T-2609

COMPARISON OF
FREQUENCY DOMAIN MIGRATION APPROACHES
WITH REGARD TO
VERTICAL VELOCITY VARIATIONS

BY

YO KATAHIRA

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

T-2609

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 in Geophysics.

Golden, Colorado

Date: Sept 2, 1952

Signed: Yo Katahira

Yo Katahira

Approved:

William A. Schneider
Thesis Advisor

Golden, Colorado

Date: Sept 2, 1952

George V. Keller
Head of Department

ABSTRACT

Migration of seismic data is an accepted part of seismic data processing flow today. It is a process which maps stacked seismic section into a section with events properly positioned in x, y, z space.

The incremental method discussed in this paper presents a recursive scheme in the frequency wavenumber domain as opposed to the variable velocity method (Stolt's development). The method projects the wavefield down successively by simple complex multiplications with downward continuation operators, consequently it handles vertical velocity gradients very well. The method is also free from frequency dispersion up to the nyquist frequency.

Synthetic model data are migrated in the presence of vertical velocity gradients and migration results are compared with results obtained by the variable velocity method. Comparisons between the two methods, given the accurate knowledge of velocity, show that the incremental method gives better results than Stolt's variable velocity method.

A simple coordinate transformation extends the applicability of the incremental method to accommodate minor lateral velocity gradients. This transformation maps an

original time section into a section governed by a mean velocity for an entire section. Then migration proceeds as before with the mean velocity. The result is encouraging but more analytical study is suggested in order to establish the scheme for laterally inhomogeneous media.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I wish to thank Drs. W. A. Schneider, F. A. Hadsell, and M. W. Majors for serving as my committee members. Special thanks go to my advisor, Dr. W. A. Schneider, for his dedicated guidance throughout the course of this study. The author is especially grateful to the late Dr. G. R. Pickett who served as a committee member until his death. Also, special thanks go to my fellow students for their encouragements, and to S. L. Faulkner for reviewing the manuscript.

Grateful acknowledgement is extended to the Integrated Geophysics Project for their financial support.

Finally, most sincere gratitude goes to my parents for their constant encouragements and financial and moral support throughout the years.

INTRODUCTION

Migration of seismic data has been important for some time in the field of exploration seismology.

Since the introduction of the finite - difference algorithm by Claerbout (1972), many studies have been published on the subject of wave equation migration.

Schneider (1978) formalized the Kirchhoff Integral technique. This method has historic ties to the conventional diffraction summation approach. He developed the explicit relationships existing between the Kirchhoff and the Frequency - Wavenumber approach discussed below.

Another approach to migration was popularized by Stolt (1978). He discussed the formulation of wave equation migration in the frequency domain (Variable velocity method) taking into account vertical velocity gradients. He showed that migration in the f - k domain is a straight mapping procedure. In his development of a migration scheme, Stolt made modifications to the original wave equation in order to incorporate a vertical velocity gradient. In his final form of the wave equation, he argued that W, a function of velocity, time, depth, etc., is supposed to take care of a vertical velocity gradient. Stolt proposed W be held constant along the entire seismic section (a value usually between 0.5 and 1.0).

Bolondi (1978) and Gazdag (1978) discussed a recursive type of scheme in the f - k domain (incremental method), in which the subsurface is discretized into layers with constant velocities and the wavefield is extrapolated downward layer by layer.

Recently a new approach to migration was introduced by Berkhout (1979, 1980). His method performs migration in the space - frequency domain (F - X migration). In his method, a seismic section is first Fourier transformed in the t - direction only and the data in the f - x domain is convolved with the downward continuation operator along the x - axis for each temporal frequency component.

This thesis work deals with the implementation and the evaluation of the incremental method on synthetic data. Also the two F - K migration schemes; namely, the variable velocity method and the incremental method, will be compared on the basis of their performances in the presence of vertical velocity gradients. Furthermore, the possible extension of the incremental method in the presence of lateral velocity gradient will be discussed.

All the computer work is done on CYBER - 720 at the Colorado School of Mines using CGG and AIMS software packages except for the incremental migration routine which was developed.

OVERVIEW OF WAVE EQUATION MIGRATION

This chapter briefly describes the migration principle and some of the major migration techniques available today; namely, Finite Difference, Kirchhoff Integral, and Frequency-Wavenumber.

The need for migration

All seismic processing today is based on the common assumption which states that the earth can be approximated by layers of rocks and that the layers are flat and horizontal. If this assumption is violated, as we often see in real geology, events are incorrectly positioned either vertically or horizontally, or even both.

Figure 1 illustrates the case where a dipping layer exists in a constant velocity medium. The source - receiver coincidence rays are drawn perpendicular to the reflector. This process is called Normal Incidence Ray Tracing and simulates a CDP stacked imaged section.

As we notice in Figure 1, seismic events (the dipping segment) are plotted directly beneath the source - receiver locations reached by the rays. This leads to an incorrect interpretation of data. A similar example of a diffractor is given in Figure 2. The diffraction curve is hyperbolic in nature, and occurs where any discontinuity exists.

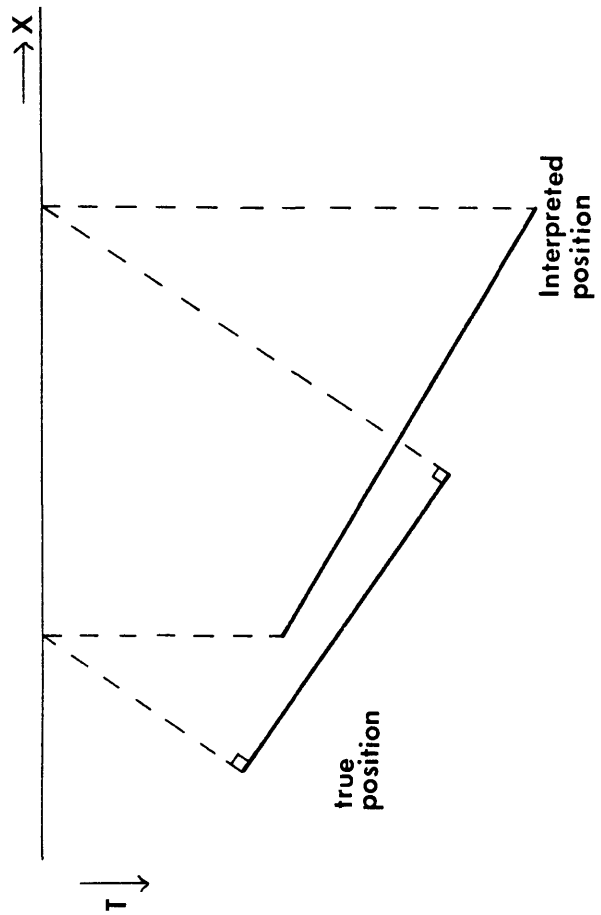

Figure 1: True and interpreted positions of a dipping segment.

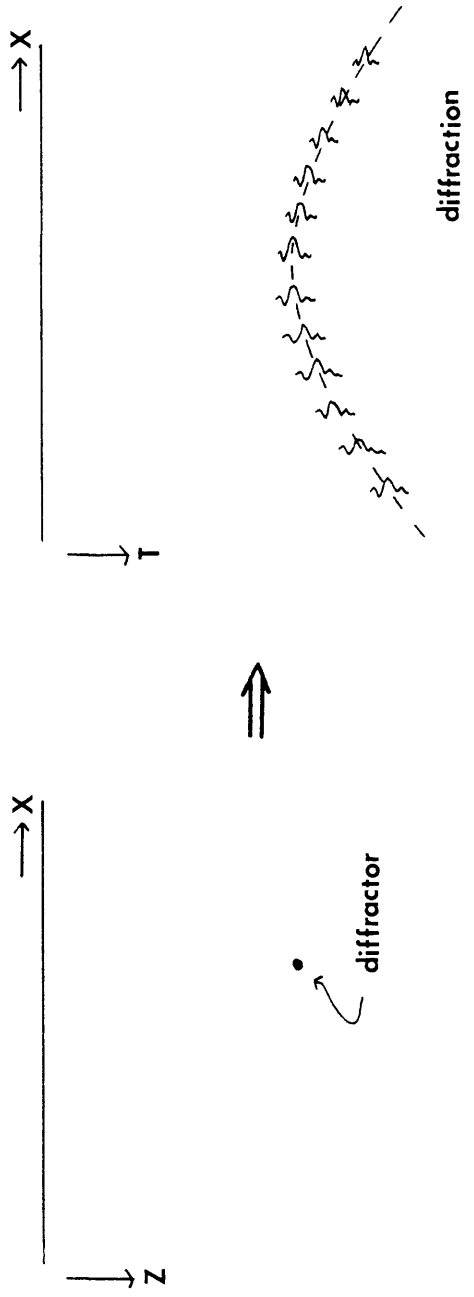

Figure 2: A diffraction from a scattering point.

Wave equation migration

Wave equation migration is the process which maps observed seismic data into either time or depth section at their correct vertical and horizontal position. It is based on numerical solution to the scalar wave equation,

$$\nabla^2 U - \frac{1}{c^2} U_{tt} = 0.$$

In implementation of two dimensional migration schemes today, there are three main assumptions:

1) data are primary reflections or diffractions,
2) cross - dip can be neglected,
3) velocity distribution is known

(Sheriff, 1978).

One of the main concerns is the problem of cross - dip. In two dimensional approach, we assume that there is no energy coming from out-of-plane structures. This is a poor assumption we have made. In real world, reflections come from all directions. Therefore, it should be emphasized that true migration can only be accomplished by processing in 3-D. It might be mentioned that the true three-dimensional migration is not obtained by separately migrating lines in two-dimensional form.

Wave equation migration is carried out in two steps (Figure 3);

1) Downward continuation.

Take the observed upgoing wave field at the surface and project it downward. This operation generates a series of new sections which estimate the recording of the wave field which would have been acquired at lower depth levels. Thus it gives us more detail and resolution.

2) Imaging.

The imaging principle states that the migrated section is composed of the downward continued recorded data for all depths, evaluated at $t = 0$.

There are a number of different migration methods available (Figure 4):

Finite - difference,
Kirchhoff integral,
Frequency - wavenumber,
Space - frequency.

All of these methods are based on the scalar wave equation. The differences in results are in the approximation made in their application rather than being intrinsic in

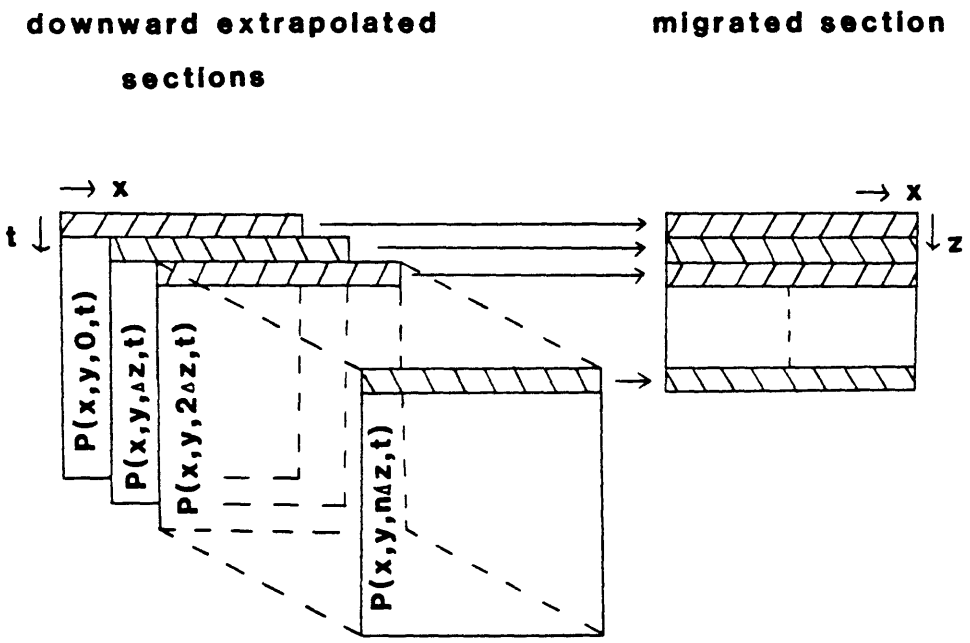

Figure 3: Migration of seismic section: Downward continuation and Imaging. (After Berkhout, 1980).

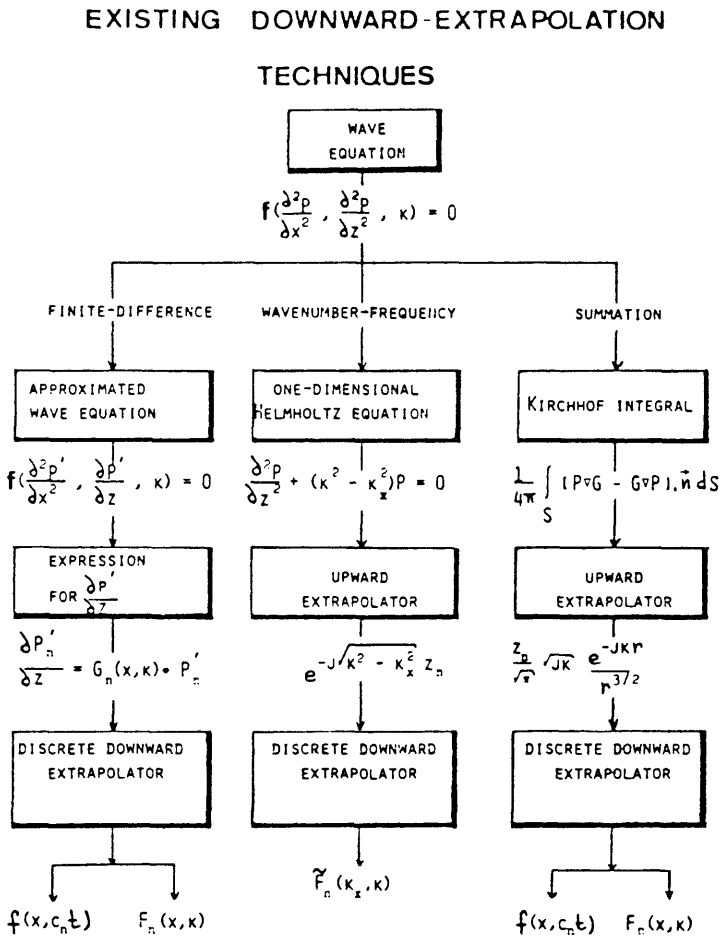

Figure 4: Some of existing downward continuation techniques: Finite difference, Kirchhoff integral, and Frequency wavenumber. (After Berkhout, 1979).

either approach (Sheriff, 1978). Table 1 summarizes the comparison of some of the techniques mentioned above.

The Finite difference method, introduced by Claerbout, transforms the scalar wave equation into difference equation. In his original development, performance was limited to about 15° dip. Refinement of his theory enables one to migrate up to 45° dip. The Finite difference algorithm handles a vertical velocity variation better than any other method. The major disadvantage is the dip limitation and large computer run time.

The Kirchhoff integral approach was introduced by French (1975) and Schneider (1978). This approach clearly shows historic ties to conventional diffraction summation approach, which simply summed trace amplitudes along diffraction curves. The main advantage of the method is good performance with steep dip, but it probably is the slowest of the current migration methods available.

Frequency wavenumber migration, popularized by Stolt (1978), has become a viable alternative to the conventional Finite difference or Kirchhoff integral approach. Unlike the other methods which migrate in the time domain, this method effects migration in the frequency wavenumber domain. With the aid of specially designed FFT hardware, computer run time is far less than any other methods. Stolt's development is a straight mapping process.

TABLE I

SUBJECTIVE SUMMARY

	F-D	"F-K"	KIRCHHOFF
SPACE	(t,x,z) (t,K_x,z)	(K_x,K_z)	(t,x,z)
COST	2	1	3
VELOCITY VARIATION	1	3	2
STEEP DIP	3	2	1
DISPERSION	3	1	2
TREATMENT OF NOISE	1	2	3
COSMETIC APPEARANCE	1	3	2
BEFORE STACK	3	2	1

Table I: Comparison of three different migration methods. "1" means best and "3" means worst. (After Johnson, 1978).

Bolondi (1978) and Gazdag (1978) discussed the idea of a recursive scheme in the f - k domain. Both approaches, the former a straight mapping and the latter a recursive scheme, theoretically migrate any dip up to 90°. Also they do not suffer from frequency dispersion within the nyquist band.

There still exists difficulties in migration. Such difficulties include noise, rapidly varying velocities, steep dip, and other problems which no current migration techniques handles perfectly (Chun, 1978).

F - K MIGRATION

In this chapter we discuss the theoretical development of migration of the frequency - wavenumber domain.

Basic theory for a constant velocity medium

Let the CDP stacked section be denoted by ψ (x, z=0, t) where x is a spatial midpoint coordinate and t is a travel time coordinate. We assume that ψ is governed by the scalar wave equation with a constant propagation velocity c,

$$\psi_{xx} + \psi_{zz} = \frac{1}{(c/2)^2} \psi_{tt}. \qquad (1)$$

Here subscripts x, z, and t denote differentiation with respect to the corresponding coordinate. The c/2 in equation (1) results from the fact that t equals the two way travel time. Now we employ the basic imaging principle; that is, migration is accomplished by transforming the surface wave field $\psi(x,0,t)$ into the wave field at a depth z, evaluated at t=0 $\psi(x,z,0)$. This is equivalent to the downward continuation of source - receivers from z=0 to a particular depth and evaluating at t=0. (Figure 5)

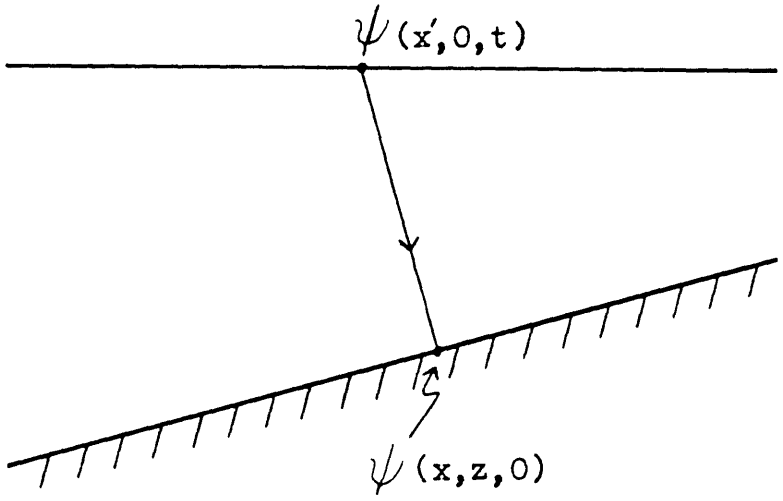

Figure 5: Simplified picture of downward continuation. x' is a surface midpoint coordinate and x is a migrated coordinate.

In order to carry out this scheme in the frequency-wavenumber domain, we have to take a two dimensional Fourier transform of equation (1) with respect to x and t. Let the two dimensional Fourier transform be defined as

$$U(k_x,f,z) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dt\ dx\ \ \psi(x,t,z)\ e^{-j2\pi(ft+xk_x)}. \qquad (2)$$

The Fourier transform of each term in equation (1) is given as follows by the differentiation property,

$$\psi_{xx} \longleftrightarrow (j2\pi k_x)^2 \, U, \tag{3-a}$$

$$\psi_{tt} \longleftrightarrow (j2\pi f)^2 \, U, \tag{3-b}$$

$$\psi_{zz} \longleftrightarrow U_{zz}. \tag{3-c}$$

Here "$\longleftrightarrow$" denotes Fourier transform pairs. Substituting equations (3-a), (3-b), and (3-c) into the original equation (1), we have

$$-4\pi^2 \, k_x^2 \, U + U_{zz} = - \frac{4}{c^2} \, 4\pi^2 \, f^2 \, U.$$

Rearranging the terms, finally we have in the transformed domain,

$$U_{zz} = -4\pi^2 \, (\frac{4f^2}{c^2} - k_x^2) \, U. \tag{4}$$

The solution to this second order differential equation is

$$U(k_x,f,z) = A \, e^{j2\pi z \, (\frac{4f^2}{c^2} - k_x^2)^{1/2}}. \tag{5}$$

The coefficient A can be determined from the boundary condition by setting z=0 and we have finally,

$$U(k_x,f,z) = U(k_x,f,0) \, e^{j2\pi z \, (\frac{4f^2}{c^2} - k_x^2)^{1/2}}. \tag{6}$$

The above equation relates the wave field at depth z to the wave field at z=0 (surface data). Qualitatively,

$U(k_x,f,z)$ .... Two dimensional Foureir transform of $\psi(x,z,t)$.

$U(k_x,f,0)$ .... Two dimensional Fourier transform of $\psi(x,0,t)$, which is the surface data.

and

$e^{j2\pi z\ (\frac{4f^2}{c^2} - k_x^2)^{1/2}}$ .... Operator to extrapolate the surface wave field downward by a distance z.

The next step we must take is to transform equation (6) back into the time domain. The two dimensional inverse Fourier transform of equation (6) is

$$\psi(x,z,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} df\ dk_x\ U(k_x,f,z)\ e^{j2\pi(ft+k_x x)}. \qquad (7)$$

Substituting equation (6) for U in equation (7), we have,

$$\psi(x,z,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} df\ dk_x\ U(k_x,f,0)\ e^{j2\pi z\ (\frac{4f^2}{c^2} - k_x^2)^{1/2}} \cdot e^{j2\pi\ (ft+k_x x)}. \qquad (8)$$

Now referring back to the imaging principle, what we want is $\psi(x,z,0)$ not $\psi(x,z,t)$, therefore we may set t=0 in equation (8), giving the migrated section,

$$\psi(x,z,0) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} df\ dk_x\ U(k_x,f,0)\ e^{j2\pi\left[z(\frac{4f^2}{c^2} - k_x^2)^{1/2} + k_x x\right]}. \quad (9)$$

The above equation, as it stands, is not very economical since for each depth z, we must perform a double integral over f and k_x. However by a simple change of variables, we can transform equation (9) into what looks like a two dimensional inverse Fourier transform.

Let

$$k_z = (\frac{4f^2}{c^2} - k_x^2)^{1/2} \quad (10)$$

then,

$$f = \frac{c}{2} (k_z^2 + k_x^2)^{1/2}, \quad (11)$$

$$df = \frac{c}{2} \frac{k_z}{(k_z^2 + k_x^2)^{1/2}} dk_z. \quad (12)$$

Substituting equations (11) and (12) into equation (9), we have

$$\psi(x,z,0) = \frac{c}{2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} dk_x \, dk_z \, Q(k_x,k_z) \, e^{j2\pi(k_z z + k_x x)} \qquad (13)$$

where

$$Q(k_x,k_z) = \frac{k_z}{(k_z^2+k_x^2)^{1/2}} \, U(k_x, \frac{c}{2}(k_z^2 + k_x^2)^{1/2}) \; . \qquad (14)$$

Finally, equations (11), (13), and (14) represent the solution to our problem.

The coordinate transformation equations (10) and (14) represent a shift of frequency from f to a lower frequency $f' = (f^2 - \frac{c^2 k_x^2}{4})^{1/2}$ for a constant k_x. This is similar to a moveout correction where f takes the place of time, and k_x of offset. Figure 6 shows the nature of this coordinate transformation. Points in the $f - k_x$ plane move vertically down to points in the $k_z - k_x$ plane. Taking an example in Figure 6, points at a constant frequency f_o map onto a circle in the $k_z - k_x$ domain such that $2f_o/c = (k_x^2 + k_z^2)^{1/2}$. In the constant velocity medium, no propagation energy exists below the line $f = c \cdot k_x$. All physically realizable energy must lie within the region bounded by $f = \pm c \, k_x$. After the migration, the wedge shaped region is transformed into a hemisphere governed by equation (11) as shown in Figure 7. Figure 8 summarizes the scheme in a simple block diagram for a constant velocity medium.

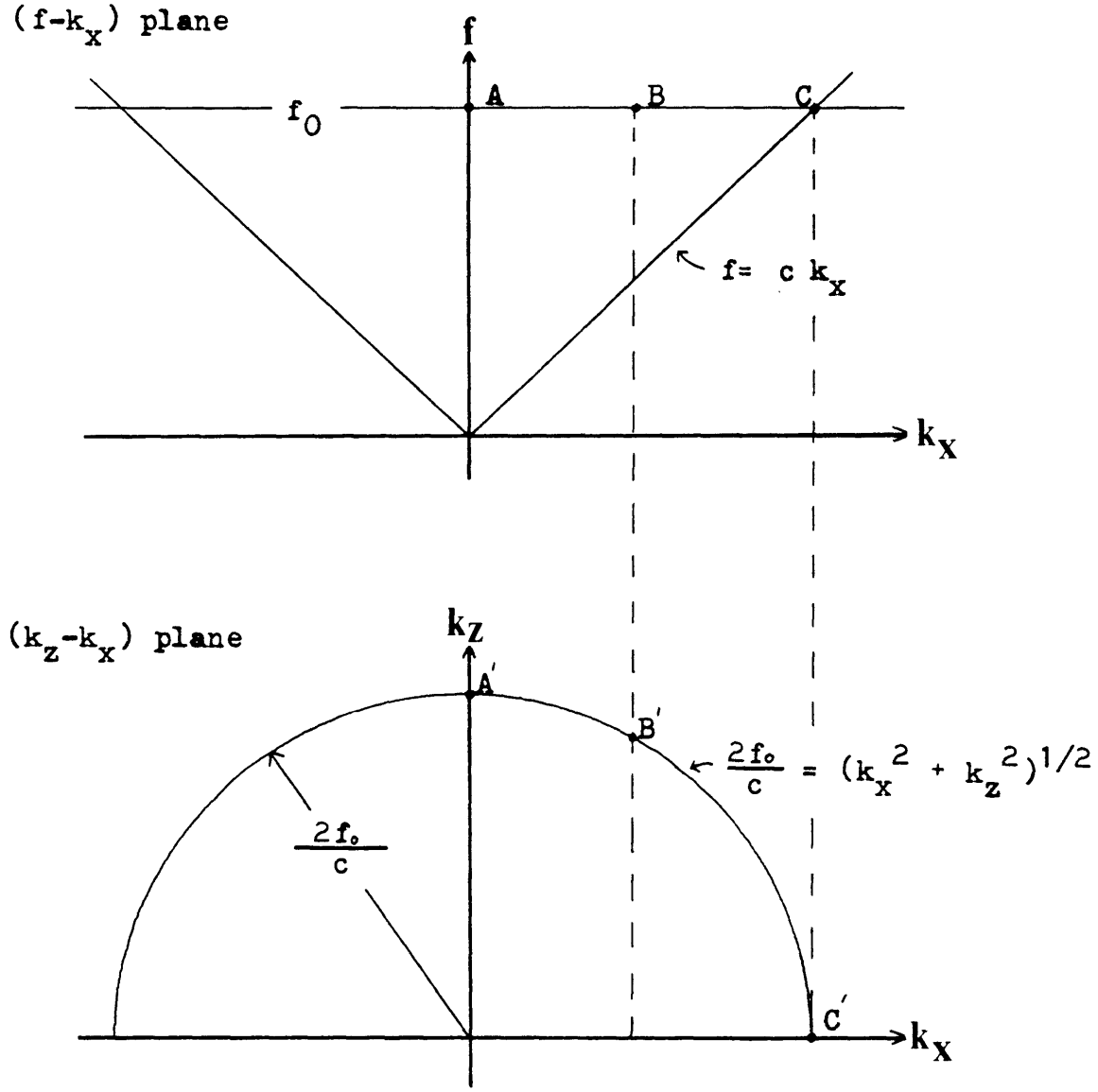

Figure 6: Migration mapping procedure for a constant velocity case.

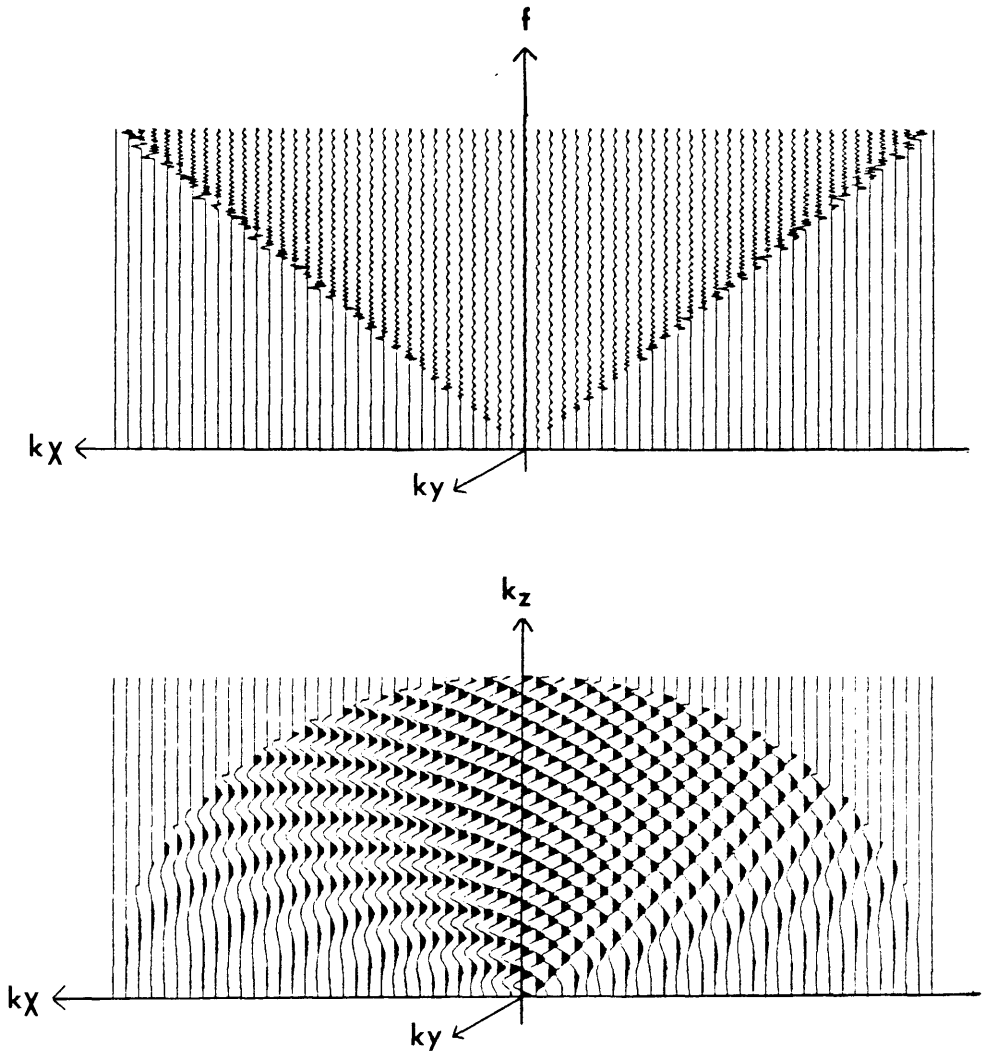

Figure 7: An example of migration mapping for a constant velocity case. f and k_x are not plotted on the same scale. This transformation from top to bottom is governed by equation(10). The wiggles shown in the plots represent moduli of Fourier spectrum.

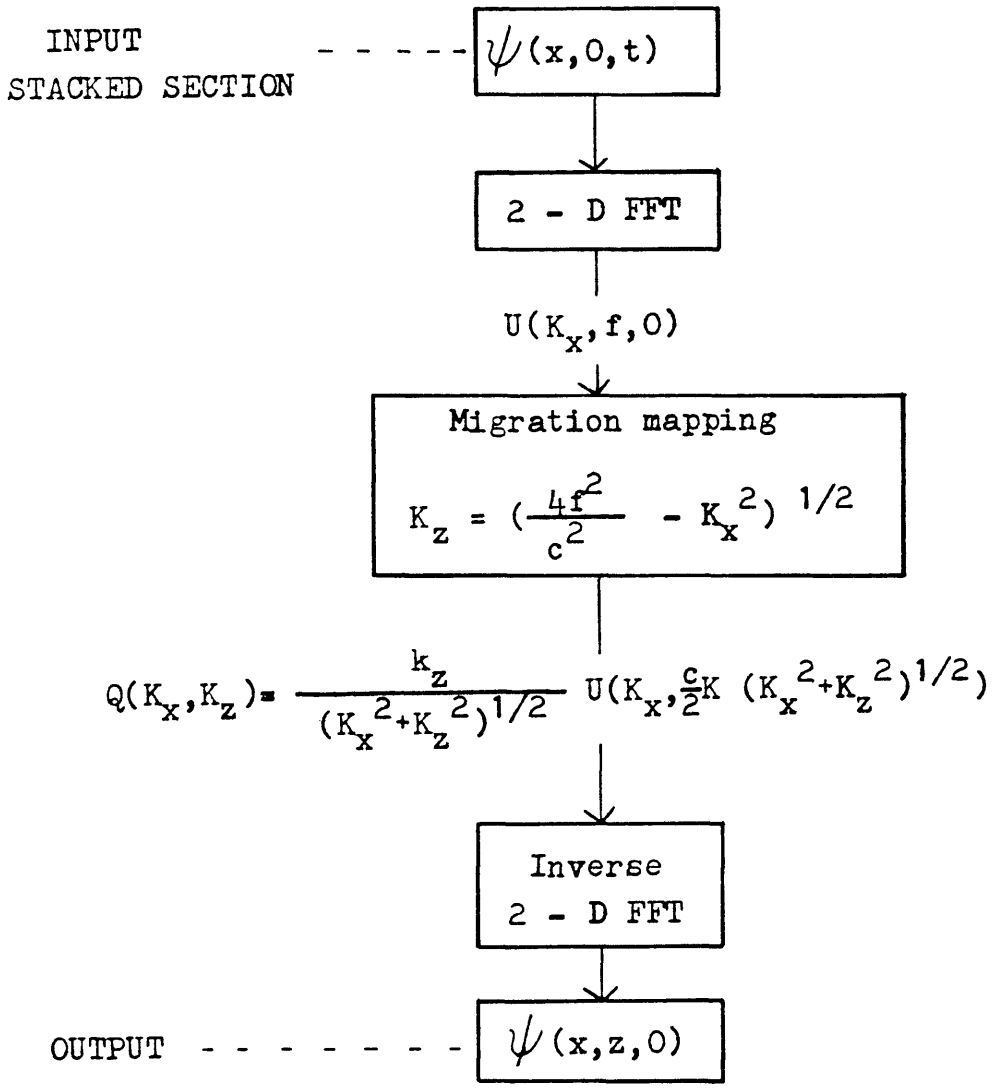

Figure 8: Simplified flow chart of constant velocity method.

Variable velocity method

The foregoing discussion assumes a constant propagation velocity c. Stolt (1978) made modifications to the constant velocity procedure in order to incorporate a vertical velocity gradient. After a series of coordinate transformations and approximations, Stolt obtained the new wave equation which is in an oblique coordinate system. Unlike the constant velocity case which is the exact solution to the scalar wave equation, this refined development of Stolt's approximates the wave equation. However, approximations he made are no worse than the ones made in other migration techniques such as Finite difference or Kirchhoff integral methods.

Starting with equations which govern $\psi(x_s,z_s,x_o,z_o,t)$,

$$\psi_{x_s x_s} + \psi_{z_s z_s} - \psi_{tt}/c(x_s,z_s)^2 = 0 \qquad (15)$$

and

$$\psi_{x_o x_o} + \psi_{z_o z_o} - \psi_{tt}/c(x_o,z_o)^2 = 0 \qquad (16)$$

where x_s,z_s is a source location and x_o,z_o is a receiver location. If we pretend that "stacked" sections are equivalent to normal incidence sections, that is, $(x_s,z_s) = (x_o,z_o)$. Defining midpoint coordinates,

$$X = (x_s + x_o)/2 \quad \text{and} \quad Z = (z_s + z_o)/2 \qquad (17)$$

and offset coordinates,

$$x = (x_o - x_s)/2 \text{ and } z = (z_o - z_s)/2 \tag{18}$$

Setting $\psi(X,x,Z,z,t) = \psi(x_s,z_s,x_o,z_o,t)$, and ignoring first derivatives with respect to x and z (see footnote on pp. 24, Stolt, 1978), equations (15) and (16) becomes

$$\psi_{XX} + \psi_{ZZ} = 4/C(X,Z)^2 \; \psi_{tt} = 0 \quad \text{(Stolt, 1978)} \tag{19}$$

Performing a series of coordinate transformations, neglecting velocity derivatives, and by setting $\phi(X,d,D) = \psi(X,Z,t)$ Stolt has come up with the equation

$$\phi_{XX} + W(X,d,D) \; \phi_{dd} + 2 \; \phi_{dD} = 0 \tag{20}$$

where

$$W = \frac{C(X,Z)^2}{\eta^2} + \frac{2d}{D} \left(1 - \frac{C(X,Z)^2 \; D^2}{\eta}\right), \tag{21}$$

$$D = \sqrt{2 \int_o^t \tau \cdot V_{rms}^2 \; (\tau) \; d\tau}, \tag{22}$$

(Stolt, 1978)

η and d are variables defined during Stolt's coordinate transformations.

Now migration proceeds from the half plane,

$$d = 0,\ D > 0,$$

to the half plane,

$$d = D,\ D > o.$$

Here D replaces time and is length dimensioned. We call this a "pseudo - depth" since only if the velocity is constant does D have a physical sense of depth. And d replaces Z which is a real depth variable of the original coordinate system. So all that these coordinate transformations amount to is that a stretching and a de-stretching of the input traces has to be carried out by using equation (22).

Figure 9 depicts an example of a diffraction stretching. The solid line indicates a real diffraction curve and the dashed line indicates a pseudo - depth converted diffraction curve.

Here we see that the pseudo - depth converted diffraction has been stretched down. This converted diffraction will not, therefore, be collapsed completely just by using the velocity of the upper medium, even though this velocity is the only parameter responsible for the diffraction. This is the reason we have to be very careful about the selection of W. We will discuss this point later in this section.

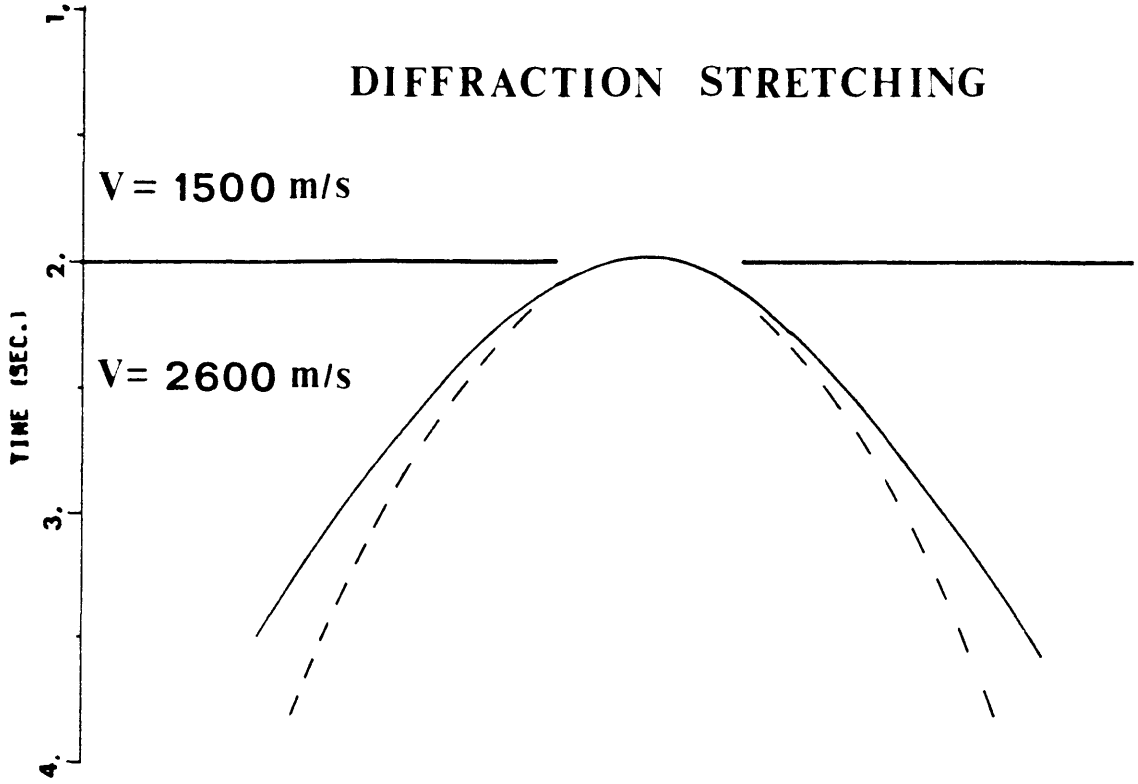

Figure 9: An example of stretching. The solid line indicates the real diffraction curve, the dashed line indicates the pseudo - depth converted diffraction curve.

Now let us proceed to obtain mapping equations from equation (20), we have

$$\phi_{XX} + W\phi_{dd} + 2\,\phi_{dD} = 0 \,. \qquad \text{(Stolt, 1978)} \tag{20}$$

By taking a two dimensional Fourier transform with respect to X and D, we have in the transformed domain

$$W\,U_{dd} + 4\pi j\,k_D U_z = 4\pi^2 k_x^{\,2}\,U \,. \tag{23}$$

The solution to this differential equation is

$$U(k_x,k_D,d) = U(k_x,k_D,O)\;e^{j2\pi d\;\left[-\frac{k_D}{W}+\sqrt{\frac{k_D^{\,2}}{W^2}-\frac{k_x^{\,2}}{W}}\right]}. \tag{24}$$

By taking the inverse two dimensional Fourier transform of equation (24), we have

$$\phi(X,d,D) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dk_x\,dk_D\,U(k_x,k_D,O)\;e^{j2\pi\left\{Xk_x+Dk_D+d\left[-\frac{k_D}{W}+\sqrt{\frac{k_D^{\,2}}{W^2}-\frac{k_x^{\,2}}{W}}\right]\right\}}. \tag{25}$$

Now we apply the imaging principle; that is, the migrated section occurs when d=D, then equation (25) becomes

$$\phi(X,D,D) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dk_x\,dk_D\,U(k_x,k_D,\,O)\;e^{j2\pi\left\{Xk_x+D\left[\frac{k_D(W-1)}{W}+\sqrt{\frac{k_D^{\,2}}{W^2}-\frac{k_x^{\,2}}{W}}\right]\right\}}. \tag{26}$$

As it has been shown in the constant velocity case, a change of variables will help us here too.

Let

$$k_z = \frac{k_D\ (W-1)}{W} + \sqrt{\frac{k_D^2}{W^2} - \frac{k_x^2}{W}}\ . \tag{27}$$

After some manipulations, we have

$$k_D = \frac{(1-W)\ k_z + \sqrt{k_z^2 + (2-W)\ k_x^2}}{(2-W)} \tag{28}$$

and

$$dk_D = \frac{k_z + (1-W)\sqrt{k_z^2 + (2-W)\ k_x^2}}{(2-W)\sqrt{k_z^2 + (2-W)\ k_x^2}}\ dk_z. \tag{29}$$

Therefore we have for the migrated section

$$\phi(X,D,D) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dk_x\ dk_z\ Q(k_x,k_z)\ e^{j2\pi(Xk_x + Dk_z)}, \tag{30}$$

where

$$Q(k_x,k_z) = \frac{k_z + (1-W)\sqrt{k_z^2 + (2-W)\ k_x^2}}{(2-W)\sqrt{k_z^2 + (2-W)\ k_x^2}}$$

$$\cdot\ U\left(k_x,\ \frac{(1-W)k_z + \sqrt{k_z^2 + (2-W)\ k_x^2}}{(2-W)}\right). \tag{31}$$

Note in equation (31), in the case of W = 1,

$$Q(k_x,k_z) = \frac{k_z}{\sqrt{k_z^2 + k_x^2}} \; U(k_x, \sqrt{k_z^2 + k_x^2})$$

which is equivalent to the constant velocity case.
From this point the migration proceeds as follows

1) time to pseudo - depth conversion with equation (22)
2) two dimensional Fourier transform
3) migration mapping with equation (31)
4) two dimensional inverse Fourier transform
5) pseudo - depth to time conversion with equation (22).

