CWP-177
April 1995

2.5-D Common Offset Inversion in Triangulated Background Models of the Earth

Boyi Ou

— Master's Thesis —
Mathematical and Computer Sciences

Center for Wave Phenomena
Colorado School of Mines
Golden, Colorado 80401
303/273-3557

ABSTRACT

Two-and-one-half dimensional inversion of reflection seismic data developed by Bleistein et al. (1987) provides a powerful tool for obtaining correct locations of interfaces while preserving a model-consistent amplitude. Here, I describe the use of ray tracing in a triangulated model of the Earth to implement 2.5-D Kirchhoff inversion. The objective of the inversion is to obtain correct locations of the interfaces and return a true amplitude (angularly dependent reflection coefficient) consistent with the underlying acoustic theory of wave propagation.

In 2.5-D common-offset inversion, a ray tracer for triangulated media is used to calculate the traveltime and other necessary constituents of the ray theory for a background model. In the background models, velocity is replaced by sloth (inverse velocity squared). The medium is assumed to consist of constant-gradient sloth in triangulated regions that characterize layers separated by arbitrary, curved interfaces. Also transmission losses through interfaces are taken into account. To improve cpu time, I generate the ray data for a relatively sparse set of upper-surface grid points and interpolate the data to obtain values at intermediate grid points. This interpolator produces good reflector maps with acceptable cpu time. This 2.5-D inversion improves on the previous code which required a background model in which the propagation velocity was piecewise-constant and the reflectors were required to extend across the entire model.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Norman Bleistein, for his terrific help, guidance and support during my time at the Center for Wave Phenomena. His enthusiasm for sciences is surpassed only by his knowledge and insight. I am indebted to Dr. Ken Larner and Dr. Erik Van Vleck for serving on my committee. I would also like to thank Zhenyue Liu and Andreas Rüger for their assistance in this research effort.

Chapter 1

INTRODUCTION

The purpose of this project is to update CWP's currently available 2.5-D Kirchhoff inversion code CXZCO which was developed by Hsu and Liu (1991) to implement the 2.5-D inversion theory of Bleistein, et al. (1987). CXZCO has some inherent limitations movitated in part by the design of two-point ray tracing (Docherty, 1988). First, the background model in CXZCO must consist of constant-velocity layers. Second, interfaces between the layers are required to cross the entire model. This limits the complexity of background models that this program can handle. Here, I describe a program, developed by me, that allows more general background models.

This code is based on the Kirchhoff integral inversion method for common-offset seismic data (Bleistein et al., 1987). This method treats amplitude in inversion in a WKBJ-consistent manner so that the output is the reflectivity function. Furthermore, Bleistein showed that, based on the stationary-phase principle, one can design weights in the Kirchhoff integral to determine quantities such as the specular reflection angle. In Bleistein's integral formulation, the integral requires ray data: traveltime, WKBJ amplitude and other ray parameters for a background model. Two commonly used approaches to calculating traveltime and the other quantities in the Kirchhoff integral are ray tracing and finite differences applied to the eikonal equation. Here, I use the former, which allows for multiple arrivals at any point. This is much more difficult for finite difference traveltime solvers.

Hale and Cohen (1991) developed software to generate data bases for two-dimensional computer models of complex geology. Their method uses a triangulation technique designed to support efficient and accurate computation of seismic wavefields for models of the earth's interior. Subsequently, Hale (1991) used this triangulation approach to perform dynamic ray tracing and create synthetic seismograms based on the method of Gaussian beams. Then, Rüger (1993) generalized that code. Based on this earlier work, I have implemented 2.5-D inversion for more complex models. The ray tracer used in this thesis is derived from code developed by Rüger, but I do not use the Gaussian beam option of his code for amplitude calculations.

In my adaptation, it is necessary to calculate ray data at output grid points. Usually, the rays do not pass through these points, so it is necessary to interpolate to obtain ray data at the grid points. Furthermore, the computation cost of traveltime and amplitude for every trajectory from surface grid points to subsurface grid points could dominate the total calculation cost of the Kirchhoff integral method. Thus, to speed up the Kirchhoff integral method, I apply a "parallel interpolation" scheme developed by Liu (1993) after computing ray data for only a limited set of surface grid points.

In this work, the velocity model is replaced by variable sloth layers separated by interfaces that are more complex than those used in CXZCO. Sloth is inverse-velocity-squared. For example, the lens-shaped structure and pinchouts shown in Figure 1.1 can be treated with the new code, but can be only poorly represented with the previous code. As we can see in this figure, interfaces do not need to cross the entire model. Furthermore, the velocity can vary both vertically and horizontally within layers, whereas, CXCZO requires constant velocity within layers. In summary, this thesis synthesizes the 2.5-D inversion theory previously implemented in CXZCO and Rüger's dynamic ray tracing to produce a 2.5-D inversion code that is applicable to a more general Earth model than that of the predecessor.

In Chapter II, I describe an inversion implementation that takes advantage of dynamic ray tracing to obtain geometrical spreading. In Chapter III, ray tracing in triangulated models is described. In Chapter IV, I describe the shooting scheme and transformation from the triangulated system to a uniform grid and parallel interpolation, designed for efficient use of cpu time. In Chapter V, I show examples of inversion of data from a horizontal-layer model, a commonly-studied synthetic model generated for Marathon Oil Company, a lens-shaped model, a salt dome, a physical tank model, and the Marmousi model.

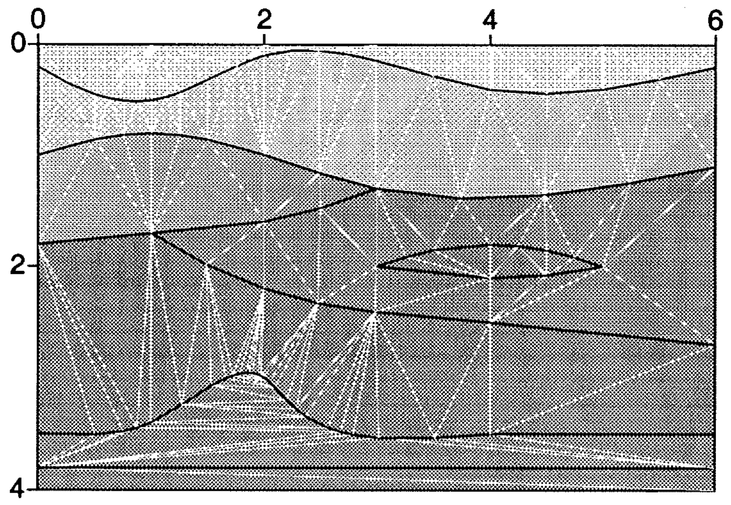

FIG. 1.1. Lens-shaped structure and pinchouts. Lithological interfaces are represented by black lines. The shading denotes the seismic velocity field. The white lines are edges of auxiliary triangles used in the model building.

Chapter 2

ALGORITHM FOR 2.5-D COMMON OFFSET INVERSION

2.1 Introduction

In this chapter, I derive an inversion formula to be programmed for the implementation of 2.5-D common offset inversion. The earlier code (Hsu and Liu, 1991) used the rate of change of ray direction, $\partial\beta(\boldsymbol{x})/\partial\xi$ [$\beta(\boldsymbol{x})$ is the incidence angle between the ray and the upward vertical direction at depth point $\boldsymbol{x}$; ξ is the location of the midpoint between source and receiver], to characterize geometrical spreading, which, in turn is used to caculcate the ray theoretic amplitude along a ray. Here, I take advantage of dynamic ray tracing to obtain the geometrical spreading factor directly. Also, in contrast to CXZCO, where rays are shot upward, here I shoot rays downward. The computation of ray data used here follows the method of Červený (1981).

2.2 Geometrical spreading

Geometrical spreading characterizes the changes in amplitude along the geometrical optics rays. In CXZCO (2.5-D common-offset inversion code in constant-velocity, layered medium), geometrical spreading is approximated by $\partial\beta(\boldsymbol{x})/\partial\xi$. However, in dynamic ray tracing, geometrical spreading is one of the quantities naturally calculated along each ray. I describe here this alternative approach to characterization of geometrical spreading.

First, consider the *Jacobian* of the transformation from Cartesian (x, z) coordinates to ray coordinates, given by

$$J = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \gamma} & \frac{\partial z}{\partial \gamma} \end{vmatrix}, \tag{2.1}$$

where ξ and γ are ray coordinates and, here and below, $|\;|$ denotes determinant of the array. The ray coordinate γ selects one ray from the whole system of rays. The coordinate ξ specifies the position of a point on the selected ray. In this discussion, ξ will be the arclength along the ray measured from an arbitrary reference point. The function J measures the expansion and contraction of the ray tube. When two neighboring rays intersect, the function J vanishes, J=0. Such points are called caustic points. In shadow zones, where rays do not exist, J is not defined. The function J can be simply expressed in "ray-centered" coordinates (ℓ,n). J is particularly easy to compute directly on the ray Ω, where n=0. As seen in Figure 2.1, the coordinate ℓ measures the arclength along the ray Ω from an arbitrary reference point; n represents a length coordinate in the direction perpendicular to Ω at ℓ.

Let us first perform the transformation from (x,z) coordinates to the ray-centered

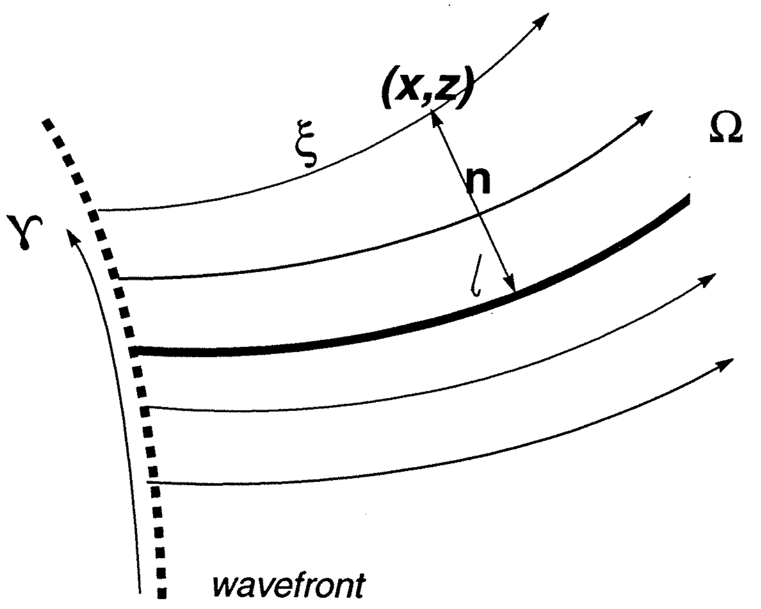

FIG. 2.1. Three coordinates systems are depicted; ray coordinates (ξ,γ), ray-center coordinates (ℓ,n) and Cartesian coordinates (x,z).

(ℓ,n) coordinates, and then from (ℓ,n) coordinates to the ray coordinates (ξ,γ). This discussion follows Červený (1981). The Jacobian of the transformation from $(x,\ z)$ to $(\xi,\ \gamma)$ coordinates is then expressed as a product of two Jacobians,

$$J = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \gamma} & \frac{\partial z}{\partial \gamma} \end{vmatrix} = \begin{vmatrix} \frac{\partial x}{\partial \ell} & \frac{\partial z}{\partial \ell} \\ \frac{\partial x}{\partial n} & \frac{\partial z}{\partial n} \end{vmatrix} \cdot \begin{vmatrix} \frac{\partial \ell}{\partial \xi} & \frac{\partial n}{\partial \xi} \\ \frac{\partial \ell}{\partial \gamma} & \frac{\partial n}{\partial \gamma} \end{vmatrix}. \tag{2.2}$$

As the ray-centered coordinate system is orthogonal with scaling factors h,1, we have

$$\begin{vmatrix} \frac{\partial x}{\partial \ell} & \frac{\partial z}{\partial \ell} \\ \frac{\partial x}{\partial n} & \frac{\partial z}{\partial n} \end{vmatrix} = h. \tag{2.3}$$

On the ray Ω, h=1, $\partial\ell/\partial\xi$=1, $\partial n/\partial\xi = 0$. From these results, it follows that

$$J = \frac{\partial n}{\partial \gamma}. \tag{2.4}$$

This is the expression of choice for the function J in the ray-centered coordinates (ℓ,n) connected with the ray Ω, as measured directly on Ω. Following Červený (1981), we introduce the system

$$\frac{dn}{d\ell} = vp_n, \quad \frac{dp_n}{d\ell} = -\frac{v_{,nn}}{v^2}n, \tag{2.5}$$

to describe the distance, n, to a neighboring ray Ω_c, near the central ray. Here p_n is the n-component of the slowness vector $\vec{p}$. Differentiating this system with respect to γ, we obtain

$$\frac{dJ}{d\ell} = vP, \quad \frac{dP}{d\ell} = -\frac{v_{,nn}}{v^2}J, \tag{2.6}$$

where

$$J = \frac{\partial n}{\partial \gamma}, \quad P = \frac{\partial p_n}{\partial \gamma}. \tag{2.7}$$

Here, J again denotes the Jacobian, and P is an auxiliary function (the derivative of the normal component of the slowness vector with respect to the ray parameter γ). We will see in the following chapter that J and P are calculated by dynamic ray tracing.

2.3 Formulation of 2.5-D Common-offset Inversion

Docherty used β_r and β_s for the take off angles from source and receiver points. Bleistein and Cohen introduced $\beta(x,z)$ for the output of their inversion formalism. For historical reasons, I use the same symbols to honor their contributions. The distinction should be clear in context.

