U.S. DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

# Some Considerations on the Steep-Dip Finite-Difference Migration

By S. Y. Suh[1], M. W. Lee[2], and J. A. Grow[2]

Open-File Report 85-219



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

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

1985

CONTENTS



ILLUSTRATIONS



TABLES

# SOME CONSIDERATIONS ON THE STEEP-DIP FINITE-DIFFERENCE MIGRATION

By S. Y. Suh, M. W. Lee, and J. A. Grow

## ABSTRACT

The practical implementation of a steep-dip migration scheme by the finite-difference method is the main concern of this paper. Successful steep-dip migration requires not only an accurate one-way wave equation but also an accurate difference approximation of the differential equation.

Accurate one-way equations can be obtained using an optimization method (Lee and Suh, 1985). This optimization method can also be applicable to the derivation of accurate difference operators.

This paper describes a practical approach to the steep-dip migration using optimized one-way and difference operators.

## INTRODUCTION

Seismic migration is a process of imaging the subsurface from the measured data. The finite-difference method is one of the migration schemes, the advantage being that it is easily applicable to spatial velocity variations. However, this method cannot accurately image reflectors having high dip angles. The dip limitation of the finite-difference method is primarily due to the inaccurate dispersion relation of the approximate one-way equation employed in the scheme.

There are two types of one-way equations: explicit and implicit. Gazdag (1980) and Berkhout (1980) introduced a finite-difference method using explicit one-way equations. In this method, the derivatives in the horizontal direction (x) are estimated either by a convolution in the spatial domain or by a multiplication in the wavenumber domain. This method effectively controls the difference-approximation error. Ma (1981) developed a solution method to high-order implicit equations. Lee and Suh (1985) introduced optimized one-way equations of the implicit type which significantly reduce the dispersion error in conventional one-way equations.

The steep-dip finite-difference migration by the implicit one-way equation is degraded by the error in derivative-to-difference approximation. The apparent wavenumber in x-direction increases from zero to the true wavenumber as the dip increases from zero to 90 degrees. Therefore, the difference error is more significant in the steep-dip migration than in the gentle-dip migration.

This paper is organized as follows. The first section reviews the theory of finite-difference migration. The second section describes a synthetic example on steep-dip finite-difference migration by the explicit and implicit schemes, which shows that each scheme has its own problem for the steep-dip migration. Finally, a new method of the difference approximation is presented. This paper also includes an appendix, which is a computer program written mainly in array-processor assembler language.

## THEORY OF FINITE-DIFFERENCE MIGRATION

The finite-difference migration employs an one-way equation for the wave-field extrapolation. According to the type of one-way equation, it is divided into two schemes: explicit and implicit. This section briefly reviews the theory of finite-difference migration.

The extrapolation equation of the two-dimensional wave P(x, z) is given by the following square-root equation (Claerbout, 1976, p. 202),

$$P_z(x,z) = ik\,(1 + \partial_{xx}/k^2)^{1/2}\,P(x,z) \tag{1}$$

where x is the horizontal distance, z is depth, k is the wave number, and $P_z$ is the derivative of P with respect to z. Equation (1) represents a wave propagating in the one z-direction, i.e., positive or negative, and is called the one-way equation. The finite-difference migration uses an approximate one-way equation which is a rational approximation of equation (1) with respect to $\partial_{xx}$. An explicit approximation uses the Taylor series,

$$P_z = ik\,(1 + \partial_{xx}/2k^2 + \partial_{xxxx}/8k^4 + \cdots)\,P. \tag{2}$$

In the wave extrapolation, wave field P and its x-derivatives are known, while the z-derivative $P_z$ is unknown. Equation (2) represents the unknown $P_z$ in terms of the known explicitly. Therefore, it is an explicit equation. The explicit scheme computes the x-derivatives either by a convolution in the space domain or by a multiplication in the wavenumber domain. By computing the right-hand side of equation (2), the wave at $z + \Delta z$ is calculated by the following equation,

$$P(z+\Delta z) = P(z) + \Delta z\,P_z(z) + (\Delta z)^2 P_{zz}(z)/2 + \cdots . \tag{3}$$

One of the advantages of the explicit scheme is that the accuracy of the x-derivative is easily improved, for example, by using a long difference operator.

The implicit scheme of the finite-difference migration approximates equation (1) as

$$P_z = ik\,\Big(1 + \frac{\sum_j a_j \partial_{xx}^j}{1 + \sum_j b_j \partial_{xx}^j}\Big)\,P. \tag{4}$$

The coefficients--$a_j$ and $b_j$--may be obtained either by the continued-fractions method (Hildebrand, 1956, p. 406) or by the least-squares optimization method (Lee and Suh, 1985). A polynominal expression of equation (4) represents the unknown $P_z$ in terms of the unknown $P_{xxz}$. Therefore, equation (4) is an implicit equation. A solution method of equation (4) is developed by Ma (1981). In his method, equation (4) is divided into partial fractions as

$$P_z = ik\,\Big(1 + \sum_j \frac{\alpha_j \partial_{xx}}{1 + \beta_j \partial_{xx}}\Big)\,P. \tag{5}$$

The solution to equation (5) or, equivalently, to equation (4) may be obtained by solving the split equations separately. A split equation resembles the so-called 45-degree equation,

$$P_z = ik\,\frac{\alpha\,\partial_{xx}}{1 + \beta\,\partial_{xx}}\,P. \tag{6}$$

## MIGRATION TEST

It is generally believed that the finite-difference method is not accurate for steep-dip migration. Such a limitation is primarily due to the inaccurate dispersion relation of the one-way equation employed by the conventional method. In this section, a model experiment is given by both the explicit and implicit schemes. Figure 1 shows a synthetic reflector model with a homogeneous velocity of 3,000 m/sec. 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.

In a homogeneous medium, the propagation angle of the normal ray, with respect to the z-axis, is the same as the dip angle. Whereas, the propagation angle is generally less than the dip angle because, normally, the velocity decreases as the ray propagates from the reflector to the surface.

Figure 2 is a synthetic zero-offset section and is computed by the phase-shift method (Gazdag, 1978). The grid intervals are $\Delta x = \Delta z = 50$ m with the number of grid points in the x- and z-directions by 512 and 121, respectively. The time increment $\Delta t$ is 50 msec and the number of time samples is 256.

Figure 3 shows three migrated sections of figure 2. The result is represented in (x, t) domain, i.e., time migration. Figure 3A is computed by the program MIGRATX of DISCO (Digicon's Interactive Seismic Computer) software. The program uses the explicit finite-difference algorithm and controls the difference error quite accurately. Lee and Suh (1985) studied the dispersion relations of explicit and implicit one-way equations and showed that the former is less accurate than the latter for the same order of approximation. In figure 3A, the 45-degree segments are properly migrated, but the steeper segments are still under-migrated. The under-migration is caused by the inaccurate dispersion relation of the explicit one-way equation.

Figures 3B and 3C are computed by the implicit method using the optimized second-order equation and the optimized fourth-order equation (Lee and Suh, 1985), respectively. The figures show remarkable differences. The effect of under-migration in figure 3A is progressively diminishing in figures 3B and 3C. However, the two figures show phase error in steeper segments caused by the numerical error in the derivative-to-difference approximation.

There are two methods of suppressing the numerical error. The one is by using a smaller sampling interval, and the other is by using a more elaborate difference scheme. Figure 4 shows a migrated result of figure 2 using the optimized fourth-order equation with a new sampling interval $\Delta x = 12.5$ m. Changing the sampling interval can be accomplished by an interpolation. The result shows neither the under-migration nor the phase error. The 70-degree dip is successfully migrated.

Another criteria for evaluating the performance of these various migration schemes is to compare computing times. Table 1 summarizes the computing times for the migration together with its input parameters. The computation is done by a VAX 11/780 CPU and an AP 120B array processor.

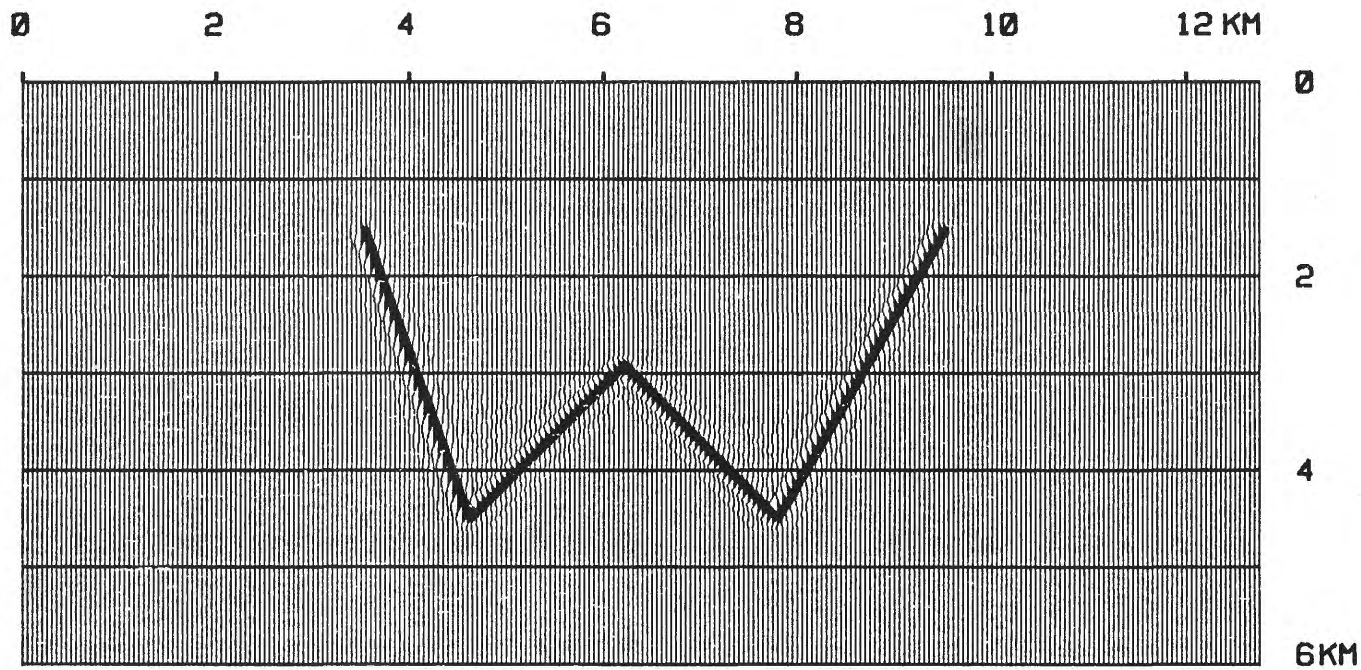


Figure 1.--A W-shaped reflector model.

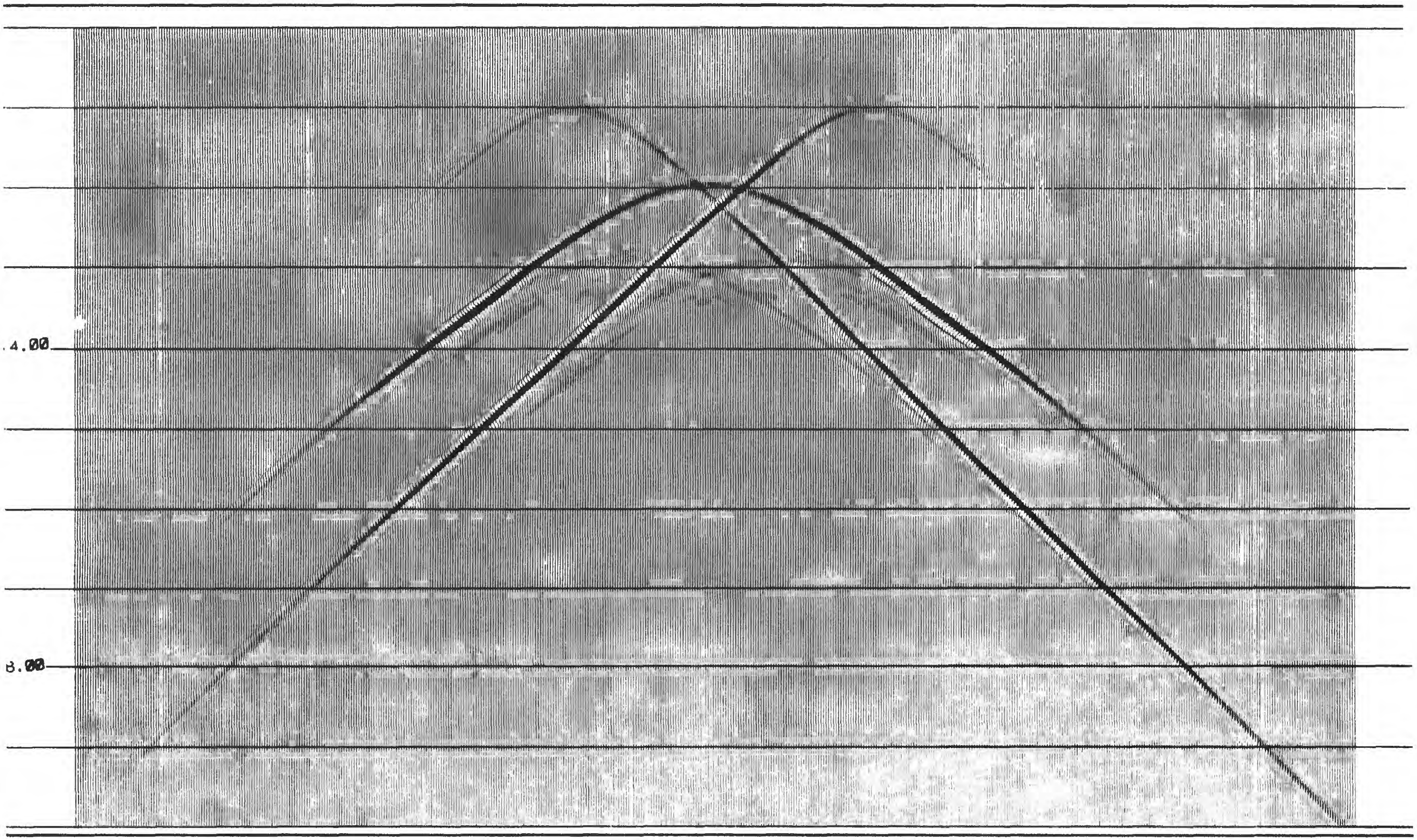


Figure 2.--Synthetic zero-offset section of figure 1.

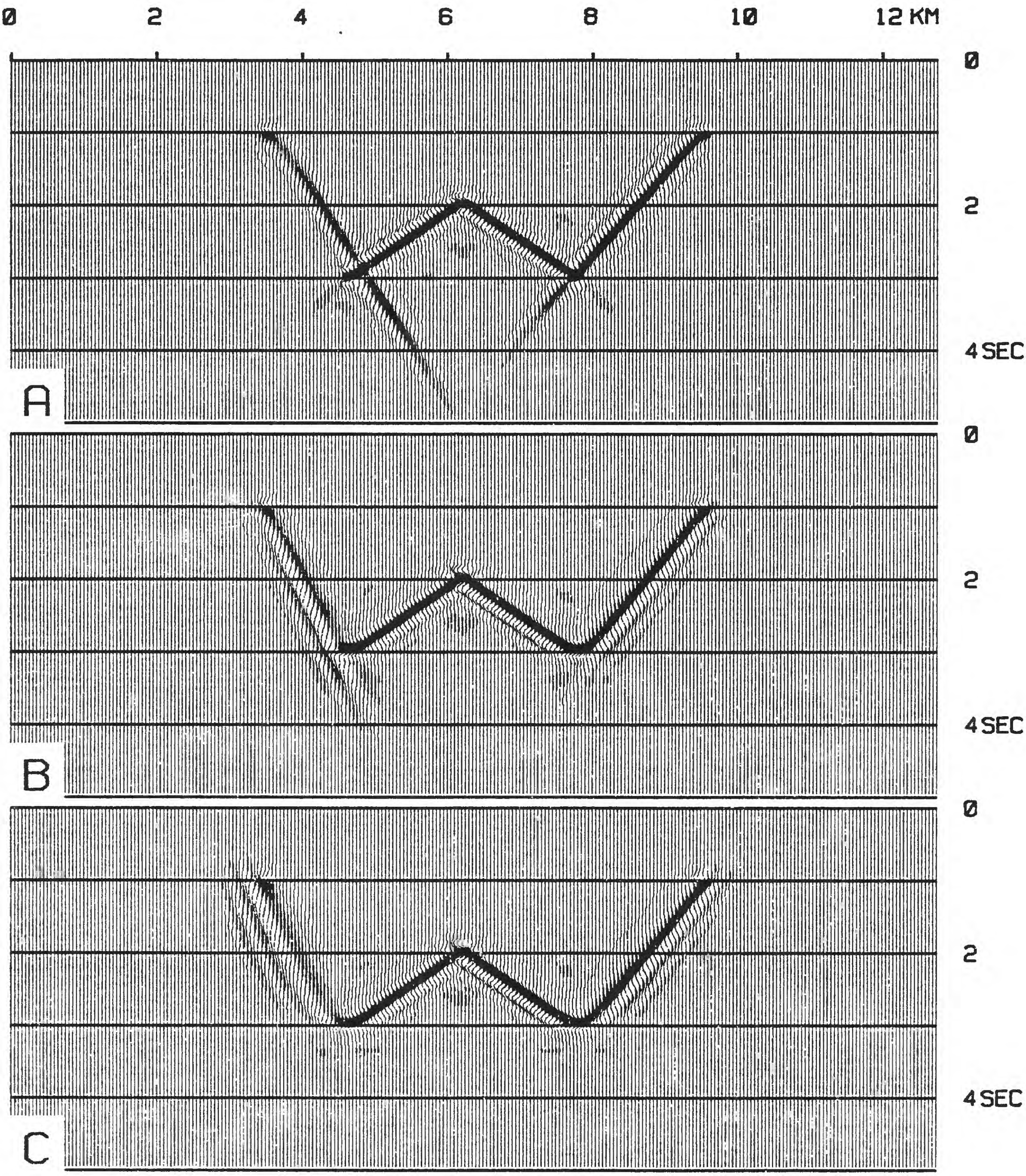


Figure 3.--Finite-difference migration of figure 2 by an explicit scheme (A), by an implicit second-order equation (B), and by an implicit fourth-order equation (C).

