2.5-D Common Offset Migration in a Triangulated Model

Boyi Ou

ABSTRACT

2.5D inversion developed by Bleistein et al (1987) provides a powerful tool for obtaining correct location of interfaces while preserving a model-consistent amplitude. Here, the use of ray tracing in a triangulated model of the earth to implement 2.5D inversion is described. In this paper, velocity is replaced by sloth to define in the model. The medium is assumed to consist of constant gradient sloth in triangulated regions, and is separated by arbitrary, not necessarily smooth interfaces. The objective of the inversion is to correct locations of the interfaces and return true amplitude consistent with the underlying theory. To date, I have focused on the program logic and traveltime calculation, so that this is a migration program, producing a reflector map but not yet producing a model-consistent amplitude. The amplitude calculation will be incorporated into the code before it is offered to the sponsors. Some computer implementations for synthetic data and physical data are presented. This new code overcomes some of the model complexity limitations of the earlier CXCZO code as described in the text.

INTRODUCTION

Bleistein et al. (1987) derived the Kirchhoff integral inversion method by using the WKBJ approximation. This method treats amplitude in migration 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 reflection angle. In Bleistein's integral formula, the integral requires ray data: traveltime, WKBJ amplitude and other ray parameters. Two commonly used approaches to calculate traveltime and the other quantities in the Kirchhoff integral are ray-tracing and finite-difference applied to the eikonal equation.

Hsu and Liu (1991) developed a 2.5D Common Offset Inversion Program CXZCO base on two point ray tracing developed by Docherty (1988). This program implements the 2.5D inversion theory (Bleistein, 1987), but it has some inherent limitations due to the algorithm used to do the two point ray tracing. First, the model must consist of constant velocity layers. Second, the model is separated by interfaces which

are required to cross the entire model. This limits the complexity of background models that this program can handle. Thus we need to develop a program to deal with more general models. Hale and Cohen (1991) developed software to generate 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 generalized that code. I take advantage of triangulation ray tracing to accomplish my goal: implement 2.5D inversion theory for more complex models. The ray tracer used in this project 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. This interpolation is complicated when the raypaths have a caustic. At present, I ignore caustics, but will address this issue in continuing research. The calculation cost of traveltime and amplitude for every midpoint would 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) and compute ray data at a sparse set of grid points. In this work, the velocity model is replaced by a sloth model separated by interfaces that are more complex than those used in CXZCO. For example, the lens-shaped structure and pinchouts shown in Figure 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 no longer need to cross the entire model. Furthermore, the velocity can vary both vertically and horizontally, whereas, CXCZO requires constant velocity in layers. In summary, this projects synthesizes the 2.5D inversion theory previously implemented in CXZCO and Rüger's dynamic ray tracing to produce a 2.5D migration code that is applicable to more general Earth model's than the predecessor. To date, I have focused on the program logic and traveltime calculation, so that this is a migration program, producing a reflector map but not yet producing a mode-consistent amplitude. The amplitude calculation will be incorporated into the code before it is offered to the sponsors. The common shot inversion code CXZCS (Dong, 1989), will be similarly upgraded in the near future.

ALGORITHM OF 2.5D COMMON OFFSET INVERSION

The Kirchhoff integral method can be represented by

$$output = integral\{weight \cdot input\} \tag{1}$$

For inversion problems, the input is seismic traces, the output is a structural image. To accomplish inversion, quantities to be calculated are as listed below:

τ traveltime.

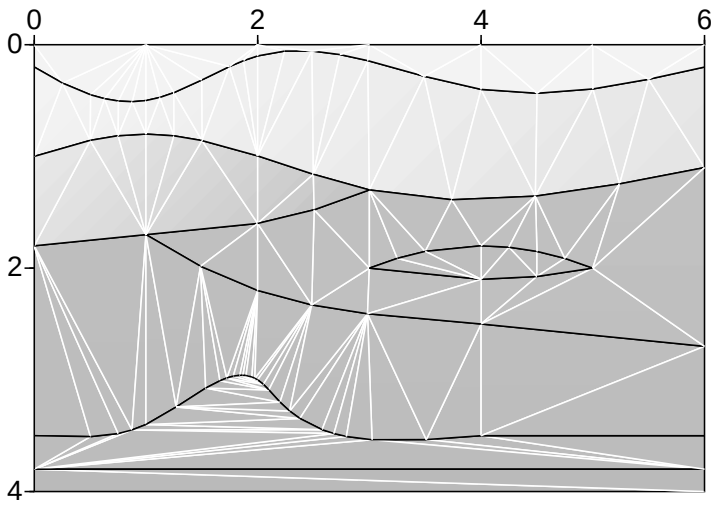

FIG. 1. Lens-shaped structure and Pinchouts. This model includes lens-type structure and a pinchout. Lithological interfaces are represented by black lines. The shading denotes the seismic velocity field. The white lines are edges of auxiliary triangles used in the model building.

ϕ	propagation angle.
σ	running ray parameter.
$\beta(x_s)$	take off angle from source.
$\beta(x_r)$	take off angle from receiver.
$\partial\beta(x_s)/\partial x$	geometrical spreading parameter for shot.
$\partial\beta(x_r)/\partial x$	geometrical spreading parameter for receiver.

In this list $\partial\beta(x_s)/\partial x$ and $\partial\beta(x_r)/\partial x$ are used to approximate geometrical spreading. However, in dynamic ray tracing geometrical spreading is one of the quantities naturally calculated by the method along each ray. I will describe here this alternative approach to characterization of geometrical spreading. First, I introduce the *Jacobian* of the transformation from Cartesian to the ray coordinates, given by

$$J = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \gamma} & \frac{\partial z}{\partial \gamma} \end{vmatrix}. \tag{2}$$

Here ξ and γ are ray coordinates. 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 could 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 very simply expressed in the ray-centered coordinates (ℓ,n), connected with the ray Ω, especially when we compute it directly at Ω. As seen in Figure 2, the coordinate ℓ measures the arclength along the ray Ω from an arbitrary

