MICROCOMPUTER-BASED VERSION OF SIPT--A PROGRAM FOR THE INTERPRETATION OF SEISMIC-REFRACTION DATA (TEXT)

By F. P. Haeni, Deborah G. Grantham, and Karl Ellefsen

Open-File Report 87-103-A

Hartford, Connecticut

1987

UNITED STATES DEPARTMENT OF THE INTERIOR

DONALD PAUL HODEL, Secretary

GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information,
write to:
Chief, Connecticut Office
U.S. Geological Survey, WRD
450 Main Street, Room 525
Hartford, CT 06103

Copies of this report can be
purchased from:
Books & Open-File Reports Section
U.S. Geological Survey
Box 25425, Federal Center
Denver, Co 80225
Telephone: (303) 236-7476

CONTENTS

ILLUSTRATIONS

TABLES

CONVERSION FACTORS AND ABBREVIATIONS

This report uses metric (International System) units as the primary system of measurements. The units are commonly abbreviated using the notations shown in parentheses. Metric units can be converted to inch-pound units by multiplying by the factors given in the following list.

Multiply metric unit	by	to obtain inch-pound unit
	Length	
meter (m)	3.281	foot (ft)
kilometer (km)	0.622	mile (mi)
	Area	
square kilometer (km^2)	0.386	square mile (mi^2)
	Velocity	
meter per second (m/s)	3.281	foot per second (ft/s)
kilometer per second (km/s)	0.622	mile per second (mi/s)

MICROCOMPUTER-BASED VERSION OF SIPT--A PROGRAM FOR THE INTERPRETATION OF SEISMIC-REFRACTION DATA

by F. P. Haeni, Deborah G. Grantham, and Karl Ellefsen

ABSTRACT

This report documents the use of a microcomputer-based FORTRAN program, SIPT1.FOR, for the interpretation of seismic-refraction data and outlines an interpretation procedure using SIPT1.FOR.

SIPT1.FOR is an adaptation of SIPT22.FOR and SIPT23.FOR, programs developed for use on mainframe computer systems. The program has been adapted for interactive use on microcomputers that use a DOS 3.0 operating system.

SIPT1.FOR can be used to interpret seismic-refraction data for complex field situations and geologic settings. The program is a two-dimensional modeling process in which the delay-time technique is used to obtain a first approximation of model layers and then an iterative ray tracing technique is used to refine the model. Error detection, feedback, and automatic adjustment are written into the program. The program can handle multilayer dipping bed problems for multiple geophone spreads and complex field situations such as offset shotpoints, nonlinear geophone spacing, and variable land-surface topography.

The documentation includes instructions for the use of SIPT1.FOR, a discussion of the interpretation process, and an example problem.

INTRODUCTION

Seismic-refraction data can be used to infer the subsurface geology and hydrology of a study site. Many papers have described the use of the seismic-refraction method in hydrologic studies, including: Bonini and Hickok, 1958; Warrick and Winslow, 1960; Gill and others, 1965; Eaton and Watkins, 1967; Birch, 1976; Haeni, 1978; Morrissey, 1983; Tolman and others, 1983; and Haeni, 1986a. Depending upon the scope of the hydrologic study and the complexity of the hydrogeology at a site, it may be necessary to use computer-assisted interpretation techniques to obtain more detailed and accurate hydrogeologic information.

Several computer programs have been developed to assist in the analysis of complex seismic-refraction investigations. These programs are able to solve multilayer problems for multiple geophone spreads in field areas having topographic relief and nonlinearly spaced shotpoints and geophones.

SIPT is one of these programs and has been used successfully by the U.S. Geological Survey under varying geologic and hydrologic field conditions. SIPT is a computer-modeling procedure based on the delay-time technique of Barthelmes (1946), modified by Pakiser and Black (1957), and further developed by Scott and others (1972) and Scott (1977a, 1977b).

The original FORTRAN IV[1/] program code and its documentation for batch processing using a Burroughs computer system are given in Scott and others (1972). The documentation for a revised batch-processing version is described by Scott (1977a), and documentation for an interactive version of the same program is provided by Scott (1977b). Other versions of the program have run on Multics, Prime, IBM, and VAX computer systems. The SIPT1.FOR is a microcomputer-based, interactive version of SIPT22 and SIPT23, versions of SIPT running on the U.S. Geological Survey Distributed Information Systems computer system. SIPT1.FOR is running on the IBM-PC and IBM-AT and IBM-compatible computers using a DOS (Disk Operating System) 3.0. SIPT1.FOR has been compiled on both the Microsoft FORTRAN compiler and the IBM Professional FORTRAN compiler. The current release, SIPT1.FOR, is compiled on the IBM Professional compiler.

The documentation of SIPT by Scott is complete and the discussion of the seismic-refraction theory and the program algorithm in this documentation will be brief.

This documentation and the source code for the SIPT1.FOR program are available on paper (U.S. Geological Survey Open-File Report 87-103-A) from the Books and Open-File Reports Section, U.S. Geological Survey, Box 25425, Federal Center, Denver, CO, 80225. The compiled, executable program and the source code are available on floppy disk (U.S. Geological Survey Open-File Report 87-103-B) from the Books and Open-File Reports Section.

This report documents the use of the microcomputer-based FORTRAN program, SIPT1.FOR, for the interpretation of seismic-refraction data and outlines an interpretation procedure using SIPT1.FOR.

[1/] Use of brand, firm, or trade names in this report is for identification purposes only and does not constitute endorsement by the U.S. Geological Survey.

SEISMIC-REFRACTION THEORY

Numerous textbooks and journal articles present the details of seismic-refraction theory (Heiland, 1940; Slotnick, 1959; Grant and West, 1965; Griffiths and King, 1965; Musgrave, 1967; Dobrin, 1976; Telford and others, 1976; Parasnis, 1979; and Mooney, 1981). The following discussion will review the basic principles and limitations of the seismic-refraction method.

It must be emphasized that the absence of an extensive section on the theory of seismic refraction does not minimize the importance of the topic. Hydrologists must possess a good understanding of the technique and its limitations prior to using either the technique or the SIPT1.FOR program.

The foundation of seismic-refraction theory is Snell's Law, which governs the refraction of sound or light waves across the boundary between layers of media with different acoustic or light transmitting properties. As sound propagates through one layer and impinges on another layer with faster seismic velocity characteristics, part of the energy is refracted (or bent) and part is reflected back into the first layer (see raypath 1 in fig. 1). As the angle of incidence approaches the critical angle, an angle where the refracted ray grazes the surface of the contact between two media and is equal to arcsin v_1/v_2, the compressional energy is transmitted along the surface of the second layer (see raypath 2 in fig.1). As this energy propagates along the surface of layer 2, it generates new sound waves in the upper medium according to Huygen's principle, which states that every point on an advancing wave front can be regarded as the source of a new sound wave; these new sound waves propagate back to the surface through layer 1. When these refracted waves arrive at the land surface, the geophones along the path of the wavefront are activated and the signal is recorded on the seismograph.

For a spread of geophones and a shotpoint, the arrival times can be plotted as a function of the shot-to-detector distances; this results in a time-distance plot or time-distance curve, shown in figure 2. The crossover distance (x_c) is the source-to-receiver distance at which refracted sound waves from a deep layer overtake direct sound waves from a shallow layer. For any distance less than the crossover distance, the sound travels directly from the source to the detectors. This compressional wave travels a known distance in a known time and the velocity of the first layer can be directly calculated by $v_1 = x/t$ where x is the distance traveled by the sound wave through layer 1 in time t and v_1 is the velocity of sound in layer 1. The data are plotted on the time-distance plot, time on the ordinate axis and distance on the abscissa, and v_1 is equal to the inverse slope of the first line segment.

Beyond the first crossover distance the compressional sound wave that has traveled through layer 1, along the interface of the high-speed layer (layer 2) and back up to the surface through layer 1, arrives before the compressional wave that has traveled only in layer

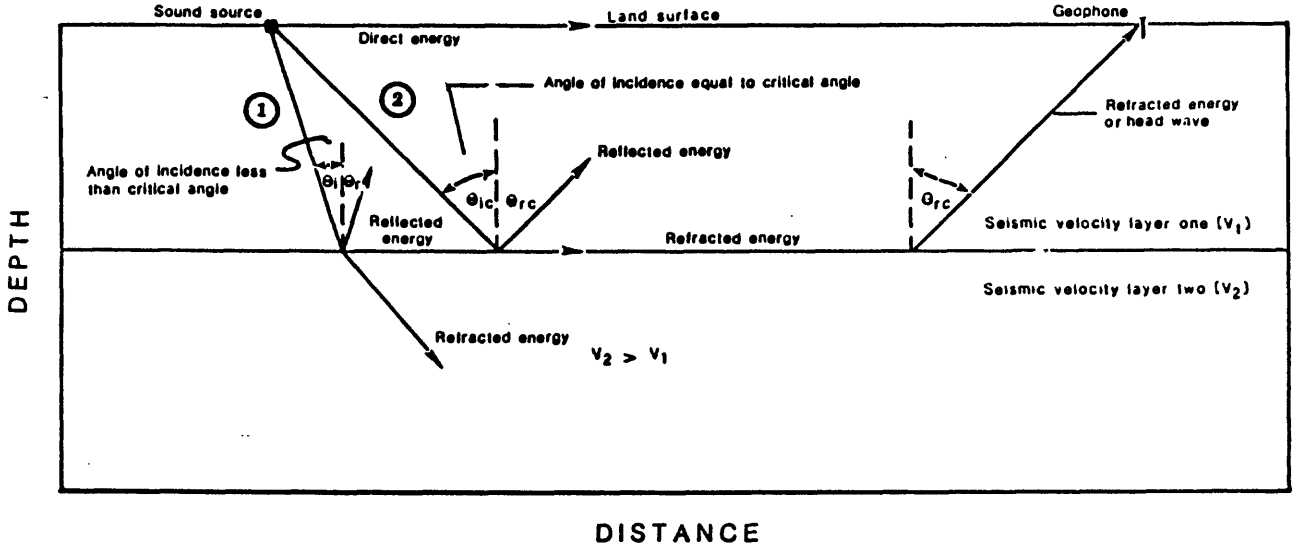


Figure 1.--Ray path diagram showing the principles of seismic refraction techniques (after Haeni, 1986b).

1 (the slow velocity layer). All first-compressional wave arrivals at geophones more distant than the crossover distance will be refracted or head waves from layer 2. The slope of the line segment formed by plotting these first-compressional wave arrivals on the time-distance plot will be equal to the apparent velocity of layer 2 (fig. 2). This line segment intersects the time axis at some point greater than zero, called the intercept time (t_i). The intercept time and the crossover distance are directly dependent on the velocity of sound in the two materials and the thickness of the first layer and, therefore, can be used to determine the thickness of the first layer (z).

The seismic velocities and thicknesses of additional deeper layers can be determined in the same way.

Interpretation Formulas

Intercept times and crossover distance-depth formulas have been derived in the literature (Grant and West, 1965; Zohdy and others, 1974; Dobrin, 1976; Telford and others, 1976; Parasnis, 1979; and Mooney, 1981) and will not be presented in this documentation.

For solution of the intercept times and crossover distance formulas, the total traveltime of the sound waves from the shot point to the geophones is measured, the velocity in each layer is calculated from the time-distance plot, and the ray-path geometry can be calculated. The only unknown is the depth to the high-speed refractor. These calculations can be made manually, with a hand-held calculator (Ballantyne and others, 1981), or by means of several computer programs.

Assumptions

The most commonly used and published formulas are based on the following assumptions (Haeni, 1986b):

1) The boundaries between layers are planes that are either horizontal or dipping at a constant angle.

2) There is no land-surface relief.

3) Each layer is homogeneous and isotropic, and;

4) The seismic velocity of the layers increases with depth.

These assumptions must be considered by the hydrologist before planning a field investigation or interpreting data.

Some versions of the formulas are capable of handling more complex situations than indicated in the above assumptions and are published by Dobrin (1976, p. 301-313).

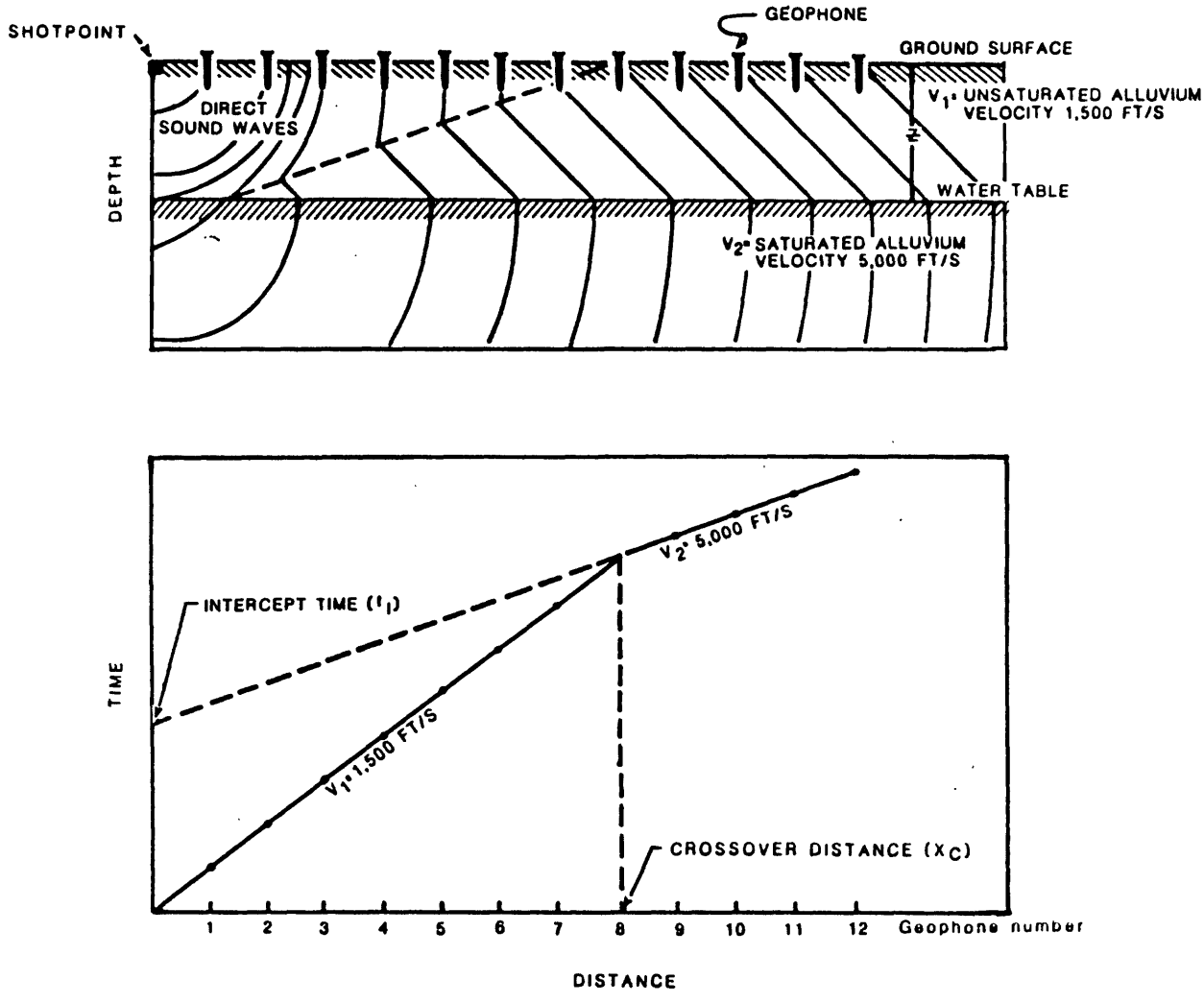


Figure 2.--Seismic wave front diagram and corresponding time-distance plot (after Haeni, 1986b).

Advanced Interpretation Methods

More sophisticated theoretical solutions of seismic-refraction data such as delay-time procedures (Dobrin, 1976, p. 313-314), the GRM (generalized reciprocal method) (Palmer, 1980), and a method described by Ackermann and others (1983) have been developed and are capable of handling complex geologic settings. These procedures have been implemented on various computer systems and are widely used for the interpretation of seismic-refraction data.

Field-Related Corrections

In addition to the theoretical solutions of seismic-refraction problems, corrections for field-related problems also have been developed. The two main field problems are variations in land-surface elevation and thickness of the weathered layer. Both elevation and weathering corrections are used to adjust field-derived traveltimes to some selected datum plane, so that straight-line segments on the time-distance plot can be associated with subsurface refractors. These corrections can be applied manually (Dobrin, 1976, p. 335) or by computer (Scott and others, 1972).

Limitations of the Seismic-Refraction Method

The seismic-refraction technique does not work well in the following situations (Domzalski, 1956; Burke, 1967; Wallace, 1970):

1) The presence of thin intermediate seismic-velocity refractors;

2) Insufficient seismic-velocity contrasts between hydrologic units;

3) The presence of low seismic-velocity units underlying high seismic-velocity units, and;

4) Long geophone spreads over deep refractors.

In the first situation--a thin intermediate seismic-velocity layer in the geologic section--cannot be detected by the refraction method (Soske, 1959). This problem is critical in water-resources investigations because the intermediate layer may be the zone of interest, the saturated aquifer material, in the geologic section. These intermediate layers cannot be defined by any alternative location of the geophones or shallow shot points. Deep shot holes may overcome this problem (Soske, 1959) but are usually impractical under normal field conditions. In the absence of drill-hole data, an unexpected velocity change in the time-distance plot should warn the hydrologist that a thin intermediate seismic-velocity layer may be present and that a qualified interpretation is in order. If the

presence of such a layer is suspected, calculations can be made to determine its minimum and maximum thickness. Methods for performing these calculations are presented in Soske (1959), Hawkins and Maggs (1961), Green (1962), Redpath (1973), and Mooney (1981).

The second situation--insufficient seismic-velocity contrasts between layers--may cause significant hydrologic units to remain undetected. For example, weathered rock surfaces may have approximately the same seismic velocity as saturated unconsolidated materials (Burwell, 1940). These layers cannot be differentiated using the seismic-refraction technique.

The third situation--a seismic-velocity inversion--will cause the sound wave to be refracted toward the lower velocity layer, according to Snell's Law, so the low velocity unit will not be detected and the calculated depth to a deep refractor will be in error. If a velocity inversion is known to exist, and the depth of the slow layer and seismic velocity are known from drill-hole or geologic data, the true depth to the deeper refractor can be estimated (Morgan, 1967; Mooney, 1981).

The fourth problem is a result of the assumption that the seismic velocity in each layer is constant throughout the entire spread. This limitation is not severe for short spread lengths, but may impose severe restrictions on the interpretation for long spreads over deep refractors. The U.S. Geological Survey (Ackerman, 1983) has developed a computer interpretation program that overcomes this shortcoming. This procedure is more difficult to use than the one described here, but it is a better interpretation scheme when large spreads and very deep refractors are being studied. The GRM (Palmer, 1980) also avoids this problem. The details of these interpretation procedures will not be covered here because the procedure is well documented.

These problems are mentioned not to discourage the use of seismic-refraction techniques, but rather to make hydrologists aware of potential pitfalls. These situations, recognized early in the study, can be accounted for in the planning, data acquisition, and interpretation phases of the study.

PROGRAM DESCRIPTION

The SIPT1.FOR computer program is a two-dimensional modeling process in which the delay-time technique is used to obtain a first approximation of model layers and then an iterative ray-tracing technique is used to refine the model. Error detection, feedback, and automatic adjustment are written into the program. The following sections describe the concepts and procedures used in SIPT1.FOR.

Program Assumptions And Limitations

In addition to the theoretical limitations and assumptions presented under seismic-refraction theory, the following requirements, assumptions, and limitations are associated with the SIPT1.FOR program (Scott and others, 1972):

1) The number of layers represented by the data must be predetermined by the interpreter and provided as input data to the program.

2) Each refraction event as measured by the first break on the seismograph must be assigned a number that represents the layer carrying the critically refracted ray along its surface.

3) Each layer under each spread is assumed to have a constant horizontal and vertical velocity. Horizontal and vertical seismic velocities may differ from one another.

4) Each layer extends from one side of the model to the other.

5) The maximum number of layers is five.

6) The maximum number of spreads is five. Each spread has up to 48 geophones and a maximum of seven shotpoints. These limits can be changed in the program if necessary.

7) Refracted rays are assumed to represent minimum traveltime paths of compressional seismic waves, and;

8) The final interpreted model layers are defined beneath geophones that receive refracted energy from the surface of that layer and are interpolated or extrapolated to other positions.

General Description of Program

Main Program

SIPT1.FOR is divided into three main parts, described briefly in this section, that call various subroutines to carry out computations, data display, and modifications to the interpretations.

Part 1 of the main program inputs the data, estimates the velocity of layer 1, and makes the time corrections to a sloping datum plane, if necessary.

Part 2 estimates the velocity of layer 2, delineates the interface between layers 1 and 2, removes the effect of layer 1 from observed refraction traveltimes, optionally draws the time-distance graph with layer 1 removed, and estimates the velocities of the layers beneath layer 2. The velocity of layer 2 is estimated by two methods. The first fits a line through the traveltimes of the rays refracted along the top of layer 2 by least-squares regression. The second method, the Hobson-Overton method, uses a variation of delay-time procedures.

Part 3 delineates the layers beneath layer 2, one at a time from the shallowest to the deepest. The delay-time method gives a first approximation and then an iterative ray-tracing technique is used to refine the model. The resulting geologic model and depth tables are then printed.

Delay-Time and Ray-Tracing Procedures

Delay time, or intercept time, is the additional time required for a wave to travel a segment of a ray trajectory over the time that would be required for the same wave to travel the horizontal component of that segment at the highest velocity along the trajectory (Pakiser and Black, 1957). Delay-time procedures use the delay or intercept times to determine the depths at each end of the segment of the ray trajectory in the case that the segment is not horizontal or uniformly dipping. SIPT1.FOR uses a delay-time procedure developed by Barthelmes (1946).

The Hobson-Overton method, developed by Hobson and Overton of the Geological Survey of Canada (Scott, 1973), uses a velocity such that the variation of total delay-time differences for a group of geophones is minimized. This procedure requires that at least two geophones receive refracted rays traveling in opposite directions from two different shot points. The program will indicate, on the terminal screen, a lack of data for the use of the Hobson-Overton method and will use the least-squares method.

Ray tracing is next used to determine the true position of points of entrance and emergence of rays along each refracting horizon, taking into account the dip of the refracting horizon and the dips and thicknesses of all layers above. SIPT1.FOR adjusts the model by tracing raypaths upwards from depths computed using the delay-time

method, calculating traveltimes based on the lengths of these ray-paths and then comparing the calculated traveltimes with the observed traveltimes. SIPT1.FOR can perform up to three iterations of the ray-tracing procedure, in which entrance and emergence points are adjusted to minimize the discrepancies between the calculated traveltimes and the observed traveltimes.

Exit Points

SIPT1.FOR has several exit points so that the user can obtain specific information without running through the entire program. The exit number is specified in the input data but may be changed when the program is run (Scott and others, 1972).

Exit 0 draws a time-distance graph, with no corrections applied to the data.

Exit 1 applies terrain corrections to the data before drawing the time-distance graph.

Exit 2 draws the time-distance graph with the near surface layer stripped away.

Exits 3 through 6 are points in part 3 of the main program that provide varying degrees of data processing in delineating the layers beneath layer 2.

Exit 3 terminates the program after the delay-time procedure has been used to delineate layers deeper than layer 2.

Exit 4 terminates the program after the delay-time procedure and one ray-tracing iteration has been completed. The ray-tracing procedure adjusts layer 2 so that any sharp irregularities in layer 2 are not carried through to the delineation of deeper layers.

Exit 5 terminates the program after the delay-time procedure and two ray-tracing iterations have been completed.

Exit 6 allows the program to run for a full 3 ray-tracing iterations.

The plus or minus sign preceding an exit number indicates full or brief output, depending on the requirements of the user.

USER PROCEDURES

An example application of SIPT1.FOR is described in this section to demonstrate user procedures. The example is taken from a field investigation of the Minister Brook area in Simsbury, Conn., conducted by the U.S. Geological Survey.

An IBM or compatible personal computer with either a floppy disk drive or hard disk, DOS 3.0, and a minimum of 320K bytes of memory is required to run this program.

The interpretation process starts with the following steps:

1) Pick arrival times from seismograph records, assign preliminary layer numbers to each refraction event, and record the times on data sheet (fig. 3).

2) Plot the position of all shotpoints and geophones using an arbitrary scale X-Y coordinate system (fig. 4).

3) Plot the elevation of all shotpoints and geophones (fig. 4).

4) Choose appropriate scales for the elevation plot, the horizontal plot, and the time-distance plot. The scales are ft/col (feet per column), ft/row (feet per row), and ms/col (milliseconds per column) and the default values are 5.0 ft/col, 8.333 ft/row, and 1.0 ms/col respectively. Default values can only be changed by entering new values in the appropriate positions of the input file.

5) Enter information on computer data input form (fig. 5). For detailed data entry instruction, see Scott and others, 1972; Scott, 1973; and Scott, 1977a;b.

6) Enter the information into a computer file created using an editor, such as WordStar. SIPT1.FOR accepts either free or fixed field data. Tables 1 and 2 show examples of data sets of free and fixed field data, respectively. All data must be in lower case letters and the geophone numbers must be enclosed by quotation marks. Data is stored in files that are 132 characters in width.

SEISMIC REFRACTION FIRST ARRIVAL TIME RECORD SHEET

Site name _____

Spread # _____

Shot number _____ Shot direction _____

Geophone #	First arrival time (in ms)	Seismograph delay time (in ms)	Total travel time (in ms)	Preliminary layer assignment	Notes
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

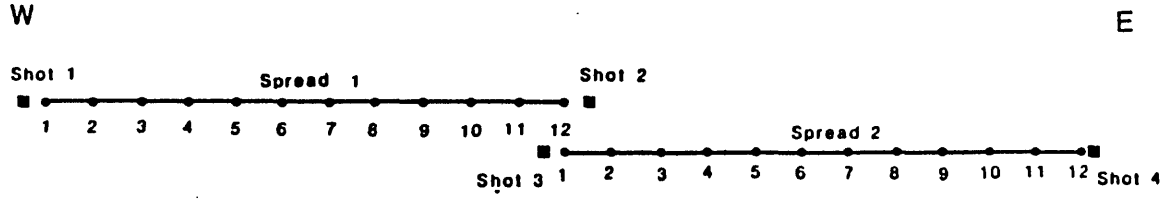
Shot number _____ Shot direction _____

Geophone #	First arrival time (in ms)	Seismograph delay time (in ms)	Total travel time (in ms)	Preliminary layer assignment	Notes
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

Figure 3.--Seismic refraction first arrival record sheet (after Haeni, 1986b).

FARMINGTON

(A) Spread location diagram



(B) Topographic profile

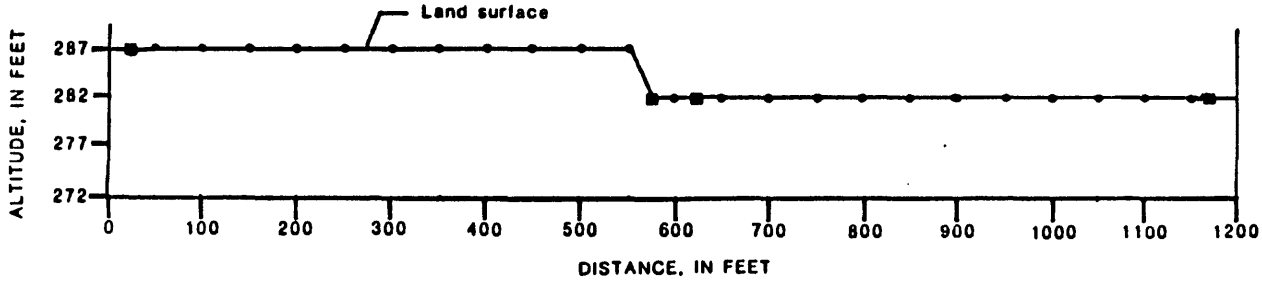


Figure 4.--Example of shotpoint and geophone locations and an elevation plot (after Haeni, 1986b).

Note: each number is followed by a comma when entering data into the computer.

One set per problem (1 to 5 spreads)

PROBLEM IDENTIFICATION LINE (UP TO 78 COLUMNS OF TEXT OR NUMBERS)

PROBLEM CONTROL LINE

Number of spreads	Program exit point	Number of Layers	Number of velocity cards	Elevation plot scale	Horizontal distance plot scale	Time plot scale	
							0,0,0,0,0,0,0,0,0,0,0

VELOCITY LINE (ONE PER LAYER)

Layer Number	Vertical velocity spread #1	Horizontal velocity spread #1	
		0.	0,0

(Number of velocity pairs depends on number of spreads, i.e., one spread would have 0,0; two spreads would have 0,0,0,0.)

One set per each spread

SPREAD CONTROL LINE

Spread number	Number of shot Points	Number of geophones	
			0,0

SHOT POINT LINES

Shot point number	Elevation of Shot point	In line coordinate	Transverse coordinate	Depth of shot point	
			0.		0,0,0

Multiple shotpoints must be listed in increasing in-line coordinate order.

GEOPHONE LINES (Travel times from shot points must be in the same order as on shot point line)

Geophone number	Elevation of geophone	In line coordinate	Transverse coordinate	Travel time Shot point #	Layer	Travel time Shot point #	Layer	Travel time Shot point #	Layer	Travel time Shot point #	Layer	Travel time Shot point #	Layer	Travel time Shot point #	Layer
1															
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															

Figure 5.--Data input form for entering data into SIPT1.FOR (after Haeni, 1986b).

Table 1:--Example of free-field data-input set for SIPT1.FOR

Format for input data to SIPT1.FOR program	Explanation of data lines
Simsbury Minister Brook (Htfd. Fire Ins. Co.),	Title
2,6,3,1,5.,16.66,2.0,0,0,0,0,0,0,0,0,0,0,0	Problem control
1,700.,0.,0.,0	Velocity override
`1',2.,12.,0.,0.	Spread 1 control data
`1',173.,0.,0.,8.,0.,0.,0	Shot 1 data
`2',69.,1000.,0.,8.,0.,0.,0	Shot 2 data
1,173.,200.,0.,63.,2.,133.,3	Spread 1, geophone
2,173.,250.,0.,75.,2.,130.,3	locations, arrival
3,173.,300.,0.,85.,3.,126.,3	times, and layer
4,173.,350.,0.,90.,3.,123.,3	selection
5,173.,400.,0.,93.,3.,120.,3	
6,173.,450.,0.,97.,2.,116.,3	
7,173.,500.,0.,103.,3.,115.,3	
8,173.,550.,0.,107.,3.,112.,3	
9,173.,600.,0.,110.,3.,104.,2	
10,173.,650.,0.,115.,3.,94.,2	
11,172.,700.,0.,119.,3.,85.,2	
12,171.,750.,0.,124.,3.,74.,2	
`2',2.,12.,0.,0.	Spread 2 control data
`3',173.,400.,0.,10.,0.,0.,0	Shot 3 data
`4',155.,1600.,0.,4.,0.,0.,0	Shot 4 data
1,171.,750.,0.,94.,2.,152.,3	Spread 2 geophone
2,169.,800.,0.,104.,3.,150.,3	locations, arrival
3,169.,850.,0.,109.,3.,146.,3	times, and layer
4,169.,900.,0.,113.,3.,142.,3	selection
5,169.,950.,0.,118.,3.,139.,3	
6,169.,1000.,0.,120.,3.,134.,3	
7,169.,1050.,0.,125.,3.,130.,3	
8,169.,1100.,0.,128.,3.,129.,3	
9,169.,1150.,0.,132.,3.,126.,3	
10,168.,1200.,0.,137.,3.,123.,3	
11,165.,1250.,0.,140.,3.,119.,3	
12,164.,1300.,0.,143.,3.,116.,3	

Table 2:--Example of fixed-field data-input set for SIPT1.FOR

Format for input data to SIPT1.FOR program	Explanation of data lines
Simsbury Minister Brook (Htfd. Fire Ins. Co.)	Title
2 6 3 1 5.0 16.66 2. 0.0 0.0 0.0 0.0 0.0 0.0	Problem control
0.0 0.0 1 1 1[1/]	
1 700. 0. 0. 0.	Velocity override
1 2 12 0. 0	Spread 1 control data
1 1 173. 0. 0. 8. 0. 0. 0	Shot 1 data
1 2 69. 1000. 0. 8. 0. 0. 0	Shot 2 data
1 1 173. 200. 0. 63. 2 133. 3	Spread 1, geophone
1 2 173. 250. 0. 75. 2 130. 3	locations, arrival
1 3 173. 300. 0. 85. 3 126. 3	times, and layer
1 4 173. 350. 0. 90. 3 123. 3	selection
1 5 173. 400. 0. 93. 3 120. 3	
1 6 173. 450. 0. 97. 2 116. 3	
1 7 173. 500. 0. 103. 3 115. 3	
1 8 173. 550. 0. 107. 3 112. 3	
1 9 173. 600. 0. 110. 3 104. 2	
1 10 173. 650. 0. 115. 3 94. 2	
1 11 172. 700. 0. 119. 3 85. 2	
1 12 171. 750. 0. 124. 3 74. 2	
2 2 12 0. 0	Spread 2 control data
2 3 173. 400. 0. 10. 0. 0. 0	Shot 3 data
2 4 155. 1600. 0. 4. 0. 0. 0	Shot 4 data
2 1 171. 750. 0. 94. 2 152. 3	Spread 2, geophone
2 2 169. 800. 0. 104. 3 150. 3	locations, arrival
2 3 169. 850. 0. 109. 3 146. 3	times, and layer
2 4 169. 900. 0. 113. 3 142. 3	selection
2 5 169. 950. 0. 118. 3 139. 3	
2 6 169. 1000. 0. 120. 3 134. 3	
2 7 169. 1050. 0. 125. 3 130. 3	
2 8 169. 1100. 0. 128. 3 129. 3	
2 9 169. 1150. 0. 132. 3 126. 3	
2 10 168. 1200. 0. 137. 3 123. 3	
2 11 165. 1250. 0. 140. 3 119. 3	
2 12 164. 1300. 0. 143. 3 116. 3	

[1/] These five data items belong on the problem control line in the input file in columns 66-72, 73-76, 78, 79, and 80 respectively. The last three items cause the following to be done, respectively: 1) the ray trace is printed; 2) the offset shotpoint to end-of-spread time adjustment is made; and 3) the open spaces between refracting horizons are filled in with the appropriate layer number. Values of 0.0 will cancel these procedures.

The interactive interpretation program, SIPT1.FOR, is now called from an on-line library on the computer or from one of the disk drives. To load the program, type the following command:

SIPT1.FOR <CR>[1/]

The program provides a series of prompts that allow the interpreter a number of choices during the interpretation process. Only the most frequently used options are discussed here.

1) Enter input file name (or <CR> to exit): (prompt)

   SIMS 2A (response)

   Discussion: SIMS 2A is the file name of the input-data file.

2) Enter input FMT type: C = Card, F = Free Field: (prompt)

   F (response)

   Discussion: Format type can be card image--that is--fixed fields of data or free field where the data elements are separated by commas. F tells the program that the input data set is in the free field format (each piece of data is separated by a comma) or C tells the program that the input data set is in the fixed field format (each piece of data is placed in a particular row and column).

3) Enter output unit: P = LPT, T = Terminal, B = Both: (prompt)

   T (response)

   Discussion: T is for small 72- or 80-column terminals, and is the most common choice. B will place a 132-column output file on the machine's disk storage device for later retrieval by a line printer.

