A comparison of four closely related depth migration methods

by

Baoniu Han

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

Golden, Colorado

Date _____

Signed: _____
Baoniu Han

Approved: _____
Dr. Kenneth L. Larner
Professor of Geophysics
Thesis Advisor

Golden, Colorado

Date _____

_____
Dr. Terence K. Young
Professor and Head
Department of Geophysics

ABSTRACT

Much research has focused on ray-based migration methods, such as Kirchhoff and Gaussian beam, with special effort required to overcome the multi-pathing problems that arise in complicated media. Despite advances that enhance the ray-based methods, many problems related to complex structures have not been fully solved as yet. Though relatively expensive compared with the ray-based methods, migration methods based on wavefield extrapolation exhibit advantages in handling complex structures. Although ray-based methods are still the dominant imaging tool for 3-D prestack depth migration, with the rapid advancement of computer hardware technology, we can expect migration algorithms based on wavefield extrapolation to play a more important role in the near future.

In this thesis I discuss four such related one-way wavefield-extrapolation algorithms for depth migration of prestack and poststack data: phase-shift-plus-interpolation (PSPI), split-step Fourier (SSF), implicit $\omega - x$ finite-difference (FD), and Fourier finite-difference (FFD). All the algorithms work in the frequency domain, resulting in natural and similar parallelization of the codes. These straightforward implementations of one-way methods yield accurate imaging in complicated geological structures such as the Marmousi model, where ray-based methods require complicated ray-tracing efforts to image the target zone.

For the most part, these algorithms obtained somewhat comparable migration results for the two model data sets tested in this thesis (especially the Marmousi model in Chapter 4). Such a result is not surprising. First, all of them are derived from the same one-way acoustic wave equation. Second, they are closely linked with one another, the SSF and FFD methods form the intermediate steps between the phase-shift and finite-difference methods.

A good choice of migration algorithm is largely dependent on the balance of computation cost and accuracy. The SSF and 65° FD algorithms are relatively low-cost wavefield extrapolation methods, but they also suffer some loss of accuracy. Being based on a small-perturbation assumption, the SSF method cannot deal with strong lateral velocity contrast; the 65° FD algorithm, on the other hand, has accuracy limited to dips less than about 65°. About twice as expensive as the SSF and 65° methods, the FFD and 80° algorithms have the ability to handle both rapid lateral velocity variation and large dips (around 80°). The PSPI and 90° FD algorithms

are accurate for dips up to 90°, but the 10° increase in the dip ability of these two algorithms requires a doubling of the cost relative to that of the FFD and 80° FD methods.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Ken Larner for his advice, and direction during this work. Also I thank members of my M.S. committee, Ilya Tsvankin, Norm Bleistein and Roel Snieder, for their suggestions during the writing of this thesis.

I want to give my deepest gratitude to my parents, for their continuous encouragement and support for my education. Finally I would like to dedicate this thesis to my wife Joy, for her support during these years.

Chapter 1

INTRODUCTION

1.1 Ray tracing versus wavefield extrapolation?

Many methods for migration of seismic data have been developed and put into practice over the years. For purpose here, they can be put into either of two categories — ray-based methods and methods based on wavefield extrapolation, although all of them are derived from some form of the acoustic or elastic wave equation and depend more or less on a high-frequency approximation.

Ray-based migration methods, such as Kirchhoff-summation (Schneider, 1978) and Gaussian beam (Hill, 1990), are currently preferred methods for 3-D prestack depth migration, mainly because of their target-oriented feature, flexibility in handling irregularly sampled 3-D data, generally good imaging results and relative computational efficiency. Considering the high computation cost in processing 3-D data sets, Kirchhoff and Gaussian beam methods are presently the only viable ways to do 3-D prestack depth migration. The defining characteristic of ray-based methods is their dependence on ray tracing to get the traveltimes needed for the migration mapping. In complex geological structures, however, multiple arrivals are a complication that often causes ray tracing to fail to obtain proper traveltimes, hence leading to poor subsurface imaging. Recently, several methods have been developed to help overcome such problems; examples include semi-recursive Kirchhoff migration (Bevc, 1997), maximum-energy traveltime methods (Nichols, 1996), common-angle image-gather migration (Xu *et al.*, 1998), and asymptotic inversion (De Hoop, 1998). These methods, however, only partially solve the multi-pathing problem, and do so at increased algorithm complexity and computing costs, thus losing some of the advantage of ray-based methods.

In contrast, methods based on wavefield extrapolation, such as implicit finite-difference methods (Claerbout, 1985) and the split-step Fourier method (Stoffa *et al.*, 1990), can naturally handle the multi-pathing problem, leading to more accurate subsurface imaging. Therefore, although relatively expensive compared with ray-based methods, wavefield-extrapolation migration methods have often been preferred for 2-D and 3-D

poststack migration, where cost is a less serious issue than for prestack migration.

Compared with ray-based algorithms, another major advantage for wavefield extrapolation methods is their simplicity. Without the need for complicated traveltime computation, solving the partial differential equations (PDE) is nothing more than a combination of phase-shift and finite-difference operators. Such simplicity also benefits implementation of these algorithms; usually it takes only days to get a new algorithm into practice, given a suitable starting platform, while the implementation of an adequate ray-tracing code can take months.

1.2 Some clarification

As I am emphasizing the accuracy of wavefield-extrapolation migration algorithms over that of ray-based algorithms, I shall make some clarification about how the advantages are made possible.

First, the trade-off for selecting a migration algorithm is governed by the balance between cost and accuracy. Although wavefield-extrapolation algorithms are superior in accuracy, this improvement is largely due to the more expensive numerical methods used to solve the wave equation. Typically, wavefield-extrapolation methods are an order of magnitude more costly than the ray-based methods in 2-D prestack migration. Years ago, one could not afford to do prestack wavefield extrapolation migration; now it is feasible because of the rapid advance of computer hardware/software technology in recent years. The price drop for fast RAMs, CPUs and hard-drives makes the computing power more accessible than in the past; nevertheless, the relative cost of various migration algorithms remains an important issue. In short, we should not take the accuracy of wavefield-extrapolation algorithms for granted.

Another issue is how different are the wavefield extrapolation and ray-based migration algorithms. Before answering this question, let us first look at the similarities between them. For the majority of migration algorithms, only one-way propagation of wavefields is carried out inside the algorithms, i.e., propagation that ignores transmission and reflection across the layer boundaries, and treats the up-going and down-going waves separately (usually, with the downgoing wavefield ignored). By doing so, we linearize the seismic inversion problem, rendering the process applicable.

In ray-based algorithms, the Green's function for one-way wave equation is computed by ray-tracing. As most ray-tracing codes yield only a single primary arrival from source to receiver, usually the first arrival (which often differs from the maximum-energy arrival), the Green's function obtained by ray-tracing is incomplete. Thus,

unless extra (costly) steps are taken, ray-based methods are usually single-arrival methods.

In wavefield extrapolation algorithms, the Green's function is computed by downward continuation of wavefields recorded at the surface to all depths. Not only the primary arrivals, but the total (one-way) wavefield at a given depth is computed; thus, these approaches yield a more complete Green's function than do ray-based ones.

For media with simple structure, the Green's function describing only primary arrivals is adequate for good imaging; for media with complex structure, however, the complete Green's function for one-way wave-equation may be essential.

1.3 Comparison of four one-way algorithms

Among the wavefield-extrapolation migration methods, some, such as reverse-time migration (Baysal *et al.*, 1983), are derived from the two-way wave equation, while most others depend on approximate solutions to the one-way acoustic wave equation. Two-way solutions, which are accurate but also expensive, are used mainly for generating synthetic data. As two-way methods generate unwanted internally scattered waves for migration, special treatment is needed to suppress these annoying artifacts, thus making the results of two-way migration resemble more closely those of one-way migration. In practice, most migration algorithms are based on the less costly one-way solution. With certain optimization efforts (Jenner *et al.*, 1997), the one-way solution can reach a high accuracy.

Here, I compared four related, one-way depth-migration methods: phase-shift-plus-interpolation (Gazdag & Sguazzero, 1984), implicit ω-x finite-difference (Claerbout, 1985), Fourier finite-difference (Ristow & Rühl, 1994), and split-step Fourier (Stoffa *et al.*, 1990). I selected these four algorithms from among other one-way algorithms because of their close relationship to one another (Rühl & Ristow, 1995). It is of interest to review their inter-relationship as well as their relative migration performance.

Another reason I single out those four algorithms is that they include and generalize two important members of one-way wavefield-extrapolation techniques, i.e., phase-shift and finite-difference. Thus, understanding the strengths and weaknesses of the above algorithms will help us deduce the expected performance of any newly developed algorithm, which largely depends on extension of the existing methods or some combination of them.

The four algorithms I compare were developed mainly in the 80s and 90s, with the exception of the implicit ω-x finite-difference method, which was devised in the early 70s and greatly refined throughout 80s. Over these years, many comparisons of these algorithms have been made. After viewing the results of these comparison, however, I have identified the following shortcomings in the comparisons.

1. In some papers, such as Gazdag & Sguazzero (1984) and Popovici (1992), comparisons were done on relatively simple models so that migration performance, such as the maximum dip for accurate imaging, could be easily measured. No examples for complicated models have been shown in these papers.

2. Some comparisons, such as Zhu & Lines (1998), were done for the complex Marmousi model but against Kirchhoff type algorithms to show their strength over the ray-based methods; no comparisons with other wavefield-extrapolation algorithms were done.

3. Theoretical comparisons were done in papers by Ristow & Rühl (1994) and Le Rousseau & De Hoop (1998) showing that results favor one algorithm over another. These theoretical analyses, however, do not guarantee that one algorithm is better than another for complex subsurface structure because the testing environment used may not be representative of that for field data.

4. When migration results on a common data set have been published by different authors, it has not been easy to do comparisons because variations in picture quality, display method, and scale influence our relative impressions of the performance of these algorithms. A uniform comparison is needed.

5. The four algorithms that I compared here have never been tested together for data from the same complex models. It is desirable to test them in a common environment to assess similarities and differences.

Based on these observations, emphasis of this thesis is on comparing algorithm performance on realistic and complicated subsurface model data. I have tested the above four algorithms on two known models, — the Marmousi model (Versteeg & Grau, 1990) and the SEG-EAGE salt model (Huang & Fehler, 1997). Both models are two dimensional (2-D) with complicated geological structure, and have posed difficulties for accurate imaging with ray-based algorithms. With access to prestack seismic data for the Marmousi model, I have performed common-shot-gather migrations on the data with both the exact velocity model and smoothed ones. On the other hand, I applied only poststack migration to the zero-offset data for the SEG-EAGE model.

To make the comparison fair, all the codes used in the experiments were implemented by the author, and all were developed on a common coding platform. At the time of writing this thesis, the migration codes have been released to the public domain for two years. With extensive tests by people from industry and academic institutes, most errors and bugs in the codes have been fixed, thus reducing the likelihood that such errors might influence the imaging quality. Also, all the codes have been parallelized to run on multi-processors machines and computer networks, speeding up the experiments greatly.

This thesis is organized as follows. Chapter 2 describes the four algorithms and their inter-relationships. Chapter 3 compares zero-offset migration results on SEG-EAGE salt model data. Chapter 4 compares prestack migration results on data from the Marmousi model. Following a discussion on the migration results in Chapter 3 and 4, in Chapter 5 I draw conclusions on strengths and weaknesses of these methods. Also, I discuss finished and on-going extensions and developments for these algorithms. Appendix A describes how I have parallelized the algorithms and shows their improvement in performance over that of sequential codes.

1.4 Contribution of this thesis

The contribution of this thesis has the following aspects:

1. As emphasized in the beginning of this chapter, the choice of migration algorithm is largely dependent on the cost of the algorithm and its accuracy obtained under such a cost. Through the efforts to make fair comparisons of various one-way wavefield-extrapolation migration algorithms in this thesis, I have attempted to show readers how those algorithms work for two complicated models and thus help them make decisions on the choice of migration algorithm best suited for a specific imaging target.

   The quality of any seismic migration is governed not only by the accuracy of the algorithm, but also by the choice of parameters for migration. By tuning the migration parameters in examples shown in Chapters 3 and 4, I show alterations in images that suggest that differences among the migration results of various algorithms can often be minimized by suitable choice of parameters. Thus, the preference for a migration method should also consider the ease of handling and adjusting migration parameters.

2. By developing the migration codes used in these experiments, we obtain valuable tools that might be useful for other research tasks. One example is static-

correction research done by Tjan (1995). For that research, accurate migration and inverse-migration (demigration) tools are needed. With little modification, the existing migration codes can be turned into demigration and modeling codes useful for that research. (The major difference between demigration and modeling is that demigration uses the migrated section as the reflectivity function, while modeling usually use some other function, such as the derivative of the specified velocity model, as the reflectivity function.) For the study here, I have implemented exploding-reflector modeling codes by modifying the poststack migration codes.

Through implementation of these algorithms, I have developed a common coding platform for efficient development of new migration algorithms. One example is the development of an anisotropic prestack migration (Han *et al.*, 1998) based on the finite-difference algorithm; extension to anisotropic media was finished within a week because only the core wavefield-propagation part had to be modified to create the new algorithm.

3. Releasing the code in the public domain [free code through Seismic Unix (SU)] greatly benefits both academia and small-size companies, where commercial software is too expensive to be affordable. Also, apparently, no single commercial data-processing package includes all the algorithms compared here; thus, even large companies can benefit from these codes as well. With the parallelized migration codes, large-scale migration can be run on both costly multi-processor computers and inexpensive, heterogeneous PC clusters, providing efficient and cost-effective solutions for data processing.

Before looking at the migration results, I first review these four algorithms in Chapter 2, giving necessary background knowledge about the various migration methods.

Chapter 2

A COMMON BASIS

Let us start with the 2-D acoustic wave equation. Given a homogeneous velocity structure, in the frequency-wavenumber domain we have

$$k_x^2 P - \frac{\partial^2 P}{\partial z^2} = \frac{\omega^2}{v^2} P, \tag{2.0.1}$$

where P is the pressure wavefield, z is the depth, ω is frequency and v is velocity. Also k_x is the wavenumber in the lateral (i.e., x-) direction. Taking the spatial Fourier transform in the z-direction, gives the dispersion relationship

$$k_x^2 + k_z^2 = \frac{\omega^2}{v^2}, \tag{2.0.2}$$

or

$$k_z = \pm\sqrt{\frac{\omega^2}{v^2} - k_x^2}, \tag{2.0.3}$$

and its corresponding one-way wave equation,

$$\frac{\partial P}{\partial z} = \pm i k_z P. \tag{2.0.4}$$

Here the $\pm$ signs are for the downward and upward wavefields, respectively. Analytic solutions of this one-way wave equation are

$$P(z + dz, k_x, \omega) = P(z, k_x, \omega) \exp^{\pm i k_z dz}. \tag{2.0.5}$$

Thus, wavefield extrapolation involves just a simple phase shift in the frequency-wavenumber domain. To obtain the migrated image, we need to apply the imaging condition. For poststack migration, this involves evaluating the wavefield when $t = 0$,

which is just a summation of all the frequency components for position (k_x, z). Then the final image is obtained by inverse Fourier transformation from the wavenumber domain to the space domain. That is,

$$Image(x,z) = (FFT)^{-1}[\sum_{\omega_i} P(x,k_x,\omega_i)]. \tag{2.0.6}$$

This formula is the essence of the phase-shift migration method often called Gazdag migration (Gazdag, 1978). The phase-shift migration method, however, requires laterally homogeneous [i.e., $v(z)$] media, making it essentially a time-migration method. Advantages of this method are that it is stable, with no special requirement for the grid spacing, and it is accurate, capable of migrating reflectors with dip up to and beyond 90^0. Given these advantages, people have attempted to extend it to depth migration, that is, to deal with lateral velocity variation in a $v(x,z)$ medium.

2.1 Phase-shift-plus-interpolation migration (PSPI)

The pure phase-shift migration method cannot accommodate velocity fields that have lateral variation because it requires computation in the spatial wavenumber domain. Gazdag and Sguazzero (1984) offered a method to treat lateral variation in velocity that uses the central features of the phase-shift method. Instead of only one velocity at each depth step, they made use of several reference velocities to account for the lateral velocity variation. At each depth step, the wavefield is propagated in the (ω, k_x) domain with each of these reference velocities, yielding multiple reference wavefields. Then an inverse Fourier transform brings these reference wavefields back to the (ω, x) domain. The true wavefield is obtained by linearly interpolating the reference wavefields based on the relationship of the local velocity to the reference velocities.

To maintain high accuracy for small dip, Gazdag and Sguazzero (1984) introduced a laterally varying time-shift in the (ω, x) domain as a preprocessor for the input data. Specifically, they defined a modified field $P^*(z)$ in the space domain

$$P^*(z) = P(z)\exp[\pm i\frac{\omega}{v(x)}dz], \tag{2.1.7}$$

Then they transformed $P^*(z)$ into the wavenumber domain with FFT. In the wavenumber domain, the influence of the previous time-shift is compensated by the $\frac{\omega}{v_r}dz$ term

in the following formula,

$$P(z+dz) = \hat{P}^*(z) \exp\left[\pm i(k_z \mp \frac{\omega}{v_r})dz\right]. \tag{2.1.8}$$

Here, v_r is the reference velocity, k_z is the vertical wavenumber, defined as $\sqrt{(\frac{\omega}{v_r})^2 - {k_x}^2}$, and the $\hat{P}^*$ is the Fourier transform of P^* from (ω, x) to (ω, k_x). Such a time-shift term is important in the implementation of the PSPI method. Later, when we examine all the other three algorithms, we find that the spatially varying time-shift is nothing but the *thin-lens term*, which appears in all these algorithms. With such a term in PSPI, the split-step Fourier method (SSF), which has the same term, can be considered as a special case of PSPI when only one reference velocity is used in PSPI.

Since it works on the complex wavefield, we have two choices for ways to do the linear interpolation. One is to interpolate the amplitude and phase angle individually, and the other is to interpolate the real and imaginary parts individually. The periodicity of the phase angle (2π ambiguity) suggests that we choose the latter because it has no phase unwrapping problem. Also, the experiments with these two interpolation schemes show that interpolating the real and imaginary parts individually introduces less numerical noise and thus produces cleaner image.

The accuracy of PSPI is directly related to the number of reference velocities used at each depth step, that number depending on the amount of lateral velocity variation at that depth. In an effort to do automatic reference-velocity picking, Bagaini et al. (1995) proposed an adaptive choice for the reference velocities at any depth step determined by the statistical distribution of the velocity within that depth step. His method determines the reference velocities as those that are most representative of the velocity distribution at that depth level. In this way, the total computation cost in PSPI is controlled. Figure 2.1 shows the number of reference velocities I used for migrating the Marmousi data (see Chapter 4) as a function of depth. Interested readers may refer to Bagaini et al. (1995) for a full description of the procedure for the velocity selection.

Another issue for the PSPI method is the wrap-around introduced by the spatial Fourier transformation from the (ω, x) domain to the (ω, k_x) domain. Wrap-around is a phenomenon in which the periodically extended wavefield on either side of the computation domain propagates in from the sides, interfering with the actual wavefield. For PSPI, although I don't have an explanation for it, the experimental results show that the wrap-around is not serious if the number of reference velocities is not large. In one poststack migration experiment not shown here (Han, 1998a), when the

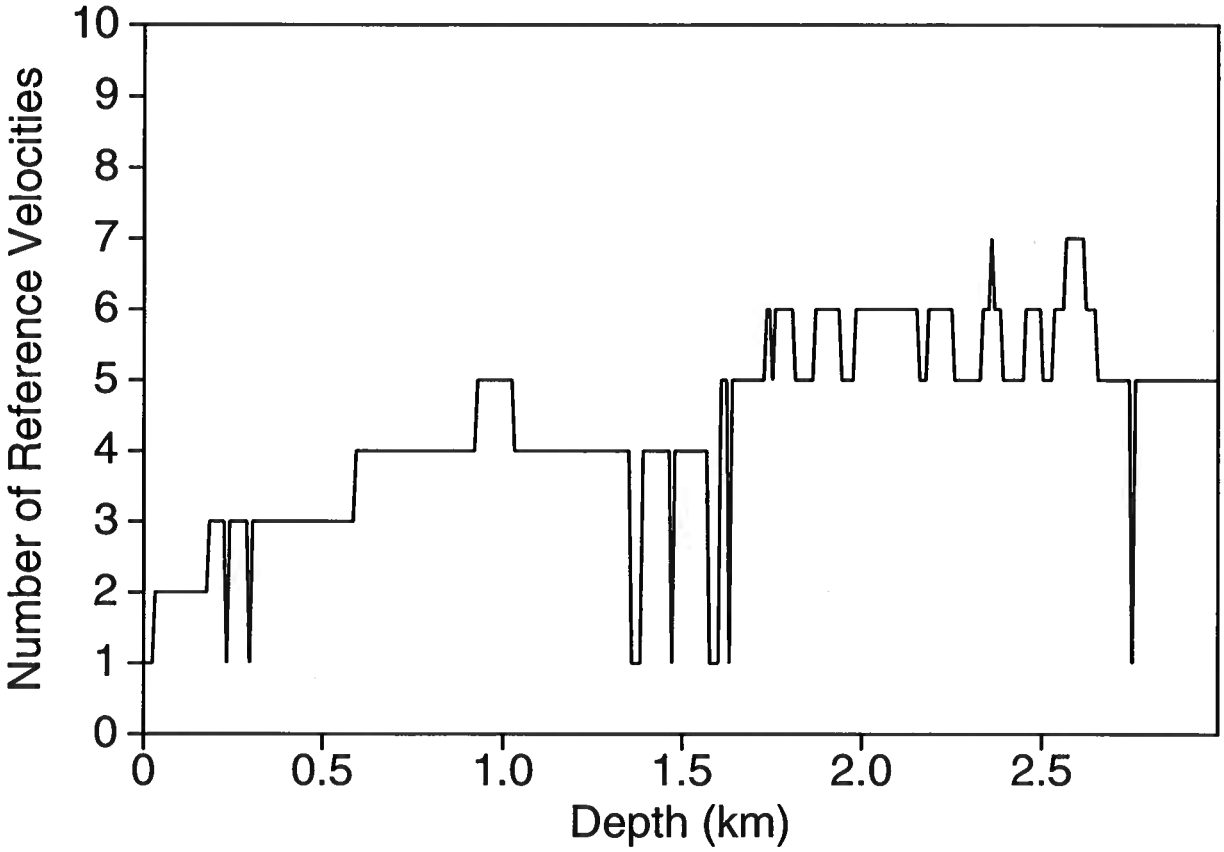

FIG. 2.1. The number of reference velocities used in migrating the Marmousi data with the PSPI algorithm. These numbers are determined by the statistical method of Bagaini et al. (1995).

number of reference velocities is less than five, the wrap-around is negligible, but as a result, the steep reflectors are mistreated because a finer increment in reference velocities is needed. When we introduce too many reference velocities, however, image quality in the deep section is degraded. To attack this problem, following Cerjan et al. (1985) I have implemented a damping region on each side of the depth section. For example, on the left side, I set the damping region empirically to 20 traces as follows: for $1 \leq i_x \leq 20$ (i_x is the output grid-point index in x-direction), I multiply the wavefield by a Gaussian-shaped spatial function $g = \exp -[a(20 - i)^2]$ with a set to 0.015. The damping region also acts to reduce unwanted reflections from the side-boundary. Since split-step Fourier (SSF) and Fourier finite-difference (FFD) methods also work in the dual domains, (ω, x) and (ω, k_x), I apply this damping technique to them as well.

