# SEISMIC IMAGING ON MASSIVELY PARALLEL COMPUTERS

Curtis C. Ober, Ron Oldfield, David E. Womble, John VanDyke, and Sudip Dosanjh *
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185
email:dewombl@cs.sandia.gov



**Keywords:** Parallel computing, Computer performance, Seismic imaging.

## ABSTRACT

Fast, accurate imaging of complex, oil-bearing geologies, such as overthrusts and salt domes, is the key to reducing the costs of domestic oil and gas exploration. Geophysicists say that the known oil reserves in the Gulf of Mexico could be significantly increased if accurate seismic imaging beneath salt domes was possible. A range of techniques exist for imaging these regions, but the highly accurate techniques involve the solution of the wave equation and are characterized by large data sets and large computational demands. Massively parallel computers can provide the computational power for these highly accurate imaging techniques.

A brief introduction to seismic processing will be presented, and the implementation of a seismic-imaging code for distributed memory computers will be discussed. The portable code, Salvo, performs a wave-equation-based, 3-D, prestack, depth imaging and currently runs on the Intel Paragon and the Cray T3D. It uses MPI for portability, and has sustained 22 Mflops/sec/proc (compiled FORTRAN) on the Intel Paragon.

## INTRODUCTION

A key to reducing the risks and costs of associated with oil and gas exploration is the fast, accurate imaging of complex geologies. Prestack depth migration generally yields the most accurate images, and one approach to this is to solve the scalar wave equation using finite differences. As part of an ongoing Advanced Computational Technologies Initative (ACTI) project, a finite difference, 3-D prestack, depth migration code for a range of platforms has been developed. The goal of this work is to demonstrate that massively parallel computers (thousands of processors) can be used efficiently for seismic imaging, and that sufficient computing power exists (or soon will exist) to make finite difference, prestack, depth migration practical for oil and gas exploration.

Several problems have been addressed to obtain an efficient code. These include efficient I/O, efficient parallel tridiagonal solves, and high single-node performance. Furthermore, portability considerations have restricted the code to the use of high-level programming languages and interprocessor communications using MPI.

Efficient I/O is one of the problems that have been addressed. The initial input to our seismic imaging code is a sequence of seismic traces, which are scattered across all the raids in the I/O subsystem and may or may not be in any particular order. The traces must be read, Fourier transformed and redistributed to the appropriate processors for computation. In Salvo, the input is performed by a subset of the nodes, while the remaining nodes perform the pre-computations in the background.

A second problem that has been addressed is the efficient use of thousands of processors. There are a couple types of parallelism available in a finite difference solution of the wave equation for seismic imaging. The first and most obvious is frequency parallelism; however, this limits the available parallelism to hundreds of processors and restricts the size of problem that can be solved in-core. Spatial parallelism addresses both of these problems, but introduces another issue. Specifically, an alternating direction implicit (ADI) method (or a variant) is typically used for the solution at each depth level, which means that tridiagonal solves must be parallelized. Parallelizing individual tridiagonal solves is difficult, so the problem has been handled by pipelining many tridiagonal solves.

The remainder of this paper describes in more detail the algorithms and implementation used in Salvo and presents some numerical results.

*This work was supported by the United States Department of Energy under Contract DE-AC04-94AL85000, and by Mathematical Information and Computational Sciences.

## IMAGING ALGORITHM

The following development is an industry-standard approach [Claerbout 1985, Yilmaz 1987, Li 1991], and is repeated here for reference. The equation used to model the propagation of pressure waves through the earth is

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

where $P(x, y, z, t)$ is pressure, and $v(x, y, z)$ is the acoustic velocity of the media. This equation is transformed to a Helmholtz equation and then to the paraxial wave equation,

$$\frac{\partial P}{\partial z} = \left\{ \pm \frac{i\omega}{v} \left[ 1 + \frac{v^2}{\omega^2} \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \right]^{1/2} \right\} P, \tag{2}$$

where $\omega$ is the frequency of the propagating wave. The positive and negative signs correspond to upcoming and downgoing wave fields.

