UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

Finite-Difference Migration by Optimized One-Way Equations

By Myung W. Lee[1] and Sang Y. Suh[2]

Open-File Report 85-289

[1]U.S. Geological Survey, Box 25046, MS 960, Denver Federal Center, Denver, Colorado 80225

[2]Korea Institute of Energy and Resources, 219-5, Garibong-dong, Guro-gu, Seoul, Korea.

1985

CONTENTS

ILLUSTRATIONS

TABLE

FINITE-DIFFERENCE MIGRATION BY THE OPTIMIZATION OF ONE-WAY EQUATIONS

By Myung W. Lee and Sang Y. Suh

ABSTRACT

The principle component of the finite-difference migration scheme is the approximation of a one-way wave extrapolation operator or a downward continuation operator. Usually, this approximation can be carried out either by a Taylor series expansion or a continued fraction method of an exact one-way extrapolation operator (or a "square root" operator). Due to the truncation of the expansion, the dispersion relation of the conventional one-way equation is very inaccurate, particularly for high propagation angles. This dispersion error controls the limit of the dip angle to be properly migrated by the finite-difference method.

In order to improve the dispersion relation, an optimization method was investigated. The basic concept of the optimization is to modify the coefficients of the conventional equation in such a way that the modified coefficients provide a better dispersion relation. This optimization can be implemented by minimizing the weighted dispersion error using a least-squares method.

This study shows that the optimization improves the dispersion relation of the one-way wave equation substantially. For example, within the relative dispersion error of 1 percent, the optimized second-order equation is reliable up to the propagation angle of 65 degrees; while the conventional second-order equation, the so-called 45 degree equation, is reliable up to 45 degrees. Since the only differences between the optimum and conventional one-way equations are the coefficients in the rational approximation of the square-root equation, the conventional finite-difference migration computer program can be modified easily in order to improve the performance of steep-dip migration.

INTRODUCTION

One of the main purposes of reflection seismology is to image the subsurface structure by measuring the reflected waves on the surface. To accomplish this objective, the data is subjected to computer processing. Seismic migration is a process of reconstructing the subsurface structure from the measured data and has received a great deal of attention within the last decade. There are three different approaches in modern seismic migration, i.e., the finite-difference method (Claerbout and Doherty, 1972; Claerbout, 1976), the Kirchhoff integral method (French, 1975; Schneider, 1978), and the F-K method (Stolt, 1978; Gazdag, 1978). All of the methods are based on the wave equation and are conveniently called wave-equation migrations.

Wave-equation migration is accomplished by two steps--downward extrapolation and imaging. The three methods of migration differ in their approach on the extrapolation of the wave field. The difference between depth migration and time migration is the imaging of the extrapolated wave field. Since imaging is simply a method of representing the wave field, the method of extrapolation is of great importance in most migration schemes.

The theory of wave extrapolation is based on the square root equation. The conventional wave equation represents waves which propagate in positive and in negative z-directions simultaneously. On the contrary, the square root equation represents waves propagating in one z-direction only. In laterally homogeneous media, exact wave extrapolation can be achieved by the square-root equation in the frequency-wavenumber domain.

In laterally heterogeneous media, wave extrapolations can be accomplished by use of the finite-difference method. In this method, a wavefield can be extrapolated by a one-way equation which is a rational approximation of the square-root equation. The one-way equation is further approximated by a difference equation in the actual computation. There are two methods of rationalizing the square-root equation. One is by a Taylor series and the other is by continued fractions (Hildebrand, 1956, p. 406). The Taylor series method may be called an explicit scheme while the continued-fractional method may be called an implicit scheme. The conventional 15-degree equation is the first-order approximation of the square root equation by either method. The 45-degree is the second-order approximation by the implicit scheme.

One of the disadvantages in the finite-difference method is that it cannot handle steep-dip structures. This is due to the inaccurate dispersion relation of the one-way equation used in the method. The inaccuracy comes from the truncation of the exact series expression of the square-root equation. Therefore, more terms must be used, i.e., migration by higher order equations. Berkhout (1980) and Gazdag (1980) used explicit high-order equations in their migration. An advantage of this method is the accuracy of the derivatives, which are evaluated by either a convolution in the space domain or by a multiplication in the wavenumber domain. This method alone cannot handle the steep-dip limitation satisfactorily, as is demonstrated in a later section. Besides, this method is numerically unstable (Gazdag and Sguazzero, 1984). Ma (1981) developed a more practical approach to finite-difference migration using the high-order implicit equations. In his method, the high-order equation is split into a series of low-order equations that are solved separately.

Berkhout (1979) studied the dispersion relations of the first-, second-, and third-order approximate equations and found that the dispersion error could be reduced by modifying the coefficients of the equations. His results suggest that the dip limitation can be avoided by using a higher order equation with modified coefficients. Ma (1981) attempted a similar procedure for the fifth-order implicit equation.

In the first section of this paper, the theory of wave extrapolation is briefly reviewed and the dispersion relations of one-way equations both explicit and implicit are analyzed. In the next section, optimization of the implicit one-way equations (using a least-squares method) to minimize the dispersion error is discussed. Finally, the optimized equations are tested on three examples: (1) extrapolation of a monochromatic wave, (2) migration of a synthetic model, and (3) migration of field data. Two appendices--the finite-difference formulation of the one-way equation and a version using only CPU (Central Processing Unit) of the migration program--are included.

ONE-WAY EQUATIONS

Wave-equation migration consists of two steps: extrapolation and imaging. The finite-difference extrapolation uses the one-way equation which is a rational approximation of the square-root equation. In this section, the basic concept of wave extrapolation is reviewed and the various one-way equations in terms of their dispersion relations are investigated.

The theory of wave extrapolation starts from an assumption that the wave field p(x, z, t) satisfies the two-dimensional scalar wave equation:

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial z^2} = \frac{1}{v^2} \frac{\partial^2 P}{\partial t^2} \tag{1}$$

where x is the horizontal distance, z is the depth, t is time, and v is the velocity. By introducing k_x and ω, which are the apparent horizontal wavenumber and the angular frequency, respectively, the wave field can be expressed by the following double Fourier transform:

$$p(x,z,t) = \sum_{k_x} \sum_{\omega} \hat{p}(k_x, z, \omega) \exp\left[i(k_x x + \omega t)\right]. \tag{2}$$

For the moment, we assume v is independent of x. Substituting equation (2) into equation (1) gives an ordinary differential equation:

$$\frac{d^2 \tilde{p}}{dz^2} = \left(-\frac{\omega^2}{v^2} + k_x^2\right) \tilde{p}. \tag{3}$$

Equation (3) has two solutions,

$$\tilde{p}(k_x, z+\Delta z, \omega) = \tilde{p}(k_x, z, \omega) \exp(\pm i k_z \Delta z) \tag{4}$$

where k_z is given by

$$k_z = \frac{\omega}{v}\left[1 - \left(\frac{v k_x}{\omega}\right)^2\right]^{1/2}. \tag{5}$$

One of the two solutions represents the wave propagating in the negative z-direction, while the other is in the positive z-direction. In order to extrapolate the wave field, we must choose only one of the two solutions for the following reason.

The wave equation is second order in z and has two independent solutions. Therefore, we need two boundary conditions to extrapolate waves in the z-direction. But we are given only one boundary condition in seismic migration, the wave observed on the surface. If the wave propagates in one direction only, we may discard the unnecessary solution in equation (4). This reduces the wave equation to the first order in z which can be solved with one boundary condition.

We choose the positive sign in equation (4) which implies that the wave propagating in the negative z-direction is considered. With the positive sign in equation (4), the extrapolation of the wave at $z = z_o$ to $z = z_o + \Delta z$ can be accomplished by the correcting the phase angle, given by $k_z \Delta z$.

This is the basic concept of the phase shift method of migration (Gazdag, 1978). The above approach indicates that the wave extrapolation in the (k_x, z, ω) domain might be represented by the following square-root equation,

$$\tilde{P}_z(k_x, z, \omega) = i \frac{\omega}{v} \left[1 - \left(\frac{v k_x}{\omega} \right)^2 \right]^{1/2} \tilde{P}(k_x, z, \omega) \tag{6}$$

where the subscript z represents the derivative with respect to that variable. Equation (6) is appropriate for the zero-offset migration. For the non-zero offset migration, a double square-root equation (Yilmaz and Claerbout, 1980) should be used.

The phase shift method is valid only for laterally homogeneous media. If the velocity changes horizontally, the extrapolation can be done in (x, z, ω) or in (x, z, t) domain with the one-way equation which is a rational approximation of equation (6), assuming that the logarithmic variation of velocity is less than that of the wave. There are two approaches in the approximation--explicit and implicit. The explicit method uses the Taylor series expansion while the implicit method uses continued fractions. Both of the methods require truncation of the exact expression. This is known as the paraxial approximation. Because of the truncation, the derived one-way equation is always different from the square-root equation. The difference may be represented by the dispersion relations. We investigate the dispersion relations of the one-way equations derived by both methods.

The simplest method of approximating the square-root expression of k_z given in equation (5) into k_x is the Taylor series method,

$$k_z = \frac{\omega}{v} \left(1 - \frac{1}{2} \frac{v^2 k_x^2}{\omega^2} - \frac{1}{8} \frac{v^4 k_x^4}{\omega^4} - \cdots \right) \tag{7}$$

Let us define $k_z^{(n)}$, the n-th order approximation of k_z, by taking the first n+1 terms in the series expression. The first-order approximation of k_z produces the following one-way equation,

$$\tilde{P}_z = i \frac{\omega}{v} \left(1 - \frac{1}{2} \frac{k_x^2 v^2}{\omega^2} \right) \tilde{P} \tag{8}$$

in the (k_x, z, ω) domain. By inverse-transforming equation (8), the one-way equation can be written in the (x, z, ω) domain as

$$P_z = i \frac{\omega}{v} \left(1 - \frac{1}{2} \frac{v^2}{\omega^2} \partial_{xx} \right) P. \tag{9}$$

This equation is known as a 15-degree equation. In the wave extrapolation, the wave P and its x-derivative are known, but the z-derivative is unknown. Equation (9) represents the unknown P_z in terms of the known. Therefore,

the equation is an explicit one-way equation. The second-order approximation of k_z produces the one-way equation in (x, z, ω) domain as

$$P_z = i \frac{\omega}{v} \left(1 + \frac{1}{2} \frac{v^2}{\omega^2} \partial_{xx} - \frac{1}{8} \frac{v^4}{\omega^4} \partial_{xxxx}\right) P \tag{10}$$

which also is an explicit one-way equation. Generally, a one-way equation derived from the Taylor series method is the explicit equation.

The accuracy of the approximate one-way equation is determined in terms of its dispersion relation. Let us consider a plane wave in the direction of θ with respect to the z-axis. The apparent wave numbers in x- and z-directions are

$$k_x = k \sin\theta \tag{11}$$

$$k_z = k \cos\theta \tag{12}$$

respectively, where k is the true wave number ω/v. The dispersion relation of the first-order approximate one-way equation is found by substituting equation (11) into equation (7) resulting in

$$k_z^{(1)} = k \left(1 - \frac{1}{2} \sin^2\theta\right). \tag{13}$$

Therefore, the dispersion error, $\Delta k_z^{(1)}$ is given by

$$\Delta k_z^{(1)} = k_z^{(1)} - k_z = k \left(1 - \frac{1}{2} \sin^2\theta - \cos\theta\right) \tag{14}$$

and is a function of the propagation angle θ and of the wavenumber k. The relative dispersion error is defined by the dispersion error normalized by the wavenumber, which is a function of only the propagation angle. For a reflector having dip angle ϕ, the propagation angle of the normal ray is also ϕ with respect to the z-axis. For successful migration, a one-way equation which is accurate up to the propagation angle ϕ should be used.

Let us define the relative dispersion error limit as 1 percent. Figure 1 shows the relative dispersion error of explicit one-way equations obtained by the Taylor series method. The numbers on the figure are the order of the approximation. The higher order approximation gives the more accurate dispersion relation. Gazdag (1980) and Berkhout (1980) used the explicit higher order equations for migration. The x-derivatives were computed by convolution in (x, z, ω) domain or by multiplication in (k_x, z, ω) domain.

This method is effective in reducing the numerical error resulting from the approximation of derivatives by differences. However, the methods were not effective for steep-dip migration. Figure 1 indicates that a sixteenth order equation or more should be used to migrate a 75-degree dipping structure.

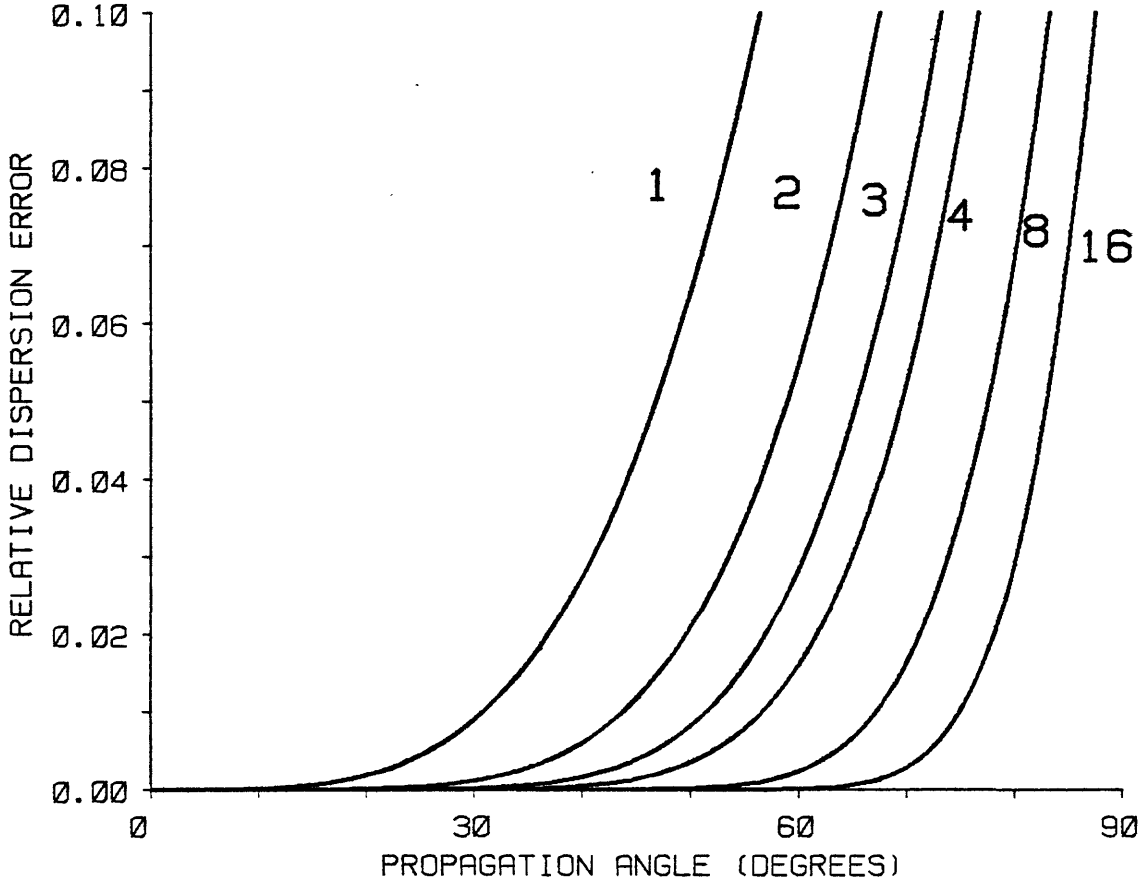

Figure 1.--Dispersion error in explicit one-way equations obtained by Taylor series approximation of the wave equation. Numbers annotated next to the curves indicate the order of approximation.