Repeating equation (2.0.3), the dispersion relationship is

$$k_z = \pm\sqrt{\frac{\omega^2}{v^2} - k_x^2}.$$

Note that for $k_x^2 \geq \frac{\omega^2}{v^2}$, k_z is an imaginary number, so the waves are inhomogeneous. In a typical phase-shift algorithm, such inhomogeneous waves are ignored in the downward continuation by putting the corresponding amplitude to zero. Because such a treatment of the inhomogeneous wavefield is equivalent to applying a boxcar filter on the data, it generates unwanted numerical noise. In my experiments, the inhomogeneous wavefield is attenuated by a relatively smoothing wavenumber filter. Handling the wavefield this way actually benefits the imaging of steep dipping reflectors. These steep features corresponds to the large wavenumbers in the wavenumber domain. Abrupt truncation of the high-wavenumber portion of the wavefield introduces numerical noise in the form of artifacts with large moveout. The wavenumber filter is designed as follows. For $|k_x| \geq |w/v|$, I substitute

$$k_z = \pm i\sqrt{k_x^2 - \frac{\omega^2}{v^2}}$$

into equation (2.0.5), so that the inhomogeneous-wave contribution to the analytic solution of downward continuation of the wavefield becomes

$$P(z + dz, k_x, \omega) \;=\; P(z, k_x, \omega) \exp^{\pm i \cdot i\sqrt{k_x^2 - \frac{\omega^2}{v^2}}dz},$$

$$= P(z, k_x, \omega) \exp^{\mp\sqrt{k_x^2 - \frac{\omega^2}{v^2}}dz}. \tag{2.1.9}$$

Strictly, physical reality requires that we treat the downgoing and upgoing waves in different ways — damping downgoing waves and increasing the amplitude of upgoing (backward-propagating) waves. For numerical stability in the presence of additive and computational noise, however, we must avoid exponentially boosting the amplitude during the wavefield extrapolation. Since the primary idea here is to cause the inhomogeneous waves to decay smoothly, I select filters with negative exponent. Therefore, in the implementation, I choose the negative sign in the exponent of equation (2.1.9).

2.2 Implicit (ω-x) finite-difference migration (FD)

Since Claerbout (1985) developed the FD method in the early 1970s, it has been extensively used in seismic imaging. Unlike the phase-shift migration method, the FD method works in the frequency-space domain, thus making it suitable to handle lateral velocity variation. In deriving the FD method, one starts again from the one-way acoustic wave equation [equation (2.0.4)] in the frequency-wavenumber domain and approximates the square-root appearing in k_z with continued fractions. The idea of continued fractions is to use the recurrence relationship to approximate a function. For the square-root function

$$R = \sqrt{1 - S^2}, \tag{2.2.10}$$

where S is some number, the following recurrence relation holds:

$$R_{n+1} = 1 - \frac{S^2}{1 + R_n}. \tag{2.2.11}$$

Note, as $n \to \infty$, R_n converges to R. For the second-order approximation, we obtain

$$R_2 = 1 - \frac{S^2}{2 - \frac{1}{2}S^2}. \tag{2.2.12}$$

Using this equation to approximate the square-root function (2.0.3) in k_z, and defining $S = \frac{v}{\omega}k_x$, we get

$$k_z = \frac{\omega}{v}(1 - \frac{0.5S^2}{1 - 0.25S^2}). \tag{2.2.13}$$

Now associating the partial differential operator $-i\partial_x$ with k_x, and $-i\partial_z$ with k_z, we get the second-order approximation of one-way wave equation (2.0.4):

$$\frac{\partial P}{\partial z} = i\frac{\omega}{v}(1 + \frac{0.5S_x^2}{1 + 0.25S_x^2})P, \tag{2.2.14}$$

with

$$S_x \equiv \frac{v}{\omega}\partial_x. \tag{2.2.15}$$

Now, as equation (2.2.14) and (2.2.15) are in the frequency-space (ω-x) domain, the v can freely take the form $v(x, z)$; i.e., velocity can vary from one grid cell of the model to another. Note that the factor multiplying P on the right side of equation (2.2.14) contains two terms. The first factor $i\frac{\omega}{v}$ called the *thin-lens term*, is responsible for the lateral variation in velocity; where the velocity is constant, applying this term is nothing but a constant time shift of the wavefield. The second term, called the *diffraction term*, serves to collapse diffraction hyperbolas in the seismic data.

To solve the resulting second-order approximation of the one-way wave equation, the splitting method is usually used. That is, we first apply the thin-lens term to the input wavefield, then the final wavefield is obtained by applying the diffraction term to the intermediate wavefield. The thin-lens term is solved analytically, and the diffraction term is solved by a method of finite differences. Based on the implicit finite-difference approximation, the diffraction term is formulated into a tri-diagonal linear system that can be solved efficiently by Crank-Nicolson method. For the full recipe for solving the tri-diagonal linear system, refer to (Claerbout, 1985).

Because we used only the second-order continued-fractions approximation, the implicit FD method is accurate for dips up to about only 45°. Also the structure of continued-fractions approximation further makes it hard to implement high-order approximations efficiently. On the other hand, many situations require that FD method handle dips beyond 45°. To resolve the dip limitation of the implicit finite-difference algorithm, Lee & Suh (1985) used a least-squares optimization technique. First, they use a cascaded series of equations to match the dispersion relationship,

$$k_{z_{opt}} = \frac{\omega}{v}(1 - \sum_i \frac{a_i S^2}{1 - b_i S^2}). \tag{2.2.16}$$

Here $k_{z_{opt}}$ is the optimized k_z. Then they use least-squares optimization to minimize the difference between $k_{z_{opt}}$ and k_z by adjusting the variable parameters a_i and b_i over the range 0° to 90°. Table 1 (Lee & Suh, 1985), lists values of the maximum dip that the implicit finite-difference method can handle after optimization. With the help of optimized equations, the maximum dip for implicit finite-difference can be increased up to 90°. Also for no increase in computation cost, the optimization extends the dip accuracy from 45° to 65°. Applying the implicit FD for dip beyond the 80°, however, yields diminishing returns in terms of improved accuracy versus increased cost.

Dip Accuracy	a_i	b_i
45°	0.5	0.25
65°	0.4782	0.3763
80°	0.0042	0.8739
	0.4572	0.2226
87°	0.0042	0.9729
	0.0813	0.7441
	0.4142	0.1508
90°	0.0005	0.9940
	0.0148	0.9194
	0.1175	0.6145
	0.3670	0.1057

Table.1 Optimized parameters for implicit finite-difference algorithm (Lee & Suh, 1985).

2.3 Split-step Fourier migration (SSF)

The split-step Fourier method, used since the 1970s, has been applied to fields such as underwater acoustics and light propagation in optical fibers (Lee *et al.*, 1991). Stoffa (1990) reintroduced this method into seismic imaging. In more recent years, this method has been implemented in both 2-D prestack migration (Roberts *et al.*, 1997) and 3-D poststack migration (Tanis & Stoffa, 1997).

Like PSPI, SSF also involves a wavefield extrapolation in the frequency-wavenumber

domain and a local velocity correction with the thin-lens term in the frequency-space domain, but its sequence of operations is just opposite to that of PSPI. For PSPI, the thin-lens correction is done first, prior to the wavefield extrapolation in wavenumber domain. Instead of using multi-reference velocities to propagate the wavefield, SSF uses only one reference velocity, which can be an average velocity, the minimum velocity, the root-mean-square velocity or an inverse of the average slowness at each depth step . The choice of the reference velocity among these alternatives is not critical for the migration results.

SSF is based on perturbation theory, according to which one can split the laterally varying velocity field into a constant term and a small perturbation term,

$$v(x,z) = v_0(z) + \delta v(x,z), \tag{2.3.17}$$

First we use $v_0(z)$ to propagate wavefields in the (ω,k_x) domain,

$$\hat{P}(z+dz,k_x,\omega) = P(z,k_x,\omega)\exp\left[\pm i\sqrt{(\frac{\omega}{v_0(z)})^2 - k_x^2}dz\right]. \tag{2.3.18}$$

Then $\hat{P}(z+dz,k_x,\omega)$ is transformed back to the space domain, and a thin-lens term is applied to it to account for the lateral velocity variation,

$$P(z+dz,x,\omega) = \hat{P}(z+dz,x,\omega)\exp\left[\pm i(\frac{\omega}{v(x,z)} - \frac{\omega}{v_0(z)})dz\right]. \tag{2.3.19}$$

In the above two equations, the $\hat{P}$ is the intermediate wavefield.

Lee et al. (1991) further showed that the accuracy of SSF can be improved by making the propagator symmetric, that is, splitting the thin-lens term into two identical parts,

$$\pm i(\frac{\omega}{v(x,z)} - \frac{\omega}{v_0(z)})dz = 2 * \pm i(\frac{1}{2})(\frac{\omega}{v(x,z)} - \frac{\omega}{v_0(z)})dz, \tag{2.3.20}$$

and applying them in the space domain, before and after carrying out the wavefield extrapolation in the wavenumber domain.

For strong lateral variation of the velocity field, however, the perturbation theory fails, and more than one reference velocity has to be used for SSF. Logic for use of multi-reference velocities (MRVL) was introduced into SSF by Kessinger (1992), ne-

cessitating an interpolation for the overlapping region between two reference velocities (Wu & Jin, 1997). With multi-reference velocities, however, the cost of SSF increases to that of PSPI. In my implementation, I chose not to use MRVL in SSF because there is no need to repeat the job of PSPI. Also, in a later example of prestack migration, we shall see that strong lateral velocity variation is not a serious problem for SSF because we can choose a difference single reference velocity for each shot gather at no greater computational effort than that for a single velocity across the entire line. This advantage, however, is not available for poststack migration.

2.4 Fourier finite-difference migration (FFD)

Ristow and Rühl (1994) introduced the Fourier finite-difference method as an extension of the split-step Fourier method. In addition to the SSF operator, an adaptive FD operator is introduced into the wave-propagator. Below, without derivation, is the expression for wave propagator in FFD.

$$\sqrt{\frac{\omega^2}{v(x,z)^2}+\frac{\partial^2}{\partial x^2}} = I + II + III, \tag{2.4.21}$$

$$I = \sqrt{\frac{\omega^2}{v_r^2}+\frac{\partial^2}{\partial x^2}},$$

$$II = \frac{\omega}{v(x,z)} - \frac{\omega}{v_r},$$

$$III = \frac{\omega}{v(x,z)}\left(1-\frac{v_r}{v(x,z)}\right)\frac{S_x^2}{a+bS_x^2}.$$

Here, as before, $S_x^2 = v(x,z)^2/\omega^2 \cdot \partial^2/\partial x^2$, and v_r is a reference velocity. For stability, v_r should be chosen to be the minimum velocity along the layer $(z,\ z+dz)$. Following Ristow and Rühl, a and b are adjustable coefficients, with a set to 2 and $b = \frac{1}{2}[(v_r/v)^2 + v_r/v + 1]$.

As an extension of the SSF method, the FFD algorithm retains the character of the SSF method — the combination $I + II$ is the exact operator shown in the SSF algorithm. The III factor is a higher-order term that improves the accuracy of the approximated dispersion relationship in the FFD algorithm. With a form close to the diffraction term in the FD algorithm, the III factor can also be formulated into a tri-diagonal linear system and solved by the Crank-Nicolson method (Claerbout,

1985). The main difference between this operator and the traditional diffraction term is the choice of the b coefficient, which is adapted to the velocity variation in FFD method.

For end-case situations, FFD has different forms. For examples, for a $v(z)$ medium, since $v(x,z) = v_r(z)$, the II and III terms drop, so the FFD method is the same as the pure phase-shift method. When the lateral variation in velocity is strong, FFD has action somewhere between that of SSF and FD. In fact, if we choose $v_r \ll v(x,z)$, then

$$
\begin{aligned}
I &= \sqrt{\frac{\omega^2}{v_r^2} + \frac{\partial^2}{\partial x^2}} \approx \frac{\omega}{v_r}, \\
I + II &\approx \frac{\omega}{v(x,z)}, \\
III &\approx \frac{\omega}{v(x,z)} \frac{S_x^2}{a + bS_x^2}.
\end{aligned}
\tag{2.4.22}
$$

That is, FFD simply reverts back to a pure FD method. In general, FFD is a hybrid of SSF and FD, combining features of each.

2.5 Relationships among the various methods

When Stoffa (1990) introduced the SSF method into the seismic literature, he separated the velocity field into a constant background term and a small perturbation term. Using the perturbation term as a secondary source in the wave equation, he derived the SSF method from the phase-shift method.

Rühl and Ristow (1995) further showed that SSF and FFD form a link between the phase-shift method and the finite-difference method. Similar to Stoffa's derivation, they also use the perturbation term as a secondary source in the wave equation. They then derived the FD method from the phase-shift method in the presence of strong lateral velocity variation. The SSF and FFD methods are just intermediate steps of that derivation; specifically, the SSF method is just zero-order approximation of the FFD method.

It is not surprising that the above four algorithms have such a close relationship. As approximate solutions to the one-way acoustic wave equation, all of them originate

from the same dispersion relationship; only the approximation techniques — either continued fractions or Taylor expansion or combination of both — differ.

For depth imaging, all four algorithms have similar terms that account for lateral velocity variation. SSF, FD and FFD, have an explicit thin-lens term that makes them depth-migration algorithms; in PSPI, the interpolation of wavefields in the frequency-space domain accomplishes the action of the thin-lens term.

While we have emphasized the similarities among them, we cannot ignore the minor differences that distinguish them from one another. First, they work in different domains. While the FD method works only in frequency-space domain, the other three algorithms work in dual domains (frequency-wavenumber and frequency-space); the Fourier is used to transform the wavefield between these two domains.

In a typical dual-domain algorithm that shuffles between the space and wavenumber domains, two operations are performed. One is a global operation, i.e., the phase shift in wavenumber domain with a constant reference velocity; the other is the local phase shift that corrects the difference between the local velocity and reference velocity in the space domain, usually with a small-perturbation assumption. The PSPI and SSF are examples of this case.

One may not think that PSPI also has the small-perturbation assumption in it, but notice that PSPI requires that multiple reference velocities represent the velocity distribution in the model as close as possible. This requires that PSPI needs the difference between reference velocity and local velocity to be small, which a variant of the small-perturbation requirement.

If we see the FFD method as the SSF plus a FD operator, then we must acknowledge that the FD operator plays an important role in eliminating the small-perturbation requirement. That is the reason why the FD operator in FFD is called an adaptive operator: when the small-perturbation assumption holds, that FD operator is not important in the overall migration process, but once the small-perturbation assumption breaks, the FD operator becomes the part that primarily influences the final migration image. We will see this in comparisons between the migration results of SSF and FFD in Chapter 3, where the velocity ratio around the salt body can reach 1:2.

Contrary to the dual-domain method, the FD algorithm works only in single domain, the frequency-space domain. Derived in the wavenumber domain, then transformed back to space domain, the FD algorithm approximates the dispersion relationship with its localized FD operator. Being a purely local method, the FD method will not put any limitation on the strength of lateral velocity variation, making it a excellent

candidate for depth migration. On the other hand, as both differential operators and square-root operator are approximated, errors are introduced from these two sources. When high-order equations, such as the 90° equation, are used in the FD method, only the error from approximation to the square-root operator is reduced. Additional care needs to be taken to reduce the error in the finite-difference approximation to the differential operator. The second-order differential operator is approximated as

$$\frac{\partial^2}{\partial x^2} \approx \frac{\frac{\delta^2}{\delta x^2}}{1+\gamma(dx)^2\frac{\delta^2}{\delta x^2}}, \tag{2.5.23}$$

and the $\frac{\delta}{\delta x}$ is the differencing operator.

With this formula, the relative accuracy of difference operator is increased because high-order terms are included in the differencing operation. The coefficient γ is an adjustable parameter that, in principle, would be obtained by maximizing the accuracy of (2.5.23) over some specified range of k_x. It should be set to 1/12 based on the Taylor series expansion. However, according to Claerbout (1985), $\gamma = 1/6$ gives a better approximation to the differential operator over a wider range of k_x and is a value in common use. Therefore, γ is called "1/6 trick" parameter (Claerbout, 1985).

A common issue for all the algorithms is how to suppress inhomogeneous waves. As discussed for the PSPI method, for dual-domain methods (PSPI, SSF, FFD) we can use a smooth wavenumber filter to remove the inhomogeneous waves in the wavenumber domain. The design of such a filter is related to the dispersion relationship. For the FD algorithm, we can also use the same wavenumber filter in the wavenumber domain but at additional computation cost. To control the computing cost for the FD methods, we use a filter in the space domain. Claerbout (1985) suggested use of a dip filter (to be exact, it is still a wavenumber filter) to suppress the wavefield at higher wavenumbers. Without explanation, he suggested that use of a complex frequency with a small imaginary component for R_0 [equation (2.2.11)] in the continued-fraction approximation to the dispersion relationship in the implicit FD method could accomplish the desired dip filtering. By making frequency complex in R_0, the FD algorithm has an implicitly built-in dip filter at the computation cost of the original FD algorithm.

To show the action of this filter in suppressing inhomogeneous waves, substitute the complex frequency into the derivation of the continued-fraction approximation. Then the diffraction term in the 45° FD algorithm [equation (2.2.16)] becomes,

$$k_z \frac{v}{\omega} = \frac{a(\frac{v}{\omega})^2 k_x^2}{1 - b(\frac{v}{\omega})^2 k_x^2 \frac{1}{1+i\frac{\epsilon}{2\omega}}}, \tag{2.5.24}$$

where $a = 0.5$, $b = 0.25$, and $|\epsilon/\omega| \ll 1$.

Defining the normalized wavenumbers as $K_1 \equiv k_x \frac{v}{\omega}$, $K_3 \equiv k_z \frac{v}{\omega}$, and $\epsilon^0 \equiv \frac{\epsilon}{\omega}$, equation (2.5.24) becomes

$$K_3 = \frac{aK_1^2}{1 - bK_1^2 \frac{1}{1+i\ 0.5\epsilon^0}}. \tag{2.5.25}$$

Simplifying the above equation and omitting all the ϵ^0 terms of second order and above, leads to the imaginary component of K_3

$$\frac{-i\ 0.5ab\epsilon^0\ K_1^4}{(1 - bK_1^2)^2}. \tag{2.5.26}$$

In terms of the normalized wavenumber K_3, the downward-continuation operator in equation 2.0.5 is just $\exp^{-iK_3}$. With $a = 0.5$, $b = 0.25$, and inserting (2.5.26) for the imaginary part of K_3, we obtain an amplitude filtering term in the 45° FD algorithm that reads

$$\exp^{\frac{-\epsilon^0 K_1^4}{(4.0-K_1^2)^2}}, \tag{2.5.27}$$

where ϵ^0 is the dip filter parameter.

To illustrate the influence of this dip filter on the input wavefield, Figure 2.2 compares the amplitude response of this dip filter for different ϵ^0 values. Also shown in the figure is the wavenumber filter used in the PSPI method. Waves are inhomogeneous for $K_1 \geq 1.0$. The major difference between the family of FD filters (a, b, and c) and the PSPI filter (d) is that the PSPI filter passes all the homogeneous waves undistorted and suppresses only the inhomogeneous waves, while the FD filter attenuates both types of waves. The filter in the PSPI algorithm suppresses most of the inhomogeneous waves rapidly, but, depending on the choice of ϵ^0, the dip filter in the FD algorithm less severely removes the inhomogeneous waves. To suppress more of the inhomogeneous waves, we have to use a large ϵ^0 value. Unfortunately, the homogeneous waves are then attenuated even more severely. The difficulty in applying this dip filter, then, is that the choice of ϵ^0 unavoidly involves a trade-off in the actions on homogeneous and

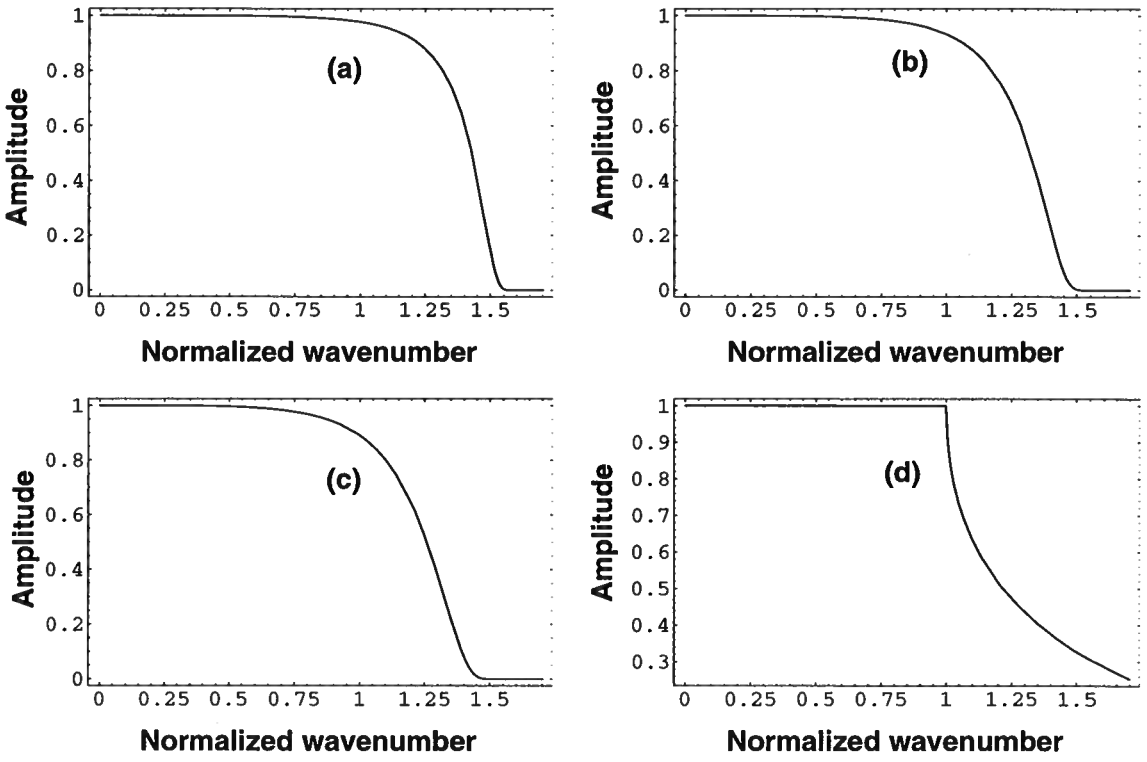

FIG. 2.2. Amplitude variations of the dip (wavenumber) filter for the 45° FD algorithm and the PSPI algorithm. From (a) to (c), ϵ^0 values are 0.1, 0.3, 0.5 for the FD algorithm. (d) is the wavenumber filter in PSPI algorithm; it has no control parameter.

inhomogeneous waves. In contrast, the PSPI filter applied in all the wavenumber-domain algorithms does not have such a shortcoming. The FD migration results with different ϵ^0 values are compared in Chapter 3.

