Chapter 3

Tomographic velocity inversion

3.1 Overview

The problem of determining interval velocities is one of long-standing interest to reflection seismologists. Many different solutions have been proposed; I won't attempt to list them all here. I will instead limit myself to kinematic methods: that is, methods that rely upon travel time, rather than upon wavelet shape or amplitude. I will further restrict the scope of my study to non-zero offset methods: methods that are valid even when the shot and geophone are not close to each other. Finally, I am interested only in methods that do not involve a priori assumptions about the velocity structure or the reflector dip.

3.1.1 Traveltime tomography

The method of traveltime tomography (Bishop et al., 1985) uses traveltimes that have been picked from pre-stack reflection seismic data. These traveltimes are picked along horizons that have been specified in advance by the interpreter; each horizon has associated with it an identifying number. Thus, a set of four parameters is found for each reflection "pick": shot position x_s, geophone position x_g, travel time t, and horizon number H. These picked parameters can be used in an inversion scheme to determine the interval velocities and the depths of the picked horizons. The inversion seeks to find all velocities and all horizons simultaneously rather than layer by layer; such an approach may be defined to be tomographic. The main disadvantage of this method is that the traveltimes must be picked by hand; there are also problems when the reflections change character or when there are faults or other discontinuities along the horizons.

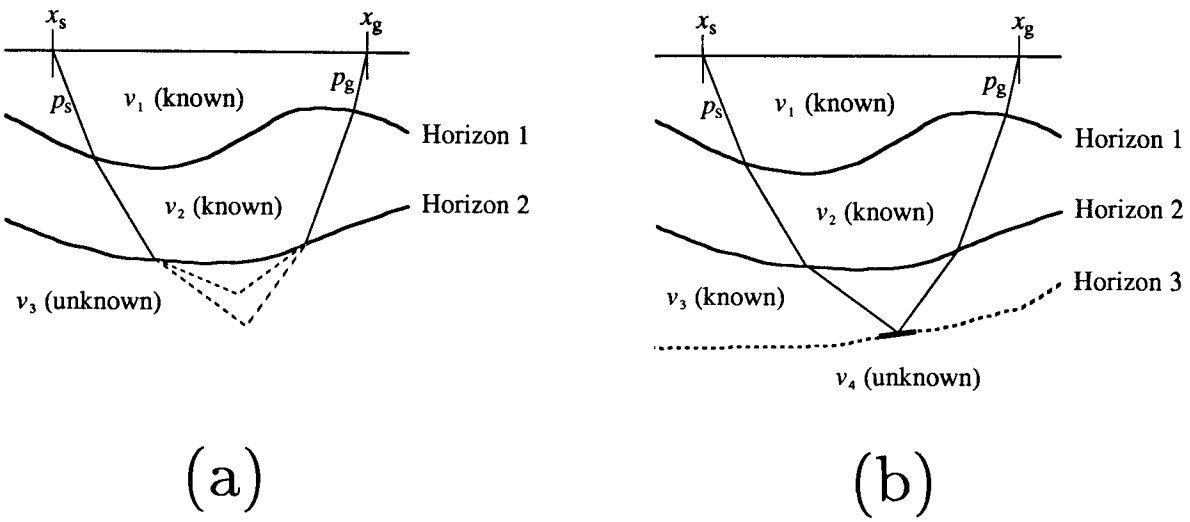

Figure 3.1: The layer-stripping method (method of reciprocal points). Diagram (a) shows how rays are traced through known layers with velocities v_1 and v_2. These rays, based on picked reciprocal parameters, are then used in a CDR velocity analysis to determine the unknown velocity v_3. Diagram (b) shows how, once v_3 is known, migrated dip bars can be used to determine the position of Horizon 3. Once the position of this horizon is known, ray-tracing can be used to find a deeper velocity, v_4.

3.1.2 Layer stripping

Traveltime tomography is an expensive approach. Less expensive, and conceptually simpler, is the method of layer stripping, known in the Soviet Union as the method of reciprocal points. This method was first invented in the U.S.S.R. (Nevinnii and Urupov, 1976; Glogovskii et al., 1977), and subsequently re-invented in the United States (Gray and Golden, 1983; Inderwiesen, 1986).

The input parameters to the layer-stripping method are the reciprocal parameters (shot position x_s, geophone position x_g, shot ray parameter p_s, geophone ray parameter p_g, and traveltime t). These can be picked according to any of the picking methods described in the previous chapter. Once these parameters are picked, their associated CDR velocities and reflector positions are found according to equations (2.25), (2.26), and (2.27). From these results, the position of the first (uppermost) reflector, and the velocity of the first layer, are determined. Rays associated with deeper reflectors are then traced through the first layer, down to the depth of the first reflector (see Figure 3.1). From there, they are

propagated into the unknown layers below. Again, a CDR velocity analysis is carried out, but the formula is more complicated than in equation (2.27), since the bending of rays as they cross the first reflector must now be accounted for. The position of the second reflector, and the velocity of the layer overlying it, are then determined.

The process of layer stripping can be carried as deep as the data will allow. This process can be implemented on an interactive work station, allowing an interpreter to eliminate unsuitable picks, and to choose suitable velocities and horizons. Non-interactive systems (Kopilevich et al., 1986) can also be used.

One difficulty with the layer-stripping technique is that the accuracy of the positions and velocities of the lower layers depends on how accurately the positions and velocities of the upper layers are determined. Ideally, the results of imaging the lower layers should be used to help correct the upper layers of the model.

3.1.3 CDR tomography

I have called the method that I am investigating "CDR tomography". Like the layer-stripping technique, it makes use of picked reciprocal parameters (x_s, x_g, p_s, p_g, and t). Like traveltime tomography, it attempts to determine the velocities in all layers simultaneously. Unlike either of these two techniques, CDR tomographic inversion is not based on the assumption that seismic velocity and reflector position are related. The CDR tomographic approach was first suggested by Harlan and Burridge (1983); I will describe it in detail below.

3.1.4 Other approaches

Other approaches to interval velocity determination exist that meet the criteria listed at the beginning of this chapter: they rely on traveltime, they work at non-zero offsets, and they are not based on any a priori assumptions about the reflectors or the velocity structure. An approach that is similar to the layer-stripping technique, but that uses only one ray parameter (p_g), has been proposed by Reshef and Kosloff (1984). Other approaches do not involve picking data at all (Fowler, 1986; Biondi, 1987; Etgen, 1987). I will not discuss these techniques here. They show great potential, but they have not yet been tested in practice.

3.2 The CDR tomographic objective function

The input data to the CDR tomographic method consist of the picked reciprocal parameters, x_s, x_g, p_s, p_g, and t, for each reflection event. The goal of CDR tomography is to find the velocity model that best fits, or is most consistent with, these picked data. To measure this fit, an objective function (a function that describes how well the model matches the data) is used.

Given the picked parameters and a proposed velocity model, there are several different objective functions that could be used. Some of these are discussed below.

3.2.1 An unstable objective function

The most straightforward objective function is as follows: rays, with ray parameters p_s and p_g, are traced from points x_s and x_g respectively. The tracing continues until the rays meet. The total predicted traveltime is calculated; the objective function consists of the squared difference between the predicted and observed traveltimes.

This straightforward method is not the one that I use. Figure 3.2 illustrates two of the reasons why. Suppose that the velocity model happens to be correct. Since the model is correct, the objective function should be at a minimum. Suppose also that x_s and x_g are near each other, and that there are slight measurement errors in p_s and p_g. The solid line in Figure 3.2 shows the actual raypath in the medium, and the dotted line shows the raypath that is predicted on the basis of the erroneous measurements of p_s and p_g. The measured traveltime is based on the travel time along the solid line, while the predicted traveltime is based on the travel time along the dotted line. As a result, therefore, of a small error in measuring the ray parameters, there is a large error in the predicted traveltime, which in turn incorrectly suggests that there is a large error in the velocity model. The first problem, then, is that small unavoidable ray-parameter errors can lead to large errors in velocity measurement.