The second method of approximating k_z into k_x is the use of continued fractions. Let us define two new variables s and Y as

$$S = -\frac{v^2 k_x^2}{\omega^2} = -\frac{k_x^2}{k^2} \tag{15}$$

$$(1+S)^{1/2} = 1 + Y \tag{16}$$

respectively. This method is the approximation of Y by a rational function of s, resulting in a quadratic equation by squaring equation (16)

$$Y^2 + 2Y - S = 0 \tag{17}$$

The smallest root of equation (17) is found by successive approximations (Claerbout, 1976, p. 207). By using n-th approximation of Y, Y_n, and (n+1)-th approximation of Y, Y_{n+1}, equation (17) can be written by

$$Y_n Y_{n+1} + 2Y_{n+1} - S = 0 \tag{18}$$

This gives a recurrence relation,

$$Y_{n+1} = \frac{S}{2+Y_n} \tag{19}$$

Starting from $Y_0 = 0$, we get the first and second approximations of Y as

$$Y_1 = \frac{S}{2} \tag{20}$$

$$Y_2 = \frac{2S}{4+S} \tag{21}$$

respectively. The third and fourth approximations are

$$Y_3 = \frac{4S+S^2}{8+4S} \tag{22}$$

$$Y_4 = \frac{8S+4S^2}{16+12S+S^2} . \tag{23}$$

The first-order approximation of the one-way equation is obtained using equations (6), (15), (16), and (20), and is the same as the first-order equation obtained by the Taylor series method. The second-order equation by continued fractions is obtained using equation (21). In the (x, z, ω) domain, the equation is

$$P_z = i\frac{\omega}{v}\left(1 + \frac{2\partial_{xx}}{4k^2+\partial_{xx}}\right)P. \tag{24}$$

Equation (24) is known as a 45-degree equation. The third and fourth order equations by continued fractions are obtained by the same method. These are

$$P_z = i\frac{\omega}{v}\left(1 + \frac{4k^2\partial_{xx} + \partial_{xxxx}}{8k^4 + 4k^2\partial_{xx}}\right)P \tag{25}$$

$$P_z = i\frac{\omega}{v}\left(1 + \frac{8k^2\partial_{xx} + 4\partial_{xxxx}}{16k^4 + 12k^2\partial_{xx} + \partial_{xxxx}}\right)P. \tag{26}$$

Equation (24) contains an x-derivative term in its denominator. If the x-derivative is considered as the spatial convolution, the derivative in the denominator corresponds to the spatial deconvolution. Equation (24) may be converted to the convolutional form by multiplying both sides of the equation by $4k^2 + \partial_{xx}$. The result represents the unknown P_z in terms of P_{xxz} which is still unknown. Therefore, equation (24) is an implicit equation. For the same reason, equations (25) and (26) are implicit equations. Generally, a one-way equation (with an order greater than two) derived by the continued fraction method is an implicit equation. Stolt (1978) derived similar equations by transforming the wave equation into the floating-time coordinate and by successively approximating the z-derivatives.

Ma (1981) introduced a convenient method of solving the higher order implicit equations in which the higher order equation is split into a series of lower order equations. The general form of Y_{2n} is given by

$$Y_{2n} = \sum_{i=1}^{n} a_i s^i \Big/ \left(1 + \sum_{i=1}^{n} b_i s^i\right). \tag{27}$$

Equation (27) may be split into partial fractions as

$$Y_{2n} = \sum_{i=1}^{n} \frac{\alpha_i s}{1 + \beta_i s}. \tag{28}$$

Therefore, the 2n-th order implicit equation is given by

$$P_z = i\frac{\omega}{v}\left(1 + \sum_{i=1}^{n} \frac{\alpha_i \partial_{xx}}{k^2 + \beta_i \partial_{xx}}\right)P \tag{29}$$

in the (x, z, ω) domain. Application of Marzuk's splitting method (Mitchell, 1969) to equation (29) gives the following series of second-order equations and a phase correction equation,

$$\begin{aligned} P_z &= i\frac{\omega}{v}\frac{\alpha_m \partial_{xx}}{k^2 + \beta_m \partial_{xx}}P, \quad m = 1, 2, \cdots n \\ P_z &= i\frac{\omega}{v}P. \end{aligned} \tag{30}$$

Thus, wave extrapolation by the 2n-th order implicit equation is accomplished by solving the second-order equations n times and by applying the $k\Delta z$ phase correction. In this method, the computing time of the extrapolation is approximately proportional to the order of the equation. It is possible to apply the splitting method to the (2n-1)th order equation, but it has no advantage over 2n-th order equation. The computing time is almost the same but the dispersion relation is less accurate.

Figure 2 shows the relative dispersion error of one-way equations obtained by the continued fractions method. The numbers shown are the order of the approximation. The dispersion relation shown in this figure is more accurate than in figure 1. With the same order of approximation, the implicit equation is superior to the explicit. Still the implicit equation is not satisfactory. Even the eighth-order implicit equation shows more than 1 percent relative dispersion error for the propagation angle of 75 degrees. In order to get an implicit equation reliable up to 90 degrees propagation angle with 1 percent relative dispersion error, n must be 100 or more. Therefore, the problem of steep dip in finite-difference migration cannot be solved by using the higher order equations alone.

OPTIMIZATION OF ONE-WAY EQUATIONS

That the implicit equation better approximates the square-root equation than does the explicit equation has been shown. Furthermore, Berkhout (1979) modified the coefficients in second and third order implicit equations. In his method, the second and third order approximations of Y in equations (20) and (21) are modified as

$$Y_2 = \frac{1}{2} \frac{S}{1+a_2 S} \tag{31}$$

$$Y_3 = \frac{S}{2} \frac{1+a_3 S}{1+0.5 S} \tag{32}$$

respectively. Instead of the original coefficients $a_2 = a_3 = 0.25$, he used $a_2 = 0.279$ and $a_3 = 0.242$ respectively. The modified equations showed better dispersion relations than the originals. Ma (1981) introduced a modified version of his sixth-order equation, which corresponds to Y_5 in this paper. In this section, the optimization of one-way equations by modifying the coefficients is discussed.

The approximation of Y into Y_{2n} produces the error $E_{2n}(s)$ as

$$E_{2n}(s) = 1 + Y_{2n} - (1+S)^{1/2}. \tag{33}$$

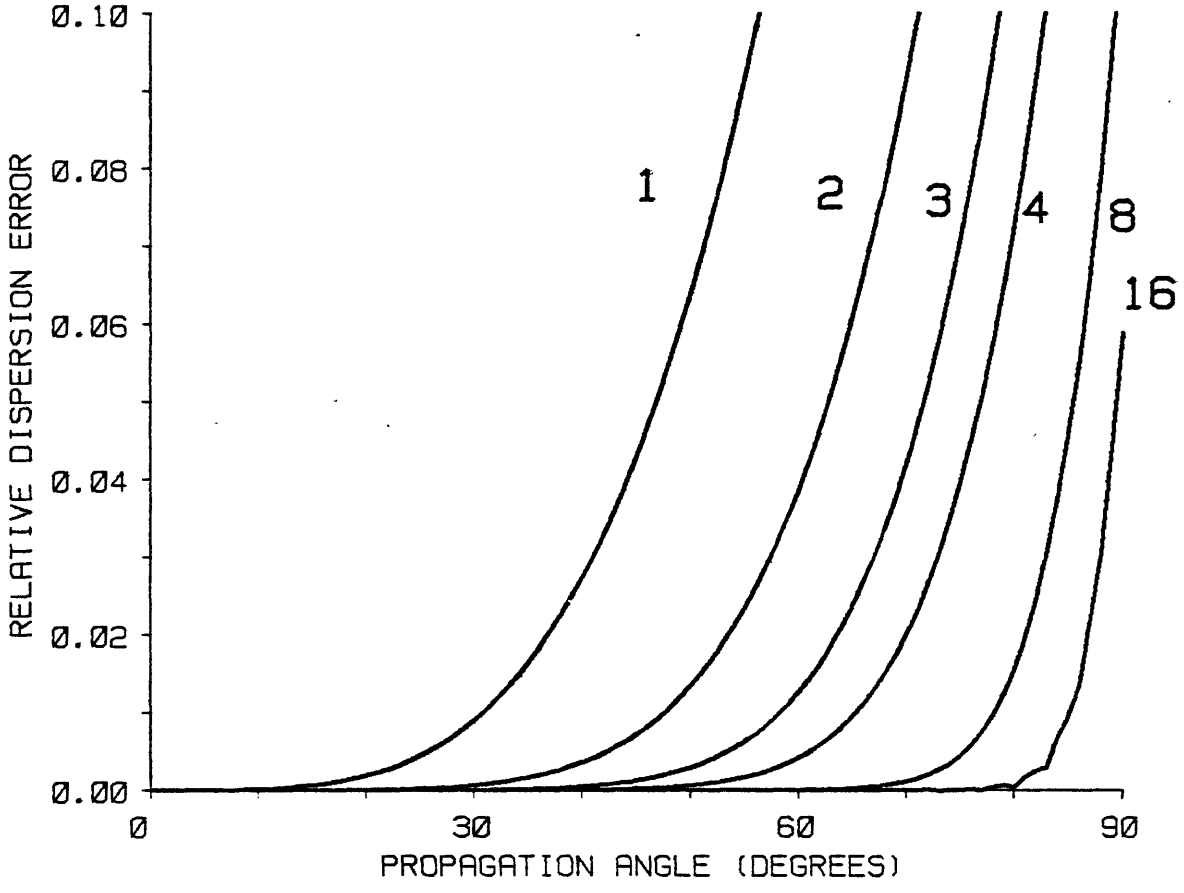

Figure 2.--Dispersion error in one-way equations obtained by the continued fractional approximation of the wave equation. Numbers annotated next to the curves indicate the order of approximation.

Substituting equation (27) into equation (33), and converting into the propagation angle θ gives

$$E_{2n}(\theta) = \sum_{i=1}^{n} a_i(-\sin^2\theta)^i \Big/ \left[1 + \sum_{i=1}^{n} b_i(-\sin^2\theta)^i\right] + 1 - \cos\theta. \tag{34}$$

The optimization procedure may be defined by finding coefficients a_i, b_i ($i = 1, 2, \ldots, n$) which minimizes the following integral,

$$J = \int_0^{\phi} E_{2n}^2(\theta)\, d\theta \tag{35}$$

where ϕ is the maximum optimization angle. The direct approach using equation (35) is very difficult and requires laborious numerical integration. Moreover, it turns out to be a nonlinear least-squares method. The authors tried to solve the problem by the Newton-Gauss method employing the iterative Taylor series expansion but the result was not satisfactory. The difficulty occurs when n is greater than 2, greater than the fourth-order equation, mainly because the solution is very sensitive to the initial guess required in the Newton-Gauss method.

To circumvent this difficulty, we used the weighted dispersion error. The weighted error E'_{2n} is defined as the result of E_{2n} times its denominator in equation (34),

$$E'_{2n}(\theta) = \sum_{i=1}^{n} a_i(-\sin^2\theta)^i + (1-\cos\theta)\left[1 + \sum_{i=1}^{n} b_i(-\sin^2\theta)^i\right] \tag{36}$$

The optimized coefficients a_i, b_i, ($i = 1, 2, \ldots, n$) can be found by minimizing the following integral

$$J' = \int_0^{\phi} \left[E'_{2n}(\theta)\right]^2 d\theta. \tag{37}$$

This is a linear least-squares method. The validity for using the weighted error can be justified by analyzing its result.

With the standard least-squares method, the optimization coefficients are the solution of the following normal equations,

$$\sum_{j=1}^{n} a_j \int (-\sin^2\theta)^{i+j} d\theta + \sum_{j=1}^{n} b_j \int (-\sin^2\theta)^{i+j}(1-\cos\theta)\, d\theta = \int (-\sin^2\theta)^i (1-\cos\theta)\, d\theta, \tag{38-a}$$

$$\sum_{j=1}^{n} a_j \int (-\sin^2\theta)^{i+j}(1-\cos\theta)\, d\theta + \sum_{j=1}^{n} b_j \int (-\sin^2\theta)^{i+j}(1-\cos\theta)^2 d\theta = \int (-\sin^2\theta)^i (1-\cos\theta)^2 d\theta, \tag{38-b}$$

for $i = 1, 2, \ldots, n$.

The integrals in the above equation can be calculated by the following integration table (Selby, 1971),

$$\int \sin^n x\, dx = -\frac{\sin^{n-1} x \cos x}{n} + \frac{n-1}{n}\int \sin^{n-2} x\, dx, \quad (39\text{-a})$$

$$\int \sin^n x \cos x\, dx = \frac{1}{n+1}\sin^{n+1} x, \quad (39\text{-b})$$

The solution of equation (38) gives the optimized one-way equation for each optimization angle θ. It is further split into a series of second-order equations, which may be solved by the conventional 45-degree migration algorithm.

The difficulty in the optimization is that equation (38) is ill-conditioned for the higher order equations. The double-precision computation employing 64 bits gives considerable error for optimization of the sixth-order and the eighth-order equations. Inversion of the ill-conditioned matrix is greatly affected by the number of significant digits. For this purpose, special Fortran subroutines are written, which compute addition, subtraction, multiplication, and division with arbitrary bits of precision. All of the computations for solving equation (38) is done by these subroutines.