4) Enter new Exit, -6 thru +6 or <CR> for old: (prompt)

   <CR> (response)

   Discussion: This statement lets the interpreter exit the program at various places. <CR> returns control to the choice assigned on the problem control line. The program will query the user concerning all data-processing options even when those options have been eliminated by the exit point. It is necessary to respond to these queries with an answer that is consistent with the exit number previously entered.

[1/] The symbol <CR> is an instruction to strike the enter or carriage return key.

5) The program title and the data on the problem control line are now printed out.

6) Table of SP & Geo data: T to type, <CR> to suppress: (prompt)

T (response)

Discussion: The table of input data should always be printed the first time through the program, because the program has editing features that will flag typographic and other obvious data entry errors. If this happens, the message "error on input cards" will be printed. Execution of the program will be terminated at this point, and the error can be corrected via the computer editor. The input geophone and shotpoint data table is now printed out. These data need to be checked for typographic errors not caught by the editor.

7) T-D plot: 1 = raw, 2 = datum, 3 = Pre-D, 4 = L1 remvd: (prompt)

1 (response)

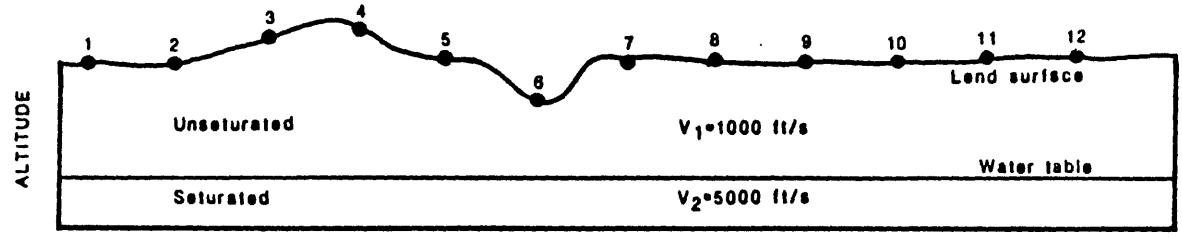
Discussion: The most common response is to plot the raw time-distance data. This plot uses the raw field data to construct a time-distance plot. If the field site has a lot of topographic relief, the raw time-distance curve may not have straight line segments and refined layer assignments may be hard to make (fig. 6). Under these conditions, selection of the datum corrected time-distance plot may help the interpreter. If this option is chosen, the raw seismic traveltimes are corrected to a datum plane constructed by a least-square fit through the geophone elevations. The result of this procedure is that the local topographic features are smoothed out and the resulting time-distance plot may aid the interpreter in deciding which layer is associated with each arrival time. Pre-D plots the arrival times just prior to depth computation of layer 1; these are not normally used.

The time-distance plot is now printed and layer 1 velocity is computed. If no layer 1 assignments are made on the time-distance plot, the default value of 457 m/s (meter per second) is used by the program.

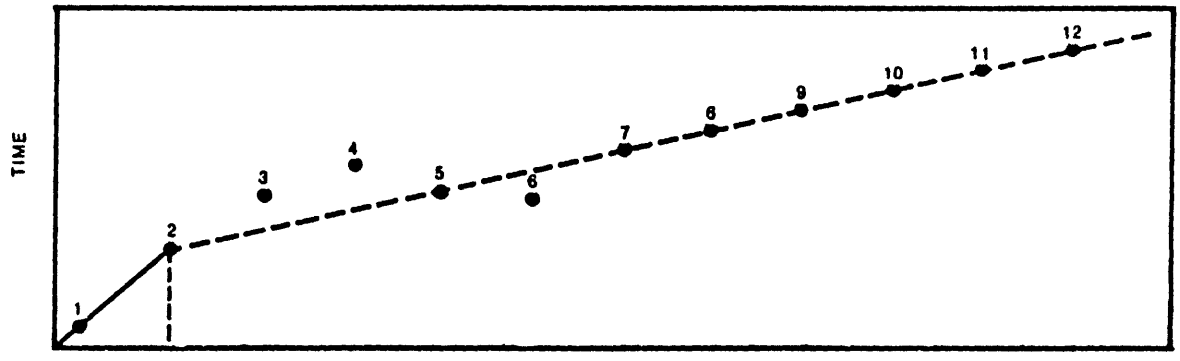
If layer 1 is very irregular, the time-distance plot still may be hard to interpret; in this case, the program provides another option. This option removes layer 1 from the refraction times and plots a new time-distance graph. The program has an exit point that allows the interpreter to end the program after the time-distance plot is printed; or it can be run to completion.

At this point in the interpretation process, the interpreter needs to spend some time working with the time-distance plot.

(A) Topographic profile and geologic section



(B) Raw time-distance curve as plotted by interpretation program.



(C) Datum-corrected time-distance plot as plotted by interpretation program.

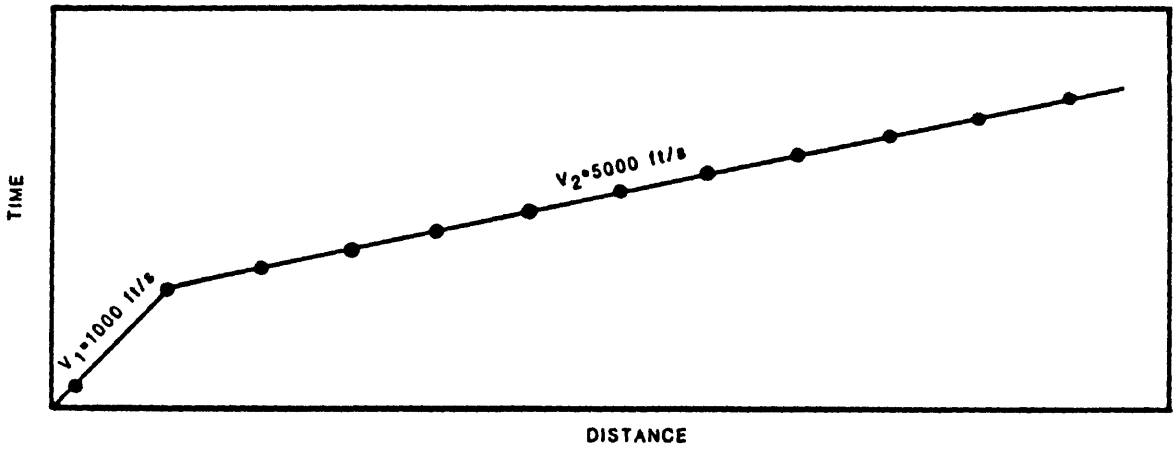


Figure 6.--Effect of topographic relief on raw and datum corrected time-distance plots (after Haeni, 1986b).

The preliminary layer assignments made in the data preparation phase are checked for obvious errors on the time-distance plot. The interpreter reconciles the general form of the time-distance plot with prior knowledge of the geology of the area. For example, if the area is known to have dry sand and gravel overlying saturated sand and gravel, overlying crystalline bedrock, the time-distance plot should show three linear segments. If the water table and bedrock are thought to be relatively flat surfaces, the layer velocities derived from the time-distance plot should be within the range of expected values.

Any unexpected results need to be analyzed before proceeding with the interpretation process. For example, a large shift in the middle of a time-distance plot segment might indicate an error in reading, recording, or entering the traveltime data. Reversed shots that plot in the same direction indicate, for example, an encoding error (fig. 7).

The time-distance plot needs to be inspected for continuity and uniformity between spreads. For example, if the refracting surface is flat over two or more spreads, the crossover distance or intercept time at all shot points would be similar. If the refracting surface is getting deeper, such as in a bedrock valley, the crossover distance or intercept time would be increasing. Two shots in opposite directions but located close to each other would have similarly shaped time-distance plots unless an abrupt change in the refractor depth exists. Figure 8 illustrates some of these principles, and the following discussion gives the symbols and generalized relationships.

Crossover distances:

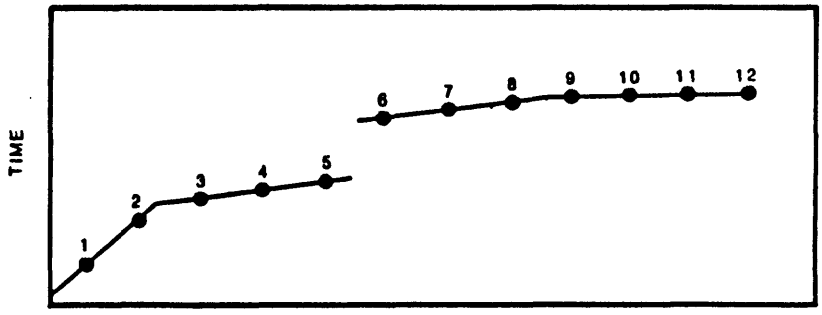
$x_{c1,1}$ = Crossover distance for interface between layers 1 and 2--that is, the water table--from shotpoint 1. The values of $x_{c1,1}$ through $x_{c1,2}$ will all be similar because the water table and land surface are relatively flat and parallel.

$$x_{c1,1} = x_{c1,2} = x_{c1,3} = x_{c1,4} = x_{c1,5} = x_{c1,6}$$

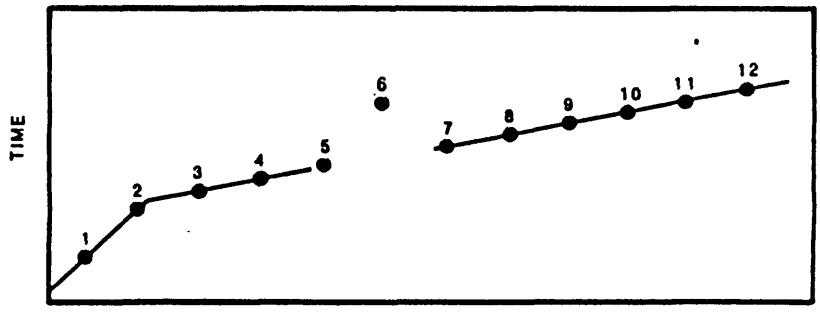
$x_{c2,1}$ = Crossover distance for interface between layers 2 and 3--that is, the bedrock surface--from shotpoint 1. These values will increase as the rock gets deeper.

$$x_{c2,1} < x_{c2,2} < x_{c2,3} < x_{c2,4} < x_{c2,5} < x_{c2,6}$$

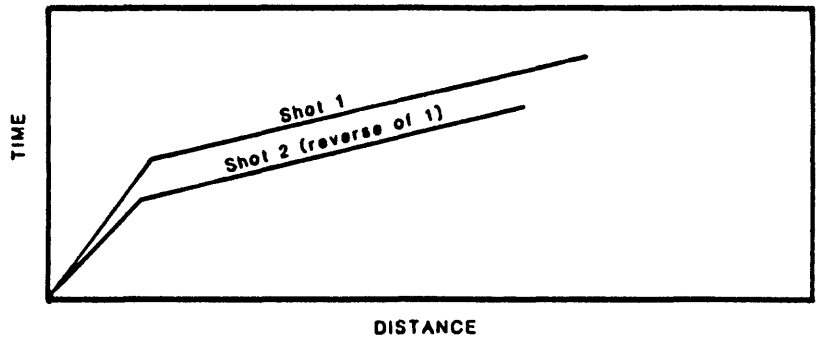
Layer velocities:



Possible error: Travel time at geophone 6 misread on seismograph record and all subsequent geophones referenced to 6



Possible errors: Just geophone 6 misread on seismograph record or typographic error in entering geophone 6 date on computer.



Possible error: Shot 2 has been encoded incorrectly since it was the reverse of shot 1.

Figure 7.--Common errors indicated by unusual time-distance plots (after Haeni, 1986b).

$v_{1,1}$ = Velocity of sound in layer 1 from shotpoint 1, unsaturated unconsolidated deposits. The values of $v_{1,1}$ through $v_{1,6}$ will all be about the same if the deposit is homogeneous.

$$v_{1,1} = v_{1,2} = v_{1,3} = v_{1,4} = v_{1,5} = v_{1,6}$$

$v_{2,1}$ = Apparent velocity of sound in layer 2 from shotpoint 1, saturated unconsolidated deposits. These values would represent the true velocity of sound in layer 2 and are about equal since the water table is a flat surface.

$$v_{2,1} = v_{2,2} = v_{2,3} = v_{2,4} = v_{2,5} = v_{2,6}$$

$v_{3,1}$ = Apparent velocity of sound in layer 3 from shotpoint 1, the bedrock. The apparent velocity of sound in layer 3 as determined from shots to the left of the spreads ($v_{3,2} = v_{3,4} = v_{3,6}$) is less than the apparent velocity as determined from shots to the right of the spreads ($v_{3,1} = v_{3,3} = v_{3,5}$) due to the dipping bedrock surface (fig. 8).

$$v_{3,1} = v_{3,3} = v_{3,5} < v_{3,2} = v_{3,4} = v_{3,6}$$

After obvious errors are reconciled and corrected, the interpreter needs to look at the time-distance plot in detail. The individual segments should be drawn in and used to refine the layer assignments further.

The straight line segments on the time-distance plot can be drawn using the following guidelines:

A. If the land surface is relatively flat, the first refracting surface is the water table, and the saturated zone has a significant thickness, a straight line segment with an inverse slope of about 1.524 km/s can be aligned with any existing data points.

B. The slow surface layer segment can now be constructed through the origin and points below the 1.524-km/s line. All available geologic data should be used to help the interpreter make the proper layer assignments. If, for example, the shot hole were drilled to the water table, the value of the crossover distance to layer 2 could be calculated from the formulas referenced in the theory section. All first-arrival times from geophones between the shot point and this crossover distance must be direct arrivals and assigned to layer 1.

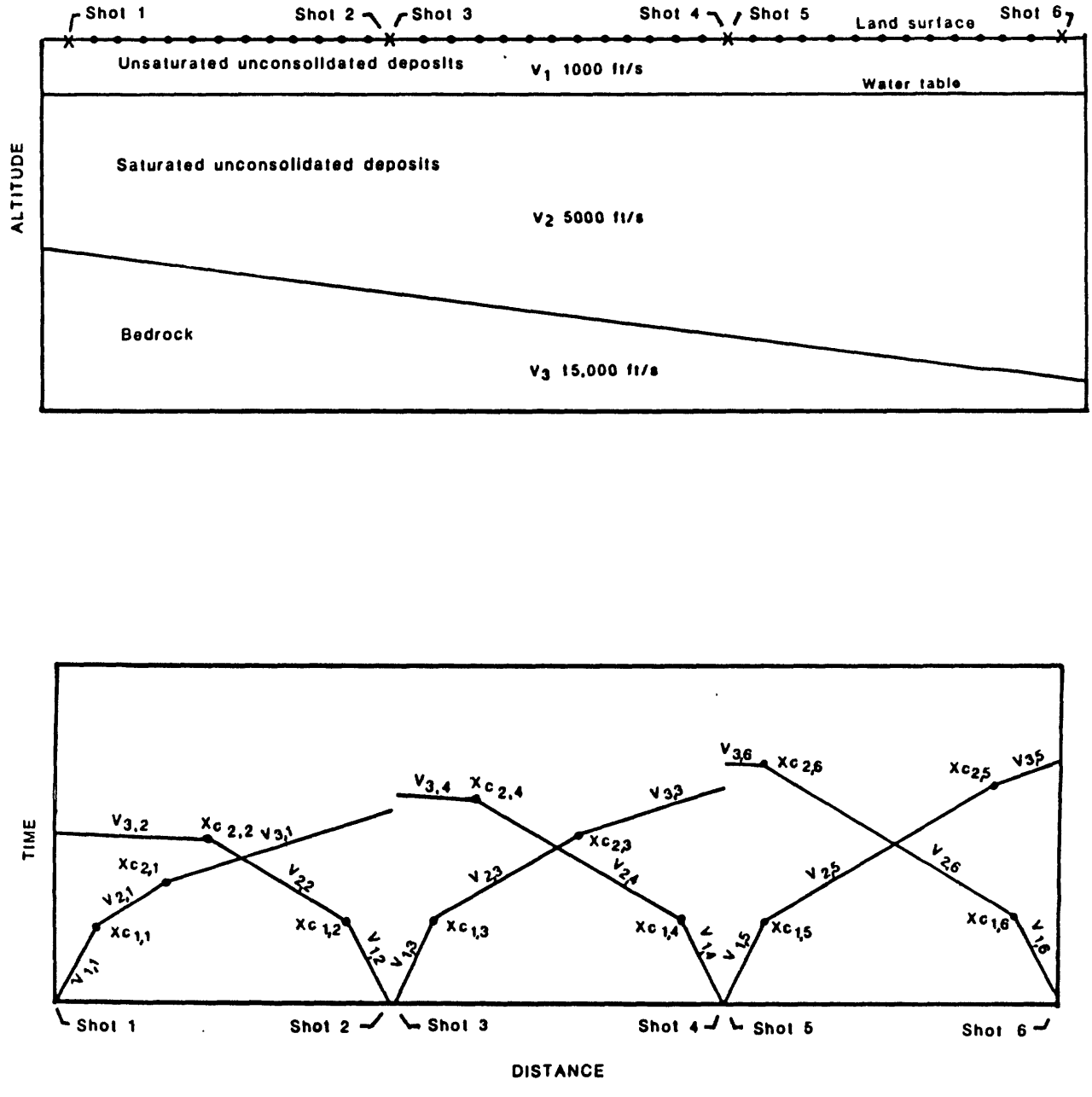


Figure 8.--Geologic section and time-distance plot showing the general relationships of seismic velocities and crossover distances between three seismic refraction spreads (after Haeni, 1986b).

C. The remaining data points are used to construct line segments that represent refracted sound waves from deeper layers. If the deep refracting layers have little or no relief, the segments on the time-distance plots would be straight lines. If there is relief on these surfaces or if the velocity of sound varies significantly in any of the overlying subsurface units, these data points will not form a straight line.

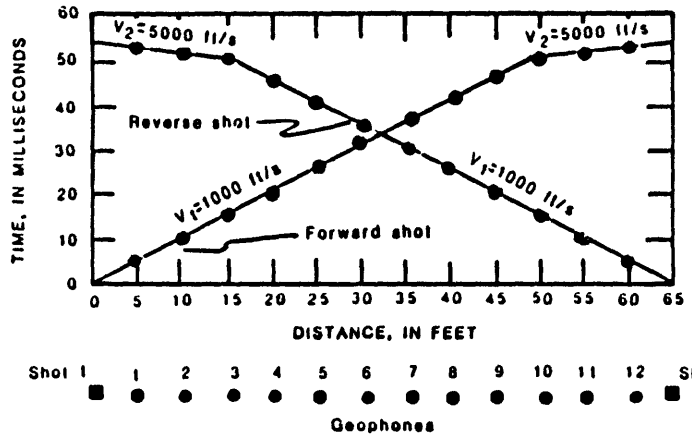
D. The principle of reciprocity also can be used to help construct traveltime plots. An examination of figure 9 indicates that the traveltime from shot point 1 to geophone 12 is the same as from shot point 2 to geophone 1. In general, the seismic traveltime from a source at point A to a geophone at point B is equal to that from a source at B to a geophone at A. For the arrangement shown in figure 9, the offsets from shot point 1 to geophone 1 and from shot point 2 to geophone 12 are small. Hence, the reciprocity principle is applicable and constrains the traveltimes for the end geophones. A good example of this principle is also shown in figure 10 for shot points 2 and 4.

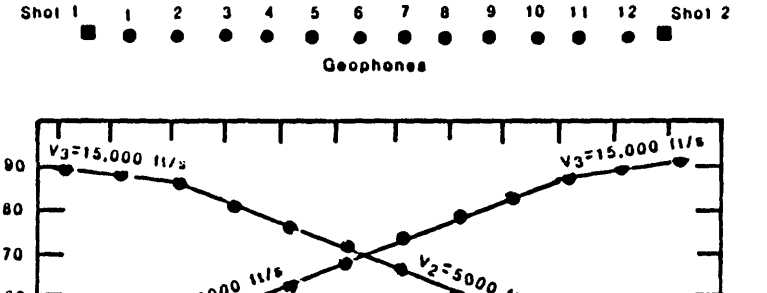
E. Extending the arrival time curves back to the time axis also may help to construct traveltime plots. The arrival times for the geophone array to the right of shot point 2 and for the array to the left of shot point 4 are shown in figure 10. Figure 10 shows that the time-distance plots for the first two velocity layers are symmetrical with respect to the time axis.

Because v_1 and v_2 are equal for both time-distance plots, the intercept times (t_i) also will be equal. Therefore, the line fit to the arrival times for the v_2 layer on each time-distance plot will meet the time axis at t_i for shot points 2 and 3. This property constrains the line fit to arrival times. In general, then, for two geophone arrays laid in opposite directions for which the shot point is halfway between the arrays, the intercept times from common horizons will be equal. This property also is applied appropriately to shot points 3, 4, and 5 in figure 8.

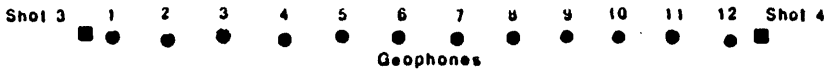
At this point in the interpretation process, some layer assignments near the crossover points may be in question. This needs to be indicated on the time-distance plot so that both options may be tried in subsequent computer runs.

8) Velocity tables: T to type, <CR> to suppress: (prompt)

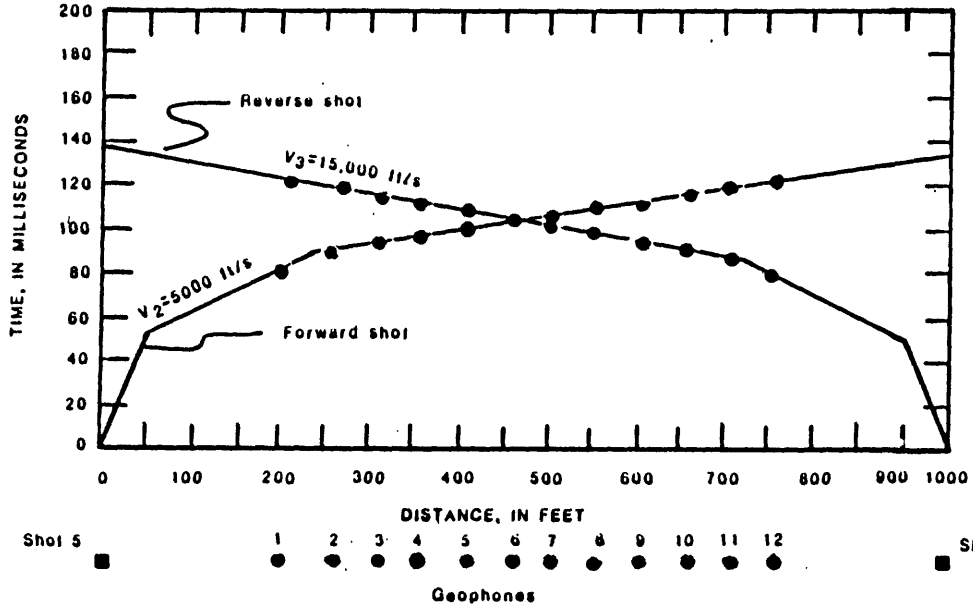




A. Field setup: five feet between geophones and 5 feet between shot point and first geophone.



B. Field setup: Twenty five feet between geophones and 10 feet between first geophone and shot point.



C. Field setup: Fifty feet between geophones and 200 feet between first geophone and shot point.

Figure 9.--Field set up and resultant time-distance plots for determining seismic velocities in a geologic section with three layers. The field set up in (a) determines v_1, that in (b) determines v_2, and that in (c) determines v_3 (after Haeni, 1986b).

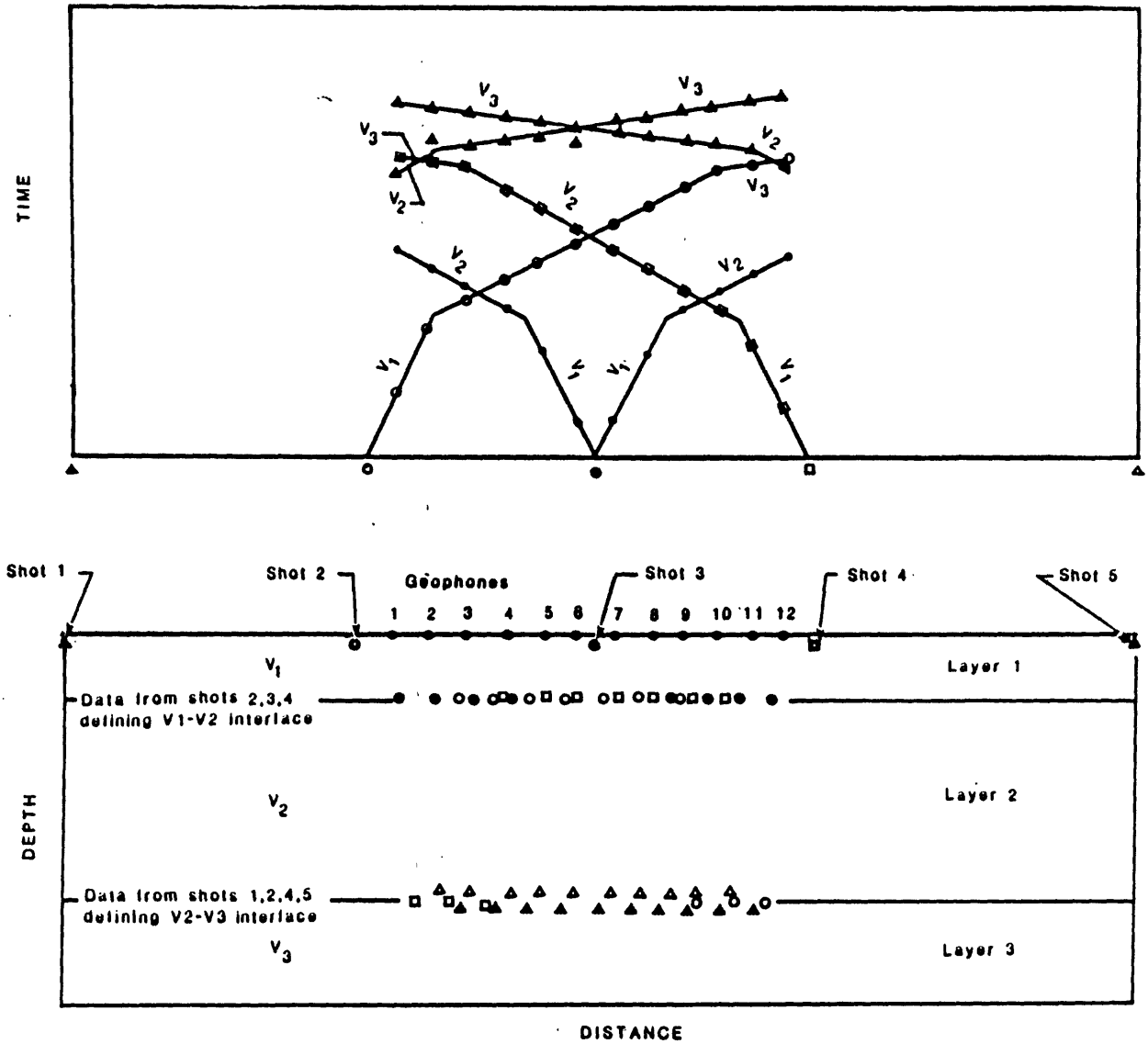


Figure 10.--Time-distance plot and interpreted seismic section resulting from a single geophone spread with five shotpoints (after Haeni, 1986b).

T (response)

Discussion: This is an important step in the interpretation process. This table needs to be printed out and thoroughly reviewed. Incorrect layer assignments or data input error of individual geophone times may cause the velocities of individual layers to seem too low or too high. For example:

If layer 1 geophones are given layer 2 assignments, the velocity of sound in layer 2, computed by regression, will be too low. Conversely, if layer 2 geophones are given layer 1 assignments, the velocity in layer 1 will be too high (fig. 11). The velocity table, therefore, helps the interpreter in assigning the correct layer to refracted geophone travel times.

The velocity tables are printed out.

NOTE: It must be remembered that the velocities computed by regression are affected by dip and are the apparent velocities. Velocities computed by the Hobson-Overton method are independent of dip effects (Scott, 1973).

9) Table of Ray End Points: T to type, <CR> to suppress: (prompt)

<CR> (response)

Discussion: Normally, this table is used for trouble shooting the program and is not used in the interpretation process.

10) Depths beneath SPS & Geos: T to type, <CR> to suppress: (prompt)

T (response)

Discussion: This table is usually printed out because it lists depths to the individual refractors. If this is the first run, the interpreter need not be concerned with the results. If the obvious errors mentioned earlier have not been corrected, the solution presented here represents initial layer assignments and incorporates any data-entry error.

11) Depth plot: Enter T to type, <CR> to suppress: (prompt)

T (response)

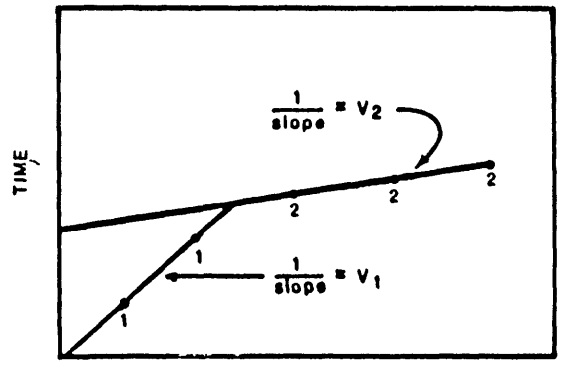
Discussion: This is usually printed because it is the final plot of the interpreted geologic section. It can be suppressed on the initial run.

12) Enter input file name or <CR> to exit: (prompt)

Enter file name for next run or <CR> to exit program.

<CR> (response)

PROPER LAYER ASSIGNMENTS

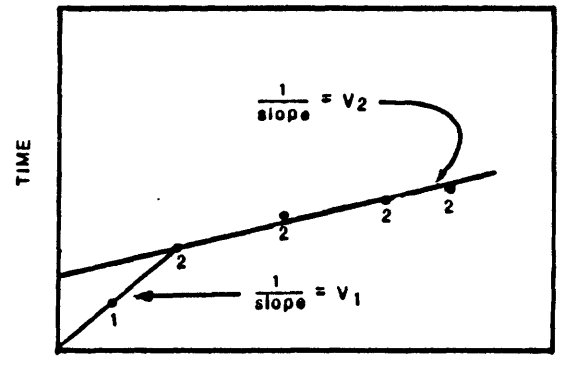


Computed seismic velocities

V_1 = 1000 ft/s

V_2 = 5000 ft/s

LAYER 1 GEOPHONE GIVE LAYER 2 ASSIGNMENT

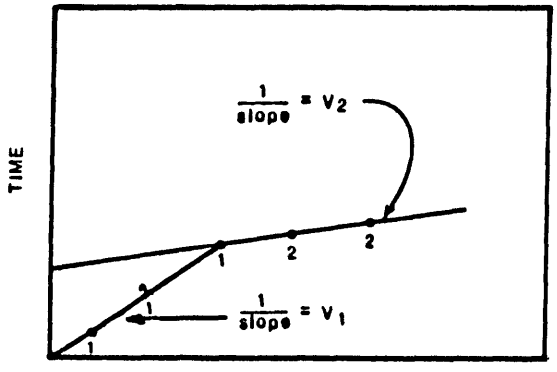


Computed seismic velocities

V_1 = 1000 ft/s

V_2 = 4000 ft/s

LAYER 2 GEOPHONE GIVEN LAYER 1 ASSIGNMENT



Computed seismic velocities

V_1 = 2500 ft/s

V_2 = 5000 ft/s

Figure 11.--Effects of incorrect layer assignments on the velocity of sound as computed by regression in SIPT1.FOR (after Haeni, 1986b).

Discussion: The final <CR> must be used to exit the program or the program file will remain open. On some computer systems, the interpreter will be prevented from accessing the program again until it is closed.

This completes the first computer run of the seismic-refraction interpretation program. As mentioned previously, the interpreter needs to now review the time-distance plot and to make some changes in layer assignments and possibly to input data values.

The data file is now corrected or changed with the computer editor, and a second run of the interpretation program is begun. This run could produce much better results than the first, and the interpreter can start looking at the depth table and the final geologic plot to assess the quality of the solution.

During the second run, the following points need to be checked again by the interpreter:

1) Input data--Were the intended changes entered properly?

2) Velocity tables--Are there still problems with layer velocities or do the velocities look reasonable?

3) Time-distance plot--Were the changes from run #1 made and is the plot now acceptable?

4) Depth table and final plot--Are any water well, shot hole, or geologic data available to check approximate depths?

In some hydrologic studies, few data are available for determining the velocity of sound in layer 1. If this layer is shallow, a completely separate field setup is required to determine the velocity of sound in the layer. Independent control on the thickness of layer 1 may be available from nearby observation wells, swamps, or shot holes. The depth to layer 2, commonly the water table, can be adjusted in the interpretation procedure by using the velocity override option. The seismic velocity for layer 1 is adjusted by trial and error until the solution for the depth to layer 2 generally agrees with field observations. For example, the computer solution often places the water table at depths greater than those observed in the field. This happens when the program uses the default value of 457 m/s for the velocity of sound in layer 1. By decreasing the velocity of sound in layer 1, the water table can be raised to agree with the independent field data. Similarly, the velocity of layer 1 may change from spread to spread. This situation can again be accounted for by using the velocity override option.

At this point in the interpretation process, two or three computer runs have been made, all the obvious encoding and typing errors have been corrected, and the depth to layer 2 generally agrees with independent field data. The interpreter is now ready to assess the quality of the cross-sectional plot, keeping in mind that several layer assignments near the crossover points on the time-distance plot still may be in question.

The best method for confirming the adequacy of the seismic interpretation is to compare the results with well or test hole control points. Generally, this is not possible, so the interpreter needs to qualitatively judge the results. One way to do this is to examine the final cross-sectional plot. The data points that define a particular refractor are printed on the plot. If the data points from reversed shot points overlap and form a continuous line with little scattering, a relatively good computer solution has been obtained. If, however, there is significant scatter in these points, the solution is not satisfactory. An example of an satisfactory and an unsatisfactory computer solution of the second refracting layer is shown in figure 12.

Several field and interpretational errors can lead to the unsatisfactory solution shown in figure 12-B. Any departure of the subsurface from the simplifying assumptions listed in the beginning of this section can lead to an unsatisfactory solution. Some common causes of this are inhomogeneous layers, such as localized buried swamp or peat deposits, or lateral lithologic facies changes. Layer misassignments and errors in field measurements also can cause unsatisfactory solutions.

If all of the arrival time data from one shot point are consistently late, the data from that shot point need to be checked and the possibility of the sound source being located in an atypical setting, for example, recent fill or swamp deposits, needs to be considered. If the atypical setting is considered likely, a provision in the program makes it possible to add or subtract a constant time delay to each geophone in the spread (see "Fudge time" in Scott and others, 1972).

It is important to realize that the best solution of the delay-time technique occurs when the refracting surface of interest has many overlapping data points from shots in opposite directions. If only a few isolated data points define a refracting layer, the computer solution is questionable, even though it may seem unambiguous. It is this fact that requires the collection of field data in a manner whereby the delay-time technique, and consequently the computer program, can produce a reasonable solution.

The questionable layer assignments indicated earlier on the time-distance plot near the crossover points can now be tested. The interpreter needs to make several computer runs, systematically varying the questionable layer assignments until a best fit is achieved on the cross-sectional plot that agrees with drill-hole data.