Now let us turn our attention back to the discussion of W, the shaping parameter. Equation (20) shows that all explicit dependence on X and Z now resides in the coefficient W of ϕ_{dd}. $W \neq 1$ reflects the fact that diffractions are not pure hyperbolas in a layered medium (Stolt, 1978). Stolt also argued that W, a function of all variables, is supposed to take care of a vertical velocity gradient. Stolt proposed W be held constant along the entire seismic section (a value usually between 0.5 and 1.0). Looking back in Figure 9, for greater dips the two curves are significantly diverging from each other.

This means that standard F - K migration of such dips will result in mislocated events. This is a threat to one of the more attractive features of the variable velocity method, which is the ability to migrate any dip up to 90°.

The detailed study done by J. P. Diet (1979) indicated that in the medium where velocity is a linear function of depth, W can be calculated using the following equation

$$W = \frac{e^{-2\alpha t} + 2\alpha t - 1}{e^{2\alpha t} + e^{-2\alpha t} - 2}, \quad \text{(Diet, 1979)} \tag{32}$$

where α is the vertical velocity gradient, and t is the one way travel time. Figure 10 displays W as a function of α and t. The idea he used is to compare the theoretical diffraction curves in a pseudo - depth domain with the real ones we want to collapse and used the parameter W to make them fit each other as well as possible. From theory Diet obtained the following equations in a pseudo - depth domain, for theoretical diffraction curves as a function of W,

$$D(x) = D_O \frac{W - 1}{W} + \frac{D_O}{W} \sqrt{1 + x^2 \frac{W}{D_O^2}}, \quad \text{(Diet, 1979)} \tag{33}$$

and for observed diffraction curves as a function of velocity ration (V_{lower}/V_{upper}) only for a small offset,

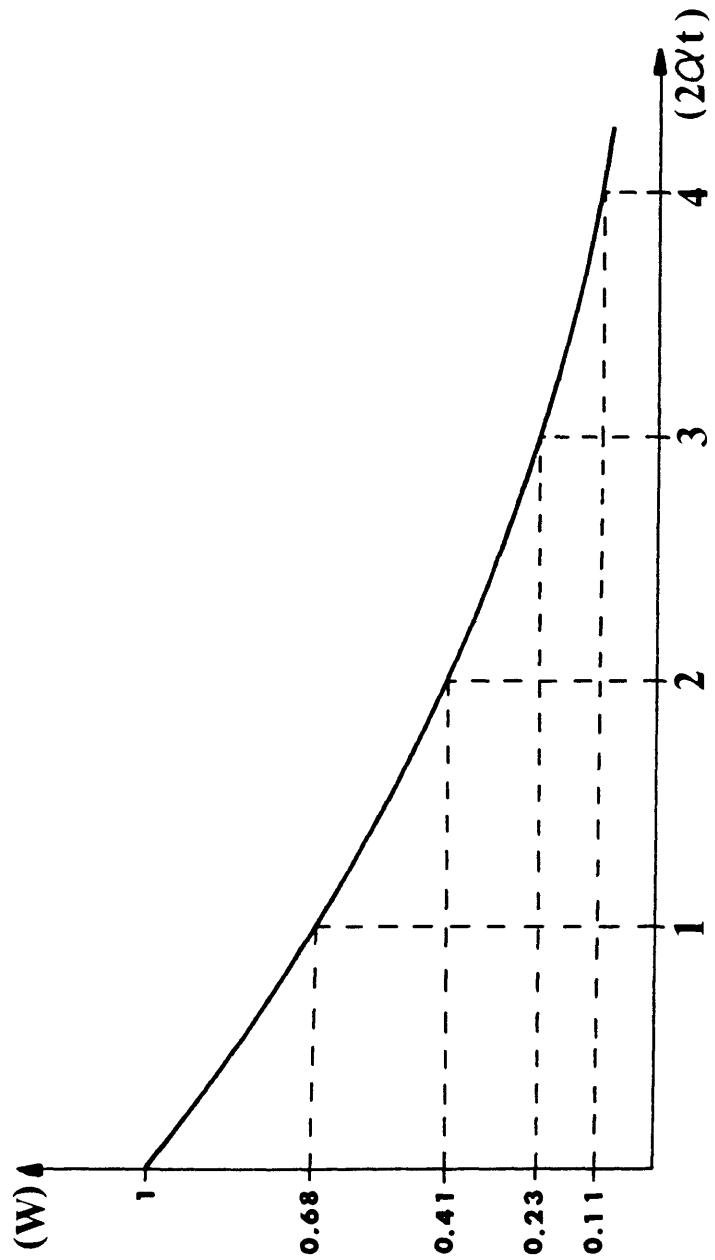

Figure 10: W plotted as a function of α and t. α is the velocity gradient and t is a one way travel time. (Replotted after Diet, 1979).

$$D(x) \approx D_O + \frac{x^2}{2D_O} - \frac{x^4}{8D_O^3}\left(2 - \frac{V_u^2}{V_1^2}\right), \quad \text{(Diet, 1979)} \tag{34}$$

where in both equations, x is the offset distance and D_O is the pseudo - depth of the diffracting point. An example of a simple two layer model is shown in Figure 11 and 12. Figure 11 shows theoretical diffraction curves in a pseudo - depth domain calculated as a function of W, and Figure 12 shows diffraction curves calculated as a function of velocity ratio (V_{lower}/V_{upper}). D_O in both figures is the pseudo - depth of the diffracting point. Notice that the curve for W = 1 in Figure 11 and the one for velocity ratio = 1 are the same because they both reflect the constant velocity case.

For a given seismic section, the value of W may be chosen in order to focus the region of interest even though the value may not be the best choice for other parts of the section.

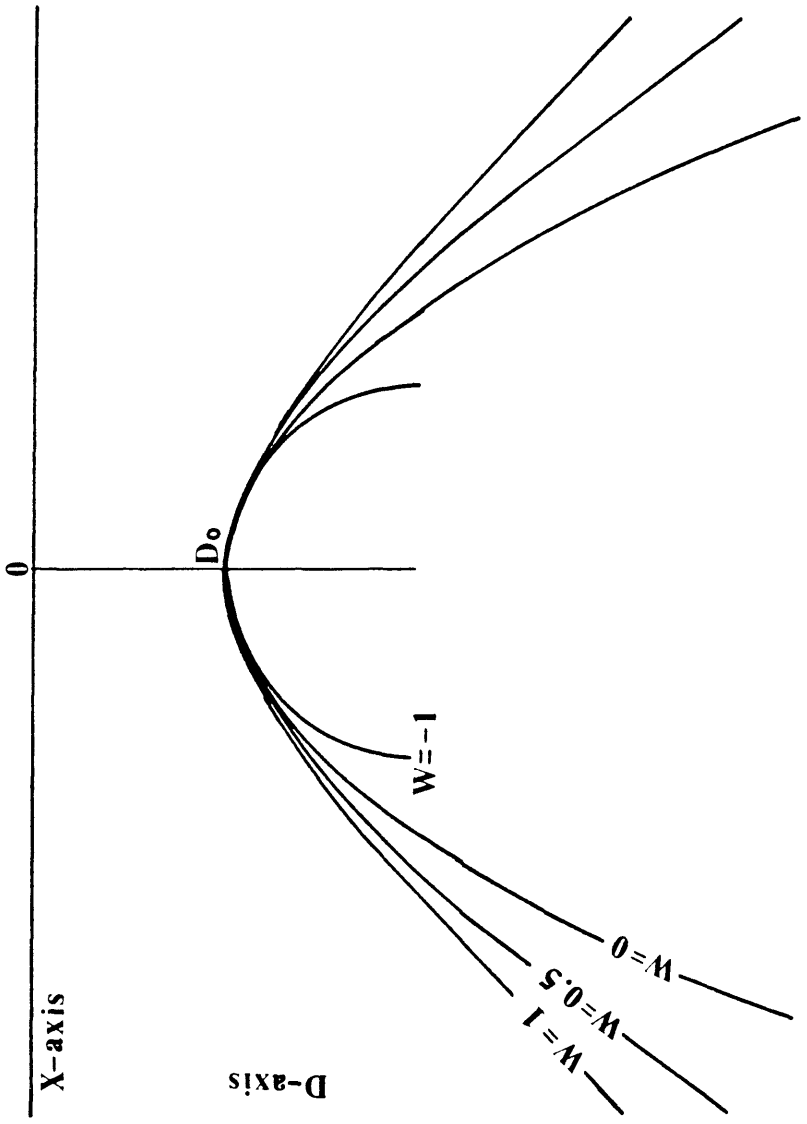

Figure 11: Pseudo - depth converted theoretical diffraction curves for different W. (Replotted after Diet,1979).

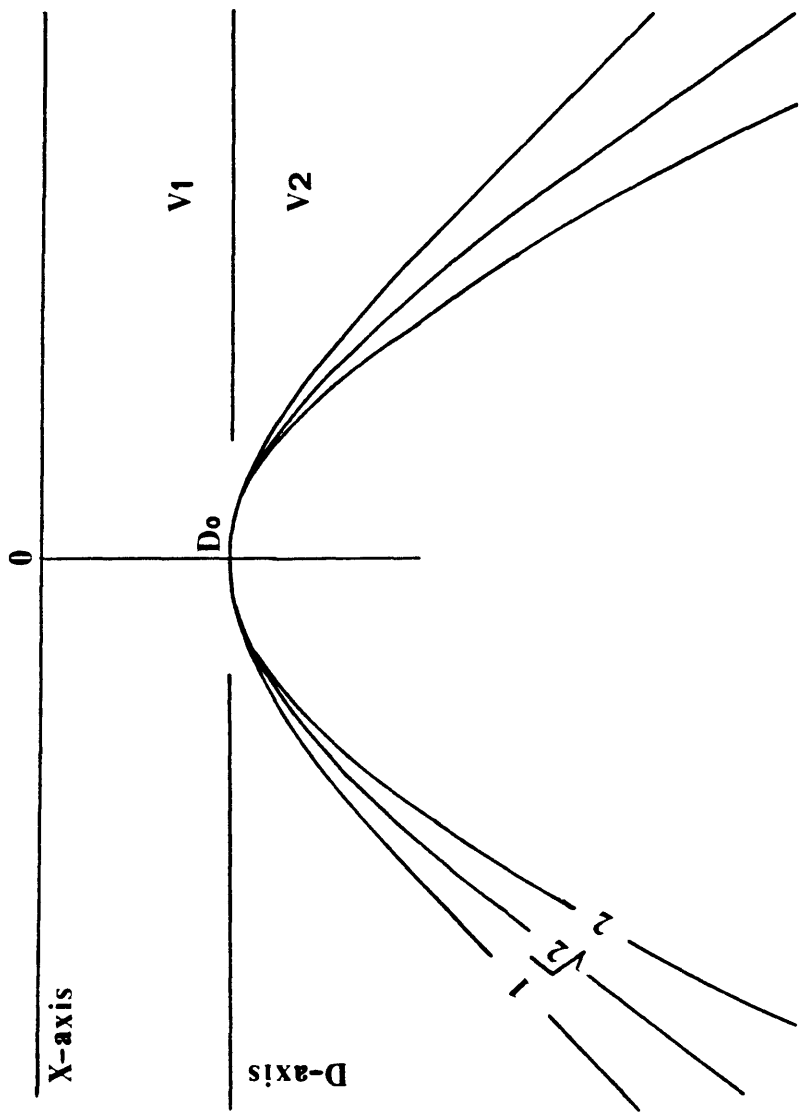

Figure 12: Pseudo - depth converted real diffraction curves for different velocity ratio (V_2 / V_1). (Replotted after Diet, 1979).

Incremental method

The incremental method also performs migration in the frequency wavenumber domain. This is a recursive type of scheme, in which the subsurface is discretized into layers with constant velocities and the wave field is extrapolated layer by layer (Figure 13).

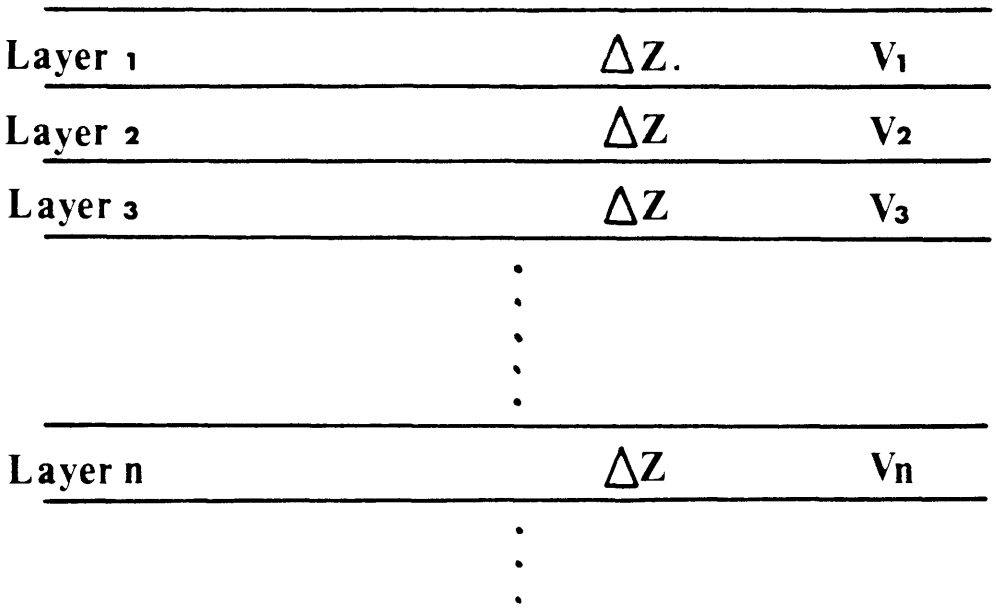

Figure 13: Diagram of the incremental model.

Naturally, one might then suspect that computation time is increased. However a big advantage of the incremental method over the variable velocity method is that the wave extrapolation is exact, since the subsurface is subdivided into constant velocity layers. Consequently, it gives us more accurate imaging of the section in the presence of a vertical velocity gradient. If one desires, the subdivision can be broken down to each sampling interval.

Let $U(k_x,f,O)$ be a two dimensional Fourier transform of the stacked section. Then we know from equation (6) in the basic theory section, the downward continued wave field across layer 1 is given by

$$U(k_x,f,\Delta z)= U(k_x,f,O)\ e^{j2\pi\Delta z\ (\frac{4f^2}{c_1^2} - k_x^2)^{1/2}}. \tag{35}$$

The extrapolation across layer 2 is given by

$$\begin{aligned} U(k_x,f,2\Delta z) &= U(k_x,f,\Delta z)\ e^{j2\pi\Delta z\ (\frac{4f^2}{c_2^2} - k_x^2)^{1/2}} \\ &= U(k_x,f,O)\ e^{j2\pi\Delta z \sum_{i=1}^{2} (\frac{4f^2}{c_i^2} - k_x^2)^{1/2}} . \end{aligned} \tag{36}$$

By successive extrapolations of the wave field we have

$$U(k_x,f,z_n) = U(k_x,f,O)\ e^{j2\pi\Delta z \sum_{i=1}^{N} (\frac{4f^2}{c_i^2} - k_x^2)^{1/2}}. \tag{37}$$

Now we have to take the two dimensional inverse Fourier transform; that is,

$$\phi(x,z_n,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} U(k_x,f,z_n)\ e^{j2\pi(xk_x + tf)}\ dk_x\ df\ , \qquad (38)$$

Referring back to the imaging principle, what we really want is the wave field at t=0 at some particular depth z_n. Then by setting t=0 in equation (38), we have

$$\phi(x,z_n,O) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} U(k_x,f,z_n)\ e^{j2\pi\ xk_x}\ df\ dk_x\ , \qquad (39)$$

Equation (39) indicates that we have to integrate over frequency then take the inverse Fourier transform along the spatial axis (Figure 14). Then the migration proceeds as follows:

1) two dimensional Fourier transform of the stacked section.

2) at each desired depth
   - multiply the wave field by the extrapolation operator
   - sum frequency at common k_x's
   - inverse one dimensional Fourier transform along the k_x axis.

Figure 15 shows a simplified flow diagram of the process.

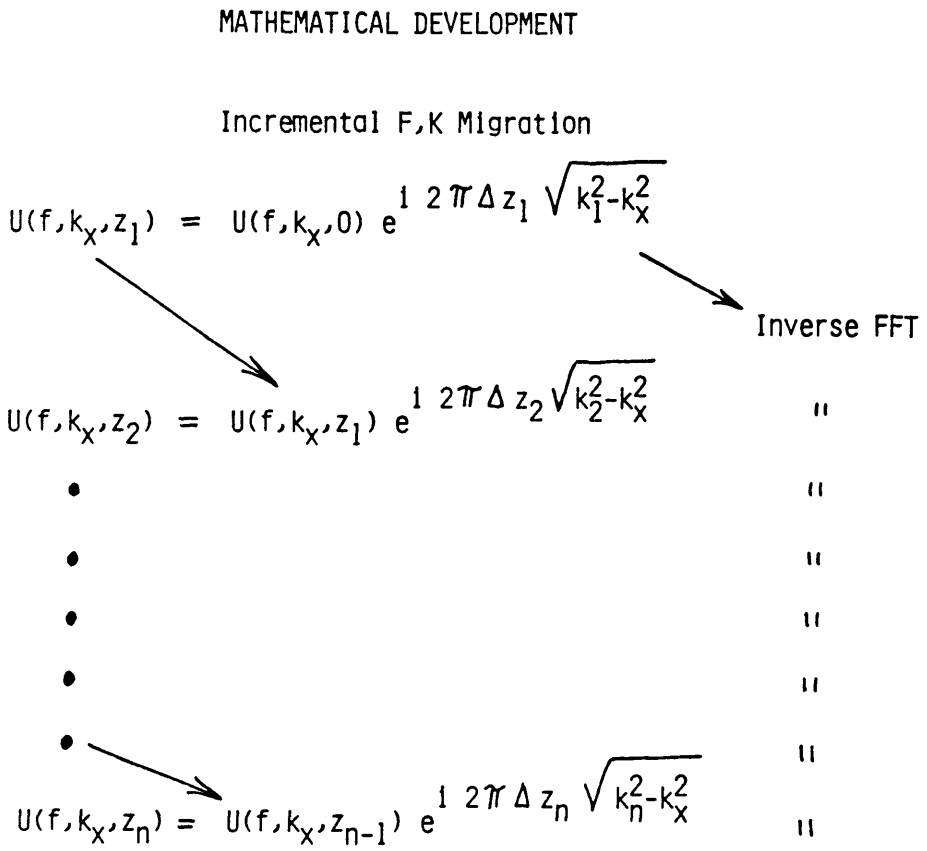

Figure 14: Simplified diagram of the incremental process.

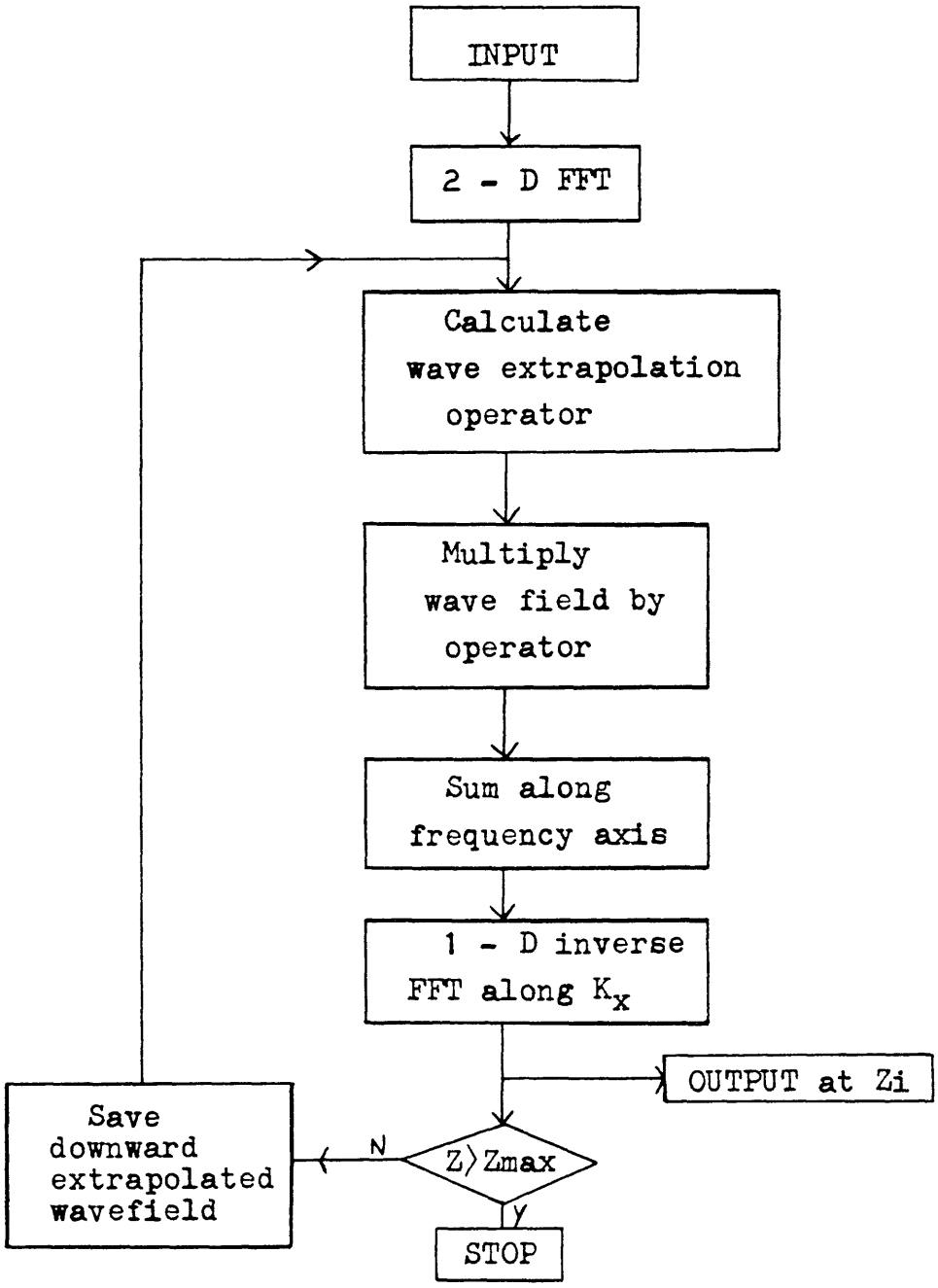

Figure 15: Simplified flow diagram of the incremental method.

COMPARISON OF FREQUENCY DOMAIN MIGRATION METHODS

In order to carry out the comparison of the two migration schemes, several examples are presented here. For forward synthetic modelling, the CGG wave equation modelling package was used. The sampling interval, the CDP spacing, and the wavelet used for synthetic time sections are 1 msec., 50 feet, and a bandpass wavelet with corner frequencies at (10,30,80,100) hz, respectively. Synthetic time sections were migrated by FKMIG (CGG) for the variable velocity method and YOMIG (written by the author) for the incremental method in order to establish a basis for comparison of the two methods.

One horizon test model (Model 1)

Model 1 was intended primarily for the testing of the incremental method and for the reconfirmation of Diet's study (1979). Figures 16, 17, and 18 show the physical parameters, normal incidence time section and wave - theory modelling of model 1, respectively. There is no geological significance to this model, densities are varied to give diffractions. Figure 19 illustrates a pseudo - depth converted real diffraction and a pseudo - depth converted theoretical diffraction curve with W = 1.0. Since the velocity ratio, 1.2, is very small, two diffraction curves in the pseudo - depth domain are very close, in fact indistinguishable in this figure. Consequently, we expect migration to be very close to perfect. And indeed, the migrated result of this synthetic model is very good. Figures 20 and 21 show migrations of model 1 by the variable velocity method, and by the incremental method, respectively. Here we observed that the variable velocity method gives us acceptable results when the velocity ratio is close to 1.0.

Next, the velocity ratio is increased to 2.0. Synthetic time section of this model is the same as the one illustrated in Figure 18. The velocity of the upper medium is the only factor responsible for the diffraction

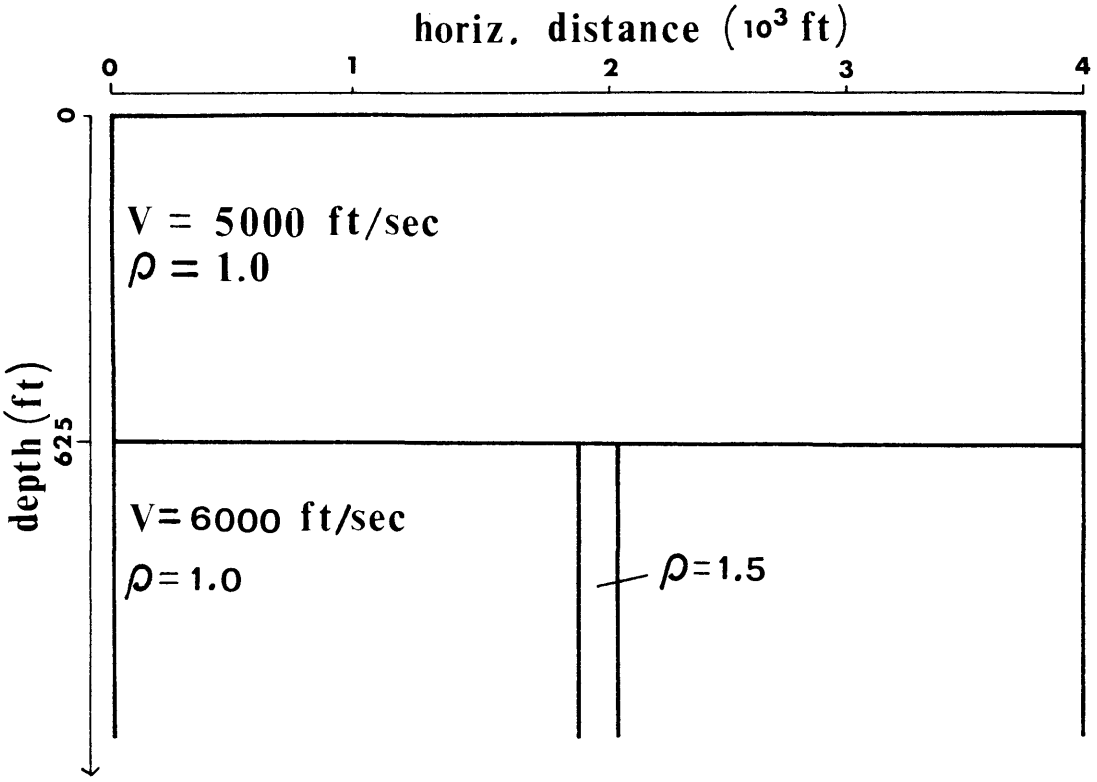

Figure 16: Physical parameters of model 1.

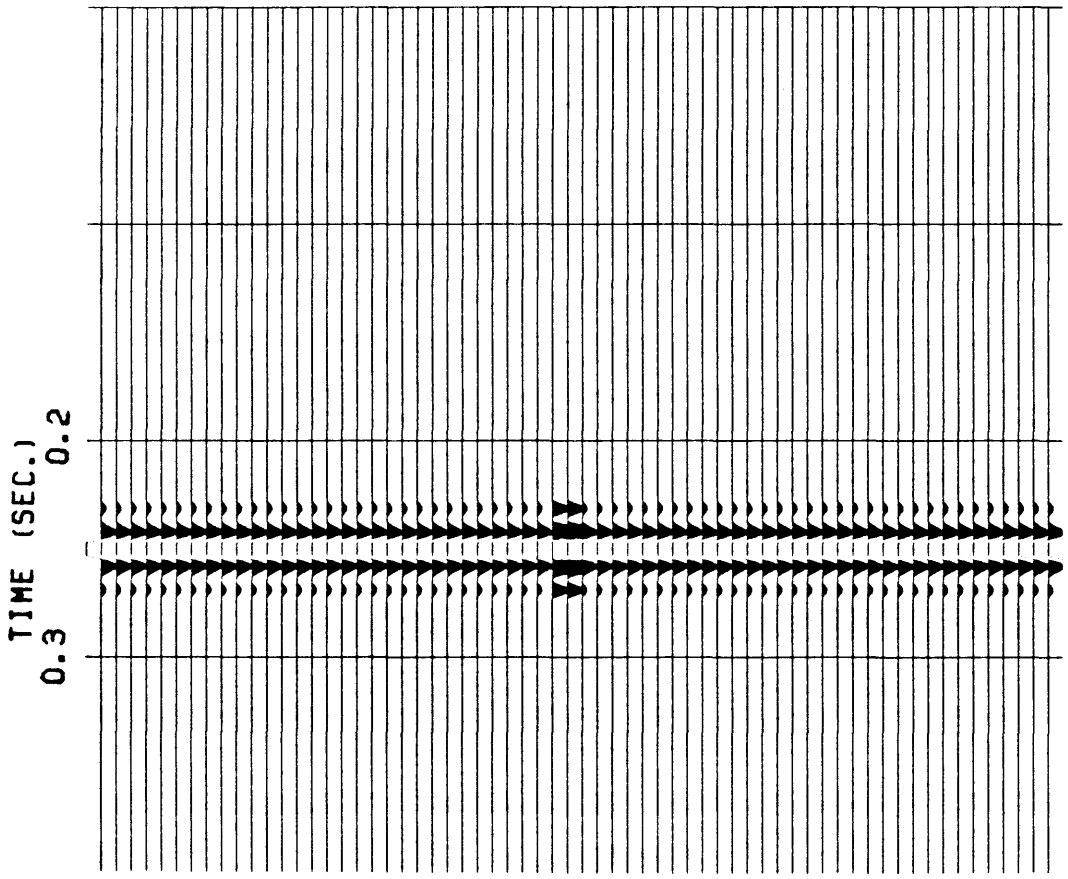

Figure 17: Normal incidence time section of model 1.

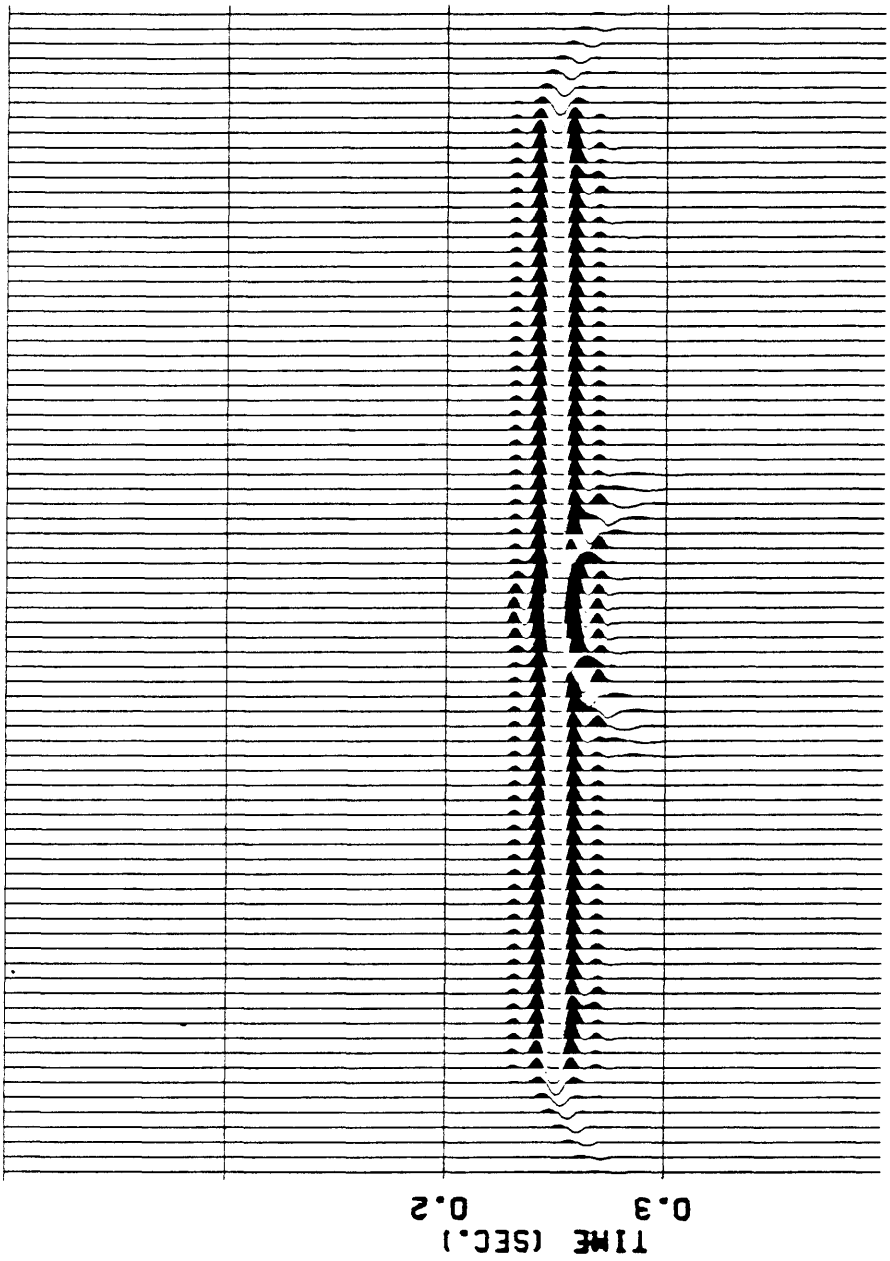

Figure 18: Wave - theory modelling of model 1.

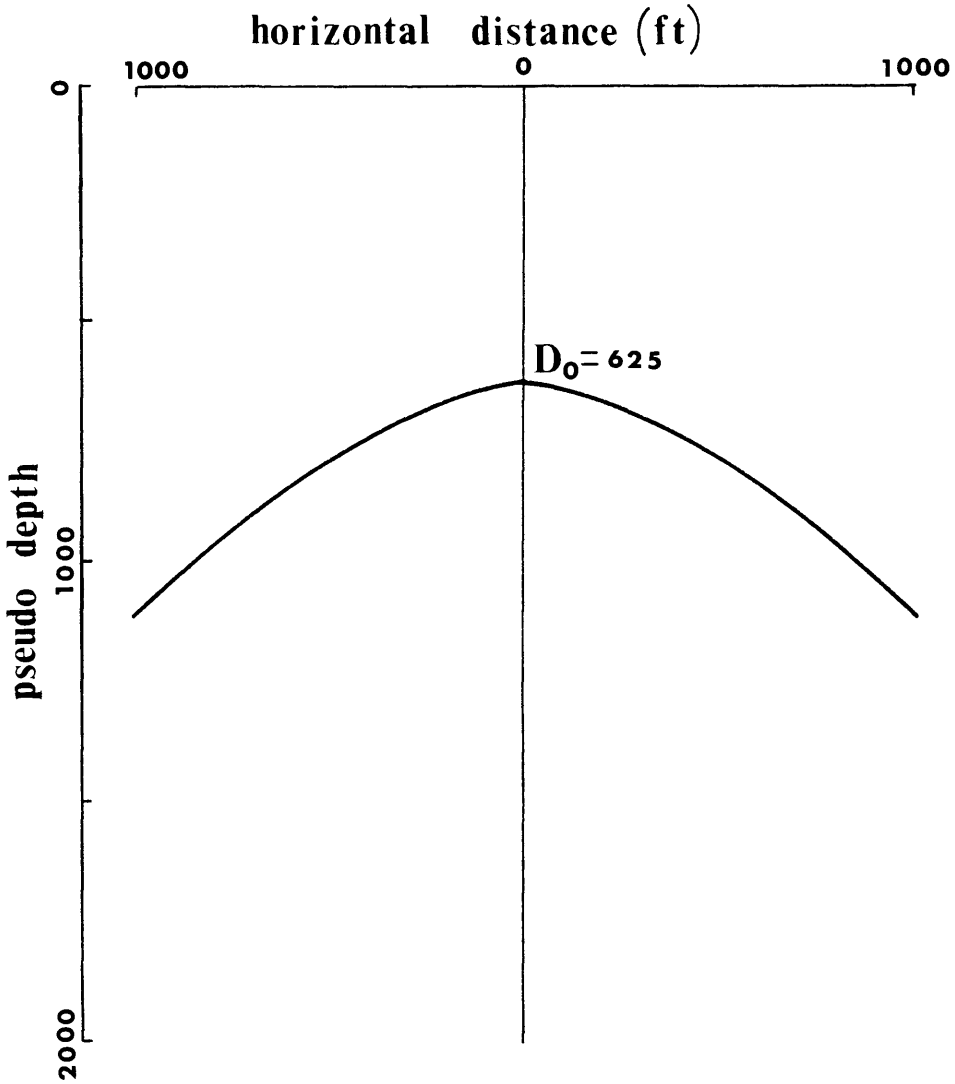

Figure 19: Real and theoretical diffraction curves in the pseudo - depth domain of model 1 for velocity ratio of 1.2 .
D_0 is pseudo - depth of the diffracting point.

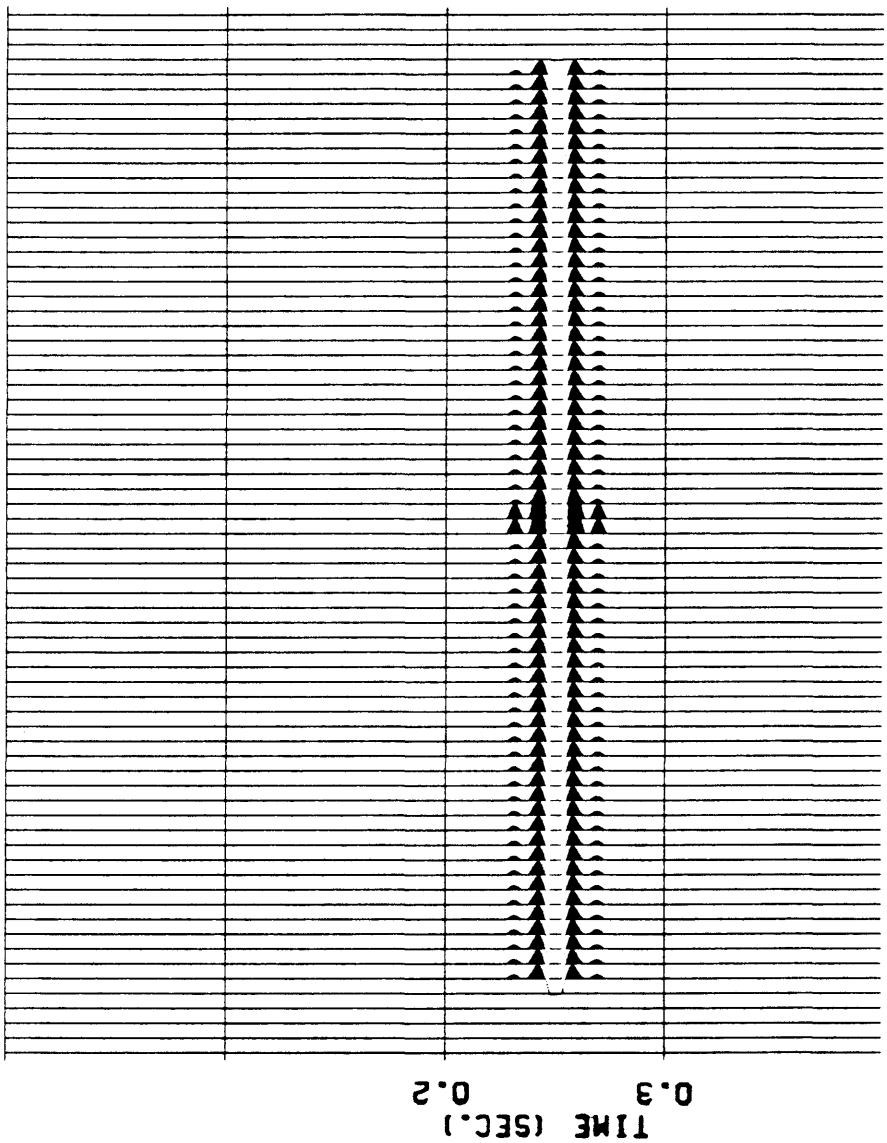

Figure 20: The variable velocity migration of model 1 with W = 1.0 for velocity ratio of 1.2 .

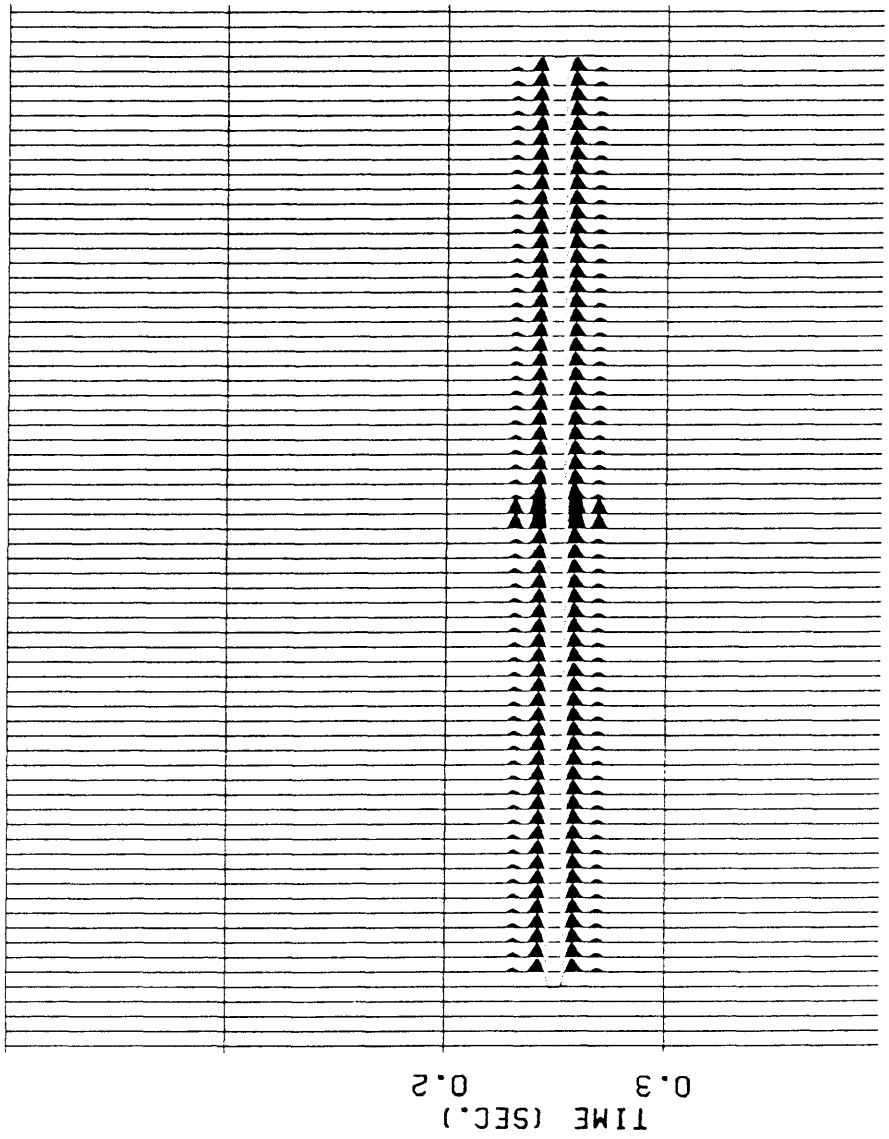

Figure 21: The incremental migration of model 1 for velocity ratio of 1.2 .

except the reflection coefficient. Figure 22 shows the theoretical and the real diffraction curves in the pseudo-depth domain. Here, the solid line indicates a pseudo - depth converted real diffraction curve and the dashed line indicates a pseudo - depth converted theoretical diffraction curve with W = 1.0. The difference between two curves is more than the one with the velocity ratio of 1.2. We expect that the migration with W = 1.0 would result in overmigration since the real diffraction curve is more open than the theoretical curve with W = 1.0. And the result shows a small amount of overmigration as is seen in Figure 23. The next step which we have to take is to change the value of W so that the two diffraction curves in the pseudo-depth domain fit closer. Looking back in Figure 12 of the variable velocity section, the value of W comes out to be about 0.0. Figure 24 shows the result of this migration with W = 0.0. The diffraction has been collapsed very well. The incremental method, as we expect, results in perfect migration (Figure 25), since this method does not suffer from stretching of data.