The computational formula derived by Docherty(1989) for $\beta(x,z)$ for a flat observation surface is

$$\beta(\boldsymbol{x}) = \frac{1}{2(2\pi)^{3/2}} \int d\xi K(x_s, x_r, \boldsymbol{x}) D(\tau_s + \tau_r, \xi), \tag{2.8}$$

where

$$K(x_s, x_r, \boldsymbol{x}) = \sqrt{\frac{1}{\sigma_s} + \frac{1}{\sigma_r}} \left[\frac{\partial \beta_s}{\partial \xi} + \frac{\partial \beta_r}{\partial \xi} \right] \cdot [A(x_s, \boldsymbol{x}) A(x_r, \boldsymbol{x})]^{-1}, \tag{2.9}$$

$$D(t, \xi) = \frac{1}{A_f} \int d\omega \sqrt{i\omega} e^{-i\omega[\tau(\boldsymbol{x}, x_s) + \tau(\boldsymbol{x}, x_r)]} U_s(\omega, \xi), \tag{2.10}$$

Here, $\boldsymbol{x}=(x,z)$, shown in Figure 2.2, is the specular reflection point on the reflector, and $\beta(\boldsymbol{x})$ depicts reflectors as bandlimited delta functions that peak on the surface of each reflector. These are called singular functions of the reflectors. The peak amplitude of each singular function is predicted by the inversion theory described in Bleistein (1986). The normalization provided by the formula ensures that this peak value is the angularly-dependent reflection coefficient, $R(\boldsymbol{x}, \theta)$, at incident angle with respect to normal, θ, of the specular ray for a given source/receiver pair in the survey. The specular choice is consistent with the background velocity model; σ_r and σ_s are ray-trace running parameters from receiver and shot to output point, respectively. Also shown in Figure 2.2, x_s and x_r are the locations of shot and receiver, respectively; β_s and β_r are the angles between ray and upward vertical at $\boldsymbol{x}$, respectively; ξ is the location of the midpoint between source and receiver; $U_s(\omega, \xi)$ denotes the observed data at x_r due to the source at x_s; $\tau(\boldsymbol{x}, x_s)$ and $\tau(\boldsymbol{x}, x_r)$ are traveltimes between output points $\boldsymbol{x}$ and shot at x_s or receiver at x_r.

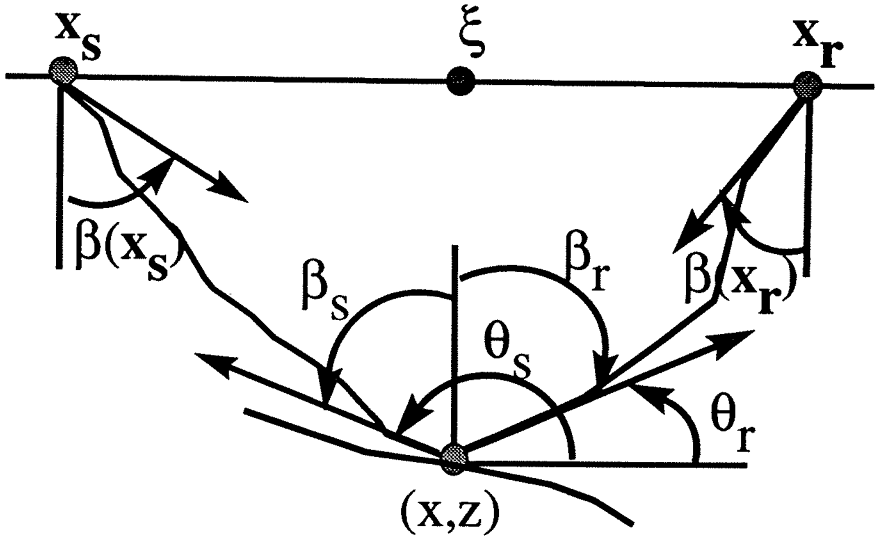

FIG. 2.2. Geometry of ray data.

The scaling factor, A_f, it is easiest to explain for synthetic data. One assumes a bandlimited delta function as the time signature of the source. That is, in the frequency domain, the Fourier transform of the source is a symmetric, real, nonnegative function, $F(\omega)$. A_f is the area under $F(\omega)$ for positive ω. For field data, the value of A_f is harder to define. It should again be the source signature, which now has to be estimated from the actual data. Further, given an estimate of the source signature, it may be further modified by filtering in the frequency domain to zero out data at the ends of the bandwidth that are judged to be unreliable. The product of the source signature and this filter would then play the role of $F(\omega)$ in the synthetic data. Also, we have

$$A(x_s, \boldsymbol{x}) = \frac{1}{4\pi}\left[\frac{c(\boldsymbol{x})}{\sigma_s \cos\beta(x_s)}\right]^{1/2}\left[\frac{\partial\beta_s}{\partial\xi}\right]^{1/2} T(x_s, \boldsymbol{x}),$$

$$A(x_r, \boldsymbol{x}) = \frac{1}{4\pi}\left[\frac{c(\boldsymbol{x})}{\sigma_r \cos\beta(x_r)}\right]^{1/2}\left[\frac{\partial\beta_r}{\partial\xi}\right]^{1/2} T(x_r, \boldsymbol{x}), \tag{2.11}$$

Here, $c(\boldsymbol{x})$ is velocity at output point $\boldsymbol{x}$; $\beta(x_s)$ and $\beta(x_r)$ are the angles between rays and downward vertical direction at x_s and x_r, respectively. Also

$$\begin{aligned} T(x_s, \boldsymbol{x}) &= \prod_{i_s=1}^{N} K_{i_s}\left[\frac{\cos\theta_{i'_s}}{\cos\theta_{i_s}}\right]^{1/2}, \\ T(x_r, \boldsymbol{x}) &= \prod_{i_r=1}^{N} K_{i_r}\left[\frac{\cos\theta_{i'_r}}{\cos\theta_{i_r}}\right]^{1/2}. \end{aligned} \tag{2.12}$$

In these last equations, $T(x_s, \boldsymbol{x})$ and $T(x_r, \boldsymbol{x})$ denote transmission effects on up-going rays from $\boldsymbol{x}$ to x_s or x_r respectively; K_{i_s} and K_{i_r} are transmission coefficients at each interface; $\theta_{i'_s}$ and θ_{i_s} are transmission angle and incidence angles with respect to the normal direction corresponding to the shot at each interface; $\theta_{i'_r}$ and θ_{i_r} are transmission angle and incidence angle with respect to the normal direction coresponding to the receiver at each interface. By using A_s and A_r as defined by (2.11) to substitute for $\partial\beta_s/\partial\xi$ and $\partial\beta_r/\partial\xi$ in (2.9), we obtain

$$\begin{aligned} K(x_s, x_r, \boldsymbol{x}) \quad &= \quad \frac{16\pi^2}{c(\boldsymbol{x})}\sqrt{\frac{1}{\sigma_s} + \frac{1}{\sigma_r}}[A(x_s, \boldsymbol{x})A(x_r, \boldsymbol{x})]^{-1} \\ &\cdot\left[\frac{A(x_s, \boldsymbol{x})^2 \sigma_s \cos\beta(x_s)}{T(x_s, \boldsymbol{x})^2} + \frac{A(x_r, \boldsymbol{x})^2 \sigma_r \cos\beta(x_r)}{T(x_r, \boldsymbol{x})^2}\right], \end{aligned} \tag{2.13}$$

Furthermore, from (Bleistein, 1986),

$$\begin{aligned} A(x_s, \boldsymbol{x}) = A(\boldsymbol{x}, x_s) &= \frac{1}{4\pi\sqrt{\sigma_s J_s}} T(\boldsymbol{x}, x_s), \\ A(x_r, \boldsymbol{x}) = A(\boldsymbol{x}, x_r) &= \frac{1}{4\pi\sqrt{\sigma_r J_r}} T(\boldsymbol{x}, x_r). \end{aligned} \tag{2.14}$$

Here, $T(\boldsymbol{x}, x_s)$ and $T(\boldsymbol{x}, x_r)$ denote transmission effects on down-going rays, J_s and J_r denote the *Jacobian* functions for shot and receiver respectively. Thus, the final expression for the inversion is

$$\beta(\boldsymbol{x}) = \frac{2\sqrt{2\pi}}{c(\boldsymbol{x})}\int d\xi K(x_s, x_r, \boldsymbol{x}) D(\tau_s + \tau_r, \xi), \tag{2.15}$$

where

$$K(x_s, x_r, \boldsymbol{x}) = \sqrt{\sigma_s + \sigma_r} \cdot \left[\sqrt{\frac{J_r}{J_s} \frac{T(\boldsymbol{x}, x_s) \cos\beta(x_s)}{T(\boldsymbol{x}, x_r) T(x_s, \boldsymbol{x})^2}} + \sqrt{\frac{J_s}{J_r} \frac{T(\boldsymbol{x}, x_r) \cos\beta(x_r)}{T(\boldsymbol{x}, x_s) T(x_r, \boldsymbol{x})^2}} \right], \quad (2.16)$$

$$D(t, \xi) = \frac{1}{A_f} \int d\omega \sqrt{i\omega} e^{-i\omega[\tau(\boldsymbol{x}, x_s) + \tau(\boldsymbol{x}, x_r)]} U_s(\omega, \xi). \quad (2.17)$$

The preprocessing of (2.17) is carried out by a program developed by Sumner (1990). My code uses the output $D(t, \xi)$ to compute $\beta(\boldsymbol{x})$ in (2.15). This is the inversion formula that I will use to calculate R, the angularly dependent reflection coefficient at each interface. Also, I calculate a second integral which yields $R\cos\theta$ as output, where θ is the incidence angle with respect to the normal direction at output points. Following is the formula for $R\cos\theta$

$$\beta_1(\boldsymbol{x}) = \frac{2\sqrt{2\pi}}{c(\boldsymbol{x})} \int d\xi K_1(x_s, x_r, \boldsymbol{x}) D(\tau_s + \tau_r, \xi), \quad (2.18)$$

where

$$K_1(x_s, x_r, \boldsymbol{x}) = \cos\left(\frac{\theta_s - \theta_r}{2}\right) K(x_s, x_r, \boldsymbol{x}). \quad (2.19)$$

Here, the function $\beta_1(\boldsymbol{x})$ is like $\beta(\boldsymbol{x})$, except for the peak value that it produces. The peak value of $\beta_1(\boldsymbol{x})$ is $R\cos\theta$. Thus, the ratio of these two outputs yields an estimate of $\cos\theta$, while the peak value of $\beta(\boldsymbol{x})$ yields an estimate of R, itself; θ_s and θ_r are incidence angles at $\boldsymbol{x}$ with respect to horizontal direction from shot and receiver, respectively, shown in Figure 2.2. Note that θ_s and θ_r are two angles that I use to calculate θ, which is the incident angle of the specular rays from source or receiver, relative to the normal to the reflector. That is, at specular $(\theta_s - \theta_r)/2$=θ.

In the next chapter, I describe ray tracing in triangulated models.

Chapter 3

RAY TRACING IN A TRIANGULATED MODEL

3.1 Introduction

In this chapter, I describe ray tracing in a triangulated model, developed by Hale (1991). The earth model used by this method is characterized by sloth, which is the inverse of velocity-squared. In a triangulated model, each triangle has constant sloth gradient. This choice allows closed-form solutions for the elements of the ray data that we need, such as the ray trajectory, traveltime and ray parameters, etc. Included among these is a geometrical spreading factor, as well. More detail about ray tracing in triangulated models is available in Hale (1991) and Rüger (1993).

3.2 Ray tracing equation

Following Červený (1987), we express the ray tracing equations in two dimensions as follows:

$$
\begin{aligned}
\frac{dx}{d\sigma} &= p_x, \\
\frac{dz}{d\sigma} &= p_z, \\
\frac{dp_x}{d\sigma} &= \frac{1}{2}\frac{\partial v^{-2}}{\partial x}, \\
\frac{dp_z}{d\sigma} &= \frac{1}{2}\frac{\partial v^{-2}}{\partial z}, \\
\frac{dt}{d\sigma} &= v^{-2}.
\end{aligned} \tag{3.1}
$$

Here v denotes velocity, x and z denote the Cartesian coordinates of the ray, and p_x and p_z are the x and z components, respectively, of the slowness vector. Furthermore, t is traveltime, and σ is a running parameter that increases monotonically along the ray, as implied by the last of equations (3.1).

The reason for writing the ray tracing equations in this form is that the partial derivatives of v^{-2} in (3.1) are constants when v^{-2} is a linear function of x and z. Specifically, we define "sloth" to be the inverse of velocity-squared; that is,

$$
s(x,z) \equiv v(x,z)^{-2}, \tag{3.2}
$$

and introduce the linear approximation,

$$s(x,z) = s_{00} + s_{,x}x + s_{,z}z, \tag{3.3}$$

in each triangle with s_{00}, $s_{,x}$ and $s_{,z}$ constants. Then the solutions to the equations (3.1) in each triangle are

$$\begin{aligned}
x(\sigma) &= x(\sigma_0) + p_x(\sigma_0)(\sigma - \sigma_0) + \frac{1}{4}s_{,x}(\sigma - \sigma_0)^2, \\
z(\sigma) &= z(\sigma_0) + p_z(\sigma_0)(\sigma - \sigma_0) + \frac{1}{4}s_{,z}(\sigma - \sigma_0)^2, \\
p_x(\sigma) &= p_x(\sigma_0) + \frac{1}{2}s_{,x}(\sigma - \sigma_0), \\
p_z(\sigma) &= p_z(\sigma_0) + \frac{1}{2}s_{,z}(\sigma - \sigma_0), \\
t(\sigma) &= t(\sigma_0) + [s_{00} + s_{,x}x(\sigma_0) + s_{,z}z(\sigma_0)](\sigma - \sigma_0) + \\
&\quad \frac{1}{2}[s_{,x}p_x(\sigma_0) + s_{,z}p_z(\sigma_0)](\sigma - \sigma_0)^2 + \\
&\quad \frac{1}{12}({s_{,x}}^2 + {s_{,z}}^2)(\sigma - \sigma_0)^3.
\end{aligned} \tag{3.4}$$

The ray path described by equation (3.4) is a parabola in σ, and the intersection of this ray path with the linear edge of a triangle may be easily computed by solving a quadratic equation in σ. The number of real roots of this equation is either zero, one, or two, corresponding to the number of possible intersections. To determine the edge at which a ray exits a triangle, one must solve three quadratic equations for σ, one for each edge of the triangle. Then, letting σ_0 denote the value of σ at the point where a ray enters the triangle, the value of σ where the ray exits is the smallest real root of these three quadratic equations that is greater than σ_0. The value of σ at the exit point then yields the parameters in equations (3.4) at that point.