After four to eight computer runs, the interpreter would have a good idea where the problems are in the solution and whether or not the controlled and deliberate changes made in the runs have any effect. Under normal circumstances, the interpreter stops the computer-interpretation process when little or no improvement is indicated.

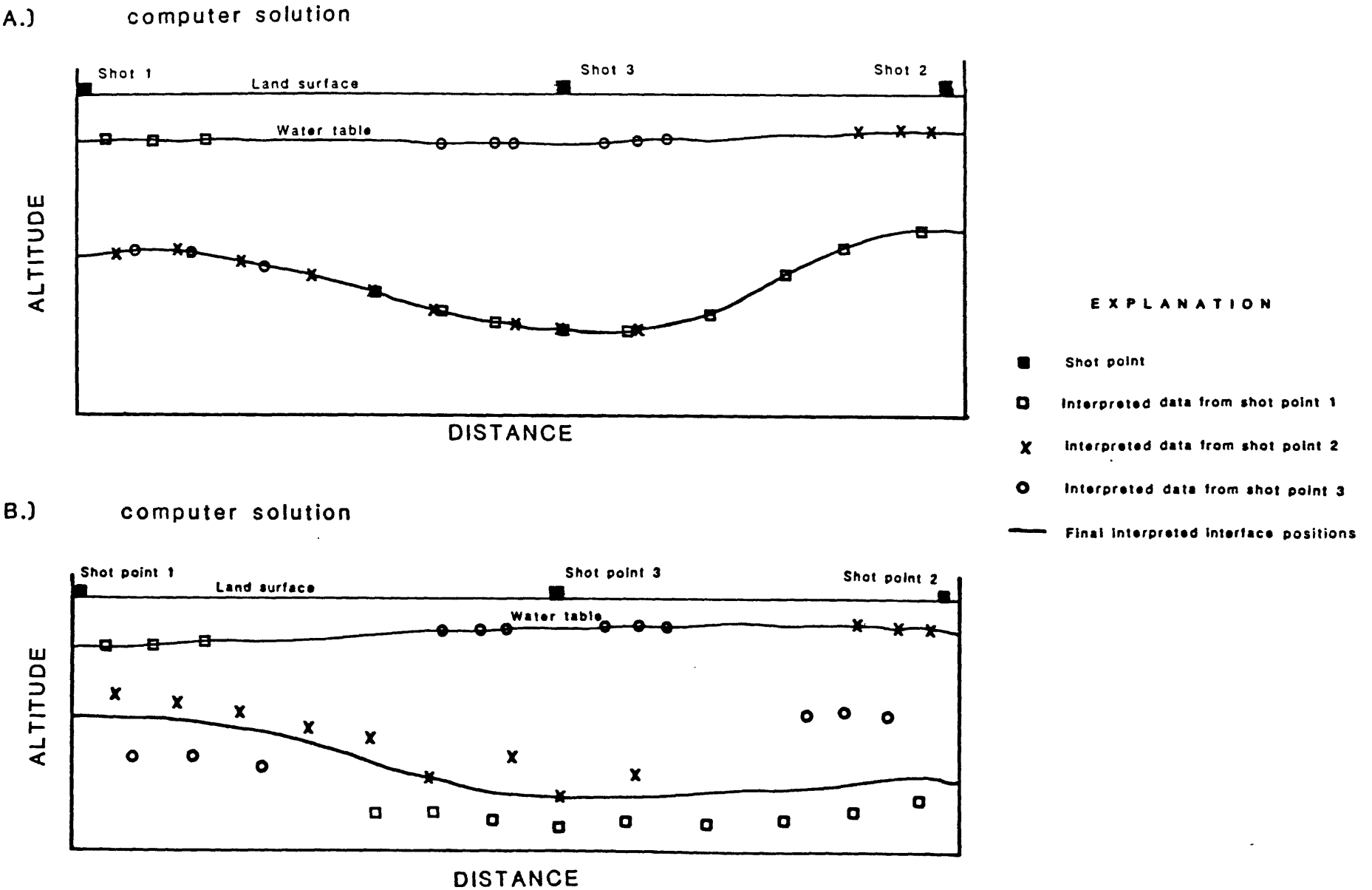


Figure 12.--Sketch showing satisfactory and unsatisfactory computer solutions (after Haeni, 1986b).

CONCLUSIONS

It must be emphasized that, because the geologic section never exactly meets the simplifying assumptions that have been made, a perfect solution is never possible. Field data must be collected with the interpretation process in mind to define the geologic layers of interest. A single geophone spread with one shot on each end rarely provides enough data to completely define a multilayer geologic section. Multiple shots and multiple spreads are needed to provide adequate data in most field situations, and overlapping velocity segments from multiple opposite shot points provide the best data for computer interpretation. In the end, the interpreter needs to make the final interpretation with the information provided by the computer-assisted seismic-refraction modeling process.

REFERENCES

Ackermann, H. D., Pankratz, L. W., and Dansereau, D. A., 1983, A comprehensive system for interpreting seismic refraction arrival time data using interactive computer methods: U.S. Geological Survey Open-File Report 82-1065, 265 p.

Ballantyne, E. J., Jr., Campbell, D. L., Mentemeier, S. H., and Wiggins, Ralphe, 1981, Manual of Geophysical Hand-calculator Programs: TI Volume: Tulsa, Oklahoma, Society of Exploration Geophysicists, 166 p.

Barthelmes, A. J., 1946, Application of continuous profiling to refraction shooting: Geophysics, v. 11, no. 1, p. 24-42.

Birch, F. S., 1976, A seismic ground-water survey in New Hampshire: Ground Water, v. 14, no. 2, p. 94-100.

Bonini, W. E., and Hickok, E. A., 1958, Seismic refraction method in ground-water exploration: Transactions of the American Institute of Mining, Metallurgical, and Petroleum Engineers, v. 211, p. 485-488.

Burke, K. B. S., 1967, A review of some problems of seismic prospecting for ground water in surficial deposits, in Morey, L. W., ed., Mining and Ground Water Geophysics: Canada Geological Survey Economic Geology Report no. 26, p. 569-579.

Burwell, E. B., 1940, Determination of ground-water levels by the seismic method: Transactions American Geophysical Union, v. 12, p. 439-440.

Dobrin, M. B., 1976, Introduction to geophysical prospecting, third edition: New York, McGraw-Hill, 630 p.

Domzalski, W., 1956, Some problems of shallow refraction investigations: Geophysical Prospecting, v. 4, no. 2, p. 140-166.

Eaton, G. P., and Watkins, J. S., 1967, The use of seismic refraction and gravity methods in hydrologic investigations, in Morey, L. W., ed., Mining and ground-water geophysics: Geological Survey of Canada, Economic Geology Report 26, p. 554-568.

Gill, H. E., Vecchioli, J., and Bonini, W. E., 1965, Tracing the continuity of Pleistocene aquifers in northern New Jersey by seismic methods: Ground Water, v. 3, no. 4, p. 33-35.

Grant, F. S., and West, G. F., 1965, Interpretation theory in applied geophysics: New York, McGraw-Hill, 583 p.

Green, R., 1962, The hidden layer problem: Geophysical Prospecting, v. 10, no. 2, p. 166-170.

Griffiths, D. H., and King, R. F., 1965, Applied geophysics for engineers geologists (2d ed.): Oxford, England, Pergamon Press, 223 p.

Haeni, F. P., 1978, Computer modeling of the ground-water availability of the Pootatuck River Valley, Newtown, Connecticut: U.S. Geological Survey Water Resources Investigations 78-77, 64 p.

_____, 1986a, Application of seismic refraction methods in ground-water modeling studies in New England: Geophysics, v. 51, no. 2, p. 236-249.

_____, 1986b, Application of seismic-refraction techniques to hydrologic studies: U.S. Geological Survey Open-File Report 84-746, 144 p.

Hawkins, L. V., and Maggs, D., 1961, Nomographs for determining maximum error and limiting conditions in seismic refraction survey with a blind-zone problem: Geophysical Prospecting, v. 9, no. 4, p. 526-532.

Heiland, C. A., 1940, Geophysical exploration: New York, Prentice-Hall, 1013 p.

Mooney, H. M., 1981, Handbook of engineering geophysics, Volume 1: Seismic: Minneapolis, Minnesota, Bison Instruments, Inc., 220 p.

Morgan, N. A., 1967, The use of continuous seismic profiles to solve hidden-layer problems: Geophysical Prospecting, v. 15, no. 1, p. 35-43.

Morrissey, D. J., 1983, Hydrology of the Little Adroscoggin River Valley aquifer, Oxford County, Maine: U.S. Geological Survey Water Resources Investigations 83-4018, 79 p.

Musgrave, A. W., ed., 1967, Seismic refraction prospecting: Tulsa, Oklahoma, Society of Exploration Geophysicists, 604 p.

Palmer, Derecke, 1980, The generalized reciprocal method of seismic refraction interpretation: Tulsa, Oklahoma, Society of Exploration Geophysicists Monograph, 104 p.

Pakiser, L. C., and Black, R. A., 1957, Exploring for ancient channels with the refraction seismograph: Geophysics, v. 22, no. 1., p. 32-47.

Parasnis, D. S., 1979, Principles of applied geophysics, 3d ed.: London, England, Chapman and Hall, (distributed in the United States by J. Wiley & Sons, New York), 275 p.

Redpath, B. B., 1973, Seismic refraction exploration for engineering site investigations: National Technical Information Service AD-768710, 51 p.

Scott, J. H., 1973, Seismic refraction modeling by computer: Geophysics, v. 38, no. 2, p. 271-284.

_____1977a, SIPB.--A seismic inverse modeling program for batch computer systems: U.S. Geological Survey Open-File Report 77-366, 40 p.

_____1977b, SIPT.--A seismic refraction inverse modeling program for timeshare terminal computer system: U.S. Geological Survey Open-File Report 77-365, 35 p.

Scott, J. H., Tibbetts, B. L., and Burdick, R. G., 1972, Computer analysis of seismic refraction data: Bureau of Mines, Report of Investigations 7595, 95 p.

Slotnick, M. M., 1959, Lessons in seismic computing: Tulsa, Oklahoma, Society of Exploration Geophysicists, 268 p.

Soske, J. L., 1959, The blind-zone problem in engineering geophysics: Geophysics, v. 24, no. 2, p. 359-365.

Telford, W. M., Geldart, L. P., Sherriff, R. E., and Keys, D. A., 1976, Applied geophysics: New York, Cambridge University Press, 806 p.

Tolman, A. L., Tepper, D. H., Prescott, J. C., Jr., and Gammon, S. O., 1983, Hydrogeology of significant sand and gravel aquifers in northern New York and southern Cumberland Counties, Maine: Maine Geological Survey, Report 83-1, 4 plates.

Wallace, D. E., 1970, Some limitations of seismic refraction methods in geohydrological surveys of deep alluvial basins: Ground Water, v. 8, no. 6, p. 8-13.

Warrick, R. E., and Winslow, J. D., 1960, Application of seismic methods to a ground-water problem in northeastern Ohio: Geophysics, v. 25, no. 2, p. 505-519.

Zohdy, A. A. R., Eaton, G. P., and Mabey, D. R., 1974, Application of surface geophysics to ground-water investigations: U.S. Geological Survey Techniques of Water-Resources Investigations, book 2, chap. D1, 116 p.

GLOSSARY

Batch-processing: Computer processing in which an entire job (usually large) is completed without further instruction by the user; work that is similar is accumulated and submitted to the computer together to increase efficiency.

Byte: Computer unit of binary digits usually in eight bits representing two numerals or one character.

Inverse modeling: Determining a model which could have given rise to observed effects; usually inverse modeling is not unique.

Migration: Interpretation of seismic data so that reflections are plotted at the locations of the reflectors rather than with respect to observation points.

Reciprocity: The seismic trace from a source at A to a geophone at B is the same as from a source at B to a geophone at A if sources and receivers are similarly coupled to the ground.

Subroutine: A computer program called for as part of a larger program.