Table 1 shows the optimized coefficients which are further split into a series of second-order equations given in equation (29). The sixth-order-or-less equations are computed with 100-bit precision; the eighth-order and tenth-order equations are computed with 200-bit and 300-bit precision, respectively. The first and second columns of the table represent the order of the equation and the maximum optimization angle. The third and fourth columns represent the numerator and denominator coefficients of the split equations.

The dispersion error of the optimized 2n-th order equation is given by

$$\Delta k_z = k\left(1 - \cos\theta - \sum_{i=1}^{n} \frac{d_i \sin^2\theta}{1 - \beta_i \sin^2\theta}\right). \quad (40)$$

The relative dispersion error is defined as Δkz divided by the wave number k. Figure 3 shows the relative dispersion errors of the various one-way equations. M_2 through M_{10} are the errors in the optimized equations, the subscripts of which represent the order. Y_2 is the error in the unmodified second-order equation, the conventional 45-degree equation. B_2 is the error in Berkhout's modified second-order equation. Berkhout's modified equation clearly is an improved version of the unmodified second-order equation. The optimized second-order equation (M_2) is better than B_2. Note that two coefficients are modified in M_2 but only one coefficient is changed in B_2.

Table 1.--Coefficients of optimized, fractioned one-way wave equations

Order	Ø	α_i	β_i
2	65	.478 242 060	.376 369 527
4	80	.040 315 157	.873 981 642
		.457 289 566	.222 691 983
6	87	.004 210 420	.972 926 132
		.081 312 882	.744 418 059
		.414 236 605	.150 843 924
8	90	.000 523 275	.994 065 088
		.014 853 510	.919 432 661
		.117 592 008	.614 520 676
		.367 013 245	.105 756 624
10	90	.000 153 427	.997 370 236
		.004 172 967	.964 827 992
		.033 860 918	.824 918 565
		.143 798 076	.483 340 757
		.318 013 812	.073 588 213

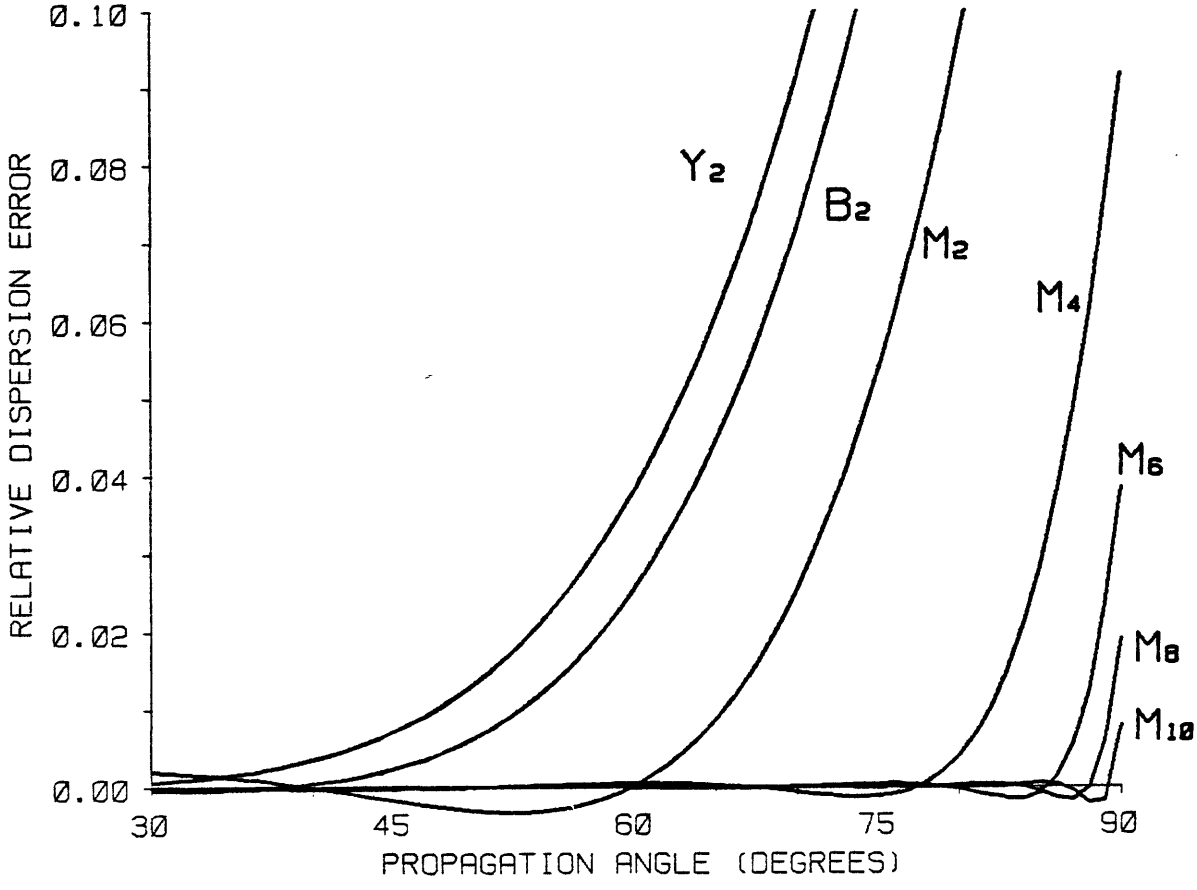

Figure 3.--Dispersion error in the optimized equations (M_2 through M_{10}), the conventional 45-degree equation (Y_2), and Berkhout's modified second-order equation (B_2).

The improvement of the optimization can be observed by comparing figures 2 and 3. Accepting the relative dispersion error limit of 1 percent, the second-order equation, which is accurate up to approximately 45 degrees in figure 2, is improved up to 65 degrees in figure 3. The fourth-order equation which is accurate up to 65 degrees in figure 2 is improved up to 82 degrees in figure 3. This behaviour can be observed in the eighth-order equation. The optimized tenth-order equation is accurate up to 90 degrees, which cannot be achieved by the 100th-order equation without the optmization. The validity for using the weighted error in the optimization procedure is established.

EXAMPLES

In this section, the optimized one-way wave equations are tested in three different examples. The first example is the extrapolation of a monochromatic cylindrical wave. The second example is the migration of a synthetic reflector model. The last example is the migration of field data.

Figure 4 is the result of monochromatic wave extrapolation by the finite-difference method. Figure 4A is computed by using the unmodified second-order equation. Figures 4B and 4C are computed by the optimized second- and fourth-order equation, respectively. The computational procedure is as follows. A two-dimensional (x, z) domain of 2,800 m by 1,200 m is divided by a regular grid having the intervals $\Delta x = \Delta z = 10$ m. With the origin on the upper left-hand corner of the domain, the grid index in x-direction (j) ranges from 0 to 280; while the grid index in z-direction (n) ranges from 0 to 120. A monochromatic wave source is located at $j = 80$ and $n = 0$. The wavelength is 100 m, which corresponds to the 30-Hz wave propagating through the 3-km/sec velocity medium. The theoretical solution of the wave is approximated by

(41)

where r is the distance from the source. The wave at the source is replaced by that of the adjacent grid point. Because of the spatial aliasing, the finite-diference method is not applicable near the source point. Therefore, the waves at $n = 1$, 2 and 3 are computed by equation (41). Waves at $n = 4$ through $n = 120$ are computed by the finite-difference method, the details of which are described in Appendix A. Figure 4 shows the real part of the wave field at even indices of j, $j = 0, 2, 4, \ldots, 280$.

The accurate wave extrapolation should be represented by a semicircular wavefront. The amplitude should be inversely proportional to the square root of the distance from the source. The wavefront in figure 4A is not a perfect semicircle. The amplitude in figure 4A does not decay properly particularly in a 60-degree deviation from the vertical. These effects are caused by the error in the dispersion relation. Figure 4B shows better results than figure 4A, and figure 4C is better than figure 4B. The optimized second-order equation is better than the original second-order equation. The higher order equation is superior to the lower order equation. Figure 4C, which is computed by the optimized fourth-order equation, shows almost perfect results except for a minor amplitude anomaly in the upper-right corner. The results are anticipated from the previous discussion on one-way equations.

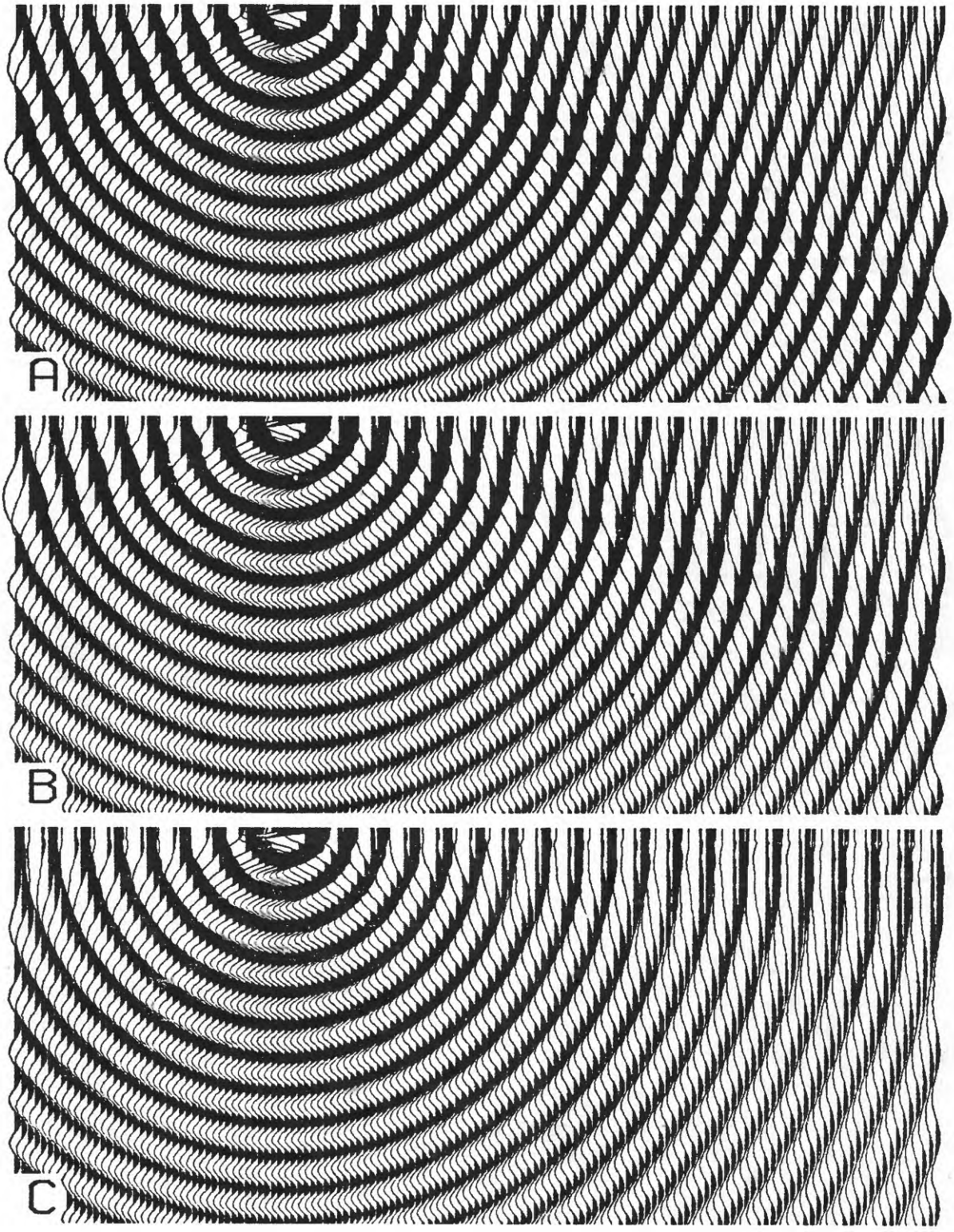

Figure 4.--Extrapolation of cylindrical wave by the conventional 45-degree equation (A), by the optimized second-order equation (B), and by the optimized fourth-order equation (C).

The result of extrapolation by the optimized sixth or higher order equation is almost the same as that by the optimized fourth-order equation. The minor amplitude error in the upper-right corner of figure 4C is not improved. This implies that factors other than the dispersion relation affect the high-angle extrapolation. It is apparent that one of the factors is the numerical error occurring from the derivative to the difference approximation. The split one-way equations resemble the second-order equation in the floating-time coordinate. In the floating-time coordinate, the apparent wave number in z-direction is less than the fixed-time coordinate for the propagation angle smaller than 60 degrees (Berkhout, 1980). If the propagation angle approaches 90 degrees, the zero apparent wavenumber in z-direction in the fixed-time coordinate increases to ω/v in the floating-time coordinate. The finite-difference approximation of P_z described in Appendix A requires 18 or more grid points per one wavelength to give less than 1 percent approximation error. To handle a higher propagation angle, a more elaborate difference scheme is required. Otherwise, the grid interval Δz should be reduced. Another factor affecting high-angle extrapolation is the initial condition. Equation (41) is an asymptotic solution which is valid only if distance from the source is far greater than the wavelength. The minimum distance in figure 4 is 30 m, i.e., 0.3 times the wavelength. There must be considerable error in the initial condition.

Figure 5 is the newly computed result of wave extrapolation with a finer grid interval and improved initial condition. Figure 5A is computed by the optimized fourth-order equation. Figures 5B and 5C are computed by the optimized sixth- and eighth-order equations, respectively. In the computation, Δz is reduced to 5 m. The grid index in z-direction ranges from 0 to 240. To avoid the inaccurate initial condition, the finite-difference extrapolation starts from n = 20. The waves at n = 0 through n = 19 are computed by equation (41). The minimum distance from the source is now extended to 100 m which is the same as the wavelength. It is found that figure 5B is better than 5A, and 5C is better than 5B. The result shows that it is possible, at least theoretically, that high-angle extrapolation can be accomplished by the optimized high-order equations. The result by the optimized tenth-order equation is amost the same as the result by the optimized eighth-order equation. Figure 5 represents the propagation angle of up to 87 degrees.

Figure 6 is a synthetic reflector model for the migration test. The reflector is asymmetrically W-shaped, the leftmost segment representing a 70-degree dip, and the rightmost segment representing a 60-degree dip, respectively. Two segments in the central part simulate a 45-degree dip. There are four different velocities in the model, i.e., 2 km/sec, 3 km/sec, 3.5 km/sec, and 4 km/sec. The horizontal dimension of the model is 12.75 km and the vertical dimension is 6 km. This is not a realistic model, but it was chosen to test the migration algorithm for the steeply dipping events in the heterogeneous medium.