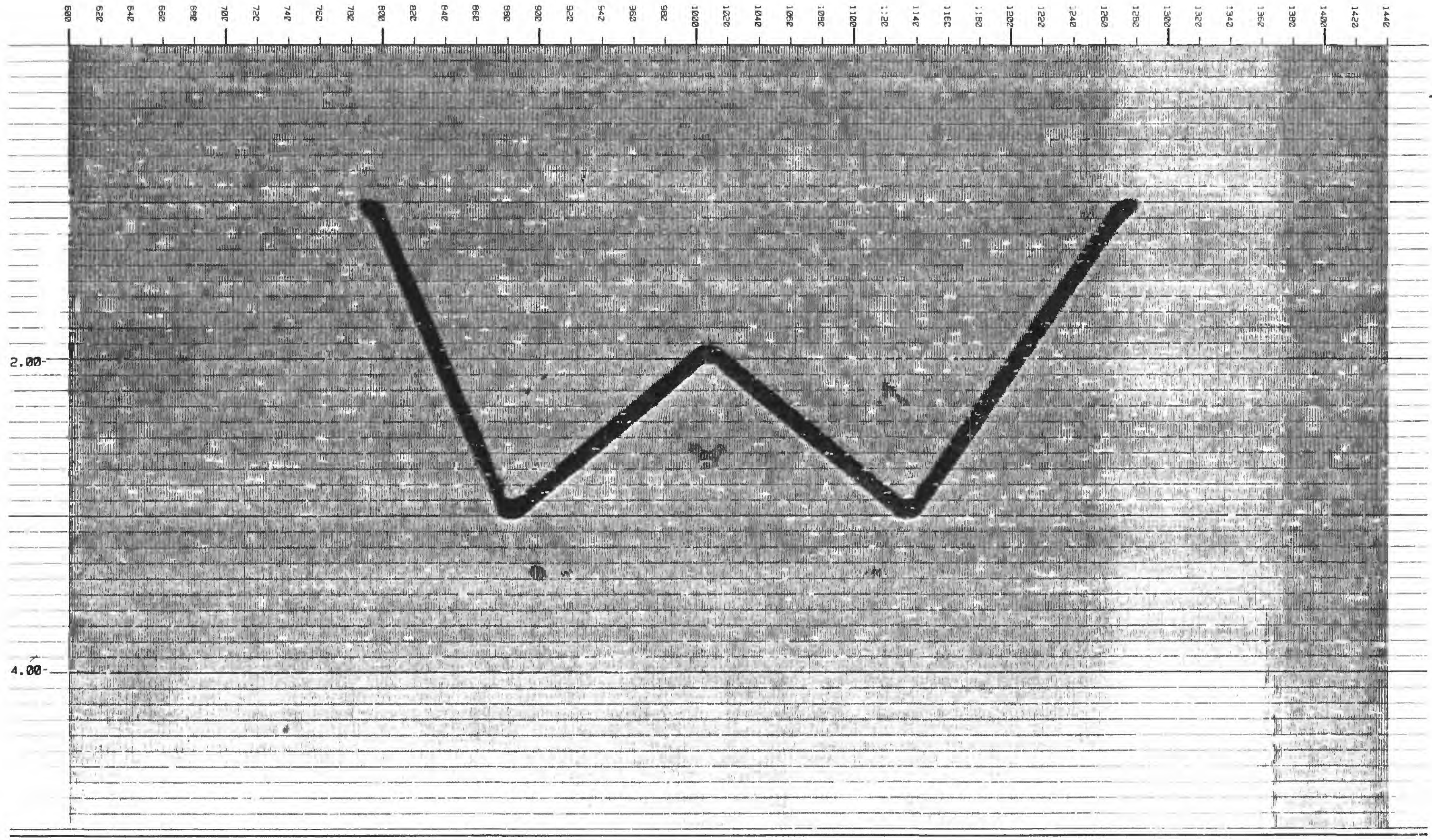


Figure 4.--Finite-difference migration of figure 2 by the implicit fourth-order equation with reduced trace interval.

The array processor has 65,536 words of memory and has a 167 nanoseconds cycle time. The first and second columns of the table show the figure name and the number of traces in the input data. The third column shows the sample rate of the input data in msec. The sample rates are different from that of the synthetic zero-offset section of figure 2, i.e., the inputs are interpolated in the time axis. The fourth column of the table shows the downward-continuation interval in msec. The fifth and sixth columns indicate the CPU time and the AP time, respectively, in seconds. The last column represents the number of page faults which greatly affects the computing time in a virtual memory system. Page faults can be reduced by increasing the size of a working set in a VAX computer. However, a common working set size was used for this comparison.

Table 1.--Run-time report on the finite-difference migrations

| Fig. No. | No. of traces | $\Delta t$ (msec) | $\Delta \tau$ (msec) | CPU time (sec) | AP time (sec) | Page faults |
|---|---|---|---|---|---|---|
| 3A | 512 | 4 | 20 | 1,247.24 | 3,855.13 | 107,057 |
| 3B | 512 | 20 | 20 | 42.99 | 331.04 | 3,435 |
| 3C | 512 | 20 | 20 | 43.12 | 514.40 | 3,454 |
| 4 | 2,048 | 10 | 10 | 15,917.66 | 5,049.24 | 1,153,956 |

There are two kinds of computing time in the migration. One is the time for wave-field extrapolation, and the other is the time for matrix transposition. The extrapolation requires a lot of arithmetic operations and is the task of the array processor. The AP time in table 1 is the time for extrapolation. On the other hand, the matrix transposition in migration requires a lot of physical memory and is accomplished by the CPU. The matrix transposition used most of the CPU time in table 1.

The AP time of the implicit scheme is generally less than that of the explicit scheme. The CPU time is highly dependent on the number of samples of the input data. Figure 4 takes a tremendous amount of CPU time for the matrix transpose, which is caused by the resampling in x-direction. This suggests that interpolation is not an effective method for the finite-difference scheme.

## DERIVATIVE-TO-DIFFERENCE APPROXIMATION

One of the major concerns of the finite-difference method is to reduce the numerical dispersion error caused by the finite sampling interval. Decreasing the sampling interval is one of the accepted methods of reducing numerical error, but it is time-consuming and inefficient. Thus, more accurate derivative-to-difference approximations are desirable.

For a steep-dip migration, the apparent wavenumber in the x-direction increases according to the dip angle, thereby reducing the apparent wavelengths. As a result, the effect of numerical errors is more obvious in the steep-dip migration. Therefore, a more accurate derivative-to-difference approximation in the x-direction is required. Claerbout (1976) introduced an implicit difference approximation of the second derivative as

$$\partial_{xx}^{(2)} = \frac{1}{(\Delta x)^2} \frac{\delta_{xx}}{1+\delta_{xx}/12} \tag{7}$$

where

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

Equation (7) is a rational approximation to the first two terms of the following series expression of $\partial_{xx}$.

$$\partial_{xx} = \frac{1}{(\Delta x)^2}\left(\delta_{xx} - \frac{1}{12}\delta_{xx}^2 + \frac{1}{90}\delta_{xx}^3 - \frac{1}{560}\delta_{xx}^4 + \cdots\right). \tag{8}$$

The finite-difference formulation of equation (6) by equation (7) results in a tri-diagonal equation. The formulation by the first term of equation (8) also results in a tri-diagonal equation. Therefore, equation (7) takes little more computing time than the first term approximation but provides more accurate results.

The 2n-th order approximation of $\partial_{xx}$ may be written in the following rational form,

$$\partial_{xx}^{(2n)} = \frac{1}{(\Delta x)^2} \frac{\sum_{j=1}^{n} c_j \delta_{xx}^j}{1+\sum_{j=1}^{n} d_j \delta_{xx}^j}. \tag{9}$$

The coefficients--$c_j$ and $d_j$--may be found by converting equation (8) into the rational form. The accuracy of the approximation is observed in the wavenumber domain. The Fourier transform of $\partial_{xx}$ is

$$\tilde{\partial}_{xx} = -k_x^2. \tag{10}$$

For the unit grid interval, the Fourier transform of $\partial_{xx}^{(2n)}$ is

$$\tilde{\partial}_{xx}^{(2n)} = \frac{\sum c_j \tilde{\delta}_{xx}^j}{1+\sum d_j \tilde{\delta}_{xx}^j} \tag{11}$$

where

$$\tilde{\delta}_{xx} = 2(\cos k_x - 1).$$

Therefore, the error $E^{(2n)}$ of equation (9) is the difference between equation (11) and equation (10), i.e.,

$$E^{(2n)}(k_x) = \frac{\sum c_j \tilde{\delta}_{xx}^j}{1+\sum d_j \tilde{\delta}_{xx}^j} + k_x^2. \tag{12}$$

Equation (12) is very similar to the dispersion error of the 2n-th order implicit equation (Lee and Suh, 1984). This suggests that coefficients $c_j$ and $d_j$ may be found by a least-squares method. Introducing a weighted error, i.e.,

$$\hat{E}^{(2n)}(k_x) = \sum_{j=1}^{n} c_j \tilde{\delta}_{xx}^{j} + k_x^2 \left(1 + \sum_{j=1}^{n} d_j \tilde{\delta}_{xx}^{j}\right) \tag{13}$$

the least-squares method is reduced to the linear problem. Therefore, the coefficients are found by the minimization of the integral

$$J \triangleq \int_0^{\phi} \left[\hat{E}^{2n}(k_x)\right]^2 dk_x \tag{14}$$

where ∅ is the maximum wavenumber of the optimization. The wavenumber does not exceed the Nyquist wavenumber (180 degrees).

Table 2 shows the least-squares solution for fourth- and sixth-order approximation. The first column indicates the order of the approximation. The second column shows the maximum wavenumber of the optimization. The third and fourth columns show the estimated coefficients for the numerator and denominator of equation (9), respectively. The last column indicates the subscript of each coefficient.

Table 2.--Estimated coefficients of the difference operators in equation (9)

| Order | ∅ | $c_j$ | $d_j$ | j |
|---|---|---|---|---|
| 4 | 135 | 1 | .260 013 893 | 1 |
| | | .175 550 990 | .011 521 881 | 2 |
| 6 | 180 | 1 | .494 329 422 | 1 |
| | | .410 176 191 | .071 525 600 | 2 |
| | | .040 479 975 | .002 565 484 | 3 |

Figure 5 shows the amplitude spectra of the derivative operators. Graph A is the spectrum of the exact second-derivative operator. Graph B is the spectrum of the second-order approximate operator given in equation (7). Graphs C and D are the spectra of the fourth- and sixth-approximate operators. Figure 5 shows that graph B is acceptable up to about 50 percent of the total bandwidth, while graphs C and D are acceptable up to about 75 percent and 90 percent, respectively. This suggests that the conventional difference operator given in equation (7) may be replaced by the more accurate operators which are fourth- and sixth-order approximation of $\partial_{xx}$.

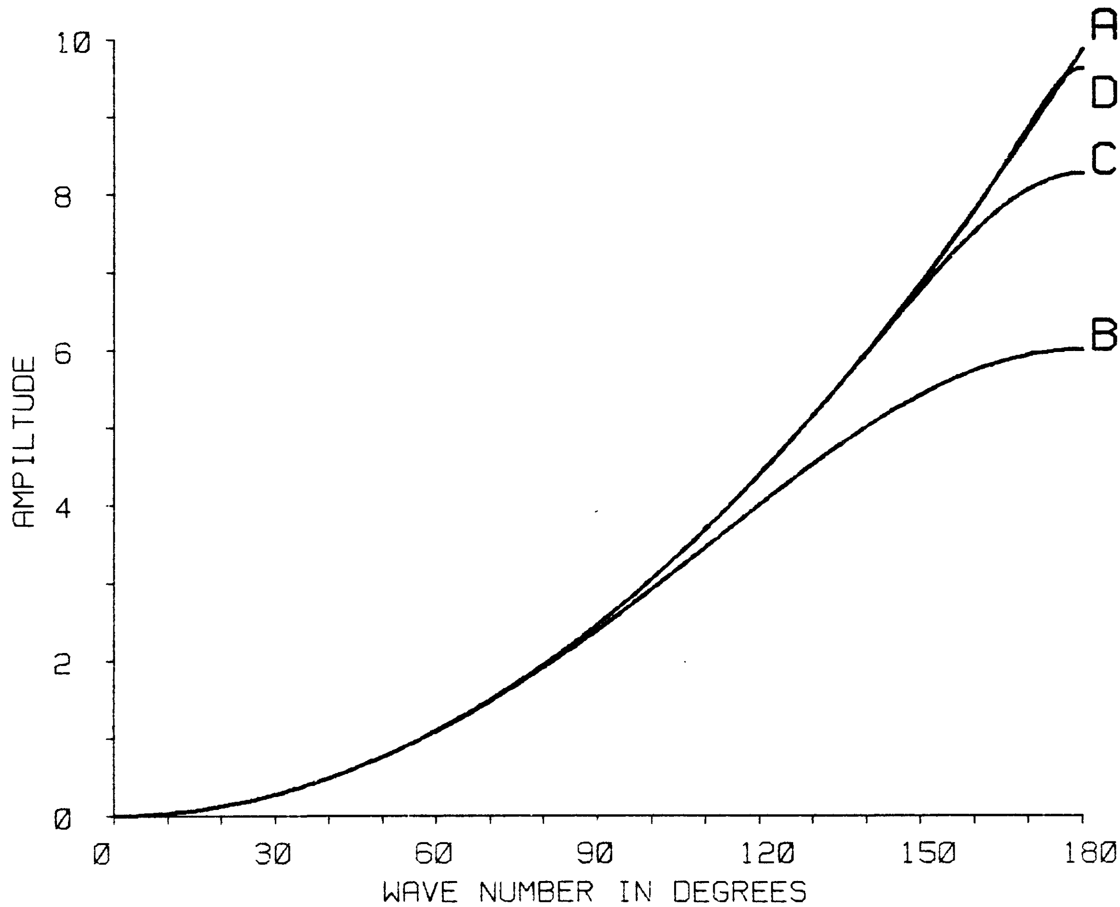


Figure 5.--Amplitude spectra of second-derivative operators: A is the exact second-derivative operator; B, a second-order approximate operator; C, a fourth-order approximate operator; and D, a sixth-order approximate operator.

The finite-difference formulation of equation (6) by high-order difference approximation does n ot result in a tri-diagonal equation, i.e., the fourth-order approximation gives a pentagonal equation and the sixth-order approximation gives a heptagonal equation. The general form of the heptagonal equation is

$$A_k T_{k-3} + B_k T_{k-2} + C_k T_{k-1} + D_k T_k + E_k T_{k+1} + F_k T_{k+2} + G_k T_{k+3} = H_k \tag{15}$$

for k = 3, 4, ..., N-3. In the equation, $A_k$ through $H_k$ are known coefficients and T is the unknown. The solution to equation (15) employs an auxilliary equation of

$$T_k = P_k T_{k+3} + Q_k T_{k+2} + R_k T_{k+1} + S_k, \tag{16}$$

where the coefficients $P_k$ through $S_k$ are firstly found by comparing to equation (15). The unknown $T_k$ are then computed by equation (16).

The number of arithmetic operations in the finite-difference method increases according to the order of the difference approximation. Let us assume that there are N input traces for the migration. The number of significant elements in the tri-diagonal matrix is approximately 3N, while those in the pentagonal and heptagonal matrices are 5N and 7N, respectively. Therefore, the number of arithmetic operations in high-order difference approximation increases approximately by the same ratio.

Table 3 shows an estimated time report to migrate figure 2, by using the high-order difference approximations. The first column of the table shows the order of the optimized one-way equation. The second column shows the order of the difference approximations. The third, fourth and fifth columns show the number of traces, the sample rate ($\Delta t$), and the downward-continuation interval ($\Delta\tau$), respectively. The sixth and seventh columns show the estimated CPU time and AP time, respectively, in seconds. The CPU time is easily predictable from table 1. The AP times are computed by scaling the corresponding AP time by the factor of 1.67 or 2.33 depending on the order of the difference approximation.