3.3 Dynamic ray tracing equations

The equations of the previous section do not include the necessary variable to determine geometrical spreading. Here, I extend the dependent variable set to include geometrical spreading by using dynamic ray tracing. This method also detects caustics. Dynamic ray tracing in two dimensions requires the solution of the following system of coupled differential equations (Červený, 1987):

$$\begin{aligned}
\frac{dq}{d\sigma} &= p \\
\frac{dp}{d\sigma} &= -\frac{v_{,nn}}{v^3}q,
\end{aligned} \tag{3.5}$$

where again $v_{,nn}$ denotes the second partial derivative of velocity v with respect to distance n in a direction normal to the ray. Given a central ray described by equation (3.1), parameters p and q describe a nearby (paraxial) ray. Specifically, p is the partial derivative of traveltime t with respect to n for the paraxial ray, and q is the normal distance n from the central ray to the paraxial ray. The ratio p/q is the second partial derivative of t with respect to n, and is related to the curvature of the wavefront normal to the ray. Also, from this pair of equations, if we use $d\sigma = vd\ell$, we obtain (2.6) with J replaced by q and P replaced by p. Thus, q is just the *Jacobian* required in our inversion. For the linear sloth used here,

$$\begin{aligned} \frac{dq}{d\sigma} &= p, \\ \frac{dp}{d\sigma} &= -\frac{3}{4}\frac{(s_{,x}p_z - s_{,z}p_x)^2}{s^2}q. \end{aligned} \tag{3.6}$$

The solution to these equations is

$$\begin{aligned} q(\sigma) &= \frac{1}{\sqrt{s_0 s}}\left[[s_0 + \frac{s_1(\sigma-\sigma_0)}{2}][q(\sigma_0) + p(\sigma_0)(\sigma-\sigma_0)] + (\frac{{s_1}^2}{4s_0} - s_2)q(\sigma_0)(\sigma-\sigma_0)^2\right] \\ p(\sigma) &= \frac{1}{\sqrt{s_0 s}}\left[[s_0 + s_1(\sigma-\sigma_0)]p(\sigma_0) + [\frac{s_1}{2} + (\frac{{s_1}^2}{2s_0} - 2s_2)(\sigma-\sigma_0)]q(\sigma_0)\right] \\ &\quad -[\frac{s_1 + 2s_2(\sigma-\sigma_0)}{2s}]q(\sigma), \end{aligned} \tag{3.7}$$

where s_0, s_1, s_2 are constants defined by

$$\begin{aligned} s_0 &\equiv s[x(\sigma_0), z(\sigma_0)], \\ s_1 &\equiv s_{,x}p_x(\sigma_0) + s_{,z}p_z(\sigma_0), \\ s_2 &\equiv \frac{1}{4}({s_{,x}}^2 + {s_{,z}}^2). \end{aligned} \tag{3.8}$$

Also, from (3.3) (3.4) and (3.8), one can show that

$$s = s[x(\sigma), z(\sigma)] = s_0 + s_1(\sigma-\sigma_0) + s_2(\sigma-\sigma_0)^2 \tag{3.9}$$

Given the values $q(\sigma_0)$ and $p(\sigma_0)$ at the point where a ray enters a triangle, equations (3.7) yield the values $q(\sigma)$ and $p(\sigma)$ at the point where the ray exits that triangle. In summary, we use (3.4) and (3.7) to obtain the ray path, the function J and other ray data that we need at specific nonuniformly spaced outpoints. In the next chapter I describe the problem of obtaining ray data at grid points.

3.4 The KMAH index

In dynamic ray tracing a quantity called the KMAH index (Hale, 1991) records the number of times that a paraxial ray from a point source crosses its central ray. The

KMAH index plays an important role in situations in which the ray family has caustics or envelopes. We tabulate the KMAH index by counting the zeros of $q(\sigma)$ along the ray. The KMAH index counts the number of phase shifts of $\pi/2$ required in the solution for rays that pass through caustics.

Most applications of dynamic ray tracing equations require the solutions $q(\sigma)$ and $p(\sigma)$ of equation (3.5) for the initial conditions $q(0) = 0$ and $p(0) = 1$. Following Červený et al. (1982) the solutions corresponding to these initial conditions are usually labeled $q_2(\sigma)$ and $p_2(\sigma)$, the solutions corresponding to a point source.

Chapter 4

DETERMINATION OF RAY DATA ON A UNIFORM GRID

In this chapter, I describe the process used to transform ray data from a triangulated system to a uniform grid. I also describe parallel interpolation for efficiency in computation.

4.1 Transformation from a triangulated system to a uniform grid

Transformation of ray data from a triangulated system to a uniform grid is crucial to my application of this modeling technique to inversion. Here, I call the set of source or receiver grid points on the upper-surface S and the number of such points, N_s. Also, I call the set of grid points in the x-direction, X and the set of grid points in the z-direction, Z with the number of them N_x and N_z, so that the total number of such grid points, (x, z) is $N_x \times N_z$. We denote the combined set of depth points by $\{X, Z\}$.

Normally, I set the increment dz, to be one quarter of wavelength at the dominant frequency of the data. For dx, the increment in the $x-$direction, I usually use the same value as the midpoint spacing for the desired data on the upper-surface, although these two values can be chosen independently. The objective is to obtain ray data—traveltime, amplitude, etc.—from each point in S to each point in $\{X, Z\}$. The structure of the technique for achieving this is described as follows.

1. Shoot rays from point s in S into the subsurface at some fixed distribution in take off angles, typically in approximately one-degree increments.

2. Determine the triangles of the triangulated model that contain horizontal line segments at a given depth z in Z. By solving equations (3.4) and (3.7), evaluate the ray data at the intersections of the rays with the line at depth z. For example, in Figure 4.1, the depth z was chosen as 3 km (indicated by the horizontal line at 3-km depth). The figure shows the triangles that the horizontal line at this depth crosses and it shows the intersections of the rays from a surface-point s=3.5 km with this line. In this way, the intersection points with all grid depths z are identified.

3. Since the intersection points (x values at depth z) are still not grid points, I use linear interpolation to determine ray data at grid points from ray data at intersection points. In Figure 4.2, the rays are depicted schematically as straight lines with arrows, and the grid points are shown as circles.

This scheme cannot interpolate properly in the vicinity where rays turn from downward to upward. Near turning, the distance between two intersection points is too large

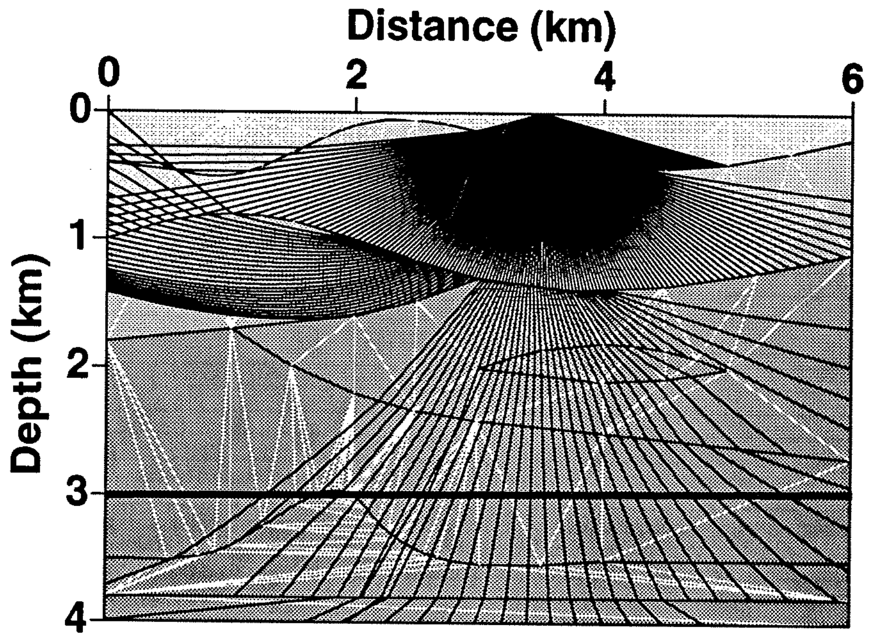

FIG. 4.1. Shooting rays from surface.

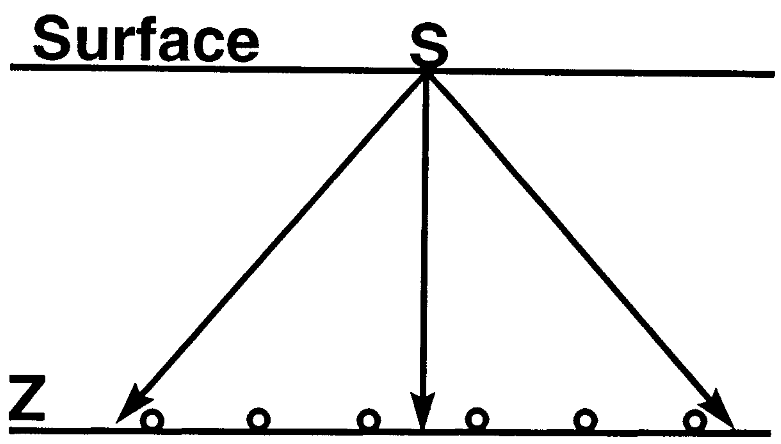

FIG. 4.2. Interpolation from non-uniform points to grid points.

for interpolation. To deal with rays propagating nearly horizontally, I apply a 2-D interpolation technique. That is, in addition to horizontal interpolation, I use vertical interpolation to improve the accuracy of the interpolated values. Consider Figure 4.3, which depicts two nearly horizontal rays near a grid point which is represented by a circle. I obtain the ray data at four points: two at the level of interest, one level higher and one level lower. In the figure, these points are labeled 1,2,3,4. I linearly interpolate ray data at points 3 and 4 to the grid point and retain the distance between points 3 and 4, $d_{3,4}$, as well. Similarly, I use the points 1 and 2 to obtain interpolated ray data at the grid point and I compute $d_{1,2}$, which is distance between points 1 and 2. I compare $d_{1,2}$ and $d_{3,4}$. If $d_{1,2}$ is smaller than $d_{3,4}$, I would use the interpolated values from points 1 and 2 to provide ray data at the grid point.

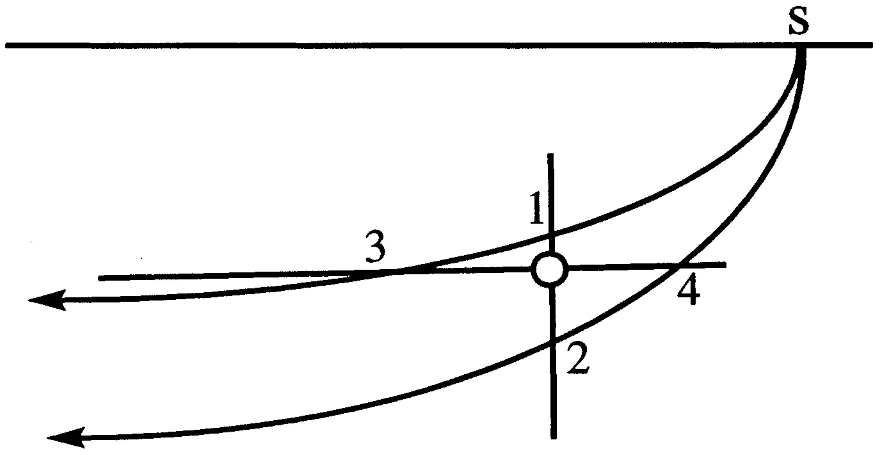

FIG. 4.3. 2-D interpolation.

4.2 Parallel interpolation between source points

The use of ray tracing to implement inversion basically involves two major steps: (1) computation of ray data for grid points, and (2) computation of the integral in equation (2.15). If the scheme described in the previous section were carried out for each point in $\mathcal{S}$ and each point in $\{X, Z\}$, then $N_s \times N_x \times N_z$ sets of ray data would need to be computed. This may require an unacceptable amount of CPU time and, as will be shown below, is often unnecessary. Instead, we compute ray data for only some subset of $\mathcal{S}$, say, every fifth or every tenth point, and all of $\{X, Z\}$ and then use a parallel interpolation scheme developed by Liu (1993) to fill in the missing data for the remainder of the points in $\mathcal{S} \times \{\mathcal{X}, \mathcal{Z}\}$. [The program has a parameter called SKIP that provides the user the option of choosing this more sparse set of points as a fraction (1/skip) of N_s]. The parallel interpolation can be described as follows.

1. Define the spacing of selected sparse initial surface-points, which can be either shot or receiver.

2. Obtain ray data for points in this sparse subset of $\mathcal{S}$. Suppose that S is a point in $\mathcal{S}$, not in the sparse subset and that S_1 and S_2 are points in the sparse subset for which ray data have been calculated, by the above description, for all points at depth in the grid $\{X, Z\}$. Define L_1 as the distance from S to S_1 and L_2 to be the distance from S to S_2 (Figure 4.4). At a depth z, we need ray data at each x on the grid for all source points between S_1 and S_2. Given a value of x on the grid, define x_1 to be the point on the grid at distance L_1 from x—to the left of x, the same distance by which S_1 is to the left of S—and x_2 to be the point a distance L_2 from x on the other side. The ray data from S_1 to x_1 and from S_2 to x_2 are known since S_1 and S_2 were part of the sparse set of initial points of rays, and (x_1, z) and (x_2, z) are points on the grid, $\{X, Z\}$. Use these ray data to obtain the corresponding data from S to (x, z) by interpolation. This step is illustrated in Figure 4.4.

   The formula used for this interpolation is

$$\rho(x, z, S) = \lambda\rho(x_2, z; S_2) + (1 - \lambda)\rho(x_1, z; S_1), \tag{4.1}$$

   where, ρ is any element of the ray data and $\lambda = L_1/(L_1 + L_2)$.

3. Repeat step 2 for each x at depth z.

4. Repeat steps 2 and 3 for each z.

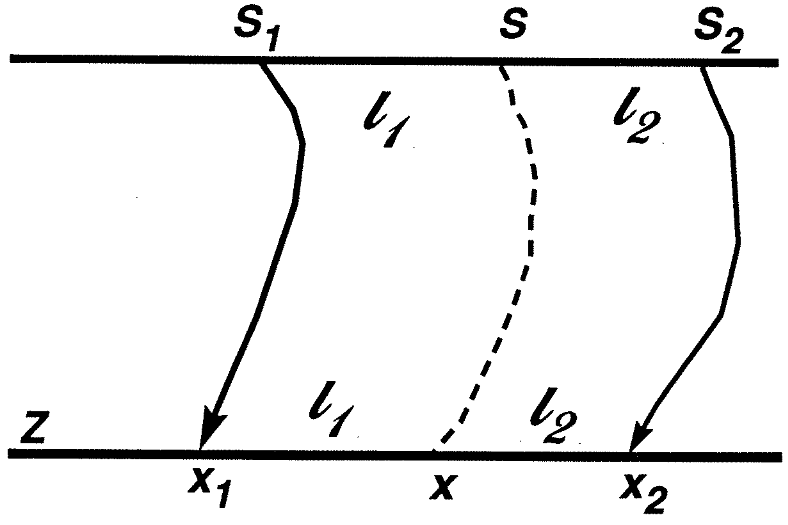

FIG. 4.4. Parallel interpolation

When the velocity is only depth-dependent, this linear interpolation is exact because the ray data are invariant for lateral displacement; that is for any h,

$$\rho(x+h, z; S+h) = \rho(x, z; S) \tag{4.2}$$

For a general velocity function $v(x, z)$, the interpolation error depends on $\partial v / \partial x$, but the parallel interpolation should work with some degree of success so long as the lateral variations are not too large. In the next chapter, using cpu time without parallel interpolation as a benchmark, I show results that demonstrate reduced cpu time with little loss in accuracy achieved by this interpolation method.

