>
</tr>
<tr>
<td>4518</td>
<td>6.892</td>
<td>3.406</td>
<td>0.030</td>
</tr>
<tr>
<td>4519</td>
<td>6.139</td>
<td>3.286</td>
<td>0.040</td>
</tr>
<tr>
<td>4520</td>
<td>5.343</td>
<td>3.136</td>
<td>0.030</td>
</tr>
<tr>
<td>4521</td>
<td>4.587</td>
<td>3.007</td>
<td>0.030</td>
</tr>
<tr>
<td>4522</td>
<td>3.788</td>
<td>2.827</td>
<td>0.030</td>
</tr>
<tr>
<td>4523</td>
<td>2.775</td>
<td>2.688</td>
<td>0.030</td>
</tr>
<tr>
<td>4524</td>
<td>1.595</td>
<td>2.457</td>
<td>0.030</td>
</tr>
<tr>
<td>4525</td>
<td>1.634</td>
<td>2.507</td>
<td>0.030</td>
</tr>
<tr>
<td>4526</td>
<td>2.922</td>
<td>2.717</td>
<td>0.030</td>
</tr>
<tr>
<td>4527</td>
<td>3.991</td>
<td>2.897</td>
<td>0.030</td>
</tr>
<tr>
<td>4528</td>
<td>5.063</td>
<td>3.067</td>
<td>0.040</td>
</tr>
<tr>
<td>4529</td>
<td>6.043</td>
<td>3.238</td>
<td>0.030</td>
</tr>
<tr>
<td>4530</td>
<td>7.112</td>
<td>3.368</td>
<td>0.040</td>
</tr>
<tr>
<td>4531</td>
<td>8.166</td>
<td>3.508</td>
<td>0.050</td>
</tr>
<tr>
<td>4532</td>
<td>9.193</td>
<td>3.688</td>
<td>0.030</td>
</tr>
<tr>
<td>4533</td>
<td>10.201</td>
<td>3.808</td>
<td>0.030</td>
</tr>
<tr>
<td>4534</td>
<td>11.255</td>
<td>3.978</td>
<td>0.030</td>
</tr>
<tr>
<td>4535</td>
<td>12.340</td>
<td>4.128</td>
<td>0.030</td>
</tr>
<tr>
<td>4536</td>
<td>13.351</td>
<td>4.278</td>
<td>0.040</td>
</tr>
<tr>
<td>4537</td>
<td>14.409</td>
<td>4.449</td>
<td>0.050</td>
</tr>
<tr>
<td>4538</td>
<td>15.456</td>
<td>4.608</td>
<td>0.050</td>
</tr>
<tr>
<td>4539</td>
<td>16.507</td>
<td>4.748</td>
<td>0.040</td>
</tr>
<tr>
<td>4540</td>
<td>17.513</td>
<td>4.939</td>
<td>0.050</td>
</tr>
<tr>
<td>4541</td>
<td>18.611</td>
<td>5.089</td>
<td>0.040</td>
</tr>
<tr>
<td>4542</td>
<td>19.621</td>
<td>5.259</td>
<td>0.050</td>
</tr>
<tr>
<td>4543</td>
<td>20.737</td>
<td>5.389</td>
<td>0.050</td>
</tr>
<tr>
<td>4544</td>
<td>21.669</td>
<td>5.529</td>
<td>0.040</td>
</tr>
<tr>
<td>4545</td>
<td>22.622</td>
<td>5.659</td>
<td>0.060</td>
</tr>
<tr>
<td>4546</td>
<td>23.567</td>
<td>5.790</td>
<td>0.050</td>
</tr>
<tr>
<td>4547</td>
<td>24.601</td>
<td>5.959</td>
<td>0.060</td>
</tr>
<tr>
<td>4548</td>
<td>25.539</td>
<td>6.039</td>
<td>0.060</td>
</tr>
<tr>
<td>4549</td>
<td>26.581</td>
<td>6.159</td>
<td>0.060</td>
</tr>
<tr>
<td>4550</td>
<td>27.547</td>
<td>6.299</td>
<td>0.050</td>
</tr>
<tr>
<td>4551</td>
<td>28.514</td>
<td>6.449</td>
<td>0.050</td>
</tr>
<tr>
<td>4552</td>
<td>29.521</td>
<td>6.569</td>
<td>0.060</td>
</tr>
<tr>
<td>4553</td>
<td>30.505</td>
<td>6.650</td>
<td>0.060</td>
</tr>
<tr>
<td>4554</td>
<td>31.490</td>
<td>6.771</td>
<td>0.050</td>
</tr>
<tr>
<td>4555</td>
<td>32.448</td>
<td>6.881</td>
<td>0.060</td>
</tr>
<tr>
<td>4556</td>
<td>33.488</td>
<td>7.010</td>
<td>0.060</td>
</tr>
<tr>
<td>4557</td>
<td>34.478</td>
<td>7.171</td>
<td>0.060</td>
</tr>
<tr>
<td>4558</td>
<td>35.434</td>
<td>7.310</td>
<td>0.060</td>
</tr>
<tr>
<td>4559</td>
<td>36.452</td>
<td>7.470</td>
<td>0.060</td>
</tr>
</tbody>
</table>

Receiver 4 Shot line 2N

Shot  | Range  | Time  | Err  |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3336</td>
<td>40.928</td>
<td>8.180</td>
<td>0.090</td>
</tr>
<tr>
<td>3337</td>
<td>40.098</td>
<td>8.121</td>
<td>0.080</td>
</tr>
<tr>
<td>3338</td>
<td>39.252</td>
<td>8.071</td>
<td>0.070</td>
</tr>
<tr>
<td>3339</td>
<td>38.424</td>
<td>8.041</td>
<td>0.090</td>
</tr>
<tr>
<td>3340</td>
<td>37.560</td>
<td>7.887</td>
<td>0.060</td>
</tr>
<tr>
<td>3341</td>
<td>36.717</td>
<td>7.809</td>
<td>0.070</td>
</tr>
<tr>
<td>3342</td>
<td>35.888</td>
<td>7.721</td>
<td>0.070</td>
</tr>
<tr>
<td>3343</td>
<td>35.027</td>
<td>7.668</td>
<td>0.070</td>
</tr>
<tr>
<td>3344</td>
<td>34.195</td>
<td>7.551</td>
<td>0.060</td>
</tr>
<tr>
<td>3345</td>
<td>33.406</td>
<td>7.442</td>
<td>0.060</td>
</tr>
<tr>
<td>3346</td>
<td>32.555</td>
<td>7.328</td>
<td>0.060</td>
</tr>
<tr>
<td>3347</td>
<td>31.724</td>
<td>7.220</td>
<td>0.050</td>
</tr>
<tr>
<td>3348</td>
<td>30.854</td>
<td>7.080</td>
<td>0.050</td>
</tr>
<tr>
<td>3349</td>
<td>30.007</td>
<td>6.974</td>
<td>0.050</td>
</tr>
<tr>
<td>3350</td>
<td>29.160</td>
<td>6.858</td>
<td>0.050</td>
</tr>
<tr>
<td>3351</td>
<td>28.308</td>
<td>6.732</td>
<td>0.040</td>
</tr>
<tr>
<td>3352</td>
<td>27.493</td>
<td>6.639</td>
<td>0.050</td>
</tr>
<tr>
<td>3353</td>
<td>26.702</td>
<td>6.539</td>
<td>0.040</td>
</tr>
<tr>
<td>3354</td>
<td>25.962</td>
<td>6.419</td>
<td>0.060</td>
</tr>
<tr>
<td>3355</td>
<td>25.201</td>
<td>6.338</td>
<td>0.040</td>
</tr>
<tr>
<td>3356</td>
<td>23.663</td>
<td>6.081</td>
<td>0.050</td>
</tr>
<tr>
<td>3357</td>
<td>22.854</td>
<td>5.916</td>
<td>0.050</td>
</tr>
<tr>
<td>3358</td>
<td>22.049</td>
<td>5.813</td>
<td>0.040</td>
</tr>
<tr>
<td>3359</td>
<td>21.302</td>
<td>5.662</td>
<td>0.040</td>
</tr>
<tr>
<td>3360</td>
<td>20.585</td>
<td>5.540</td>
<td>0.040</td>
</tr>
<tr>
<td>3361</td>
<td>19.896</td>
<td>5.433</td>
<td>0.040</td>
</tr>
<tr>
<td>3362</td>
<td>19.192</td>
<td>5.325</td>
<td>0.040</td>
</tr>
<tr>
<td>3363</td>
<td>18.519</td>
<td>5.230</td>
<td>0.040</td>
</tr>
<tr>
<td>3364</td>
<td>17.842</td>
<td>5.133</td>
<td>0.040</td>
</tr>
</tbody>
</table>

