3D Traveltime Computation Using Second–Order ENO Scheme

Seongjai Kim* and Richard Cook†

Abstract

We consider a second-order finite difference scheme to solve the eikonal equation. Upwind differences are requisite to sharply resolve discontinuities in the traveltime derivatives, while centered differences improve the accuracy of the computed traveltime. A second-order upwind *Essentially Non-Oscillatory* (ENO) scheme satisfies these requirements. It is implemented with a dynamic *Down 'N' Out* (DNO) marching, an expanding box approach. To overcome the instability of such an expanding box scheme, the algorithm is incorporated with an efficient *Post Sweeping* (PS), a correction-by-iteration method. Near the source, an efficient and accurate mesh-refinement initialization scheme is suggested for the DNO marching. The resulting algorithm, ENO-DNO-PS, turns out to be unconditionally stable, of second-order accuracy, and efficient; for various synthetic/real velocity models having large contrasts, two PS iterations produce traveltimes accurate enough to complete the computation.

1 Introduction

Seismic techniques based on high frequency asymptotic representation of the 3D acoustic Green's function require efficient solution methods for the computation of traveltimes. The use of ray tracing followed by interpolation of traveltimes is a popular and robust method for computing diffraction trajectories for small or moderate

* *Current Address*: Shell E&P Technology Co., P.O. Box 481, 3747 Bellaire Blvd., Houston, TX 77001-0481 USA Email: skim@shellus.com; *Permanent address*: Department of Mathematics, University of Kentucky, Lexington, KY 40506-0027, USA E–mail: skim@ms.uky.edu

† Shell E&P Technology Co., P.O. Box 481, 3747 Bellaire Blvd., Houston, TX 77001-0481 USA Email: cook@shellus.com

velocity contrasts. An alternative to the method is to compute traveltimes by solving directly the eikonal equation on a regular grid by finite difference schemes; see (Reshef and Kosloff, 1986; Vidale, 1988; Vidale, 1990; van Trier and Symes, 1991; Dellinger, 1991; Qin et al., 1992; Gray and May, 1994; Symes et al., 1994; Dellinger and Symes, 1997; Popovici and Sethian, 1997; Kim et al., 1997; Kim et al., 1998). For a comprehensive treatment of ray theory, see e.g. (Friedlander, 1958), (Cerveny et al., 1977), and (Cerveny, 1985).

A drawback of the finite difference eikonal solvers is that it is not easy to control the propagation angle as it is with ray tracing methods. Nontheless the finite difference eikonal solvers have their advantages: they are much more efficient computationally than ray tracers and rarely produce shadow zones. For regions with high velocity contrasts, ray tracing methods can produce quite large shadow zones where the computed traveltime field should be interpolated. Such ray–tracing/interpolation processes are cumbersome and computationally expensive, especially in three dimensions.

The finite difference eikonal solvers propagate the solution with either the *expanding box* strategy (Vidale, 1988; Vidale, 1990; Dellinger, 1991; Dellinger and Symes, 1997; Kim et al., 1998) or the *expanding wavefront* scheme (Qin et al., 1992; Popovici and Sethian, 1997). The expanding wavefront schemes are more expensive than the expanding box schemes, since the traveltimes in the wavefronts must be sorted by magnitude in each step of the computation; however, they are (unconditionally) stable because of their built-in causality. Even though the expanding box schemes can be unstable for turning rays, they have the following advantages: (1) they are computationally efficient and (2) they can be easily modified in order to achieve a high-order accuracy; see (Osher and Shu, 1991) and (Kim et al., 1998). The expanding wavefront schemes are first-order accurate and hardly extendible for a higher-order accuracy. The main goal of this paper is to suggest a second-order, expanding-box finite difference scheme and an efficient post processing scheme, called *Post Sweeping* (PS), as a remedy for the instability of the expanding box scheme.

The finite difference scheme we want to introduce is one member of the so called *Essentially Non–Oscillatory* (ENO) schemes proposed by (Osher and Sethian, 1988) and (Osher and Shu, 1991) for Hamilton–Jacobi initial–boundary value problems, and adopted by (Symes et al., 1994) and (El-Mageed, 1996; Kim et al., 1997) for the eikonal equation in 2D and 3D, respectively. We implement the algorithm in which the expanding box is dynamically expanded to best control the propagation angles of the rays. We call it the *dynamic Down 'N' Out* (DNO) marching. The basic idea of the DNO marching was first introduced in (Vidale, 1988) and adopted and named so by (Dellinger and Symes, 1997); both were implemented in a pre-determined (nondynamic) way.