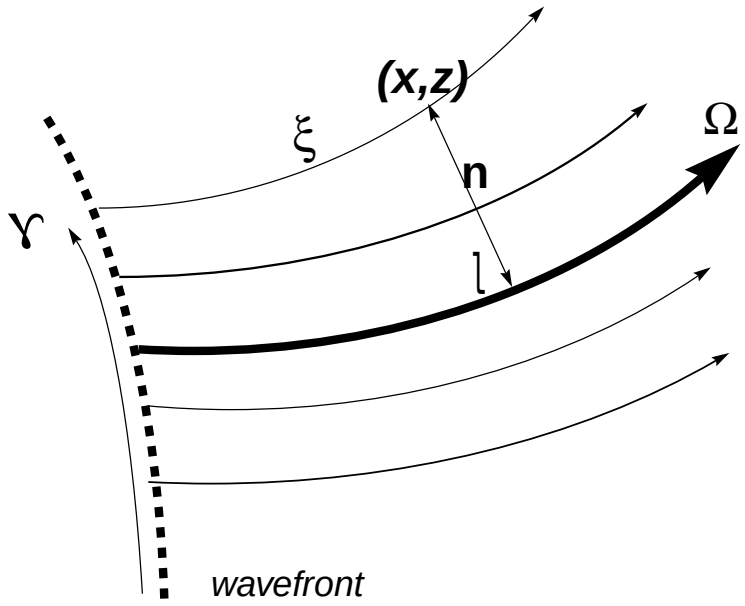

FIG. 2. In this figure, three coordinates systems are put together, they are coordinate (ξ,γ), ray center coordinate (ℓ,n) and Cartesian coordinate (x,z)

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

$$J = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \gamma} & \frac{\partial z}{\partial \gamma} \end{vmatrix} = \begin{vmatrix} \frac{\partial x}{\partial \ell} & \frac{\partial z}{\partial \ell} \\ \frac{\partial x}{\partial n} & \frac{\partial z}{\partial n} \end{vmatrix} \cdot \begin{vmatrix} \frac{\partial \ell}{\partial \xi} & \frac{\partial n}{\partial \xi} \\ \frac{\partial \ell}{\partial \gamma} & \frac{\partial n}{\partial \gamma} \end{vmatrix}. \tag{3}$$

The ray centered coordinate system is orthogonal with scaling factors (h,1), with

$$h = 1 + \frac{v_{,n}}{v} n. \tag{4}$$

Thus,

$$\begin{vmatrix} \frac{\partial x}{\partial \ell} & \frac{\partial z}{\partial \ell} \\ \frac{\partial x}{\partial n} & \frac{\partial z}{\partial n} \end{vmatrix} = h. \tag{5}$$

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

$$J = \frac{\partial n}{\partial \gamma}. \tag{6}$$

This is the final expression for the function J in the ray-centered coordinates (ℓ,n) connected with the ray Ω, directly at Ω. Following Červený (1981), we introduce the system:

$$\frac{dn}{d\ell} = vp_n, \quad \frac{dp_n}{d\ell} = -\frac{v_{,nn}}{v^2}n. \tag{7}$$

to describe a ray Ω_c near the central ray. Differentiating this system with respect to γ, we obtain

$$\frac{dJ}{d\ell} = vP, \quad \frac{dP}{d\ell} = -\frac{v_{,nn}}{v^2}J. \tag{8}$$

where

$$J = \frac{\partial n}{\partial \gamma}, \quad P = \frac{\partial p_n}{\partial \gamma}. \tag{9}$$

Here, J again denotes the Jacobian, and P is the auxiliary function (the derivative of the normal component of the slowness vector with respect to the ray parameter γ). Thus, we can use dynamic ray tracing along the ray to determine the function J and auxiliary function P.

We use J_s and J_r, obtained from dynamic ray tracing, to replace $\partial\beta(x_s)/\partial x$ and $\partial\beta(x_r)/\partial x$. The original formula derived by Docherty (1989), for the reflectivity $\beta(x,z)$ is

$$\begin{aligned}\beta(x,z) \quad \sim \quad \frac{1}{2(2\pi)^{3/2}A_f} \quad \cdot \quad & \int d\xi \sqrt{\frac{1}{\sigma_s}+\frac{1}{\sigma_r}}\left[\frac{\partial\beta(x_s)}{\partial\xi}+\frac{\partial\beta(x_r)}{\partial\xi}\right] \\ & \cdot[A(x_s,x)A(x_r,x)]^{-1} \\ & \cdot\int d\omega\sqrt{i\omega}e^{-i\omega[\tau(x,x_s)+\tau(x,x_r)]}U_s(\omega,\xi)\end{aligned} \tag{10}$$

Docherty use $\beta(x_s)$ and $\beta(x_r)$ for the emergence angle and $\beta(x,z)$ for reflectivity. Here, we have,

$$\begin{aligned}A(x_s,x) &= \frac{1}{4\pi}\left[\frac{c(x)}{\sigma_s\sqrt{g_s}\hat{n}_s\cdot\hat{p}(x_s,x)}\right]^{1/2}\left[\frac{\partial\beta(x_s)}{\partial\xi}\right]^{1/2}T(x_s,x), \\ A(x_r,x) &= \frac{1}{4\pi}\left[\frac{c(x)}{\sigma_s\sqrt{g_r}\hat{n}_r\cdot\hat{p}(x_r,x)}\right]^{1/2}\left[\frac{\partial\beta(x_r)}{\partial\xi}\right]^{1/2}T(x_r,x),\end{aligned} \tag{11}$$

where,

$$T(x_s,x)=\prod_{i_s=1}^{N}K_{i_s}\left[\frac{\cos\theta_{i'_s}}{\cos\theta_{i_s}}\right]^{1/2},$$
$$T(x_r,x)=\prod_{i_r=1}^{N}K_{i_r}\left[\frac{\cos\theta_{i'_r}}{\cos\theta_{i_r}}\right]^{1/2}. \tag{12}$$

