T-2977

Acoustic Tomography in Boreholes using an Algebraic Reconstruction Technique.

by

Kingsley L. Smith

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Geophysics).

Golden, Colorado

Date 4 Oct. 1984

Signed: Kingsley L. Smith

Kingsley L. Smith

Approved: Norman Bleistein

Dr. Norman Bleistein,

Thesis Advisor

Golden, Colorado

Date 8 Oct 84

P R Romig

Dr. Phillip R. Romig,

Head of the Geophysics Department

ABSTRACT

An algebraic reconstruction technique (ART) is described for the seismic tomography of velocities from the travel times of a multiple offset vertical seismic profile. ART concentrates on the production of a reconstructed field whose projected data (travel times) agree with the observed data. This reconstructed field is modified by altering the data for each ray such that when this data is back-projected, the new image agrees with the original data. Because the paths of the rays must be known to calculate the expected travel times, the problem is linearized by using raypath approximations as determined from either a constant or a linear c(z) velocity medium.

Imaging of synthetic data revealed that the orientation of the anomaly affects both the rate of convergence and the resolution of the reconstructed field. Some smoothing of the velocity anomalies occurred along the direction of the rays.

Noisy data sets developed problems in the reconstructed velocity field. Huge single point anomalies appeared along the model's edge in the reconstructed image. Elimination of these errant anomalies was necessary to obtain a reasonable velocity reconstruction. Because isotropy was assumed, the algorithm poorly reconstructed data which was collected over the strongly anisotropic Pierre Shale.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

NOTATION

α	Angle formed by the length of the raypath.
c_o	Velocity at the surface.
$c(z)$	Velocity.
d^{mn}	Sum of chord lengths.
e_{ij}^{mn}	Residual error time for chord length.
E_{ij}	Residual error time for entire ray.
i	Angle of incidence.
ij	Subscript - from the i'th source to the j'th receiver.
l_{ij}^{mn}	Chord length.
L_{ij}	Total length of the ray.
mn	Superscript for parameter circle around point m,n.
m_o	Slope of the linear c(z) velocity.
p	Ray parameter.
r	Radius of any parameter circle.
R	Radius of curvature for the ray.
s^{mn}	Slowness matrix.
$\tilde{s}^{mn}$	Slowness correction matrix.
t_{ij}^{mn}	Residence time.
T_{ij}	Calculated travel time for the ray ij.
$\tilde{T}_{ij}$	Observed travel time for the ray ij.
w_{ij}^{mn}	Weighting factor.
x_i, z_j	Offset to source, depth to receiver.

x_m, z_n	Offset and depth to parameter point mn.
x_o, z_o	Origin of the radius of curvature for the ray.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to Dr. Norman Bleistein for his help as thesis advisor as well as for his support and encouragement of this study. Also, I wish to thank Dr. James E. White and Dr. Frank Hadsell for serving on my thesis committee.

1. INTRODUCTION

One of the many goals of vertical seismic profiling (VSP) is to determine the lateral variations which occur away from the borehole. These lateral changes may reflect porosity, permeability, or facies changes within the rock unit. Such occurrences greatly influence the migration and accumulation of petroleum. Therefore, methods are needed to delineate these discrepancies between the borehole wall and the surrounding medium.

This investigation was geared at determining velocity inhomogeneities using only the first arrival times from a multiple offset VSP. The travel time through any medium is the integral of the slowness (reciprocal of the velocity) along the raypath. Unfortunately, this raypath itself is dependent upon the velocities within the medium. Thus, as posed, the problem is non-linear.

Linearizing this problem requires that the raypaths are known, which, in turn, necessitates a prior knowledge of the velocity structure. However, if we assume some general velocity trend for the medium, we can approximate the paths which the rays take. Then, the problem becomes one of finding the velocity perturbation from the assumed background velocity. Furthermore, we assume that all raypaths are not influenced by this perturbation velocity; this assumption holds true for small perturbations. Since the rays are independent of the velocity perturbation, the problem becomes linear.

This research utilized two background models:

1) Constant velocity medium in which the rays form straight lines from the source to the receiver.

2) Linear c(z) velocity medium where the raypaths follow arcs along a circle.

The residual time is defined as the observed time minus the expected time as calculated from the velocity model. Fawcett (1983) utilized a Radon transform method for his seismic tomography of the slowness field from reflected residual travel times. Neumann (1981) used a least-squares inversion of residual travel times in his study on reflection seismics. Christofferson and Husebye (1979) applied a least-squares approach on P-wave residual times in a three-dimensional case. In contrast, McMechan (1983) and Mason (1981) opted for an algebraic reconstruction technique (ART). I have opted for the last approach, as well.

In comparision to a least-squares inversion, ART possesses several advantages:

1) ART programs are computationally faster and they are easier to program.

2) Constraints are easily incorporated into the program to accommodate any prior knowledge of the medium.

3) ART can be easily applied to any source/receiver geometry without difficulties.

ART has its origins within the medical field. Gordon, Bender, and Herman (1970) first introduced ART for the reconstruction of images from x-ray pictures. Later, Herman, Lent, and Rowland (1973) improved the early ART algorithms and Gordon (1974) summarized the various ART developments within the medical profession.

Several authors worked on reconstruction techniques which are applicable to the more general reconstruction problem rather than the specific x-ray case. Mersereau and Oppenheim (1974) and Mueller, Kaveh, and Wade (1979) applied ART to density reconstruction problems using the Fourier domain. Horn (1978) developed a density reconstruction algorithm for any arbitrary ray sampling scheme.

Geophysical tomography is relatively new. Dines and Lytle (1979) reconstructed pictures of electromagnetic properties in the region between a pair of boreholes. By assuming only straight raypaths, seismic velocity structures were obtained by both Mason (1981) and McMechan (1983). Mason concerned himself with inverting seismic data shot between two boreholes, while McMechan imaged data shot from a borehole to a second borehole as well as to the surface.

The VSP model used in this paper consisted of ten point sources, which were equally spaced at 100 foot intervals along the surface. Similarly, a 100 foot spacing was used for the ten receivers placed down the borehole. No source or receiver resided at the top of the borehole (see Fig. 1). For a coordinate system, whose origin rested at the surface expression of the borehole, the x variable coincided with the surface offset distance and the z variable reflected the depth below the surface.

2. ALGEBRAIC RECONSTRUCTION TECHNIQUE - ART

The ART algorithm used here was adapted from Mason (1981). It involves a nine step process:

1) The medium is modeled by casting a set of overlapping circles on a uniform grid.

2) The total length of ray segments about a parameter is the sum of all the chord lengths formed by rays which pass through any given parameter circle.

3) The travel times associated with each of these chord lengths are determined by the velocity background model and summed.

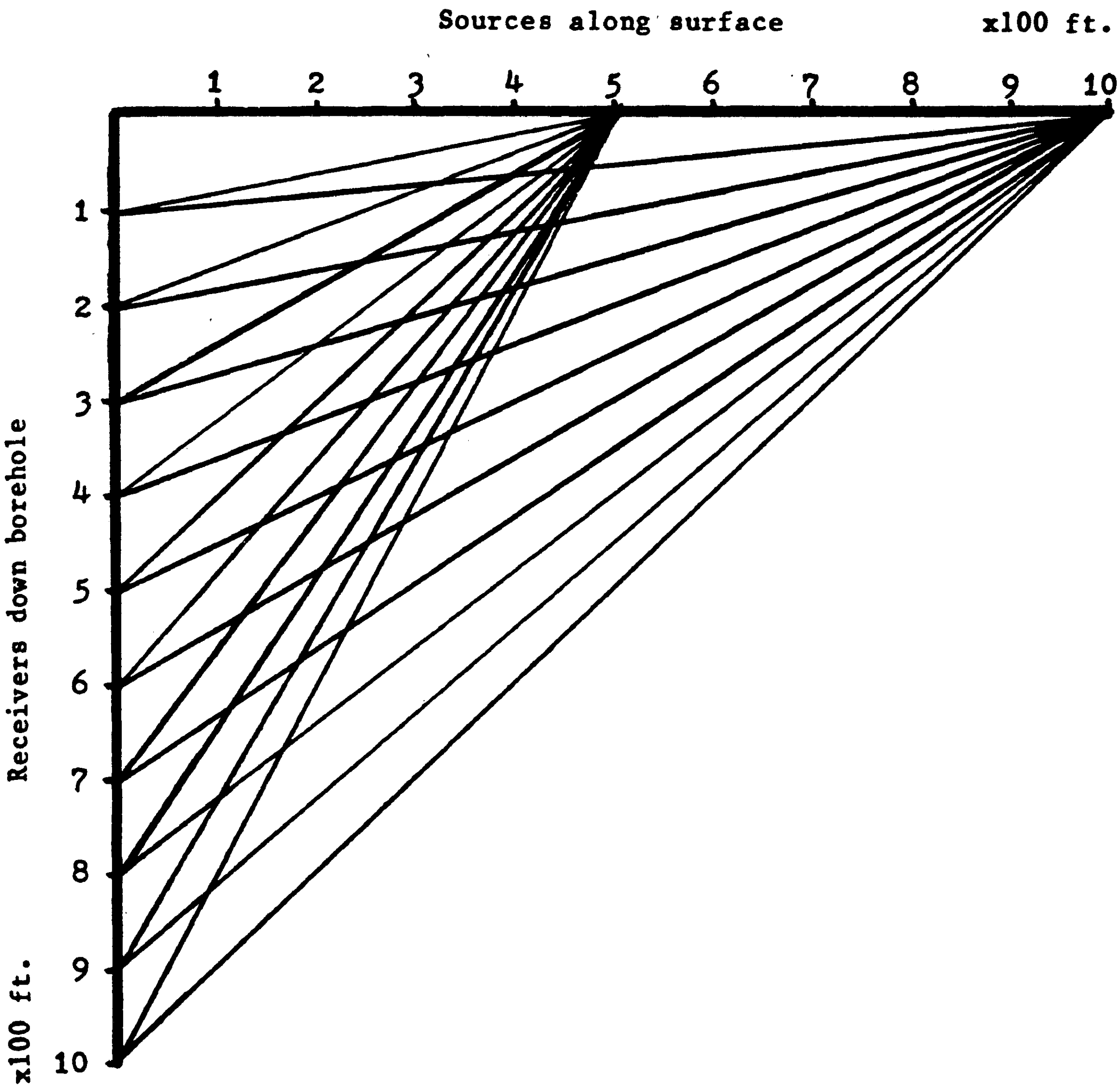


Figure 1. Multiple offset VSP geometry.

4) The average slowness at the center of a parameter circle is the difference of 2) and 3). These outputs form a matrix indexed by mn.

5) Using the generated slowness matrix from 4), the expected travel times are calculated.

6) The residual time is the observed minus the calculated travel time.

7) This residual error time is equally distributed along the entire length of the ray.

8) A slowness correction matrix is computed in the manner of steps 2), 3), and 4).

9) Steps 5) through 8) are iterated.

2.1 Constant background

As stated earlier, this investigation used two different velocity backgrounds for the medium. First the constant velocity case will be discussed before going into the more complicated linear c(z) medium.

Using a constant velocity reference, the raypaths become straight lines from the i'th source to the j'th receiver. Denote the length of this ray as L_{ij}. Because the ray follows a straight course from $(x_i,0)$ to $(0,z_j)$, the raypath is represented by the equation:

$$z = -\left[\frac{z_j}{x_i}\right] x + z_j. \tag{1}$$

Over the two-dimensional medium, a set of overlapping circles is drawn, whose centers form an uniform square grid (Fig. 2). These circles must completely cover the medium such that every point within the medium is contained in at least one of these parameter circles. The average slowness within any circle is assumed to be the slowness at the origin of the circle. These circles serve as a truncated least-distance averaging method which lies at the basis of Horn's algorithm (1978). Also, circles prove to be more convenient than a grid of squares.

Sources are positioned at the center of some, or all, of the circles along the surface and receivers are located at the origin of some, or all, of the circles at the borehole edge of the model. Clearly, the source or receiver spacing defines the largest possible grid size, or equivalently, the density of the parameter circles. Resolution is lost with too small of a density of parameter circles. Conversely, a greater grid density will increase the resolution only to the limit of the data.

Let the known value r denote the radius of any parameter circle. Then the circle, around the point (x_m, z_n), is mathematically expressed as:

$$(x - x_m)^2 + (z - z_n)^2 = r^2. \tag{2}$$

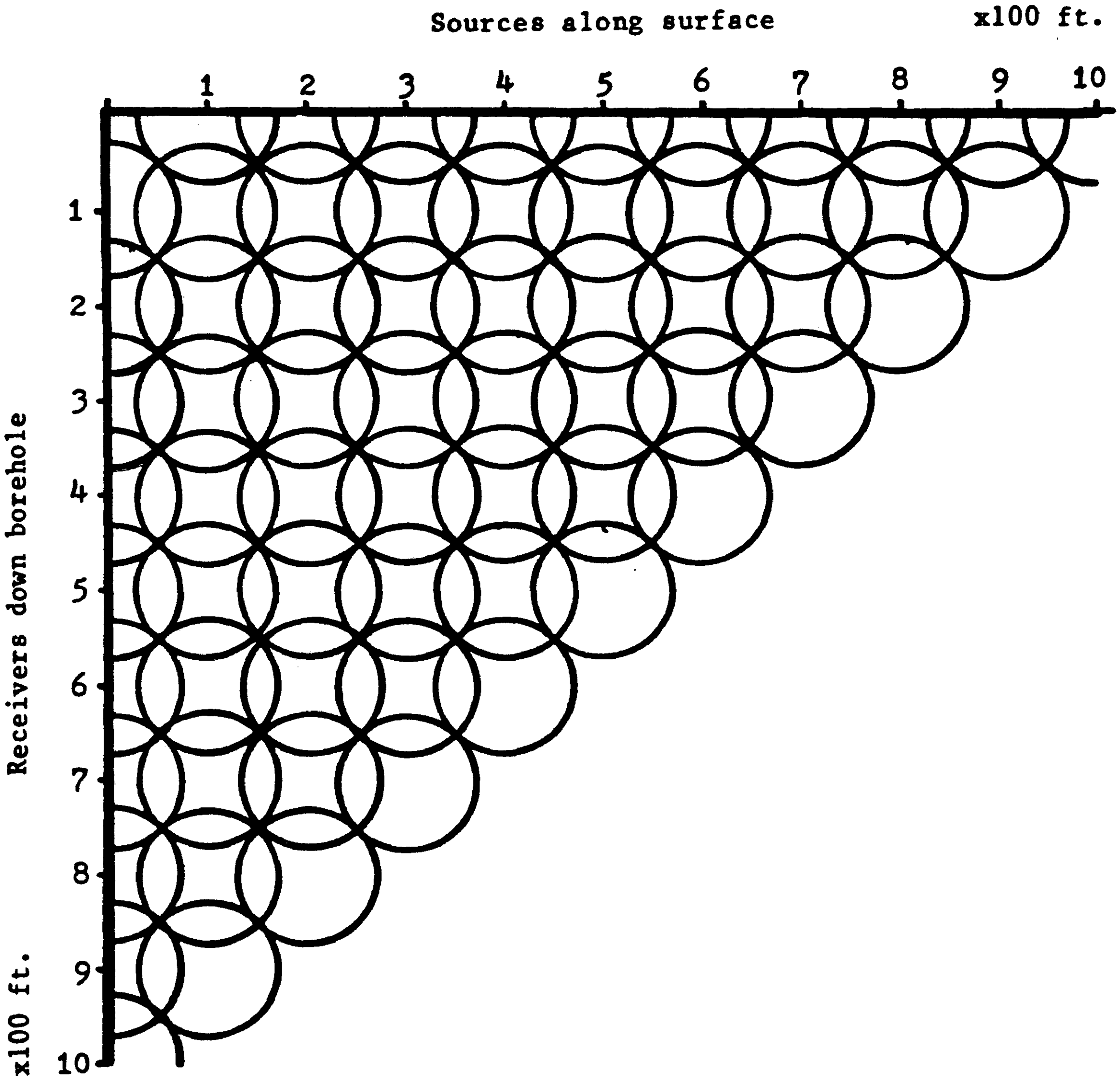


Figure 2. Modeling the medium using overlapping circles.

If the ray in question intersects the parameter circle, then equation (1) and equation (2) will have two points in common. A quadratic equation is formed by substituting (1) into (2). The quadratic is solved using the quadratic formula. Should both points prove to be imaginary, then the ray fails to intersect the circle.

The chord length l_{ij}^{mn} is calculated using the distance formula. These chord lengths are formed by all rays ij which intersect a given parameter circle around the point mn. The total length of ray segments d^{mn}, around the parameter mn, is the sum of all these chord lengths.

$$d^{mn} = \sum_{ij} l_{ij}^{mn}. \tag{3}$$

Every ray ij, which passes through a parameter circle mn, will have a residence time t_{ij}^{mn} within that circle as calculated from the observed travel time $\tilde{T}_{ij}$ and a weighting factor w_{ij}^{mn}.

$$t_{ij}^{mn} = \tilde{T}_{ij}\, w_{ij}^{mn}. \tag{4}$$

This weighting factor depends on the background velocity and can be viewed as the ratio of the expected time spent inside the circle to the total expected time of the ray. For a constant c_o velocity background:

$$w_{ij}^{mn} = \frac{\left[\frac{l_{ij}^{mn}}{c_o}\right]}{\left[\frac{L_{ij}}{c_o}\right]} = \frac{l_{ij}^{mn}}{L_{ij}}. \tag{5}$$

The total time spent about a parameter point mn is the sum of all the residence times. The average slowness at the point mn is given by:

$$s^{mn} = \frac{\sum_{ij} t_{ij}^{mn}}{\sum_{ij} l_{ij}^{mn}}. \tag{6}$$

2.2 Constant background - Iterative algorithm

An iterative technique is applied to improve upon this first guess model. First, a forward model is needed to compute an estimated travel time. Using straight raypaths, the ray is divided into equal segments Δl. The slowness along the entire segment length is assumed to equal the slowness at the midpoint of the segment, which is estimated by fitting a two-dimensional parabola to the surrounding four points of the slowness matrix. The estimated travel time T_{ij} is the sum of the slowness along the raypath.

$$T_{ij} = \sum_{(x_i,0)}^{(0,z_j)} s(x,z)\ \Delta l. \tag{7}$$

The total residual travel time E_{ij} along ray ij is given by:

$$E_{ij} = \tilde{T}_{ij} - T_{ij}. \tag{8}$$

This residual error time is equally distributed along the length of the ray. The slowness correction matrix is the ratio of the total error times about a parameter to the total length of the ray segments.

$$\tilde{s}^{mn} = \frac{\sum_{ij} e_{ij}^{mn}}{\sum_{ij} l_{ij}^{mn}}, \qquad \text{where} \quad e_{ij}^{mn} = E_{ij} \left[\frac{l_{ij}^{mn}}{L_{ij}}\right]. \tag{9}$$

This slowness correction matrix is added to the previous slowness matrix to obtain an updated model. Iteration is performed by computing a new correction matrix, based on the current slowness model, until the mean square of the residual times reaches a steady state.

$$\underset{\text{old}}{\sum_{ij} (E_{ij})^2} - \underset{\text{new}}{\sum_{ij} (E_{ij})^2} < \varepsilon. \tag{10}$$

2.3 Linear c(z) medium

For a linear c(z) medium, the raypaths are no longer straight, but instead are curved. The local radius of curvature is derived using Snell's Law.

$$\frac{\sin i}{c(z)} = p = \text{constant}. \qquad (11)$$

Taking the derivative of the ray parameter, and noting that the derivative of any constant is zero, we find that

$$dp = \frac{\partial p}{\partial i}\, di + \frac{\partial p}{\partial z}\, dz, \qquad (12)$$

or

$$0 = \frac{\cos i}{c(z)}\, di - \frac{\sin i}{c^2(z)} \frac{dc(z)}{dz}\, dz. \qquad (13)$$

From the geometry shown in Fig. 3:

$$dS = R\,di, \quad \text{and} \quad \cos i = \frac{dz}{dS}. \qquad (14)$$

By combining the two equations (14), we conclude that

$$di = \frac{dz}{R \cos i}. \qquad (15)$$

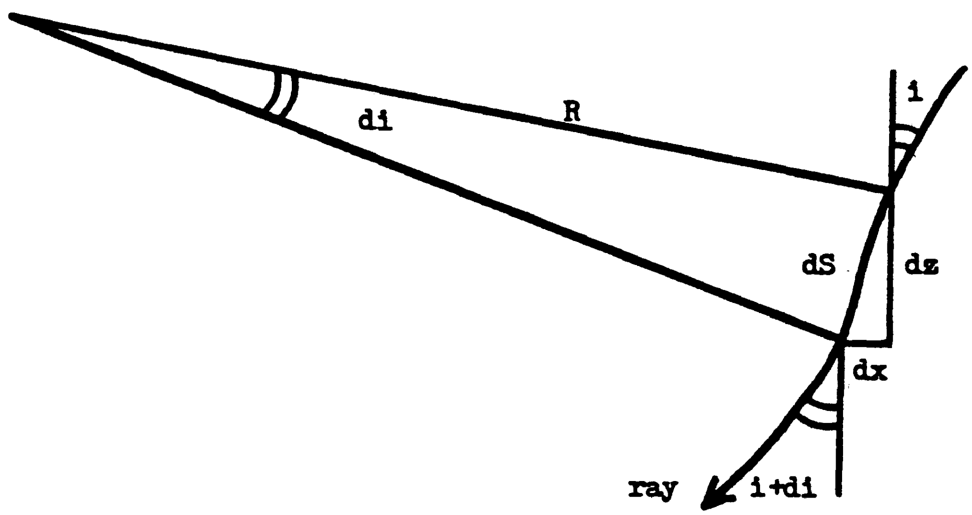


Figure 3. Derivation of the local radius of curvature.

Substituting (15) into (13):

$$0 = \frac{dz}{R\,c(z)} - \frac{\sin i}{c^2(z)} \frac{dc(z)}{dz}\, dz, \tag{16}$$

or

$$\frac{1}{R} = \frac{\sin i}{c(z)} \frac{dc(z)}{dz} = p\, \frac{dc(z)}{dz}. \tag{17}$$

So for a linear velocity $c(z) = c_o + m_o z$, the radius of curvature becomes a constant, which implies that the raypath is simply the arc along some circle; that is,

$$\frac{1}{R} = p\, m_o. \tag{18}$$

Using the Wiechert-Herglotz integral (Aki and Richards, 1980), the ray parameter can be determined (See appendix A). The result is

$$p = \frac{2\, m_o x_i}{\sqrt{[m_o^2 x_i^2 + c_o^2 + c^2(z_j)]^2 - 4\, c_o^2 c^2(z_j)}}. \tag{19}$$

From the ray parameter, the radius of curvature becomes known using equation (18). With two points on this circular path known, namely the shot and receiver locations, the ray follows a path on the circle:

$$(x-x_o)^2 + (z-z_o)^2 = R^2. \tag{20}$$

Using the geometry in Fig. 4, the center of the circle can be determined:

$$\left|x_o\right| = R \cos(i+\alpha); \tag{21}$$

$$\left|z_o\right| = R \sin i. \tag{22}$$

Both angles are calculated from the geometry. The results are

$$\sin\frac{\alpha}{2} = \frac{x_i{}^2 + z_j{}^2}{2R}; \tag{23}$$

$$\tan(i+\frac{\alpha}{2}) = \frac{x_i}{z_j}. \tag{24}$$

Thus for a linear c(z) medium, the rays are arcs of a circle. The overall length of the ray is:

$$L_{ij} = R\alpha. \tag{25}$$

Over the two-dimensional medium, a grid of overlapping circles is

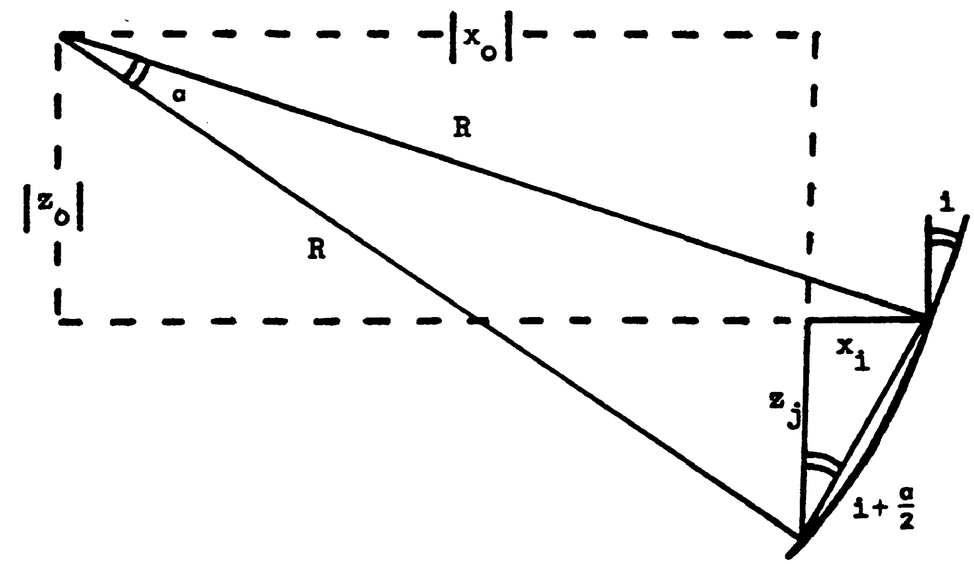


Figure 4. Origin determination.

superimposed. Any ray will intersect a circle provided that:

$$\left| d - R \right| < r, \qquad \text{where} \quad d^2 = (x_m - x_o)^2 + (z_n - z_o)^2. \tag{26}$$

The arc length l_{ij}^{mn} formed by the ray ij, within the circle mn, is computed using the law of cosines (Fig. 5).