The evaluation of the square-root operator is numerically difficult, hence it is approximated by a series that has its origin in a continued fraction expansion [Claerbout 1985, p. 84] [Yilmaz 1987, p. 513]. The continued fraction expansion can be represented by ratios of polynomials [Ma 1981] and the polynomial coefficients can be optimized for propagation angle [Lee and Suh 1985]. With these approximations, the paraxial wave equation can be written as

$$\frac{\partial P}{\partial z} = \pm \frac{i\omega}{v} \left[ 1 + \sum_{\ell=1}^{m} \frac{\alpha_\ell S}{1 + \beta_\ell S} \right] P, \tag{3}$$

where

$$S = S_x + S_y = \frac{v^2}{\omega^2}\frac{\partial^2}{\partial x^2} + \frac{v^2}{\omega^2}\frac{\partial^2}{\partial y^2} \tag{4}$$

and $\alpha_\ell$ and $\beta_\ell$ are the expansion coefficients [Lee and Suh 1985].

The terms of the expansion,

$$\begin{aligned} \frac{\partial P}{\partial z} = \pm \frac{i\omega}{v} \Bigg[ & P + \frac{\alpha_1 S}{1 + \beta_1 S} P + \frac{\alpha_2 S}{1 + \beta_2 S} P + \\ & \cdots + \frac{\alpha_\ell S}{1 + \beta_\ell S} P + \cdots + \frac{\alpha_m S}{1 + \beta_m S} P \Bigg], \end{aligned} \tag{5}$$

are separated by the method of fractional steps [Fletcher 1988] and a sequence of $(m + 1)$ equations are solved (*i.e.*, one for each term on the right-hand side of Eq. (5)). The solution from one step in the sequence is fed into the next step until the last step produces the solution at the next depth level.

The solution to the first equation,

$$\frac{\partial P_\ell}{\partial z} = \pm \frac{i\omega}{v} P_\ell,$$

is simply a complex exponential. The primary computational load is the solution of equations of the form

$$\frac{\partial P_\ell}{\partial z} = \pm \frac{i\omega}{v} \frac{\alpha_\ell S}{1 + \beta_\ell S} P_\ell.$$

Another operator splitting similar to the method of fractional steps is performed but this time in the $x$ and $y$ directions. To convert the operator, $S$, to a linear combination of $S_x$ and $S_y$, we write

$$\begin{aligned} \frac{\partial P_\ell}{\partial z} = \pm \frac{i\omega}{v} \Bigg[ & \frac{\alpha_\ell S_x}{1 + \beta_\ell S_x} + \frac{\alpha_\ell S_y}{1 + \beta_\ell S_y} \\ & + \frac{-2\alpha_\ell \beta_\ell S_x S_y}{(1 + \beta_\ell S)(1 + \beta_\ell S_x)(1 + \beta_\ell S_y)} \\ & + \frac{-\alpha_\ell \beta_\ell^2 (S_x S_x S_y + S_x S_y S_y)}{(1 + \beta_\ell S)(1 + \beta_\ell S_x)(1 + \beta_\ell S_y)} \Bigg] P_\ell, \end{aligned}$$

or

$$\frac{\partial P_\ell}{\partial z} \approx \pm \frac{i\omega}{v} \left[ \frac{\alpha_\ell S_x}{1 + \beta_\ell S_x} + \frac{\alpha_\ell S_y}{1 + \beta_\ell S_y} \right] P_\ell. \tag{6}$$

The operators in Eq. (6) are once again split by method of fractional steps to produce the sequence of equations

$$\frac{\partial P_\ell}{\partial z} = \pm \frac{i\omega}{v} \frac{\alpha_\ell S_x}{1 + \beta_\ell S_x} P_\ell \tag{7}$$

and

$$\frac{\partial P_\ell}{\partial z} = \pm \frac{i\omega}{v} \frac{\alpha_\ell S_y}{1 + \beta_\ell S_y} P_\ell, \tag{8}$$

These equations produce tridiagonal systems that can be solved efficiently.

To correct for errors produced by neglecting the last two terms in Eq. 6, a filter is used [Graves and Clayton 1990]. This amounts to solving the equation

$$\frac{\partial P}{\partial z} = \mp i 2\alpha\beta \frac{\omega^3}{v^3} \frac{\partial^2}{\partial x^2} \frac{\partial^2}{\partial y^2} P. \tag{9}$$

Finally we apply the absorbing boundary conditions described in [Clayton and Engquist 1980].

### I/O

Seismic datasets consisting of recorded pressure waves are often large. Even if the computations can be performed in-core, the time required to read the initial seismic data, read the velocity models and write the images is substantial. In Salvo, the effect of the "I/O bottleneck" is mitigated by performing preliminary computations and data redistribution using nodes not directly involved in the I/O.