ATTACHMENT A: LISTING OF SIPT1.FOR CODE

```
c ************************ MAIN.FOR ************************************
c
c Program SIPT1.FOR -- version 1.0 -- USGS, Hartford, CT -- 08-15-86
c
c Micro-computer implementation of sipt22.fortran (on Denver PRIME).
c Compiled on an IBM PC using the IBM Professional FORTRAN compiler
c (version 1.0).
c Program consists of 9 modules: MAIN.FOR, PART2.FOR, PART3.FOR,
c INITIAL.FOR, S1.FOR, S2.FOR, S3.FOR, S4.FOR and S5.FOR.
c
c This program interprets seismic refraction data for up to 5 spreads,
c each of which with up to 48 geophones and 7 shotpoints, and for 2 to
c 5 layers in the subsurface.
c
c Part 1 computes elevation corrections and plots corrected t-d graph.
c Part 2 computes velocities, makes weath correction, and replots t-d gra
c Part 3 computes migrated depths and makes smoothed depth interpretatior
c
c ********************************************************************
c
c explicit variable specification
c
 character*2 jchar
 character*32 din,dout,blank
        character*1 ifprint,jtrl,jprt,jout,idspr,idsp,krs2,krg,ip,it,
     1        ity,idtest,ibl,is,isq,iast,iplus,idash,icoln,idee,iques,
     2        icee,itee,ipee,ibee,ix,il,jvel
       character*1 begin,br2002
        character*1 enter2
 character*4 ident,iend
c
c common block assignments
c
 common ibl,iques,ip,it,icoln,iplus,isq,idash,is,iast,idee,il,
     1          ident
 common/ibm1/nl,ln,p,tscale,xscale,xsco2,escale,xlim1,xlim2,
     1              iplot
 common/blk0/lg
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,ity
 common/blk2/xg,erp,slope,idip
 common/blk3/ta,tr,dsg
 common/blk4/vva,vha
 common/blk5/idspr,idsp
 common/ibm3/kl,kr,d
 common/blk6/trp,jjoff
 common/blk7/ja,jb,trs,ers,xsp,esp,ls
 common/blk8/prg,erg,prs2,ers2,zrsp,zrg
 common/ibm4/krg,krs2
 common/blk9/eg,es
 common/blk10/blim,itrace,tansg
```

```
 common/blk11/vreg,preg
 common/blk12/vhob,phob
        common/real1/a,avvdd,big,bl,diff,dmy0,dmy1,dmy2,dmy3,ebar,edat1,
     1    edat2,edg,edsp,eg2,eintg,eref,erg2,erx,es2,gpt,gtc,pg2,prg2,
     2    prp,ps2,pts,pts1,pts2,pts3,sppt,sptc,sum1,sum2,sum3,tcg,
     3    tfudge,tg2,tlim,trg,trg2,trs2,tuh,tvg,tvs,vdd,vv,xbar,xcg,
     4    xdat1,xdat2,xintg,xlast,xref,xs,xshift,xtie,xvg,yg,ysp,zsg,
     5    zsp
        common/real2/b,denex,dg2,edif,eint,hv,pbar,prsh,pt,rad,rads,ray,
     1    t,tbar,tc,th,ts,v,vocosg,vvcosg,x1,x2,xc,xint,xtru,z,ztan,ztru
        common/real3/dg,ds,dz,e1,e2,ells,hv2,p1,p2,s1,s2,sgn,tcall,tcor,
     1    tg,tlls,v1,xd,xlls
        common/char1/blank,din,dout,ibee,icee,idtest,iend,ifprint,ipee,
     1    itee,ix,jout
        common/char2/begin,br2002
       common/char3/enter2
        common/integ1/i,i1,iexit,iopen,iover,irep,ixit,ixtrue,iz,j,j1,
     1    jc,jexit,jn,jopen,jr,k,kn,ktest,l,l1,l2,lr,m,mj,mk,ml,ml1,mm,
     2    nixit,nv
        common/integ2/icall,iflag,ii,j2,jcall,jj,jjj,jjtie,k1,k11,k2,
     1    k22,kcall,kk,kt1,kt2,lcall,mcall
        common/integ3/isplt,ixrep,jt,ll,m1,none
c
c arrays
c
 dimension ident(20),ip(103),it(53),
     1   il(5),vreg(5),vhob(5),preg(5),phob(5),zsg(4),
     2   idspr(5),nj(5),nk(5),xshift(5),ja(5),jb(5),bl(5),
     3   vva(5,5),vha(5,5),
     4   tvs(7),avvdd(7),zrsp(7),idsp(7,5),esp(7,5),
     5   xsp(7,5),ysp(7,5),zsp(7,5),tuh(7,5),edsp(7,5),
     6   tfudge(7,5),es(7,5),kl(7,5),kr(7,5),sptc(7,5),sppt(7,5),
     7   ls(7,5),zrg(48,7),
     8   ps2(7,5,2),es2(7,5,2),trs2(7,5,2),
     9   ers(7,5,4),trs(7,5,4),
     1   ers2(7,5,4,2),prs2(7,5,4,2),krs2(7,5,4,2),
     1   tvg(48),xvg(48),dsg(48),prp(48),xs(48),erx(48),
     2   prg2(48,2),erg2(48,2),trg2(48,2),
     3   tcg(48,2),xcg(48,2),xintg(48,2),eintg(48,2),
     4   edg(48,5),eg(48,5),xg(48,5),yg(48,5),
     5   gtc(48,5),gpt(48,5),
     6   pg2(48,5,2),eg2(48,5,2),tg2(48,5,2),
     7   erp(48,5,4),trp(48,5,4),ta(48,7,5),
     8   tr(48,7,5),lg(48,7,5),d(48,7,5),vdd(48,7,5),p(48,7,5),
     9   prg(48,7,5),erg(48,7,5),trg(48,7,5),krg(48,7,5)
c
```

```
c ***********  part 1
*********************************************************
c
c data initialization
c
       mm=5
        mj=7
        mk=48
        ml=5
        ml1=4
        iz=0
        i1=1
        big=9999999.
       iend='end'
        ibl=' '
        is='s'
        isq='#'
        iast='*'
        iplus='+'
        idash='-'
        icoln=':'
        idee='>'
        iques='?'
        icee='c'
        itee='t'
        ipee='p'
        ibee='b'
        ix='x'
        il(1)='1'
        il(2)='2'
        il(3)='3'
        il(4)='4'
        il(5)='5'
        blank='                                        '
        br2002='n'
        enter2='n'
  din=blank
 iopen=0
        jopen=0
c
c input data file, format type, and output unit specification from termin
c
1 write(*,10000)
10000 format(/,
     1' Welcome to the PC refraction interpretation program SIPT1.'/
     2' If you have any questions about the use of this program, please'
     3/' refer to the documentation in the USGS Open File Report.'/
     4' Please use lower case for all input from the terminal.  Good luc
     5k!',
     6//,' Enter input file name (or <cr> to exit):',/)
 read (*,10002) din
10002 format(a32)
 if(din.eq.blank) go to 9999
        if(iopen.eq.0) go to 10003
```

```
        close (10)
10003   open (10,file=din,status='old',form='formatted')
        iopen=1
        begin='n'
 ifprint=ibl
c
 write(*,10004)
10004 format(' Enter input format type (c=card, f=free field):',/)
 read (*,10006) ifprint
10006 format(a1)
 if(ifprint.ne.icee) ifprint=itee
c
 write(*,10007)
10007 format(
     1' Enter output unit (p=printer, t=terminal, b=both):',/)
 jprt=ix
 jtrl=ix
 read (*,10006) jout
 if (jout.eq.ipee) jtrl=ibl
 if (jout.eq.itee) jprt=ibl
c
c initialization of arrays, etc.
10008 call initial
c
c read input data file
c title card
 read(10,9) ident
9 format (20a4)
 if (ident(1).eq.iend) go to 1
c problem control card
 if(ifprint.eq.icee) read(10,11) nm,iexit,nl,nv,escale,xscale,
     1   tscale,edat1,xdat1,edat2,xdat2,slope,a,blim,tlim,itrace,
     2   jjoff,idip
11 format (i1,i2,2(1x,i1),1x,3(f4.0,1x),4f7.1,f7.4,f7.1,f7.2,f4.1,
     1  1x,3i1)
 if(ifprint.eq.itee) read(10,*) nm,iexit,nl,nv,escale,xscale,
     1   tscale,edat1,xdat1,edat2,xdat2,slope,a,blim,tlim,itrace,jjoff,
     2   idip
 ixit=0
 if (iabs(iexit).le.6) ixit=iexit
 if (nl.le.1) nl=2
 if (nm.eq.0) nm=1
 if (blim.eq.0.0) blim=0.5
 if (escale.eq.0.) escale=5.0
 if (xscale.eq.0.) xscale=8.3333333
 if (tscale.eq.0.) tscale=1.0
 if (tlim.eq.0.0) tlim=10.0
 ln=nl-1
c
c exit point specification from terminal
c
 write(*,10012)
10012 format(' Enter exit point (+6 to -6):',/)
 read(*,'(i2)') ixit
```

```
c read (*,11012) jchar
c11012 format(a2)
c decode(jchar,11112) jexit  --> invalid statement in IBM PROFORT
c11112 format(i2)
c if(jexit.ne.0.and.iabs(jexit).le.6) ixit=jexit
 if(jprt.eq.ibl) go to 11013
c
c output file name specification from terminal
c
        write(*,20012)
20012   format(' Enter output file name:',/)
        read (*,10002)dout
        if(jopen.eq.1) close(20)
21112   open (20,file=dout,status='new',form='formatted')
 jopen=1
c
c print input data
c
22012 write(20,13) ident,nm,ixit,nl,nv,escale,xscale,tscale,
     1 edat1,xdat1,edat2,xdat2,slope,a,blim,tlim,itrace,jjoff,idip
13 format (/,50x,13h** SIPT1 **  //,20a4,//,
     1     17hcontrol card data,
     2   13x,20hp l o t  s c a l e s,5x,9hd a t u m,4x,
     3   15ho v e r r i d e,
     4   4x,11hv a l u e s,/,30x,4helev,4x,5hhoriz,
     5  3x,4htime,4x,29hp o i n t   1   p o i n t   2/1h ,5hsprds,2x,
     6  44hexit  layers  vcards  ft/col  ft/row  ms/col,
     7  2(3x,4helev,4x,5hx pos),3x,28hslope  intcpt   blim    tlim,3x,
     8  13htrace off dip/1h ,5h-----,2x,4h----,13(2x,6h------),2x,
     9  13h----- --- ---/1h ,i3,2i7,i8,2x,7f8.1,f8.4,f7.1,f8.2,f8.1,i6,
     1  i5,i4)
11013   if(jtrl.ne.ibl)write(*,10013) ident,nm,ixit,nl,nv,escale,
     1 xscale,tscale,edat1,xdat2,slope,a,blim,tlim,itrace,jjoff,idip,
     2 edat2,xdat2
10013 format(/30x,'** SIPT1 ** '//1x,20a4//' control card data'//
     1 ' s x l v  ft/c  ft/r  ms/c edat1/2 xdat1/2  slope intcp blim  t
     2lim ',' t o d'/1x,i1,3i2,3f6.1,2f8.1,f7.4,f6.1,f5.2,f6.1,2x,3i2/,
     3  26x,2f8.1)
 nixit=0
 if(ixit.ge.0) go to 14
 ixit=iabs(ixit)
 nixit=1
14 if (nv.eq.0) go to 24
 if(jprt.eq.ibl) go to 11015
 write(20,15) (m,m=1,nm)
15 format (//,14hvelocity cards,//,4x,5(6x,'spread',i2))
 write(20,10014) (ibl,m=1,nm)
10014 format(2x,'layer',5(a1,'   vv     vh  '))
 write(20,11014) (ibl,m=1,nm)
11014 format(2x,'-----',5(a1,'------ ------'))
11015 if(jtrl.eq.ibl) go to 10016
 write(*,10015)(m,m=1,nm)
10015 format(' velocity cards'//5(6x,'spread',i2))
 write(*,11016) (ibl,m=1,nm)
```

```
11016 format(1x,'l',5(a1,'   vv     vh '))
10016 do 20 i=1,nv
c
c read velocity override card
c
 if(ifprint.eq.icee)read(10,17) l,(vva(m,l),vha(m,l),m=1,nm)
17 format (i1,10f6.0)
 if(ifprint.eq.itee) read(10,*) l,(vva(m,l),vha(m,l),m=1,nm)
 if(jprt.ne.ibl) write(20,19) l,(vva(m,l),vha(m,l),m=1,nm)
19 format (4x,i1,2x,10f7.0)
 if(jtrl.ne.ibl)write(*,10019)l,(vva(m,l),vha(m,l),m=1,nm)
10019 format(1x,i1,10f7.0)
 do 20 m=1,nm
 vva(m,l)=vva(m,l)/1000.
 vha(m,l)=vha(m,l)/1000.
20 continue
24 ity=ibl
 if(jtrl.eq.ibl) go to 25
c
c sp & geo data table specification from terminal
c
  write(*,22)
22 format(//
     1' Table of sp & geo data? (t to type, <cr> to suppress):'/)
 read (*,10006)ity
c
c read spread control card
c
25 do 60 m=1,nm
 if(ifprint.eq.icee) read(10,27) idspr(m),nj(m),nk(m),xshift(m),
     1 ixtrue
27 format (a1,2i3,f11.1,1x,i1)
 if(ifprint.eq.itee) read(10,*) idspr(m),nj(m),nk(m),xshift(m),
     1 ixtrue
 if(jprt.ne.ibl)write(20,29) idspr(m),nj(m),nk(m),xshift(m),
     1   ixtrue
29 format(//28h shotpoint and geophone data//2x,6hspread,1x,a1,1h,,
     1 i2,1x,11hshotpoints,,i3,1x,19hgeophones, xshift =,f8.1,
     2  9h, xtrue =,i2//2x,
     3  55hsp   elev     x loc     y loc     depth uphole t  fudge t,
     1  2x,6hend sp/2x,10h-- -------,6(1x,8h--------))
 if(jtrl.ne.ibl.and.ity.ne.ibl)write(*,29)idspr(m),nj(m),nk(m),
     1 xshift(m),ixtrue
 xlast=-big
 jn=nj(m)
c
c read shotpoint control cards
c
 do 35 j=1,jn
 if(ifprint.eq.icee) read(10,31) idtest,idsp(j,m),esp(j,m),
     1 xsp(j,m),ysp(j,m),zsp(j,m),tuh(j,m),tfudge(j,m),jc
31 format (a1,2x,a1,3f7.1,3f5.1,1x,i1)
 if(ifprint.eq.itee) read(10,*) idsp(j,m),esp(j,m),
     1 xsp(j,m),ysp(j,m),zsp(j,m),tuh(j,m),tfudge(j,m),jc
```

```
 if (xsp(j,m).lt.xlast) go to 9990
 if(ifprint.eq.icee.and.idtest.ne.idspr(m)) go to 9990
 if(jprt.ne.ibl) write(20,33) idsp(j,m),esp(j,m),xsp(j,m),
     1 ysp(j,m),zsp(j,m),tuh(j,m),tfudge(j,m),jc
33 format (1h ,2x,a1,f8.1,5f9.1,i6)
 if(jtrl.ne.ibl.and.ity.ne.ibl)write(*,33)idsp(j,m),esp(j,m),
     1 xsp(j,m),ysp(j,m),zsp(j,m),tuh(j,m),tfudge(j,m),jc
 es(j,m)=esp(j,m)-zsp(j,m)
 if (jc.eq.1) ja(m)=j
 if (jc.eq.2) jb(m)=j
 xlast=xsp(j,m)
35 continue
 if(jprt.eq.ibl) go to 10040
 write(20,41) (idsp(j,m),j=1,jn)
41 format (/,13x,
     1  46harrival times + fudge t and layers represented,
     2  //,30h  geo  elev     x loc    y loc,1x,7(5x,3hsp ,a1))
 write(20,42) (ibl,j=1,jn)
42 format('  --- ------',2(' --------'),2x,7(a2,'-------'))
10040 if(jtrl.eq.ibl.or.ity.eq.ibl) go to 10043
 write(*,10041)(idsp(j,m),j=1,jn)
10041 format(/13x,46harrival times + fudge t and layers represented,
     1  //,1x,21h g  elev      x   y  ,7(3x,3hsp ,a1))
 write(* ,10042) (ibl,j=1,jn)
10042 format('  - ------ ------ --- ',7(a1,'------'))
10043 xlast=-big
 kn=nk(m)
c
c read geophone control card data
c
 do 55 k=1,kn
 if(ifprint.eq.icee) read(10,43) idtest,ktest,eg(k,m),xg(k,m),
     1 yg(k,m),(ta(k,j,m),lg(k,j,m),j=1,jn)
43 format (a1,i3,3f7.1,7(f5.1,1x,i1))
 if(ifprint.eq.itee) read(10,*) ktest,eg(k,m),xg(k,m),
     1 yg(k,m),(ta(k,j,m),lg(k,j,m),j=1,jn)
 if(ktest.ne.k.or.xg(k,m).lt.xlast) go to 9990
 do 44 j=1,jn
 if (ta(k,j,m).ne.0.0) ta(k,j,m)=ta(k,j,m)+tfudge(j,m)
 if (ta(k,j,m).le.0.) lg(k,j,m)=0
 tr(k,j,m)=ta(k,j,m)
44 continue
 if(jprt.ne.ibl)write(20,45) k,eg(k,m),xg(k,m),yg(k,m),
     1 (ta(k,j,m),lg(k,j,m),j=1,jn)
45 format (2x,i2,f8.1,2f9.1,2x,7(f7.1,i2))
 if(jtrl.ne.ibl.and.ity.ne.ibl)write(*,10045)k,eg(k,m),xg(k,m),
     1 yg(k,m),(ta(k,j,m),lg(k,j,m),j=1,jn)
10045 format(1x,i2,2f7.0,f4.0,1x,7(f5.0,i2))
 xlast=xg(k,m)
55 continue
 if((m.ne.nm.or.nixit.eq.0).and.jprt.ne.ibl)write(20,57) ident
57 format (///,20a4)
60 continue
c
```

```
c the following routine (lines 60 -95) corresponds to SIPT rev22.
c (rev23 contains a different algorithm).
c
c if ixtrue=0, correct x-distances.  ixtrue=0 implies that x-distances
c for sps & geos are measured along ground surface and no correction has
c been made for slope.
c
 if(ixtrue.ne.0) go to 95
 do 90 m=1,nm
 jn=nj(m)
 kn=nk(m)
 xs(1)=xg(1,m)
 do 62 k=2,kn
 xs(k)=xs(k-1)+sqrt((xg(k,m)-xg(k-1,m))**2-(eg(k,m)-eg(k-1,m))**2)
62 continue
 do 64 j=1,jn
 if(xsp(j,m).gt.xg(1,m)) go to 66
64 continue
 j=jn+1
66 j1=j-1
 if(j1.eq.0) go to 70
 xtie=xs(1)
 xref=xg(1,m)
 eref=eg(1,m)
 do 68 j=1,j1
 jr=j1-j+1
 xtie=xtie-sqrt((xref-xsp(jr,m))**2-(eref-esp(jr,m))**2)
 xref=xsp(jr,m)
 eref=esp(jr,m)
 xsp(jr,m)=xtie
68 continue
70 j1=j1+1
 do 72 j=j1,jn
 if(xsp(j,m).gt.xg(kn,m)) go to 74
 do 71 k=2,kn
 if(xg(k,m).lt.xsp(j,m)) go to 72
 xsp(j,m)=xs(k-1)+(xs(k)-xs(k-1))*(xsp(j,m)-xg(k-1,m))/
     1 (xg(k,m)-xg(k-1,m))
71 continue
72 continue
 go to 78
74 j1=j
 xtie=xs(kn)
 xref=xg(kn,m)
 eref=eg(kn,m)
 do 76 j=j1,jn
 xtie=xtie+sqrt((xsp(j,m)-xref)**2-(esp(j,m)-eref)**2)
 xref=xsp(j,m)
 eref=esp(j,m)
 xsp(j,m)=xtie
76 continue
78 do 80 k=1,kn
80 xg(k,m)=xs(k)
90 continue
```

```
c
c compute v1 using direct dist dd from shot to geos for which lg=1
c
95      if(nixit.eq.0)then
 if ((jprt.eq.ix).and.(jtrl.eq.ix))then
           write(20,100)
          write(*,100)
       elseif((jprt.eq.ix).and.(jtrl.eq.ibl))then
           write(20,100)
        elseif((jprt.eq.ibl).and.(jtrl.eq.ix))then
           write(*,100)
        endif
        endif
c
100 format (//,2x,42hv1 for direct rays and direct distances dd)
 sum1=0.0
 pts1=0.0
 do 150 m=1,nm
 sum2=0.0
 pts2=0.0
 jn=nj(m)
 do 120 j=1,jn
 sum3=0.0
 pts3=0.0
 kn=nk(m)
 do 110 k=1,kn
 if (lg(k,j,m).ne.1) go to 110
 d(k,j,m)=sqrt ((eg(k,m)-es(j,m))**2+(xg(k,m)-xsp(j,m))**2+
     1 (yg(k,m)-ysp(j,m))**2)
 vdd(k,j,m)=d(k,j,m)/ta(k,j,m)
 pts3=pts3+1.0
 sum3=sum3+vdd(k,j,m)
 pts2=pts2+1.0
 sum2=sum2+vdd(k,j,m)
 pts1=pts1+1.0
 sum1=sum1+vdd(k,j,m)
110 continue
 if (pts3.eq.0.) go to 115
 avvdd(j)=sum3/pts3
 go to 120
115 avvdd(j)=0.0
120 continue
 if (pts2.eq.0.) go to 126
 sum2=sum2/pts2
126 if (nixit.eq.1) go to 150
        if((jprt.eq.ix).and.(jtrl.eq.ix))then
          write(20,130) idspr(m)
          write(*,130)  idspr(m)
        elseif((jprt.eq.ix).and.(jtrl.eq.ibl))then
          write(20,130) idspr(m)
        elseif((jprt.eq.ibl).and.(jtrl.eq.ix))then
          write(*,130)  idspr(m)
        endif
130 format(/,2x,'spread',1x,a1,5x,'sp  geo    dd       v1   avg v1',
```

```
     1/,15x,'--  ---',3('  ------'))
 do 140 j=1,jn
 do 134 k=1,kn
 if(vdd(k,j,m).eq.0.) go to 134
        if((jprt.eq.ix).and.(jtrl.eq.ix))then
    write(20,132) idsp(j,m),k,d(k,j,m),vdd(k,j,m)
          write(*,132)  idsp(j,m),k,d(k,j,m),vdd(k,j,m)
        elseif((jprt.eq.ix).and.(jtrl.eq.ibl))then
          write(20,132) idsp(j,m),k,d(k,j,m),vdd(k,j,m)
        elseif((jprt.eq.ibl).and.(jtrl.eq.ix))then
          write(*,132)  idsp(j,m),k,d(k,j,m),vdd(k,j,m)
        endif
c
132 format(16x,a1,i5,f8.1,3pf8.0)
134 continue
 if(avvdd(j).ne.0.) then
        if((jprt.eq.ix).and.(jtrl.eq.ix))then
          write(20,136) avvdd(j)
          write(*,136)  avvdd(j)
        elseif((jprt.eq.ix).and.(jtrl.eq.ibl))then
          write(20,136) avvdd(j)
 elseif((jprt.eq.ibl).and.(jtrl.eq.ix))then
          write(*,136)  avvdd(j)
        endif
        endif
136 format(38x,3pf8.0)
140 continue
150 continue
 if (pts1.eq.0.) go to 152
 sum1=sum1/pts1
152 vreg(1)=sum1
 if (vreg(1).le.0.0) vreg(1)=1.5
156 if (nixit.eq.0)then
           if((jprt.eq.ix).and.(jtrl.eq.ix))then
                 write(20,157) sum1
                 write(*,157)  sum1
            elseif((jprt.eq.ix).and.(jtrl.eq.ibl))then
                 write(20,157) sum1
           elseif((jprt.eq.ibl).and.(jtrl.eq.ix))then
                 write(*,157)  sum1
            endif
        endif
157 format(27x,'  avg of all',3pf7.0,/,27x,'  ----------------',/)
c
c apply xshift to xsp and xg arrays
c also compute dir dist between (xsp,ysp,esp) and (xg,yg,eg) if
c lg.ne.1 and compute plot positions p
c
 do 180 m=1,nm
 jn=nj(m)
 kn=nk(m)
 do 165 k=1,kn
 xg(k,m)=xg(k,m)+xshift(m)
165 continue
```

```
 do 175 j=1,jn
 xsp(j,m)=xsp(j,m)+xshift(m)
 do 170 k=1,kn
 if (lg(k,j,m).ne.1) d(k,j,m)=sqrt((xg(k,m)-xsp(j,m))**2+(yg(k,m)-
     1 ysp(j,m))**2+(eg(k,m)-esp(j,m))**2)
 if (xg(k,m)-xsp(j,m)) 168,166,167
166 if (k.eq.kn) go to 168
167 p(k,j,m)=xsp(j,m)+d(k,j,m)
 go to 170
168 p(k,j,m)=xsp(j,m)-d(k,j,m)
170 continue
175 continue
180 continue
c
c find number of geo to the left, kl(j,m), and right, kr(j,m) of each sp.
c if none, kl or kr set to zero.
c
 do 188 m=1,nm
 jn=nj(m)
 kn=nk(m)
 do 186 j=1,jn
 if (xsp(j,m).le.xg(1,m)) go to 183
 if (xsp(j,m).ge.xg(kn,m)) go to 184
 do 181 k=2,kn
 if (xg(k,m).gt.xsp(j,m)) go to 182
181 continue
182 kl(j,m)=k-1
 kr(j,m)=k
 go to 186
183 kl(j,m)=0
 kr(j,m)=1
 go to 186
184 kl(j,m)=kn
 kr(j,m)=0
186 continue
188 continue
c
c find end sps for each spread and set limits of x axis for t-d graph
c
 xsco2=xscale/2.0
        if(jjoff.ne.0) go to 201
 do 200 m=1,nm
 jn=nj(m)
 kn=nk(m)
 if (ja(m).ne.0) go to 192
 do 190 j=1,jn
 if (xsp(j,m).le.xg(1,m)) go to 190
 ja(m)=j-1
 if (j.gt.1) go to 192
 ja(m)=1
 if (m.eq.1) xlim1=xg(1,1)
 go to 194
190 continue
 ja(m)=jn
```

```
192 if (m.ne.1) go to 195
 j=ja(m)
 xlim1=amin1(xsp(j,1),xg(1,1))
194 xlim1=xlim1+xsco2
195 if (jb(m).ne.0) go to 197
 do 196 j=1,jn
 if (xsp(j,m).lt.xg(kn,m)) go to 196
 jb(m)=j
 go to 197
196 continue
 jb(m)=jn
 if (m.ne.nm) go to 200
 xlim2=xg(kn,nm)
 go to 200
197 if (m.ne.nm) go to 200
 j=jb(m)
 xlim2=amax1(xsp(j,nm),xg(kn,nm))
200 continue
        go to 203
201     do 202 m=1,nm
        jn=nj(m)
        ja(m)=1
202     jb(m)=jn
        xlim1=xsp(1,1)
        xlim2=xsp(jn,nm)
c
c t-d plot selection from terminal
c
203 iplot=0
 if(nixit.eq.1) go to 204
 write(*,10200)
10200 format(/
     1' T-D plot? (0=no plot, 1=raw, 2=datum, 3=pre-d, 4=l1 remvd):',/)
 read (*,*) iplot
c
c test for doing raw time t-d plot
c
10204 if(iplot.eq.1) call plot(1)
204 if (ixit.eq.0) go to 1
c
c fit straight line thru geo elevations
c
205 if (slope.ne.0.0.or.a.ne.0.0) go to 225
 if (xdat1.ne.0.0.or.xdat2.ne.0.0) go to 220
 sum1=0.0
 sum2=0.0
 pts=0.0
 do 212 m=1,nm
 kn=nk(m)
 do 210 k=1,kn
 sum1=sum1+xg(k,m)
 sum2=sum2+eg(k,m)
 pts=pts+1.0
210 continue
```

```
212 continue
 xbar=sum1/pts
 ebar=sum2/pts
 sum1=0.0
 sum2=0.0
 do 216 m=1,nm
 kn=nk(m)
 do 214 k=1,kn
 diff=xg(k,m)-xbar
 sum1=sum1+diff*eg(k,m)
 sum2=sum2+diff**2
214 continue
216 continue
 slope=sum1/sum2
 a=ebar-slope*xbar
 go to 225
220 slope=(edat2-edat1)/(xdat2-xdat1)
 a=edat1-slope*xdat1
225 do 230 m=1,nm
 kn=nk(m)
 do 226 k=1,kn
 edg(k,m)=a+slope*xg(k,m)
226 continue
 jn=nj(m)
        do 228 j=1,jn
 edsp(j,m)=a+slope*xsp(j,m)
228 continue
230 continue
c
c velocity override card analysis -- set any zero vh=vv if vv nonzero
c
 iover=0
 if (vva(1,1).ne.0.0) go to 242
 do 240 m=1,nm
 vva(m,1)=vreg(1)
240 continue
 go to 246
242 iover=1
 if (nm.le.1) go to 246
 do 244 m=2,nm
 if (vva(m,1).eq.0.0) vva(m,1)=vva(1,1)
244 continue
246 do 248 m=1,nm
 do 247 l=2,nl
 if (vva(m,l).eq.0.0) go to 247
 if (vha(m,l).eq.0.0) vha(m,l)=vva(m,l)
247 continue
248 continue
 if (nm.le.1) go to 300
 do 250 l=2,nl
 do 249 m=2,nm
 if (vva(m,l).eq.0.0) vva(m,l)=vva(m-1,l)
 if (vha(m,l).eq.0.0) vha(m,l)=vha(m-1,l)
249 continue
```

```
250 continue
c
c compute and apply vertical time corrections to datum -- print results
c
300 if(nixit.eq.1) go to 303
 if(jtrl.eq.ibl) go to 301
 write(*,10303)vreg(1)
10303 format(/1x,'computed v1 =',3pf8.0/)
 if(iover.eq.0) go to 301
 if(nm.gt.1) write(*,10304) (m,m=1,nm)
10304 format(6x,'spread',5(i8))
 write(*,10305) (vva(m,1),m=1,nm)
 write(*,10301)
10305 format(1x,'override v1 =',5(3pf8.0))
301 if(jprt.eq.ibl) go to 303
 if(iover.eq.0) go to 10302
 if(nm.gt.1) write(20,10304) (m,m=1,nm)
 write(20,10305) (vva(m,1),m=1,nm)
 write(20,10301)
10301 format(1x,7('--------'))
10302 write(20,57) ident
 write(20,302) a, slope
302 format (//,46harrival times corrected to datum (datum elev =,
     1  f8.1,4h + (,f8.4,25h)x), and plot positions d)
303 do 400 m=1,nm
 vv=vva(m,1)
 jn=nj(m)
 kn=nk(m)
c first precompute geo time corr
 do 304 k=1,kn
 tvg(k)=(edg(k,m)-eg(k,m))/vv
304 continue
c then compute sp time corr
 do 306 j=1,jn
 tvs(j)=0.
 if (j.lt.ja(m).or.j.gt.jb(m)) go to 306
 tvs(j)=(edsp(j,m)-es(j,m))/vv
 if (tuh(j,m).ne.0.0) tvs(j)=tvs(j)*tuh(j,m)*vv/zsp(j,m)
306 continue
c apply datum corrections
312 do 332 j=1,jn
 do 330 k=1,kn
 if (lg(k,j,m).ne.1.and.tr(k,j,m).ne.0.0) ta(k,j,m)=tr(k,j,m) +
     1 tvs(j) + tvg(k)
330 continue
332 continue
c
c print results
c
 if (nixit.eq.1.or.jprt.eq.ibl) go to 400
 write(20,383) idspr(m),(idsp(j,m),j=1,jn)
383 format (//,1x,7hspread ,a1,8x,7(10x,'sp ',a1))
 write(20,384) (edsp(j,m),j=1,jn)
384 format(/7x,'elev . . . . .',7(f10.1,4x))
```

```
 write(20,385) (tvs(j),j=1,jn)
385 format (9x,1h./9x,1h.,3x,6hcorr t,7(f12.1,2x))
 write(20,387) (ibl,j=1,jn)
387 format (' geo      .',10x,7(a1,'---t-----d---'))
 do 390 k=1,kn
 write(20,389) k,edg(k,m),tvg(k),(ta(k,j,m),p(k,j,m),j=1,jn)
389 format (1x,i2,2f8.1,1x,14f7.1)
390 continue
400 continue
c
c plot t-d graph
c
450 if(iplot.eq.2) call plot(1)
c
c exit point test and branch
c
 if (ixit.eq.1) go to 1
22222  call part2
        if(begin.eq.'y')go to 1
        call part3
        if(enter2.eq.'y')go to 22222
        if(begin.eq.'y')go to 1
c
9990 if(jprt.ne.ibl)write(20,9991)
9991 format (/,
     1' Error on input cards, computation halted.  See documentation.'/)
 if(jtrl.ne.ibl)write(*,9991)
 go to 1
9999    stop
 end
```

```
 subroutine part2
c ********************  PART2.FOR  ************************************
c
c Program SIPT1.FOR -- version 1.0 -- USGS, Hartford, CT -- 08-15-86
c
c Micro-computer implementation of sipt22.fortran (on Denver PRIME).
c Compiled on an IBM PC using the IBM Professional FORTRAN compiler
c (version 1.0).
c Program consists of 9 modules: MAIN.FOR, PART2.FOR, PART3.FOR,
c INITIAL.FOR, S1.FOR, S2.FOR, S3.FOR, S4.FOR and S5.FOR.
c
c ***************************************************************************
c
c explicit variable specification
c
 character*2 jchar
 character*32 din,dout,blank
        character*1 ifprint,jtrl,jprt,jout,idspr,idsp,krs2,krg,ip,it,
     1        ity,idtest,ibl,is,isq,iast,iplus,idash,icoln,idee,iques,
     2        icee,itee,ipee,ibee,ix,il,jvel
       character*1 begin,br2002
        character*1 enter2
 character*4 ident,iend
c
c common block assignments
c
 common ibl,iques,ip,it,icoln,iplus,isq,idash,is,iast,idee,il,
     1          ident
 common/ibm1/nl,ln,p,tscale,xscale,xsco2,escale,xlim1,xlim2,
     1               iplot
 common/blk0/lg
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,ity
 common/blk2/xg,erp,slope,idip
 common/blk3/ta,tr,dsg
 common/blk4/vva,vha
 common/blk5/idspr,idsp
 common/ibm3/kl,kr,d
 common/blk6/trp,jjoff
 common/blk7/ja,jb,trs,ers,xsp,esp,ls
 common/blk8/prg,erg,prs2,ers2,zrsp,zrg
 common/ibm4/krg,krs2
 common/blk9/eg,es
 common/blk10/blim,itrace,tansg
 common/blk11/vreg,preg
 common/blk12/vhob,phob
        common/real1/a,avvdd,big,bl,diff,dmy0,dmy1,dmy2,dmy3,ebar,edat1,
     1    edat2,edg,edsp,eg2,eintg,eref,erg2,erx,es2,gpt,gtc,pg2,prg2,
     2    prp,ps2,pts,pts1,pts2,pts3,sppt,sptc,sum1,sum2,sum3,tcg,
     3    tfudge,tg2,tlim,trg,trg2,trs2,tuh,tvg,tvs,vdd,vv,xbar,xcg,
     4    xdat1,xdat2,xintg,xlast,xref,xs,xshift,xtie,xvg,yg,ysp,zsg,
     5    zsp
        common/real2/b,denex,dg2,edif,eint,hv,pbar,prsh,pt,rad,rads,ray,
```

```
     1    t,tbar,tc,th,ts,v,vocosg,vvcosg,x1,x2,xc,xint,xtru,z,ztan,ztru
         common/real3/dg,ds,dz,e1,e2,ells,hv2,p1,p2,s1,s2,sgn,tcall,tcor,
     1    tg,tlls,v1,xd,xlls
         common/char1/blank,din,dout,ibee,icee,idtest,iend,ifprint,ipee,
     1    itee,ix,jout
         common/char2/begin,br2002
        common/char3/enter2
         common/integ1/i,i1,iexit,iopen,iover,irep,ixit,ixtrue,iz,j,j1,
     1    jc,jexit,jn,jopen,jr,k,kn,ktest,l,l1,l2,lr,m,mj,mk,ml,ml1,mm,
     2    nixit,nv
         common/integ2/icall,iflag,ii,j2,jcall,jj,jjj,jjtie,k1,k11,k2,
     1    k22,kcall,kk,kt1,kt2,lcall,mcall
         common/integ3/isplt,ixrep,jt,ll,m1,none
c
c arrays
c
 dimension ident(20),ip(103),it(53),
     1   il(5),vreg(5),vhob(5),preg(5),phob(5),zsg(4),
     2   idspr(5),nj(5),nk(5),xshift(5),ja(5),jb(5),bl(5),
     3   vva(5,5),vha(5,5),
     4   tvs(7),avvdd(7),zrsp(7),idsp(7,5),esp(7,5),
     5   xsp(7,5),ysp(7,5),zsp(7,5),tuh(7,5),edsp(7,5),
     6   tfudge(7,5),es(7,5),kl(7,5),kr(7,5),sptc(7,5),sppt(7,5),
     7   ls(7,5),zrg(48,7),
     8   ps2(7,5,2),es2(7,5,2),trs2(7,5,2),