The finite difference DNO marching schemes require an efficient and accurate initialization of the traveltime near the source. Since the wavefronts have a large curvature near the source, we need to pay a special attention to the initialization. We suggest an initialization scheme utilizing the *locally uniform mesh refinement* technique (Berger and Oliger, 1984; McCormick and Thomas, 1986; Bramble et al., 1988).

The PS approach can be viewed as a variant of the *correction-by-iteration* method suggested in (Schneider et al., 1992). There the correction-by-iteration scheme has been applied line by line, i.e. it corrects traveltimes for a grid line (using the Fermat's principle) until satisfying a given stopping criterion and move to the next grid line to perform the same as before. It continues through all the grid lines and returns back to the first grid line as many times as required to achieve global convergence. Here we use the idea in a different and more efficient way for PS.

Our resulting algorithm, ENO-DNO-PS, has been successfully tested for synthetic and real velocity models having large contrasts. It turns out to be unconditionally stable, second–order accurate, and efficient. The PS scheme converges so fast that two iterations produce traveltimes accurate enough to complete the computation for all models we have tested.

In the following section, we present the upwind second–order ENO scheme, details of the dynamic DNO marching, the mesh-refinement initialization, and the PS scheme. The next section shows numerical results demonstrating the accuracy and efficiency of ENO-DNO-PS. The algorithm is compared with a second-order correction-by-iteration method that is very similar to the one in (Schneider et al., 1992) incorporating the nonlinear interpolation scheme. The last section includes conclusions.

2 The algorithm

2.1 Finite difference scheme

The eikonal equation for an isotropic medium is given by

$$|\nabla\tau(\mathbf{x}_s,\mathbf{x})|^2 = \frac{1}{v^2(\mathbf{x})},$$

where $\tau(\mathbf{x}_s,\mathbf{x})$ denotes the traveltime of the acoustic wave from the source $\mathbf{x}_s$ to a subsurface point $\mathbf{x}$, and v is the wave velocity. It can be rewritten as an evolution equation in depth:

$$\frac{\partial\tau}{\partial z} = \sqrt{s^2 - \left(\frac{\partial\tau}{\partial x}\right)^2 - \left(\frac{\partial\tau}{\partial y}\right)^2}, \tag{2.1}$$

where $s\,(=1/v)$ is the slowness. For the eikonal solver for isotropic waves, we utilize the second-order upwind ENO Runge-Kutta scheme suggested in (Kim et al., 1997; Kim et al., 1998) (see also Appendix A):

$$\begin{aligned} \tau_{i,j}^{k+\frac{1}{2}} &= \tau_{i,j}^{k} + \Delta z_{\mathrm{CFL}}\cdot H[\tau]_{i,j}^{k}, \\ \tau_{i,j}^{k+1} &= \frac{1}{2}\Big(\tau_{i,j}^{k} + \tau_{i,j}^{k+\frac{1}{2}} + \Delta z_{\mathrm{CFL}}\cdot H[\tau]_{i,j}^{k+\frac{1}{2}}\Big), \end{aligned} \tag{2.2}$$

where

$$H[\tau]_{i,j}^{k} = \sqrt{\left(s_{i,j}^{k}\right)^2 - \left(\widehat{D}_x\tau_{i,j}^{k}\right)^2 - \left(\widehat{D}_y\tau_{i,j}^{k}\right)^2},$$

where $\widehat{D}_x\tau_{i,j}^{k}$ and $\widehat{D}_y\tau_{i,j}^{k}$ are respectively the upwind ENO differences for $\partial\tau/\partial x$ and $\partial\tau/\partial y$, and Δz_{CFL} is determined by the Courant-Friedrichs-Levy (CFL) condition presented in Appendix B. The difference $\widehat{D}_x\tau$ is a second-order ENO modification of the Godunov-type flux. It has been observed from a various numerical experiments that $\widehat{D}_x\tau$ is mostly-centered and sharply resolves shocks. Hence, the algorithm is of *mostly* second-order accuracy.

Since the velocity model is often provided in a uniform mesh of the cell size $(\Delta x, \Delta y, \Delta z)$, a certain number of substeps (of size Δz_{CFL}) should be imposed in a step (of size Δz) to satisfy the CFL condition. That is, the traveltime computation for a cell layer should be performed by marching several substeps of size Δz_{CFL}. The number of substeps increases when rays propagate at large angles to the vertical.