236
<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>3365</td>
<td>17.849</td>
<td>5.095</td>
<td>0.040</td>
</tr>
<tr>
<td>3366</td>
<td>17.212</td>
<td>4.944</td>
<td>0.040</td>
</tr>
<tr>
<td>3367</td>
<td>16.613</td>
<td>4.914</td>
<td>0.040</td>
</tr>
<tr>
<td>3368</td>
<td>16.005</td>
<td>4.877</td>
<td>0.040</td>
</tr>
<tr>
<td>3369</td>
<td>15.467</td>
<td>4.733</td>
<td>0.040</td>
</tr>
<tr>
<td>3370</td>
<td>14.949</td>
<td>4.656</td>
<td>0.040</td>
</tr>
<tr>
<td>3371</td>
<td>14.471</td>
<td>4.518</td>
<td>0.040</td>
</tr>
<tr>
<td>3372</td>
<td>14.016</td>
<td>4.375</td>
<td>0.040</td>
</tr>
<tr>
<td>3373</td>
<td>13.651</td>
<td>4.302</td>
<td>0.040</td>
</tr>
<tr>
<td>3374</td>
<td>13.312</td>
<td>4.265</td>
<td>0.040</td>
</tr>
<tr>
<td>3375</td>
<td>12.883</td>
<td>4.234</td>
<td>0.040</td>
</tr>
<tr>
<td>3376</td>
<td>12.676</td>
<td>4.173</td>
<td>0.040</td>
</tr>
<tr>
<td>3377</td>
<td>12.649</td>
<td>4.196</td>
<td>0.050</td>
</tr>
<tr>
<td>3378</td>
<td>12.677</td>
<td>4.207</td>
<td>0.040</td>
</tr>
<tr>
<td>3379</td>
<td>12.839</td>
<td>4.175</td>
<td>0.040</td>
</tr>
<tr>
<td>3380</td>
<td>12.968</td>
<td>4.162</td>
<td>0.040</td>
</tr>
<tr>
<td>3381</td>
<td>13.135</td>
<td>4.203</td>
<td>0.040</td>
</tr>
<tr>
<td>3382</td>
<td>13.360</td>
<td>4.203</td>
<td>0.040</td>
</tr>
<tr>
<td>3383</td>
<td>13.601</td>
<td>4.245</td>
<td>0.040</td>
</tr>
<tr>
<td>3384</td>
<td>13.886</td>
<td>4.267</td>
<td>0.060</td>
</tr>
<tr>
<td>3385</td>
<td>14.237</td>
<td>4.324</td>
<td>0.050</td>
</tr>
<tr>
<td>3386</td>
<td>14.603</td>
<td>4.347</td>
<td>0.040</td>
</tr>
<tr>
<td>3387</td>
<td>14.982</td>
<td>4.404</td>
<td>0.040</td>
</tr>
<tr>
<td>3388</td>
<td>15.354</td>
<td>4.383</td>
<td>0.040</td>
</tr>
<tr>
<td>3389</td>
<td>15.823</td>
<td>4.424</td>
<td>0.040</td>
</tr>
<tr>
<td>3390</td>
<td>16.320</td>
<td>4.502</td>
<td>0.040</td>
</tr>
<tr>
<td>3391</td>
<td>16.838</td>
<td>4.576</td>
<td>0.040</td>
</tr>
<tr>
<td>3392</td>
<td>17.351</td>
<td>4.633</td>
<td>0.080</td>
</tr>
<tr>
<td>3393</td>
<td>17.919</td>
<td>4.696</td>
<td>0.080</td>
</tr>
<tr>
<td>3394</td>
<td>18.441</td>
<td>4.754</td>
<td>0.050</td>
</tr>
<tr>
<td>3395</td>
<td>19.604</td>
<td>4.816</td>
<td>0.050</td>
</tr>
<tr>
<td>3396</td>
<td>19.631</td>
<td>4.891</td>
<td>0.040</td>
</tr>
<tr>
<td>3400</td>
<td>20.254</td>
<td>4.979</td>
<td>0.050</td>
</tr>
<tr>
<td>3401</td>
<td>20.840</td>
<td>5.075</td>
<td>0.040</td>
</tr>
<tr>
<td>3402</td>
<td>21.447</td>
<td>5.202</td>
<td>0.050</td>
</tr>
<tr>
<td>3403</td>
<td>22.089</td>
<td>5.295</td>
<td>0.050</td>
</tr>
<tr>
<td>3404</td>
<td>22.750</td>
<td>5.402</td>
<td>0.060</td>
</tr>
<tr>
<td>3405</td>
<td>23.413</td>
<td>5.542</td>
<td>0.040</td>
</tr>
<tr>
<td>3406</td>
<td>24.045</td>
<td>5.683</td>
<td>0.040</td>
</tr>
<tr>
<td>3407</td>
<td>24.753</td>
<td>5.743</td>
<td>0.040</td>
</tr>
<tr>
<td>3408</td>
<td>25.437</td>
<td>5.899</td>
<td>0.040</td>
</tr>
<tr>
<td>3409</td>
<td>26.122</td>
<td>5.980</td>
<td>0.050</td>
</tr>
<tr>
<td>3410</td>
<td>26.835</td>
<td>6.124</td>
<td>0.060</td>
</tr>
<tr>
<td>3411</td>
<td>27.561</td>
<td>6.261</td>
<td>0.050</td>
</tr>
<tr>
<td>3412</td>
<td>28.295</td>
<td>6.342</td>
<td>0.050</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3413</td>
<td>29.041</td>
<td>6.506</td>
<td>0.050</td>
</tr>
<tr>
<td>3414</td>
<td>29.830</td>
<td>6.624</td>
<td>0.060</td>
</tr>
<tr>
<td>3415</td>
<td>30.845</td>
<td>6.732</td>
<td>0.060</td>
</tr>
<tr>
<td>3416</td>
<td>31.630</td>
<td>6.802</td>
<td>0.080</td>
</tr>
<tr>
<td>3417</td>
<td>32.423</td>
<td>6.872</td>
<td>0.080</td>
</tr>
<tr>
<td>3418</td>
<td>33.212</td>
<td>7.230</td>
<td>0.100</td>
</tr>
<tr>
<td>3419</td>
<td>33.974</td>
<td>7.488</td>
<td>0.100</td>
</tr>
<tr>
<td>3420</td>
<td>34.779</td>
<td>7.588</td>
<td>0.100</td>
</tr>
<tr>
<td>3421</td>
<td>35.518</td>
<td>7.516</td>
<td>0.150</td>
</tr>
</tbody>
</table>