2.6 Relative computation costs

It is difficult to compare computation costs of these algorithms with precision. The difficulty arises from the fact that we have no good way to determine the cost for the different numerical operations on various computer platforms. We know that the + and - operations are far less costly than the * and / operations. On the other hand, all the methods compared here work in the frequency domain; thus complex operations cannot be avoided, and usually the cost of such operations is more than several times that of the pure float operation. Here, to simplify the analysis, I just count the number of float operations performed in each algorithm. To convert the counts of complex and other operations to those of float operations, I made the following assumptions.

1. float multiply and division are the same, counted as one float operation,
2. addition and subtraction for both complex and float numbers are ignored.
3. one complex multiply is equal to four float operations,
4. one complex division is equal to seven float operations,
5. complicated function calls, such as sin, cos, exp, and sqrt, are each treated as five float operations.

Also, since all the methods work in the frequency domain, and each frequency component is independent of one another, I will just analyze the computing cost for a single frequency across just one depth interval.

First assume that n, the number of horizontal samples in the space domain, is a power of 2; then the number of wavenumbers, n_k, is just $\frac{n}{2}$. Now, let us list the count of float operation numbers for the major operations in the four algorithms.

* FFT between space and wavenumber domains $\rightarrow n \log_2 n$,
* Phase shift in the wavenumber domain $\rightarrow 24\ n_k = 12\ n$,
* FD operator for the FFD method in the space domain $\rightarrow 57\ n$,

* FD operator for the FD method in the space domain $\rightarrow 50\ n$,
* Thin-lens term in the space domain $\rightarrow 15\ n$,
* interpolation in the space domain $\rightarrow 6\ n$,

Combining the costs for above operations, we have the computation costs for the four algorithms:

PSPI with n_{ref} reference velocities $\rightarrow 21\ n + n\ \log_2 n + n_{ref}\ (n\ \log_2 n + 12\ n)$

Thin-lens term $\rightarrow 15\ n$

interpolation $\rightarrow 6\ n$

FFT $\rightarrow (1 + n_{ref})\ n\ \log_2 n$

Phase shift $\rightarrow n_{ref}\ 12\ n$

SSF $\rightarrow 27\ n + 2\ n\ \log_2 n$

Thin-lens term $\rightarrow 15\ n$

FFT $\rightarrow 2\ n\ \log_2 n$

Phase shift $\rightarrow 12\ n$

FD with n_{FD} cascaded FD operators $\rightarrow (15 + 50\ n_{FD})\ n$

Thin-lens term $\rightarrow 15\ n$

FD operator $\rightarrow n_{FD}\ 50\ n$,

FFD $\rightarrow 84\ n + 2\ n\ \log_2 n$

Thin-lens term $\rightarrow 15\ n$

FFT $\rightarrow 2\ n\ \log_2 n$

Phase shift $\rightarrow 12\ n$

FD operator of the FFD method in the space domain $\rightarrow 57\ n$,

Based on these estimates for number counts, the estimated costs of the different algorithms are shown in Table 2. Clearly, the SSF method is the fastest one. PSPI costs a factor of n_{ref} more than SSF, and the cost of FFD is between that of the 65° and 80° FD methods, being closer to the latter. In Chapter 3 and 4, I discuss measured computing times in detail.

Method \ n	128	256	512	1024	2048
PSPI ($n_{ref} = 3$)	9780	20300	42000	86800	179000
PSPI ($n_{ref} = 4$)	11900	24800	51300	106000	219000
SSF	4700	9750	20200	41800	86500
65° FD ($n_{FD} = 1$)	8320	16600	33300	66600	133000
80° FD ($n_{FD} = 2$)	14700	29400	58900	110000	236000
FFD	20000	24300	49400	100000	203000

Table.2 Computation counts for various algorithms on different input data sizes. All the numbers are computed using the counts shown in this section and rounded to keep only 3-digit significant numbers. Here n is the input data size; n_{ref} stands for the number of reference velocities in the PSPI method; and n_{FD} means the number of diffraction terms used for different orders of the FD algorithm.

Chapter 3

EXAMPLES OF ZERO-OFFSET MIGRATION

For actual seismic surveys, it is impossible to put sources and receivers together, so we never have zero-offset data. For simple geological structures, the stacked CMP (common midpoint) gathers after NMO (normal moveout) correction are close to zero-offset data. So, for field data, poststack migration is done on stacked CMP data instead of zero-offset data.

Having reviewed the theoretical background of the four closely related algorithms in Chapter 2, I now compare their performance for zero-offset migration of data from the SEG-EAGE salt model (Huang & Fehler, 1997), which was generated for the exploding reflector model using one-way propagation with a high-order finite-difference scheme.

3.1 Description of the SEG-EAGE model

Figure 3.1 shows the velocity model used for the comparisons. It is a 2-D slice through the SEG-EAGE 3-D salt model. This velocity profile contains 320 by 1024 points with a minimum velocity of 1524 m/s and a maximum velocity of 4480 m/s, and the grid spacing is about 12 meters in both horizontal and vertical directions. Figure 3.2 shows the zero-offset data generated by Huang & Fehler (1997) with a finite-difference scheme based on the exploding-reflector model. These data have 1024 traces, with midpoint spacing of 12 m; each trace has 1024 samples at a sampling interval of 8 ms.

The data provided by Huang & Fehler (1997) and used for comparisons of the different algorithms resulted in generally satisfactory imaging results. One shortcoming of those data, however, is that the frequency content of the data was limited because a 20-Hz Ricker wavelet was used as the source function. In a later section, I also generated zero-offset synthetic data for this model with my finite-difference modeling code in order to do experiments with data that have much higher frequencies.

Figure 3.3 shows the "pseudo-reflectivity" section derived from the velocity model

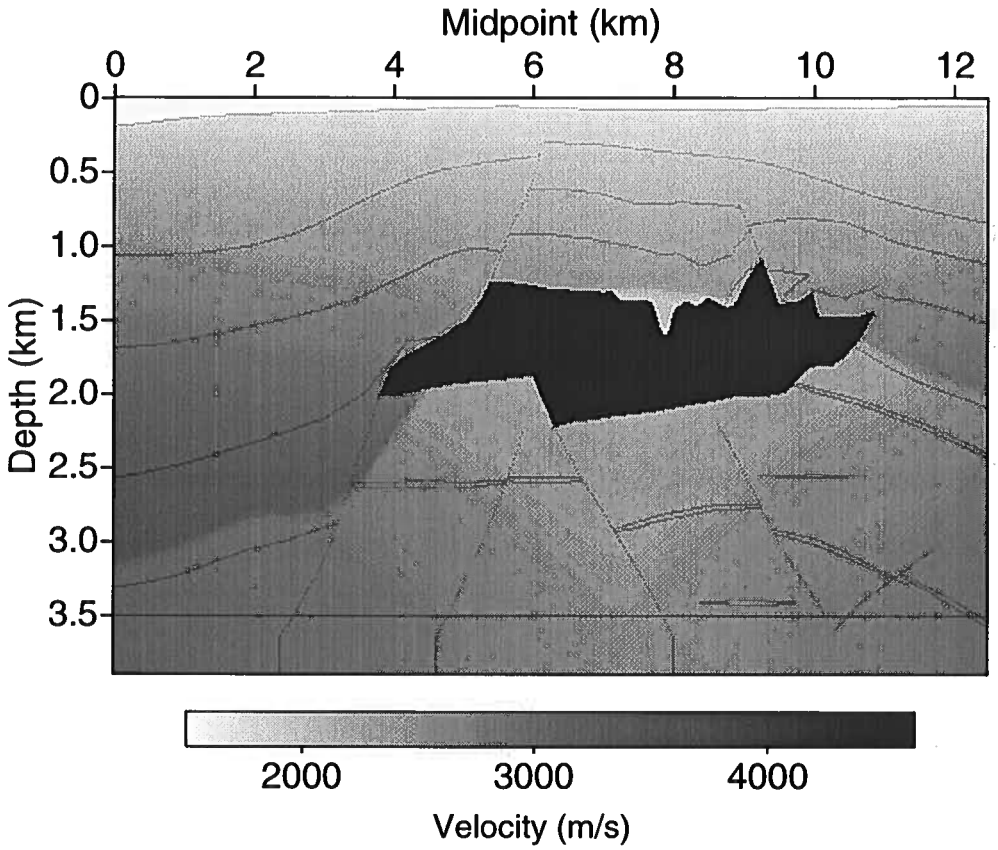

FIG. 3.1. 2-D slice of the SEG/EAGE 3-D salt model. The minimum velocity is 1524 m/s and the maximum velocity is 4480 m/s.

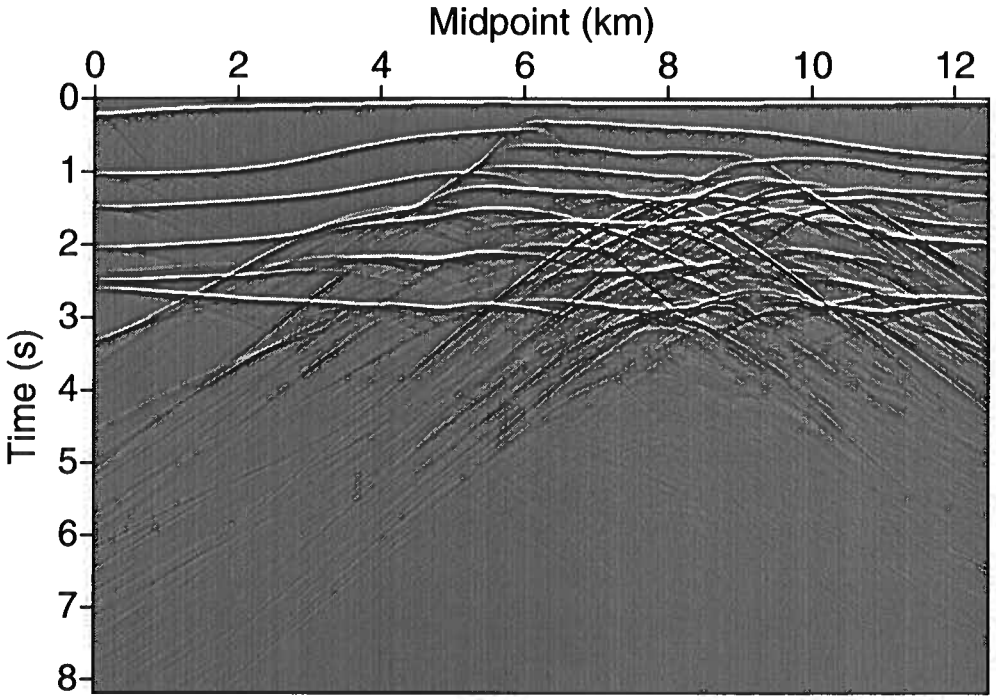

FIG. 3.2. Exploding reflector data for the salt model.

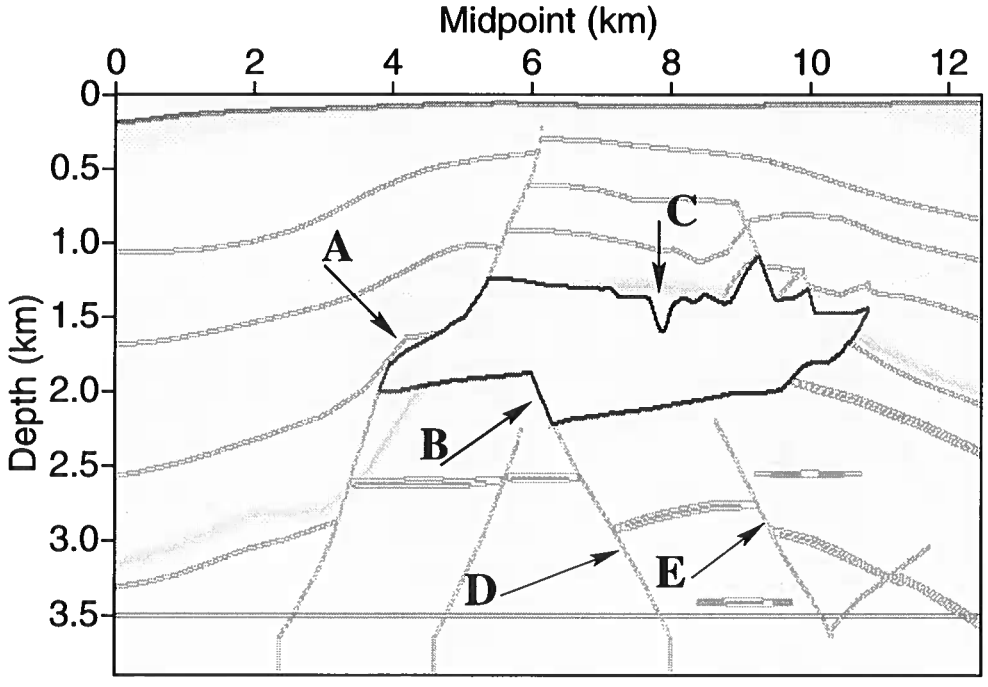

FIG. 3.3. Pseudo-reflectivity section computed using equation (3.1.1).

in Figure 3.1. Migration results with the various algorithms will be compared with this section, the same reflectivity was also used as input for the exploding-reflector modeling code. Given the velocity model, computation of this section is simple. The reflectivity is just the derivative of velocity in horizontal and vertical directions, written as

$$R = \sqrt{(dv/dz)^2 + (dv/dx)^2}. \tag{3.1.1}$$

Some definitions of pseudo-reflectivity are computed from only vertical derivatives. Because I include the horizontal derivative in the above formula, the three vertical reflectors at the bottom of the velocity model also appear in the pseudo-reflectivity section. These vertical reflectors were not included in the synthetic data provided by Huang & Fehler (1997).

The overall structure in the SEG-EAGE salt model is relatively simple compared with that of the Marmousi model (Chapter 4). As seen in the velocity model (Figure 3.1) and the pseudo-reflectivity model (Figure 3.3), the top portion is characterized by gently dipping reflectors, separated by a major fault in the middle at midpoint locations around 6 km. The dip for this fault is about 56 degrees, and the velocity is essentially just depth dependent, as can be seen in the change of gray shades in the velocity model. Beneath the salt body, the structure is also relatively simple. The section under the salt is separated into five blocks by four major faults, with dips ranging from 48 to 54 degrees. Within each block, reflectors generally have mild dip; the medium is nearly homogeneous in the region below the salt.

The difficulty arises for the wavefield close to and inside the salt body, where the salt has much higher velocity than that of the surrounding media, which creates strong lateral velocity variation. The irregular shape of salt body, combined with the high velocity contrast, cause further trouble for seismic imaging of the region close to the salt body. The arrows in Figure 3.3 highlight five regions in the velocity model where we will concentrate on the relative quality of imaging by the four algorithms. Region C contains a sharp depression at the top of the salt that introduces rapid lateral velocity change. The dip for this structure is about 52 degrees. In region A, the reflectors are close to one other; a test of these migration algorithms would be their ability to restore neighboring reflectors. Region B contains a steep reflector that is a sharp boundary between salt and the surrounding media that is an extension of a steep fault beneath the salt body. The dip of this feature is about 50 degrees. In discussing the results below, I focus attention on the performance of the various algorithms in these three areas. Regions D and E pertain to the problem of imaging steep features beneath the salt, which will be discussed later.

As the synthetic data are free from noise, we need not to worry about the input data quality. The only preprocessing I applied to the data is a constant time shift for all the data traces. A 20-Hz Ricker wavelet with peak amplitude delayed by 50 ms had been used to generate the data. I therefore applied a 52-ms time advance (50 ms rounded to the nearest 4 ms) to the data so that the peak time for maximum-energy arrivals would correspond to computed reflection times.

3.2 Migration with the exact velocity model

Before showing the migration results, we can first try to predict the performance of various methods. As the model has simple structures and mild velocity variation above the salt body, all the algorithms should work well for the top portion. For features near the salt body, the SSF method will most likely fail because the small-perturbation assumption breaks in the presence of high lateral velocity contrast. Although the PSPI method involves a small-perturbation assumption as well, with use of a fine increment in the reference velocities the PSPI method should be able to handle these features well. As for the FFD and FD method, accurate treatment in the presence of large velocity variation is their stipulated strength, so they should work well for this model.

3.2.1 Migration comparisons

All the migrations were done with the exact velocity model shown in Figure 3.1. A frequency range 2-55 Hz was selected to be consistent with the bandwidth of the source function — the 20-Hz Ricker wavelet. All the experiments, with parallelized codes of those migration algorithms, made use of about 15-18 CPUs per run. To show the relative computing time of various algorithms, I recorded computation times for processing with just a single four-processor computer. One reason is that most of the computers used in Center for Wave Phenomena (CWP) are also desktop PCs for personal usage. From time to time, the load of jobs on different computers can change rapidly. The second reason is that migrations on the poststack data took very little time when I used more than 15 CPUs; most of the algorithms took no more than 60 s. For such a short CPU time, many factors, such as the network traffic can greatly influence the time for migration. For the timing comparisons, the four-processor machine was dedicated for the computation; I made the timing experiments only when nobody else was using it. By doing so, the comparisons of computing time are more fair.

Method	*Time* (s)	*Method*	*Time* (s)
SSF	127	FD65	120
FFD	205	FD80	199
PSPI	1024		

Table.3 Computation time of various algorithms for the SEG-EAGE salt model.

Table 3 shows the computation time for various algorithms. Among them, the SSF and 65^0 FD methods, have close computation time, as do the FFD and 80^0 FD methods. In Chapter 2, I discussed the theoretical cost for various algorithms. According to the computation counts there, the SSF and FFD should be faster than the 65^0 FD and 80^0 FD method by, respectively, 30% and 17%, which is different from the observed results here. The difference might be due to the varying costs of the fast Fourier transform (FFT) and the rough approximation when I made the counts. The PSPI algorithm is the slowest by far. I used an average eight reference velocities per depth step; hence the method was about eight times slower than the SSF method. Also in this case, PSPI was also about five times more expensive than the other two relatively costly methods — FFD and 80° FD.

The migration results obtained with the different algorithms are shown in Figures 3.5 through 3.13. As expected, all the algorithms give good results in the shallow region where the velocity structure is simple. All the gently dipping reflectors and the 50 degree fault crossing them are properly imaged. Most of the top of the salt body is also well imaged. Beneath the salt, all the algorithms yield variously degraded results; only the horizontal and mildly dipping reflectors are imaged. Faults D and E below the salt body shown in the pseudo-reflectivity model (Figure 3.3) are missing in all the migrated sections. Because all the algorithms have this same problem for the subsalt imaging, it is natural to question whether this is a data problem or a problem of imaging. The most direct guess is that the reflections from these missing faults likely have not been properly recorded in the zero-offset data. To answer these questions, I address this problem in a later section.

Differences lie along the boundary of the high-velocity salt body. As mentioned above, the SSF method will most likely fail when the small-perturbation assumption is violated. As seen in Figure 3.4, all the three small problematic regions in the previous section were somehow poorly imaged, suggesting that the wrong velocity has been used in the migration. In Region A, the low boundary of the salt body does not terminate well, crossing through the fault beside it. For Region B, the dipping event around 6 km has been shifted compared with migration results from other

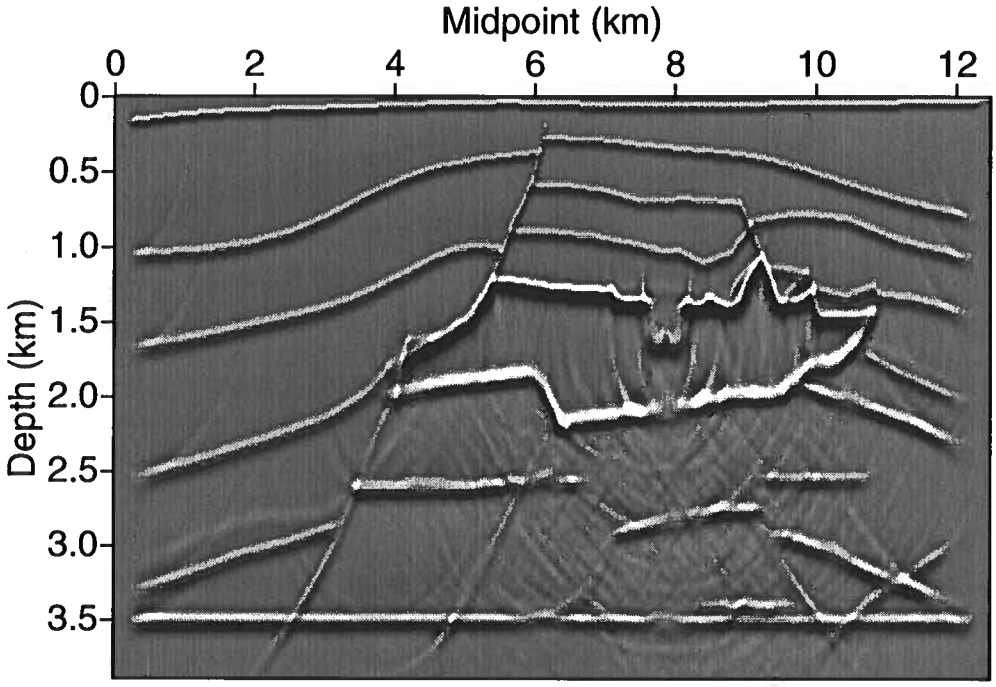

FIG. 3.4. SSF migration.

algorithms. Both detailed features are better seen in the zoomed pictures shown in Figures 3.14 and 3.15. For the depression region (Region C), the SSF method totally failed to yield a focused image of the top of the salt; an apparently over-migrated bowl-shape feature appears inside the salt body.