$$\cos\frac{\beta}{2} = \frac{R^2 + d^2 - r^2}{2Rd}; \tag{27}$$

$$l_{ij}^{mn} = R\beta. \tag{28}$$

The total ray distance d^{mn}, about a given parameter, is the sum of all the arc lengths.

$$d^{mn} = \sum_{ij} l_{ij}^{mn}. \tag{29}$$

Every ray ij, which passes through a parameter circle mn, will have a residence time t_{ij}^{mn} within that circle as calculated from the observed travel time $\tilde{T}_{ij}$ and a weighting factor w_{ij}^{mn}.

$$t_{ij}^{mn} = \tilde{T}_{ij} \, w_{ij}^{mn}. \tag{30}$$

The weighting factor is the ratio of the expected time spent inside

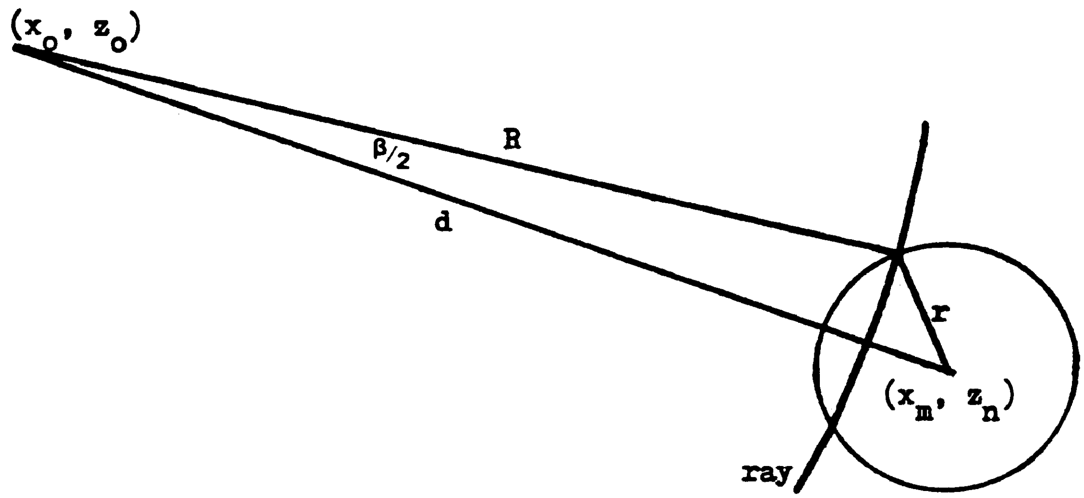


Figure 5. Arc length determination.

the parameter circle to the overall travel time for the entire ray.

$$w_{ij}^{mn} = \frac{\left[\frac{l_{ij}^{mn}}{c(z_n)}\right]}{T_{ij}}. \qquad (31)$$

Here $c(z_n)$ is the velocity at the origin of the parameter circle and T_{ij} is the theoretical travel time of the ray in the assumed linear background (see appendix B).

$$T_{ij} = \frac{1}{m_o} \ln\left[\frac{c(z_j)\,(1 + \sqrt{1 - p^2 c_o^2}\,)}{c_o\,(1 + \sqrt{1 - p^2 c^2(z_j)}\,)}\right]. \qquad (32)$$

The average slowness at the point mn is given by:

$$s^{mn} = \frac{\sum_{ij} t_{ij}^{mn}}{\sum_{ij} l_{ij}^{mn}}. \qquad (33)$$

2.4 Linear c(z) medium - Iterative algorithm

The slowness matrix can be improved by an iteration technique. The curved ray is divided into equal segments Δl. The slowness along the entire segment length is assumed to equal the slowness at the midpoint of the segment, which is estimated by fitting a two-dimensional paraboloid to the surrounding four points of the slowness matrix. An estimated travel time is computed by summing the slownesses at the mid-points of the segments.

$$T_{ij} = \sum_{(x_i,0)}^{(0,z_j)} s(x,z)\ \Delta l. \tag{34}$$

The total residual time E_{ij} along the ray ij is given by:

$$E_{ij} = \tilde{T}_{ij} - T_{ij}. \tag{35}$$

This residual error time is distributed equally along the raypath. The slowness correction matrix is the ratio of the sum of the local residual times to the sum of the arc lengths around each parameter:

$$\tilde{s}^{mn} = \frac{\sum_{ij} e_{ij}^{mn}}{\sum_{ij} l_{ij}^{mn}}, \quad \text{where} \quad e_{ij}^{mn} = E_{ij} \left[\frac{l_{ij}^{mn}}{L_{ij}}\right]. \tag{36}$$

This slowness correction matrix is added to the slowness matrix to obtain a revised slowness model. Iteration is performed, by computing a new correction matrix based on the updated slowness model, until the mean square of the residual times reaches a steady state:

$$\sum_{\substack{ij \\ \text{old}}} (E_{ij})^2 - \sum_{\substack{ij \\ \text{new}}} (E_{ij})^2 < \varepsilon. \tag{37}$$

3. TESTING THE ALGORITHMS

Both the constant and linear velocity background ART algorithms were tested using synthetic data sets. For a given slowness model, the travel times were generated using the forward modeling scheme in either the constant or linear velocity case. The slowness was integrated along either straight or circular paths as dictated by the general velocity trend of the input velocity model. Where the velocity increased linearly with x instead of z, the reciprocity principle of interchanging the shot and receiver locations was applied. Such data was calculated by interchanging the x and z variables and using the technique of section 2.4 to compute the synthetic data.

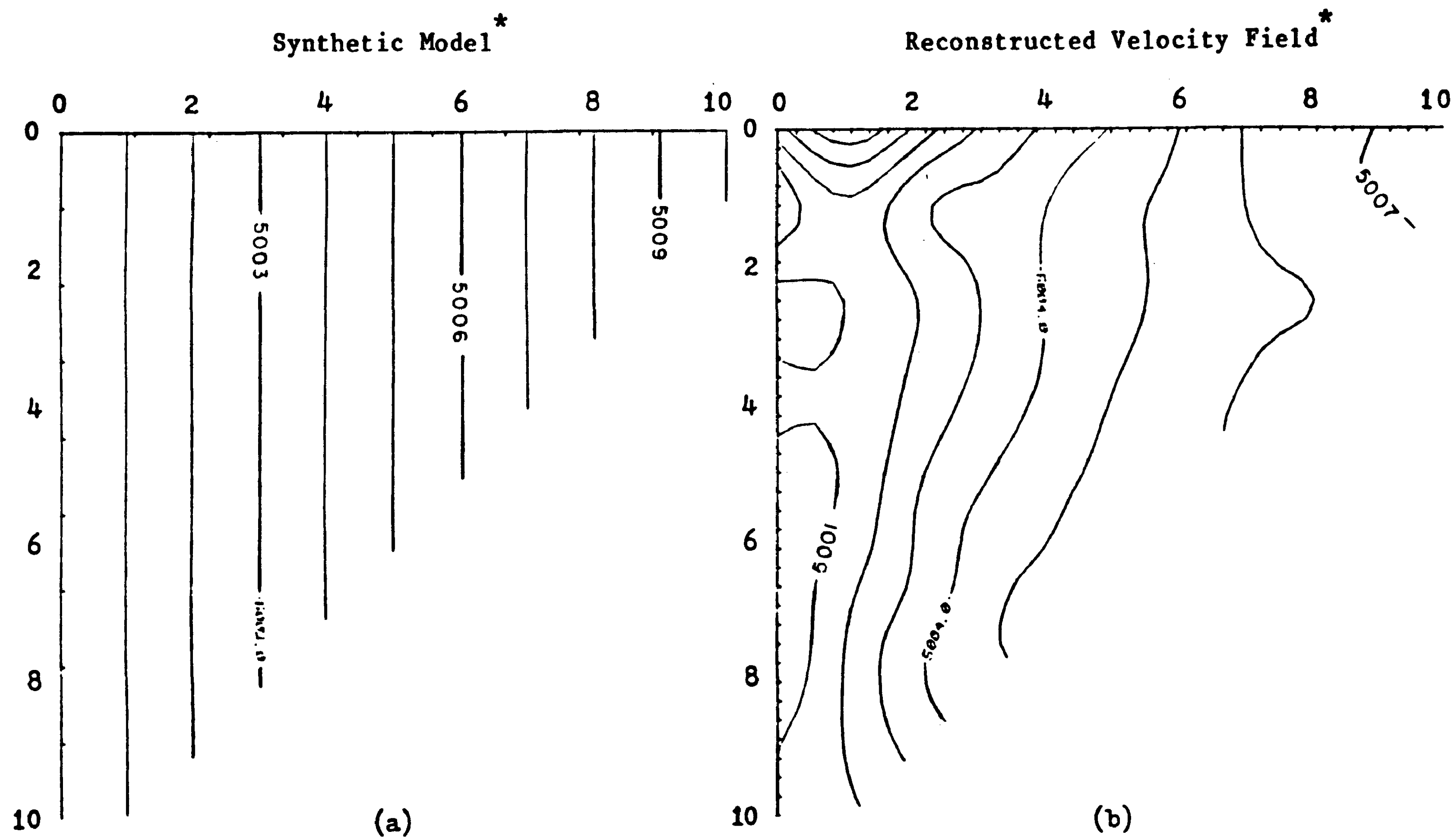
Clearly, the synthetic data does not correspond exactly to the perturbations which would be measured in the real case, but instead, the travel times represent a first order approximation for relatively small amplitude velocity anomalies. Therefore, my results test the accuracy of the reconstruction method and not the validity of linearizing the problem.

4. RESULTS

ART produces the best tomographical results when the velocity function is smoothly varying. Using synthetic data sets, accurately inverted velocity values occurred within the model, but the velocity values varied from the synthetic input near the edges of the model (Fig. 6, 7, 8, 9, 10, 11).

The greatest deviation occurred near the origin (Fig 6b, 7b, 8b, 11b, 11c). The first velocity down the well is erroneously high while the first velocity along the surface is erroneously low. The apparent cause of this phenomenon rests within the raypaths themselves and the way in which the ART algorithm handles these raypaths.

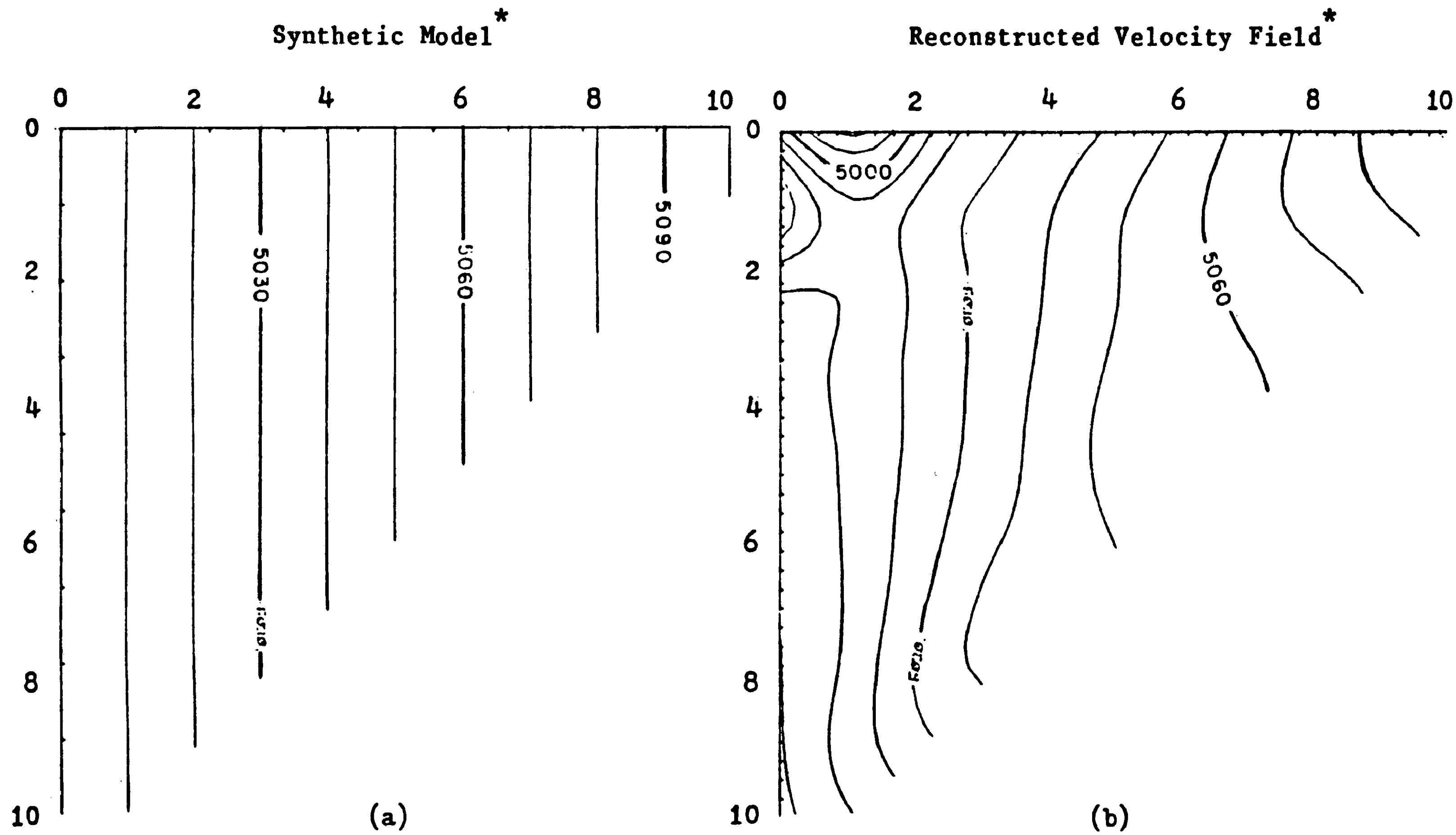
The first parameter down the well reflects the slownesses along the rays from all ten shot locations. However, the shortest raypath length most accurately reflects the true velocity at this point, while the longer rays reflect less of the velocity in question and more of the velocities further away. Because the ART algorithm equally weights all rays, regardless of their relative lengths, the slowness down the hole reflects the velocity change across the surface of the model. Appropriate constraints down the well will eliminate this problem, since the slowness in a borehole are usually known from the sonic log (Fig 11d).



*Travel times calculated using curved rays based on a background velocity - $v(x) = 5000 + .01(x_i)$.

*After 7 iterations using straight rays.

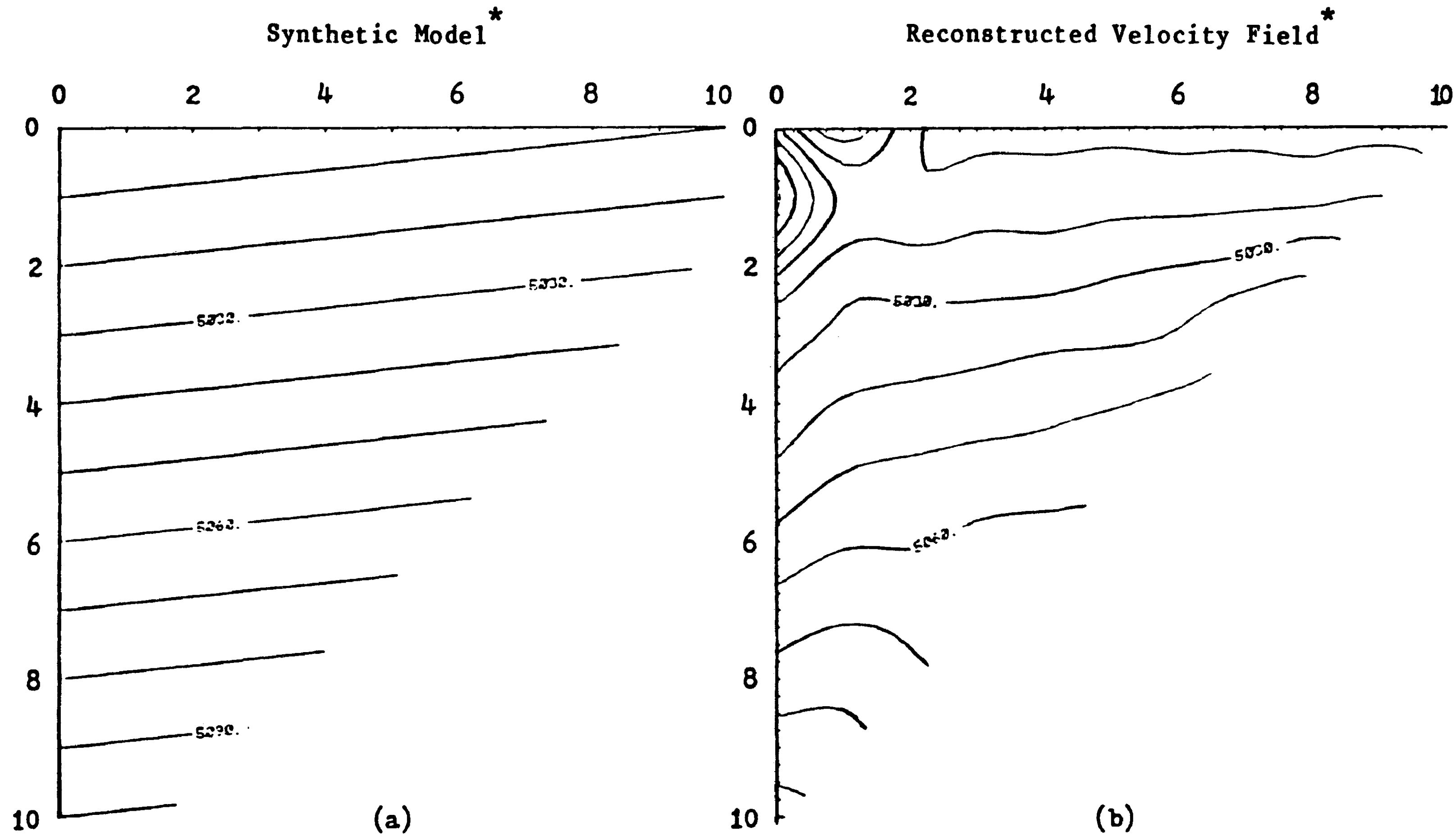
Figure 6. Velocity increasing with x.



*Travel times calculated using curved rays based on a background velocity - $v(x) = 5000 + .1(x_i)$.

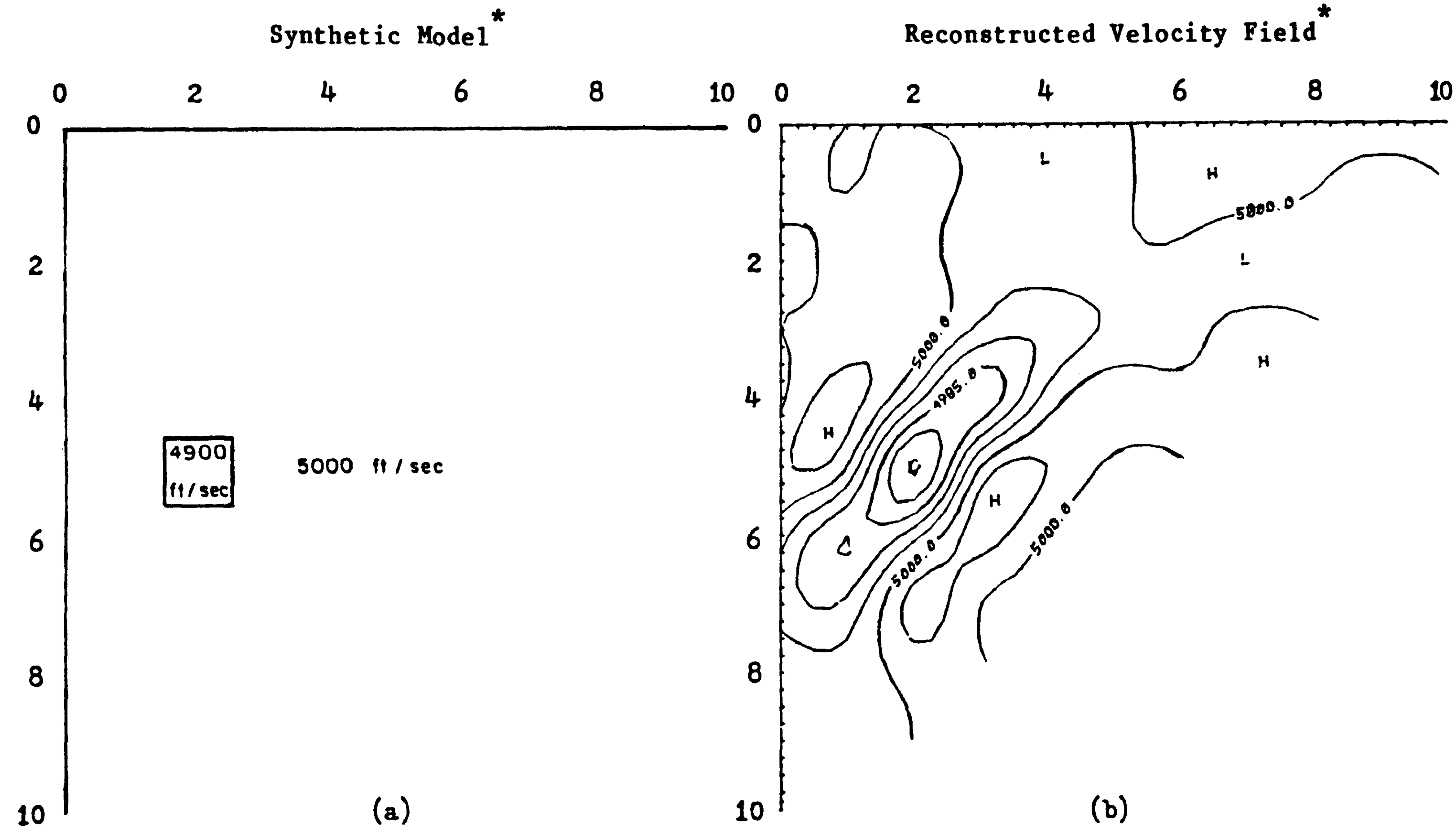
*After 9 iterations using straight rays.

Figure 7. Velocity increasing with x.



*Travel times calculated using straight rays. *After 11 iterations using straight rays.

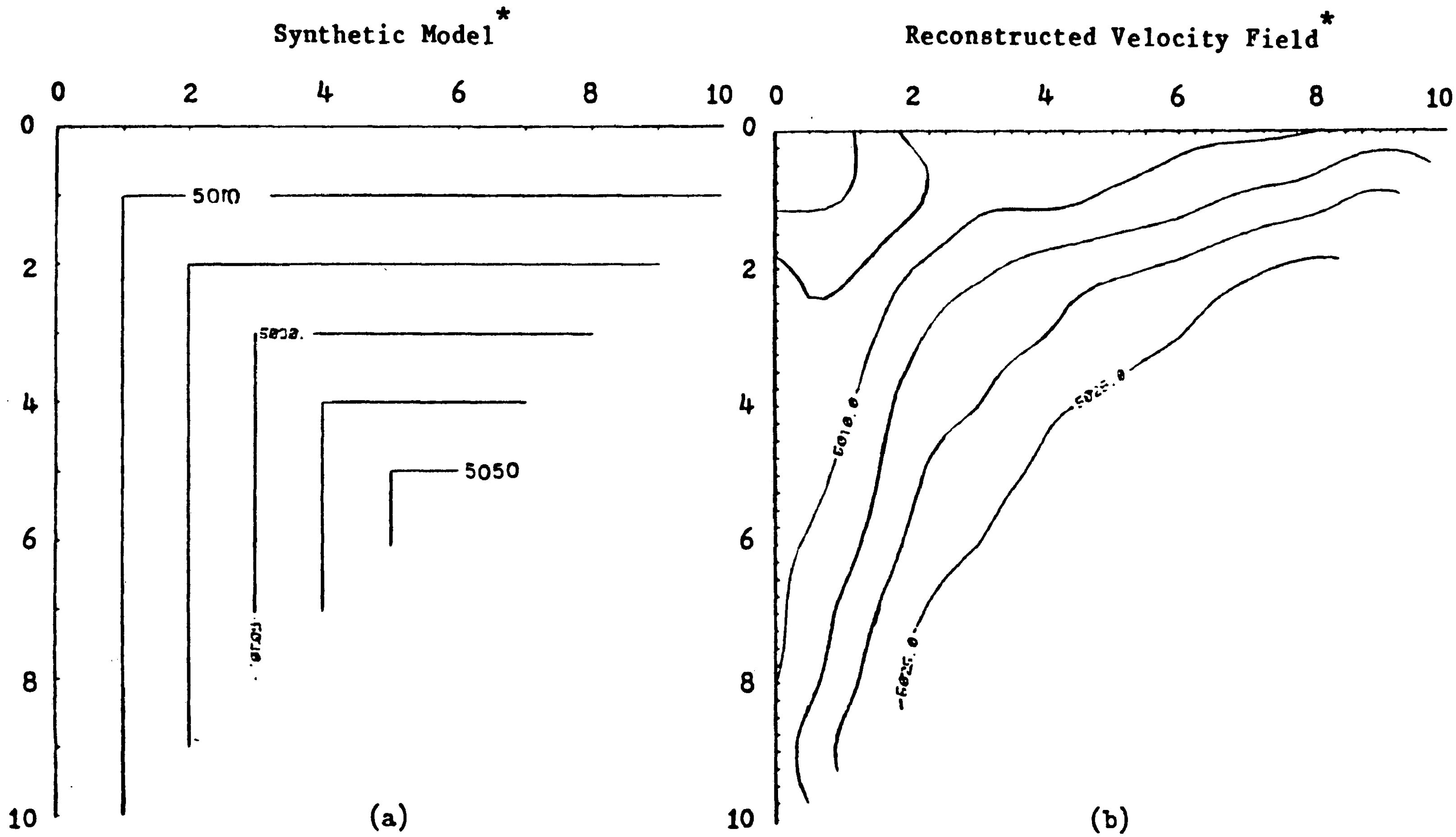
Figure 8. Dipping model.