Receiver 4 Shot line 4N
<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>3243</td>
<td>24.368</td>
<td>6.408</td>
<td>0.080</td>
</tr>
<tr>
<td>3244</td>
<td>23.691</td>
<td>6.202</td>
<td>0.080</td>
</tr>
<tr>
<td>3245</td>
<td>23.026</td>
<td>6.026</td>
<td>0.070</td>
</tr>
<tr>
<td>3246</td>
<td>22.362</td>
<td>5.946</td>
<td>0.090</td>
</tr>
<tr>
<td>3247</td>
<td>21.711</td>
<td>5.918</td>
<td>0.060</td>
</tr>
<tr>
<td>3248</td>
<td>21.053</td>
<td>5.942</td>
<td>0.060</td>
</tr>
<tr>
<td>3249</td>
<td>20.364</td>
<td>5.996</td>
<td>0.060</td>
</tr>
<tr>
<td>3250</td>
<td>19.637</td>
<td>5.893</td>
<td>0.060</td>
</tr>
<tr>
<td>3251</td>
<td>18.965</td>
<td>5.786</td>
<td>0.060</td>
</tr>
<tr>
<td>3252</td>
<td>18.283</td>
<td>5.614</td>
<td>0.050</td>
</tr>
<tr>
<td>3253</td>
<td>17.597</td>
<td>5.395</td>
<td>0.060</td>
</tr>
<tr>
<td>3254</td>
<td>16.921</td>
<td>5.244</td>
<td>0.050</td>
</tr>
<tr>
<td>3255</td>
<td>16.212</td>
<td>5.142</td>
<td>0.080</td>
</tr>
<tr>
<td>3256</td>
<td>15.446</td>
<td>4.874</td>
<td>0.060</td>
</tr>
<tr>
<td>3257</td>
<td>14.659</td>
<td>4.736</td>
<td>0.060</td>
</tr>
<tr>
<td>3258</td>
<td>13.893</td>
<td>4.597</td>
<td>0.050</td>
</tr>
<tr>
<td>3259</td>
<td>13.323</td>
<td>4.431</td>
<td>0.050</td>
</tr>
<tr>
<td>3260</td>
<td>12.622</td>
<td>4.312</td>
<td>0.050</td>
</tr>
<tr>
<td>3261</td>
<td>11.873</td>
<td>4.200</td>
<td>0.050</td>
</tr>
<tr>
<td>3262</td>
<td>11.080</td>
<td>4.042</td>
<td>0.040</td>
</tr>
<tr>
<td>3263</td>
<td>10.171</td>
<td>3.803</td>
<td>0.040</td>
</tr>
<tr>
<td>3264</td>
<td>9.257</td>
<td>3.531</td>
<td>0.040</td>
</tr>
<tr>
<td>3265</td>
<td>8.316</td>
<td>3.288</td>
<td>0.040</td>
</tr>
<tr>
<td>3266</td>
<td>7.356</td>
<td>3.082</td>
<td>0.040</td>
</tr>
<tr>
<td>3267</td>
<td>6.394</td>
<td>2.810</td>
<td>0.050</td>
</tr>
<tr>
<td>3268</td>
<td>5.388</td>
<td>2.610</td>
<td>0.040</td>
</tr>
<tr>
<td>3269</td>
<td>4.431</td>
<td>2.482</td>
<td>0.040</td>
</tr>
<tr>
<td>3270</td>
<td>3.470</td>
<td>2.407</td>
<td>0.050</td>
</tr>
<tr>
<td>3271</td>
<td>2.522</td>
<td>2.207</td>
<td>0.030</td>
</tr>
<tr>
<td>3272</td>
<td>1.625</td>
<td>2.033</td>
<td>0.030</td>
</tr>
<tr>
<td>3273</td>
<td>0.870</td>
<td>1.833</td>
<td>0.030</td>
</tr>
<tr>
<td>3274</td>
<td>0.489</td>
<td>1.752</td>
<td>0.030</td>
</tr>
<tr>
<td>3275</td>
<td>1.131</td>
<td>1.897</td>
<td>0.030</td>
</tr>
<tr>
<td>3276</td>
<td>2.053</td>
<td>2.209</td>
<td>0.030</td>
</tr>
<tr>
<td>3277</td>
<td>2.950</td>
<td>2.389</td>
<td>0.030</td>
</tr>
<tr>
<td>3278</td>
<td>3.841</td>
<td>2.589</td>
<td>0.040</td>
</tr>
<tr>
<td>3279</td>
<td>4.780</td>
<td>2.784</td>
<td>0.040</td>
</tr>
<tr>
<td>3280</td>
<td>5.695</td>
<td>3.034</td>
<td>0.040</td>
</tr>
<tr>
<td>3281</td>
<td>6.417</td>
<td>3.154</td>
<td>0.050</td>
</tr>
<tr>
<td>3282</td>
<td>8.430</td>
<td>3.569</td>
<td>0.040</td>
</tr>
<tr>
<td>3284</td>
<td>9.328</td>
<td>3.736</td>
<td>0.040</td>
</tr>
<tr>
<td>3285</td>
<td>10.285</td>
<td>3.914</td>
<td>0.060</td>
</tr>
<tr>
<td>Receiver 5 Shot line 2N</td>
<td>Shot Range</td>
<td>Time</td>
<td>Err</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>3343</td>
<td>51.246</td>
<td>9.927</td>
<td>0.120</td>
</tr>
<tr>
<td>3344</td>
<td>50.555</td>
<td>9.810</td>
<td>0.150</td>
</tr>
<tr>
<td>3345</td>
<td>49.896</td>
<td>9.771</td>
<td>0.150</td>
</tr>
<tr>
<td>3346</td>
<td>49.207</td>
<td>9.666</td>
<td>0.150</td>
</tr>
<tr>
<td>3347</td>
<td>48.519</td>
<td>9.578</td>
<td>0.120</td>
</tr>
<tr>
<td>3348</td>
<td>47.807</td>
<td>9.488</td>
<td>0.120</td>
</tr>
<tr>
<td>3349</td>
<td>47.091</td>
<td>9.332</td>
<td>0.130</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Receiver 5 Shot line 2N</th>
<th>Shot Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>3351</td>
<td>45.730</td>
<td>9.140</td>
<td>0.150</td>
</tr>
<tr>
<td>3352</td>
<td>45.043</td>
<td>9.078</td>
<td>0.140</td>
</tr>
<tr>
<td>3353</td>
<td>44.374</td>
<td>9.088</td>
<td>0.130</td>
</tr>
<tr>
<td>3354</td>
<td>43.727</td>
<td>9.027</td>
<td>0.120</td>
</tr>
<tr>
<td>3355</td>
<td>43.078</td>
<td>8.926</td>
<td>0.100</td>
</tr>
<tr>
<td>3356</td>
<td>42.430</td>
<td>8.809</td>
<td>0.080</td>
</tr>
<tr>
<td>3357</td>
<td>41.777</td>
<td>8.679</td>
<td>0.100</td>
</tr>
<tr>
<td>3358</td>
<td>41.103</td>
<td>8.604</td>
<td>0.100</td>
</tr>
<tr>
<td>3359</td>
<td>40.439</td>
<td>8.511</td>
<td>0.080</td>
</tr>
<tr>
<td>3360</td>
<td>39.834</td>
<td>8.430</td>
<td>0.070</td>
</tr>
<tr>
<td>3361</td>
<td>39.271</td>
<td>8.388</td>
<td>0.070</td>
</tr>
<tr>
<td>3362</td>
<td>38.784</td>
<td>8.391</td>
<td>0.090</td>
</tr>
<tr>
<td>3363</td>
<td>38.337</td>
<td>8.262</td>
<td>0.070</td>
</tr>
<tr>
<td>3364</td>
<td>37.813</td>
<td>8.238</td>
<td>0.070</td>
</tr>
<tr>
<td>3365</td>
<td>37.300</td>
<td>8.103</td>
<td>0.060</td>
</tr>
<tr>
<td>3366</td>
<td>36.823</td>
<td>8.022</td>
<td>0.060</td>
</tr>
<tr>
<td>3367</td>
<td>36.315</td>
<td>7.962</td>
<td>0.060</td>
</tr>
<tr>
<td>3368</td>
<td>35.803</td>
<td>7.865</td>
<td>0.060</td>
</tr>
<tr>
<td>3369</td>
<td>35.265</td>
<td>7.761</td>
<td>0.050</td>
</tr>
<tr>
<td>3370</td>
<td>34.734</td>
<td>7.674</td>
<td>0.060</td>
</tr>
<tr>
<td>3371</td>
<td>34.229</td>
<td>7.555</td>
<td>0.060</td>
</tr>
<tr>
<td>3372</td>
<td>33.675</td>
<td>7.429</td>
<td>0.100</td>
</tr>
<tr>
<td>3373</td>
<td>33.166</td>
<td>7.230</td>
<td>0.070</td>
</tr>
<tr>
<td>3374</td>
<td>32.817</td>
<td>7.181</td>
<td>0.060</td>
</tr>
<tr>
<td>3375</td>
<td>32.309</td>
<td>7.151</td>
<td>0.080</td>
</tr>
<tr>
<td>3376</td>
<td>31.617</td>
<td>7.004</td>
<td>0.070</td>
</tr>
<tr>
<td>3377</td>
<td>31.167</td>
<td>6.920</td>
<td>0.060</td>
</tr>
<tr>
<td>3378</td>
<td>30.609</td>
<td>6.880</td>
<td>0.080</td>
</tr>
<tr>
<td>3379</td>
<td>30.167</td>
<td>6.992</td>
<td>0.080</td>
</tr>
<tr>
<td>3380</td>
<td>29.999</td>
<td>6.804</td>
<td>0.070</td>
</tr>
<tr>
<td>3381</td>
<td>29.942</td>
<td>6.951</td>
<td>0.060</td>
</tr>
<tr>
<td>3382</td>
<td>29.940</td>
<td>6.874</td>
<td>0.080</td>
</tr>
<tr>
<td>3383</td>
<td>29.966</td>
<td>6.821</td>
<td>0.060</td>
</tr>
<tr>
<td>3384</td>
<td>29.987</td>
<td>6.780</td>
<td>0.080</td>
</tr>
<tr>
<td>3385</td>
<td>30.047</td>
<td>6.701</td>
<td>0.060</td>
</tr>
<tr>
<td>3386</td>
<td>30.071</td>
<td>6.609</td>
<td>0.060</td>
</tr>
<tr>
<td>3387</td>
<td>30.156</td>
<td>6.613</td>
<td>0.050</td>
</tr>
<tr>
<td>3388</td>
<td>30.348</td>
<td>6.602</td>
<td>0.050</td>
</tr>
<tr>
<td>3389</td>
<td>30.495</td>
<td>6.560</td>
<td>0.050</td>
</tr>
<tr>
<td>3390</td>
<td>30.656</td>
<td>6.682</td>
<td>0.050</td>
</tr>
<tr>
<td>3391</td>
<td>30.825</td>
<td>6.718</td>
<td>0.050</td>
</tr>
<tr>
<td>3392</td>
<td>31.019</td>
<td>6.776</td>
<td>0.050</td>
</tr>
<tr>
<td>3393</td>
<td>31.427</td>
<td>6.862</td>
<td>0.060</td>
</tr>
<tr>
<td>3394</td>
<td>31.738</td>
<td>6.949</td>
<td>0.050</td>
</tr>
<tr>
<td>3395</td>
<td>32.032</td>
<td>7.012</td>
<td>0.060</td>
</tr>
<tr>
<td>3396</td>
<td>32.320</td>
<td>7.058</td>
<td>0.060</td>
</tr>
<tr>
<td>3397</td>
<td>32.679</td>
<td>7.098</td>
<td>0.050</td>
</tr>
</tbody>
</table>