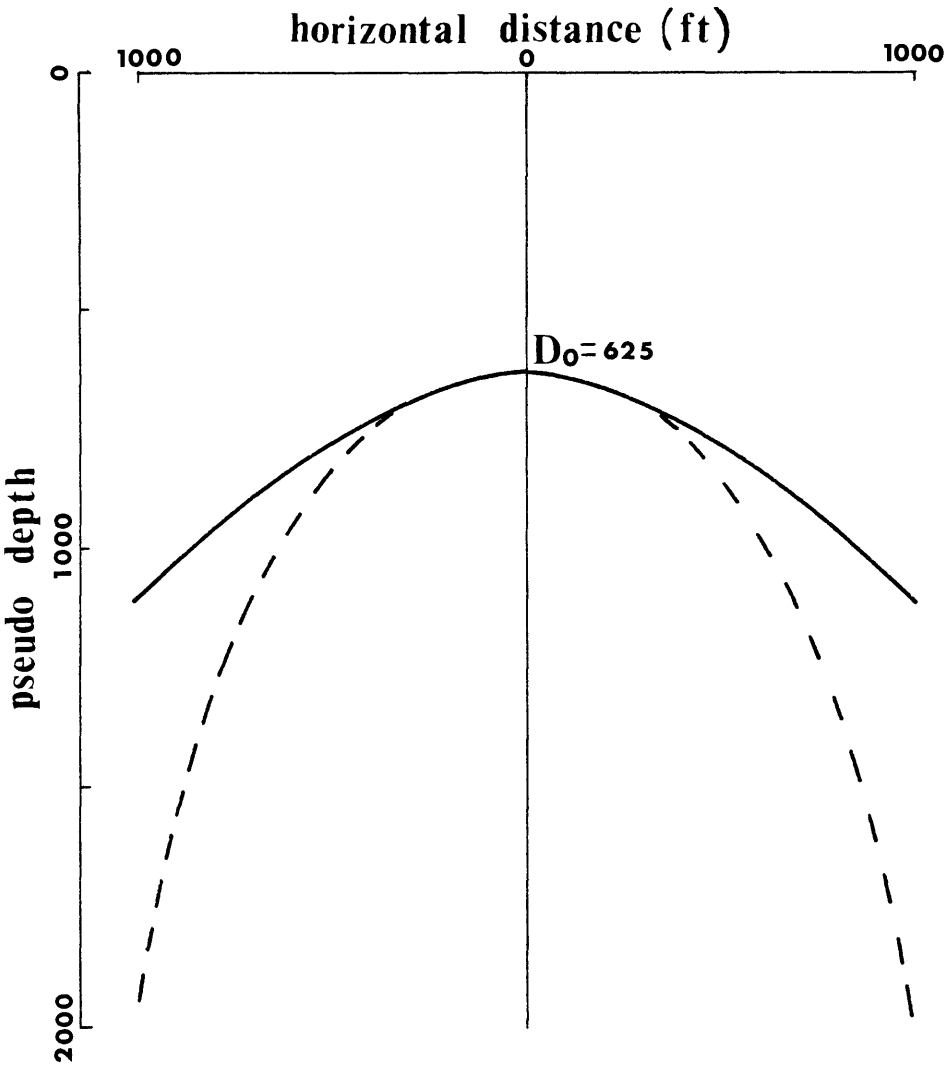

Figure 22: Real and theoretical diffraction curves in the pseudo - depth domain of model 1 for velocity ratio of 2.0 .
D_0 is pseudo - depth of the diffracting point.

TABLE II

Offset	Theoretical (W=1.0)	Real (V.R.=1.2)	Real (V.R.= 2.0)
0.0	625.0	625.0	625.0
50.0	627.0	627.0	627.01
100.0	632.95	632.97	633.10
150.0	642.75	642.85	643.52
200.0	656.22	656.54	658.64
250.0	673.15	673.88	679.0
300.0	693.27	694.68	705.29
350.0	716.33	718.70	738.37
400.0	742.04	745.66	779.21
450.0	770.15	775.24	828.99
500.0	800.39	807.08	889.0
550.0	832.54	840.76	960.70
600.0	866.39	875.84	1045.71

Table II: Theoretical and real diffraction curves in a pseudo - depth domain. Numbers are in feet.

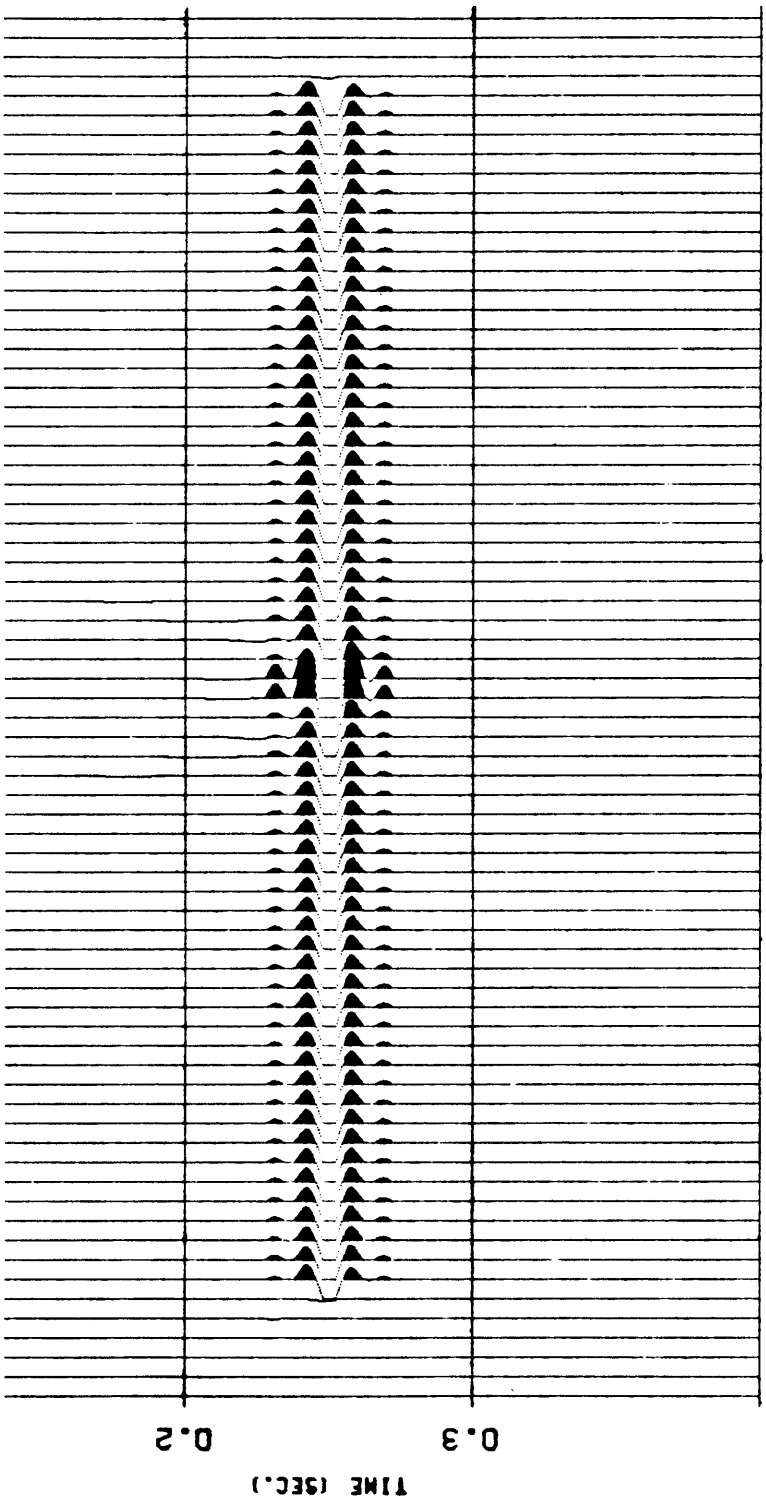

Figure 23: The variable velocity migration of model 1 for velocity ratio of 2.0 . W = 1.0 was used here.

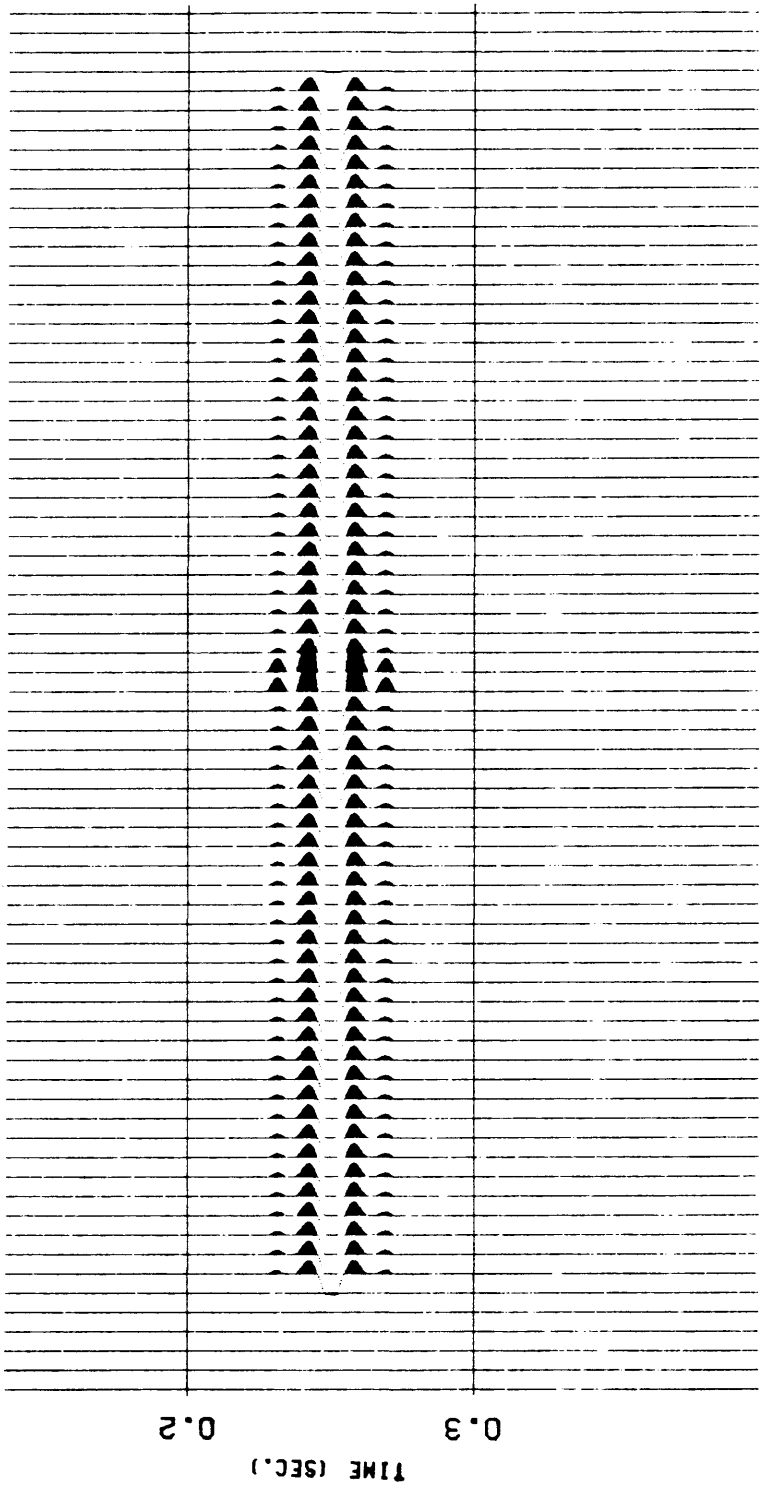

Figure 24: The variable velocity migration of model 1 for velocity ratio of 2.0 . W = 0.0 was used here.

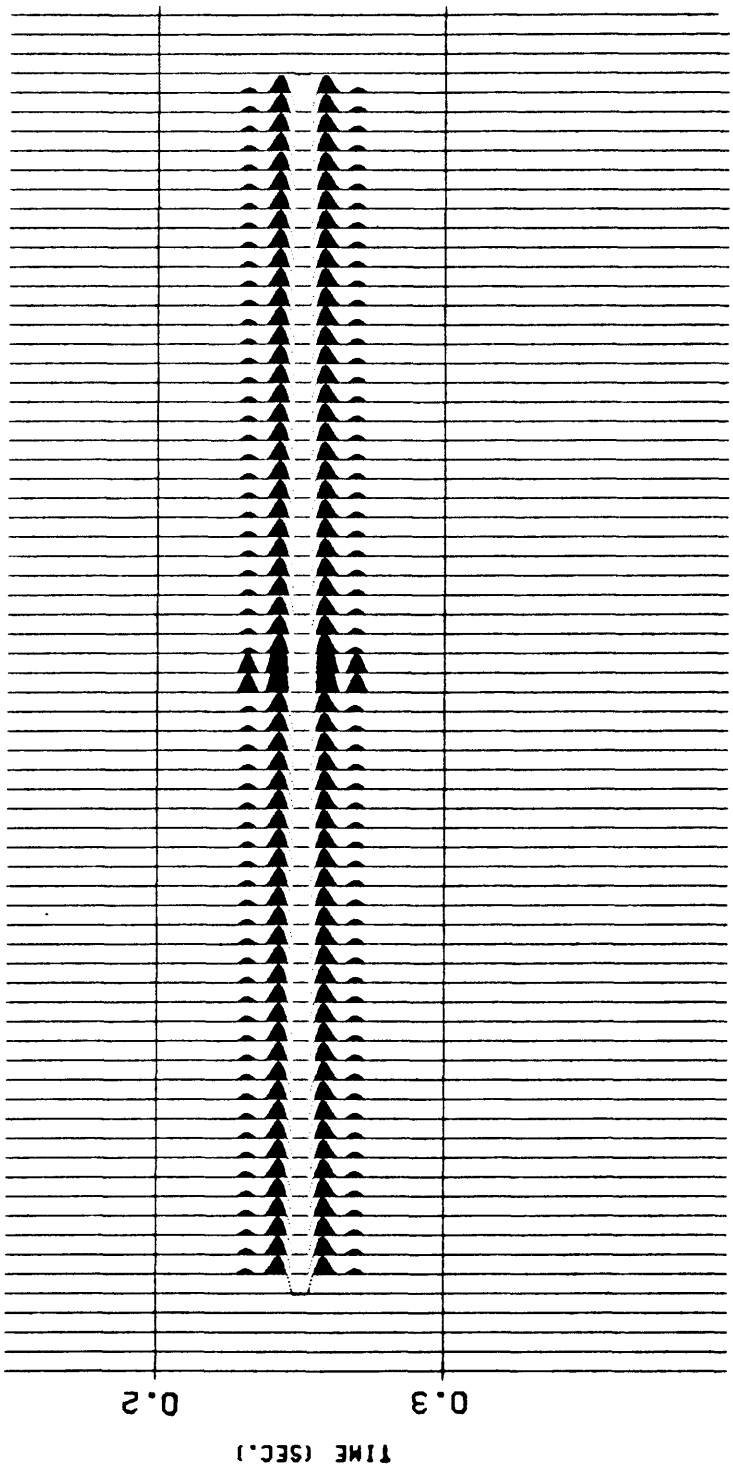

Figure 25: The incremental migration of model 1 for velocity ratio of 2.0.

Four horizons with gradual velocity increase with depth (Model 2)

This model is more complex in terms of determining the value of W. Figure 26 shows the physical parameters for this model which again do not have any geological significance. Velocity increases with depth at a fairly uniform rate. Figures 27 and 28 shows the corresponding normal incidence and wave - theory modelling sections, respectively. The variable velocity migration was carried out with W = 1.0, standard F - K migration, and is shown in Figure 29. This is a very good result, but the deepest diffraction at 0.4 sec. is not collapsed completely. This suggests to us that a change in the value of W might focus the deepest diffraction better. Diet (1979) pointed out that if velocity is a linear function of depth, equation (32) can be used to determine an optimum value of W. Figure 30 shows the velocity distribution of this model. By smoothing this discretized velocity model, an approximate value of α may be found. α is the velocity gradient in the form of $V = V_o + \alpha z$, where V_o is the initial velocity and α is the vertical velocity gradient. α obtained by this method came out to be about 2.3. Consequently, W is obtained by using either equation (32) or the chart in Figure 10 and for α = 2.3, W is about 0.6.

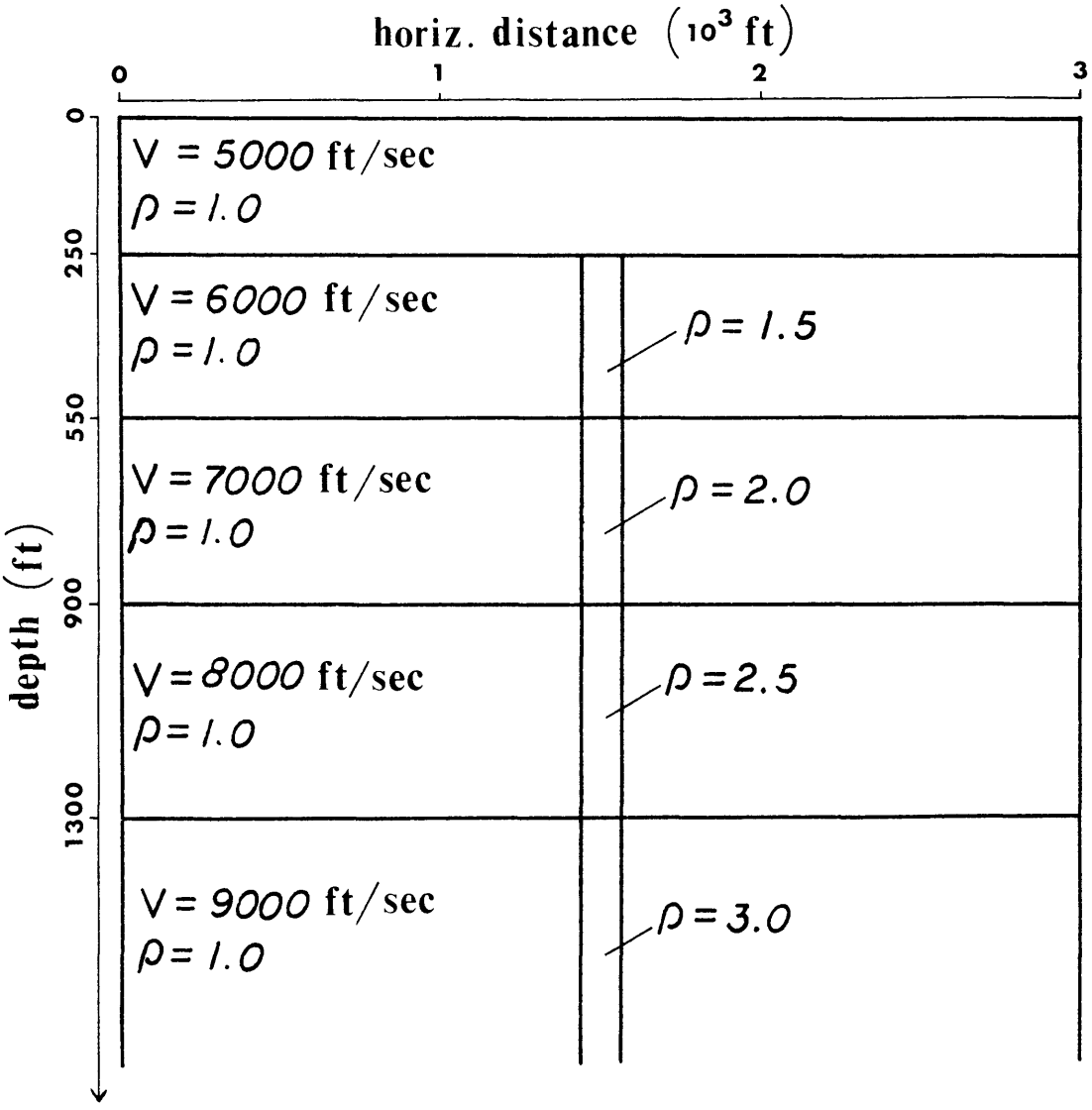

Figure 26: Physical parameters of model 2 .

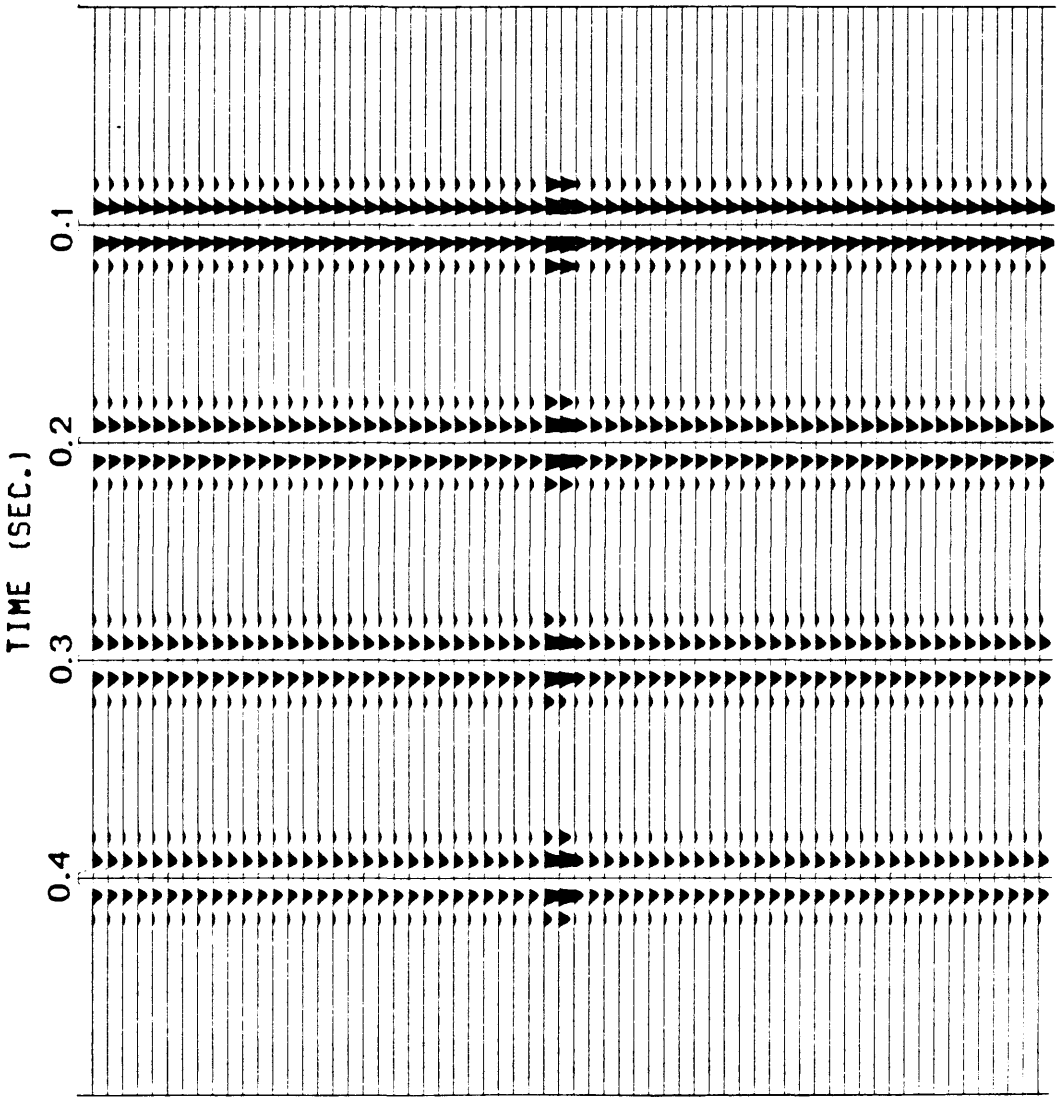

Figure 27: Normal incidence time section of model 2.

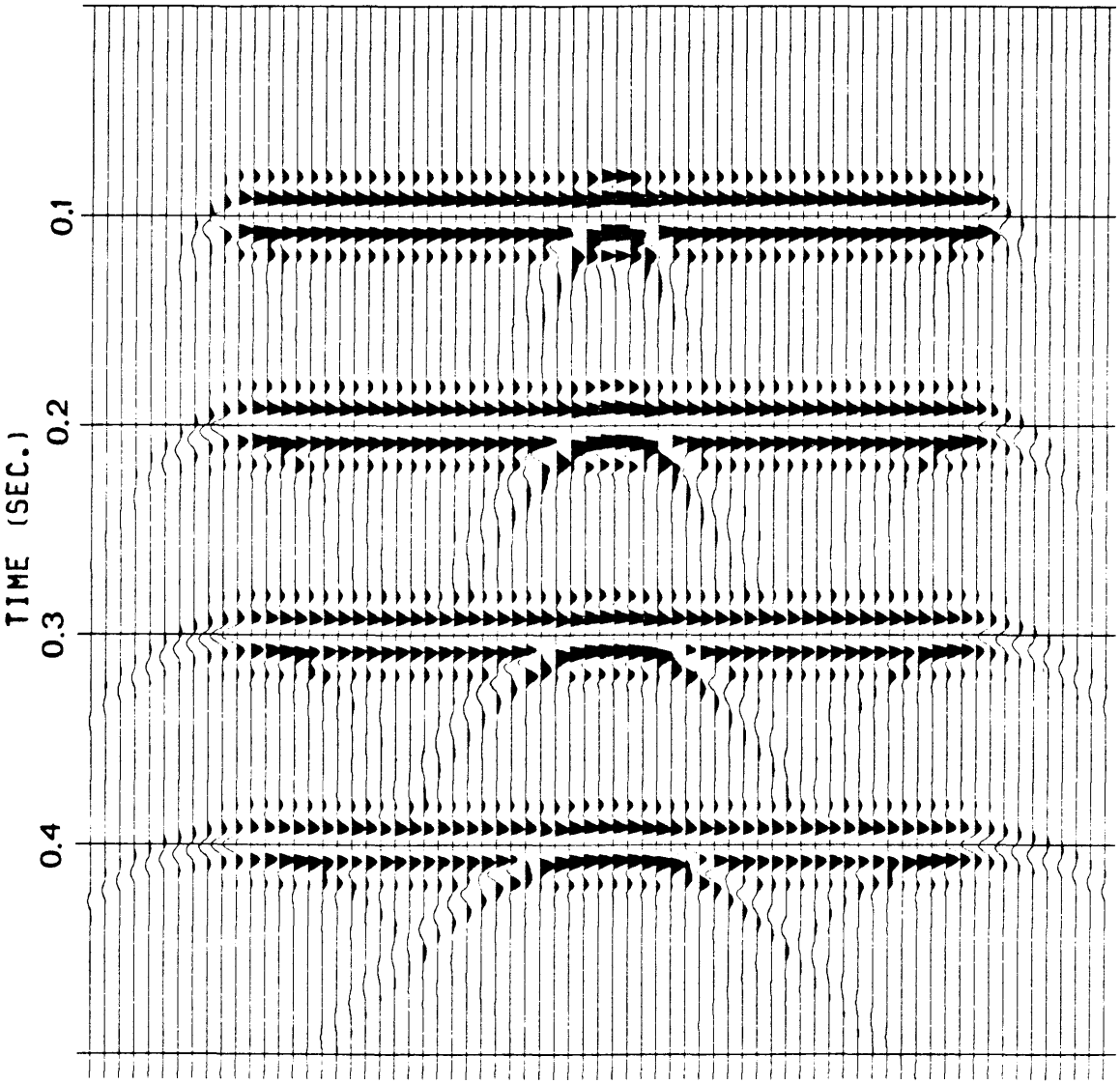

Figure 28: Wave - theory modelling of model 2.

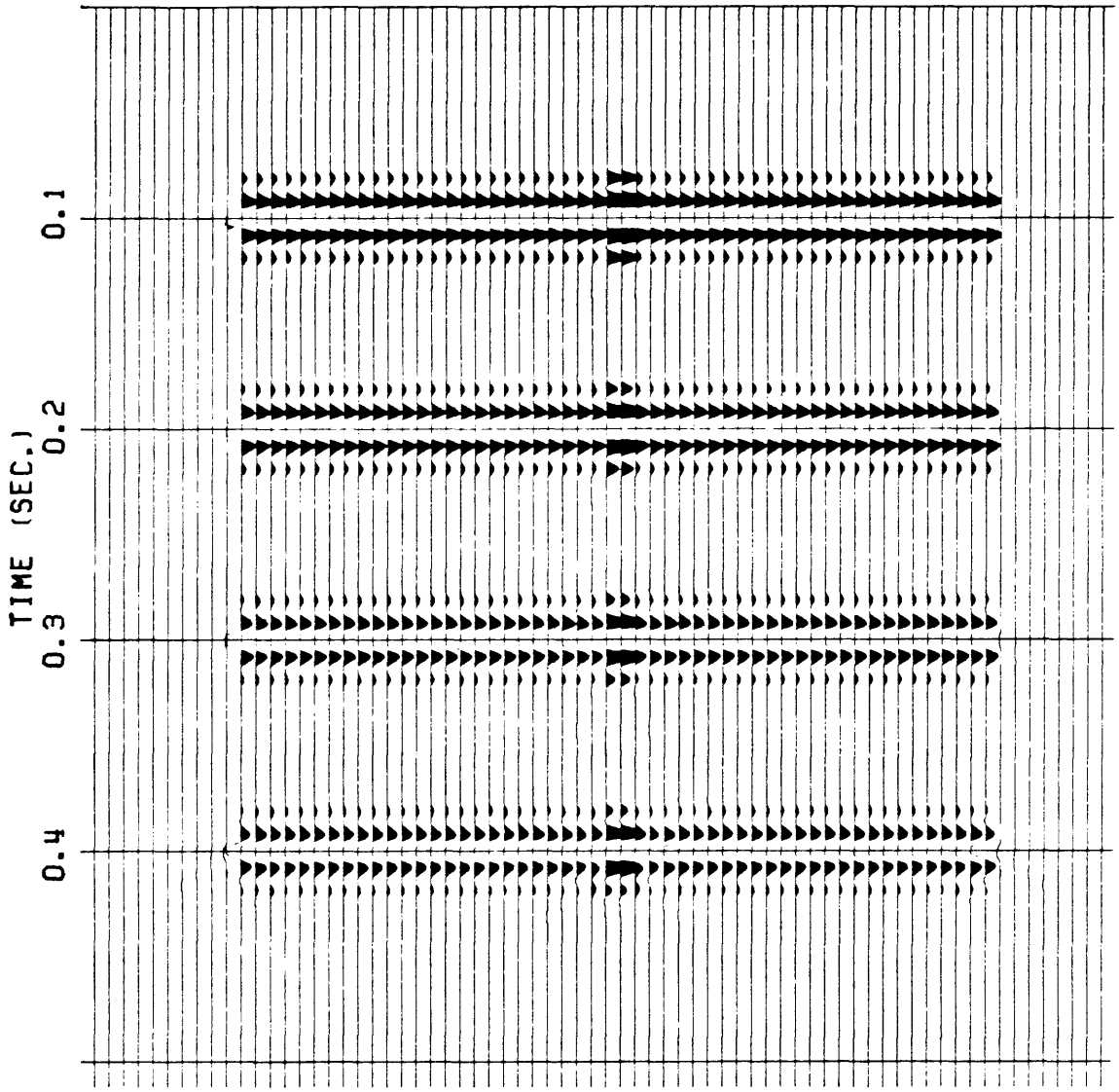

Figure 29: The variable velocity migration of model 2 with W = 1.0 , standard F - K migration.

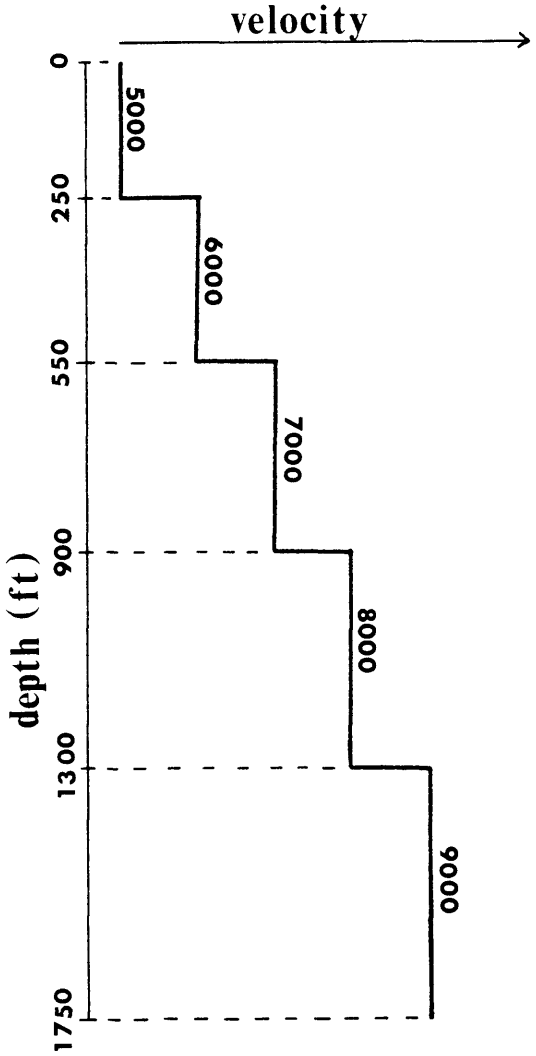

Figure 30: Velocity distribution of model 2, velocities are in feet / sec. .

Figure 31 shows the result of the variable velocity migration with W = 0.6. Unfortunately, the result does not look as good as the one with W = 1.0, even though 0.6 was calculated to be an optimum value of W. Figure 32 shows the result by the incremental method. Again the migration result is in very good shape even down to the deepest horizon.

The simplified picture in Figure 33 illustrates what we think happened to the diffractions in the pseudo - depth domain. The solid lines indicate the theoretical curves with W = 1.0, and the dashed lines indicate the real diffraction curves. The upper three dashed lines are so similar to the solid lines that they are indistinguishable from each other. The top three real diffractions fit very well with the theoretical ones, and this is reflected in the migration result. Notice that for the deepest diffraction the two curves, the theoretical and the real, are not similar enough to give us the correct result with this value of W. Since 1.0 is the maximum value we could assign to W, this should give us the best result. In order to confirm this observation, the synthetic section was migrated with W = 0.0. Figure 34 is the corresponding migrated section. As we have expected, the result is not as good as the one with W = 1.0. The third horizon is now out of focus as well as the fourth one.

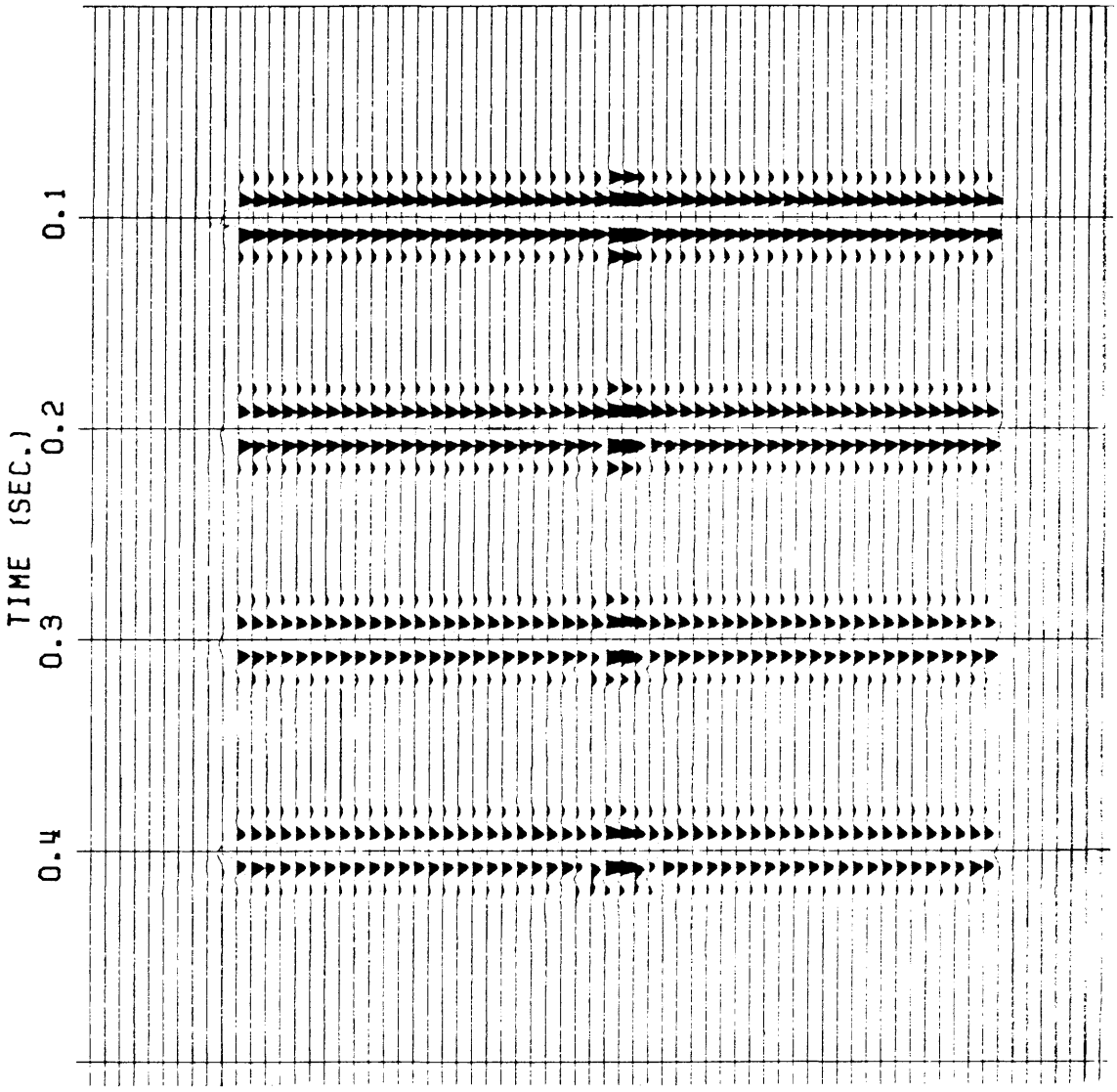

Figure 31: The variable velocity migration of model 2 with W = 0.6 .

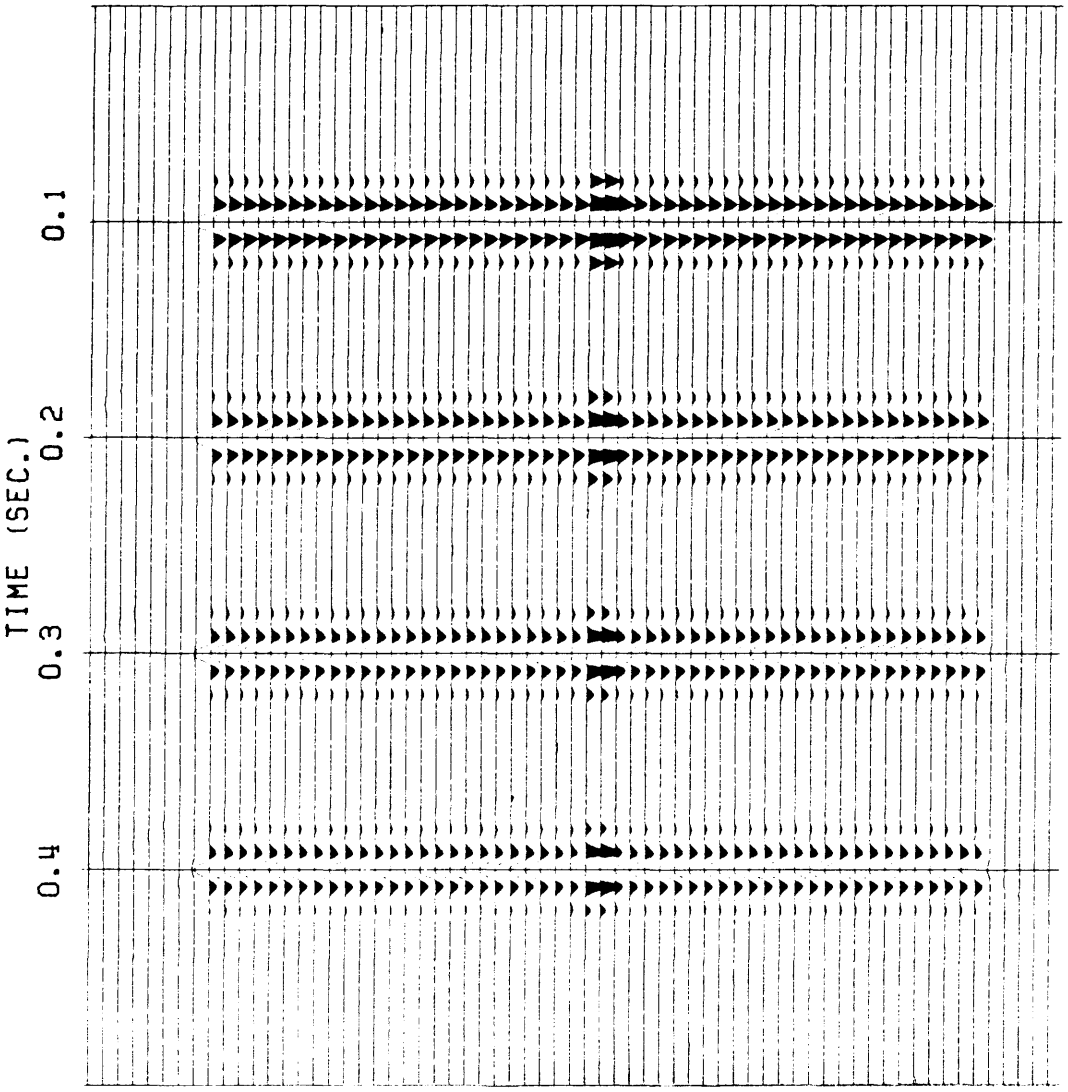

Figure 32: The incremental migration of model 2.