The second problem has to do with the way the velocity model is updated. In tomographic problems, the velocity is typically updated along raypaths where a velocity error has been detected. This means, for instance, that in the situation shown Figure 3.2, the velocity will be updated not only from the surface to the reflector depth, as it should be, but also from the reflector down to the point where the two erroneous rays intersect. It appears, then, that traveltime error is not a good measurement to use in the objective

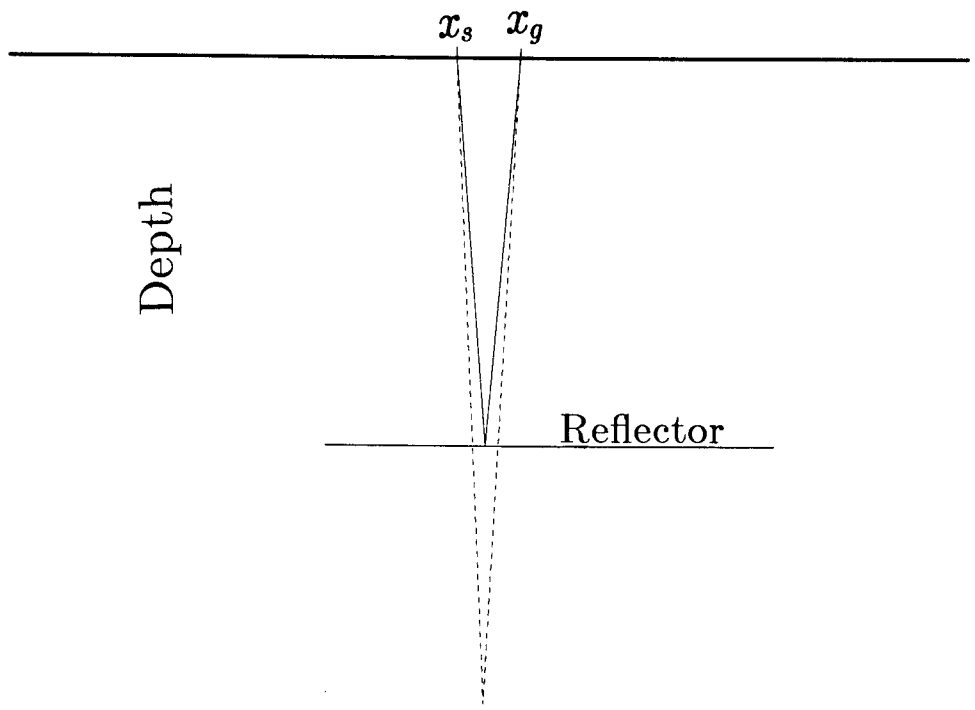

Figure 3.2: The result of small errors in p_s and p_g. When x_s and x_g are near each other, small errors in the measurement of the ray parameters can lead to large errors in the predicted traveltime. Here, the solid line shows the actual raypath in the medium, while the dotted line shows the raypath computed on the basis of slightly erroneous ray-parameter measurements.

function.

There is yet another reason for not trying to minimize errors in traveltime. In tomographic inversion problems, one does not use an objective function that minimizes, for instance, the discrepancy in x_s, the shot position. The shot position is assumed to be known exactly. Instead, one tries to minimize the discrepancy in another parameter (t, perhaps) that is more likely to have measurement errors. Similarly, when the five-parameter CDR data sets described in this paper are being used, there is no sense in trying to minimize the discrepancies in the parameters x_s, x_g, or t, which are assumed to be known relatively precisely. The objective function should be based on minimizing discrepancies in the ray parameters p_s and p_g, not on minimizing discrepancies in the measured versus the predicted traveltimes.

This last reason can be stated mathematically. Suppose that the objective function Φ

is described according to the equation

$$\Phi = \frac{1}{\sigma_{xs}^2}\sum_j(\hat{x}_{s(j)} - x_{s(j)})^2 + \frac{1}{\sigma_{xg}^2}\sum_j(\hat{x}_{g(j)} - x_{g(j)})^2 + \\ + \frac{1}{\sigma_{ps}^2}\sum_j(\hat{p}_{s(j)} - p_{s(j)})^2 + \frac{1}{\sigma_{pg}^2}\sum_j(\hat{p}_{g(j)} - p_{g(j)})^2 + \frac{1}{\sigma_t^2}\sum_j(\hat{t}_{(j)} - t_{(j)})^2, \tag{3.1}$$

where the parameters with the hats ($\hat{x}_{s(j)}$, etc.) are the predicted data values for a particular velocity model **v**. The hatless parameters are the measured data values, the σ terms are weighting terms, and j gives the measurement number. Each σ term is set to be proportional to the expected standard deviation (measurement error) for that parameter. If some parameters have small measurement errors, their weights ($1/\sigma^2$) are large. As the errors get smaller, these weights approach infinity and become constraints. Thus, if t were considered much less reliable than any of the other parameters, then equation (3.1) would become

$$\Phi = \frac{1}{\sigma_t^2}\sum_j(\hat{t}_{(j)} - t_{(j)})^2, \tag{3.2}$$

with the constraints

$$\begin{aligned} x_{s(j)} &= \hat{x}_{s(j)}, \qquad x_{g(j)} = \hat{x}_{g(j)}, \\ p_{s(j)} &= \hat{p}_{s(j)}, \qquad p_{g(j)} = \hat{p}_{g(j)}, \end{aligned} \tag{3.3}$$

for all j's. Thus, the objective function is based on discrepancies in the least reliable parameter.

3.2.2 An expensive objective function

For the above reasons, it seems likely that an objective function based on minimizing errors in p_s and p_g would be more stable that one based on minimizing errors in traveltime. There are difficulties, however, in basing an objective function directly on such errors. I will not expand on this, but it should not be too difficult to see that to measure these errors, an excessive amount of ray tracing would have to be performed. For a given velocity model, different trial ray parameters would be tried, until a pair was found that produced rays whose total computed traveltime to the point of intersection equaled the measured traveltime. Such testing of different trial ray parameters would be expensive.

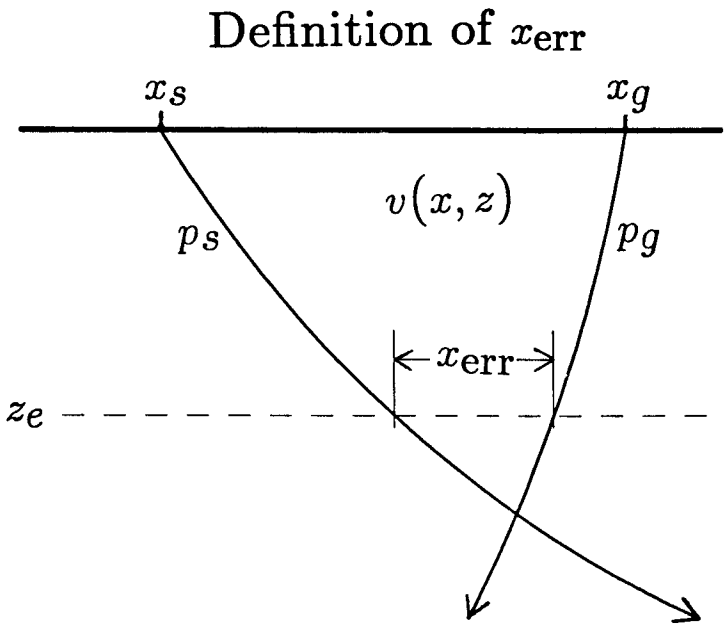

Figure 3.3: A better objective function. Two rays are traced down to depth z_e, the depth where their combined computed traveltime equals the measured (picked) travel time. The distance between the rays at this depth is defined to be x_{err}. An objective function based on x_{err} is reasonably stable in the presence of errors in the measured ray parameters.

3.2.3 A stable objective function

I have chosen to use a different objective function, one that is neither overly sensitive to ray-parameter errors, nor expensive to compute.