Here, K_{i_s} and K_{i_r} are transmission coefficient on each interface. If we use A_s and A_r to substitute $\partial\beta(x_s)/\partial\xi$ and $\partial\beta(x_r)/\partial\xi$, we obtain

$$\begin{aligned}\beta(x,z) \sim\ & \frac{2\sqrt{2\pi}}{A_f c(x)}\int d\xi\sqrt{\frac{1}{\sigma_s}+\frac{1}{\sigma_r}}[A(x_s,x)A(x_r,x)]^{-1}\\ &\cdot\left[\frac{A(x_s,x)^2\sigma_s\sqrt{g_s}\hat{n}_s\cdot\hat{p}(x_s,x)}{T(x_s,x)^2}+\frac{A(x_r,x)^2\sigma_r\sqrt{g_r}\hat{n}_r\cdot\hat{p}(x_r,x)}{T(x_r,x)^2}\right]\\ &\cdot\int d\omega\sqrt{i\omega}e^{-i\omega[\tau(x,x_s)+\tau(x,x_r)]}U_s(\omega,\xi).\end{aligned} \tag{13}$$

Furthermore, from(Bleistein, 1986), we have,

$$\begin{aligned}A(x_s,x)=A(x,x_s)&=\frac{1}{4\pi\sqrt{\sigma_s J_s}}T(x,x_s),\\ A(x_r,x)=A(x,x_r)&=\frac{1}{4\pi\sqrt{\sigma_r J_r}}T(x,x_r).\end{aligned} \tag{14}$$

Thus, the final expression for our inversion is as follows

$$\begin{aligned}\beta(x,z) \sim\ & \frac{2\sqrt{2\pi}}{A_f c(x)}\int d\xi\sqrt{\sigma_s+\sigma_r}\\ &\cdot\left[\sqrt{\frac{J_r}{J_s}}\frac{T(x,x_s)\sqrt{g_s}\hat{n}_s\cdot\hat{p}(x_s,x)}{T(x,x_r)T(x_s,x)^2}+\sqrt{\frac{J_s}{J_r}}\frac{T(x,x_r)\sqrt{g_r}\hat{n}_r\cdot\hat{p}(x_r,x)}{T(x,x_s)T(x_r,x)^2}\right]\\ &\cdot\int d\omega\sqrt{i\omega}e^{-i\omega[\tau(x,x_s)+\tau(x,x_r)]}U_s(\omega,\xi).\end{aligned} \tag{15}$$

In this work, I only consider a flat observation surface, for which

$$\begin{aligned}\sqrt{g_s}=1,\qquad &\hat{n}\cdot\hat{p}(x_s,x)=cos\beta(x_s),\\ \sqrt{g_r}=1,\qquad &\hat{n}\cdot\hat{p}(x_r,x)=cos\beta(x_r).\end{aligned} \tag{16}$$

Therefore, in this case the formula for the reflectivity, $\beta(x,z)$, becomes:

$$\begin{aligned}\beta(x,z) \quad \sim \quad & \frac{2\sqrt{2\pi}}{A_f c(x)} \int d\xi \sqrt{\sigma_s + \sigma_r} \\ & \cdot \left[\sqrt{\frac{J_r}{J_s}} \frac{T(x,x_s)\cos\beta_s}{T(x,x_r)T(x_s,x)^2} + \sqrt{\frac{J_s}{J_r}} \frac{T(x,x_r)\cos\beta_r}{T(x,x_s)T(x_r,x)^2} \right] \\ & \cdot \int d\omega \sqrt{i\omega} e^{-i\omega[\tau(x,x_s)+\tau(x,x_r)]} U_s(\omega,\xi). \end{aligned} \tag{17}$$

The meaning of each symbol as follows:

$\sigma_s; \sigma_r$	Ray trace running parameter from shot and receiver to output, respectively;
$\beta_s; \beta_r$	Angles between ray and downward vertical at x_s and x_r;
J_s, J_r	Jacobian function for shot and receiver respectively;
$T(x, x_s)$	transmission factor along down-going ray for shot;
$T(x, x_r)$	transmission factor along down-going ray for receiver;
$T(x_s, x)$	transmission factor along up-going ray for shot;
$T(x_r, x)$	transmission factor along up-going ray for receiver.

This is the inversion formula that I will use. Below, I will describe ray tracing in triangulated models.

Ray tracing in triangulated model

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

$$\begin{aligned} \frac{dx}{d\sigma} &= p_x, \\ \frac{dz}{d\sigma} &= p_z, \\ \frac{dp_x}{d\sigma} &= \frac{1}{2}\frac{\partial v^{-2}}{\partial x}, \\ \frac{dp_z}{d\sigma} &= \frac{1}{2}\frac{\partial v^{-2}}{\partial z}, \\ \frac{dt}{d\sigma} &= v^{-2}. \end{aligned} \tag{18}$$

Here v denotes velocity, x and z denote the Cartesian coordinates of the ray, p_x and p_z are the x and z component, respectively, of the slowness vector. Furthermore, t is traveltime, σ is a running parameter that increases monotonically along the ray, as implied by the last of equations (18).

The reason for writing the ray tracing equations in this form is that the partial derivatives of v^{-2} in (18) are constants when v^{-2} is a linear function of x and z. Specifically, we define "sloth", the inverse of velocity-squared, as follows

$$s(x,z) \equiv v(x,z)^{-2} \tag{19}$$

and introduces the linear approximation,

$$s(x,z) = s_{00} + s_{,x}x + s_{,z}z \tag{20}$$

in each triangle. Then the solutions to the equations (18) in each triangle are

$$\begin{aligned}
x(\sigma) &= x(\sigma_0) + p_x(\sigma_0)(\sigma-\sigma_0) + \frac{1}{4}s_{,x}(\sigma-\sigma_0)^2, \\
z(\sigma) &= z(\sigma_0) + p_z(\sigma_0)(\sigma-\sigma_0) + \frac{1}{4}s_{,z}(\sigma-\sigma_0)^2, \\
p_x(\sigma) &= p_x(\sigma_0) + \frac{1}{2}s_{,x}(\sigma-\sigma_0), \\
p_z(\sigma) &= p_z(\sigma_0) + \frac{1}{2}s_{,z}(\sigma-\sigma_0), \\
t(\sigma) &= t(\sigma_0) + [s_{00} + s_{,x}x(\sigma_0) + s_{,z}z(\sigma_0)](\sigma-\sigma_0) + \\
&\quad \frac{1}{2}[s_{,x}p_x(\sigma_0) + s_{,z}p_z(\sigma_0)](\sigma-\sigma_0)^2 + \\
&\quad \frac{1}{12}({s_{,x}}^2 + {s_{,z}}^2)(\sigma-\sigma_0)^3.
\end{aligned} \tag{21}$$