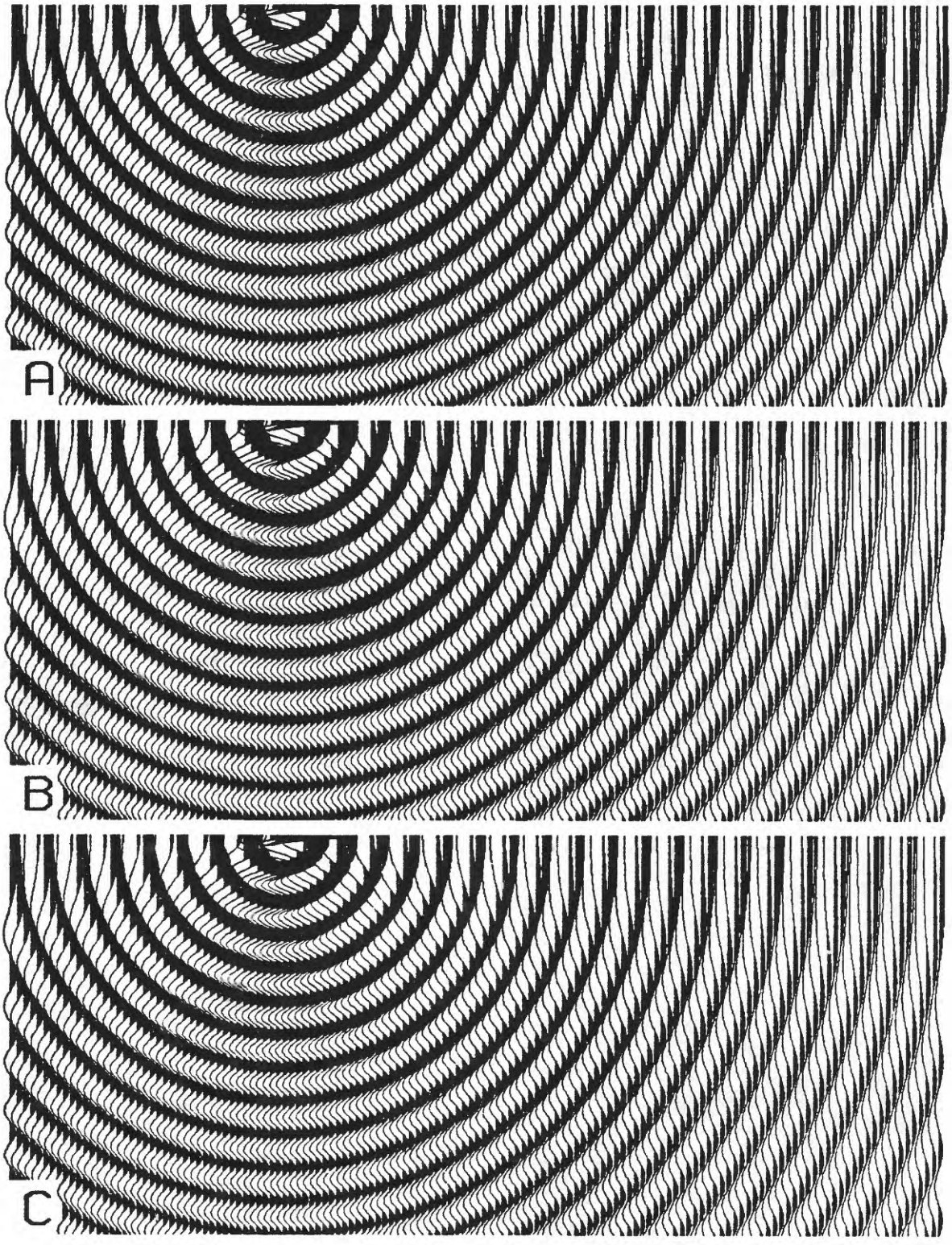

Figure 5.--Extrapolation of cylindrical wave with finer grid spacing by the fourth-order equation (A), sixth-order equation (B), and the eighth-order equation (C).

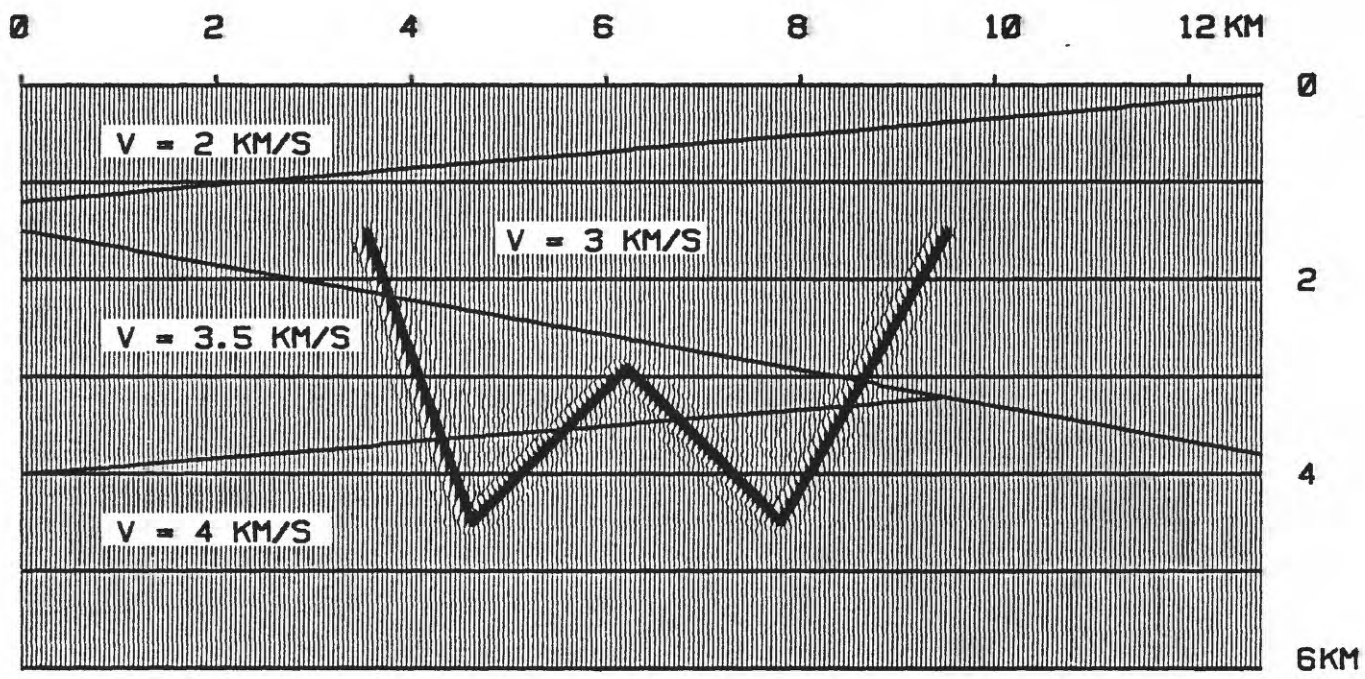

Figure 6.--A W-shaped reflector model simulating 45°, 60°, and 70° dips, simultaneously.

Figure 7 is a synthetic zero-offset section of figure 6. It was computed by the phase shift plus interpolation (PSPI) method (Gazdag and Sguazzero, 1984). In the computation, the grid interval is chosen as $\Delta x = \Delta z = 50$ m to give the number of grid points N_x and N_z by 256 and 121, respectively. The time increment Δt is 50 msec and the number of time samples N_t is 256. By taking N_x and N_t to the powers of two, the Fourier transforms required in the PSPI method is accomplished by the fast Fourier transform (FFT) routine. The PSPI method computes the diffraction which is difficult in the ray-tracing method. The dispersion relation of PSPI method approaches that of the exact square root equation as the number of reference velocities increases. The number of reference velocities used are four which is the same as that of the synthetic model. The computation should be accurate except in the region of the velocity boundary. The section shows a small amount of wrap-around effect, especially at the left-side boundary.

Figure 8 is the result of finite-difference migration of figure 7. Figure 8A is computed by the conventional 45-degree equation. Figures 8B and 8C are computed by the optimized second- and fourth-order equations, respectively. Figure 8A shows inaccurate migration except for the 45-degree dip reflector represented by the central part of W. Figure 8B shows better results than 8A. The 60-degree dipping segment on the right-hand side of W is almost accurately migrated in 8B. The 70-degree dipping segment is still distorted. Figure 8C, which is computed by the optimized fourth-order equation does not meet our anticipation satisfactorily. Although the 70-degree dipping segment is more accurately migrated in figure 8C than in 8B, it still shows a considerable amount of phase error and minor amplitude error. From the previous dispersion analysis, the equation should handle up to 80 degrees. The inaccuracy in the migration may be summarized as follows.

The amplitude error is primarily due to abrupt changes in velocity both in the synthesis of the zero-offset section and in the migration. The lateral velocity variation should be smooth to get an accurate synthetic section by the PSPI method. To make the velocity variation smooth, a high cut filter should be applied on the velocity boundary. This will give us many velocities in the filtered model, which require a number of reference velocities in the PSPI method. The computation time by the PSPI method increases almost linearly to the number of reference velocities. It took about 3,000 CPU seconds by the VAX 11/780 computer to produce figure 7 using four reference velocities.

The phase error, which is more significant than the amplitude error, in the migrated result in figure 8 is due to the numerical error occurring from derivative to difference approximation. In the migration, the spatial grid intervals Δx and Δz are 50 m. The time interval Δt is 50 msec. A 10 Hz signal propagating through the model having the average half velocity of 1,500 m/sec has a wavelength of 150 m. Therefore, the spatial grid interval of 50 m is too gross for the wavelength.

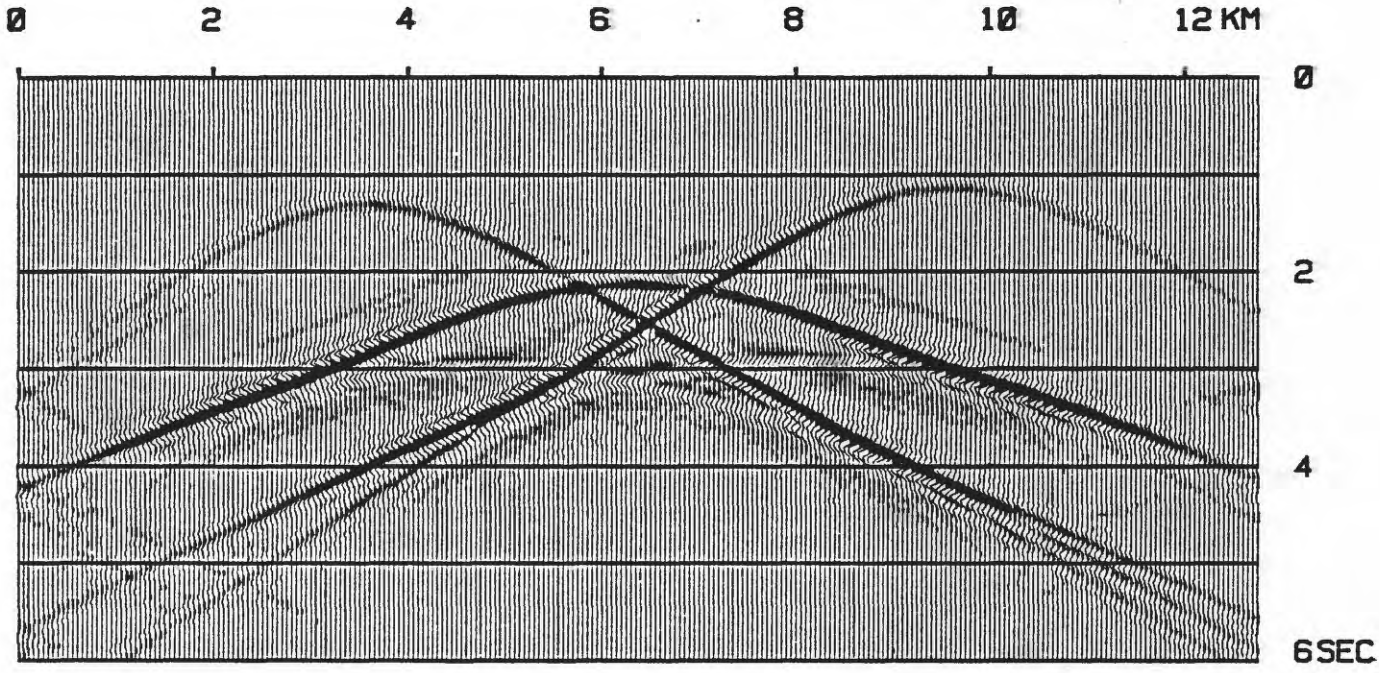

Figure 7.--Synthetic zero-offset time section of figure 6.