239
<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>3252</td>
<td>39.357</td>
<td>8.371</td>
<td>0.100</td>
</tr>
<tr>
<td>3253</td>
<td>38.643</td>
<td>8.272</td>
<td>0.100</td>
</tr>
<tr>
<td>3254</td>
<td>37.935</td>
<td>8.121</td>
<td>0.100</td>
</tr>
<tr>
<td>3255</td>
<td>37.187</td>
<td>7.989</td>
<td>0.100</td>
</tr>
<tr>
<td>3256</td>
<td>36.446</td>
<td>7.841</td>
<td>0.090</td>
</tr>
<tr>
<td>3257</td>
<td>35.674</td>
<td>7.703</td>
<td>0.090</td>
</tr>
<tr>
<td>3258</td>
<td>34.900</td>
<td>7.514</td>
<td>0.080</td>
</tr>
<tr>
<td>3259</td>
<td>34.134</td>
<td>7.418</td>
<td>0.080</td>
</tr>
<tr>
<td>3260</td>
<td>33.395</td>
<td>7.269</td>
<td>0.080</td>
</tr>
<tr>
<td>3261</td>
<td>32.626</td>
<td>7.198</td>
<td>0.070</td>
</tr>
<tr>
<td>3262</td>
<td>31.781</td>
<td>7.080</td>
<td>0.070</td>
</tr>
<tr>
<td>3263</td>
<td>30.905</td>
<td>6.901</td>
<td>0.070</td>
</tr>
<tr>
<td>3264</td>
<td>29.985</td>
<td>6.679</td>
<td>0.070</td>
</tr>
<tr>
<td>3265</td>
<td>29.095</td>
<td>6.476</td>
<td>0.070</td>
</tr>
<tr>
<td>3266</td>
<td>28.211</td>
<td>6.340</td>
<td>0.070</td>
</tr>
<tr>
<td>3267</td>
<td>27.298</td>
<td>6.138</td>
<td>0.060</td>
</tr>
<tr>
<td>3268</td>
<td>26.349</td>
<td>5.938</td>
<td>0.060</td>
</tr>
<tr>
<td>3269</td>
<td>25.429</td>
<td>5.811</td>
<td>0.070</td>
</tr>
<tr>
<td>3271</td>
<td>23.528</td>
<td>5.666</td>
<td>0.060</td>
</tr>
<tr>
<td>3272</td>
<td>22.587</td>
<td>5.612</td>
<td>0.050</td>
</tr>
<tr>
<td>3273</td>
<td>21.668</td>
<td>5.512</td>
<td>0.050</td>
</tr>
<tr>
<td>3274</td>
<td>20.551</td>
<td>5.401</td>
<td>0.050</td>
</tr>
<tr>
<td>3275</td>
<td>19.640</td>
<td>5.256</td>
<td>0.060</td>
</tr>
<tr>
<td>3276</td>
<td>18.727</td>
<td>5.178</td>
<td>0.050</td>
</tr>
<tr>
<td>3277</td>
<td>17.875</td>
<td>5.049</td>
<td>0.050</td>
</tr>
<tr>
<td>3278</td>
<td>16.962</td>
<td>5.009</td>
<td>0.070</td>
</tr>
<tr>
<td>3279</td>
<td>16.035</td>
<td>4.894</td>
<td>0.040</td>
</tr>
<tr>
<td>3280</td>
<td>15.139</td>
<td>4.754</td>
<td>0.050</td>
</tr>
<tr>
<td>3281</td>
<td>14.422</td>
<td>4.634</td>
<td>0.040</td>
</tr>
<tr>
<td>3282</td>
<td>13.357</td>
<td>4.442</td>
<td>0.040</td>
</tr>
<tr>
<td>3283</td>
<td>12.441</td>
<td>4.269</td>
<td>0.040</td>
</tr>
<tr>
<td>3284</td>
<td>11.543</td>
<td>4.097</td>
<td>0.050</td>
</tr>
<tr>
<td>Shot</td>
<td>Range</td>
<td>Time</td>
<td>Err</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>3372</td>
<td>35.124</td>
<td>7.577</td>
<td>0.110</td>
</tr>
<tr>
<td>3375</td>
<td>32.686</td>
<td>7.235</td>
<td>0.120</td>
</tr>
<tr>
<td>3376</td>
<td>31.892</td>
<td>7.239</td>
<td>0.100</td>
</tr>
<tr>
<td>3377</td>
<td>30.923</td>
<td>7.108</td>
<td>0.100</td>
</tr>
<tr>
<td>3378</td>
<td>30.431</td>
<td>7.039</td>
<td>0.100</td>
</tr>
<tr>
<td>3379</td>
<td>29.771</td>
<td>7.021</td>
<td>0.100</td>
</tr>
<tr>
<td>3380</td>
<td>29.017</td>
<td>6.800</td>
<td>0.080</td>
</tr>
<tr>
<td>3381</td>
<td>28.448</td>
<td>6.701</td>
<td>0.090</td>
</tr>
<tr>
<td>3382</td>
<td>27.666</td>
<td>6.540</td>
<td>0.100</td>
</tr>
<tr>
<td>3383</td>
<td>26.945</td>
<td>6.441</td>
<td>0.120</td>
</tr>
<tr>
<td>3384</td>
<td>26.393</td>
<td>6.535</td>
<td>0.100</td>
</tr>
<tr>
<td>3385</td>
<td>25.926</td>
<td>6.257</td>
<td>0.100</td>
</tr>
<tr>
<td>3386</td>
<td>24.981</td>
<td>6.134</td>
<td>0.080</td>
</tr>
<tr>
<td>3387</td>
<td>24.305</td>
<td>6.039</td>
<td>0.080</td>
</tr>
<tr>
<td>3388</td>
<td>23.583</td>
<td>5.862</td>
<td>0.090</td>
</tr>
<tr>
<td>3389</td>
<td>22.971</td>
<td>5.775</td>
<td>0.080</td>
</tr>
<tr>
<td>3390</td>
<td>22.584</td>
<td>5.637</td>
<td>0.100</td>
</tr>
<tr>
<td>3391</td>
<td>21.638</td>
<td>5.523</td>
<td>0.080</td>
</tr>
<tr>
<td>3392</td>
<td>21.160</td>
<td>5.454</td>
<td>0.090</td>
</tr>
<tr>
<td>3393</td>
<td>20.648</td>
<td>5.337</td>
<td>0.090</td>
</tr>
<tr>
<td>3394</td>
<td>20.049</td>
<td>5.260</td>
<td>0.080</td>
</tr>
<tr>
<td>3395</td>
<td>19.528</td>
<td>5.235</td>
<td>0.080</td>
</tr>
<tr>
<td>3396</td>
<td>18.809</td>
<td>5.184</td>
<td>0.080</td>
</tr>
<tr>
<td>3397</td>
<td>18.302</td>
<td>4.989</td>
<td>0.080</td>
</tr>
<tr>
<td>3398</td>
<td>17.791</td>
<td>4.928</td>
<td>0.100</td>
</tr>
<tr>
<td>3399</td>
<td>17.426</td>
<td>4.848</td>
<td>0.100</td>
</tr>
<tr>
<td>3400</td>
<td>16.780</td>
<td>4.800</td>
<td>0.100</td>
</tr>
<tr>
<td>3401</td>
<td>16.262</td>
<td>4.690</td>
<td>0.090</td>
</tr>
<tr>
<td>3402</td>
<td>15.939</td>
<td>4.704</td>
<td>0.080</td>
</tr>
<tr>
<td>3403</td>
<td>15.449</td>
<td>4.663</td>
<td>0.100</td>
</tr>
<tr>
<td>3404</td>
<td>15.096</td>
<td>4.480</td>
<td>0.070</td>
</tr>
<tr>
<td>3405</td>
<td>14.770</td>
<td>4.610</td>
<td>0.080</td>
</tr>
<tr>
<td>3406</td>
<td>14.530</td>
<td>4.604</td>
<td>0.080</td>
</tr>
<tr>
<td>3407</td>
<td>14.274</td>
<td>4.574</td>
<td>0.090</td>
</tr>
<tr>
<td>3408</td>
<td>14.157</td>
<td>4.558</td>
<td>0.070</td>
</tr>
<tr>
<td>3409</td>
<td>14.030</td>
<td>4.514</td>
<td>0.060</td>
</tr>
<tr>
<td>3410</td>
<td>13.859</td>
<td>4.490</td>
<td>0.100</td>
</tr>
<tr>
<td>3411</td>
<td>13.737</td>
<td>4.544</td>
<td>0.100</td>
</tr>
<tr>
<td>3412</td>
<td>13.779</td>
<td>4.422</td>
<td>0.110</td>
</tr>
<tr>
<td>3413</td>
<td>13.974</td>
<td>4.442</td>
<td>0.080</td>
</tr>
<tr>
<td>3414</td>
<td>14.158</td>
<td>4.467</td>
<td>0.090</td>
</tr>
<tr>
<td>3415</td>
<td>14.252</td>
<td>4.453</td>
<td>0.080</td>
</tr>
</tbody>
</table>

Receiver 205 Shot line 2N

Receiver 201 Shot line 1N

<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>3416</td>
<td>14.548</td>
<td>4.474</td>
<td>0.090</td>
</tr>
<tr>
<td>3417</td>
<td>14.819</td>
<td>4.555</td>
<td>0.090</td>
</tr>
<tr>
<td>3418</td>
<td>15.121</td>
<td>4.553</td>
<td>0.090</td>
</tr>
<tr>
<td>3419</td>
<td>15.551</td>
<td>4.683</td>
<td>0.100</td>
</tr>
<tr>
<td>3420</td>
<td>15.889</td>
<td>4.870</td>
<td>0.100</td>
</tr>
<tr>
<td>3421</td>
<td>16.341</td>
<td>4.964</td>
<td>0.120</td>
</tr>
<tr>
<td>3422</td>
<td>35.124</td>
<td>7.577</td>
<td>0.110</td>
</tr>
<tr>
<td>3423</td>
<td>32.686</td>
<td>7.235</td>
<td>0.120</td>
</tr>
<tr>
<td>3424</td>
<td>31.892</td>
<td>7.239</td>
<td>0.100</td>
</tr>
<tr>
<td>3425</td>
<td>30.923</td>
<td>7.108</td>
<td>0.100</td>
</tr>
<tr>
<td>3426</td>
<td>30.431</td>
<td>7.039</td>
<td>0.100</td>
</tr>
<tr>
<td>3427</td>
<td>29.771</td>
<td>7.021</td>
<td>0.100</td>
</tr>
<tr>
<td>3428</td>
<td>29.017</td>
<td>6.800</td>
<td>0.080</td>
</tr>
<tr>
<td>3429</td>
<td>28.448</td>
<td>6.701</td>
<td>0.090</td>
</tr>
<tr>
<td>3430</td>
<td>27.666</td>
<td>6.540</td>
<td>0.100</td>
</tr>
<tr>
<td>3431</td>
<td>26.945</td>
<td>6.441</td>
<td>0.120</td>
</tr>
<tr>
<td>3432</td>
<td>26.393</td>
<td>6.535</td>
<td>0.100</td>
</tr>
<tr>
<td>3433</td>
<td>25.926</td>
<td>6.257</td>
<td>0.100</td>
</tr>
<tr>
<td>3434</td>
<td>24.981</td>
<td>6.134</td>
<td>0.080</td>
</tr>
<tr>
<td>3435</td>
<td>24.305</td>
<td>6.039</td>
<td>0.080</td>
</tr>
<tr>
<td>3436</td>
<td>23.583</td>
<td>5.862</td>
<td>0.090</td>
</tr>
<tr>
<td>3437</td>
<td>22.971</td>
<td>5.775</td>
<td>0.080</td>
</tr>
<tr>
<td>3438</td>
<td>22.584</td>
<td>5.637</td>
<td>0.100</td>
</tr>
<tr>
<td>3439</td>
<td>21.638</td>
<td>5.523</td>
<td>0.080</td>
</tr>
<tr>
<td>3440</td>
<td>21.160</td>
<td>5.454</td>
<td>0.090</td>
</tr>
<tr>
<td>3441</td>
<td>20.648</td>
<td>5.337</td>
<td>0.090</td>
</tr>
<tr>
<td>3442</td>
<td>20.049</td>
<td>5.260</td>
<td>0.080</td>
</tr>
<tr>
<td>3443</td>
<td>19.528</td>
<td>5.235</td>
<td>0.080</td>
</tr>
<tr>
<td>3444</td>
<td>18.809</td>
<td>5.184</td>
<td>0.080</td>
</tr>
<tr>
<td>3445</td>
<td>18.302</td>
<td>4.989</td>
<td>0.080</td>
</tr>
<tr>
<td>3446</td>
<td>17.791</td>
<td>4.928</td>
<td>0.100</td>
</tr>
<tr>
<td>3447</td>
<td>17.426</td>
<td>4.848</td>
<td>0.100</td>
</tr>
<tr>
<td>3448</td>
<td>16.780</td>
<td>4.800</td>
<td>0.100</td>
</tr>
<tr>
<td>3449</td>
<td>16.262</td>
<td>4.690</td>
<td>0.090</td>
</tr>
<tr>
<td>3450</td>
<td>15.939</td>
<td>4.704</td>
<td>0.080</td>
</tr>
<tr>
<td>3451</td>
<td>15.449</td>
<td>4.663</td>
<td>0.100</td>
</tr>
<tr>
<td>3452</td>
<td>15.096</td>
<td>4.480</td>
<td>0.070</td>
</tr>
<tr>
<td>3453</td>
<td>14.770</td>
<td>4.610</td>
<td>0.080</td>
</tr>
<tr>
<td>3454</td>
<td>14.530</td>
<td>4.604</td>
<td>0.080</td>
</tr>
<tr>
<td>3455</td>
<td>14.274</td>
<td>4.574</td>
<td>0.090</td>
</tr>
<tr>
<td>3456</td>
<td>14.157</td>
<td>4.558</td>
<td>0.070</td>
</tr>
<tr>
<td>3457</td>
<td>14.030</td>
<td>4.514</td>
<td>0.060</td>
</tr>
<tr>
<td>3458</td>
<td>13.859</td>
<td>4.490</td>
<td>0.100</td>
</tr>
<tr>
<td>3459</td>
<td>13.737</td>
<td>4.544</td>
<td>0.100</td>
</tr>
<tr>
<td>3460</td>
<td>13.779</td>
<td>4.422</td>
<td>0.110</td>
</tr>
<tr>
<td>3461</td>
<td>13.974</td>
<td>4.442</td>
<td>0.080</td>
</tr>
<tr>
<td>3462</td>
<td>14.158</td>
<td>4.467</td>
<td>0.090</td>
</tr>
</tbody>
</table>