Table 3.--Estimated time report on finite-difference migration of figure 2

| Order of one-way equation | Order of difference equation | No. of traces | $\Delta t$ (msec) | $\Delta\tau$ (msec) | CPU time (sec) | AP time (sec) |
|---|---|---|---|---|---|---|
| 2 | 4 | 512 | 20 | 20 | 43 | 550 |
| 2 | 6 | 512 | 20 | 20 | 43 | 770 |
| 4 | 4 | 512 | 20 | 20 | 43 | 833 |
| 4 | 6 | 512 | 20 | 20 | 43 | 1,170 |
| 4 | 6 | 512 | 20 | 10 | 43 | 2,340 |

## CONCLUSIONS

Two finite-difference migration schemes, i.e., the explicit and the implicit, were tested using a synthetic model with a maximum dip of 70 degrees. The tests showed that the explicit scheme is subject to the dip limitation of the one-way equation while the implicit scheme is subject to the numerical dispersion. The numerical disperion can be suppressed by the interpolation of the input data, but it takes a tremendous amount of CPU time in the matrix transposition. A new method of controlling the numerical dispersion effect is presented in this paper which employs optimized high-order difference approximations. Because this suggested method is computatively more efficient than the conventional method, it is worth considering for implementing steep-dip migration.

## REFERENCES CITED


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

Claerbout, J. F., 1976, Fundamentals of geophysical data processing: New York, McGraw Hill Book Co., 274 p.

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.

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

Lee, M. W., and Suh, S. Y., 1985, Finite-diffference migration by optimized one-way wave equations: U.S. Geological Survey Open-File Report, in press.

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.

## APPENDIX: ARRAY PROCESSOR VERSION OF THE MIGRATION PROGRAM

Migration is one of the most time-consuming processes in seismic processing. Even by use of a modern high-speed computer that can compute almost one million floating-point operations in a second, the process takes hours of computing time. By introducing the concept of the array processor (AP), the time problem of migration is greatly reduced. The array processor computes multiple operations simultaneously. The operations are further divided into several steps and computed in a pipe-line mode; thereby millions of arithmetic additions and multiplications can be calculated within a second. This version of the migration program uses practically all the resources of the array processor. The imput goes to the array processor, and the output, which is the migrated result, comes from the array processor. The I/O operation between the CPU and AP is minimized. This is accomplished by rewriting the main algorithm into AP assembler language.

The current version of the program supports time migration by the optimized one-way wave equations from the second-order to the tenth-order equation (Lee and Suh, 1985), and by the optimized second-order difference operator. Since the migration is a part of a stream of other seismic processes, the program is designed to handle the output of other processes. Therefore, the program is further converted to be compatible with one of the most commonly used software packages in the exploration industry, the DISCO (Digicon's Interactive Seismic Computer). The following describes the program in the form of the DISCO user's manual as well as the source statements. The name of the program is MIGRHI.

MIGRHI

```
"*CALL"   "MIGRHI"  DX        ORDER     DAMP      ADDFAC

"KEY"     KEYNAM    KYV1ST    KYVLST    KYVINC

"LAYER"   DTAU      ETIME     F1        F2

"VELOCT"  VTYPE     VIDENT
C
C   PARAMETER DESCRIPTIONS
C       LU: LOWER LIMIT
C       UL: UPPER LIMIT
C       DEF: DEFAULT VALUE
C

"*CALL"   "MIGRHI"  DX        ORDER     DAMP      ADDFAC

DX        FLOATING PT  'HORIZONTAL TRACE SPACING IN FEET
                        OR METERS'
     LL :
     UL :
     DEF:
ORDER     INTEGER      'ORDER OF THE ONEWAY EQUATION.'
     LL :  '2'
     UL :  '10'
     DEF:  '2'
DAMP      FLOATING PT  'FACTOR OF THE NUMERICAL DIP-FILTER.
                 DAMP=
     LL :  '0'
     UL :  '5.'
     DEF:  '0'
ADDFAC    FLOATING PT  'FACTOR OF DATA EXTENSION TO PREVENT
                        THE WRAP-AROUND EFFECT OF F-X-Z
                        MIGRATION ALGORITHM'
     LL :  '1.'
     UL :  '2.'
     DEF:  '1.'
```

KEY

```
"KEY"     KEYNAM    KYV1ST    KYVLST    KYVINC

KEYNAM    C* 8         'HEADER ENTRY NAME OF VELOCITY CONTRO
                        POINTS.'
     DEF:  'CDP'
KYV1ST    INTEGER      'FISRT KEY VALUE OF THE TRACE TO BE
                        MIGRATED.'
```

```
     LL :
     UL :
     DEF:
KYVLST     INTEGER       'LAST  KEY VALUE OF THE TRACE TO BE
                          MIGRATED.'
     LL :
     UL :
     DEF:
KYVINC     INTEGER       'INCREMENT OF THE KEY VALUE.
                          NOTE:
                          IF(KYVINC.GT.1) DX MUST BE SET TO
                          DX*KYVINC.
     LL :  '1'
     UL :
     DEF:  '1'
```

LAYER

```
"LAYER"    DTAU      ETIME      F1         F2
DTAU       FLOATING PT  'LAYER THICKNESS IN MSEC.'
     LL :
     UL :
     DEF:  '40.'
ETIME      FLOATING PT  'END MIGRATION TIME IN MSEC.'
     LL :
     UL :
     DEF:
F1         FLOATING PT  'LOWER FREQUENCY LIMIT IN HERZ.'
     LL :  '0'
     UL :
     DEF:  '0'
F2         FLOATING PT  'UPPER FREQUENCY LIMIT IN HERZ.
                         NOTE:
                         TO SAVE COMPUTING TIME, SUPPLY SIGNA
                         BAND ONLY. THE BAND MAY BE LOCATED B
                         SPECTRUM ANALYSIS OF INPUT DATA.'
     LL :
     UL :
     DEF:
```