Suppose that the up-going and down-going rays have both been traced down to some depth z in the velocity model (see Figure 3.3). Since the starting positions and the take-off angles (ray parameters) of both rays are known, the raypaths are unique. The *predicted traveltime* at depth z can be defined as the computed traveltime along the down-going ray, plus the computed traveltime along the up-going ray, as measured at depth z. Then, for each pair of rays one can determine a depth z_e, the depth where the predicted traveltime equals the observed traveltime. Once the value of z_e has been determined, the horizontal distance between the two rays at that depth can be measured; this distance is defined to be x_{err}. The objective function to be minimized is x_{err}^2. The value of x_{err} is clearly zero if the ray parameters and traveltime have been measured correctly and if the velocity model has been chosen correctly. If the ray parameters p_s and p_g have been measured incorrectly, x_{err} will be non-zero, but it won't vary disproportionately, even when the shot-geophone distance is small (compare this behavior to that of the objective function

based on traveltime errors). The rays need be computed only once per iteration; no expensive two-point ray-tracing algorithm is necessary.

It is clear that when the shot-geophone offset is small, x_{err} is stable relative to small fluctuations in p_{s} and p_{g}. It might appear, though, that x_{err} is unstable when the offset is large, or the raypaths are nearly horizontal. It can be shown, however, for the constant-velocity case, that dx_{err}/dp is, within a factor of 2, the same for large and small offsets. This result may seem counter-intuitive until one remembers that a change in p results not only in a change in raypath, but in a change in z_{e} as well. These two changes partially cancel each other, resulting in a smaller change in x_{err} than one might expect.

In theory, the velocity model used in the inversion can be arbitrary. It can consist of layers, or basis functions, or some combination of these, or it can consist of a grid of velocity values. I have chosen to use a velocity model where the velocity is defined at regularly spaced grid points. In Appendix C are given the details of how ray tracing may be accomplished in such a model.

3.3 Minimizing the objective function

Once an objective function has been chosen, a way of finding its minimum must be devised. Some definitions are necessary before proceeding further:

The model is defined to consist of N boxes. The velocity in box i is v_i (there may be a horizontal velocity gradient in each box as well; see Appendix C). The vector $\mathbf{v}$ is made up of all N of the velocity terms v_i.

The data consist of M sets of picked parameters; the jth set of parameters contains $p_{\mathrm{s}(j)}$, $p_{\mathrm{g}(j)}$, $x_{\mathrm{s}(j)}$, $x_{\mathrm{g}(j)}$, and $t_{(j)}$.

For the jth set of picked parameters, there is an associated error term $x_{\mathrm{err}(j)}$. This term is found, as described in the previous section, by tracing rays. It is clearly a function of $\mathbf{v}$, and it is conveniently written as $x_{\mathrm{err}(j)}(\mathbf{v})$. The vector composed of all M of the error terms is written as $\mathbf{x}_{\mathrm{err}}$. The objective function Φ, then, is

$$\Phi(\mathbf{v}) \equiv \sum_j x^2_{\mathrm{err}(j)}(\mathbf{v}) = \|\mathbf{x}_{\mathrm{err}}(\mathbf{v})\|^2. \tag{3.4}$$

3.3.1 Damping

To prevent the velocity from varying too wildly in each iteration, it is necessary to include damping factors. Since the damping should be based on the overall velocity model, rather than on the change in velocity at a single iteration step (Tarantola and Valette, 1982), the damping factors are incorporated into the objective function. As a result, the objective function becomes

$$\Phi(\mathbf{v}) \equiv \sum_j x_{\mathrm{err}(j)}^2 + \lambda_x^2 \sum_i \left(\frac{\partial v_i}{\partial x}\right)^2 + \lambda_z^2 \sum_i \left(\frac{\partial v_i}{\partial z}\right)^2, \tag{3.5}$$

where i is the index over model velocities, j is the index over picked parameters, and λ_x and λ_z are the horizontal and vertical damping terms, respectively.

This damping function is not entirely arbitrary. It was chosen with the idea that if the velocity in a region is unknown, then that velocity should be constant. This is a physically sensible constraint; in practice, it works better than the assumption that there is a linear velocity gradient in depth. The damping function and some of its problems are discussed in more detail in section 3.3.3 below.

3.3.2 The Gauss-Newton method

The minimization problem is easily stated: find the value of $\mathbf{v}$ that minimizes $\Phi(\mathbf{v})$. This is a non-linear minimization problem. From the many possible ways to solve such problems (Gill et al., 1981), I have chosen to use a modified Gauss-Newton method. Bishop et al. (1985) use a similar iterative method.

In this iterative method, rays are traced through the current best guess of the velocity model. The value of the objective function is found, as are differential values that describe how the raypaths (and thus the objective function) change with respect to small changes in velocity. The results are used to set up a linearized least-squares inversion problem. This least-squares problem is solved by an iterative conjugate-gradient method, and the solution is used to update the velocity model. Rays are then traced in the new velocity model, thus re-linearizing the problem, and the sequence of iterations continues. Note that there are two sorts of iterations: *internal* iterations, to solve the linear least-squares problem, and *external* iterations, which involve tracing rays through the velocity model.

Let $\mathbf{v}^{(k)}$ be the current velocity model, the result of the kth external iteration. The

desired new velocity model is $\mathbf{v}^{(k+1)}$, which is to be found according to the formula

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + \Delta\mathbf{v}. \tag{3.6}$$

In the Gauss-Newton method $\Delta\mathbf{v}$ is found by solving the least-squares system

$$\underset{\sim}{\mathbf{A}}^{(k)}\Delta\mathbf{v} \approx -\mathbf{x}_{\text{err}}^{(k)}, \tag{3.7}$$

where $\mathbf{x}_{\text{err}}^{(k)} \equiv \mathbf{x}_{\text{err}}(\mathbf{v}^{(k)})$, $\underset{\sim}{\mathbf{A}}^{(k)}$ is the Fréchet matrix derived from $\mathbf{x}_{\text{err}}^{(k)}$, and $\Delta\mathbf{v}$ is the difference between the current velocity model and the desired model. The Fréchet matrix is defined according to the formula

$$A_{ij}^{(k)} \equiv \left(\frac{\partial x_{\text{err}(j)}}{\partial v_i}\right)_{\mathbf{v}=\mathbf{v}^{(k)}}. \tag{3.8}$$

This matrix can be obtained through a finite-differencing method: perturb each velocity term v_i slightly, then see how all the components of $\mathbf{x}_{\text{err}}$ vary in response. In practice, this method is too expensive; alternative methods are described in Appendix D for gridded velocity structures.

A few details must be added to this general description. Since the objective function contains damping terms (see equation (3.5)), these terms must be incorporated into the iterations. A differencing matrix $\underset{\sim}{\mathbf{D}}$ is defined such that

$$\underset{\sim}{\mathbf{D}}\begin{pmatrix} v_1 \\ \vdots \\ v_N \end{pmatrix} \approx \left(\lambda_x\frac{\partial v_1}{\partial x},\ldots,\ \lambda_x\frac{\partial v_N}{\partial x},\ \lambda_z\frac{\partial v_1}{\partial z},\ldots,\ \lambda_z\frac{\partial v_N}{\partial z}\right)^T. \tag{3.9}$$

The terms λ_x and λ_z have the same meaning as in equation (3.5). I use a simple two-point differencing operator, where

$$\frac{\partial v_x}{\partial x} \approx \frac{v_{x+1} - v_x}{\Delta x}. \tag{3.10}$$

When damping is included, the least-squares system no longer takes the form of equation (3.7), but instead is written as

$$\begin{pmatrix} \underset{\sim}{\mathbf{A}} \\ \underset{\sim}{\mathbf{D}} \end{pmatrix}\Delta\mathbf{v} \approx \begin{pmatrix} -\mathbf{x}_{\text{err}} \\ -\underset{\sim}{\mathbf{D}}\mathbf{v} \end{pmatrix}. \tag{3.11}$$

So long as λ_x and λ_z are non-zero, equation (3.11) represents a sparse over-determined system of linear equations, even if M is less than N, as is the case with my synthetic

examples. Equation (3.11) can be solved numerically by the conjugate gradient subroutine LSQR (Paige and Saunders, 1982).