The ray path described by equation (21) 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 (21) at that point. In addition to the solution of equation (18), dynamic ray tracing in two dimensions requires the solution of the following system of coupled differential equations (*Červený*, 1987):

$$\begin{aligned}
\frac{dq}{d\sigma} &= p \\
\frac{dp}{d\sigma} &= -\frac{v_{,nn}}{v^3}q
\end{aligned} \tag{22}$$

From the first two equations in (18), one can show that $d\sigma = v d\ell$. Using this in the above equations, we obtain (8) with J replaced by q and P replaced by p. Thus, q is just the Jacobian J required in our inversion. For the linear sloth used here,

$$\begin{aligned}
\frac{dq}{d\sigma} &= p, \\
\frac{dp}{d\sigma} &= -\frac{3}{4}\frac{(s_{,x}p_z - s_{,z}p_x)^2}{s^2}q.
\end{aligned} \tag{23}$$

The solution to these equations is

$$\begin{aligned} q(\sigma) &= \frac{1}{\sqrt{s_0 s}}\left[[s_0+\frac{s_1(\sigma-\sigma_0)}{2}][q(\sigma_0)+p(\sigma_0)(\sigma-\sigma_0)]+(\frac{{s_1}^2}{4s_0}-s_2)q(\sigma_0)(\sigma-\sigma_0)^2\right] \\ p(\sigma) &= \frac{1}{\sqrt{s_0 s}}\left[[s_0+s_1(\sigma-\sigma_0)]p(\sigma_0)+[\frac{s_1}{2}+(\frac{{s_1}^2}{2s_0}-2s_2)(\sigma-\sigma_0)]q(\sigma_0)\right] \\ &\quad -[\frac{s_1+2s_2(\sigma-\sigma_0)}{2s}]q(\sigma). \end{aligned} \tag{24}$$

where s_0, s_1, s_2 are constants defined by

$$\begin{aligned} s_0 &\equiv s[x(\sigma_0), z(\sigma_0)], \\ s_1 &\equiv s_{,x}p_x(\sigma_0)+s_{,z}p_z(\sigma_0), \\ s_2 &\equiv \frac{1}{4}({s_{,x}}^2+{s_{,z}}^2). \end{aligned} \tag{25}$$

Also, from (18) (20) and (25), one can show that

$$s = s[x(\sigma), z(\sigma)] = s_0 + s_1(\sigma-\sigma_0) + s_2(\sigma-\sigma_0)^2 \tag{26}$$

Given the values $q(\sigma_0)$ and $p(\sigma_0)$ at the point where a ray enters a triangle, equations (24) yield the values $q(\sigma)$ and $p(\sigma)$ at the point where the ray exits that triangle. In summary, we use (18) and (23) to obtain the ray path, the function J and other ray data we need at the edges of triangles. Now we must address the problem of obtaining these data at grid points.

Transformation from triangular system to uniform grid

Transformation of ray data from a triangular system to a uniform grid is crucial to my application of this modelling technique to inversion. The structure of my technique for achieving this is described below.

1. Define many horizontal levels. Normally, the distance between levels is 1/4 wavelength. Refine the grid when necessary to avoid aliasing.

2. Shoot a ray from shot or receiver down to subsurface; determine which horizontal level is between the entering point and exiting point of triangle. Thus we can know the exact the location of z relative to the triangles. From this we can obtain all information we need by solving the equations (18) and (23) at z .

3. Since the intersection points (x values at depth z) are still not grid points, I use linear interpolation to transform ray data from intersection points to grid points.

4. On every grid point, compute ray data.

Parallel interpolation between source points

Calculation cost of traveltime and amplitude for every midpoint would dominate the total cost of the Kirchhoff integral method. I use a parallel interpolation technique on sparse grids developed by Liu (1993) to reduce computation time of this part of my program. The interpolation between the sources or receivers proceeds as follows: As shown in Figure 3, traveltime and amplitude are calculated from selected sources

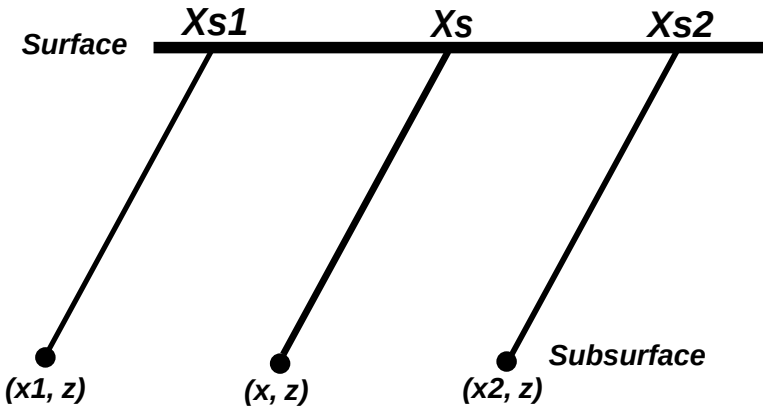

FIG. 3.

or receivers, and then these values are interpolated for the other sources or receivers in between. Suppose that the traveltime functions $\tau(x_1, z; x_{s1})$ and $\tau(x_2, z; x_{s2})$ from two sources x_{s1} and x_{s2} have been calculated, I interpolate a traveltime function of sources x_s at position $(x,\ z)$ $(x_{s1} \leq x_s \leq x_{s2})$ by

$$\tau(x, z, x_s) = \lambda\tau(x_2, z; x_{s2}) + (1 - \lambda)\tau(x_1, z; x_{s1}) \tag{27}$$

where $\lambda = (x_s - x_{s1})/(x_{s2} - x_{s1})$. I do same thing for the amplitude.

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

$$\tau(x + h, z; x_s + h) = \tau(x, z; x_s) \tag{28}$$

For a general velocity function $v(x, z)$, the interpolation error theoretically depends on $\partial v/\partial x$. In the next section, I will show some results which demonstrate reduced CPU time with little loss in accuracy achieved by this interpolation method.