VELOCT

```
"VELOCT"   VTYPE     VIDENT
VTYPE      INTEGER       'VELOCITY TYPE:
               0 = RMS VELOCITY
               1 =
     LL :  '0'
     UL :  '1'
     DEF:  '0'
VIDENT     C*16          'VELOCITY IDENT NAME DEFINED TO SEISD
     DEF:
C
C.THE FOLLOWING DATA DECK WAS USED TO COMPUTE FIGURE 3B IN THE TEXT
```

```
C

*JOB      KOREA    L8053
*ALT      MIGRHI   [LEE.MIGRHI]
*CALL     DSKRD    [RIFLE.NIDSSS]HYBRID5
*CALL     RESAMP   20.0
*CALL     MIGRHI   50       2
KEY       CHAN     1        512      1
LAYER     20       5000     1.       10.
VELOCT    1        V3000
*CALL     SECPLOT  LR       VAWG     20       1.25
LABEL     CHAN     20
TITLE     FIG. 3B.
TRANGE    0        5000
TIMING    2        1        0        0
SETAMP    PEAK                                         FIRST
GAIN      2
MAXTR     512
*END
```

```
C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C       PROGRAM   **   M I G R H I   **   VERSION TM.8   NOV/24/84
C
C       FINITE-DIFFERENCE WAVE EQUATION TIME-MIGRATION.
C
C       ALGORITHM DESCRIPTION
C
C       1.  THE EXACT SQUARE-ROOT ONE-WAY EQUATION IS OPTIMIZED BY
C           A ONEWAY EQUATION, THE ORDER OF WHICH MAY BE DETERMINED
C           BY THE USER.  FIVE DIFFERENT ONEWAY EQUATIONS ARE
C           SUPPORTED, I.E., SECOND, FOURTH, SIXTH, EIGTH, AND TENTH
C           ORDER EQATIONS,  THE DISPERSION RELATION OF WHICH ARE
C           ACCURATE WITHIN 1 % LIMIT FOR THE PROPAGATION ANGLE OF
C           UP TO 65, 80, 87, 89, AND 90 DEGREES RESPECTIVELY.
C       2.  THE OPTIMIZED ONE-WAY EQUATION IS EXPANDED TO THE FINITE-
C           DIFFERENCE EQUATIONS EMPLOYING AN IMPLICIT DIFFERENCE
C           EQUATION, WHICH IS BASICALLY SAME AS CLAERBOUT' METHOD.
C           BUT THE APPROXIMATION OF X-DERIVATIVE TO DIFFERENCE
C           IS IMPROVED IN THIS VERSION, SO THAT IT CAN HANDLE
C           UP TO THE QUATER WAVELENGTH.
C       3.  ANOTHER REMARK GOES ON THE BOUNDARY CONDITION, WHICH,
C           PRACTICALLY, IS TRANSPARENT AT THE BOUNDARY. THEREFORE
C           YOU DO NOT HAVE TO ADD ZERO-TRACES AT THE BOUNDARY.
C
C       4.  REFER TO LEE AND SUH, 1984, FOR THE DETAIL.
C
C       AUTHORS
C
C       1.  DR. MYUNG W. LEE, RESEARCH GEOPHYSICIST, SEISMIC STRATI-
C               GRAPHY GROUP, THE BRANCH OF OIL AND GAS RESOURCES,
C               U.S. GEOLOGICAL SURVEY, DENVER FEDERAL CENTER,
C               DENVER, COLORADO 80225, U. S. A.    (303) 236-5753
C
C       2.  DR. SANG  Y. SUH, SENIOR GEOPHYSICIST, DEPARTMENT OF
C               GEOPHYSICAL RESEARCH, KOREA INSTITUTE OF ENERGY AND
C               RESOURCES(KIER), 219-5 GARIBONG-DONG, GURO-GU,
C               SEOUL, REPUBLIC OF KOREA.           (002) 856-0041
C
C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C       EDIT-PHASE PROGRAMS
C
C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
        SUBROUTINE MIGRHI_EDITP
        INCLUDE    'MONFORT/NOLIST'
        INCLUDE    'MIGRHI.CMB/NOLIST'
C
C       CONTENTS OF MIGRHI.CMB + + + + + + + + + + + + + + + + + + +
C
        COMMON / MIGRHI_BLK1 /  NX,     NZ,     NT,     DX,     DZ,
     +  DDT,    NTFFT,  NTOUT,  NX1,    NZ1,    NX2,    NW,     NW2,
     +  DW,     SW0,    WIMAG,  AMIMAG, NESTEQ, NZMINR, NZGRND, NWFOLD,
     +  NWGLOB, JWSTAT, EXTBUF(18,5)
```

```
      CHARACTER*56            FILVEL, FILHDR, FILFRQ
      CHARACTER*16            VIDENT
      CHARACTER*8             KEYNAM
      COMMON / MIGRHI_BLK2 /  FILVEL, FILHDR, FILFRQ, LDVVEL, LDVHDR,
     + LDVFRQ, KEYNAM, KEYLEN, KEYFMT, KEYIND, KYV1ST, KYVLST, KYVINC,
     + VIDENT, IVTYPE, NCNTRL
      COMMON / MIGRHI_BLK3 /  MAXAPS, LMIGCN, IGBUF,  IAOLD, IEXTBF,
     + IVBUF, IWBUF,  ISBUF,   IAMBF,  IDZ,    ISW0,   IDW,   IDDT
C
C     END OF FILE MIGRHI.CMB + + + + + + + + + + + + + + + + + + + + +
C
C
      CHARACTER*10    JOBCHR
C
C
      CALL MIGRHI_GETCRD
C
C     ASSIGN FILE NUMBERS AND FILE NAMES
C
      FILVEL  =  'SCR1$DISK:[SEISMIC]MIGR0000.SC1'
      FILHDR  =  'SCR1$DISK:[SEISMIC]MIGR0000.SC2'
      FILFRQ  =  'SCR1$DISK:[SEISMIC]MIGR0000.SC3'
      CALL INFOGET ('JOBNO', JOBNO)
      ENCODE (10, 700, JOBCHR)  JOBNO  +  1000
      FILVEL(24:27)   =  JOBCHR (7:10)
      FILHDR(24:27)   =  JOBCHR (7:10)
      FILFRQ(24:27)   =  JOBCHR (7:10)
      CALL GETIOU (LDVVEL)
      CALL GETIOU (LDVHDR)
      CALL GETIOU (LDVFRQ)
C
C     GET VELOCITY INFORMATION AND RESERVE IT.
C
      CALL GETTCOR (4, 1000, IV)
      CALL MIGRHI_VELGET (VIBAR, RCORE(IV), RCORE(IV+500))
      AMIMAG  =  WIMAG  *  VIBAR
C
C     ALLOCATE MEMORY OF THE GLOBAL BUFFER
C
      MAXAPS  =  65536
      NWMINR  =  (MAXAPS - 201 - 11*NX)  /  (3 + NX + NX)
      IF (NWMINR .LT. 1)      STOP    'INSUFFICIENT AP-MEMORY'
      NBFIMG  =  NX  *  (NTOUT  +  1)
      NBFFRQ  =  MAX0 (MAXAPS, NX*NW*2)
      CALL MEMLCL (NBFIMG  +  NBFFRQ)
      CALL APMEM  (MAXAPS)
      CALL MIGRHI_EXTCON
      CALL SETOPT (OP$EOF)
      RETURN
 700  FORMAT (I10)
      END


      SUBROUTINE MIGRHI_GETCRD
C
```

```
C       READ CONTROL PARAMETERS
C
        INCLUDE     'MONFORT/NOLIST'
        INCLUDE     'MIGRHI.CMB/NOLIST'
        CHARACTER*8  NAMTAB (3)
        DATA         NAMTAB  / 'KEY', 'LAYER', 'VELOCT'  /
        DATA         NUMTAB  /  3  /
        CALL SETGBL (NCARDS)
        DX       =  FPARM ('DX',      000, 0., 0., 0.)
        NESTEQ   =  IPARM ('ORDER',   111, 2, 10,  2 )  /  2
        WPCNT    =  FPARM ('DAMP',    111, 0., 5., 0.)
        FFTFAC   =  FPARM ('ADDFAC', 111, 1., 2., 1.)
C
C       GET LIST PARAMETERS
C
        DO 400 K = 1, NCARDS
        CALL NXTLST (NAMTAB,  NUMTAB,  INDEX, NREP)
        GO TO (100, 200, 300), INDEX
C
C       GET KEY PARAMETERS
C
 100    KEYNAM   =  CPARM ('KEYNAM', 001, 'CDP')
        KYV1ST   =  IPARM ('KYV1ST', 000, 0,  0,  0 )
        KYVLST   =  IPARM ('KYVLST', 000, 0,  0,  0 )
        KYVINC   =  IPARM ('KYVINC', 101, 1,  0,  1 )
        GO TO 400
C
C       READ LAYER PARAMETERS
C
 200    DZ       =  FPARM ('DTAU', 001, 0., 0., 40.)
        STIME    =  0.
        ETIME    =  FPARM ('ETIME',000, 0., 0., 0. )
        SW1      =  FPARM ('F1',    101, 0., 0., 0. )
        SW2      =  FPARM ('F2',    000, 0., 0., 0. )
        GO TO 400
C
C       READ  VELOCITY CARD
C
 300    IVTYPE   =  IPARM ('VTYPE',   111, 0,  1,  0 )
        VIDENT   =  CPARM ('VIDENT', 000,     00000 )
 400    CONTINUE
C
C       NOW CHECK THE PARAMETERS          * * * * * * * * * * * * *
C
        IF (DX .LE. 0.)                     GO TO 800
        NESTEQ   =  MAX0 (1, MIN0 (5, NESTEQ))
        WPCNT    =  AMAX1 (0., AMIN1 (5., WPCNT))
        FFTFAC   =  AMAX1 (1., AMIN1 (2., FFTFAC))
        IF (THDRGET(KEYNAM, KEYLEN, KEYFMT, KEYIND, 'E') .EQ. 0)
     +                                      GO TO  801
        NX       =  (KYVLST  -  KYV1ST)  /  KYVINC  +  1
        IF (NX .LT. 2)                      GO TO  802
        DDT      =  DT  /  1000.
        NZMINR   =  DZ  /  DDT  +  0.5
        DZ       =  DDT *  NZMINR
```

```
      NT       =  LENGTH
      TMAX     =  DDT   *   (NT   -   1)
      ETIME    =  AMIN1 (ETIME, TMAX)
      NTOUT    =  LENGTH
      NZ       =  ETIME   /   DZ    +   1.5
      DDT      =  DDT        /   1000.
      DZ       =  DZ         /   1000.
      FNYQ     =  0.5 /   DDT
      IF (SW1.GE.SW2 .OR. SW2.GT.FNYQ) GO TO 803
C
C     COMPUTE NTFFT
C
      NUMBER   =  NT    *   AMAX1 (1., AMIN1 (2., FFTFAC))
      NTFFT    =  1
      DO 10 K = 1, 15
      NTFFT    =  NTFFT   +   NTFFT
      IF (NTFFT .GE. NUMBER)   GO TO 12
 10   CONTINUE
      STOP     'ERROR IN NTFFT'
 12   CONTINUE
      PI       =  ACOS (-1.)
      DW       =  2.   *   PI   /   (NTFFT    *   DDT)
      SW1      =  2.   *   PI   *   SW1
      SW2      =  2.   *   PI   *   SW2
C
      IW1      =  SW1   /   DW   +   1.5
      IW4      =  SW2   /   DW   -   1.5
      NW       =  IW4   -   IW1   +   1
      SW0      =  DW   *    IW1
      SWMAJ    =  (SW1   +   SW2)   /   2.
      WIMAG    =  WPCNT *   SWMAJ /   100.
      RETURN
C
C     ERROR TERMINAL SECTION
C
 800  STOP     'ERROR 800: DX'
 801  STOP     'ERROR 801: KEYNAM'
 802  STOP     'ERROR 802: KEYVAL'
 803  STOP     'ERROR 803: F1/F2'
      END


C
C
      SUBROUTINE MIGRHI_EXTCON
C
C     INITIALIZE THE EXTPOLATION BUFFER
C
      INCLUDE     'MONFORT/NOLIST'
      INCLUDE     'MIGRHI.CMB/NOLIST'
      DIMENSION   DFCBUF(2)
      DOUBLE PRECISION A(15), B(15)
      DATA     MODE     /  1  /
C
      DATA     A(1)  /  0. 376 369 527 234 052  /       !  65
```

```
      DATA     B(1)   /  0. 478 242 059 603 743   /
C
      DATA     A(2)   /  0. 873 981 642 171 890   /          !  80
      DATA     B(2)   /  0. 040 315 156 988 852   /
      DATA     A(3)   /  0. 222 691 982 666 100   /
      DATA     B(3)   /  0. 457 289 565 835 625   /
C
      DATA     A(4)   /  0. 972 926 131 694 782   /          !  87
      DATA     B(4)   /  0. 004 210 419 911 239   /
      DATA     A(5)   /  0. 744 418 058 525 258   /
      DATA     B(5)   /  0. 081 312 882 016 760   /
      DATA     A(6)   /  0. 150 843 924 026 968   /
      DATA     B(6)   /  0. 414 236 604 654 513   /
C
      DATA     A(7)   /  0. 991 834 774 675 097   /          !  89
      DATA     B(7)   /  0. 000 737 959 542 660   /
      DATA     A(8)   /  0. 911 282 437 100 351   /
      DATA     B(8)   /  0. 016 329 891 492 279   /
      DATA     A(9)   /  0. 602 498 780 802 238   /
      DATA     B(9)   /  0. 120 110 756 314 730   /
      DATA     A(10)  /  0. 102 624 305 081 323   /
      DATA     B(10)  /  0. 362 806 692 332 044   /
C
      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 (MODE .GE. 0)          THEN     !  MIGRATION MODE
              DX       =  ABS (DX)
              DZ       =  ABS (DZ)
      ELSE                               !  MODELING MODE
              DX       = -ABS (DX)
              DZ       = -ABS (DZ)
      ENDIF
C
      K0         =   (NESTEQ * (NESTEQ - 1))  /  2
      DO 10 K = 1, NESTEQ
      AA         =   1.  /  A(K+K0)
      BB         =   AA  *  B(K+K0)
      CALL MIGRHI_CFBNDG (EXTBUF(17,K), A(K+K0), B(K+K0))
 10   CALL MIGRHI_CFINTI (EXTBUF(1,K), AA, BB, DFCBUF(1), DFCBUF(2))
      RETURN
      END
```

```
      SUBROUTINE MIGRHI_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
              S6      =  S4   *  5.  /  6.
              S3      =  2.   /  3.
              S5      =  S3   *  0.8
      ENDIF
C
C     COMPUTE NORMAL EQUATION COEFFICIENTS
C
      A11      =  B * B * S6
      A12      =  B  *  (S4   -   A * S6)
      A22      =  S2 - 2.*A*S4  +  A*A*S6
      B1       =  B * B * S5
      B2       =  B * (S3   -   A * S5)
C
C     SOLVE THE EQUATION
C
      DETERM   =  1.  /  (A11 * A22  -  A12 * A12)
      AA       =  DETERM * (B1 * A22  -  B2 * A12)
      BB       =  DETERM * (B2 * A11  -  B1 * A12)
C
C     CONVERT COEFFICIENTS IN OLD FORMAT, I.E.,
C
C     BOLD(AA,BB)     =  Y  /  (AA  +  BB*Y)
C
      BCOF(1) =  1.  /  BB
      BCOF(2) =  AA  /  BB
      RETURN
      END
      SUBROUTINE MIGRHI_CFINTI (CFM, B, C, DFCNMR, ALPHA)
C
C     INITIALIZE DIFFERENCE COEFFICIENTS OF THE MAIN REGION
C     SEE P42R EQ. (6).
C
      INCLUDE    'MIGRHI.CMB/NOLIST'
C
      DIMENSION  CFM(16)
```

```
C
      BETA     =  ALPHA    -  0.5
      F1       =  (B + B) * DX * DX / DFCNMR
      F2       =  (F1   +  F1)  *  AMIMAG
      G2       =    C   *  DZ
      G1       =  -G2   *  WIMAG
C
      CALL MIGRHI_CFINTJ (CFM(1), F1, F2, G1, G2, ALPHA, -1., -1.)
      CALL MIGRHI_CFINTJ (CFM(5), F1, F2, G1, G2, ALPHA, -1.,  1.)
      CALL MIGRHI_CFINTJ (CFM(9), F1, F2, G1, G2, BETA,   2.,  2.)
      CALL MIGRHI_CFINTJ (CFM(13),F1, F2, G1, G2, BETA,   2., -2.)
      RETURN
      END

      SUBROUTINE MIGRHI_CFINTJ (COF, F1, F2, G1, G2, ALPHA, R1, R2)
C
C     DO THE COMPUTATION FOR CFINT2-INTERIOR
C
      DIMENSION  COF(4)
C
      COF(1)  =  R1  +  R1  +  R2  *  G1
      COF(2)  =  ALPHA  *  R1  *  F1
      COF(3)  =  R2  *  G2
      COF(4)  =  R1  *  F2  *  ALPHA
      RETURN
      END


      SUBROUTINE MIGRHI_SKIPCD (LDEV, NCARD)
      DO 10 ICARD = 1, NCARD
 10   READ (LDEV, 50)
 50   FORMAT (1X)
      RETURN
      END


      SUBROUTINE MIGRHI_VELGET (VIBAR, TIME, VELO)
C
C     GET AND RESERVE THE VELOCITY INFORMATION
C
      INCLUDE    'MONFORT/NOLIST'
      INCLUDE    'MIGRHI.CMB/NOLIST'
C
      DIMENSION   TIME(2), VELO(2)
      CHARACTER*50    FILDFN
      CHARACTER*8     LINE
C
      CALL INFOGET ('LINE', LINE)
      CALL FNAMGET ( LINE, 'VELDEFN', VIDENT, FILDFN, IFSTAT)
      IF (IFSTAT .NE. HERR$OK) STOP   'CAN NOT FIND VIDENT'
      CALL GETIOU  (LDVDFN)
      OPEN (UNIT = LDVDFN, FILE = FILDFN, TYPE = 'OLD', READONLY)
      OPEN (UNIT = LDVVEL, FILE = FILVEL, TYPE = 'SCRATCH',
```

```
     +         FORM = 'UNFORMATTED')
C
        CALL MIGRHI_SKIPCD (LDVDFN, 2)
        READ (LDVDFN, 50) NCNTRL
        CALL MIGRHI_SKIPCD (LDVDFN, 29)
        DO 20 ICNTRL = 1, NCNTRL
        READ (LDVDFN, 50) ICHECK
C       IF (ICHECK .NE. 1)
C    +          STOP    'VELOCITY NOT CORRECTLY DEFINED'
        READ (LDVDFN, 50) NPAIR, JCNTRL
        CALL MIGRHI_SKIPCD (LDVDFN, 17)
        DO 10 I = 1, NPAIR
 10     READ (LDVDFN, 52) TIME(I), VELO(I)
        VALKEY  =  JCNTRL
 20     WRITE (LDVVEL) VALKEY, NPAIR, (TIME(I), VELO(I), I = 1, NPAIR)
        VIBAR   =  1. / VELO(1)
        CLOSE (LDVDFN)
        CALL RELIOU (LDVDFN)
        RETURN
 50     FORMAT (I9)
 52     FORMAT (2F14.1)
        END


C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C       PROCESSING PHASE PROGRAMS
C
C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
        SUBROUTINE MIGRHI_PROCP (TRACE, THDR, IFLAG)
C
        INCLUDE    'MONFORT/NOLIST'
        INCLUDE    'MIGRHI.CMB/NOLIST'
        LOGICAL   NOMIGR
        DATA      NOMIGR  /  .TRUE.  /
C
C
C
        IF (PHASE .EQ. PH$PROC) THEN
          CALL MIGRHI_TM2FRQ (RCORE(FWACOR), TRACE, THDR)
          IFLAG =  FLG$MULTI
        ELSE
          IF (NOMIGR)    THEN
            CALL MIGRHI_TM2FRF (RCORE(FWACOR))
            CALL MIGRHI_DOMIGR
            NOMIGR = .FALSE.
          ENDIF
          CALL MIGRHI_OUTPUT (TRACE, THDR, IFLAG, RCORE(FWACOR+NX))
        ENDIF
        RETURN
        END


        SUBROUTINE MIGRHI_DOMIGR
C
```

```
C         DO THE MAJOR MIGRATION
C
          INCLUDE     'MONFORT/NOLIST'
          INCLUDE     'MIGRHI.CMB/NOLIST'
C
C         INITIALIZE THE PARAMETERS AND EDIT VELOCITY DATA.
C
          NX1      =  NX  -  1
          NZ1      =  NZ  -  1
          NX2      =  NX  +  NX
          MXNXNZ   =  MAX0 (NX, NZ1)
          I1       =  MXNXNZ  +  FWACOR
          I2       =  MXNXNZ  +  I1
          I3       =  MXNXNZ  +  I2
          I4       =  MXNXNZ  +  I3
          CALL MIGRHI_VELEDT (RCORE(I1), RCORE(I2), RCORE(I3),
     +         RCORE(I4), RCORE(FWACOR))
C
C         GET FREQUENCY DATA IN MUX-FORMAT.
C
          IWCPU    =  FWACOR  +  NX  *  (NTOUT  +  1)
          CALL MIGRHI_GETMXC (LDVFRQ, RCORE(IWCPU), NX, NW)
          CLOSE (LDVFRQ)
          CALL RELIOU (LDVFRQ)
C
C         FOLD THE FREQUENCY AND DO MIGRATION.
C
          NWGLOB   =  NW
          NWMINR   =  (MAXAPS - 201 - 11 * NX)  /  (3 + NX2)
          NWFOLD   =  (NWGLOB  -  1)  /  NWMINR  +  1
          CALL APEX_APINIT (0, 0, ISTAT)
          DO 10 IWFOLD = 1, NWFOLD
          JWSTAT   =  (IWFOLD  -  1) * NWMINR  +  1
          NW       =  MIN0 (NWMINR, NWGLOB - JWSTAT + 1)
          NW2      =  NW  +  NW
          CALL MIGRHI_PUTCON (RCORE(FWACOR))
          CALL APEX_APPUT (RCORE(IWCPU+(JWSTAT-1)*NX2), IQBUF, NX2*NW, 2
          CALL APEX_APWD
          IF (IWFOLD .EQ. 1) THEN
            CALL MIGRHI_MLTST1 (RCORE(FWACOR), RCORE(FWACOR + NX))
          ELSE
            CALL MIGRHI_MLTSTP (RCORE(FWACOR), RCORE(FWACOR + NX))
          ENDIF
 10       CONTINUE
          CALL APEX_APRLSE
          CLOSE (LDVVEL)
          CALL RELIOU (LDVVEL)
          RETURN
          END


          SUBROUTINE MIGRHI_PUTCON (BUF)
C
C         PUT MIGRATION CONSTANTS TO THE ARRAY-PROCESSOR.
C
          INCLUDE     'MIGRHI.CMB/NOLIST'
```

```
      DIMENSION  BUF(200)
C
C     AP ADDRESS DEFINITIONS
C
      LMIGCN  =  200
      IDZ     =  1
      ISWO    =  IDZ   +  5
      IDW     =  ISWO  +  1