The LSQR subroutine must be applied with caution. It is tempting to stop LSQR after only a few internal iterations, but this choice should be avoided. The reason is that there are problems with the $\underset{\sim}{\mathbf{D}}$ matrix used in this inversion; these problems will be discussed in the next section. If LSQR is halted prematurely, the damping terms do not take full effect, the minimization condition on equation (3.5) is not fulfilled, and as a result the solution does not converge quickly. Bishop et al. discovered a similar effect when they terminated their least-squares solver prematurely (Cutler et al., 1985).

An additional complication is that equation (3.6) should actually be written in the form

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + \alpha \Delta \mathbf{v}, \tag{3.12}$$

where α is the scalar that minimizes $\Phi(\mathbf{v}^{(k)} + \alpha \Delta \mathbf{v})$. If Φ behaves quadratically, α will be close to 1.0. In other cases, however, setting α equal to 1.0 may not produce good results. In practice, it is expensive to find α exactly, since ray tracing must be performed every time Φ is calculated. I search for α by selecting some trial values of α, calculating Φ for these values, and fitting a polynomial to find the minimum of $\Phi(\alpha)$. More details are found in Appendix B.

In theory, the external iterations could proceed indefinitely. I have obtained adequate results, however, by stopping the iteration process when Φ decreases by less than 1% between external iterations.

It is possible to apply weighting to the data, since some sets of reciprocal parameters may be considered more reliable than others. The easiest sort of weighting is by amplitude—the higher the amplitude of the picked peak, the more weight is given to the picked reciprocal parameters. It is useful, in addition, to apply weighting that is inversely proportional to z_e (z_e is defined in section 3.2.3). This inverse-z_e weighting is intended to compensate for the increase of x_{err} with depth.

The first part of Appendix D contains the details of how the Fréchet matrix $\underset{\sim}{\mathbf{A}}$, defined in equation (3.8), is calculated when the velocity model is laterally homogeneous. One interesting result is that A_{ij} in such a medium depends only on $p_{s(j)}$, $p_{g(j)}$, v_i, and $v_{I(j)}$, where $v_{I(j)}$ is the velocity in layer $I(j)$, the layer containing the endpoints of the jth pair of rays. This result helps make the problem a little bit more linear, since it means that A_{ij} does not depend on v_l, unless $l = i$ or $l = I(j)$.

Finding the Fréchet matrix is a more difficult task when v is a function of both x and z. Over and above the expected difficulties of computing derivatives in a layer containing a horizontal gradient, there is an additional problem. It has just been noted that in laterally homogeneous media, A_{ij} depends only on v_i and $v_{I(j)}$, where $I(j)$ is the index of the layer containing the endpoints of the jth ray. When the medium is laterally inhomogeneous, however, A_{ij} depends on the velocities in all the boxes that the ray passes through on its way from the ith box to the endpoint. As a consequence, calculating A_{ij} is more complicated than for the flat-layer case. The details of how these calculations are carried out are found in the second part of Appendix D.

3.3.3 Problems with damping

A damping operator $\underset{\sim}{\mathbf{D}}$ is defined in equations (3.9) and (3.10). Equation (3.5), which defines the damped objective function, can now be rewritten in the form:

$$\Phi(\mathbf{v}) = \mathbf{x}_{\text{err}}^{\text{T}}\mathbf{x}_{\text{err}} + \mathbf{v}^{\text{T}}\underset{\sim}{\mathbf{D}}^{\text{T}}\underset{\sim}{\mathbf{D}}\mathbf{v}. \tag{3.13}$$

According to the theory of the maximum-likelihood approach (Menke, 1984, pp. 92–96), any choice of $\underset{\sim}{\mathbf{D}}$ implies an a priori knowledge of the covariance in the velocity model. More specifically,

$$\underset{\sim}{\mathbf{D}}^{\text{T}}\underset{\sim}{\mathbf{D}} = (\text{cov}\,\mathbf{v})^{-1}, \tag{3.14}$$

where cov $\mathbf{v}$ is the a priori covariance in the velocity model $\mathbf{v}$. The covariance matrix implied by my choice of $\underset{\sim}{\mathbf{D}}$ may not be the best one.

One problem with my choice of $\underset{\sim}{\mathbf{D}}$ is that it is singular. Figure 3.4 illustrates the problem; it shows the one-dimensional frequency-domain version of the damping function. At zero frequency, the transformed damping function goes to zero. As a result of this zero value, $\underset{\sim}{\mathbf{D}}$ is singular. It should be noted, however, that although $\underset{\sim}{\mathbf{D}}$ is singular, the inversion itself is not necessarily poorly conditioned; during inversion the $\underset{\sim}{\mathbf{D}}$ matrix is combined with an $\underset{\sim}{\mathbf{A}}$ matrix, as shown in equation (3.11).

A possible solution to the singularity problem is to add some small constant ϵ to the function in Figure 3.4, thus lifting the zero point to some value other than zero. Adding this constant would be equivalent to adding the term $\epsilon\mathbf{v}^{\text{T}}\mathbf{v}$ to equation (3.13). It would pull the velocity in any poorly determined region toward zero; the effect would be most pronounced at low spatial frequencies, where ϵ is large compared to the original value of the function. This effect is neither desirable nor physically meaningful.

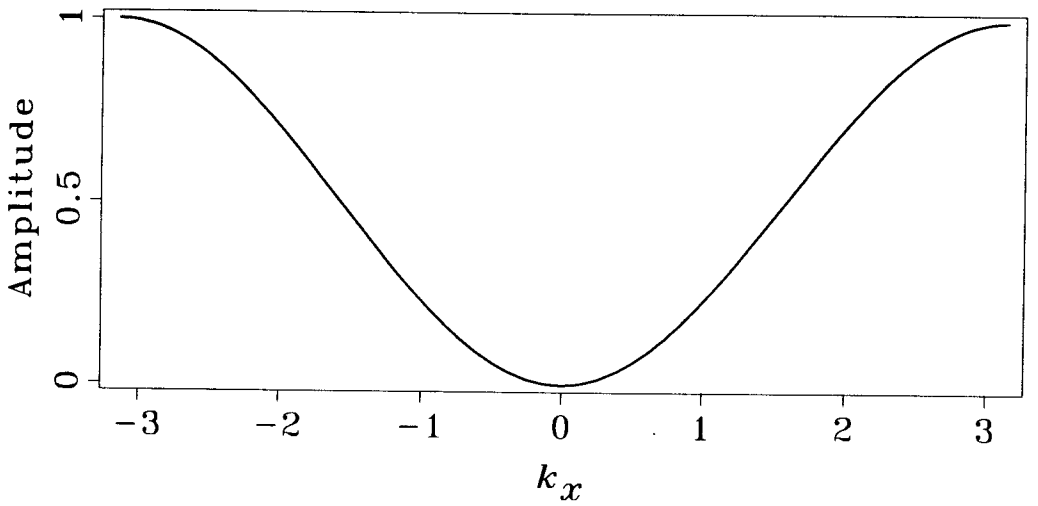

Figure 3.4: The damping function (frequency domain). In the frequency domain, in one dimension, the damping function in equation (3.5) take the form shown in this figure. Near zero frequency, the damping function approaches zero value; this behavior leads to problems with conditioning.

A second problem with my choice of $\underset{\sim}{\mathbf{D}}$ is that it represents a very localized two-point differentiation operator. In order for the entire velocity model to feel the effects of the damping function, the LSQR routine must perform enough iterations so that the damping can propagate out from the areas of known velocity to areas of unspecified velocity. The speed of propagation is inversely proportional to the locality of the operator, so that the two-point differentiation operator requires many iterations to take full effect.