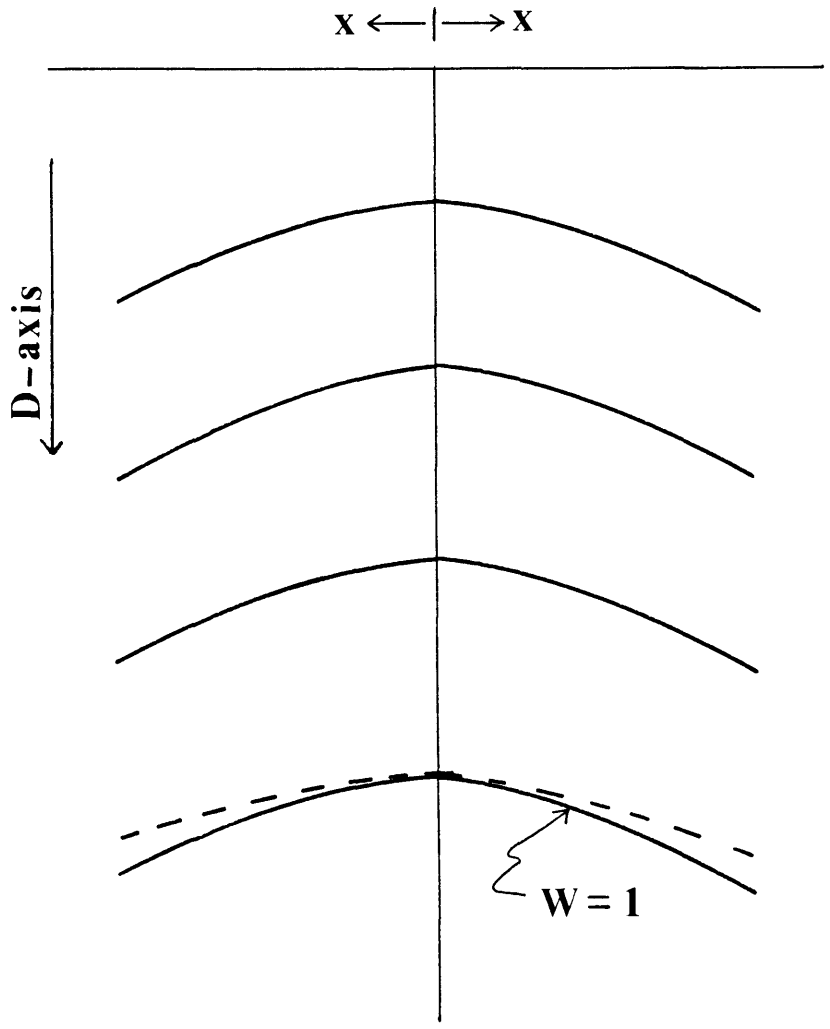

Figure 33: Schematic of real and theoretical diffraction curves in the pseudo - depth domain. Solid lines indicate theoretical curves and dashed lines indicate real curves.

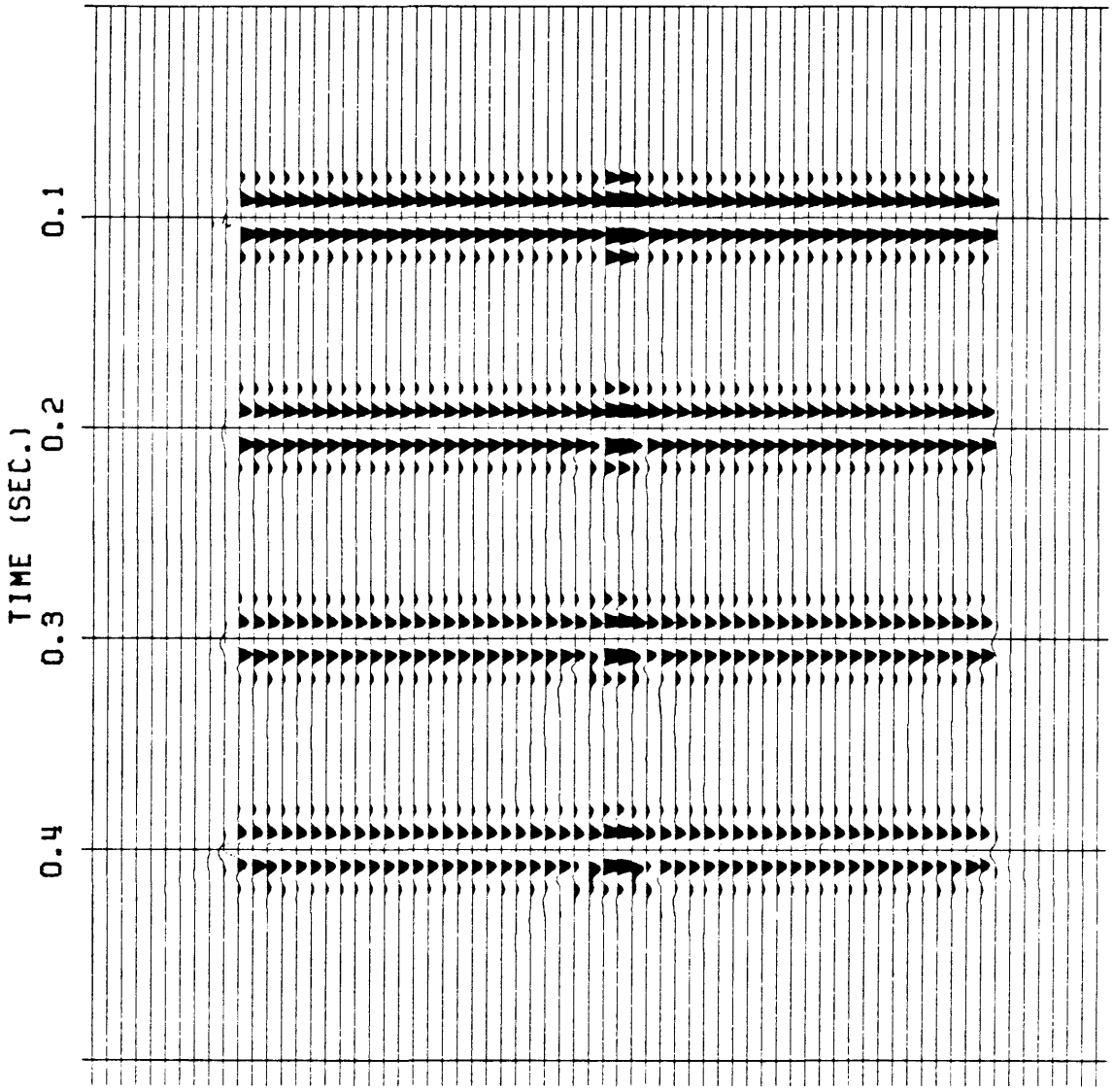

Figure 34: The variable velocity migration of model 2 with W = 0.0 .

We have investigated performances of the two F - K migration schemes with the model having a fairly uniform rate of velocity increase with depth. Both schemes worked well in this type of model. However for the variable velocity method, the optimum value of W did not give us the best result.

Three horizons with large velocity jump. (Model 3)

Model 3 (Figure 35) is very similar to model 2. The difference is that larger steps in the velocity distribution have been made, as is shown in Figure 36. This way, the determination of an optimum W may be difficult. Figures 37 and 38 show the normal incidence and wave - theory modelling sections of this model, respectively. As we have done in model 2, we could calculate an optimum value of W. However, let us proceed to migrate the synthetic section with W = 1.0, standard F - K migration. Figure 39 shows this migration result. Again the deepest horizon is undermigrated, and this time it is worse than the previous example. This fact tells us that the theoretical diffraction curve with W = 1.0 is less open than the real diffraction curve in the pseudo - depth domain. This observation is depicted in Figure 40, again the solid lines indicate the theoretical curves and the dashed lines indicate the real curves.

α and the optimum W obtained for this model are about 2.3 and 0.6, respectively. These are almost the same values as those in the previous example. If we are to migrate this data with W = 0.6, the result is expected to be less focused. Indeed it is not quite as good of a

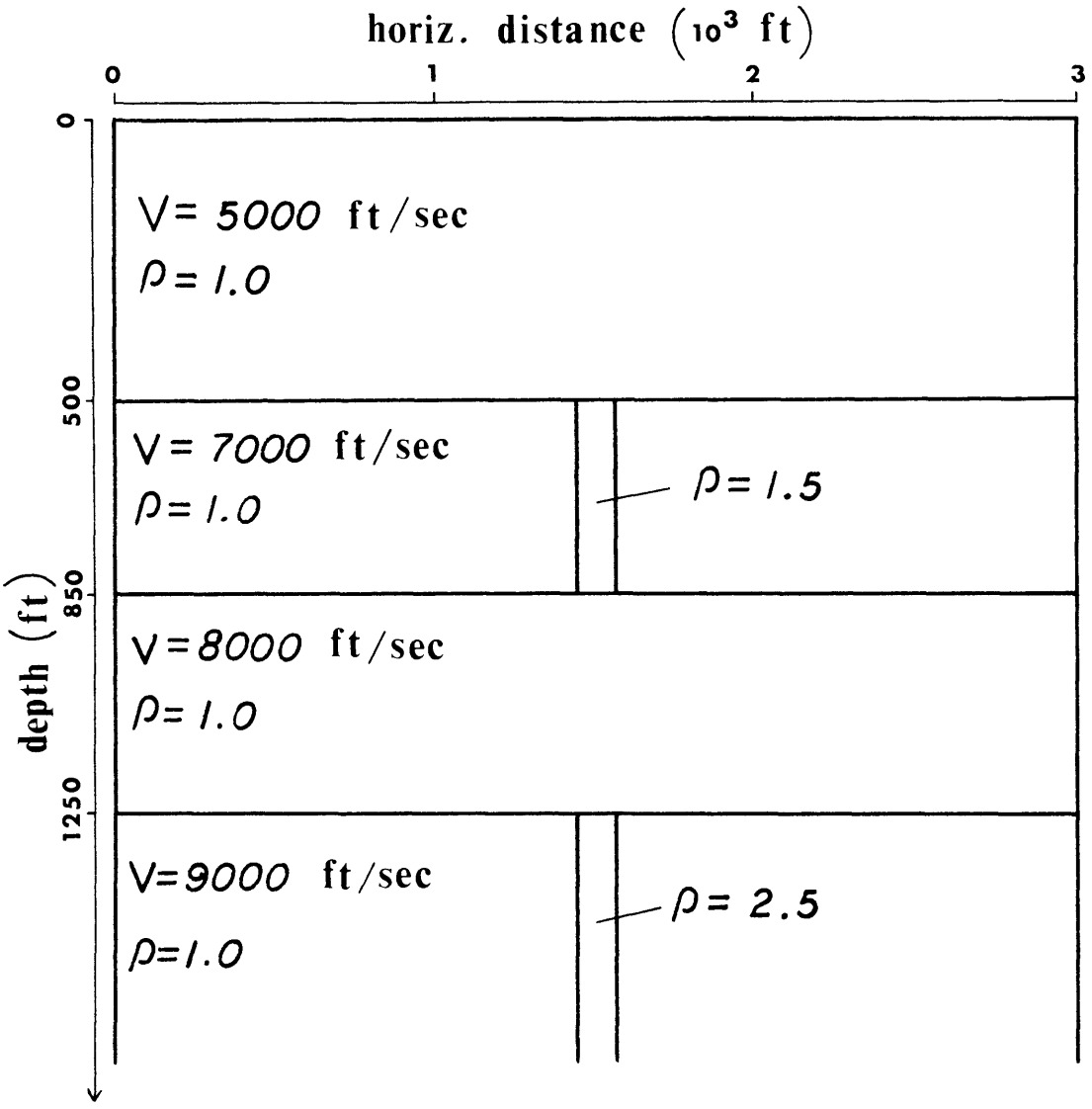

Figure 35: Physical parameters of model 3.

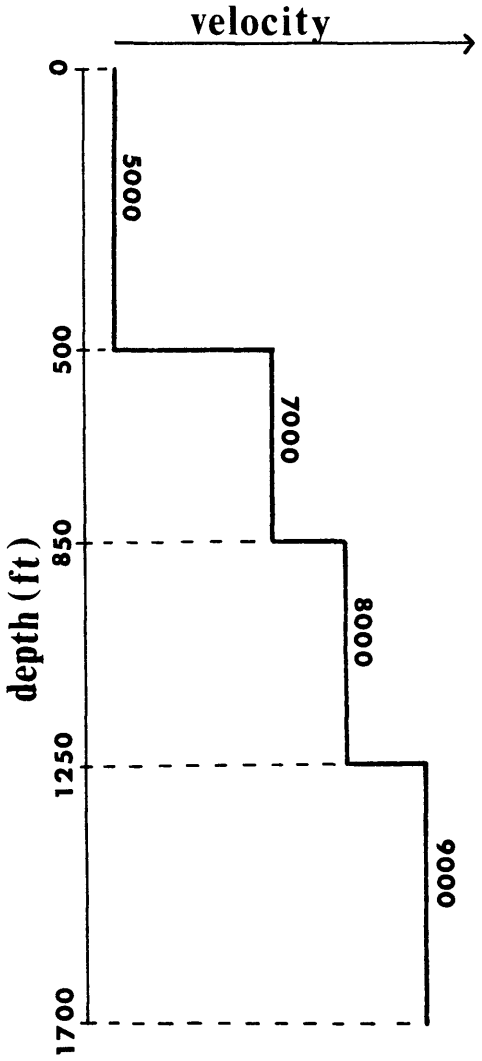

Figure 36: Velocity distribution of model 3, velocities are in feet / sec. .

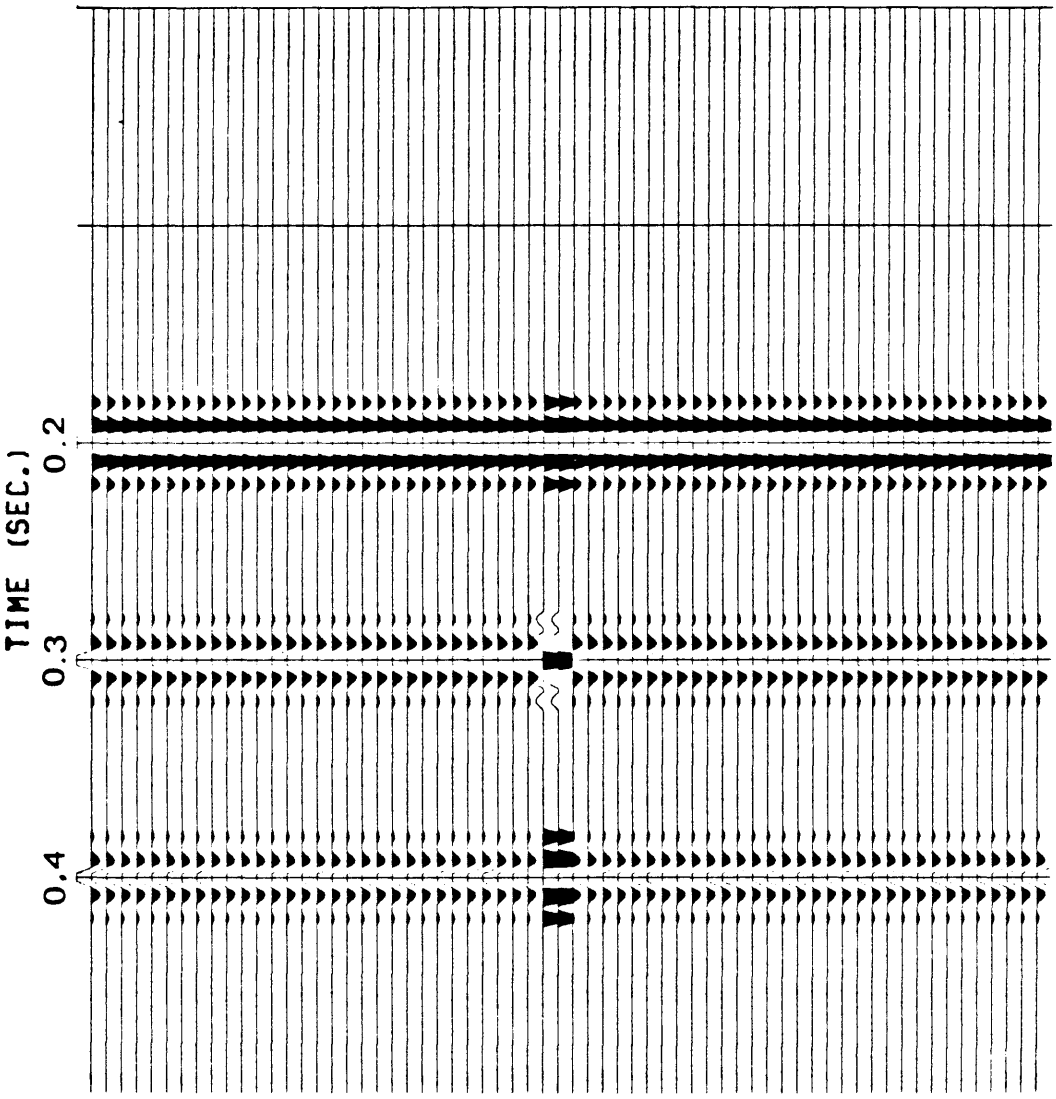

Figure 37: Normal incidence time section of model 3.

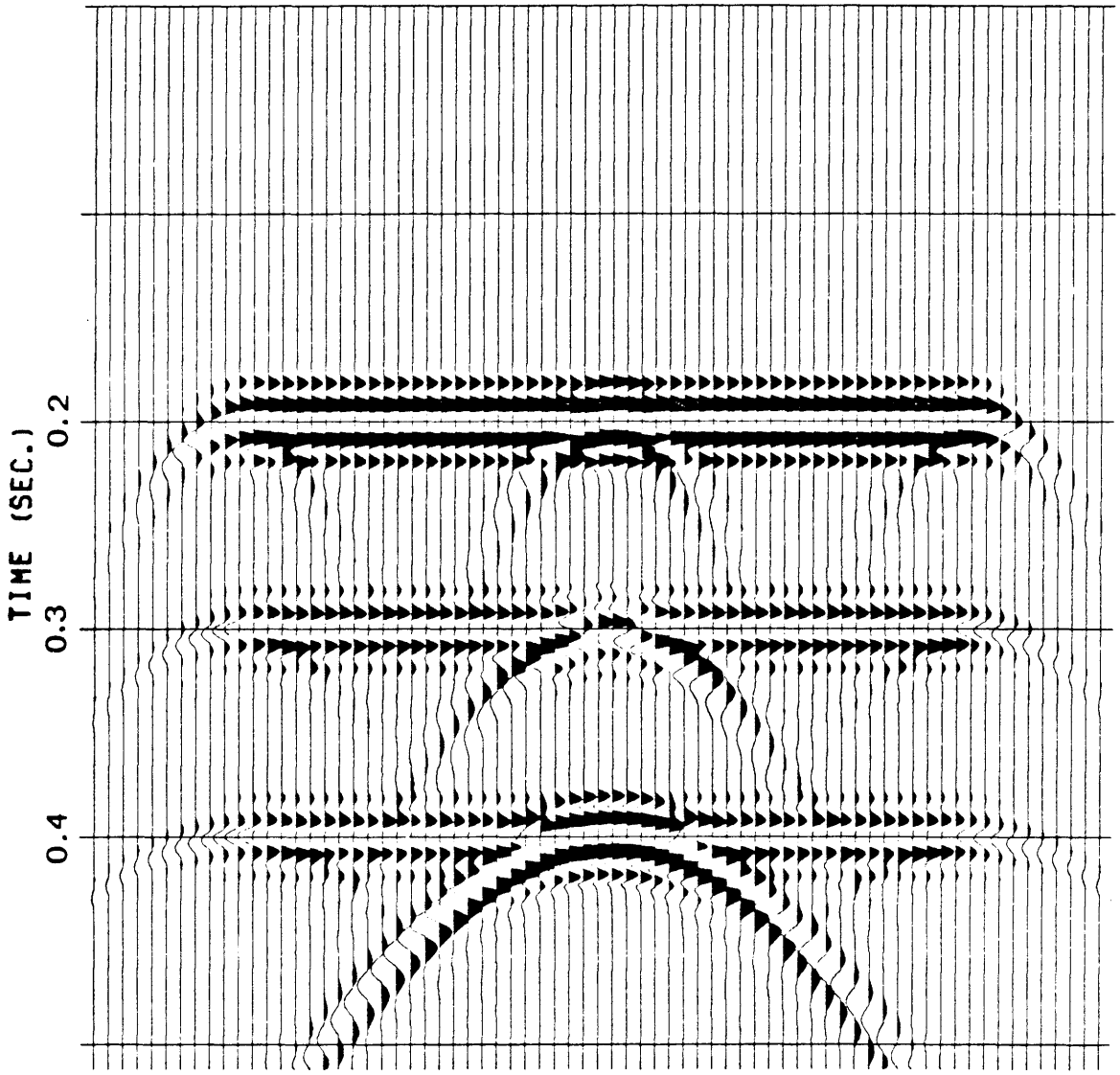

Figure 38: Wave - theory modelling of model 3.

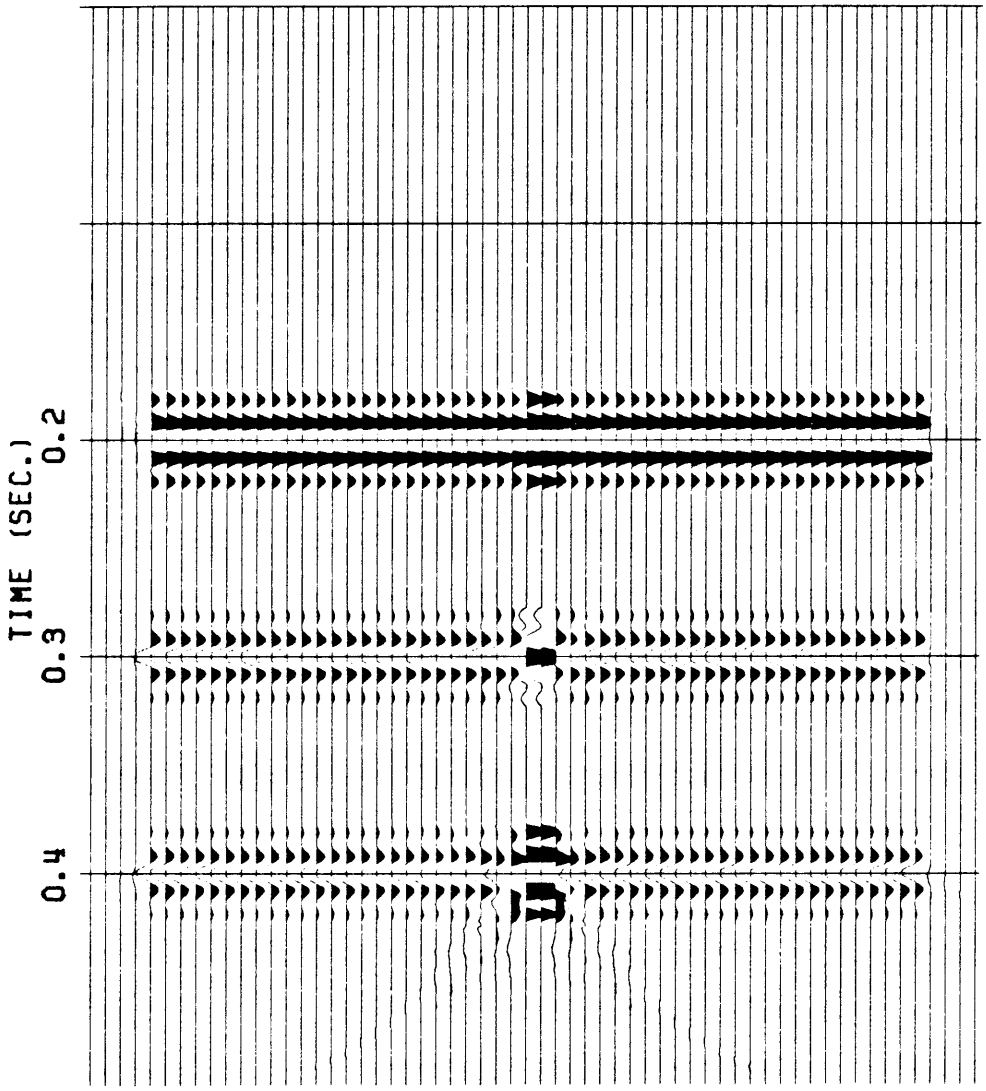

Figure 39: The variable velocity migration of model 3 with W = 1.0 , standard F - K migration.

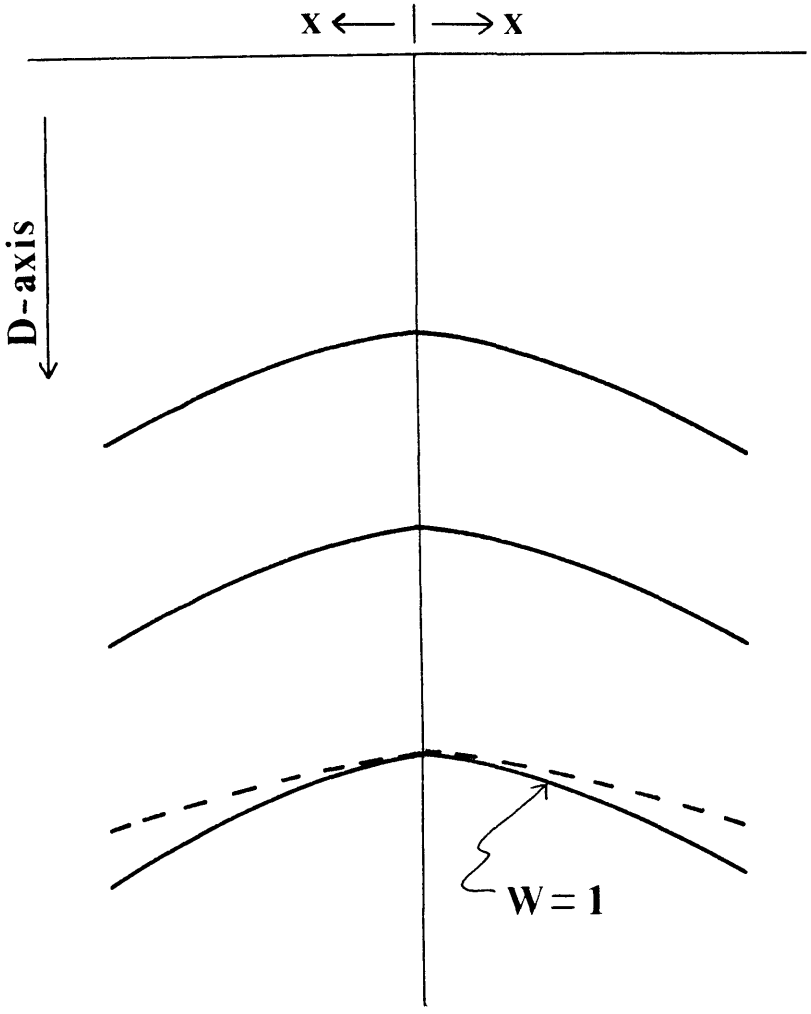

Figure 40: A schematic of real and theoretical diffraction curves in the pseudo - depth domain. Solid lines indicate theoretical curves and dashed lines indicate real curves.

result as the one with W = 1.0. Figure 41 shows the corresponding migrated section with W = 0.6.

On the other hand, the incremental method gives us a well focused result as is shown in Figure 42. There is some undermigrated energy around the deepest horizon, however this result is superior to the one obtained by the variable velocity method.

Figures 43, 44, and 45 show a sequence of migration results with W = 0.4,-0.5, and -0.75, respectively. We observe that the migration results become worse with each decrease in W.

In this model, the large step in velocity is introduced. Since this large velocity step violates the rule of linearly increasing velocity with depth used to calculate the value for W (Diet, 1979), the variable velocity scheme with this value of W did not quite give us an acceptable result.

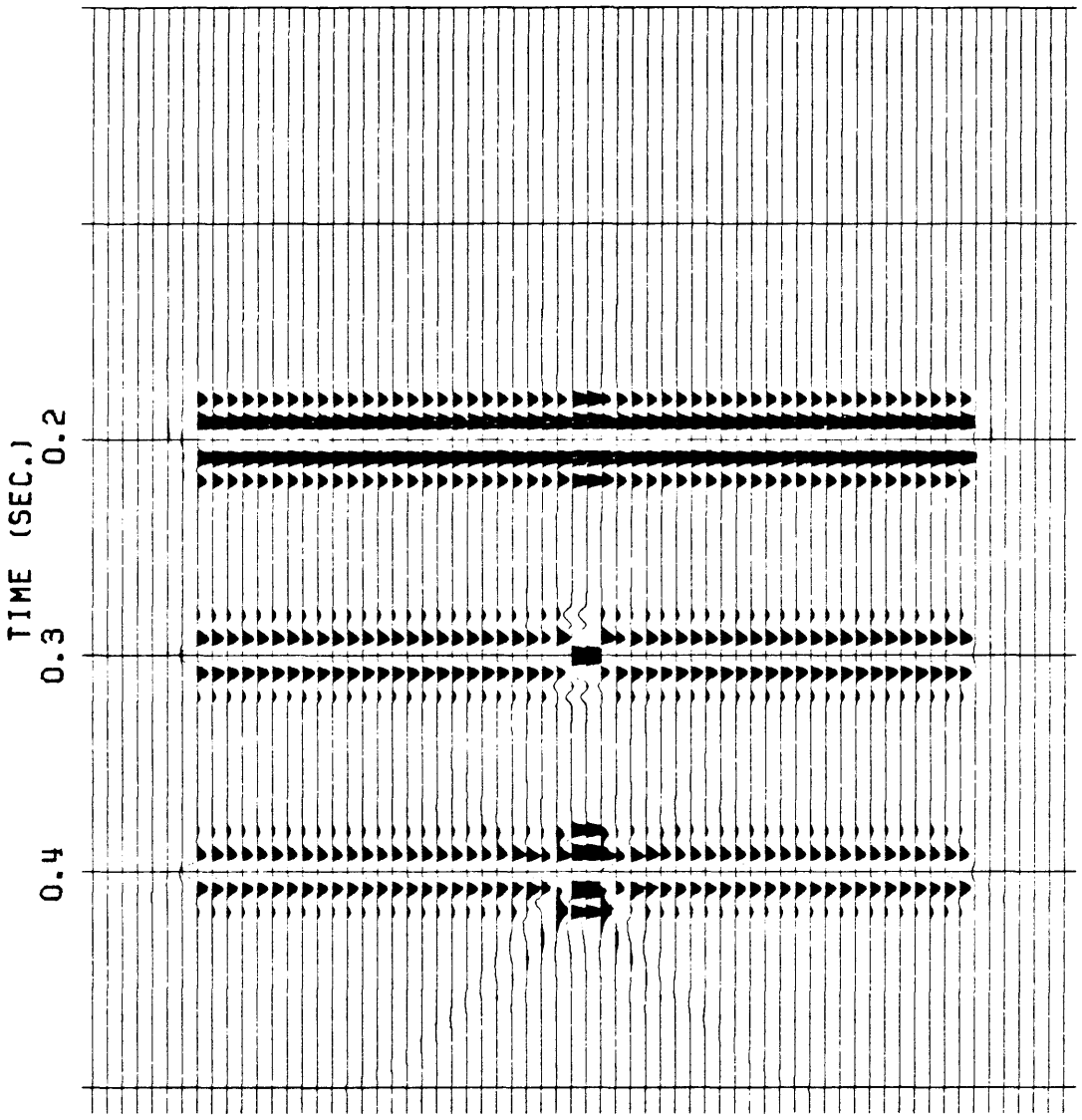

Figure 41: The variable velocity migration of model 3 with W = 0.6 .

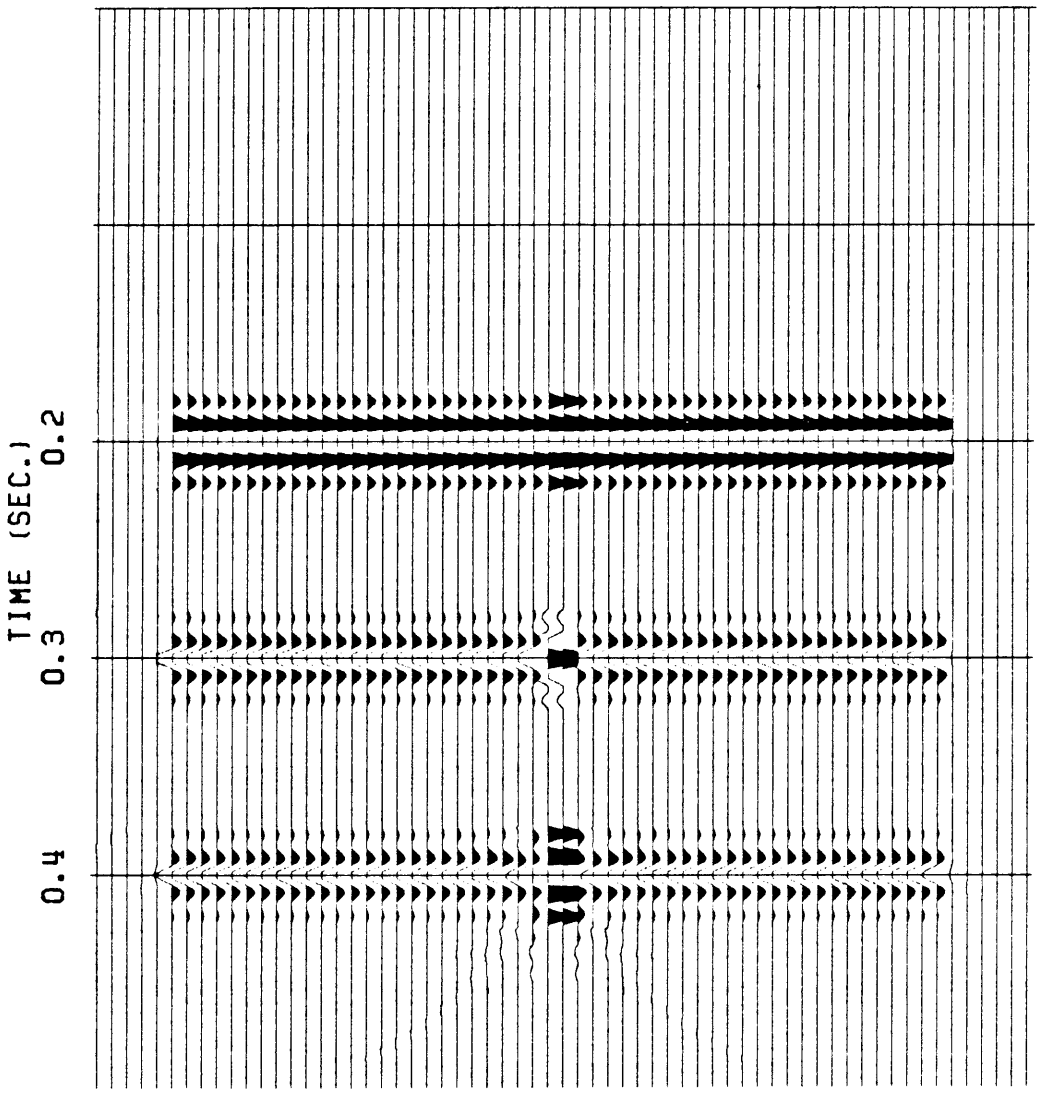

Figure 42: The incremental migration of model 3.

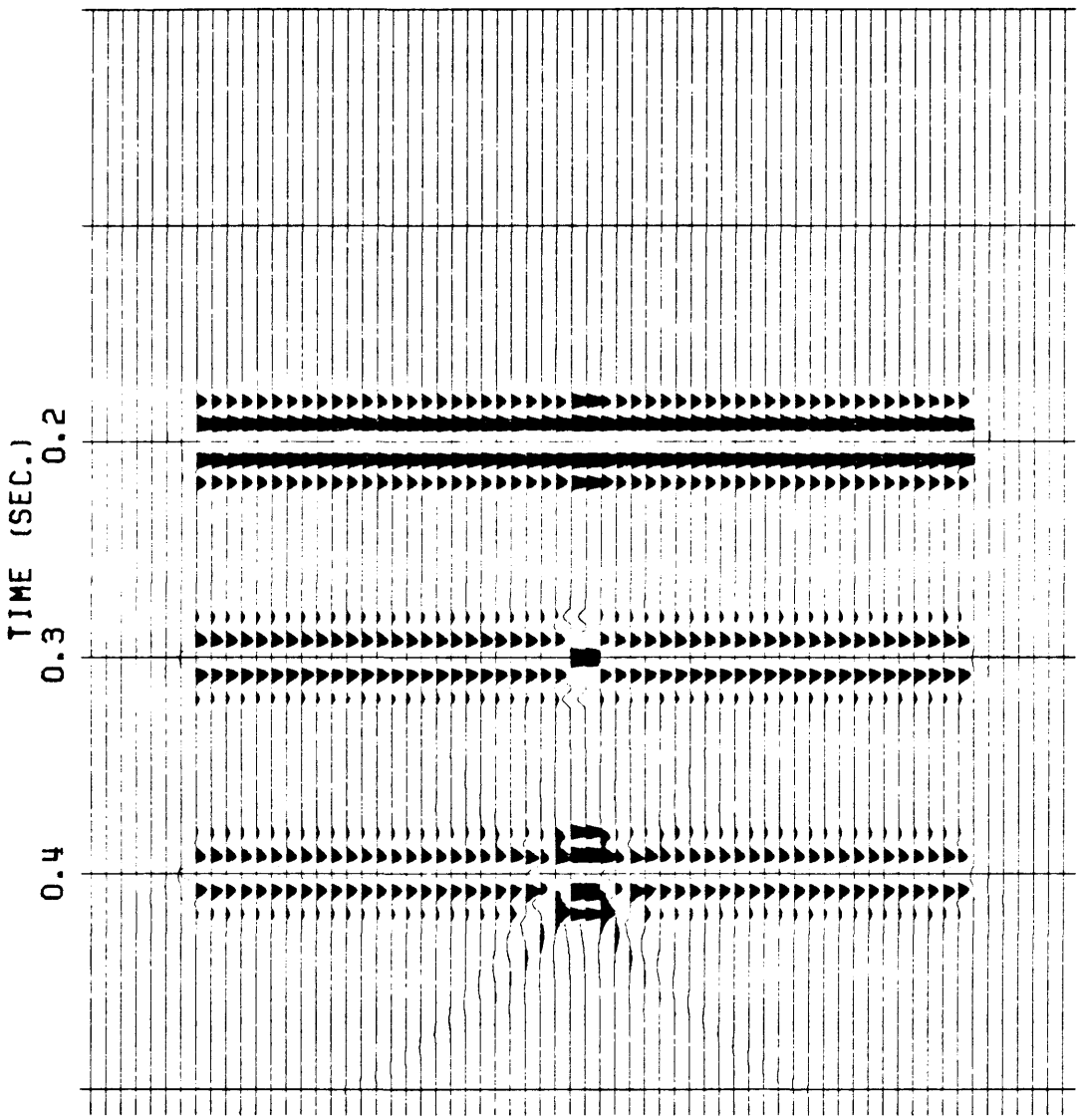

Figure 43: The variable velocity migration of model 3 with W = 0.4 .

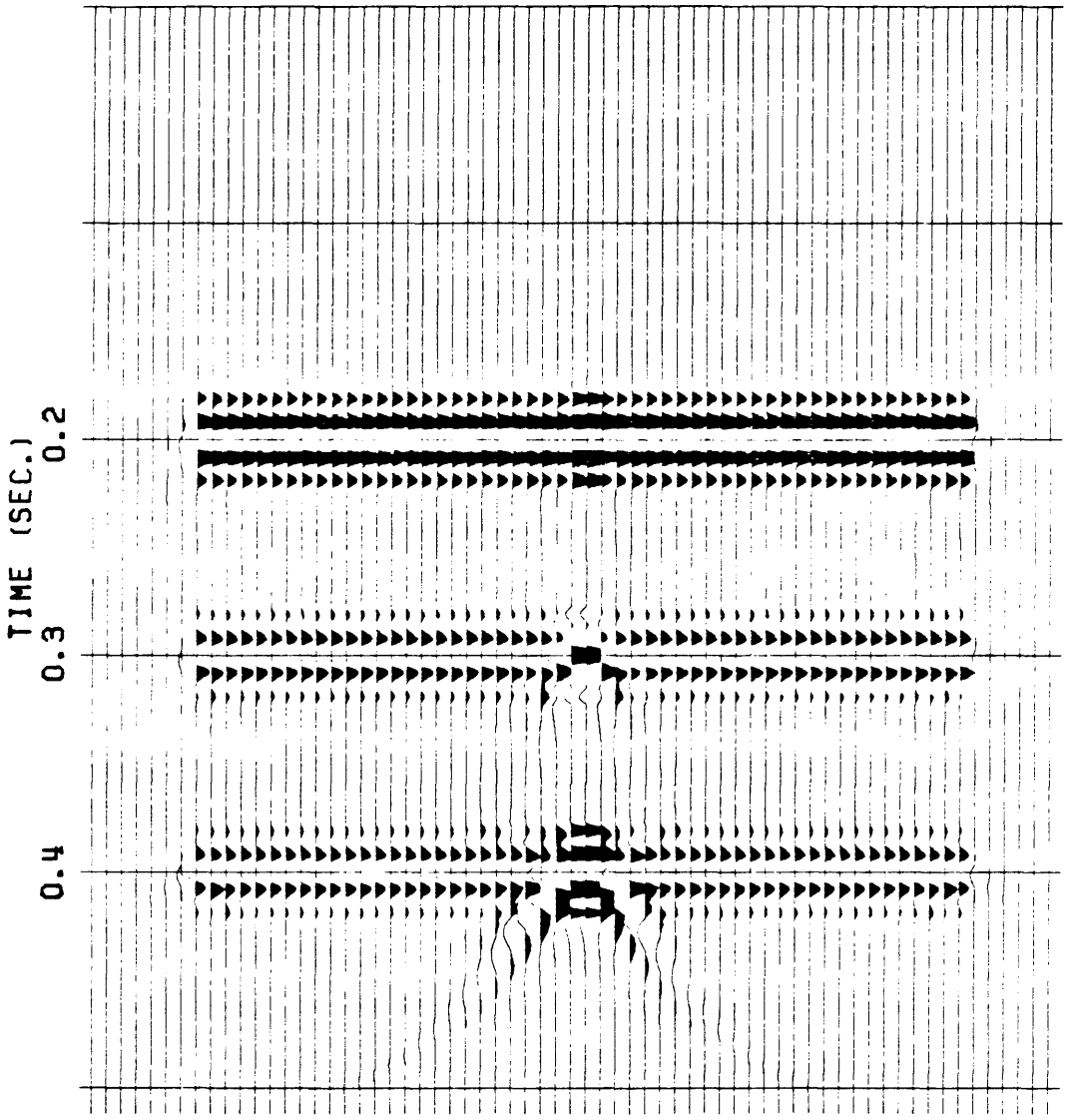

Figure 44: The variable velocity migration of model 3 with W = -0.5 .