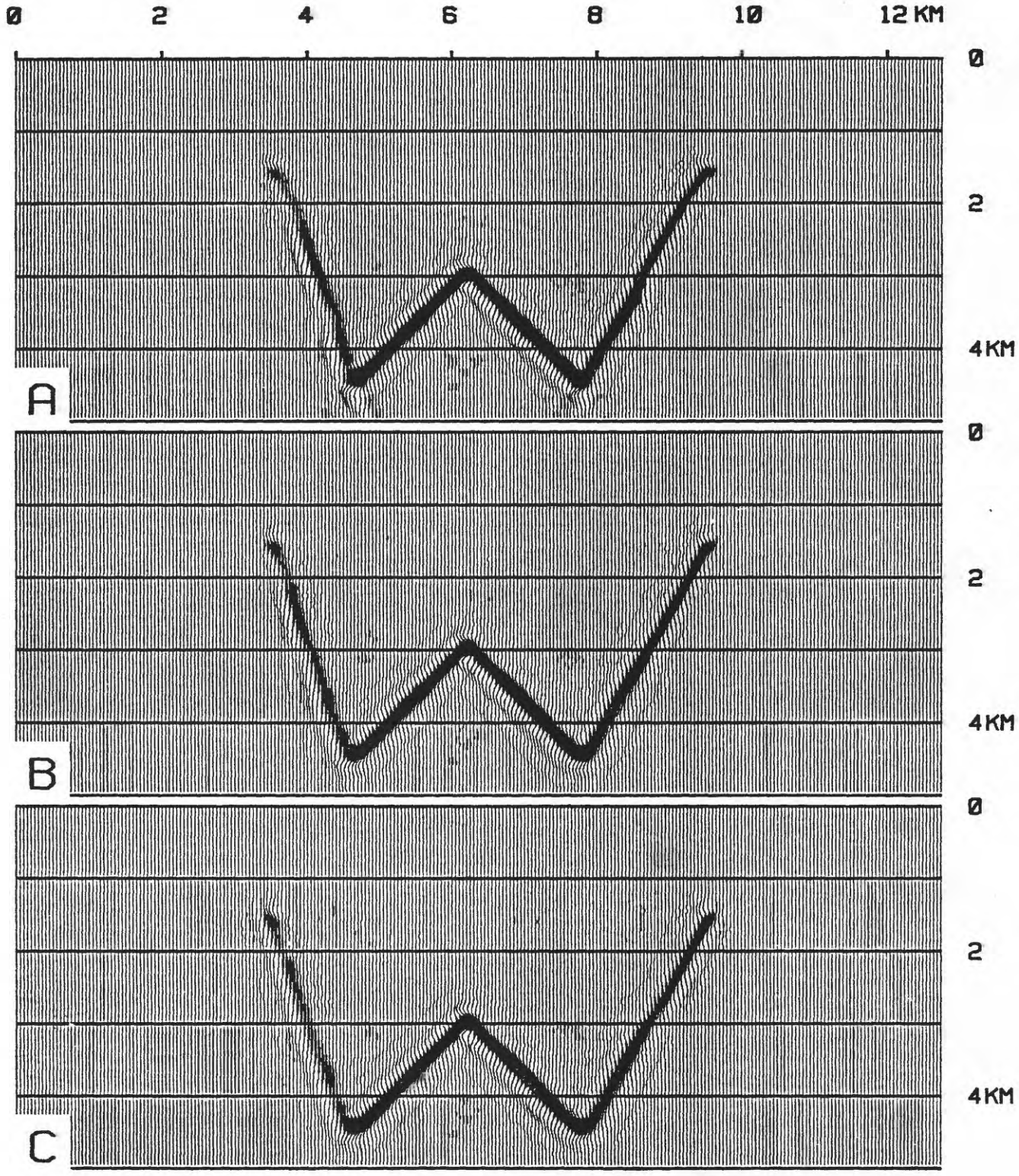

Figure 8.--Finite-difference migration of figure 7 by the conventional 45-degree equation (A), the optimized second-order equation (B), and the optimized fourth-order equation (C).

To accurately approximate the derivatives to differences within a 1 percent limit, the finite-difference method described in Appendix A requires about five grid points in x-direction and about eighteen grid points in z-direction, respectively. Therefore, the spatial grid intervals Δx and Δz should be reduced to less than 30 m and 10 m, respectively. The reduction gives us an increase in the number of grid points about ten times that of the previous grid points. It took approximately 2,640 CPU seconds by the VAX 11/780 computer to migrate figure 7 by the fourth-order equation with 50 m grid spacing. Migration with finer grid spacing will take hours of CPU time and days of clocktime by a time-sharing computer. The problem of computing time may be reconciled by using an array processor. The next example is actually computed by the array processor.

Figure 9 is a stacked section. The data were collected at Block 7 of the Korean continental shelf in 1980 and were processed using the seismic data processing facilities of the U.S. Geological Survey. The main hardware of the facility consists of a VAX 11/780 computer and FPS-120B array processors. The operating seismic software is DISCO (Digicon's Interactive Seismic Computer). The common depth point (CDP) interval of figure 9 is 25 m, the number of traces are 501, and the horizontal dimension is 12.5 km.

Figure 10 shows the migrated section of figure 9 by employing the optimized second-order equation. The stacking velocity produced by NMO analysis is directly used for the migration. As the stacking velocity is given in the time domain, the time migration algorithm is applied. The variable z in the depth migration is replaced by the two-way travel time of the image ray. The extrapolation step of the migration Δt is 48 msec. The maximum frequency is 40 Hz. To speed up the computation, almost all of the migration algorithm is written in the assembler language of the array processor. The computing time of figure 10 is 59 CPU seconds and 271 AP seconds.

Figure 9 contains considerable noise in the lower half of the section. The noise is primarily considered as the P-S converted wave. Therefore, it is hardly expected that a good migrated result in that portion will be achieved. On the contrary, the upper half of the section shows relatively good signal-to-noise ratio. The effect of migration is well observed in figure 10. Several diffractions are nicely focused. The bowtie at 5 km and on 1.3 sec in figure 9 is successfully migrated to give the syncline structure in figure 10.

CONCLUSION

One of the disadvantages in the finite-difference migration is that it cannot handle a steep-dip structure. This is due to the inaccurate dispersion relation of the one-way wave equation utilized in the scheme. In this paper, an optimization method is studied in order to improve the dispersion relation of the implicit one-way wave equations. The basic concept of the optimization is modifying the coefficients, which has been attempted by Berkhout (1979) and Ma (1981). In our optimization, all of the adjustable coefficients are subject to the modification, i.e., the number of optimzed coefficients is the same as the order of the one-way equation.

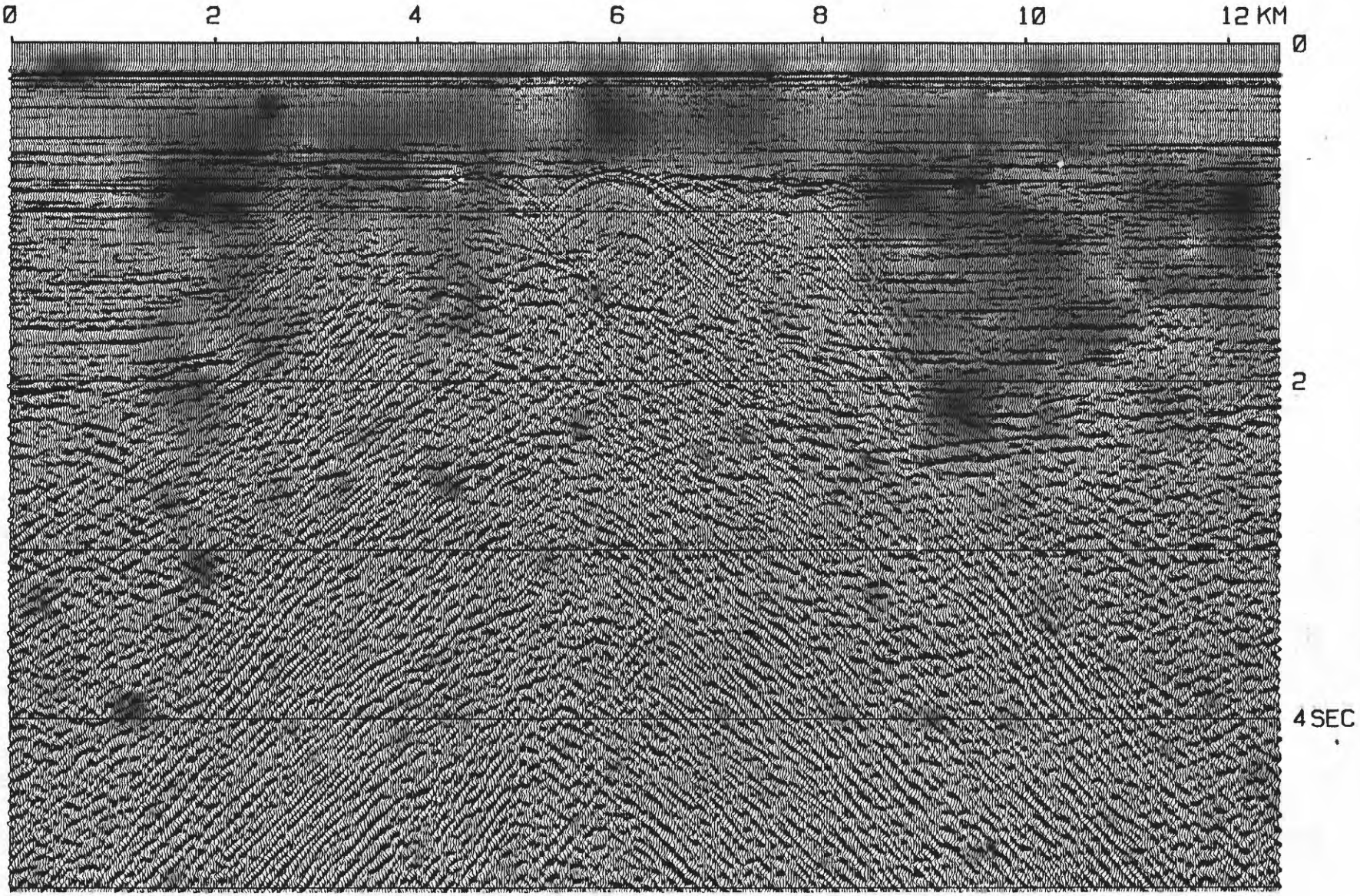

Figure 9.--CMP stack of a Korean offshore line.

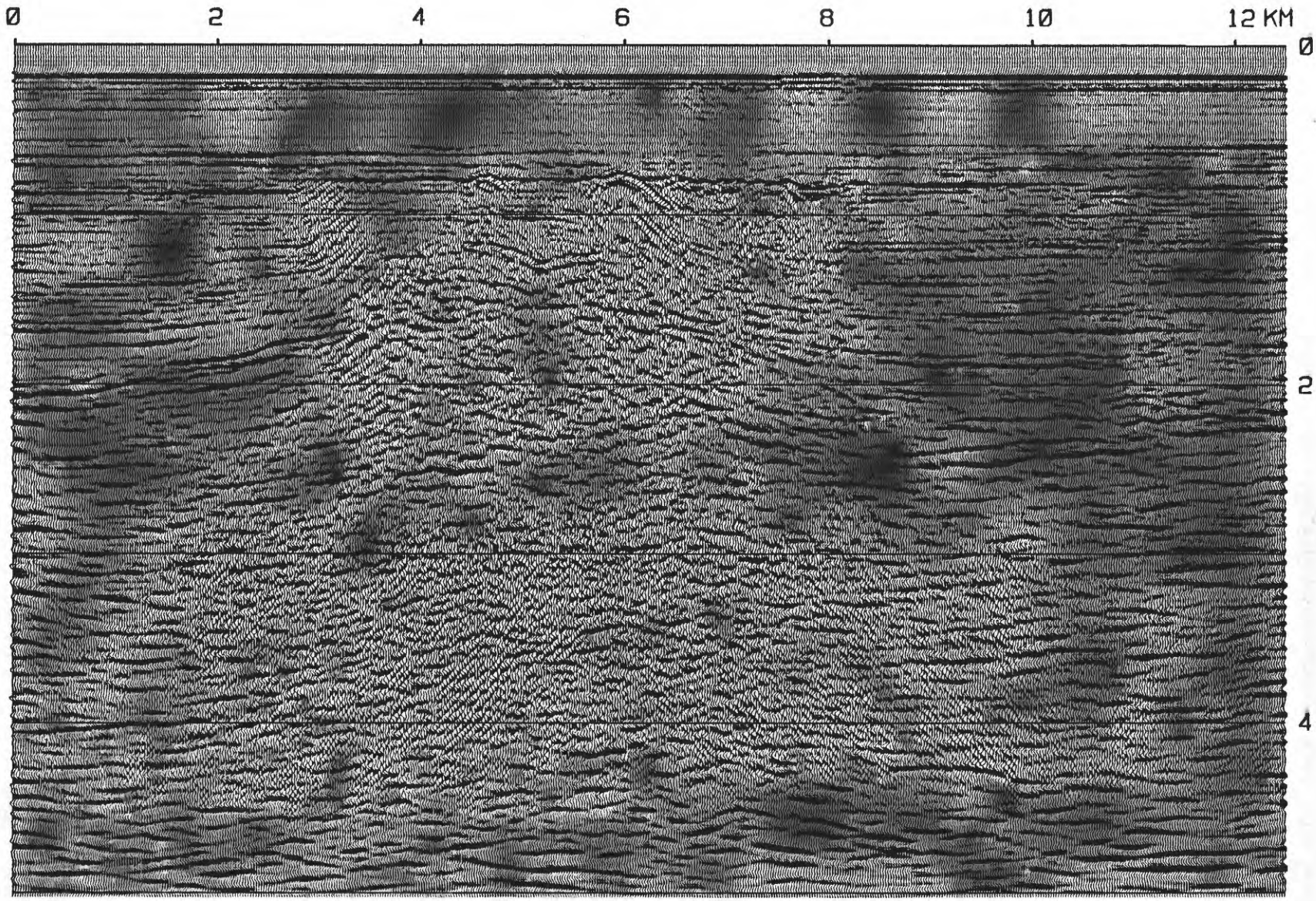

Figure 10.--Finite-difference time migration of figure 9 by the optimized second-order equation.

A least squares method, which minimizes the weighted dispersion error, is investigated. By introducing the weighted error, the least-squares method is reduced to the linear problem. Optimized coefficients are computed for up to the tenth- order equation. Each of the optimized equations shows significantly improved dispersion relation over the corresponding unoptimized equation. Using the optimized equations, the steep-dip migration can be achieved accurately without going to the higher order equation.

The optimized equations are tested on three examples. The first example is on the extrapolation of the monochromatic cylindrical wave. In the example, the optimized second-order equation shows better results than the unoptimized second-order equation which is known as the 45-degree equation. The computing times by the two equations are exactly the same. The optimized higher order equations show substantially better results than the lower order equations. The second example which is on the migration of a synthetic model, shows similar results. A stack section is migrated by the optimized second-order equation. The result is considered as acceptable.

The extrapolation of high-angle waves requires suppression of numerical error occurring from the derivative to the difference approximation as well as the one-way equations having accurate dispersion relation up to that angle. One of the methods of suppressing the numerical error is by working with finer grid intervals. Use of a finer grid interval will result in many hours of computing time. A method should be developed to control the numerical error effectively.

REFERENCES CITED

Berkhout, A. J., 1979, Steep dip finite-difference migration: Geophysical Prospecting, v. 27, p. 196-213.

Berkhout, A. J., 1980, Seismic Migration: Amsterdam, Elsevier Publishing, 339 p.

Claerbout, J. F., 1976, Fundamentals of Geophysical Data Processing: New York, McGraw Hill, 274 p.

Claerbout, J. F., and Doherty, S. M., 1972, Downward continuation of moveout corrected seismograms: Geophysics, v. 37, p. 741-768.

Clayton, R. W., and Engquist, V., 1980, Absorbing boundary conditions for wave-equation migration: Geophysics, v. 45, p. 895-904.

French, W., 1975, Computer migration of oblique seismic reflection profiles: Geophysics, v. 40, p. 961-980.

Gazdag, J., 1978, Wave equation migration with the phase shift method: Geophysics, v. 43, p. 1342-1351.