Computer Implementation

To check my program, I applied it to synthetic data and physical data. Some of these results are described here. I define a parallel interpolation parameter Skip which characterized the sparsity of the points used to shoot rays. For example, Skip = 5 mean every fifth midpoint is used to shoot rays.

Synthetic Data

Example 1

In Example 1, the model is shown in Figure 1. It consists of eight reflectors, with a lens-shaped inclusion. The dark lines are the fixed interfaces and the white lines are the triangulars' edges of the model generated by Gbmodel. In some layers the velocity is constant, and in other layers sloth changes both horizontally and vertically. Rüger's GBmod code requires data described in a file of parameter lines called sfill. Below is part of this shell file which I use to generate the model.

sfill = x, z, x0, z0, s00, dsdx, dsdz
sfill =0.1, 0.1, 0, 0, 1, 0, 0
sfill = 2., 0.8, 0, 0, 1, -0.04, -0.04
sfill = 1, 1. 1, 0, 0, 1, -0.2, -0.2
sfill = 2, 2, 0, 0, 0.1111111, 0, -0.01
sfill = 0.1, 2, 0,2, 0.0625, 0, -0.01
sfill = 4, 2, 0, 0, 0.0625, 0, 0
sfill = 0.1, 3.7, 0, 0, 0.04, 0, 0
sfill = 0.1, 3.9, 0, 0, 0.01, 0,0

When we want to use Gbmodel to generate a sloth model, one way is to use sfill to fill sloth information into each layer. Let $(x,\ z)$ denote any point inside a closed region. Sloth inside this region is determined by $\mathrm{s}(x,\ z) = \mathrm{s}00 + (x - x0)\times dsdx + (z - z0)\times dsdz$. So in my lens-shaped model, there are eight separated regions corresponding to each sfill in the shell file. For example, in the first layer the sloth is equal to 1. After building a model, I use gbseis to generate a common-offset data set whose offset is 0.3 km (Figure 4). Figure 5 is the migration result without using interpolation between sources or receivers. Figure 6 is the migration result with the use of parallel interpolation between sources. Here, I shoot rays for every fifth midpoint. Figure 5 and Figure 6 show comparable results except that run times are quite different. The latter is 3 times faster than the former. Actually, several factors can influence the speed of program including, density of rays and number of rays. So, the speed of the program is flexible, but parallel interpolation is the main factor in this speed-up.

Example 2

Figure 7 shows the Marathon model used in example 2. In each layer the sloth is constant. Instead of synthetic data, I use physical tank data to test my program. Figure 8 is the largest offset physical data, and Figure 9 is the corresponding migration result. In fact, I use five common offset physical data sets. After migration of these data, I stack five outputs. The result is shown in Figure 10 to Figure 14 with different Skip. Surprisingly, it is not easy to degrade the result even I use very large Skip. At

skip:20, the fault at 7kft starts to degrade. At skip:25 the lowest (flat) reflector starts to degrade and some of the sawteeth have dismissed. For this test, the hightest skip which can obtain good result is 15. Also, for comparison, I show the inversion result which is generated by Suinvxz developed by Liu (1994) in Figure 15. We both use velocity model obtained by Liu (1994), and they give us comparable result.

CONCLUSION

In this paper, ray tracing in triangulated sloth models is used to calculate ray data for 2.5-D inversion with linear interpolation used to obtain ray data on uniform grids. Also, parallel interpolation following Liu is used to speed up the program. This method has some difficulties dealing with caustics and turning waves. It requires a good way to sort caustic rays and turning rays together, separated from other rays. Also, it is possible to have some shadow zones that are not covered by rays. This can be caused by range of take-off angle, caustic region, and post-critical angle incidence at interfaces. These anomalies can degrade ray data calculations less accurate. However, the program can really handle more complex and general model than CXZCO.

ACKNOWLEDGMENTS

I wish to thank my advisor Dr. Norman Bleistein for his terrific help, guidance and support on this project. I also wish to thank Zhenyue Liu and Andreas Rüger for their assistance in this research effort.

REFERENCES

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., 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., 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.

Dong, W, 1989, Finite Difference Ray Tracing And Common Shot Inversion: CWP-084, 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.

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

Liu, Z, 1994, Migration Velocity Analysis.

Liu, Z, and Hsu C-H, 1991, CXZCO: A 2.5D Common Offset Inversion Program in a c(x, z) Medium: CWP-U15, Colorado School of Mines.

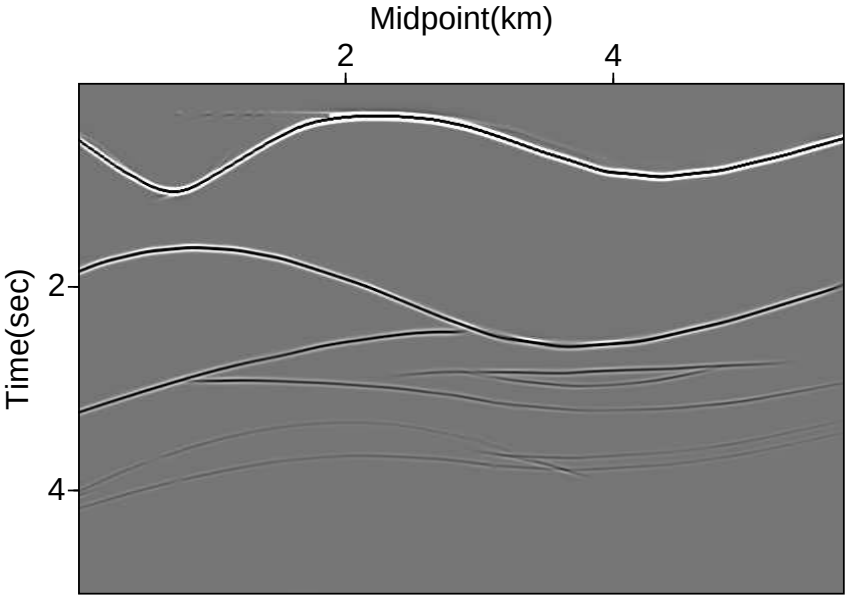

FIG. 4. Synthetic data generated by Gaussian beam method (offset = 0.3(km))