Large propagation angles are needed for imaging steeply dipping reflectors and practical eikonal solvers must efficiently compute accurate traveltimes for large propagation angles. To accomplish this, we limit the number of substeps (equivalently, the

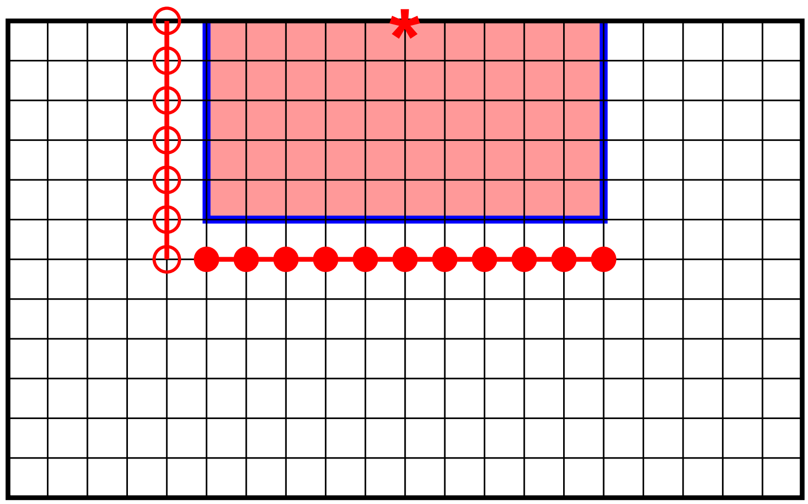

Figure 1: The expanding box of the dynamic DNO marching. The traveltimes at the points in the shaded area are already determined.

propagation angles), take care to ensure that traveltimes are smoothly over-estimated when the CFL condition is violated, and then correct the over-estimated traveltimes by propagating the solution in different directions. The remainder of this section describes these features.

2.2 Down 'N' Out (DNO)

Suppose that the traveltimes on the shaded area (up to its boundary) in Figure 1 are already determined. We call the boundary of the area the *expanding box* and the plane of the expanding box facing the $(c\pm)$-direction will be called the $(c\pm)$-*plane*, where c is x, y, or z.

The solution for the traveltimes proceeds by marching the expanding box until exhausting the whole computational domain. One may consider pre-determined (nondynamic) marching directions as in (Dellinger and Symes, 1997), e.g.

$$(z+),\ (x-),\ (x+),\ (y-),\ (y+),\ (z+),\ (x-),\ \cdots.$$

However, the nondynamic DNO marching is yet to be improved. For the simplest nonhomogeneous velocity model: $v = v(z) = v_0 + v_z z$, $v_z > 0$, for example, rays except the vertical one are turning out and up. The larger the shooting angle is, the more quickly the ray turns up. In this case more vertical steps then horizontal ones should precede to control the ray angles (and to compute traveltimes) in a more efficient way. Here we want to suggest a dynamic way of DNO marching for the traveltime computation in general media in which the new marching direction is found by checking ray angles on the edges of the expanding box. The scheme is given as follows:

- For all points on the edges of the $(z+)$-plane that are not on the boundary of the computational domain, compute the z-derivatives and outer derivatives of the traveltime. By the outer derivatives we mean the x-derivatives and the y-derivatives on the edges in the $(x\pm)$-directions and the $(y\pm)$-directions, respectively.
- Along each edge of the $(z+)$-plane, count the number of points at which the outer derivative is greater than the z-derivative, and divide it by the total number of points on the edge. We call the resulting four numbers the *outgoing indices*. (The outgoing index for the $(z+)$-direction is a prescribed number, e.g. 0.5. For the plane of the expanding box that have reached the boundary of the computational domain, their outgoing indices are set to be larger than 1.)
- Choose the marching direction in which the outgoing index is the smallest.

Suppose that the new marching direction is found to be $(z+)$. Then, the plane of solid circles in Figure 1 will be timed by the down marching of the algorithm (2.2). Now, we compute the outgoing indices to find the next marching direction; suppose $(x-)$ is the one. Then, the plane of open circles in Figure 1 will be timed by the algorithm marching in the $(x-)$ direction. Since the outside of the expanding box is not yet timed, the finite differences should be one-sided at the points on the edges of the expanding box.

2.3 Initialization

Initialization of the traveltimes in a small area near the source is an interesting research subject. Since the wavefront has a large curvature near the source, more efficient or higher–order formulas are required to get accurate traveltimes.