241
### Receiver 210 Shot line 2N

<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>3342</td>
<td>21.949</td>
<td>5.904</td>
<td>0.070</td>
</tr>
<tr>
<td>3344</td>
<td>20.169</td>
<td>5.720</td>
<td>0.090</td>
</tr>
<tr>
<td>3345</td>
<td>19.424</td>
<td>5.620</td>
<td>0.100</td>
</tr>
<tr>
<td>3346</td>
<td>18.587</td>
<td>5.407</td>
<td>0.080</td>
</tr>
<tr>
<td>3347</td>
<td>17.483</td>
<td>5.251</td>
<td>0.080</td>
</tr>
<tr>
<td>3348</td>
<td>16.489</td>
<td>5.139</td>
<td>0.090</td>
</tr>
<tr>
<td>3349</td>
<td>15.635</td>
<td>4.986</td>
<td>0.080</td>
</tr>
<tr>
<td>3350</td>
<td>14.696</td>
<td>4.876</td>
<td>0.070</td>
</tr>
<tr>
<td>3351</td>
<td>13.375</td>
<td>4.791</td>
<td>0.081</td>
</tr>
<tr>
<td>3352</td>
<td>12.871</td>
<td>4.615</td>
<td>0.090</td>
</tr>
<tr>
<td>3353</td>
<td>11.989</td>
<td>4.489</td>
<td>0.070</td>
</tr>
<tr>
<td>3354</td>
<td>11.157</td>
<td>4.415</td>
<td>0.090</td>
</tr>
<tr>
<td>3355</td>
<td>10.228</td>
<td>4.211</td>
<td>0.080</td>
</tr>
<tr>
<td>3356</td>
<td>9.442</td>
<td>4.068</td>
<td>0.080</td>
</tr>
<tr>
<td>3357</td>
<td>8.554</td>
<td>3.908</td>
<td>0.080</td>
</tr>
<tr>
<td>3359</td>
<td>6.864</td>
<td>3.562</td>
<td>0.060</td>
</tr>
<tr>
<td>3360</td>
<td>5.942</td>
<td>3.404</td>
<td>0.040</td>
</tr>
<tr>
<td>3362</td>
<td>4.507</td>
<td>3.230</td>
<td>0.040</td>
</tr>
<tr>
<td>3363</td>
<td>3.766</td>
<td>3.072</td>
<td>0.040</td>
</tr>
<tr>
<td>3364</td>
<td>3.150</td>
<td>2.915</td>
<td>0.040</td>
</tr>
<tr>
<td>3365</td>
<td>2.907</td>
<td>2.812</td>
<td>0.040</td>
</tr>
<tr>
<td>3366</td>
<td>2.805</td>
<td>2.781</td>
<td>0.040</td>
</tr>
<tr>
<td>3368</td>
<td>3.345</td>
<td>2.849</td>
<td>0.050</td>
</tr>
<tr>
<td>3369</td>
<td>3.904</td>
<td>2.933</td>
<td>0.060</td>
</tr>
<tr>
<td>3370</td>
<td>4.525</td>
<td>3.056</td>
<td>0.060</td>
</tr>
<tr>
<td>3371</td>
<td>5.372</td>
<td>3.190</td>
<td>0.060</td>
</tr>
<tr>
<td>3372</td>
<td>6.212</td>
<td>3.306</td>
<td>0.060</td>
</tr>
<tr>
<td>3373</td>
<td>7.016</td>
<td>3.427</td>
<td>0.080</td>
</tr>
<tr>
<td>3376</td>
<td>9.564</td>
<td>3.928</td>
<td>0.080</td>
</tr>
<tr>
<td>3377</td>
<td>10.316</td>
<td>4.059</td>
<td>0.100</td>
</tr>
<tr>
<td>3379</td>
<td>11.898</td>
<td>4.310</td>
<td>0.080</td>
</tr>
<tr>
<td>3380</td>
<td>12.711</td>
<td>4.421</td>
<td>0.110</td>
</tr>
<tr>
<td>3381</td>
<td>13.506</td>
<td>4.593</td>
<td>0.090</td>
</tr>
<tr>
<td>3382</td>
<td>14.435</td>
<td>4.690</td>
<td>0.110</td>
</tr>
<tr>
<td>3383</td>
<td>15.130</td>
<td>4.772</td>
<td>0.120</td>
</tr>
<tr>
<td>3384</td>
<td>15.692</td>
<td>4.860</td>
<td>0.090</td>
</tr>
<tr>
<td>3386</td>
<td>17.353</td>
<td>5.052</td>
<td>0.090</td>
</tr>
<tr>
<td>3387</td>
<td>18.333</td>
<td>5.222</td>
<td>0.100</td>
</tr>
<tr>
<td>3388</td>
<td>19.080</td>
<td>5.280</td>
<td>0.100</td>
</tr>
<tr>
<td>3390</td>
<td>20.870</td>
<td>5.518</td>
<td>0.080</td>
</tr>
<tr>
<td>3391</td>
<td>21.531</td>
<td>5.589</td>
<td>0.100</td>
</tr>
<tr>
<td>3392</td>
<td>22.109</td>
<td>5.654</td>
<td>0.100</td>
</tr>
</tbody>
</table>