Gazdag, J., 1980, Wave equation migration with the accurate space derivative method: Geophysical Prospecting, v. 28, p. 60-70.

Gazdag, J., and Sguazzero, P., 1984, Migration of seismic data by phase shift plus interpolation: Geophysics, v. 49, p. 124-131.

Hildebrand, F. B., 1956, Introduction to numerical analysis: New York, McGraw Hill, 511 p.

Ma, Zaitian, 1981, Finite-difference migration with higher-order approximation: Presented at the 1981 joint meeting of China Geophysical Society and Society of Exploration Geophysicists, Peking, China, 14 p.

Mitchell, A. R., 1969, The finite difference methods in partial differential equations: New York, J. Wiley and Sons, 272 p.

Schneider, W. A., 1978, Integral formulation for migration in two and three dimensions, Geophysics, v. 43, p. 49-76.

Selby, S. M., 1971, Standard Mathematical Tables: Cleveland, Chemical Rubber Company, 710 p.

Stolt, R. H., 1978, Migration by Fourier transform: Geophysics, v. 43, p. 23-48.

Yilmaz, O., and Claerbout, J. F., 1980, Prestack partial migration: Geophysics, v. 45, p. 1753-1779.

APPENDIX A. Finite-Difference Formulation of the One-Way Equation.

A 2n-th order implicit equation is split into n second-order implicit equation. Therefore, it is sufficient to describe the solution method of the second order equation which is given by

$$P_z = i \frac{\omega}{v} \frac{\alpha \partial_{xx}}{k^2 + \beta \partial_{xx}} P. \tag{A-1}$$

A two-dimensional wave field is defined on (x, z) domain. The domain is divided into a regular grid, the intervals of which are Δx and Δz. The wave field at an intersection, $P(j\Delta x, N\Delta z)$ is denoted by P_j^n. The problem of extrapolation is to find wave $P_j^{n+1}(j = 0, 1, \ldots, J)$ from the given $P_j^n(j = 0, 1, \ldots, J)$. The finite-difference method employs the difference approximation of equation (A-1).

The z-derivative in equation (A-1) is approximated by

$$\partial_z \approx \frac{2}{\Delta z} \frac{\delta_z}{\gamma_z} \tag{A-2}$$

where

$$\delta_z f(z) = f(z+\Delta z) - f(z)$$
$$\gamma_z f(z) = f(z+\Delta z) + f(z)$$

This is an expression of Crank-Nicolson's scheme of the finite-difference method. The x-derivative in equation (A-1) is approximated by

$$\partial_{xx} \approx \frac{1}{(\Delta x)^2} \frac{\delta_{xx}}{1 + 1/12 \, \delta_{xx}} \tag{A-3}$$

where

$$\delta_{xx} f(x) = f(x+\Delta x) - 2 f(x) + f(x-\Delta x).$$

Equation (A-2) is accurate within 1 percent if 18 or more grid points exist per one wavelength. Equation (A-3) is accurate within 1 percent if 5 or more grid points exist per one wavelength (Claerbout, 1976). Substituting equations (A-2) and (A-3) to equation (A-1) gives

$$2 \delta_z \left[k^2 \Delta x^2 + (\beta + k^2 \Delta x^2/12) \delta_{xx} \right] P_j^n = i k \alpha \Delta z \gamma_z \delta_{xx} P_j^n \tag{A-4}$$

Expanding equation (A-4) in z index gives

$$2\left[k^2 \Delta x^2 + (\beta + k^2 \Delta x^2/12) \delta_{xx}\right] (P_j^{n+1} - P_j^n) = i k \alpha \Delta z \delta_{xx} (P_j^{n+1} + P_j^n). \tag{A-5}$$

Expanding equation (A-5) in x index gives

$$A_j P_{j-1}^{n+1} + B_j P_j^{n+1} + C_j P_{j-1}^{n+1} = D_j \tag{A-6}$$

where

$$A_j = 2\left(\beta + k^2\Delta x^2/12\right) - ik\alpha\Delta z$$

$$B_j = 2\left[k^2\Delta x^2 - 2\left(\beta + k^2\Delta x^2/12\right)\right] + 2ik\alpha\Delta z$$

$$C_j = A_j$$

$$D_j = A_j^* P_{j-1}^n + B_j^* P_j^n + C_j^* P_{j+1}^n ,$$

for $j = 1, 2, \ldots, J-1$,

where A_j^*, B_j^*, and C_j^* indicate the complex conjugate of A_j, B_j, and C_j, respectively.

Equation (A-6) gives J-1 equations for J+1 unknowns. The remaining two equations are given from the boundary conditions. A conventional boundary condition such as Dirichlet or Neumann is subject to artificial reflections at the boundary. To suppress the artificial reflections, Clayton and Engquist (1980) suggest

$$P_z = i\frac{\omega}{v}\,\frac{a + b\partial_x/k}{1 + c\partial_x/k}\,P \qquad \text{(A-7)}$$

in the fixed-time coordinate. The coefficients a, b, and c are determined so that the dispersion relation of equation (A-7) approximates the quarter circle. The boundary condition for equation (A-1) may be in the form of

$$P_z = i\frac{\omega}{v}\,\frac{d\,\partial_x}{k + c\,\partial_x}\,P \qquad \text{(A-8)}$$

where the coefficients c and d are determined as the dispersion relation approximates to one-half that of the interior. If it approximates $+k_x$ axis, it is used at the +x boundary; if it approximates $-k_x$ axis, it is used at the -x boundary. The coefficients are found by the least squares method which is very similar to the optimization of one-way equations.

The finite-difference formulation of equation (A-9) uses equation (A-2). The formulation at x = 0 boundary is

$$\left(k\Delta x + 2c\,\delta_x\right)\delta_z P_o^n = ikd\Delta z\,\delta_x\gamma_z P_o^n \qquad \text{(A-9)}$$

Expanding equation (A-9) in the z index gives

$$\left(k\Delta x + 2c\,\delta_x\right)\left(P_o^{n+1} - P_o^n\right) = ikd\Delta z\,\delta_x\left(P_o^{n+1} + P_o^n\right) \qquad \text{(A-10)}$$

Expanding equation (A-10) in x index gives

$$B_o P_o^{n+1} + C_o P_1^{n+1} = D_o \qquad \text{(A-11)}$$

where

$$B_0 = k\Delta x - 2C + ikd\Delta z$$

$$C_0 = k\Delta x + 2C - ikd\Delta z$$

$$D_0 = B_0^* P_0^n + C_0^* P_0^n$$

The finite-difference equation at $x = J\Delta x$ boundary can be derived by changing the subscripts as 0 to J and 1 to J-1, and the coefficients C to A as

$$A_J P_{J-1}^{n+1} + B_J P_J^{n+1} = D_J \qquad \text{(A-12)}$$

where

$$A_J = k\Delta x + 2C - ikd\Delta z$$

$$B_J = k\Delta x - 2C + ikd\Delta z$$

$$D_J = A_J^* P_{J-1}^n + B_J^* P_J^n .$$

Equation (A-11) is added to the top of equation (A-6) and equation (A-12) is appended to the bottom of equation (A-6). Introducing vectors $\underline{P}^{n+1}$ and $\underline{D}$ which represent P_j^{n+1} and $D_j (j = 0, 1, \ldots, J)$, the result will be

$$\underline{\underline{M}}\ \underline{P}^{n+1} = \underline{D} \qquad \text{(A-13)}$$

where matrix $\underline{\underline{M}}$ is given by a tridiagonal matrix as

$$\underline{\underline{M}} = \begin{bmatrix} B_0 & C_0 & & & \text{Zero} \\ A_1 & B_1 & C_1 & & \\ & A_2 & B_2 & C_2 & \\ & & \ddots & \ddots & \ddots \\ & & & A_{J-1} & B_{J-1} & C_{J-1} \\ \text{Zero} & & & & B_J & C_J \end{bmatrix}$$

The solution to equation (A-13) may be found by introducing an auxilliary equation

$$P_{J-1}^{n+1} = E_J P_J^{n+1} + F_J P_{J+1}^{n+1} . \qquad \text{(A-14)}$$

Coefficients E_j and F_j are found by comparing the coefficients to equation (A-13). The unknowns are then computed by backward substitution (Claerbout, 1976).

APPENDIX B. CPU Version of the Migration Program

This program computes a depth-migrated section from a zero-offset seismic section. The name of the program is FXZMIG. It is written in Fortran 77 language. Five files are referenced by the program. The first file is named by FXZMIG.DAT and is intended for reading the input parameters only. A more thorough description on the parameters may be found later. The second file is the trace file containing the zero-offset seismic section. The format of the file is UNFORMATTED, i.e., the information is represented by the internal format. Each trace consists of one record. There is no header in the trace, i.e., the file should contain amplitude information only. The third file is the velocity file containing the migration velocities in (x, z) domain. Like the trace file, it is UNFORMATTED and has no header. The velocity should be defined at every trace and at every depth step. The fourth file is named by FXZMIG.OUT and will contain the migrated result in (x, z) domain after the successful run of the program. The last file is a scratch file intended for the program internally.

Four lines of input parameters are required for the program to run. The first line defines six parameters: NX, NZ, NT, DX, DZ, and DT. NX is the number of traces; NZ, the number of depth steps; NT, the number of time samples; DX and DZ, the spatial intervals in meters; and DT, the temporal interval in msec. The second line defines two parameters, NESTDR and WPCNT. NESTDR represents the order of the one-way equation which will be used for the migration. The number indicates half of the order, i.e., 1 is the optimized second order, 2 is the optimized fourth order, and so on. It should not exceed 5. If NESTDR is less than 1, the conventional 45-degree equation will be used for the migration. WPCNT is a parameter for the numerical dip-filter. If it is 0, no dip-filter will be applied. The dip-filter algorithm may be found in Claerbout (1976).

The third line describes two parameters, SW1 and SW4, i.e., the lowcut and highcut frequencies, respectively, in Hz. The highcut frequency should not exceed Nyquist frequency. The fourth line defines the name of two input files, i.e., the velocity file and the trace file. All the lines should be physically separated by a carriage return for an interactive job. For a batch job, the lines must be on separate cards. The parameters within a line should be separated by a comma and/or spaces, i.e., in free format of the Fortran language. Each file name should be enclosed in a pair of apostrophies. The following example shows the input parameters used in the computation of figure 8C.

Example of the input parameters

```
256          120       130       50        50        50
  2            0
  1            9

'VEL008', 'MDLPHS.OUT'
```