Chapter 5

COMPUTER IMPLEMENTATION

In this chapter, I present the results of tests of the inversion program. The objective of these examples is (i) to test the accuracy of reflector location, (ii) to show the accuracy of the prediction of the angularly dependent reflection coefficient, and (iii) to test the accuracy of using a sparse set of initial points on the upper-surface followed by interpolation to generate ray data at all grid points. In the first four examples, model velocity is used for inversion; in the last two examples, velocities obtained by Liu's velocity analysis (1991 and 1995) are used to obtain images of reflector location.

5.1 Example 1

This example provides a simple test of positioning and amplitude accuracy. To test amplitude accuracy in Bleistein's theory (1986), we need reliable amplitude in data for $U_s(\omega, \xi)$ in (2.17). In this example and next example, I use CSHOT (Docherty, 1991), which uses two-point ray tracing to generate amplitude 2.5-D synthetic data. Here, I chose a simple model consisting of constant-velocity layers, as shown in Figure 5.1. Based on Snell's law and the formula for the reflection coefficient, I can obtain analytic results for R and $R\cos\theta$ at each point on each reflector. For the single offset used in this example, these values are constant on each reflector. In this model, layer velocities are 5, 6 and 7 km/s from the shallowest to the deepest layer, respectively. Figure 5.2 shows model data with offset=2 km. The inversion produces two outputs: one for $\beta(\boldsymbol{x})$, the other for $\beta_1(\boldsymbol{x})$.

The output for $\beta_1(\boldsymbol{x})$ is similar to $\beta(\boldsymbol{x})$ in the sense that it provides a reflector map; the amplitudes differ by the factor $\cos(\theta_s - \theta_r)/2$ which appears in the integrand in (2.18) for $\beta_1(\boldsymbol{x})$, but not in the integrand of (2.15) for $\beta(\boldsymbol{x})$. Since the model velocity is used for this example, so location of the peak amplitude of the output on each reflector is virtually at the correct depth and the peak values on each reflector provide estimates of R and $R\cos\theta$.

For the first interface, R is 0.144, $R\cos\theta$ is 0.120, and θ is 33.7 degrees. In my result, R is 0.148, $R\cos\theta$ is 0.123 and θ is 33.8 degrees. The percentage errors here are 2.8, 2.7 and 0.2 percent respectively. For the second interface, the exact value of R is 0.095, $R\cos\theta$ is 0.087 and θ is 24.3 degrees. In my result, R is 0.092, $R\cos\theta$ is 0.084 and θ is 24.35 degrees. The percentage errors here are 2.4, 2.7 and 0.2 percent, respectively. The fact that the inversion code has taken transmission losses into account is important to the accuracy obtained for values at the second interface. The code thus produces output that is consistent with Bleistein's theory (1986). It provides both a reflector map and model-consistent estimates of R and $\cos\theta$ as long as the background velocity used

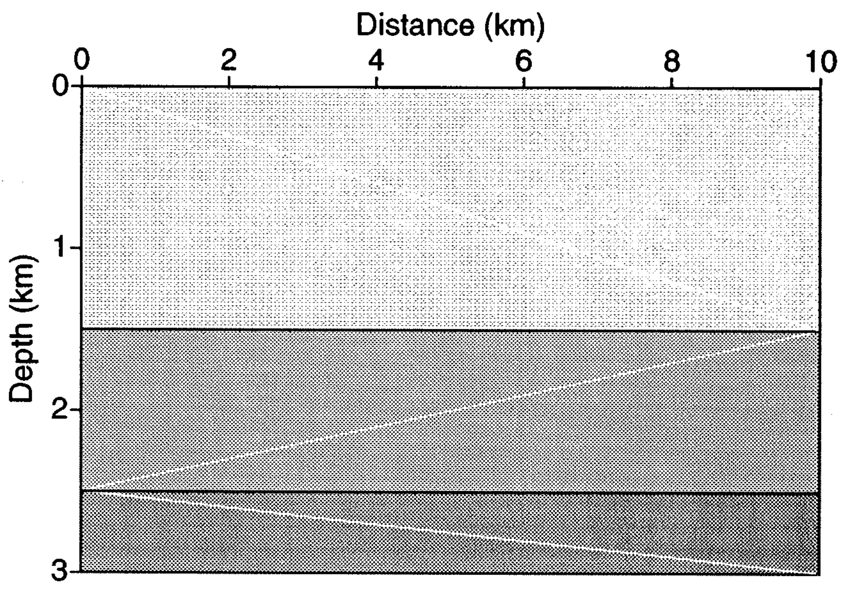

FIG. 5.1. Model of horizontal layers.

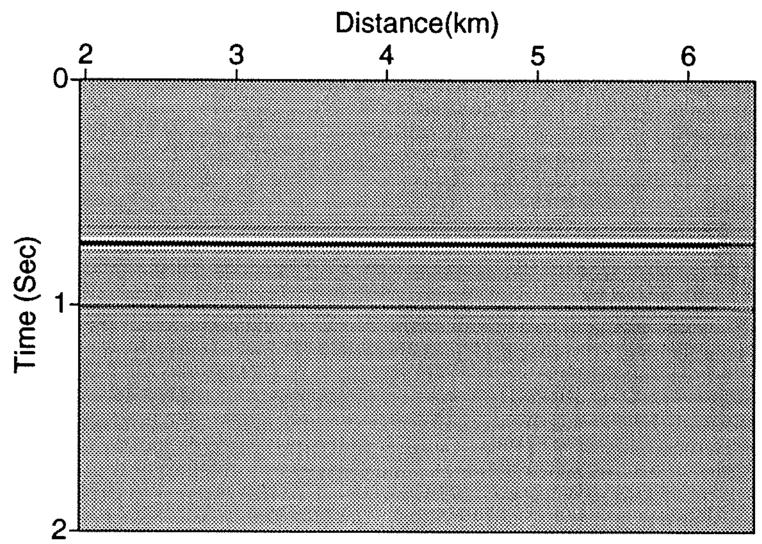

FIG. 5.2. Synthetic data generated by CSHOT with offset=2 km.

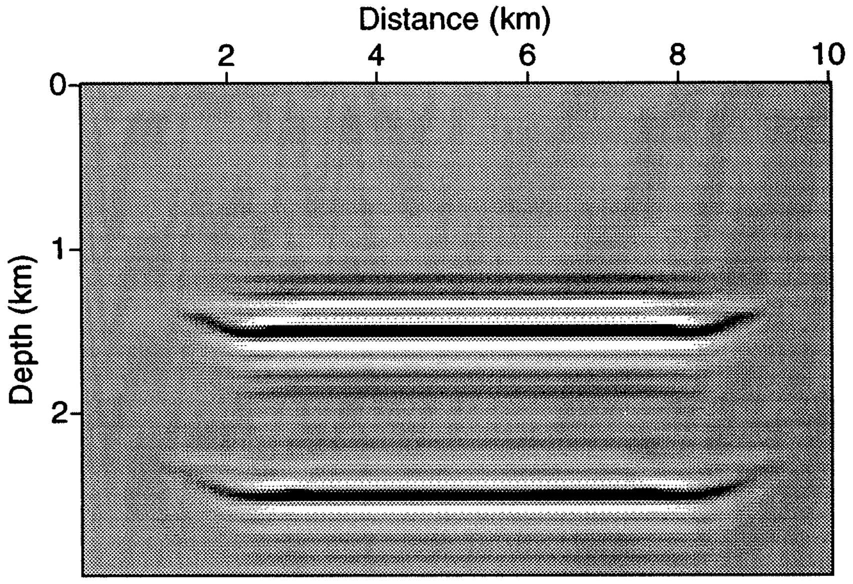

FIG. 5.3. R.

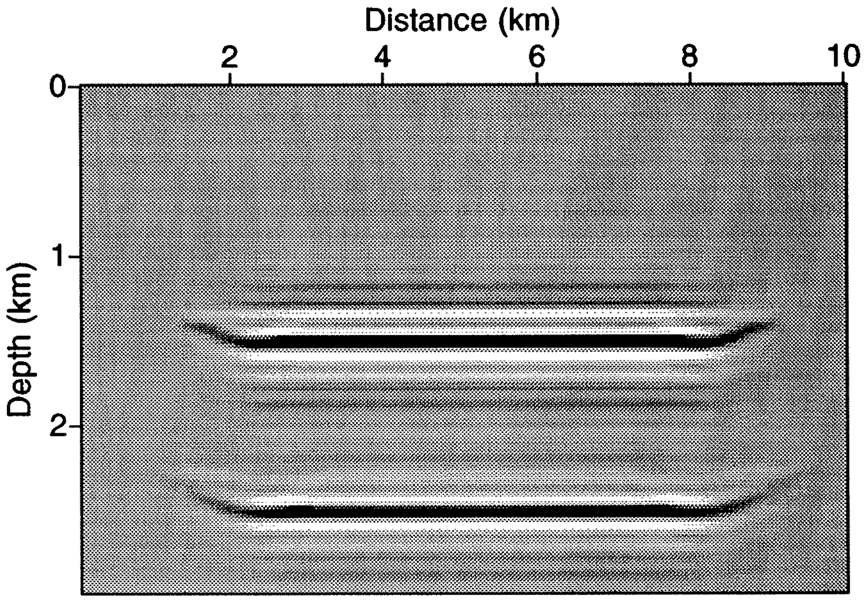

FIG. 5.4. $R\cos\theta$.

for the inversion is the correct one for the layered medium. Also, since this is analytic test of amplitude, we know the amplitudes in the input data very well. In practice, we do not know amplitude of input data perfectly. However, from the results here we can be confident that the inversion treats relative amplitude properly. Therefore, based on relative amplitude of output, we can at least do bright-spot analysis and amplitude-versus-offset (AVO) analysis.

5.2 Example 2

In this example, to test amplitude, I use synthetic data for an experiment similar to the "Marathon tank model," which has parameters of a physical tank model provided by Marathon Oil Company (Liu, 1991). The model is illustrated in Figure 5.5. Since this is a more complex model than that of Example 1, it is more difficult to directly compute R and θ as benchmark for our tests. Therefore, let us use an indirect test of accuracy based on the formula for the reflection coefficient (Bleistein, 1987),

$$R(\boldsymbol{x},\theta) = \frac{\cos\theta - \sqrt{v_1^2/v_2^2 - \sin\theta^2}}{\cos\theta + \sqrt{v_1^2/v_2^2 - \sin\theta^2}} \tag{5.1}$$

Here, $\boldsymbol{x}$ is the location of reflector, v_1 and v_2 denote velocities above and below the reflector, respectively, $R(\boldsymbol{x},\theta)$ is the reflection coefficient at $\boldsymbol{x}$ at specular incidence angle θ. From the discussion following equation (2.17), the peak values of $\beta(\boldsymbol{x})$ and $\beta_1(\boldsymbol{x})$ on the reflectors provide values of R and $R\cos\theta$. Given these values and using the model value for the velocity above the reflector, one can compute the velocity below the reflector from (5.1) and compare with the exact value.

This is a constant-velocity-layer model; the velocities are 11.75, 15.75, 22.40 , 19.90 and 30.00 kft/s, from the shallowest to deepest layers. Figure 5.6 shows, for offset 2400 ft, synthetic data generated by CSHOT (Docherty, 1991), which uses two-point ray tracing to generate the data. Figure 5.7 and Figure 5.8 are the inversion results for R and $R\cos\theta$, again based on computations in which the background velocity is known exactly.

To test amplitude, I chose a few trace locations to estimate the velocity below each interface, assuming I know the velocity above. Tables 5.1 through 5.5 compare my estimated velocities v_2' with the true layer velocities v_2.

The velocity estimates are good except at a few locations. At point (3.2,8.8) the error is greater than 10 percent. With the aid of Figure 5.9, we can provide an explanation of this large error. The figure shows normal rays from the sawtooth reflector to the upper-surface. For zero-offset data, these would be the specular rays for this model from this reflector. We expect that, qualitatively, the features of these rays are similar to the analogous features for the specular rays for small offset. For the point (3.2,8.8), specular rays for this reflection point emerge at the upper surface near the edge of the model. If they were right at the edge, Bleistein's (1986) theory would predict peak amplitudes for $\beta(\boldsymbol{x})$ and $\beta_1(\boldsymbol{x})$ to be $R/2$ and $R/2\cos\theta$. Therefore I used twice the peak amplitudes, replacing 0.0434 and 0.0430 by 0.0868 and 0.086, as values of R and $R\cos\theta$. Using these

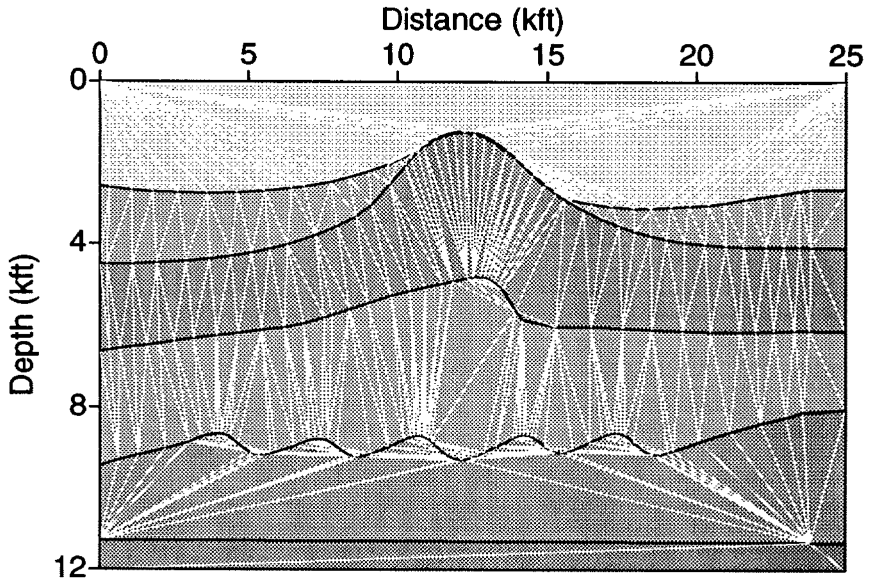

FIG. 5.5. Triangulated synthetic marathon model.