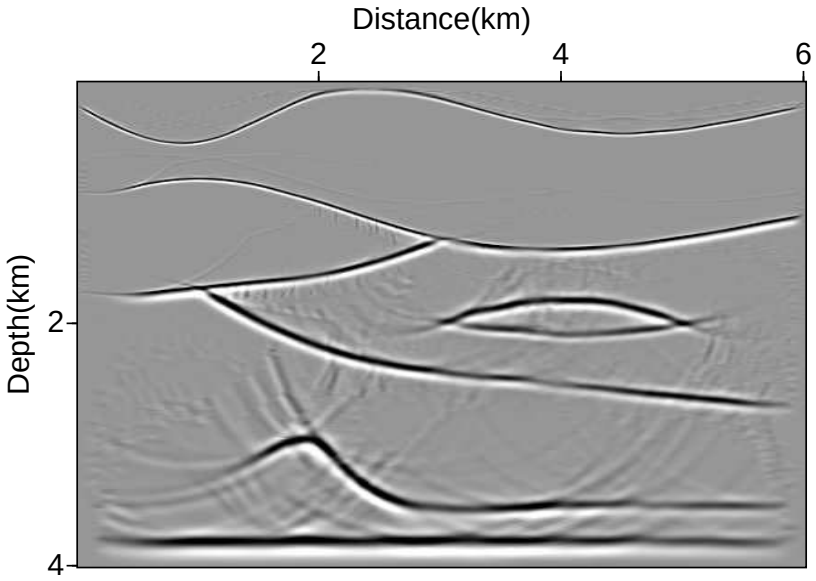

FIG. 5. Migration result using ray tracing without doing interpolation between midpoints. (offset = 0.3 (km) Skip = 1)

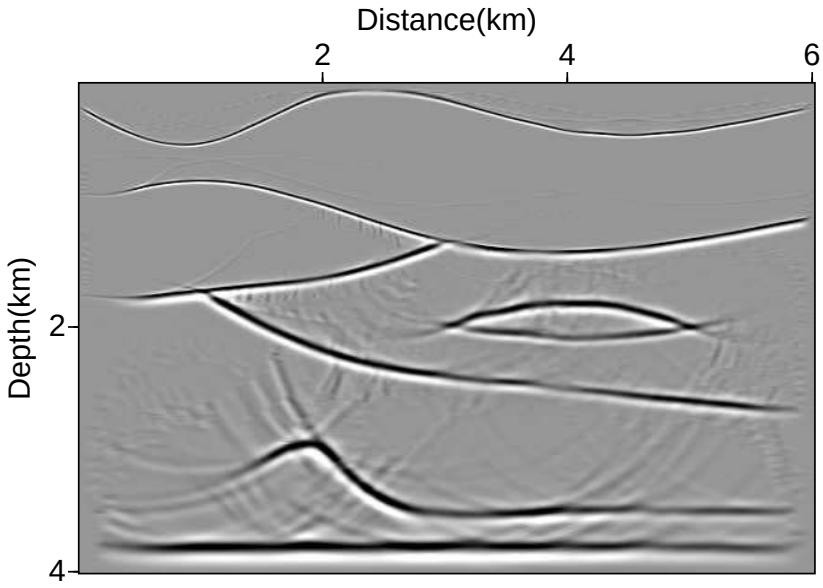

FIG. 6. Migration result with doing interpolation (offset = 0.3 (km) Skip = 5)

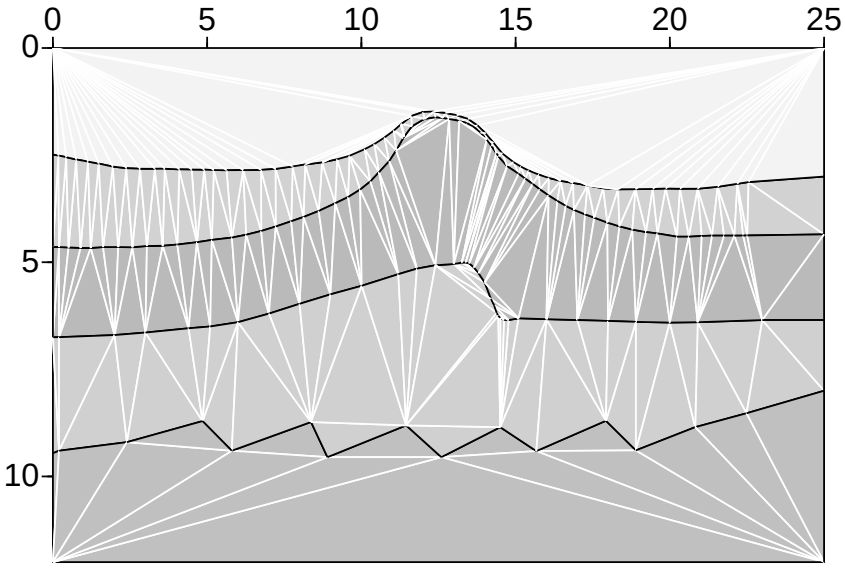

FIG. 7. Marathon model built by Gbmodel

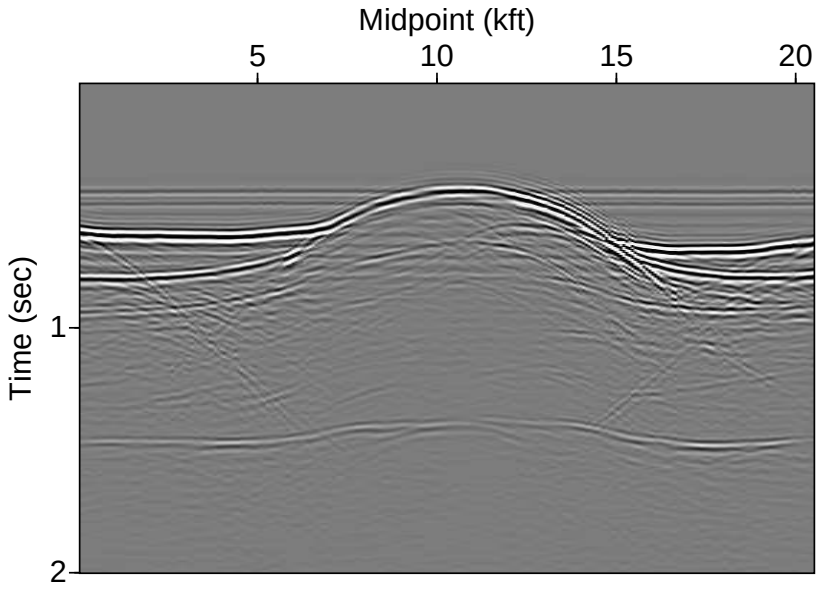

FIG. 8. Physical Tank Data (offset = 4080 (ft))