The program statements are as follows.

```
C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C       PROGRAM   **   F X Z M I G   **   VERSION HG.3   SEP/28/84
C
C       FINITE-DIFFERENCE WAVE EQUATION MIGRATION IN (F,X,Z) DOMAIN
C
C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
        PROGRAM  FXZMIG
C
C       INCLUDE   'FXZMIG.CMB/LIST'
C
C       CONETENTS OF FILE FXZMIG.CMB
C
        COMMON / BLK1 /
     1  NX,              !  NO. OF TRACES,
     2  NZ,              !  NO. OF SAMPLES
     3  NT,              !  NO. OF TIME SAMPLES,
     4  DX,              !  TRACE INTERVAL,
     5  DZ,              !  Z-STEP INTERVAL,
     6  DT,              !  SAMPLING TIME,
     7  NX2,NY,NX1,      !  NX*2, NX-2, NX-1,
     8  NTFFT,           !  FFT LENGTH IN TIME,
     9  NW,DW,           !  NO. OF FREQUENCIES, AND INCREMENT,
     A  SW0,SW1,SW4,     ! START(SW0,SW1) AD END(SW4) FRQUENCIES
     B  WIMAG,AMIMAG     ! DIP-FILTER PARAMETERS
C
        COMMON / BLK2 /
     1  EXTBUF(12,5),    !  EXTRAPOLATION MATRIX,
     2  BNDBUF(2,5),     !  BOUNDARY MATRIX
     3  NESTDR           !  ORDER OF THE NEST
C
        CHARACTER*56    FILVEL, FILTRC
        COMMON / BLKF / FILVEL, FILTRC
C
        PARAMETER
     1  LDVVEL  =  11,          !  VELOCITY FILE
     2  LDVTRC  =  12,          !  TRACE FILE
     3  LDVMIG  =  13           !  MIGRATED OUTPUT
CCC
C       END OF FILE FXZMIG.CMB
C
        DIMENSION  BUF (320 000)
        DATA  MAXBFS  / 320 000 /
C
C
        CALL GETCRD
        CALL ALCMEM (MAXBFS, IV, IM, JM, IA, IB, IC, ID, IQ)
        CALL EDTVEL (BUF)
        CALL TM2FRQ (BUF, BUF(IQ))
        CALL EXTCON
        CALL MLTSTP (BUF(IV), BUF(IM), BUF(JM), BUF(IA), BUF(IB),
     +                    BUF(IC), BUF(ID), BUF(IQ))
        CALL OUTPUT (BUF)
        STOP    'NORMAL COMPLETION'
        END
```

```
      SUBROUTINE GETCRD                                                 57
C                                                                       58
C     READ INPUT DATA FROM CARD.                                        59
C                                                                       60
      INCLUDE   'FXZMIG.CMB/NOLIST'                                     61
C                                                                       62
C     READ/WRITE SECTION.                                               63
C                                                                       64
      OPEN (UNIT = 5, FILE = 'FXZMIG', TYPE = 'OLD', READONLY)          65
      WRITE (6, 60)                                                     66
      READ  (5,  *)  NX, NZ, NT, DX, DZ, DT                             67
      WRITE (6, 62)  NX, NZ, NT, DX, DZ, DT                             68
      READ  (5,  *)  NESTDR, WPCNT                                      69
      WRITE (6, 64)  NESTDR, WPCNT                                      70
      READ  (5,  *)  SW1, SW4                                           71
      WRITE (6, 66)  SW1, SW4                                           72
      READ  (5,  *)  FILVEL, FILTRC                                     73
      WRITE (6, 68)  FILVEL, FILTRC                                     74
      CLOSE   (5)                                                       75
C                                                                       76
C     FORMAT SECTION                                                    77
C                                                                       78
 60   FORMAT (1H1///35(' *')//                                          79
     1          ' WELCOME TO   *** F X Z M I G ***'5X'VERSION  HG.3'/   80
     2          ' FINITE-DIFFERENCE WAVE EQ. MIGRATION IN (F,X,Z)',    81
     3          ' DOMAIN'// 35(' *')//                                 82
     4          ' INPUT DATA SUMMARY'/)                                83
 62   FORMAT (T9'NX'T19'NZ'T29'NT'T38'DX'T49'DZ'T59'DT'/                 84
     +          3I10, 3F10.1)                                          85
 64   FORMAT ('0'T5'NESTDR'T16'WPCNT'/ I10,F10.1)                       86
 66   FORMAT ('0'T8'SF1'T18'SF4' / 2F10.1)                              87
 68   FORMAT ('0 FILVEL =',2X,A/'  FILTRC =',2X,A)                      88
C                                                                       89
C     PARAMETER EXAMINATION SECTION                                     90
C                                                                       91
      NX2      =  NX  +  NX                                             92
      NY       =  NX  -  2                                              93
      NX1      =  NX  -  1                                              94
      DT       =  DT     /  1000.                                       95
      NTFFT    =  KPOWR2 (NT)                                           96
      FNYQ     =  0.5 /  DT                                             97
C                                                                       98
      IF (SW1.GE.SW4 .OR. SW4.GT.FNYQ) STOP 'ERROR IN SFRQ'             99
C                                                                      100
C                                                                      101
      PI       =  ACOS (-1.)                                           102
      DW       =  2.  *  PI  /  (NTFFT  *  DT)                         103
      SW1      =  2.  *  PI  *  SW1                                    104
      SW4      =  2.  *  PI  *  SW4                                    105
C                                                                      106
      IW1      =  SW1  /  DW  +  1.0                                   107
      IW4      =  SW4  /  DW  -  0.5                                   108
      NW       =  IW4  -  IW1  +  1                                    109
      SW0      =  DW  *   IW1                                          110
      SWMAJ    =  (SW1  +  SW4)  /  2.                                 111
      WIMAG    =  - WPCNT *  SWMAJ /  100.                             112
```

```
      RETURN                                                            113
      END                                                               114


      SUBROUTINE ALCMEM (MAXBFS, IV, IM, JM, IA, IB, IC, ID, IQ)        115
C                                                                       116
C     ALLOCATE MEMORY OF THE GLOBAL BUFFER AND                          117
C     CHECK THE MAXIMUM BUFFER SIZE.                                    118
C                                                                       119
      INCLUDE   'FXZMIG.CMB/NOLIST'                                     120
C                                                                       121
      IV      =  1                        !  VBUF START ADDRESS         122
      IM      =  IV  +  NX                !  AMBUF                      123
      JM      =  IM  +  NX                !  AM2BUF                     124
      IA      =  JM  +  NX                !  ABUF                       125
      IB      =  IA  +  NX2               !  BBUF                       126
      IC      =  IB  +  NX2               !  CBUF                       127
      ID      =  IC  +  Nx2               !  DBUF                       128
      IQ      =  ID  +  NX2               !  QBUF                       129
      IQ      =  MAX0 (IQ, IV + NTFFT*2 - 1)                            130
      NBFEXT  =  IQ  +  NX2 * NW  -  1                                  131
      NBFDMX  =  NX  *  NZ                                              132
      NBFREQ  =  MAX0 (NBFEXT, NBFDMX)                                  133
C                                                                       134
      IF (NBFREQ .GT. MAXBFS) STOP  'INSUFFICIENT BFS'                  135
      RETURN                                                            136
      END                                                               137


      SUBROUTINE MLTSTP (VBUF, AMBUF, AM2BUF, ABUF, BBUF, CBUF, DBUF,   138
     +                   QBUF)                                          139
C                                                                       140
C     MULTISTEP EXTRAPOLATION AND IMAGING.                              141
C                                                                       142
      INCLUDE   'FXZMIG.CMB/NOLIST'                                     143
      COMPLEX    QBUF(NX,NW)                                            144
      DIMENSION  VBUF(NX)                                               145
C                                                                       146
C                                                                       147
      REWIND LDVVEL                                                     148
      OPEN (UNIT = LDVMIG, FILE = 'FXZMIG.OUT', FORM = 'UNFORMATTED',   149
     +  TYPE = 'UNKNOWN')                                              150
      CALL IMAGEZ (QBUF, ABUF)                                          151
      DO 20 IZ = 2, NZ                                                  152
      READ    (LDVVEL)          VBUF                                    153
      DO 10 JW = 1, Nw                                                  154
      SW      =  SW0  +  DW  *  (JW  -  1)                              155
      CALL VSMULT (NX, VBUF, SW, AMBUF)                                 156
 10   CALL EXTPOL (QBUF(1,JW), ABUF, BBUF, CBUF, DBUF, AMBUF, AM2BUF)   157
      CALL IMAGEZ (QBUF, ABUF)                                          158
      TYPE *, IZ, '-TH STEP COMPLETED'                                  159
 20   CONTINUE                                                          160
      RETURN                                                            161
      END                                                               162
```

```
      SUBROUTINE TM2FRQ (BUF, QBUF)                                     163
C                                                                       164
C     GET TIME-DOMAIN DATA, TRANSFORM, AND SAVE ON QBUF IN              165
C     DEMULTIPLEXED FORM.                                               166
C                                                                       167
      INCLUDE   'FXZMIG.CMB/NOLIST'                                     168
C                                                                       169
      COMPLEX    QBUF(NX,NW)                                            170
C     DIMENSION  BUF(NTFFT*2)                                           171
      DIMENSION  BUF(2)                                                 172
C                                                                       173
      OPEN (UNIT = LDVTRC, FILE = FILTRC, FORM = 'UNFORMATTED',         174
     + TYPE = 'OLD', READONLY)                                          175
      NT2     =  NT  +  NT                                              176
      NTFFT2  =  NTFFT  +  NTFFT                                        177
      IW1     =  2 * IFIX(SW0 / DW  +  0.5)  +  1                       178
C                                                                       179
      DO 10 IX = 1, NX                                                  180
      CALL STORE  (NTFFT2, BUF, 0.)                                     181
      READ  (LDVTRC) (BUF(J), J = 1, NT2, 2)                            182
      CALL FFTC   (NTFFT, BUF, 1.)                                      183
      CALL MOVEJC (NW, BUF(IW1), QBUF(IX,1), 1, NX)                     184
 10   CONTINUE                                                          185
      CLOSE   (LDVTRC)                                                  186
      RETURN                                                            187
      END                                                               188


      SUBROUTINE OUTPUT (BUF)                                           189
      INCLUDE   'FXZMIG.CMB/NOLIST'                                     190
      DIMENSION  BUF(NX,NZ)                                             191
      REWIND  LDVMIG                                                    192
      DO 10 JZ = 1, NZ                                                  193
 10   READ    (LDVMIG)  (BUF(IX,JZ), IX = 1, NX)                       194
      REWIND  LDVMIG                                                    195
      DO 20 IX = 1, NX                                                  196
 20   WRITE   (LDVMIG)  (BUF(IX,JZ), JZ = 1, NZ)                       197
      RETURN                                                            198
      END                                                               199


      SUBROUTINE IMAGEZ (QBUF, BUF)                                     200
C                                                                       201
C     IMAGE THE MIGRATED WAVE FILED                                     202
C                                                                       203
      INCLUDE   'FXZMIG.CMB/NOLIST'                                     204
C                                                                       205
      COMPLEX    QBUF(NX, NW)                                           206
      DIMENSION  BUF(NX)                                                207
C                                                                       208
      DO 10 IX = 1, NX                                                  209
      CALL SUMVJ  (NW, QBUF(IX,1), BUF(IX), NX2)                        210
 10   CONTINUE                                                          211
      WRITE (LDVMIG) (BUF(IX), IX = 1, NX)                              212
      RETURN                                                            213
      END                                                               214
```

```
      SUBROUTINE EXTPOL (QBUF, ABUF, BBUF, CBUF, DBUF, AMBUF, AM2BUF)   215
C                                                                       216
C     EXTRAPOLATE THE WAVE ALONG THE Z DIRECTION.                       217
C                                                                       218
      INCLUDE   'FXZMIG.CMB/NOLIST'                                     219
      COMPLEX    QBUF(NX), ABUF(NX),BBUF(NX),CBUF(NX),DBUF(NX)          220
      COMPLEX    XTINER, XTOUTR                                         221
      DIMENSION  AMBUF(NX)                                              222
C                                                                       223
      CALL VVMULT (NX, AMBUF, AMBUF, AM2BUF)                            224
      DO 10 K = 1, NESTDR                                               225
      CALL CFBND2 (AMBUF, BBUF(1), XTINER, XTOUTR, BNDBUF(1,K))         226
      ABUF(1) =  XTINER * QBUF(2)  +  XTOUTR * QBUF(1)                  227
      CALL CFBND2 (AMBUF(NX), BBUF(NX), XTINER, XTOUTR, BNDBUF(1,K))    228
      ABUF(NX)=  XTINER * QBUF(NX1)  +  XTOUTR * QBUF(NX)               229
      CALL CFINT2 (CBUF, ABUF, DBUF, BBUF, EXTBUF(1,K), AMBUF,AM2BUF)   230
      CALL CVAMMA (NY, QBUF,QBUF(3), CBUF(2), QBUF(2), DBUF(2),         231
     +                   DBUF(2))                                       232
      CALL TRIDGX (NX, QBUF, ABUF, BBUF, DBUF)                          233
 10   CONTINUE                                                          234
C                                                                       235
      DO 20 IX = 1, NX                                                  236
 20   QBUF(IX) =  QBUF(IX) * CEXP (CMPLX (0., DZ * AMBUF(IX)))         237
      RETURN                                                            238
      END                                                               239


      SUBROUTINE CFINT2 (CLF, CNF, CLD, CND, CFM, AMB, AM2)             240
C                                                                       241
C     COMPUTE DIFFERENCE COEFFICIENTS OF 2-ND ORDER WAVE EQUATION       242
C     SEE PAGE 42-R EQ. (10A).                                          243
C                                                                       244
C     VARIABLE DESCRIPTION.                                             245
C                                                                       246
C     NAME    LENGTH          DESCRIPTION                               247
C                                                                       248
C     CLF     NX2     COEFF. OF OLD/OFF-DIAGONAL                        249
C     CNF     NX2     COEFF. OF NEW/OFF-DIAGONAL                        250
C     CLD     NX2     COEFF. OF OLD/DIAGONAL                            251
C     CND     NX2     COEFF. OF NEW/DIAGONAL                            252
C     CFM     3*4     MATRIX COEFFICIENTS OF P. 42-R EQ. (10A)          253
C     AMB     NX      W / V(I)                                          254
C     AM2     NX      AM ** 2                                           255
C                                                                       256
      INCLUDE   'FXZMIG.CMB/NOLIST'                                     257
      DIMENSION  CLF(4), CNF(4), CLD(4), CND(4)                         258
      DIMENSION  CFM(3,4), AMB(NX), AM2(NX)                             259
C                                                                       260
C                                                                       261
      CALL VSMSAJ (NY, AM2(2), CFM(2,1), CFM(1,1), CLF(3),1,2)          262
      CALL VSMSAJ (NY, AM2(2), CFM(2,2), CFM(1,2), CNF(3),1,2)          263
      CALL VSMSAJ (NY, AM2(2), CFM(2,3), CFM(1,3), CLD(3),1,2)          264
      CALL VSMSAJ (NY, AM2(2), CFM(2,4), CFM(1,4), CND(3),1,2)          265
      CALL VSMULJ (NY, AMB(2), CFM(3,1),  CLF(4),  1,  2)               266
      CALL VSMULJ (NY, AMB(2), CFM(3,2),  CNF(4),  1,  2)               267
      CALL VSMULJ (NY, AMB(2), CFM(3,3),  CLD(4),  1,  2)               268
```

```
      CALL VSMULJ (NY, AMB(2), CFM(3,4),  CND(4),  1,  2)                269
C                                                                        270
      RETURN                                                             271
      END                                                                272


      SUBROUTINE CFBND2 (AMREAL, XTRATE, XTINER, XTOUTR, BCOF)           273
C                                                                        274
C     COMPUTE TRANSPARENT BOUNDARY COEFFICIENTS(XTRATE,XTINER,XTOUTR)    275
C     SEE PAGE 40-R EQ. (11)                                             276
C                                                                        277
      INCLUDE   'FXZMIG.CMB/NOLIST'                                      278
      COMPLEX  EYEB, XTRATE, XTINER, XTOUTR, CMDZ, CAMX, CSCALE, CAM     279
      DIMENSION  BCOF(2)                                                 280
C                                                                        281
C                                                                        282
      ADX     =  BCOF(1)  *  DX                                          283
      EYEB    =  CMPLX (0.,  2. * BCOF(2))                               284
      CAM     =  CMPLX (AMREAL, AMIMAG)                                  285
      CMDZ    =  CAM  *  DZ                                              286
      CAMX    =  CAM  *  ADX                                             287
      CSCALE  =  1.  /  (CMDZ  +  CAMX  +  EYEB)                         288
      XTRATE  =  CSCALE  *  ( CMDZ  -  CAMX  +  EYEB)                    289
      XTINER  =  CSCALE  *  ( CMDZ  +  CAMX  -  EYEB)                    290
      XTOUTR  =  CSCALE  *  (-CMDZ  +  CAMX  +  EYEB)                    291
      RETURN                                                             292
      END                                                                293


      SUBROUTINE CFINTI (CFM, B, C, DFCNMR, ALPHA)                       294
C                                                                        295
C     INITIALIZE CFM(3,4) EXTRAPOLATION MATRIX GIVEN IN P42-R            296
C     EQ. (19A).                                                         297
C                                                                        298
C     DEFINITION OF THE EXTRAPOLATION MATRIX : CFM(3,4)                  299
C                                                                        300
C     CFM(1,J) : CONSTANT TERM                                           301
C     CFM(2,J) : COEFF. OF AM * AM                                       302
C     CFM(3,J) : COEFF. OF  I * AM                                       303
C     J = 1    : OLD/OFF-DIAGONAL                                        304