     9   ers(7,5,4),trs(7,5,4),
     1   ers2(7,5,4,2),prs2(7,5,4,2),krs2(7,5,4,2),
     1   tvg(48),xvg(48),dsg(48),prp(48),xs(48),erx(48),
     2   prg2(48,2),erg2(48,2),trg2(48,2),
     3   tcg(48,2),xcg(48,2),xintg(48,2),eintg(48,2),
     4   edg(48,5),eg(48,5),xg(48,5),yg(48,5),
     5   gtc(48,5),gpt(48,5),
     6   pg2(48,5,2),eg2(48,5,2),tg2(48,5,2),
     7   erp(48,5,4),trp(48,5,4),ta(48,7,5),
     8   tr(48,7,5),lg(48,7,5),d(48,7,5),vdd(48,7,5),p(48,7,5),
     9   prg(48,7,5),erg(48,7,5),trg(48,7,5),krg(48,7,5)
c
c *********** beginning of part2 ***************************************
c branch to statement 1200 from part 3
 if(enter2.eq.'y')then
        enter2='n'
        go to 1200
        endif
c
c do regression velocity computation for top of layer 2
c
1000 if (nixit.eq.0.and.jprt.ne.ibl) write(20,57) ident
 jvel=ibl
 if(nixit.eq.1.or.jtrl.eq.ibl) go to 510
c
c velocity table print option
c
 write(*,500)
500 format(//' Velocity tables? (t to type, <cr> to suppress):'/)
```

```
 read (*,10006) jvel
 if(jvel.ne.ibl) write(*,505)
505 format(/)
510 l1=1
 l2=2
 call regv(l2,nixit)
 iover=1
 if(preg(2).gt.0.0) go to 1005
 if(vva(1,2).ne.0.) go to 1010
1001 if (nixit.ne.0)then
        begin='y'
        return
        endif
 if(jprt.ne.ibl) write(20,1002) l2
1002 format (/1x,46hnot enough points to define velocity for layer,i2,
     1 18h use override card)
 if(jtrl.ne.ibl)then
        write(*,1002) l2
        begin='y'
        return
 endif
c
c do hobson-overton vel computation for top of layer 2
c
1005 call hobv(l2,nixit)
c
 vv=(vreg(2)*preg(2)+2.0*vhob(2)*phob(2))/(preg(2)+2.0*phob(2))
 if(vva(1,2).ne.0.) go to 1010
 do 1008 m=1,nm
 vva(m,2)=vv
 vha(m,2)=vv
1008 continue
 iover=0
c
1010 if(nixit.ne.0) go to 1094
 if(jprt.ne.ibl)write(20,1011) l2,vv
1011 format(/' wtd avg velocity for layer',i2,2h =,3pf7.0/' -'
     1 ,6('------'))
 if(jtrl.ne.ibl)write(*,11011)l2,vv
11011 format(/' wtd avg computed v',i1,' =',3pf8.0/)
c
 if(nixit.ne.0.or.iover.eq.0) go to 1094
 if(jprt.eq.ibl) go to 1023
 if (nm.gt.1) write(20,10304) (m,m=1,nm)
 write(20,1021) l2,(vva(m,l2),m=1,nm)
 write(20,1022) l2,(vha(m,l2),m=1,nm)
 write(20,10301)
1021 format(1x,'override vv',i1,' =',5(3pf7.0,1x))
1022 format(10x,'vh',i1,' =',5(3pf7.0,1x))
1023 if(jtrl.eq.ibl) go to 1094
 if(nm.gt.1) write(*,10304) (m,m=1,nm)
 write(*,1021) l2,(vva(m,l2),m=1,nm)
 write(*,1022) l2,(vha(m,l2),m=1,nm)
 write(*,10301)
```

```
c
c compute velocities for deeper layers by regression and hobson method
c
1094    if(nl.le.2) go to 1111
     do 1100 l=3,nl
 if(nixit.eq.0.and.jprt.ne.ibl) write(20,57) ident
 if(jvel.ne.ibl) write(*,505)
 lcall=1
 call regv(lcall,nixit)
 if (preg(l).gt.0.) go to 1096
 if(vva(1,l).gt.0.) go to 1098
 l2=l
 go to 1001
1096 lcall=l
 call hobv(lcall,nixit)
 iover=1
 vv=(vreg(l)*preg(l)+2.*vhob(l)*phob(l))/(preg(l)+2.*phob(l))
 if(vva(1,l).ne.0.) go to 1098
 do 1097 m=1,nm
 vva(m,l)=vv
 vha(m,l)=vv
1097 continue
 iover=0
1098 if(nixit.ne.0.) go to 1100
 if(jprt.ne.ibl) write(20,1011) l,vv
 if(jtrl.ne.ibl) write(*,11011) l,vv
c
 if(nixit.ne.0.or.iover.eq.0) go to 1100
 if(jprt.eq.ibl) go to 1099
 if(nm.gt.1) write(20,10304) (m,m=1,nm)
 write(20,1021) l,(vva(m,l),m=1,nm)
 write(20,1022) l,(vha(m,l),m=1,nm)
1099 if(jtrl.eq.ibl) go to 1100
 if(nm.gt.1) write(*,10304) (m,m=1,nm)
 write(*,1021) l,(vva(m,l),m=1,nm)
 write(*,1022) l,(vha(m,l),m=1,nm)
 write(*,10301)
1100 continue
c
c make tie correction if jjoff.eq.0
c
1111    if(jjoff.ne.0) go to 1150
 do 1145 m=1,nm
 mcall=m
 kn=nk(m)
 j=ja(m)
 if(xg(1,m).ge.xsp(j,m)) go to 1120
c set d negative for geos left of left end sp
 k2=kl(j,m)
 do 1118 k=1,k2
 d(k,j,m)=-d(k,j,m)
1118 continue
1120 jj=j-1
 if(jj.lt.1) go to 1130
```

```
 jjtie=0
 do 1125 l=2,nl
 lcall=1
 call kends(lcall,mcall,jj,kr(jj,m),kn,k11,k22)
 if(k11.eq.0) go to 1125
 kt1=kr(j,m)
 if(kt1.eq.0) go to 1125
 call tie(lcall,mcall,j,jj,k11,k22,kt1,kn,kn,1,jjtie)
1125 continue
 j=j-1
 go to 1120
1130 j=jb(m)
 jn=nj(m)
 if(xg(kn,m).le.xsp(j,m)) go to 1135
c set d negative for geos right of right end sp
 k1=kr(j,m)
 do 1132 k=k1,kn
 d(k,j,m)=-d(k,j,m)
1132 continue
1135 jj=j+1
 if(jj.gt.jn) go to 1145
 jjtie=0
 do 1140 l=2,nl
 lcall=1
 call kends(lcall,mcall,jj,1,kl(jj,m),k11,k22)
 if(k11.eq.0) go to 1140
 kt2=kl(j,m)
 if(kt2.eq.0) go to 1140
 call tie(lcall,mcall,j,jj,k11,k22,1,kt2,kn,2,jjtie)
1140 continue
 j=j+1
 go to 1135
1145 continue
c
c plot t-d graph of pre-depth time values
c
1150 do 1160 m=1,nm
 jn=nj(m)
 kn=nk(m)
 do 1160 j=1,jn
 do 1160 k=1,kn
1160 ta(k,j,m)=tr(k,j,m)
c
 if(iplot.eq.3) call plot(1)
c
c compute depth pts at base of layer 1 (for irep=1,iflag=0)
c (irep.ge.2 is done in part 3)
c
c spread loop starts here & ends at line 1060.
c
1025 if (itrace.ne.0.and.ixit.ge.4.and.jprt.ne.ibl) write(20,1027) l2
1027 format (//,45hintermediate results of ray tracing for layer,i2)
 iflag=0
 rads=sqrt(1.+slope**2)
```

```
 do 1060 m=1,nm
 jn=nj(m)
 kn=nk(m)
 vv=vva(m,11)
 hv=vha(m,12)
 if (hv.le.vv) go to 9992
 rad=sqrt(hv**2-vv**2)
 tansg=vv/rad
 vocosg=tansg*hv
 vvcosg=vv*rad/hv
 denex=hv*rads
 if (jjoff.ne.0) go to 1029
 j1=ja(m)
 j2=jb(m)
 go to 1030
1029 j1=1
 j2=nj(m)
c
c precompute datum elev corr for all geos and store v and trig consts
c i=1 for right-going rays, 2 for left-going
c
1030 do 1034 k=1,kn
 do 1033 i=1,2
 icall=i
 call elcor(tansg,vv,hv,xg(k,m),eg(k,m),a,slope,icall,tcg(k,i),
     1  xcg(k,i),xintg(k,i),eintg(k,i))
1033 continue
1034 continue
c
c shot point loop starts here & ends at line 1049
c
 do 1049 j=j1,j2
 jj=j
c
c initialize for right-going rays
c
 i=1
 ii=2
 k1=kr(j,m)
 if (k1.eq.0) go to 1046
 k2=kn
c
c compute elev time corr and direct dists
c
1036 mcall=m
 jcall=j
 call kends(2,mcall,jcall,k1,k2,k11,k22)
 if (k11.eq.0) go to (1046,1049),i
 call elcor(tansg,vv,hv,xsp(j,m),es(j,m),a,slope,ii,tc,xc,xint,
     1 eint)
c
c re-entry point for outlying shotpoints
c
1037 do 1038 k=k11,k22
```

```
 if (lg(k,jj,m).ne.2) go to 1038
 ta(k,jj,m)=tr(k,jj,m)+tc+tcg(k,i)
 dsg(k)=d(k,j,m)+xc+xcg(k,i)
1038 continue
c
c extrap time at sp and compute coord of end pt of ray beneath sp
c
 mcall=m
 call regres(k11,k22,jj,mcall,2,v,t,pt,0)
 if (pt.eq.0.) go to (1046,1049),i
 z=t*vocosg
 ts=z/vvcosg
 ztan=z*tansg
 ray=sign(sqrt(z**2+ztan**2),z)
 b=slope
 if(i.eq.1) b=-b
 xtru=ray*(vv-b*rad)/denex
 ztru=ray*(rad+vv*b)/denex
 if(i.eq.2) xtru=-xtru
1039 prs2(jj,m,1,i)=xint+xtru
 if(i.eq.2) ztan=-ztan
 prsh=xint+ztan/rads
 ers2(jj,m,1,i)=eint-ztru
 z=es(j,m)-ers2(jj,m,1,i)
 trs2(jj,m,i)=sign(sqrt(z**2+(xsp(j,m)-prs2(jj,m,1,i))**2)/vv,z)
 ps2(jj,m,i)=xsp(j,m)
 es2(jj,m,i)=es(j,m)
 if (trs2(jj,m,i).ge.0.) go to 1041
 ts=0.
 z=0.
 trs2(jj,m,i)=0.
 ers2(jj,m,1,i)=es(j,m)
 prs2(jj,m,1,i)=xsp(j,m)
 prsh=xsp(j,m)
1041 zrsp(jj)=z
c
c compute coord of ray end pts at geos
c
 do 1043 k=k11,k22
 if (lg(k,jj,m).ne.2) go to 1043
 edif=slope*(xintg(k,i)-prsh)
 kcall=k
 jcall=j
 mcall=m
 call htime(kcall,jcall,mcall,prsh,0.,xintg(k,i),edif,hv,th)
 pg2(k,m,i)=xg(k,m)
 eg2(k,m,i)=eg(k,m)
 z=(ta(k,jj,m)-ts-th)*vocosg
 ztan=z*tansg
 ray=sign(sqrt(z**2+ztan**2),z)
 b=slope
 if(i.eq.2) b=-b
 xtru=ray*(vv-b*rad)/denex
 ztru=ray*(rad+vv*b)/denex
```

```
 if(i.eq.1) xtru=-xtru
 erg(k,jj,m)=eintg(k,i)-ztru
 prg(k,jj,m)=xintg(k,i)+xtru
 z=eg(k,m)-erg(k,jj,m)
 trg(k,jj,m)=sign(sqrt(z**2+(xg(k,m)-prg(k,jj,m))**2)/vv,z)
 if (trg(k,jj,m).ge.0.) go to 1042
 z=0.
 erg(k,jj,m)=eg(k,m)
 prg(k,jj,m)=xg(k,m)
 trg(k,jj,m)=0.00001
1042 zrg(k,jj)=z
1043 continue
c
c test for doing outlying sps left of spread, right-going rays
c
 if (i.eq.2) go to 1047
 if (jjoff.ne.0.or.jj.gt.j1) go to 1046
1044 jj=jj-1
 if (jj.gt.0) go to 1045
 sum1=0.
 sum2=0.
 sum3=0.
 pts1=0.
 jjj=0
 do 2045 jj=1,j1
 if(ers2(jj,m,l1,1).eq.0.) go to 2045
 sum1=sum1+trs2(jj,m,1)
 sum2=sum2+prs2(jj,m,l1,1)
 sum3=sum3+ers2(jj,m,l1,1)
 pts1=pts1+1.
 if(jjj.eq.0) jjj=jj
2045 continue
 if(pts1.eq.0.) go to 1046
 tbar=sum1/pts1
 pbar=sum2/pts1
 ebar=sum3/pts1
 do 3045 jj=jjj,j1
 trs2(jj,m,1)=tbar
 prs2(jj,m,l1,1)=pbar
 ers2(jj,m,l1,1)=ebar
 if(jj.lt.j1) krs2(jj,m,l1,1)=iast
3045 continue
 go to 1046
1045 mcall=m
 call kends(2,mcall,jj,kr(jj,m),kn,k11,k22)
 if (k11.eq.0) go to 1044
 go to 1037
c
c initialize for left-going rays
c
1046 i=2
 ii=1
 k1=1
 k2=kl(j,m)
```

```
 if (k2.ne.0) go to 1036
c
c test for doing outlying sps right of spread, left-going rays
c
1047 if (jjoff.ne.0.or.jj.lt.j2) go to 1049
1048 jj=jj+1
 if (jj.le.jn) go to 4048
 sum1=0.
 sum2=0.
 sum3=0.
 pts1=0.
 do 2048 jj=j2,jn
 if(ers2(jj,m,l1,2).eq.0.) go to 2048
 sum1=sum1+trs2(jj,m,2)
 sum2=sum2+prs2(jj,m,l1,2)
 sum3=sum3+ers2(jj,m,l1,2)
 pts1=pts1+1.
 jjj=jj
2048 continue
 if(pts1.eq.0.) go to 1049
 tbar=sum1/pts1
 pbar=sum2/pts1
 ebar=sum3/pts1
 do 3048 jj=j2,jjj
 trs2(jj,m,2)=tbar
 prs2(jj,m,l1,2)=pbar
 ers2(jj,m,l1,2)=ebar
 if(jj.gt.j2) krs2(jj,m,l1,2)=iast
3048 continue
 go to 1049
4048 mcall=m
 call kends(2,mcall,jj,1,kl(jj,m),k11,k22)
 if (k11.eq.0) go to 1048
 go to 1037
1049 continue
1060 continue
c
c compute avg elev of base of layer 1 beneath each geo
c
c enter after returning from part 3
c
1200 do 1400 m=1,nm
 vv=vva(m,1)
 tansg=vv/sqrt(vha(m,2)**2-vv**2)
 mcall=m
 if (iflag.eq.0) call admig(l1,mcall,bl(m))
 jn=nj(m)
 kn=nk(m)
 dg2=(xg(kn,m)-xg(1,m))/float(kn+kn-2)
 do 1255 k=1,kn
 erp(k,m,1)=0.
 trp(k,m,1)=0.
 prp(k)=0.
 erx(k)=0.
```

```
 if (k.eq.1) go to 1235
 x1=(xg(k,m)+xg(k-1,m))/2.
 go to 1240
1235 x1=xg(1,m)-dg2
 if(m.eq.1) x1=x1-dg2
1240 if (k.eq.kn) go to 1245
 x2=(xg(k+1,m)+xg(k,m))/2.
 go to 1250
1245 x2=xg(kn,m)+dg2
 if(m.eq.nm) x2=x2+dg2
1250 call avg(x1,x2,2,prp(k),erx(k))
1255 continue
 do 1285 k=1,kn
 if(erx(k).eq.0.) go to 1285
 kk=k
 if (prp(k).gt.xg(k,m)) go to 1270
1260 kk=kk+1
 if (kk.gt.kn) go to 1278
 if(erx(kk).eq.0.) go to 1260
 go to 1275
1270 kk=kk-1
 if (kk.lt.1) go to 1278
 if(erx(kk).eq.0.) go to 1270
1275 erp(k,m,1)=terp(prp(k),erx(k),prp(kk),erx(kk),xg(k,m))
 go to 1285
1278 erp(k,m,1)=erx(k)-(prp(k)-xg(k,m))*slope
1285 continue
 do 1290 k=1,kn
 if(erx(k).eq.0.) go to 1290
 prp(k)=xg(k,m)
 trp(k,m,1)=(eg(k,m)-erp(k,m,1))/vv
1290 continue
 do 1325 j=1,jn
 trs(j,m,1)=0.
 ers(j,m,1)=0.
1325 continue
1400 continue
c
c fill in missing points
 call fillin(l1)
c l1 set to 0 in fillin if no pts are defined for layer l1
 if(l1.eq.0) then
        begin='y'
        return
        endif
 if (iflag.eq.2.or.ixit.le.3.or.(iflag.eq.1.and.ixit.eq.4))
     1 go to 1061
 iflag=iflag+1
 l1=1
 l2=2
 irep=iflag+1
        br2002='y'
 return
c
```

```
c go to part 3
c
1061 if (nixit.eq.1) go to 1070
c
c print heading for results
c
 if(jprt.eq.ibl) go to 1070
 write(20,57) ident
 write(20,1068)
1068 format (//,48harrival times corrected to base of layer 1, and ,
     1 23helev of base of layer 1)
c
c compute corrected ta for t-d plot
c
1070 do 1092 m=1,nm
 jn=nj(m)
 kn=nk(m)
 do 1082 j=1,jn
 if (j.ge.ja(m).or.jjoff.ne.0) go to 1075
 jj=ja(1)
 go to 1076
1075 if (j.le.jb(m).or.jjoff.ne.0) go to 1077
 jj=jb(nm)
1076 trs(j,m,1)=trs(jj,m,1)
1077 do 1080 k=1,kn
 if (lg(k,j,m).eq.1.or.tr(k,j,m).eq.0.0) go to 1078
 ta(k,j,m)=tr(k,j,m)-trp(k,m,1)-trs(j,m,1)
 go to 1080
1078 ta(k,j,m)=0.0
1080 continue
1082 continue
c
c print results
c
 if (nixit.eq.1.or.jprt.eq.ibl) go to 1092
 write(20,383) idspr(m),(idsp(j,m),j=1,jn)
 write(20,384) (ers(j,m,1),j=1,jn)
 write(20,385) (trs(j,m,1),j=1,jn)
 write(20,387) (ibl,j=1,jn)
 do 1090 k=1,kn
 write(20,389)k,erp(k,m,1),trp(k,m,1),(ta(k,j,m),p(k,j,m),j=1,jn)
1090 continue
1092 continue
c
c plot t-d graph (layer 1 removed)
c
1110 if(iplot.eq.4) call plot(1)
 if (ixit.eq.2) then
        begin='y'
        return
        endif
c return to execute part 3
        return
c the following apply to problems with the earth model
```

```
9992 if(jprt.ne.ibl)then
        write(20,9993) l1,l2
        begin='y'
        return
        endif
 if(jtrl.ne.ibl)then
        write(*,9993) l1,l2
        begin='y'
        return
        endif
9993 format (/1x,25hvelocity inversion, layer,i2,10h and layer,i2,
     1 19h computation halted)
57      format(///,20a4)
383     format (//,1x,7hspread ,a1,6x,7(10x,'sp ',a1))
384     format (/7x,'elev . . . . .',7(f10.1,4x))
385     format(9x,1h./9x,1h.,3x,6hcorr t,7(f12.1,2x))
387     format (' geo     .',10x,7(a1,'---t-----d---'))
389     format(1x,i2,2f8.1,1x,14f7.1)
10006   format(a1)
10301   format(1x,7('--------'))
10304   format(/6x,'spread',5(i8))
        return
 end
```

```
      subroutine part3
c ***********************  PART3.FOR  ***********************************
c
c Program SIPT1.FOR -- version 1.0 -- USGS, Hartford, CT -- 08-15-86
c
c Micro-computer implementation of sipt22.fortran (on Denver PRIME).
c Compiled on an IBM PC using the IBM Professional FORTRAN compiler
c (version 1.0).
c Program consists of 9 modules: MAIN.FOR, PART2.FOR, PART3.FOR,
c INITIAL.FOR, S1.FOR, S2.FOR, S3.FOR, S4.FOR and S5.FOR.
c
c ***************************************************************************
c
c explicit variable specification
c
 character*2 jchar
 character*32 din,dout,blank
      character*1 ifprint,jtrl,jprt,jout,idspr,idsp,krs2,krg,ip,it,
     1       ity,idtest,ibl,is,isq,iast,iplus,idash,icoln,idee,iques,
     2       icee,itee,ipee,ibee,ix,il,jvel
       character*1 begin,br2002
      character*1 enter2
 character*4 ident,iend
c
c common block assignments
c
 common ibl,iques,ip,it,icoln,iplus,isq,idash,is,iast,idee,il,
     1         ident
 common/ibm1/nl,ln,p,tscale,xscale,xsco2,escale,xlim1,xlim2,
     1              iplot
 common/blk0/lg
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,ity
 common/blk2/xg,erp,slope,idip
 common/blk3/ta,tr,dsg
 common/blk4/vva,vha
 common/blk5/idspr,idsp
 common/ibm3/kl,kr,d
 common/blk6/trp,jjoff
 common/blk7/ja,jb,trs,ers,xsp,esp,ls
 common/blk8/prg,erg,prs2,ers2,zrsp,zrg
 common/ibm4/krg,krs2
 common/blk9/eg,es
 common/blk10/blim,itrace,tansg
 common/blk11/vreg,preg
 common/blk12/vhob,phob
      common/real1/a,avvdd,big,bl,diff,dmy0,dmy1,dmy2,dmy3,ebar,edat1,
     1   edat2,edg,edsp,eg2,eintg,eref,erg2,erx,es2,gpt,gtc,pg2,prg2,
     2   prp,ps2,pts,pts1,pts2,pts3,sppt,sptc,sum1,sum2,sum3,tcg,
     3   tfudge,tg2,tlim,trg,trg2,trs2,tuh,tvg,tvs,vdd,vv,xbar,xcg,
     4   xdat1,xdat2,xintg,xlast,xref,xs,xshift,xtie,xvg,yg,ysp,zsg,
     5   zsp
      common/real2/b,denex,dg2,edif,eint,hv,pbar,prsh,pt,rad,rads,ray,
     1   t,tbar,tc,th,ts,v,vocosg,vvcosg,x1,x2,xc,xint,xtru,z,ztan,ztru
```

```
      common/real3/dg,ds,dz,e1,e2,ells,hv2,p1,p2,s1,s2,sgn,tcall,tcor,
     1   tg,tlls,v1,xd,xlls
      common/char1/blank,din,dout,ibee,icee,idtest,iend,ifprint,ipee,
     1   itee,ix,jout
      common/char2/begin,br2002
     common/char3/enter2
      common/integ1/i,i1,iexit,iopen,iover,irep,ixit,ixtrue,iz,j,j1,
     1   jc,jexit,jn,jopen,jr,k,kn,ktest,l,l1,l2,lr,m,mj,mk,ml,ml1,mm,
     2   nixit,nv
      common/integ2/icall,iflag,ii,j2,jcall,jj,jjj,jjtie,k1,k11,k2,
     1   k22,kcall,kk,kt1,kt2,lcall,mcall
      common/integ3/isplt,ixrep,jt,l1,m1,none
c
c arrays
c
 dimension ident(20),ip(103),it(53),
     1   il(5),vreg(5),vhob(5),preg(5),phob(5),zsg(4),
     2   idspr(5),nj(5),nk(5),xshift(5),ja(5),jb(5),bl(5),
     3   vva(5,5),vha(5,5),
     4   tvs(7),avvdd(7),zrsp(7),idsp(7,5),esp(7,5),
     5   xsp(7,5),ysp(7,5),zsp(7,5),tuh(7,5),edsp(7,5),
     6   tfudge(7,5),es(7,5),kl(7,5),kr(7,5),sptc(7,5),sppt(7,5),
     7   ls(7,5),zrg(48,7),
     8   ps2(7,5,2),es2(7,5,2),trs2(7,5,2),
     9   ers(7,5,4),trs(7,5,4),
     1   ers2(7,5,4,2),prs2(7,5,4,2),krs2(7,5,4,2),
     1   tvg(48),xvg(48),dsg(48),prp(48),xs(48),erx(48),
     2   prg2(48,2),erg2(48,2),trg2(48,2),
     3   tcg(48,2),xcg(48,2),xintg(48,2),eintg(48,2),
     4   edg(48,5),eg(48,5),xg(48,5),yg(48,5),
     5   gtc(48,5),gpt(48,5),
     6   pg2(48,5,2),eg2(48,5,2),tg2(48,5,2),
     7   erp(48,5,4),trp(48,5,4),ta(48,7,5),
     8   tr(48,7,5),lg(48,7,5),d(48,7,5),vdd(48,7,5),p(48,7,5),
     9   prg(48,7,5),erg(48,7,5),trg(48,7,5),krg(48,7,5)
c
c *********** beginning of part3 ************************************
c
c branch to statement 2002 from part 2
 if(br2002.eq.'y')then
        br2002='n'
        go to 2002
        endif
c compute depth points at base of layer 1, l.gt.2
c
c layer loop -- refr horiz is between l1 and l2 -- loop ends at 2190
c
2000 if (nl.le.2) go to 2200
 l2=3
2001 l1=l2-1
 if (itrace.ne.0.and.jprt.ne.ibl) write(20,1027) l2
 l1=l1-1
 iflag=0
 if (irep.eq.4) go to 2002
```

```
 irep=1
c
c spread loop starts here & ends at the third line after 2090
c
c entry point from part 2  (for irep.ge.2, iflag.ne.0, l1=1,l2=2)
c
2002 m=1
 ixrep=ixit-irep
c
2003 jn=nj(m)
 kn=nk(m)
 hv=vha(m,l2)
 hv2=hv**2
 vv=vva(m,l1)
 if (hv.le.vv) go to 9992
 tansg=vv/sqrt(hv2-vv**2)
 vocosg=tansg*hv
 if(irep.ne.2) bl(m)=0.
 if (jjoff.eq.0) go to 2004
 j1=1
 j2=jn
 go to 2007
2004 j1=ja(m)
 j2=jb(m)
c
c precompute time and migr corr for all geos, spread m
c also clear working storage
c
2007 do 2011 k=1,kn
 do 2008 lr=1,2
 prg2(k,lr)=0.0
 erg2(k,lr)=0.0
 trg2(k,lr)=0.0
2008 continue
 if (irep.gt.1) go to 2011
 do 2009 lr=1,2
 pg2(k,m,lr)=0.0
 eg2(k,m,lr)=0.0
 tg2(k,m,lr)=0.0
2009 continue
 rad=sqrt(hv2-vva(m,1)**2)
 tvg(k)=trp(k,m,1)*hv/rad
 xvg(k)=(eg(k,m)-erp(k,m,1))*vva(m,1)/rad
 if (l1.le.1) go to 2011
 do 2010 l=2,l1
 rad=sqrt(hv2-vva(m,l)**2)
 tvg(k)=tvg(k)+trp(k,m,l)*hv/rad
 xvg(k)=xvg(k)+(erp(k,m,l-1)-erp(k,m,l))*vva(m,l)/rad
2010 continue
2011 continue
c
c shot point loop starts here & ends at 2090
c
 j=j1
```

```
2012 jj=j
c
c compute time and migr corr at sp j
c
 isplt=0
 if (iflag.ne.0) go to 12018
 if (es(j,m).le.ers(j,m,ll)) go to 2016
 tc=0.0
 xc=0.0
 none=0
 do 2015 l=1,ll
 if (es(j,m).lt.ers(j,m,l)) go to 2015
 if (irep.eq.1) go to 2013
 ls(j,m)=l
 go to 2018
2013 v1=vva(m,l)
 rad=sqrt (hv2-v1**2)
 if (none.eq.0) go to 2014
 xc=xc+(ers(j,m,l-1)-ers(j,m,l))*v1/rad
 tc=tc+trs(j,m,l)*hv/rad
 go to 2015
2014 xc=(es(j,m)-ers(j,m,l))*v1/rad
 tc=((es(j,m)-ers(j,m,l))*hv)/(v1*rad)
 ls(j,m)=l
 none=1
2015 continue
 go to 2017
2016 ls(j,m)=ll
 if(irep.eq.1) go to 12017
 if(es(j,m).ge.ers(j,m,ll)) go to 2018
 ls(j,m)=l2
 isplt=2
 go to 2018
12017 dz=ers(j,m,ll)-es(j,m)
 isplt=1
2017 if (isplt.eq.0) go to 2018
 tc=-dz*vocosg/vv**2
 ells=ers(j,m,ll)
 ds=-dz*tansg
 xc=ds
 go to 2018
c
c initialize for right-going rays
c
12018 if(ls(j,m).eq.2) isplt=1
2018 i=1
 ii=2
 sgn=1.0
 if(kr(j,m).eq.0) go to 2040
 call kends(l2,m,j,kr(j,m),kn,k11,k22)
c
c entry point after initialization for right- or left-going rays
c trace shotpoint rays, irep=1
c
```

```
2019 if (irep.gt.1) go to 2021
 xlls=xsp(j,m)+xc*sgn
 tlls=tc
 if(isplt.ne.0) go to 2023
 if(itrace.ne.0.and.jprt.ne.ibl)write(20,2020) irep,idspr(m),
     1 idsp(j,m),iz,ii,l2,l1,ls(j,m),xsp(j,m),es(j,m),xlls,ells,tlls,
     2 bl(m)
2020 format(//,35hirep spr sp  g  i  l ll l0   x0/xl1,7x,2he0,6x,
     1 3hxll,6x,3hell,6x,3htll,7x,2hxl,7x,2hel,7x,2htl,3x,
     2 17hbll/bl   eps    n,
     3 /1x,i3,2(3x,a1),5i3,5f9.1,27x,f9.4)
 call rayup(l2,l1,ls(j,m),j,m,ii,xsp(j,m),es(j,m),xlls,
     1 ells,tlls,dmy1,dmy2,dmy3,bl(m))
 ds=(xlls-xsp(j,m))*sgn
 go to 2023
c
c trace shotpoint rays, irep.gt.1
c
2021 if((jjoff.eq.0.and.j.eq.j1.and.i.eq.1).or.
     1(jjoff.eq.0.and.j.eq.j2.and.i.eq.2)) go to 12021
 if(k11.eq.0) go to 2031
12021 if (isplt.eq.2) go to 2022
 if(prs2(j,m,l1,i).ne.0..or.ers2(j,m,l1,i).ne.0.) go to 12020
 prs2(j,m,l1,i)=xsp(j,m)
 ers2(j,m,l1,i)=es(j,m)
12020 if(itrace.ne.0.and.jprt.ne.ibl)write(20,12022) irep,idspr(m),
     1 idsp(j,m),iz,ii,l2,l2,ls(j,m),xsp(j,m),es(j,m),ps2(j,m,i),
     2 es2(j,m,i),prs2(j,m,l1,i),ers2(j,m,l1,i),trs2(j,m,i),bl(m)
12022 format(/'irep spr sp  g  i  l ll l0   x0/xl1',7x,'e0',6x,'xl1'
     1 ,6x,'el1',6x,'tl1',7x,'xl',7x,'el',7x,'tl   bll/bl   eps'
     2 /1x,i3,2(3x,a1),5i3,4f9.1,9x,3f9.1,f9.4)
 call rayup(l2,l2,ls(j,m),j,m,ii,xsp(j,m),es(j,m),
     1 ps2(j,m,i),es2(j,m,i),dmy0,prs2(j,m,l1,i),ers2(j,m,l1,i),
     2 trs2(j,m,i),bl(m))
 go to 2023
2022 prs2(j,m,l1,i)=xsp(j,m)
 ers2(j,m,l1,i)=ers(j,m,l1)
c
c re-entry pt for outlying shotpoints
c
2023 if(k11.eq.0) go to 2031
 if(krs2(jj,m,l1,i).eq.ibee) krs2(jj,m,l1,i)=ibl
 do 2030 k=k11,k22
 if (lg(k,jj,m).ne.l2) go to 2030
c
c trace geophone rays, irep=1
c
 if (irep.gt.1) go to 2026
 if (tg2(k,m,i).ne.0.0) go to 2025
 pg2(k,m,i)=xg(k,m)-xvg(k)*sgn
 tg2(k,m,i)=tvg(k)
 if(itrace.ne.0.and.jprt.ne.ibl)write(20,2020) irep,idspr(m),
     1 idsp(jj,m),k,i,l2,l1,i1,xg(k,m),eg(k,m),pg2(k,m,i),eg2(k,m,i),
     2 tg2(k,m,i),bl(m)
```

```
 call rayup(l2,l1,1,j,m,i,xg(k,m),eg(k,m),pg2(k,m,i),eg2(k,m,i),
     1 tg2(k,m,i),dmy1,dmy2,dmy3,bl(m))
2025 dg=(xg(k,m)-pg2(k,m,i))*sgn
 ta(k,jj,m)=tr(k,jj,m)-tlls-tg2(k,m,i)
 dsg(k)=abs(d(k,j,m))-ds-dg
 go to 2030
c
c trace geophone rays, irep.gt.1
c
2026 if (trg2(k,i).ne.0.0) go to 2028
 prg2(k,i)=prg(k,jj,m)
 erg2(k,i)=erg(k,jj,m)
 trg2(k,i)=trg(k,jj,m)
 if(itrace.ne.0.and.jprt.ne.ibl)write(20,12022) irep,idspr(m),
     1 idsp(jj,m),k,i,l2,l2,i1,xg(k,m),eg(k,m),pg2(k,m,i),eg2(k,m,i),
     2 prg2(k,i),erg2(k,i),trg2(k,i),bl(m)
 call rayup(l2,l2,1,j,m,i,xg(k,m),eg(k,m),pg2(k,m,i),eg2(k,m,i),
     1 dmy0,prg2(k,i),erg2(k,i),trg2(k,i),bl(m))
 if(isplt.ne.2) go to 2028
 if(abs((prg2(k,i)-xsp(j,m))/(erg2(k,i)-es(j,m))).gt.100.)
     1 go to 2028
 dmy0=big
 prg2(k,i)=(xsp(j,m)+2.*prg2(k,i))/3.
 if(itrace.ne.0.and.jprt.ne.ibl) write(20,12022) irep,idspr(m),
     1 idsp(jj,m),k,i,l2,l2,i1,xg(k,m),eg(k,m),pg2(k,m,i),eg2(k,m,i),
     2 prg2(k,i),erg2(k,i),trg2(k,i),bl(m)
 call rayup(l2,l2,1,j,m,i,xg(k,m),eg(k,m),pg2(k,m,i),eg2(k,m,i)
     1 ,dmy0,prg2(k,i),erg2(k,i),trg2(k,i),bl(m))
2028 prg(k,jj,m)=prg2(k,i)
 erg(k,jj,m)=erg2(k,i)
 trg(k,jj,m)=trg2(k,i)
2030 continue
c
c irep=1  or  irep.gt.1
c
2031 if(irep.gt.1) go to 2035
c
c irep=1
c
c regression of end and outlying sp times to get intercept t when
c irep=1.  if jjoff=0 outlying sp times are tied to end sp times.
c
 kt1=k11
 kt2=k22
 jt=jj
 s1=0.
 s2=0.
 pt=0.
 t=0.
12033 if(kt1.eq.0) go to 12037
 do 12036 k=kt1,kt2
 if(lg(k,jt,m).ne.12) go to 12036
 if(jt.eq.jj) go to 12035
 if(tg2(k,m,i).ne.0.) go to 12034
```

```
 pg2(k,m,i)=xg(k,m)-xvg(k)*sgn
 tg2(k,m,i)=tvg(k)
 if(itrace.ne.0.and.jprt.ne.ibl)write(20,2020) irep,idspr(m),
     1 idsp(jt,m),k,i,l2,l1,i1,xg(k,m),eg(k,m),pg2(k,m,i),eg2(k,m,i),
     2 tg2(k,m,i),bl(m)
 call rayup(l2,l1,1,jt,m,i,xg(k,m),eg(k,m),pg2(k,m,i),eg2(k,m,i),
     1 tg2(k,m,i),dmy1,dmy2,dmy3,bl(m))
12034 dg=(xg(k,m)-pg2(k,m,i))*sgn
 dsg(k)=abs(d(k,j,m))-ds-dg
 ta(k,jt,m)=tr(k,jt,m)-tlls-tg2(k,m,i)
12035 s1=s1+dsg(k)
 s2=s2+ta(k,jt,m)
 pt=pt+1.
12036 continue
12037 if(jjoff.ne.0) go to 2032
 if(i.eq.2) go to 12040
 if(jj.ne.j1) go to 2032
 jt=jt-1
 if(jt.lt.1) go to 12042
 krs2(jt,m,l1,1)=iast
 call kends(l2,m,jt,kr(jt,m),kn,kt1,kt2)
 go to 12033
12040 if(jj.ne.j2) go to 2032
 jt=jt+1
 if(jt.gt.jn) go to 12042
 krs2(jt,m,l1,2)=iast
 call kends(l2,m,jt,1,kl(jt,m),kt1,kt2)
 go to 12033
c
12042 if(pt.eq.0.) go to 2040
 if(pt.eq.1.) go to 12055
 xbar=s1/pt
 tbar=s2/pt
 s1=0.
 s2=0.
 kt1=k11
 kt2=k22
 jt=jj
12043 if(kt1.eq.0) go to 12046
 do 12045 k=kt1,kt2
 if(lg(k,jt,m).ne.l2) go to 12045
 xd=dsg(k)-xbar
 s1=s1+xd*ta(k,jt,m)
 s2=s2+xd**2
12045 continue
12046 if(i.eq.2) go to 12050
 jt=jt-1
 if(jt.lt.1) go to 12052
 call kends(l2,m,jt,kr(jt,m),kn,kt1,kt2)
 go to 12043
12050 jt=jt+1
 if(jt.gt.jn) go to 12052
 call kends(l2,m,jt,1,kl(jt,m),kt1,kt2)
 go to 12043
```

```
c
12052 t=(tbar-xbar*s1/s2)/2.
 go to 12060
c
12055 t=(s2-s1/vha(m,l2))/2.
 go to 12060
c
c irep=1
c
c compute horiz time and first approx depths
c
2032 if(k11.eq.0) go to 2040
 call regres(k11,k22,jj,m,l2,v,t,pt,0)
 if(pt.eq.0.) go to 2040
12060 z=t*vocosg
 if(z.lt.0.) z=0.
 ts=z*vocosg/vv**2
 trs2(jj,m,i)=tlls+ts
 ers2(jj,m,l1,i)=ells-z
 prs2(jj,m,l1,i)=xlls+z*tansg*sgn
2033 zrsp(jj)=z
 do 2034 k=k11,k22
 if (lg(k,jj,m).ne.l2) go to 2034
 edif=slope*(pg2(k,m,i)-prs2(jj,m,l1,i))
 kcall=k
 call htime(kcall,j,m,prs2(jj,m,l1,i),0.,pg2(k,m,i),edif,hv,th)
 z=(ta(k,jj,m)-ts-th)*vocosg
 if (z.lt.0.0) z=0.0
 zrg(k,jj)=z
 tg=z*vocosg/vv**2
 trg(k,jj,m)=tg2(k,m,i)+tg
 erg(k,jj,m)=eg2(k,m,i)-z
 prg(k,jj,m)=pg2(k,m,i)-z*tansg*sgn
2034 continue
 go to 2040
c
c irep.gt.1
c
c compute horiz time, compute total ray time, and adjust depths
c to make computed time agree with measured time
c
2035 if(k11.eq.0) go to 2040
 sum1=0.
 pts1=0.
 if(isplt.eq.2) go to 3035
 p1=prs2(j,m,l1,i)
 e1=ers2(j,m,l1,i)
 go to 3135
3035 p1=xsp(j,m)
 e1=es(j,m)
 krs2(jj,m,l1,i)=ibee
3135 do 2037 k=k11,k22
 if (lg(k,jj,m).ne.l2) go to 2037
 krg(k,jj,m)=ibl
```

```
 p2=prg2(k,i)
 e2=erg2(k,i)
3036 kcall=k
 call htime(kcall,j,m,p1,e1,p2,e2,hv,th)
 tcor=tr(k,jj,m)-trs2(jj,m,i)-th-trg2(k,i)
 if(isplt.eq.0) tcor=tcor/2.
 tcall=tcor
 if(abs(tcor).le.tlim) go to 2036
 tcall=sign(tlim,tcor)
 krg(k,jj,m)=iques
2036 call raycor(pg2(k,m,i),p2,eg2(k,m,i),e2,vv,hv,tcall)
 prg(k,jj,m)=p2
 erg(k,jj,m)=e2
 trg(k,jj,m)=trg2(k,i)+tcall
 pts1=pts1+1.0
 sum1=sum1+tcor
 if (ixrep.gt.2.or.l2.eq.2) go to 2037
 gtc(k,m)=gtc(k,m)+tcor
 gpt(k,m)=gpt(k,m)+1.0
2037 continue
 if (pts1.eq.0..or.isplt.eq.2) go to 2040
 if (ixrep.gt.2.or.l2.eq.2) go to 2038
 sptc(jj,m)=sptc(jj,m)+sum1
 sppt(jj,m)=sppt(jj,m)+pts1
2038 tcor=sum1/pts1
 if(krs2(jj,m,l1,i).eq.iques) krs2(jj,m,l1,i)=ibl