The failure of the SSF method is due to the large difference between the reference velocity and local velocity. In my implementation, the reference velocity for each depth layer is determined as follows:

$$\frac{1}{v_r} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{v(x_i)}, \tag{3.2.2}$$

where v_r is the reference velocity, $v(x_i)$ is the local velocity at each horizontal sample point, and n is the number of samples in the horizontal direction. Thus, the reference velocity is just the inverse of an average slowness. As the downward continuation process goes through the salt body, the reference velocity has more contribution from high-salt velocity; thus the reflections around the salt body are migrated with a velocity that is much higher than the correct value. This is especially true for

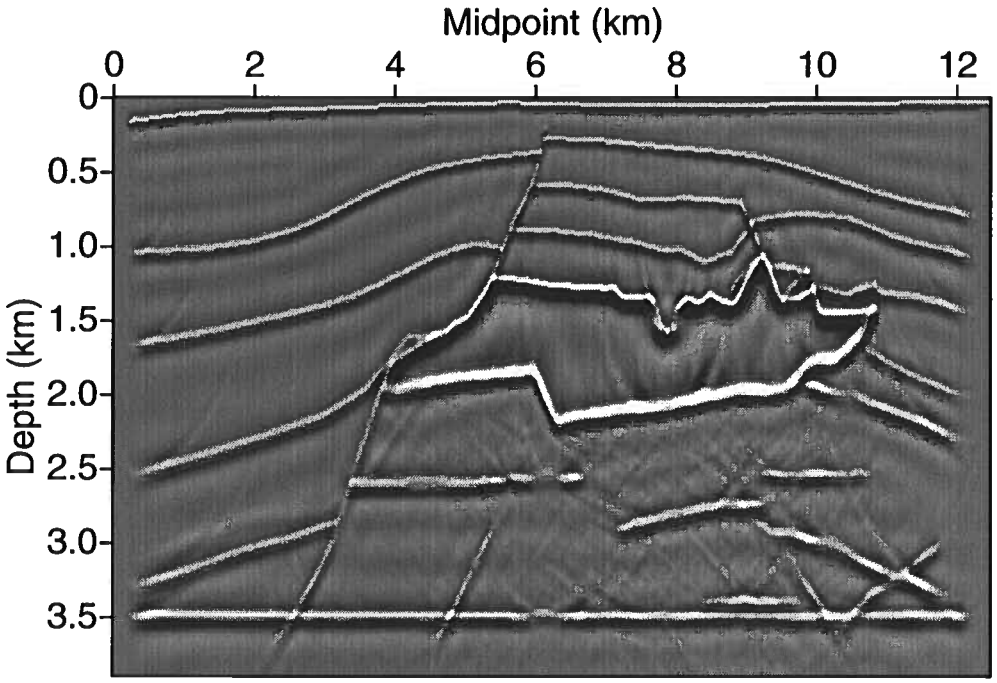

FIG. 3.5. PSPI migration with automatically picked reference velocities. The inhomogeneous wave was attenuated with the wavenumber filter shown in Figure 2.2d.

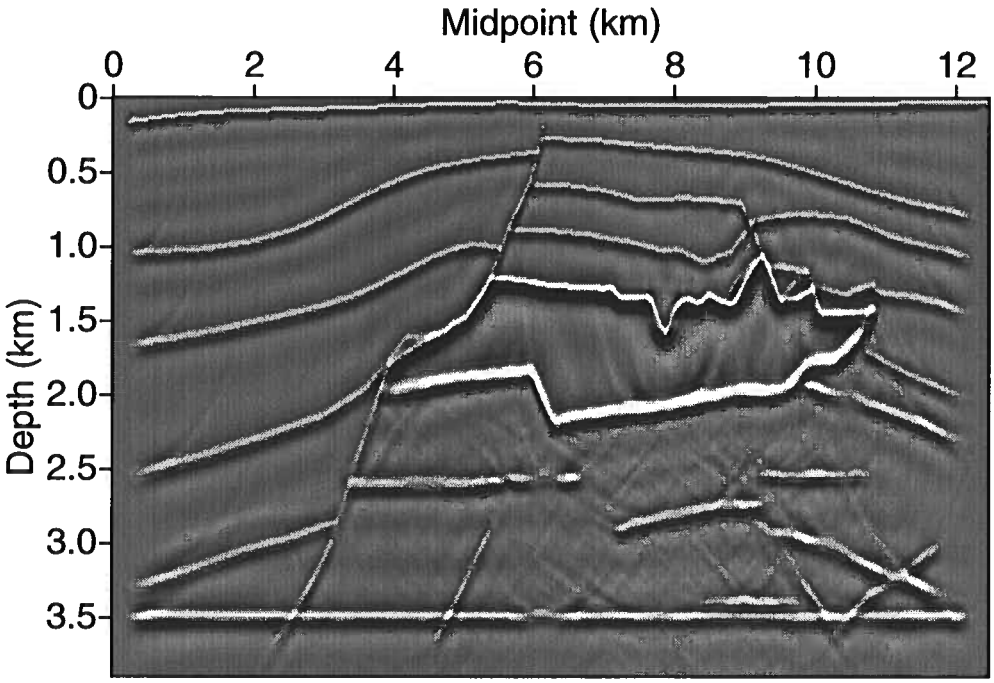

FIG. 3.6. PSPI migration with automatically picked and then manually refined reference velocities The inhomogeneous wave was attenuated with the wavenumber filter shown in Figure 2.2d.

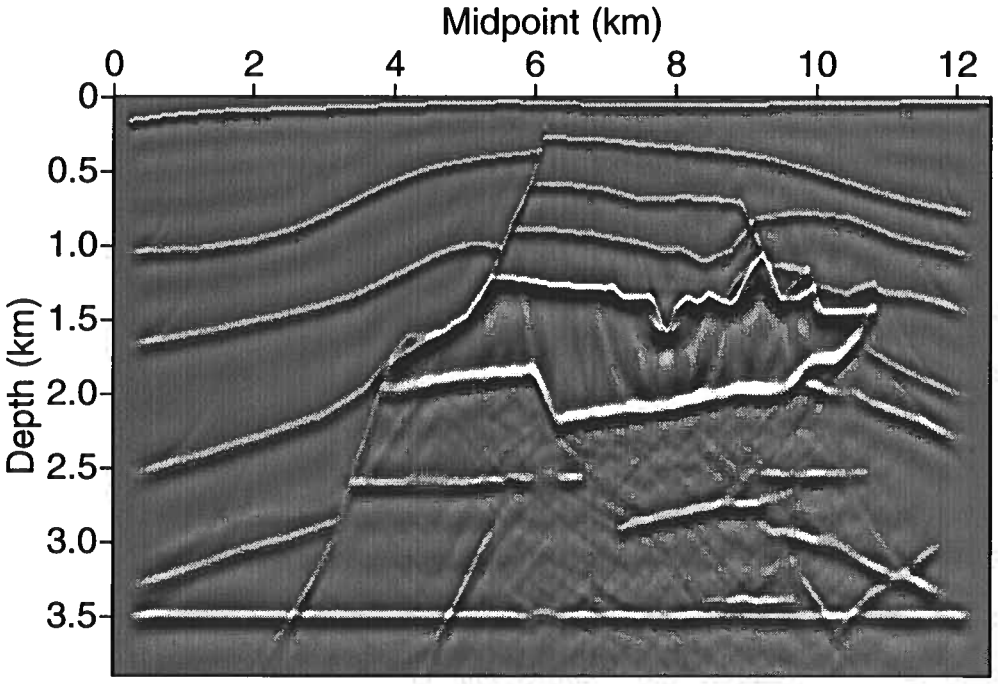

FIG. 3.7. PSPI migration with automatically picked and then manually refined reference velocities. The inhomogeneous wave was eliminated with a boxcar wavenumber filter.

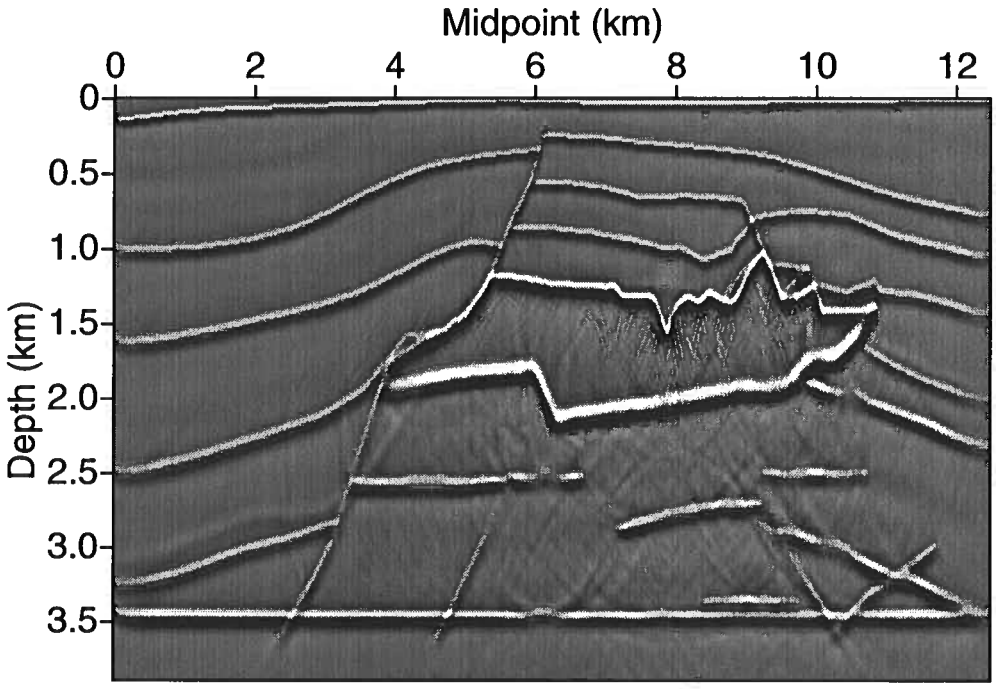

FIG. 3.8. 65^0 FD migration: 1/6 trick parameter γ set to 0.08; dip-filter parameter ϵ^0 set to 0.1.

the small depression region, which has the lowest velocity locally. The consequent velocity error can be as larger as 100%, so the SSF method has seriously overmigrated the depression area, and experiments with other reference velocities did not show improved images. Because the SSF algorithm inherently has weakness for handling strong lateral velocity variation, I did not try to push it further by tweaking the parameters in the algorithm. To obtain an improved image with SSF, the easy solution is to use multiple reference velocities in the SSF algorithm to remedy the small-perturbation assumption. As discussed in Chapter 2, however, by doing that the SSF would be changed toward the PSPI algorithm, with its increased cost.

Contrary to SSF, PSPI obtained a generally good image (Figures 3.5 through 3.7) of Regions A and B. Both reflectors are imaged at the correct location with good focusing, showing the details of the structures. The FD (Figure 3.8 through 3.10) and FFD (Figure 3.13) results had similar performance to that of PSPI for these two regions.

For the depression region (Region C), these three algorithms imaged the reflectors at the correct locations, but their results show differences in quality.

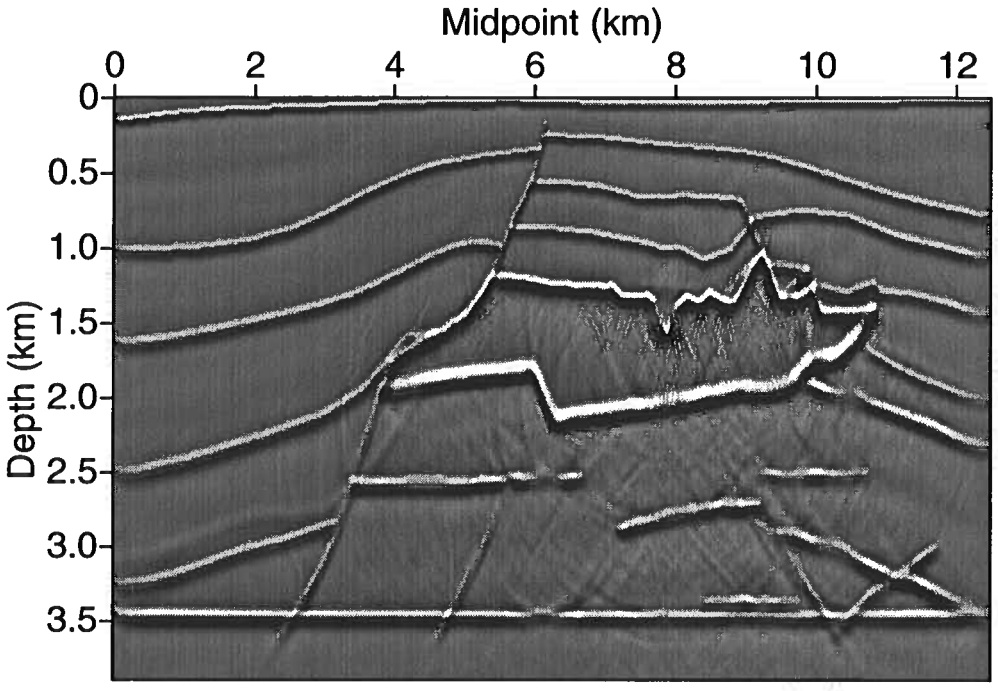

FIG. 3.9. 65^0 FD migration: 1/6 trick parameter γ set to 0.16; dip-filter parameter ϵ^0 set to 0.1.

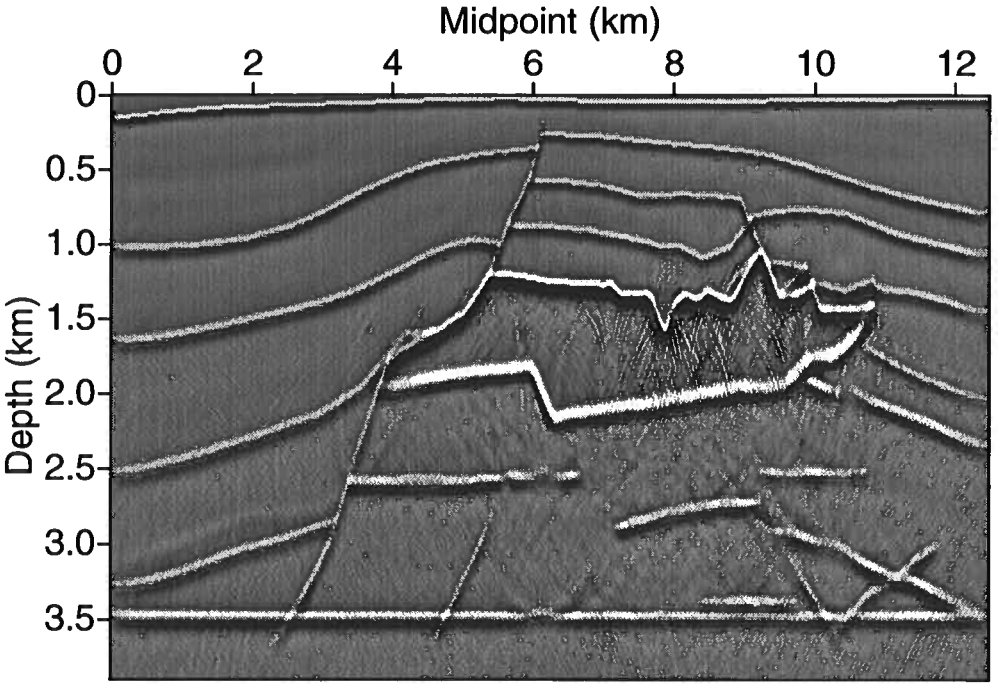

FIG. 3.10. 65^0 FD migration: 1/6 trick parameter γ set to 0.1; no dip filter.

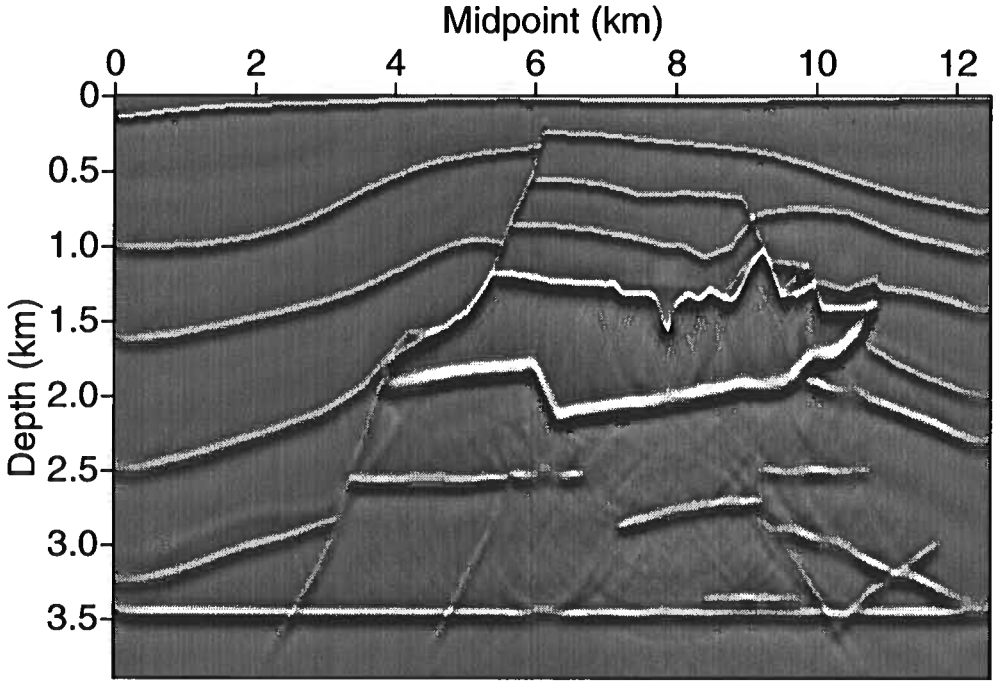

FIG. 3.11. 65^0 FD migration: 1/6 trick parameter γ set to 0.1; dip-filter parameter ϵ^0 set to 0.3.

Compared with SSF, the PSPI method obtained a far better image for Region C. We can estimate the correct location for the top of the salt dome, but depression still shows some over-migration. As we know, if the reference velocities are close to the local velocities, then a good image results. Looking at Figure 3.5, however, it is obvious that a velocity higher than the local one has been used in PSPI for imaging Region C. In the PSPI, the reference velocity is automatically determined by the method of Bagaini *et al.* (1995). Only velocities that arise with relatively high frequency of occurrence along each depth step in the velocity profile are likely to be chosen as the reference velocities. In the velocity model shown in Figure 3.1, Region C, which has the lowest local velocity and has much smaller lateral extent than the salt body beside it, is unlikely to have its low velocity automatically picked by the statistical method.

Manually, I added the velocity inside Region C to the reference-velocity table and did the migration with this new reference velocity (Figure 3.6). The new image of Region C is more accurate than that in the original result (Figure 3.5).

In Chapter 2, I suggested use of a smooth wavenumber filter to suppress the inhomogeneous wave in the PSPI method, instead of a boxcar filter that will generate

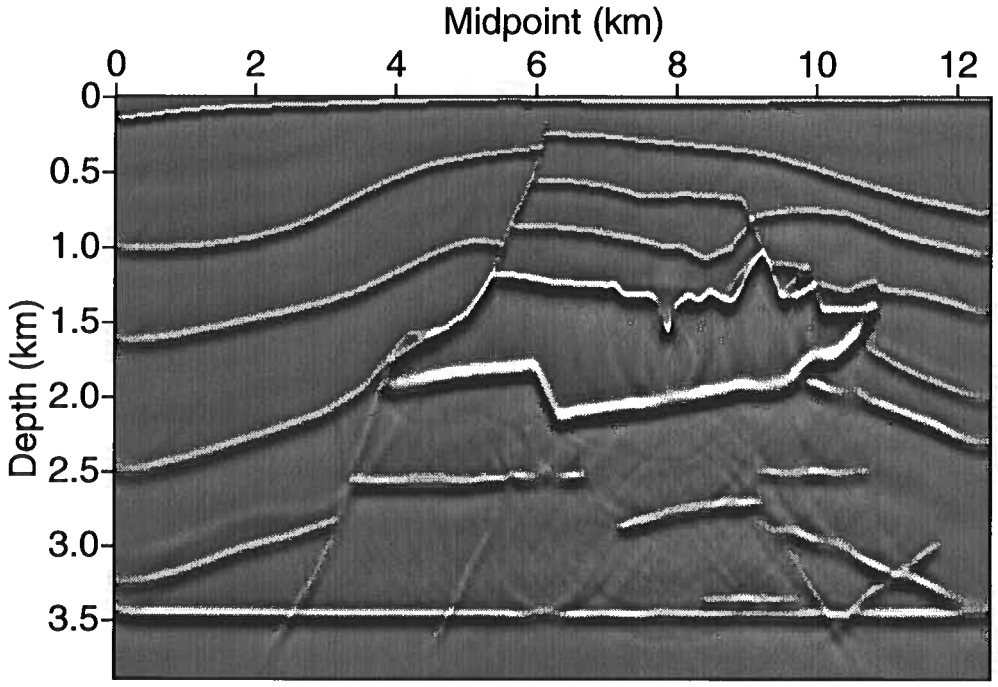

FIG. 3.12. 65^0 FD migration: 1/6 trick parameter γ set to 0.1; dip filter parameter ϵ^0 set to 0.5.

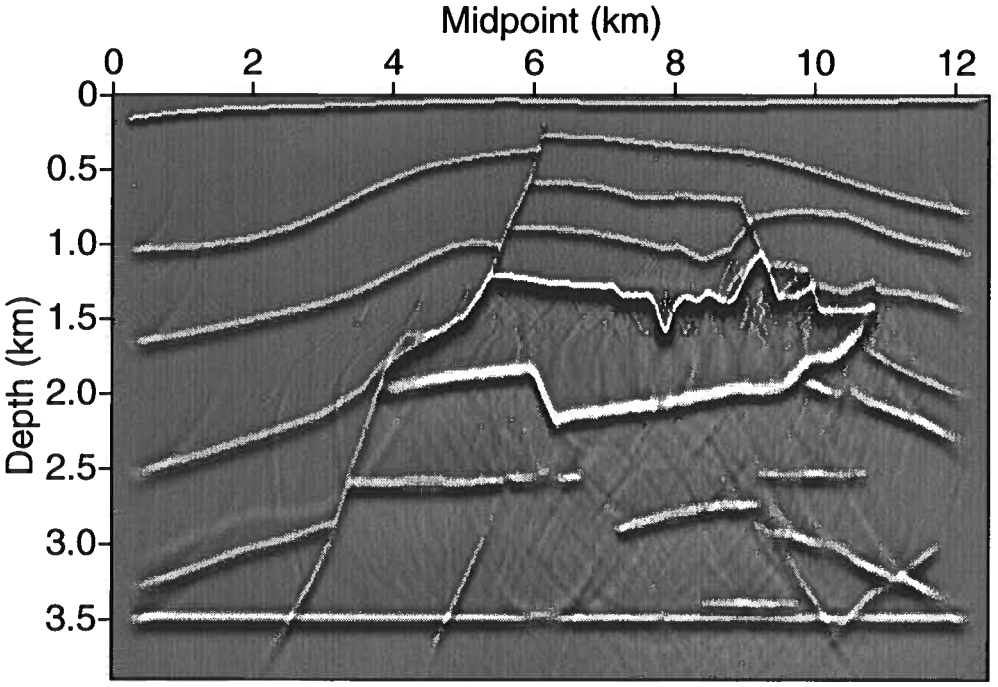

FIG. 3.13. FFD migration.