It has been observed that the finite difference scheme produces a large error when it is applied from near the source, while the error is small when it is applied from at

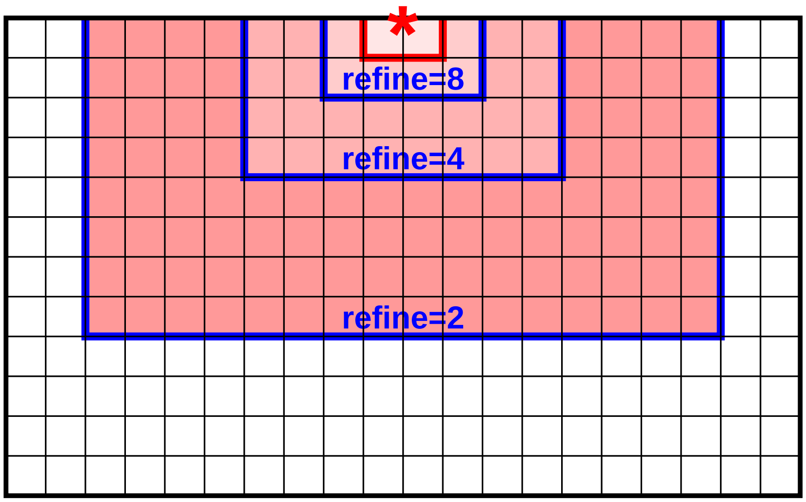

Figure 2: The mesh refinement near the source for traveltime initialization.

least 4 to 5 cell–layers away from the source; see also (Vidale, 1990). This observation has motivated the usage of *locally uniform mesh refinement* (LUMR) strategy near the source. In the literature, LUMR has been studied for the accurate and efficient solution of transient problems (Berger and Oliger, 1984; Trompert and Verwer, 1993) and elliptic problems (McCormick and Thomas, 1986; Bramble et al., 1988). Many applications of LUMR have been considered with, in particular, the multilevel schemes such as multigrid methods and domain decomposition methods; see articles in the proceedings (Keys and Xu, 1994).

For simplicity of the description, assume that the source is at a grid point on the top surface and is far from the edges of the top surface, as shown in Figure 2. The following argument is easily extended to more general source locations.

(0) <u>Analytic solver for $(2 \times 2 \times 1)$ cells:</u>
We assume that the velocity is linear in $(2 \times 2 \times 1)$ cells nearest to the source. (See Figure 2.) Then, the all points on the box of these four cells are timed by calling an analytic solver. This calculation uses the linear velocity function that best fits the given velocities in the four cells surrounding the source.

(1) <u>Refine=8 for $(4 \times 4 \times 2)$ cells:</u>
Consider the box of $(4 \times 4 \times 2)$ cells which includes the source at the top center. We refine each cell to $(8 \times 8 \times 8)$ subcells so that the box can have $(32 \times 32 \times 16)$

subcells and $(33 \times 33 \times 17)$ (sub)grid points that are to be timed in the current step. We first compute the traveltimes at the (sub)grid points that are on the boundary of the box considered in step (0), by an analytic solver. Then, apply the ENO-DNO algorithm to the remained part of the refined problem, after interpolating the velocity.

(2) Refine=4 for $(8 \times 8 \times 4)$ cells:
Now, we consider the box of $(8\times 8\times 4)$ original cells that again locates the source at its top center. In this step, each cell is refined to $(4 \times 4 \times 4)$ subcells; still the problem size is $(33 \times 33 \times 17)$, the same as before. The outermost traveltimes of the former step clearly provide the initialization for the current step traveltime computation; we perform the ENO-DNO marching for the remainder, after interpolating the velocity field.

(3) Refine=2 for $(16 \times 16 \times 8)$ cells:
For the last step of the initialization, choose the box of $(16 \times 16 \times 8)$ original cells, with each cell refined to $(2 \times 2 \times 2)$ subcells. Again the problem size is the same as before and the velocity interpolation and the ENO-DNO marching can also be performed in the same way.

Note that in each of steps (1)–(3) the ENO-DNO algorithm is applied from eight (sub)cell-layers away from the source, which surely helps the algorithm not produce a large error near the source. Furthermore, since the above initialization scheme uses finer meshes around the source, we also can enjoy a more accurate traveltimes near the source, and therefore, over the whole computational domain.