 if(abs(tcor).le.tlim) go to 2039
 if(krs2(jj,m,l1,i).eq.ibl) krs2(jj,m,l1,i)=iques
 tcor=sign(tlim,tcor)
2039 call raycor(ps2(j,m,i),p1,es2(j,m,i),e1,vv,hv,tcor)
 trs2(jj,m,i)=trs2(jj,m,i)+tcor
 prs2(jj,m,l1,i)=p1
 ers2(jj,m,l1,i)=e1
c
c test for doing outlying sps left of spreads, right-going rays
c
2040 if (i.ne.1) go to 2070
 if (jjoff.ne.0.or.jj.gt.j1) go to 2046
2041 jj=jj-1
 if (jj.gt.0) go to 2042
 if(irep.eq.1.or.j1.eq.1) go to 2046
 sum1=0.
 sum2=0.
 sum3=0.
 pts1=0.
 jjj=0
 do 3041 jj=1,j1
 if(ers2(jj,m,l1,i).eq.0..or.krs2(jj,m,l1,i).eq.iques) go to 3041
 sum1=sum1+trs2(jj,m,i)
 sum2=sum2+prs2(jj,m,l1,i)
 sum3=sum3+ers2(jj,m,l1,i)
 pts1=pts1+1.
 if(jjj.eq.0) jjj=jj
3041 continue
```

```
 if(pts1.eq.0.) go to 2046
 tbar=sum1/pts1
 pbar=sum2/pts1
 ebar=sum3/pts1
 do 4041 jj=jjj,j1
 trs2(jj,m,i)=tbar
 prs2(jj,m,l1,i)=pbar
 ers2(jj,m,l1,i)=ebar
4041 continue
 go to 2046
2042 call kends(l2,m,jj,kr(jj,m),kn,k11,k22)
 if (k11.eq.0) go to 2041
2043 trs2(jj,m,i)=trs2(j,m,i)
 prs2(jj,m,l1,i)=prs2(j,m,l1,i)
 ers2(jj,m,l1,i)=ers2(j,m,l1,i)
 go to 2023
c
c initialize for left-going rays
c
2046 i=2
 ii=1
 sgn=-1.0
 if(kl(j,m).eq.0) go to 2090
 call kends(l2,m,j,1,kl(j,m),k11,k22)
 go to 2019
c
c test for doing outlying sps right of spreads, left-going rays
c
2070 if (jjoff.ne.0.or.jj.lt.j2) go to 2090
2072 jj=jj+1
 if (jj.le.jn) go to 2074
 if(irep.eq.1.or.j2.eq.jn) go to 2090
 sum1=0.
 sum2=0.
 sum3=0.
 pts1=0.
 do 2073 jj=j2,jn
 if(ers2(jj,m,l1,i).eq.0..or.krs2(jj,m,l1,i).eq.iques) go to 2073
 sum1=sum1+trs2(jj,m,i)
 sum2=sum2+prs2(jj,m,l1,i)
 sum3=sum3+ers2(jj,m,l1,i)
 pts1=pts1+1.
 jjj=jj
2073 continue
 if(pts1.eq.0.) go to 2090
 tbar=sum1/pts1
 pbar=sum2/pts1
 ebar=sum3/pts1
 do 3073 jj=j2,jjj
 trs2(jj,m,i)=tbar
 prs2(jj,m,l1,i)=pbar
 ers2(jj,m,l1,i)=ebar
3073 continue
 go to 2090
```

```
2074 call kends(l2,m,jj,1,kl(jj,m),k11,k22)
 if (k11.eq.0) go to 2072
 go to 2043
c
2090 j=j+1
 if (j.le.j2) go to 2012
 if (irep.eq.1)call admig(l1,m,bl(m))
 m=m+1
 if (m.le.nm) go to 2003
 if (iflag.ne.0) then
        enter2='y'
        return
        endif
c
c for flag.ne.0, return to part 2
c
c end of sp loop at 2090, end of spread loop at 2090 + 3.
c
c compute avg coords(p1,e1),(p2,e2) in adjacent intervals between geos,
c and then interpolate to find smoothed elev pts (erp) at geo pos (xg)
c between the two intervals.  then compute layer vert travel time (trp)
c
 do 2096 m=1,nm
 jn=nj(m)
 kn=nk(m)
 do 2094 j=1,jn
 trs(j,m,l1)=0.0
 ers(j,m,l1)=0.0
2094 continue
 do 2095 k=1,kn
 trp(k,m,l1)=0.0
 erp(k,m,l1)=0.0
2095 continue
2096 continue
c
 m=0
 x1=2.*xg(1,1)-xg(2,1)
2100 m=m+1
 if (m.gt.nm) go to 2114
 m1=m
 kn=nk(m)
 k=1
2102 kk=k
 x2=xg(k,m)
 if (x2.le.x1) go to 2104
 call avg(x1,x2,l2,p1,e1)
 if (e1.ne.0.0) go to 2105
c
c condition in which (p1,e1) is not yet defined
c
 x1=x2
2104 k=k+1
 if (k.gt.kn) go to 2100
 go to 2102
```

```
c
c condition in which (p1,e1) is defined and (p2,e2) is sought
c
2105 erp(kk,m1,l1)=e1
 trp(kk,m1,l1)=(erp(kk,m1,l1)-erp(kk,m1,l1))/vva(m1,l1)
2106 x1=x2
2108 k=k+1
 if (k.le.kn) go to 2109
 m=m+1
 if (m.gt.nm) go to 2113
 kn=nk(m)
 k=1
2109 x2=xg(k,m)
 if (x2.le.x1) go to 2108
 call avg(x1,x2,l2,p2,e2)
 if (e2.eq.0.0) go to 2106
2110 erp(kk,m1,l1)=terp(p1,e1,p2,e2,xg(kk,m1))
 trp(kk,m1,l1)=(erp(kk,m1,l1)-erp(kk,m1,l1))/vva(m1,l1)
 kk=kk+1
 if (m.ne.m1) go to 2111
 if (kk.eq.k) go to 2112
 go to 2110
2111 if (kk.le.nk(m1)) go to 2110
 m1=m
 kk=k
 erp(kk,m1,l1)=terp(p1,e1,p2,e2,xg(kk,m1))
 trp(kk,m1,l1)=(erp(kk,m1,l1)-erp(kk,m1,l1))/vva(m1,l1)
2112 p1=p2
 e1=e2
 go to 2106
2113 x2=2.*xg(kn,nm)-xg(kn-1,nm)
 call avg(x1,x2,l2,p2,e2)
 if (e2.eq.0.0) go to 2114
 erp(kn,nm,l1)=terp(p1,e1,p2,e2,xg(kn,nm))
 trp(kn,nm,l1)=(erp(kn,m1,l1)-erp(kn,m1,l1))/vva(m1,l1)
2114 call fillin(l1)
c l1 set to 0 in fillin if no pts are defined for layer l1
 if(l1.eq.0)then
        begin='y'
        return
        endif
c
c repeat if irep=1 or irep=2
c
 if (ixrep.le.3) go to 2190
 irep=irep+1
 go to 2002
c
2190 l2=l2+1
 if (l2.le.nl) go to 2001
c
c end of layer loop at 2190
c
c final filter -- trim-up time adjust at base of layer 1
```

```
c
 if (irep.eq.4.or.ixit.le.3) go to 2200
 irep=4
c
 do 2196 m=1,nm
 jn=nj(m)
 kn=nk(m)
 do 2192 j=1,jn
 if (sppt(j,m).eq.0.0) go to 2192
 tcor=sptc(j,m)/sppt(j,m)
 ers(j,m,1)=ers(j,m,1)-tcor*vva(m,1)
 if (ers(j,m,1).gt.esp(j,m)) ers(j,m,1)=esp(j,m)
2192 continue
 do 2194 k=1,kn
 if (gpt(k,m).eq.0.0) go to 2194
 tcor=gtc(k,m)/gpt(k,m)
 erp(k,m,1)=erp(k,m,1)-tcor*vva(m,1)
 if (erp(k,m,1).gt.eg(k,m)) erp(k,m,1)=eg(k,m)
2194 continue
2196 continue
 go to 2000
c
c print results -- depths computed from refraction arrivals
c
2200 if(jprt.eq.ibl) go to 12200
 do 2222 m=1,nm
 write(20,57) ident
 jn=nj(m)
 write(20,2201) idspr(m),(idsp(j,m),j=1,jn)
2201 format (//,8h spread ,a1,3x,32hray end points beneath geophones
     1 /'  geo',14x,7(8x,'sp ',a1))
 write(20,2202) (ibl,j=1,jn)
2202 format('  ---',17x,7(a2,'--------1-'))
c
 kn=nk(m)
c
 do 2210 k=1,kn
 write(20,2203) k,(prg(k,j,m),lg(k,j,m),j=1,jn)
2203 format (1h ,i3,15x,3hpos,7(f9.1,i2,1x))
 write(20,2205) (erg(k,j,m),krg(k,j,m),j=1,jn)
2205 format (1h ,17x,4helev,7(f9.1,1x,a1,1x))
 write(20,2207)
2207 format (1x)
2210 continue
c
 write(20,2211)
2211 format (' ray end points beneath shotpoints')
c
 do 2220 l2=2,nl
 l=l2-1
 write(20,2213) l2,(prs2(j,m,l,1),j=1,jn)
2213 format (/,2hl=,i1,4x,5hright,6x,3hpos,f9.1,6f12.1)
 write(20,2205) (ers2(j,m,l,1),krs2(j,m,l,1),j=1,jn)
 write(20,2217) l2,(prs2(j,m,l,2),j=1,jn)
```

```
2217 format (/,2hl=,i1,4x,4hleft,7x,3hpos,f9.1,6f12.1)
 write(20,2205) (ers2(j,m,l,2),krs2(j,m,l,2),j=1,jn)
2220 continue
2222 continue
c
c print results -- interp-extrap pts at sps and geos
c
 do 2256 m=1,nm
 if (jjoff.eq.0) go to 2225
 j1=1
 go to 2226
2225 j1=ja(m)
2226 if (jjoff.eq.0) go to 2227
 j2=nj(m)
 go to 2228
2227 j2=jb(m)
2228 write(20,57) ident
 write(20,2231) idspr(m),(l,l=2,nl)
2231 format (//,8h spread ,a1,3x,28hsmoothed position of layers ,
     1 32hbeneath shotpoints and geophones/,24x,4(11x,'layer',i2))
c
 write(20,2233) (ibl,l=2,nl)
2233 format('  sp   position  surf elev',4(a1,5x,'depth  elev '))
 write(20,2234) (ibl,l=2,nl)
2234 format('  ---  --------  ---------',4(a4,'--------------'))
c
 do 2246 j=j1,j2
 do 2240 l=1,ln
 zsg(l)=0.0
 if (ers(j,m,l).ne.0.0) zsg(l)=esp(j,m)-ers(j,m,l)
2240 continue
 write(20,2243) idsp(j,m),xsp(j,m),esp(j,m),(zsg(l),ers(j,m,l),
     1 l=1,ln)
2243 format(3x,a1,2f11.1,4(2x,2f8.1))
2246 continue
c
 write(20,2249)
2249 format('  geo'/'  ---')
 kn=nk(m)
 do 2256 k=1,kn
 do 2252 l=1,ln
 if (erp(k,m,l).ne.0.0) zsg(l)=eg(k,m)-erp(k,m,l)
2252 continue
 write(20,2255) k,xg(k,m),eg(k,m),(zsg(l),erp(k,m,l),l=1,ln)
2255 format (1x,i3,2f11.1,4(2x,2f8.1))
2256 continue
 do 2260 m=1,nm
 write(20,2257) m,(l,l=1,nl)
2257 format(/' velocities used, spread',i2/7x,5(11x,'layer',i2))
 write(20,2258) (vva(m,l),l=1,ln)
 write(20,2259) (vha(m,l),l=2,nl)
2258 format(/8x,'vertical',3pf9.0,4f18.0)
2259 format(6x,'horizontal',9x,3p4f18.0)
2260 continue
```

```
12200 ity=ibl
 if(jtrl.eq.ibl) go to 3010
c
c table of ray end points option from terminal
c
 write(*,12201)
12201 format(//' Table of ray end pts? (t to type, <cr> to suppress):'
     1/)
 read (*,10006)ity
 if(ity.eq.ibl) go to 12223
 do 12222 m=1,nm
 jn=nj(m)
 write(*,12202) idspr(m),(idsp(j,m),j=1,jn)
12202 format(/' spread ',a1,3x,'ray end pts beneath geophones'//2x,'g'
     1 ,3x,7(5x,'sp ',a1))
        write(*,12203) (ibl,j=1,jn)
12203 format('  -',5x,7(a2,'------l'))
 kn=nk(m)
 do 12210 k=1,kn
 write(*,12204)k,(prg(k,j,m),lg(k,j,m),j=1,jn)
12204 format(/1x,i2,4x,'x',7(f7.0,i2))
 write(*,12205)(erg(k,j,m),krg(k,j,m),j=1,jn)
12205 format(6x,'el',1x,7(f7.1,1x,a1))
12210 continue
 write(*,2207)
 write(*,2211)
 do 12220 l2=2,nl
 l=l2-1
 write(*,12213)l2,(prs2(j,m,l,1),j=1,jn)
12213 format(/' l',i1,' rt x',7(f7.0,2x))
 write(*,12205)(ers2(j,m,l,1),krs2(j,m,l,1),j=1,jn)
 write(*,12217)l2,(prs2(j,m,l,2),j=1,jn)
12217 format(/' l',i1,' lf x',7(f7.0,2x))
 write(*,12205)(ers2(j,m,l,2),krs2(j,m,l,2),j=1,jn)
12220 continue
12222 continue
c
c depth beneath sps-geos option from terminal
c
12223 write(*,12224)
12224 format(//
     1' Depth beneath sps-geos? (t to type, <cr> to suppress):'/)
 ity=ibl
 read (*,10006)ity
 if(ity.eq.ibl) go to 12254
c
 do 12256 m=1,nm
 if(jjoff.eq.0) go to 12225
 j1=1
 go to 12226
12225 j1=ja(m)
12226 if(jjoff.eq.0) go to 12227
 j2=nj(m)
 go to 12228
```

```
12227 j2=jb(m)
12228 write(*,12231)idspr(m),(l,l=2,nl)
12231 format(/' spread ',a1,3x,'depths beneath sps & geos'//16x
     1 ,4(7x,'layer',i2))
 write(*,12233) (ibl,l=2,nl)
12233 format(/1x,'sp',4x,'x',6x,'el',4(a3,'depth  elev'))
 write(*,12234) (ibl,l=2,nl)
12234 format(' --   ---   ----',4(a2,'------------'))
 do 12246 j=j1,j2
 do 12240 l=1,ln
 if(ers(j,m,l).ne.0.)zsg(l)=esp(j,m)-ers(j,m,l)
12240 continue
 write(*,12243)idsp(j,m),xsp(j,m),esp(j,m),(zsg(l),ers(j,m,l),
     1 l=1,ln)
12243 format(2x,a1,f6.0,9f7.1)
12246 continue
 write(*,12249)
12249 format(/1x,'geo'/1x,'---')
 kn=nk(m)
 do 12256 k=1,kn
 do 12252 l=1,ln
 if(erp(k,m,l).ne.0.)zsg(l)=eg(k,m)-erp(k,m,l)
12252 continue
 write(*,12255)k,xg(k,m),eg(k,m),(zsg(l),erp(k,m,l),l=1,ln)
12255 format(1x,i2,f6.0,9f7.1)
12256 continue
12254 do 12260 m=1,nm
 write(*,12257) m,(l,l=1,nl)
12257 format(/' velocities used, spread',i2/2x,5(7x,'layer',i2))
 write(*,12258)(vva(m,l),l=1,ln)
 write(*,12259)(vha(m,l),l=2,nl)
12258 format(1x,'vertical',3pf7.0,4f14.0)
12259 format(1x,'horizontal',5x,3p4f14.0)
12260 continue
c
c depth plot option from terminal
c
3000 ity=ibl
 write(*,2207)
 write(*,3001)
3001 format(/' Depth plot? (t to type, <cr> to suppress:)'/)
 read(*,10006) ity
3010 if(jprt.ne.ibl.or.ity.ne.ibl) call plot(2)
        begin='y'
        return
c
9992 if(jprt.ne.ibl)write(20,9993) l1,l2
 if(jtrl.ne.ibl)write(*,9993) l1,l2
9993 format (/1x,25hvelocity inversion, layer,i2,10h and layer,i2,
     1 19h computation halted)
        begin='y'
        return
```

```
57      format(///,20a4)
1027    format(//,45hintermediate results of ray tracing for layer,i2)
10006   format(a1)
 end
```

```
      subroutine initial
c ********************** INITIAL.FOR ********************************
c
c Program SIPT1.FOR -- version 1.0 -- USGS, Hartford, CT -- 08-15-86
c
c Micro-computer implementation of sipt22.fortran (on Denver PRIME).
c Compiled on an IBM PC using the IBM Professional FORTRAN compiler
c (version 1.0).
c Program consists of 9 modules: MAIN.FOR, PART2.FOR, PART3.FOR,
c INITIAL.FOR, S1.FOR, S2.FOR, S3.FOR, S4.FOR and S5.FOR.
c
c ***********************************************************************
c
c explicit variable specification
c
 character*2 jchar
 character*32 din,dout,blank
      character*1 ifprint,jtrl,jprt,jout,idspr,idsp,krs2,krg,ip,it,
     1       ity,idtest,ibl,is,isq,iast,iplus,idash,icoln,idee,iques,
     2       icee,itee,ipee,ibee,ix,il,jvel
       character*1 begin,br2002
        character*1 enter2
 character*4 ident,iend
c
c common block assignments
c
 common ibl,iques,ip,it,icoln,iplus,isq,idash,is,iast,idee,il,
     1         ident
 common/ibm1/nl,ln,p,tscale,xscale,xsco2,escale,xlim1,xlim2,
     1              iplot
 common/blk0/lg
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,ity
 common/blk2/xg,erp,slope,idip
 common/blk3/ta,tr,dsg
 common/blk4/vva,vha
 common/blk5/idspr,idsp
 common/ibm3/kl,kr,d
 common/blk6/trp,jjoff
 common/blk7/ja,jb,trs,ers,xsp,esp,ls
 common/blk8/prg,erg,prs2,ers2,zrsp,zrg
 common/ibm4/krg,krs2
 common/blk9/eg,es
 common/blk10/blim,itrace,tansg
 common/blk11/vreg,preg
 common/blk12/vhob,phob
      common/real1/a,avvdd,big,bl,diff,dmy0,dmy1,dmy2,dmy3,ebar,edat1,
     1   edat2,edg,edsp,eg2,eintg,eref,erg2,erx,es2,gpt,gtc,pg2,prg2,
     2   prp,ps2,pts,pts1,pts2,pts3,sppt,sptc,sum1,sum2,sum3,tcg,
     3   tfudge,tg2,tlim,trg,trg2,trs2,tuh,tvg,tvs,vdd,vv,xbar,xcg,
     4   xdat1,xdat2,xintg,xlast,xref,xs,xshift,xtie,xvg,yg,ysp,zsg,
     5   zsp
      common/real2/b,denex,dg2,edif,eint,hv,pbar,prsh,pt,rad,rads,ray,
     1   t,tbar,tc,th,ts,v,vocosg,vvcosg,x1,x2,xc,xint,xtru,z,ztan,ztru
```

```
      common/real3/dg,ds,dz,e1,e2,ells,hv2,p1,p2,s1,s2,sgn,tcall,tcor,
     1   tg,tlls,v1,xd,xlls
      common/char1/blank,din,dout,ibee,icee,idtest,iend,ifprint,ipee,
     1   itee,ix,jout
      common/char2/begin,br2002
      common/char3/enter2
      common/integ1/i,i1,iexit,iopen,iover,irep,ixit,ixtrue,iz,j,j1,
     1   jc,jexit,jn,jopen,jr,k,kn,ktest,l,l1,l2,lr,m,mj,mk,ml,ml1,mm,
     2   nixit,nv
      common/integ2/icall,iflag,ii,j2,jcall,jj,jjj,jjtie,k1,k11,k2,
     1   k22,kcall,kk,kt1,kt2,lcall,mcall
      common/integ3/isplt,ixrep,jt,l1,m1,none
c
c arrays
c
 dimension ident(20),ip(103),it(53),
     1   il(5),vreg(5),vhob(5),preg(5),phob(5),zsg(4),
     2   idspr(5),nj(5),nk(5),xshift(5),ja(5),jb(5),bl(5),
     3   vva(5,5),vha(5,5),
     4   tvs(7),avvdd(7),zrsp(7),idsp(7,5),esp(7,5),
     5   xsp(7,5),ysp(7,5),zsp(7,5),tuh(7,5),edsp(7,5),
     6   tfudge(7,5),es(7,5),kl(7,5),kr(7,5),sptc(7,5),sppt(7,5),
     7   ls(7,5),zrg(48,7),
     8   ps2(7,5,2),es2(7,5,2),trs2(7,5,2),
     9   ers(7,5,4),trs(7,5,4),
     1   ers2(7,5,4,2),prs2(7,5,4,2),krs2(7,5,4,2),
     1   tvg(48),xvg(48),dsg(48),prp(48),xs(48),erx(48),
     2   prg2(48,2),erg2(48,2),trg2(48,2),
     3   tcg(48,2),xcg(48,2),xintg(48,2),eintg(48,2),
     4   edg(48,5),eg(48,5),xg(48,5),yg(48,5),
     5   gtc(48,5),gpt(48,5),
     6   pg2(48,5,2),eg2(48,5,2),tg2(48,5,2),
     7   erp(48,5,4),trp(48,5,4),ta(48,7,5),
     8   tr(48,7,5),lg(48,7,5),d(48,7,5),vdd(48,7,5),p(48,7,5),
     9   prg(48,7,5),erg(48,7,5),trg(48,7,5),krg(48,7,5)
c
c *********** beginning of initializations ******************************
c
10008 do 2 k=1,mk
 tvg(k)=0.
 xvg(k)=0.
 dsg(k)=0.
 prp(k)=0.
 xs(k)=0.
 do 2 lr=1,2
 prg2(k,lr)=0.
 erg2(k,lr)=0.
 trg2(k,lr)=0.
 tcg(k,lr)=0.
 xcg(k,lr)=0.
 xintg(k,lr)=0.
2 eintg(k,lr)=0.
 do 5 j=1,mj
 avvdd(j)=0.
```

```
 do 5 m=1,mm
 idsp(j,m)=ibl
 sptc(j,m)=0.0
 sppt(j,m)=0.0
 ls(j,m)=1
 esp(j,m)=0.
 xsp(j,m)=0.
 ysp(j,m)=0.
 zsp(j,m)=0.
 tuh(j,m)=0.
 edsp(j,m)=0.
 tfudge(j,m)=0.
 es(j,m)=0.
 kl(j,m)=0
 kr(j,m)=0
 do 3 lr=1,2
 ps2(j,m,lr)=0.
 es2(j,m,lr)=0.
3 trs2(j,m,lr)=0.
 do 4 l=1,ml1
 ers(j,m,l)=0.0
 trs(j,m,l)=0.0
 do 4 lr=1,2
 ers2(j,m,l,lr)=0.0
 prs2(j,m,l,lr)=0.0
 krs2(j,m,l,lr)=ibl
4 continue
 do 5 k=1,mk
 prg(k,j,m)=0.0
 erg(k,j,m)=0.0
 trg(k,j,m)=0.0
 krg(k,j,m)=ibl
 vdd(k,j,m)=0.0
 d(k,j,m)=0.0
 lg(k,j,m)=0
 tr(k,j,m)=0.
 ta(k,j,m)=0.
 p(k,j,m)=0.
5 continue
 do 6 m=1,mm
 do 6 l=1,ml
 vva(m,l)=0.0
 vha(m,l)=0.0
6 continue
 do 8 m=1,mm
 ja(m)=0
 jb(m)=0
 nj(m)=0
 nk(m)=0
 xshift(m)=0.
 do 8 k=1,mk
 gtc(k,m)=0.0
 gpt(k,m)=0.0
 do 7 lr=1,2
```

```
 pg2(k,m,lr)=0.
 eg2(k,m,lr)=0.
7 tg2(k,m,lr)=0.
 do 8 l=1,ml1
 erp(k,m,l)=0.0
 trp(k,m,l)=0.0
8 continue
 ixtrue=0
 irep=0
 dmy0=0.
 dmy1=0.
 dmy2=0.
 dmy3=0.
       return
 end
```

```
c ************************ S1.FOR ************************************
c
c Program SIPT1.FOR -- version 1.0 -- USGS, Hartford, CT -- 08-15-86
c
c Micro-computer implementation of sipt22.fortran (on Denver PRIME).
c Compiled on an IBM PC using the IBM Professional FORTRAN compiler
c (version 1.0).
c Program consists of 9 modules: MAIN.FOR, PART2.FOR, PART3.FOR,
c INITIAL.FOR, S1.FOR, S2.FOR, S3.FOR, S4.FOR and S5.FOR.
c
c ************ beginning of subroutines *********************************
c
c ***** subroutine TIE (#1).  Written 09/06/81  1445.3rew **************
c makes tie correction for outlying sp jj to next inner sp j
 subroutine tie(l2,m,j,jj,k11,k22,kt1,kt2,kn,i,jjtie)
 character*1 ibl,iques,ip,it,icoln,iplus,isq,idash,is,iast,idee
     1 ,il,idspr,idsp
 common ibl,iques,ip,it,icoln,iplus,isq,idash,is,iast,idee,il
 common/ibm1/nl
 common/blk0/lg
 common/blk3/ta,tr,dsg
 common/blk4/vva,vha
 common/blk5/idspr,idsp
 common/ibm3/kl,kr,d
 dimension lg(48,7,5),ta(48,7,5),dsg(48),tr(48,7,5),d(48,7,5),
     1 ip(103),it(53),il(5),idspr(5),idsp(7,5),kl(7,5),kr(7,5),vva(5,5)
     2 ,vha(5,5)
c
 call kends(l2,m,j,kt1,kt2,kt11,kt22)
 if (kt11.eq.0) go to 99
 do 10 k=1,kn
10 dsg(k)=abs(d(k,j,m))
 sum1=0.
 pts1=0.
 k1=max0(k11,kt11)
 k2=min0(k22,kt22)
 if(k2.lt.k1) go to 50
c
c overlap between j and jj arrivals
c
20 do 30 k=k1,k2
 if (lg(k,jj,m).ne.l2) go to 30
 sum1=sum1+tr(k,jj,m)-tr(k,j,m)
 pts1=pts1+1.
30 continue
 if (pts1.eq.0.) go to 50
 sum1=sum1/pts1
35 do 40 k=k11,k22
 if (lg(k,jj,m).eq.l2) tr(k,jj,m)=tr(k,jj,m)-sum1
40 continue
 if(jjtie.eq.0) go to 60
 if(i.eq.2) go to 44
 kk=k22+1
 if(kk.gt.kn) go to 99
```

```
 do 42 k=kk,kn
 if(tr(k,jj,m).ne.0) tr(k,jj,m)=tr(k,jj,m)-sum1
42 continue
 go to 99
44 kk=k11-1
 if(kk.lt.1) go to 99
 do 46 k=1,kk
 if(tr(k,jj,m).ne.0) tr(k,jj,m)=tr(k,jj,m)-sum1
46 continue
        go to 99
c
c gap between j and jj arrivals
c
50 call regres(kt11,kt22,j,m,l2,vj,tj,pt,1)
 if(pt.eq.0.) go to 99
 call regres(k11,k22,jj,m,l2,vjj,tjj,pt,1)
 if(pt.eq.0.) go to 99
 if(i.eq.2) go to 54
 dmid=(dsg(k22)+dsg(kt11))/2
52 sum1=(tjj+tjj+dmid/vjj)-(tj+tj+dmid/vj)
 go to 35
54 dmid=(dsg(k11)+dsg(kt22))/2
 go to 52
c
c tie down points for deeper layers l2+1 thru nl (sp jj)
c
60 lnext=l2+1
 if(lnext.gt.nl) go to 99
 dt1=sum1
 do 80 l=lnext,nl
 l1=l-1
 dt2=dt1*vha(m,l1)/vha(m,l)
 do 70 k=1,kn
 if(lg(k,jj,m).eq.1) tr(k,jj,m)=tr(k,jj,m)-dt2
70 continue
 dt1=dt2
80 continue
 jjtie=1
99 return
 end
c
c
c ****** Function TERP.   written 09/06/81  1445.3rew ******************
c computes interpolated value of y corresponding to x, given the 2 pts
c (x1,y1) and (x2,y2)
c
 function terp(x1,y1,x2,y2,x)
 if (abs (x2-x1).lt.0.1) go to 2
       terp=((x-x1)*(y2-y1))/(x2-x1)+y1
1  return
2 terp=(y1+y2)/2.0
       go to 1
       end
c
```

```
c
c **** Subroutine SETIP (#2).  written 09/06/81  1445.3rew ************
c sets print for line of depth pt graph
c
 subroutine setip(e,emin,escale,isym)
 character*1 ibl,iques,ip,it,icoln,iplus,isq,idash,isym,ipi,iti
 common ibl,iques,ip,it,icoln,iplus,isq,idash
 dimension ip(103),it(53)
c
 i=2+ifix ((e-emin)/escale+0.5)
 ii=(i-2)/2+2
 if (i.gt.0) go to 1
 i=1
 ii=1
 go to 2
c
1 if (i.le.103) go to 2
 i=103
 ii=53
2 ipi=ip(i)
 iti=it(ii)
 if(ipi.eq.ibl.or.ipi.eq.icoln.or.ipi.eq.iplus.or.ipi.eq.idash)
     1 go to 4
 ip(i)=isq
 go to 6
4 ip(i)=isym
6 if(iti.eq.ibl.or.iti.eq.icoln.or.iti.eq.iplus.or.iti.eq.idash)
     1 go to 8
 it(ii)=isq
 go to 10
8 it(ii)=isym
10 return
 end
c
c
c Subroutine REGV (#3).  written 09/06/81  1446.7rew ******************
c computes and prints regression velocities and intercept t for layer l
c
 subroutine regv(l,nixit)
 character*1 ibl,jtrl,jprt,jvel,idspr,idsp,idum
 common ibl
 common/blk0/lg
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,idum
 common/blk3/ta,tr,dsg
 common/blk5/idspr,idsp
 common/ibm3/kl,kr,d
 common/blk11/vreg,preg
 dimension nj(5),nk( 5),kl(7,5),kr(7,5),d(48,7,5),idspr(5),
     1 idsp(7,5),vreg(5),preg(5),lg(48,7,5),ta(48,7,5),tr(48,7,5),
     2 dsg(48)
c
 if (nixit.eq.0.and.jprt.ne.ibl) write(20,1) l
1 format (' Layer',i2,' velocity and time intercepts computed by',
```

```
     1 ' regression')
 if(jvel.ne.ibl) write(*,1) l
 sum1=0.0
 pts1=0.0
 do 50 m=1,nm
 mcall=m
 none=0
 jn=nj(m)
 kn=nk(m)
 sum2=0.0
 pts2=0.0
 do 40 j=1,jn
 jcall=j
 call kends(1,mcall,jcall,1,kl(j,m),kl1,kl2)
 if (kl1.eq.0) go to 10
 do 5 k=kl1,kl2
 dsg(k)=d(k,j,m)
5 continue
 call regres(kl1,kl2,jcall,mcall,1,vl,tlf,pt,0)
 if (pt.le.1.0) go to 10
 tlf=tlf+tlf
 sum3=pt/vl
 pts3=pt
 go to 15
10 vl=0.0
 tlf=0.0
 sum3=0.0
 pts3=0.0
15 call kends(1,mcall,jcall,kr(j,m),kn,kr1,kr2)
 if (kr1.eq.0) go to 23
 do 20 k=kr1,kr2
 dsg(k)=d(k,j,m)
20 continue
 call regres(kr1,kr2,jcall,mcall,1,vr,trt,pt,0)
 if (pt.le.1.0) go to 23
 trt=trt+trt
 pts3=pts3+pt
 sum3=sum3+pt/vr
 go to 24
23 vr=0.0
 trt=0.0
 avgt=tlf
24 sum2=sum2+sum3
 pts2=pts2+pts3
 sum1=sum1+sum3
 pts1=pts1+pts3
 if (pts3.eq.0.0) go to 40
 if (tlf.eq.0.0) avgt=trt
 if (tlf.ne.0.0.and.trt.ne.0.0) avgt=(tlf+trt)/2.0
 if (none.ne.0) go to 27
 none=1
 if (nixit.eq.0.and.jprt.ne.ibl) write(20,25) idspr(m)
25 format (/8h spread ,a1/4x,22hvel   time   geos   sp,5x,
     1  17hgeos  time    vel,8x,19havg v   avg t   pts/2x,6h------,
```

```
     2  2x,4h----,2x,5h-----,3x,3h---,3x,5h-----,2x,4h----,2x,6h------,
     3  6x,6h------,3x,5h-----,3x,3h---)
 if(jvel.ne.ibl) write(*,25) idspr(m)
27 sum3=pts3/sum3
 if (nixit.eq.1) go to 40
 if(jprt.ne.ibl) write(20,31) vl,tlf,kl1,kl2,idsp(j,m),kr1,kr2,
     1 trt,vr,sum3,avgt,pts3
31 format (1x,3pf7.0,0pf6.1,1x,2i3,4x,a1,3x,2i3,f6.1,3pf8.0,f12.0,
     1  0pf8.1,f6.0)
 if(jvel.ne.ibl) write(*,31)vl,tlf,kl1,kl2,idsp(j,m),kr1,kr2,trt,
     1 vr,sum3,avgt,pts3
40 continue
c
 if (none.eq.0) go to 50
 sum2=pts2/sum2
 if (nixit.eq.1) go to 50
 if(jprt.ne.ibl) write(20,41) sum2,pts2
 if(jvel.ne.ibl) write(*,41)sum2,pts2
41 format (/51x,3havg,3pf7.0,0pf14.0)
42 format (/44x,10havg of all,3pf7.0,0pf14.0)
50 continue
c
 if (pts1.eq.0.0) go to 56
 vreg(1)=pts1/sum1
 preg(1)=pts1
 if (nm.eq.1.or.nixit.eq.1) go to 60
 if(jprt.ne.ibl) write(20,55)
55 format (/55x,7h-------,10x,3h---)
 if(jprt.ne.ibl) write(20,42) vreg(1),pts1
 if(jvel.eq.ibl) go to 60
 write(*,55)
 write(*,42)vreg(1),pts1
 go to 60
56 vreg(1)=0.0
 preg(1)=0.0
c
 if (nixit.eq.0.and.jprt.ne.ibl) write(20,57)
57 format(//,8h--none--)
 if(jvel.ne.ibl) write(*,57)
60 return
 end
c
c
c ***** Subroutine HOBV (#4).  Written 09/06/81  1446.7rew **********
c computes horiz vel of layer 1 by hobson-overton method
c
 subroutine hobv(1,nixit)
 character*1 ibl,jtrl,jprt,idspr,idsp,jvel,idum
 common ibl
 common/blk0/lg
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,idum
 common/blk3/ta,tr,dsg
 common/blk5/idspr,idsp
```

```
 common/ibm3/kl,kr,d
 common/blk12/vhob,phob
 dimension ep(48),dx(48),dt(48),nj(5),nk(5),idspr(5),lg(48,7,5),
     1  ta(48,7,5),kr(7,5),kl(7,5),idsp(7,5),vhob(5),phob(5),d(48,7,5),
     2  dsg(48),ki(5),pe(5),tr(48,7,5)
c
 if (nixit.eq.0.and.jprt.ne.ibl) write(20,2) l
2 format (//,' Layer',i2,' velocity computed by hobson-overton',
     1 ' method')
 if(jvel.ne.ibl) write(*,2)l
 sum2=0.0
 pts2=0.0
c
 do 22 m=1,nm
 mcall=m
 none=0
 jn=nj(m)
 kn=nk(m)
 sum3=0.0
 pts3=0.0
 j2=jn-1
c
 do 18 j=1,j2
 jcall=j
 call kends(l,mcall,jcall,kr(j,m),kn,kr1,kr2)
 if (kr1.eq.0) go to 18
 j1=j+1
c
 do 16 jj=j1,jn
 jjcall=jj
 call kends(l,mcall,jjcall,1,kl(jj,m),kl1,kl2)
 k1=max0(kr1,kl1)
 k2=min0(kr2,kl2)
 if (kl1.eq.0.or.(k2-k1).le.0) go to 16
c
c begin hobson-overton routine
c
 sdx=0.0
 sdx2=0.0
 sdt=0.0
 sdtdx=0.0
 seep=0.0
 pt=0.0
 do 3 k=1,12
 ep(k)=0.0
3 continue
 do 4 k=k1,k2
 if (lg(k,j,m).ne.1.or.lg(k,jj,m).ne.1) go to 4
 dx(k)=abs(d(k,j,m))-abs(d(k,jj,m))
 sdx=sdx+dx(k)
 sdx2=sdx2+dx(k)**2
 dt(k)=ta(k,j,m)-ta(k,jj,m)
 sdt=sdt+dt(k)
 sdtdx=sdtdx+dx(k)*dt(k)
```

```
 pt=pt+1.0
4 continue
 if (pt.le.1.0) go to 9
 v=(sdx2-sdx**2/pt)/(sdtdx-(sdx*sdt)/pt)
 tdsp=(sdt-sdx/v)/pt
 do 6 k=k1,k2
 if (lg(k,j,m).ne.1.or.lg(k,jj,m).ne.1) go to 6
 ep(k)=dt(k)-dx(k)/v-tdsp
 seep=seep+ep(k)**2
6 continue
 seep=sqrt (seep/pt)
 do 8 ik=1,5
 ki(ik)=0
 pe(ik)=0.
 do 7 k=k1,k2
 if(abs(ep(k)).le.abs(pe(ik))) go to 7
 pe(ik)=ep(k)
 ki(ik)=k
7 continue
 if(ki(ik).eq.0) go to 8
 k=ki(ik)
 ep(k)=0.
8 continue
 go to 10
9 v=0.0
 pt=0.0
c
c end hobson-overton routine
c
10 if (pt.eq.0.0) go to 16
 sum3=sum3+v*pt
 pts3=pts3+pt
 if (nixit.eq.1) go to 16
 if (none.ne.0) go to 12
 if(jprt.ne.ibl) write(20,11) idspr(m)
11 format(/1x,' spread ',a1,39x,'5 highest eps'/7x,
     1    'vel   sps   geos',
     2 '  tdsp  se ep',5(4x,'ep   geo')/5x,'------',2x,
     3  '---  -----  ----',2x,'-----',5(3x,'---------'))
12 if(jprt.ne.ibl) write(20,13) v,idsp(j,m),idsp(jj,m),k1,k2,tdsp,
     1 seep,(pe(k),ki(k),k=1,5)
13 format (1x,3pf10.0,2x,a1,1x,a1,1x,2i3,0pf6.1,f7.3,5(f8.3,i4))
 if(jvel.eq.ibl) go to 19
 if(none.ne.0) go to 17
 write(*,14) idspr(m)
14 format(/1x,'spread ',a1,42x,'4 highest eps'/3x,
     1  'vel   sps   geos','  tdsp  se ep',4(4x,'ep   geo')/2x,
     2  '------',1x,'---  -----  ----',2x,'-----',4(3x,'---------'))
17 write(*,15) v,idsp(j,m),idsp(jj,m),k1,k2,tdsp,seep,
     1 (pe(k),ki(k),k=1,4)
15 format(1x,3pf7.0,1x,a1,1x,a1,1x,2i3,0pf6.1,f7.3,4(f8.3,i4))
19 none=1
16 continue
18 continue
```

```
c
 sum2=sum2+sum3
 pts2=pts2+pts3
 if (pts3.eq.0.0) go to 22
 sum3=sum3/pts3
 if (nixit.eq.0.and.jprt.ne.ibl) write(20,20) sum3,pts3
 if(jvel.ne.ibl) write(*,20)sum3,pts3
20 format (//,5x,'avg=',3pf7.0,' for',0pf5.0,' points')
22 continue
c
 if (pts2.eq.0.0) go to 26
 sum2=sum2/pts2
 if (nm.eq.1.or.nixit.eq.1) go to 28
 if(jprt.ne.ibl) write(20,25) sum2,pts2
 if(jvel.ne.ibl) write(*,25) sum2,pts2
25 format(//,1x,'avg of all=',3pf7.0,' for',0pf5.0,' points')
 go to 28
26 if(nixit.eq.0.and.jprt.ne.ibl) write(20,27)
 if(jvel.ne.ibl) write(*,27)
27 format(/3x,'not enough points')
c
28 vhob(l)=sum2
 phob(l)=pts2
 return
 end
c
c
c ***** Subroutine REGRES (#5).  Written 09/06/81  1446.7rew ************
c computes velocity v by regression of time pts (ta) at distances d from
c sp j to geos between indices k1 and k2 for layer l, spread m.
c only nonzero ta for which lg=l are used in regression.
c half intercept time at sp j is given by t, num of regressed pts is pt.
c