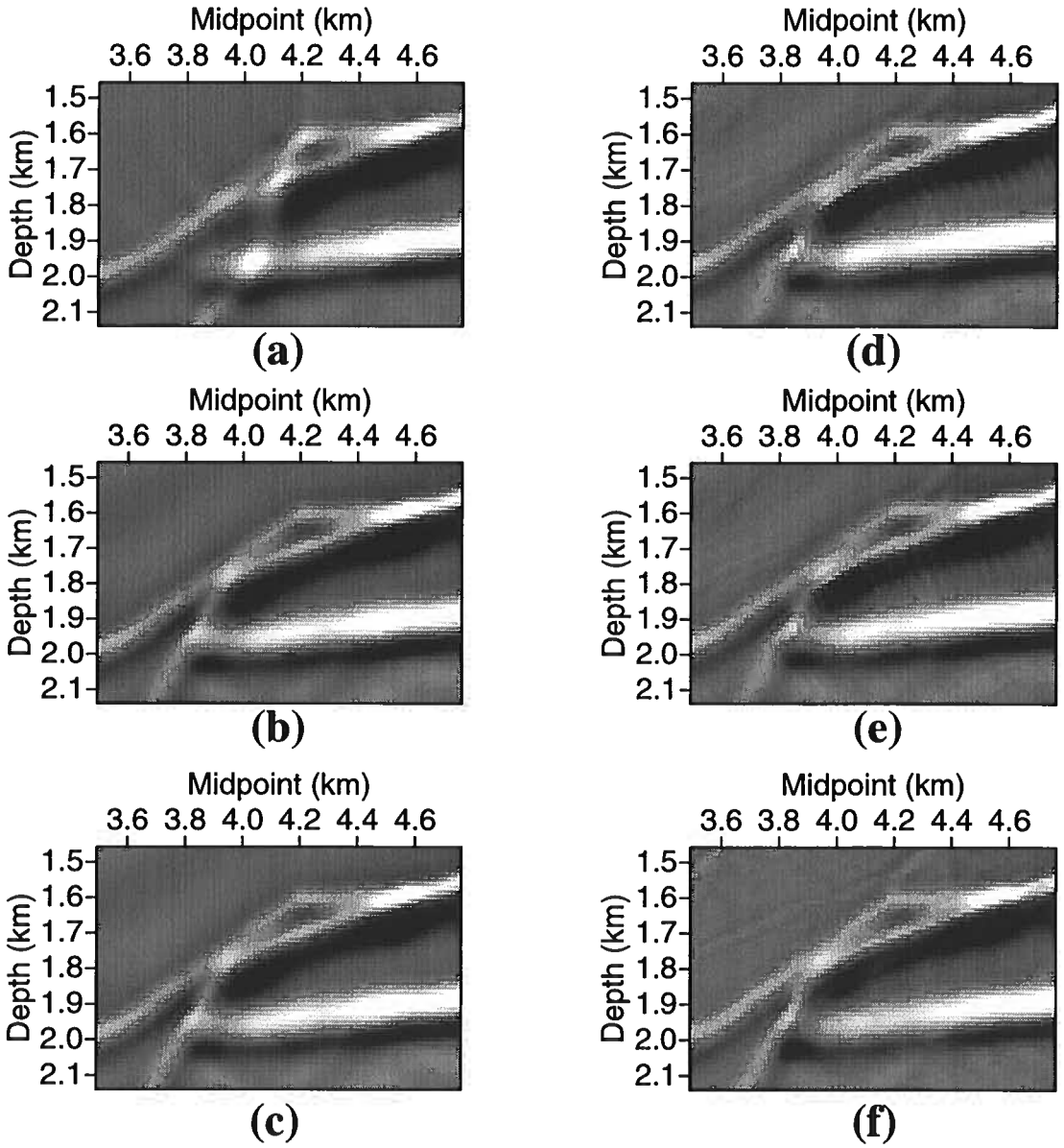

FIG. 3.14. Various migration results for Region A; (a) SSF, (b) PSPI with automatically picked reference velocities, (c) PSPI with automatically picked and then manually refined reference velocities, (d) 65° FD with 1/6 trick parameter γ equal to 0.08, dip-filter parameter ϵ^0 equal to 0.1, (e) 65° FD with 1/6 trick parameter γ equal to 0.16, dip-filter parameter ϵ^0 equal to 0.1, (f) FFD.

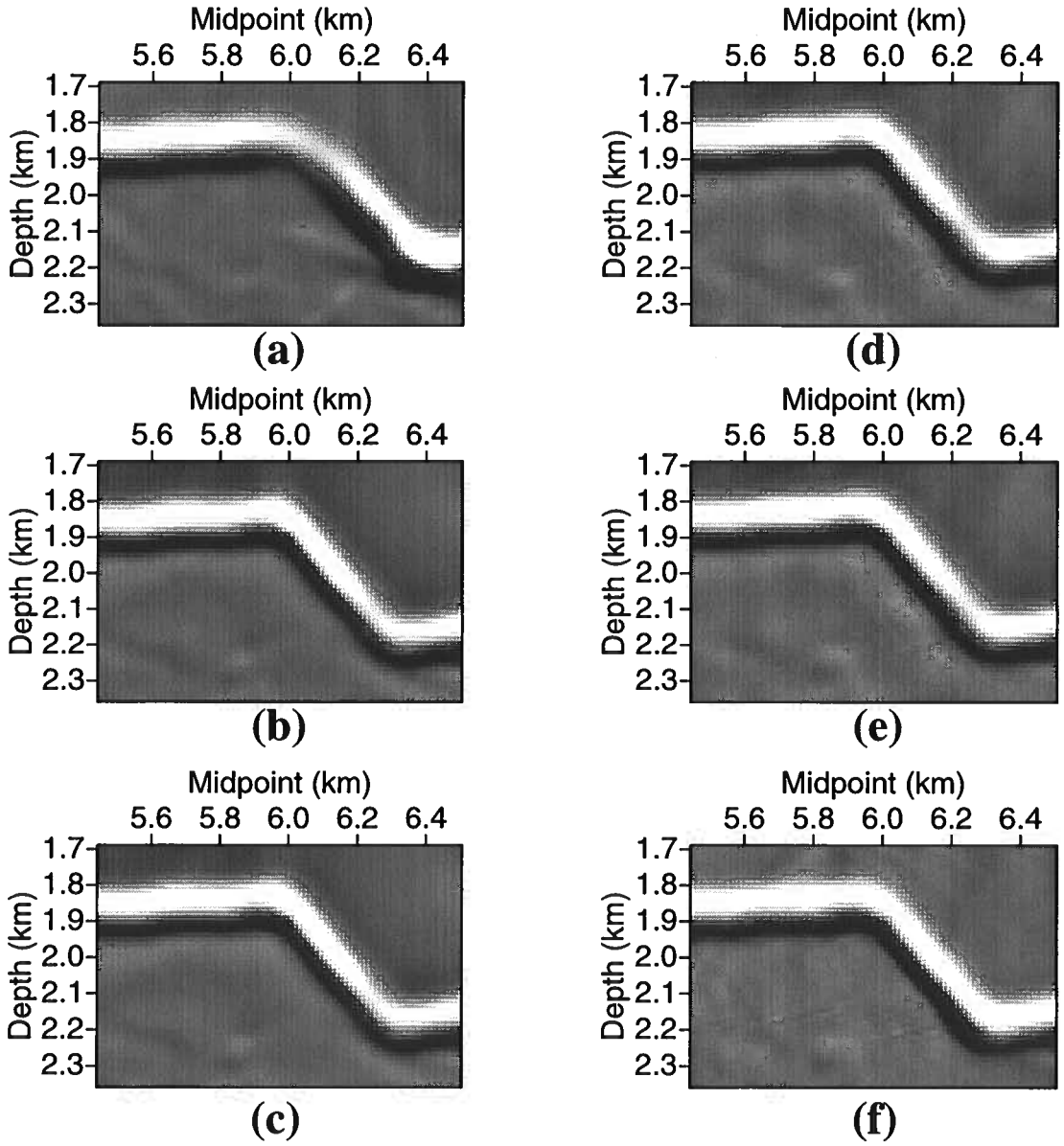

FIG. 3.15. Various migration results for Region B; (a) SSF, (b) PSPI with automatically picked reference velocities, (c) PSPI with automatically picked and then manually refined reference velocities, (d) 65° FD with 1/6 trick parameter γ equal to 0.08, dip-filter parameter ϵ^0 equal to 0.1, (e) 65° FD with 1/6 trick parameter γ equal to 0.16, dip-filter parameter ϵ^0 equal to 0.1, (f) FFD.

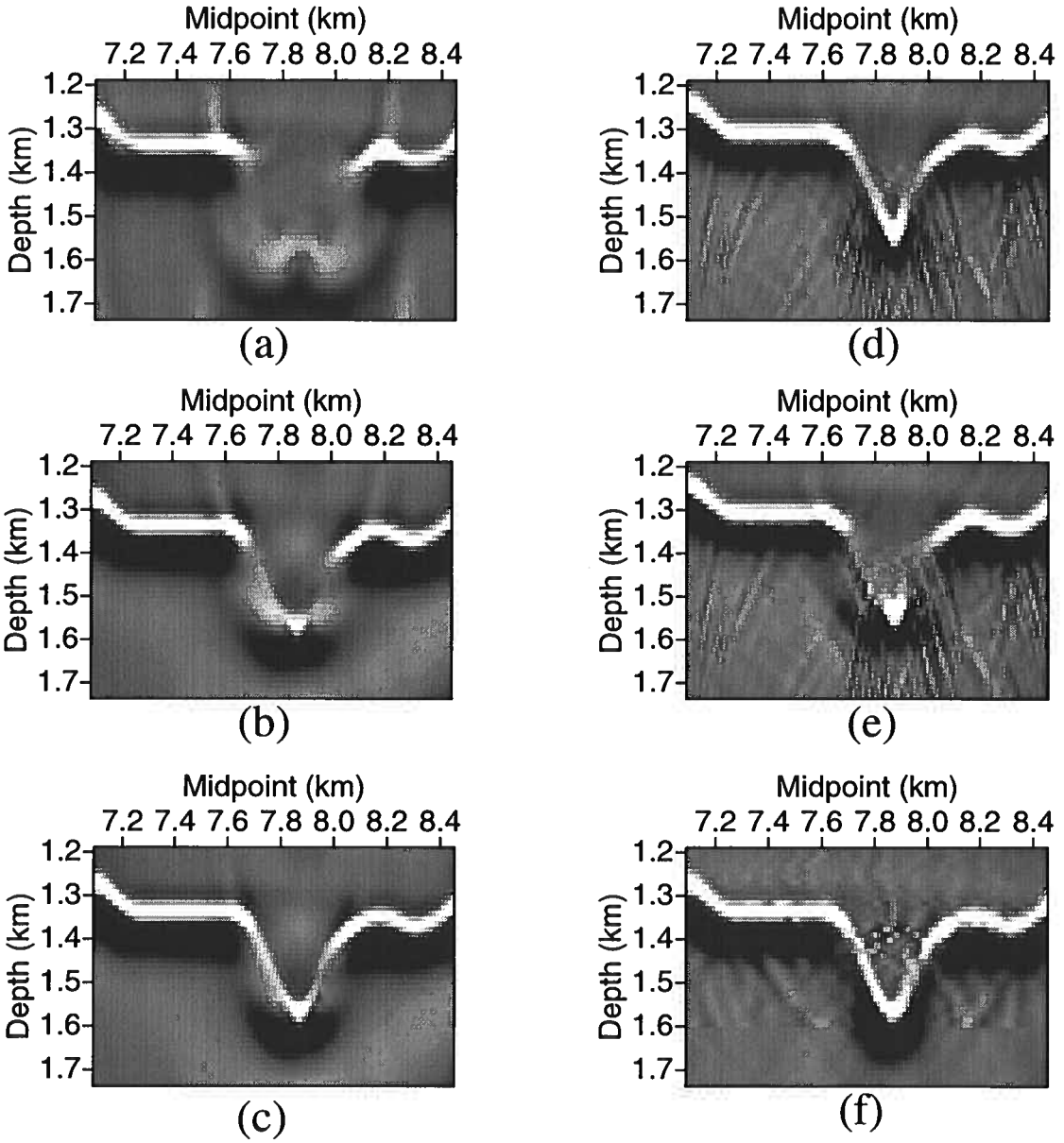

FIG. 3.16. Various migration results for Region C; (a) SSF, (b) PSPI with automatically picked reference velocities, (c) PSPI with automatically picked and then manually refined reference velocities, (d) 65° FD with 1/6 trick parameter γ equal to 0.08, dip-filter parameter ϵ^0 equal to 0.1, (e) 65° FD with 1/6 trick parameter γ equal to 0.16, dip-filter parameter ϵ^0 equal to 0.1, (f) FFD.

numerical noise. Here, the difference in performance for these two choices can be seen in Figures 3.6 and 3.7. Figure 3.7 shows a poorly imaged top boundary of the salt, and at the base of the depression we see artifacts that do not appear in Figure 3.6.

Because all the dipping reflectors in the SEG-EAGE salt model have dips no more than 60°, I mainly used the 65^0 FD algorithm described in Chapter 2 for FD imaging of this model. This implicit FD method can handle strong lateral variation in velocity because the FD operator is a localized operator, so we expect the FD method to yield a good image in migrating the model data. In Figure 3.8, the FD method has imaged the reflectors in Region C. Both walls of those reflectors have been imaged at correct location. Nevertheless, the result of the FD algorithm still could be improved. For example, note that the relatively strong numerical noise that interferes with the reflections from the flanks of the depression.

As I did with the PSPI algorithm, I adjusted the parameters for the 65^0 equation in an attempt to improve the image. The FD algorithm has two parameters: one is the 1/6 trick parameter γ mentioned in Chapter 2, which helps to increase the accuracy of difference operation; the other is the dip-filter parameter ϵ^0 described by Claerbout (1985) and also discussed in Chapter 2, which is used to attenuate inhomogeneous waves.

The FD result in Figure 3.8 is obtained by setting γ to be 0.08. Changing the value of γ to 0.16, the image in Figure 3.9 has degraded compared with that in Figure 3.8, especially the depression (Region C). Although Claerbout (1985) suggested use of 1/6 for γ, for the migration of the SEG-EAGE model, $\gamma = 1/12$ gave a better result.

For the above two results, the dip-filter parameter ϵ^0 has been set to 0.1. Now let us remove the dip filter (set it to zero). The migration result (Figure 3.10) shows greatly improved amplitude for the reflectors in Region C. Meanwhile, we also see more numerical noise in the background throughout the image. Figures 3.11 and 3.12 show the migration results obtained by setting ϵ^0 to 0.3 and 0.5. Compared with Figure 3.10, they have less numerical noise because of the increased attenuation of inhomogeneous waves with use of larger ϵ^0 values. Notice, however, that the reflections in Region C have been severely attenuated as well. Consistent with the theoretical analysis in Chapter 2, when applying the dip (wavenumber) filter in FD algorithm, we attenuate both inhomogeneous and homogeneous waves. Since the attenuated homogeneous waves relate to steep features in the SEG-EAGE model, the attenuation of homogeneous waves with use of larger ϵ^0 values is manifested as the disappearance of the steep portions of the depression in Region C. Because the use of the dip filter in the FD algorithm can degrade the migration image, we need to be careful about the choice of ϵ^0. The plots of the filter responses shown in Figure 2.2 suggest that $\epsilon = 0.5$ is not a severe filter. Nevertheless, the migration result with

$\epsilon = 0.5$ significantly attenuates the depression in Region C. The mismatch might be due to the breakdown of the approximation $\epsilon \ll 1.0$ for the choice $\epsilon = 0.5$.

Finally, let us see the result from the FFD method (Figure 3.13). The FFD result is comparable to the improved result of PSPI and FD. The FFD method also has an FD operator with a similar 1/6 trick parameter γ. Benefitting from the experiments with the FD algorithm, I set γ to 0.1. Different from the FD algorithm, the dip filtering in the FFD method works in wavenumber domain; it uses the same wavenumber filter as in PSPI and SSF to suppress inhomogeneous waves. Thus, the image of FFD has less numerical noise than that of the FD result without dip filter, but its image of the depression reflector in Region C is as sharp as that of that FD result.

3.2.2 Zoomed display of the three problematic regions

In previous section, I compared the performance of the four algorithms primarily on the three regions that most show their differences. Here, let us zoom into those regions to take a close look. Regions A and B are shown in Figure 3.14 and 3.15. Only the SSF method shows difference in imaging from that of the other algorithms. As discussed above, this is due to the breakdown of small-perturbation assumption in the SSF method. Other methods work well for regions A and B because they can handle severe velocity variation.

Images of the depression in Region C by various algorithms are shown in Figure 3.16. The SSF method did a particularly poor job in imaging this feature; the velocity contrast for this reflector is beyond capability of the SSF method. All other three algorithms do better in imaging the depression, but still with significant differences in image quality, which can be altered by adjusting the special parameters in each algorithm.

3.2.3 Missing steep features beneath salt

Most of the algorithms did adequately in imaging the salt body and the shallow portion above the salt. None of them, however, obtained a good image of steep features beneath the salt. To see whether or not the problem is due to shortcomings in the algorithms, I used the 90° FD method to migrate the model data (Figure 3.17). This result shows no improvement over that of other approaches and that of the low-order FD algorithms. The horizontal and mildly dipping features below the salt body have been properly imaged. The steep events, especially in Regions D and E in the middle of the velocity section (Figure 3.1), however, are totally missing.

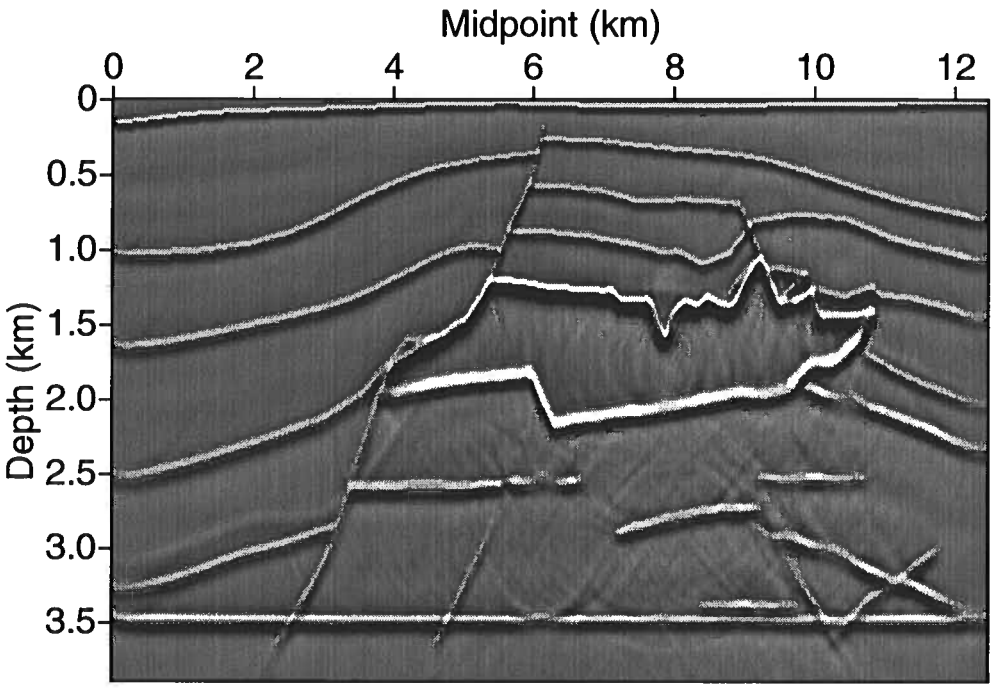

FIG. 3.17. Second-order 90^0 FD migration.

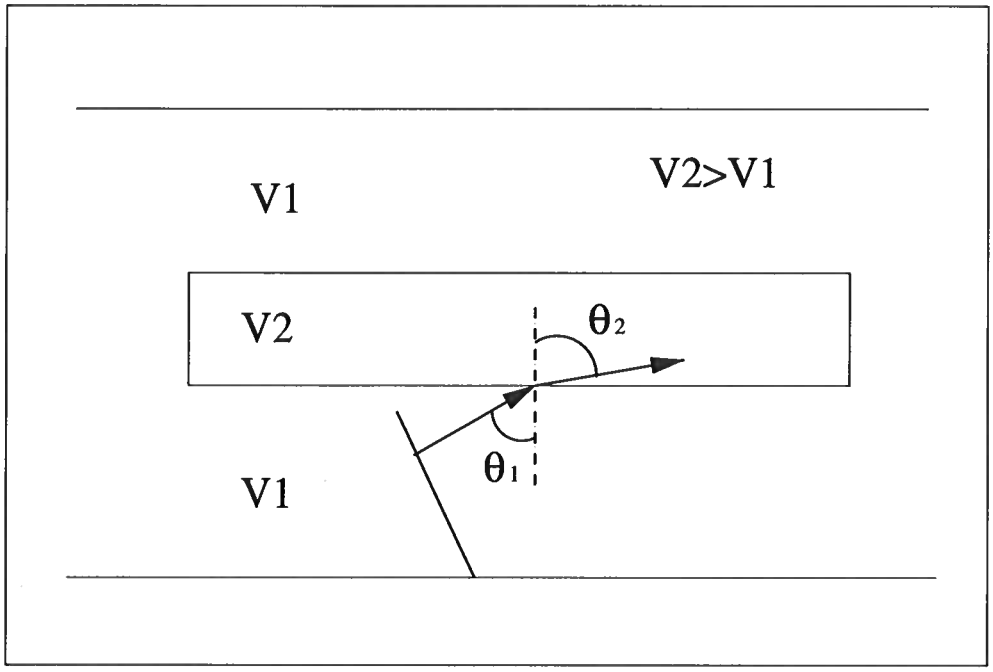

FIG. 3.18. Simplified salt model with steep reflector beneath the salt

To explain the problem of missing steep events under salt body, let us see how zero-offset rays (normal incident on the reflectors) behave for this situation. Figure 3.18 illustrates a simplified salt model, wherein the medium surrounding the salt has a velocity v_1=2500 m/s, and the salt body is represented by a rectangular block with velocity v_2=5000 m/s. Thus the velocity ratio between salt and media around it is 2.0, similar to the ratio in the SEG-EAGE model. The steep reflectors below the salt have dips of about 50° in the SEG-EAGE model. In exploding-reflector modeling, zero-offset rays are normal to the reflectors. When a ray reaches the salt boundary, the incidence angle is the same as the dip of that reflector; thus, in Figure 3.18, θ_1 is 50°. According to Snell's law, the incidence and transmission angles have the relationship,

$$\sin\theta_2 = \sin\theta_1 \frac{v_2}{v_1}.$$

Here $\sin\theta_2$=2 $\sin 50°$=1.53. As $\sin\theta_2 \geq 1.0$, the transmitted waves become inhomogeneous, and thus cannot be observed in the unmigrated data. Inspection of the unmigrated data (Figure 3.2) confirms the absence of reflections from the steep subsalt features. This is not the result of any aperture-size limitation, so simply increasing aperture size for modeling and migration will not solve the imaging problem. Because no reflections from the steep subsalt events exist in the unmigrated data, those features cannot be imaged. Therefore, the final migrated image has no contributions from those steep subsalt features.

Chapter 4

EXAMPLES OF PRESTACK MIGRATION

In this chapter, the four algorithms are applied to prestack data, on the Marmousi model (Versteeg & Grau, 1990). Before showing results, let us consider differences between prestack and poststack migration.

For poststack migration, the stacked data are treated as zero-offset data, and then one-way propagation is applied to these data using the exploding reflector model (Claerbout, 1985). Finally, the imaging condition involves just taking the wavefield at zero time for each depth step through the downward continuation, which corresponds to summing up all the frequency components in the frequency domain.