A solution to the problem of operator locality would be to use a more non-local operator. For instance, the differentiation in equation (3.5) could be combined with leaky integration; the effect of this integration would be to narrow the valley in Figure 3.4, and to widen the damping operator. Preliminary tests have shown that this approach works on one-dimensional problems: the number of required LSQR iterations is reduced. The leaky-integration approach is unsuccessful on two-dimensional problems, however: the number of LSQR iterations actually increases. This increase may be the result of applying a one-dimensional theory to a two-dimensional problem, or it may simply be the result of an undetected programming bug. In any case, the examples in this thesis were all computed using the unsophisticated two-point differentiation operator given in equations (3.9) and (3.10).

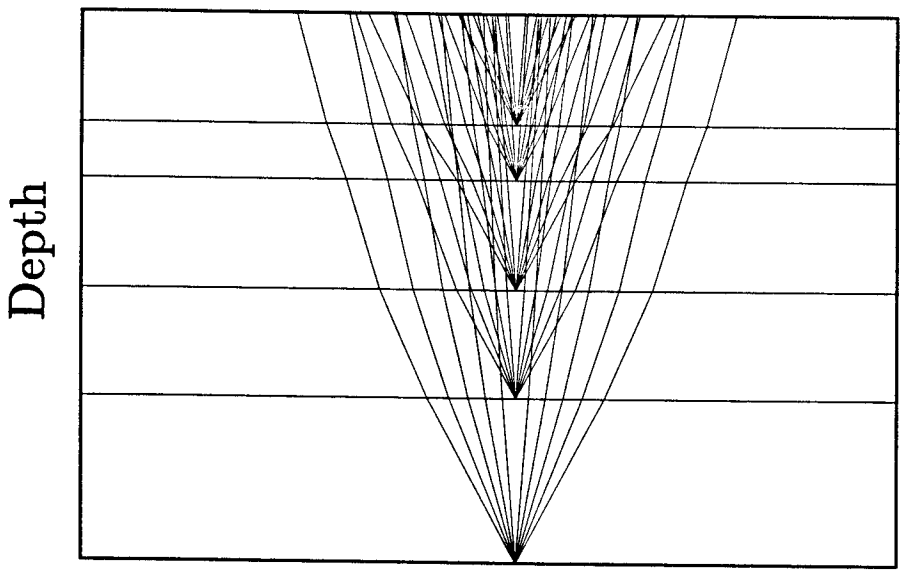

Figure 3.5: Generation of synthetic data. This model consists of 5 layers; velocity is constant within each layer. The third layer from the top is a low-velocity zone. There is no vertical exaggeration.

3.4 Synthetic examples

The CDR tomographic inversion method has been tested on synthetic data, including data from a vertically stratified (laterally homogeneous) model and from a fully inhomogeneous model. In all cases, the picked reciprocal parameters (x_s, x_g, p_s, p_g, t) were obtained directly from the modeling program, so the picking procedures described in the previous chapter were not used.

3.4.1 Synthetic data from a layered medium

The first synthetic example is from a vertically stratified model. The model consisted of five layers, with a constant velocity within each layer. Figure 3.5 shows the raypaths generated during the modeling. The surface-position parameters, x_s and x_g, were arbitrary; they depended only on where the rays happened to intersect the surface. A total of 25 sets of reciprocal parameters were generated. No noise was added to these data.

These synthetic data were used as input to the inversion algorithm described in section 3.3. A vertically-stratified model was used in the inversion, with layers a distance of $\Delta z = .01$ apart; the total depth of the model was 1.0. The velocity boxes were each as wide as the model, so that there was only one box per layer. The units used are unimportant, but the reader is free to choose any set of units that seems reasonable (distance in tens of

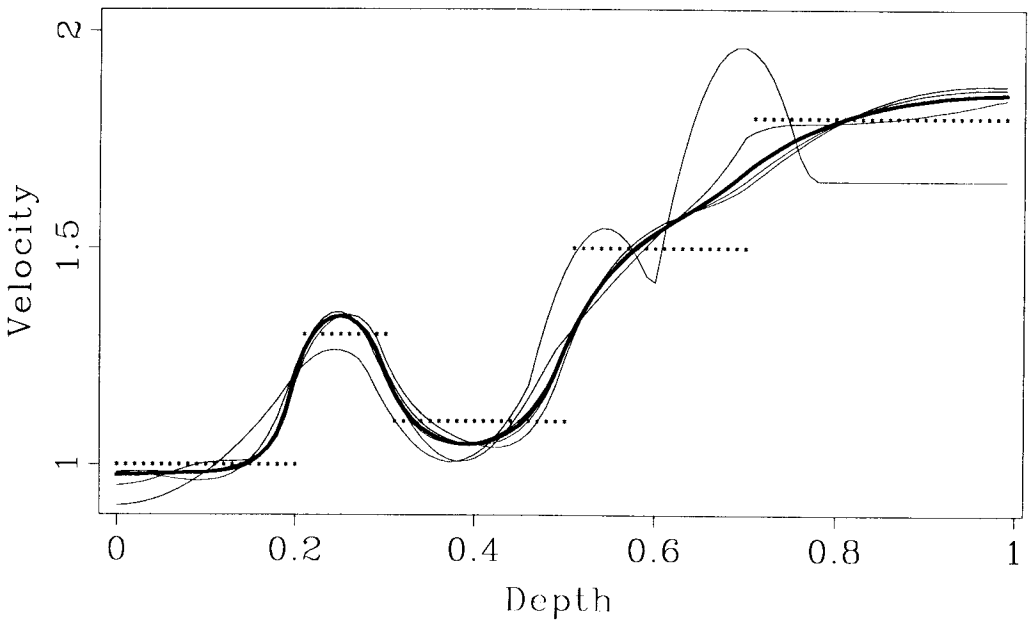

Figure 3.6: Iterative inversion. The synthetic data generated by the rays in Figure 3.5 were inverted for interval velocity. The results of the first 6 iterations are shown here; the result of the 6th iteration is drawn with a thicker line. The dotted line shows the correct velocity model.

kilometers and time in seconds, for example). A vertical damping coefficient of $\lambda_z = .0001$ was used, and the initial velocity model was a constant velocity of 1.0. Figure 3.6 shows the results of the first 6 iterations plotted as functions of velocity versus depth. The results are encouraging. The undamped inversion problem would be underdetermined, since there were only 25 sets of picked parameters and 100 layers in the model; the damping thus played an important role. The calculation required about 30 seconds of CPU time on a Convex C-1 computer.

3.4.2 Synthetic data from a laterally heterogeneous model

It is an interesting exercise to use tomography to invert data from a vertically stratified medium, but it is not necessarily practical, since there are many faster ways of determining the velocity structure in such media. More interesting is to invert data from a medium that is both vertically and laterally inhomogeneous. In this sub-section, I invert synthetic data from a model similar to that used by Červený as a test of his ray-tracing program. This model is shown in Figure 3.7, as are all the raypaths from a typical shot.