 subroutine regres(k1,k2,j,m,l,v,t,pt,it)
 common/blk0/lg
 common/blk3/ta,tr,d
 common/blk4/vva,vha
 dimension d(48),ta(48,7,5),tr(48,7,5),lg(48,7,5),vha(5,5),
     1 vva(5,5),tar(48)
c
 if(it.eq.1) go to 2
 do 1 k=k1,k2
1 tar(k)=ta(k,j,m)
 go to 4
2 do 3 k=k1,k2
3 tar(k)=tr(k,j,m)
4 s1=0.0
 s2=0.0
 pt=0.0
 t=0.0
 v=0.0
 do 5 k=k1,k2
 if (lg(k,j,m).ne.l) go to 5
 s1=s1+d(k)
```

```
 s2=s2+tar(k)
 pt=pt+1.0
5 continue
c
 if (pt.le.1.0) go to 15
 xbar=s1/pt
 tbar=s2/pt
 s1=0.0
 s2=0.0
 do 10 k=k1,k2
 if (lg(k,j,m).ne.l) go to 10
 xd=d(k)-xbar
 s1=s1+xd*tar(k)
 s2=s2+xd**2
10 continue
 v=abs (s2/s1)
 t=(tbar-xbar*s1/s2)/2.0
12 return
c
15 if (pt.eq.0.0) go to 12
 v=vha(m,l)
 if (v.le.0.0) go to 12
 t=(s2-s1/v)/2.0
 go to 12
 end
```

```
c ************************ S2.FOR ************************************
c
c Program SIPT1.FOR -- version 1.0 -- USGS, Hartford, CT -- 08-15-86
c
c Micro-computer implementation of sipt22.fortran (on Denver PRIME).
c Compiled on an IBM PC using the IBM Professional FORTRAN compiler
c (version 1.0).
c Program consists of 9 modules: MAIN.FOR, PART2.FOR, PART3.FOR,
c INITIAL.FOR, S1.FOR, S2.FOR, S3.FOR, S4.FOR and S5.FOR.
c
c *********** subroutines resumes ************************************
c
c ****** Subroutine RAYUP (#6).  Written 09/06/81  1446.7rew ***********
c traces ray from starting point on top of layer l or ll to ending point
c (x0,e0) within or on the upper boundary of layer l0.  refracting horiz
c is the top of layer l.  computes and returns corrected coord of start
c point and total travel time. for irep=1, ll=l-1, and ray start point
c is taken as (xll,ell) on top of layer ll,  for irep=2 or 3, ll=l, and
c start point is taken as (xl,el) on top of layer l. also for irep=2 or
c 3, ray intersection with top of layer l-1 is outputted as (xll,ell)
c and time from this point as tll. input parameter bll is precomputed
c as avg dip of refractor over entire spread. if bll is nonzero on input
c it is used in place of interval dip between  geo pairs which is
c otherwise computed internally. input parameter i is preset to 1 for
c rays going up and right, 2 for rays going up and left.
c for the case where shot is below refractor (es(j,m).lt.ers(j,m,ll)),
c tll=big on input and ray is traced from shot to geo.
c
 subroutine rayup(l,ll,l0,j,m,i,x0,e0,xll,ell,tll,xl,el,tl,bll)
 character*1 ibl,jtrl,jprt,jvel,idum
 character*12 ipath
 common ibl
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,idum
 common/blk2/xg,erp,slope,idip
 common/blk4/vva,vha
 common/blk6/trp,jjoff
 common/blk7/ja,jb,trs,ers,xsp,esp,ls
 common/blk9/eg,es
 common/blk10/blim,itrace,tansg
        dimension nk(5),xg(48,5),erp(48,5,4),vva(5,5),vha(5,5),nj(5),
     1   ja(5),jb(5),trs(7,5,4),ers(7,5,4),xsp(7,5),esp(7,5),ls(7,5),
     2   es(7,5),eg(48,5),trp(48,5,4)
 data big,small/999999.,0.000001/
c
 none=0
 xlls=xll
 tlls=tll
 xls=xl
 tls=tl
 if (l.eq.ll) xll=xl
2 iblsw=0
3 xrefl=xll
 xrefll=xll
```

```
 tll=0.0
 tl=0.0
 m1=m
 l2=ll
 l1=l2-1
c compute slope of ray from starting point
 inval=0
 mflag=0
 if(bll.ne.0.) bl=bll
c first find k of geos bounding xll
4 kn=nk(m1)
 k3=kn-1
 k1=1
 k2=2
 if (xll.lt.xg(1,m1)) go to 8
 k1=kn-1
 k2=kn
 if (xll.gt.xg(kn,m1)) go to 11
 do 6 k1=1,k3
 k2=k1+1
 if (xll.le.xg(k2,m1)) go to 15
6 continue
 go to 14
c case of xll left of spread m1
8 if (m1.eq.1) go to 9
 if (inval.gt.0) go to 15
 inval=-1
 m1=m1-1
 go to 4
c case of xll left of spread 1
9 if(bll.ne.0..or.idip.ne.0) go to 10
 kmid=kn/2
 k1=1
 k2=2
 if(kmid-k1.lt.5) kmid=k1+5
 if(kmid.gt.kn) kmid=kn
 call dip(l1,m1,k1,kmid,k1,a,bl)
10 if(jjoff.eq.0) go to 310
 j2=ja(1)
 if(xll.ge.xsp(j2,1).or.j2.le.1) go to 310
110 j1=j2-1
 if(xll.ge.xsp(j1,1).or.j1.eq.1) go to 210
 j2=j2-1
 go to 110
210 ell=terp(xsp(j1,1),ers(j1,1,l1),xsp(j2,1),ers(j2,1,l1),xll)
 go to 410
310 if(idip.ne.0.and.bll.eq.0.) bl=slope
 if(bl.gt.blim) bl=blim
 if(bl.lt.-blim) bl=-blim
 ell=erp(k1,m1,l1)+bl*(xll-xg(k1,m1))
410 mflag=1
 go to 16
c case of xll right of spread m1
11 if (m1.eq.nm) go to 12
```

```
 if (inval.lt.0) go to 15
 inval=1
 m1=m1+1
 go to 4
c case of xll right of spread nm
12 if(bll.ne.0..or.idip.ne.0) go to 13
 kmid=kn/2+1
 k1=kn-1
 k2=kn
 if(k2-kmid.lt.5) kmid=k2-5
 if(kmid.lt.1) kmid=1
 call dip(l1,m1,kmid,k2,k2,a,bl)
13 if(jjoff.eq.0) go to 313
 jn=nj(nm)
 j1=jb(nm)
 if(xll.le.xsp(j1,nm).or.j1.ge.jn) go to 313
113 j2=j1+1
 if(xll.le.xsp(j2,nm).or.j2.eq.jn) go to 213
 j1=j1+1
 go to 113
213 ell=terp(xsp(j1,nm),ers(j1,nm,l1),xsp(j2,nm),ers(j2,nm,l1),xll)
 go to 413
313 if(idip.ne.0.and.bll.eq.0.) bl=slope
 if(bl.gt.blim) bl=blim
 if(bl.lt.-blim) bl=-blim
 ell=erp(k2,m1,l1)+bl*(xll-xg(k2,m1))
413 mflag=2
 go to 16
14 k1=kn-1
 k2=kn
c k1, k2, and m1 now known. compute ell and bl for layer l1 at xll.
15 ell=terp(xg(k1,m1),erp(k1,m1,l1),xg(k2,m1),erp(k2,m1,l1),xll)
16 el=ell
 erefl=ell
 erefll=ell
 if (none.ne.0) go to 17
 ells=ell
 els=ell
17 if (bll.eq.0.0) go to 18
 bl=bll
 go to 19
18 if (iblsw.ne.0) go to 19
 denom=xg(k2,m1)-xg(k1,m1)
 if (denom.gt.0.1) go to 118
 bl=slope
 go to 19
118 if(mflag.ne.0) go to 119
 bl=(erp(k2,m1,l1)-erp(k1,m1,l1))/denom
119 if (bl.gt.blim) bl=blim
 if (bl.lt.-blim) bl=-blim
19 blref=bl
 if(tlls.lt.big) go to 219
c
c ray traced from es when es is below refractor.
```

```
c
 denom=sqrt(((xll-xsp(j,m))**2+(ell-es(j,m))**2)*(1.+bl**2))
 if(denom.lt.0.001) go to 219
 sinr=(ell-es(j,m)-bl*(xll-xsp(j,m)))/denom
 if(abs(sinr).gt.0.999) go to 219
 sini=sinr*vva(m1,l1)/vha(m1,l)
 tani=sini/sqrt(1.-sini**2)
 go to 120
219 tani=vva(m1,l1)/sqrt(vha(m1,l)**2-vva(m1,l1)**2)
c compute slope of ray emerging from l2
120 if (i.eq.2) tani=-tani
c entry pt for rays after 1st one
20 denom=tani-bl
c decrement l1 in preparation for finding intersection w/ horizon above
 l1=l1-1
 l2=l2-1
c vertical ray test
 if (abs(denom).lt.small) go to 39
c nonvertical ray
 bray=(tani*bl+1.0)/denom
 aray=ell-bray*xll
c test for uppermost ray -- if so compute xl1, tll, tl, and then exit
 if (l2.gt.l0) go to 23
 xl1=(e0-aray)/bray
 t=sqrt ((xl1-xll)**2+(e0-ell)**2)/vva(m1,l2)
 if (e0.gt.ell) go to 22
121 t=0.00001
21 l3=l2+1
 t=t*vva(m1,l2)/vva(m1,l3)
22 tll=tll+t
 tl=tl+t
 go to 46
c not uppermost ray -- compute tentative intersection w/ horizon above
23 inval=0
24 if(bll.ne.0.) go to 25
 denom=xg(k2,m1)-xg(k1,m1)
 if(denom.gt.0.1) go to 124
 bl=slope
 go to 25
124 if(mflag.eq.0) go to 125
 if(idip.eq.0) go to 224
 bl=slope
 go to 25
224 if(mflag.eq.2) go to 225
c mflag=1 (xll or xl1 left of spread m1)
324 kmid=kn/2
 k1=1
 k2=1
 if(kmid-k1.lt.5) kmid=k1+5
 if(kmid.gt.kn) kmid=kn
 call dip(l1,m1,k1,kmid,k1,a,bl)
 if(jjoff.eq.0.or.inval.eq.0) go to 25
 j2=ja(1)
 if(xl1.ge.xsp(j2,1).or.j2.le.1) go to 25
```

```
424 j1=j2-1
 if(xl1.ge.xsp(j1,1).or.j1.eq.1) go to 524
 j2=j1
 go to 424
524 bl=(ers(j2,m1,l1)-ers(j1,m1,l1))/(xsp(j2,m1)-xsp(j1,m1))
 go to 25
c mflag=2 (xll or xl1 right of spread m1)
225 kmid=kn/2+1
 k1=kn
 k2=kn
 if(k2-kmid.lt.5) kmid=k2-5
 if(kmid.lt.1) kmid=1
 call dip(l1,m1,kmid,k2,k2,a,bl)
 if(jjoff.eq.0.or.inval.eq.0) go to 25
 jn=nj(nm)
 j1=jb(nm)
 if(xl1.le.xsp(j1,nm).or.j1.ge.jn) go to 25
625 j2=j1+1
 if(xl1.le.xsp(j2,nm).or.j2.eq.jn) go to 524
 j1=j2
 go to 625
125 bl=(erp(k2,m1,l1)-erp(k1,m1,l1))/denom
25 if(bl.gt.blim) bl=blim
 if(bl.lt.-blim) bl=-blim
 al=erp(k1,m1,l1)-bl*xg(k1,m1)
c test for ray parallel with horizon above
 denom=bray-bl
 if (abs (denom).ge.small) go to 28
 if (bray) 32,26,36
26 go to (36,32), i
c test for valid intersection
28 xl1=(al-aray)/denom
 if(k1.eq.k2) go to 30
 if (xl1.lt.xg(k1,m1)) go to 32
 if (xl1.gt.xg(k2,m1)) go to 36
c valid intersection found
30 el1=al+bl*xl1
 if (abs (bray).lt.small) bray=sign(small,bray)
 if (bl.gt.blim) bl=blim
 if (bl.lt.-blim) bl=-blim
 denom=1.-bl/bray
 if(abs(denom).lt.small) go to 52
 tanr=(bl+1.0/bray)/denom
 t=sqrt ((xl1-xll)**2+(el1-ell)**2)/vva(m1,12)
31 tl=tl+t
c test for case where tll starts accumulating at l-1, not l
 if (ll.eq.l.and.l2.eq.(l-1)) go to 43
 tll=tll+t
 go to 44
c intersection not valid -- search to left
32 if (inval.gt.0) go to 30
 if (k1.eq.1) go to 34
 k2=k1
 k1=k2-1
```

```
33 inval=-1
  go to 24
34 if (m1.eq.1) go to 35
 m1=m1-1
 kn=nk(m1)
 k2=kn
 k1=k2-1
 inval=-1
 go to 24
c case of xl1 left of spread 1
35 inval=1
 mflag=1
 go to 324
c intersection not valid -- search to right
36 if (inval.lt.0) go to 30
 if (k2.eq.kn) go to 38
 k1=k2
 k2=k1+1
37 inval=1
 go to 24
38 if (m1.eq.nm) go to 138
 m1=m1+1
 kn=nk(m1)
 k1=1
 k2=2
 inval=1
 go to 24
c case of xl1 right of spread nm
138 inval=-1
 mflag=2
 go to 225
c vertical ray -- test if uppermost -- if so compute tl, tll and exit.
39 xl1=xll
 if (l2.gt.l0) go to 40
 t=(e0-ell)/vva(m1,l2)
 if(t.le.0.) go to 121
 go to 22
c vertical ray -- not uppermost one
40 if(bll.ne.0.) go to 141
 denom=xg(k2,m1)-xg(k1,m1)
 if(denom.gt.0.1) go to 41
 bl=slope
 go to 141
41 bl=(erp(k2,m1,l1)-erp(k1,m1,l1))/denom
 if (bl.gt.blim) bl=blim
 if (bl.lt.-blim) bl=-blim
141 al=erp(k1,m1,l1)-bl*xg(k1,m1)
42 el1=al+bl*xl1
 tanr=bl
 t=(el1-ell)/vva(m1,l2)
 go to 31
c
43 xrefll=xl1
 erefll=el1
```

```
 ells=el1
c prepare to continue tracing ray upward
44 xll=xl1
 ell=el1
 sini=vva(m1,l1)*tanr/(sqrt(1.0+tanr**2)*vva(m1,l2))
 tani=sini/sqrt (1.0-sini**2)
 go to 20
c exit from ray-trace routine -- prepare to trace more rays if necessary
46 if (none.gt.0) go to 50
c first ray traced -- store results
 none=none+1
 xs1=x0-xl1
 xrll1=xrefll
 xrl1=xrefl
 erl1=erefl
 erll1=erefll
 tl1=tl
 tll1=tll
 blref1=blref
        epss=abs(xs1)
 if (itrace.ne.0.and.jprt.ne.ibl) write(20,47) none,xl1,xrefll,
     1 erefll,tll,xrefl,erefl,tl,xs1
47 format (22x,'ray',i2,f9.1,9x,6f9.1,f15.1)
 if (abs (xs1).lt.0.5) go to 67
49 xll=xrefl+xs1
 go to 2
c second ray traced -- store results
50 none=none+1
 xs2=x0-xl1
 xrl2=xrefl
 xrll2=xrefll
 erl2=erefl
 erll2=erefll
 tl2=tl
 tll2=tll
 blref2=blref
        if (abs(xs2).ge.epss) go to 51
 epss=abs(xs2)
 xlls=xrefll
 ells=erefll
 tlls=tll
 xls=xrefl
 els=erefl
 tls=tl
51 if (itrace.ne.0.and.jprt.ne.ibl) write(20,47) none,xl1,xrefll,
     1 erefll,tll,xrefl,erefl,tl,xs2
c make tests for accepting first two rays traced
 if (xs1*xs2.lt.0.0) go to 53
c the two rays are on same side of sp or geo
 if (abs (xs2).lt.abs (xs1)) go to 54
c the 2nd ray is not closer to sp or geo than 1st ray
 if (none.gt.4) go to 52
 xl1=xrefl+xs1
        go to 56
```

```
c give up and resort to using saved input values, then return
52 xll=xlls
 ell=ells
 tll=tlls
 xl=xls
 el=els
 tl=tls
 ipath='div: give up'
 go to 63
c test if 2nd ray comes within 10 ft of objective
53 if (abs(xs2).le.10.0) go to 58
c not within 10 ft. if 4 or less rays traced try once more after
c interpolating bl. if more than 4 rays traced, accept last pair.
 if (none.gt.4) go to 58
 iblsw=1
 xll=(xrl1*xs2-xrl2*xs1)/(xs2-xs1)
        bl=terp(xrl1,blref1,xrl2,blref2,xll)
 xrl1=xrl2
 xrll1=xrll2
 erl1=erl2
 erll1=erll2
 tl1=tl2
 tll1=tll2
 xs1=xs2
 go to 3
c test if extrapolation is permissible
54 if (abs (xs2).le.abs (xs1-xs2)) go to 57
 if (none.gt.4) go to 52
c readjust starting point and then retrace 2nd ray
55 xll=terp(xs1,xrl1,xs2,xrl2,0.0)
56 xrl1=xrl2
 xrll1=xrll2
 erl1=erl2
        erll1=erll2
 tl1=tl2
 tll1=tll2
 xs1=xs2
 blref1=blref2
 go to 2
c test if 2nd ray within 10 ft of objective
57 if (abs(xs2).le.10.0) go to 58
c not within 10 ft. if 4 or less rays traced try once more, otherwise
c accept the last pair traced.
 if (none.gt.4) go to 58
 go to 55
c interpolate or extrapolate to obtain xll,ell,tll,xl,el,tl, then return
58 xl=terp(xs1,xrl1,xs2,xrl2,0.0)
 xll=terp(xs1,xrll1,xs2,xrll2,0.0)
 ell=terp(xrll1,erll1,xrll2,erll2,xll)
 el=terp(xrl1,erl1,xrl2,erl2,xl)
        tll=terp(xrll1,tll1,xrll2,tll2,xll)
 tl=terp(xrl1,tl1,xrl2,tl2,xl)
 ipath='interp/extrp'
 if (tl.lt.tll) go to 52
```

```
63 if (itrace.ne.0.and.jprt.ne.ibl) write(20,65) ipath,xll,ell,
     1 tll,xl,el,tl,bl
65 format (15x,a12,18x,6f9.1,f9.4)
 if(tlls.ge.big) tll=0.
 return
c very close approximation (.lt.0.5 ft). no further ray tracing needed
67 xll=xrefll
 xl=xrefl
 ell=erefll
 el=erefl
 go to 63
 end
c
c
c ******* Subroutine RAYCOR (#7).  Written 09/06/81  1446.7rew ********
c adjusts coordinates of bottom end point of ray entering or leaving the
c refracting horizon so that total time of computer-traced ray agrees
c with total observed time.
c
 subroutine raycor(x1,x2,e1,e2,vv,hv,tcor)
c
 dx=x1-x2
 de=e1-e2
 denom=sqrt (dx**2+de**2)/vv-abs (dx)/hv
 if (denom.lt.0.1) denom=0.1
 fctr=tcor/denom
 x2=x2-dx*fctr
 e2=e2-de*fctr
5 return
 end
```

```
c ************************ S3.FOR ************************************
c
c Program SIPT1.FOR -- version 1.0 -- USGS, Hartford, CT -- 08-15-86
c
c Micro-computer implementation of sipt22.fortran (on Denver PRIME).
c Compiled on an IBM PC using the IBM Professional FORTRAN compiler
c (version 1.0).
c Program consists of 9 modules: MAIN.FOR, PART2.FOR, PART3.FOR,
c INITIAL.FOR, S1.FOR, S2.FOR, S3.FOR, S4.FOR and S5.FOR.
c
c ************ subroutine resumed ************************************
c
c **** Subroutine PLOT (#8).  written 09/06/81  1446.7rew **************
c plots t-d graph (igoto=1) or depth graph (igoto=2)
c
 subroutine plot(igoto)
        character*1 ibl,iques,ip,it,icoln,iplus,isq,idash,is,iast,
     1   idee,il,jprt,jtrl,idspr,idsp,krs2,krg,isymbl,jsymbl,lblj,
     2   lblm,jvel,idum
 character*4 ident
 common ibl,iques,ip,it,icoln,iplus,isq,idash,is,iast,idee,il,
     1    ident
 common/ibm1/nl,ln,p,tscale,xscale,xsco2,escale,xlim1,xlim2,
     1    iplot
 common/blk0/lg
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,idum
 common/blk2/xg,erp,slope,idip
 common/blk3/ta,tr,dsg
 common/blk5/idspr,idsp
 common/ibm3/kl,kr,d
 common/blk6/trp,jjoff
 common/blk7/ja,jb,trs,ers,xsp,esp,ls
 common/blk8/prg,erg,prs2,ers2,zrsp,zrg
 common/ibm4/krg,krs2
 common/blk9/eg,es
 dimension ident(20),il(5),idspr(5),ja(5),jb(5),nj(5),nk(5),
     1  xsp(7,5),esp(7,5),es(7,5),idsp(7,5),kl(7,5),kr(7,5),ip(103),
     2  it(53),ers(7,5,4),prs2(7,5,4,2),ers2(7,5,4,2),krs2(7,5,4,2),
     3  xg(48,5),eg(48,5),p(48,7,5),ta(48,7,5),lg(48,7,5), dsg(48),
     4  prg(48,7,5),erg(48,7,5),krg(48,7,5),erp(48,5,4),d(48,7,5),
     5  trs(7,5,4),trp(48,5,4),ls(7,5),tr(48,7,5),zrsp(7),zrg(48,7)
c internal arrays
 dimension lb(11),prgall(1680),ergall(1680),isymbl(1680),
     1  prsall(280),ersall(280),jsymbl(280) ,
     2  enext(4),elast(4),erite(4),
     3  jptr(5),kptr(5),
     4  kjpt(7,5)
 data big/9999999./
c
 go to (500,3000),igoto
c
c t-d plot routine
c
```

```
500 lbinc=ifix (10.0*tscale)
c
c main t-d graph plot loop
c
 lb(1)=0
 do 510 i=2,11
 lb(i)=lb(i-1)+lbinc
510 continue
 if(jprt.ne.ibl)write(20,57) ident
57 format (//,20a4)
 go to (515,520,525,530),iplot
515 if(jprt.ne.ibl)write(20,516)
 if(jtrl.ne.ibl)write(*,518)
518 format(/1x,'s',6x,'t-d plot -- raw data, no corrections')
516 format (//,2h s,20x,
     1  40htime-distance plot  --  raw data with no,
     2  20h corrections applied)
 go to 540
520 if(jprt.ne.ibl)write(20,521)
 if(jtrl.ne.ibl)write(*,10521)
10521 format(/1x,'s',6x,'t-d plot -- t corrected to datum elev')
521 format (//,2h s,20x,
     1  42htime-distance plot  --  times corrected to,
     2 16h datum elevation)
 go to 540
525 if(jprt.ne.ibl) write(20,526)
 if(jtrl.ne.ibl) write(*,527)
526 format('  s',20x,'time-distance plot -- pre-depth values with'
     1  ,' tie corr if jjoff=0')
527 format(/' s',6x,'t-d plot -- pre-depth w/ tie corr if jjoff=0')
 go to 540
530 if(jprt.ne.ibl)write(20,531)
 if(jtrl.ne.ibl)write(*,10531)
10531 format(/1x,'s',6x,'t-d plot -- layer 1 removed')
531 format (//,2h s,40x,39htime-distance plot  --  layer 1 removed)
540 if(jprt.ne.ibl)write(20,541)
 if(jtrl.ne.ibl)write(*,10541)
10541 format(1x,'p  g',65x,'d'/1x,'r  e  s',62x,'i'/' e  o  p',62x,
     1  's',/1x,'a',68x,'t'/1x,'d',13x,
     2  't i m e    ( m i l l i s e c o n d s )')
541 format (1h ,5h p  g,62x,50x,1hd/1h ,8h r  e  s,59x,50x,1hi/1h ,
     1  8h e  o  p,59x,50x,1hs/1h ,2h a,62x,53x,1ht/1h ,2h d,40x,
     2 38ht i m e    ( m i l l i s e c o n d s ))
 if(jprt.ne.ibl)write(20,584) (lb(i),i=1,11)
 if(jtrl.ne.ibl)write(*,10584)(lb(i),i=1,11,2)
10584 format(1x,i9,5i10)
584 format (1h ,11i10)
 if(jprt.ne.ibl)write(20,585)
 if(jtrl.ne.ibl)write(*,10585)
10585 format(9x,'+',5('---------+'))
585 format (1h ,9x,1h+,10(10h---------+))
c
c initialize pointers and xlim
c jptr(m) for sp labels
```

```
c kptr(m) for geo labels
c kjpt(j,m) for arrival times
c
 do 595 m=1,nm
 kptr(m)=1
 if (m.eq.1.or.jjoff.eq.0) go to 586
 j=1
 go to 587
586 j=ja(m)
587 if (xsp(j,m).ge.(xlim1-xscale)) go to 588
 j=j+1
 if (j.le.nj(m)) go to 587
 j=nj(m)
588 jptr(m)=j
 jn=nj(m)
 do 590 j=1,jn
 kjpt(j,m)=1
590 continue
595 continue
 xlim=xlim1
 xmin=xlim1-xscale
627 kflag=0
 lblj=ibl
 lblm=ibl
 xmid=xlim-xsco2
c
c set print for printing lblm (spread) and lblj (sp), also set kflag
c kflag=-1 for m and j label
c kflag= 0 for no label (blank)
c kflag= 1 for m, k and j label
c
 do 645 m=1,nm
 kn=nk(m)
 jn=nj(m)
 if (m.eq.nm) jn=jb(nm)
 kp=kptr(m)
 if (kp.gt.kn) go to 635
 if (xg(kp,m).ge.xlim) go to 635
 kflag=1
 lblm=idspr(m)
 lk=kp
 dist=xg(kp,m)
 kptr(m)=kptr(m)+1
635 jp=jptr(m)
 if (jp.gt.jn) go to 645
 if (xsp(jp,m).ge.xlim.or.xsp(jp,m).lt.xmin) go to 645
 if (kflag.eq.1) go to 638
 kflag=-1
638 lblm=idspr(m)
 lblj=idsp(jp,m)
 dist=xsp(jp,m)
 jptr(m)=jptr(m)+1
645 continue
c
```

```
c set print for background and border of graph
c
 do 650 i=1,103
 ip(i)=ibl
650 continue
 do 655 ii=1,53
655 it(ii)=ibl
 if (kflag.eq.0) go to 662
 do 660 i=2,102,10
660 ip(i)=iplus
 do 661 ii=2,52,10
661 it(ii)=iplus
 go to 665
662 ip(2)=idash
 ip(102)=idash
 it(2)=idash
 it(52)=idash
c
c set print for arrival times
c
665 do 710 m=1,nm
 kn=nk(m)
 jn=nj(m)
 do 700 j=1,jn
 kk=kjpt(j,m)
 if (kk.gt.kn) go to 668
 if (p(kk,j,m).ge.xlim) go to 666
 i=ifix(ta(kk,j,m)/tscale + 2.5)
 ii=(i-2)/2+2
 go to 677
666 if (kk.ne.1) go to 669
c
c prepare to interp between time=0 at ja(m) and ta(1,j,m)
c
 if (j.ne.ja(m)) go to 700
 t2=ta(1,j,m)
 if (t2.eq.0.0) go to 700
 p1=xsp(j,m)
 p2=p(1,j,m)
 t1=0.0
 go to 674
c
c prepare to interp between time=0 at jb(m) and ta(kn,j,m)
c
668 if (j.ne.jb(m)) go to 700
 t1=ta(kn,j,m)
 if (t1.eq.0.0) go to 700
 p1=p(kn,j,m)
 p2=xsp(j,m)
 t2=0.0
 go to 674
c
c prepare to interp for kk=k2=2 to kn and k1=k2-1 using ta(k1,..ta(k2,..
c
```

```
669 k2=kk
 k1=k2-1
670 t1=ta(k1,j,m)
 if (t1.ne.0.0) go to 671
 if (k1.eq.kr(j,m).or.k1.eq.kl(j,m)) go to 700
 k1=k1-1
 if (k1.lt.1) go to 700
 go to 670
c
671 t2=ta(k2,j,m)
 if (t2.ne.0.0) go to 672
 if (k2.eq.kr(j,m).or.k2.eq.kl(j,m)) go to 700
 k2=k2+1
 if (k2.gt.kn) go to 700
 go to 671
c
c nonzero pair of ta found -- test if sp j occurs between p1 and p2
c if so prepare to interp between time=0 at sp and appropriate ta
c if not prepare to interp between ta pair
c
672 p1=p(k1,j,m)
 p2=p(k2,j,m)
 if (xsp(j,m).gt.p2.or.xsp(j,m).lt.p1) go to 674
 if (xmid.le.xsp(j,m)) go to 673
 p1=xsp(j,m)
 t1=0.0
 go to 674
c
673 p2=xsp(j,m)
 t2=0.0
c
c interpolate to compute i and then set ip(i)
c
674 if(xmid.lt.p1.or.xmid.gt.p2) go to 700
 i=ifix (terp(p1,t1,p2,t2,xmid)/tscale+2.5)
 ii=(i-2)/2+2
 if (i.lt.2.or.i.gt.102) go to 700
 if (ip(i).eq.ibl.or.ip(i).eq.iplus.or.ip(i).eq.idash)
     1     ip(i)=icoln
 if(it(ii).eq.ibl.or.it(ii).eq.iplus.or.it(ii).eq.idash)
     1  it(ii)=icoln
 go to 700
677 if (i.gt.1) go to 678
 i=1
 ii=1
 go to 684
678 if (i.lt.103) go to 681
 i=103
 ii=53
684 ip(i)=isq
 it(ii)=isq
 go to 690
681 if(ip(i).ne.ibl.and.ip(i).ne.iplus.and.ip(i).ne.idash) ip(i)=isq
 if(it(ii).ne.ibl.and.it(ii).ne.iplus.and.it(ii).ne.idash)
```

```
     1 it(ii)=isq
686 lgp=lg(kk,j,m)
 if (lgp.eq.0) go to 687
 if(ip(i).ne.isq)ip(i)=il(lgp)
 if(it(ii).ne.isq)it(ii)=il(lgp)
 go to 690
687 if(ip(i).ne.isq)ip(i)=idsp(j,m)
 if(it(ii).ne.isq)it(ii)=idsp(j,m)
690 kjpt(j,m)=kjpt(j,m)+1
700 continue
710 continue
c
c write(20,a line of graph
c
 if (kflag) 725,720,730
720 if(jprt.ne.ibl)write(20,722) (ip(i),i=1,103)
722 format (1h ,8x,50a1,53a1)
 if(jtrl.ne.ibl)write(*,723)(it(ii),ii=1,53)
723 format(8x,53a1)
 go to 745
725 if(jprt.ne.ibl)write(20,727) lblm,lblj,(ip(i),i=1,103),dist
727 format (1h ,1x,a1,5x,51a1,53a1,f10.0)
 if(jtrl.ne.ibl)write(*,728)lblm,lblj,(it(ii),ii=1,53),dist
728 format(1x,a1,5x,54a1,f10.0)
 go to 745
730 if(jprt.ne.ibl)write(20,732) lblm,lk,lblj,(ip(i),i=1,103),dist
732 format (1h ,1x,a1,i3,2x,51a1,53a1,f10.0)
 if(jtrl.ne.ibl)write(*,733)lblm,lk,lblj,(it(ii),ii=1,53),dist
733 format(1x,a1,i3,2x,54a1,f10.0)
c
c increment xlim for next line and loop back unless graph is complete
c
745 if (xlim.gt.xlim2) go to 750
 xlim=xlim+xscale
 go to 627
c
c graph is complete -- write(20,bottom border
c
750 if(jprt.eq.ibl) go to 752
 write(20,585)
 write(20,584) (lb(i),i=1,11)
752 if(jtrl.eq.ibl) go to 800
 write(*,10585)
 write(*,10584)(lb(i),i=1,11,2)
800 return
c
c end of t-d plot routine.  beginning of depth plot routine.
c first find highest surface elev (emax)
c
3000 emax=-big
 do 3014 m=1,nm
 kn=nk(m)
 if (jjoff.eq.0) go to 3002
 j1=1
```

```
 j2=nj(m)
 go to 3008
3002 j1=ja(1)
 j2=jb(m)
3008 do 3010 j=j1,j2
 if (esp(j,m).gt.emax) emax=esp(j,m)
3010 continue
 do 3012 k=1,kn
 if (eg(k,m).gt.emax) emax=eg(k,m)
3012 continue
3014 continue
c
c compute elev scale labels lb(i) -- compute emin
c
 lbinc=ifix (10.0*escale+0.5)
 lmax=lbinc*(ifix (emax+escale)/lbinc+1)
 lb(1)=lmax-10*lbinc
 emin=float (lb(1))
 do 3016 i=2,11
 lb(i)=lb(i-1)+lbinc
3016 continue
c
c print heading and labels
c
 if(jprt.eq.ibl) go to 3020
 write(20,57) ident
 write(20,3018)
3018 format (1h ,60x,58x,1hs/1h ,4x,1hd,60x,50x,4hg  p/1h ,4x,1hi,
     1   60x,47x,7hs  e  r/1h ,4x,1hs,38x,
     2   32he l e v a t i o n    ( f e e t ),
     3  37x,7hp  o  e/1h ,4x,1ht,60x,53x,1ha/1h ,60x,58x,1hd)
 write(20,584) (lb(i),i=1,11)
 write(20,585)
3020 if(jtrl.eq.ibl) go to 3024
 write(*,3022)
3022 format(68x,'s'/5x,'d',59x,'g  p'/5x,'i',56x,'s  e  r'/5x,'s',13x
     1 ,'e l e v a t i o n    ( f e e t )',11x,'p  o  e'/5x,'t',62x,'a'
     2  /68x,'d')
        write(*,10584)(lb(i),i=1,11,2)
 write(*,10585)
c
c sort prg(k,j,m),erg(k,j,m) arrays into prgall(kjm),ergall(kjm) columns
c
3024 kjm=1
3030 pmin=big
 do 3034 m=1,nm
 jn=nj(m)
 kn=nk(m)
 do 3033 j=1,jn
 do 3032 k=1,kn
 if (prg(k,j,m).ge.pmin.or.erg(k,j,m).eq.0.0) go to 3032
 pmin=prg(k,j,m)
 k1=k
 j1=j
```

```
 m1=m
3032 continue
3033 continue
3034 continue
c
 if (pmin.eq.big) go to 3040
 prgall(kjm)=pmin
 ergall(kjm)=erg(k1,j1,m1)
 isymbl(kjm)=krg(k1,j1,m1)
 if (isymbl(kjm).eq.ibl) isymbl(kjm)=idsp(j1,m1)
 kjm=kjm+1
c
 erg(k1,j1,m1)=0.0
 go to 3030
c
3040 nkjm=kjm-1
c
c sort and merge 4-d arrays prs2(j,m,l,i) and ers2(j,m,l,i) into column
c arrays prsall(jml) and ersall(jml).
c
 jml=1
3050 pmin=big
 do 3056 m=1,nm
 jn=nj(m)
 do 3055 j=1,jn
 do 3054 l=1,ln
 do 3052 lr=1,2
3051 if (prs2(j,m,l,lr).gt.pmin.or.ers2(j,m,l,lr).eq.0.) go to 3052
 pmin=prs2(j,m,l,lr)
 j1=j
 m1=m
 l1=l
 i=lr
c
3052 continue
3054 continue
3055 continue
3056 continue
c
 if (pmin.eq.big) go to 3070
 prsall(jml)=pmin
 ersall(jml)=ers2(j1,m1,l1,i)
 jsymbl(jml)=krs2(j1,m1,l1,i)
 ers2(j1,m1,l1,i)=0.0
 if (jsymbl(jml).eq.ibl) jsymbl(jml)=is
 jml=jml+1
 go to 3050
c
3070 njml=jml-1
c
c set pointers to initial values -- jptr(m) for sp  labels and plot pts
c                                  kptr(m) for geo labels and plot pts
c  and initialize xlim             jml     for prsall(jml)
c  after extending xlim1 and xlim2 kjm     for prgall(kjm)
```

```
c  to range xsp(j1,1)-xsp(j2,nm)
c  or prgall(1)-prgall(nkjm), whichever is larger
c
 kn=nk(1)
 dx=xg(kn,1)-xg(1,1)
 dx3=dx+dx+dx
 j1=ja(1)
 j2=jb(nm)
 if(jjoff.eq.0) go to 3071
 j1=1
 j2=nj(nm)
3071 xlim=xsp(j1,1)
 do 13071 kjm=1,nkjm
 if(isymbl(kjm).ne.iques) go to 13072
13071 continue
 kjm=nkjm
13072 if(prgall(kjm).lt.xlim) xlim=prgall(kjm)
 if (xlim.lt.xlim1-dx3) xlim=xlim1-dx3
 if (xlim1.le.xlim) go to 3072
 xlim1=xlim+xsco2
3072 xlim=xsp(j2,nm)
 do 3073 kjm=1,nkjm
 kjmr=nkjm-kjm+1
 if(isymbl(kjmr).ne.iques) go to 13073
3073 continue
 kjmr=1
13073 if(prgall(kjmr).gt.xlim) xlim=prgall(kjmr)
 if (xlim.gt.xlim2+dx3) xlim=xlim2+dx3
 if (xlim2.lt.xlim) xlim2=xlim
c
3074 do 3080 m=1,nm
 kptr(m)=1
 if (jjoff.eq.0) go to 3076
 j=1
 go to 3077
3076 j=ja(m)
3077 if (xsp(j,m).ge.(xlim1-xscale)) go to 3078
 j=j+1
 if (j.le.nj(m)) go to 3077
 j=nj(m)
3078 jptr(m)=j
3080 continue
 jml=1
 kjm=1
 xlim=xlim1
 xmin=xlim1-xscale
 xmid=xlim-xsco2
 plast=xlim1
 plsur=plast
 elsur=esp(1,1)
 if(jjoff.ne.0) go to 3084
 jleft=ja(1)
 jrite=jb(nm)
 go to 3086
```

```
3084 jleft=1
 jrite=nj(nm)
3086 pjb=xsp(jrite,nm)
 b1=slope
 b2=slope
 do 3090 l=1,ln
 if(idip.ne.0) go to 3088
 kmid=nk(1)/2
 if(kmid.lt.6) kmid=6
 if(kmid.gt.nk(1)) kmid=nk(1)
 lcall=l
 call dip(lcall,1,1,kmid,1,a,b1)
 kmid=nk(nm)/2+1
 if(kmid.lt.7) kmid=nk(nm)-5
 if(kmid.lt.1) kmid=1
 call dip(lcall,nm,kmid,nk(nm),nk(nm),a,b2)
3088 elast(l)=ers(jleft,1,l)-b1*(xsp(jleft,1)-xlim1)
 erite(l)=ers(jrite,nm,l)+b2*(xlim2-pjb)
3090 continue
 elsur=esp(jleft,1)-slope*(xsp(jleft,1)-xlim1)
c
c main plot
loop*****************************************************************
c
3100 kflag=0
 lblj=ibl
 lblm=ibl