### Receiver 210 Shot line 4N

<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>4587</td>
<td>32.875</td>
<td>7.577</td>
<td>0.150</td>
</tr>
<tr>
<td>4588</td>
<td>32.294</td>
<td>7.439</td>
<td>0.100</td>
</tr>
<tr>
<td>4589</td>
<td>31.693</td>
<td>7.349</td>
<td>0.100</td>
</tr>
<tr>
<td>4590</td>
<td>30.820</td>
<td>7.185</td>
<td>0.090</td>
</tr>
<tr>
<td>4592</td>
<td>29.970</td>
<td>6.989</td>
<td>0.090</td>
</tr>
<tr>
<td>4594</td>
<td>28.893</td>
<td>6.896</td>
<td>0.080</td>
</tr>
<tr>
<td>4595</td>
<td>28.404</td>
<td>6.802</td>
<td>0.090</td>
</tr>
<tr>
<td>4596</td>
<td>27.940</td>
<td>6.749</td>
<td>0.080</td>
</tr>
<tr>
<td>4597</td>
<td>27.143</td>
<td>6.595</td>
<td>0.100</td>
</tr>
<tr>
<td>4598</td>
<td>26.418</td>
<td>6.491</td>
<td>0.080</td>
</tr>
<tr>
<td>4599</td>
<td>25.811</td>
<td>6.448</td>
<td>0.090</td>
</tr>
<tr>
<td>4600</td>
<td>25.171</td>
<td>6.277</td>
<td>0.070</td>
</tr>
<tr>
<td>4607</td>
<td>20.654</td>
<td>5.702</td>
<td>0.100</td>
</tr>
<tr>
<td>4609</td>
<td>19.229</td>
<td>5.563</td>
<td>0.100</td>
</tr>
<tr>
<td>4610</td>
<td>18.616</td>
<td>5.423</td>
<td>0.080</td>
</tr>
<tr>
<td>4613</td>
<td>17.135</td>
<td>5.024</td>
<td>0.060</td>
</tr>
<tr>
<td>4616</td>
<td>15.732</td>
<td>4.683</td>
<td>0.060</td>
</tr>
<tr>
<td>4617</td>
<td>15.288</td>
<td>4.663</td>
<td>0.060</td>
</tr>
<tr>
<td>4618</td>
<td>14.981</td>
<td>4.596</td>
<td>0.060</td>
</tr>
<tr>
<td>4619</td>
<td>14.631</td>
<td>4.646</td>
<td>0.070</td>
</tr>
<tr>
<td>4620</td>
<td>14.298</td>
<td>4.605</td>
<td>0.080</td>
</tr>
<tr>
<td>4621</td>
<td>14.058</td>
<td>4.517</td>
<td>0.060</td>
</tr>
<tr>
<td>4622</td>
<td>13.820</td>
<td>4.460</td>
<td>0.100</td>
</tr>
<tr>
<td>Shot</td>
<td>Range</td>
<td>Time</td>
<td>Err</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>4491</td>
<td>39.163</td>
<td>8.210</td>
<td>0.100</td>
</tr>
<tr>
<td>4492</td>
<td>38.418</td>
<td>8.085</td>
<td>0.088</td>
</tr>
<tr>
<td>4493</td>
<td>37.676</td>
<td>8.035</td>
<td>0.100</td>
</tr>
<tr>
<td>4494</td>
<td>36.947</td>
<td>7.885</td>
<td>0.088</td>
</tr>
<tr>
<td>4495</td>
<td>36.220</td>
<td>7.785</td>
<td>0.088</td>
</tr>
<tr>
<td>4496</td>
<td>35.504</td>
<td>7.598</td>
<td>0.075</td>
</tr>
<tr>
<td>4497</td>
<td>34.766</td>
<td>7.573</td>
<td>0.088</td>
</tr>
<tr>
<td>4498</td>
<td>34.040</td>
<td>7.522</td>
<td>0.075</td>
</tr>
<tr>
<td>4499</td>
<td>33.303</td>
<td>7.372</td>
<td>0.075</td>
</tr>
<tr>
<td>4500</td>
<td>32.579</td>
<td>7.272</td>
<td>0.063</td>
</tr>
<tr>
<td>4501</td>
<td>31.855</td>
<td>7.173</td>
<td>0.063</td>
</tr>
<tr>
<td>4502</td>
<td>31.120</td>
<td>7.060</td>
<td>0.063</td>
</tr>
<tr>
<td>4503</td>
<td>30.385</td>
<td>6.885</td>
<td>0.063</td>
</tr>
<tr>
<td>4504</td>
<td>29.656</td>
<td>6.860</td>
<td>0.100</td>
</tr>
<tr>
<td>4505</td>
<td>28.913</td>
<td>6.760</td>
<td>0.075</td>
</tr>
<tr>
<td>4506</td>
<td>28.182</td>
<td>6.760</td>
<td>0.063</td>
</tr>
<tr>
<td>4507</td>
<td>27.434</td>
<td>6.585</td>
<td>0.063</td>
</tr>
<tr>
<td>4508</td>
<td>26.688</td>
<td>6.460</td>
<td>0.050</td>
</tr>
<tr>
<td>4509</td>
<td>25.941</td>
<td>6.423</td>
<td>0.075</td>
</tr>
<tr>
<td>4510</td>
<td>25.204</td>
<td>6.298</td>
<td>0.063</td>
</tr>
<tr>
<td>4511</td>
<td>24.453</td>
<td>6.173</td>
<td>0.063</td>
</tr>
<tr>
<td>Shot</td>
<td>Range</td>
<td>Time</td>
<td>Err</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>4561</td>
<td>23.328</td>
<td>5.723</td>
<td>0.088</td>
</tr>
<tr>
<td>4562</td>
<td>24.223</td>
<td>5.835</td>
<td>0.075</td>
</tr>
<tr>
<td>4563</td>
<td>25.161</td>
<td>5.960</td>
<td>0.088</td>
</tr>
<tr>
<td>4564</td>
<td>23.759</td>
<td>6.460</td>
<td>0.100</td>
</tr>
<tr>
<td>4565</td>
<td>23.018</td>
<td>6.310</td>
<td>0.088</td>
</tr>
<tr>
<td>4566</td>
<td>22.280</td>
<td>6.298</td>
<td>0.088</td>
</tr>
<tr>
<td>4567</td>
<td>21.567</td>
<td>6.110</td>
<td>0.100</td>
</tr>
<tr>
<td>4568</td>
<td>20.841</td>
<td>5.897</td>
<td>0.075</td>
</tr>
<tr>
<td>4569</td>
<td>20.113</td>
<td>5.772</td>
<td>0.100</td>
</tr>
<tr>
<td>4570</td>
<td>18.655</td>
<td>5.598</td>
<td>0.075</td>
</tr>
<tr>
<td>4571</td>
<td>17.899</td>
<td>5.448</td>
<td>0.075</td>
</tr>
<tr>
<td>4572</td>
<td>17.161</td>
<td>5.298</td>
<td>0.075</td>
</tr>
<tr>
<td>4573</td>
<td>16.402</td>
<td>5.147</td>
<td>0.075</td>
</tr>
<tr>
<td>4574</td>
<td>15.648</td>
<td>5.022</td>
<td>0.063</td>
</tr>
<tr>
<td>4575</td>
<td>14.873</td>
<td>4.860</td>
<td>0.075</td>
</tr>
<tr>
<td>4576</td>
<td>14.088</td>
<td>4.685</td>
<td>0.088</td>
</tr>
<tr>
<td>4577</td>
<td>13.315</td>
<td>4.585</td>
<td>0.063</td>
</tr>
<tr>
<td>4578</td>
<td>12.653</td>
<td>4.485</td>
<td>0.063</td>
</tr>
<tr>
<td>4579</td>
<td>12.002</td>
<td>4.385</td>
<td>0.063</td>
</tr>
<tr>
<td>4580</td>
<td>11.409</td>
<td>4.272</td>
<td>0.075</td>
</tr>
<tr>
<td>4581</td>
<td>10.790</td>
<td>4.198</td>
<td>0.063</td>
</tr>
<tr>
<td>4582</td>
<td>10.211</td>
<td>4.160</td>
<td>0.075</td>
</tr>
<tr>
<td>4583</td>
<td>9.639</td>
<td>4.022</td>
<td>0.063</td>
</tr>
<tr>
<td>4584</td>
<td>9.092</td>
<td>3.947</td>
<td>0.063</td>
</tr>
<tr>
<td>4585</td>
<td>8.592</td>
<td>3.860</td>
<td>0.063</td>
</tr>
<tr>
<td>4586</td>
<td>8.020</td>
<td>3.772</td>
<td>0.063</td>
</tr>
<tr>
<td>4587</td>
<td>7.353</td>
<td>3.735</td>
<td>0.075</td>
</tr>
<tr>
<td>4588</td>
<td>6.864</td>
<td>3.635</td>
<td>0.063</td>
</tr>
<tr>
<td>4589</td>
<td>6.535</td>
<td>3.560</td>
<td>0.063</td>
</tr>
<tr>
<td>4590</td>
<td>6.337</td>
<td>3.535</td>
<td>0.063</td>
</tr>
<tr>
<td>4591</td>
<td>6.252</td>
<td>3.497</td>
<td>0.075</td>
</tr>
<tr>
<td>4592</td>
<td>6.218</td>
<td>3.510</td>
<td>0.063</td>
</tr>
<tr>
<td>4593</td>
<td>6.216</td>
<td>3.522</td>
<td>0.063</td>
</tr>
<tr>
<td>4594</td>
<td>6.221</td>
<td>3.497</td>
<td>0.075</td>
</tr>
<tr>
<td>4595</td>
<td>6.156</td>
<td>3.485</td>
<td>0.063</td>
</tr>
<tr>
<td>4596</td>
<td>6.090</td>
<td>3.472</td>
<td>0.050</td>
</tr>
<tr>
<td>4597</td>
<td>6.215</td>
<td>3.472</td>
<td>0.063</td>
</tr>
<tr>
<td>4598</td>
<td>6.402</td>
<td>3.497</td>
<td>0.063</td>
</tr>
<tr>
<td>4600</td>
<td>6.634</td>
<td>3.535</td>
<td>0.063</td>
</tr>
<tr>
<td>4601</td>
<td>6.923</td>
<td>3.535</td>
<td>0.063</td>
</tr>
<tr>
<td>4602</td>
<td>7.245</td>
<td>3.548</td>
<td>0.063</td>
</tr>
<tr>
<td>4603</td>
<td>7.636</td>
<td>3.610</td>
<td>0.075</td>
</tr>
</tbody>
</table>

Receiver 526 Shot line 4N
### Receiver 608 Shot line 1N

<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>3242</td>
<td>32.885</td>
<td>7.260</td>
<td>0.100</td>
</tr>
<tr>
<td>3247</td>
<td>29.321</td>
<td>6.903</td>
<td>0.090</td>
</tr>
<tr>
<td>3248</td>
<td>28.575</td>
<td>6.948</td>
<td>0.080</td>
</tr>
<tr>
<td>3249</td>
<td>27.799</td>
<td>7.013</td>
<td>0.080</td>
</tr>
<tr>
<td>3250</td>
<td>27.024</td>
<td>6.902</td>
<td>0.080</td>
</tr>
<tr>
<td>3253</td>
<td>24.981</td>
<td>6.478</td>
<td>0.080</td>
</tr>
<tr>
<td>3254</td>
<td>24.209</td>
<td>6.359</td>
<td>0.080</td>
</tr>
<tr>
<td>3256</td>
<td>22.810</td>
<td>6.031</td>
<td>0.080</td>
</tr>
<tr>
<td>3257</td>
<td>22.079</td>
<td>5.893</td>
<td>0.080</td>
</tr>
<tr>
<td>3258</td>
<td>21.313</td>
<td>5.776</td>
<td>0.070</td>
</tr>
<tr>
<td>3259</td>
<td>20.692</td>
<td>5.611</td>
<td>0.070</td>
</tr>
<tr>
<td>3260</td>
<td>19.930</td>
<td>5.504</td>
<td>0.060</td>
</tr>
<tr>
<td>3261</td>
<td>19.211</td>
<td>5.404</td>
<td>0.060</td>
</tr>
<tr>
<td>3262</td>
<td>18.440</td>
<td>5.298</td>
<td>0.060</td>
</tr>
<tr>
<td>3263</td>
<td>17.634</td>
<td>5.170</td>
<td>0.060</td>
</tr>
<tr>
<td>3264</td>
<td>16.885</td>
<td>4.918</td>
<td>0.070</td>
</tr>
<tr>
<td>3265</td>
<td>16.151</td>
<td>4.717</td>
<td>0.070</td>
</tr>
<tr>
<td>3266</td>
<td>15.380</td>
<td>4.532</td>
<td>0.060</td>
</tr>
<tr>
<td>3267</td>
<td>14.679</td>
<td>4.352</td>
<td>0.060</td>
</tr>
<tr>
<td>3268</td>
<td>13.940</td>
<td>4.152</td>
<td>0.060</td>
</tr>
<tr>
<td>3269</td>
<td>13.195</td>
<td>4.075</td>
<td>0.060</td>
</tr>
<tr>
<td>3270</td>
<td>12.502</td>
<td>4.143</td>
<td>0.080</td>
</tr>
<tr>
<td>3271</td>
<td>11.909</td>
<td>3.973</td>
<td>0.060</td>
</tr>
<tr>
<td>3272</td>
<td>11.202</td>
<td>3.900</td>
<td>0.060</td>
</tr>
<tr>
<td>3273</td>
<td>10.612</td>
<td>3.882</td>
<td>0.060</td>
</tr>
<tr>
<td>3274</td>
<td>10.154</td>
<td>3.852</td>
<td>0.060</td>
</tr>
<tr>
<td>3275</td>
<td>9.701</td>
<td>3.778</td>
<td>0.060</td>
</tr>
<tr>
<td>3276</td>
<td>9.382</td>
<td>3.810</td>
<td>0.060</td>
</tr>
<tr>
<td>3277</td>
<td>9.240</td>
<td>3.853</td>
<td>0.070</td>
</tr>
<tr>
<td>3281</td>
<td>9.141</td>
<td>3.812</td>
<td>0.060</td>
</tr>
<tr>
<td>3282</td>
<td>9.474</td>
<td>3.832</td>
<td>0.060</td>
</tr>
<tr>
<td>3283</td>
<td>9.737</td>
<td>3.830</td>
<td>0.060</td>
</tr>
<tr>
<td>3285</td>
<td>10.596</td>
<td>3.908</td>
<td>0.080</td>
</tr>
<tr>
<td>3286</td>
<td>11.033</td>
<td>4.046</td>
<td>0.070</td>
</tr>
<tr>
<td>3288</td>
<td>12.227</td>
<td>4.198</td>
<td>0.070</td>
</tr>
<tr>
<td>3289</td>
<td>12.971</td>
<td>4.329</td>
<td>0.070</td>
</tr>
<tr>
<td>3290</td>
<td>13.547</td>
<td>4.399</td>
<td>0.070</td>
</tr>
<tr>
<td>3291</td>
<td>14.219</td>
<td>4.476</td>
<td>0.100</td>
</tr>
<tr>
<td>3292</td>
<td>15.069</td>
<td>4.618</td>
<td>0.080</td>
</tr>
<tr>
<td>3293</td>
<td>15.785</td>
<td>4.779</td>
<td>0.090</td>
</tr>
<tr>
<td>3294</td>
<td>16.518</td>
<td>4.900</td>
<td>0.080</td>
</tr>
<tr>
<td>3295</td>
<td>17.353</td>
<td>4.969</td>
<td>0.080</td>
</tr>
</tbody>
</table>