2.4 Post sweeping (PS)

For more general velocity models in which rays change their propagation directions sharply and/or more than once, the solution of the DNO marching (dynamic or nondynamic) may not be stable; the instability appears in a form of over-estimation of traveltimes. For a numerical test, we choose a high-contrast velocity model where a cube of a lower velocity ($v = 2000$ m/s) is embedded inside the computational domain of a constant background velocity ($v = 4500$ m/s). As shown in Figure 3(a), ENO-DNO has over-estimated the traveltimes below the lower velocity zone. The over-estimation (instability) can happen when rays propagate with a large angle from the

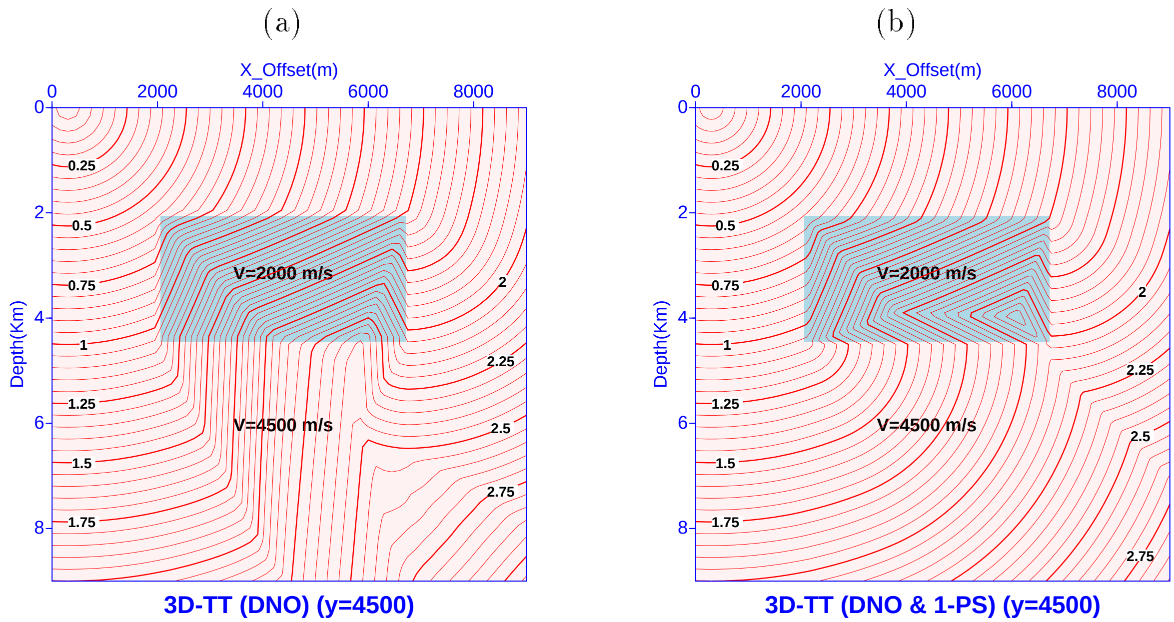

Figure 3: The finite difference traveltimes for the dynamic DNO marching (a) without PS iteration and (b) with one PS iteration, for the velocity model where a cube of a lower velocity ($v = 2000$ m/s) is embedded inside the computational domain of a constant background velocity ($v = 4500$ m/s).

finite difference marching direction. To cure the instability, we introduce the following iteration called *Post Sweeping* (PS), to be performed after the DNO marching:

Set $\tau^{(0,0)}$ to be the solution of the DNO marching;
Choose a stopping criterion ε;
Select directions: DIR(:)= $(x+)$, $(x-)$, $(y+)$, $(y-)$, $(z+)[$, $(z-)]$;
Do $j = 0, 1, \cdots$
  For each DIR(i)
    march the algorithm in the DIR(i) direction to get $\tau^{(i,j)}$;
    set $\tau^{(i,j)} = \min(\tau^{(i,j)}, \tau^{(i-1,j)})$;
  End for
  Set $\tau^{(0,j+1)} = \tau^{(i_{\max},j)}$;
  If $|\tau^{(0,j+1)} - \tau^{(0,j)}| \le \varepsilon$, stop;
End do

We have tested the resulting algorithm, ENO-DNO-PS, for various velocity models having large contrasts as in those of the Gulf of Mexico. For all velocity models we have tested, traveltimes after two PS iterations become accurate enough and the third iteration changes the solution in a negligible amount. For the lower-velocity-embedded model, the second iteration does not improve the solution; the traveltime after one PS iteration is accurate enough and plotted in Figure 3(b).

3 Numerical results

This section presents numerical experiments for the traveltime computation using ENO-DNO-PS and LUMR initialization. It is successfully tested for real velocity models from the Gulf of Mexico having large contrasts as well as synthetic models. Its computational cost and accuracy are compared with those of the 3D Minimum Traveltime Algorithm (MTA3D), a second-order algorithm similar to the one in (Schneider et al., 1992) incorporating the nonlinear interpolation scheme. The traveltime computation is performed on a single processor of an Ultra Sparc (300MHz) machine.