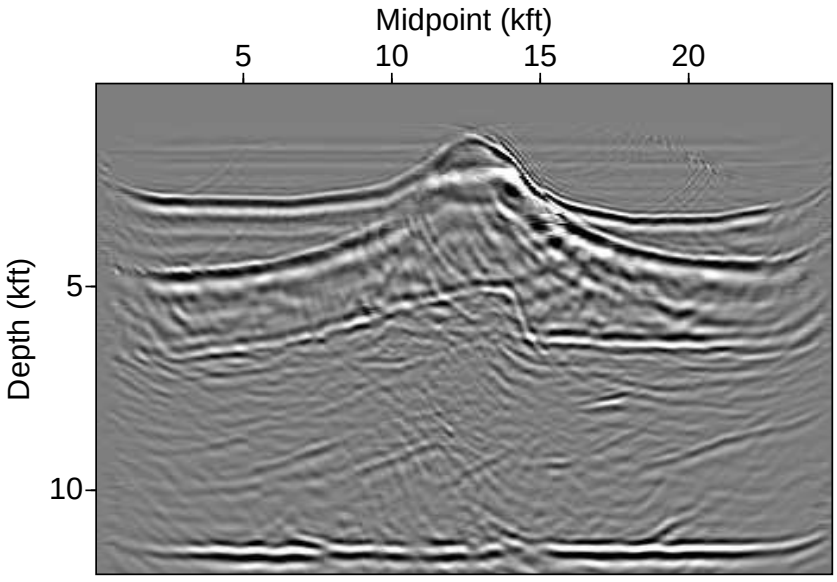

FIG. 9. Migration result of physical tank data (offset = 4080 ft Skip = 5)

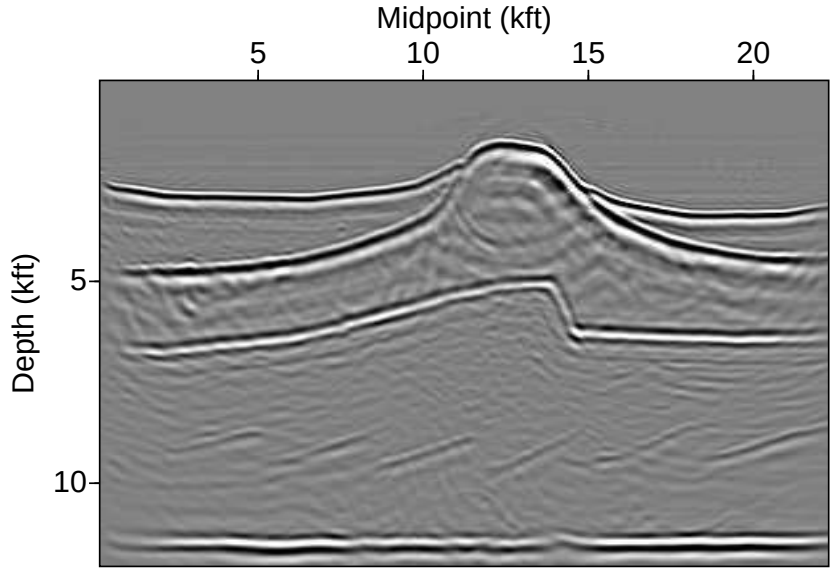

FIG. 10. Stack migration data which is generated by stacking five common offset data. (Skip = 5)

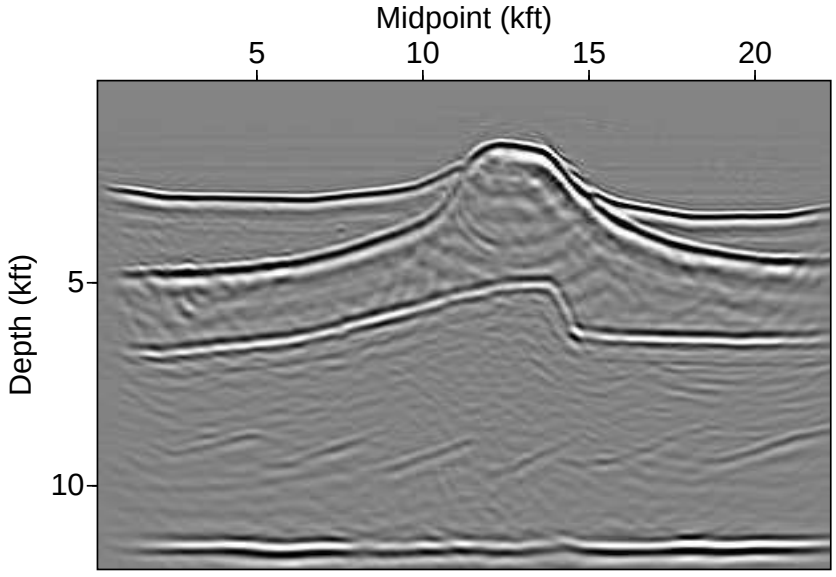

FIG. 11. Stack migration data (Skip = 15)