      IDDT    =  IDW   +  1
      IEXTBF  =  14
      IWBUF   =  201
      ISBUF   =  IWBUF  +  NW
      NW2     =  NW  +  NW
      IVBUF   =  ISBUF  +  NW2
      IAMBF   =  IVBUF  +  NX
      IAOLD   =  IAMBF  +  NX2
      IQBUF   =  IAOLD  +  NX2 * 4
C
C     PUT CONSTANTS TO THE AP.
C
      BUF(IDZ)          =  DZ
      BUF(IDZ+1)        =  WIMAG
C     BUF(IDZ+2)        =  B  +  B                          ! RESERVED
C     BUF(IDZ+3)        =  A  *  DX  *  NESTDF              ! RESERVED
      BUF(IDZ+4)        =  AMIMAG
C
      BUF(ISWO)         =  SWO  +  (JWSTAT  -  1) * DW
      BUF(IDW)          =  DW
      BUF(IDDT)         =  DDT
C
      DO 10 I = 1, NESTEQ
      EXTBUF(17,I)      =  DX  *  EXTBUF(17,I)
 10   EXTBUF(18,I)      =  2.  *  EXTBUF(18,I)
      CALL MIGRHI_MOVE  (90, EXTBUF, BUF(IEXTBF))
      CALL APEX_APPUT (BUF, IDZ, LMIGCN, 2)
      CALL APEX_APWD
C
      CALL FPS_VRAMP  (ISWO, IDW, IWBUF, 1, NW)
      CALL FPS_VSMUL  (IWBUF, 1, IDDT, IQBUF, 1, NW)
      CALL FPS_VCOS   (IQBUF, 1, ISBUF, 2,       NW)
      CALL FPS_VSIN   (IQBUF, 1, ISBUF+1, 2,     NW)
      CALL APEX_APWR
      RETURN
      END


      SUBROUTINE MIGRHI_MLTSTP (VBUF, BUF)
C
C     MULTISTEP EXTRAPOLATION AND IMAGING.
C
      INCLUDE    'MIGRHI.CMB/NOLIST'
      DIMENSION  VBUF(NX)
C
      REWIND LDVVEL
      CALL MIGRHI_IMAGEI
      CALL MIGRHI_IMAGEZ (VBUF, BUF)
```

```
      DO 10 IZ = 1, NZ1
      READ (LDVVEL) VBUF
      CALL APEX_APPUT  (VBUF, IVBUF, NX, 2)
      CALL APEX_APWD
      CALL MIGRHI_FPS_EXTPOL (NX,IAMBF,IAOLD,IQBUF,IWBUF,NESTEQ,NW)
      CALL MIGRHI_IMAGEZ (VBUF, BUF)
 10   CONTINUE
      CALL MIGRHI_IMAGEF (VBUF, BUF)
      CALL APEX_APWR
      RETURN
C
C     MULTISTEP EXTRAPOLATION FOR JWSTAT = 1
C
      ENTRY   MIGRHI_MLTST1 (VBUF, BUF)
      REWIND LDVVEL
      CALL MIGRHI_IMAGEI
      CALL MIGRHI_IMAGEA (VBUF, BUF)
      DO 20 IZ = 1, NZ1
      READ (LDVVEL) VBUF
      CALL APEX_APPUT  (VBUF, IVBUF, NX, 2)
      CALL APEX_APWD
      CALL MIGRHI_FPS_EXTPOL (NX,IAMBF,IAOLD,IQBUF,IWBUF,NESTEQ,NW)
      CALL MIGRHI_IMAGEA (VBUF, BUF)
 20   CONTINUE
      CALL MIGRHI_IMAGAF (VBUF, BUF)
      CALL APEX_APWR
      RETURN
      END


      SUBROUTINE MIGRHI_IMAGEZ (XBUF, BUF)
C
C     IMAGE THE WAVE-FIELD, AND  REMOVE IT FROM THE QBUF.
C
      INCLUDE    'MIGRHI.CMB/NOLIST'
C
      DIMENSION  XBUF(NX), BUF(NX, NTOUT)
C
 11   DO 20 IZ = 1, NLOUT
      JZOUT   =  JZOUT  +  1
      IF (JZOUT .GT. NTOUT)  RETURN
      CALL APEX_APWR
      CALL MIGRHI_FPS_IMAGEW (NX, NW, IQBUF, IVBUF)
      CALL APEX_APWR
      CALL APEX_APGET  (XBUF, IVBUF, NX, 2)
      CALL APEX_APWD
      CALL MIGRHI_FPS_PHSHFT (NX, NW, ISBUF, IQBUF)
      CALL MIGRHI_VVADD (NX, XBUF, BUF(1,JZOUT), BUF(1,JZOUT))
 20   CONTINUE
      GO TO 99
C
      ENTRY   MIGRHI_IMAGEF (XBUF, BUF)
      IF (JZOUT .GE. NTOUT)   RETURN
      NLOUT   =  NTOUT  -  JZOUT
      GO TO 11
C
```

```
      ENTRY   MIGRHI_IMAGEI
      JZOUT   =  0
      NLOUT   =  NZMINR
      GO TO 99
C
C     IMAGE FOR JWSTAT = 1
C
      ENTRY   MIGRHI_IMAGEA (XBUF, BUF)
C
 22   DO 30 IZ = 1, NLOUT
      JZOUT   =  JZOUT  +  1
      IF (JZOUT .GT. NTOUT)  RETURN
      CALL APEX_APWR
      CALL MIGRHI_FPS_IMAGEW (NX, NW, IQBUF, IVBUF)
      CALL APEX_APWR
      CALL APEX_APGET  (BUF(1,JZOUT), IVBUF, NX, 2)
      CALL APEX_APWD
      CALL MIGRHI_FPS_PHSHFT (NX, NW, ISBUF, IQBUF)
 30   CONTINUE
      GO TO 99
C
      ENTRY   MIGRHI_IMAGAF (XBUF, BUF)
      IF (JZOUT .GE. NTOUT)   RETURN
      NLOUT   =  NTOUT  -  JZOUT
      GO TO 22
 99   RETURN
      END


      SUBROUTINE MIGRHI_TM2FRQ (BUF, TRACE, THDR)
C
C     TRANSFORM TO FREQUENCY DOMAIN AND SAVE ON QBUF IN MUX-ORDER
C
      INCLUDE    'MONFORT/NOLIST'
      INCLUDE    'MIGRHI.CMB/NOLIST'
      INTEGER    THDR (THDRLEN)
      LOGICAL    VIRGIN
      DATA       VIRGIN  /  .TRUE.  /
C
      DIMENSION  BUF(2)
C
      IF (VIRGIN)      THEN
              NW2      =  NW  +  NW
              IBLSZ    =  MIN0 (8*NW2, 32764)
              OPEN (UNIT = LDVHDR, FILE = FILHDR, TYPE = 'SCRATCH',
     +        FORM = 'UNFORMATTED')
              OPEN (UNIT = LDVFRQ, FILE = FILFRQ, TYPE = 'SCRATCH',
     +        FORM = 'UNFORMATTED', RECL = NW2, RECORDTYPE = 'FIXED'
     +        BLOCKSIZE = IBLSZ, INITIALSIZE = NX/2, BUFFERCOUNT = 2
              KYV1ST   =  MAX0 (KYV1ST, THDR(KEYIND))
              KEYNXT   =  KYV1ST
              NX       =  0
              NFOLDS   =  MAXAPS / NTFFT
              IFOLDS   =  0
              VIRGIN   =  .FALSE.
      ENDIF
```

```
C
        KEYVAL   =  THDR (KEYIND)
        IF (KEYVAL.LT.KEYNXT .OR. KEYVAL.GT.KYVLST)      RETURN
        NZEROT   =  (KEYVAL   -  KEYNXT)  /  KYVINC
C
C       CORRECT FOR ZERO TRACES
C
        IF (NZEROT .GT. 0)       THEN
                DO 10 ITRACE = 1, NZEROT
                NX       =  NX   +  1
                THDR (KEYIND)    =  KEYNXT
                KEYNXT           =  KEYNXT   +  KYVINC
                WRITE (LDVHDR) THDR
                IFOLDS   =  IFOLDS  +  1
                IF (IFOLDS .GT. NFOLDS) THEN
                        CALL MIGRHI_FTRANS (NFOLDS, BUF)
                        IFOLDS   =  1
                ENDIF
                CALL MIGRHI_STORE (NT, BUF((IFOLDS-1)*NT+1), 0.)
 10             CONTINUE
        ENDIF
C
C       NOW TREAT THE INPUT TRACE.
C
        IFOLDS   =  IFOLDS   +  1
        IF (IFOLDS .GT. NFOLDS) THEN
                CALL MIGRHI_FTRANS (NFOLDS, BUF)
                IFOLDS   =  1
        ENDIF
        CALL MIGRHI_MOVE (NT, TRACE, BUF((IFOLDS-1)*NT+1))
        NX       =  NX   +  1
        THDR  (KEYIND)   =  KEYNXT
        WRITE (LDVHDR)      THDR
        KEYNXT           =  KEYNXT   +  KYVINC
        RETURN
C
C       END-PROCEDURE
C
        ENTRY MIGRHI_TM2FRF(BUF)
C
        CALL MIGRHI_FTRANS (IFOLDS, BUF)
        RETURN
        END


        SUBROUTINE MIGRHI_FTRANS (NTRACE, BUF)
C
C       TRANSFORM NTRACE DATA TO FREQUENCY
C
        INCLUDE    'MIGRHI.CMB/NOLIST'
C
        DIMENSION  BUF(2)
        LOGICAL VIRGIN
        DATA    VIRGIN  /  .TRUE.  /
C
C ***** SET UP
```

```
C
      IF (VIRGIN)      THEN
            NW2      =  NW  +  NW
            IW0      =  2  *  IFIX(SW0  /  DW  +  0.5)
            VIRGIN   =  .FALSE.
      ENDIF
      CALL APEX_APINIT (0, 0, ISTAT)
      CALL APEX_APPUT  (BUF, 0, NTRACE * NT, 2)
      CALL APEX_APWD
      CALL MIGRHI_FPS_RFFTMM (NTRACE, NT, NTFFT, NW2, IW0,
     +                        NTRACE*NT,  NTRACE*NTFFT)
      CALL APEX_APWR
      CALL APEX_APGET  (BUF, IW0, NTRACE*NW2, 2)
      CALL APEX_APWD
      CALL APEX_APRLSE
C
C  ***  MOVE THE RESULT TO QBUF IN MUX-FORM
C
      J1      =  1
      DO 50 IX = 1, NTRACE
      J2      =  J1  +  NW2  -  1
      WRITE (LDVFRQ) (BUF(J), J = J1, J2)
 50   J1      =  J2  +  1
      RETURN
      END


      SUBROUTINE MIGRHI_GETMXC (LDEV, CBUF, NX, NW)
C
C     GET FREQUENCY DATA IN MUX-ORDER.
C
      COMPLEX     CBUF(NX,NW)
C
      REWIND  LDEV
      DO 10 IX = 1, NX
 10   READ   (LDEV) (CBUF(IX,JW), JW = 1, NW)
      RETURN
      END


      SUBROUTINE MIGRHI_OUTPUT (TRACE, THDR, IFLAG, BUF)
C
C     PASS THE MIGRATED RESULT.  ONE TRACE AT A TIME.
C
      INCLUDE     'MONFORT/NOLIST'
      INCLUDE     'MIGRHI.CMB/NOLIST'
      DIMENSION   THDR(THDRLEN)
      DIMENSION   BUF (NX, NTOUT)
      LOGICAL     VIRGIN
      DATA        VIRGIN  /  .TRUE.  /
C
C
      IF (VIRGIN)      THEN
            IX       =  1
            REWIND LDVHOR
            VIRGIN   =  .FALSE.
```

```
      ELSE
              IX      =  IX  +  1
      ENDIF
      IF (IX  -  NX)  10, 20, 30
 10   IFLAG   =  FLG$MULTO
      GO TO 22
 20   IFLAG   =  FLG$NORM
 22   CALL MIGRHI_MOVEJ (NTOUT, BUF(IX,1), TRACE, NX, 1)
      READ (LOVHOR) THOR
      RETURN
 30   IFLAG   =  FLG$MULTI
      RETURN
      END


      SUBROUTINE MIGRHI_VELEDT (TIME, VRMS, VINT, XCNTRL, BUF)
C
C     EDIT VELOCITY DATA.
C
      INCLUDE     'MIGRHI.CMB/NOLIST'
      DIMENSION   TIME(2), VRMS(2), VINT(2), XCNTRL(2), BUF(NZ1, 2)
C
      REWIND LOVVEL
      DDZ      =  DZ  *  1000.
      DO 20 JCNTRL = 1, NCNTRL
      READ (LOVVEL) XCNTRL(JCNTRL), NVAL, (TIME(I),VRMS(I),I=1,NVAL)
      IF (IVTYPE .EQ. 0)       THEN
              CALL MIGRHI_RMS2MV (NVAL, TIME, VRMS, VINT)
      ELSE
              CALL MIGRHI_INT2MV (NVAL, TIME, VRMS, VINT)
      ENDIF
      I1       =  1
      DO 10 I  =  1, NVAL
      I2       =  MIN0 (NZ1, IFIX(TIME(I)/DDZ + 1.001))
      IF (I1 .LE. I2)  CALL MIGRHI_STORE (I2-I1+1,  BUF(I1,JCNTRL),
     +                 VINT(I))
 10   I1       =  I2  +  1
      IF (I1 .LE. NZ1) CALL MIGRHI_STORE (NZ1-I1+1, BUF(I1, JCNTRL),
     +                 VINT(NVAL))
 20   CONTINUE
      CLOSE (LOVVEL)
      OPEN (UNIT = LOVVEL, FILE = FILVEL, TYPE = 'SCRATCH',
     + FORM = 'UNFORMATTED', RECL = NX, RECORDTYPE = 'FIXED',
     + BLOCKSIZE = 8*NX, INITIALSIZE = NZ/2, BUFFERCOUNT = 2)
C
C     NOW INTERPOLATE IN THE HORIZONTAL DIRECTION.
C     NOW VRMS=XOUT AND TIME=VOUT.
C
      CALL MIGRHI_VRAMP (NX, VRMS, FLOAT(KYV1ST), FLOAT(KYVRMSC))
      DO 30 IZ = 1, NZ1
      CALL MIGRHI_MOVEJ (NCNTRL, BUF(IZ, 1), VINT, NZ1, 1)
      CALL MIGRHI_LINVBP(NCNTRL, XCNTRL, VINT, NX, VRMS, TIME)
 30   WRITE (LOVVEL) (TIME(I), I = 1, NX)
      RETURN
      END
```

```
      SUBROUTINE MIGRHI_RMS2MV (N, TIME, VRMS, VMIG)
C
C     TRANSFORM RMS VELOCITIES TO MIGRATION VELOCITIES, I.E.,
C     MIG VELOCITY =  1.  /  (0.5  *  INTERVAL VELOCITY)
C
      DIMENSION  TIME(N), VRMS(N), VINT(N), VMIG(N)
      VMIG(1) =  2.0  /  VRMS(1)
      IF (N .LT. 2)  GO TO 99
      DO 10  I = 2, N
      HOLD  = VRMS(I)**2 * TIME(I) - VRMS(I-1)**2 * TIME(I-1)
      IF (HOLD .LE. 0.) STOP  'ERROR IN RMS VELOCITY'
 10   VMIG(I)  =  2. * SQRT((TIME(I) - TIME(I-1)) / HOLD)
      GO TO 99
C
C     TRANSFORM INTERVAL VELOCITIES TO MIGRATION VELOCITIES.
C
      ENTRY   MIGRHI_INT2MV (N, TIME, VINT, VMIG)
C
      DO 20 I =  1, N
 20   VMIG(I) =  2.0  /  VINT(I)
 99   RETURN
      END


      SUBROUTINE MIGRHI_LINVBP (NINPT, XIN, YIN, NOUTPT, XOUT, YOUT
C
C     LINEAR INTERPOLATION OF VARIABLE BASE POINTS.
C
      DIMENSION  XIN(NINPT), YIN(NINPT), XOUT(NOUTPT), YOUT(NOUTPT)
C
      DO 20 J = 1, NOUTPT
      XX      =  XOUT(J)
      IF (XX .LE. XIN(1))     THEN
              YOUT(J) =  YIN(1)
      ELSE
              DO 10 I = 2, NINPT
              IF (XX .LE. XIN(I))     GO TO  12
 10           CONTINUE
              YOUT(J) =  YIN(NINPT)
              GO TO 20
 12           SLOPE   =  (YIN(I) - YIN(I-1)) / (XIN(I) - XIN(I-1))
              YOUT(J) =  SLOPE * (XX - XIN(I))  +  YIN(I)
      ENDIF
 20   CONTINUE
      RETURN
      END


;
;     * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
;     SUBROUTINES WRITTEN IN VAX 11/780 ASSEMBLER LANGEAGE
;
;     * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
;
      .TITLE  MIGRHI_MARSUB
```

```
        .IDENT   / 65.03 /
;
;
        .ENTRY   MIGRHI_MOVE,^M<R8,R9,R10,R11>
        MOVL     @4(AP),R8                  ;   N
        MOVL     8(AP),R9                   ;   A
        MOVL     12(AP),R11                 ;   B
        CLRL     R10                        ;   SET UP INDEX
MOVE_LOOP:
        MOVL     (R9)[R10],(R11)[R10]       ;   MOVE A TO B
        AOBLSS   R8,R10,MOVE_LOOP
        RET
;
;
        .ENTRY   MIGRHI_MOVEJ,^M<R4,R5,R6,R7,R8,R9,R10,R11>
        MOVL     @4(AP),R11                 ;   N
        MOVL     8(AP),R10                  ;   A
        MOVL     12(AP),R9                  ;   B
        MOVL     @16(AP),R8                 ;   JMPA
        MOVL     @20(AP),R7                 ;   JMPB
        CLRL     R6                         ;   SET UP A-INDEX(JA)
        CLRL     R5                         ;   SET UP B-INDEX(JB)
        CLRL     R4                         ;   SET UP LOOP COUNT
MOVEJ_LOOP:
        MOVL     (R10)[R6],(R9)[R5]         ;   B(JB) = A(JA)
        ADDL2    R8,R6                      ;   INCREMENT JA
        ADDL2    R7,R5                      ;   INCREMENT JB
        AOBLSS   R11,R4,MOVEJ_LOOP
        RET
;
;
        .ENTRY   MIGRHI_STORE,^M<R8,R9,R10,R11>
        MOVL     @4(AP),R8          ;   N
        MOVL     8(AP),R9           ;   X
        MOVL     @12(AP),R11        ;   CONST
        CLRL     R10                ;   SET UP X-INDEX
STORE_LOOP:
        MOVL     R11,(R9)[R10]      ;   STORE CONST TO X(I)
        AOBLSS   R8,R10,STORE_LOOP
        RET
;
        .ENTRY   MIGRHI_VINVRS,^M<R8,R9,R10,R11>
        MOVL     @4(AP),R8                  ;   N
        MOVL     8(AP),R9                   ;   A
        MOVL     12(AP),R11                 ;   B
        CLRL     R10                        ;   SET UP INDEX
VINVRS_LOOP:
        DIVF3    (R9)[R10],#^F1.0,(R11)[R10]        ; TAKE INVERS
        AOBLSS   R8,R10,VINVRS_LOOP
        RET
;
;
        .ENTRY   MIGRHI_VRAMP,^M<R7,R8,R9,R10,R11>
        MOVL     @4(AP),R11                 ;   N
        MOVL      8(AP),R10                 ;   X
        MOVL     @12(AP),R9                 ;   X0
```

```
        MOVL    @16(AP),R8                      ;  DEL_X
        CLRL    R7                              ;  SET UP INDEX
VRAMP_LOOP:
        MOVL    R9,(R10)[R7]
        ADDF2   R8,R9
        AOBLSS  R11,R7,VRAMP_LOOP
        RET
;
;
        .ENTRY  MIGRHI_VSMULT,^M<R7,R8,R9,R10,R11>
        MOVL    @4(AP),R11                      ;  N
        MOVL    8(AP),R10                       ;  A
        MOVL    @12(AP),R9                      ;  S
        MOVL    16(AP),R8                       ;  C
        CLRL    R7                              ;  CLEAR LOOP COUNT
VSMULT_LOOP:
        MULF3   R9,(R10)[R7],(R8)[R7]           ;  C(I) = S * A(I)
        AOBLSS  R11,R7,VSMULT_LOOP
        RET
;
;
        .ENTRY  MIGRHI_MOVEJC,^M<R4,R5,R6,R7,R8,R9,R10,R11>
        MOVL    @4(AP),R11                      ;  N
        MOVL    8(AP),R10                       ;  A
        MOVL    12(AP),R9                       ;  B
        MOVL    @16(AP),R8                      ;  JMPA
        MOVL    @20(AP),R7                      ;  JMPB
        CLRL    R6                              ;  SET UP A-INDEX(JA)
        CLRL    R5                              ;  SET UP B-INDEX(JB)
        CLRL    R4                              ;  SET UP LOOP COUNT
MOVEJC_LOOP:
        MOVQ    (R10)[R6],(R9)[R5]              ;  B(JB) = A(JA)
        ADDL2   R8,R6                           ;  INCREMENT JA
        ADDL2   R7,R5                           ;  INCREMENT JB
        AOBLSS  R11,R4,MOVEJC_LOOP
        RET
;
;
        .ENTRY  MIGRHI_VVADD,^M<R7,R8,R9,R10,R11>
        MOVL    @4(AP),R11                      ;  N
        MOVL     8(AP),R10                      ;  A
        MOVL    12(AP),R9                       ;  B
        MOVL    16(AP),R8                       ;  C
        CLRL    R7                              ;  SET UP LOOP COUNT
VVADD_LOOP:
        ADDF3   (R10)[R7],(R9)[R7],(R8)[R7]          ;  C(I) = A(I) + B(I)
        AOBLSS  R11,R7,VVADD_LOOP
        RET
;
;
        .END


"
"       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
"
```

```
"       SUBROUTINES WRITTEN IN AP-120B ASSEMBLER LANGUAGE
"
"       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
"
        $TITLE  EXTPOL
        $ENTRY  EXTPOL,7
        $EXT    EXTPL1,EXTPL2
"
" S-PAD DEFINITION SECTION
"
NX      $EQU    0
IAMBF   $EQU    1
IAOLD   $EQU    2
IQBUF   $EQU    3
IWBUF   $EQU    4
NESTDF  $EQU    5
NW      $EQU    6
"
JQBUF   $EQU    IQBUF
JSW     $EQU    IWBUF
"
CTRNW   $EQU    6
CTRDFC  $EQU    7
EGTEEN  $EQU    8.
"
SP0     $EQU    0
SP1     $EQU    1
SP2     $EQU    2
SP3     $EQU    3
SP4     $EQU    4
SP5     $EQU    5
"
"       SAVE NW AND NESTDF TO DPX(4) AND DPX(-5)
"
EXTPOL: INCDPA
        MOV     NW,NW;  DPX(3)<SPFN;
                DECDPA
        DECDPA
        MOV     NESTDF,NESTDF;DPX(-4)<SPFN;
                INCDPA
"
"       DO 20 I = 1, NW
"       CALL EXTPL1 (NX, IAMBF, *, *,, JSW)
"
LOOP20: JSR     EXTPL1
"
"       DO 10 J = 1, NESTDF
"10     CALL EXTPL2 (NX, IAMBF, IAOLD, JQBUF, JSW, IIEXT(J))
"
        DECDPA
        INCDPA; DPY(-4)<DPX(-4)
        LDSPI   SP5; DB=14.             "IIEXT(1) == 14.
LOOP10: JSR     EXTPL2