If the geologic structure is complicated, the stacked data cannot properly represent zero-offset data, and poststack migration cannot cure shortcomings in the data. We need to perform prestack migration. Similar to poststack migration, in prestack migration, the wavefield propagation is still performed by a one-way propagator. Now, however, the source and receiver wavefields are propagated separately, i.e., forward propagation of the source wavefield and backward propagation of the receiver wavefield.

The imaging condition in prestack migration is significantly different from that of poststack migration. Suppose both source and receiver wavefields have been properly propagated to a certain depth step; let the source wavefield be denoted by S (incident wave), the receiver wavefield by R (reflected wave), and the reflectivity at that depth by W (Earth impulse response). These three functions are linked by the convolution operation in the time-space domain; expressed in the frequency domain, the relationship is

$$R = SW. \tag{4.0.1}$$

The goal of prestack imaging is to obtain the reflectivity W; thus, the imaging condition is just division in the frequency domain

$$W = \frac{R}{S}, \tag{4.0.2}$$

which, in practice, is modified to

$$W = \frac{RS^*}{SS^* + \epsilon}. \tag{4.0.3}$$

where S^* is the complex conjugate of source wavefield, and ϵ has a small constant value to keep the numerical stability, avoiding the possibility of division by zero.

If we assume that the source wavefield is a propagating Dirac delta function, SS^* is just unity in the frequency domain, so the denominator becomes just a scaling factor. The imaging condition can thus be further simplified to

$$W = RS^*. \tag{4.0.4}$$

Aside from the difference in imaging condition, prestack migration differs from poststack migration in the input data format as well. Unlike the poststack migration, which works on only a single trace at each midpoint, prestack data sets are much larger and can be sorted as either common-shot gathers, common-receiver gathers, or common-offset gathers. Sorting of the data is based on the geometry of source and receiver positions for the given input traces.

The shot record, or common-shot gather, is a collection of traces that have the same source position, and is thus just the physical wavefield recorded along the Earth's surface when a source is excited. The common-receiver gather is, in a sense, a reciprocal of the common-shot gather, wherein source and receiver positions are switched. That is, it is a collection of traces that were recorded at the same receiver position. Prestack migrations using these two different data arrangements yield the same results for the PP data (P-wave source and reflected P-wave data), but they differ for mode-conversion data such as the PS data (P-wave source and reflected SV-wave data). In this case, the common-shot gather might be PS data, while the common-receiver gather is SP data (effectively, SV-wave source and reflected P-wave data). In common-offset data, traces are organized by constant offset between source and receiver.

The migration algorithms based on wavefield extrapolation, such as the four algorithms compared in this thesis, can work with either common-shot gathers or common-receiver gathers. For them to work on common-offset data, they can be

applied repeatedly, each time with a single-trace migration (each trace is treated as a shot gather). Single-trace migration is just what Kirchhoff-type migration does. In that approach, a traveltime table is first built by ray tracing, providing an incomplete Green function that accounts for only primary arrivals, as opposed to the whole wavefield. Then all the traces in the input data are processed one by one — a single trace is read in, and its amplitudes are distributed to all the possible output points according to information in the traveltime table. So Kirchhoff migration operates independent of the input data arrangement. For wavefield extrapolation techniques, however, it is computationally expensive to mimic what Kirchhoff migration does; building the complete Green function and storing it is impractical for routine processing of 3-D prestack data (2-D prestack migration is also expensive, but with advances in computer hardware technology, it does not pose a computational challenge, even with a low-cost PC cluster).

4.1 Marmousi model

The Marmousi model (Versteeg & Grau, 1990) originated from a 1990 workshop for testing different migration velocity analysis (MVA) techniques. A complicated geological model was designed by Institute Francais Du Petrole (IFP), and synthetic data were computed using a finite-difference solution to the 2-D acoustic wave equation. A band-limited filter was applied to suppress some of the numerical noise from the synthetic data. Since then, the Marmousi model have been used extensively as an industry benchmark for migration velocity analysis and prestack migration algorithms.

Within the Center for Wave Phenomena (CWP), at Colorado School of Mines, this model has also been studied by several students over the years. Liu (1995) applied his analytic migration velocity analysis technique to the Marmousi model data using an implementation of Kirchhoff migration. Salinas (1996) used and extended Liu (1995)'s Kirchhoff migration code to study the influence of near-surface time anomalies on imaging the complex structure, testing the Kirchhoff datuming method on the Marmousi model data. Tjan (1995) worked on the residual static-correction technique for the complex structure based on implicit finite-difference migration and demigration; he demonstrate the effectiveness of this technique on the Marmousi data contaminated by arbitrary static shifts. Later, Jenner et al. (1997) imaged this model with a more accurate implicit FD migration algorithm; he also attempted to repeat the work of Tjan (1995) with that implicit FD migration/demigration algorithm. Le Rousseau (1998) tested his implementation of the generalized phase-screen migration algorithm on the Marmousi model as well.

The velocity model of the Marmousi data, shown in Figure 4.1, contains 369 (lateral)

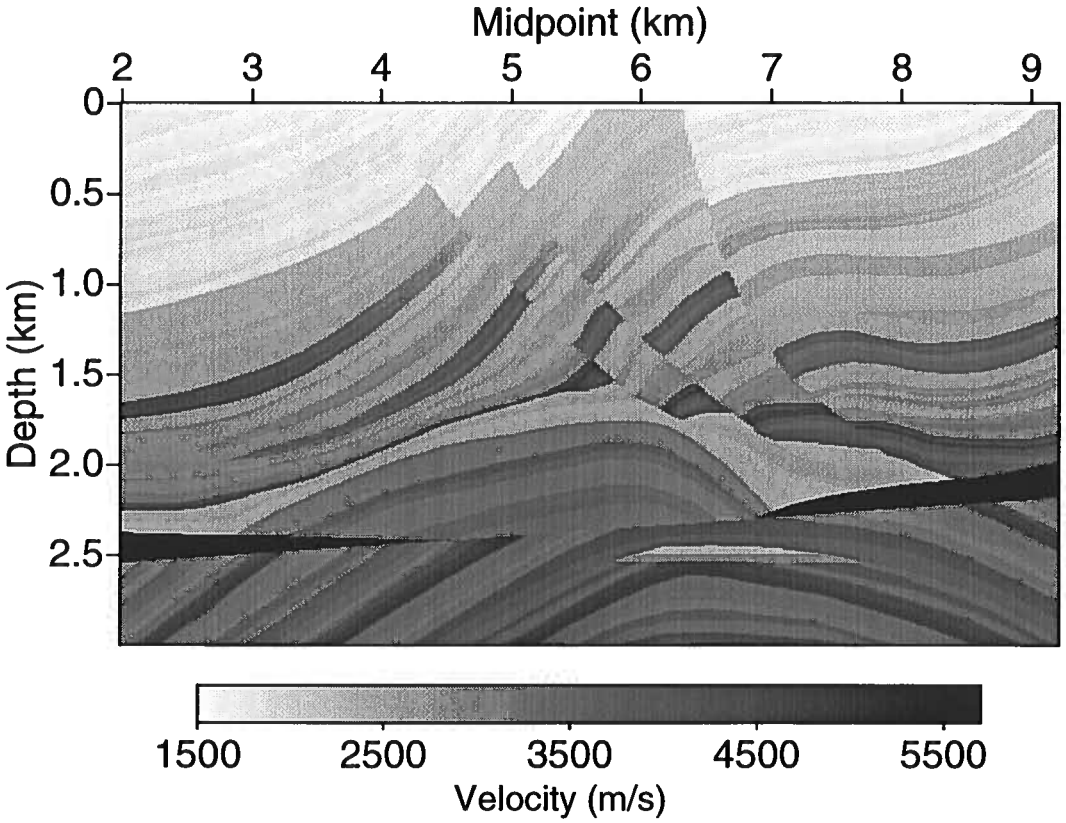

FIG. 4.1. Velocity model for the Marmousi data. The minimum velocity is 1500 m/s and the maximum velocity is 5500 m/s.

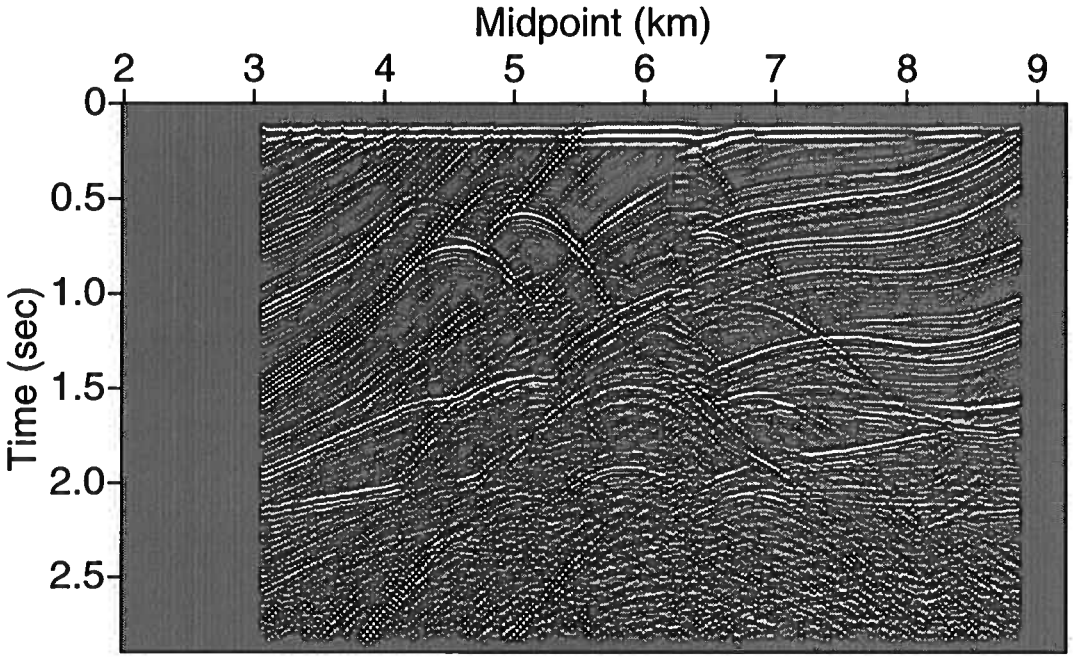

FIG. 4.2. Shortest-offset (200 m) Marmousi data.

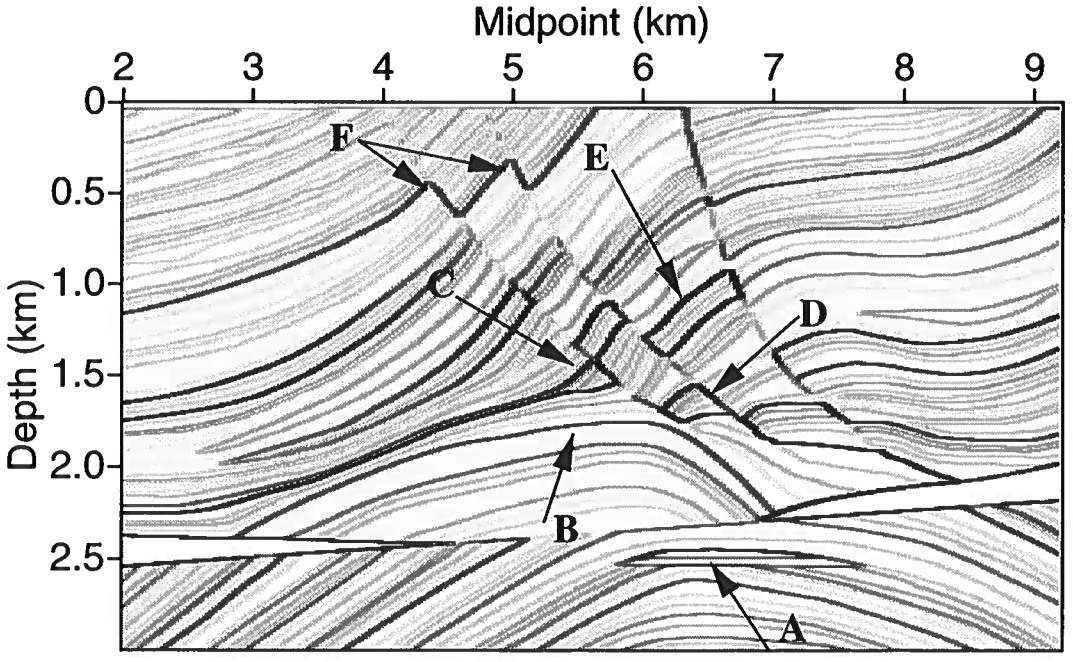

FIG. 4.3. Pseudo-reflectivity model for the Marmousi data.

by 750 (vertical) grid points with lateral spacing of 25 m and vertical step size of 4 m. An initial test showed that resampling the vertical interval to 8 m generates reasonable imaging results for all the migration algorithms, so the velocity model has been resampled to 369x375 points for all the migration results shown here.

The 2-D synthetic data (a shortest-offset section is shown in Figure 4.2) were computed by a finite-difference scheme that solves the acoustic two-way wave equation. A total of 240 shot gathers were generated, with a shot spacing of 25 m. The source position starts at midpoint position 2 km and ends at 8 km. Within each shot gather are 96 traces with a 25 m group interval; the near-offset is 200 m and the far offset is 2750 m; with 750 samples at 4-ms interval, each trace contains 3 s of data.

Similar to the SEG-EAGE model, a pseudo-reflectivity model is computed via equation (3.1.1) and shown in Figure 4.3. Most of the imaging difficulties highlighted later are at locations indicated by black arrows in this model.

Compared with the SEG-EAGE model, the difficulties in imaging the Marmousi model are more related to the complexity of the geological structures than to strong velocity variation. We do have two particularly high-velocity layers on the left and right sides at the depth of 2.5 km, with the highest velocity in the model — 5500 m/s. These two blocks have a simple wedge shape, with a dip less than 5°, so they are not difficult to image. The strong velocity contrast in these two areas make them standout with strong amplitude in the final image.

The main region of difficulty in imaging the Marmousi data is the target zone (Region A), a low-velocity wedge located at (x=6.5 km, z=2.5 km). As waves from the surface propagate through the complicated structures above the target zone, errors that arise from inaccurate imaging algorithms (such as conventional Kirchhoff algorithms) will accumulate; once reaching the target zone, these accumulated errors distort the wavefield, thus degrading the target zone image unacceptably. In the migration results shown below, however, all the four algorithms compared here readily image the target zone, so for the wavefield-extrapolation techniques used here, the target zone is no longer a difficulty.

Above the target zone is an anticline structure (Region B). This structure contains several thin layers within it, and the velocity contrast between each layer is weak. Though the structure of this anticline is simple, the image for this structure can again be distorted by accumulated errors due to complexity of the overburden.

In the middle of the model, are three major faults spanning midpoints from 4 km to 7.5 km, going as deep as 1.8 km. From left to right, the dips for these faults are about 37, 50, and 70 degrees. The shallow part (Region F) of these three faults has strong

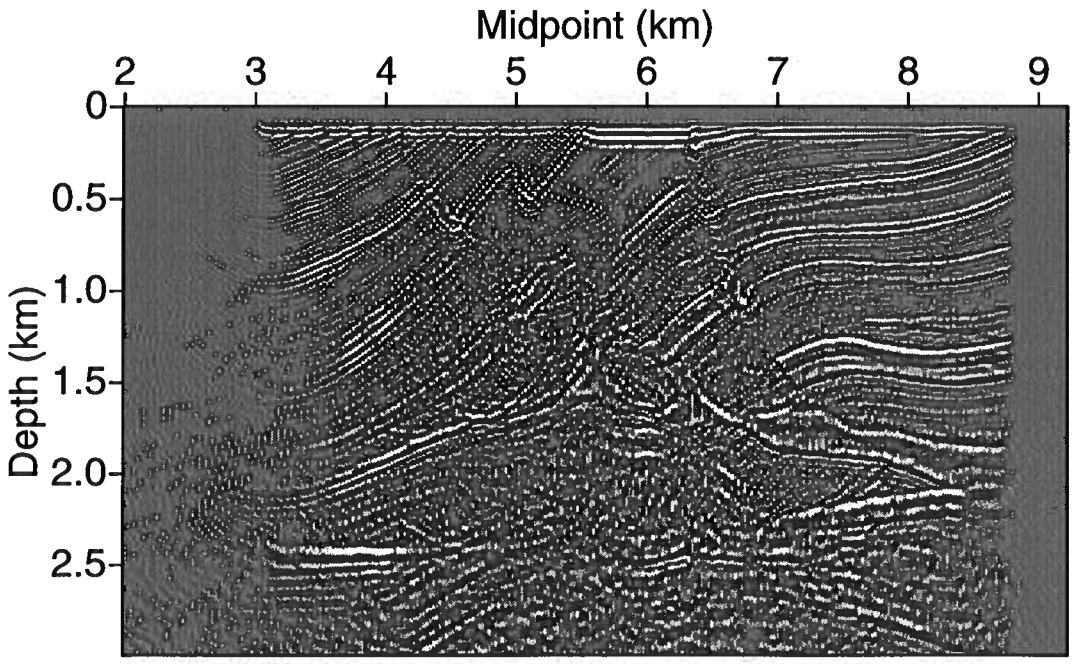

FIG. 4.4. Zero-offset FFD migration on the shortest-offset (200 m) data with the exact velocity model shown in Figure 4.1.

velocity contrast; thus the amplitude in the pseudo-reflectivity model (Figure 4.3) is strong. The deep part of these faults has mild velocity contrast, so we do not expect to see the image of these faults directly in the migration results shown later. Instead, the positions of the faults can be estimated by comparing the images of gently dipping reflectors on either side of the faults. Within the faulted zone, several subtle structures (Regions C, D, and E) need our special attention because images of those small structures can be distorted by errors in migration.

Away from the faulted area, on the left and right sides of the shallow section, the model has relatively simple structure. The velocity variation is weak both laterally and vertically, and most of the reflectors have dips less than 10 degrees. For imaging these two regions, we can expect even a time-migration algorithm, such as the phase shift method (see Chapter 2), to work well.

4.2 Zero-offset migration on shortest offset data

Let us first see how zero-offset migration works on this complicated model. As mentioned in the description of the Marmousi model, zero-offset data are not generated for this model, so the nearest offset (200 m) data (shown in Figure 4.2) were used in the experiments, treating it as zero-offset data. Because all the algorithms gave the similar imaging results, only the results of the FFD algorithm are shown. Figure 4.4 is the FFD migration result using the exact velocity model (shown in Figure 4.1). Although this zero-offset migration on the shortest-offset data presents all the major features, the imaging result is noisy and inaccurate. Most of the reflectors are not imaged at their exact position. Even though the migration is done on synthetic data, the imaging result is not free from noise. This noise is mainly the numerical noise in the synthetic data inherited from the modeling process; such noise cannot be totally removed by the bandpass filter applied to the model data.

The poor image of zero-offset migration shows the misfit between the migration algorithm and the shortest-offset data. The poststack migration is based on the exploding-reflector model, which requires the input data to be zero-offset data. Although the shortest-offset data have been used in this migration, the 200-m offset made enough difference between the required zero-offset data and the nonzero-offset data used in migration to result in a distorted image. We thus have to use prestack depth migration algorithms to obtain a proper image of the complex Marmousi model.

4.3 Prestack migration with the exact velocity model

In the prestack migration of Marmousi data, a 25-Hz Ricker wavelet was used as the source function for the downward continuation of the source wavefield. To compensate the 40-ms time shift of the peak in the wavelet of the supplied Marmousi synthetic data from zero time, I have shifted the source function the same amount time.

The computations made use of the parallel version of the prestack migration codes implemented in the PVM environment (Han, 1998b). Using 19 Pentium PCs, the total processing time for the Marmousi data was about 25 minutes for the SSF method (the fastest of the methods).

The relative computation times for various algorithms are close to those predicted from the computation counts discussed in Chapter 2, and the relative efficiency of those algorithms is similar to that for the results in Chapter 3. Because an average of 4.3 reference velocities at each depth step was used for the PSPI migration, the

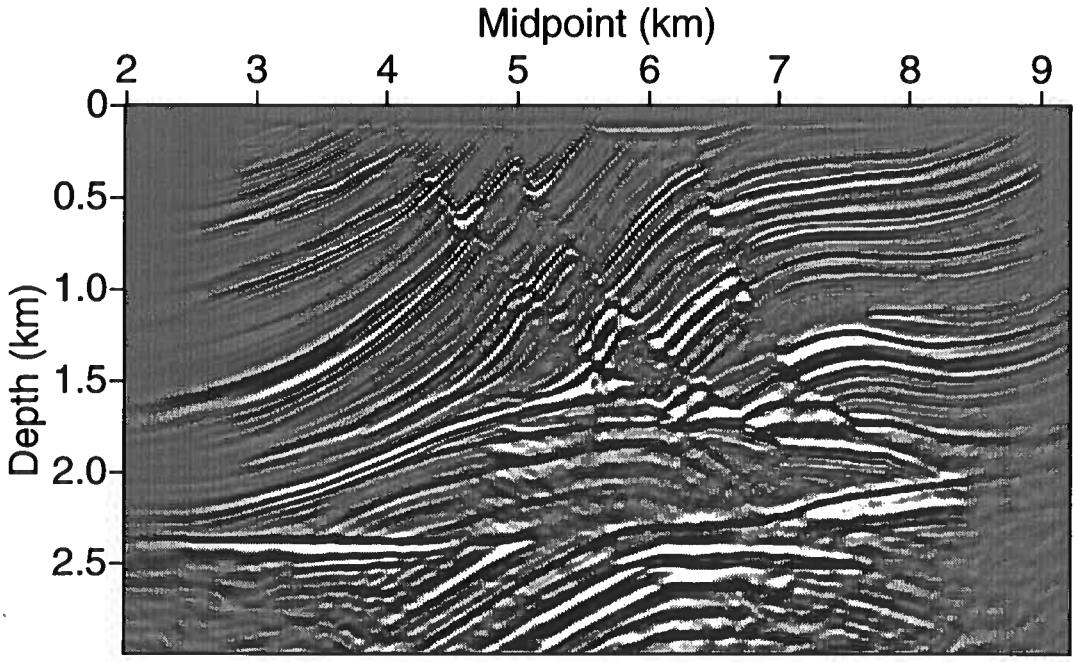

FIG. 4.5. SSF migration.