3.1 Linear model

To test accuracy and efficiency, we begin with the linear velocity model:

$$v(x, y, z) = 1000 + 0.4x + 0.2y + 0.8z \text{ m/s}.$$

The computational domain is $(0, 5000\text{m})^3$. Hence the velocity has the minimum value 1000 and the maximum value 8000. The uniform grid size is selected: $\Delta x = \Delta y = \Delta z =: h$. We choose the grid size $h = 50$ m so that the problem has $101 \times 101 \times 101$ grid points to be timed. The source is put at the center of the top, i.e. $\mathbf{x}_s = (2500, 2500, 0)$. Define the numerical traveltime "error" in millisecond (ms) as follows:

$$\text{error} = 1000 \cdot (\tau_{\text{computed}} - \tau_{\text{analytic}}).$$

In Figure 4, the traveltime errors (computed with the dynamic DNO marching) are plotted for an analytic initialization of (a) one and (b) five cell-distances from the source; the maximum errors are 2.3 and 0.4 ms respectively. One can see from the figure that most error appears near the source and it contaminates the whole domain. This has been observed independently on the mesh size h, which implies that

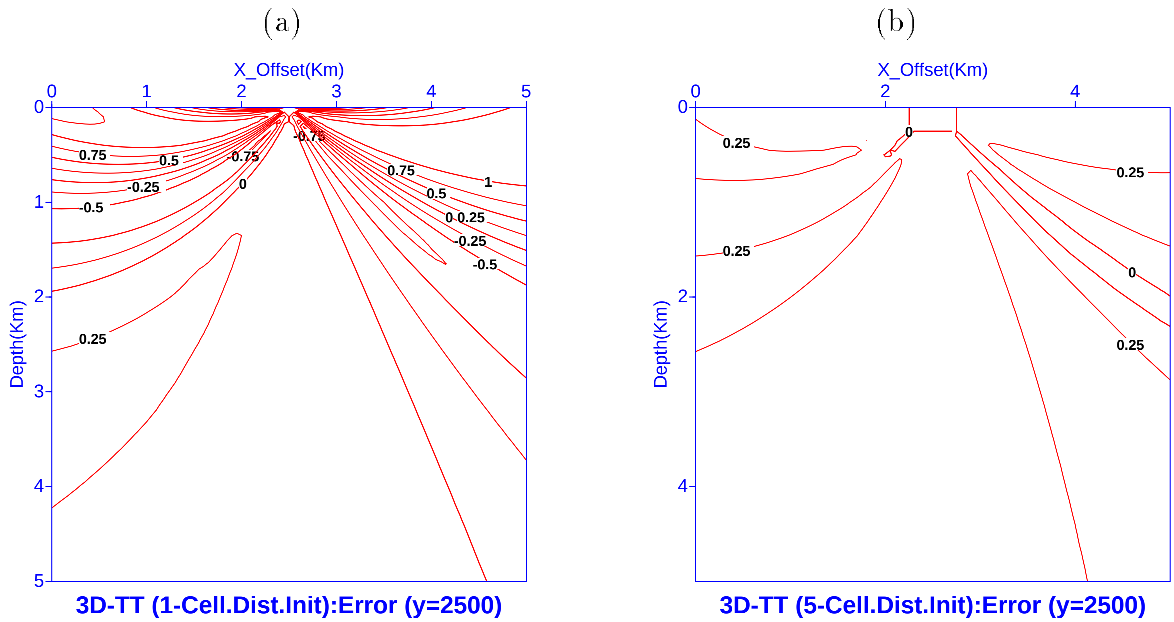

Figure 4: The traveltime errors (in ms) for the DNO algorithm. In this computation, the traveltimes are initialized on the box of (a) one and (b) five cell-distances, using an analytic solver. The maximum traveltime errors are 2.3 and 0.4 ms for (a) and (b), respectively.

the second-order finite difference scheme should be applied from a certain number of cell-distances from the source to maintain a high accuracy. We divide the cells into subcells in order for the finite difference scheme to start from a large number of *sub*cell-distances from the source. This is the basic idea of the LUMR initialization presented in the previous section. Figure 5 includes the traveltime and the error for the ENO-DNO scheme with the LUMR initialization. The initialization takes 0.15 seconds, the remainder takes 2.98 seconds (3.14 seconds for the whole computation), and the maximum error is 0.34 ms (0.027 %). It should be noticed that the LUMR initialization is quick (less than 5% of whole computation time) and produces very accurate traveltimes near the source; compare Figure 5(b) with Figure 4(b).