The trace dataset is distributed across many disks to increase the total disk-to-memory bandwidth. A subset of the available nodes is assigned to handle the I/O, and each node in this subset, termed an I/O node, is assigned to handle I/O from one file system.

The remaining nodes, termed compute nodes, can complete computations and communications necessary before the migration can begin. Each compute node is assigned to an I/O node, and performs the pre-computations on the data read by its I/O node. Currently the pre-computation comprises fast Fourier transforms (FFTs), but other computations could also be performed. If we assign a sufficient number of compute nodes to each I/O node, the time to read a block of seismic data will be greater than the time required to compute the FFTs and distribute the frequencies to the correct nodes for later computations. Thus, the computation time will be hidden behind the I/O time.

A model of the I/O and pre-computations and communications can be developed to determine the proper balance between I/O nodes and compute nodes. The I/O node begins by reading a block of data from a disk and distributing this data to a set of compute nodes. The time required for this operation is approximately

$$\Phi b + c\left[\alpha + \beta\left(\frac{b}{c}\right)\right],$$

where $\Phi$ is the disk bandwidth, $b$ is the blocksize, $\alpha$ is communication latency, $\beta$ is the time to communicate one byte, and $c$ is the number of compute nodes.

The time to compute the FFTs, $\tau$, is machine and library dependent. Because $\tau$ can be measured easily on most platforms, it is not further decomposed into computational rates.

After completing an FFT, the compute node must distribute each frequency to the processor assigned to perform the seismic migration for that $x$ and $y$ location and frequency. The time to evenly distribute the frequencies of one trace is approximated by

$$p_\omega\left[\alpha + \beta\left(\frac{ng}{p_\omega}\right)\right],$$

where $p_\omega$ is the number of nodes at a specific $x$ and $y$ location, that is, the number of nodes in the frequency decomposition, $n$ is the number of words in a frequency trace, $g$ is the size of one word of data ($g = 8$ for single precision, complex numbers). The total time required to FFT the traces and redistribute frequencies for $b/(c\,n\,g)$ traces, (*i.e.*, the number of traces which one compute node processes) is approximately

$$\frac{b}{c\,n\,g}\left\{p_\omega\left[\alpha + \beta\left(\frac{ng}{p_\omega}\right)\right] + \tau\right\},$$

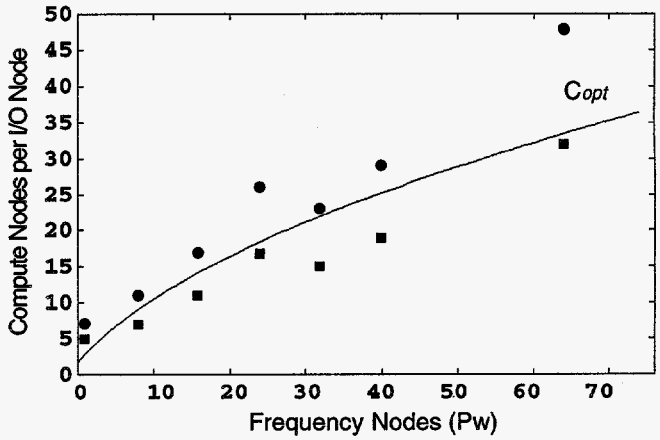



Figure 1: The graph shows $c_{opt}$ as a function of $p_\omega$. Circles correspond to actual runs in which I/O nodes had no idle time; squares correspond to actual runs in which I/O node were idle for part of the run.

To determine the minimum number of compute nodes for each I/O node, $c_{opt}$, the time required to read and distribute a block of data must be equal to or greater than the time required to FFT the time traces and redistribute frequencies. This yields

$$c_{opt} = \frac{-b\,(\Phi + \beta) + \sqrt{\kappa}}{2\alpha}, \tag{10}$$

where

$$\kappa = (\Phi b + \beta b)^2 + 4\alpha\left\{p_\omega\left[\alpha + \beta\left(\frac{ng}{p_\omega}\right)\right] + \tau\right\}.$$

All of the variables in the expression for $c_{opt}$, except $p_\omega$, are either machine constants or defined by the problem size. Figure 1 shows the $c_{opt}$ as a function of $p_\omega$ and points indicating several "real" runs. We see that the model does a good job of predicting whether the run time is dominated by disk reads or by computation and communication.