### Receiver 608 Shot line 3N

<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>4526</td>
<td>56.353</td>
<td>10.424</td>
<td>0.080</td>
</tr>
<tr>
<td>4527</td>
<td>55.872</td>
<td>10.339</td>
<td>0.080</td>
</tr>
<tr>
<td>4528</td>
<td>55.472</td>
<td>10.259</td>
<td>0.080</td>
</tr>
<tr>
<td>4530</td>
<td>54.812</td>
<td>10.205</td>
<td>0.080</td>
</tr>
<tr>
<td>4531</td>
<td>54.463</td>
<td>10.138</td>
<td>0.080</td>
</tr>
<tr>
<td>4532</td>
<td>54.131</td>
<td>10.051</td>
<td>0.080</td>
</tr>
<tr>
<td>4533</td>
<td>53.911</td>
<td>9.982</td>
<td>0.100</td>
</tr>
<tr>
<td>4534</td>
<td>53.562</td>
<td>9.942</td>
<td>0.080</td>
</tr>
<tr>
<td>4535</td>
<td>53.285</td>
<td>9.877</td>
<td>0.080</td>
</tr>
<tr>
<td>4536</td>
<td>53.028</td>
<td>9.838</td>
<td>0.100</td>
</tr>
<tr>
<td>4537</td>
<td>52.788</td>
<td>9.773</td>
<td>0.080</td>
</tr>
</tbody>
</table>
## Receiver 608 Shot line 4N

<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>4582</td>
<td>57.687</td>
<td>10.542</td>
<td>0.100</td>
</tr>
<tr>
<td>4583</td>
<td>57.163</td>
<td>10.028</td>
<td>0.100</td>
</tr>
<tr>
<td>4584</td>
<td>56.539</td>
<td>9.919</td>
<td>0.080</td>
</tr>
<tr>
<td>4585</td>
<td>55.874</td>
<td>9.875</td>
<td>0.100</td>
</tr>
<tr>
<td>4586</td>
<td>55.252</td>
<td>9.977</td>
<td>0.100</td>
</tr>
<tr>
<td>4587</td>
<td>54.488</td>
<td>9.866</td>
<td>0.100</td>
</tr>
<tr>
<td>4588</td>
<td>53.384</td>
<td>9.949</td>
<td>0.080</td>
</tr>
<tr>
<td>4589</td>
<td>52.342</td>
<td>9.811</td>
<td>0.100</td>
</tr>
<tr>
<td>4590</td>
<td>51.239</td>
<td>9.679</td>
<td>0.080</td>
</tr>
<tr>
<td>4591</td>
<td>50.099</td>
<td>9.603</td>
<td>0.080</td>
</tr>
<tr>
<td>4592</td>
<td>49.239</td>
<td>9.448</td>
<td>0.080</td>
</tr>
<tr>
<td>4593</td>
<td>48.619</td>
<td>9.441</td>
<td>0.080</td>
</tr>
<tr>
<td>4594</td>
<td>47.445</td>
<td>9.284</td>
<td>0.080</td>
</tr>
<tr>
<td>4595</td>
<td>46.699</td>
<td>9.198</td>
<td>0.080</td>
</tr>
<tr>
<td>4596</td>
<td>46.118</td>
<td>9.028</td>
<td>0.080</td>
</tr>
<tr>
<td>4597</td>
<td>45.436</td>
<td>8.961</td>
<td>0.090</td>
</tr>
<tr>
<td>4598</td>
<td>44.817</td>
<td>8.873</td>
<td>0.070</td>
</tr>
<tr>
<td>4599</td>
<td>44.199</td>
<td>8.817</td>
<td>0.090</td>
</tr>
<tr>
<td>4600</td>
<td>43.517</td>
<td>8.696</td>
<td>0.080</td>
</tr>
<tr>
<td>4601</td>
<td>42.936</td>
<td>8.550</td>
<td>0.070</td>
</tr>
<tr>
<td>4602</td>
<td>42.319</td>
<td>8.470</td>
<td>0.090</td>
</tr>
<tr>
<td>4603</td>
<td>41.636</td>
<td>8.376</td>
<td>0.080</td>
</tr>
<tr>
<td>4604</td>
<td>41.056</td>
<td>8.396</td>
<td>0.080</td>
</tr>
<tr>
<td>4605</td>
<td>40.236</td>
<td>8.270</td>
<td>0.080</td>
</tr>
<tr>
<td>4606</td>
<td>39.517</td>
<td>8.180</td>
<td>0.070</td>
</tr>
<tr>
<td>4607</td>
<td>38.698</td>
<td>8.103</td>
<td>0.070</td>
</tr>
<tr>
<td>4608</td>
<td>37.980</td>
<td>7.990</td>
<td>0.090</td>
</tr>
<tr>
<td>4609</td>
<td>37.161</td>
<td>7.928</td>
<td>0.080</td>
</tr>
<tr>
<td>4610</td>
<td>36.443</td>
<td>7.759</td>
<td>0.080</td>
</tr>
<tr>
<td>4611</td>
<td>35.590</td>
<td>7.615</td>
<td>0.080</td>
</tr>
<tr>
<td>4612</td>
<td>34.055</td>
<td>7.362</td>
<td>0.070</td>
</tr>
<tr>
<td>4613</td>
<td>33.204</td>
<td>7.185</td>
<td>0.070</td>
</tr>
<tr>
<td>4614</td>
<td>32.489</td>
<td>7.065</td>
<td>0.060</td>
</tr>
<tr>
<td>4615</td>
<td>31.742</td>
<td>7.003</td>
<td>0.060</td>
</tr>
<tr>
<td>4616</td>
<td>30.926</td>
<td>6.883</td>
<td>0.060</td>
</tr>
<tr>
<td>4617</td>
<td>30.183</td>
<td>6.762</td>
<td>0.070</td>
</tr>
<tr>
<td>4618</td>
<td>29.338</td>
<td>6.769</td>
<td>0.060</td>
</tr>
<tr>
<td>4619</td>
<td>27.049</td>
<td>6.425</td>
<td>0.060</td>
</tr>
<tr>
<td>4620</td>
<td>26.342</td>
<td>6.272</td>
<td>0.060</td>
</tr>
<tr>
<td>4621</td>
<td>25.507</td>
<td>6.156</td>
<td>0.080</td>
</tr>
<tr>
<td>4622</td>
<td>23.972</td>
<td>5.949</td>
<td>0.060</td>
</tr>
<tr>
<td>4623</td>
<td>23.143</td>
<td>5.890</td>
<td>0.070</td>
</tr>
</tbody>
</table>