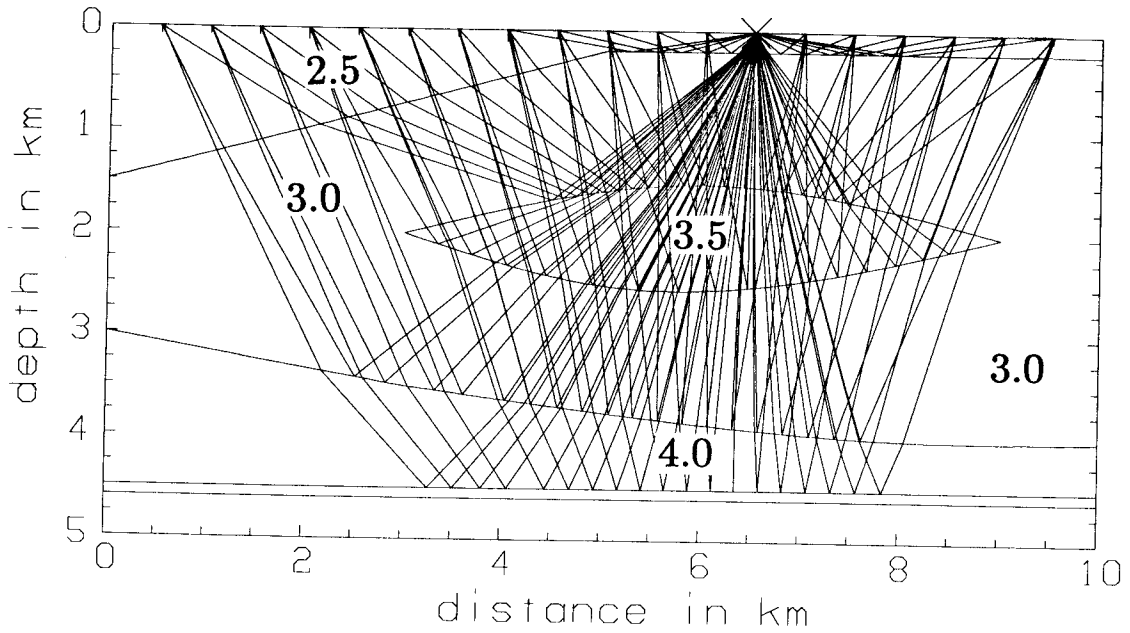

Figure 3.7: Synthetic model. This model was used to generate synthetic picked parameters. Note the depth of the deepest reflector. Shown are all the rays from a single shot. Some of these rays were not used in the subsequent inversion: they arrived at too shallow an angle. The shots and geophones are distributed along the surface, at intervals of .5 km, starting at the .5 km mark and going to the 9.5 km mark. Velocities are in km/sec.

The ray-tracing program SEIS83 (Červený and Pšenčík, 1984b) was used to generate the picked parameters. There were 19 geophone locations and 19 shotpoints. It might be preferable to say that 19 shot arrays and 19 geophone arrays were used, since in practice the ray parameters p_s and p_g can be determined only by analyzing the data from arrays of adjacent shots and geophone. Any rays that arrived at the surface at too shallow an angle (more than 63.5 degrees from the vertical) were rejected. The program did not generate diffracted rays (although the inversion process is able to treat them correctly). The output of the modeling program consisted of about 1,400 sets of picked parameters. Figure 3.8 shows the result of time migrating these picked parameters at velocity v_{CDR} (see section 2.3.3, page 28).

Noise-free data

The picked parameters, with no noise added, were used as input to the tomographic inversion program. The model used in the inversion consisted of a grid 10 km wide and 5.5 km deep, with a horizontal and vertical spacing of .05 km between adjacent grid points. Thus, there were 22,000 grid points in the inversion model. The initial velocity model

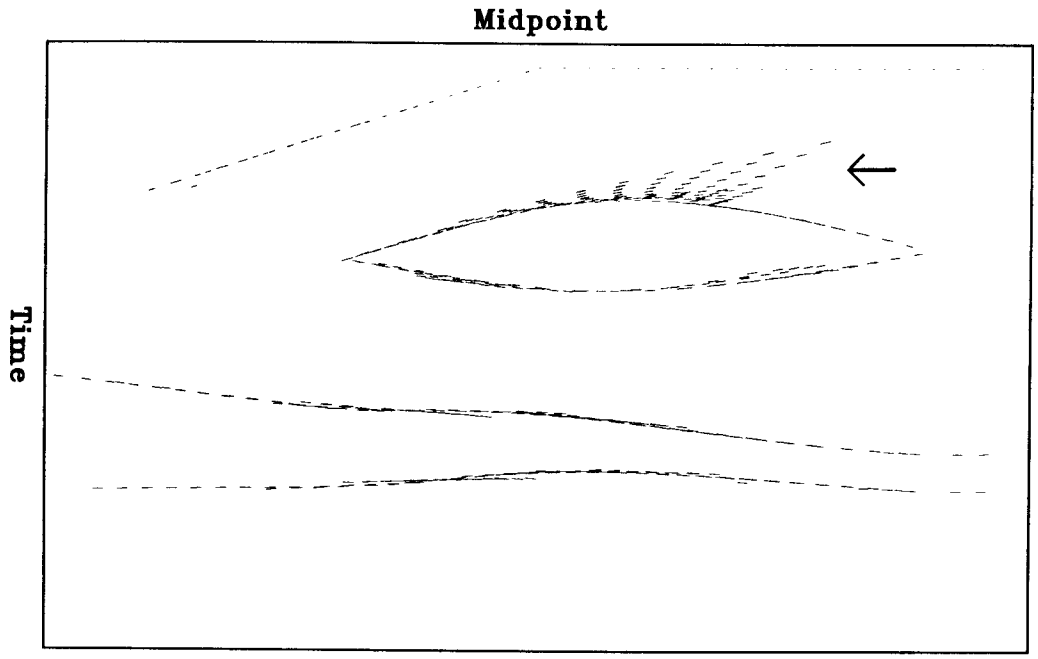

Figure 3.8: Migrated time section. The synthetic data, generated by tracing rays through the model in Figure 3.7, are plotted after time migration at velocity v_{CDR}. This figure is analogous to Figure 2.7 (page 29). An arrow points to some oddly placed dip bars that resulted from rays passing through the dipping boundary on the left half of the model.

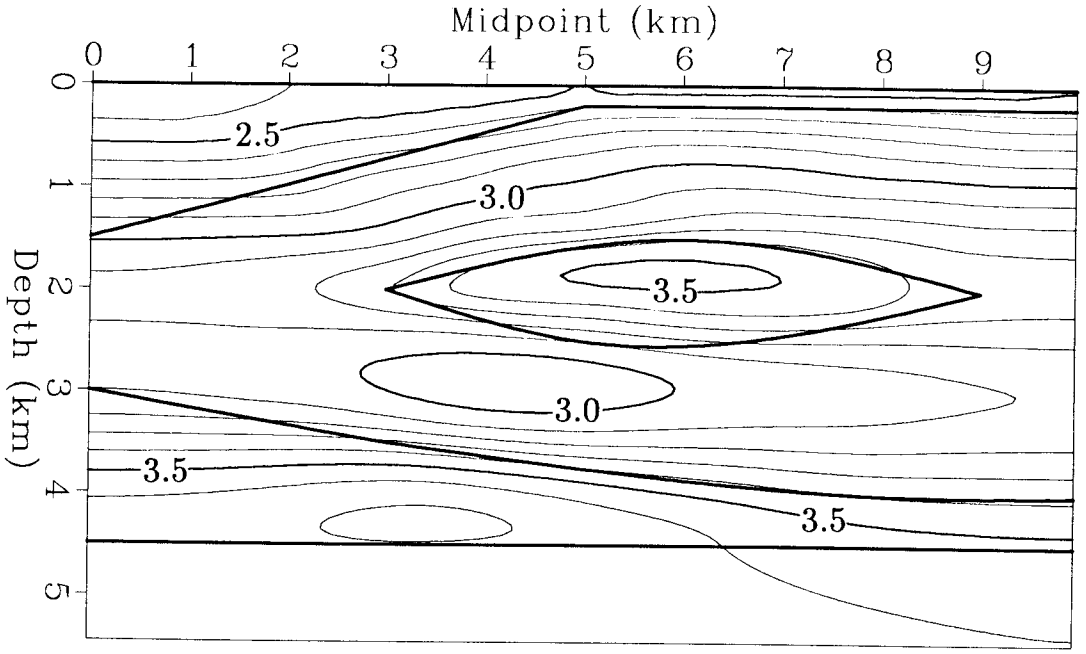

Figure 3.9: Result of inverting with a 2-D model. This figure shows the results of inverting the picked parameters from Figure 3.7. The original model is overlaid for comparison; velocities are in km/sec.