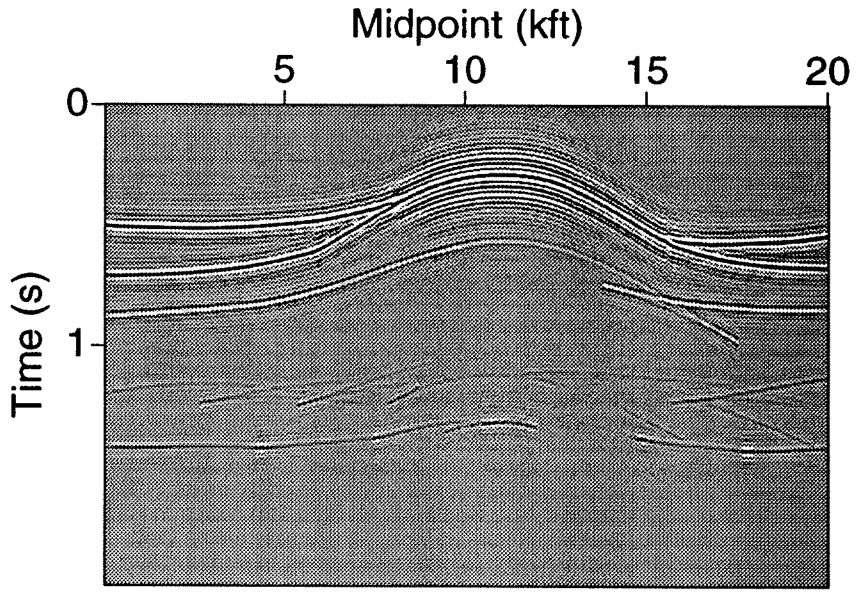

FIG. 5.6. Synthetic data for offset=2400 ft.

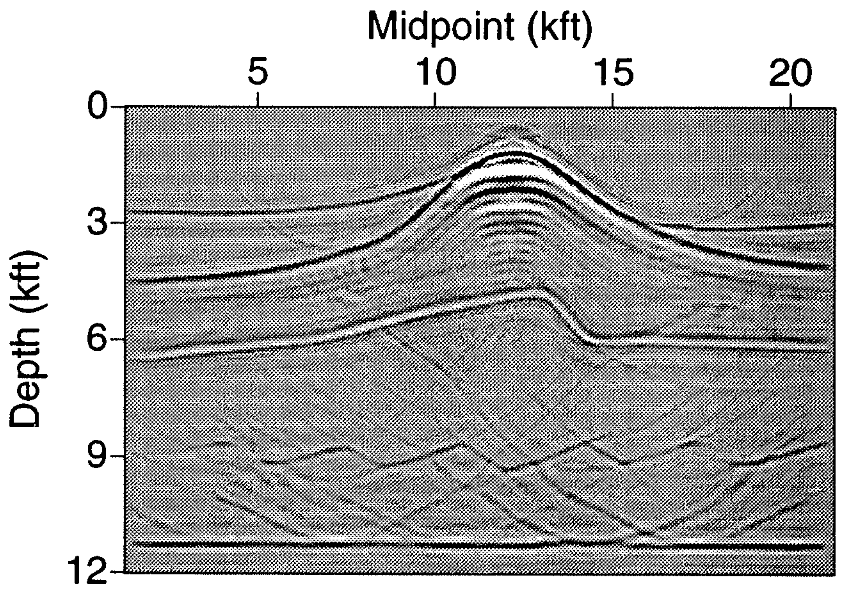

FIG. 5.7. R.

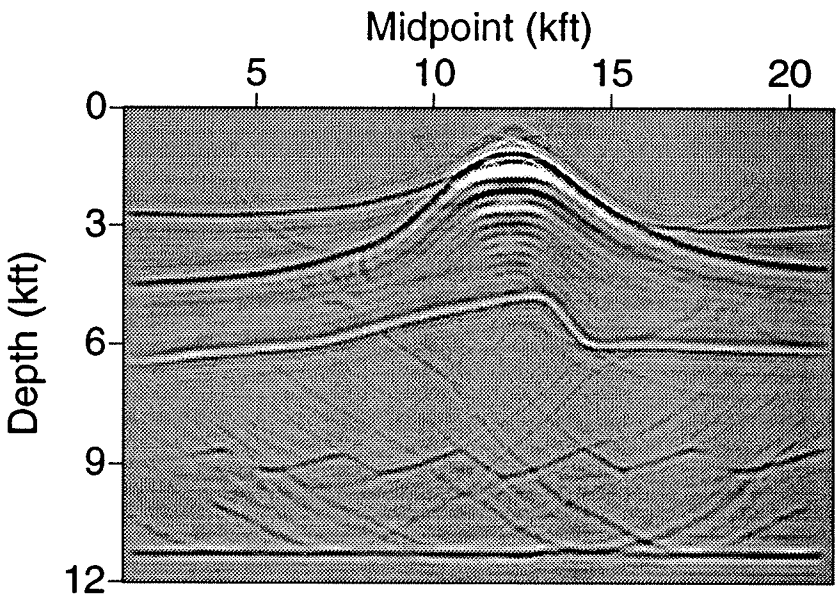

FIG. 5.8. $R \cos\theta$.

Table 5.1.

output points (kft)	R	R$\cos\theta$	v_2 (kft/s)	v_2' (kft/s)	error %
(3.2, 2.8)	0.1562	0.1433	15.75	16.04	2
(3.2, 4.4)	0.1809	0.1685	22.40	22.49	0
(3.2, 6.4)	-0.1789	-0.1757	15.75	16.02	2
(3.2, 8.8)	0.0434	0.0430	19.90	17.31	13
(3.2, 11.3)	0.1819	0.1802	30.00	28.72	4.3

Table 5.2.

output points (kft)	R	R$\cos\theta$	v_2 (kft/s)	v_2' (kft/s)	error %
(7.2, 2.6)	0.1786	0.1487	15.75	15.98	1
(7.2, 3.8)	0.1775	0.1560	22.40	21.86	2.4
(7.2, 6.0)	-0.1681	-0.1623	15.75	16.78	6.5
(7.2, 8.8)	0.1226	0.1214	19.90	20.21	1.5
(7.2, 11.3)	0.2366	0.2355	30.00	32.14	6.6

Table 5.3.

output points (kft)	R	R$\cos\theta$	v_2 (kft/s)	v_2' (kft/s)	error %
(8.8, 2.4)	0.1790	0.1539	15.75	16.20	2.5
(8.8, 3.3)	0.1533	0.1332	22.40	21.19	5.3
(8.8, 5.6)	-0.1670	-0.1612	15.75	16.82	6.7
(8.8, 9.2)	0.1044	0.1032	19.90	19.52	2.5
(8.8, 11.3)	0.1172	0.1148	30.00	25.36	15.5

Table 5.4.

output points (kft)	R	R$\cos\theta$	v_2 (kft/s)	v_2' (kft/s)	error %
(19.2, 3.2)	0.1590	0.1483	15.75	16.16	2.6
(19.2, 4.0)	0.1809	0.1611	22.40	22.06	3.6
(19.2, 5.9)	-0.1605	-0.1573	15.75	16.68	6.0
(19.2, 9.2)	0.1010	0.1007	19.90	19.31	3
(19.2, 11.3)	0.1689	0.1672	30.00	27.98	6.6

Table 5.5.

output points (kft)	R	R$\cos\theta$	v_2 (kft/s)	v_2' (kft/s)	error %
(20.4, 3.0)	0.1651	0.1483	15.75	16.17	2.6
(20.4, 4.0)	0.2080	0.1933	22.40	23.30	4.0
(20.4, 6.2)	-0.1539	-0.1511	15.75	16.86	7.0
(20.4, 8.8)	0.1337	0.1324	19.90	20.66	3.8
(20.4, 11.3)	0.2454	0.2435	30.00	32.65	8.8

values, the revised estimated velocity is 18.85 kft/s and the error is reduced from 13 to 5.2 percent. Presumably, the specular rays are not right at the edges, but near enough that this correction improves our estimate of velocity.

At location (8.8,11.3), the error is 15.5 percent. From the ray picture in Figure 5.10, in this region the density of normal rays at surface is higher than elsewhere. From this, I conjecture that there is a caustic nearby in the ray family of the Green's function. The underlying theory for estimation of the reflection coefficient (Bleistein, 1986) breaks down in such regions and thus, the inaccuracy here is not surprising.

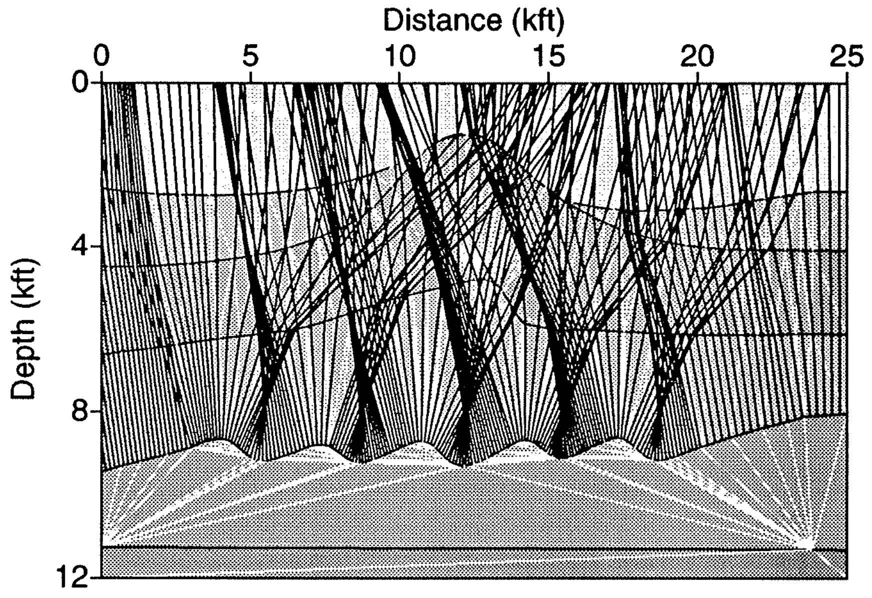

FIG. 5.9. Normal rays from sawtooth reflector.

This example demonstrates that the inversion output produces reliable velocity estimates in more complex models, inferring, indirectly, that the computed reflectivity and reflection angles are reliably computed as well. The significance of the inference is important because it suggests that even if absolute value of amplitude of the data

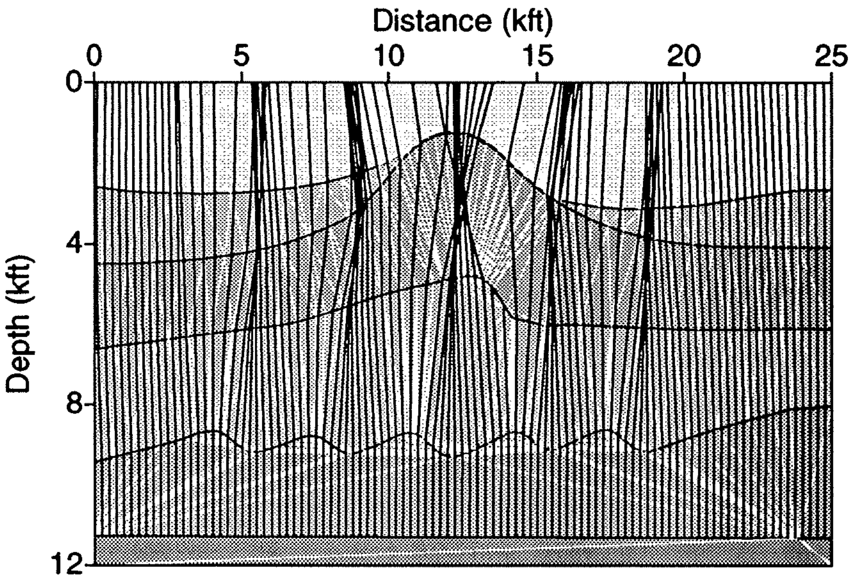

FIG. 5.10. Normal rays from horizontal reflector.

are not known, the results of inversion can be reliably used in bright-spot analysis and amplitude-versus-offset (AVO) analysis.

5.3 Example 3

In this example, I test the trade-off between cost and accuracy with the use of parallel interpolation in the complex model shown in Figure 1.1. This model consists of eight reflectors, with a lens-shaped inclusion. The dark lines define the fixed interfaces and the white lines are the triangle's edges in the model generated by GBMODEL (Hale, 1991), which generates the triangulated earth model used in the ray tracing. In some layers the velocity is constant, and in other layers sloth changes linearly in both horizontal and vertical directions. After building the model, I use GBSEIS (Rüger, 1993), which generates synthetic data by the Gaussian Beam method, to generate a common-offset data set with offset 0.3 km (Figure 5.11). Figure 5.12 is the inversion result for which ray data were computed by ray tracing from every surface-point (spacing Δs=.015km, here and below; Δs denotes the surface-point spacing for ray tracing) to every grid point at depth—$N_s \times N_x \times N_z$ sets of ray data. This is the benchmark against which to compare inversion with interpolated ray data generated by starting with a sparser set of initial points on the upper-surface.

The program has a parameter called "skip" that characterizes the sparsity of surface-points from which ray data are directly calculated. For example, when skip=50, rays are shot for every 50th surface-point. This parameter is set by the user. For the benchmark described above, skip=1.

Since the amplitude of synthetic data generated by the Gaussian beam method is not reliable, I will not consider amplitude in this example. Figure 5.13 is an inversion result with the use of parallel interpolation between surface-points. Here, I computed ray data for a subset of surface-points and then use parallel interpolation to obtain ray data on grid points for all surface-points. In this first test, rays were shot for every tenth surface-point, $\triangle s = .15$ km or skip=10. Figure 5.12 and Figure 5.13 show comparable results. However, the run times are quite different. The latter is five times faster than the former.

On the other hand, when the midpoint spacing for ray tracing is too large, the imaging is unacceptably distorted, as shown in Figure 5.14 for which $\triangle s$ is 0.75 km (skip=50). Intermediate results suggest that skip=10 is about the acceptable limit of sparsity for this example.

It is worthwhile to examine some artifacts of the output here. Note, that the image of the first interface becomes thicker at horizontal distance around 2.5 km. This happens because the reflector is close to the recording surface; the ray incidence angle is relatively large compared to its value at deeper points. The underlying theory (Bleistein, etal, 1987) predicts that the range of wavenumbers in the output is proportional to $\cos\theta$ and hence decreases with increasing θ. A lower range of wavenumbers leads to a broader peak on output.