 do 3122 i=1,103
3122 ip(i)=ibl
 do 3123 i=1,53
3123 it(i)=ibl
 ip(2)=idash
 ip(102)=idash
 it(2)=idash
 it(53)=idash
c
c set print for printing spread and sp labels, dist, and plot pts
c
c set kflag=-1 for m and j labels and dist
c     kflag= 0 for no labels
c     kflag= 1 for m, j and k labels and dist
c
 do 3120 m=1,nm
 kn=nk(m)
 if (jjoff.eq.0) go to 3101
 jn=nj(m)
 go to 3102
3101 jn=jb(m)
3102 jp=jptr(m)
 if (jp.gt.jn) go to 3110
 if (xsp(jp,m).ge.xlim) go to 3110
 if(xsp(jp,m).lt.xmin) go to 3108
 kflag=-1
 lblm=idspr(m)
```

```
 lblj=idsp(jp,m)
 dist=xsp(jp,m)
 do 3103 i=2,102,10
3103 ip(i)=iplus
 do 3104 i=2,52,10
3104 it(i)=iplus
 elsur=esp(jp,m)
 plsur=xsp(jp,m)
 call setip(es(jp,m),emin,escale,iast)
 do 3105 l=1,ln
 call setip(ers(jp,m,l),emin,escale,icoln)
 elast(l)=ers(jp,m,l)
3105 continue
 plast=xsp(jp,m)
c
3108 jptr(m)=jptr(m)+1
 go to 3102
c
3110 kp=kptr(m)
 if (kp.gt.kn) go to 3120
 if (xg(kp,m).ge.xlim) go to 3120
 if(xg(kp,m).lt.xmin) go to 3118
 if (kflag.ne.0) go to 3115
c
 dist=xg(kp,m)
 do 3111 i=2,102,10
3111 if(ip(i).eq.ibl) ip(i)=iplus
 do 3112 i=2,52,10
3112 if(it(i).eq.ibl) it(i)=iplus
c
3115 lk=kp
 lblm=idspr(m)
 kflag=1
 e=eg(kp,m)+escale
 call setip(e,emin,escale,idee)
 elsur=eg(kp,m)
 plsur=xg(kp,m)
 do 3117 l=1,ln
 call setip(erp(kp,m,l),emin,escale,icoln)
 elast(l)=erp(kp,m,l)
3117 continue
 plast=xg(kp,m)
c
3118 kptr(m)=kptr(m)+1
 go to 3110
3120 continue
c
3124 if (jml.gt.njml) go to 3126
 if (prsall(jml).ge.xlim) go to 3126
 call setip(ersall(jml),emin,escale,jsymbl(jml))
 jml=jml+1
 go to 3124
c
3126 if (kjm.gt.nkjm) go to 3128
```

```
 if (prgall(kjm).ge.xlim) go to 3128
 call setip(ergall(kjm),emin,escale,isymbl(kjm))
 kjm=kjm+1
 go to 3126
c
c interpolate to find layer boundaries between sps and geos
c
3128 if (kflag.ne.0) go to 3171
 if(xlim.le.pjb) go to 13129
 pnext=xlim2
 do 13128 l=1,ln
13128 enext(l)=erite(l)
 pnsur=xlim2
 ensur=esp(jrite,nm)+slope*(xlim2-pjb)
 go to 3140
13129 pnext=big
 pnsur=big
 knm=nk(nm)
 do 3138 m=1,nm
 jp=jptr(m)
 kp=kptr(m)
 if (jjoff.eq.0) go to 3129
 jn=nj(m)
 go to 3130
3129 jn=jb(m)
3130 kn=nk(m)
 if (jp.gt.jn.or.kp.gt.kn) go to 3131
 if (xg(kp,m).le.xsp(jp,m)) go to 3132
 go to 3134
3131 if(kp.gt.kn.and.jp.gt.jn) go to 3138
 if(kp.gt.kn) go to 3134
 if(jp.gt.jn) go to 3132
 if(xsp(jp,m).lt.xg(kp,m)) go to 3134
3132 if (xg(kp,m).gt.pnext) go to 3138
 pnext=xg(kp,m)
 pnsur=pnext
 ensur=eg(kp,m)
 do 3133 l=1,ln
3133 enext(l)=erp(kp,m,l)
 go to 3138
c
3134 if (xsp(jp,m).gt.pnext.and.xsp(jp,m).le.xg(knm,nm)) go to 3138
 pnext=xsp(jp,m)
 pnsur=pnext
 ensur=esp(jp,m)
 do 3135 l=1,ln
3135 enext(l)=ers(jp,m,l)
3138 continue
c
3140 i1=2+ifix ((terp(plsur,elsur,pnsur,ensur,xmid)-emin)/escale+0.5)
 i2=(i1-2)/2+2
 if (i1.le.0) i1=1
 if(i2.le.0) i2=1
 if(i2.gt.53) i2=53
```

```
 if (i1.gt.103) i1=103
 if(it(i2).eq.ibl) it(i2)=icoln
 if (ip(i1).eq.ibl) ip(i1)=icoln
 do 3169 l=1,ln
 i1=2+ifix ((terp(plast,elast(l),pnext,enext(l),xmid)-emin)/escale
     1  +0.5)
 i2=(i1-2)/2+2
 if (i1.le.0) i1=1
 if(i2.le.0) i2=1
 if (i1.gt.103) i1=103
 if(i2.gt.53) i2=53
 if (ip(i1).eq.ibl) ip(i1)=icoln
 if(it(i2).eq.ibl) it(i2)=icoln
3169 continue
c
c write(20,a line of graph
c
3171 if(jprt.eq.ibl) go to 3184
 if (kflag) 3172,3176,3180
3172 write(20,3174) dist,(ip(i),i=1,103),lblj,lblm
3174 format (1h ,f8.0,50a1,53a1,1x,a1,5x,a1)
 go to 3184
3176 write(20,3178) (ip(i),i=1,103)
3178 format (1h ,8x,50a1,53a1)
 go to 3184
3180 write(20,3182) dist,(ip(i),i=1,103),lblj,lk,lblm
3182 format (1h ,f8.0,50a1,53a1,1x,a1,i3,2x,a1)
3184 if(jtrl.eq.ibl) go to 3200
 if(kflag) 3186,3190,3194
3186 write(*,3188)dist,(it(i),i=1,53),lblj,lblm
3188 format(1x,f7.0,53a1,1x,a1,5x,a1)
 go to 3200
3190 write(*,3192)(it(i),i=1,53)
3192 format(8x,53a1)
 go to 3200
3194 write(*,3196)dist,(it(i),i=1,53),lblj,lk,lblm
3196 format(1x,f7.0,53a1,1x,a1,i3,2x,a1)
c
c prepare to loop back to compute next line, or if done write(20,border
c
3200 if (xlim.gt.xlim2) go to 3210
 xmin=xlim
 xlim=xlim+xscale
 xmid=xlim-xsco2
 kn=nk(nm)
 jn=jb(nm)
 if(jjoff.ne.0) jn=nj(nm)
 go to 3100
c
3210 if(jprt.eq.ibl) go to 3212
 write(20,585)
 write(20,584) (lb(i),i=1,11)
 write(20,3018)
3212 if(jtrl.eq.ibl) go to 3214
```

```
 write(*,10585)
 write(*,10584)(lb(i),i=1,11,2)
3214 return
 end
c
c
c ***** Subroutine KENDS (#9).  written 09/06/81  1446.7rew ***********
c finds index of leftmost (k11) and rightmost (k22) geo representing
c layer l for sp j, spread m. k1 and k2 are end pts of range to be
c tested, and are input values.k11 and k22 are end pts found (output).
c both k11 and k22 set to zero if no pts found
c
 subroutine kends(l,m,j,k1,k2,k11,k22)
       common/blk0/lg
 dimension lg(48,7,5)
       k11=0
       k22=0
       if (k1.eq.0.or.k2.eq.0) go to 12
       do 1 k=k1,k2
       if (lg(k,j,m).ne.l) go to 1
       k11=k
       go to 3
1  continue
       go to 12
3  do 5 k=k11,k2
       if(lg(k,j,m).eq.l) k22=k
5  continue
12 return
       end
c
c
c ****** Subroutine HTIME (#10).  written 09/06/81  1447.8rew **********
 subroutine htime(k,j,m,p1,e1,p2,e2,hv,th)
 character*1 idspr,idsp
 common/blk2/xg,erp,slope,idip
 common/blk5/idspr,idsp
 common/ibm3/kl,kr,d
 common/blk7/ja,jb,trs,ers,xsp,esp,ls
 common/blk9/eg,es
 common/blk10/blim,itrace,tansg
 dimension xg(48,5),erp(48,5,4),ja(5),jb(5),trs(7,5,4),
     1   ers(7,5,4),xsp(7,5),esp(7,5),ls(7,5),eg(48,5),es(7,5),
     2   idspr(5),idsp(7,5),kl(7,5),kr(7,5),d(48,7,5)
c
 de=e2-e1
 dp=p2-p1
 dsx=xg(k,m)-xsp(j,m)
 dse=eg(k,m)-es(j,m)
 sh=sqrt(dsx**2+dse**2)
 d1=d(k,j,m)
 dp=sign(dp,dsx*dp*d1)
 if (dsx.eq.0..or.dp.eq.0.) go to 10
 if (abs(de/dp).gt.blim) de=blim*dp
 th=sign(d1*sqrt(dp**2+de**2)/sh,dp)/hv
```

```
 return
10 dh=d1-abs(p1-xsp(j,m))-abs(p2-xg(k,m))
 th=sign(sqrt(dh**2+de**2),dh)/hv
 return
 end
```

```
c ************************ S4.FOR ************************************
c
c Program SIPT1.FOR -- version 1.0 -- USGS, Hartford, CT -- 08-15-86
c
c Micro-computer implementation of sipt22.fortran (on Denver PRIME).
c Compiled on an IBM PC using the IBM Professional FORTRAN compiler
c (version 1.0).
c Program consists of 9 modules: MAIN.FOR, PART2.FOR, PART3.FOR,
c INITIAL.FOR, S1.FOR, S2.FOR, S3.FOR, S4.FOR and S5.FOR.
c
c ************** subroutine resumed ************************************
c
c ***** Subroutine FILLIN (#11).  written 09/06/81  1447.8rew **********
c computes missing (zero) elev and times at geos (erp and trp) and at
c sps (ers and trs) by interp or extrap of nonzero values of erp and trp
c
 subroutine fillin(l)
        character*1 ibl,jtrl,jprt,idspr,idsp,krs2,krg,jvel,idum
 common ibl
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,idum
 common/blk2/xg,erp,slope,idip
 common/blk4/vva,vha
 common/blk5/idspr,idsp
 common/ibm3/kl,kr,d
 common/blk6/trp,jjoff
 common/blk7/ja,jb,trs,ers,xsp,esp,ls
 common/blk8/prg,erg,prs2,ers2,zrsp,zrg
 common/ibm4/krg,krs2
 common/blk9/eg,es
 common/blk10/blim,itrace,tansg
 dimension ka(5),kb(5),nj(5),nk(5),xg(48,5),eg(48,5),trp(48,5,4),
     1  erp(48,5,4),trs(7,5,4),ers(7,5,4),xsp(7,5),esp(7,5),ja(5),jb(5),
     2  es(7,5),vva(5,5),vha(5,5),ls(7,5),prg(48,7,5),erg(48,7,5),
     3  krg(48,7,5),prs2(7,5,4,2),ers2(7,5,4,2),krs2(7,5,4,2),idspr(5),
     4  idsp(7,5),kl(7,5),kr(7,5),d(48,7,5),zrsp(7),zrg(48,7)
c
c first interp to fill in gaps within each spread
c
 do 20 m=1,nm
 kn=nk(m)
c search for 1st nonzero value
 do 4 k=1,kn
 if (trp(k,m,l).ne.0.0) go to 6
4 continue
 ka(m)=0
 kb(m)=0
 go to 20
c nonzero value found -- search for zero value
6 ka(m)=k
 kb(m)=k
 k1=k+1
8 if (k1.gt.kn) go to 20
 do 10 k=k1,kn
```

```
 if (trp(k,m,l).eq.0.0) go to 12
 kb(m)=k
10 continue
 go to 20
c zero value found -- store index of preceding nonzero value and search
c for next nonzero value
12 k11=k-1
 k1=k+1
 if (k1.gt.kn) go to 20
 do 14 k=k1,kn
 if (trp(k,m,l).ne.0.0) go to 16
14 continue
 go to 20
16 k22=k
 kb(m)=k
 k1=k11+1
 k2=k22-1
 do 18 k=k1,k2
 erp(k,m,l)=terp(xg(k11,m),erp(k11,m,l),xg(k22,m),erp(k22,m,l),
     1  xg(k,m))
 trp(k,m,l)=terp(xg(k11,m),trp(k11,m,l),xg(k22,m),trp(k22,m,l),
     1  xg(k,m))
18 continue
 k1=k22+1
 go to 8
20 continue
c
c connect pts between spreads
c
 if (nm.ne.1) go to 21
 m111=1
 m222=1
 go to 100
21 m=nm-1
 do 52 m1=1,m
 kn1=nk(m1)
 m2=m1+1
 kn2=nk(m2)
 k1=kb(m1)
 k2=ka(m2)
 if (k1.eq.0.or.k2.eq.0) go to 52
 if (xg(1,m2).ge.xg(kn1,m1)) go to 44
c end geos of the two spreads overlap
 if (xg(k2,m2).lt.xg(k1,m1)) go to 30
c end pts where erp is defined dont overlap -- interpolate in this intvl
 k11=k1+1
 kk1=k1
 if (k11.gt.kn1) go to 24
 do 22 k=k11,kn1
 if (xg(k,m1).gt.xg(k2,m2)) go to 24
 erp(k,m1,l)=terp(xg(k1,m1),erp(k1,m1,l),xg(k2,m2),erp(k2,m2,l),
     1  xg(k,m1))
 trp(k,m1,l)=terp(xg(k1,m1),trp(k1,m1,l),xg(k2,m2),trp(k2,m2,l),
     1  xg(k,m1))
```

```
 kk1=k
22 continue
24 k22=k2-1
 kk2=k2
 if (k22.lt.1) go to 28
 none=0
 do 26 k=1,k22
 if (xg(k,m2).lt.xg(k1,m1)) go to 26
 erp(k,m2,l)=terp(xg(k1,m1),erp(k1,m1,l),xg(k2,m2),erp(k2,m2,l),
     1  xg(k,m2))
 trp(k,m2,l)=terp(xg(k1,m1),trp(k1,m1,l),xg(k2,m2),trp(k2,m2,l),
     1  xg(k,m2))
 if (none.eq.1) go to 26
 none=1
 kk2=k
26 continue
28 k1=kk1
 kb(m1)=k1
 k2=kk2
 ka(m2)=k2
c now the end pts do overlap
c first fill in spread m1 going to the right
30 k11=k1+1
        k22=kb(m2)-1
 if (k11.gt.kn1.or.k22.lt.1) go to 37
 do 36 k=k11,kn1
 do 32 kk=k2,k22
 if (xg(k,m1).ge.xg(kk,m2).and.xg(k,m1).le.xg(kk+1,m2)) go to 34
32 continue
 go to 36
34 if (trp(kk,m2,l).eq.0.0.or.trp(kk+1,m2,l).eq.0.0) go to 36
 erp(k,m1,l)=terp(xg(kk,m2),erp(kk,m2,l),xg(kk+1,m2),
     1 erp(kk+1,m2,l),xg(k,m1))
 trp(k,m1,l)=terp(xg(kk,m2),trp(kk,m2,l),xg(kk+1,m2),
     1 trp(kk+1,m2,l),xg(k,m1))
 kb(m1)=k
36 continue
c then fill in spread m2 going to the left
37 k11=ka(m1)+1
 k22=k2-1
 if (k11.gt.kn1) go to 52
 if (k22.lt.1) go to 52
 none=0
 do 42 k=1,k22
 do 38 kk=k11,k1
 if (xg(k,m2).ge.xg(kk-1,m1).and.xg(k,m2).le.xg(kk,m1)) go to 40
38 continue
 go to 42
40 if (trp(kk-1,m1,l).eq.0.0.or.trp(kk,m1,l).eq.0.0) go to 42
 erp(k,m2,l)=terp(xg(kk-1,m1),erp(kk-1,m1,l),xg(kk,m1),
     1 erp(kk,m1,l),xg(k,m2))
 trp(k,m2,l)=terp(xg(kk-1,m1),trp(kk-1,m1,l),xg(kk,m1),
     1 trp(kk,m1,l),xg(k,m2))
 if (none.ne.0) go to 42
```

```
 none=1
 ka(m2)=k
42 continue
 go to 52
c
c end geos dont overlap -- interpolate between nonzero end pts
c first go right on spread m1
c
44 k11=k1+1
 if (k11.gt.kn1) go to 48
 do 46 k=k11,kn1
 erp(k,m1,l)=terp(xg(k1,m1),erp(k1,m1,l),xg(k2,m2),erp(k2,m2,l),
     1 xg(k,m1))
 trp(k,m1,l)=terp(xg(k1,m1),trp(k1,m1,l),xg(k2,m2),trp(k2,m2,l),
     1 xg(k,m1))
46 continue
 kb(m1)=kn1
c then go left on spread m2
48 k22=k2-1
 if (k22.lt.1) go to 52
 do 50 k=1,k22
 erp(k,m2,l)=terp(xg(k1,m1),erp(k1,m1,l),xg(k2,m2),erp(k2,m2,l),
     1 xg(k,m2))
 trp(k,m2,l)=terp(xg(k1,m1),trp(k1,m1,l),xg(k2,m2),trp(k2,m2,l),
     1 xg(k,m2))
50 continue
 ka(m2)=1
52 continue
c
c fill in intermediate spreads with no depth pts defined
c
 do 54 m1=1,nm
 if (ka(m1).ne.0) go to 58
54 continue
 if(jprt.ne.ibl)write(20,56) l
 if(jtrl.ne.ibl)write(*,56)l
56 format (/1x,'no depth points defined for base of layer',i2,
     1 17h--quit in fillin )
 l=0
 go to 200
58 m111=m1
60 m11=m1+1
 if (m11.ge.nm) go to 80
 do 62 mm=m11,nm
 if (ka(mm).eq.0) go to 64
 m1=mm
62 continue
c no such spreads occur
 go to 80
64 mm=mm+1
 do 66 m2=mm,nm
 if (ka(m2).ne.0) go to 68
66 continue
 go to 80
```

```
c such a spread does occur
68 k1=kb(m1)
 k2=ka(m2)
 m11=m1+1
 m22=m2-1
 do 72 mm=m11,m22
 kn=nk(mm)
 do 70 k=1,kn
 erp(k,mm,l)=terp(xg(k1,m1),erp(k1,m1,l),xg(k2,m2),erp(k2,m2,l),
     1 xg(k,mm))
 trp(k,mm,l)=terp(xg(k1,m1),trp(k1,m1,l),xg(k2,m2),trp(k2,m2,l),
     1 xg(k,mm))
70 continue
 ka(mm)=1
 kb(mm)=kn
72 continue
 m1=m2
 go to 60
c
c fill in intermediate spreads with only one depth pt defined
c
80 do 88 m=m111,nm
 kn=nk(m)
 if (ka(m).eq.0) go to 90
 kk=ka(m)
 if (kk.ne.kb(m)) go to 88
 if (m.eq.m111) go to 84
c do to the left of pt
 m1=m-1
 k1=kb(m1)
 k2=kk-1
 if (k2.lt.1) go to 84
 do 82 k=1,k2
 erp(k,m,l)=terp(xg(k1,m1),erp(k1,m1,l),xg(kk,m),erp(kk,m,l),
     1 xg(k,m))
 trp(k,m,l)=terp(xg(k1,m1),trp(k1,m1,l),xg(kk,m),trp(kk,m,l),
     1 xg(k,m))
82 continue
 ka(m)=1
c do to the right of pt
84 m2=m+1
 if (m2.gt.nm) go to 88
 if (ka(m2).eq.0) go to 92
 k1=kk+1
 if (k1.gt.kn) go to 88
 k2=ka(m2)
 do 86 k=k1,kn
 erp(k,m,l)=terp(xg(kk,m),erp(kk,m,l),xg(k2,m2),erp(k2,m2,l),
     1   xg(k,m))
 trp(k,m,l)=terp(xg(kk,m),trp(kk,m,l),xg(k2,m2),trp(k2,m2,l),
     1  xg(k,m))
86 continue
 kb(m)=kn
88 continue
```

```
 m222=nm
 go to 100
90 m=m-1
92 m222=m
c
c extrapolate end geos and end spreads
c left of spread
c
100 kmid=(kb(m111)-ka(m111))/2
 if(kmid-ka(m111).lt.5) kmid=ka(m111)+5
 if(kmid.gt.kb(m111)) kmid=kb(m111)
 call dip(l,m111,ka(m111),kmid,ka(m111),a1,b1)
 k1=ka(m111)-1
 if (k1.lt.1) go to 102
 call extrp(l,m111,1,k1,a1,b1,vva(m111,l))
102 if (m111.eq.1) go to 106
 m11=m111-1
 do 104 m=1,m11
 mcall=m
 call extrp(l,mcall,1,nk(m),a1,b1,vva(m,l))
104 continue
c right of spread
106 kmid=(kb(m222)-ka(m222))/2+1
 if(kb(m222)-kmid.lt.5) kmid=kb(m222)-5
 if(kmid.lt.ka(m222)) kmid=ka(m222)
 call dip(l,m222,kmid,kb(m222),kb(m222),a2,b2)
 kn=nk(m222)
 k2=kb(m222)+1
 if (k2.gt.kn) go to 108
 call extrp(l,m222,k2,kn,a2,b2,vva(m222,l))
108 if (m222.eq.nm) go to 112
 m22=m222+1
 do 110 m=m22,nm
 mcall=m
 call extrp(l,mcall,1,nk(m),a2,b2,vva(m,l))
110 continue
c
c interp-extrap elev and time at sps and prevent criss-cross of layers
c
112 do 160 m=1,nm
 if (jjoff.eq.0) go to 113
 j1=1
 j2=nj(m)
 go to 116
113 j1=ja(m)
 j2=jb(m)
116 kn=nk(m)
 do 146 j=j1,j2
 mm=m
 knm=nk(m)
 if (trs(j,m,1).ne.0.0) go to 142
 if (xsp(j,m).ge.xg(1,mm)) go to 118
117 if (mm.eq.1) go to 136
 mm=mm-1
```

```
 knm=nk(mm)
 if (xsp(j,m).lt.xg(1,mm)) go to 117
 go to 122
118 if (xsp(j,m).le.xg(knm,mm)) go to 122
119 if (mm.eq.nm) go to 138
 mm=mm+1
 knm=nk(mm)
 if (xsp(j,m).gt.xg(knm,mm)) go to 119
c
122 if (xg(2,mm).ge.xsp(j,m)) go to 128
 if (xg(knm-1,mm).le.xsp(j,m)) go to 126
 do 124 k=2,knm
 if (xg(k,mm).ge.xsp(j,m)) go to 134
124 continue
c
126 k1=knm-1
 x1=xg(k1,mm)
 e1=erp(k1,mm,1)
 x2=xg(knm,mm)
 e2=erp(knm,mm,1)
 go to 140
c
128 x1=xg(1,mm)
 e1=erp(1,mm,1)
 x2=xg(2,mm)
 e2=erp(2,mm,1)
 go to 140
c
134 k1=k-1
 x1=xg(k1,mm)
 e1=erp(k1,mm,1)
 x2=xg(k,mm)
 e2=erp(k,mm,1)
 go to 140
c
c case where mm=1
c
136 kmid=knm/2
 if(kmid.lt.6) kmid=6
 if(kmid.gt.knm) kmid=knm
 call dip(1,1,1,kmid,1,a1,b1)
 if(jjoff.eq.0.or.j.ge.ja(1)) go to 7136
 e1=0.
 jj2=j-1
 if(jj2.lt.1) go to 2136
 do 1136 jj=1,jj2
 jr=jj2-jj+1
 if(ers2(jr,1,1,1).eq.0) go to 1136
 x1=prs2(jr,1,1,1)
 e1=ers2(jr,1,1,1)
 go to 2136
1136 continue
2136 jj1=j
 jj2=ja(1)-1
```

```
3136 do 4136 jj=jj1,jj2
 if(ers2(jj,1,l,1).eq.0.) go to 4136
 x2=prs2(jj,1,l,1)
 e2=ers2(jj,1,l,1)
 jlast=jj
 go to 6136
4136 continue
 if(e1.eq.0.) go to 7136
5136 x2=xg(1,1)
 e2=erp(1,1,l)
 go to 140
6136 if(e1.ne.0) go to 140
 x1=x2
 e1=e2
 jj1=jlast+1
 if(jj1.le.jj2) go to 3136
 go to 5136
7136 ers(j,m,l)=erp(1,mm,l)-b1*(xg(1,mm)-xsp(j,m))
 go to 142
c
c case where mm=nm
c
138 kmid=knm/2+1
 if(knm-kmid.lt.5) kmid=knm-5
 if(kmid.lt.1) kmid=1
 call dip(l,nm,kmid,knm,knm,a2,b2)
 if(jjoff.eq.0.or.j.le.jb(nm)) go to 7138
 e2=0.
 jn=nj(nm)
 jj1=j+1
 if(jj1.gt.jn) go to 2138
 do 1138 jj=jj1,jn
 if(ers2(jj,nm,l,2).eq.0.) go to 1138
 x2=prs2(jj,nm,l,2)
 e2=ers2(jj,nm,l,2)
 go to 2138
1138 continue
2138 jj2=j
 jj1=jb(nm)+1
3138 do 4138 jj=jj1,jj2
 jr=jj2-jj+jj1
 if(ers2(jr,nm,l,2).eq.0.) go to 4138
 x1=prs2(jr,nm,l,2)
 e1=ers2(jr,nm,l,2)
 jlast=jr
 go to 6138
4138 continue
 if(e2.eq.0.) go to 7138
5138 x1=xg(knm,nm)
 e1=erp(knm,nm,l)
 go to 140
6138 if(e2.ne.0.) go to 140
 x2=x1
 e2=e1
```

```
 jj2=jlast-1
 if(jj1.le.jj2) go to 3138
 go to 5138
7138 ers(j,m,l)=erp(knm,mm,l)+b2*(xsp(j,m)-xg(knm,mm))
 go to 142
140 ers(j,m,l)=terp(x1,e1,x2,e2,xsp(j,m))
142 if (l.eq.1) go to 144
 if (ers(j,m,l).le.ers(j,m,l-1)) go to 143
 ers(j,m,l)=ers(j,m,l-1)-.00001
 trs(j,m,l)=0.00001
 go to 146
143 trs(j,m,l)=(ers(j,m,l-1)-ers(j,m,l))/vva(m,l)
 go to 146
144 if (ers(j,m,l).le.es(j,m)) go to 145
 ls(j,m)=2
 trs(j,m,l)=0.00001
 if (ers(j,m,l).gt.esp(j,m)) ers(j,m,l)=esp(j,m)-.00001
 go to 146
145 trs(j,m,l)=(es(j,m)-ers(j,m,1))/vva(m,l)
146 continue
 if (l.eq.1) go to 150
 do 148 k=1,kn
 if (erp(k,m,l-1)-erp(k,m,l).ge.0.00001) go to 148
 erp(k,m,l)=erp(k,m,l-1)-0.00001
 trp(k,m,l)=0.00001
148 continue
 go to 160
c
150 do 152 k=1,kn
 if (eg(k,m)-erp(k,m,l).ge.0.00001) go to 152
 trp(k,m,l)=0.00001
 erp(k,m,l)=eg(k,m)-0.00001
152 continue
c
160 continue
c
200 if(itrace.eq.0.or.jprt.eq.ibl) go to 300
 do 260 m=1,nm
 kn=nk(m)
 l2=l+1
 if(jjoff.ne.0) go to 206
 j1=ja(m)
 j2=jb(m)
 go to 208
206 j1=1
 j2=nj(m)
208 write(20,210) l2,idspr(m)
210 format(/' output of subr fillin for layer',i2,'  spread',a1)
 do 230 j=j1,j2
 write(20,220) idsp(j,m),xsp(j,m),esp(j,m),ers(j,m,l),trs(j,m,l)
220 format('  sp=',1x,a1,'  xsp=',f8.1,'  esp=',f8.1,'  ers=',f12.4,
     1 '  trs=',f12.4)
230 continue
 do 250 k=1,kn
```

```
 write(20,240) k,xg(k,m),eg(k,m),erp(k,m,l),trp(k,m,l)
240 format('  k=',i2,'   xg=',f8.1,'   eg=',f8.1,'  erp=',f12.4,
     1 '  trp=',f12.4)
250 continue
260 continue
300 return
 end
```

```
c ************************ S5.FOR ************************************
c
c Program SIPT1.FOR -- version 1.0 -- USGS, Hartford, CT -- 08-15-86
c
c Micro-computer implementation of sipt22.fortran (on Denver PRIME).
c Compiled on an IBM PC using the IBM Professional FORTRAN compiler
c (version 1.0).
c Program consists of 9 modules: MAIN.FOR, PART2.FOR, PART3.FOR,
c INITIAL.FOR, S1.FOR, S2.FOR, S3.FOR, S4.FOR and S5.FOR.
c
c ************* subroutines resumed ************************************
c
c ***** Subroutine EXTRP (#12).  written 09/06/81  1447.8rew ***********
c computes erp and trp between geos k1 and k2 using equation consts a,b
c
 subroutine extrp(l,m,k1,k2,a,b,vv)
 common/blk2/xg,erp,slope,idip
 common/blk6/trp,jjoff
 common/blk9/eg,es
 dimension xg(48,5),erp(48,5,4),trp(48,5,4),eg(48,5),es(7,5)
c
 do 2 k=k1,k2
 erp(k,m,l)=a+b*xg(k,m)
 if (l.eq.1) go to 1
 trp(k,m,l)=(erp(k,m,l-1)-erp(k,m,l))/vv
 go to 2
1 trp(k,m,l)=(eg(k,m)-erp(k,m,l))/vv
2 continue
 return
 end
c
c
c ****** Subroutine ELCOR (#13).
************************************written
c computes time corr (tc) and x corr (xc) for surf-to-datum slant rays
c for layer 1, given x pos of shot or geo (xsg), elev of shot or geo
c (esg), intcpt and slope of datum (ad and bd), and sign of boa=cot i
c (i=1 for plus, 2 for minus). xint,eint is coord of intersection of
c ray and datum.
c
 subroutine elcor(tani,vv,hv,xsg,esg,ad,bd,i,tc,xc,xint,eint)
 character*1 jtrl,jprt,jvel,idum
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,idum
 dimension nj(5),nk(5)
c
 bi=tani
 if(i.eq.2) bi=-bi
 boa=(1.-bi*bd)/(-bi-bd)
 aoa=esg-boa*xsg
 xint=(ad-aoa)/(boa-bd)
 eint=ad+bd*xint
 oa=sqrt ((xsg-xint)**2+(esg-eint)**2)
 tc=oa/vv
```

```
 xc=oa*vv/hv
 if (esg.le.eint) go to 4
3 tc=-tc
 xc=-xc
4 return
 end
c
c
c ***** Subroutine DIP (#14).  written 09/06/81  1447.8rew *************
c computes equation constants a and b (y=a+bx) for regression line
c fitted to erp over geos k1 to k2 with line passing through point
c xg(kk,m),erp(kk,m,l)
c
 subroutine dip(l,m,k1,k2,kk,a,b)
 common/blk2/xg,erp,slope,idip
 dimension xg(48,5),erp(48,5,4)
c
 if (idip.eq.0.and.xg(k1,m).lt.xg(k2,m)) go to 1
 a=erp(kk,m,l)
 b=slope
 go to 4
1 sum1=0.0
 sum2=0.0
 do 2 k=k1,k2
 sum1=sum1+xg(k,m)
 sum2=sum2+1.0
2 continue
 xbar=sum1/sum2
 sum1=0.0
 sum2=0.0
 do 3 k=k1,k2
 xd=xg(k,m)-xbar
 sum1=sum1+xd*erp(k,m,l)
 sum2=sum2+xd**2
3 continue
 b=sum1/sum2
 a=erp(kk,m,l)-b*xg(kk,m)
4 return
 end
c
c
c ****** Subroutine AVG (#15).  written 09/06/81  1447.8rew *************
c computes avg coord (pa,ea) of all pts in arrays (prg,erg) and
c (prs2,ers2) whose x position is ge x1 .and. lt x2. and whose refractor
c is layer l2. if pts.le.2.and.l2.gt.2 interval is expanded by
c dx=x2-x1 on each side.
c
 subroutine avg(x1,x2,l2,pa,ea)
 character*1 ibl,jtrl,jprt,krs2,krg,jvel,idum
 common ibl
 common/blk0/lg
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,idum
 common/blk8/prg,erg,prs2,ers2,zrsp,zrg
```

```
 common/ibm4/krg,krs2
 dimension lg(48,7,5),nj(5),nk(5),prg(48,7,5),erg(48,7,5),
     1  krg(48,7,5),prs2(7,5,4,2),ers2(7,5,4,2),krs2(7,5,4,2),zrsp(7),
     1  zrg(48,7)
c
 sum1=0.0
 sum2=0.0
 pts=0.0
 l1=l2-1
 x11=x1
 x22=x2
 dx=0.0
c
1 do 14 m=1,nm
 jn=nj(m)
 kn=nk(m)
 do 12 j=1,jn
 i=1
2 if (ers2(j,m,l1,i).eq.0..or.krs2(j,m,l1,i).ne.ibl) go to 3
 e1=ers2(j,m,l1,i)
 p1=prs2(j,m,l1,i)
 go to 10
3 if (i.eq.2) go to 5
4 i=2
 go to 2
5 k=1
6 if (lg(k,j,m).ne.l2.or.erg(k,j,m).eq.0.0.or.krg(k,j,m).ne.ibl)
     1  go to 8
 e1=erg(k,j,m)
 p1=prg(k,j,m)
 i=3
 go to 10
8 k=k+1
 if (k.gt.kn) go to 12
 go to 6
10 if (p1.lt.x11.or.p1.ge.x22) go to 11
 sum1=sum1+p1
 sum2=sum2+e1
 pts=pts+1.0
11 go to (4,5,8),i
12 continue
14 continue
 if (pts.eq.0.0) go to 18
 if (pts.le.2..and.l2.gt.2.and.x1.eq.x11) go to 19
 pa=sum1/pts
 ea=sum2/pts
16 return
18 pa=0.0
 ea=0.0
 go to 16
19 dx=x22-x11
 x11=x11-dx
 x22=x22+dx
 go to 1
```

```
 end
c
c
c ***** Subroutine ADMIG (#16).  written 09/06/81  1447.8rew ***********
c computes tan of avg angle of dip (b) of bottom of l1 for spread m
c by regression of points (prs2,ers2) and (prg,erg).
c
 subroutine admig(l1,m,b)
 character*1 ibl,jtrl,jprt,idspr,idsp,krs2,krg,jvel,idum
 common ibl
 common/blk0/lg
 common/blk1/nm,nj,nk
 common/ibm2/jtrl,jprt,jvel,idum
 common/blk2/xg,erp,slope,idip
 common/blk5/idspr,idsp
 common/ibm3/kl,kr,d
 common/blk6/trp,jjoff
 common/blk7/ja,jb,trs,ers,xsp,esp,ls
 common/blk8/prg,erg,prs2,ers2,zrsp,zrg
 common/ibm4/krg,krs2
 common/blk9/eg,es
 common/blk10/blim,itrace,tansg
 dimension nj(5),nk(5),kl(7,5),kr(7,5),lg(48,7,5),ls(7,5),
     1  prs2(7,5,4,2),ers2(7,5,4,2),krs2(7,5,4,2),prg(48,7,5),
     2  erg(48,7,5),idspr(5),idsp(7,5),ers(7,5,4),xsp(7,5),esp(7,5),
     3  d(48,7,5),krg(48,7,5),trp(48,5,4),ja(5),jb(5),trs(7,5,4),
     4   xg(48,5),eg(48,5),es(7,5),erp(48,5,4),zrsp(7),zrg(48,7)
c
 b=0.
 jn=nj(m)
 kn=nk(m)
 l0=l1-1
 l2=l1+1
c
c compute average prg and prs2 (xbar)
c
 s1=0.
 s2=0.
 p1=0.
 p2=0.
c
 do 22 j=1,jn
 jcall=j
c
c up-right and down-left rays
c
 if(kr(j,m).eq.0) go to 18
 call kends(l2,m,jcall,kr(j,m),kn,k11,k22)
 if(k11.eq.0) go to 18
 do 17 k=k11,k22
 if(lg(k,j,m).ne.l2) go to 17
 s1=s1+prg(k,j,m)
 p1=p1+1.
17 continue
```

```
 if((jjoff.eq.0.and.j.lt.ja(m)).or.krs2(j,m,l1,1).ne.ibl) go to 18
 s2=s2+prs2(j,m,l1,1)
 p2=p2+1.
c
c up-left and down-right rays
c
18 if(kl(j,m).eq.0) go to 22
 call kends(l2,m,jcall,1,kl(j,m),k11,k22)
 if(k11.eq.0) go to 22
 do 20 k=k11,k22
 if(lg(k,j,m).ne.l2) go to 20
 s2=s2+prg(k,j,m)
 p2=p2+1.
20 continue
 if((jjoff.eq.0.and.j.gt.jb(m)).or.krs2(j,m,l1,2).ne.ibl) go to 22
 s1=s1+prs2(j,m,l1,2)
 p1=p1+1.
22 continue
c
 if((p1+p2).lt.2.) go to 99
 xbar1=0.
 xbar2=0.
 if(p1.ne.0.) xbar1=s1/p1
 if(p2.ne.0.) xbar2=s2/p2
c
c compute average dip (b)
c
 s1=0.
 s2=0.
 ss1=0.
 ss2=0.
 p1=0.
 p2=0.
 do 52 j=1,jn
 jcall=j
c
c up-right and down-left rays
c
 if (kr(j,m).eq.0) go to 48
 call kends(l2,m,jcall,kr(j,m),kn,k11,k22)
 if (k11.eq.0) go to 48
 do 47 k=k11,k22
 if(lg(k,j,m).ne.l2.or.xbar1.eq.0.) go to 47
 x=prg(k,j,m)-xbar1
 s1=s1+x*erg(k,j,m)
 ss1=ss1+x**2
 p1=p1+1.
47 continue
 if((jjoff.eq.0.and.j.lt.ja(m)).or.krs2(j,m,l1,1).ne.ibl.or.
     1 xbar2.eq.0.) go to 48
 x=prs2(j,m,l1,1)-xbar2
 s2=s2+x*ers2(j,m,l1,1)
 ss2=ss2+x**2
 p2=p2+1.
```

```
c
c up-left and down-right rays
c
48 if(kl(j,m).eq.0) go to 52
 call kends(l2,m,jcall,1,kl(j,m),k11,k22)
 if(k11.eq.0) go to 52
 do 50 k=k11,k22
 if(lg(k,j,m).ne.l2.or.xbar2.eq.0.) go to 50
 x=prg(k,j,m)-xbar2
 s2=s2+x*erg(k,j,m)
 ss2=ss2+x**2
 p2=p2+1.
50 continue
 if((jjoff.eq.0.and.j.gt.jb(m)).or.krs2(j,m,l1,2).ne.ibl.or.
     1 xbar1.eq.0.) go to 52
 x=prs2(j,m,l1,2)-xbar1
 s1=s1+x*ers2(j,m,l1,2)
 ss1=ss1+x**2
 p1=p1+1.
52 continue
c
        if((p1+p2).lt.2.) go to 99
 r1=0.
 r2=0.
 if(ss1.ne.0.) r1=s1/ss1
 if(ss2.ne.0.) r2=s2/ss2
 b=(p1*r1+p2*r2)/(p1+p2)
 if(idip.ne.0) b=slope
 if(b.gt.blim) b=blim
 if(b.lt.-blim) b=-blim
c
c migrate ray end points
c
 b1=b
 if(l1.eq.1) b1=(b-slope)/(1.+b*slope)
 cosa=1./sqrt(1.+b1**2)
 zrk=(1.+b1*tansg)*cosa-1.
 xrk=tansg-cosa*(tansg-b1)
 zlk=(1.-b1*tansg)*cosa-1.
 xlk=tansg-cosa*(tansg+b1)
c
c up-right and down-left rays
c
 do 72 j=1,jn
 jcall=j
 if(kr(j,m).eq.0) go to 68
 call kends(l2,m,jcall,kr(j,m),kn,k11,k22)
 if(k11.eq.0) go to 68
 do 67 k=k11,k22
 if(lg(k,j,m).ne.l2.or.xbar1.eq.0.) go to 67
 erg(k,j,m)=erg(k,j,m)-zrg(k,j)*zrk
 prg(k,j,m)=prg(k,j,m)+zrg(k,j)*xrk
67 continue
 if(xbar2.eq.0..or.krs2(j,m,l1,1).ne.ibl) go to 68
```

```
 ers2(j,m,l1,1)=ers2(j,m,l1,1)-zrsp(j)*zlk
 prs2(j,m,l1,1)=prs2(j,m,l1,1)-zrsp(j)*xlk
c
c up-left and down-right rays
c
68 if(kl(j,m).eq.0) go to 72
 call kends(l2,m,jcall,1,kl(j,m),k11,k22)
 if(k11.eq.0) go to 72
 do 70 k=k11,k22
 if(lg(k,j,m).ne.l2.or.xbar2.eq.0) go to 70
 erg(k,j,m)=erg(k,j,m)-zrg(k,j)*zlk
 prg(k,j,m)=prg(k,j,m)-zrg(k,j)*xlk
70 continue
 if(xbar1.eq.0..or.krs2(j,m,l1,2).ne.ibl) go to 72
 ers2(j,m,l1,2)=ers2(j,m,l1,2)-zrsp(j)*zrk
 prs2(j,m,l1,2)=prs2(j,m,l1,2)+zrsp(j)*xrk
72 continue
99 return
 end
```