        LDSPI   EGTEEN; DB=18.
        ADD     EGTEEN,SP5; DECDPA
        LDSPI   CTRDFC; DB=DPY(-4)
```

```
        DEC     CTRDFC; DPY(-4)<SPFN; INCDPA
        BGT     LOOP10
"
"       RETURN TO LOOP 20
"
        INC     JSW
        ADD     NX,JQBUF
        ADD     NX,JQBUF; INCDPA
        LDSPI   CTRNW; DB=DPX(3)
        DEC     CTRNW; DPX(3)<SPFN; DECDPA
        BGT     LOOP20
        RETURN
        $END


        $TITLE  EXTPL1
        $ENTRY  EXTPL1
        $EXT    VMUL,VSMUL
NX0     $EQU    0
IAMBF0  $EQU    1
IAOLD0  $EQU    2
JQBUF0  $EQU    3
JSW0    $EQU    4
"
IAMBF   $EQU    8.
IAOLD   $EQU    9.
JQBUF   $EQU    10.
JSW     $EQU    11.
"
SP0     $EQU    0
SP1     $EQU    1
SP2     $EQU    2
SP3     $EQU    3
SP4     $EQU    4
SP5     $EQU    5
SP6     $EQU    6
"
EXTPL1: MOV     IAMBF0,IAMBF
        MOV     IAOLD0,IAOLD
        MOV     JQBUF0,JQBUF
        MOV     JSW0,JSW
"
" CALL VSMUL (IVBUF, 1, JSW, IAMBF, 1, NX)
"
        MOV     NX0,SP5
        LDSPI   SP4;DB=1
        MOV     IAMBF,SP3
        MOV     JSW,SP2
        LDSPI   SP1;DB=1
        MOV     IAMBF,SP0                          "IVBUF = IAMBF - NX
        SUB     SP5,SP0
        JSR     VSMUL
"
" CALL VMUL (IAMBF, 1, IAMBF, 1, ISMSBF, 1, NX)
"
        MOV     SP5,SP6
```

```
        LDSPI   SP5;DB=1
        MOV     IAMBF,SP4                        "IAMSBF = IAMBF + NX
        ADD     SP6,SP4
        LDSPI   SP3;DB=1
        MOV     IAMBF,SP2
        LDSPI   SP1;DB=1
        MOV     IAMBF,SP0
        JSR     VMUL
"
" RETURN INPUT VARIABLES
"
        MOV     SP6,NX0
        MOV     IAMBF,IAMBF0
        MOV     IAOLD,IAOLD0
        MOV     JQBUF,JQBUF0
        MOV     JSW,JSW0
        RETURN
        $END


        $TITLE  EXTPL2
        $ENTRY  EXTPL2,6
        $EXT    DIV,VSMUL,VSMSA,CVAMMA
"
NX0     $EQU    0
IAMBF0  $EQU    1
IAOLD0  $EQU    2
JQBUF0  $EQU    3
JSW0    $EQU    4
JEXT0   $EQU    5
"
NX      $EQU    7.
IAMBF   $EQU    8.
IAMBF1  $EQU    IAMBF
IAMSB1  $EQU    IAMBF
IAOLD   $EQU    9.
IAOLD2  $EQU    IAOLD
IAOLD3  $EQU    IAOLD
TWO     $EQU    10.
THREE   $EQU    TWO
FOUR    $EQU    TWO
CTRBN   $EQU    TWO
JQBUF   $EQU    11.
JEXT    $EQU    12.
JSW     $EQU    13.
NX2     $EQU    14.
"
SP0     $EQU    0
SP1     $EQU    1
SP2     $EQU    2
SP3     $EQU    3
SP4     $EQU    4
SP5     $EQU    5
SP6     $EQU    6
"
EXTPL2: MOV     NX0,NX
```

```
        MOV     IAMBF0,IAMBF
        MOV     IAOLD0,IAOLD
        MOV     JQBUF0,JQBUF
        MOV     JSW0,JSW
        MOV     JEXT0,JEXT
        MOVL    NX,NX2
"
" SAVE S-PADS
"
"
" CALL CFBND2 (IAMBF, IANEW, IBNEW, JQBUF1, JQBUF, JSW, ADX)
"
        MOV     IAMBF,SP0
        MOV     IAOLD,SP1                   "IANEW = IAOLD + NX2 + NX2
        ADD     NX2,SP1
        ADD     NX2,SP1
        MOV     SP1,SP2                     "IBNEW = IANEW + NX2
        ADD     NX2,SP2
        MOV     JQBUF,SP3                   "JQBUF1 = JQBUF + 2
        INC     SP3
        INC     SP3
        MOV     JQBUF,SP4
        MOV     JSW,SP5
        LDSPI   SP6;DB=16.                  "ADX = 16 + JEXT
        ADD     JEXT,SP6
"       JSR     CFBND2
"
        LDSPI   CTRBN;DB=2                  "SET UP CTR FOR CFBND2
"
"       BOUNDARY CODES FOR MIGRATION
"
"       SEE PAGE 43-R EQ. (11)
"
AMREAL  $EQU    0
ACOF    $EQU    1
BCOF    $EQU    2
QINNER  $EQU    3
QOUTER  $EQU    4
SW      $EQU    5
ADX     $EQU    6
"
"
"DZ     ==   1
"WIMAG  ==   2
"AMIMAG ==   5
"EYEB   ==   ADX   +   1.
"
"
LOOPBN: LDMA;DB=5                                   "FETCH AMIMAG;
        LDTMA;DB=!ONE                               "FETCH 1.0
        LDMA;DB=1                                   "FETCH DZ
        MOV SW,SW;SETMA;                            "FETCH SW;
                DPY<MD                                "SAVE AMIMAG
        LDMA;DB=2                                   "FETCH WIMAG
        INC ADX;SETMA;                      "FETCH EYEB;
                DPX<MD                                "HOLD DZ
```

```
        DEC ADX;SETMA;                              "FETCH ADX;
                        FMUL DPX,MD                           "CWDZ REAL
        MOV AMREAL,AMREAL;SETMA;                         "FETCH AMREAL;
                        FMUL DPX,MD                           "CWDZ IMAG
                DPX(3)<MD;DPY(3)<MD;                     "SAVE EYEB;
                        FMUL;
                                FADD ZERO,MD                      "DEN IMAG FILL
                DPY(-4)<FM;                              "SAVE CWDZ REAL;
                 DPX<MD;                                  "HOLD ADX;
                        FMUL;
                                FADD FM,ZERO                      "DEN REAL FILL
                DPY(-3)<FM;                              "SAVE CWDZ IMAG;
                        FMUL DPX,MD;                          "CAMX REAL;
                                FADD FM,FA                        "DEN IMAG ADD
                        FMUL DPX,DPY                          "CAMX IMAG
                        FMUL
        BR      .+2;
                DPX(-4)<FM;                              "SAVE CAMX REAL;
                        FMUL;
                                FADD FM,FA                        "DEN REAL ADD
LDRBN1: BR      LOOPBN
                DPX(-3)<FM;                              "SAVE CAMX IMAG;
                                FADD FM,FA                        "DEN IMAG ADD
                DPY(-2)<FA;                              "SAVE DEN REAL;
                 DPX(-2)<FA;                              "SAVE CA+CW REAL;
                                FSUB DPY(-4),DPX(-4)      "CW-CA REAL
                DPY(-1)<FA;                              "SAVE DEN IMAG;
                        FMUL DPY(-2),DPY(-2);                 "A*A;
                                FSUBR DPX(3),FA                   "CA+CW IMAG
                DPX(-4)<FA;                              "SAVE CW-CA REAL;
                        FMUL DPY(-1),DPY(-1);                 "B*B;
                                FADD
                        FMUL;
                                FSUBR DPX(3),FA                   "CA+CW-EB IMAG
                DPY<FM;
                        FMUL;
                                FSUB DPY(-3),DPX(-3)      "CW-CA IMAG
                DPX(-1)<FA;                              "SAVE CA+CW-EB IMAG;
                                FADD FM,DPY                       "A*A + B*B
                DPY(0)<TM;                               "1.0 AT NUMERATOR;
                 DPX(-3)<FA;                              "SAVE CW-CA IMAG;
                                FADD
                DPX(0)<FA                                "A*A + B*B AT DENOM
        JSR     DIV
                                FSUB DPX(-3),DPY(3)       "CW-CA-EB IM
        BR      .+2;
                        FMUL DPX(0),DPY(-2);                  "DEN REAL
                                FADD
LDRBN2: BR      LDRBN1
                 DPX(2)<FA;                               "SAVE CW-CA-EB IM;
                        FMUL DPX(0),DPY(-1)                   "DEN IMAG
                        FMUL;
                                FADD DPX(-3),DPY(3)       "CW-CA+EB IM
                DPY(-2)<FM;                              "SAVE SCALE REAL;
                        FMUL FM,DPX(-2);                      "XIN REAL;
```

```
                                FADD
        MOV QINNER,QINNER;SETMA;                        "FETCH QINNER;
                DPY(-1)<FM;                               "SAVE SCALE IMAG;
                 DPX(-3)<FA;                               "SAVE CW-CA+EB IM;
                        FMUL FM,DPX(-2);                    "XIN IMAG-;
                                FADD DPY(3),FA                  "CA-CW+EB IM
        INC QINNER;SETMA;                               "FETCH QINNER IMAG;
                        FMUL DPY(-1),DPX(-1);               "XIN REAL;
                                FADD
                        FMUL DPY(-2),DPX(-1);               "XIN IMAG+;
                                FADD FM,ZERO                    "XIN REAL FILL
                DPY(-4)<MD;                               "SAVE QIN REAL;
                        FMUL DPY(-2),DPX(-4);               "XRT REAL;
                                FSUBR FM,ZERO                   "XIN IMAG FILL
                DPY(-3)<MD;                               "SAVE QIN IMAG;
                        FMUL DPY(-1),DPX(-4);               "XRT IMAG-;
                                FADD FM,FA                      "XIN REAL ADD
                        FMUL DPY(-1),DPX(-3);               "XRT REAL;
                                FADD FM,FA                      "XIN AIMG ADD
                DPX(-2)<FA;                               "SAVE XIN REAL;
                        FMUL DPY(-2),DPX(-3);               "XRT IMAG+;
                                FADD FM,ZERO                    "XRT REAL FILL
                DPX(-1)<FA;                               "SAVE XIN IMAG;
                        FMUL DPY(-3),FA;                    "QINX REAL-;
                                FSUBR FM,ZERO                   "XRT IMAG FILL
        BR      .+2;
                        FMUL DPY(-3),DPX(-2);               "QINX IMAG;
                                FADD FM,FA                      "XRT REAL ADD
LDRBN3: BR      LDRBN2
                        FMUL DPY(-4),DPX(-2);               "QINX REAL+;
                                FADD FM,FA                      "XRT IMAG ADD
        MOV BCOF,BCOF;SETMA;MI<FA;                      "WRITE BCOF REAL;
                        FMUL DPY(-4),DPX(-1);               "QINX IMAG;
                                FSUBR FM,ZERO                   "QINX REAL FILL
        INC BCOF;SETMA;MI<FA;                           "WRITE BCOF IMAG;
                        FMUL DPY(-2),DPX(-4);               "XOUT REAL-;
                                FADD FM,ZERO                    "QINX IMAG FILL
                        FMUL DPY(-2),DPX(2);                "XOUT IMAG-;
                                FADD FM,FA                      "QINX REAL ADD
        MOV QOUTER,QOUTER;SETMA;                        "FETCH QOUTER REAL;
                        FMUL DPY(-1),DPX(2);                "XOUT REAL-;
                                FADD FM,FA                      "QINX IMAG ADD
        INC QOUTER;SETMA;                               "FETCH QOUTER IMAG;
                DPX(-4)<FA;                               "SAVE QINX REAL;
                        FMUL DPY(-1),DPX(-4);               "XOUT IMAG+;
                                FSUBR FM,ZERO                   "XOUT REAL FILL
                DPX(-3)<FA;                               "SAVE QINX IMAG;
                        FMUL;
                                FSUBR FM,ZERO                   "XOUT IMAG FILL
                DPY(-2)<MD;                               "SAVE QOUTER REAL;
                        FMUL;
                                FSUBR FM,FA                     "XOUT REAL SUBR
                DPY(-1)<MD;                               "SAVE QOUTER IMAG;
                                FADD FM,FA                      "XOUT IMAG ADD
                DPX<FA;                                   "HOLD XOUT REAL;
```

```
                      FMUL DPY(-2),FA;               "QBNX REAL+;
                              FADD DPX(-4),ZERO         "ACOF REAL FILL
              DPX(1)<FA;                           "HOLD XOUT IMAG;
                      FMUL DPY(-2),FA;               "QBNX IMAG;
                              FADD DPX(-3),ZERO         "ACOF IMAG FILL
        BR    .+2;
                      FMUL DPY(-1),DPX(1)            "QBNX REAL-
LDRBN4: BR    LDRBN3
                      FMUL DPY(-1),DPX;              "QBNX IMAG;
                              FADD FM,FA                "ACOF REAL ADD
                      FMUL;
                              FADD FM,FA                "ACOF IMAG ADD
                      FMUL;
                              FSUBR FM,FA               "ACOF REAL SUBR
                              FADD FM,FA                "ACOF IMAG ADD
        MOV ACOF,ACOF;SETMA;MI<FA;                 "WRITE ACOF REAL;
                              FADD
        INC ACOF;SETMA;MI<FA                       "WRITE ACOF IMAG
" CALL CFBND2 (IAMBF9, IANEW9, IBNEW9, JQBUF8, JQBUF9, JSW, IDZ)
"
        MOVL    NX,NX2                     "RECREATE NX2
        BR      .+2;    MOV SP5,JSW        "RECREATE JSW
LDRBN5: BR      LDRBN4
        ADD     NX,SP0                     "IAMBF9 = IAMBF + NX - 1
        DEC     SP0
        MOV     NX2,SP6                    "MAKE SP6 = NX2-3
        DEC     SP6
        DEC     SP6
        DEC     SP6
        ADD     SP6,SP1                    "IANEW9 = (IANEW+1) + (NX2-3)
        ADD     SP6,SP2                    "IBNEW9 = (IBNEW+1) + (NX2-3)
        ADD     SP6,SP4                    "JQBUF9 = (JGBUF+1) + (NX2-3)
        MOV     SP4,SP3                    "JQBUF8 = JQBUF9 - 2
        DEC     SP3
        DEC     SP3
"       "       "SP5"                      "INVARIANT
        LDSPI   SP6;DB=16.                 "ADX ==  16  +  JEXT
        ADD     JEXT,SP6
"       JSR     CFBND2
"
        DEC     CTRBN
        BGT     LDRBN5
" CALL VSMUL (JEXT(3), 4, JSW, IAIW, 1, 4)  SCR = (0, 3, 15)
"
        MOV     JEXT,SP0                           "JEXT(3) = JEXT + 2
        INC     SP0                                "JEXT(3) = JEXT + 2
        INC     SP0
        LDSPI   SP1;    DB=4
        MOV     JSW,SP2
        LDSPI   SP3;    DB=10.                     "IAIW == 10.
        LDSPI   SP4;    DB=1
        LDSPI   SP5;    DB=4
        JSR     VSMUL
" CALL VSMSA (IAMSB1, 1, JEXT(2), JEXT, IAOLD2, 2, NY)
"
```

```
        ADD     NX,IAMBF                              "IAMSB1 = IAMBF + 1
        INC     IAMBF
        MOV     IAMSB1,SP0
        LDSPI   SP1;    DB=1
        MOV     JEXT,SP2                              "JEXT(2) = JEXT + 1
        INC     SP2
        MOV     JEXT,SP3
        LDSPI   SP5;    DB=2
        ADD     SP5,IAOLD                             "IAOLD2 = IAOLD + 2
        MOV     IAOLD2,SP4
        MOV     NX,SP6                                "NY = NX - 2
        SUB     SP5,SP6
        JSR     VSMSA
"
" CALL VSMSA (IAMSB1, 1, JEXT(6), JEXT(5), IANEW2, 2, NY)
"
        MOV     IAMSB1,SP0
"       "       "SP1"   INVARIANT
        LDSPI   FOUR;DB=4
        ADD     FOUR,SP2
        ADD     FOUR,SP3
        MOV     IAOLD2,SP4                            "IANEW2=IAOLD2+NX2*2
        ADD     NX2,SP4
        ADD     NX2,SP4
"       "       "SP5,AP6"       INVARIANT
        JSR     VSMSA
"
" CALL VSMSA (IAMSB1, 1, JEXT(10), JEXT(9), IBOLD2, 2, NY)
"
        MOV     IAMSB1,SP0
"       "       "SP1"   "INVARIANT
        ADD     FOUR,SP2
        ADD     FOUR,SP3
        MOV     IAOLD2,SP4                            "IBOLD2 = IAOLD2 + NX2
        ADD     NX2,SP4
"       "       "SP5,SP6"       "INVARIANT"
        JSR     VSMSA
"
" CALL VSMSA (IAMSB1, 1, JEXT(14), JEXT(13), IBNEW2, 2, NY)
"
        MOV     IAMSB1,SP0
"       "       "SP1"   "INVARIANT
        ADD     FOUR,SP2
        ADD     FOUR,SP3
        MOV     IAOLD2,SP4                            "IBNEW2=IAOLD2+NX2*3
        ADD     NX2,SP4
        ADD     NX2,SP4
        ADD     NX2,SP4
"       "       "SP5,SP6"       "INVARIANT
        JSR     VSMSA
"
" CALL VSMSA (IAMBF1, 1, JEXT(4), IAIW, IAOLD3, 2, NY)
"
        SUB     NX,IAMSB1                   "IAMBF1 = IAMSB1 - NX
        MOV     IAMBF1,SP0
```

```
"       "       "SP1"   "INVARIANT
        LDSPI   SP3;    DB=10.                          "IAIW == 10.
        SUB     SP3,SP2                                 "JEXT(4) = JEXT(14)-10.
        INC     IAOLD2                                  "IAOLD3 = IAOLD2 + 1
        MOV     IAOLD3,SP4
"       "       "SP5,SP6"       "INVARIANT
        JSR     VSMSA
" CALL  VSMSA (IAMBF1, 1, JEXT(8), IAIW1, IANEW3, 2, NY)
"
        MOV     IAMBF1,SP0
"       "       "SP1"   "INVARIANT
        ADD     FOUR,SP2
        INC     SP3
        MOV     IAOLD3,SP4                              "IANEW3 = IAOLD3+NX2*2
        ADD     NX2,SP4
        ADD     NX2,SP4