*Travel times calculated using straight rays.

*After 12 iterations using straight rays.

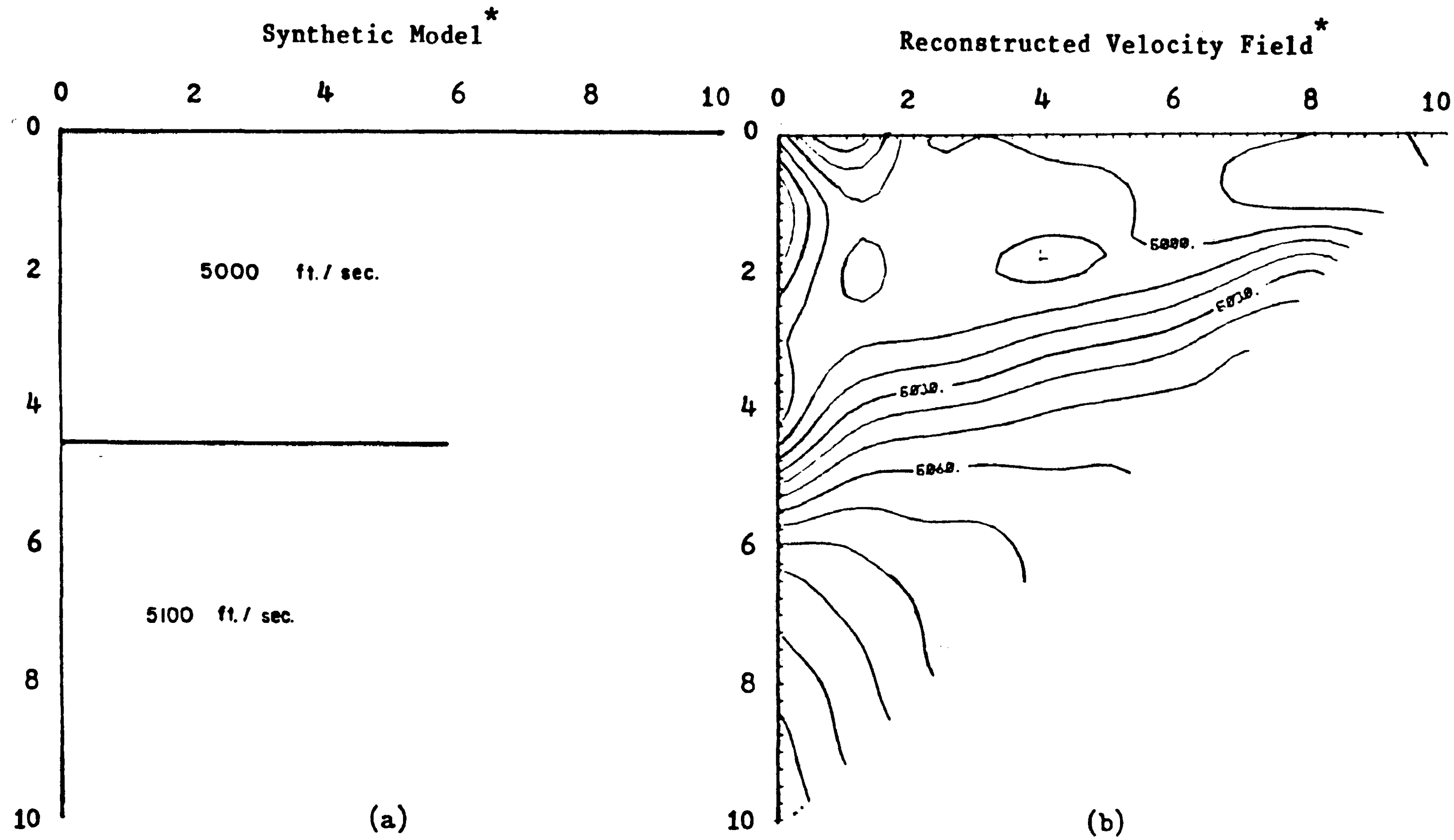
Figure 9. Model of a low velocity anomaly.



*Travel times calculated using straight rays.

**After 7 iterations using straight rays.

Figure 10. Velocity increasing with x and z.

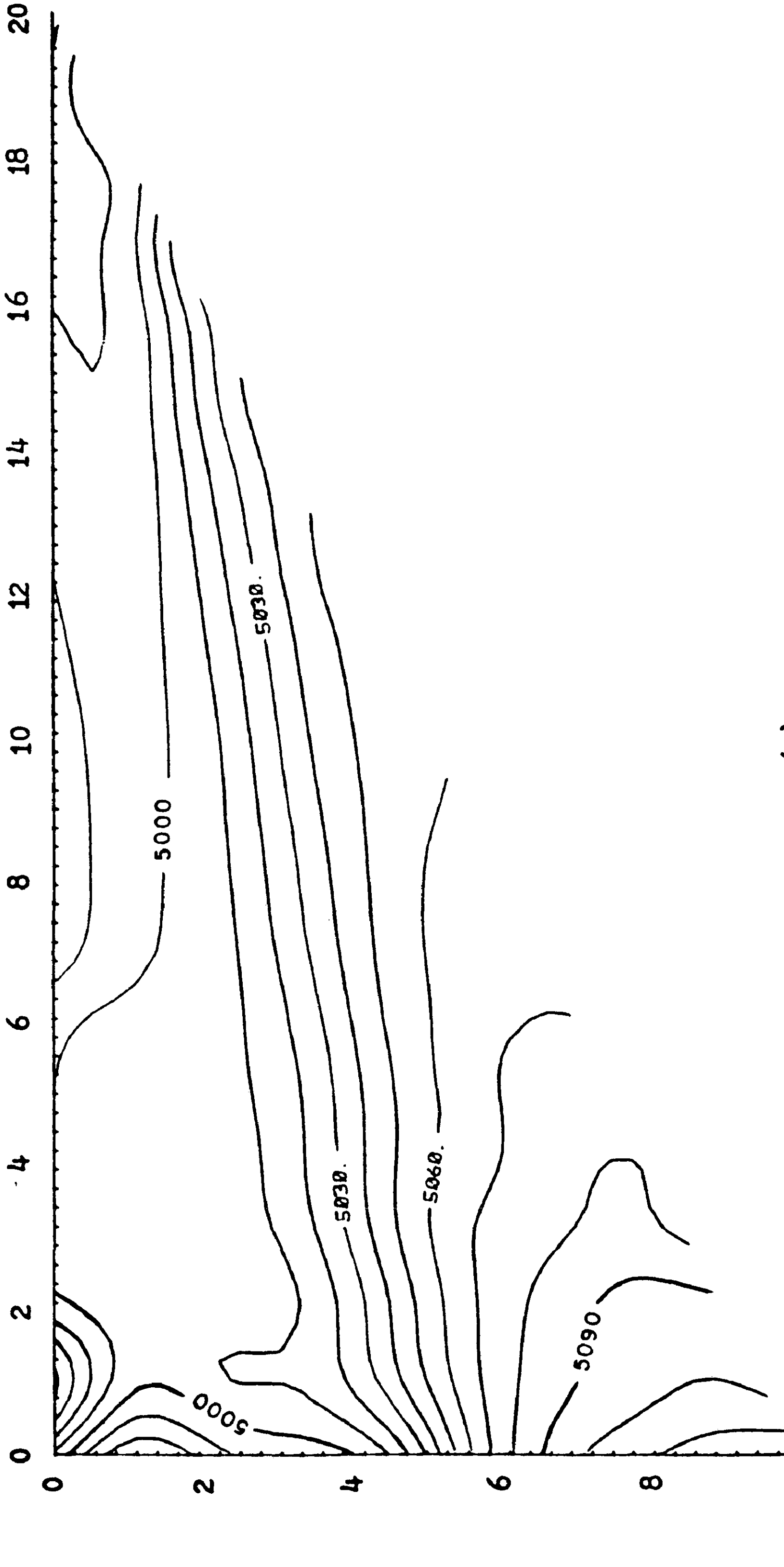


*Travel times calculated using straight rays.

*After 13 iterations using straight rays.

Figure 11. Two-layered model.

Reconstructed Velocity Field*

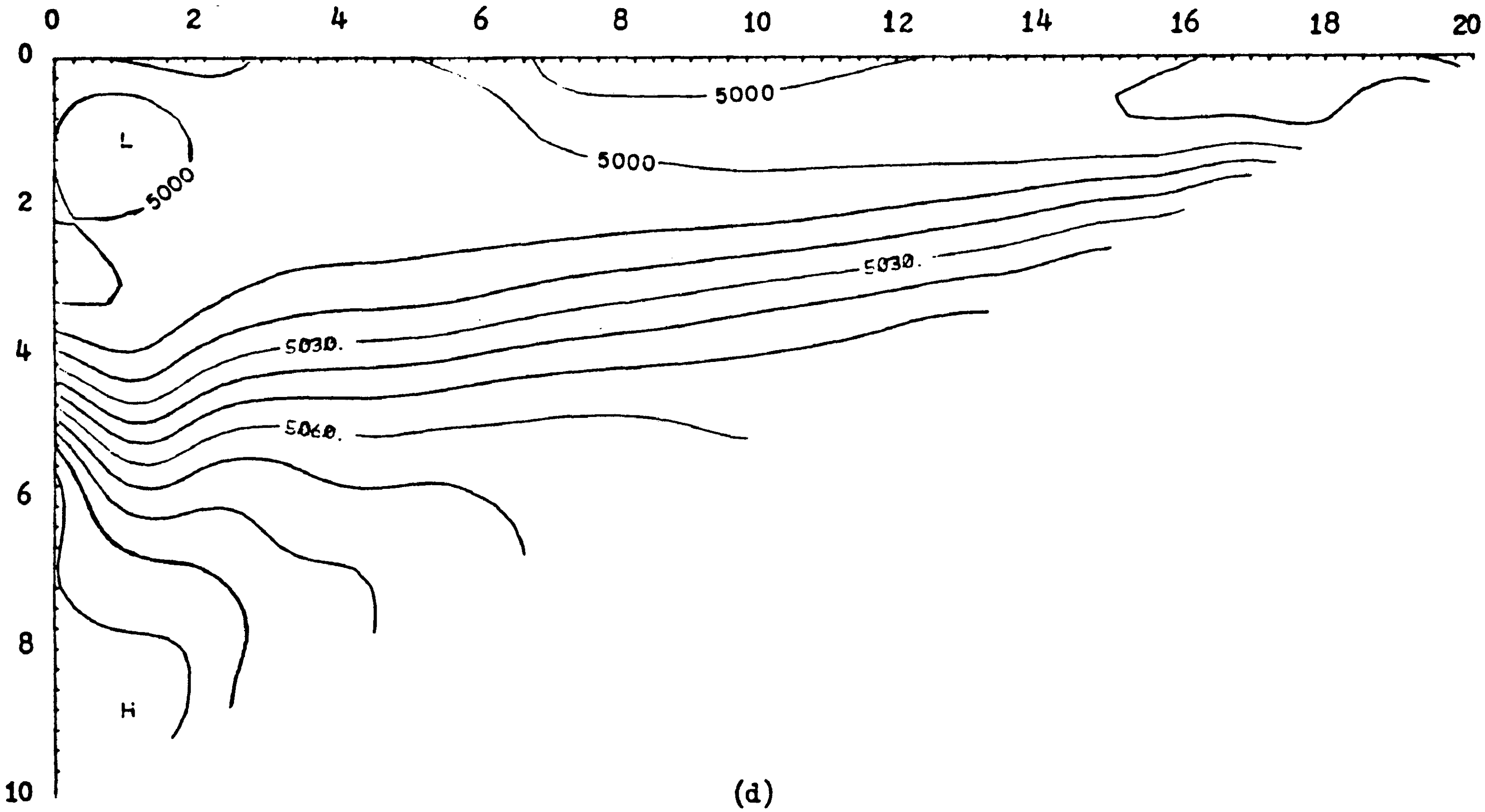


(c)

*After 10 iterations using straight rays.

Figure 11. Two-layered model.

Reconstructed Velocity Field*



(d)

*After 12 iterations using straight rays. Velocity parameters were constrained down the well (z-axis).

Figure 11. Two-layered model.

Along the edges, the velocity values fluctuated from the synthetic model. These deviations resulted from two factors. First, the erroneous origin velocities greatly influence the surface and borehole edges. Second, the diagonal edge parameters are poorly constrained.

Unconstrained parameters are defined by only one ray. All parameters within my model had at least three rays defining each parameter. However, along the diagonal edge, these three rays are nearly identical and as such, they are defined by nearly the same parameters. Therefore, these rays fail to be completely independent of each other. Thus, the parameters along the diagonal edge are poorly constrained. In unconstrained cases, ART tends to average the velocities equally along the raypath resulting in resolution problems.

By using a delta function as the input velocity structure (Fig 9), we can better illustrate this resolution problem. Resolution along the raypaths is poor, but between the rays, the resolution increases. This accounts for the skewness of the inverted delta function into an elliptical form elongated along the dominant raypath direction. Still, the location of the slowness delta point is accurately defined, though not its true velocity value.

This poor resolution along the raypaths contributes to uniqueness problems (Fig 10). In general, an infinite number of models can produce the same travel time data. ART converges to the model which is the closest to the assumed background velocity and not necessarily a representative of the true medium.

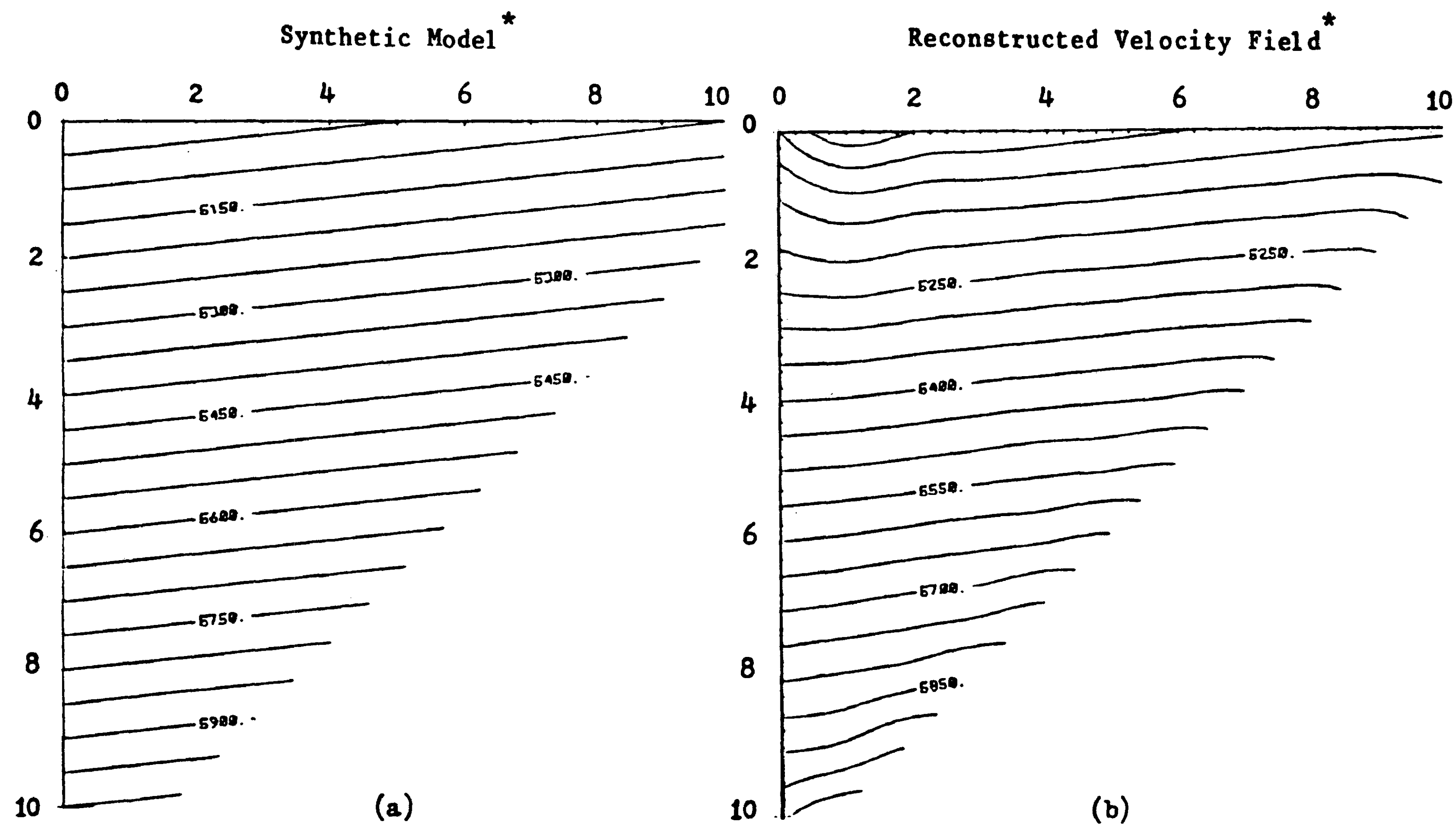
Uniqueness requires that all the slowness parameters are over constrained; that is more than one ray intersects each parameter circle. Furthermore, these rays must be independent. A close examination of my model reveals that at least three rays intersect each parameter; however, these rays travel approximately along the same direction and thus, these rays reflect many of the same slowness parameters. Clearly, these rays are not independent of each other.

The back-projected algorithm along curved rays was tested using synthetic data generated on linear trending, velocity models. In each case, the curved ART algorithm accurately reconstructed the synthetic models (Fig. 12, 13, 14).

5. PIERRE SHALE EXPERIMENT

The multiple offset VSP, which was taken by the CSM Exploration Research Laboratory of the Geophysics Department, provided a real data test of the algorithms. Field work was done, in November 1980, at the CSM test site in northeastern Colorado, about five miles south of Brush Colorado (Southwest 1/4, Section 28, Township 3N, Range 55W). Sixta (1982) describes the data aquisition and data processing of this VSP experiment.

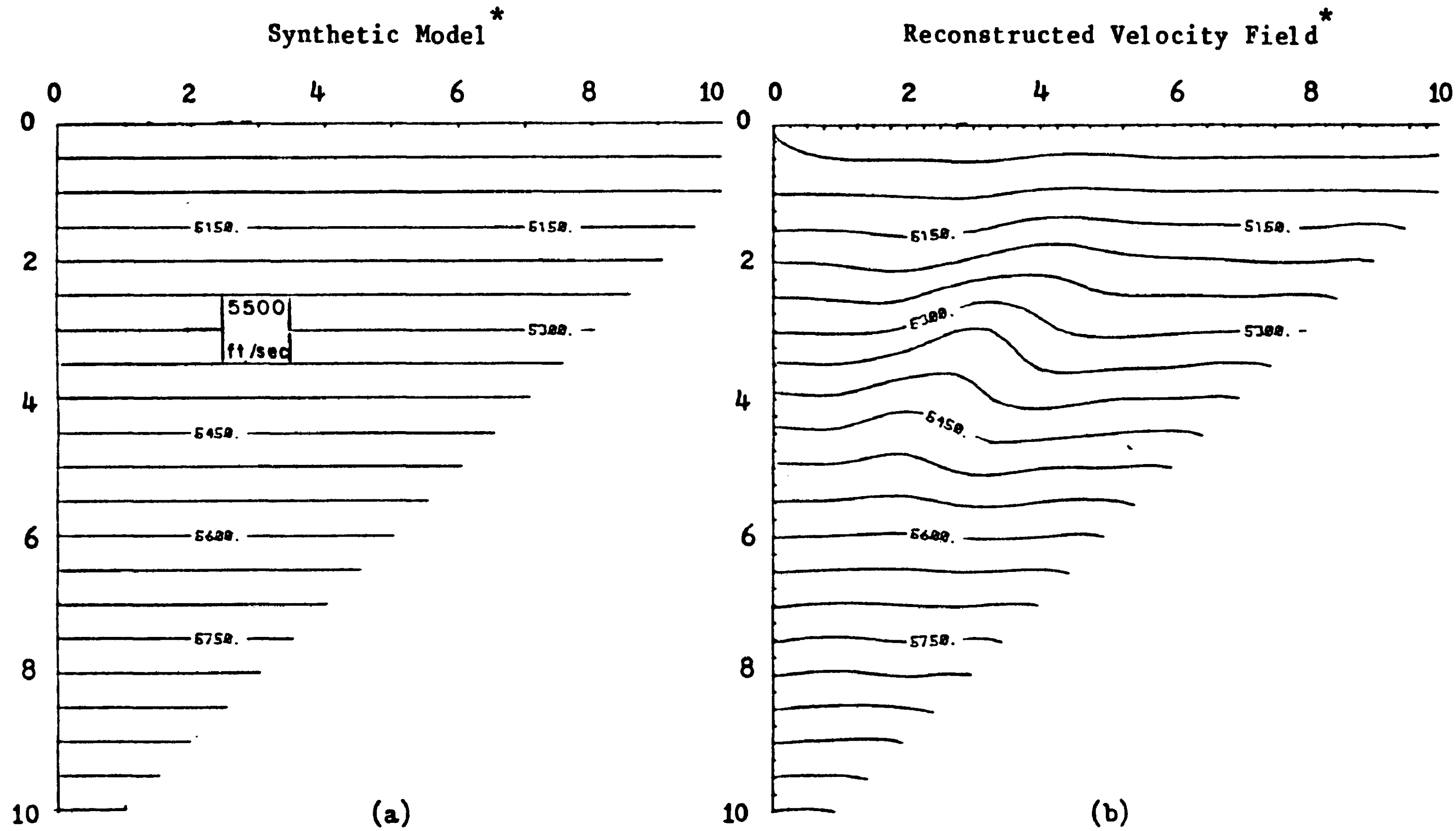
This site was chosen for its simple geologic setting. A 40 foot



*Travel times calculated using curved rays based on a background velocity - $v(z) = 5000 + z_j$.

*After 10 iterations using curved rays.

Figure 12. Dipping model.



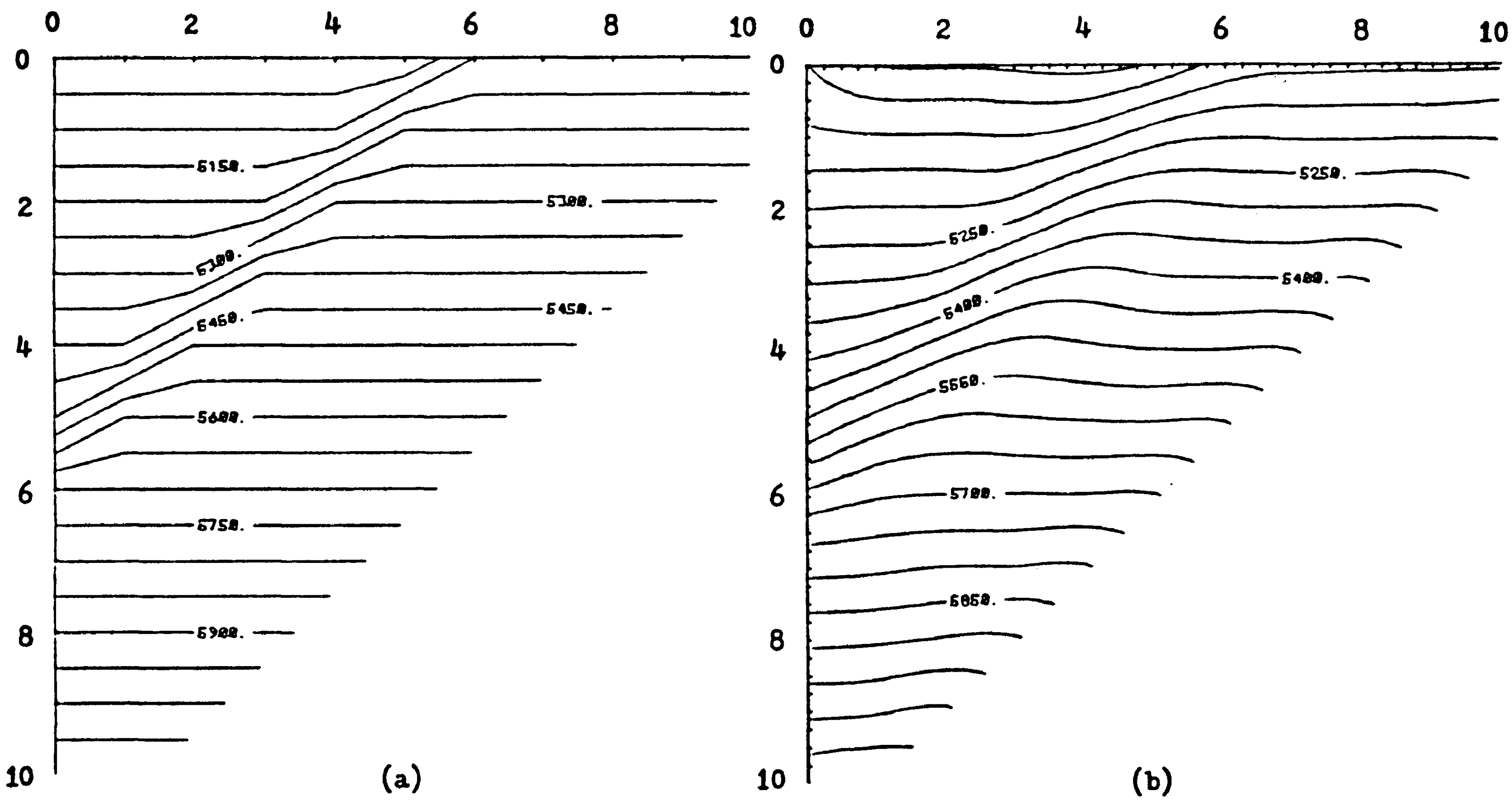
*Travel times calculated using curved rays based on a background velocity - $v(z) = 5000 + z_j$.