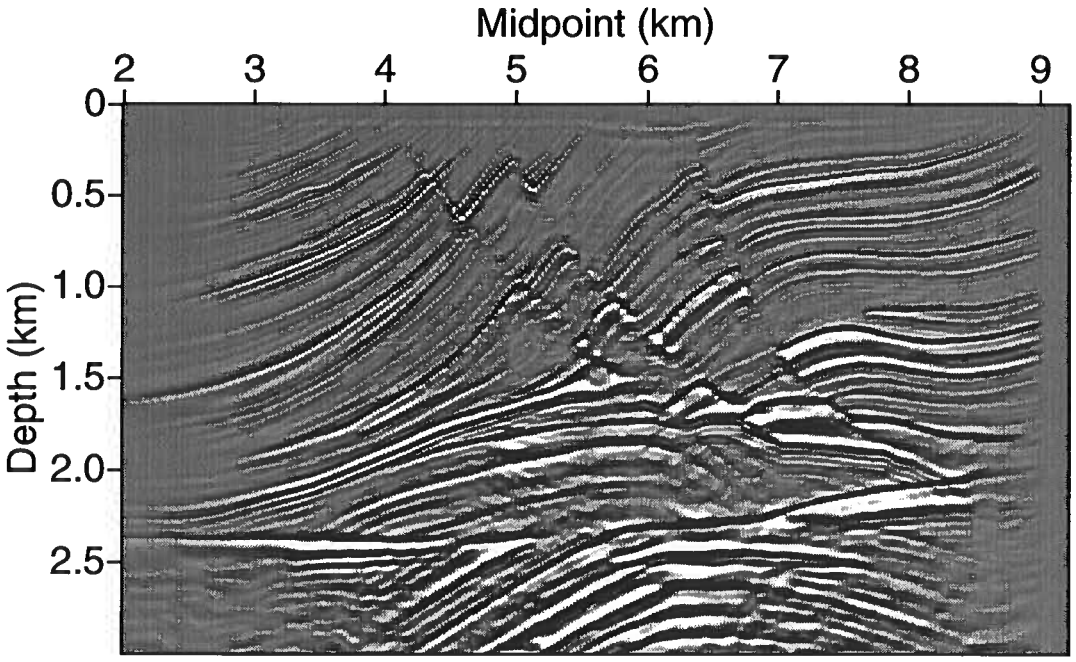

FIG. 4.6. PSPI migration with automatically picked reference velocities. An average of 4.3 reference velocities per depth step was used in the migration.

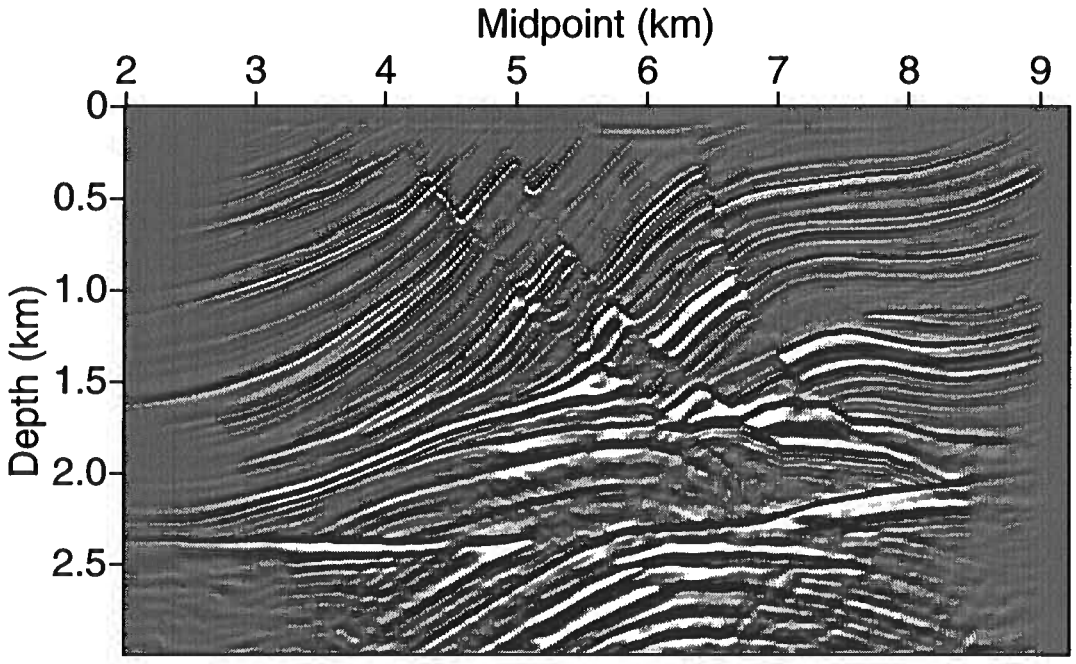

FIG. 4.7. PSPI migration with automatically picked and then manually refined reference velocities.

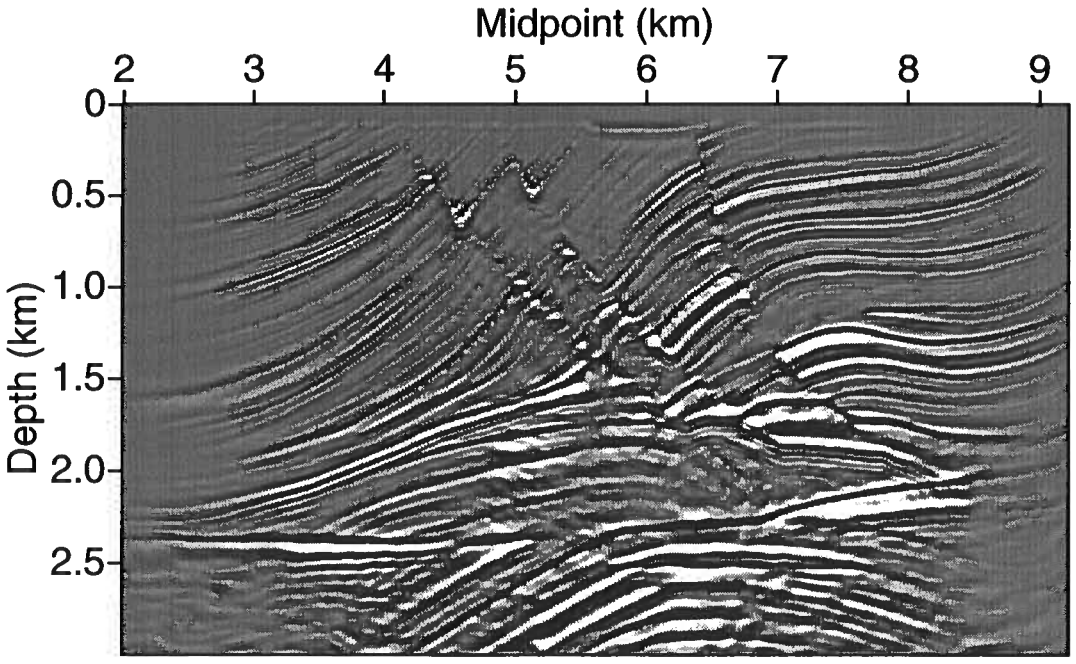

FIG. 4.8. 45^0 FD migration.

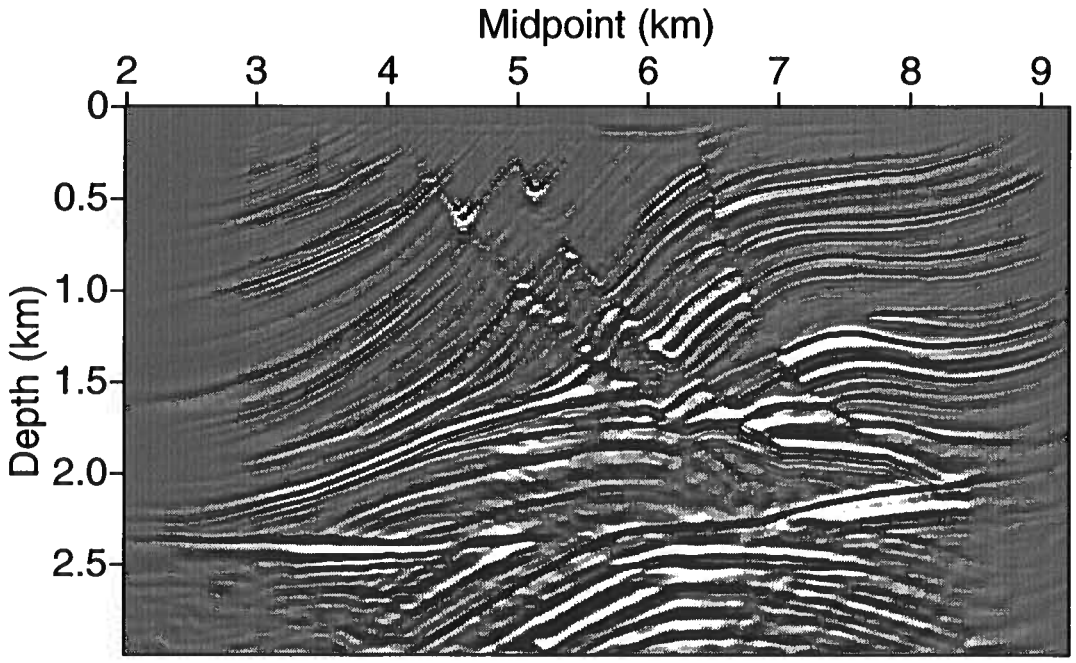

FIG. 4.9. 65^0 FD migration.

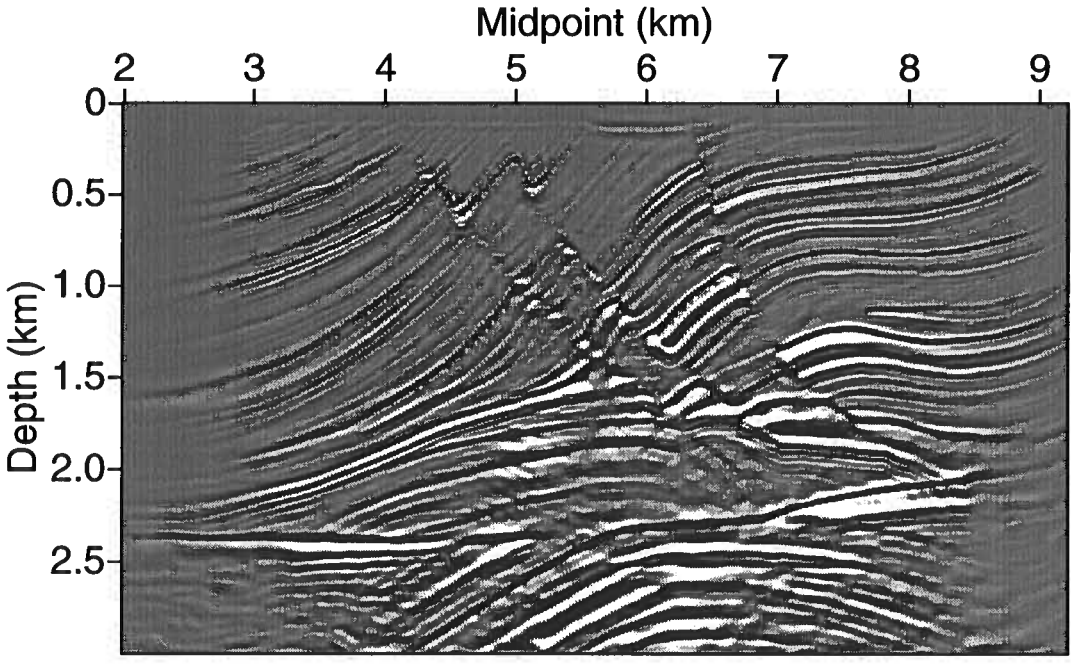

FIG. 4.10. 80^0 FD migration.

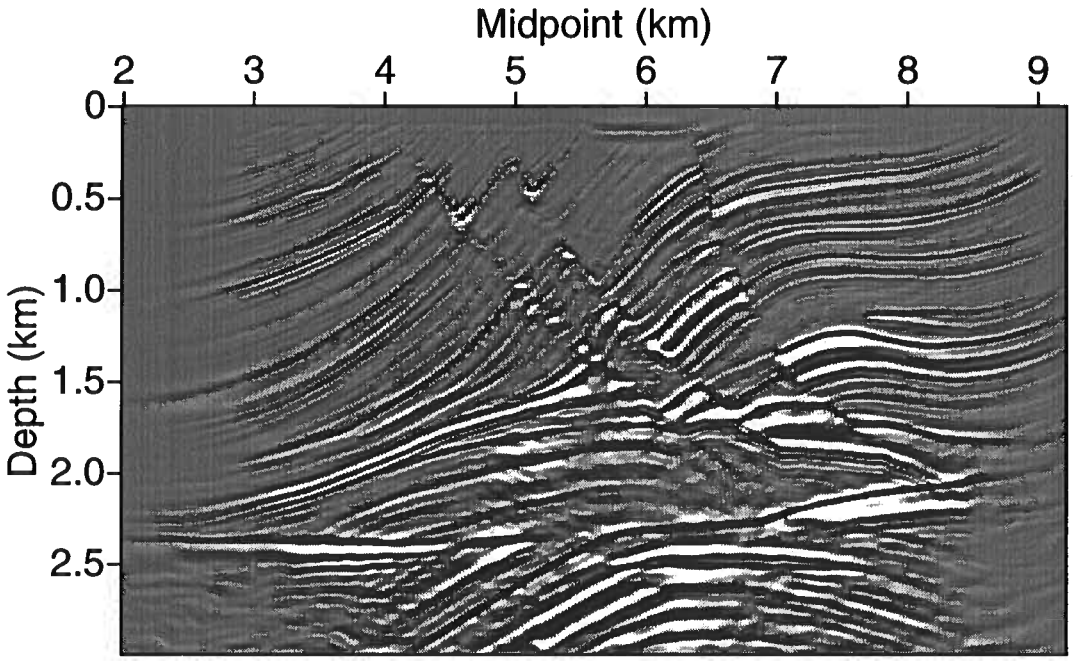

FIG. 4.11. 90^0 FD migration.

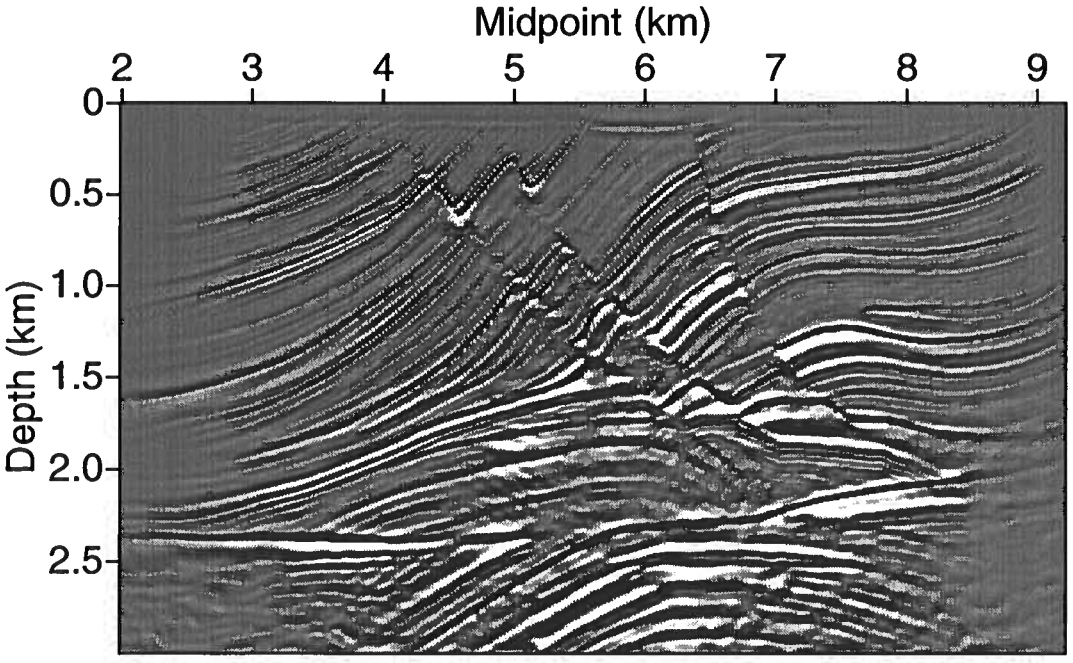

FIG. 4.12. FFD migration.

PSPI method is about four times more costly than SSF. The cost for FFD is slight less than twice that of the SSF, and the cost of FD65 is between that of SSF and FFD. The cost of FD80 is about 20% higher than that of FFD.

As seen in the migrated depth sections shown in Figure 4.5 through 4.12, all four algorithms provided good imaging of the Marmousi model. Specifically, all of them did well in imaging the anticline in the deeper section. The prestack migration results are greatly superior to that of zero-offset migration on the shortest-offset Marmousi model data.

The excellent result (Figure 4.5) for the SSF method may seem a little surprising because, in Chapter 3, the zero-offset result with the SSF method was not as good as those for the other three algorithms. Here, for a highly complicated model, the SSF method generated a result that is comparable in quality to that of the other algorithms. The explanation is as follows. In prestack shot-gather migration, each of the shot gathers has limited illumination area. Even though the overall geological structure might be complicated, the local region illuminated by such a shot gather has relatively limited velocity contrast. Therefore, a different velocity can be used for each shot gather. Consequently, the SSF method does not face problem of the extreme lateral velocity variation that it does in zero-offset migration. For the same reason, we can infer that SSF will not work well if we use it to do prestack *common-offset* migration because the velocity variation across the entire common-offset section is large. Thus, SSF will not be a good choice for common-offset migration of the Marmousi data.

The PSPI result shown in Figure 4.6 was obtained using the automatic picking of reference velocities based on the statistical methods proposed by (Bagaini *et al.*, 1995). The geological structure is well restored with this PSPI result; however, amplitudes are somewhat weak for the shallow portion. I manually refined the reference velocities to make sure velocities in some small but important areas has been included in the reference velocities. The amplitude (especially Region E) has been improved in the revised PSPI result (Figure 4.7). Thus, the amplitude behavior of PSPI seems related to the choice of reference velocities, but the interpolation of the wavefield in PSPI is another possible source of problems. A full investigation of amplitude treatment by wavefield-extrapolation algorithms (not done here) is needed to understand the problem.

The different-order FD algorithms yield similar results in Figures 4.8 through 4.11. Among them, the computational effort for 80° FD algorithm is twice of that for 65° algorithm, and that for the 90° algorithm is about four time that of the 65° algorithm. Although the faulted area has steep faults larger than 65°, the reflections from those faults are weak and do not contribute significantly to the final image. Thus, for the

Marmousi model, the higher-order FD algorithm cannot show its advantage over the 45° and 65° algorithms.

The FFD algorithm yields a result similar to that of the FD algorithms. The advantage of FFD is that it can handle strong velocity contrast, while attenuating inhomogeneous waves (using the same smoothing filter as in PSPI) better than do the FD algorithms. Here, in the Marmousi example, however, FFD did not show much advantage because the low-cost SSF and 65° FD methods obtained similar results with only half the cost of the FFD algorithm. Therefore, FFD has no advantage here.

The marked regions in the pseudo-reflectivity section (Figure 4.3), such as regions C, D and F, show some differences (such as in the continuity of the reflectors and in amplitude) among the various migration results. Those differences can be reduced by tweaking the special parameters for each of the migration algorithms. However, in practice with field data, without knowledge of the exact model in mind, it would be difficult to know which result is closer to representing the truth.

The similarity in migration results for all the algorithms suggest two things. First, all the algorithms can handle a model as complex as the Marmousi model. The Marmousi model is a good example to show the inability of Kirchhoff migration to cope with complex situations such as multi-pathing problems (Zhu & Lines, 1998; Bevc, 1997). In contrast, for wavefield-extrapolation methods when the exact velocity model is known, migration of moderately steep features in complex structures (without huge velocity contrast) is not a challenge. As shown in Chapter 3, however, migration with sharp velocity contrast poses a big challenge for these algorithms.

Second, the Marmousi model is not a good example to show the difference in these algorithms. Although this model is complicated, it lacks sharp velocity contrast in areas covered by different shot records; thus we cannot see differences in migration results such as in those for the SEG-EAGE model. In the upper portion of the Marmousi model are steep faults. However, reflections from those faults are so weak that they hardly contribute to the final images. We mainly identify those faults through the less steep layers cut through by those faults. Therefore, the Marmousi model is not suitable to demonstrate the ability of algorithms to handle steep features.

Chapter 5

CONCLUSIONS AND FUTURE WORK

Of the four one-way depth-migration methods tested on the two model data here, SSF was the fastest. Although in the example of the zero-offset migration on SEG-EAGE model with extreme lateral variation, the SSF method was distinctly less accurate than other methods, its performance on the Marmousi model is comparable to that of other three methods. This good performance for prestack migration is understandable because, in shot-gather migration, different reference velocities can be chosen for each gather at no additional cost since each shot gather has limited illumination area; thus prestack SSF has less velocity error in the presence of large lateral velocity variation than it would for zero-offset (or constant-offset) SSF. Thus, although a good candidate for shot-record migration, the SSF method will not be a good candidate for prestack common-offset migration.

The other three methods gave comparable results for the migration of the model data. That is because they are derived from the same dispersion relationship and are able to handle strong lateral variation in velocity. Their differences in the migration results are mainly a quality-control issue that can be solved by adjusting the specific parameters in each algorithm. Given the comparable accuracy in migration results, choosing a proper migration method out of those three will largely depend on the specific requirements of the target project.

The PSPI method is the most expensive of the methods because accurate results require that the increment in the reference velocities are small enough to accommodate the full range of lateral velocity variation encountered for any given velocity model. This method often requires three to four reference velocities per depth step as compared to only one reference velocity in the SSF method. On the other hand, the PSPI algorithm can handle steep dips, up to 90°. Thus, for imaging targets that contain steep features in complex structure, the PSPI method will be a good choice.

The FD algorithm uses a local approximation to the dispersion relationship in the space domain. Such a feature enables it to handle strong velocity contrasts. The FD algorithm is dip-limited, however, due to the truncation error in approximating the dispersion relationship. The dip-limitation can be removed by using more cascaded

diffraction terms at several times of additional computation costs. In general, the FD algorithm is preferable when the imaging target has strong velocity variation, but moderate dips.

The FFD algorithm was designed as an extension for the SSF method, wherein a FD operator is added to the SSF method to enhance the ability to handle strong lateral velocity variation. Thus FFD overcomes shortcomings in the SSF method due to the small-perturbation assumption. Adding the FD operator, however, makes FFD twice as expensive as the SSF method. In general, the cost and accuracy of the FFD method are comparable to those of the 80° FD algorithm.

Except for the FD approach, all approaches are dual-domain algorithms that shuttle between the frequency-wavenumber and frequency-space domain. One benefit of this shuttling between domains is that it is convenient to design filters in the wavenumber domain to attenuate post-critical inhomogeneous waves. As pointed out by Li (1991) and Graves & Clayton (1989), these dual-domain methods consequently have more flexibility than the FD algorithm in eliminating numerical noise caused by the inhomogeneous wavefield, thus yielding images with less numerical noise. In the FD algorithm, an implicit dip filter (actually a wavenumber filter) is used to suppress the inhomogeneous waves. The advantage of this filter is that it can be implemented at no additional computation cost. However, trying to attenuate inhomogeneous waves will inevitably attenuate some homogeneous waves as well. One suggestion will be to implement a wavenumber domain filter for the FD algorithm as well, but with 100% and 50% increase in computation cost for the 65° and 80° FD algorithms, respectively. For such a case, the FFD method is preferable because its cost is close to that of the 80° FD algorithm and already has the wavenumber-domain filter built into it.

Besides comparing these four algorithms, I addressed the problem of missing steep subsalt events encountered for all the algorithms. Using Snell's law, we find that the wavefield related to steep subsalt features is inhomogeneous within the salt and thus would not be recorded. Consequently, no reflections from the steep reflectors are available for imaging them, no matter how large the aperture we used in modeling and migration.