## TRIDIAGONAL SOLVES

At each depth step the algorithm solves a sequence of tridiagonal systems. It is difficult to parallelize the solution of a single tridiagonal system, but this difficulty is offset because there are many such systems. Salvo takes advantage of this by setting up a pipeline. That

is, in the first stage of the pipeline, processor one starts a tridiagonal solve. In the second stage of the pipeline, processor two continues the first tridiagonal solve, while processor one starts a second tridiagonal solve. This process continues until all processors are busy.

In the implementation of a pipeline, there are two sources of parallel inefficiency. The first is communication between processors. This communication time is dominated by the message latency since very small amounts of data must be transferred. This can be offset by grouping several tridiagonal solves into each stage of the pipeline.

The second source of parallel inefficiency is processor idle time associated with the pipeline is being filled or emptied. This is dominated by the computation time of each pipeline stage. It can be reduced by reducing the computation time, but it is increased by grouping several tridiagonal solves in each stage of the pipeline.

The total parallel overhead can be minimized by choosing how many tridiagonal solves are grouped into each stage of the pipeline. The number of tridiagonal solves to group is based on the following model. The communication time is approximated by

$$T_{comm} = N\left(2\frac{\alpha}{b} + 24\beta\right),$$

where $N$ is the total number of tridiagonal solves, $b$ is the number to be grouped into each stage of the pipeline, $\alpha$ is the communication latency, and $\beta$ is time to communicate one byte. The pipeline idle time is approximated by

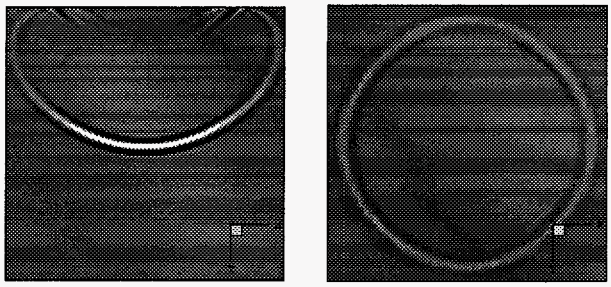
$$T_{pipe} = W\,b\,n\,\gamma + p\,(2\alpha + 24b\beta),$$

where $W$ is the total number of floating point operations required at each grid point, $n$ is the number of points in each stage of the pipeline, $p$ is the number of processors in the pipeline, and $\gamma$ is the computational time required for one floating point operation.

The value of $b$ that minimizes the total overhead, $b_{min}$ is computed by summing $T_{comm}$ and $T_{pipe}$, differentiating with respect to $b$, setting the result equal to zero and solving for $b$. This yields

$$b_{min} = \left(\frac{2\,N\,\alpha}{W\,n\,\gamma + 24\,p\,\beta}\right)^{1/2}.$$

We have found this model to be quite accurate, and all results presented later in this paper use this value of $b_{min}$ to improve performance.



(a) (b)

Figure 2: Impulse response for a filtered migration. An $x$-$z$ section through the center of the migration is shown in (a), and an $x$-$y$ section through the 60° propagation angle (b).

## RESULTS

To validate Salvo, several tests were performed to ensure accurate imaging of reflecting layers. The problems selected for the test cases include a simple impulse response from a hemispherical reflector, the poststack migration of the French Model [French 1974], and the prestack migration of an SEG/EAEG-Overthrust-Model section [Aminzadeh *et al.* 1994].

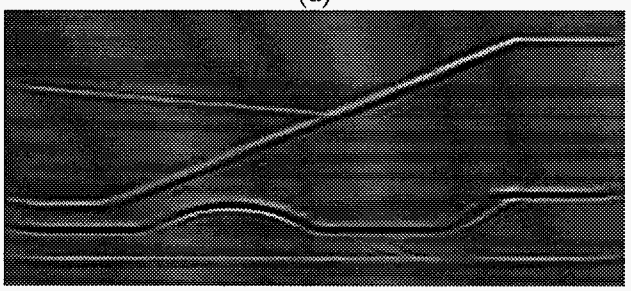
The impulse-response problem is a good initial problem, because of the simple inputs and the simple solution. The test can be described as a source impulse which is initiated at the center of the hemispherical reflector. This impulse propagates into the earth as a hemispherical wave. The reflected impulse coalesces at the center of the hemispherical reflector, and is recorded by a geophone. Thus, the inputs for this test are a source trace with an impulse at some time, a receiver trace with an impulse at some later time, and a constant velocity field.