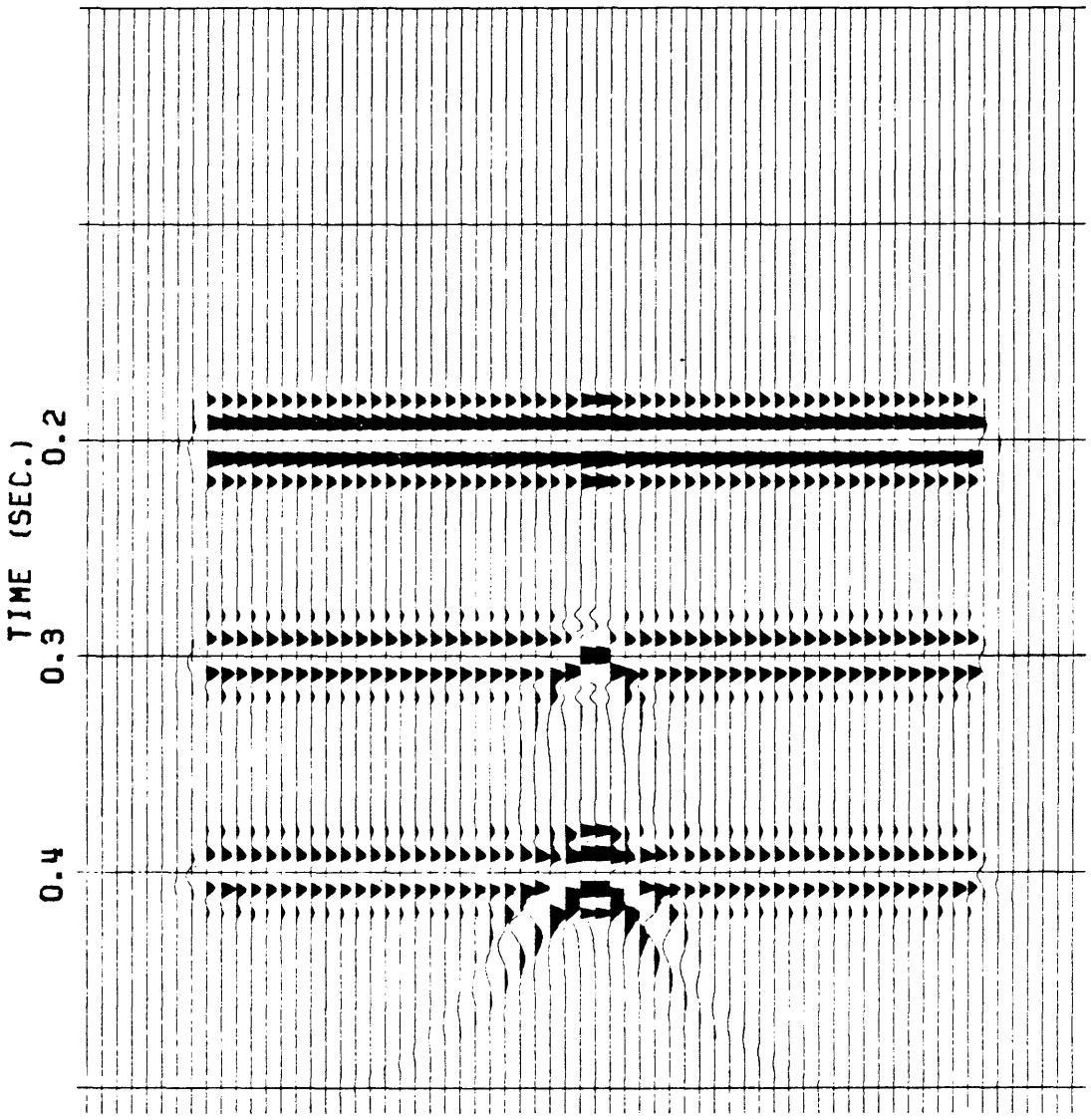

Figure 45: The variable velocity migration of model 3 with W = -0.75 .

POSSIBLE EXTENSION OF THE INCREMENTAL METHOD IN THE PRESENCE OF LATERAL VELOCITY VARIATIONS

From the investigation carried out for vertical velocity gradients, it is obvious that the incremental method is superior to the variable velocity method.

Naturally, the next thing we try is to consider the applicability of the incremental method in the presence of both vertical and lateral velocity variations.

In order for this scheme to be properly applied, velocity inhomogenuities must be in the z - direction only. In this scheme for vertical velocity variations, we assumed constant interval velocities within discretized layers. Therefore we really do not have any analytical expressions for lateral velocity variations. But there is a possible detour around this difficulty.

If a velocity function of x and t; that is, V(x,t) may be approximately by a new velocity function of t, V(t), then the incremental method can be used to migrate sections.

For the variable velocity method, we understand that the minor lateral velocity variation is taken care of by the pseudo - depth conversion. The idea we use here is similar to the pseudo - depth conversion.

Let $\overline{V}(t)$ be a mean average velocity function for an entire section, then a depth function, $\overline{D}(t)$, associated with $\overline{V}(t)$ can be obtained by

$$\overline{D}(t) = \frac{\overline{V}(t) \cdot t}{2} \tag{40}$$

There is a velocity function associated with each CDP, so a depth function at a CDP is described by

$$D_j(x,t) = \frac{V_j(x,t) \cdot t}{2} \tag{41}$$

where j is the index for CDP.

Then the next step is to find new times t's on the average depth scale ($\overline{D}(t)$) for depths at each CDP. Figures 46 and 47 illustrate the ways to find new time t's, and to interpolate traces.

Then the migration procedure is modified as follows:

1) Interpolation of traces (t - t')

2) Migration with $\overline{V}(t)$

3) Interpolation of traces back to original time (t' - t).

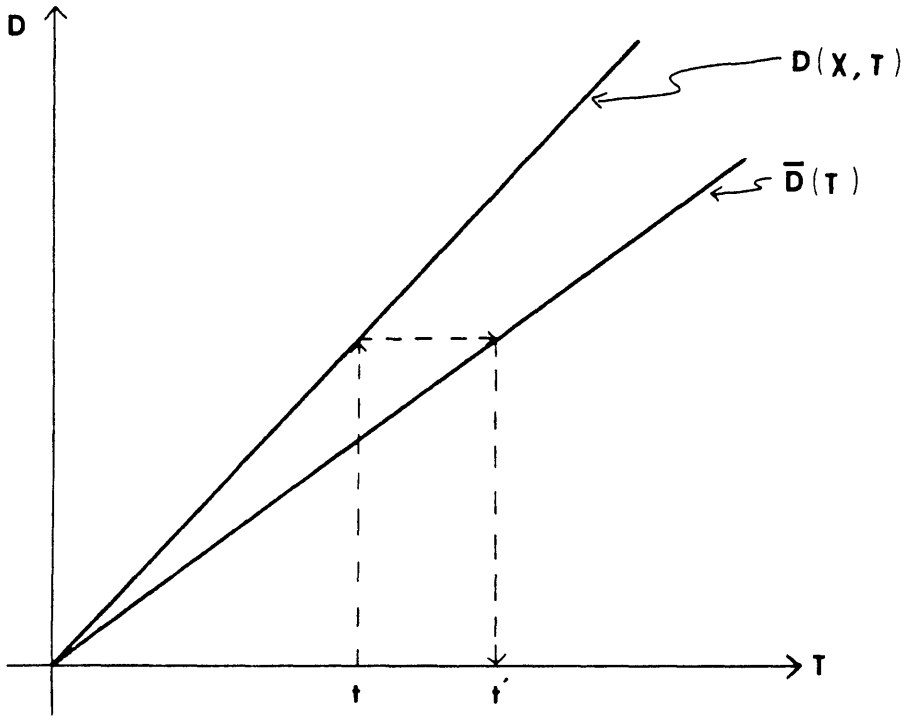

Figure 46: The simplified diagram showing how to obtain new time t' .

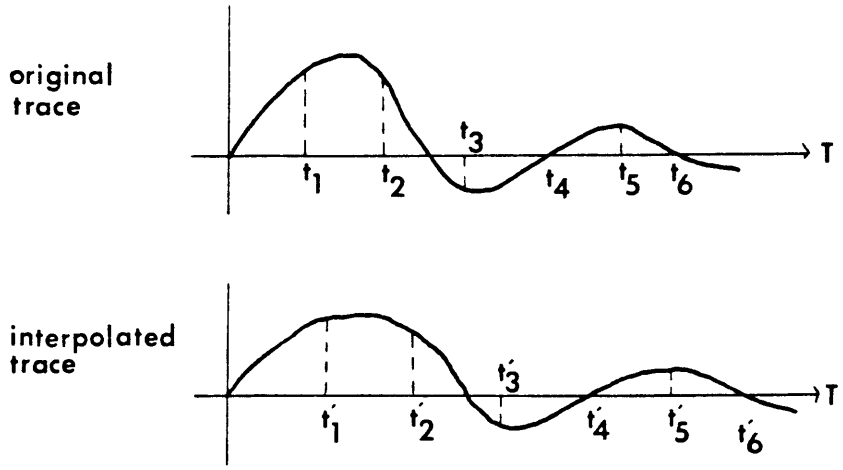

Figure 47: Schematic of trace interpolation.

Dipping layer with diffractor. (Model 4)

This model tests the applicability of the incremental method on a simple example of a lateral velocity variation. Figure 48 shows the physical parameters of model 4. This is intended to simulate a water bottom reflection with a dip of about 14°. A diffractor is added beneath the dipping water bottom. It might be noted that the large velocity contrast is used here in order to have sufficient lateral shift of a diffraction apex discussed below. Figure 49 is the synthetic seismic section with CDP spacing of 50 feet generated by a ray - tracing program. A spike section was convolved with a zero - phase symmetrical Ricker wavelet with a pseudo period of 36 Hz. to produce Figure 49.

Note that in Figure 49 the true apex position, shown by the arrow, of the diffraction has shifted to the left. Note also the asymmetric shape of the diffraction. The lateral shift of the apex is about 450 feet and this is caused by presence of the dipping bottom layer as is shown in Figure 50.

Migration of this section with W = 1.0, standard F - K migration, is shown in Figure 51. Velocity functions were input at both edges of the section and the program is supposed to interpolate to obtain velocity functions for CDPs between them. The apex has shifted back closer to

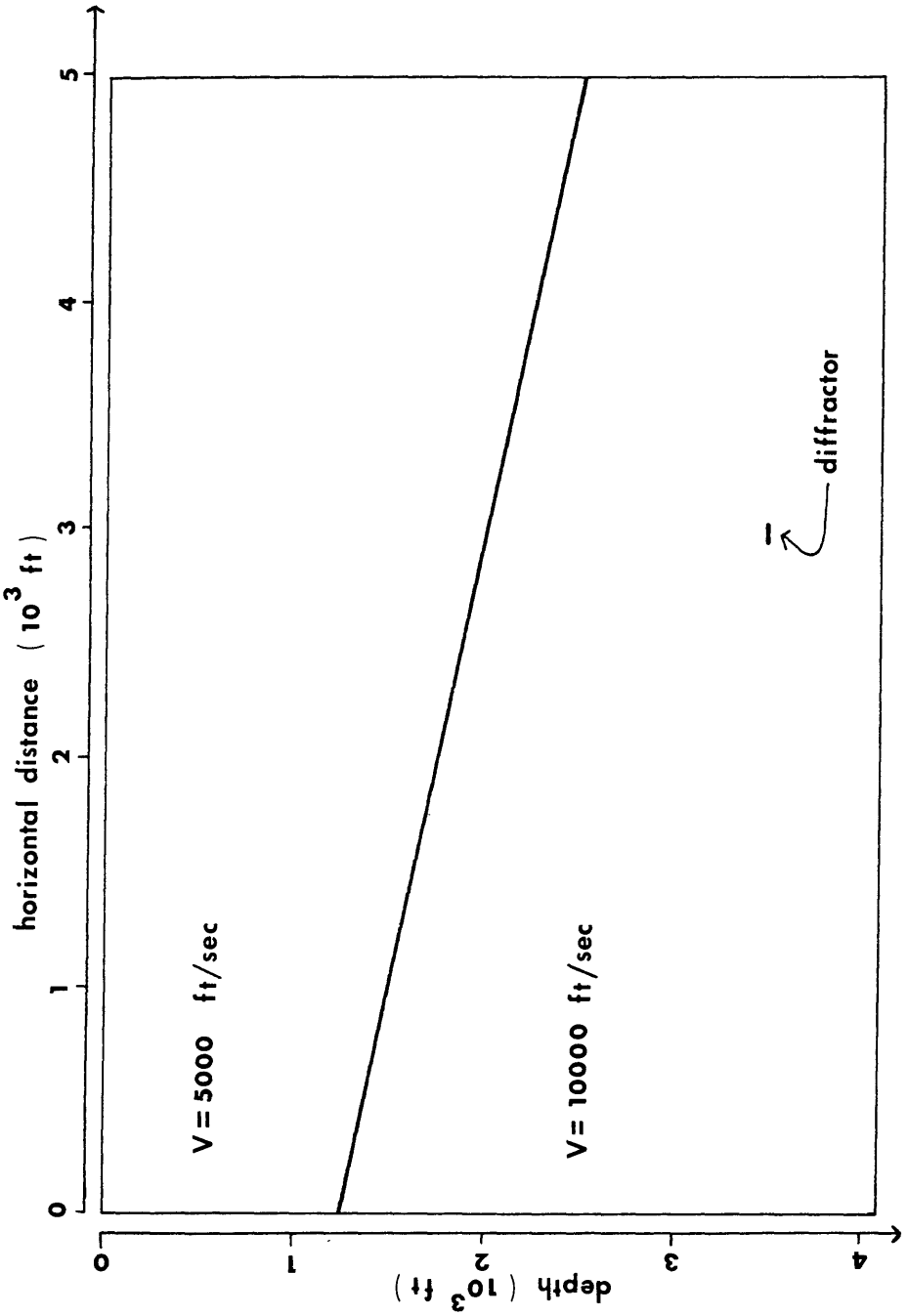

Figure 48: Physical parameters of model 4.

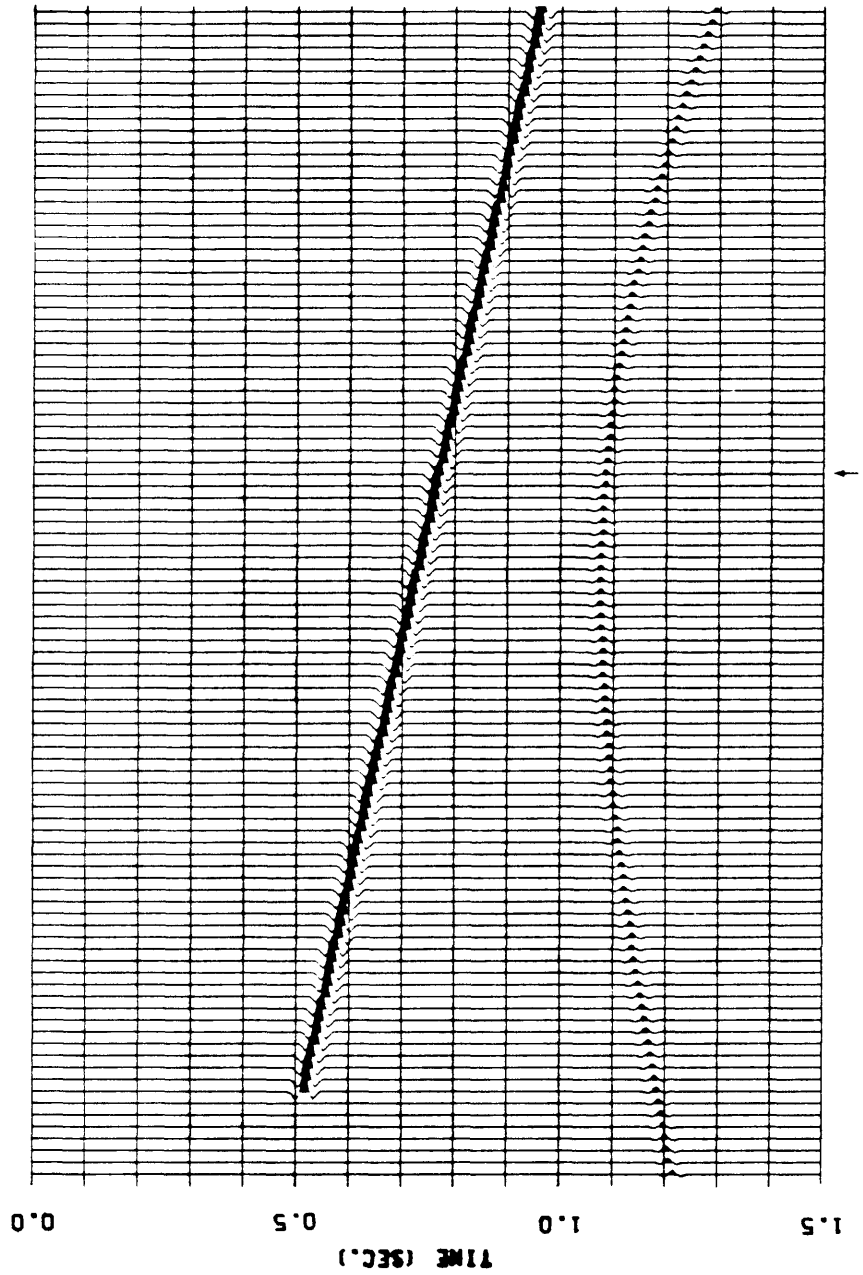

Figure 49: Synthetic time section of model 4. The arrow shows the correct position of the diffractor apex.

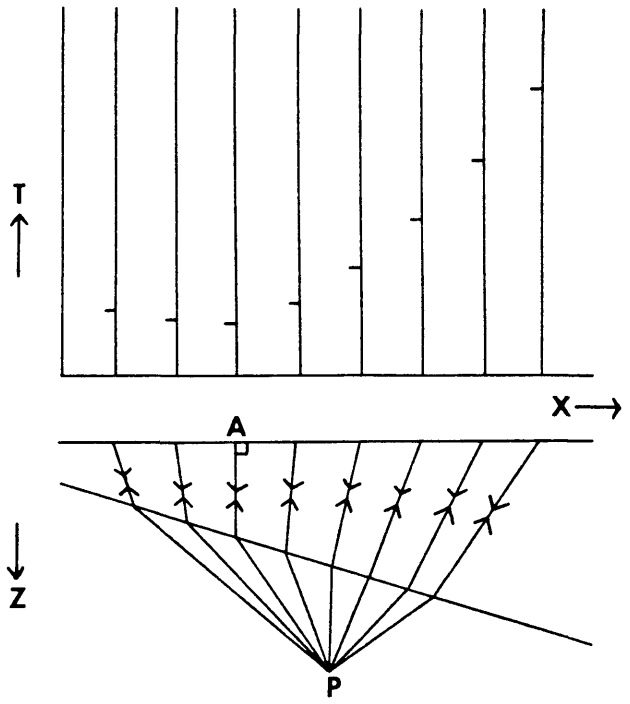

Figure 50: Bottom: raypaths from a scatterer below a dipping interface. The interface separates a low-velocity layer above from a high-velocity layer below. Top: resulting diffraction curve on a time section.
(After Larner et al., 1981).

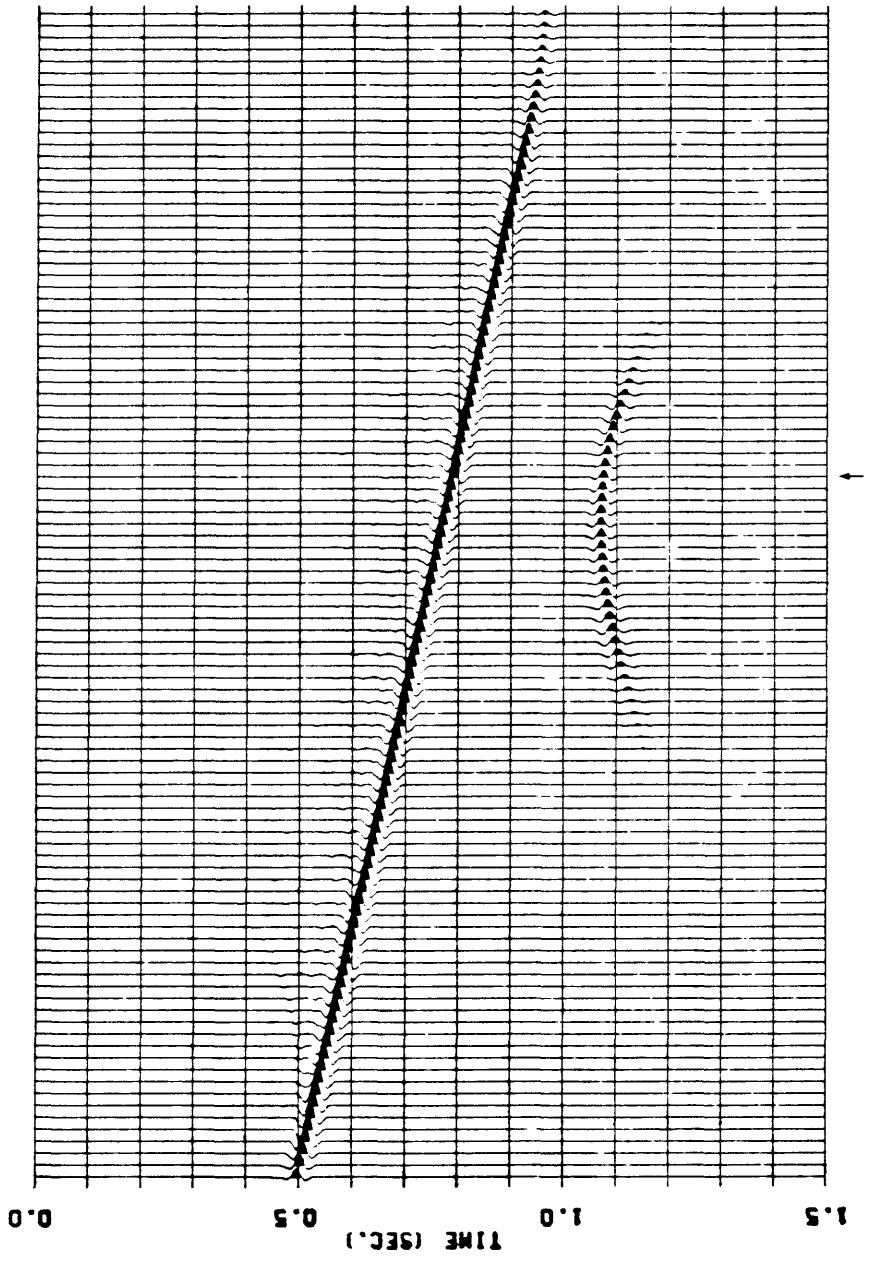

Figure 51: The variable velocity migration of model 4 with W = 1.0 . The arrow shows the correct position of the apex.

the true position, denoted by the arrow, but the diffraction has not been collapsed completely. This indicates that the migration velocity used is too slow.

Figures 52, 53, 54, and 55 show variable velocity migration results with W = -0.43, -0.2, +0.2, and +0.43, respectively. There are not prominent changes among the results. This probably is due to the introduction of a lateral velocity variation, which consequently caused the ineffectiveness of W, the shaping parameter.

Since W has no effects on the results, and the velocity seems too slow, the input velocity was increased to observe changes in the migration results. Figures 56, 57, 58, 59, and 60 correspond to migration results with velocity increases of +10%, +15%, +20%, +25%, and +30%, respectively. As we notice here, the diffraction becomes better focused with the increases of velocity. But also notice that these increases in velocity caused the over-migration of the dipping layer. The optimum migration result is obtained with 25% increase in velocity. Even though the apex is moved very close to the true position, it is a very large adjustment of velocity to have to make.

A detailed study of the diffraction may be done in order to understand the asymmetric diffraction pattern. The plots of T^2 vs. X^2 have been produced for the original synthetic section of Figure 49. Figure 61 is for traces

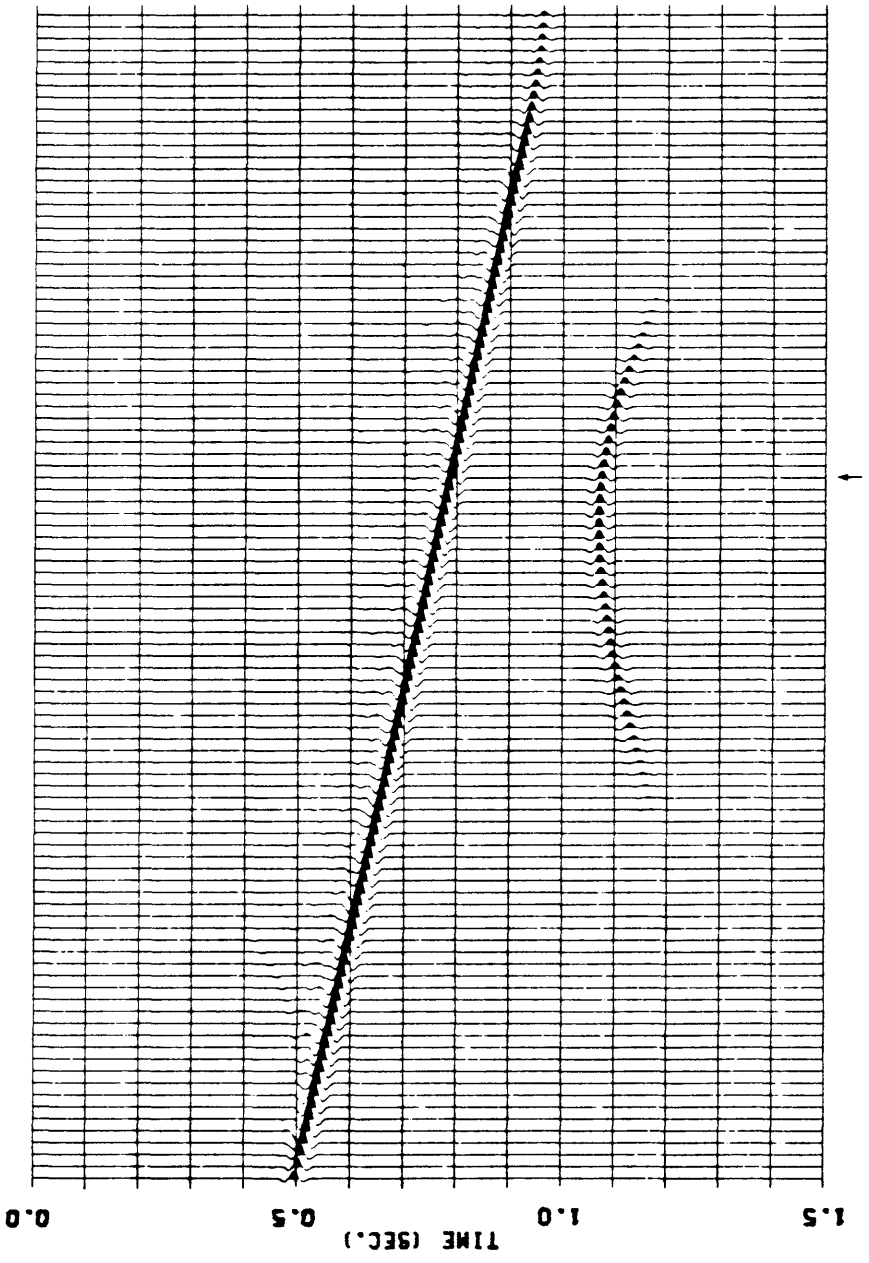

Figure 52: The variable velocity migration of model 4 with W = -0.43. The arrow shows the correct position of the apex.

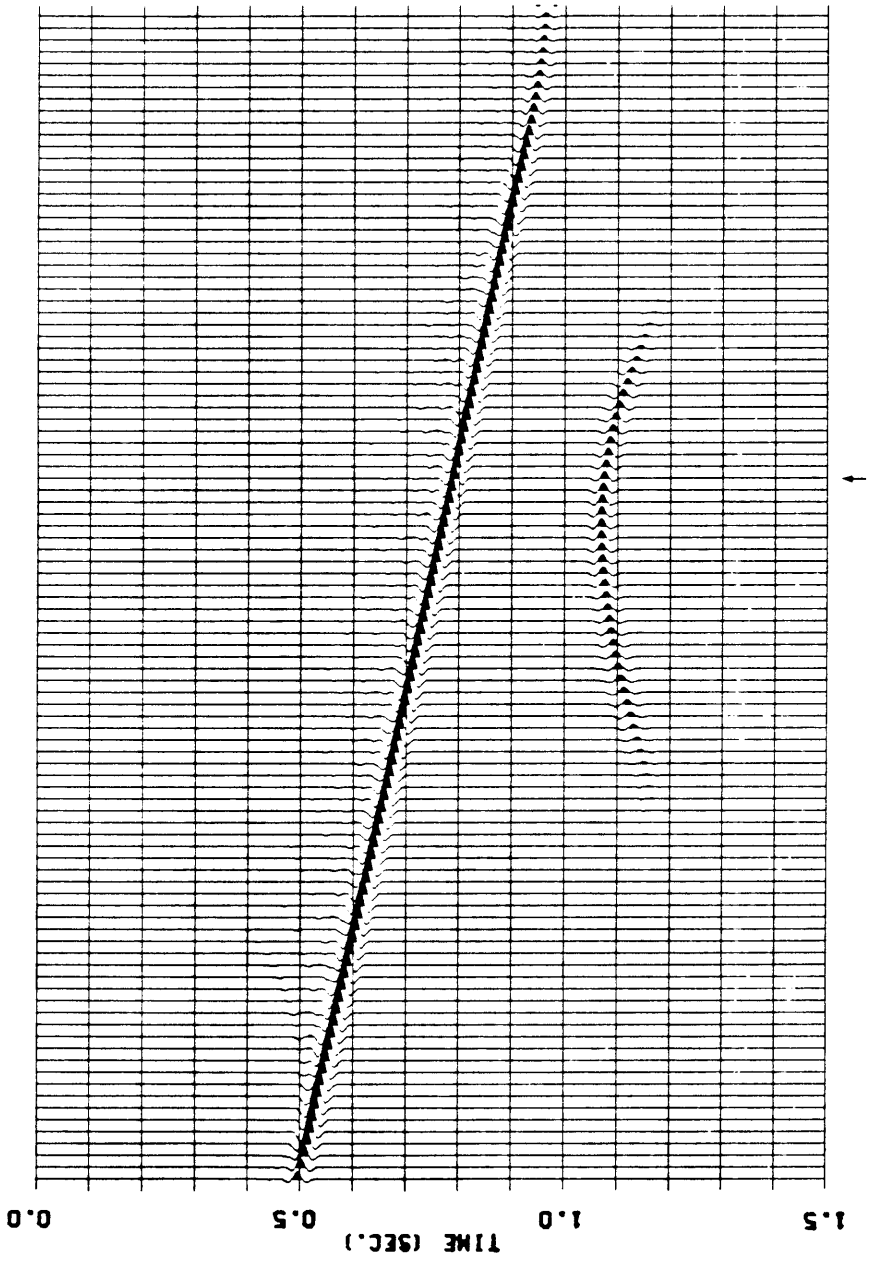

Figure 53: The variable velocity migration of model 4 with W = -0.2, The arrow shows the correct position of the apex.

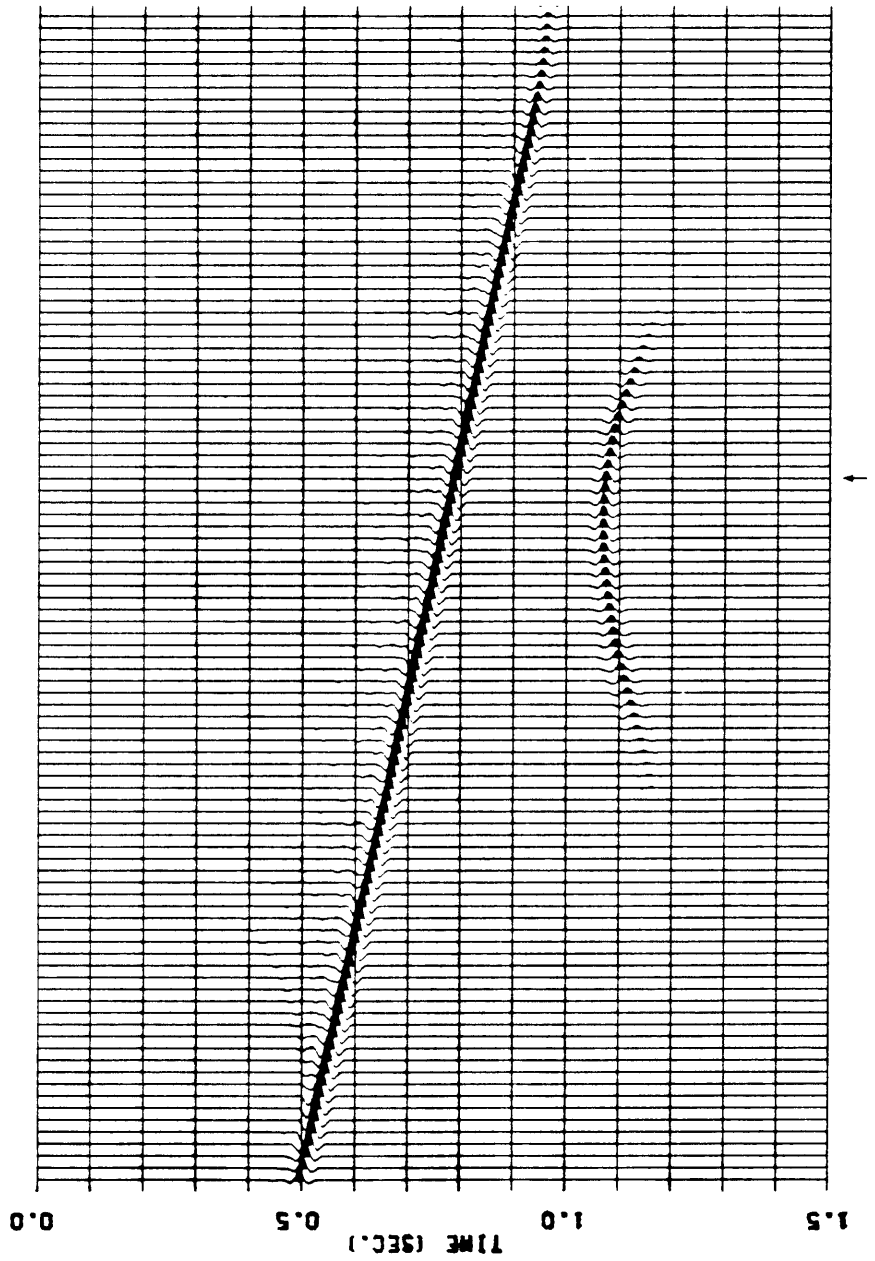

Figure 54: The variable velocity migration of model 4 with W = 0.2 , The aarow shows the correct position of the apex.

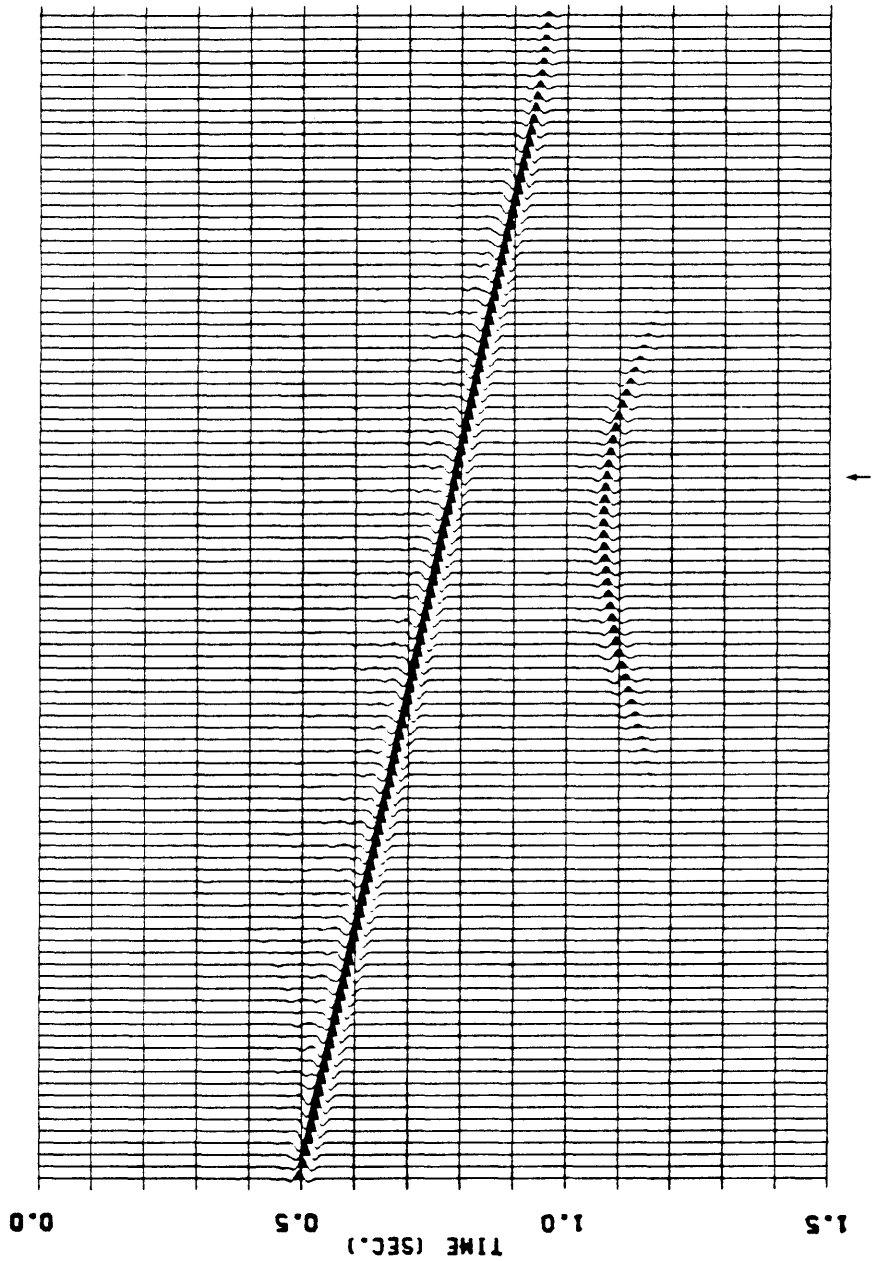

Figure 55: The variable velocity migration of model 4 with W = 0.43 , The arrow shows the correct position of the apex.

TABLE III

Velocity	Interval velocity Layer 1	Layer 2	Corresponding figure
+ 0 %	5000.0	10000.0	Figure 55
+ 10 %	5500.0	11000.0	Figure 56
+ 15 %	5750.0	11500.0	Figure 57
+ 20 %	6000.0	12000.0	Figure 58
+ 25 %	6250.0	12500.0	Figure 59
+ 30 %	6500.0	13000.0	Figure 60

Table III: Table of velocity increase used in the variable velocity migration.

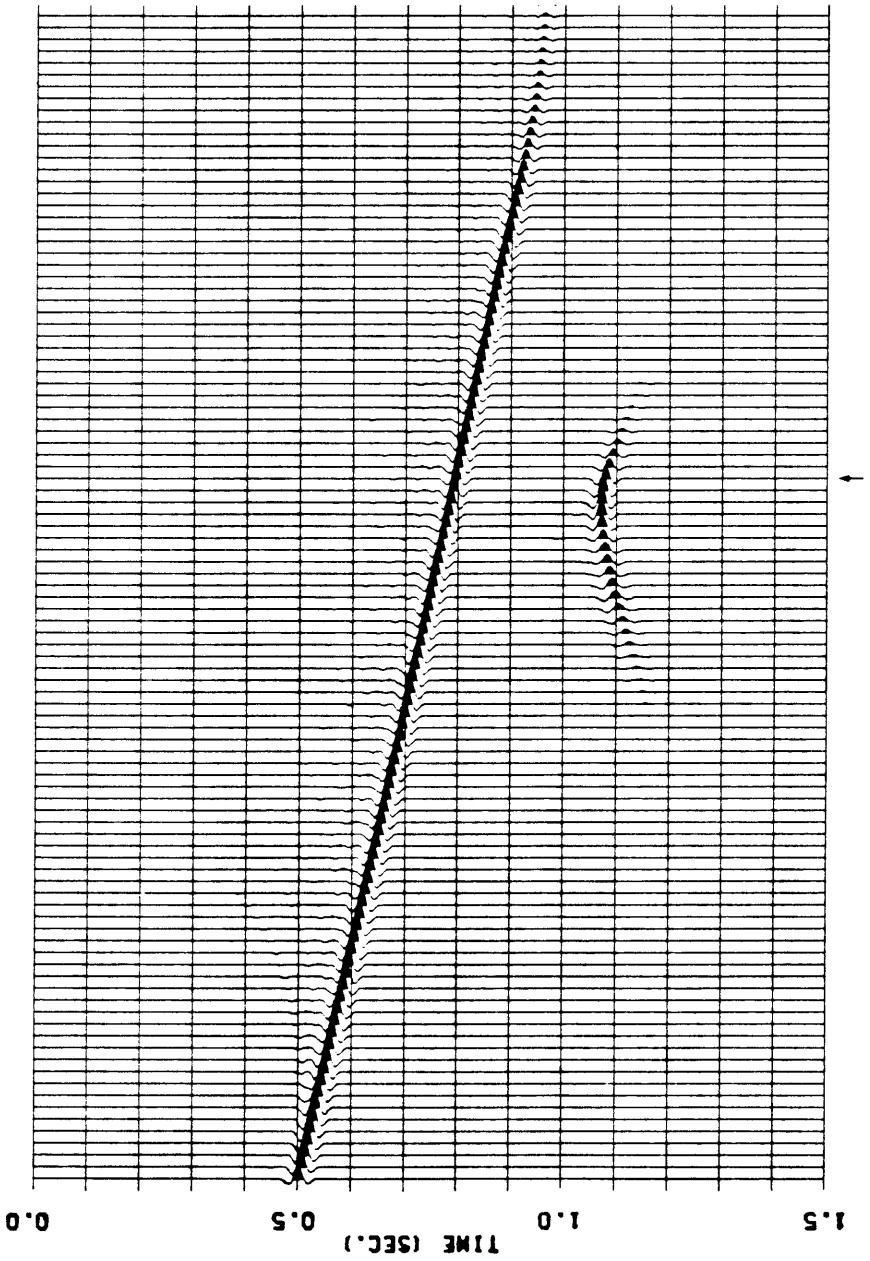

Figure 56: The variable velocity migration of model 4 with W = 0.43 , and velocity increase by 10 %. The arrow shows the correct position of the apex.