*After 10 iterations using curved rays.

Figure 13. High velocity anomaly within a linear trend.

Synthetic Model*

Reconstructed Velocity Field*



*Travel times calculated using curved rays based on a background velocity - $v(z) = 5000 + z_j$ ft./sec.

*After 10 iterations using curved rays.

Figure 14. Faulted model.

thick layer of eolian sand rests on the surface. Beneath this sand layer, a 40 foot thick layer of clay grades into the Pierre Shale below. These thicknesses vary laterally away from the borehole. Because the upper 1000 feet of the Pierre Shale is well noted for its homogeneous nature, refracted events should not appear on the records and therefore, the direct ray will be the first arrival.

The field layout is sketched in Fig. 15. Geophones were spaced down the cemented well every 100 feet within the interval between 400 and 1000 feet. Dynamite, placed in shot holes at a depth of 100 feet deep, were shot at 200 foot intervals across the surface to a distance of 1400 feet from the borehole. The travel times were picked at the first trough on the records (Fig. 16) and are tabulated in the table below.

Table 1: Travel times in msec.

	Horizontal offset in ft.						
Depth in ft.	200	400	600	800	1000	1200	1400
400	50.25	62.00	82.75	106.25	133.50	160.00	187.00
500	64.50	74.00	91.75	113.00	138.50	163.50	189.75
600	78.75	86.50	101.75	121.00	144.50	168.00	193.50
700	92.25	99.00	112.25	129.75	151.50	174.00	198.25
800	106.00	111.75	123.50	139.25	159.50	180.75	203.50
900	119.25	124.50	135.00	149.00	168.00	188.00	210.00
1000	133.50	138.00	147.00	160.00	177.75	196.50	217.00

Picking the arrival times at the first trough results in a lag time of about 3 msec. from the true onset of the waveform. This error in the travel time corresponds to a similar error in the velocity value. The

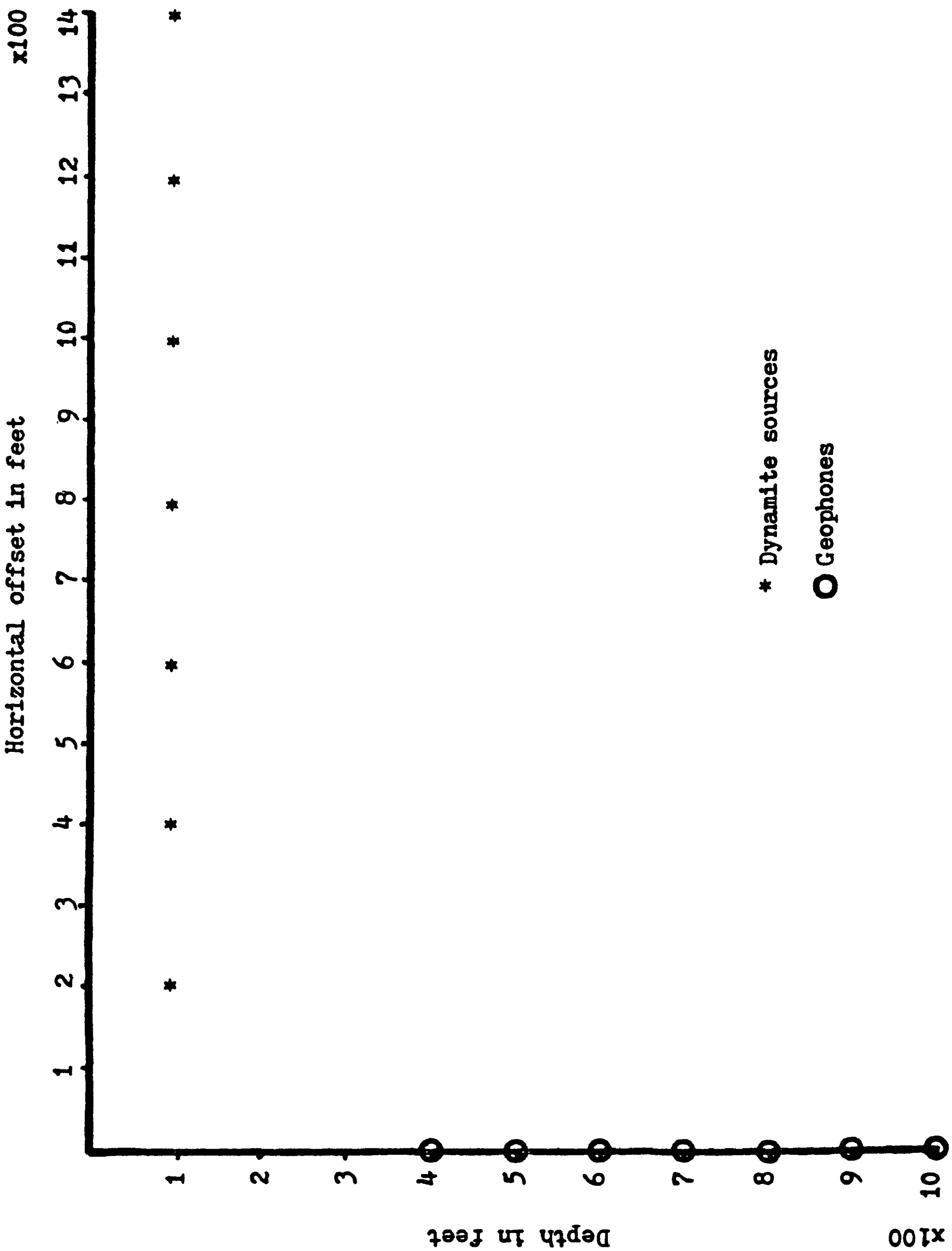


Figure 15. Field experiment layout.

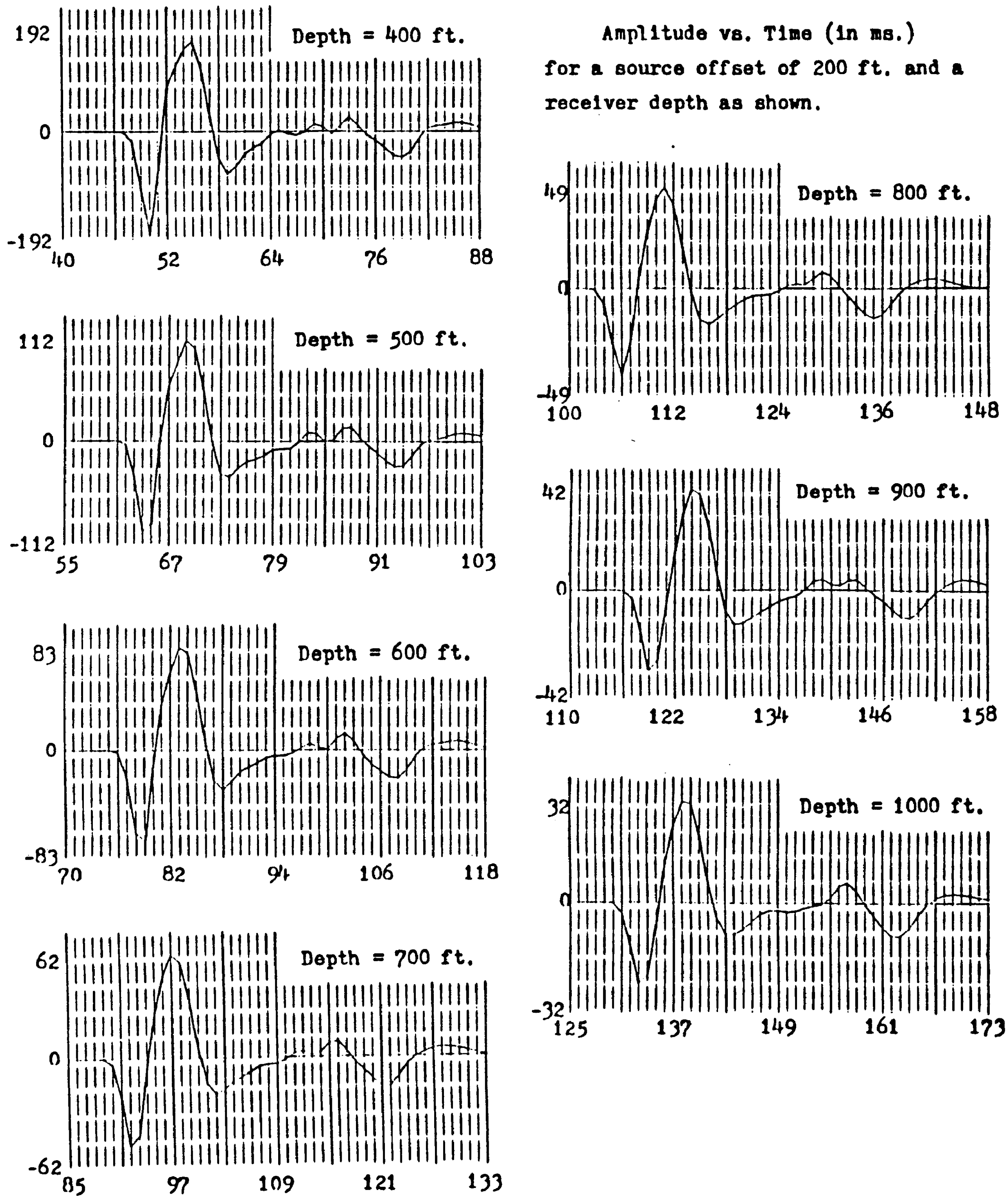


Figure 16. Typical waveforms.

travel time is the integral of the slowness along the raypath.

$$t = \int \frac{1}{c} dL, \qquad \text{or} \qquad t \approx \frac{L}{c}. \tag{38}$$

Differentiating (38), we obtain

$$dt \approx -L\frac{dc}{c^2}. \tag{39}$$

Dividing (38) into (39), yields the result

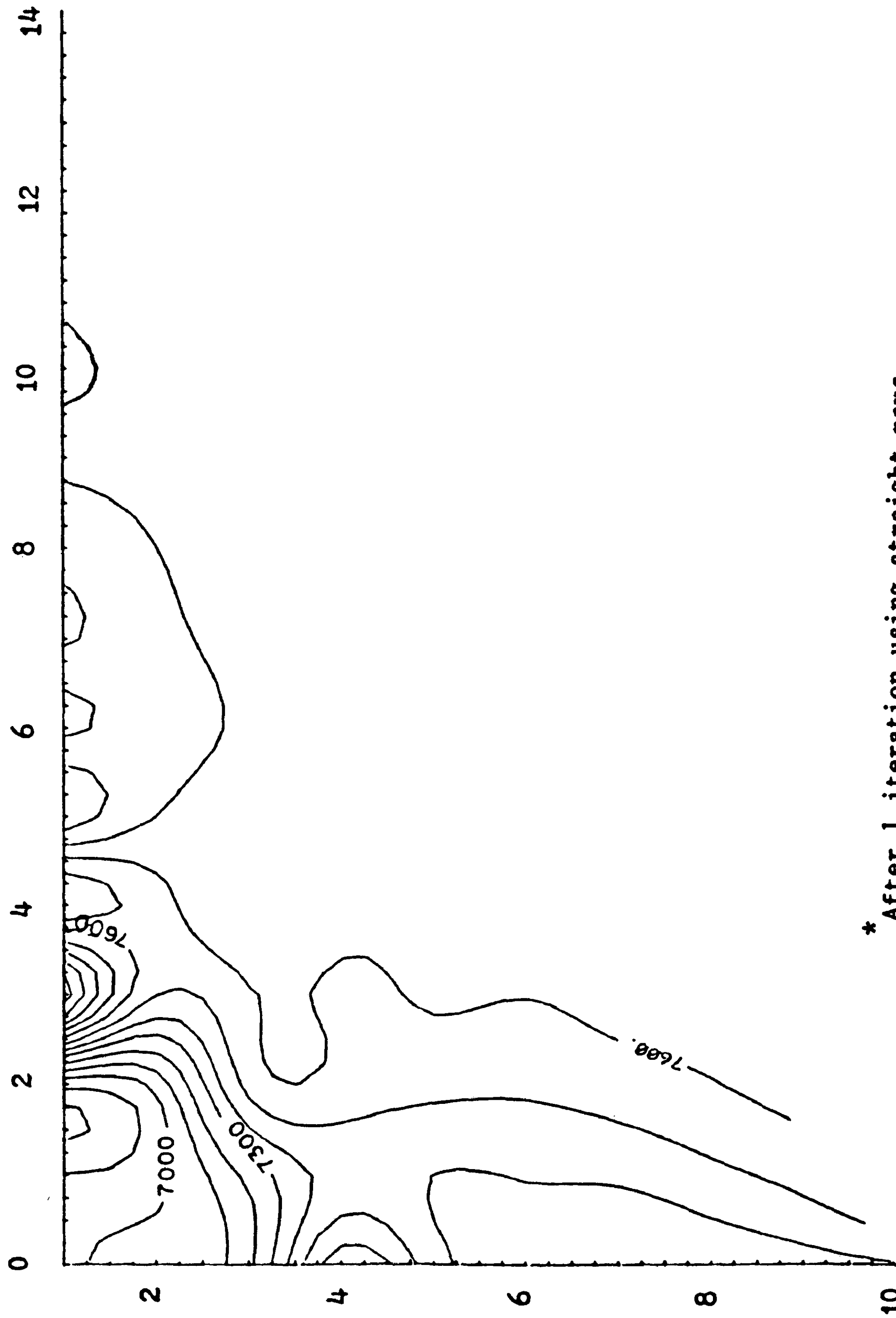
$$\frac{dt}{t} \approx -L\frac{dc}{c}. \tag{40}$$

By setting dt=3 msec. and noting that t ranges from 50.25 to 217.0 msec., the true velocity values range from 6.0 to 1.3% faster than those which are shown in the reconstructed fields.

The Pierre Shale is well noted for its homogeneous nature; however, our reconstructions fail to support this characteristic. Imaging was performed both without parameter constraints and with constraining the slowness parameters down the borehole to within five percent of the measured slowness values off the sonic log.

Both the constrained (Fig. 18a, 18b) and the unconstrained (Fig. 17a, 17b) reconstructions reveal a questionable high velocity cell near the surface. Although the sonic log for the well records a velocity of

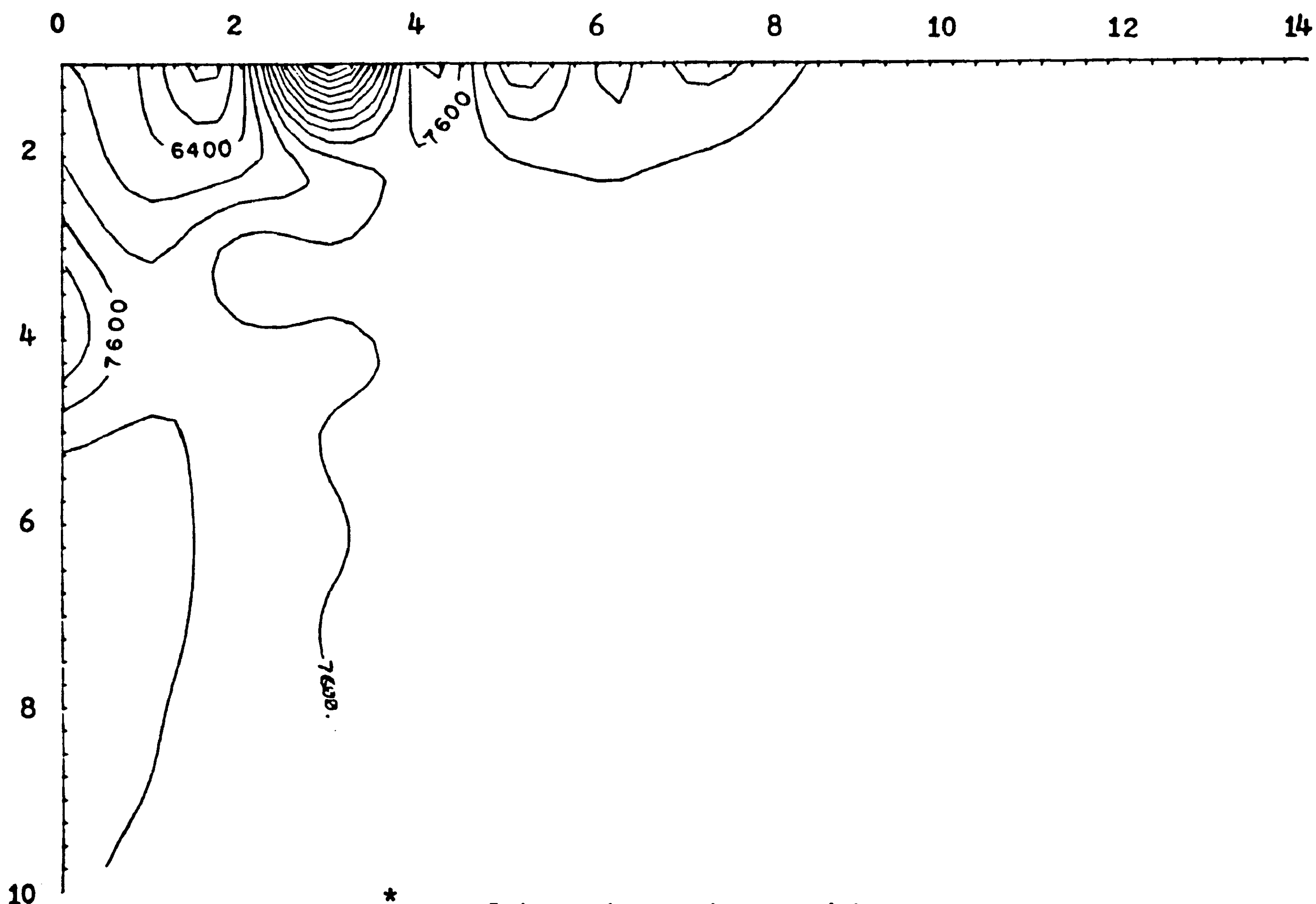
Reconstructed Velocity Field*



*After 1 iteration using straight rays.

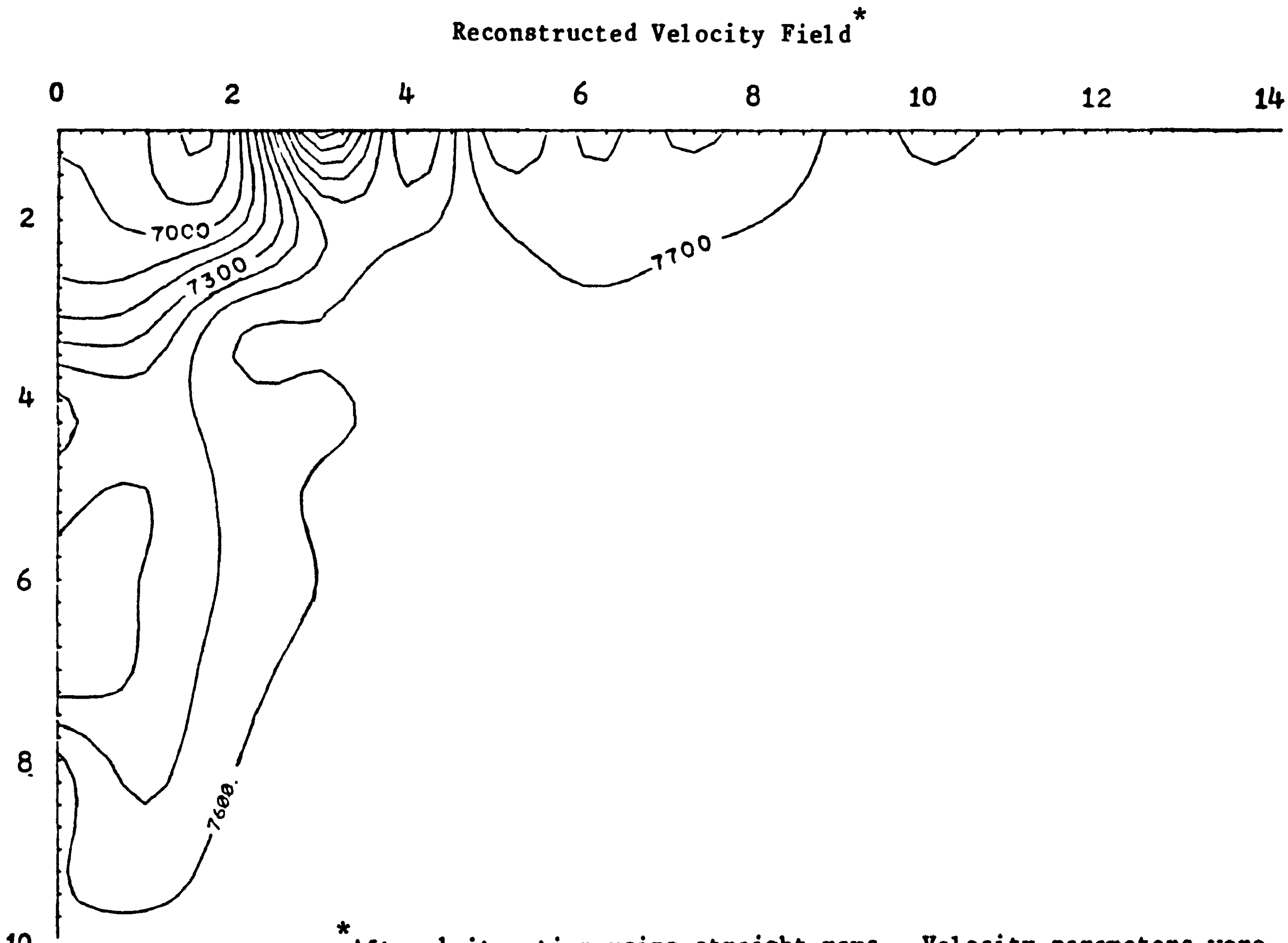
Figure 17a. Acoustic tomography from the Pierre Shale VSP data.

Reconstructed Velocity Field*



*After 5 iterations using straight rays.

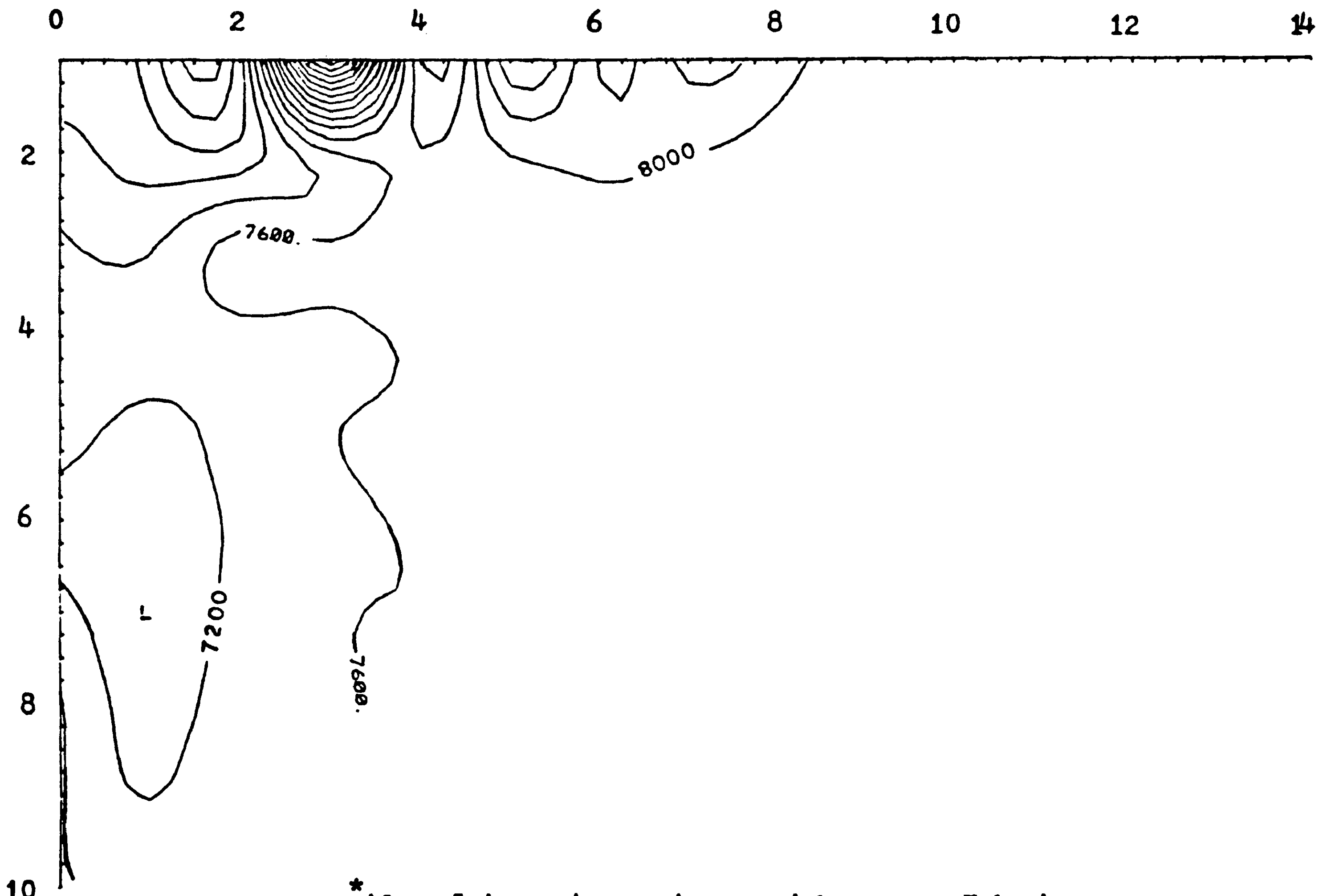
Figure 17b. Acoustic Tomography from the Pierre Shale VSP data.



*After 1 iteration using straight rays. Velocity parameters were constrained down the well (z-axis).

Figure 18a. Acoustic tomography from the Pierre Shale VSP data.