At great depths, the thickness of the image of the interface is seen to increase, as well. Here, $\cos\theta$ is nearer to one, but the background velocity is larger than at shallower depths. The range of wavenumbers varies inversely with velocity, hence, again the range of wavenumbers diminishes with increasing depth, but, this time, as a consequence of increasing velocity.

Figure 5.15 shows how cpu time on a RISC6000 decreases with increasing surface-point spacing for ray tracing. The horizontal asymptote at approximately 600 s is an estimate of the cost of computing the integral (sum), $\beta(x,\theta)$, in (2.15), with the time between the asymptote and the curve representing the portion of the cpu time that is attributable to the calculation of ray data.

From this picture, one can see that for rays shot from every surface-point, the ray-data cpu time is the dominant cost of the processing. For rays traced from every 10th surface-point, the ray-data cpu time is reduced dramatically to a more tolerable percentage of the total cpu time for processing data. Although the speed of the program depends on many issues—such as ray density, for example—parallel interpolation to produce ray data on the entire grid from ray data starting from a sparser set of surface-points is the main factor in the speed-up demonstrated here.

5.4 Example 4

Imaging a salt dome with overturned flanks is more challenging than were the previous two examples because we must accommodate ray trajectories near horizontal and turning. To obtain ray data on grid points from ray data on trajectories, I use the 2-D interpolation scheme described in the previous chapter, that does vertical as well as hor-

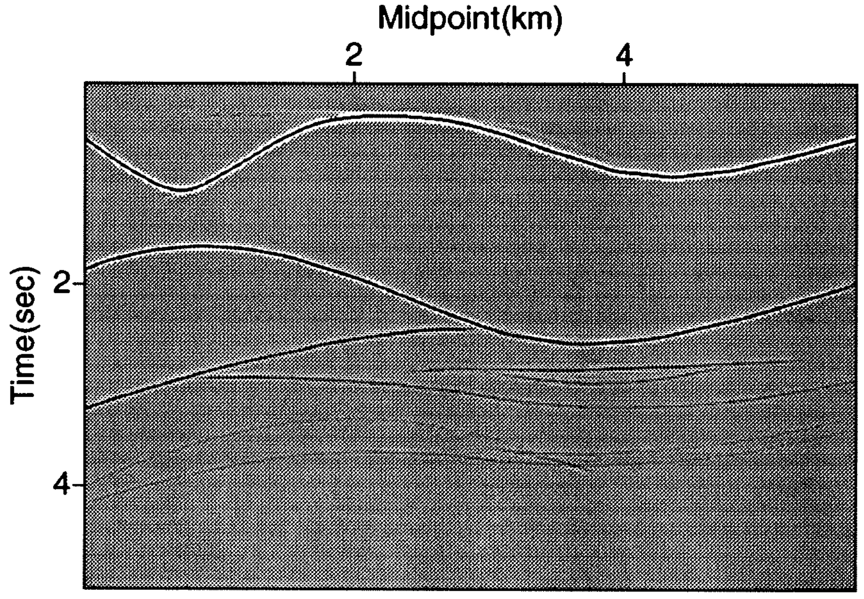

FIG. 5.11. Synthetic data generated by the Gaussian beam method (offset = 0.3 km).

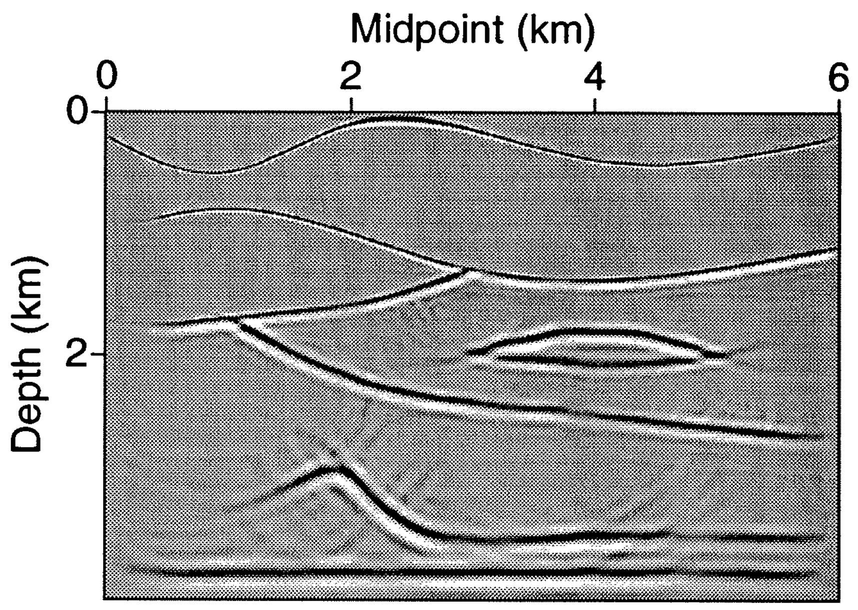

FIG. 5.12. Inversion result using ray tracing without interpolation between midpoints (offset = 0.3 km; $\triangle s$ = 0.015 km).

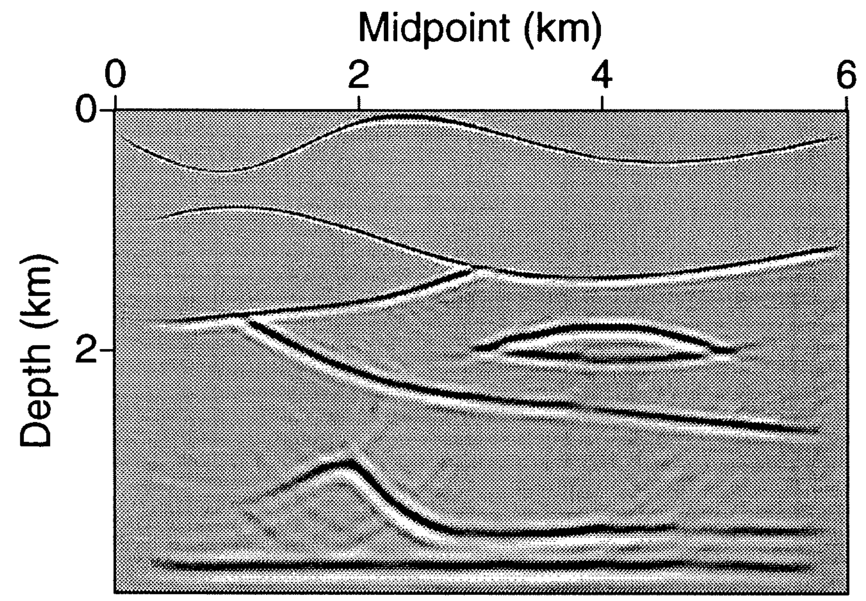

FIG. 5.13. Inversion result with interpolation ($\triangle s = 0.15$ km; skip=10).

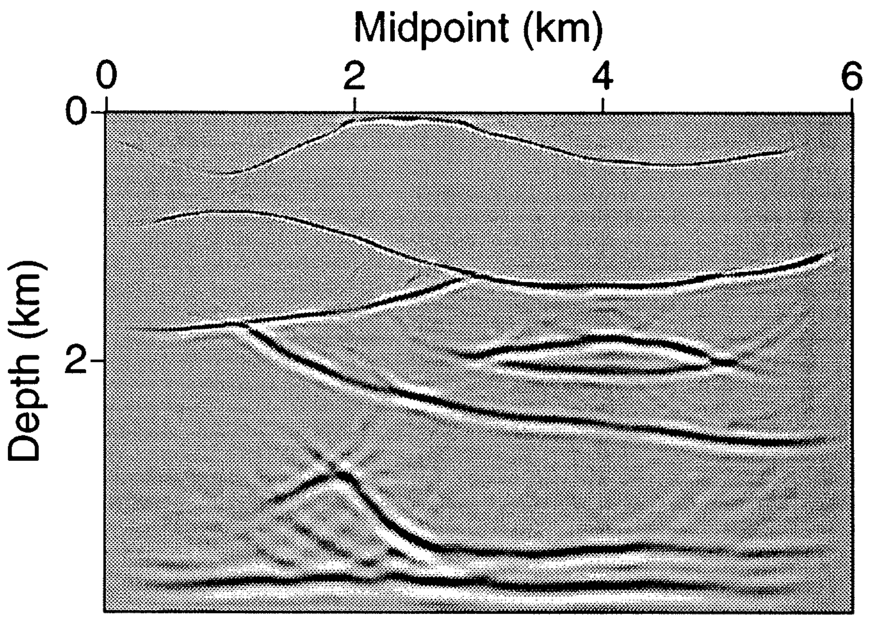

FIG. 5.14. Inversion result with large midpoint spacing ($\triangle s = 0.75$ km; skip=50).

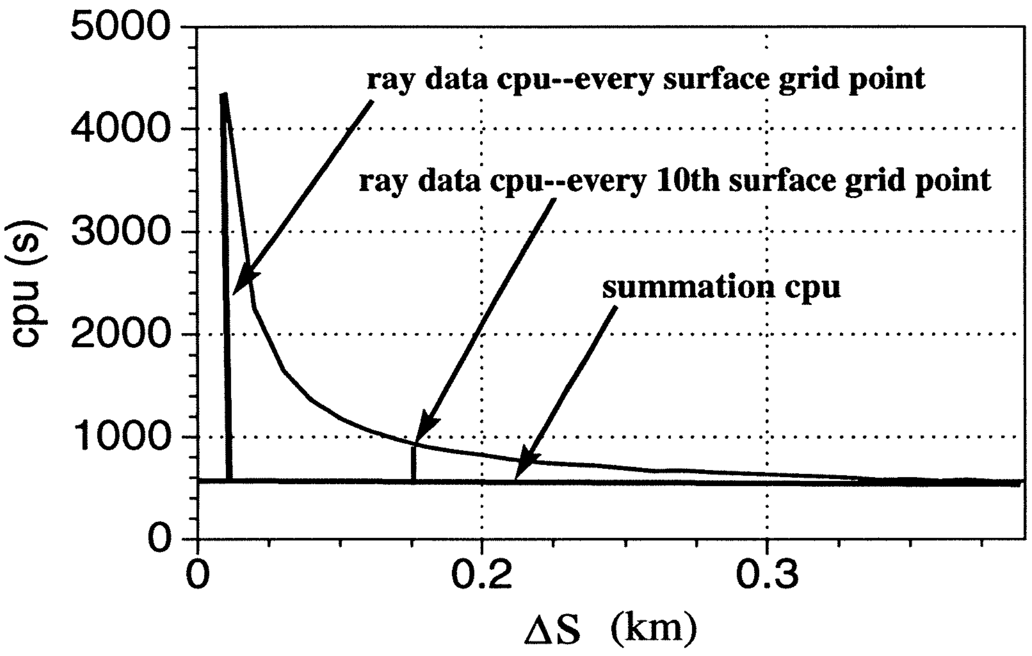

FIG. 5.15. CPU time versus the surface-point spacing for ray tracing.

izontal interpolation. With this technique, we can image reflections with dip beyond 90 degrees.

Figure 5.16 is a salt dome model for which Figure 5.17 is a common-offset synthetic data generated by GBSEIS for 400-m offset. Figure 5.18 shows the output of my code. All features in the model are imaged well, including the salt dome, the lens-shaped structure and the reflector beneath it. In particular, from Figure 5.19, which depicts the normal rays from the salt dome, we see that there are many rays near horizontal arising from the region around the overturn of the salt dome. Without the vertical interpolation option of my code, the ray data for this region would be too inaccurate to produce the image that we see in Figure 5.18.

5.5 Example 5

Figure 5.20 is a cross section of a tank model for which experimental data were gathered by the Allied Geophysical Laboratory at the University of Houston under support of Marathon Oil Company and provided to us. It is this model that motivated the synthetic model in Example 2, Figure 5.5. A major difference between the two figures is that the sawtooth reflector at about 9.5 kft deep in the figure here is sharper than in the synthetic of Example 2.

Figure 5.21 is the largest-offset physical data, and Figure 5.22 is the corresponding inversion result for that offset. I used five such common-offset physical data sets and stacked the outputs of five inversions. Figures 5.23 through Figure 5.27 show the result

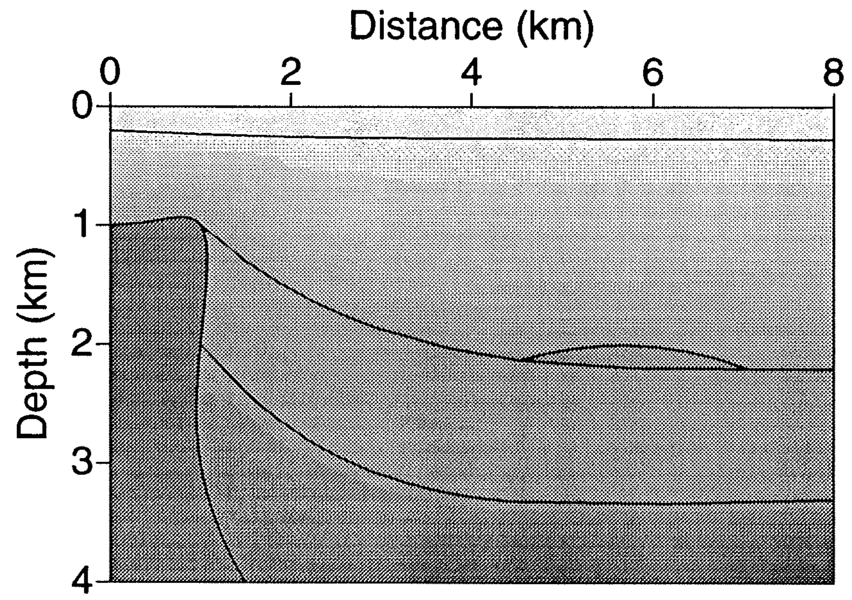

FIG. 5.16. Salt dome model.