"       "       "SP5,SP6"       "INVARIANT
        JSR     VSMSA
"
" CALL  VSMSA (IAMBF1, 1, JEXT(12), IAIW2, IBOLD3, 2, NY)
"
        MOV     IAMBF1,SP0
"       "       "SP1"   "INVARIANT
        ADD     FOUR,SP2
        INC     SP3
        MOV     IAOLD3,SP4                              "IBOLD3 = IAOLD3+NX2
        ADD     NX2,SP4
"       "       "SP5,SP6"       "INVARIANT
        JSR     VSMSA
"
" CALL VSMSA (IAMBF1, 1, JEXT(16), ISIW3, IBNEW3, 2, NY)
"
        MOV     IAMBF1,SP0
"       "       "SP1"                                   "INVARIANT
        ADD     FOUR,SP2
        INC     SP3
        MOV     IAOLD3,SP4                              "IBNEW3=IAOLD3+NX2*3
        ADD     NX2,SP4
        ADD     NX2,SP4
        ADD     NX2,SP4
"       "       "SP5,SP6"                               "INVARIANT
        JSR     VSMSA
"
" RESET IAOLD3 TO IAOLD
"
        LDSPI   THREE;DB=3
        SUB     THREE,IAOLD3
"
" CALL CVAMMA (NX, JQBUF, IAOLD, IBOLD, IAOLD)
"
        MOV     NX,SP0
        MOV     JQBUF,SP1
        MOV     IAOLD,SP2
        MOV     IAOLD,SP3                               "IBOLD = IAOLD + NX2
        ADD     NX2,SP3
```

```
        MOV     IAOLD,SP4
        JSR     CVAMMA
"
" SOLVE TRIDIAGONAL SIMULTANEOUS EQUATIONS
"
" CALL TRIDGX (NX, JQBUF, IANEW, IBNEW, IAOLD)
"
        MOV     NX,SP0
        MOV     JQBUF,SP1
        MOV     IAOLD,SP2                          "IANEW = IAOLD+NX2*2
        ADD     NX2,SP2
        ADD     NX2,SP2
        MOV     SP2,SP3                            "IBNEW = IANEW + NX2
        ADD     NX2,SP3
        MOV     IAOLD,SP4
        MOV     JSW,SP6                            "SAVE JSW IN SP6
"
"       SUBROUTINE TRIDGX (N, T, A, B, D)
"       COMPLEX    T(N), A(N), B(N), D(N), DEN
"       DO 10 I = 2, N-1
"       DEN     =  1. / (B(I)  +  A(I) * B(I-1))
"       B(I)    =  -A(I) * DEN
"10     A(I)    =  (D(I) - A(I) * A(I-1)) * DEN
"       T(N)    =  (A(N-1) * B(N)  +  A(N)) / (1. - B(N) * B(N-1))
"       DO 20 J =  1, N-1
"       I       =  N  -  J
"20     T(I)    =  B(I) * T(I+1)  +  A(I)
"       RETURN
"END
"
N       $EQU    0
T       $EQU    1
A       $EQU    2
B       $EQU    3
D       $EQU    4
CTR     $EQU    5
"
" DATA PAD VARIABLES * * * * * * * * * * * * * * *
"
" DPX(-4,-3) : A(I) PRESTORED
" DPY(-4,-3) : B(I) PRESTORED
" DPX(-2,-1) : D(I) PRESTORED
"            : T(I+1) PRESTORED
" DPY(-2,-1) : DEN
" DPX( 0, 1) : TEMPORARY REGISTERS/ JSR DIV
" DPY( 0, 1) : TEMPORARY REGISTERS/ JSR DIV/ A(I)
" DPX( 2, 3) : A(I-1) PRESTORED
" DPY( 2, 3) : B(I-1) PRESTORED
"
" * * * * * * * * * * * * * * * * * * * * * * * *
"
        MOV B,B; SETMA                             "FETCH B(1) RL
        INC B  ; SETMA                             "FETCH B(1) IM
        INC B  ; SETMA                             "FETCH B(2) RL
        INC B  ; SETMA;                            "FETCH B(2) IM
```

```
            DPY(2)<MD                                      "  STORE B(1) RL
      MOV A,A; SETMA;                                      "FETCH A(1) RL
            DPY(3)<MD                                      "  STORE B(1) IM
      INC A  ;  SETMA;                                     "FETCH A(1) IM
            DPY(-4)<MD                                     "  STORE B(2) RL
      INC A  ;  SETMA;                                     "FETCH A(2) RL
            DPY(-3)<MD                                     "  STORE B(2) IM
      INC A  ;  SETMA;                                     "FETCH A(2) IM
            DPX(2)<MD                                      "  STORE A(1) RL
      CLR TWO;DPX(3)<MD                                    "  STORE A(1) IM
      INCL TWO;                                            "TWO = 2
            DPX(-4)<MD;                                    "  STORE A(2) RL
                    FMUL DPY(2),MD                            "A2B1 REAL+
      ADD TWO,D; SETMA;                                    "FETCH D(2) RL
            DPX(-3)<MD;                                    "  STORE A(2) IM
                    FMUL DPY(2),MD                            "A2B1 IMAG
      INC D  ; SETMA;                                      "FETCH D(2) IM,
                    FMUL DPY(3),DPX(-3)                       "A2B1 REAL-
      MOV N,CTR;
                    FMUL DPY(3),DPX(-4);                      "A2B1 IMAG
                            FADD FM,ZERO                          "DEN REAL FILL
      SUB TWO,CTR;                                         "SET UP COUNTER
            DPX(-2)<MD;                                    "  STORE D(2) RL,
                    FMUL;
                            FSUBR FM,ZERO                         "DEN IMG* FILL
      ADD N,T;
            DPX(-1)<MD;                                    "  STORE D(2) IM,
                    FMUL;
                            FSUBR FM,FA                           "DEN REAL ADD
      ADD N,T;                                             "SET UP T(N),
                            FSUBR FM,FA                           "DEN IMG* ADD
"END OF PRELIMINARY CODES
"
" DO LOOP 10
"
LOOP10: LDTMA;DB=!ONE                                      "TM = 1.0
                            FADD DPY(-4),FA                       "DEN REAL ADD
                            FSUBR DPY(-3),FA                      "DEN IMG* ADD
            DPY(-2)<FA;                                      "SAVE DEN REAL;
                            FADD
            DPY(-1)<FA;                                      "SAVE DEN IMG*;
                    FMUL DPY(-2),DPY(-2)                   "     A*A
                    FMUL DPY(-1),DPY(-1)                   "     B*B
                    FMUL                                       "PUSH
            DPX<FM;                                          "TEMPORALILY HOLD,
                    FMUL
                            FADD FM,DPX                    "     A*A  +  B*B
            DPY(0)<TM;                                       "JSR DIV(NUM)=1.0
                            FADD
            DPX(0)<FA                                        "JSR DIV(DEN)=(AA+BB)
      JSR DIV
                    FMUL DPX,DPY(-2)                           "SCALE REAL
                    FMUL DPX,DPY(-1)                           "SCALE IMAG
      BR      .+2;                                         "SKIP LADDER
            DPY(0)<DPX(-4);                                  "MOVE A(I)R TO DPY(0)
```

```
                        FMUL
LADDR2: BR      LOOP10
                DPY(-2)<FM;                             "SAVE SCALE REAL;
                        FMUL FM,DPX(-4)                   "B(I) REAL-
                DPY(-1)<FM;                             "SAVE SCALE IMAG;
                        FMUL FM,DPX(-4)                   "B(I) IMAG-
        INC B; SETMA;                                  "FETCH NEXT B(I) RL,
                DPY(1)<DPX(-3);                          "MOV A(I)I TO DPY(1);
                        FMUL DPX(-3),DPY(-1)              "B(I) REAL+
        INC B; SETMA;                                  "FETCH NEXT B(I) IM,
                        FMUL DPX(-3),DPY(-2);             "B(I) IMAG-;
                                FSUBR FM,ZERO                 "B(I) REAL FILL
        SUB TWO,B;                                     "BACK TO B(I),
                        FMUL DPY(0),DPX(2);               "A1A2 REAL+;
                                FSUBR FM,ZERO                 "B(I) IMAG FILL
                DPY(-4)<MD;                              "STORE B(I+1) RL,
                        FMUL DPY(0),DPX(3);               "A1A2 IMAG;
                                FADD FM,FA                    "B(I) REAL ADD
                DPY(-3)<MD;                              "STORE B(I+1) IM,
                        FMUL DPY(1),DPX(3);               "A1A2 REAL-;
                                FSUBR FM,FA                   "B(I) IMAG ADD
        DEC B; SETMA;MI<FA;                            "WRITE B(I) RL,
                DPY(2)<FA;                               "FOR NEXT B(I-1) RL,
                        FMUL DPY(1),DPX(2);               "A1A2 IMAG;
                                FSUBR FM,ZERO                 "A1A2 REAL FILL
        INC B; SETMA;MI<FA;                            "WRITE B(I) IM,
                DPY(3)<FA;                               "FOR NEXT B(I-1) IM,
                        FMUL;
                                FSUBR FM,ZERO                 "A1A2 IMAG FILL
        BR      .+2;                                   "SKIP THE LADDER,
                        FMUL;
                                FADD FM,FA                    "A1A2 REAL ADD
LADDR1: BR      LADDR2
        ADD TWO,B;                                     "GO B(I+1),
                                FSUBR FM,FA                   "A1A2 IMAG ADD
                                FADD DPX(-2),FA               "ADEN REAL ADD
                                FADD DPX(-1),FA               "ADEN IMAG ADD
        INC A;SETMA;                                   "FETCH A(I+1) RL,
                DPX<FA;                                  "TEMORARY,
                        FMUL DPY(-2),FA;                  "CA REAL+;
                                FADD
        INC A;SETMA;                                   "FETCH A(I+1) IM,
                        FMUL DPY(-2),FA                   "CA IMAG
        INC D;SETMA;                                   "FETCH D(I+1) RL,
                        FMUL DPY(-1),FA                   "CA REAL-
        INC D;SETMA;                                   "FETCH D(I+1) IM,
                DPX(-4)<MD;                              "STORE A(I+1) RL,
                        FMUL DPY(-1),DPX;                 "CA IMAG;
                                FADD FM,ZERO                  "CA REAL FILL
        SUB TWO,A;                                     "BACK TO A(I),
                DPX(-3)<MD;                              "STORE A(I+1) IM,
                        FMUL DPX(-4),DPY(2);              "A3B2 REAL+;
                                FADD FM,ZERO                  "CA IMAG FILL
                DPX(-2)<MD;                              "STORE D(I) RL,
                        FMUL DPX(-4),DPY(3);              "A3B2 IMAG;
```

```
                                   FSUBR FM,FA                "CA REAL ADD
           DPX(-1)<MD;                                    "STORE D(I) IM,
                       FMUL DPX(-3),DPY(3);                 "A3B2 REAL-;
                                   FADD FM,FA                 "CA IMAG ADD
       DEC A;SETMA;MI<FA;                               "WRITE A(I) RL,
           DPX(2)<FA;                                     "FOR NEXT A(I-1) RL,
                       FMUL DPX(-3),DPY(2);                 "A3B2 IMAG;
                                   FADD FM,ZERO               "DEN REAL FILL
       INC A;SETMA;MI<FA;                               "WRITE A(I) IM,
           DPX(3)<FA;                                     "FOR NEXT A(I-1) IM,
                       FMUL;
                                   FSUBR FM,ZERO              "DEN IMG* FILL
       DEC CTR;
                       FMUL;
                                   FSUBR FM,FA                "DE REAL ADD
       ADD TWO,A;BGT LADDR1;                            "GO A(I+1), BRANCH
                                   FSUBR FM,FA                "DEN IMG* ADD
" END OF DO LOOP 10
"
" STATEMENT 15.
"
       LDTMA;DB=!ONE
           DPX(-2)<DPY(-4)                                "MOVE B(N)R TO DPX
           DPX(-1)<DPY(-3)                                "MOVE B(N)I TO DPX
                       FMUL DPX(-2),DPY(2)                  "RL +
                       FMUL DPX(-2),DPY(3);                 "IM
                                   FADD TM,ZERO               "FILL 1.0
                       FMUL DPX(-1),DPY(3);                 "RL -
                                   FADD
                       FMUL DPX(-1),DPY(2);                 "IM,
                                   FSUBR FM,FA                "1. - RL
                       FMUL;
                                   FADD FM,ZERO               "FILL IM
                       FMUL;
                                   FADD FM,FA                 "ADD RL
                                   FADD FM,FA                 "ADD IM
           DPY(-2)<FA;                                    "SAVE A,
                                   FADD
           DPY(-1)<FA;                                    "SAVE B,
                       FMUL DPY(-2),DPY(-2)                 "A*A
                       FMUL DPY(-1),DPY(-1)                 "B*B
                       FMUL
           DPX<FM;                                        "TEMPORARY SAVE
                       FMUL
                                   FADD FM,DPX                "A*A + B*B
           DPY(0)<TM;                                     "PUT 1.0 ON NUMERATO
                                   FADD
           DPX(0)<FA                                      "PUT A*A+B*B ON DENO
       JSR         DIV                                  "FORM INVERSE
           FMUL DPX,DPY(-2)
           FMUL DPX,DPY(-1)
" DENOMINATOR COMPLETED
"
" THE NUMERATOR CODES ARE:
"
```

```
                  FMUL DPX(2),DPY(-3)            "IM
            DPY(-2)<FM;                        "SAVE DEN RL,
                  FMUL DPX(2),DPY(-4)            "RL +
            DPY(-1)<FM;                        "SAVE DEN IM,
                  FMUL DPX(3),DPY(-4)            "IM
                  FMUL DPX(3),DPY(-3);           "RL -,
                        FADD FM,ZERO                "FILL IM
                  FMUL;
                        FADD FM,ZERO                "FILL RL
                  FMUL;
                        FADD FM,FA                  "ADD IM
                        FSUBR FM,FA                 "ADD RL
                        FADD DPX(-3),FA             "ADD IM
                        FADD DPX(-4),FA             "ADD RL
            DPX<FA;                            "TEMPORARY,
                  FMUL DPY(-2),FA;               "IM,
                        FADD
                  FMUL DPY(-2),FA                "RL +
                  FMUL DPY(-1),FA                "IM
      MOV N,CTR;                             "COUNTER FOR LOOP 20,
                  FMUL DPY(-1),DPX;              "RL -,
                        FADD FM,ZERO                "FILL IM
      DEC CTR;                               "COUNTER FOR LOOP 20,
                  FMUL;
                        FADD FM,ZERO                "FILL RL
      SUB TWO,B;                             "SET UP B FOR LP 20,
            DPY(-3)<DPY(3);                    "FOR NEXT B(I) IM,
                  FMUL;
                        FADD FM,FA                  "ADD IM
      DEC B;                                 "SET UP B FOR LP 20,
            DPY(-4)<DPY(2);                    "FOR NEXT B(I) RL,
                        FSUBR FM,FA                 "ADD RL
      DEC T;SETMA;MI<FA;                     "WRITE T(N) IM,
            DPX(-1)<FA;                        "SAVE T(N) FOR NEXT,
                        FADD
      DEC T;SETMA;MI<FA;                     "WRITE T(N) RL,
            DPX(-2)<FA                         "SAVE T(N) RL.
" END OF STATEMENT 15.
"
" DO LOOP 20
"
      DEC A;                                 "ADJUST A-INDEX
                  FMUL DPX(-1),DPY(-4)           "IM
                  FMUL DPX(-2),DPY(-4)           "RL +
LOOP20: DEC A;SETMA;                         "FETCH A(I) IM,
                  FMUL DPX(-2),DPY(-3)           "IM
      DEC A;SETMA;                           "FETCH A(I) RL,
                  FMUL DPX(-1),DPY(-3);          "RL -
                        FADD FM,ZERO                "FILL IM
      DEC B;SETMA;                           "FETCH B(I-1) IM,
                  FMUL;
                        FADD FM,ZERO                "FILL RL
      DEC B;SETMA;                           "FETCH B(I-1) RL,
            DPX(-3)<MD;                        "STORE A(I) IM,
                  FMUL;
```

```
                          FADD FM,FA                "ADD IM
        DPX(-4)<MD;                               "STORE A(I) RL,
                          FSUBR FM,FA               "ADD RL