Reconstructed Velocity Field*



*After 5 iterations using straight rays. Velocity parameters were constrained down the well (z-axis).

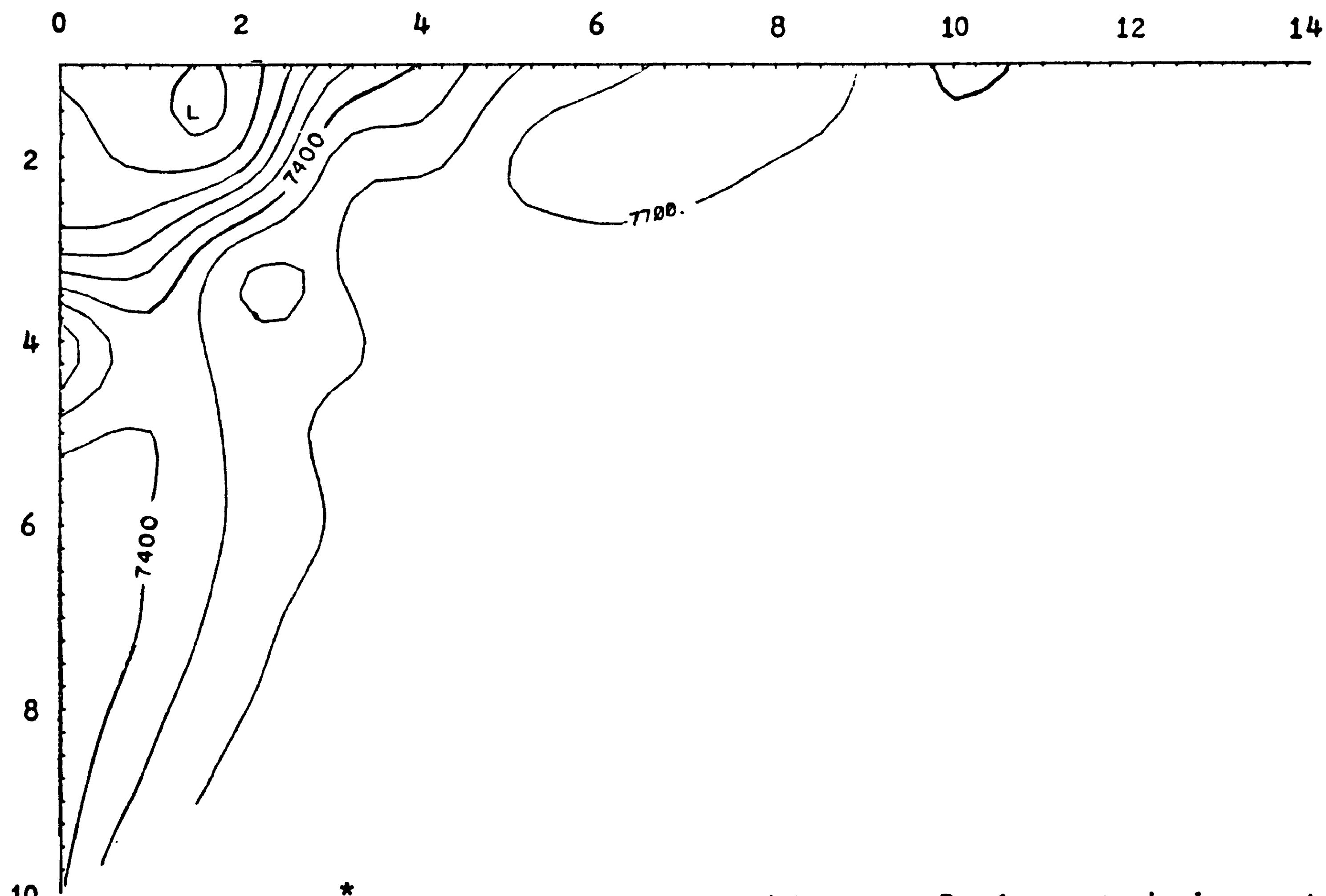
Figure 18b. Acoustic tomography from the Pierre Shale VSP data.

6250 ft/sec at a depth of 100 feet, the reconstructed high, only 600 feet away, reaches 11,900 ft/sec for the unconstrained case and 12,000 ft/sec for the constrained case. However, these velocity highs consisted of a single, poorly-constrained parameter along the surface, which clearly introduces uncertainty into this velocity value.

By eliminating any poorly constrained parameters along the surface, the imaging process is improved (Fig. 19, 20). These errant velocity values were ignored only when the reconstructed velocity field was contoured, and thus, they were used throughout the reconstruction algorithm. A similar step was applied to the reconstructions using circular rays based on the linear velocity trend exhibited by the sonic log (Fig. 21, 22). Each reconstruction shows a high velocity ridge of 8,000 ft/sec. hovering between a horizontal offset of 200 to 900 feet at a depth around 200 to 300 feet. Additionally, a low velocity cell exists at 100 foot offset and a depth of 500 to 800 feet. This low cell fluctuates between velocities of 7,000 to 7,200 ft/sec. depending upon the reconstruction conditions.

To test the validity of these reconstructions, the observed travel times were compared to theoretical data from a linearly increasing medium. This synthetic model, based on sonic log information, initially started at a velocity of 7,000 ft/sec. and increased to 8,000 ft/sec. at the total depth of the well. For comparision, both the synthetic and the Pierre Shale VSP travel times were plotted for each geophone posi-

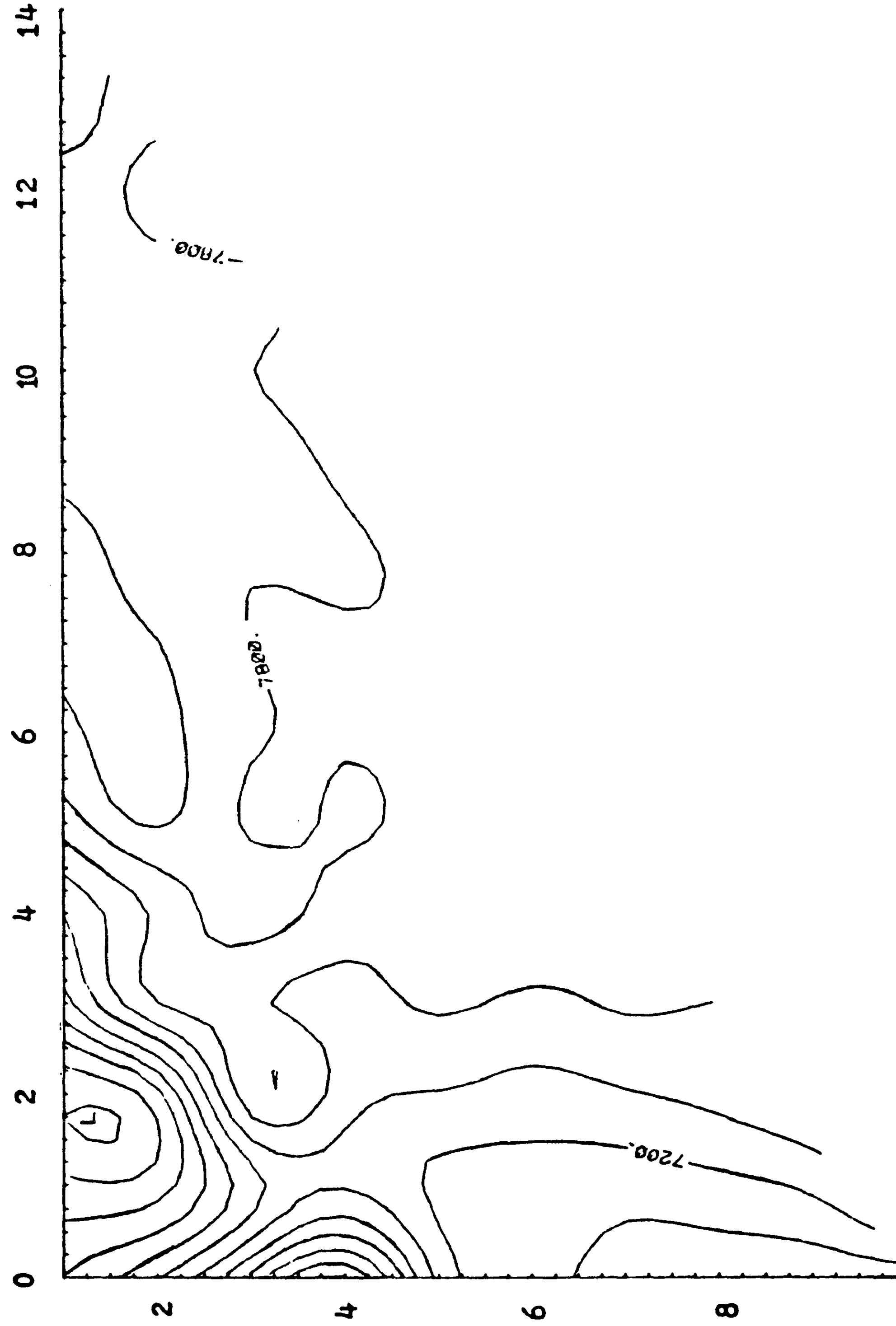
Reconstructed Velocity Field*



*After 1 iteration using straight rays. Poorly-constrained parameters along the surface were eliminated from the image.

Figure 19a. Acoustic tomography from the Pierre Shale VSP data.

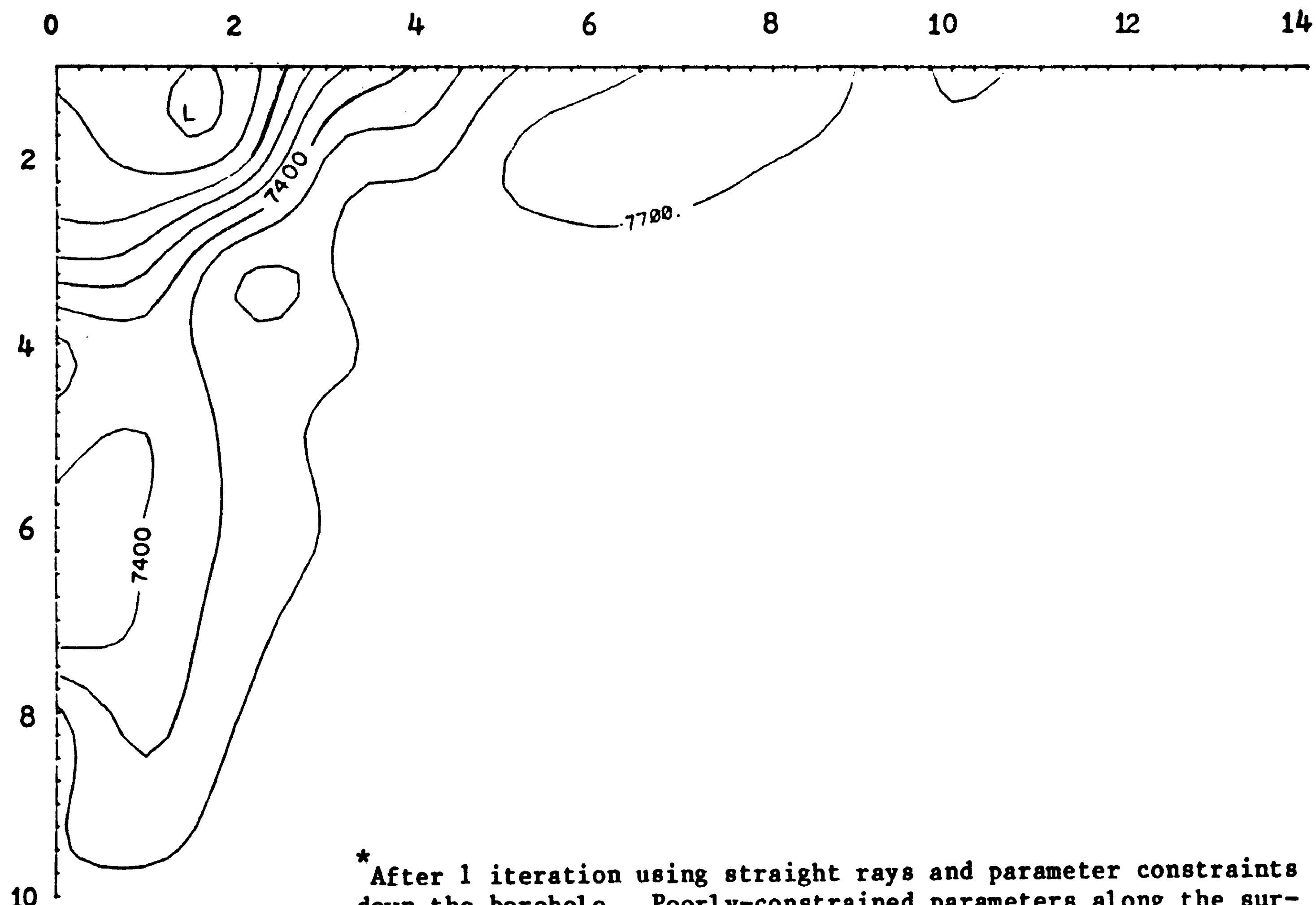
Reconstructed Velocity Field*



*After 5 iterations using straight rays. Poorly-constrained parameters along the surface were eliminated from the image.

Figure 19b. Acoustic tomography from the Pierre Shale VSP data.

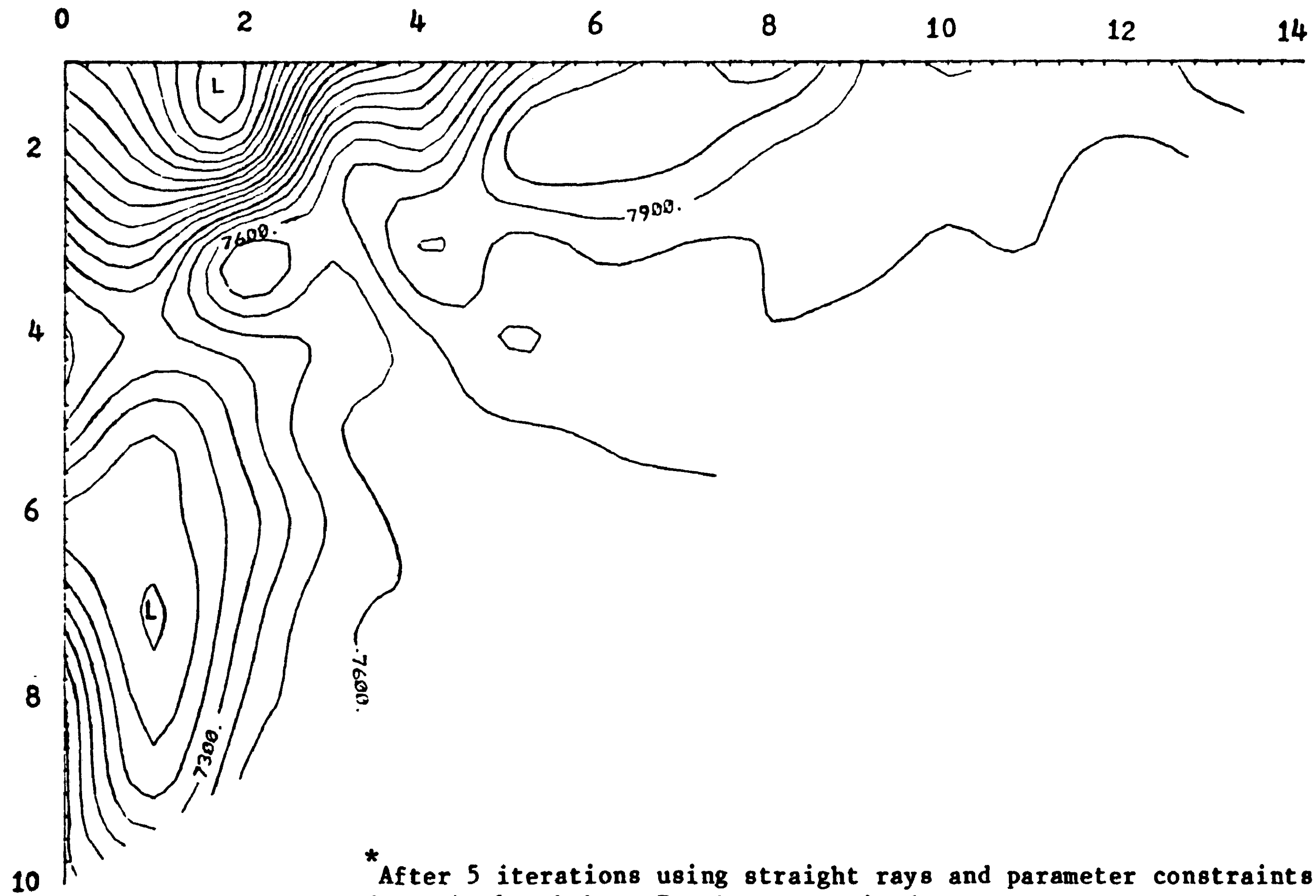
Reconstructed Velocity Field*



*After 1 iteration using straight rays and parameter constraints down the borehole. Poorly-constrained parameters along the surface were eliminated from the image.

Figure 20a. Acoustic tomography from the Pierre Shale VSP data.

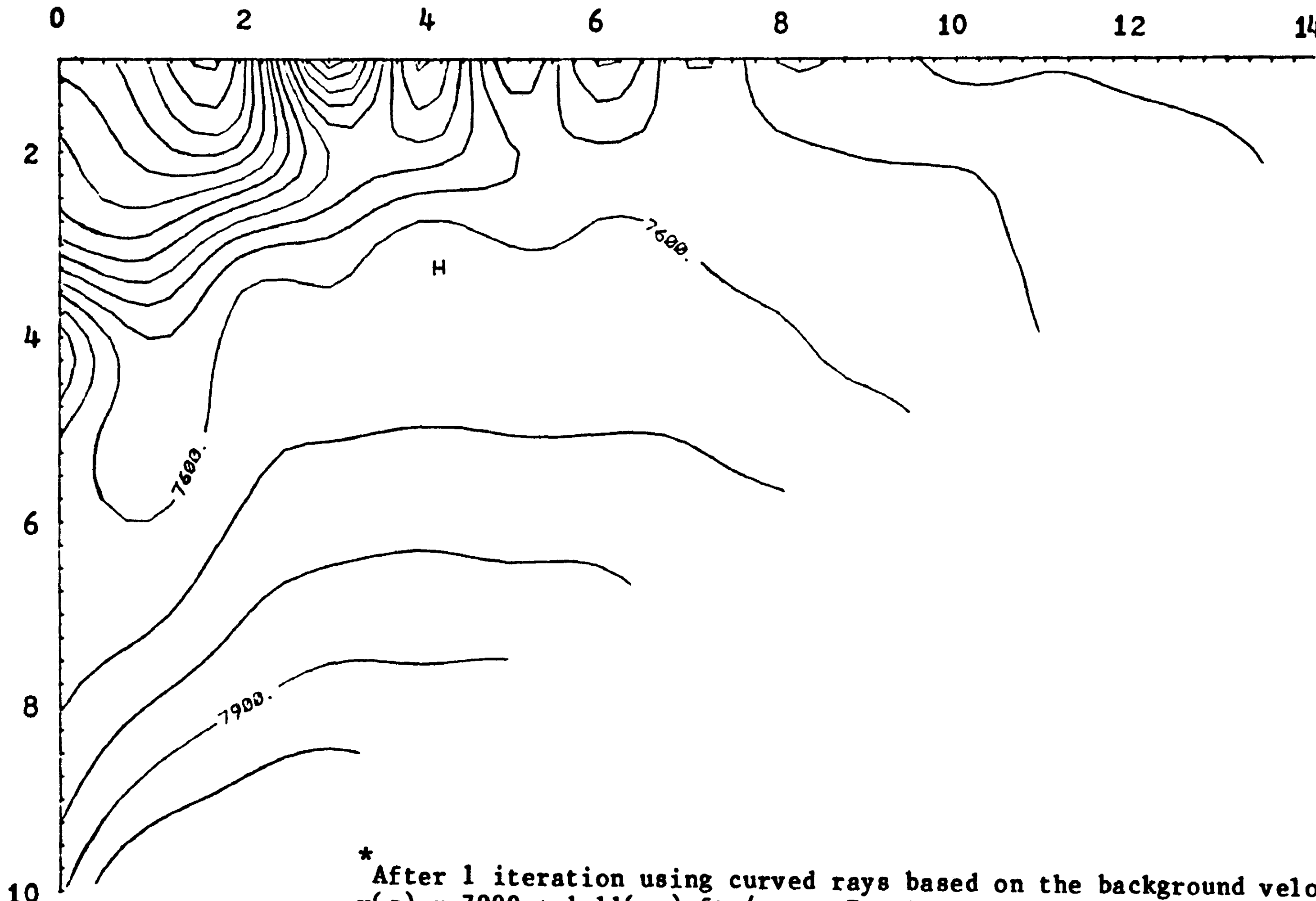
Reconstructed Velocity Field*



*After 5 iterations using straight rays and parameter constraints down the borehole. Poorly-constrained parameters along the surface face were eliminated from the image.

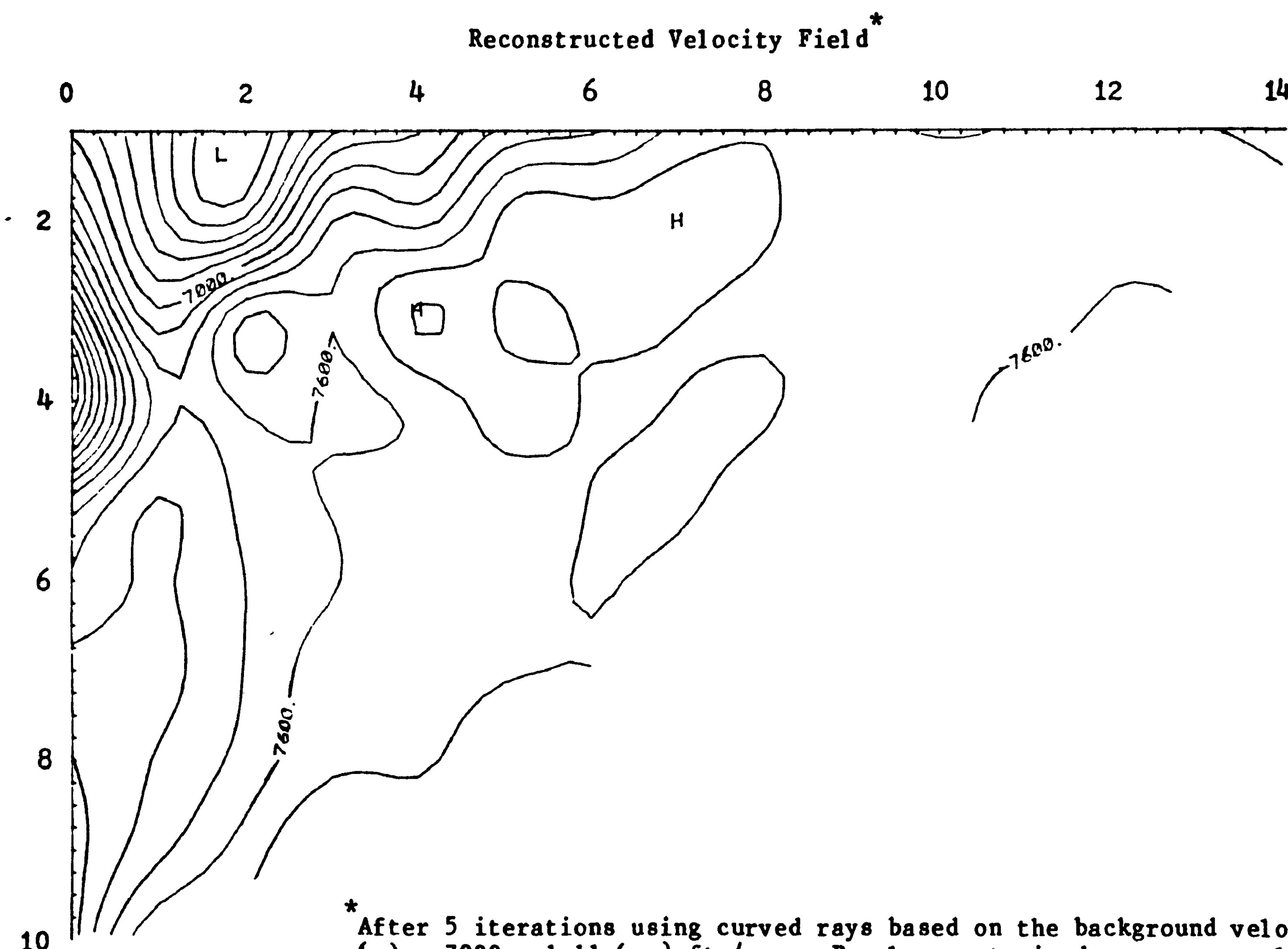
Figure 20b. Acoustic tomography from the Pierre Shale VSP data.

Reconstructed Velocity Field*



*After 1 iteration using curved rays based on the background velocity - $v(z) = 7000 + 1.11(z_1)$ ft./sec. Poorly-constrained parameters along the surface were eliminated from the image.