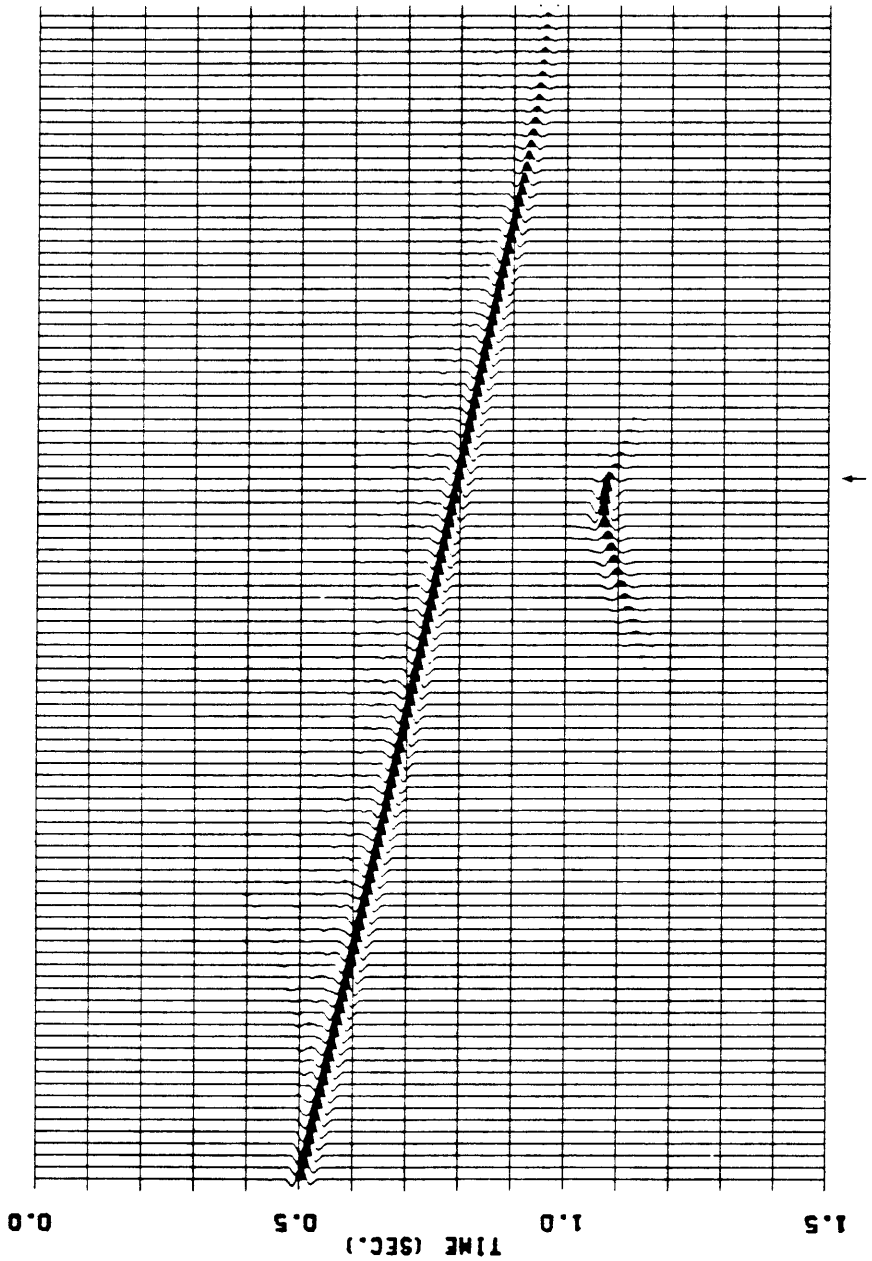

Figure 57: The variable velocity migration of model 4 with W = 0.43 , and velocity increase by 15 %. The arrow shows the correct position of the apex.

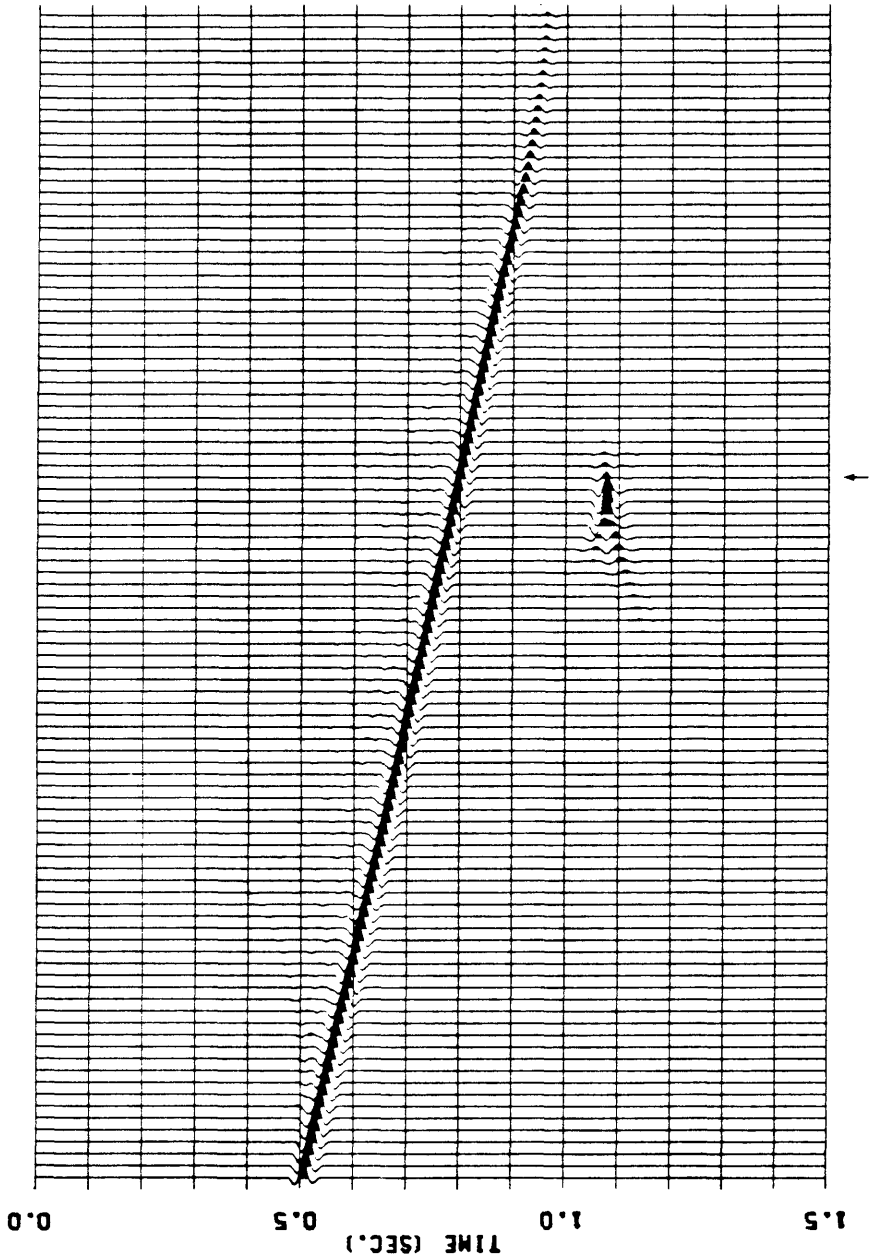

Figure 58: The variable velocity migration of model 4 with W = 0.43 , and velocity increase by 20 % . The arrow shows the correct position of the apex.

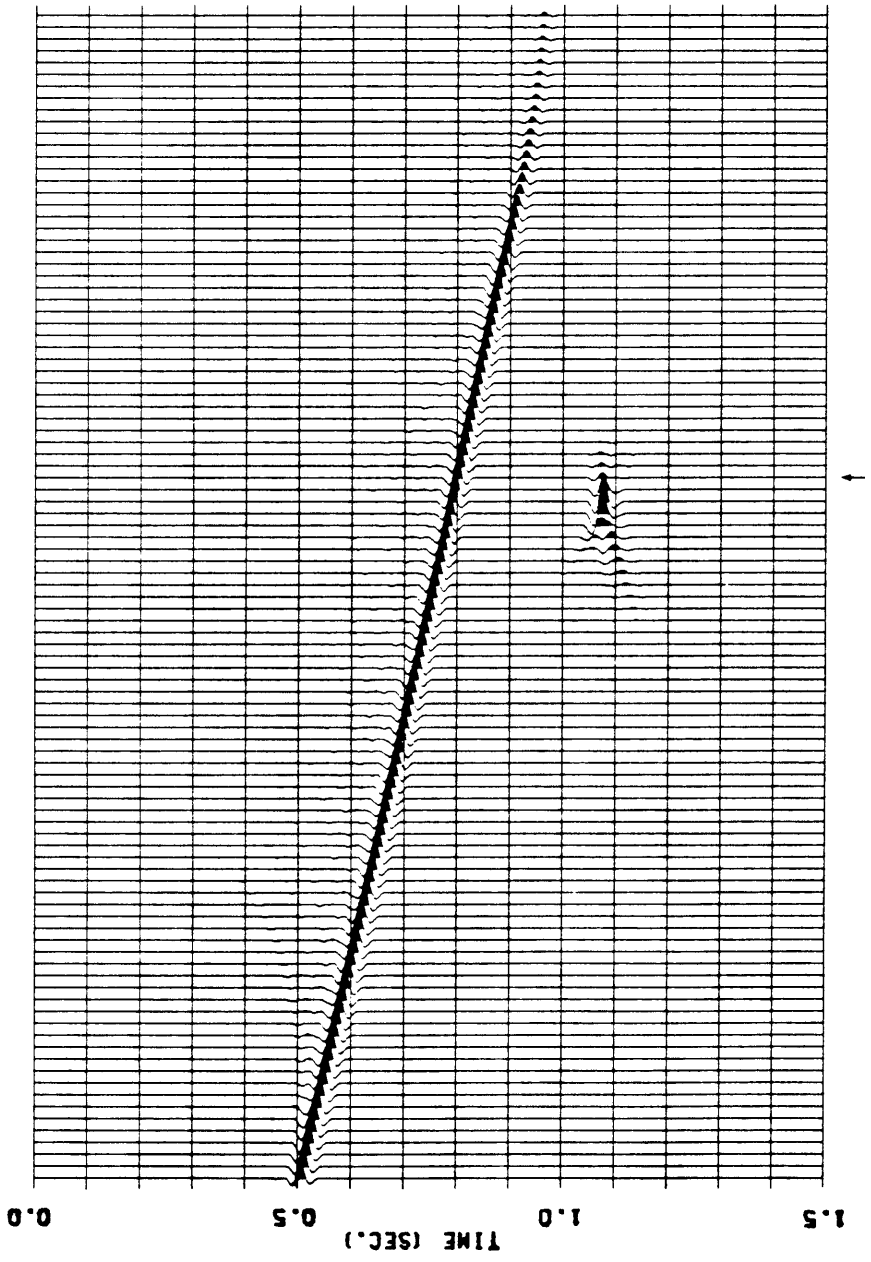

Figure 59: The variable velocity migration of model 4 with W = 0.43 , and velocity increase by 25 % . The arrow shows the correct position of the apex.

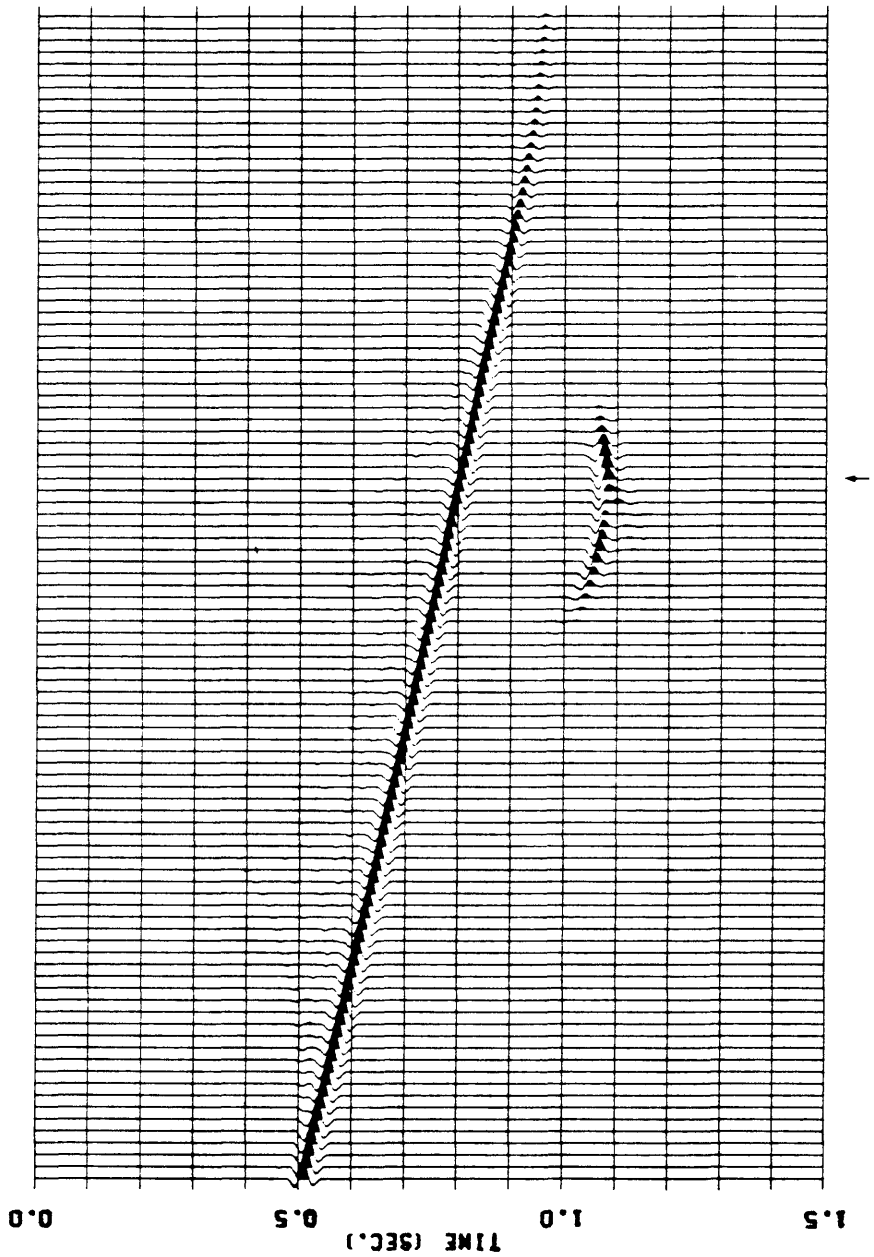

Figure 60: The variable velocity migration of model 4 with W = 0.43 , and velocity increase by 30 % . The arrow shows the correct position of the apex.

1 through 50 and RMS velocity came out to be about 8118 ft/sec., and Figure 62 is for traces 50 through 100 and RMS velocity is about 5641 ft/sec. This is reflected in the migration result in Figure 59; the diffraction on the right side of the apex has been collapsed but on the left side it is still undermigrated.

Now let us turn our attentions to the incremental method. As it was mentioned, a mean average velocity, $\overline{V}(t)$, has to be approximated form $V(x,t)$. For our simple test case, let $\overline{V}(t)$, shown in Figure 63, be a mean average velocity. It can be seen in Figure 63 that we have a transition zone from 0.5 sec. to 1.0 sec. The velocity is increased linearly from 5000 ft/sec. to 10000 ft/sec. in the transition zone. There are no justifications for using this velocity function, it is strictly the test case. The lower plot in Figure 63 shows corresponding depth functions at CDP 1, CDP 100, and $\overline{D}(t)$.

The incremental migration of the section without interpolation before and after migration is shown in Figure 64. The apex stayed at the original recorded position, which is at CDP 50, and there is some unmigrated energy present. This result is reasonable because the algorithm tries to focus the diffracted energy at the apex on the observed time section.

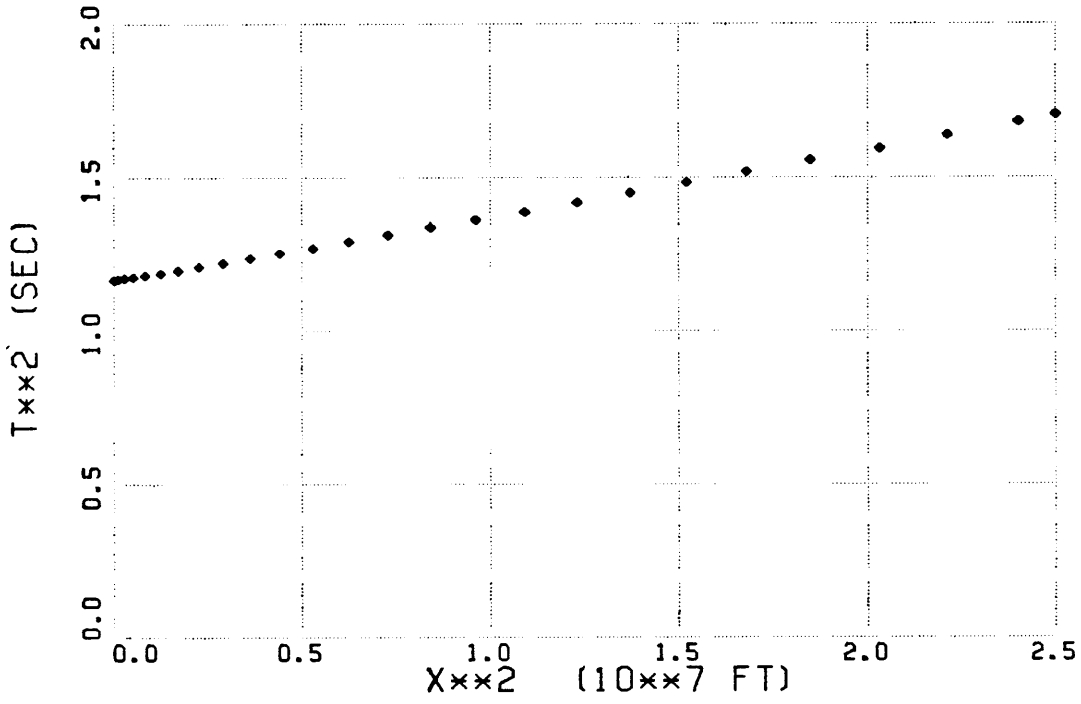

Figure 61: The plot of T^2 vs. X^2 of original synthetic section (Fig. 49) for traces 1 through 50. RMS velocity is about 8118 ft/sec..

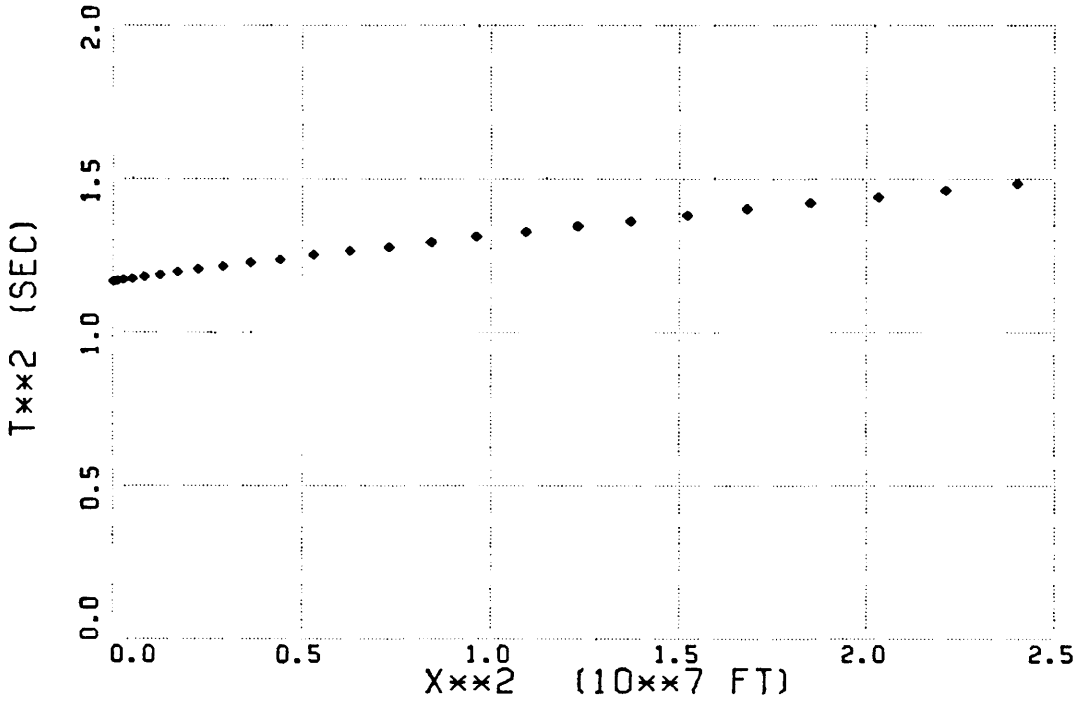

Figure 62: The plot of T^2 vs. X^2 of original synthetic section (Fig. 49) for traces 50 through 100. RMS velocity is about 5641 ft/sec. .

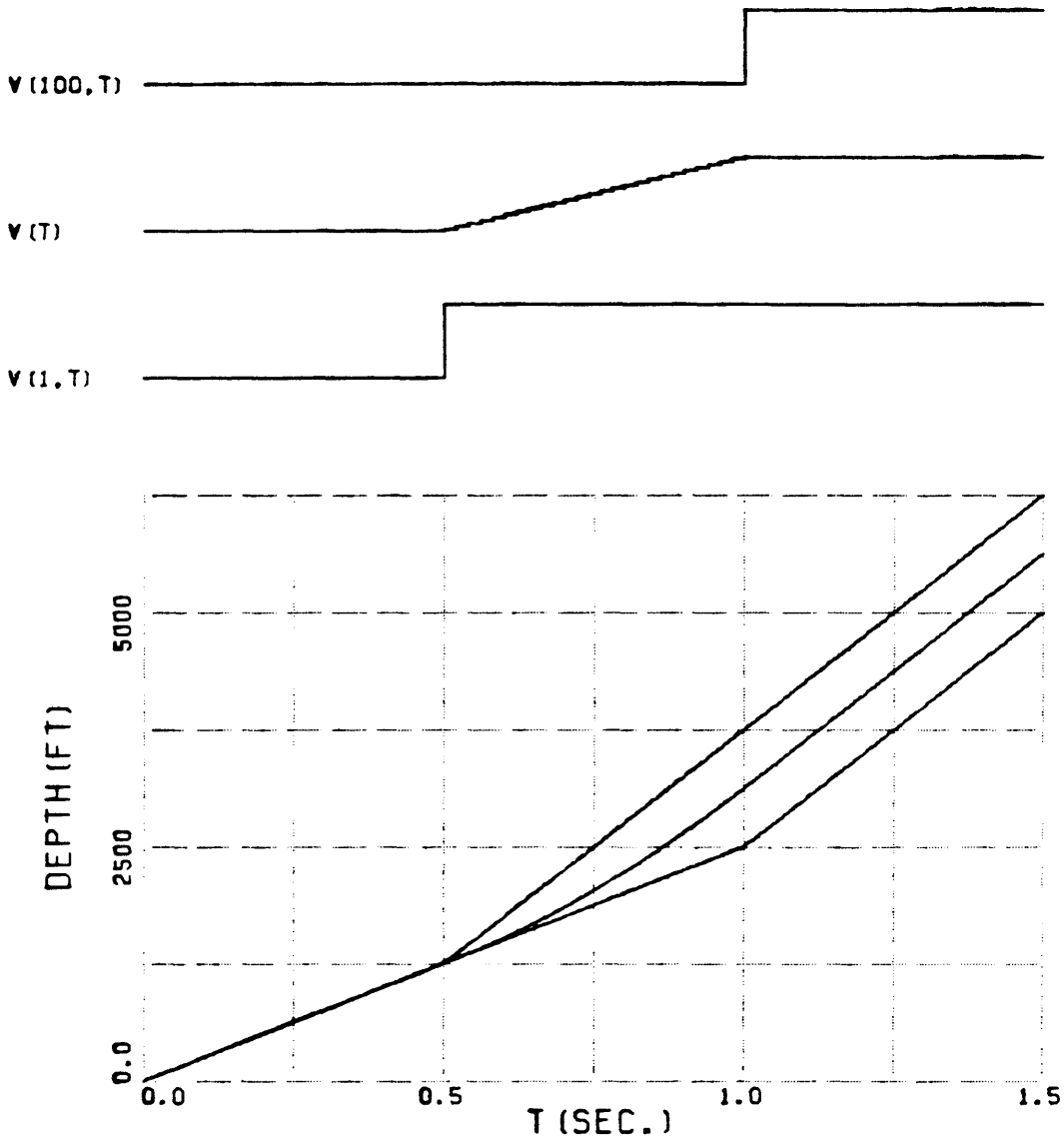

Figure 63: Top: mean velocity used for the incremental migration. Bottom: corresponding depth function.

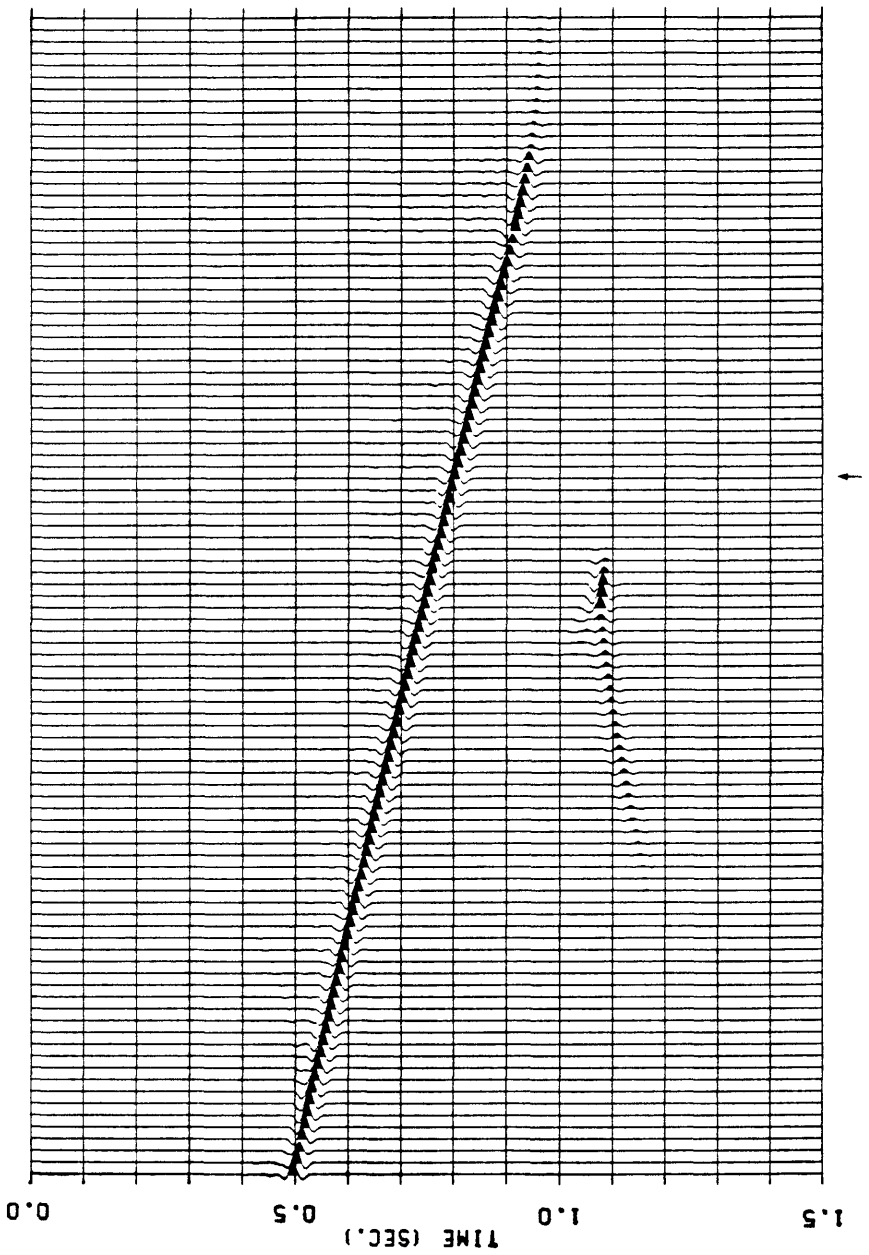

Figure 64: The incremental migration of model 4 without interpolation of trace before and after the migration. The arrow shows the correct position of the apex.

Now we go back to an idea we proposed earlier; the interpolation of traces before and after migration. Figure 65 is an interpolated section before migration. The diffraction looks symmetrical but the apex of the diffraction has been shifted to around CDP 64, about 4 traces too far. This may mean that $\overline{V}(t)$ used here is not the correct mean velocity. But let us proceed to migrate the section. Figure 66 shows the migrated section with this mean velocity. The diffraction has not been collapsed completely and close examination shows that the apex is shifted down about 5 msec. The dipping layer is now shifted a little upward. These errors were caused by the interpolation of traces. Figure 67 shows the migration result with velocity increased by 10%. As we see here, the diffraction is fairly well collapsed but the same phenomenon occurs here; the shift in time of the apex. Also the frequency content of the dipping events become higher. We feel that this probably is caused during the process of interpolation, also.

Here the possibility of extending the incremental method in the laterally inhomogeneous media was investigated in a rather experimental manner. A more analytical study of the process will help in understanding the difficulties encountered here.

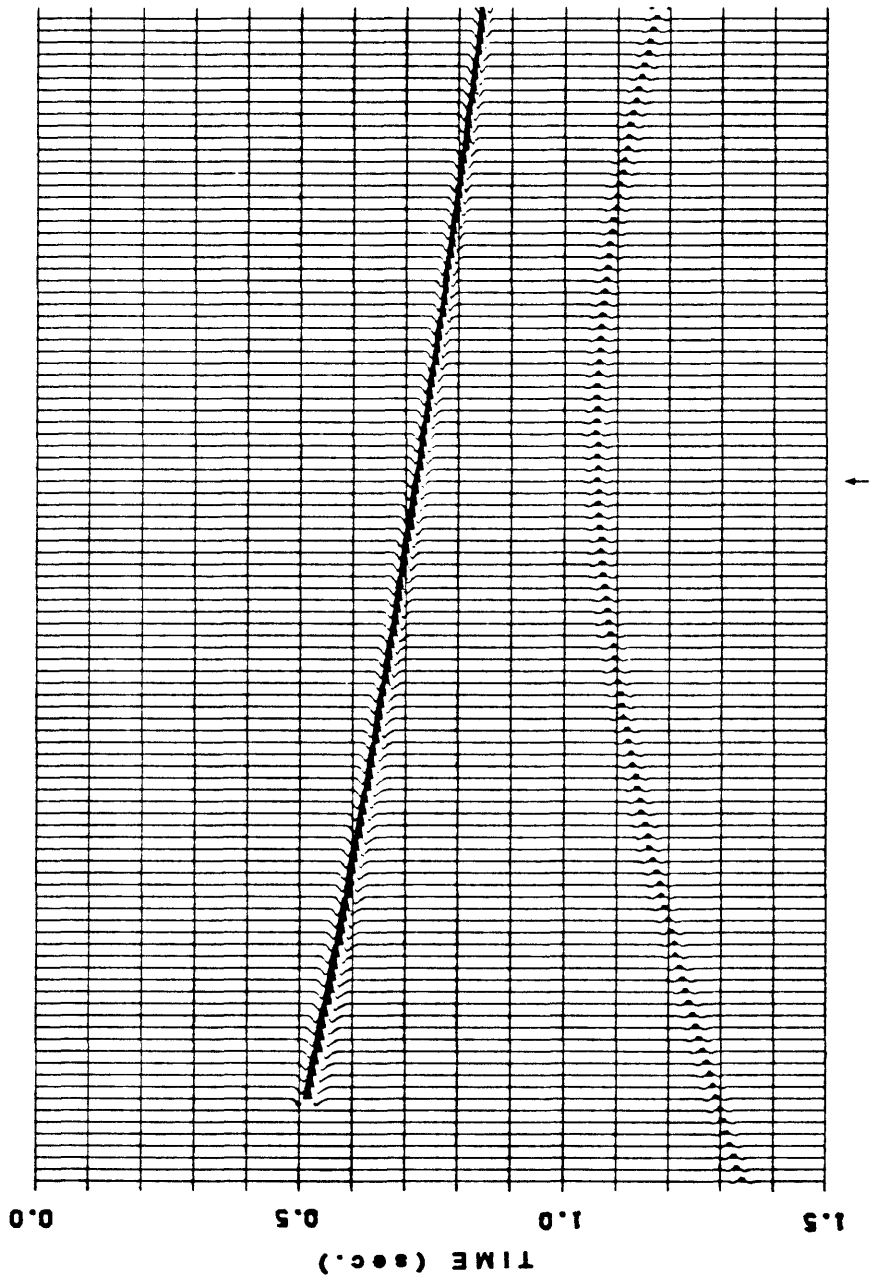

Figure 65: The interpolated section before migration using the mean velocity of figure 61. The arrow shows the correct position of the apex.

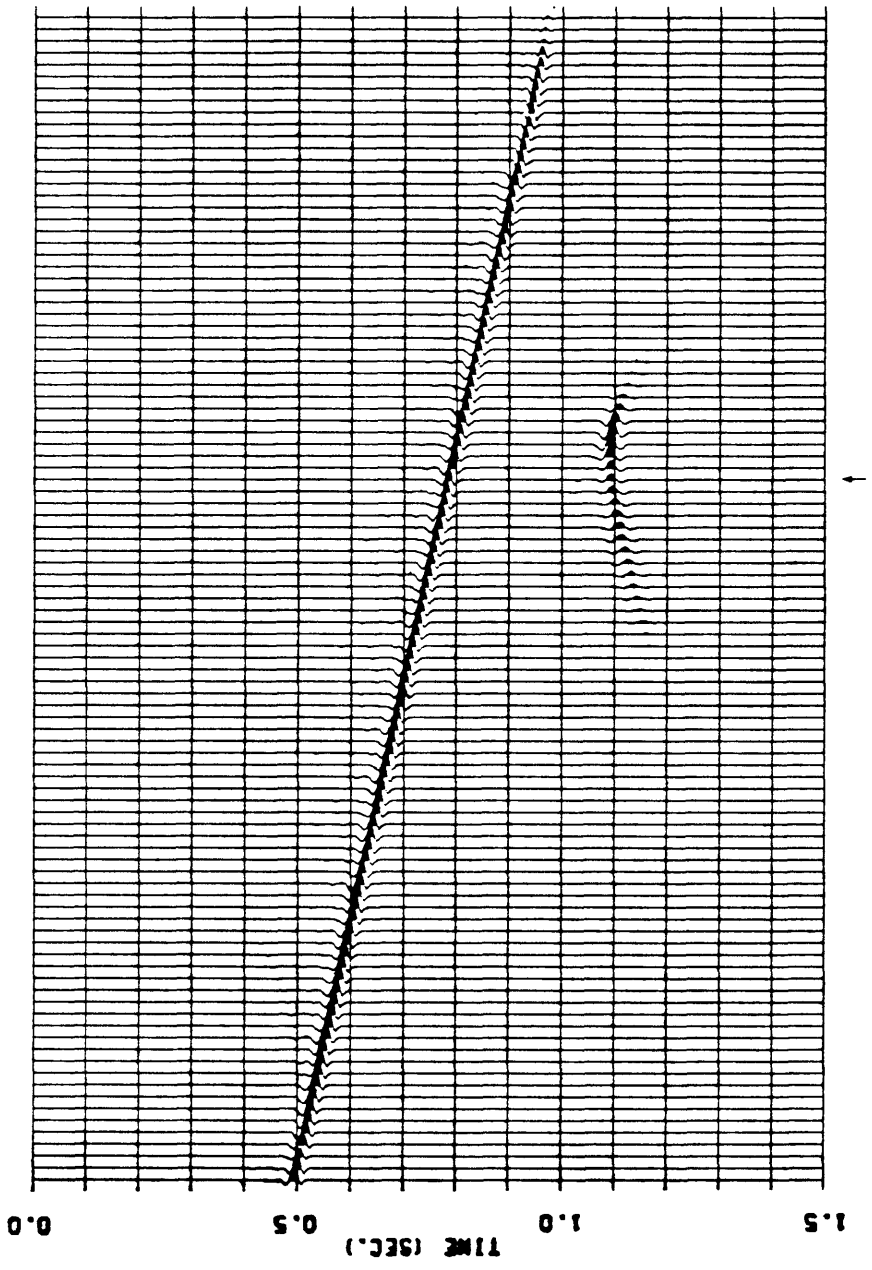

Figure 66: The incremental migration of model 4 using the mean velocity of figure 61. The arrow shows the correct position of the apex.

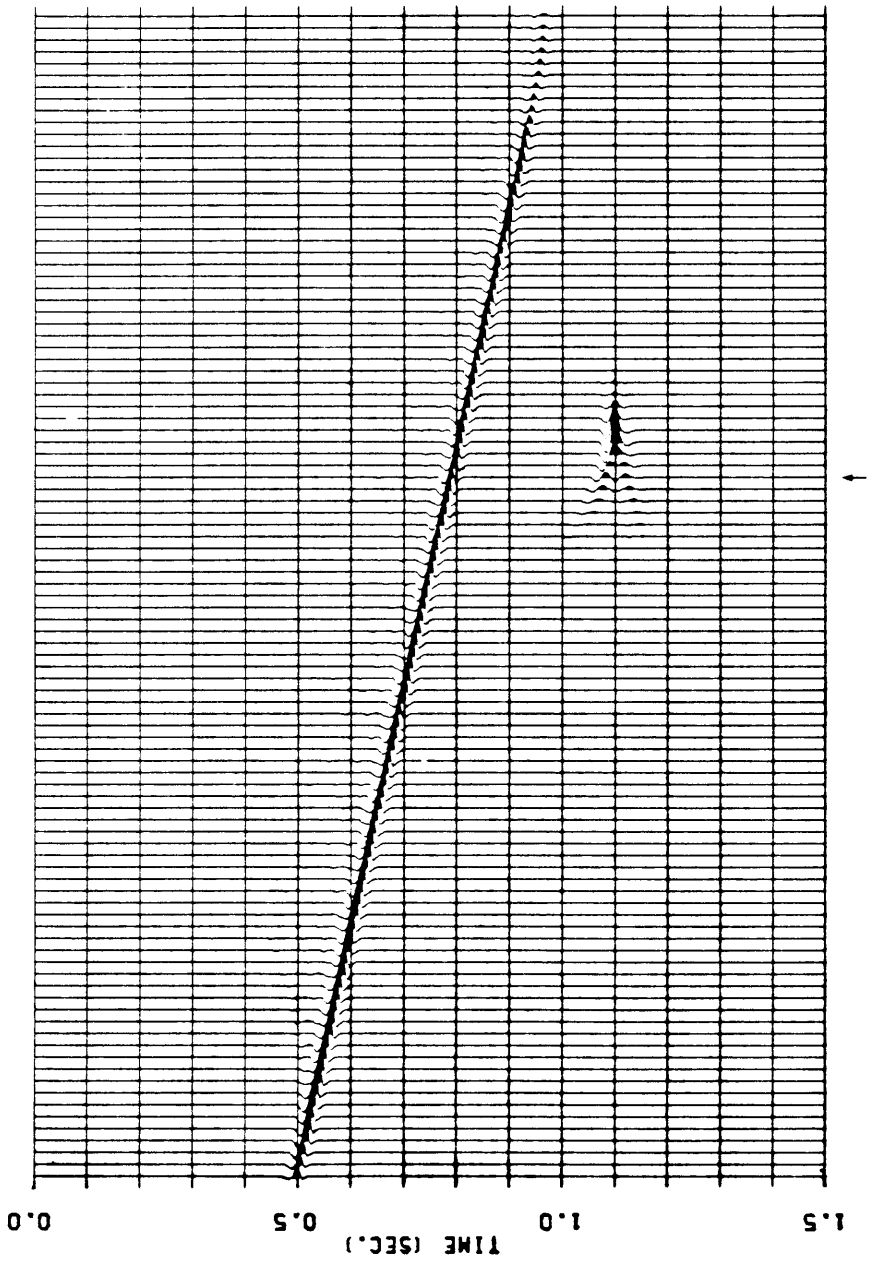

Figure 67: The incremental migration of model 4 using the mean velocity increased by 10 %. The arrow shows the correct position of the apex.

SUMMARY AND CONCLUSIONS

The incremental approach to frequency wavenumber domain migration was developed in this paper. This is a recursive scheme in the F - K domain as opposed to the variable velocity method which is a straight mapping scheme.

Several examples have been presented illustrating the comparisons of the two methods in the presence of vertical velocity gradients. Also, the effort was extended to account for a minor lateral velocity gradient.

From the study carried out here, several conclusions can be drawn,

1) Determination of a correct value of shaping parameter, W, is essential to the success of the variable velocity method.

2) The incremental method gives better results in the presence of vertical velocity gradients.

3) Ineffectiveness of shaping parameter, W, is observed in the presence of lateral velocity gradients. And finally,

4) With more regorous analytical study, the incremental method may be applied in the presence of both lateral and vertical velocity gradients.

BIBLIOGRAPHY

Berkhout, A. J., and Van Wuefften Palthe, D. W., 1979, Migration in terms of spatial deconvultion: Geophysical Prospecting, V. 27, pp. 261-291.

_____ , 1980, Seismic migration: Elsevier Scientific Publishing Company.

Bolondi, G., Rocca, F., and Savelli, S., 1978, A frequency domain approach to two-dimensional migration: Geophysical Prospecting, V. 26, pp. 750-772.

Brigham, E. O., 1974, The fast fourier transform: Prentice-Hall, Inc.

Chun, J. H., and Jacewitz, C. A., 1981, Fundamental of frequency domain migration: Geophysics, V. 46, No. 5, pp. 714-733.

Claerbout, J. F., and Doherty, S. M., 1972, Downward continuation of moveout corrected seismograms: Geophysics, V. 37, No. 5, pp. 741-768.

Diet, J. P., 1979, Determination of the shape parameter W in Stolt's frequency domain migration scheme: reprint of paper S. 19 SEG 1979.

Gazdag, J., 1978, Wave equation migration with the phase shift method: Geophysics, V. 43, No. 7, pp. 1342.

Hatton, L., Larner, K. L., and Gibson, B. S., 1981, Migration of seismic data from inhomogeneous media: Geophysics, V. 46, No. 5, pp. 751-767.

Hood, P., 1978, Finite difference and wave number migration: Geophysical Prospecting, V. 24, pp. 773-789.

Johnson, J. D., 1979, Comparisons of finite difference, f - k, and kirchhoff approaches to wave equation migration:

Larner, K. L., Hatton, L., Gibson, B. S., and Hsu, I-C., 1981, Depth migration of imaged time sections: Geophysics, V. 46, No. 5, pp. 734-750.

Lowenthal, D., Lu, L., Roberson, R., and Sherwood, J., 1976, The wave equation applied to migration: Geophysical Prospecting, V. 24, pp. 380-399.

Neale, G. H., Stone, D. G., and Atherton, J. L., 1978, Migration in the frequency domain: Geophysical Society of Houston (Seminar).

Papoulis, A., 1962, The fourier integral and its applications: McGraw - Hill Book Company, Inc.

Schneider, W. A., 1971, Developments in seismic data processing and analysis (1968-1970): Geophysics, V. 36, No. 6, pp. 1043-1073.

_____ , 1978, Integral formation for migration in two and three dimensions: Geophysics, V. 43, No. 1, pp. 49-76.

Sheriff, R. E., 1978, History and overview of seismic migration methods: Geophysical Society of Houston (Seminar).

Stolt, R. H., 1978, Migration by fourier transform: Geophysics, V. 43, No. 1, pp. 23-48.

APPENDIX A

DESCRIPTION OF THE COMPUTER PROGRAM

COMPUTER PROGRAM

In order to test the incremental scheme, computer program has been developed. First the program was written in a regular Fortran program format to undergo tests. Later it was converted to CGG module format because of the following advantages:

1) easy access to synthetic data generated by the CGG modelling program, and also output to the CGG plotting routine.

2) easy to make use of special routines and MAP 3 array processor calls already existing within the CGG library.

General structure of the program is borrowed from FKMIG (CGG software) and necessary changes were made by the author.

INPUT

The module, YOMIG, takes the same format as other modules in the CGG package.