Figure 2 shows a typical output for this problem. The parameters used for this run are

$$\begin{aligned} n_x &= 101, & \Delta x &= 5 \text{ m}, \\ n_y &= 101, & \Delta y &= 5 \text{ m}, \\ n_z &= 100, & \Delta z &= 5 \text{ m}, \\ n_t &= 128, & \Delta t &= 0.004 \text{ s}, \\ n_\omega &= 63, & \text{v} &= 3000 \text{ m/s}. \end{aligned}$$

In Fig. 2(a), the shape of the hemispherical image in comparison to the reflector is accurately determined up to a propagation angle of 65 degrees. Beyond 65 degrees, the image curls back to the center of the domain.


(a)



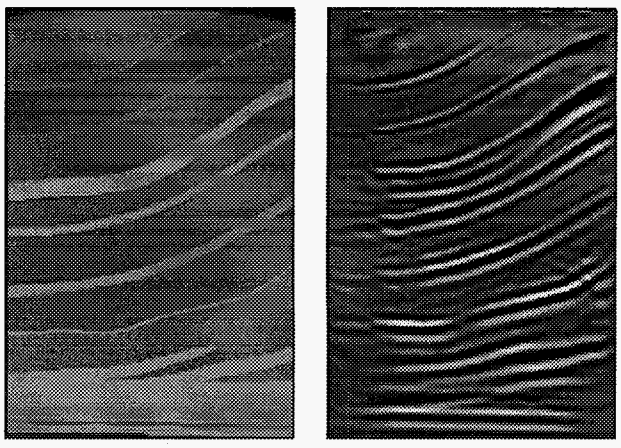
(b)

Figure 3: The French Model acoustic velocity (a) and Salvo solution (b).

This structure is termed the cardioids of the solution [Bunks 1995]. They are caused by the approximations to the square-root operator in Eq. (3), where evanescent energy has been introduced.

In Fig. 2(b), an $x$-$y$ plane through the solution is shown. The depth of the plane was selected so that the 60 degree propagation angle is located on the hemispherical reflector. The cross-section of the hemispherical reflector is nearly circular, which should be the case since this is within the 65 degree approximation limits. So although a slight diamond shape remains, any further refinements in the filter would add little value to the solution.

The French Model [French 1974] is a velocity model with trace data generated from an exploding reflector algorithm. The velocity model has $111 \times 111 \times 250$ grid points with a grid spacing of 100 ft. $\times$ 100 ft. $\times$ 20 ft. Therefore, the total velocity-model volume is 11,000 ft $\times$ 11,000 ft $\times$ 5000 ft. A 2-D section through the 3-D velocity model is shown in Fig. 3(a). There are several constant velocity layers at different dip angles and two dome structures. (Only one dome is shown in the figure.) All the flat dipping reflectors are angled into the page so that the worst case, reflectors along the line $y = \pm x$, is tested.



(a) (b)

Figure 4: SEG/EAEG Overthrust 2-D Section: velocity model (a) and Salvo solution (b).

The trace dataset is generated by an exploding reflector algorithm and requires poststack migration. With a slight modification, the Salvo code can handle poststack data and perform the poststack migration. A calculated solution is shown in Fig. 3(b) using the French velocity model and the poststack traces. Good agreement with the velocity model is seen.

Finally, a small region of the synthetic SEG/EAEG Overthrust Model was used to evaluate the Salvo code. This model has more variations in velocity, both in depth and in the horizontal directions. The velocity model for the entire Overthrust Model has $801 \times 801 \times 187$ grid points with 25 m spacing in each direction. The selected subvolume has $100 \times 100 \times 150$ grid points, and a 2-D slice of this subvolume is shown in Fig. 4(a).

The trace dataset was generated with the original SEG acoustic-wave-propagation code and used as input to the Salvo code. The trace dataset was used in its raw form and did not have deconvolution performed or first arrivals removed. The latter caused noise near the surface. The 2-D section of the 3-D Salvo solution is shown in Fig. 4(b), and again, good agreement with the velocity model is evident.

We are continuing to test and validate Salvo.