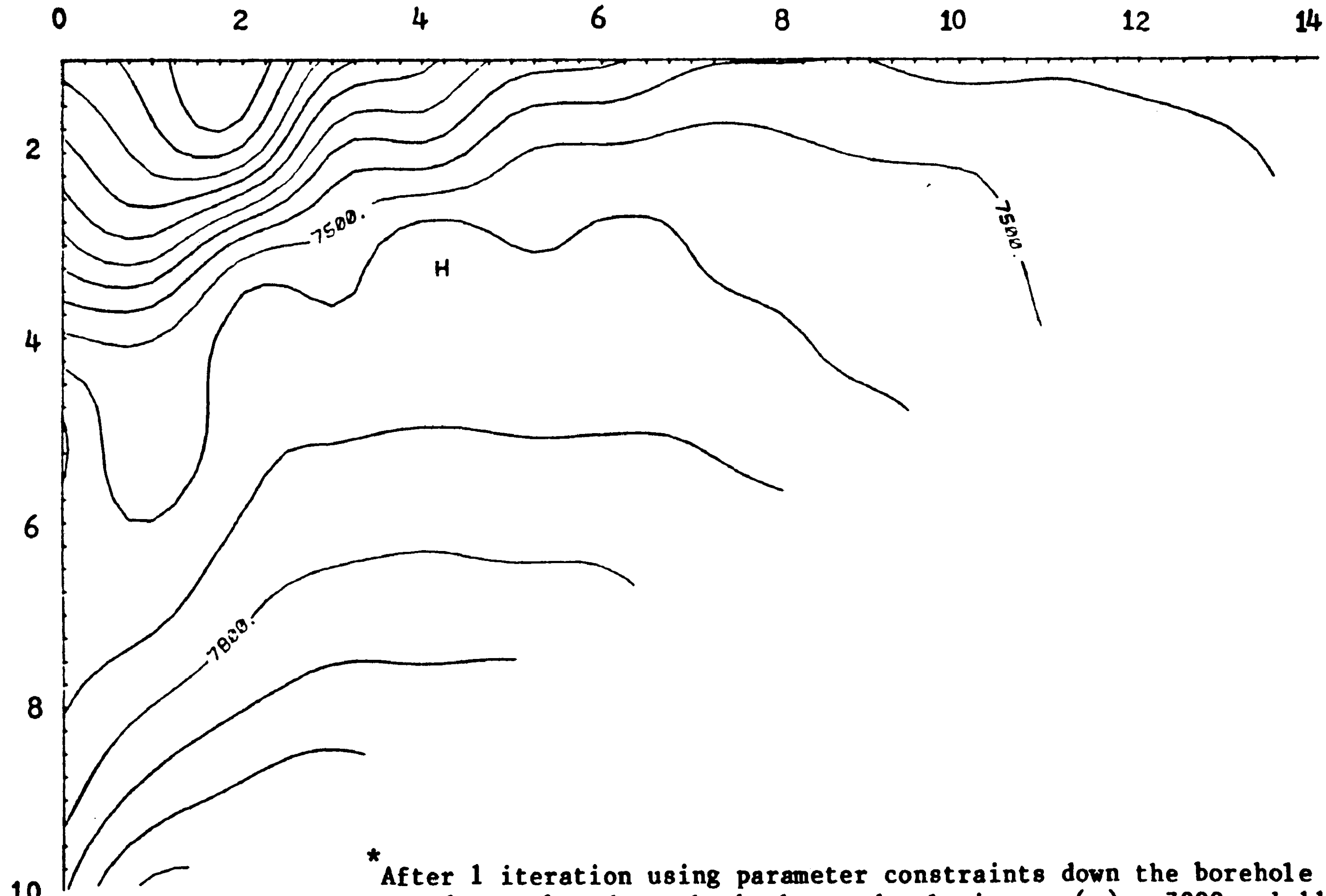
Figure 21a. Acoustic tomography from the Pierre Shale VSP data.



*After 5 iterations using curved rays based on the background velocity - $v(z) = 7000 + 1.11\ (z_i)$ ft./sec. Poorly-constrained parameters along the surface were eliminated from the image.

Figure 21b. Acoustic tomography from the Pierre Shale VSP data.

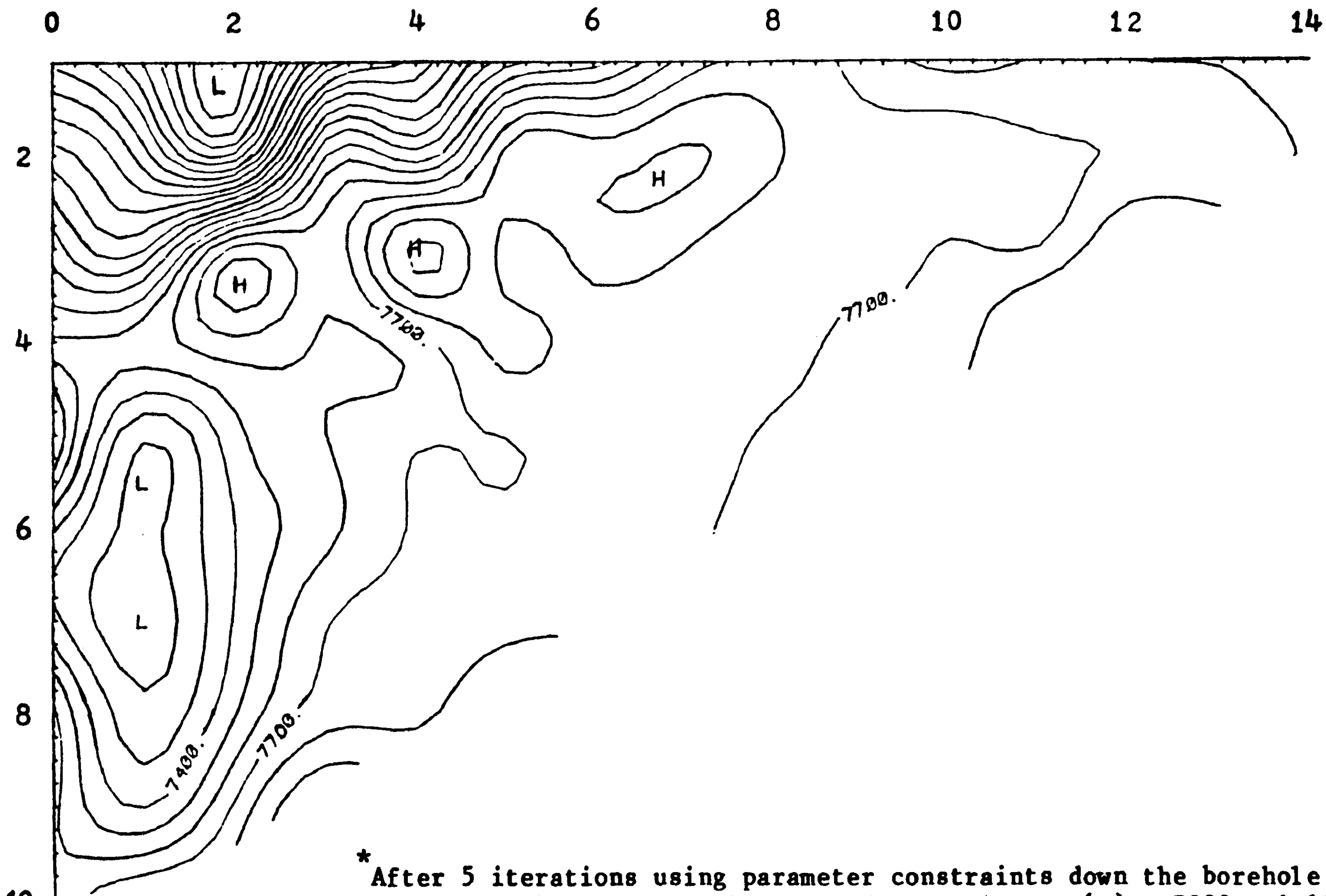
Reconstructed Velocity Field*



*After 1 iteration using parameter constraints down the borehole and curved rays based on the background velocity - $v(z) = 7000 + 1.11(z_j)$ ft./sec. Poorly-constrained parameters along the surface were eliminated from the image.

Figure 22a. Acoustic tomography from the Pierre Shale VSP data.

Reconstructed Velocity Field*



*After 5 iterations using parameter constraints down the borehole and curved rays based on the background veolocity - $v(z) = 7000 + 1.11(z_i)$ ft./sec. Poorly-constrained parameters along the surface were eliminated from the image.

Figure 22b. Acoustic tomography from the Pierre Shale VSP data.

tion (Fig. 23). Where the synthetic curve arrives after the observed times, a velocity increase is required. Where the observed times lag behind the synthetic, lower velocities are expected. Relative to the synthetic model, we should expect higher velocities at the large offset distances, but lower velocities at small offsets and large depth.

Because surface velocities are typically very slow, the high surface velocity is contrary to expectations. Three feasible explanations arise for this velocity anomaly. Either the velocity structure exists, or the data or the collection of the data is questionable, or the basic theory is inadequate for the earth conditions exhibited by the Pierre Shale. Poorly placed shot locations could result in an artificial high at the surface, if the errant shot locations resulted in a shorter raypath.

A more feasible explanation arises from the strong anisotropy exhibited by the Pierre Shale. After using the same data collected near Brush, Colorado, White, Martineau-Nicoletis, and Monash (1983) concluded that the horizontal velocity component was 10 to 20% faster than the vertical velocity component. Qualitatively, the rays from the large offset sources possess mostly a horizontal component and thus, should exhibit a greater than expected velocity. Similarly, the deeper traveling rays are mostly vertical and would travel at the slower velocity. Therefore, the anisotropy of the Pierre Shale explains the high surface

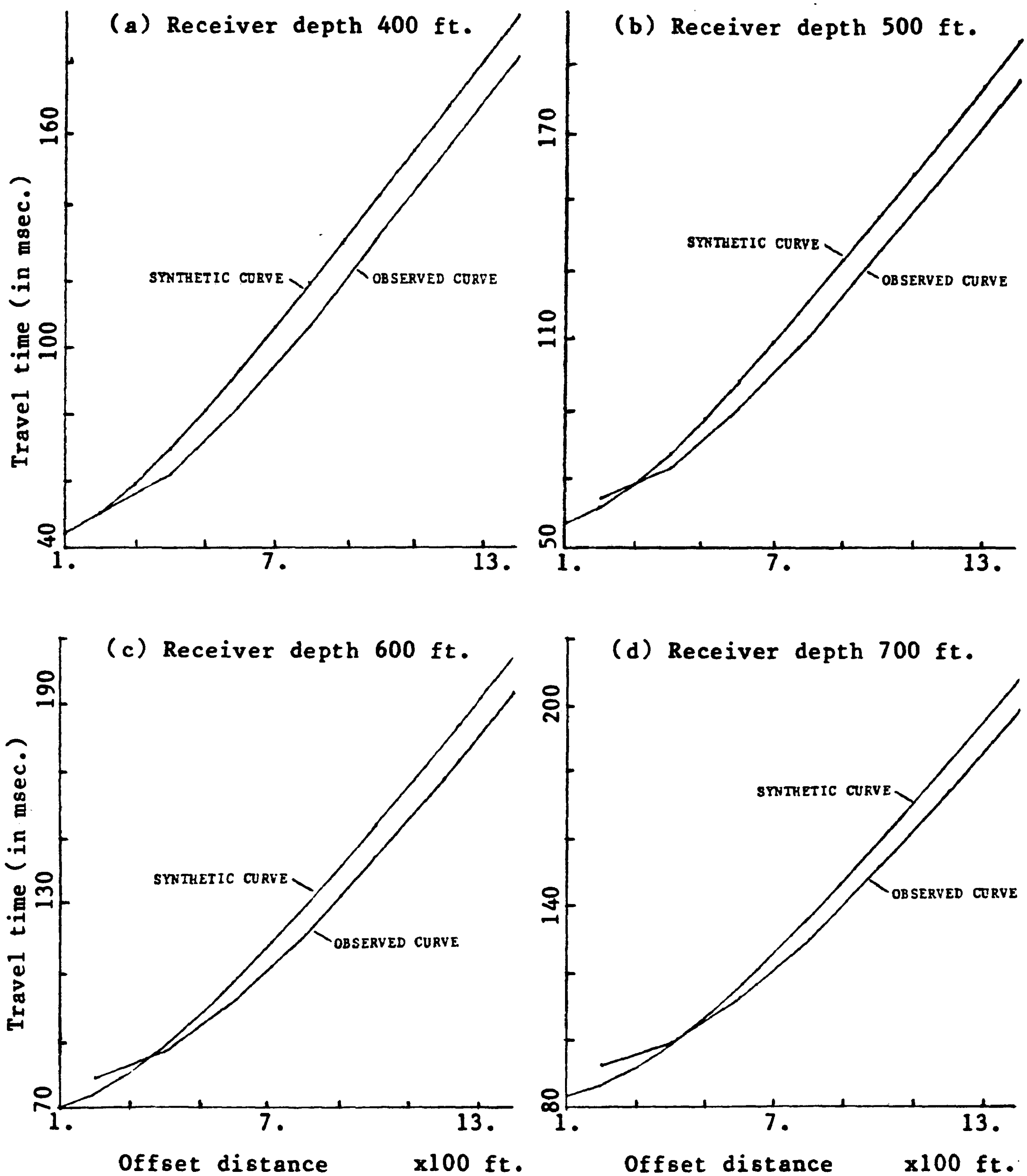


Figure 23. Travel time curves.

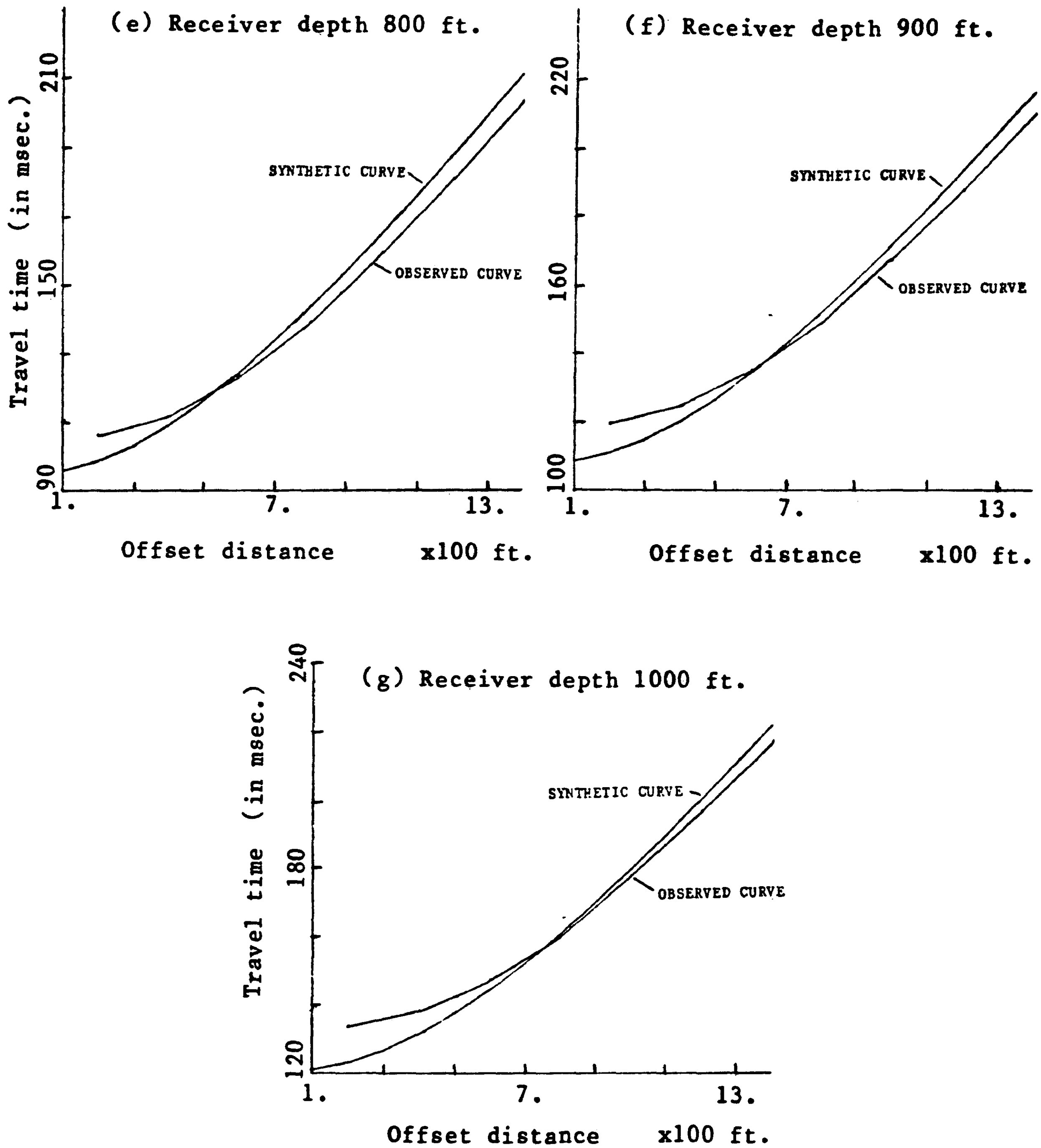


Figure 23. Travel time curves.

velocity and the low velocity cell at depth of the reconstructed velocity fields.

6. CONCLUSIONS

Algebraic reconstruction technique (ART) concentrates on the production of a reconstructed field whose projected data (travel times) agree with the observed data. This reconstructed field is modified by altering the data for each ray such that when this data is back-projected, the new image agrees with the original data.

ART possesses many inherent advantages. The flexibility of the algorithm easily allows for the incorporation of constraints. For example, sonic log information predetermines the velocity parameters down the borehole. This flexibility also permits the application of ART to any field setup of sources and receivers.

With noiseless data, ART picks a projection which agrees with the observed data. It is not in ART's nature to introduce spurious images on good data sets. However, the resolution of the image is affected by the orientation of the anomaly relative to the direction of the rays. While good resolving power occurs perpendicular to the rays, ART tends to smooth the image along the direction of the rays.

With noisy data, ART can produce single point anomalies of highly unrealistic velocities among the poorly-constrained parameters along the edges of the model. ART will produce a reasonable reconstructed velocity field when constraints are incorporated from the well log information or by simply ignoring any poorly-constrained parameters within the image. Lastly, data, which was collected over a strongly anisotropic medium, will result in poor velocity reconstructions, because such data defies the assumption of isotropy which was used in these algorithms.

REFERENCES

Aki, R. and P. G. Richards, Quantitative Seismology: Theory and Methods, Vol. II, W. H. Freeman and Company: San Francisco, (1980).

Christofferson, A. and E. S. Husebye, "On 3-D inversion of P-wave time residuals option for geological modeling," J. Geophys. Res., 84 (1979) 6168-6176.

Dines, K. A. and R. J. Lytle, "Computerized geophysical tomography," Proc. IEEE, 67 (1979) 1065-1073.

Fawcett, J. A., "I. Three dimensional ray-tracing and ray-inversion in layered media. II. Inverse scattering and curved ray tomography with applications to seismology." PhD Thesis: California Institute of Technology, Pasadena, (1983) 229 pp.

Gordon, R., "A tutorial on ART," IEEE Trans. Nucl. Sci., NS-21 (June 1974) 78-93.

Gordon, R., R. Bender, and G. T. Herman, "An algebraic reconstruction technique (ART) for 3-D electron microscopy and x-ray photography," J. Theor. Biol., 29 (1970) 471-518.

Herman, G. T., A. Lent, and S. W. Rowland, "ART: Mathematics and applications," J. Theor. Biol., 42 (1973) 1-32.

Horn, B. K. P., "Density reconstruction using arbitrary ray-sampling schemes," Proc. IEEE, 66 (1978) 551-562.

Mason, I. M., "Algebraic reconstruction of a two-dimensional velocity inhomogeneity in the High Hazles seam of Thoresby colliery," Geophysics, 46 (1981) 298-308.

McMechan, G. A., "Seismic tomography in boreholes," Geophys. J., 74 (1983) 601-621.

Mersereau, R. M., and A. V. Oppenheim, "Digital reconstruction of multidimensional signals from their projections," Proc. IEEE, 62 (1974) 1319-1338.

Mueller, R. K., M. Kaveh, and G. Wade, "Reconstructive tomography and applications to ultrasonics," Proc. IEEE, 67 (1979) 567-587.

Neumann, G., "Determination of lateral inhomogeneities in reflection seismics by inversion of travel time residuals," Geophys. Prosp., 29 (1981) 161-177.

Sixta, D. P., "Comparision and analysis of downgoing waveforms from land seismic sources," M.S. Thesis: Colorado School of Mines, Golden, (1982) 451 pp.

White, J. E., L. Martineau-Nicoletis, and C. Monash, "Measured anisotopy in Pierre Shale," Geophys. Prosp., 31 (1983) 709-725.

APPENDIX A: RAY PARAMETER DERIVATION

The ray parameter within a linear c(z) medium is determined from the Wiechert-Herglotz integral (Aki and Richards, 1980).

$$x_i = p \int_0^{z_j} \frac{dz}{\sqrt{\eta^2 - p^2}}, \qquad \text{(A-1)}$$

where $\eta = \frac{1}{c(z)}$ is the slowness,

x_i is the source offset,

z_j is the receiver depth.

The medium is assumed to possess a linear velocity structure.

$$c(z) = c_o + m_o z; \qquad c_o = c(0), \text{ the surface velocity.} \qquad \text{(A-2)}$$

Substituting (A-2) into (A-1), we obtain

$$x_i = p \int_0^{z_j} \frac{(c_o + m_o z)dz}{\sqrt{1 - p^2 (c_o + m_o z)^2}}. \qquad \text{(A-3)}$$

The integration may be carried out explicitly. The result is

$$x_i = \frac{1}{m_o p} \sqrt{1 - p^2 c^2(z)} \Big|_{z=0}^{z=z_j}, \qquad \text{(A-4)}$$

$$x_i = \frac{1}{m_o p}\left[\sqrt{1 - p^2c^2(z_j)} - \sqrt{1 - p^2 c_o^2}\right]. \quad \text{(A-5)}$$

Squaring both sides, we find

$$(x_i p m_o)^2 = (1-p^2c_o^2) + (1-p^2c^2(z_j)) - 2\sqrt{(1-p^2c_o^2)(1-p^2c^2(z_j))}, \quad \text{(A-6)}$$

or

$$(x_i p m_o)^2 - 2 + p^2(c_o^2 + c^2(z_j)) = -2\sqrt{(1 - p^2c_o^2)(1 - p^2c^2(z_j))}. \quad \text{(A-7)}$$

Again squaring both sides, we find

$$p^4[m_o^2x_i^2 + c_o^2 + c^2(z_j)]^2 - 4c_o^2p^2[m_o^2x_i^2 + c_o^2 + c^2(z_j)] + 4 = 4(1-p^2c_o^2)(1-p^2c^2(z_j)), \quad \text{(A-8)}$$

or

$$p^4\Big[[m_o^2x_i^2 + c_o^2 + c^2(z_j)]^2 - 4c_o^2c^2(z_j)\Big] - 4p^2m_o^2x_i^2 = 0. \quad \text{(A-9)}$$

Solving for p, we obtain

$$p^2 = 0, \qquad p^2 = \frac{4m_o^2x_i^2}{[m_o^2x_i^2 + c_o^2 + c^2(z_j)]^2 - 4c_o^2c^2(z_j)}. \quad \text{(A-10)}$$

However, the ray parameter must be positive for a linearly increasing medium. Thus, we conclude that

$$p = \frac{2\,m_o\,x_i}{\sqrt{[m_o^2 x_i^2 + c_o^2 + c^2(z_j)]^2 - 4\,c_o^2 c^2(z_j)}}. \tag{A-11}$$

APPENDIX B: TRAVEL TIME CALCULATION

The travel time in a linear c(z) medium is calculated by the Wiechert-Herglotz inversion method (Aki and Richards, 1980).

$$T_{ij} = \int_0^{z_j} \frac{dz}{c(z)\sqrt{1 - p^2c^2(z)}}. \qquad \text{(B-1)}$$

We introduce the new variable of integration θ:

$$p\,c(z) = \sin\theta, \qquad \text{(B-2)}$$

$$\sqrt{1 - p^2c^2(z)} = \cos\theta. \qquad \text{(B-3)}$$

Differentiating (B-3), we obtain

$$p\,m_o\,dz = \cos\theta\;d\theta. \qquad \text{(B-4)}$$

Substituting (B-2) and (B-4) into (B-1) yields:

$$T_{ij} = \frac{1}{m_o}\int\frac{d\theta}{\sin\theta}. \qquad \text{(B-5)}$$

$$T_{ij} = \frac{-1}{m_o} \ln [\csc \theta + \cot \theta]. \tag{B-6}$$

In terms of z, this yields the result

$$T_{ij} = \frac{1}{m_o} \ln \left[\frac{1 + \sqrt{1 - p^2c^2(z)}}{pc(z)}\right]\Bigg|_{z=z_j}^{z=0}, \tag{B-7}$$

$$T_{ij} = \frac{1}{m_o} \ln \left[\frac{c(z_j)\,(1 + \sqrt{1 - p^2c_o^2}\,)}{c_o\,(1 + \sqrt{1 - p^2c^2(z_j)}\,)}\right]. \tag{B-8}$$