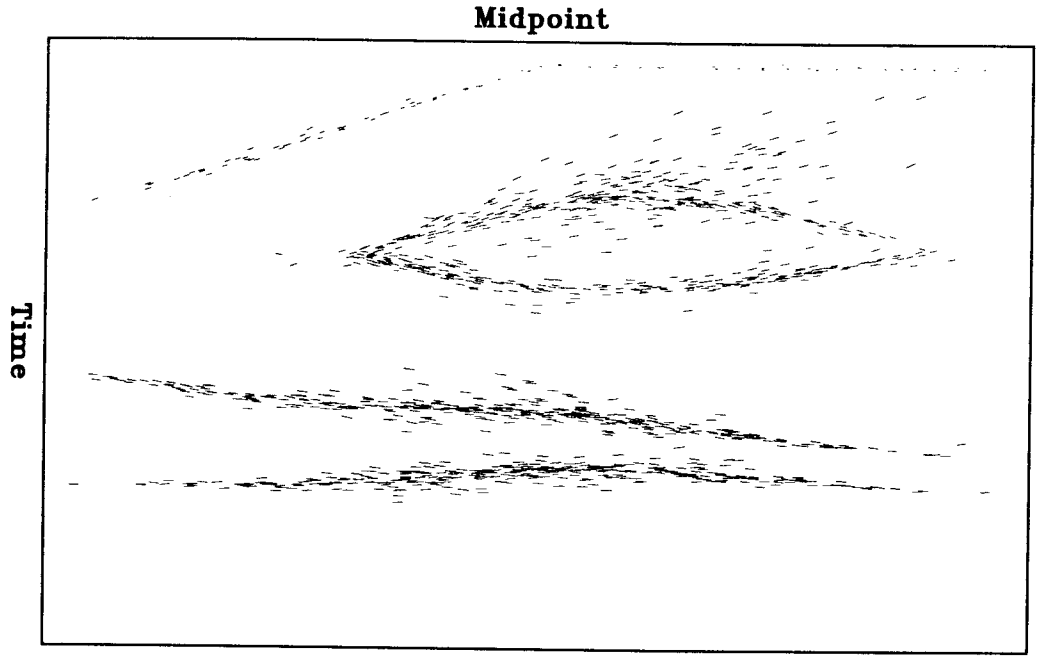

Figure 3.10: Time migration of noisy synthetic data. This figure is similar to Figure 3.8, except that random noise has been added to the synthetic data.

(first guess) was homogeneous, with a constant velocity of 1.5 km/sec.

Figure 3.9 shows the result of the 10th iteration of the inversion. The damping factors were $\lambda_x = .0756$ and $\lambda_z = .0126$. The three-place precision in the damping factors is illusory; they have been converted from other units, where they had only one-place precision. Because of the damping, necessary to achieve stable convergence, the result shows only the low-frequency terms of the original model. In the top half of the section, the results of the inversion agree well with the original model. In the bottom half of the section, the low-velocity zone under the high-velocity lens is correctly found; the velocity in the lowest layer is somewhat low.

Noisy data

Gaussian random noise was added to the ray parameters p_s and p_g of the synthetic data in order to test the inversion process on noisy data. Figure 3.10 shows a v_{CDR} time migration of the noisy picked data. A comparison of this figure with Figure 3.8 gives a qualitative idea of the variance of the noise. Notice that this data is somewhat noisier than the marine data of the previous chapter. Because of the noise, it was necessary to use larger damping factors in the tomographic inversion: $\lambda_x = .1140$ and $\lambda_z = .0190$. Fewer iterations were

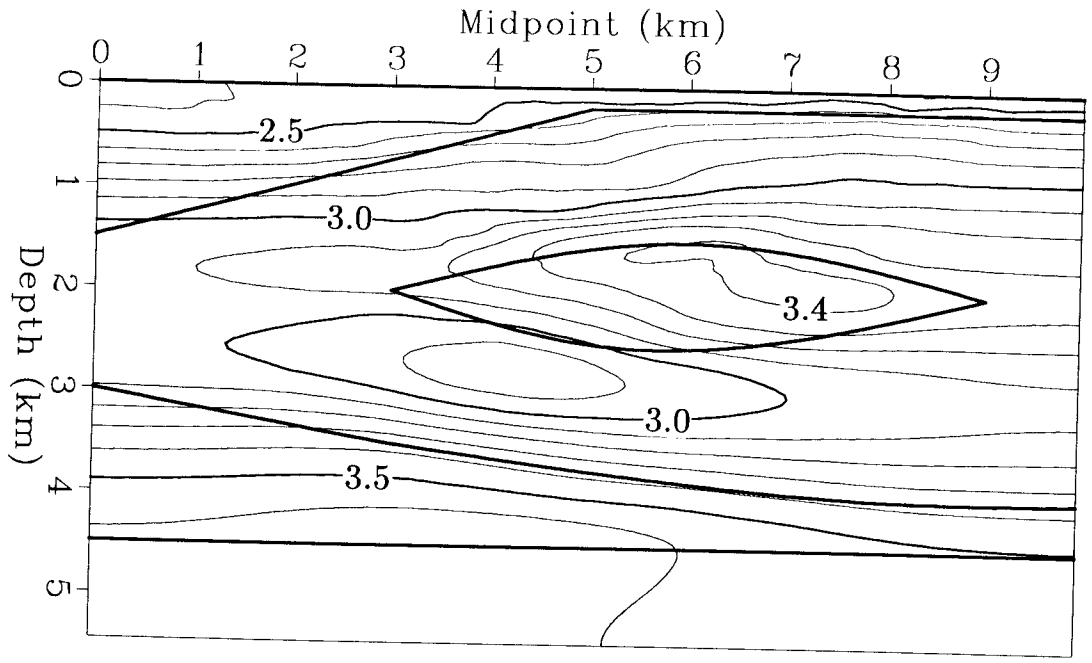

Figure 3.11: Inversion results: noisy synthetic data. This figure is analogous to Figure 3.9, except that before the inversion, random noise was added to the synthetic picked parameters.

necessary: only 5. The results of the inversion are shown in Figure 3.11. The low-velocity zone under the lens was correctly found, but the lens itself is not correctly positioned.

3.5 Imaging the reflectors

Once the velocity model is found by inversion, full CDR depth migration can be performed on the picked parameters. In essence, the rays corresponding to each set of picked parameters are traced down to the depth z_e (defined in section 3.2.3), and a dip bar is drawn halfway between the endpoints of the shot and geophone rays (see Figure 3.12). The angle of each dip bar is chosen so that the angle of incidence equals the angle of reflection. In Figure 3.13 are shown the results of performing this process using the velocity model in Figure 3.9. The width of each dip bar is equal to x_{err}, so this figure gives some idea of how well the inversion process worked.

Most reflection seismologists (other than old-timers and adherents of CDR) are unaccustomed to seeing sections composed of dip bars, so I have rasterized and tapered the dip bars and performed some time filtering. The results are shown in Figure 3.14. The widths

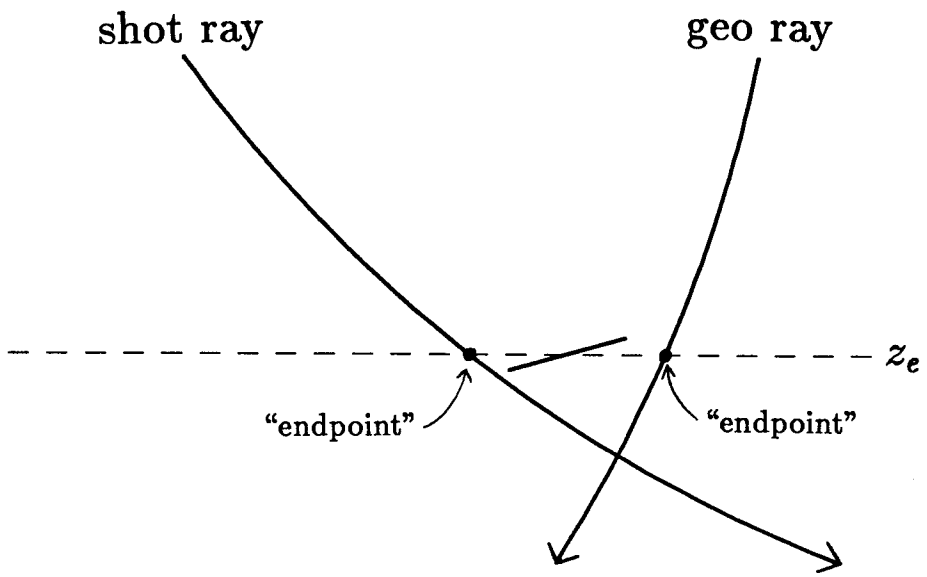

Figure 3.12: Basis of CDR migration. Given a set of picked parameters and a velocity structure, a dip bar can be drawn to represent a reflector segment. This dip bar is drawn at depth z_e (defined in section 3.2.3), halfway between the endpoints of the shot and geophone rays. The tilt of the dip bar corresponds to the dip of the reflector that would have generated the two rays.

of the dip bars are now based on travel times of the rays, rather than on x_{err}.