In this thesis, I focus the comparison of these algorithms on their performances in 2D migration. We can expect the relative accuracy seen here to hold for 3D data as well. The extension from 2D to 3D is straightforward for PSPI and SSF, involving only an additional Fourier transform in the crossline direction. It is more difficult to take FD and FFD to the 3D domain because of the numerical anisotropy caused by splitting the FD operator in the inline and crossline directions. To compensate for such numerical noise, multi-direction splitting (Ristow & Rühl, 1995) may be required, but at increased computation cost for FD and FFD. It seems, therefore,

that 3D imaging will give more preference to the PSPI and SSF methods.

Because the Earth's subsurface is generally anisotropic, data processing techniques that take anisotropy into account are required when the magnitude of anisotropy is so large that it cannot be ignored. Also, with ocean bottom seismic (OBS) data, in some cases converted-wave sections may be superior to conventional P-wave data. All four algorithms can be readily modified to handle both the anisotropy and mode conversion because of their similar and related algorithm structures (i.e., the common platform used here.).

In Han (2000), I present a implementation of two converted-wave migration algorithms in transversely isotropic media with a vertical symmetry axis (VTI). Derived from the isotropic phase-shift-plus-interpolation (PSPI) and implicit finite-difference (FD) methods, these two algorithms inherit the accuracy of those wavefield-extrapolation migration methods. Based on an analytic solution of the Christoffel equation, the anisotropic PSPI algorithm can handle arbitrary magnitude of anisotropic parameters. The anisotropic FD algorithm, however, employs the weak-anisotropy assumption and thus is less accurate when anisotropy is strong. On the other hand, the anisotropic PSPI algorithm lacks the flexibility to deal with rapid spatial variation in the Thomsen parameters ϵ and δ (Thomsen, 1986) unless the reference wavefield is computed for each set of parameters. The anisotropic FD algorithm, however, can handle rapid variations of Thomsen parameters because each of the localized finite-difference operators is a function of the local parameters.

Another area that needs to be explored is how to use these algorithms effectively in iterative migration velocity analysis (MVA). So far, most MVA techniques use the Kirchhoff method because the iterative analysis requires that the migration algorithm be fast. Can we develop a new set of MVA techniques that is designed to take advantage of the accuracy of these algorithms and, at the same time, run with acceptable computation costs?

REFERENCES

Bagaini, C., Bonomi, E., & Pieroni, E. 1995. Data parallel implementation of 3D PSPI. *65th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 188–191.

Baysal, E., Kosloff, D. D., & Sherwood, J. W. C. 1983. Reverse time migration. *Geophysics*, **48**, 1514–1524.

Bevc, D. 1997. Imaging complex structures with semirecursive Kirchhoff migration. *Geophysics*, **62**(2), 577–588.

Cerjan, C. C., Kosloff, D., Kosloff, R., & Reshef, M. 1985. A nonreflecting boundary condition for discrete acoustic and elastic wave equations. *Geophysics*, **50**(4), 705.

Claerbout, J. F. 1985. *Imaging the Earth's interior.* Oxford, England: Blackwell Science Publishers.

De Hoop, M. V. 1998. Asymptotic inversion: multi pathing and caustics. *68th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 1534–1537.

Gazdag, J. 1978. Wave equation migration with the phase shift method. *Geophysics*, **43**, 1342–1351.

Gazdag, J., & Sguazzero, P. 1984. Migration of seismic data by phase shift plus interpolation. *Geophysics*, **49**(2), 124–131.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., & Sunderam, V. 1994. *PVM: Parallel Virtual Machines, A users guide and tutorial for networked parallel computing.* Cambridge, Massachusetts: MIT Press.

Geist, G. A., Kohl, J. A., & Papadopoulos, P. M. 1996. PVM and MPI: a comparison of features. *http://www.netlib.org/pvm.*

Gouveia, W. 1996. SUDREF: a distributed elastic reflectivity modeling algorithm. *66th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 986–989.

Graves, R. W., & Clayton, R. W. 1989. Modeling acoustic waves with paraxial extrapolators. *Geophysics*, **55**(3), 306–319.

Han, B. 1998a. A comparison of four depth-migration methods. *68th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 1104–1107.

Han, B. 1998b. Parallel implementation of four depth-migration algorithms in a heterogeneous network. *CWP Project Review, Center for Wave Phenomena, Colorado School of Mines, CWP-279.*

Han, B. 2000. Two prestack converted-wave migration algorithms for vertical transverse isotropy. *CWP Project Review, Center for Wave Phenomena, Colorado School of Mines, CWP-344.*

Han, B., Galikeev, T., Grechka, V., Le Rousseau, J., & Tsvankin, I. 1998. A synthetic example of anisotropic P-wave processing for a model from the Gulf of Mexico. *CWP Project Review, Center for Wave Phenomena, Colorado School of Mines, CWP-275.*

Hill, N. R. 1990. Gaussian beam migration. *Geophysics*, **55**(11), 1416–1428.

Huang, L., & Fehler, M. C. 1997. Extended pseudo-screen migration with multiple reference velocities. *67th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 1750–1753.

Jenner, E., de Hoop, M., Larner, K., & Van Stralen, M. 1997. Imaging using optimal rational approximation to the paraxial wave equation. *67th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 1750–1753.

Kessinger, W. 1992. Extended split-step Fourier migration. *62nd Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 917–920.

Le Rousseau, J. H., & De Hoop, M. V. 1998. Modeling and imaging with the generalized screen algorithms. *68th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts.*

Lee, M., Mason, L. M., & Jackson, G. M. 1991. Split-step Fourier shot-record migration with deconvolution imaging. *Geophysics*, **56**(11), 1786–1793.

Lee, M. W., & Suh, S. Y. 1985. Optimization of one-way wave equation. *Geophysics*, **50**(10), 1634–1637.

Li, Z. 1991. Compensating finite-difference errors in 3-D migration and modeling. *Geophysics*, **56**(10), 1650–1660.

Liu, Z. 1995. Migration velocity analysis. *Ph.D thesis, Center for Wave Phenomena, Colorado School of Mines, CWP-168.*

Murillo, A. E. 1996. DSU: Distributed parallel processing with Seismic Unix. *66th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 997–1000.

Murillo, A. E., & Uzcategui, O. 1997. Three strategies for parallelizing a 3D seismic migration algorithm on distributed memory environments. *CWP Project Review, Center for Wave Phenomena, Colorado School of Mines, CWP-259.*

Nichols, D.E. 1996. Maximum energy traveltimes calculated in the seismic frequency band. *Geophysics*, **61**(1), 253–263.

Popovici, A. M. 1992. Phase shift plus interpolation and split-step Fourier migration. *SEP-72, SEP report, Stanford University.*

Ristow, D., & Rühl, T. 1994. Fourier finite-difference migration. *Geophysics*, **59**(12), 1882–1893.

Ristow, D., & Rühl, T. 1995. 3-D implicit finite-difference migration by multiway-splitting. *65th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 1220–1223.

Roberts, P., Huang, L., Burch, C., Fehler, C., & Hildebrand, S. 1997. Prestack depth migration for complex 2D structure using phase-screen propagators. *67th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 1282–1285.

Rühl, T., & Ristow, D. 1995. Fourier FD migration : The missing link between phase-shift and FD migration. *65th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 1232–1235.

Salinas, T. 1996. The influence of near-surface time anomalies in the imaging process. *Ph.D thesis, Center for Wave Phenomena, Colorado School of Mines, CWP-237.*

Schneider, W. A. 1978. Integral formulation for migration in two and three dimensions. *Geophysics*, **43**, 49–76.

Stoffa, P. L., Fokkema, J. T., Freire, R. M., & Kessinger, W. P. 1990. Split-step Fourier migration. *Geophysics*, **55**, 410–421.

Tanis, M. C., & Stoffa, P. L. 1997. Parallel implementation of 3-D Split-step Fourier depth migration algorithm on T3E. *67th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 1433–1436.

Thomsen, L. 1986. Weak elastic anisotropy. *Geophysics*, **51**(10), 1954–1966.

Tjan, T. 1995. Residual statics estimation for data from structurally complex areas using prestack depth migration. *Master's thesis, Center for Wave Phenomena, Colorado School of Mines, CWP-186.*

Versteeg, R., & Grau, G. 1990. The Marmousi Experience: Proceedings of the 1990 EAEG workshop on Practical Aspects of Seismic Data Inversion. *52nd EAEG Meeting. Eur. Asoc. Expl. Geophys.*

Wu, R., & Jin, S. 1997. Windowed GSP(Generalized Screen Propagators) migration applied to SEG-EAEG salt model data. *67th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 1746–1749.

Xu, S., Chauris, H., Lambare, G., & Nobel, M. 1998. Common angle image gather: a strategy for imaging complex media. *68th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts*, 1538–1541.

Zhu, J., & Lines, L. 1998. Comparison of Kirchhoff and reverse-time migration methods with applications to prestack depth imaging of complex structures. *Geophysics*, **63**(4), 1166–1176.

Appendix A

A PARALLELIZED COMMON PLATFORM

A.1 Introduction

Migration algorithms that work in the frequency-space and frequency-wavenumber domains parallelize naturally because frequency components are independent of each other, and hence can be processed separately. Here, I discuss the parallel implementation of four different depth-migration methods: phase-shift-plus-interpolation (PSPI), split-step Fourier (SSF), implicit ω-x finite-difference (FD), and Fourier finite-difference (FFD). Because of their similarities, all four methods are built from a common software platform. Therefore, these implementations required little effort to convert from previous sequential migration code to the parallel code. Most of the initial converting work for all four methods was done within a weekend, though debugging of these codes took another few days.

The PVM (parallel virtual machine) library (Geist *et al.*, 1994), as opposed to MPI (message passing interface) specification, is the communication library I used, the main reason for this choice being that the PVM library has more flexibility and power for computing within a heterogeneous network than does MPI, although some efficiency is sacrificed to achieve such characteristics. Murillo and Uzcategui (1997) suggested a general strategy for parallelizing of migration algorithms, and the PVM implementation here benefited from suggestions in Gouveia (1996) and Murillo (1996).

The motivation of this research is the need to compare the relative performance of different algorithms for migration applied to large volumes of 2-D poststack and prestack data. This parallel implementation turned out to be highly rewarding in that the complex Marmousi data (Versteeg & Grau, 1990), for example, can be migrated at CWP within half an hour compared to the nine hours required by the sequential codes. Benchmark tests performed on both shared and distributed computer architectures showed an almost linear relationship between the number of processors and speedup factors.

Though four algorithms are now parallelized, a pair of master-slave codes developed

to accommodate the different algorithms into a common structure allows the ready generation of new migration code; only the downward-propagation part needs to be modified, facilitating the coding work for other algorithms.

A.2 Heterogeneous computing environment

When I did the experiments, the Center for Wave Phenomena (CWP) at Colorado School of Mines had about thirty workstations connected via an Ethernet network, forming the following heterogeneous computing environment.

1. different hardware architectures

   Among the thirty computers, 15 are PC workstations with Intel Pentium CPUs, their clock frequency ranging from 90 Hz to 120 Hz. Nine of them are PC workstations with Intel Pentium Pro CPUs having a clock frequency of 200 Hz, and we have a six-processor SGI Power Challenge; the clock frequency of the MIPS-R8000 processor is at 75 Hz.

2. different software environment

   All the PC workstations were installed with the Linux operating system using free GNU development tools from FSF (Free Software Foundation). Although most of them have the latest Linux operating system, Slakware 3.4 with 2.0.30 kernel, some of them are still working with Slakware 3.2 (2.0.27 kernel) or 3.1 (2.0.0 kernel). The reason that we kept some of the old operating system is for the compatibility with older software that can run on only the old Linux system. As to the SGI Power Challenge, its operating system is IRIX 6.1.

3. different network bandwidth

   Most of the workstations are connected via the slow Ethernet network, with peak bandwidth at 10 Mbyte/s. Eight workstations are connected with a fast Ethernet network; it has a peak bandwidth of 100 Mbyte/s. When I test my implementation, I usually run a parallel job using 20 workstations, so my only choice is to cope with the 10 Mbyte/s slow network, which becomes a bottleneck for communication-intensive jobs.

A.3 Implementation considerations

A.3.1 PVM versus MPI

To convert the sequential code to parallel code, I needed a communication library to adopt the master-slave topology. Two good choices are PVM (parallel virtual machine) and MPI (message passing interface) specification. Both of them are freely available from the web site of Netlib (http://www.netlib.org), with good documentation and some sample code. In my implementation, I chose PVM as the communication library based on a fair comparison of the above two systems by Geist et al. (1996).

Though generally less efficient than the MPI specification, PVM was designed at the outset to work for a heterogeneous-network computing environment. It has better portability and inter-operability performance than any other existing communication library. Specifically, the code written in PVM can be compiled and run on any hardware architecture and operating system without changing the source code; PVM jobs on different hardware architecture can communicate and exchange data with one another; PVM also supports both C and F77. Moreover, C and Fortran code can exchange data between them even though arrays in C and Fortran have opposite storage sequences; for example, one can code a master program in C and a slave program in Fortran without any ill-effect. As to the MPI specification, though the source code written in MPI can also be ported among different architectures without any modification, it lacks the inter-operability feature: jobs on different hardware platforms cannot exchange data with one another, and codes in C and Fortran cannot communicate with each other. The high efficiency of MPI can be achieved only in a homogeneous, distributed environment, thus sacrificing the flexibility that is necessary for heterogeneous network computing.

Furthermore, PVM contains all the necessary requirements of process control. It can start tasks, keep track of the running processes and stop them. In contrast, in the widely available MPI-1 standard, no function has been defined to start a parallel task within a program. Although MPI-2 will accommodate some of the process-control specification, no public-domain MPI-2 implementation has been available to date.

Though all the source codes I have implemented are written in PVM, considering the similarity in the communication functions between PVM and MPI, if desired, they could be converted to utilize the MPI functions without any difficulty.

A.3.2 A common platform for different algorithms

All four algorithms were derived from the one-way acoustic wave equation based on downward-continuation theory, and all work in the frequency-space domain. These algorithms are naturally parallelized; that is, all the frequency components are independent of one another, and hence can be processed separately.

In my implementation, I have adapted a master-slave topology. The master program distributes the frequency components to different slave tasks, and collects and stacks the final image after sending out all the data. Since the only differences among these migration algorithms are in the downward-continuation propagator, I code a common platform for all of them to make the code reusable, consequently saving implementation effort and facilitating future development of other migration algorithms. The master code is identical for all the migration algorithms. In fact, at the same time one can run a master program to communicate to the slave tasks with different downward-continuation propagators. The communication part of the slave code is also the same among the different algorithms; only the core part — the propagator — differs. Changing a parallel code from algorithm A to B can be easily achieved by substituting the new propagator for the old one.

A.3.3 Automatic load balancing

In migration, the total computational cost is determined by the size of the migration aperture, the frequency range of the input data, the grid spacing, the number of depth steps and the migration algorithm itself. Once the above parameters have been fixed, the total computing cost will be static and unchanged for a given hardware configuration. As opposed to the static computing expense, the available computing resources in a heterogeneous computing environment can and do change dynamically. Besides the difference in the speed of the processor (for example, the Intel Pentium Pro 200 Hz processor is about twice as fast as the Pentium 90 Hz processor in floating-point computation.), the load of different workstations varies dynamically. Also, typically a parallel job needs to compete for the available resources with those of other users of the parallel network . In such situations, it is necessary to incorporate dynamic load balancing to take care of differences in computer speed and process load.

In parallelizing the four depth-migration algorithms, dynamic load balancing can be easily and readily put into the codes. Instead of dividing the entire data just by the available number of processors N, I multiply N by a data-dependent factor a (in my codes, it is usually set to 3 or 5), then partition the data using the new number aN.

Smaller-size data sets are sent to different slave processors, and, once processing is done for any one of the data set, a signal is sent back to the master process to request more data. In this way, with faster processors processing more data than slower ones, the data load among different workstations will be balanced. To reduce the network loading, all the intermediate migrated images are kept in the slave processors instead of sending them back to the master program. When all the frequency components have been processed, the master program will collect the partial images from the slave processors and stack them together to get the final image. Hence, the parallel implementation here will be generally computation-, as opposed to communication-, intensive, as is needed in a low-bandwidth network.

A.4 Benchmarking comparison

The performance measurements, done here on both shared and distributed computer structures, involve prestack migration on the Marmousi data (Versteeg & Grau, 1990).

A.4.1 Description of testing model and the measuring procedure.

The migrated data were constructed with 60 frequency components uniformly incremented between 15 and 35 Hz.

To test the performance of the parallel codes, instead of migrating the entire data set I migrated only 15 shot gathers and saved the computing time for each shot gather to a file. After migration, the latter ten of the computing times were averaged to get a mean value. A speedup factor was computed by dividing the sequential time for a shot gather by the averaged parallel time. To test the relationship between the total computing time and the data-block size sent each time to the processors, I ran tests with, respectively, 1, 3, and 5 frequency components sent each time from the master process to the slave tasks.

The above tests have been repeated for all four algorithms with similar speedup results, showing that these implementations are independent of the algorithms themselves. Below, is a discussion of the result for the 65^0 FD algorithm, as an example.

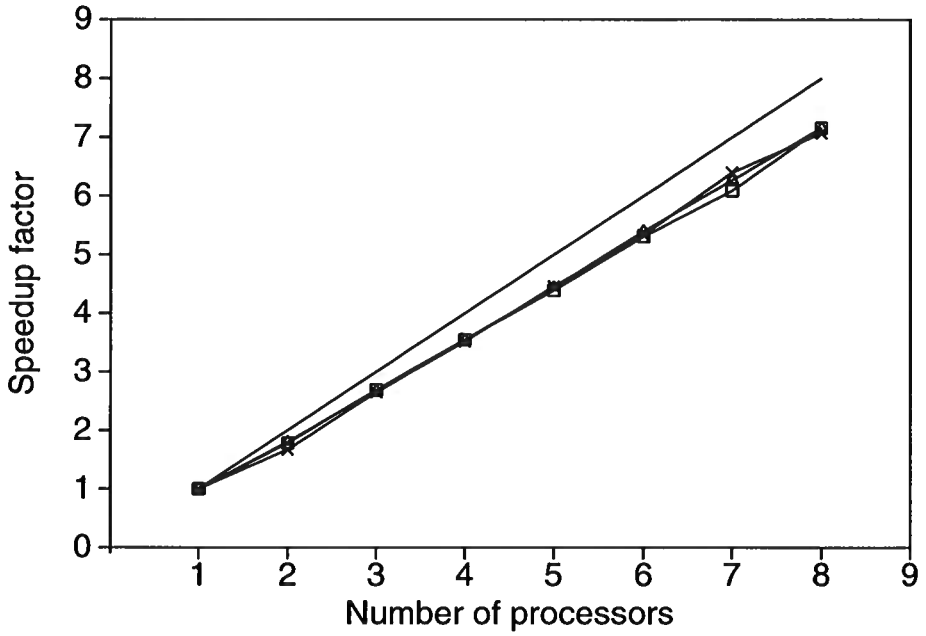

FIG. A.1. Speedup results for the 65^0 FD algorithm using an eight Pentium Pro based network; the straight line without marks is the ideal linear increase; the marked lines are for three different tests, in which different block-data sizes are sent each time from the master to slave tasks.

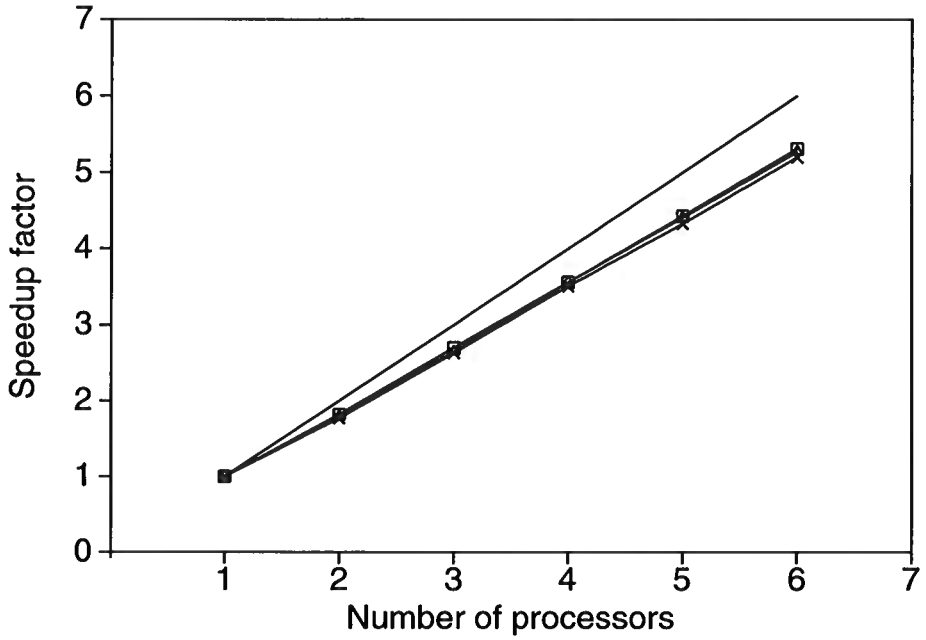

FIG. A.2. Speedup results for the 65^0 FD algorithm using a six-processor SGI Power Challenge; the straight line without marks is the ideal linear increase.

A.4.2 Performance comparison for the PC and SGI Power Challenge

To see what speedup we can get from these codes, the performance tests were done on two homogeneous platforms: a PC-based network with 8 Pentium Pro workstations and the 6-processor SGI Power Challenge.

The speedups achieved on the two platforms are shown in Figures A.1 and A.2. The speedup factors increase linearly on both testing platforms, although they still could not catch up with the ideal speedup factors. The difference between the ideal expectation and the test results is due to the time spent on communication through the network.

The communication time takes only a small portion of the total computing time, so the total computing time is almost inversely proportional to the number of processors, resulting in the somewhat linear increase of speedup factors in both figures. Though I tried different data sizes sent each time from the master to slaves, their results almost coincide with one another, again because these implementations are computation-intensive rather than communication-intensive.

In addition to the speedup tests here, I also performed load balancing tests for a

heterogeneous network that include five different workstations: Pentium 90, Pentium 120, Pentium Pro 200, SGI Power Challenge, and SGI Indigo 2. For the codes without load balancing, each slave workstation works on the same amount of data. Using the SSF algorithm, for migration of a single shot gather of the Marmousi data, the code without loading balancing took about 68 as compared to the 39 seconds achieved by the code with load balancing. Thus without load balancing, more than 70% of additional time was wasted in waiting for the slower processors to finish their jobs.

A.5 Conclusions

The parallel codes appear to be stable (I ran PVM jobs on 20 workstations for 9 hours with no problem) and give satisfactory speedup factors. Using 60 frequency components (15 – 35 Hz) in a 16-PC heterogeneous network, the migration of the multi-offset Marmousi data took from half an hour for the 65^0 FD algorithm to two hours for the PSPI algorithm — the most expensive of the four approaches considered here.