APPENDIX C: COMPUTER PROGRAM LISTING

```
**********************************************************************
*                                                                    *
*      VSPART: Vertical Seismic Profile inversion of travel time     *
*              data by Algebraic Reconstruction Technique            *
*                                                                    *
*      Programmed by KLS--May 1984                                   *
*                                                                    *
*      ****    VARIABLE USAGE  *****   VARIABLE USAGE  *****         *
*                                                                    *
*      CO      Initial velocity at the surface (in ft/sec)           *
*      C1      Increase in velocity with a 'dx' increase in depth    *
*              (in ft/sec-grid)                                      *
*      CMIN    Minimum acceptable chord length for ART               *
*      DCOM    Valid commands                                        *
*      DFILE   Data file name                                        *
*      DX      Spacing between geophones and shots (in ft)           *
*              also conversion factor -- x ft=1 grid unit            *
*      ERLAST  Previous iteration's mean square error                *
*      ERROR   Mean square error in calculated times                 *
*      ICOM    Inputted command                                      *
*                                                                    *
*      LART    .FALSE.         Use first guess ART routine           *
*              .TRUE.          Use iterative ART routine             *
*      LBACK   .FALSE.         Constant velocity background          *
*              .TRUE.          Linear velocity background            *
*                                                                    *
*      M       # of source locations                                 *
*      MROW    Maximum # of sources                                  *
*      N       # of recievers down borehole                          *
*      NCOL    Maximum # of recievers                                *
*      NDCOM   # of valid commands                                   *
*      N ITER  # of iterations performed                             *
*      RDELTA  Length of 'dx' segment for integration                *
*                                                                    *
*      *****   COMMON STORAGE  *****                                 *
*                                                                    *
*      SLOW()  Slowness at parameter point m,n (in sec/grid)         *
*                                                                    *
*      TIME()  First arrival time from m th source to n th receiver  *
*                                                                    *
*      LCONST  .FALSE. Unconstrained parameters down well            *
*              .TRUE.  Constrained parameters down well              *
*      VELVAR  Allowable slowness deviation from constrained value   *
*      CONST() Constrained parameter values                          *
*                                                                    *
```

```
********************************************************************************
        PARAMETER        (MROW=20,NCOL=10)
        IMPLICIT         LOGICAL(L)
        INTEGER          L
        CHARACTER        DFILE*10,       DCOM*65,        ICOM*5
        COMMON/CONSTR/   LCONST,VELVAR,CONST(NCOL)
        COMMON/TIMES/    TIME(MROW,NCOL)
        COMMON/VELOCI/   SLOW(0:MROW,0:NCOL)
        DATA DCOM/'      ART  BACKGCONSTDATA EXIT HELP KEYINMAP  PARAM
     *TIME VELOC     '/
        DATA    NDCOM/11/,       LART/.FALSE./,  LCONST/.FALSE./
        DATA    CMIN/.1/,        RDELTA/.1/

*****   Command input   *****

        TYPE 1000
10      TYPE 1001
        ACCEPT '(A5)',  ICOM
1000    FORMAT('1',79('*')//T25,'Vertical Seismic Profiling'/T24,
     A'Inversion of Travel Time Data'/T20,'By Algebraic Reconstruction
     B Technique'//1X,79('*')//T23,'Programmed by KLS - - July 1984')
1001    FORMAT(/1X,'5-letter command?',2X,$)

*****   Command branching         *****
*****   Accepts unique abbr. for commands       *****

        CALL ABBREV      (DCOM,NDCOM,ICOM,5)
        IF(ICOM.EQ.'ART  ')      GOTO 200
        IF(ICOM.EQ.'BACKG')      GOTO 350
        IF(ICOM.EQ.'CONST')      GOTO 450
        IF(ICOM.EQ.'DATA ')      GOTO 100
        IF(ICOM.EQ.'EXIT ')      STOP
        IF(ICOM.EQ.'HELP ')      GOTO 50
        IF(ICOM.EQ.'KEYIN')      GOTO 150
        IF(ICOM.EQ.'MAP  ')      GOTO 400
        IF(ICOM.EQ.'PARAM')      GOTO 60
        IF(ICOM.EQ.'TIME ')      GOTO 250
        IF(ICOM.EQ.'VELOC')      GOTO 300
        TYPE 1010
        GOTO 10
1010    FORMAT(1X,'INVALID COMMAND - - - Type HELP for legal commands')

*****   Help command    *****

50      TYPE 1050
        GOTO 10
```

```
1050    FORMAT(/1X,'***** LEGAL COMMANDS *****'//
     A1X,'ART    Algebraic reconstruction technique - iterate once'/
     B1x,'BACKG  Choose linear or constant velocity background'/
     C1X,'CONST  Constrain the slowness parameters down the well'/
     D1X,'DATA   Input travel time date from data file'/
     E1X,'EXIT   End program'/
     H1X,'HELP   Prints this list'/
     K1X,'KEYIN  Keyin travel time data'/
     M1X,'MAP    Creates a datafile for the contouring programs'/
     P1X,'PARAM  Change parameters for ART'/
     T1X,'TIME   Print travel time grid onto the screen'/
     V1X,'VELOC  Print velocity grid onto the screen'/
     Z/1X,'Accepts unique abbrev. for any commands')

*****   Parameter command       *****

60      TYPE 1060
        TYPE 1062µ      ACCEPT *, RDELTAµ       RDELTA=RDELTA/DX
        TYPE 1064µ      ACCEPT *, CMINµ   CMIN=SQRT(2.0)*CMIN/100.
        GOTO 10
1060    FORMAT(/1X,'Routine to change parameters used in ART')
1062    FORMAT(/1x,'In the forward model, travel times are computed by i
     *ntegrating the slowness'/1X,'along the raypath.'//1X,'Enter the le
     *ngth of a πdxπ segment (in feet) for integration.',2X,∫)
1064    FORMAT(/1X,'In calculating the average slownesses, problems aris
     *e, due to computer errors,'/1X,'whenever the raypath just barely i
     *ntesects a parameter circle.  To avoid this'/1X,'problem, the data
     * is ignored if the intersecting chord length is smaller than'/1x,'
     *some minimum value.'//1x,'Enter this minimum value as a percentage
     * of the maximum value.',2X,∫)

*****   Data command    *****

100     TYPE 1100
        ACCEPT 1101,    DFILE
        OPEN(UNIT=1,DEVICE='DSK',FILE=DFILE,DISPOSE='SAVE')
        READ(1,1102,END=10)     M,N,DX,((TIME(I,J),I=1,M),J=1,N)
        LART=.FALSE.µ   N ITER=0
        GOTO 10
1100    FORMAT(/1X,'Input data file name (ie DATA.DAT)',2X,∫)
1101    FORMAT(A10)
1102    FORMAT(T20,'ART.FOR  Datafile'/'# of sources=',I2,5X,
     *  '# of receivers=',I2,5X,'Spacing=',F5.1,2X,'ft'/(5F10.5))

*****   Keyin command   *****

150     TYPE 1150µ      ACCEPT *, M
        TYPE 1151µ      ACCEPT *, N
```

```
        TYPE 1152μ       ACCEPT *, DX
        TYPE 1155
        DO 160  I=1,M
        DO 160  J=1,N
                TYPE 1160, I,J
                ACCEPT *, TIME(I,J)
160     CONTINUE
        TYPE 1100
        ACCEPT 1101,     DFILE
        OPEN(UNIT=1,DEVICE='DSK',FILE=DFILE)
        WRITE(1,1102)    M,N,DX, ((TIME(I,J),I=1,M),J=1,N)
        LART=.FALSE.μ    N ITER=0
        GOTO 10
1150    FORMAT(/1X,'KEYIN DATA ROUTINE'//1X,'Input # of sources ',∫)
1151    FORMAT(/1X,'Input # of receivers',2X,∫)
1152    FORMAT(/1X,'Input spacing (ft) between geophones/shots',2X,∫)
1155    FORMAT(/1X,'Input first arrival times - Enter zero (0.0) for
     * no data')
1160    FORMAT(1X,'Source=',I2,5X,'Receiver=',I2,5X,'Time=?',2X,∫)

*****   ART command     *****

200     IF(LBACK)        THEN
                IF(LART)         THEN
                        CALL ARTLB2(M,N,RDELTA,CMIN,C0/DX,C1/DX,ERROR)
                ELSE
                        CALL ARTLB1(M,N,RDELTA,CMIN,C0/DX,C1/DX,ERROR)
                        LART=.TRUE.
                END IF
        ELSE
                IF(LART)         THEN
                        CALL ART2        (M,N,RDELTA,CMIN,ERROR)
                ELSE
                        CALL ART1        (M,N,RDELTA,CMIN,ERROR)
                        LART=.TRUE.
                END IF
        END IF
        TYPE 1200,       ERROR
        N ITER=N ITER + 1
        IF(N ITER.GT.1) TYPE 1210,       ERLAST
        ERLAST=ERROR
        GOTO 10
1200    FORMAT(/1X,'Mean square error is ',F12.7)
1210    FORMAT(1X,'Last error value was ',F12.7)

*****   Time command    *****

250     TYPE 1250
```

```
        IF(M.LE.9)       THEN
                M1=1μ   M2=M
        ELSE
                TYPE 1301μ       ACCEPT *, M1
                TYPE 1302μ       ACCEPT *, M2
        END IF
        IF(N.LE.20)      THEN
                N1=1μ   N2=N
        ELSE
                TYPE 1303μ       ACCEPT *, N1
                TYPE 1304μ       ACCEPT *, N2
        END IF
        TYPE 1310, (I,I=M1,M2)μ          TYPE 1311
        DO 260  J=N1,N2
                TYPE 1260,       J,(TIME(I,J),I=M1,M2)
260     CONTINUE
        GOTO 10
1250    FORMAT(/1X,'TRAVEL TIME INPUT'/)
1260    FORMAT(1X,I2,3X,14F8.5)

*****   Velocity command         *****

300     TYPE 1300
        IF(M.LE.8)       THEN
                M1=0μ   M2=M
        ELSE
                TYPE 1301μ       ACCEPT *, M1
                TYPE 1302μ       ACCEPT *, M2
        END IF
        IF(N.LE.19) THEN
                N1=0μ   N2=N
        ELSE
                TYPE 1303μ       ACCEPT *, N1
                TYPE 1304μ       ACCEPT *, N2
        END IF
        TYPE 1305, N ITER
        TYPE 1310,  (I,I=M1,M2)μ         TYPE 1311
        DO 310  J=N1,N2
                TYPE 1312,       J,(VEL(DX,SLOW(I,J)),I=M1,M2)
310     CONTINUE
        GOTO 10
1300    FORMAT(/1X,'*****  Routine to Print Velocity Structure  *****')
1301    FORMAT(/1X,'Input starting shot point to print',2X,$)
1302    FORMAT(/1X,'Input ending shot point to print',2X,$)
1303    FORMAT(/1X,'Input starting geophone to print',2X,$)
1304    FORMAT(/1X,'Input ending geophone to print',2X,$)
1305    FORMAT(//T5,'*****   Velocity Structure After',I3,' Iterations',
     *3X,5('*'))
```

```
1310    FORMAT(/T20,'Shot Points'/T11,14(I2,6X))
1311    FORMAT(1X,'Depth')
1312    FORMAT(1X,I2,3X,14F8.0)

*****   Background command       *****

350     TYPE 1350
        ACCEPT '(A5)', ICOMµ    CALL UCASE2(.TRUE.,5,ICOM)
        IF(ICOM(1:1).EQ.'C')    GOTO 360
        IF(ICOM(1:1).EQ.'L')    GOTO 370
        GOTO 350
360     LBACK=.FALSE.µ  GOTO 10
370     LBACK=.TRUE.
        TYPE 1370µ      ACCEPT *, C0
        TYPE 1380µ      ACCEPT *, C1
        TYPE 1390
        GOTO 10
1350    FORMAT(/1X,'CONSTANT or LINEAR velocity background?',2X,∫)
1370    FORMAT(/1X,'Input initial velocity (in feet/sec)',2X,∫)
1380    FORMAT(/1X,'Input velocity increase w/ geophone spacing',2X,∫)
1390    FORMAT(/T20,'***  WARNING  ***'/1X,'If the velocity background
     *is nearly linear, then variable overflows may occur'//
     *T20,'***  WARNING  ***'/1X,'If the linear velocity background is
     *too steep, then the rays will turn and'/1x,'travel upwards.
     *Square roots of negative numbers occur.')

*****   Map command      *****

400     TYPE 1400
        ACCEPT '(A10)', DFILE
        OPEN(UNIT=1,DEVICE='DSK',FILE=DFILE)
        DO 410  K=0,M
        DO 410  L=0,N
        IF(SLOW(K,L).LE.0.0)    GOTO 410
        WRITE(1,*)      FLOAT(K),       -FLOAT(L),      DX/SLOW(K,L)
410     CONTINUE
        CLOSE(UNIT=1)
        GOTO 10
1400    FORMAT(/1X,'*****   MAPPING ROUTINE   *****'//1X,
     *'A datafile is created for input into the contouring programs'
     *//T10,'MNL:KGRID,  MNL:FIT,  and  MNL:TOPO'
     *//1X,'Enter datafile name.ext (ie KGRID.DAT)',2X,∫)

*****   Constrain parameters command     *****

450     TYPE 1450
        ACCEPT '(A5)',  ICOMµ   CALL UCASE2(.TRUE.,5,ICOM)
        IF(ICOM(1:1).EQ.'U')    GOTO 460
```

```
        IF(ICOM(1:1).EQ.'C')    GOTO 470
        GOTO 450
460     LCONST=.FALSE.µ GOTO 10
470     LCONST=.TRUE.
        TYPE 1470
        DO 480  J=1,N
                TYPE 1480, J*DX
                ACCEPT *,  S
                CONST(J)=DX*S/1000000.0 ! in sec/grid units
480     CONTINUE
        TYPE 1490
        ACCEPT *, VELVAR
        GOTO 10
1450    FORMAT(/1X,'CONSTrain or UNCONstrain the slowness parameters
     * down the well?',2x,∫)
1470    FORMAT(/1X,'Input the slowness in micro seconds per foot for
     * the depth given'/1X,'Enter zero for an unconstrained parameter')
1480    FORMAT(T10,'depth',F6.0,1X,'ft',5X,'slowness?',2X,∫)
1490    FORMAT(/1X,'Enter the allowable deviation of these constrained
     * parameters'/1X,'(ie 0.0 for a completely constrained parameters'/
     */1X,'Enter value in percent',2X,∫)

        END


        SUBROUTINE ART1 (MROW,NCOL,DELTAR,CMIN,ERROR)
**********************************************************************
*                                                                    *
*       ART: Algebraic Reconstruction Technique                      *
*               Programmed by KLS  March 1984                        *
*                                                                    *
*       REF: Mason, I. M., πAlgebraic Reconstruction of a Two        *
*               Dimensional Velocity Inhomogeneity in the High Hazles *
*               Seam of Thoresby Collieryπ Geophysics 46 (1981) 298-308*
*                                                                    *
*       *****   CALLING VARIABLES       *****                        *
*                                                                    *
*       MROW    # of sources along the x axis                        *
*       NCOL    # of receivers along the z axis                      *
*       DELTAR  Approx length of ray segment for integration         *
*       CMIN    Minimum acceptable chord length to use to calculate  *
*               the average slownesses                               *
*                                                                    *
*       *****   OUTPUT VARIABLES        *****                        *
*                                                                    *
*       ERROR   Mean square error                                    *
*                                                                    *
```

```
*       *****   COMMON STORAGE  *****   INPUT   *****                   *
*                                                                       *
*       LCONST  Logical switch for constraining the slowness parameters*
*               down the borehole (known from the sonic log)            *
*                                                                       *
*       TIME()  First arrival times from m'th source to n'th geophone   *
*               Zero travel time signifies a lack of data               *
*                                                                       *
*       *****   COMMON STORAGE  *****   OUTPUT  *****                   *
*                                                                       *
*       SLOW(M,N)       Slowness array                                  *
*                                                                       *
*       TCALC(M,N)      Calculated travel times                         *
*                                                                       *
*************************************************************************
        PARAMETER        (M=20,N=10)
        LOGICAL          LCONST
        DIMENSION        CSUM(0:M,0:N),TSUM(0:M,0:N)
        COMMON/CONSTR/   LCONST
        COMMON/VELOCI/   SLOW(0:M,0:N)
        COMMON/CALCUL/   TCALC(0:M,0:N)
        COMMON/TIMES/    TIME(M,N)
        EQUIVALENCE      (TSUM(0,0),TCALC(0,0))
*************************************************************************
*                                                                       *
*       *****   VARIABLE USAGE  *****   VARIABLE USAGE  *****           *
*                                                                       *
*       A       X-squared coeffecent of quadratic equation              *
*       B       X coefficient of quadratic equation                     *
*       C       Coefficient of quadratic equation                       *
*       CSUM()  Sum of chord lengths around m,n parameter               *
*       CLEN    Chord length from X1,Z1 to X2,Z2                        *
*       ERR     Residual error                                          *
*       I       Position of source                                      *
*       J       Position of reciever                                    *
*       K       X parameter coordinate                                  *
*       L       Z parameter coordinate                                  *
*       NSEG    # of ray segments                                       *
*       RAD     Radical of quadratic solution                           *
*       RLEN    Length of ray path from ith source to jth receiver      *
*       SEG     True length of ray path                                 *
*       TSUM()  Sum of times along chord lengths around m,n             *
*       X,Z     Coordinates of midpoint of ray segment                  *
*       X1,Z1   Point of chord on circle                                *
*       X2,Z2   Second point of chord on circle                         *
*                                                                       *
*       *****   UNITS OF DIMENSIONS     *****                           *
*                                                                       *
```

```
*       Distance        in grid units (spacing between geophones)      *
*       Time            in seconds                                     *
*       Slowness        in seconds/grid unit                           *
*                                                                      *
************************************************************************

*****   Clear sumations *****

        DO 05   K=0,MROW
        DO 05   L=0,NCOL
                TSUM(K,L)=0.0
                CSUM(K,L)=0.0
05      CONTINUE

*****   For each ray path...    *****

        DO 20   I=1,MROW
        DO 20   J=1,NCOL

*****   For no data on ray      *****

                IF(TIME(I,J).LE.0.0)    GOTO 20

*****   Ray length      *****

                RLEN=SQRT(FLOAT(I*I + J*J))

*****   For each parameter circle...    *****

                ZRANGE=1.0 / (SQRT(2.0)*SIN(ATAN(I/FLOAT(J))))
        DO 10   K=0,I
                Z=J - J*K/FLOAT(I)
                L1=NINT(Z-ZRANGE+0.5)µ  IF(L1.LT.0)     L1=0
                L2=Z+ZRANGEµ    IF(L2.GT.J)     L2=J
        DO 10   L=L1,L2

*****   Find intersection of two equations      *****
*****   Equation one: Circle (X-K)**2 + (Z-L)**2 = 1/2
*****   Equation two: Line   Z = J - J*X/I
*****   Solve quadratic equation        AX**2 + 2BX + C

                A=1.0 + J*J/FLOAT(I*I)
                B=(L-J)*J/FLOAT(I) - K
                C=K*K + (J-L)**2 - 0.5
                RAD=B*B - A*C

*****   Imaginary solution means the lines do not intersect     *****
```

```
                IF(RAD.LE.0.0)  GOTO 10
                RAD=SQRT(RAD)

*****   Locate intersection points      *****
*****   B.C. - Ray path starta at (I,0) and ends at (0,J) *****

                IF(K.EQ.I .AND. L.EQ.0) THEN
                        X1=Kμ   Z1=0
                ELSE
                        X1=(-B+RAD)/A
                        Z1=J-J*X1/I
                END IF
                IF(K.EQ.0 .AND. L.EQ.J) THEN
                        X2=0μ   Z2=L
                ELSE
                        X2=(-B-RAD)/A
                        Z2=J-J*X2/I
                END IF

*****   Chord length    *****

                CLEN=SQRT((X1-X2)**2 + (Z1-Z2)**2)
                IF(CLEN.LT.CMIN)        GOTO 10

*****   Summation of parameter times    *****

                TSUM(K,L)=TSUM(K,L)+CLEN*TIME(I,J)/RLEN

*****   Summation of chord lengths      *****

                CSUM(K,L)=CSUM(K,L)+CLEN
10      CONTINUE
20      CONTINUE

*****   Average slowness at all points  *****

        DO 30   K=0,MROW
        DO 30   L=0,NCOL
                IF(CSUM(K,L).EQ.0.0)    THEN
                        SLOW(K,L)=0.0
                ELSE
                        SLOW(K,L)=TSUM(K,L)/CSUM(K,L)
                END IF
30      CONTINUE

*****   Constrained parameters  *****

        IF(LCONST)      CALL PARCON(NCOL)
```

```
*****   For each ray path...    *****

        DO 50   I=1,MROW
        DO 50   J=1,NCOL
                IF(TIME(I,J).LE.0.0)    GOTO 50
                TCALC(I,J)=0.0

*****   Divide ray into equal segments  *****

                RLEN=SQRT(FLOAT(I*I + J*J))
                NSEG=INT(RLEN/DELTAR)
                SEG=RLEN/NSEG

*****   Interpolate slowness at midpoint of segments    *****

                XINC=FLOAT(I)/FLOAT(NSEG)
                XSTART=XINC/2.0
                XEND=FLOAT(I)
                DO 40   X=XSTART,XEND,XINC
                        Z=J-J*X/FLOAT(I)
                        TCALC(I,J)=TCALC(I,J) + CARINT(X,Z)*SEG
40              CONTINUE
50      CONTINUE

*****   Mean square error       *****

        ERROR=0.0
        DO 60   I=1,MROW
        DO 60   J=1,NCOL
                IF(TIME(I,J).LE.0.0)    GOTO 60
                ERR=TCALC(I,J)-TIME(I,J)
                ERROR=ERROR+ERR*ERR
60      CONTINUE
        RETURN
        END


        SUBROUTINE ART2 (MROW,NCOL,DELTAR,CMIN,ERROR)
**********************************************************************
*                                                                    *
*       ART2: Algebraic Reconstruction Technique                     *
*               iterative portion                                    *
*               Programmed by KLS  March 1984                        *
*                                                                    *
*       REF: Mason, Geophysics 1981                                  *
*                                                                    *
*       *****   CALLING VARIABLES       *****                        *
```

```
*                                                                      *
*       MROW    # of sources along the x axis                          *
*       NCOL    # of receivers along the z axis                        *
*       DELTAR  Approx. length of ray segment for integration          *
*       CMIN    Minimum acceptable chord length to use to calculate    *
*               average slownesses                                     *
*                                                                      *
*       *****   OUTPUT VARIABLES        *****                          *
*                                                                      *
*       ERROR   Mean square error                                      *
*                                                                      *
*       *****   COMMON STORAGE  *****   INPUT   *****                  *
*                                                                      *
*       LCONST  Logical switch for constraining the slowness parameters*
*               down the borehole (known from the sonic log)           *
*                                                                      *
*       TIME()  First arrival times from m'th source to n'th receiver  *
*               Zero travel time signifies a lack of data              *
*                                                                      *
*       *****   COMMON STORAGE  *****   I/O     *****                  *
*                                                                      *
*       SLOW(M,N)       Slowness array                                 *
*                                                                      *
*       TCALC(M,N)      Calculated travel times                        *
*                                                                      *
************************************************************************
        PARAMETER       (M=20,N=10)
        LOGICAL         LCONST
        DIMENSION       CSUM(0:M,0:N),ESUM(0:M,0:N)
        COMMON/CONSTR/  LCONST
        COMMON/VELOCI/  SLOW(0:M,0:N)
        COMMON/CALCUL/  TCALC(0:M,0:N)
        COMMON/TIMES/   TIME(M,N)
************************************************************************
*                                                                      *
*       *****   VARIABLE USAGE  *****   VARIABLE USAGE  *****          *
*                                                                      *
*       A       X-squared coefficent of quadratic equation             *
*       B       X coeffiecient of quadratic equation                   *
*       C       Coeffiecient of quadratic equation                     *
*       CSUM()  Sum of chord lengths around m,n parameter              *
*       CLEN    Chord length from x1,z1 to x2,z2                       *
*       ERR     Residual error                                         *
*       ESUM()  Sum of errors associated with parameter m,n            *
*       I       Position of source                                     *
*       J       Position of reciever                                   *
*       K       X parameter coordinate                                 *
*       L       Z parameter coordinate                                 *
```

```
*       NSEG     # of ray segments                                          *
*       RAD      Radical of quadratic solution                              *
*       RLEN     Length of ray path from ith source to jth receiver         *
*       SEG      True length of ray path                                    *
*       X,Z      Coordinates of midpoint of ray segment                     *
*       X1,Z1    Point of chord on circle                                   *
*       X2,Z2    Second point of chord on circle                            *
*                                                                           *
*       *****    UNITS OF DIMENSIONS     *****                              *
*                                                                           *
*       Distance         in grid units (spacing between geophones)          *
*       Time             in seconds                                         *
*       Slowness         in seconds/grid unit                               *
*                                                                           *
*****************************************************************************

*****   Clear summations        *****

        DO 10    K=0,MROW
        DO 10    L=0,NCOL
                 CSUM(K,L)=0.0
                 ESUM(K,L)=0.0
10      CONTINUE

*****   For each ray path...    *****

        DO 30    I=1,MROW
        DO 30    J=1,NCOL

*****   For no data on ray      *****

                 IF(TIME(I,J).LE.0.0)    GOTO 30
                 ERR=TIME(I,J)-TCALC(I,J)

*****   Ray length      *****

                 RLEN=SQRT(FLOAT(I*I + J*J))

*****   For each parameter...   *****

                 ZRANGE=1.0 / (SQRT(2.0)*SIN(ATAN(I/FLOAT(J))))
        DO 20    K=0,I
                 Z=J-J*K/FLOAT(I)
                 L1=NINT(Z-ZRANGE+0.5)µ  IF(L1.LT.0)     L1=0
                 L2=Z+ZRANGEµ     IF(L2.GT.J)     L2=J
        DO 20    L=L1,L2

*       Find intersection of two equations
```

```
*       Equation one: Circle (X-K)**2 + (Z-L)**2 = 1/2
*       Equation two: Line   Z = J - J*X/I
*       Solve Quadratic Equation        AX**2 + 2BX + C

                A=1.0 + J*J/FLOAT(I*I)
                B=(L-J)*J/FLOAT(I) - K
                C=K*K + (J-L)**2 - 0.5
                RAD=B*B - A*C

*****   Imaginary solution means the lines do not intersect    *****

                IF(RAD.LE.0.0)  GOTO 20
                RAD=SQRT(RAD)

*****   Locate intesection points       *****
*****   B.C. - Ray path starts at (I,0) and ends at (0,J) *****

                IF(K.EQ.I .AND. L.EQ.0) THEN
                        X1=Kμ   Z1=0
                ELSE
                        X1=(-B+RAD)/A
                        Z1=J-J*X1/I
                END IF
                IF(K.EQ.0 .AND. L.EQ.J) THEN
                        X2=0μ   Z2=L
                ELSE
                        X2=(-B-RAD)/A
                        Z2=J-J*X2/I
                END IF

*****   Chord length    *****

                CLEN=SQRT((X1-X2)**2 + (Z1-Z2)**2)
                IF(CLEN.LT.CMIN)        GOTO 20

*****   Summation of chord lengths      *****

                CSUM(K,L)=CSUM(K,L) + CLEN

*****   Summation of error around parameter     *****

                ESUM(K,L)=ESUM(K,L) + ERR*CLEN/RLEN