        DPY(-3)<MD;                               "STORE B(I-1) IM,
                          FADD DPX(-3),FA           "ADD IM
        DPY(-4)<MD;                               "STORE B(I-1) RL,
                          FADD DPX(-4),FA           "ADD RL
    DEC T;SETMA;MI<FA;                           "WRITE T(I) IM,
        DPX(-1)<FA;                               "SAVE T(I-1) IM,
                          FADD
    DEC T;SETMA;MI<FA;                           "WRITE T(I) RL,
        DPX(-2)<FA                                "SAVE T(I-1) RL
    DEC CTR;
                     FMUL DPX(-1),DPY(-4)          "IM
    BGT LOOP20;                                  "SEE IF DONE????
                     FMUL DPX(-2),DPY(-4)          "RL +
" END OF DO LOOP 20.
"
" RECREATE INPUT VARIABLES
"
        MOV     NX,NX0
        DEC     IAMBF1
        MOV     IAMBF,IAMBF0
        MOV     IAOLD,IAOLD0
        MOV     JQBUF,JQBUF0
        MOV     SP6,JSW0
        MOV     JEXT,JEXT0
        RETURN
        $END


        $TITLE  CVAMMA
        $ENTRY  CVAMMA
"
"       D(I) = A(I) * (Q(I-1) + Q(I+1))  +  B(I) * Q(I), I = 2, N
"
N       $EQU    0
Q       $EQU    1
A       $EQU    2
B       $EQU    3
D       $EQU    4
CTR     $EQU    0
TWO     $EQU    5
"
"       DP REGISTER MAP    * * * * * * * * * * * * * * *
"
"       DPX(-4,-3) : Q(I-1), Q(I)
"       DPY(-4,-3) : Q(I),   Q(I+1)
"       DPX(-2,-1) : Q(I+1)
"       DPY( 2, 3) : A(I)
"
CVAMMA: MOV Q,Q;SETMA                            "FETCH Q(1) RL
        INC Q;SETMA                              "FETCH Q(1) IM
        INC Q;SETMA                              "FETCH Q(2) RL
        INC Q;SETMA;                             "FETCH Q(2) IM,
```

```
                              FADD ZERO,MD              "FILL QSUM RL
      INC Q;SETMA;                                    "FETCH Q(3) RL,
                              FADD ZERO,MD              "FILL QSUM IM
      INC Q;SETMA;                                    "FETCH Q(3) IM,
              DPX(-4)<MD                                "SAVE Q(2) RL
      INC A;                                          "SKIP A(1),
              DPX(-3)<MD                                "SAVE Q(2) IM
      INC A;SETMA;                                    "FETCH A(2) RL,
              DPX(-2)<MD                                "SAVE Q(3) RL
      INC A;SETMA;                                    "FETCH A(2) IM,
              DPX(-1)<MD;                               "SAVE Q(3) IM,
                              FADD DPX(-2),FA           "QSUM RL
      INC D;                                          "SKIP D(1),
                              FADD DPX(-1),FA           "QSUM IM
      DEC CTR;                                        "CTR = N-1,
              DPY(2)<MD                                 "SAVE A(2) RL
      DEC CTR;                                        "CTR = N-2,
              DPY(3)<MD                                 "SAVE A(2) IM
      INC B                                           "SKIP B(1)
"
LOOP:         DPX<FA;                                   "HOLD QSUM RL,
                      FMUL DPY(2),FA;                     "A(I)*QSUM RP,
                              FADD                      "PUSH QSUM IM
      INC B;SETMA;                                    "FETCH B(I) RL,
              DPY(-4)<DPX(-2);                          "MOVE Q(I+1) TO Q(I)
                      FMUL DPY(2),FA                      "A(I)*QSUM IM
      INC B;SETMA;                                    "FETCH B(I) IM,
              DPY(-3)<DPX(-1);                          "MOVE Q(I+1) TO Q(I)
                      FMUL DPY(3),FA                      "A(I)*QSUM RM
                      FMUL DPY(3),DPX;                    "A(I)*QSUM IM,
                              FADD FM,ZERO              "FILL AQSUM RL
      INC Q;SETMA;                                    "FETCH Q(I+2) RL,
              DPY<MD;                                   "HOLD B(I) RL,
                      FMUL DPX(-4),MD;                    "Q(I)*B(I) RP,
                              FADD FM,ZERO              "FILL AQSUM IM
      INC Q;SETMA;                                    "FETCH Q(I+2) IM,
                      FMUL DPX(-4),MD;                    "Q(I)*B(I) IM,
                              FSUBR FM,FA               "ADD AQSUM RL
      INC A;SETMA;                                    "FETCH A(I+1) RL,
                      FMUL DPX(-3),MD;                    "Q(I)*B(I) RM,
                              FADD FM,FA                "ADD AQSUM IM
      INC A;SETMA;                                    "FETCH A(I+1) IM,
              DPX(-2)<MD;                               "SAVE Q(I+2) RL,
                      FMUL DPX(-3),DPY;                   "Q(I)*B(I) IM,
                              FADD FM,FA                "ADD AQSQB RL
              DPX(-1)<MD;                               "SAVE Q(I+2) IM
                      FMUL;                               "PUSH QB RL,
                              FADD FM,FA                "ADD AQSQB IM
              DPY(2)<MD;                                "SAVE A(I+1) RL,
                      FMUL;                               "PUSH QB IM,
                              FSUBR FM,FA               "ADD AQSQB RM
              DPY(3)<MD;                                "SAVE A(I+1) IM,
                              FADD FM,FA                "ADD AQSQB IM
      INC D;SETMA;MI<FA;                              "WRITE D(I) RL,
                              FADD DPX(-4),ZERO         "FILL QSUM RP
```

```
        INC D;SETMA;MI<FA;                                    "WRITE D(I) IM,
                                FADD DPX(-3),ZERO               "FILL QSUM IM
        DEC CTR;                                              "CTR = CTR - 1,
                DPX(-4)<DPY(-4);                                "MOVE Q2 TO Q1 RL,
                                FADD DPX(-2),FA                 "ADD QSUM RL
        BGT LOOP;                                             "SEE IF DONE????,
                DPX(-3)<DPY(-3);                                "MOVE Q2 TO Q1 IM,
                                FADD DPX(-1),FA                 "ADD QSUM IM

        RETURN
        $END


        $TITLE  CVCSML
        $ENTRY  CVCSML,4
N       $EQU    0
X       $EQU    1
S       $EQU    2
Z       $EQU    3
CVCSML: MOV S,S;SETMA                                         "GET SR
        MOV X,X;SETMA                                         "GET X1R
        INC S;SETMA                                           "GET SI
        INC X;SETMA;                                          "GET X1I;
                DPY<MD                                          "SAVE SR
                DPX<MD;                                         "HOLD X1R;
                        FMUL DPY,MD                               "X1R*SR
        INC X;SETMA;                                          "GET X2R;
                DPY(1)<MD;                                      "SAVE SI;
                        FMUL DPX,MD                               "X1R*SI
        INC X;SETMA;                                          "GET X2I;
                DPX(1)<MD;                                      "HOLD X1I;
                        FMUL DPY(1),MD                            "X1I*SI
                        FMUL DPX(1),DPY;                          "X1I*SR;
                                FADD FM,ZERO                        "FILL Z1 REAL
        DEC Z;                                                "ADJUST Z-INDEX;
                DPX<MD;                                         "HOLD X2R;
                        FMUL;
                                FADD FM,ZERO                        "FILL Z1 IMAG
        INC X;SETMA;                                          "GET X3R;
                DPX(1)<MD;                                      "HOLD X2I;
                        FMUL DPX,DPY;                             "X2R*SR;
                                FSUBR FM,FA                         "Z1 REAL ADD
"
"
LOOP:   INC X;SETMA;                                          "GET X(I+2) IMAG;
                        FMUL DPX, DPY(1);                         "X(I+1)R * SI;
                                FADD FM,FA                          "Z(I) IMAG ADD
        INC Z;SETMA;MI<FA;                                    "PUT Z(I) REAL;
                        FMUL DPX(1),DPY(1);                       "X(I+1)I * SI;
                                FADD
        INC Z;SETMA;MI<FA;                                    "PUT Z(I) IMAG;
                DPX<MD;                                         "HOLD X(I+2) REAL;
                        FMUL DPX(1),DPY;                          "X(I+1)I * SR;
                                FADD FM,ZERO                        "Z(I+1) R-FILL
        DEC N;                                                "AJUST LOOP COUNT;
```

```
                DPX(1)<MD;
                        FMUL;
                                FADD FM,ZERO            "Z(I+1) I-FILL
        INC X;SETMA;                            "GET X(I+3) REAL;
                        FMUL DPX,DPY;               "X(I+2)R * SR;
                                FSUBR FM,FA;            "Z(I+1) R-ADD
        BGT LOOP                                "GO BACK TO THE LOOP.
        RETURN
        $END


" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
"       SUBROUTINE RFFTMM (NX, NT, NTFFT, NW2, IWO, NXNT, NXNTFF)
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
        $TITLE  IMAGEW
        $ENTRY  IMAGEW,4
        $EXT    SVE
"
"       DO 10 IX = 1, NX
"10     CALL SVE (IQ+IX+IX-2, NX2, IG+IX-1, NW)
"
"
NXIN    $EQU    0
NWIN    $EQU    1
QIN     $EQU    2
GIN     $EQU    3
"
NX      $EQU    15.
NW      $EQU    14.
Q       $EQU    13.
TWO     $EQU    12.
"
SP0     $EQU    0
SP1     $EQU    1
SP2     $EQU    2
SP3     $EQU    3
"
IMAGEW: MOV NXIN,NX                     "SAVE NX
        MOV NWIN,NW                     "SAVE NW
        MOV QIN,Q                       "SAVE Q
        MOV GIN,SP2                     "SAVE G
        MOV NX,SP1                      "NX2 = NX + NX
        ADD NX,SP1
        LDSPI TWO; DB=2                 "TWO = 2
        MOV G,SP0
LOOP:   MOV NW,SP3
        JSR SVE
        ADD TWO,Q
        INC SP2
        DEC NX
        MOV Q,SP0; BGT LOOP
        RETURN
        $END
```

```
        $TITLE  PHSHFT
        $ENTRY  PHSHFT,4
        $EXT    CVCSML
"
"       DO 10 J = 1, NW
"10     CALL CVCSML (NX, IQ+NX2*(J-1), IS+J+J-2, IQ+NX2*(J-1))
"
NXIN    $EQU    0
NWIN    $EQU    1
SIN     $EQU    2
QIN     $EQU    3
"
SP0     $EQU    0
SP1     $EQU    1
SP2     $EQU    2
SP3     $EQU    3
"
NX      $EQU    14.
NW      $EQU    13.
Q       $EQU    12.
NX2     $EQU    11.
"
PHSHFT: MOV     NXIN,NX                     "SAVE NX
        MOV     NWIN,NW                     "SAVE NW
        MOV     QIN,Q                       "SAVE Q
        MOVL    NX,NX2                      "NX2 = NX + NX
        MOV     NX,SP0                      "SET UP SP0
LOOP:   MOV     Q,SP1                       "SET SP1
        MOV     Q,SP3
        JSR     CVCSML
        ADD     NX2,Q
        INC     SP2                         "SP2 INCREMENTED 1 BY CVCSML.
        DEC     NW
        MOV     NX,SP0; BGT LOOP
        RETURN
        $END


        $TITLE  RFFTMM
        $ENTRY  RFFTMM,7
        $EXT    VCLR            "SCRATCH = SP0, SP15
        $EXT    VMOV            "SCRATCH = SP0, SP2, SP15
        $EXT    RFFT            "SCRATCH = SP2 TO SP15
"
" INPUT ARGUMENT DESCRIPTION.
"
INX     $EQU    0
INT     $EQU    1
INTFFT  $EQU    2
INW2    $EQU    3
IIW0    $EQU    4
INXNT   $EQU    5
INXNTF  $EQU    6
"
" INTERNAL SCRATCH VARIABLES DESCRIPTION.
"
```

```
NX      $EQU    7.
NT      $EQU    8.
NTFFT   $EQU    9.
NW2     $EQU    10.
IW0     $EQU    11.
IMDSRC  $EQU    IW0
IMDDST  $EQU    12.
NCLEAR  $EQU    NW2
CTR     $EQU    14.
"
"
SP0     $EQU    0
SP1     $EQU    1
SP2     $EQU    2
SP3     $EQU    3
SP4     $EQU    4
SP5     $EQU    5
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
"       NCLEAR  =  NTFFT -  NT
"       IMDSRC  =  NXNT  -  1
"       IMDDST  =  NXNTFF  -  1
"       DO 10 J = 2, NX
"       CALL VCLR (IMDDST, -1, NCLEAR)
"       IMDDST  =  IMDDST  -  NCLEAR
"       CALL VMOV (IMDSRC, -1, IMDDST, -1, NT)
"       IMDSRC  =  IMDSRC  -  NT
"10     IMDDST  =  IMDDST  -  NT
"       CALL VCLR (IMDDST, -1, NCLEAR)
"
"
RFFTMM: INCDPA;                                   "GO TO NEXT DPA;
                MOV INT,NT                          "MOVE NT
                MOV IIW0,IW0;                       "MOVE IW0;
                        DPX(3)<SPFN                     "SAVE IW0
        INCDPA;                                   "GO TO NEXT DAP;
                MOV INX,NX;                         "MOVE NX;
                        DPY(3)<SPFN                     "SAVE NX
                MOV INTFFT,NTFFT;                   "MOVE NTFFT;
                        DPX(3)<SPFN                     "SAVE NTFFT
        DECDPA;                                   "BACK DPA;
                MOV INW2,NW2;                       "MOVE NW2;
                        DPY(3)<SPFN                     "SAVE NW2
        DECDPA;                                   "BACK TO ORIGIN;
                MOV NT,SP4                          "VMOV SP4
                LDSPI SP1; DB=-1                    "VCLR SP1
                LDSPI SP3; DB=-1                    "VCLR SP3
                MOV INXNT,IMDSRC
                MOV INXNTF,IMDDST
                DEC IMDSRC
                DEC IMDDST
                MOV NTFFT,NCLEAR
                SUB NT,NCLEAR
                MOV NX,CTR
                DEC CTR
```

```
LOOP10: BEQ EXIT10;
                MOV IMDDST,SP0
                MOV NCLEAR,SP2
        JSR     VCLR
                SUB NCLEAR,IMDDST
                MOV IMDSRC,SP0
                MOV IMDDST,SP2
        JSR     VMOV
                SUB NT,IMDSRC
                SUB NT,IMDDST
                DEC CTR;
        BR      LOOP10
EXIT10:         MOV IMDDST,SP0
                MOV NCLEAR,SP2
        JSR     VCLR
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
"       IMDSRC  =  0
"       DO 20 J =  1, NX
"       CALL RFFT (IMDSRC, NTFFT, 1)
"20     IMDSRC  =  IMDSRC  +  NTFFT
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
        DECDPA;                                         "GO DPA(-1)
                CLR SP0
        INCDPA;                                         "GO DPA;
                MOV NX,CTR;                               "SET UP COUNTER;
                        DPX(-4)<SPFN                            "SAVE COUNTER
                MOV NTFFT,SP1
LOOP20:         LDSPI SP2; DB=1
        JSR     RFFT
        DECDPA;
                ADD SP1,SP0
                LDSPI CTR; DB=DPX(-4)
        INCDPA;
                DEC CTR; DPX(-4)<SPFN
        BGT     LOOP20
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
"       IMDSRC  =  IW0
"       IMDDST  =  IW0
"       DO 30 J = 2, NX
"       IMDSRC  =  IMDSRC  +  NTFFT
"       IMDDST  =  IMDDST  +  NW2
"30     CALL VMOV (IMDSRC, 1, IMDDST, 1, NW2)
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
"       GET IMDSRC(=IW0), CTR(=NX), NTFFT, AND SP4(=NW2)
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
        INCDPA; MOV SP1,NTFFT
```

```
                LDSPI IMDSRC; DB=DPX(3)
        INCDPA;
                LDSPI CTR;    DB=DPY(3)
        DECDPA;
                LDSPI SP4;    DB=DPY(3)
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
"       ADJUST COUNTER, RESET DPA, SET SP1, SP2 TO 1, 1.
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
        DECDPA;
                DEC CTR
                LDSPI SP1; DB=1
                LDSPI SP3; DB=1
                MOV IMDSRC,IMDDST
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
"       DO 30 J = 2, NX
"
" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
"
LOOP30: BEQ DONE;
                ADD NTFFT,IMDSRC
                ADD SP4,  IMDDST
                MOV IMDSRC,SP0
                MOV IMDDST,SP2
        JSR     VMOV
        BR      LOOP30; DEC CTR
DONE:   RETURN
        $END
```