CONTROL CARD

Column	Field	Content
1	Identification	*
3 - 7	Name	YOMIG
9 - 10	Primary option	Not used
12 - 13	Secondary option	Not used
15 - 16	Input Buffer	Disk Buffer
23 - 24	Output Buffer	Disk Buffer, must be different from input buffer
31 - 80	DATA (all the numbers are integer)	
NTa	a = number of traces to process, maximum 350.	
SIi	i = sampling interval in milliseconds.	
Ec	c = distance between CDP.	
RLd	d = length of the trace to process in milliseconds.	
RLMe	d = maximum time to migrate. (default, RLM=RL)	

ECSf f = limitation of ECs requirements; when necessary, mentioning ECSf in the data field will reduce the requirements to f words. (default, f = 60000)

MNPADg g = number of traces to be padded in x-direction before FFT. NX + MNPAD had to be less than 512.

FMAXh h = maximum frequency to migrate.

LEVCj j = number of bit shift to apply, if necessary, for output.

NOMAP YOMIG normally uses the MAP 3. Coding NOMAP forces the program to execute the calculations outside the MAP 3.

ZYKMV YOMIG normally goes through the execution without any pause. Coding ZYKMV forces the program to stop after the overlay ZYKMV and print out the necessary parameters for migration.

MV=TkVIl, TmVIn, ... , VFo

input format of migration velocity.

k = time in miliseconds.

l = interval velocity up to time k.

and same format for m, n.

o = final interval velocity

total of 30 pairs plus 1 final velocity can be input.

FILES

1) IHEDR

- It is created by the overlay ZYKRT, initialized with zeros.
- It stores input traces, managed by FICH pachage.
- It consists of one group and as many sub-groups as input traces.

2) MUXR

- It is created by the overlay ZYFKA.
- It stores all the (F - K) information with different structures according to the computation phase.
- It is managed by XED package for fast DISK - ECS direct transfer.

OVERLAY NAME		Routine used in addition to common SOS routine	
YOMIG	YOMIG1	PCLRW	
ZYINI	ZYINI1		
ZYKRT	ZYKRT1		
ZYKMV	ZYKMV1	ZYKMV2	
ZYFKA	ZKMIA21	FFTINF NPFMAX BMUX1 RFFT	PCLRW SINE,PCLRW,FOUR1, FFOUR1
ZYFKB	ZKMIB21	KTRANS1	SINE,PCLRW,FOUR1, CALSPEC,FREQSUM, FFOUR1
ZYKFC	ZKMIC21	RIFFT BDMUX	SINE,FOUR1,FFOUR1 LOWER

```
/JOB
YO,T1500.
USER,M749019,KATA.
CHARGE,I,WA*SCHNEIDER.
CPT(1234567,1234,A,1234A,1234)
ATTACH,SOSIN=MODEL3.
PURGE,MODEL3M/NA.
DEFINE,SOS001=MODEL3M.
ECSFILE,TABSOS.
SOS.
GOTO,1.
EXIT.
1,DAYFILE.
REPLACE,OUTPUT=DAYYOM.
/EOR
**  EXAMPLE-YOMIG
**34567890123456789012345678901234567890123456789012345678901234567890
* DEFIN                  1     1000 EA,DA(4*24),DB(4*24)
* LIBRI TR                          B1009,F1=(R1T1-R1T71,I1,G1)
* LIBRI BD                          UNITO1=B9999(RW)
* LIBRI HB  1                     2 YOMIG
                                   3 MODEL 3
* BOUCL                  1
* INPTR                    EA       RL500
* TRANS          EA        DA
* FINBO
* BOUCL                  2
* YOMIG          DA        DB       NT71,SI1,E50,RL500,FMAX100,
                                    ECS60000,MNPAD40,LEVC3,
                                    MV=T100VI5000,T200VI6000,
                                    T300VI7000,T400VI8000,VF9000
* FINBO
* BOUCL                  3
* TRANS          DB        EA
* RAGOU          EA                ECH750,G35,PAS50,LS0,GD,NEX1,
                                    LHB1,CT100,AG
* OUTBD          EA          500   UNITO1
* FINBO
* PROCS                            71(B1)+B2+71(B3)
/EOR
/EOF
```

OVERLAY YOMIG

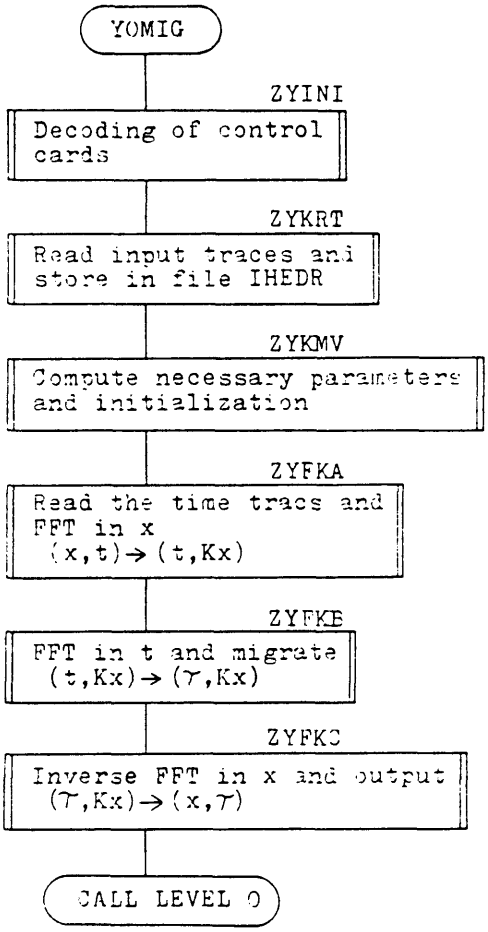

Figure A - 1 : Flow diagram of overlay YOMIG(1,0).

OVERLAY ZYFKA

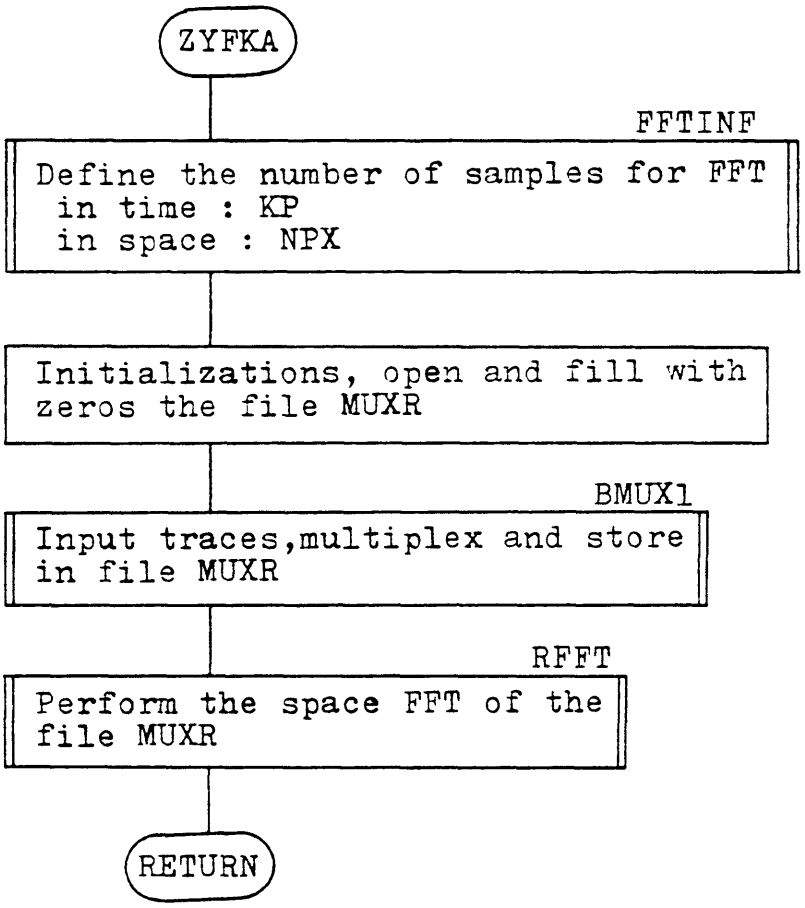

Figure A - 2 : Flow diagram of overlay ZYFKA(1,1).

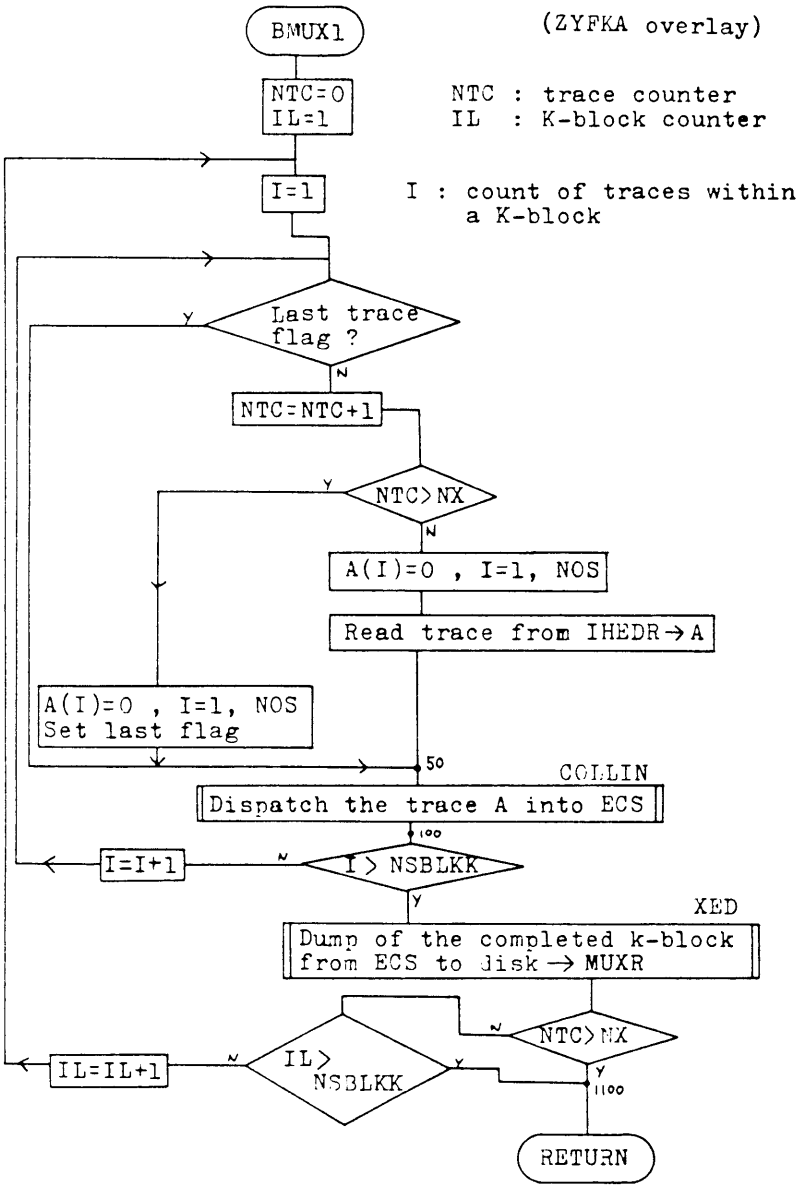

Figure A - 3 : Flow diagram of subroutine BMUX1 in overlay ZYFKA.

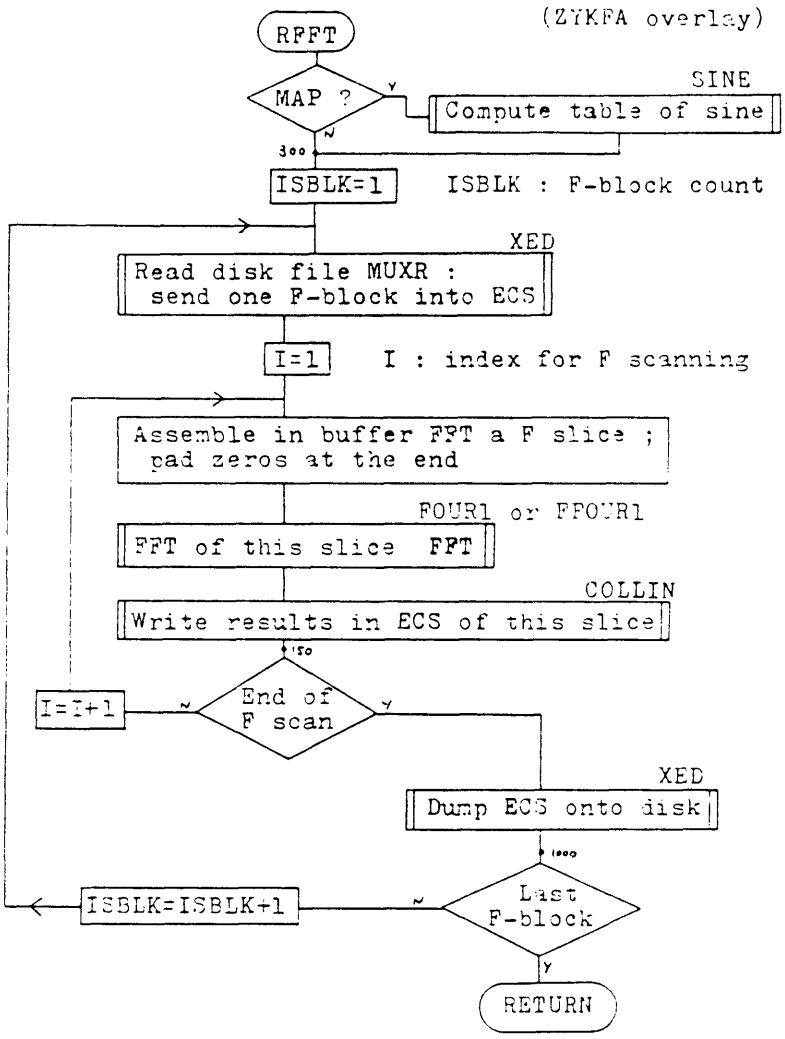

Figure A - 4 : Flow diagram of subroutine RFFT in overlay ZYFKA.

OVERLAY ZYFKB

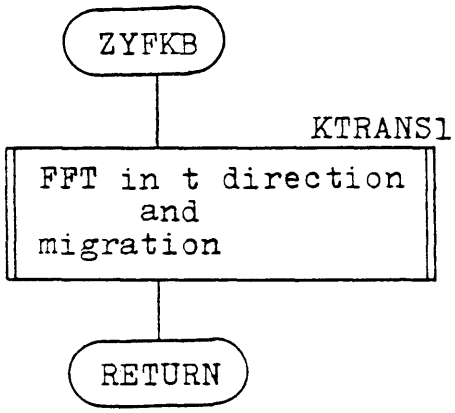

Figure A - 5 : Flow diagram of overlay ZYFKB(1,1).

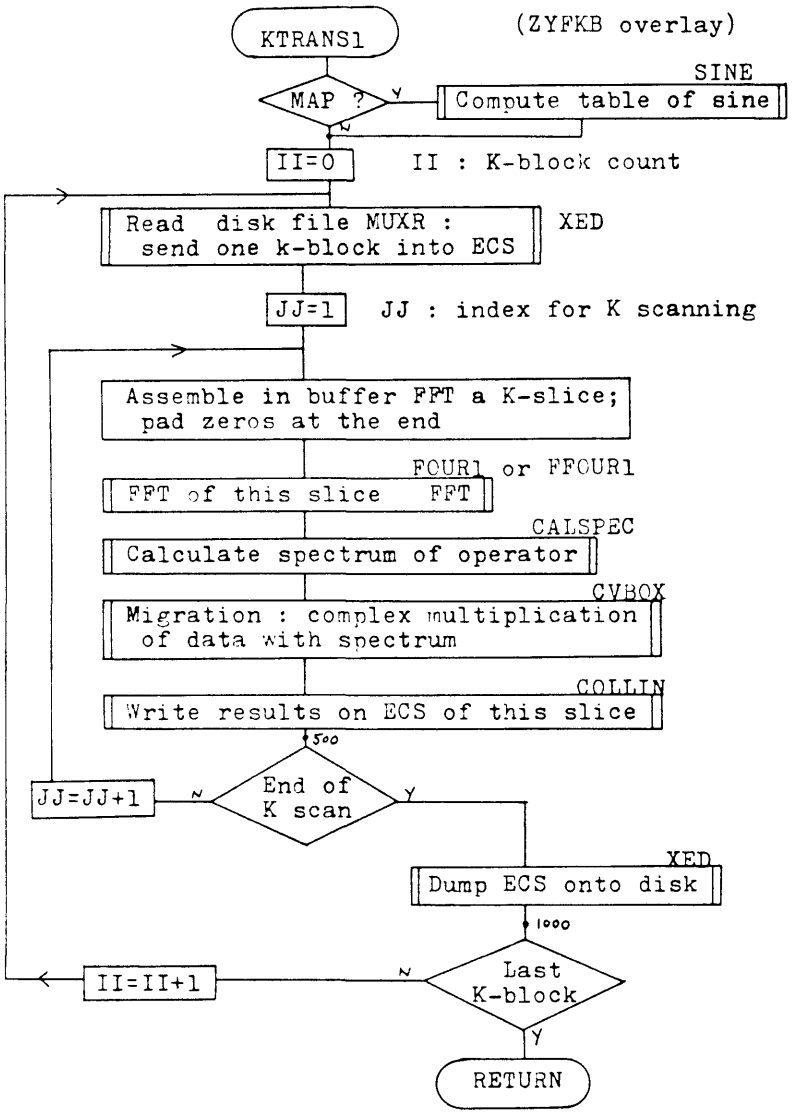

Figure A - 6 : Flow diagram of subroutine KTRANS1 in overlay ZYFKB.

OVERLAY ZYFKC

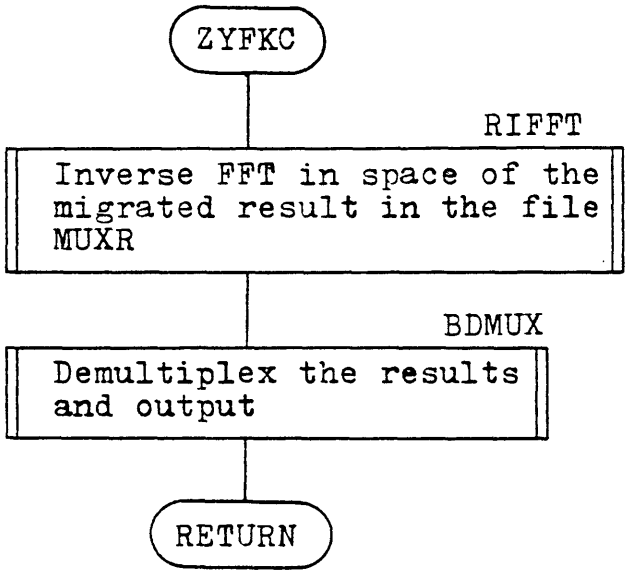

Figure A - 7 : Flow diagram of overlay ZYFKC(1,1).

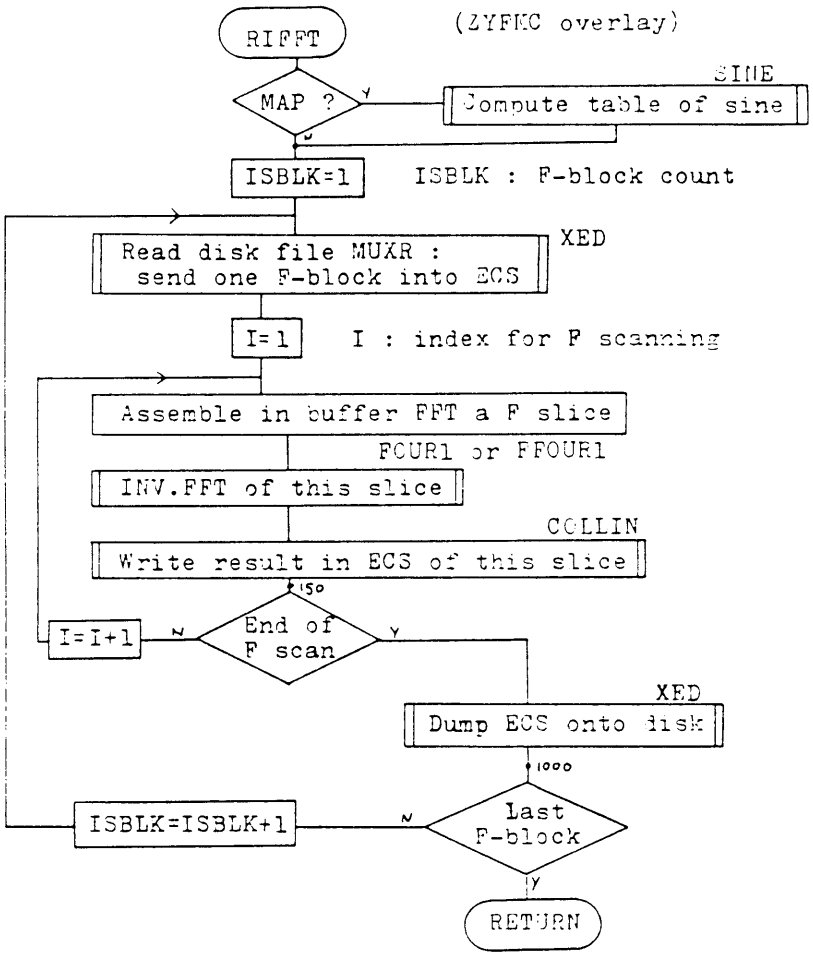

Figure A - 8 : Flow diagram of subroutine RIFFT in overlay ZYFKC.

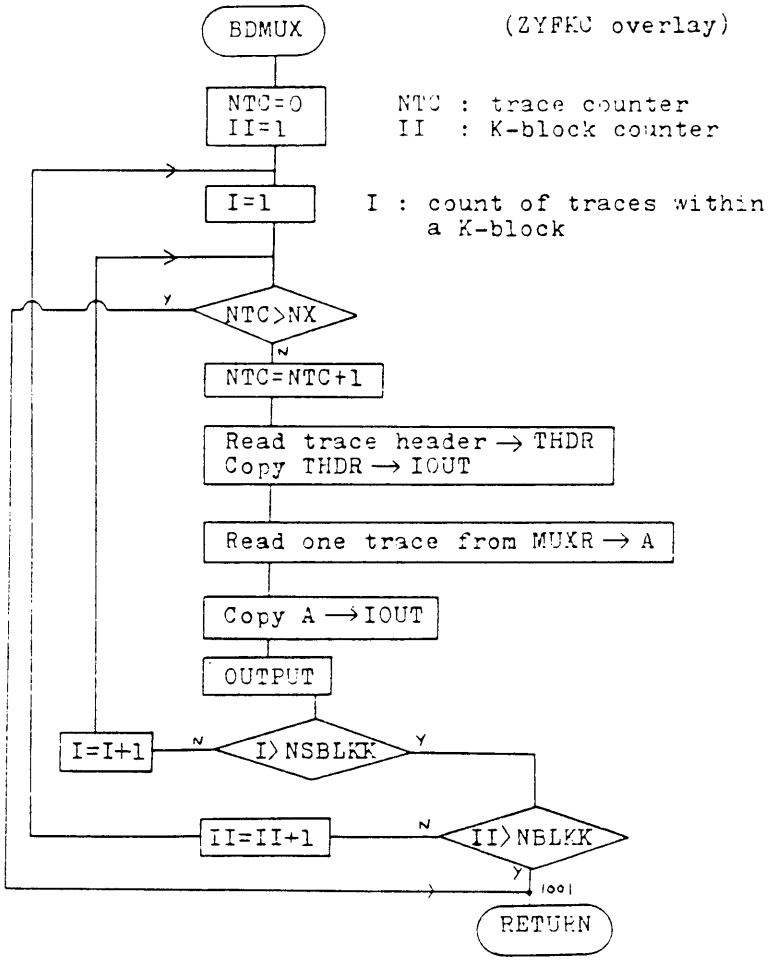

Figure A - 9 : Flow diagram of subroutine BDMUX in overlay ZYFKC.