Thus, I have taken data from a synthetic model (Figure 3.7), inverted the data to find the velocity structure (Figure 3.9), and used this velocity structure to perform a pre-stack CDR migration on the data (Figure 3.14). The velocity inversion required about 225 minutes of CPU time on a Convex C-1 computer. This corresponds to 23 CPU minutes per iteration, with about 40% of this time spent in ray tracing, and the remainder spent in solving the linear least-squares problem. The subsequent CDR pre-stack migration required less than 5 minutes.

3.6 Resolution

Figures 3.6 and 3.9 show that the method of CDR tomographic inversion is able to resolve only the low-frequency components of the velocity field. This is not a surprising result; traveltime information does not contain enough information for the resolution of the high-frequency components. The cut-off frequency is dependent in a complex way on the size of the damping factors λ_x and λ_z. In general, the higher the values of λ, the lower the

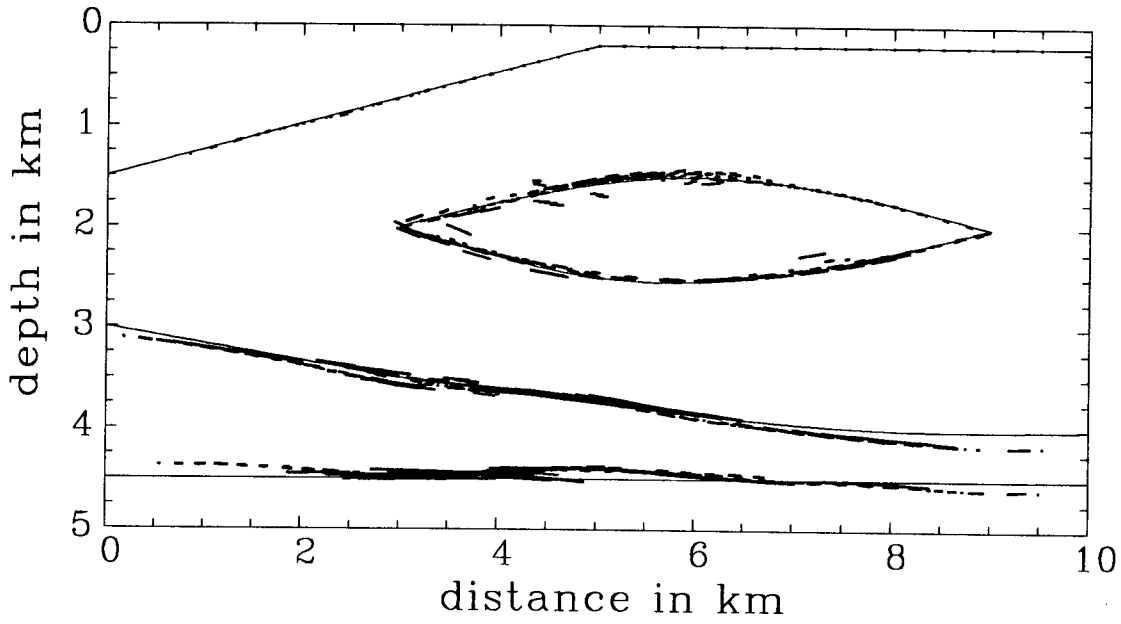

Figure 3.13: Results of CDR migration. Based on the picked parameters generated from the model in Figure 3.7, and on the velocity structure shown in Figure 3.9, dip bars have been drawn according to the technique illustrated in Figure 3.12; the length of each dip bar equals x_{err}. The original model is overlaid for comparison. Notice how closely the results fit the original model.

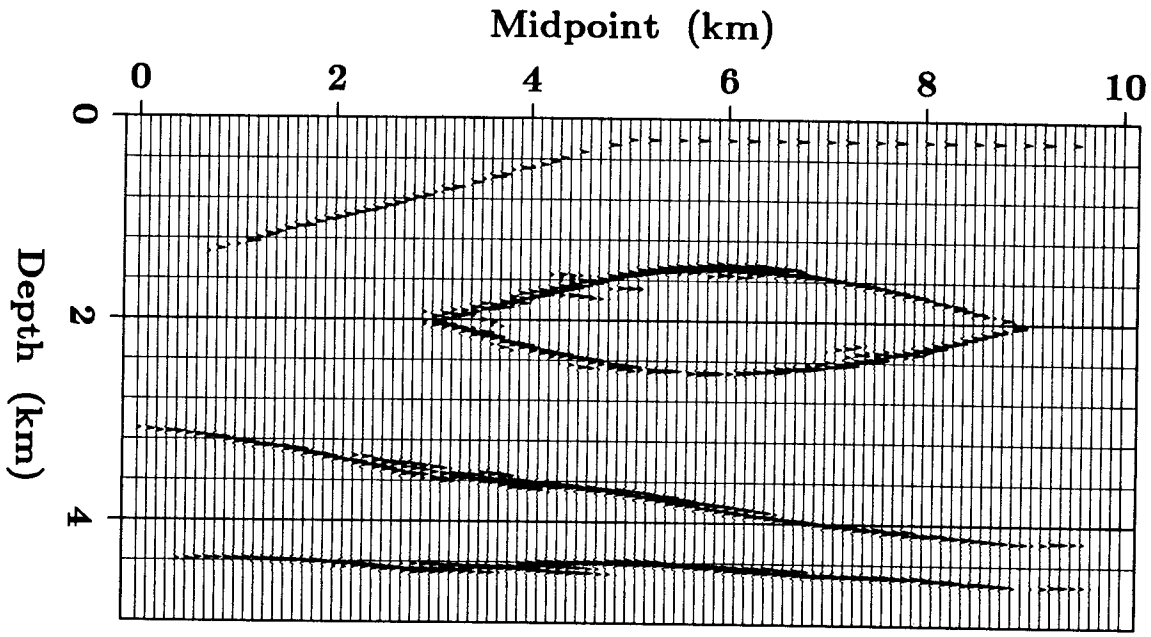

Figure 3.14: Filtered dip bars. The dip bars shown in Figure 3.13, after rasterization and filtering.

cut-off frequency.

One wants as high a frequency resolution as possible, of course. The limiting factor is that when noise in the data induces spurious high-frequency lateral velocity variations, the rays shoot off in unrealistic directions, resulting in meaningless values of x_{err}. Thus, enough horizontal damping must be provided to prevent these spurious lateral velocity variations. The vertical damping, in turn, must not be much smaller than the horizontal damping, or else the velocity inversion will tend to produce horizontally-layered models. I have found that it is best to let λ_x (horizontal damping) be about 6 times larger than λ_z (vertical damping).

Unfortunately, there does not seem to be any easy way to determine the best values of the damping factors, other than by trial and error. These values depend in a complicated way on the density of the observation system, the number of sets of reciprocal parameters, the weighting applied to each set of reciprocal parameters, and the density of the velocity grid.