## Receiver 611 Shot line 1N

<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>3260</td>
<td>52.169</td>
<td>9.784</td>
<td>0.100</td>
</tr>
<tr>
<td>3261</td>
<td>51.391</td>
<td>9.675</td>
<td>0.100</td>
</tr>
<tr>
<td>3262</td>
<td>49.609</td>
<td>9.463</td>
<td>0.100</td>
</tr>
<tr>
<td>3263</td>
<td>46.044</td>
<td>8.618</td>
<td>0.090</td>
</tr>
<tr>
<td>3264</td>
<td>45.155</td>
<td>8.478</td>
<td>0.100</td>
</tr>
<tr>
<td>3265</td>
<td>43.171</td>
<td>8.090</td>
<td>0.090</td>
</tr>
<tr>
<td>3266</td>
<td>40.481</td>
<td>8.063</td>
<td>0.090</td>
</tr>
<tr>
<td>3267</td>
<td>38.588</td>
<td>7.810</td>
<td>0.090</td>
</tr>
<tr>
<td>3268</td>
<td>34.915</td>
<td>7.514</td>
<td>0.080</td>
</tr>
<tr>
<td>3269</td>
<td>34.025</td>
<td>7.476</td>
<td>0.080</td>
</tr>
<tr>
<td>3270</td>
<td>33.247</td>
<td>7.279</td>
<td>0.080</td>
</tr>
<tr>
<td>3271</td>
<td>32.138</td>
<td>7.099</td>
<td>0.080</td>
</tr>
<tr>
<td>3272</td>
<td>31.361</td>
<td>6.958</td>
<td>0.080</td>
</tr>
<tr>
<td>3273</td>
<td>30.251</td>
<td>6.807</td>
<td>0.080</td>
</tr>
<tr>
<td>3274</td>
<td>28.590</td>
<td>6.526</td>
<td>0.070</td>
</tr>
<tr>
<td>3275</td>
<td>27.486</td>
<td>6.414</td>
<td>0.080</td>
</tr>
<tr>
<td>3276</td>
<td>26.823</td>
<td>6.329</td>
<td>0.070</td>
</tr>
<tr>
<td>3277</td>
<td>25.718</td>
<td>6.161</td>
<td>0.100</td>
</tr>
<tr>
<td>3278</td>
<td>24.946</td>
<td>6.002</td>
<td>0.080</td>
</tr>
<tr>
<td>3279</td>
<td>24.065</td>
<td>5.820</td>
<td>0.070</td>
</tr>
<tr>
<td>3280</td>
<td>23.081</td>
<td>5.683</td>
<td>0.060</td>
</tr>
<tr>
<td>3281</td>
<td>22.203</td>
<td>5.525</td>
<td>0.070</td>
</tr>
<tr>
<td>3282</td>
<td>21.326</td>
<td>5.366</td>
<td>0.070</td>
</tr>
<tr>
<td>3283</td>
<td>20.340</td>
<td>5.226</td>
<td>0.070</td>
</tr>
<tr>
<td>3284</td>
<td>19.356</td>
<td>5.094</td>
<td>0.060</td>
</tr>
<tr>
<td>3285</td>
<td>18.388</td>
<td>4.966</td>
<td>0.060</td>
</tr>
<tr>
<td>3286</td>
<td>17.411</td>
<td>4.845</td>
<td>0.060</td>
</tr>
<tr>
<td>Shot</td>
<td>Range</td>
<td>Time</td>
<td>Err</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>3299</td>
<td>16.328</td>
<td>4.718</td>
<td>0.050</td>
</tr>
<tr>
<td>3300</td>
<td>15.122</td>
<td>4.563</td>
<td>0.050</td>
</tr>
<tr>
<td>3301</td>
<td>14.375</td>
<td>4.463</td>
<td>0.050</td>
</tr>
<tr>
<td>3302</td>
<td>13.284</td>
<td>4.295</td>
<td>0.050</td>
</tr>
<tr>
<td>3303</td>
<td>12.113</td>
<td>4.133</td>
<td>0.050</td>
</tr>
<tr>
<td>3304</td>
<td>11.142</td>
<td>4.032</td>
<td>0.050</td>
</tr>
<tr>
<td>3305</td>
<td>10.313</td>
<td>3.901</td>
<td>0.050</td>
</tr>
<tr>
<td>3306</td>
<td>9.357</td>
<td>3.811</td>
<td>0.090</td>
</tr>
<tr>
<td>3307</td>
<td>8.345</td>
<td>3.491</td>
<td>0.050</td>
</tr>
<tr>
<td>3308</td>
<td>7.465</td>
<td>3.311</td>
<td>0.040</td>
</tr>
<tr>
<td>3309</td>
<td>6.472</td>
<td>3.125</td>
<td>0.040</td>
</tr>
<tr>
<td>3310</td>
<td>5.494</td>
<td>2.958</td>
<td>0.050</td>
</tr>
<tr>
<td>3311</td>
<td>4.716</td>
<td>2.836</td>
<td>0.050</td>
</tr>
<tr>
<td>3312</td>
<td>3.733</td>
<td>2.655</td>
<td>0.040</td>
</tr>
<tr>
<td>3313</td>
<td>3.658</td>
<td>2.594</td>
<td>0.040</td>
</tr>
<tr>
<td>3314</td>
<td>3.895</td>
<td>2.652</td>
<td>0.040</td>
</tr>
<tr>
<td>3315</td>
<td>4.163</td>
<td>2.744</td>
<td>0.040</td>
</tr>
<tr>
<td>3316</td>
<td>4.399</td>
<td>2.853</td>
<td>0.050</td>
</tr>
<tr>
<td>3317</td>
<td>4.871</td>
<td>2.958</td>
<td>0.050</td>
</tr>
<tr>
<td>3318</td>
<td>5.195</td>
<td>3.058</td>
<td>0.050</td>
</tr>
<tr>
<td>3319</td>
<td>5.921</td>
<td>3.171</td>
<td>0.050</td>
</tr>
<tr>
<td>3320</td>
<td>6.428</td>
<td>3.260</td>
<td>0.050</td>
</tr>
<tr>
<td>3321</td>
<td>7.064</td>
<td>3.355</td>
<td>0.060</td>
</tr>
<tr>
<td>3322</td>
<td>7.727</td>
<td>3.469</td>
<td>0.060</td>
</tr>
<tr>
<td>3323</td>
<td>8.901</td>
<td>3.646</td>
<td>0.050</td>
</tr>
<tr>
<td>3324</td>
<td>9.613</td>
<td>3.778</td>
<td>0.060</td>
</tr>
<tr>
<td>3325</td>
<td>10.335</td>
<td>3.861</td>
<td>0.060</td>
</tr>
<tr>
<td>3326</td>
<td>11.066</td>
<td>3.964</td>
<td>0.060</td>
</tr>
<tr>
<td>3327</td>
<td>11.803</td>
<td>4.083</td>
<td>0.060</td>
</tr>
<tr>
<td>3328</td>
<td>12.557</td>
<td>4.175</td>
<td>0.060</td>
</tr>
<tr>
<td>3329</td>
<td>13.288</td>
<td>4.282</td>
<td>0.060</td>
</tr>
<tr>
<td>3330</td>
<td>14.036</td>
<td>4.393</td>
<td>0.060</td>
</tr>
<tr>
<td>3331</td>
<td>14.803</td>
<td>4.494</td>
<td>0.060</td>
</tr>
<tr>
<td>3332</td>
<td>15.547</td>
<td>4.596</td>
<td>0.060</td>
</tr>
</tbody>
</table>

Receiver 612  Shot line 2N

<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>4626</td>
<td>43.449</td>
<td>8.667</td>
<td>0.120</td>
</tr>
<tr>
<td>4627</td>
<td>42.586</td>
<td>8.656</td>
<td>0.100</td>
</tr>
<tr>
<td>4628</td>
<td>42.074</td>
<td>8.655</td>
<td>0.120</td>
</tr>
<tr>
<td>4629</td>
<td>41.882</td>
<td>8.328</td>
<td>0.100</td>
</tr>
<tr>
<td>4630</td>
<td>41.554</td>
<td>8.276</td>
<td>0.090</td>
</tr>
<tr>
<td>4631</td>
<td>41.328</td>
<td>8.283</td>
<td>0.120</td>
</tr>
<tr>
<td>4632</td>
<td>41.104</td>
<td>8.290</td>
<td>0.120</td>
</tr>
<tr>
<td>4633</td>
<td>40.919</td>
<td>8.354</td>
<td>0.100</td>
</tr>
<tr>
<td>4634</td>
<td>40.665</td>
<td>8.327</td>
<td>0.100</td>
</tr>
<tr>
<td>4635</td>
<td>40.432</td>
<td>8.382</td>
<td>0.100</td>
</tr>
<tr>
<td>4636</td>
<td>40.310</td>
<td>8.490</td>
<td>0.120</td>
</tr>
<tr>
<td>4637</td>
<td>40.083</td>
<td>8.346</td>
<td>0.100</td>
</tr>
<tr>
<td>4638</td>
<td>40.020</td>
<td>8.351</td>
<td>0.120</td>
</tr>
<tr>
<td>4639</td>
<td>39.861</td>
<td>8.339</td>
<td>0.120</td>
</tr>
<tr>
<td>4640</td>
<td>39.775</td>
<td>8.298</td>
<td>0.100</td>
</tr>
<tr>
<td>4641</td>
<td>39.656</td>
<td>8.415</td>
<td>0.100</td>
</tr>
<tr>
<td>4642</td>
<td>39.335</td>
<td>8.309</td>
<td>0.100</td>
</tr>
<tr>
<td>4643</td>
<td>39.179</td>
<td>8.413</td>
<td>0.100</td>
</tr>
<tr>
<td>4644</td>
<td>39.031</td>
<td>8.479</td>
<td>0.080</td>
</tr>
<tr>
<td>4645</td>
<td>38.831</td>
<td>8.453</td>
<td>0.100</td>
</tr>
<tr>
<td>4646</td>
<td>38.625</td>
<td>8.519</td>
<td>0.100</td>
</tr>
<tr>
<td>4647</td>
<td>38.460</td>
<td>8.562</td>
<td>0.100</td>
</tr>
</tbody>
</table>

Receiver 611  Shot line 4N

<table>
<thead>
<tr>
<th>Shot</th>
<th>Range</th>
<th>Time</th>
<th>Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>3360</td>
<td>15.368</td>
<td>4.985</td>
<td>0.050</td>
</tr>
<tr>
<td>3361</td>
<td>16.051</td>
<td>5.023</td>
<td>0.060</td>
</tr>
<tr>
<td>3362</td>
<td>18.168</td>
<td>5.476</td>
<td>0.060</td>
</tr>
<tr>
<td>3363</td>
<td>21.807</td>
<td>6.291</td>
<td>0.060</td>
</tr>
<tr>
<td>3364</td>
<td>22.797</td>
<td>6.395</td>
<td>0.060</td>
</tr>
<tr>
<td>3365</td>
<td>23.573</td>
<td>6.447</td>
<td>0.070</td>
</tr>
<tr>
<td>3366</td>
<td>24.355</td>
<td>6.677</td>
<td>0.080</td>
</tr>
<tr>
<td>3367</td>
<td>25.141</td>
<td>6.756</td>
<td>0.080</td>
</tr>
<tr>
<td>3368</td>
<td>26.729</td>
<td>6.946</td>
<td>0.070</td>
</tr>
<tr>
<td>3369</td>
<td>28.206</td>
<td>7.185</td>
<td>0.070</td>
</tr>
<tr>
<td>3370</td>
<td>29.118</td>
<td>7.307</td>
<td>0.070</td>
</tr>
<tr>
<td>3371</td>
<td>29.902</td>
<td>7.480</td>
<td>0.080</td>
</tr>
<tr>
<td>3372</td>
<td>30.576</td>
<td>7.777</td>
<td>0.100</td>
</tr>
<tr>
<td>3373</td>
<td>31.582</td>
<td>7.672</td>
<td>0.080</td>
</tr>
<tr>
<td>3374</td>
<td>32.367</td>
<td>7.751</td>
<td>0.080</td>
</tr>
<tr>
<td>3375</td>
<td>33.041</td>
<td>7.817</td>
<td>0.080</td>
</tr>
<tr>
<td>3376</td>
<td>34.047</td>
<td>7.941</td>
<td>0.080</td>
</tr>
<tr>
<td>3377</td>
<td>34.953</td>
<td>7.990</td>
<td>0.080</td>
</tr>
<tr>
<td>3378</td>
<td>37.418</td>
<td>8.323</td>
<td>0.080</td>
</tr>
<tr>
<td>3379</td>
<td>38.314</td>
<td>8.396</td>
<td>0.080</td>
</tr>
</tbody>
</table>
3390 38.989 8.423 0.080 4527 27.833 7.447 0.070
3391 39.774 8.512 0.080 4529 28.663 7.320 0.070
3392 40.669 8.615 0.100 4530 29.071 7.422 0.080
3393 41.455 8.803 0.100 4531 29.510 7.374 0.080
3396 44.131 8.987 0.100 4532 29.977 7.697 0.080
3397 44.917 9.057 0.100 4533 30.574 7.517 0.080
3403 49.828 9.872 0.100 4534 31.313 7.556 0.080
3404 50.502 9.939 0.100 4535 31.677 7.550 0.090
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

248
REFERENCES


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


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