APPENDIX B

LIST OF THE COMPUTER PROGRAM

```
*DECK YOMIG
*..                    MODUL   YOMIG,ECS,1,8
*..          YOMIG     BTYPE   XI,0,X0,0,SCM,-4
*..                    BTYPE   SCM,-2,CM,151,CM,141
*..                    BTYPE   CECS,2040,SCECS,-4,SCECS,100
      OVERLAY(YOMIG,1,0)
      PROGRAM YOMIG
      EXTERNAL YOMIG1
C
C
C   =  INCREMENTAL F-K MIGRATION PROGRAM  =
C
C                 WRITTEN   BY
C
C                  YO KATAHIRA
C
C                     1982
C
C
C    MAIN OVERLAY YOMIG (1,0) LEVEL
C    INPUT FORMAT IS SAME AS OTHER CGG MODULE.
C    EXPLANATION OF COMMON /COMM/ IS GIVEN IN OVERLAY
C    ZYKMV.
C
C    THIS OVERLAY CALLS ALL OTHER SUBOVERLAYS (1,1) LEVEL
C
C     ZYINI - DECODING OF CONTROL CARD.
C     ZYKRT - READ INPUT TRACES ON FILE IHEDR.
C     ZYKMV - INITIALIZATION AND CALCULATION
C             OF NECESSARY PARAMETERS.
C     ZYFKA - MULTIPLEX AND FFT IN X - DIRECTION.
C     ZYFKB - FFT IN T - DIRECTION AND MIGRATE.
C     ZYFKC - DEMULTIPLEX AND OUTPUT.
C
C
      CALL START(YOMIG1)
      CALL LEVELO
      CALL START 1 (YOMIG1)
      END
C
```

```
C
      SUBROUTINE YOMIG1(NCO,NDA,LDA,NIN,LIN,NOU,LOU,
     *A,LAA,IOUT,LOUT,ICM,LCM,ICMN,LCMN,IECS,LECS,
     *KSCS,LSCS,KSS,LSS)
C
C
      DIMENSION NCO(1),NDA(1),NOU(1),A(1),IOUT(1),ICM(1)
      DIMENSION ICMN(1),NIN(1)
C
      COMMON/COMM/ISZCOM,NOS,NSI,NX,NGI,IHEDR,MUXR,FMAX,
     *NTN,NSBLKF,NBLKF,NSBLKK,NBLKK,NPX,KP,MNPAD,ISTFLAG,
     *NOTIME,NMTIME,NOSMIG,THICK(31),DELZ(31),LEVC,
     *DTIME(31),NITER(31),VI(31),NOL
C
C ------------------------------------------------
C

C
      IMV=20C0B
C
C
      DO 1000 I=1,LCM
1000  ICM(I)=0
C
      DO 1001 I=1,LCMN
1001  ICMN(I)=0
C
      DO 1002 I=1,20C0
1002  A(I)=0
      CALL WRITEC(A,IECS,2000)
C
C
C  INITIALIZE AND DECORDING OF CARD
C
      CALL CHARGOV(5LZYINI,IMV,0)
C
C  READ TRACES
C
      CALL CHARGOV(5LZYKRT,IMV,0)
C
      CALL CHARGOV(5LZYKMV,IMV,0)
      IF(ISTFLAG.EQ.1) GO TO 2000
C
      CALL CHARGOV(5LZYFKA,IMV,0)
C
      CALL CHARGOV(5LZYFKB,IMV,0)
C
      CALL CHARGOV(5LZYFKC,IMV,0)
C
C
2000  RETURN
      END
```

```
C
C   ***********************************************
C
C
      SUBROUTINE PCLR(C,IC,N,FLAG)
C
C    ZERO AN ARRAY
C
C        C - INPUT ARRAY TO BE ZEROED.
C       IC - INCREMENT OF ARRAY * 2
C        N - NUMBER OF POINTS IN C
C     FLAG - UNUSED.
C
      DIMENSION C(N)
150   CONTINUE
      J=IC/2
      JJ=1
      DO 120 L=1,N
      C(JJ)=0.
      JJ=JJ+J
120   CONTINUE
      RETURN
      ENTRY PCLRW
      GO TO 150
      ENTRY PWAIT
      RETURN
      END
C
C
```

```
      SUBROUTINE FFTINF(NSAMPS,NSR,NPAIRS,DF)
C
C    OBTAIN INFORMATION CONCERNING FFT
C
C   INPUT
C     NSAMPS - NUMBER OF POINTS TO BE FFT.
C              (DOES NOT HAS TO BE MULTIPLE OF 2,
C               ROUTINE AUTOMATICALLY FINDS NAPIRS WHICH
C               IS MULTIPLE OF 2)
C         NSR - SAMPLING INTERVAL IN MSEC.
C   OUTPUT
C     NPAIRS - NUMBER OF PAIRS FOR FFT/2
C         DF - INCREMENT IN FREQUENCY DOMAIN.
C
C
      N=NSAMPS
      DO 1 I=1,15
      N=N/2
      IF(N.LE.1) GO TO 2
1     CONTINUE
2     NP=2**I
      IF(NP.LT.NSAMPS) I=I+1
      NPAIRS=(2**I)/2
C
      P=FLOAT(NPAIRS)
      SR=FLOAT(NSR)/1000.
      DF=0.5/(SR*P)
      RETURN
      END
```

```
C
C
      SUBROUTINE LOWER(KSC,NOS,LEVC)
C
C   BIT SHIFT
C    SET MAX. BETWEEN 2**24 AND 2**23
C    KSC CONTAINS NOS + 64 HEADER WORD FOR EACH TRACE.
C    ROUTINE PUTS BIT SHIFT IN 13TH WORD ,AND
C    MAXIMUM NUMBER IN 14TH WORD OF KSC.
C
C
C   KSC - INPUT AND OUTPUT ARRAY.
C   NOS - NUMBER OF DATA POINTS.
C   LEVC - NUMBER OF BIT SHIFT TO BE APPLIED.
C
C
      DIMENSION KSC(1)
      LT=NOS+64
      BIG=FLOAT(KSC(14))
      RMAX=2.**24
      RMIN=2.**23
      LL=0
1     CONTINUE
      IF(BIG.LT.RMAX) GO TO 3
      LL=LL+1
      BIG=BIG/2.
      GO TO 1
3     CONTINUE
      IF(BIG.GT.RMIN) GO TO 5
      LL=LL-1
      BIG=BIG*2.
      GO TO 3
5     CONTINUE
      KSC(13)=KSC(13)+LL+LEVC
      KSC(14)=INT(FLOAT(KSC(14))/(2.**LL))
      DO 10 KK=65,LT
      KSC(KK)=INT(FLOAT(KSC(KK))/(2.**LL))
10    CONTINUE
      RETURN
      END
```

```
C
CC
C
      OVERLAY(ZYINI,1,1,C2000)
      PROGRAM ZYINI
      EXTERNAL ZYINI1
C
CC   OVERLAY ZYINI
C
C      DECODING OF CONTROL CARD.
C
C
      CALL START (ZYINI1)
      CALL LEVEL1
      END
```

```
C
      SUBROUTINE ZYINI1(NCO,NDA,LDA,NIN,LIN,NOU,LOU,A,LAA,
     *IOUT,LOUT,ICM,LCM,ICMN,LCMN,IECS,LECS,KSCS,LSCS,
     *KSS,LSS)
C
C
      DIMENSION NCO(1),NDA(1),NOU(1),A(1),IOUT(1),ICM(1)
      DIMENSION NIN(1),ICMN(1)
C
      DIMENSION TABLE(17)
C
      EQUIVALENCE(NUM,FNUM)
C
      COMMON/COMM/ISZCOM,NOS,NSI,NX,NGI,IHEDR,MUXR,FMAX,
     *NTN,NSBLKF,NBLKF,NSBLKK,NBLKK,NPX,KP,MNPAD,ISTFLAG,
     *NOTIME,NMTIME,NOSMIG,THICK(31),DELZ(31),LEVC,
     *DTIME(31),NITER(31),VI(31),NOL
C
      DATA(TABLE(I),I=1,17)/3RNT1,3RSI1,2RE1,3RRL1,
     *4RRLM1,4RECS1,2RB1,5RMV=T1,3RVI1,1R,,2RT1,3RVF1,
     *6RMNPAD1,5RFMAX1,5RLEVC1,5RZYKMV,5RNOMAP/
C
C
      ICM(15)=KSCS+100
      ICM(1)=118000
      ICM(32)=0
      KODE=0
      NBS=17
      NC=31
      IV=50
      ITM=60
      INPS=0
      ISTFLAG=0
C
C   DECODE CONTROL CARD
C
2000  CALL CARDIN(TABLE,NBS,NDA,LDA,NC,NUM,KODE)
      GO TO (1,2,3,4,5,6,7,8,9,2000,11,12,13,14,
     *15,16,17,2001,9888) ,KODE
C
C   NT
C
1     NX=NUM
      ICM(13)=NUM
      GO TO 2000
C
C   SI
C
2     NSI=NUM
      ICM(2)=NUM
      GO TO 2000
C
C   E
C
3     NGI=NUM*100
      ICM(8)=NUM
      GO TO 2000
C
C   RL
C
4     NOTIME=NUM
      GO TO 2000
C
```

```
C   RLM
C
5       NMTIME=NUM
        GO TO 2000
C
C   ECS
C
6       ICM(1)=NUM
        IF(ICM(1).LT.45000) ICM(1)=45000
        GO TO 2000
C
C   POST LOOP B
C
7       ICM(50)=NUM
        GO TO 2000
C
C   MV=T
C
8       INPS=INPS+1
        ITM=ITM+1
        ICM(48)=INPS
        ICM(ITM)=NUM
        KT=1
        GO TO 2000
C
C   VI
C
9       IV=IV+1
        ICM(IV)=NUM*100
        KT=KT-1
        GO TO 2000
C
C   T
C
11      INPS=INPS+1
        IF(KT.EQ.1) GO TO 1007
        KT=1
        IF(INPS.GT.30) GO TO 1008
        ITM=ITM+1
        ICM(48)=INPS
        ICM(ITM)=NUM
        IF(ICM(ITM).LE.ICM(ITM-1)) GO TO 1009
        GO TO 2000
        ICM(ITM)=NUM
        GO TO 2000
C
C   VF
C
12      ICM(111)=NUM*100
        GO TO 2000
C
C   MNPAD
13      MNPAD=NUM
        GO TO 2000
C
C   FMAX
C
14      FMAX=FLOAT(NUM)
        GO TO 2000
C
C   LEVC
C
15      LEVC=NUM
```

```
      GO TO 2000
C
C   ZYKMV
C
16    ISTFLAG=1
      GO TO 2000
C
C   NOMAP
C
17    ICM(32)=3
      GO TO 2000
C
C
2001  CONTINUE
      IF(LECS.GE.ICM(1)) GO TO 3000
      LLECS=ICM(1)
      CALL MODIF(1,7,LLECS,1)
3000  CONTINUE
C
C
      IF(MNPAD.EQ.0) GO TO 4000
      IF(NX.EQ.0) GO TO 1004
      IF(FMAX.EQ.0.0) GO TO 1005
      IF(NX.GT.350) GO TO 1010
      IF(NX+MNPAD.GT.512) GO TO 1011
C       IF(NGI.EQ.0) GO TO 1002
C       IF(NOTIME.EQ.0) GO TO 1003
      IF(NSI.EQ.0) GO TO 2002
      NOS=NOTIME/NSI
      NOSMIG=NOS
      IF(NMTIME.EQ.0) GO TO 1500
      NOSMIG=NMTIME/NSI
      GO TO 1501
1500  NMTIME=NOTIME
1501  CONTINUE
      FNY=1000./(2.*FLOAT(NSI))
      IF(FMAX.GT.FNY) GO TO 1006
2002  CONTINUE
C
C
C
C
      GO TO 9999
C
4000  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(29H  = MNPAD MUST BE SPECIFIED =)
      GO TO 9889
1001  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(27H  = NSI MUST BE SPECIFIED =)
      GO TO 9889
1002  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(27H  = NGI MUST BE SPECIFIED =)
      GO TO 9889
1003  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(26H  = RL MUST BE SPECIFIED =)
      GO TO 9889
1004  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(26H  = NT MUST BE SPECIFIED =)
      GO TO 9889
1005  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(28H  = FMAX MUST BE SPECIFIED =)
      GO TO 9889
1006  CALL REMARK(16H** YOMIG ERR. **)
```

```
      CALL REMARK(20H  = FMAX .GT. FNYQ =)
      GO TO 9889
1007  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(22H  = VI MUST FOLLOW T =)
      GO TO 9889
1008  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(21H  = NO. PAIRS GT 10 =)
      GO TO 9889
1009  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(26H  = T MUST BE INCREASING =)
      GO TO 9889
1010  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(17H  = NX .GT. 350 =)
      GO TO 9889
1011  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(23H  = NX+MNPAD .GT. 512 =)
      GO TO 9889
9990  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(23H** VI MUST BE SPECIFIED)
      GO TO 9889
C
9888  CONTINUE
      CALL REMARK(28H** YOMIG ERR. = ILLIGAL CARD)
9889  NCO(1)=NCO(1).OR.1L5
      CALL LEVELO
9999  CONTINUE
C
C
      RETURN
      END
```

```
C
C
      OVERLAY(ZYKRT,1,1,C2000)
      PROGRAM ZYKRT
      EXTERNAL ZYKRT1
C
C   OVERLAY ZYKRT
C
C    READ INPUT TRACES INTO FILE IHEDR
C
      CALL START (ZYKRT1)
      CALL LEVEL 1
      END
```

```
C
      SUBROUTINE ZYKRT1(NCO,NDA,LDA,NIN,LIN,NOU,LOU,
     *                  A,LAA,IOUT,LOUT,
     *                  ICM,LCM,ICMN,LCMN,
     *                  IECS,LECS,KSCS,LSCS,
     *                  KSS,LSS)
C
      DIMENSION NCO(1),NDA(1),NIN(1),NOU(1),
     *          A(1),IOUT(1),ICM(1),ICMN(1)
C
      COMMON/COMM/ISZCOM,NOS,NSI,NX,NGI,IHEOR,MUXR,FMAX,
     *NTN,NSBLKF,NBLKF,NSBLKK,NBLKK,NPX,KP,MNPAD,ISTFLAG,
     *NOTIME,NMTIME,NOSMIG,THICK(31),DELZ(31),LEVC,
     *DTIME(31),NITER(31),VI(31),NOL
C
C   INITIALIZE DISK FILE IHEOR
C     READ TRACES FROM DISK BUFFER AND
C     STORE IN IHEOR
C
      IF(NOTIME.EQ.0.OR.NSI.EQ.0.OR.NGI.EQ.0) GO TO 90
      JIPUT=1
      GO TO 100
C
90    CONTINUE
      JIPUT=2
      CALL RTRACE(1,1,IOUT)
      CALL PRETE(1,1)
C
      IF(NSI.EQ.0) NSI=IOUT(9)
      IF(NGI.EQ.0) NGI=(IOUT(15).AND.000000000001777778)
     * *50
      IF(NOTIME.EQ.0) NOTIME=IOUT(10)
      NOS=NOTIME/NSI
      NOSMIG=NOS
      IF(NMTIME.EQ.0) GO TO 210
      NOSMIG=NMTIME/NSI
      GO TO 110
210   NMTIME=NOTIME
110   CONTINUE
      FNY=1000./(2.*FLOAT(NSI))
      IF(FMAX.GT.FNY) GO TO 1006
      GO TO 100
1006  CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(20H  = FMAX .GT. FNYQ =)
      NCO(1)=NCO(1).OR.1L5
      CALL LEVELO
100   CONTINUE
C
      LONG=NOS*64
      INOM=5LIHEOR
      DO 1700 I=1,LONG
      A(I)=0.
1700  CONTINUE
C
      L=1
      LL=16
      NG=1
      NSGR=NX
      NSG=NSGR+1
      KED=ICM(15)
      ICM(15)=ICM(15)+NSG
      CALL CREFICH(INOM,ICMN(L),LL,KED,NSG,1,NSGR)
      CALL CREGR(INOM,1,LONG)
```

```
      DO 1701 I=1,NSGR
      CALL WRFICH(INOM,1,I,A,LONG,0)
      CALL ATTENTE(INOM)
1701  CONTINUE
C
      IF(JIPUT.EQ.1) GO TO 300
C
      NAL=IOUT(13)
      DO 1801 J=1,64
      A(J)=FLOAT(IOUT(J))
1801  CONTINUE
      IF(IOUT(11).EQ.7) GO TO 1850
      MOTON=IOUT(11)
      IOUT(11)=7
      IOUT(13)=0
      NAL=IOUT(13)
      DO 1851 J=1,NOS
      IOUT(64+J)=0
1851  CONTINUE
      DO 1849 J=1,64
      A(J)=FLOAT(IOUT(J))
1849  CONTINUE
      IOUT(11)=0
C
1850  CONTINUE
      DO 1852 J=1,NOS
      A(64+J)=IOUT(64+J)*(2.**NAL)
1852  CONTINUE
      CALL SUPRSGR(INOM,1,1)
      CALL WRFICH(INOM,1,1,A,LONG,0)
      CALL ATTENTE(INOM)
C
300   CONTINUE
      DO 1900 I=JIPUT,NX
      CALL RTRACE(1,I,IOUT)
      CALL PRETE(1,I)
      NAL=IOUT(13)
      DO 1901 J=1,64
      A(J)=FLOAT(IOUT(J))
1901  CONTINUE
      IF(IOUT(11).EQ.7) GO TO 1950
      MOTON=IOUT(11)
      IOUT(11)=7
      IOUT(13)=0
      NAL=IOUT(13)
      DO 1951 J=1,NOS
      IOUT(64+J)=0
1951  CONTINUE
      DO 1949 J=1,64
      A(J)=FLOAT(IOUT(J))
1949  CONTINUE
      IOUT(11)=0
C
1950  CONTINUE
      DO 1952 J=1,NOS
      A(64+J)=IOUT(64+J)*(2.**NAL)
1952  CONTINUE
      CALL SUPRSGR(INOM,1,I)
      CALL WRFICH(INOM,1,I,A,LONG,0)
      CALL ATTENTE(INOM)
1900  CONTINUE
      RETURN
      END
```

```
C
C
      OVERLAY(ZYKMV,1,1,C2C00)
      PROGRAM ZYKMV
      EXTERNAL ZYKMIV1
C
C   OVERLAY ZYKMV
C
C    INITIALIZATION AND CALCULATION
C    OF NECESSARY PARAMETERS.
C
      CALL START 1 (ZYKMIV1)
      CALL LEVEL 1
      END
```

```
C
      SUBROUTINE ZYKMIV1(NCO,NDA,LDA,NIN,LIN,NOU,LOU,
     *                   A,LAA,IOUT,LOUT,
     *                   ICM,LCM,ICMN,LCMN,
     *                   IECS,LECS,KSCS,LSCS,
     *                   KSS,LSS,NSCA,NSCB)
      DIMENSION NCO(1),NDA(1),NOU(1),A(1),NIN(1),
     *          IOUT(1),ICM(1),ICMN(1)
C
      DIMENSION NSCA(1)
C
C
      KA=13000
      CALL RFLCM(KA)
      CALL ZYKMV2(NSCA(1),NSCA(101),A,IECS,ICM,ICMN,NCO)
      CALL REDUCE(2)
      RETURN
      END
```

```
C
C
      SUBROUTINE ZYKMV2(A,B,C,IECS,ICM,ICMN,NCO)
C
C   THIS SUBROUTINE INITIALIZE PARAMETERS
C   FIND DELZ,DTIME,NITER, ETC ...
C
      DIMENSION NCO(1),A(1),B(1),C(1),ICM(1),ICMN(1)
C
      COMMON/COMM/ISZCOM,NOS,NSI,NX,NGI,IHEDR,MUXR,FMAX,
     *NTN,NSBLKF,NBLKF,NSBLKK,NBLKK,NPX,KP,MNPAD,ISTFLAG,
     *NOTIME,NMTIME,NOSMIG,THICK(31),DELZ(31),LEVC,
     *DTIME(31),NITER(31),VI(31),NOL
C
C     FOR COMMON /COMM/
C
C     ISZCOM          SIZE OF THIS COMMON
C     NOS             NUMBER OF SAMPLES
C     NSI             SAMPLING INTERVAL(MSEC)
C     NX              NO. OF TRACES TO PROCESS
C     NGI             COP INTERVAL *100
C     IHEDR           LOGICAL NO. FOR FILE IHEDR
C     MUXR            LOGICAL NO. FOR FILE MUXR
C     FMAX            MAX. FREQ. TO MIGRATE
C     NTN             NO. OF PTS IN F UP TO FMAX
C     NSBLKF          NO. OF PTS IN ONE F BLOCK
C     NBLKF           NO. OF BLOCKS IN F
C     NSBLKK          NO. OF PTS IN ONE K BLOCK
C     NBLKK           NO. OF BLOCKS IN K
C     NPX             NO. OF PTS IN K FOR FFT /2
C     KP              NO. OF PTS IN F FOR FFT
C     MNPAD           NO. OF ZERO TRACSE TO PAD FOR FFT
C     ISTFLAG         FLAG FOR ZYKMV , 1 FOR STOP
C                                      0 FOR NO-STOP
C     NOTIME          INPUT RECORD LENGTH
C     NMTIME          MAX. TIME TO MIGRATE
C     NOSMIG          NO. OF PTS FOR NMTIME
C     THICK           THICKNESS FOR EACH LAYER
C     DELZ            Z INCREMENTS FOR EACH LAYER
C     LEVC            NO. OF BIT SHIFT WHICH CAN
C                     INCRESED OR DECRESED
C     DTIME           DEL TIME FOR EACH LAYER
C     NITER           NO. OF ITERATION WITHIN LAYER
C     VI              INTERVAL VEL. FOR EACH LAYER
C     NOL             NO. OF LAYER
C
C
C
      IHEDR=5
      MUXR=1
      NPAIR=ICM(48)
      IV=50
      ITM=80
      VI(1)=FLOAT(ICM(51))/100
      THICK(1)=(FLOAT(ICM(81)*ICM(51))/100000)/2
      IF(NPAIR.EQ.1) GO TO 6
C
      DO 5 I=2,NPAIR
      VI(I)=FLOAT(ICM(IV+I))/100
      THICK(I)=(FLOAT((ICM(IV+I)*(ICM(ITM+I)-ICM(ITM+I-1)
     *)))/100000)/2
5     CONTINUE
6     CONTINUE
```

```
      VI(NPAIR+1)=FLOAT(ICM(111))/100
      KTIME=NMTIME-ICM(ITM+NPAIR)
      THICK(NPAIR+1)=(FLOAT(KTIME*ICM(111))/100000)/2
C
C
      NOL=NPAIR+1
      SIM=FLOAT(NSI)/1000
C
C   SET UP NUMBER OF ITERATION AT DESIRED
C   SAMPLING INTERVAL
C
      DO 10 I=1,NOL
      TTEST=THICK(I)/VI(I)
      IF(TTEST.LT.SIM/2) GO TO 500
10    CONTINUE
      GO TO 12
C
500   CALL REMARK(16H** YOMIG ERR. **)
      CALL REMARK(17H* DELZ .LT. SIM *)
      NCO(1)=NCO(1).OR.1L5
      CALL LEVELO
C
12    CONTINUE
      DDTIME=0.
      DO 15 I=1,NOL
      TIMEINT=THICK(I)/VI(I)-DDTIME
      DELZ(I)=SIM*VI(I)/2
      NITER(I)=INT(TIMEINT*2/SIM)+2
17    DDT=SIM*NITER(I)/2
      ADDT=ABS(DDT-TIMEINT)
      IF(ADDT.LE.SIM/100) GO TO 20
      IF(DDT.LT.TIMEINT) GO TO 22
      NITER(I)=NITER(I)-1
      GO TO 17
22    DF1=TIMEINT-DDT
      IF(DF1.GE.SIM/2) GO TO 24
      DTIME(I)=DF1
      DDTIME=SIM/2-DF1
      GO TO 15
24    NITER(I)=NITER(I)+1
      GO TO 17
20    DDTIME=0.0
      DTIME(I)=0.0
15    CONTINUE
C
      CALL LOCK
      PRINT 1070,FMAX
1070  FORMAT(/,2X,*MAX. FREQ. TO MIGRATE =*,F10.2,/)
      PRINT 1071
1071  FORMAT(//,4X,*THICK*,5X,*VI*,4X,*NITER*,4X,*DELZ*,
     *4X,*DTIME*,/)
      DO 7 I=1,NOL
      PRINT 1072, THICK(I),VI(I),NITER(I),DELZ(I),DTIME(I)
1072  FORMAT(3X,F7.2,2X,F7.0,3X,I3,4X,F5.2,3X,F8.6)
7     CONTINUE
      CALL DELOCK
C
C
      CALL RELEASE
      RETURN
      END
```

```
C
C
      OVERLAY (ZYFKA,1,1,C2000)
      PROGRAM ZYFKA
      EXTERNAL ZKMIA21
C
C   OVERLAY ZYFKA
C
C    BMUX1 - MULTIPLEX OF DATA TO FILE MUXR
C    RFFT  - FFT IN X - DIRECTION
C
      CALL START (ZKMIA21)
      CALL LEVEL1
      END
```

```
C*****
      SUBROUTINE ZKMIA21(NCO,NDA,LDA,NIN,LIN,NOU,LOU,
     *                   A,LAA,
     *                   IOUT,LOUT,
     *                   ICM,LCM,ICMN,LCMN,
     *                   IECS,LECS,KSCS,LSCS,
     *                   KSS,LSS,NSCA,NSCB)
      DIMENSION NCO(1),NDA(1),NOU(1),NIN(1),
     *          A(1),
     *          IOUT(1),
     *          ICM(1),ICMN(1)
      COMMON/COMM/ISZCOM,NOS,NSI,NX,NGI,IHEDR,MUXR,FMAX,
     *NTN,NSBLKF,NBLKF,NSBLKK,NBLKK,NPX,KP,MNPAD,ISTFLAG,
     *NOTIME,NMTIME,NOSMIG,THICK(31),DELZ(31),LEVC,
     *DTIME(31),NITER(31),VI(31),NOL
*$
*$    FIND NBLKK,NSBLKK,NBLKF,NSBLKF ETC ..
*$    FOR MULTIPLEXING DATA , AND CALL SUBROUTINES
*$    BMUX1,RFFT
C
C
      INTEGER Z,ZZ
      DIMENSION NSCA(1)
C
C
C
C       ******    COMPUTE BUFFER SIZES    ******
C
      CALL FFTINF(NOS,NSI,KP,DUMMY)
      KP=KP*2
C  ----- FIND NO. OF POINT FOR FMAX
      KPP=NPFMAX(FMAX,NSI,KP)
      NTN=KPP
      IF(NTN.GT.KP/2+1) NTN=KP/2+1
      IF(NOSMIG.GT.KPP) KPP=NOSMIG
C
      IA=KP
C
      CALL FFTINF(NX+MNPAD,NSI,NPX,DUMMY)
      NPXX=NPX*4
C
      JFLAG=0
      IF(ICM(1).NE.0) GO TO 80
C
C         COMPUTING SIZE ECS ARRAY
C
      ICM(1)=70000
81    IF(JFLAG.NE.0) ICM(1)=ICM(1)+10000
80    IF(ICM(1).GT.130000) GO TO 16
      IF(ICM(1).LT.45000) GO TO 16
      ICOR=ICM(1)-20000
      ICSIZE=ICOR-200
C          OR SIZE OF LARGEST FFT BUFFER
      IB=NPXX*2
      IF(IB.GT.IA) IA=IB
C          SET ICSIZE FOR MPX BUFFER COMPUTE
      ICSIZE=ICSIZE-IA
C
C          COMPUTE CONSTANTS FOR MPX/DMPX ARRAY
C          NO SAMPLES PER BLOCK OF F --- NSBLKF
      NSBLKF=((ICSIZE/NPXX)/8)*8
C          NO SAMPLES PER BLOCK OF K --- NSBLKK
      NSBLKK=ICSIZE/KPP
```

```
C
      IF(NSBLKF.GT.KPP) NSBLKF=KPP
      IF(NSBLKK.GT.NPXX) NSBLKK=NPXX
C
C          GET NO BLOCKS IN F --- NBLKF
15    CONTINUE
      NBLKF=KPP/NSBLKF
      IF(NBLKF*NSBLKF.LT.KPP) NBLKF=NBLKF+1
C
C          MAKE NSBLKK EVEN FOR FMIG3
      NSBLKK=NSBLKK/2*2
C          GET NO BLOCKS IN K --- NBLKK
20    CONTINUE
      NBLKK=NPXX/NSBLKK
      IF(NBLKK*NSBLKK.LT.NPXX) NBLKK=NBLKK+1
C
      IC=NBLKK*NSBLKK*NSBLKF
      IF(IC.LE.ICSIZE) GO TO 30
      NSBLKF=NSBLKF-8
      IF(NSBLKF.GT.0) GO TO 15
      JFLAG=10
      GO TO 81
      IF(NSBLKF.LE.0) GO TO 16
      GO TO 15
16    CALL REMARK (17H** YOMIG ERROR **)
      CALL REMARK(33H= NO OF SAMPLES/BLOCK TOO SMALL =)
      NCO(1)=NCO(1).OR.1L5
      CALL LEVELO
30    CONTINUE
C
C          TOTAL NUMBER OF BLOCKS
C          GET LARGER NO SAMPS TOTAL K OR F SLICE
      IC=NBLKK*NSBLKK*NSBLKF
      IB=NBLKF*NSBLKK*NSBLKF
      IF(IC.GT.IB) IB=IC
C
C          FIND NO 16 BIT WORD PER DISK DUMP
C
      NSPDMP=NSBLKK*NSBLKF
      Z=NBLKF
      ZZ=NBLKK+2
      ZZ=Z*ZZ
C
C          CHECK CORE
      NCORE=IA+IB
      IF(NCORE.LE.ICOR) GO TO 40
      NSBLKK=NSBLKK-2
      GO TO 20
40    CONTINUE
C
C           INITIALIZE DISK FILE
C
C
      DO 2000 I=1,LAA
2000  A(I)=0.
C
C           MUXR FILE
C
      KKLMUX=NSPDMP+100
      CALL RFLCM(KKLMUX)
      DO 2005 I=1,NSPDMP
2005  NSCA(I)=0
      INCM=4LMUXR
```

```
      LL=69
      ICMN(LL-1)=LOCF(ICMN)
      L=16
      NSGR=ZZ
      NSG=ZZ+1
      KEDS=ICM(15)
      LONG=NSPDMP
      CALL WRITEC(NSCA,IECS+2000,LONG)
      CALL REDUCE(2)
      CALL XEDOPEN(INOM,LONG,ICMN(105),KSS,ISTAT)

      DO 2001 I=1,NSGR
      CALL XEDWRIT(ICMN(105),IECS+2000,I,I,0,ISTAT)
      CALL XEDWAIT(ICMN(105),ISTAT)
2001  CONTINUE
C
C          COMPUTE ARRAY INDEXES
C
      LTC=1
C
      ITHDR=1+LTC+NX
C
      IXD=ITHDR+64
C
      IDEP=IXD+NOS+22
C
      IWORK=IDEP
C
C          MUX - DMUX ARRAY
      MPX=IWORK+NOS+65
C
C
C          START OF F SLICE FFT ARRAY
      IFFT=1
C          MUX ARRAY WHEN DOING F SLICE FFT
      MPX2=IFFT+NPXX+NPXX+1
C
C
C                         DO BIG MULTIPLEX
      KA=MPX+500
      IF(KA.LT.(MPX2+500)) KA=MPX2+500
      CALL RFLCM(KA)
      CALL BMUX1(NSCA(ITHDR),NSCA(IXD),NSCA(MPX),A,
     *NSCA(LTC),ICM,IOUT,LOUT,IECS,ICMN,KSCS)
C
C                          DO F SLICE FFT,S
      CALL RELEASE
      CALL RFFT(NSCA(IFFT),NSCA(MPX2),NSBLKF,NBLKF,
     *NSBLKK,NBLKK,NPX,NX,NOS,IECS,NSCA,ICMN,ICM)
      CALL REDUCE(2)
C
      CALL RELEASE
      RETURN
      END
```

```
C
C
C  ************************************************************
C
C
*$TITLE                 -BMUX1
*$ABSTRACT              -TO READ THE DATA FROM TAPE AND
*$                      -MULTIPLEX IT TO THE DISK
*$SAMPLE CALL           -CALL BMUX1(ITHDR,A,C,IWORK,B)
*$CALLING PARAMETER
*$        -ITHDR        -TRACE HEADER ARRAY
*$        -A            -INPUT ARRAY
*$        -B            -WORK ARRAY
*$        -C            -MPX WORK ARRAY
*$        -IWORK        -WORK ARRAY
C
      SUBROUTINE BMUX1(ITHDR,A,C,IWORK,B,ICM,IOUT,
     *LOUT,IECS,ICMN,KSCS)
C
C
      COMMON/COMM/ISZCOM,NOS,NSI,NX,NGI,IHEDR,MUXR,FMAX,
     *NTN,NSBLKF,NBLKF,NSBLKK,NBLKK,NPX,KP,MNPAD,ISTFLAG,
     *NOTIME,NMTIME,NOSMIG,THICK(31),DELZ(31),LEVC,
     *DTIME(31),NITER(31),VI(31),NCL
C
      DIMENSION ICM(1)
      DIMENSION IOUT(1)
      DIMENSION ITHDR(1),A(1),C(1),B(1)
      DIMENSION IWORK(1)
      DIMENSION ICMN(1)
      IADS=IECS+2000
C
      JHDRI=0
      ILAST=0
      NTC=0
      NMB=1
      NSBFK=NSBLKK*NSBLKF
      NSA=NOS+64
C
C
C
C           NBLKK TOTAL BLOCKS K
      DO 1000 IL=1,NBLKK
C           NSBLKK TRACES/BLOCK K
      DO 100 I=1,NSBLKK
      IF(ILAST.EQ.1) GO TO 50
      NTC=NTC+1
      IF(NTC.GT.NX) GO TO 5000
C------- READ A TRACE
      DO 101 IJKL=1,NOS
101   A(IJKL)=0.
      INOM=5LIHEDR
      LONG=NOS+64
      CALL RDFICH(INOM,1,NTC,ITHDR,64,0)
      CALL ATTENTE(INOM)
      CALL RDFICH(INOM,1,NTC,A,NOS,1)
      CALL ATTENTE(INOM)
C
C
50    CONTINUE
      MX=1
      MXX=(I-1)*NSBLKF+1
      MXJ=MXX
```

```
C
C         NBLKF TOTAL BLOCK F
          INOM=4LMUXR
          IADSS=IADS+MXJ
          CALL COLLIN(1,A(MX),NSBLKF,IADS+MXJ,NSBFK,NBLKF,
         *NSBLKF)
          GO TO 100
C           END OF INPUTS,COME HERE
5000      CONTINUE
          CALL PCLRW(A,2,NOS,15)
          ILAST=1
          GO TO 50
C
C
100       CONTINUE
C
C           CORE FULL  NOW DAMP TO DISK
C
          MXX=1
          NMB2=NMB+NBLKF-1
          CALL XEDWRIT(ICMN(105),IADS+MXX,NMB,NMB2,0,ISTAT)
          CALL XEDWAIT(ICMN(105),ISTAT)
          NMB=NMB+NBLKF
C
C
          IF(NTC.GT.NX) GO TO 1100
C
1000      CONTINUE
1100      CONTINUE
          RETURN
          END
```

```
C
C
      FUNCTION NPFMAX(FMAX,NSI,KP)
C
C     FIND NO OF POINT FOR FMAX
C
C      FMAX - MAXIMUM FREQUENCY OF INTEREST.
C       NSI - SAMPLING INTERVAL IN MSEC.
C         KP - NUMBER OF POINTS IN FREQUENCY DOMAIN.
C
C
      DF=1000./(FLOAT(NSI)*FLOAT(KP))
      IPOINT=0
      DDF=0.
      DO 100 I=1,10000
      IF(DDF.GT.FMAX) GO TO 200
      DDF=DDF+DF
      IPOINT=IPOINT+1
100   CONTINUE
200   CONTINUE
      NPFMAX=IPOINT+1
      RETURN
      END
```

```
C
C
C  ************************************************************
C
C
*$TITLE                -RFFT
*$NAME                 -RFFT
*$ABSTRACT             -TO PERFORM FIRST PART OF THE
*$                     -TWO DIMENSIONAL TRANSFORM
*$SAMPLE CALL          -CALL RFFT(FFT,A,NSBLKF,NBLKF,
*$                     -NSBLKK,NBLKK,NPX,NX,NOS)
*$CALLING PARAMETER
*$        -FFT         -FFT ARRAY
*$        -A           -ARRAY TO HOLD MPLX. DATA
*$        -NSBLKF      -SAMPLES/BLOCK F
*$        -NBLKF       -BLOCKS OF F
*$        -NSBLKK      -SAMPLES/BLOCK K
*$        -NBLKK       -BLOCKS OF K
*$        -NPX         -(NO. OF PAIRS IN F SLICE FFT)/2
*$        -NX          -NO. TRACES
*$        -NOS         -NO. SAMPLES PER TRACE
*$
      SUBROUTINE RFFT(FFT,A,NSBLKF,NBLKF,NSBLKK,
     *NBLKK,NPX,NX,NOS,IECS,NSCA,ICMN,ICM)
C
C
      DIMENSION IADSS(2)
      DIMENSION FFT(1),A(1)
      DIMENSION ICMN(1)
      DIMENSION NSCA(1)
      DIMENSION ICM(1)
C
      IADS=IECS+2000
C
      IADSS(1)=((IECS+ICM(1)-(NPX*4+500))/8)*8
      IADSS(2)=IADSS(1)+NPX*2+70B
      JOB=147B
C
      IARG=ICM(32)
C
      INOM=4LMUXR
      NSBF2=NSBLKF*2
      N=2*NX+1
      L=4*NPX
      L=L-N
      NPX4=NPX*4
      NSINC=NSBLKK*NSBLKF
      LONG=NSINC
C                  NBLKF   NO. BLOCKS/TRACE
C
      IF(IARG.EQ.3) GO TO 300
      NNN=NPX*2
      CALL SINE(FFT,IADSS,NNN)
      NB=NNN
300   CONTINUE
      DO 1000 ISBLK=1,NBLKF
C
C          READ UP DATA FOR F SLICE FFT
      MX=1
      IBLK=ISBLK
      DO 10 II=1,NBLKK
C           DONT DO MORE THAN NEEDED
      IB=(II-1)*NSBLKK
```

```
      IF(IB.GT.NX) GO TO 20
      CALL XEDREAD(ICMN(105),IADS+MX,IBLK,IBLK,0,ISTAT)
      CALL XEDWAIT(ICMN(105),ISTAT)
      IBLK=IBLK+NBLKF
      MX=MX+NSINC
10    CONTINUE
C
C           DO NSBLKF F SLICE FFTS
C               SET LOOP COUNTER SO NOT
C               TO DO MORE THAN NEEDED
20    CONTINUE
      NSBF=NSBLKF
      IC=NSBLKF*ISBLK
      IF(IC.GT.NOS) NSBF=NSBF-(IC-NOS)
C
      DO 150 I=1,NSBF
C           GET F SLICE TO FFT
      MX=I
      CALL COLLIN(2,FFT(1),2,IADS+MX,NSBLKF,NX,1)
C             CLEAR THE TAIL
      CALL PCLRW(FFT(2),4,NX,15)
      CALL PCLRW(FFT(N),2,L,15)
C           FFT THE TRACE
      S=1.0
      NNN=NPX*2
      IF(IARG.NE.3) GO TO 310
      CALL FOUR1(FFT,NNN,1)
      GO TO 320
310   CALL FFOUR1(FFT,IADSS,NB,JOB,1)
320   CONTINUE
C
C           MOVE FFTED F SLICE BACK TO MPLX. ARRAY
      CALL COLLIN(1,FFT,1,IADS+MX,NSBLKF,NPX4,1)
C
150   CONTINUE
C
C           WRITE FFTED BLOCKS BACK TO DISK
C           NBLKK NO. BLOCKS TO HOLD F SLICE FFTS
C
C
      IBLK=ISBLK
      MX=1
      DO 200 II=1,NBLKK
      CALL XEDWRIT(ICMN(105),IADS+MX,IBLK,IBLK,0,ISTAT)
      CALL XEDWAIT(ICMN(105),ISTAT)
      IBLK=IBLK+NBLKF
      MX=MX+NSINC
200   CONTINUE
C
C
1000  CONTINUE
      RETURN
      END
```

```
C
C
C
      OVERLAY (ZYFKB,1,1,C2000)
      PROGRAM ZYFKB
      EXTERNAL ZKMIB21
C
C   OVERLAY ZYFKB
C
C    FFT IN T - DIRECTION AND MIGRATE
C
      CALL START (ZKMIB21)
      CALL LEVEL1
      END
```

```
C*****
      SUBROUTINE ZKMIB21(NCO,NDA,LDA,NIN,LIN,NOU,LOU,
     *                   A,LAA,
     *                   IOUT,LOUT,
     *                   ICM,LCM,ICMN,LCMN,
     *                   IECS,LECS,KSCS,LSCS,
     *                   KSS,LSS,NSCA,NSCB)
      DIMENSION NCO(1),NDA(1),NOU(1),NIN(1),
     *          A(1),
     *          IOUT(1),
     *          ICM(1),ICMN(1)
      DIMENSION NSCA(1)
      COMMON/COMM/ISZCOM,NOS,NSI,NX,NGI,IHEOR,MUXR,FMAX,
     *NTN,NSBLKF,NBLKF,NSBLKK,NBLKK,NPX,KP,MNPAD,ISTFLAG,
     *NOTIME,NMTIME,NOSMIG,THICK(31),DELZ(31),LEVC,
     *DTIME(31),NITER(31),VI(31),NOL
*$
*$ABSTRACT              -TO DO LAST PART OF TRANSFORM
*$                      -AND MIGRATE
*$
C
C                       SET ARRAY INDEXES
      IFFT=1
      IWORK=IFFT+NBLKF*NSBLKF*2+5
      IWORK1=IWORK+NOSMIG*2+4
      IB=IWORK1+66
C
C                       CHECK FOR ENOUGH CORE
      IC=ICM(1)-17000
      MPLX=IB
      NWS=0
      IF(IB.GT.MPLX)NWS=IB-MPLX+4
510   CONTINUE
C
C                       GO MIGRATE DATA
      KA=IB+500
      CALL RFLCM(KA)
      CALL KTRANS1(NSCA,NSCA(MPLX),NSCA(IFFT),
     *NSCA(IWORK),A,NWS,IECS,ICMN,ICM)
      CALL REDUCE(2)
C
      CALL RELEASE
      RETURN
      END
```

```
C
C
*$TITLE                  -KTRANS1
*$NAME                   -KTRANS1
*$ABSTRACT               -TO DO THE LAST PART OF
*$                       -THE TRANSFORM AND MIGRATE
*$                       -THE DATA
*$
*$SAMPLE CALL            -CALL KTRANS1(NSCA,A,FFT,FFT1,WORK,
*$                           NWRIT)
*$CALLING PARAMETER
*$        -A             -ARRAY FOR MPLX DATA
*$        -FFT           -ARRAY FOR UNMIGRATED SPECTRUM
*$        -WORK          -ARRAY FOR DOWNWARD CONTINUATION
*$                        SPECTRUM
*$        -FFT1          -ARRAY FOR MIGRATED SPECTRUM
*$        -NWRIT         -NO OF 12 WORDS TO WRITE TO DISC
*$
      SUBROUTINE KTRANS1(NSCA,A,FFT,FFT1,WORK,NWRIT,IECS,
     *ICMN,ICM)
      EXTERNAL PREMP91
C
      DIMENSION ICMN(1)
      DIMENSION ICM(1)
      DIMENSION FFT(1),WORK(1),FFT1(1)
      DIMENSION NSCA(1)
      DIMENSION IADSS(2)
      DIMENSION A(1)
      INTEGER VV0(5),VV1(5),VV2(5),ISTM(2)
C
      COMMON/COMM/ISZCOM,NCS,NSI,NX,NGI,IHEDR,MUXR,FMAX,
     *NTN,NSBLKF,NBLKF,NSBLKK,NBLKK,NPX,KP,MNPAD,ISTFLAG,
     *NOTIME,NMTIME,NOSMIG,THICK(31),DELZ(31),LEVC,
     *DTIME(31),NITER(31),VI(31),NCL
C
C
C
C    INITIALIZATION FOR MAP CMULT
C
      VV0(1)=LOCF(PREMP91)
      VV0(2)=91
      VV0(3)=NTN
      VV0(4)=0
      VV0(5)=0
      VV1(1)=2LFF
      VV1(2)=1
      VV1(3)=NTN*2
      VV1(4)=0
      VV1(5)=1478
      VV2(1)=2LFF
      VV2(2)=2
      VV2(3)=NTN*2
      VV2(4)=1
      VV2(5)=1478
C
C       COMPUTE CONSTANT
      IADS=IECS+1000
      DELTAK=100./(FLOAT(NGI)*FLOAT(NPX)*2.)
      DELTAF=1000./(FLOAT(NSI)*FLOAT(KP))
      KP2=KP*2
      IADSS(1)=((IECS+ICM(1)-(KP2+500))/8)*8
      IADSS(2)=IADSS(1)+KP+708
```

```
      JOB=1478
      IARG=ICM(32)
      IF(IARG.EQ.3) GO TO 600
      NNN=KP
      CALL SINE(FFT,IADSS,NNN)
      NB=NNN
600   CONTINUE
C
C---------- IDSK IS DISC BLK COUNT
C
      KQ=KP
      NSBFK=NSBLKF*NSBLKK
      NSBF2=NSBLKF*2
      NS2=NOS*2
      L=KP2-NS2
      N=NS2+1
      ICNT=0
      IB=1
C
      SIM=FLOAT(NSI)/1000
C
      DO 1000 II=1,NBLKK
C                 READ UP NSBLKK/2 K SLICES
C
      MX=1
      IBLK=IB
      IBLK2=IBLK+NBLKF-1
      CALL XEDREAD(ICMN(105),IADS+MX,IBLK,IBLK2,0,ISTAT)
      CALL XEDWAIT(ICMN(105),ISTAT)
C
      NSBLKD=NSBLKK/2
      DO 500 JJ=1,NSBLKD
      MI=(JJ-1)*NSBF2+1
C               GET K SLICE
      ICNT=ICNT+1
      IF(ICNT.GT.2*NPX) GO TO 500
      MX=MI
      MJ=1
      DO 100 J=1,NBLKF
      CALL COLLIN(2,FFT(MJ),2,IADS+MX,1,NSBLKF,1)
      CALL COLLIN(2,FFT(MJ+1),2,IADS+MX+NSBLKF,1,NSBLKF,1)
      MX=MX+NSBFK
      MJ=MJ+NSBF2
100   CONTINUE
C
C       CLEAR TAIL
      CALL PCLRm(FFT(N),2,L,15)
C
C       COMPLEX FFT THE K SLICE
      S=1.0
      NNN=KP
      IF(IARG.NE.3) GO TO 610
621   CONTINUE
      CALL FOUR1(FFT,NNN,1)
      GO TO 620
610   IF(NNN.GE.4096) GO TO 621
      CALL FFOUR1(FFT,IADSS,NB,JOB,1)
620   CONTINUE
C
C
C  MIGRATION
C
      IJCNT=ICNT
```

```
      IF(ICNT.GT.NPX+1) IJCNT=NPX*2-ICNT+2
C
      CALL FREQSUM(FFT,NTN,DELTAF,PRI,PII)
      FFT1(1)=PRI
      FFT1(2)=PII
C
      NOSM=2
      DO 700 KI=1,NOL
      VEL2=VI(KI)*VI(KI)
      CALL CALSPEC(IJCNT,DELTAK,DELTAF,NTN,VEL2,DELZ(KI),
     *WORK)
      INITER=NITER(KI)
      DO 710 IK=1,INITER
C
      ISTM(1)=1
      CALL CVBOX(VVO,ISTM,2,FFT,VV1,WORK,VV2)
C
      CALL FREQSUM(FFT,NTN,DELTAF,PRI,PII)
      FFT1(NOSM*2-1)=PRI
      FFT1(NOSM*2)=PII
C
      NOSM=NOSM+1
      IF(NOSM.GT.NOSMIG) GO TO 800
710   CONTINUE
C
C  THIS PART TAKES CARE OF INTERFACE
C
      IF(DTIME(KI).EQ.0.0) GO TO 700
      DDEP=DTIME(KI)*VI(KI)
      CALL CALSPEC(IJCNT,DELTAK,DELTAF,NTN,VEL2,DDEP,
     *WORK)
      ISTM(1)=1
      CALL CVBOX(VVO,ISTM,2,FFT,VV1,WORK,VV2)
      IF(KI.EQ.NOL) GO TO 755
      VEL2=VI(KI+1)*VI(KI+1)
      GO TO 760
755   VEL2=VI(KI)*VI(KI)
760   DDTIME=SIM/2-DTIME(KI)
      DDDEP=DDTIME*SQRT(VEL2)
      CALL CALSPEC(IJCNT,DELTAK,DELTAF,NTN,VEL2,DDDEP,
     *WORK)
      ISTM(1)=1
      CALL CVBOX(VVO,ISTM,2,FFT,VV1,WORK,VV2)
      CALL FREQSUM(FFT,NTN,DELTAF,PRI,PII)
      FFT1(NOSM*2-1)=PRI
      FFT1(NOSM*2)=PII
      NOSM=NOSM+1
      IF(NOSM.GT.NOSMIG) GO TO 800
C
700   CONTINUE
C
800   CONTINUE
C
C
C
C          PUT K SLICE BACK
C
      MX=MI
      MJ=1
      DO 200 J=1,NBLKF
      CALL COLLIN(1,FFT1(MJ),2,IADS+MX,1,NSBLKF,1)
      CALL COLLIN(1,FFT1(MJ+1),2,IADS+MX+NSBLKF,1,NSBLKF,1)
      MX=MX+NSBFK
```

```
      MJ=MJ+NSBF2
200   CONTINUE

490   CONTINUE
500   CONTINUE
C
C     WRITE DOWN DATA IN DISC
C
      MX=1
      IBLK=IB
      IBLK2=IBLK+NBLKF-1
      CALL XEDWRIT(ICMN(105),IADS+MX,IBLK,IBLK2,0,ISTAT)
      CALL XEDWAIT(ICMN(105),ISTAT)
C
      IB=IB+NBLKF
1000  CONTINUE
C
      RETURN
      END
```

```
C
C
*$        -CALSPEC(JX,DKX,DF,III,VEL2,THICK,SPECT)
*$        -CALCULATE COMPLEX SPECTRUM OF MIGRATION
*$         OPERATOR
*$
*$              -JX         INDEX IN KX DIRECTION
*$              -DKX        DELTA INCREMENT IN KX
*$              -DF         DELTA INCREMENT IN F
*$              -III        NUMBER OF PAIRS IN F-DIRECTION
*$              -VEL2       VEL**2
*$              -THICK      THICKNESS OF LAYER
*$
*$              -SPECT      OUTPUT SPECTRUM
*$
      SUBROUTINE CALSPEC(JX,DKX,DF,NTN,VEL2,THICK,SPECT)
      DIMENSION SPECT(1)
      DATA PAI2/6.2831852/
C
      AKX=(JX-1)*DKX
      AKX2=AKX*AKX
C
      JJJ=1
      DO 10 JZ=1,NTN
      AF=(JZ-1)*DF
      AF2=AF*AF
      DIS=(4.0*AF2/VEL2-AKX2)
      IF(DIS.LT.0.0) GO TO 20
      D=SQRT(DIS)
      CC=-PAI2*D*THICK
      SPECT(JJJ)=COS(CC)
      SPECT(JJJ+1)=SIN(CC)
      GO TO 30
20    SPECT(JJJ)=0.0
      SPECT(JJJ+1)=0.0
30    JJJ=JJJ+2
10    CONTINUE
      RETURN
      END
```

```
C
C ***********************************************************
C
      SUBROUTINE FREQSUM(COMP,NTN,DF,PRI,PII)
C
C  SUM ALONG FREQUENCY
C
C     COMP - INPUT ARRAY.
C      NTN - NUMBER OF PAIRS IN COMP.
C       DF - UNUSED.
C      PRI - SUMMED REAL PART.
C      PII - SUMMED IMAGINARY PART.
C
C
      DIMENSION COMP(1)
      PRI=0.
      PII=0.
      IJ1=1
      IJ2=2
C
      DO 10 JZ=1,NTN
      PRI=PRI+COMP(IJ1)
      PII=PII+COMP(IJ2)
      IJ1=IJ1+2
      IJ2=IJ2+2
10    CONTINUE
      RETURN
      END
```

```
C
C
      OVERLAY (ZYFKC,1,1,C2000)
      PROGRAM ZYFKC
      EXTERNAL ZKMIC21
C
C   OVERLAY ZYFKC
C
C    INVERSE FFT IN X - DIRECTION AND OUTPUT
C
      CALL START (ZKMIC21)
      CALL LEVEL1
      END
```

```
C
      SUBROUTINE ZKMIC21(NCO,NDA,LDA,NIN,LIN,NOU,LOU,
     *                   A,LAA,
     *                   IOUT,LOUT,
     *                   ICM,LCM,ICMN,LCMN,
     *                   IECS,LECS,KSCS,LSCS,
     *                   KSS,LSS,NSCA,NSCB)
      DIMENSION NCO(1),NDA(1),NOU(1),NIN(1),
     *          A(1),
     *          IOUT(1),
     *          ICM(1),ICMN(1)
      DIMENSION NSCA(1)
C
      COMMON/COMM/ISZCOM,NOS,NSI,NX,NGI,IHEDR,MUXR,FMAX,
     *NTN,NSBLKF,NBLKF,NSBLKK,NBLKK,NPX,KP,MNPAD,ISTFLAG,
     *NOTIME,NMTIME,NOSMIG,THICK(31),DELZ(31),LEVC,
     *DTIME(31),NITER(31),VI(31),NOL
C
C
C-----------------------------------------------------------
C
C
C
C           COMPUTE ARRAY INDEXES
C              START OF F SLICE IFFT ARRAY
      IFFT=1
C              TRACE HEADER
      ITHDR=1
C              TRACE DATA
      IXD=ITHDR+64
C              VELOCITY TABLE
      IXV=IXD
C
      IWORK=IXD
C              WORK ARRAY NOSMIG LONG
      IDEP=IXD+NOSMIG+65
C
C              MUX - DMUX ARRAY
      MPX=IDEP+NOSMIG+2
C         DMUX ARRAY WHILE DOING F SLICE IFFT
      MPX2=IFFT+NPX*4+1
C
C
C     DO F SLICE IFFT
C
      KA=MPX+500
      IF(KA.LT.(MPX2+500)) KA=MPX2+500
      CALL RFLCM(KA)
      CALL RIFFT(NSCA,NSCA,A,NSBLKF,NBLKF,NSBLKK,
     *NBLKK,NPX,NX,NOSMIG,MUXR,IECS,KP,ICMN,ICM)
C
C          DO DMUX AND OUTPUT
C
      CALL BDMUX(NSCA(ITHDR),NSCA(IXD),A,IECS,KSCS,
     *A,ICM,IOUT,LOUT,ICMN)
      CALL REDUCE (2)
C
C
      CALL XEDCLOS(ICMN(105),KSS,ISTAT)
      CALL RELEASE
      RETURN
      END
```

```
C
C
*$TITLE                -RIFFT
*$NAME                 -RIFFT
*$ABSTRACT             -TO DO F SLICE IFFT,ON THE
*$                     -MULTIPLEX DATA
*$SAMPLE CALL          -CALL RIFFT(MSCA,FFT,A,NSBLKF,NBLKF,
*$                     -NSBLKK,NBLKK,NPX,NX,NOSMIG,MUXR)
*$CALLING PARAMETER
*$        -FFT         -FFT ARRAY
*$        -A           -ARRAY TO HOLD MPLX. DATA
*$        -NSBLKF      -SAMPLE/BLOCK F
*$        -NBLKF       -BLOCKS OF F
*$        -NSBLKK      -SAMPLES/BLOCK K
*$        -NBLKK       -BLOCKS OF K
*$        -NPX         -(NO. OF PAIRS IN F SLICE FFT)/2
*$        -NX          -NO. OF TRACES
*$        -NOSMIG      -NO. MIGRATED SAMPLES PER TRACE
*$        -MUXR        -LOGICAL FILE OF MPLX. DATA
*$        -KP          -NO. OF PAIRS IN K SLICE FFT
*$
      SUBROUTINE RIFFT(NSCA,FFT,A,NSBLKF,NBLKF,NSBLKK,
     *NBLKK,NPX,NX,NOSMIG,MUXR,IECS,KP,ICMN,ICM)
C
C
      DIMENSION A(1)
      DIMENSION ICM(1)
      DIMENSION ICMN(1)
      DIMENSION NSCA(1)
      DIMENSION IADSS(2)
      DIMENSION FFT(1)
      IADS=IECS+2000
C
C ------------------------------------------------------
C
      IADSS(1)=((IECS+ICM(1)-(NPX*4+500))/8)*8
      IADSS(2)=IADSS(1)+NPX*2+708
      JOB=1478
      IARG=ICM(32)
      IF(IARG.EQ.3) GO TO 610
      NNN=NPX*2
      CALL SINE (FFT,IADSS,NNN)
      NB=NNN
610   CONTINUE
C
C     COMPUTE CONSTANTS
C
      INOM=4LMUXR
      NN=NPX*2+1
      NSBF2=NSBLKF*2
      NPX4=NPX*4
      NSINC=NSBLKK*NSBLKF
C
C
C             READ UP FFTED F SLICE
      DO 1000 ISBLK=1,NBLKF
      MX=1
      IBLK=ISBLK
      DO 10 II=1,NBLKK
      CALL XEDREAD(ICMN(105),IADS+MX,IBLK,IBLK,0,ISTAT)
      CALL XEDWAIT(ICMN(105),ISTAT)
      IBLK=IBLK+NBLKF
      MX=MX+NSINC
```

```
10     CONTINUE
C
C
       NSBF=NSBLKF
       IC=NSBLKF*ISBLK
       IF(IC.GT.NOSMIG) NSBF=NSBF-(IC-NOSMIG)
       DO 150 I=1,NSBF
       MX=I
       CALL COLLIN(2,FFT,1,IADS+MX,NSBLKF,NPX4,1)
C
C              FOLD BEFORE IFFT
C                IFFT THE F SLICE
C
       S=1.0
       NNN=NPX*2
       IF(IARG.NE.3) GO TO 650
       CALL FOUR1(FFT,NNN,-1)
       GO TO 660
650    CALL FFOUR1(FFT,IADSS,NB,JOB,-1)
660    CONTINUE
C      COEF=SQRT(2./(NPX*KP*2))
C      IF(IARG.NE.3) GO TO 2001
C      DO 2000 IJJ=1,NX
C      FFT(IJJ)=FFT(IJJ)*COEF
C2000  CONTINUE
C
C2001  CONTINUE
C
C           MOVE F SLICE BACK INTO MPLX ARRAY
C
       CALL COLLIN(1,FFT(1),2,IADS+MX,NSBLKF,NX,1)
150    CONTINUE
C
C             WRITE F SLICE DATA BACK TO DISK
C
       IBLK=ISBLK
       MX=1
       DO 200 II=1,NBLKK
       IB=(II-1)*NSBLKK
       IF(IB.GT.NX) GO TO 999
       CALL XEDWRIT(ICMN(105),IADS+MX,IBLK,IBLK,0,ISTAT)
       CALL XEDWAIT(ICMN(105),ISTAT)
       IBLK=IBLK+NBLKF
       MX=MX+NSINC
200    CONTINUE
C
C
999    CONTINUE
1000   CONTINUE
C
       RETURN
       END
```

```
C
C
C
*$TITLE                 -BDMUX
*$NAME                  -BDMUX
*$ABSTRACT              -TO DO DEMULTIPLEX AND OUTPUT
*$SAMPLE CALL           -CALL BDMUX(THDR,A,B,IWORK)
*$CALLING PARAMETER
*$        -THDR         -TRACE HEADER ARRAY
*$        -A            -TIME CONVERTED OUTPUT ARRAY
*$        -B            -SCRATCH ARRAY
*$        -IWORK        -SCRATCH ARRAY
*$
      SUBROUTINE BDMUX(THDR,A,B,IECS,KSCS,IWORK,ICM,
     *IOUT,LOUT,ICMN)
C
      INTEGER Z
C
      COMMON/COMM/ISZCOM,NCS,NSI,NX,NGI,IHEDR,MUXR,FMAX,
     *NTN,NSBLKF,NBLKF,NSBLKK,NBLKK,NPX,KP,MNPAD,ISTFLAG,
     *NOTIME,NMTIME,NOSMIG,THICK(31),DELZ(31),LEVC,
     *DTIME(31),NITER(31),VI(31),NOL
C
      DIMENSION ICM(1)

      DIMENSION IOUT(1),ICMN(1)
      DIMENSION THDR(1),A(1),IWORK(1),B(1)
C
C
C
      IADS=IECS+2000
      IADSS=IECS+ICM(1)-10000
      INCM=4LMUXR
      INOM1=5LIHEDR
C
      JHDRI=0
      NTC=0
      NSBKF=NSBLKF*NSBLKK
C
C          READ UP NSBLKK SLICES OF K
      DO 1000 II=1,NBLKK
      IBLK=NBLKF
      Z=II-1
      IBLK=Z*IBLK+1
      MXX=1
      LONG=NSBKF
      IBLK2=IBLK+NBLKF-1
      CALL XEDREAD(ICMN(105),IADS+MXX,IBLK,IBLK2,0,ISTAT)
      CALL XEDWAIT(ICMN(105),ISTAT)
C
      DO 500 I=1,NSBLKK
      MXX=(I-1)*NSBLKF+1
      MX=1
      MXJ=MXX
      NTC=NTC+1
      IF(NTC.GT.NX) GO TO 1001
C
C
      CALL COLLIN(2,A(MX),NSBLKF,IADS+MXJ,NSBKF,NBLKF,
     *NSBLKF)
C
C    READ TRACE HEADER
C
```

```
      CALL RDFICH(INOM1,1,NTC,THDR,64,0)
      CALL ATTENTE(INOM1)
      DO 599 KILM=1,64
      IOUT(KILM)=INT(THDR(KILM))
599   CONTINUE
      DO 601 KILL=1,NOSMIG
      IOUT(64+KILL)=A(KILL)/(2.**IOUT(13))
601   CONTINUE
      IOUT(1)=NSI*NOSMIG
      IOUT(10)=NSI*NOSMIG
      CALL MAXR(IOUT(65),NOSMIG,IOUT(14))
      CALL LOWER(IOUT,NOSMIG,LEVC)
      CALL WTRACE(2,NTC,IOUT)
      CALL PRETE(2,NTC)
C
500   CONTINUE
1000  CONTINUE
C
1001  CONTINUE
      RETURN
      END
```