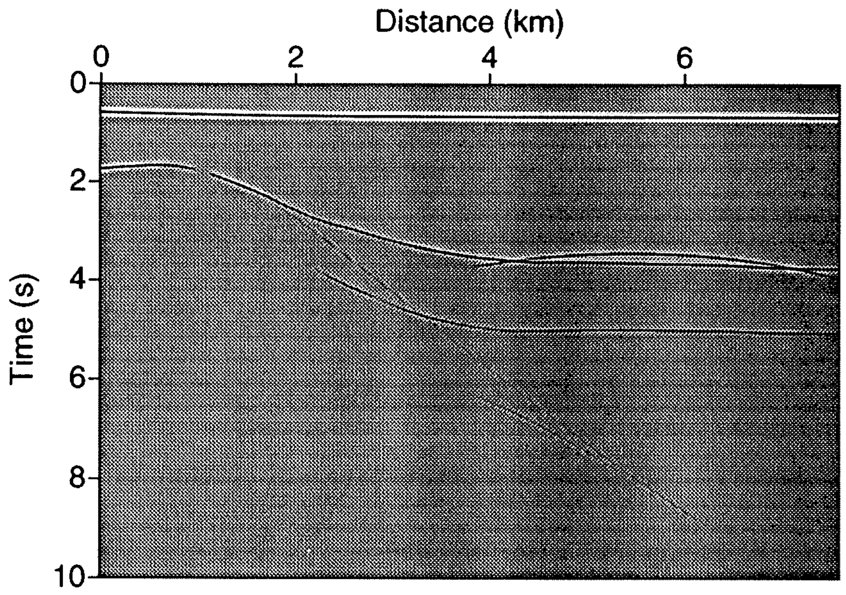

FIG. 5.17. Synthetic data with offset=0.4 km.

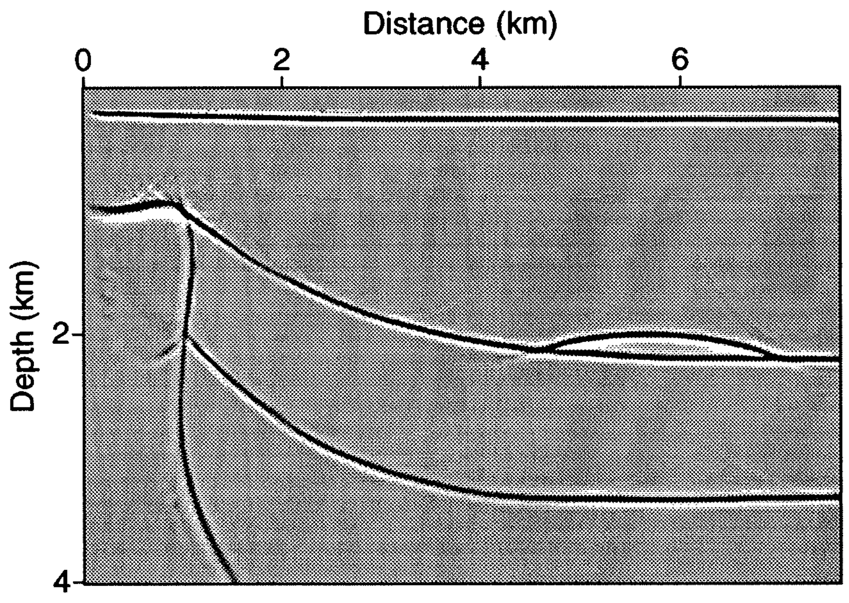

FIG. 5.18. Inversion of salt dome data.

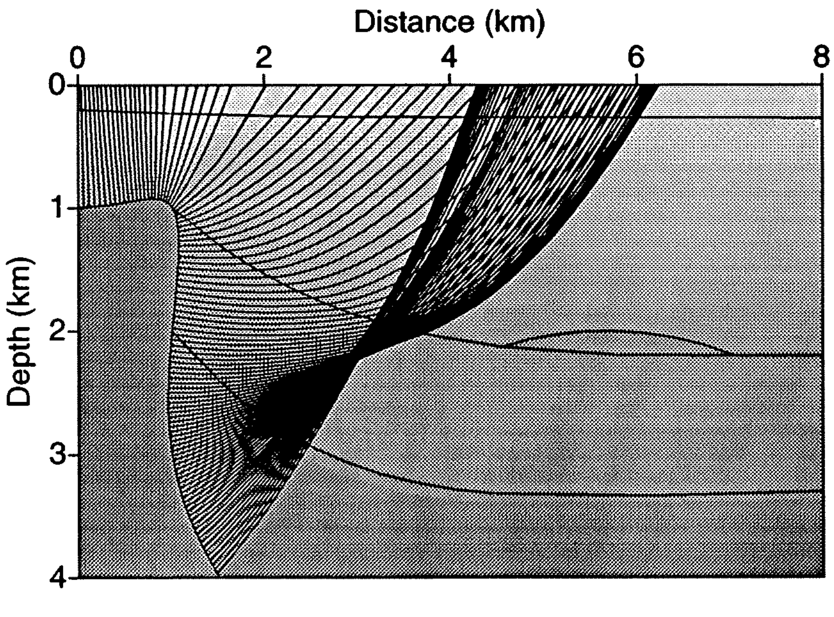

FIG. 5.19. Normal rays from the boundary of the salt dome.

of a stacking the inversion of five different offsets (880, 1680, 2480, 3280, and 4080 ft) with different choices of spatial density (skip parameter) in the ray tracing. The original spacing of midpoints was 80 ft. At Δs = 1.6 kft (skip= 20) in Figure 5.25, the fault at a depth of 7 kft and horizontal distance of 13 kft to 15 kft starts to degrade. At Δs= 2 kft (skip = 25) in Figure 5.26, the deepest (horizontal) reflector starts to degrade and some of the sawteeth in the reflector above have faded. For this example, Δs = 1.2 kft (skip=15) is the largest value of Δs for which an image is obtained that does not exhibit degradation of any reflectors.

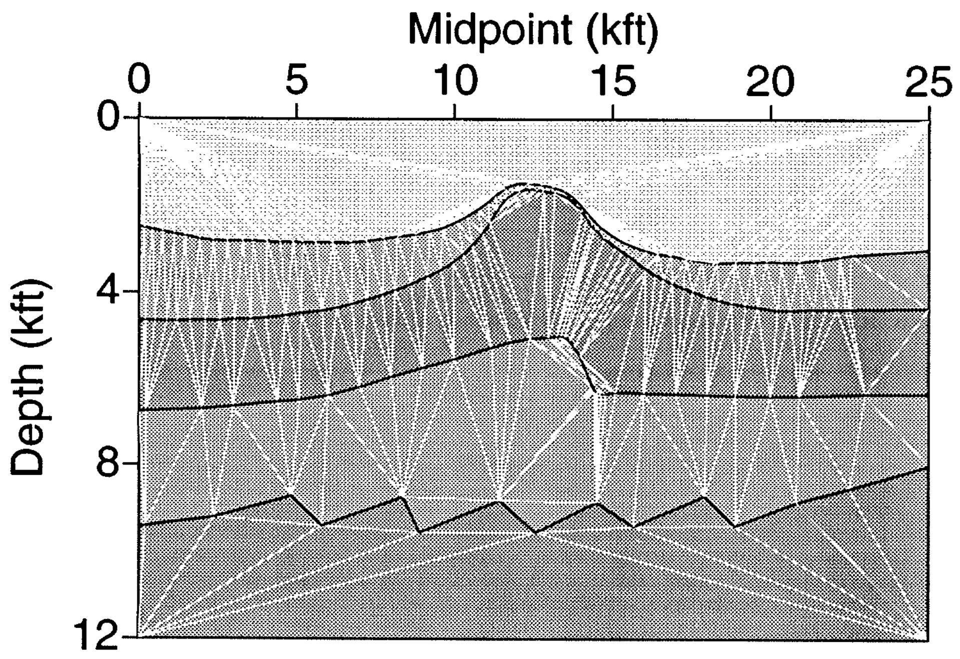

FIG. 5.20. Triangulated actual Marathon model.

5.6 Example 6

In this example, I applied my program to the Marmousi (Versteeg, 1994) data. Instead of using the correct velocity model, which consists of 160 thin layers, I used the velocity obtained by Liu's (1995) velocity analysis. Compared with the true velocity, it has just 18 layers. Figure 5.28 is a triangulated Marmousi model. Figure 5.29 is the result obtained by stacking the inversion of 19 different offset data sets. For the Marmousi data, an important challenge is the area beneath the lens-shaped structure (below depth 2.5 km, between horizontal distance of 3 km and 7 km). Since I used Liu's background model, I will compare my output with Liu's, shown in Figure 5.30. We have produced comparable images of the target zone. Liu's processing is done with a Kirchhoff migration code that uses finite difference to generate ray data. That code does not produce estimates of the reflection coefficient; it was designed to produce a reflector

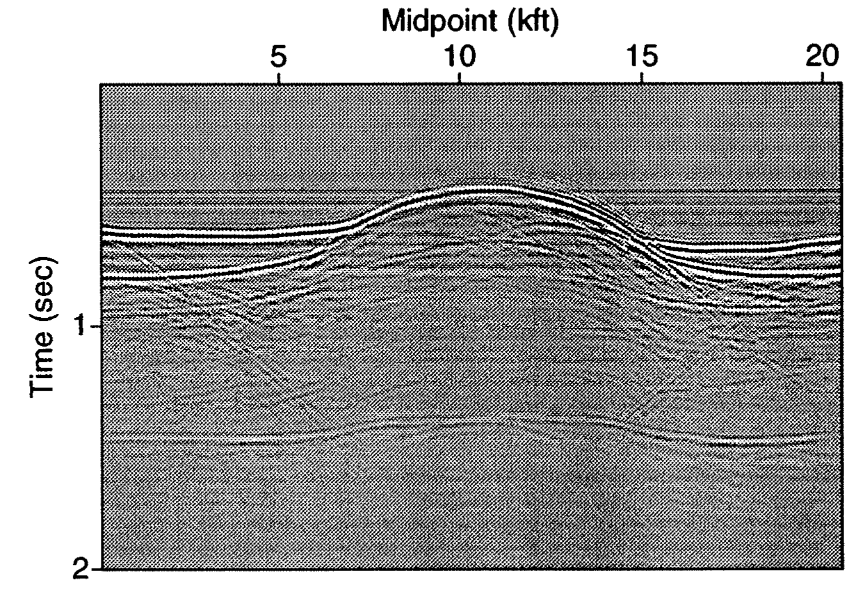

FIG. 5.21. Physical tank data with offset = 4080 ft.

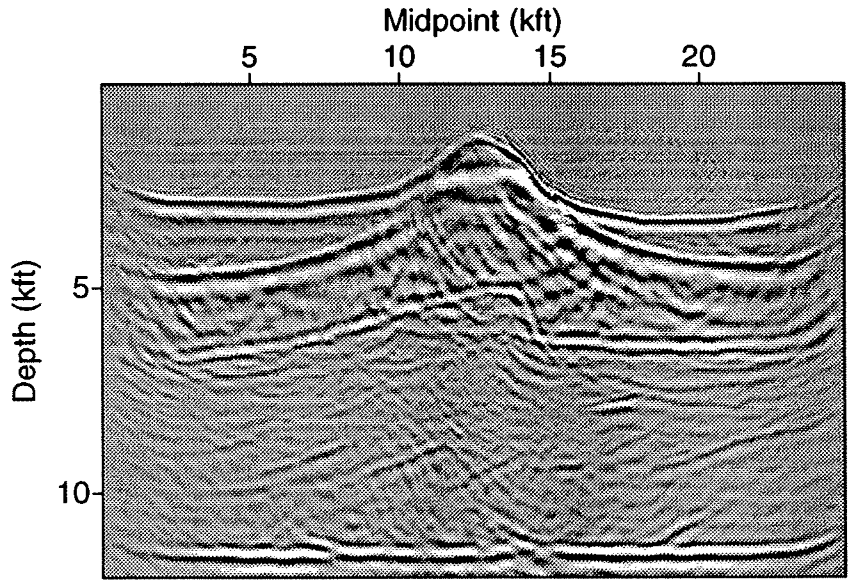

FIG. 5.22. Inversion result of physical tank data (offset = 4080 ft; $\triangle s$ = 400 ft; skip=5).

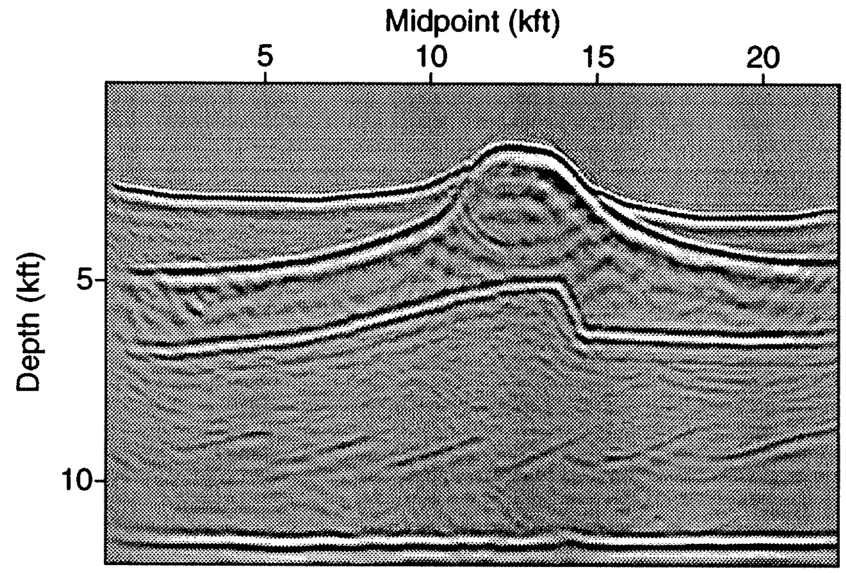

FIG. 5.23. Stacked inversion data generated from five common-offset inversions ($\triangle s$= 400 ft; skip=5).

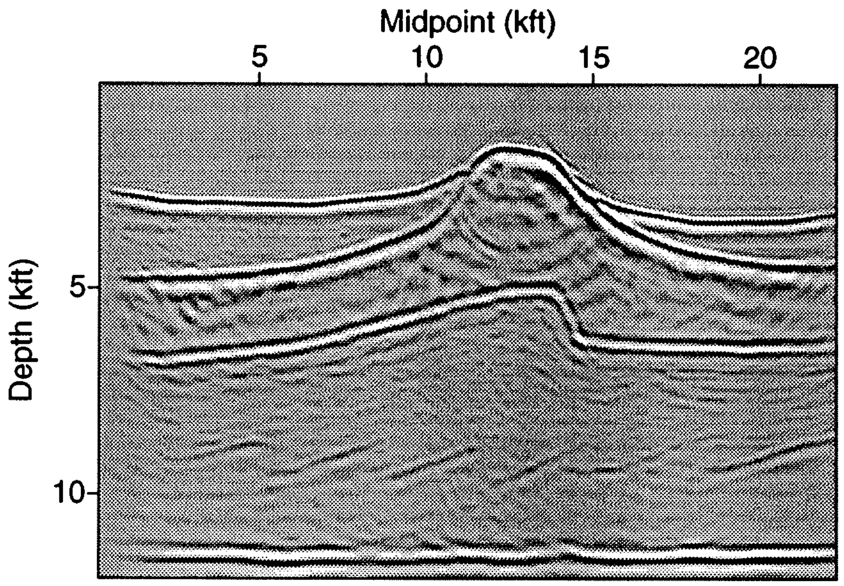

FIG. 5.24. Stacked inversion data ($\triangle s$ = 1200 ft; skip=15).