## PERFORMANCE

To test the computational performance of Salvo, the sample impulse problem was used. The spatial size of the impulse problem has been adjusted so that each proces-

| $p_x \times p_y \times p_\omega$ | Runtime (sec.) | Efficiency (%) |
|---|---|---|
| Spatial Parallelism | | |
| $1 \times 1 \times 1$ | 84.1 | 100.0 |
| $2 \times 1 \times 1$ | 92.4 | 91.0 |
| $2 \times 2 \times 1$ | 103.2 | 81.5 |
| $3 \times 3 \times 1$ | 108.7 | 77.4 |
| $4 \times 4 \times 1$ | 108.9 | 77.2 |
| $5 \times 5 \times 1$ | 112.2 | 75.0 |
| $6 \times 6 \times 1$ | 114.8 | 73.3 |
| $7 \times 7 \times 1$ | 115.6 | 72.8 |
| $8 \times 8 \times 1$ | 116.2 | 72.4 |
| Frequency Parallelism | | |
| $1 \times 1 \times 1$ | 84.1 | 100.0 |
| $1 \times 1 \times 2$ | 42.21 | 99.6 |
| $1 \times 1 \times 4$ | 21.19 | 99.2 |
| $1 \times 1 \times 8$ | 10.63 | 98.9 |
| $1 \times 1 \times 16$ | 5.35 | 98.2 |
| $1 \times 1 \times 32$ | 2.71 | 97.0 |
| $1 \times 1 \times 64$ | 1.40 | 93.8 |

Table 1: Timings for a sample impulse problem for spatial, frequency, and mixed parallelism. Single processor times are estimated. All other times are measured.

sor has approximately a $101 \times 101$ spatial grid. Sixty-four frequencies have been retained for the solution independent of how many frequency processor were used.

Timings for the sample impulse run are shown in Table 1. From these numbers, we can make a few statements about the parallelism of the migration routine. First, the spatial parallelism is very efficient as soon as the pipeline is fully utilized (after $3 \times 3 \times 1$ processor mesh). However there is a penalty for introducing the pipeline in each direction, which is about 10% for each (*i.e.*, $1 \times 1 \times 1$ at 100% to 91% for $2 \times 1 \times 1$, and to 81% for $2 \times 2 \times 1$). The origins of this "overhead" is still under investigation.

Second, the frequency parallelism is very efficient, staying in the upper 90's for most of the problems. This is expected, since frequency parallelism requires little communication during the solve. The primary communications are a broadcast of velocity data at the beginning of each depth step and a summation to produce an image at the end of each depth step.

## CONCLUSIONS

In this paper, an implementation of a wave-equation-based, finite difference, prestack, depth migration code for MPP computers has been presented. The results of several test runs were presented to show the accuracy of the code. Also, timing results and performance models have been presented to show that the code can be tuned to run efficiently on MPP computers.

## REFERENCES


Aminzadeh, F.; N. Burkhard; L. Nicoletis; F. Rocca; K. Wyatt. 1994. "SEG/EAEG 3-D Modeling Project: 2nd Update." *Leading Edge.* September, 949–952.

Bunks, C. 1995. "Effective Filtering of Artifacts for Implicit Finite-Difference Paraxial Wave Equation Migration." *Geophysical Prospecting,* vol. 43: 203–220.

Claerbout, J. F. 1985. *Imaging the Earth's Interior.* Blackwell Scientific Publications, Boston.

Clayton, R. and B. Engquist. 1980 "Absorbing Boundary Conditions for Wave-Equation Migration," *Geophysics,* vol. 45, 895–904.

Fletcher, C. 1988 *Computational Techniques for Fluid Dynamics Vol. I.* Springer-Verlag, Berlin.

French, W. S. 1974. "Two-Dimensional and Three-Dimensional Migration of Model-Experiment Reflection Profiles." *Geophysics,* vol. 39, 265–277.

Graves, R. and R. Clayton. 1990. "Modeling Acoustic Waves with Paraxial Extrapolators." *Geophysics,* vol. 55, 306–319.

Lee, M. W. and S. Y. Suh. 1985. "Optimization of One-Way Wave Equations." *Geophysics,* vol. 50, 1634–1637.

Li, Z. 1991. "Compensating Finite-Difference Errors in 3-D Migration and Modeling." *Geophysics,* vol. 56, 1650–1660.

Ma, Z. 1981 "Finite-Difference Migration with Higher Order Approximation." In *1981 Joint Mtg. Chinese Geophys. Soc. and Society of Exploration Geophysicists,* Society of Exploration Geophysicists.

Yilmaz, O. 1987. *Seismic Data Processing, Investigations in Geophysics No. 2.* P.O. Box 702740, Tulsa, OK 74170-2740: Society of Exploration Geophysicists.

## DISCLAIMER