To check the efficiency of the DNO strategy, the algorithm is allowed to choose the number of substeps not larger than 40. The dynamic DNO marching takes 3.14 seconds (including the LUMR initialization), while the pre-determined DNO marching takes 9.22 seconds. For the down marching with five cell-layer initialization of analytic values, it takes 27.57 seconds. (It is not stable though; see Figure 6.) The dynamic DNO marching is most efficient since its CFL condition requires a few num-

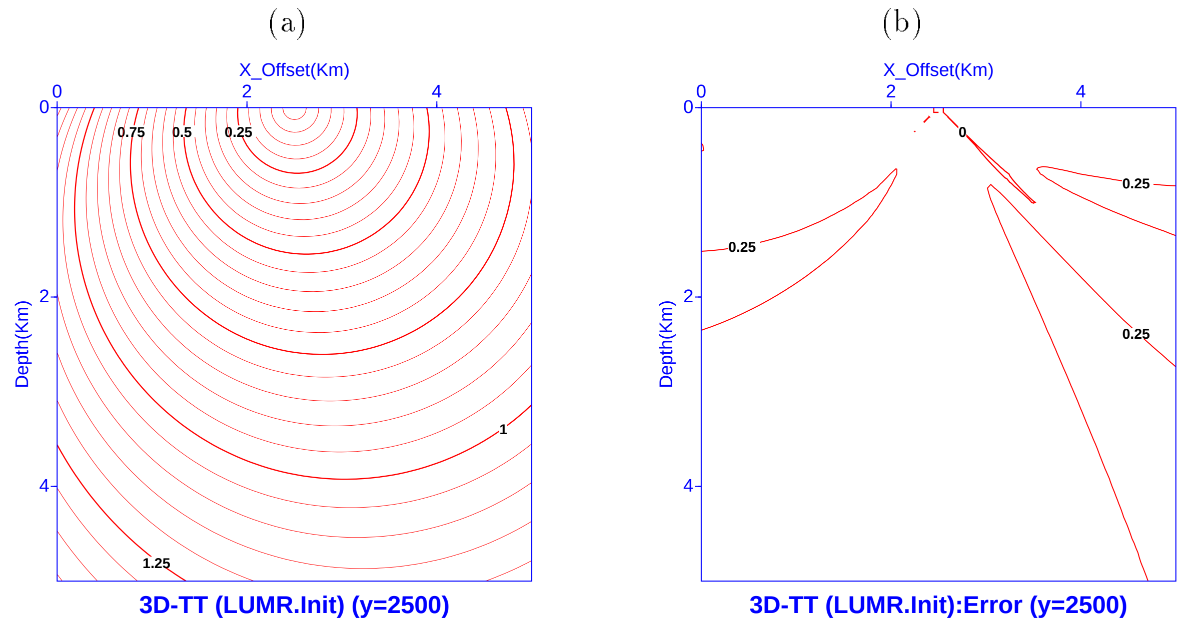

Figure 5: (a) The traveltime (in second) and (b) the error (in ms) for the dynamic DNO marching with the LUMR initialization. The maximum traveltime error is 0.34 ms (0.027 %) over the whole domain.

ber of substeps to march a cell-layer. In this experiment, the maximum numbers of substeps determined by the CFL condition are 4, 24, and 40 for the dynamic DNO marching, the pre-determined DNO marching, and the down marching, respectively. The efficiency of the dynamic DNO marching has been observed from various numerical experiments including real velocity models.

In Figure 7, we present the traveltime and the error for the down marching with five cell-layer initialization followed by one PS iteration. (The PS iteration is carried out with the maximum number of substeps not larger than 2.) The over-estimation (shown as in Figure 6) is clearly corrected even after one PS iteration. The PS iteration takes 4.30 seconds; the maximum traveltime error is 0.34 ms. The error in Figure 7(b) is comparable with that in Figures 4(b) and 5(b) over the whole domain.