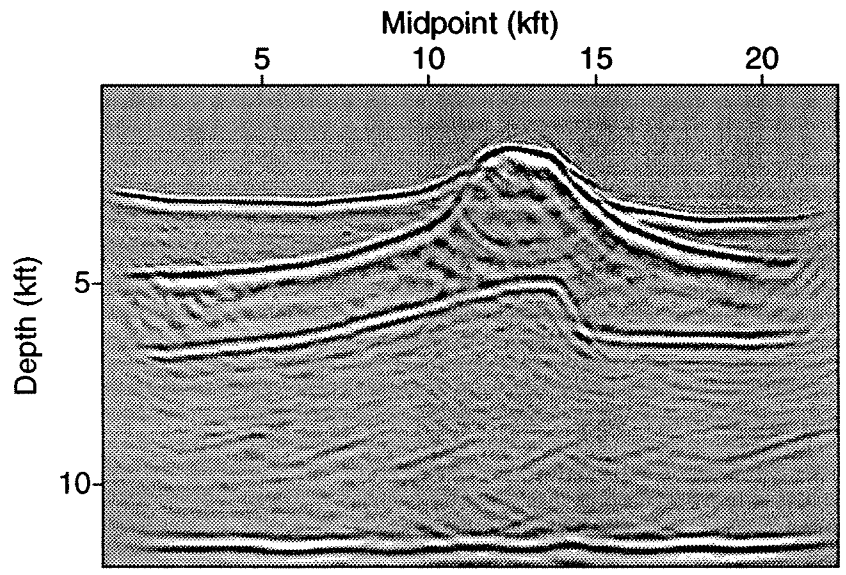

FIG. 5.25. Stacked inversion data ($\triangle s$ = 1600 ft; skip=20).

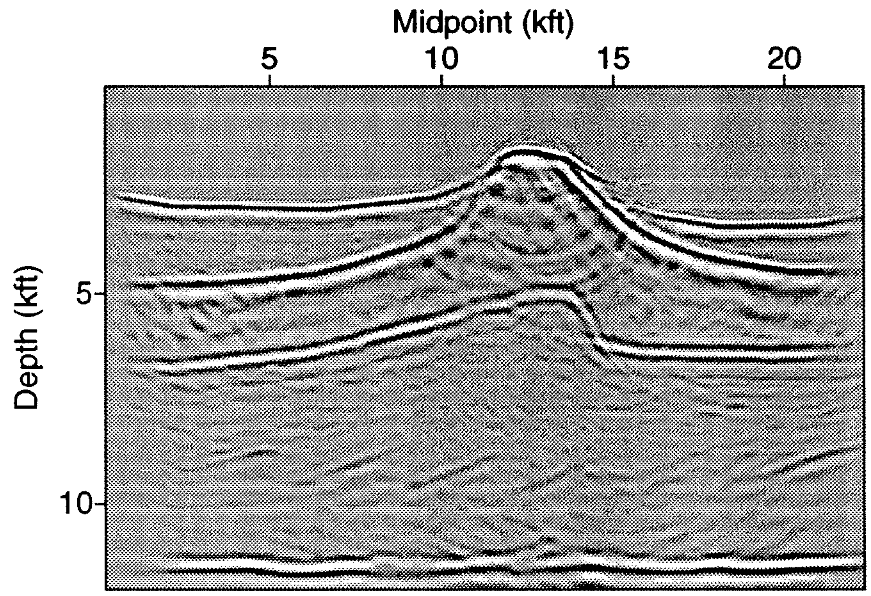

FIG. 5.26. Stacked inversion data ($\triangle s$ = 2000 ft; skip=25).

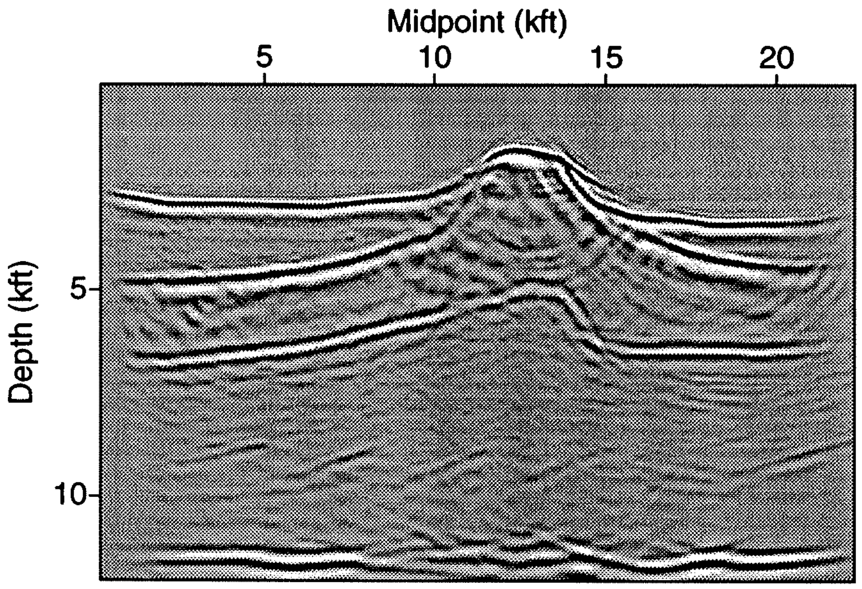

FIG. 5.27. Stacked inversion data (Δs = 2400 ft; skip=30).

map only, and was used by Liu (1995) in his recursive velocity analysis and migration method. Thus, the only purpose of this example was to provide a test of the imaging capability of my code on a complex model.

Two shortcomings of this method brought out by this example are: (1) I have not organized the program to efficiently handle multi-offset data. A desirable reorganization would involve separating out the module that generates the ray data to produce a table of the ray data from all surface-points to all subsurface points. With the ray data computed at the outset, the same ray data could then be used for each of the multiple-offset inversions. (2) Using GBMOD to build model requires digitization of all the interfaces of model, and supplying sloth in each block. If sloth model is complex, such as the Marmousi model, this entails much work. It is desirable to have some interactive model builder to reduce the effort required.

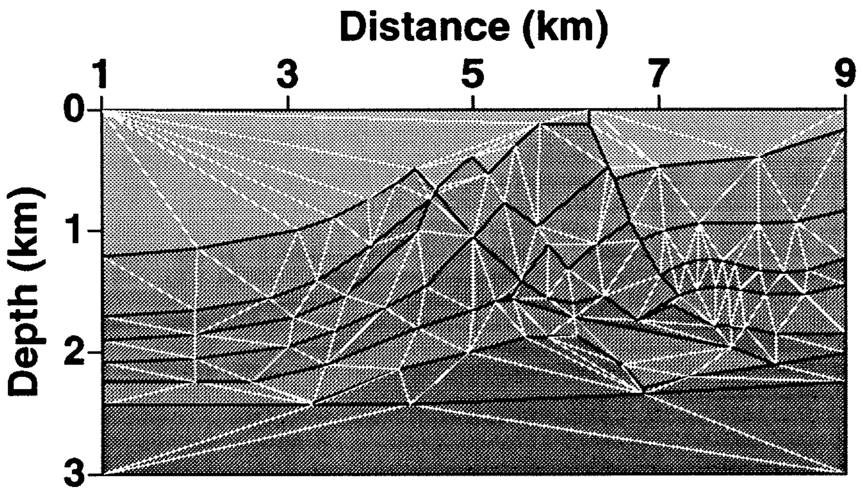

FIG. 5.28. Triangulated Marmousi Model.

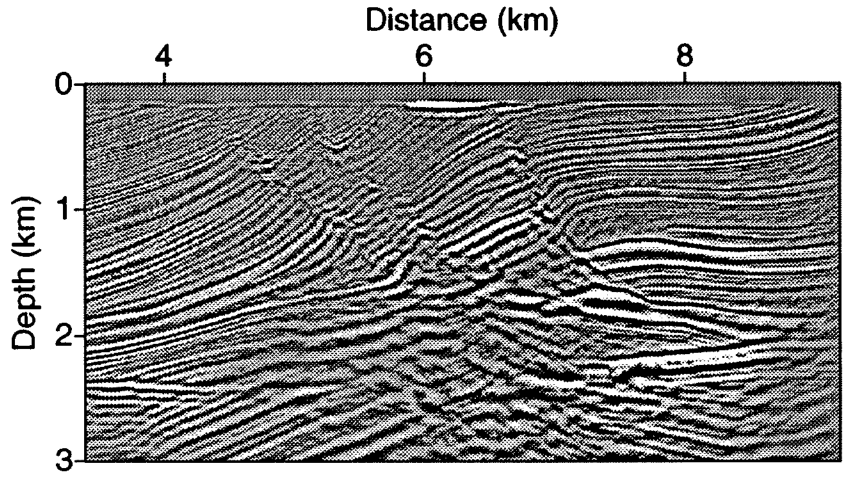

FIG. 5.29. Stacked inversion result ($\triangle s$=0.1 km; skip=4).

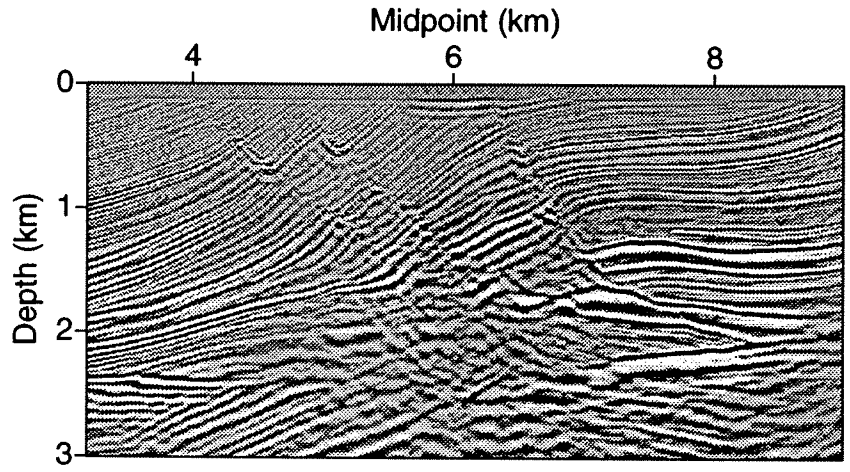

FIG. 5.30. Result of Liu.

Chapter 6

CONCLUSION

I have developed a common-offset inversion code that overcomes some of the limitations of the previous code CXZCO. It takes advantage of dynamic ray tracing in a triangulated model to accomplish 2.5-D inversion. Linear interpolation is used to obtain ray data from a ray grid to a Cartesian grid, and parallel interpolation following Liu (1993) is used for program efficiency. I also derived a 2-D interpolation scheme to estimate ray data on a Cartesian grid when the ray trajectories are nearly horizontal. With the help of 2-D interpolation, one can image steep structures, such as salt domes. The ultimate output of the inversion is the reflector map and an amplitude that is consistent with the Bleistein/Cohen inversion theory. From this amplitude we obtain an estimate of the angularly dependent reflection coefficient at the specular incidence angle and an estimate of the cosine of that angle. An advantage of a Kirchholf inversion, such as this one, is that we can image a selected part of model instead of the whole model, when our purpose is to obtain an inversion result for a particular target zone. This is in contrast to finite-difference migration or inversion, which needs to process data for the entire model. One the other hand, this method has some disadvantages. In its present form, the code does not lend itself to efficient processing of multi-offset data sets, since the ray data need to be recomputed each time. This can be easily overcome. We expect that this code will be a useful tool for seismic inversion in complex media, and can be improved to overcome its disadvantages.

REFERENCES

Rüger, A., 1993, Dynamic ray tracing and its application in triangulated media, Thesis, CWP-139.

Bleistein, N, 1986, Two-and-one-half dimensional in-plane wave propagation: Geophysical Prospecting, 34, 686-703.

Bleistein, N, Cohen, J. K., and Hagin F. G., 1987, Two and one-half dimensional Born inversion with an arbitrary reference: Geophys, 52, 26-36.

Červený, V., 1981, Computation of geometrical spreading by dynamic ray tracing: SEP-28, 61-73, Standford University.

Červený, V., 1982, Computation of wave fields in inhomogeneous media - Gaussian beam approach: Geophys. J.R. Astr. Soc., 70, 109-128.

Červený, V., 1987, Ray tracing algorithms in three-dimensional laterally varying layered structures, in Nolet, G., Ed., Seismic tomography: D. Reidel Publishing Co., 99-133.

Docherty, P., 1991, Documentation for the 2.5-D common-shot modeling program CWP-U08R, Colorado School of Mines.

Docherty, P., 1988, Documentation for the 2.5-D common shot program CSHOT: Computer program documentation CWP-U08R, Colorado School of Mines.

Docherty, P., 1989, Ray theoretical modeling, migration and inversion in two-and-one-half-dimensional layered acoustic media, CWP-051, Colorado School of Mines.

Hale, D, 1991, Dynamic ray tracing in triangulated subsurface models: CWP-107, Colorado School of Mines.

Hale, D, and Cohen, J. K., 1991, Triangulated models of the Earth's subsurface: CWP-107, Colorado School of Mines.

Hsu C-H and Liu, Z, 1991, CXZCO: A 2.5-D common offset inversion program in a c(x, z) medium: CWP-U15, Colorado School of Mines.

Liu, Z, 1995, PH.D Thesis: Migration velocity analysis CWP-168, Colorado School of Mines.

Liu, Z, 1993, A Kirchhoff approach to seismic modeling and prestack depth migration: CWP-137, Colorado School of Mines, 217-240.

Liu, Z and Norman Bleistein, 1991, Velocity analysis by inversion: CWP Project Review CWP-107, Colorado School of Mines.

Sumner, B., 1990, CZ1: Fortran program for 2.5-D stratified velocity inversion, descriptions, and instructions: Computer program documentation CWP-U06R, Colorado School of Mines.

Versteeg, R., 1994, The Marmousi experience: velocity model determination on a synthetic complex data set: The Leading Edge, **13**, 927-936.