20      CONTINUE
30      CONTINUE

*****   Update parameter matrix *****

        DO 40   K=0,MROW
```

```
      DO 40    L=0,NCOL
               IF(CSUM(K,L).EQ.0.0)    THEN
                       SLOW(K,L)=0.0
               ELSE
                       SLOW(K,L)=SLOW(K,L) + ESUM(K,L)/CSUM(K,L)
               END IF
40    CONTINUE

*****   Constrained parameters   *****

      IF(LCONST)       CALL PARCON(NCOL)

*****   For each ray path...     *****

      DO 60    I=1,MROW
      DO 60    J=1,NCOL
               IF(TIME(I,J).LE.0.0)    GOTO 60
               TCALC(I,J)=0.0

*****   Divide ray into equal segments   *****

               RLEN=SQRT(FLOAT(I*I + J*J))
               NSEG=INT(RLEN/DELTAR)
               SEG=RLEN/NSEG

*****   Interpolate slowness at midpoint of segment     *****

               XINC=FLOAT(I)/FLOAT(NSEG)
               XSTART=XINC/2.0
               XEND=FLOAT(I)
               DO 50    X=XSTART,XEND,XINC
                       Z=J-J*X/FLOAT(I)
                       TCALC(I,J)=TCALC(I,J) + CARINT(X,Z)*SEG
50             CONTINUE
60    CONTINUE

*****   Mean square error       *****

      ERROR=0.0
      DO 70    I=1,MROW
      DO 70    J=1,NCOL
               IF(TIME(I,J).LE.0.0)    GOTO 70
               ERR=TCALC(I,J)-TIME(I,J)
               ERROR=ERROR+ERR*ERR
70    CONTINUE
      END
```

```
      SUBROUTINE ARTLB1      (MROW,NCOL,DELTAR,CMIN,C0,C1,ERROR)
************************************************************************
*                                                                      *
*     ARTLB1: Algebraic Reconatrustion Technique for a linear          *
*             velocity background (ie. curved raypaths)                *
*                                                                      *
*     Programmed by KLS--May 1984                                      *
*                                                                      *
*     *****   CALLING VARIABLES        *****                           *
*                                                                      *
*     MROW    # of sources along x-axis                                *
*     NCOL    # of recievers along z-axis                              *
*     DELTAR  Approx. length of ray segment for intergration           *
*     CMIN    Minimum acceptable chord length to use to calculate      *
*             average slownesses                                       *
*     C0      Velocity at surface                                      *
*     C1      Change in velocity with a 'dx' increase in depth         *
*                                                                      *
*     *****   OUTPUT VARIABLES         *****                           *
*                                                                      *
*     ERROR   Mean square error of residual travel times               *
*                                                                      *
*     *****   COMMON STORAGE  *****   INPUT   *****                    *
*                                                                      *
*     LCONST  Logical switch for constraining the slowness parameters*
*             down the borehole (known from the sonic log)             *
*                                                                      *
*     TIME()  First arrival times from the mth shot to the nth         *
*             geophone                                                 *
*             Zero travel time signifies a lack of data                *
*                                                                      *
*     *****   COMMON STORAGE  *****   OUTPUT  *****                    *
*                                                                      *
*     SLOW()  Slowness array                                           *
*                                                                      *
*     TCALC() Calculated travel times                                  *
*                                                                      *
************************************************************************
      PARAMETER         (M=20,N=10)
      LOGICAL           LCONST
      DIMENSION         CSUM(0:M,0:N),TSUM(0:M,0:N)
      COMMON/CONSTR/    LCONST
      COMMON/VELOCI/    SLOW(0:M,0:N)
      COMMON/TIMES/     TIME(M,N)
      COMMON/CALCUL/    TCALC(0:M,0:N)
      EQUIVALENCE       (TCALC(0,0),TSUM(0,0))
************************************************************************
*                                                                      *
```

```
*       *****   VARIABLE USAGE  *****   VARIABLE USAGE  *****           *
*                                                                       *
*       Alpha   Angle of ray path                                       *
*       Angle i Angle of incidence for ray at K,0                       *
*       Beta    Angle of intersection                                   *
*       CSUM()  Sum of chord lengths around m,n parameter               *
*       CLEN    Chord (arc) length intersecting parameter circle        *
*       DIST    Distance between parameter point and origin of          *
*               the radius of curvature for the raypath                 *
*       ERR     Residual error                                          *
*       I       Position of source                                      *
*       J       Position of reciever                                    *
*       K       X parameter coordinate                                  *
*       L       Z parameter coordinate                                  *
*       NSEG    # of ray segments                                       *
*       R       The radius of curvature for the ray                     *
*       RAYPAR  Ray parameter for the ray from ith shot to jth geophone*
*       RLEN    Length of ray path from i th source to j th receiver    *
*       SEG     True length of ray path                                 *
*       TSUM(M,N)       Sum of times along chord lengths around M,N     *
*       X,Z     Midpoint of segment for integration                     *
*       Xo,Zo   Origin of the radius of curvature                       *
*               always a negative value                                 *
*                                                                       *
*       *****   UNITS OF DIMENSIONS     *****                           *
*                                                                       *
*       Angles          in radians                                      *
*       Distance        in grid units (spacing between geophones)       *
*       Time            in seconds                                      *
*       Slowness        in seconds/grid unit                            *
*       Velocity        in grid units/second                            *
*                                                                       *
*************************************************************************

*****   Clear summations        *****

        DO 10   K=0,MROW
        DO 10   L=0,NCOL
                CSUM(K,L)=0.0
                TSUM(K,L)=0.0
10      CONTINUE

*****   For each ray path...    *****

        DO 30   I=1,MROW
        DO 30   J=1,NCOL

*****   For no data on ray      *****
```

```
          IF(TIME(I,J).LE.0.0)    GOTO 30

*****     Find radius of curvature for the ray    *****

          RAYPAR=2.0*C1*I / SQRT((C0*C0+(C0+J*C1)**2+C1*C1*I*I) **2 - 4*C0
     *    *C0*(C0+C1*J)**2 )
          R=1.0/(RAYPAR*C1)

*****     Find estimated travel time      *****

          T est=LOG((C0+C1*J)*(1+ SQRT(1 - (C0*RAYPAR)**2)) /
     *          (C0 + SQRT(C0*C0 - (C0*(C0+C1*J)*RAYPAR)**2))) / C1

*****     Use geometry    *****

          alpha=2.0*ASIN(0.5*SQRT(FLOAT(I*I + J*J))/R)
          angle i=        ATAN(I/FLOAT(J)) - alpha/2.0
          RLEN=R*Alpha
          Xo=-R*COS(angle i + alpha)
          Zo=-R*SIN(angle i)

*****     For each parameter circle...    *****

          DO 20   K=0,I
                  L1=NINT(Zo + SQRT((R-1./SQRT(2.))**2 - (K-Xo)**2) + .5)
                  IF(L1.LT.0)     L1=0
                  L2=INT(Zo + SQRT((R+1./SQRT(2.))**2 - (K-Xo)**2))
                  IF(L2.GT.J)     L2=J
          DO 20   L=L1,L2

*         Find intersection of two equations
*         Equation one: Circle (X-K)**2 + (Z-L)**2 = 1/2
*         Equation two: Circle (X-Xo)**2 + (Z-Zo)**2 = R**2

*****     Ray intersects parameter circle ???     *****

                  DIST=SQRT((K-Xo)**2+(L-Zo)**2)
                  IF(ABS(DIST - R) .GE. 1./SQRT(2.0))    GOTO 20

*****     Locate intersection angle       *****

*****     Law of cosines  *****

*****     TEMP avoids computer roundoff errors    *****

                  TEMP=(R*R + DIST*DIST - 0.5) / (2.0*R*DIST)
                  IF(TEMP.GT.1.0) GOTO 20
                  beta=2.0*ACOS(TEMP)
```

```
*****   B.C. - Ray path starts at (I,0) and ends at (0,J) *****

                IF(K.EQ.I .AND. L.EQ.0)        beta=beta/2
                IF(K.EQ.0 .AND. L.EQ.J)        beta=beta/2

*****   Chord (arc) length      *****

                CLEN=R*beta
                IF(CLEN.LT.CMIN)        GOTO 20

*****   Summation of parameter times     *****

        TSUM(K,L)=TSUM(K,L)+TIME(I,J)*(CLEN/(C0+L*C1))/T est

*****   Summation of chord (arc) lengths        *****

                CSUM(K,L)=CSUM(K,L)+CLEN
20      CONTINUE
30      CONTINUE

*****   Average slowness at all points   *****

        DO 40   K=0,MROW
        DO 40   L=0,NCOL
                IF(CSUM(K,L).EQ.0.0)    THEN
                        SLOW(K,L)=0.0
                ELSE
                        SLOW(K,L)=TSUM(K,L)/CSUM(K,L)
                END IF
40      CONTINUE

*****   Constrained parameters   *****

        IF(LCONST)      CALL PARCON(NCOL)

*****   For each ray path...    *****

        DO 70   I=1,M
        DO 70   J=1,N
                IF(TIME(I,J).LE.0.0)    GOTO 70
                TCALC(I,J)=0.0

        RAYPAR=2.0*C1*I / SQRT((C0*C0+(C0+J*C1)**2+C1*C1*I*I) **2 - 4*C0
     *  *C0*(C0+C1*J)**2 )
        R=1.0/(RAYPAR*C1)
        alpha=2*ASIN(0.5*SQRT(FLOAT(I*I + J*J))/R)
        angle i=        ATAN(I/FLOAT(J)) - alpha/2.0
        RLEN=   R*alpha
```

```
        NSEG=   INT(RLEN/DELTAR)
        SEG=    RLEN/NSEG
        Xo=-R*COS(angle i + alpha)
        Zo=-R*SIN(angle i)

*****   Interpolate slowness at midpoints of segment    *****

        DO 60   S=SEG/2.0,RLEN,SEG
                beta=S/R
                X=R*COS(ANGLE I + BETA) + XO
                Z=R*SIN(ANGLE I + BETA) + ZO
                TCALC(I,J)=TCALC(I,J) + SEG*CARINT(X,Z)
60      CONTINUE
70      CONTINUE

*****   Mean square error        *****

        ERROR=0.0
        DO 80   I=1,MROW
        DO 80   J=1,NCOL
                IF(TIME(I,J).LE.0.0)    GOTO 80
                ERR=TCALC(I,J)-TIME(I,J)
                ERROR=ERROR+ERR*ERR
80      CONTINUE
        RETURN
        END


        SUBROUTINE ARTLB2       (MROW,NCOL,DELTAR,CMIN,C0,C1,ERROR)
***********************************************************************
*                                                                     *
*       ARTLB2: Algebraic Reconstruction Technique for a Linear       *
*               velocity Background (ie. curved raypaths)             *
*               Iterative portion                                     *
*                                                                     *
*       Programmed by KLS--May 1984                                   *
*                                                                     *
*       *****   CALLING VARIABLES       *****                         *
*                                                                     *
*       MROW    # of sources along x-axis                             *
*       NCOL    # of recievers along z-axis                           *
*       DELTAR  Approx. length of ray segment for integration         *
*       CMIN    Minimum acceptable chord length to use to calculate   *
*               average slownesses                                    *
*       C0      Velocity at surface                                   *
*       C1      Change in velocity with a 'DX' increase in depth      *
*                                                                     *
```

```
*       *****   OUTPUT VARIABLES        *****                              *
*                                                                          *
*       ERROR   Mean square error of residual travel times                 *
*                                                                          *
*       *****   COMMON STORAGE  *****   INPUT   *****                      *
*                                                                          *
*       LCONST  Logical switch for constraining the slowness parameters*
*               down the borehole (known from the sonic log)               *
*                                                                          *
*       TIME()  First arrival times from the m th shot to the n th         *
*               geophone                                                   *
*               Zero travel time signifies a lack of data                  *
*                                                                          *
*       *****   COMMON STORAGE  *****   I/O     *****                      *
*                                                                          *
*       SLOW()  Slowness array                                             *
*                                                                          *
*       TCALC() Calculated travel times                                    *
*                                                                          *
****************************************************************************
        PARAMETER        (M=20,N=10)
        LOGICAL          LCONST
        DIMENSION        CSUM(0:M,0:N),  ESUM(0:M,0:N)
        COMMON/CONSTR/   LCONST
        COMMON/VELOCI/   SLOW(0:M,0:N)
        COMMON/TIMES/    TIME(M,N)
        COMMON/CALCUL/   TCALC(0:M,0:N)
****************************************************************************
*                                                                          *
*       *****   VARIABLE USAGE  *****   VARIABLE USAGE  *****              *
*                                                                          *
*       Alpha   Angle of ray path                                          *
*       Angle i Angle of incidence for ray at K,0                          *
*       Beta    Angle of intersection                                      *
*       CSUM()  Sum of chord lengths around M,N parameter                  *
*       CLEN    Chord (arc) length intersecting parameter circle           *
*       DIST    Distance between parameter point and origin of             *
*               the radius of curvature for the raypath                    *
*       ERR     Residual error                                             *
*       ESUM()  Sum of errors associated with parameter M,N                *
*       I       Position of source                                         *
*       J       Position of reciever                                       *
*       K       X parameter coordinate                                     *
*       L       Z parameter coordinate                                     *
*       NSEG    # of ray segments                                          *
*       R       The radius of curvature for the ray                        *
*       RAYPAR  Ray parameter for the ray from iTH shot to jTH geophone*
*       RLEN    Length of ray path from ith source to jth receiver         *
```

```
*       SEG     True length of ray path                                      *
*       X,Z     Midpoint of segment for integration                          *
*       Xo,Zo   Origin of the radius of curvature                            *
*               always a negative value                                      *
*                                                                            *
*       *****   UNITS OF DIMENSIONS     *****                                *
*                                                                            *
*       Angles          in radians                                           *
*       Distance        in grid units (spacing between geophones)            *
*       Time            in seconds                                           *
*       Slowness        in seconds/grid unit                                 *
*       Velocity        in grid units/second                                 *
*                                                                            *
******************************************************************************

*****   Clear summations        *****

        DO 10   K=0,MROW
        DO 10   L=0,NCOL
                CSUM(K,L)=0.0
                ESUM(K,L)=0.0
10      CONTINUE

*****   For each ray path...    *****

        DO 30   I=1,MROW
        DO 30   J=1,NCOL

*****   For no data on ray      *****

                IF(TIME(I,J).LE.0.0)    GOTO 30
                ERR=TIME(I,J)-TCALC(I,J)

*****   Use geometry    *****

        RAYPAR=2.0*C1*I / SQRT((C0*C0+(C0+J*C1)**2+C1*C1*I*I) **2 - 4*C0
     *  *C0*(C0+C1*J)**2 )
                R=1.0/(RAYPAR*C1)
                alpha=2.0*ASIN(0.5*SQRT(FLOAT(I*I + J*J))/R)
                angle i=ATAN(I/FLOAT(J)) - alpha/2.0
                RLEN=R*alpha
                Xo=-R*COS(angle i + alpha)
                Zo=-R*SIN(angle i)

*****   For each parameter circle...    *****

        DO 20   K=0,I
                L1=NINT(Zo + SQRT((R-1./SQRT(2.))**2 - (K-Xo)**2) + .5)
```

```
          IF(L1.LT.0)     L1=0
          L2=INT(Zo + SQRT((R+1./SQRT(2.))**2 - (K-Xo)**2))
          IF(L2.GT.J)     L2=J
      DO 20   L=L1,L2

*     Find intersection of two equations
*     Equation one: Circle (X-K)**2 + (Z-L)**2 = 1/2
*     Equation two: Circle (X-Xo)**2 + (Z-Zo)**2 = R**2

*****  Ray intersects parameter circle ???    *****

          DIST=SQRT((K-Xo)**2+(L-Zo)**2)
          IF(ABS(DIST - R) .GE. 1./SQRT(2.0))    GOTO 20

*****  Locate intersection angle      *****

*****  Law of cosines  *****

*****  TEMP avoids computer roundoff errors    *****

          TEMP=(R*R + DIST*DIST - 0.5) / (2.0*R*DIST)
          IF(TEMP.GT.1.0) GOTO 20
          beta=2.0*ACOS(TEMP)

*****  B.C. - Ray path starts at (I,0) and ends at (0,J) *****

          IF(K.EQ.I .AND. L.EQ.0)        beta=beta/2
          IF(K.EQ.0 .AND. L.EQ.J)        beta=beta/2

*****  Chord (arc) length     *****

          CLEN=R*beta
          IF(CLEN.LT.CMIN)        GOTO 20

*****  Summation of chord (arc) lengths       *****

          CSUM(K,L)=CSUM(K,L) + CLEN

*****  Summation of error around parameter    *****

          ESUM(K,L)=ESUM(K,L) + ERR*CLEN/RLEN

20    CONTINUE
30    CONTINUE

*****  Update parameter matrix *****

      DO 40   K=0,MROW
```

```
      DO 40   L=0,NCOL
              IF(CSUM(K,L).EQ.0.0)    THEN
                      SLOW(K,L)=0.0
              ELSE
                      SLOW(K,L)=SLOW(K,L) + ESUM(K,L)/CSUM(K,L)
              END IF
40    CONTINUE

*****  Constrained parameters  *****

      IF(LCONST)      CALL PARCON(NCOL)

*****  For each ray path...    *****

      DO 70   I=1,M
      DO 70   J=1,N
      IF(TIME(I,J).LE.0.0)    GOTO 70
      TCALC(I,J)=0.0

      RAYPAR=2.0*C1*I / SQRT((C0*C0+(C0+J*C1)**2+C1*C1*I*I) **2 - 4*C0
    * *C0*(C0+C1*J)**2 )
      R=1.0 / (RAYPAR*C1)
      alpha=  2*ASIN(0.5*SQRT(FLOAT(I*I + J*J))/R)
      angle i=ATAN(I/FLOAT(J)) - alpha/2.0
      RLEN=   R*alpha
      NSEG=   INT(RLEN/DELTAR)
      SEG=    RLEN/NSEG
      Xo=-R*COS(angle i + alpha)
      Zo=-R*SIN(angle i)

*****  Interpolate slowness at midpoints of segment    *****

      DO 60   S=SEG/2.0,RLEN,SEG
              beta=S/R
              X=R*COS(ANGLE I + BETA) + XO
              Z=R*SIN(ANGLE I + BETA) + ZO
              TCALC(I,J)=TCALC(I,J) + SEG*CARINT(X,Z)
60    CONTINUE
70    CONTINUE

*****  Mean square error       *****

      ERROR=0.0
      DO 80   I=1,MROW
      DO 80   J=1,NCOL
              IF(TIME(I,J).LE.0.0)    GOTO 80
              ERR=TCALC(I,J)-TIME(I,J)
              ERROR=ERROR+ERR*ERR
```

```
80      CONTINUE
        RETURN
        END


        FUNCTION CARINT(X,Y)
*************************************************************************
*                                                                       *
*       CARINT: CARtesian INTerpolation                                 *
*               Interpolates the value of F(X,Y)                        *
*               Assumes function values are known at integer values of *
*               X and Y -- Which are stored in array U()                *
*                                                                       *
*       *****   CALLING VARIABLES       *****                           *
*                                                                       *
*       X,Y     Coordinates of point to interpolate                     *
*                                                                       *
*       *****   COMMON STORAGE  *****   INPUT   *****                   *
*                                                                       *
*       U()     Values of function at integer values for X,Y            *
*                                                                       *
*************************************************************************
        PARAMETER       (M=20,N=10)
        COMMON/VELOCI/  U(0:M,0:N)
        I=INT(X)µ       J=INT(Y)
        P=X-FLOAT(I)µ   Q=Y-FLOAT(J)
        F00=U(I,J)µ     F10=U(I+1,J)
        F01=U(I,J+1)µ   F11=U(I+1,J+1)

*****   Two point formula       *****

        IF(.NOT.(F11.EQ.0.0.AND.F10.EQ.0.0))    GOTO 10
        CARINT=(1-Q)*F00 + Q*F01
        RETURN

*****   Three point formula     *****

10      IF(F11.NE.0.0)  GOTO 20
        CARINT=(1.-P-Q)*F00 + P*F10 + Q*F01
        RETURN

*****   Three point formula     *****

20      IF(F00.NE.0.0)  GOTO 30
        CARINT=(1-P)*F01 + (1-Q)*F10 + (P+Q-1)*F11
        RETURN
```

```
*****   Four point formula      *****

30      CARINT=(1.-P)*(1.-Q)*F00+P*(1.-Q)*F10+Q*(1.-P)*F01+P*Q*F11
        RETURN
        END


        SUBROUTINE PARCON(NCOL)
        IMPLICIT        LOGICAL(L)
        PARAMETER       (M=20,N=10)
        COMMON/CONSTR/  LCONST,VELVAR,CONST(N)
        COMMON/VELOCI/  SLOW(0:M,0:N)
************************************************************************
*                                                                      *
*       PARCON: PARameter CONstraining routine                         *
*               Fixes the slowness parameter values down the borehole  *
*               as determined by the sonic log                         *
*                                                                      *
*       *****   CALLING VARIABLES       *****                          *
*                                                                      *
*       NCOL    # of geophones positions down the well                 *
*                                                                      *
*       *****   COMMON STORAGE  *****   INPUT   *****                  *
*                                                                      *
*       CONST() Constrained parameter values                           *
*                                                                      *
*       LCONST  Logical switch                                         *
*                                                                      *
*       VELVAR  Allowable velocity variation from the fixed value      *
*               Deviation in percentage from fixed value               *
*                                                                      *
*       *****   COMMON STORAGE  *****   I/O     *****                  *
*                                                                      *
*       SLOW()  Slowness matrix                                        *
*                                                                      *
************************************************************************
        DO 10   J=1,NCOL
                IF(CONST(J).LE.0.0)     GOTO 10
                VARY=CONST(J)*VELVAR/100.
                IF(SLOW(0,J).LT.CONST(J)-VARY)  SLOW(0,J)=CONST(J)-VARY
                IF(SLOW(0,J).GT.CONST(J)+VARY)  SLOW(0,J)=CONST(J)+VARY
10      CONTINUE
        RETURN
        END


        FUNCTION VEL    (DX,SLOW)
************************************************************************
```

```
*       VEL:    Computes velocity from the slowness                    *
************************************************************************
        IF(SLOW.LE.0.0) THEN
                VEL=0.0
        ELSE
                VEL=DX/SLOW
        END IF
        RETURN
        END

        SUBROUTINE ABBREV       (CMMND,NUMCOM,WORD,NUMLET)
        IMPLICIT        LOGICAL(L)
        CHARACTER       WORD*5, CMMND*65
************************************************************************
*                                                                      *
*       ABBREV: Expands abbrev. commands to their full length          *
*                                                                      *
*       *****   CALLING VARIABLES       *****                          *
*                                                                      *
*       CMMND   Legal command strings                                  *
*       NUMCOM  # of legal commands                                    *
*       WORD    Inputed command string                                 *
*       NUMLET  # of letters in inputed command                        *
*                                                                      *
*       *****   OUTPUT VARIABLES        *****                          *
*                                                                      *
*       WORD    The abbrev command expanded to its full length         *
*                                                                      *
************************************************************************
        CALL UCASE2     (.TRUE.,NUMLET,WORD)
        DO 20   I=NUMLET+1,NUMLET*NUMCOM+NUMLET,NUMLET
                LBEFOR=.FALSE.
                LAFTER=.FALSE.
        DO 10   J=1,NUMLET
                IF(WORD(J:J).EQ.' ')    GOTO 30
                K=I+J-1
                IF(WORD(J:J).NE.CMMND(K:K))     GOTO 20
                IF(WORD(J:J).NE.CMMND(K-NUMLET:K-NUMLET)) LBEFOR=.TRUE.
                IF(WORD(J:J).NE.CMMND(K+NUMLET:K+NUMLET)) LAFTER=.TRUE.
10      CONTINUE
        RETURN
20      CONTINUE
        RETURN
30      IF(LBEFOR.AND.LAFTER)   WORD(1:NUMLET)=CMMND(I:I+NUMLET-1)
        RETURN
        END
```

```
      SUBROUTINE UCASE2       (LCASE,NUMLET,WORD)
      IMPLICIT        LOGICAL(L)
      CHARACTER       WORD*5,UCALPH*26,LCALPH*26
***********************************************************************
*                                                                     *
*     UCASE2: Changes an alphanumeric string to a single case         *
*                                                                     *
*     *****   CALLING VARIABLES       *****                           *
*                                                                     *
*     LCASE   .TRUE.  Returns all upper case alphanumerics            *
*             .FALSE. Returns all lower case alphanumerics            *
*     NUMLET  # of letters in string                                  *
*     WORD    Alphanumeric string                                     *
*                                                                     *
*     *****   OUTPUT VARIABLES        *****                           *
*                                                                     *
*     WORD    Alphanumeric string converted to upper/lower case       *
*                                                                     *
***********************************************************************
      UCALPH='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
      LCALPH='abcdefghijklmnopqrstuvwxyz'
      DO 30   I=1,NUMLET
      DO 20   J=1,26
              IF(LCASE)       THEN
                      IF(WORD(I:I).NE.LCALPH(J:J))    GOTO 10
                      WORD(I:I)=UCALPH(J:J)
                      GOTO 30
              ELSE
                      IF(WORD(I:I).NE.UCALPH(J:J))    GOTO 10
                      WORD(I:I)=LCALPH(J:J)
                      GOTO 30
10            END IF
20    CONTINUE
30    CONTINUE
      RETURN
      END
```