C     J = 2    : NEW/OFF-DIAGONAL                                        305
C     J = 3    : OLD/DIAGONAL                                            306
C     J = 4    : NEW/DIAGONAL                                            307
      INCLUDE   'FXZMIG.CMB/NOLIST'                                      308
      DIMENSION  CFM(3,4)                                                309
C                                                                        310
C                                                                        311
      BETA    =  ALPHA   -  0.5                                          312
      F1      =  (B + B) * DX * DX / DFCNMR                              313
      F2      =  (F1  +  F1)  *  AMIMAG                                  314
      G2      =    C  *  DZ                                              315
      G1      =  -G2  *  AMIMAG                                          316
C                                                                        317
      CALL CFINTJ (CFM(1,1), F1, F2, G1, G2, ALPHA, -1., -1.)            318
      CALL CFINTJ (CFM(1,2), F1, F2, G1, G2, ALPHA, -1.,  1.)            319
      CALL CFINTJ (CFM(1,3), F1, F2, G1, G2, BETA,   2.,  2.)            320
```

```
      CALL CFINTJ (CFM(1,4), F1, F2, G1, G2, BETA,   2., -2.)          321
C                                                                      322
      RETURN                                                           323
      END                                                              324


      SUBROUTINE CFINTJ (COF, F1, F2, G1, G2, ALPHA, R1, R2)           325
C                                                                      326
C     GENERATE A 3-ELEMENT EXTRAPOLATION COEFFICIENT VECTOR GIVEN IN   327
C     P 42-R EQ. (10).                                                 328
C     COF(1) : CONSTANT TERM                                           329
C     COF(2) : COEFF. OF AM * AM                                       330
C     COF(3) : COEFF. OF  I * AM                                       331
C                                                                      332
      DIMENSION  COF(3)                                                333
C                                                                      334
      COF(1)  =  R1  +  R1  +  R2  *  G1                               335
      COF(2)  =  ALPHA  *  R1  *  F1                                   336
      COF(3)  =  R1 * ALPHA * F2  +  R2 * G2                           337
      RETURN                                                           338
      END                                                              339


      SUBROUTINE EXTCON                                                340
C                                                                      341
C     GENERATE EXTRAPOLATION COEFFICIENTS                              342
C                                                                      343
      INCLUDE   'FXZMIG.CMB/NOLIST'                                    344
      DIMENSION  DFCBUF(2)                                             345
      DOUBLE PRECISION A(15), B(15)                                    346
      DATA    MODE    /  -1  /                                         347
C                                                                      348
      DATA    A(1)  /  0. 376 369 527 234 052  /       !  65           349
      DATA    B(1)  /  0. 478 242 059 603 743  /                       350
C                                                                      351
      DATA    A(2)  /  0. 873 981 642 171 890  /       !  80           352
      DATA    B(2)  /  0. 040 315 156 988 852  /                       353
      DATA    A(3)  /  0. 222 691 982 666 100  /                       354
      DATA    B(3)  /  0. 457 289 565 835 625  /                       355
C                                                                      356
      DATA    A(4)  /  0. 972 926 131 694 782  /       !  87           357
      DATA    B(4)  /  0. 004 210 419 911 239  /                       358
      DATA    A(5)  /  0. 744 418 058 525 258  /                       359
      DATA    B(5)  /  0. 081 312 882 016 760  /                       360
      DATA    A(6)  /  0. 150 843 924 026 968  /                       361
      DATA    B(6)  /  0. 414 236 604 654 513  /                       362
C                                                                      363
      DATA    A(7)  /  0. 991 834 774 675 097  /       !  89           364
      DATA    B(7)  /  0. 000 737 959 542 660  /                       365
      DATA    A(8)  /  0. 911 282 437 100 351  /                       366
      DATA    B(8)  /  0. 016 329 891 492 279  /                       367
      DATA    A(9)  /  0. 602 498 780 802 238  /                       368
      DATA    B(9)  /  0. 120 110 756 314 730  /                       369
      DATA    A(10) /  0. 102 624 305 081 323  /                       370
      DATA    B(10) /  0. 362 806 692 332 044  /                       371
C                                                                      372
```

```
      DATA     A(11) /  0. 997 370 236 438 328  /       !  90
      DATA     B(11) /  0. 000 153 427 175 533  /
      DATA     A(12) /  0. 964 827 991 878 123  /
      DATA     B(12) /  0. 004 172 967 255 246  /
      DATA     A(13) /  0. 824 918 564 779 961  /
      DATA     B(13) /  0. 033 860 917 808 142  /
      DATA     A(14) /  0. 483 340 757 434 262  /
      DATA     B(14) /  0. 143 798 075 648 762  /
      DATA     A(15) /  0. 073 588 212 879 826  /
      DATA     B(15) /  0. 318 013 812 535 422  /
C
      DATA  DFCBUF  /  1. 000 000 000,   0. 124 600 000  /
C
C
      IF (NESTDR .LT. 1)       THEN
            NESTDR  =  1
            A(1)    =  0.25
            B(1)    =  0.5
      ENDIF
      IF (MODE .GE. 0)         THEN     !  MODELING MODE
            DX      =  ABS (DX)
            DZ      =  ABS (DZ)
      ELSE                               !  MIGRATION MODE
            DX      = -ABS (DX)
            DZ      = -ABS (DZ)
      ENDIF
C
      K0      =  (NESTDR * (NESTDR - 1))  /  2
      DO 10 K = 1, NESTDR
      AA      =  1.  /  A(K+K0)
      BB      =  AA  *  B(K+K0)
      CALL CFBNDG (BNDBUF(1,K), A(K+K0), B(K+K0))
 10   CALL CFINTI (EXTBUF(1,K), AA, BB, DFCBUF(1), DFCBUF(2))
      RETURN
      END


      SUBROUTINE CFBNDG (BCOF, A, B)
C
C     GENERATE THE TRANSPARENT BOUNDARY COEFFICIENTS AS
C
C     B*XX / (1 + A*XX) === BB*Y / (1 + AA*Y), WHERE
C           XX      =  DERIV(X)**2 / M, AND
C           Y       =  I * DERIV(X)/ M.
C           BCOF(1) =  AA
C           BCOF(2) =  BB
C
      DIMENSION  BCOF(2)
      LOGICAL    VIRGIN
      DATA       VIRGIN  /  .TRUE.  /
C
C     COMPUTE INTEGRALS (S2 THRU S6)
C
      IF (VIRGIN)                      THEN
            VIRGIN  =  .FALSE.
            S2      =  ATAN (1.)
```

```
                S4      =  S2  *  0.75                                  427
                S6      =  S4  *  5.  /  6.                             428
                S3      =  2.  /  3.                                    429
                S5      =  S3  *  0.8                                   430
        ENDIF                                                           431
C                                                                       432
C       COMPUTE NORMAL EQUATION COEFFICIENTS                            433
C                                                                       434
        A11     =  B * B * S6                                           435
        A12     =  B  *  (S4  -  A * S6)                                436
        A22     =  S2 - 2.*A*S4  +  A*A*S6                              437
        B1      =  B * B * S5                                           438
        B2      =  B * (S3  -  A * S5)                                  439
C                                                                       440
C       SOLVE THE EQUATION                                              441
C                                                                       442
        DETERM  =  1.  /  (A11 * A22  -  A12 * A12)                     443
        AA      =  DETERM * (B1 * A22  -  B2 * A12)                     444
        BB      =  DETERM * (B2 * A11  -  B1 * A12)                     445
C                                                                       446
C       CONVERT COEFFICIENTS IN OLD FORMAT, I.E.,                       447
C                                                                       448
C       BOLD(AA,BB)     =  Y  /  (AA  +  BB*Y)                          449
C                                                                       450
        BCOF(1) =  1.  /  BB                                            451
        BCOF(2) =  AA  /  BB                                            452
        RETURN                                                          453
        END                                                             454


        SUBROUTINE EDTVEL (BUF)                                         455
C                                                                       456
        INCLUDE   'FXZMIG.CMB/NOLIST'                                   457
        DIMENSION  BUF(NX, NZ)                                          458
C                                                                       459
        OPEN (UNIT = LDVVEL, FILE = FILVEL, FORM = 'UNFORMATTED',       460
     +  TYPE = 'OLD', READONLY)                                          461
C                                                                       462
        DO 20 IX = 1, NX                                                463
        READ    (LDVVEL) (BUF(IX, JZ), JZ = 1, NZ)                      464
        DO 10 JZ = 1, NZ                                                465
 10     BUF(IX,JZ) =  2.  /  BUF(IX,JZ)                                 466
 20     CONTINUE                                                        467
        CLOSE   (LDVVEL)                                                468
C                                                                       469
        OPEN (UNIT = LDVVEL, FILE = 'FXZMIG.SC1', FORM = 'UNFORMATTED', 470
     +  TYPE = 'SCRATCH')                                               471
C                                                                       472
        DO 30 JZ = 2, NZ                                                473
 30     WRITE   (LDVVEL) (BUF(IX, JZ), IX = 1, NX)                      474
        AMIMAG  =  WIMAG  *  BUF(1,1)                                   475
        RETURN                                                          476
        END                                                             477


C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C 478
```

```
C                                                                       479
C       THIS IS A STACK OF ELEMENTARY SUBPROGRAMS REFERENCED            480
C       BY THE MIGRATION PROGRAM.                                       481
C                                                                       482
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C     483
C                                                                       484
        FUNCTION KPOWR2 (NUMBER)                                        485
C                                                                       486
C       FIND A NUMBER WHICH IS .GE. THE ARGUMENT AND IS ALSO            487
C       A POWER OF 2.                                                   488
C                                                                       489
        KPOWR2  =  1                                                    490
        DO 10 K = 1, 15                                                 491
        KPOWR2  =  KPOWR2  +  KPOWR2                                    492
        IF (KPOWR2 .GE. NUMBER)  RETURN                                 493
 10     CONTINUE                                                        494
        STOP    'ERROR IN KPOWR2'                                       495
        END                                                             496


        SUBROUTINE CVAMMA (N, A, B, C, D, E, F)                         497
        COMPLEX    A(N), B(N), C(N), D(N), E(N), F(N)                   498
        DO 10 I = 1, N                                                  499
 10     F(I)    =  (A(I) + B(I)) * C(I)  +  D(I) * E(I)                 500
        RETURN                                                          501
        END                                                             502


        SUBROUTINE FFTC (LX, CX, SIGNI)                                 503
C                                                                       504
C       FAST FOURIER TRANSFORM. SEE CLAERBOUT P 12.                     505
C                                                                       506
        COMPLEX CX(LX), CARG, CEXP, CW, CTEMP                           507
        J       = 1                                                     508
        DO 30 I = 1, LX                                                 509
        IF (I .GT. J)   GO TO 10                                        510
        CTEMP   = CX(J)                                                 511
        CX(J)   = CX(I)                                                 512
        CX(I)   = CTEMP                                                 513
10      M       = LX / 2                                                514
20      IF (J .LE. M)   GO TO 30                                        515
        J       = J - M                                                 516
        M       = M / 2                                                 517
        IF (M .GE. 1)   GO TO 20                                        518
30      J       = J + M                                                 519
        L       = 1                                                     520
40      ISTEP   = 2 * L                                                 521
        DO 50 M = 1, L                                                  522
        CARG    = CMPLX(0.,   3.14159265 * SIGNI * (M - 1) / L)         523
        CW      = CEXP(CARG)                                            524
        DO 50 I = M, LX, ISTEP                                          525
        CTEMP   = CW * CX(I + L)                                        526
        CX(I+L) = CX(I) - CTEMP                                         527
50      CX(I)   = CX(I) + CTEMP                                         528
        L       = ISTEP                                                 529
        IF (L .LT. LX)  GO TO 40                                        530
```

```
      RETURN                                                            531
      END                                                               532


      SUBROUTINE MOVEJC (N, A, B, JA, JB)                               533
      COMPLEX    A(N), B(N)                                             534
      IA       =  1  -  JA                                              535
      IB       =  1  -  JB                                              536
      DO 10 I  =  1, N                                                  537
      IA       =  IA  +  JA                                             538
      IB       =  IB  +  JB                                             539
10    B(IB)    =  A(IA)                                                 540
      RETURN                                                            541
      END                                                               542


      SUBROUTINE STORE (N, X, CONST)                                    543
      DIMENSION X(N)                                                    544
      IF (N .LT. 1)   GO TO 12                                          545
      DO 10 I = 1, N                                                    546
10    X(I)     = CONST                                                  547
12    RETURN                                                            548
      END                                                               549


      SUBROUTINE SUMVJ (N, X, SIGMA, JMP)                               550
      DIMENSION  X(2)                                                   551
      SIGMA    =  0.                                                    552
      IX       =  1  -  JMP                                             553
      DO 10 I  =  1, N                                                  554
      IX       =  IX  +  JMP                                            555
10    SIGMA    =  SIGMA  +  X(IX)                                       556
      RETURN                                                            557
      END                                                               558


      SUBROUTINE TRIDGX (N, T, A, B, D)                                 559
C                                                                       560
C     TRIDIAGONAL EQUATION WITH TRANSPARENT BOUNDARY CONDITION.         561
C                                                                       562
      COMPLEX    T(N), A(N), B(N), D(N), DEN                            563
C                                                                       564
      N1       = N - 1                                                  565
      DO 10 I  = 2, N1                                                  566
      DEN      = 1.  /  (B(I)  +  A(I) * B(I-1))                       567
      B(I)     =  - A(I)  *  DEN                                        568
10    A(I)     =  (D(I)  -  A(I) * A(I-1))  *  DEN                      569
      T(N)     = (A(N1) * B(N)  +  A(N)) / (1.  -  B(N) * B(N1))        570
      DO 20 J  = 1, N1                                                  571
      I        = N - J                                                  572
20    T(I)     = B(I) * T(I + 1)  +  A(I)                               573
      RETURN                                                            574
      END                                                               575


      SUBROUTINE VSMSAJ (N, A, S1, S2, C, JMPA, JMPC)                   576
```

```
      DIMENSION  A(N), C(N)                                              577
      JA       =  1  -  JMPA                                            578
      JC       =  1  -  JMPC                                            579
      DO 10 I = 1, N                                                    580
      JA       =  JA  +  JMPA                                           581
      JC       =  JC  +  JMPC                                           582
10    C(JC)    =  S1  *  A(JA)  +  S2                                   583
      RETURN                                                            584
      END                                                               585


      SUBROUTINE VSMULJ (N, A, S, C, JMPA, JMPC)                        586
      DIMENSION  A(N), C(N)                                             587
      IJA      =  1  -  JMPA                                            588
      IJC      =  1  -  JMPC                                            589
      DO 10 I = 1, N                                                    590
      IJA      =  IJA  +  JMPA                                          591
      IJC      =  IJC  +  JMPC                                          592
10    C(IJC)   =  S    *  A(IJA)                                        593
      RETURN                                                            594
      END                                                               595


      SUBROUTINE VSMULT (N, A, S, C)                                    596
      DIMENSION  A(2), C(2)                                             597
      DO 10 I = 1, N                                                    598
10    C(I)     =  S  *  A(I)                                            599
      RETURN                                                            600
      END                                                               601


      SUBROUTINE VVMULT (N, A, B, C)                                    602
      DIMENSION  A(N), B(N), C(N)                                       603
      DO 10 I =  1, N                                                   604
10    C(I)     =  A(I)  *  B(I)                                         605
      RETURN                                                            606
      END                                                               607
```