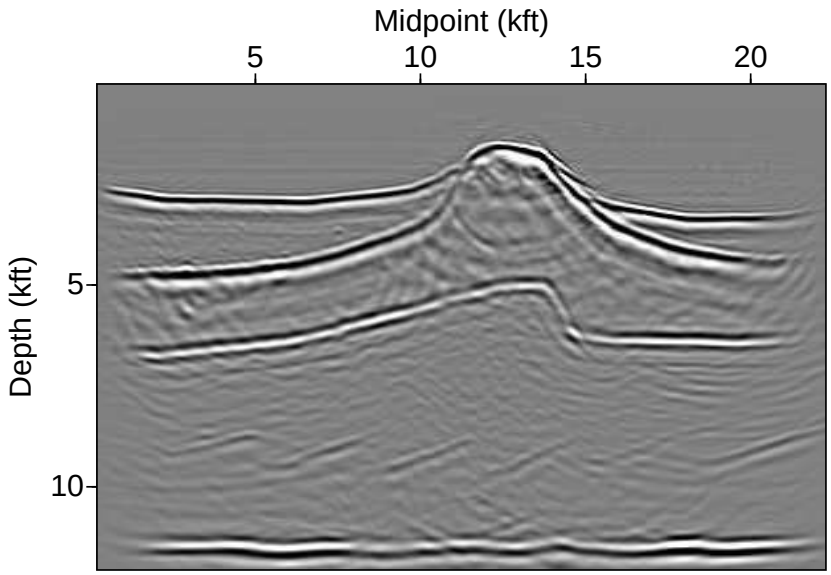

FIG. 12. Stack migration data (Skip = 20)

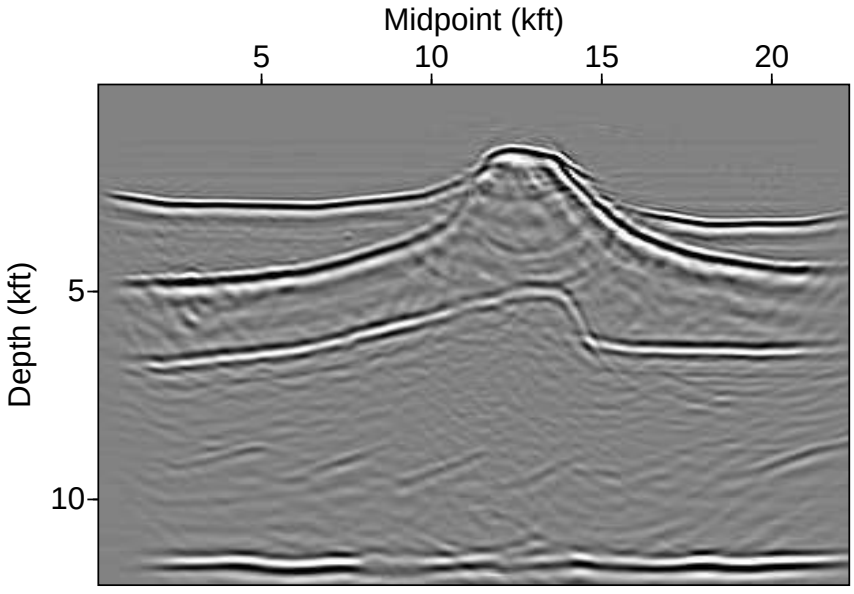

FIG. 13. Stack migration data (Skip = 25)

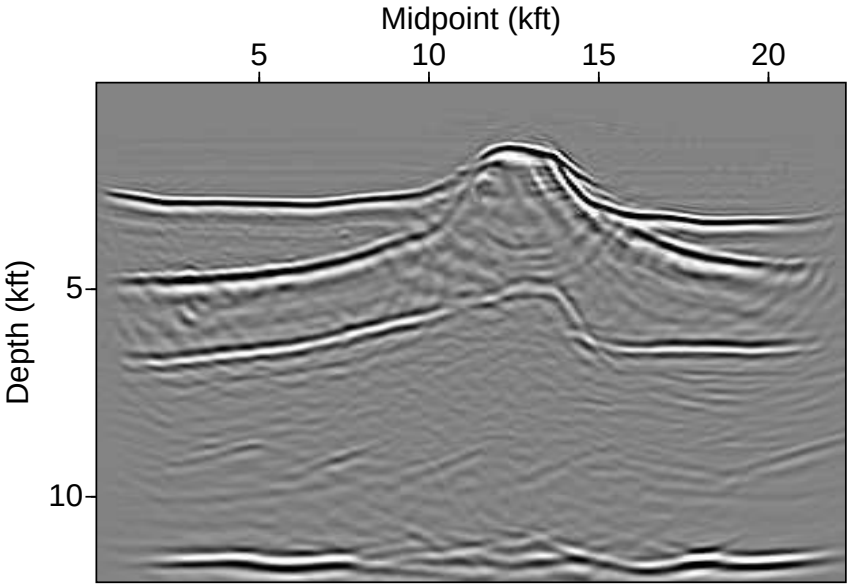

FIG. 14. Stack migration data (Skip = 30)

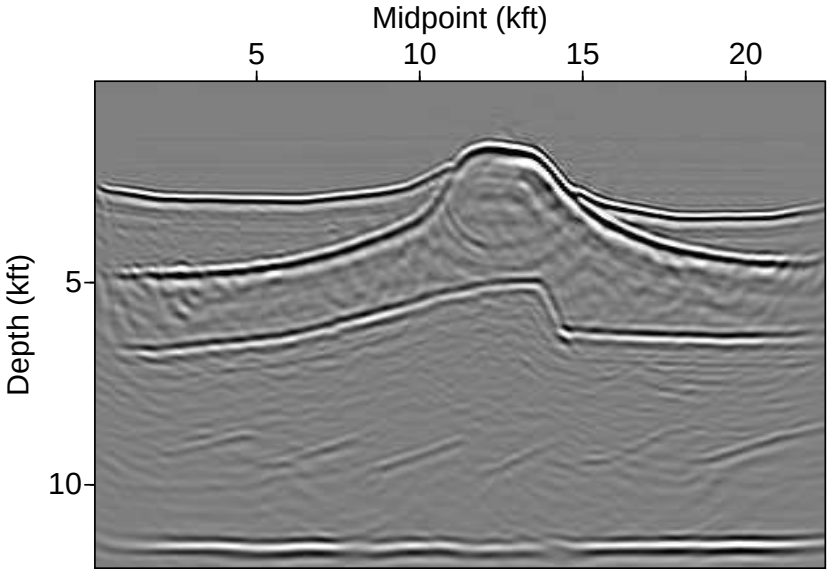

FIG. 15. Stack inversion data which is generated by Suinvvxz developed by Zhenyue Liu, this method use finite difference solve eikonal equation to accomplish inversion