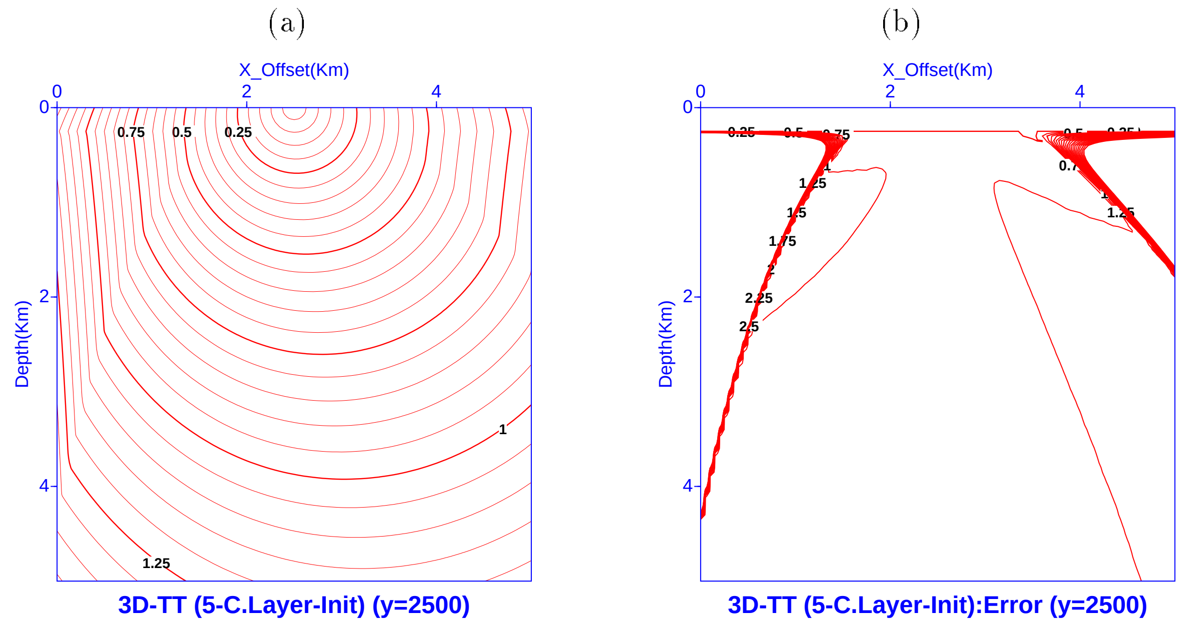

Figure 6: (a) The traveltime (in second) and (b) the error (in ms) for the down marching with five cell-layer analytic initialization. The contours for the error larger than 5 ms are not plotted.

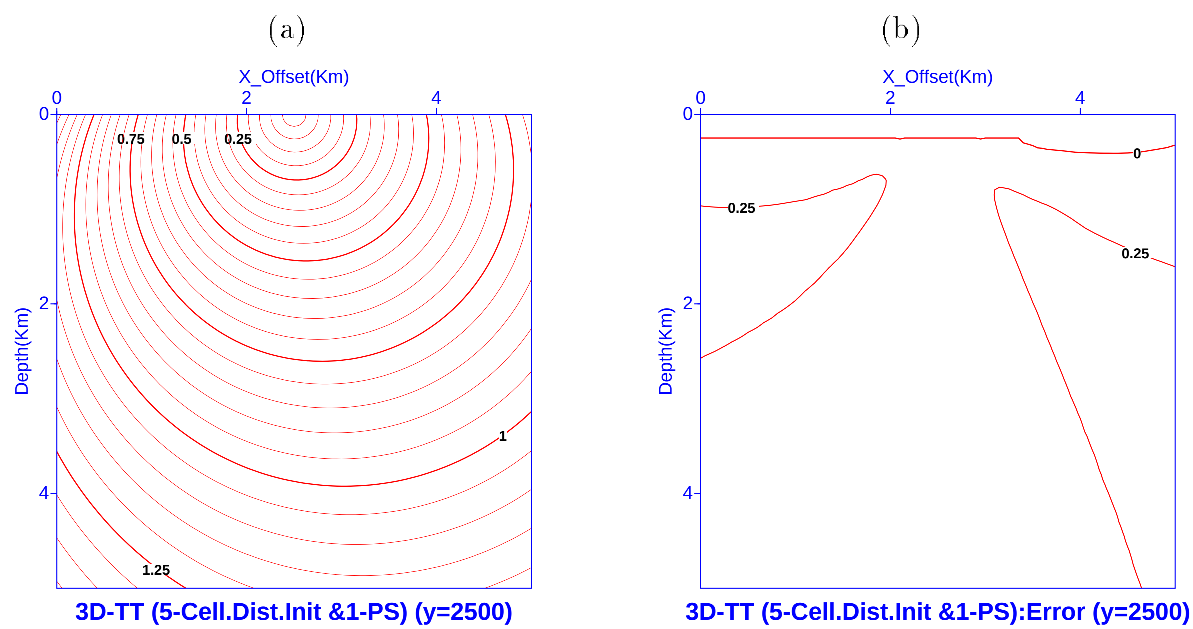

Figure 7: (a) The traveltime (in second) and (b) the error (in ms) for the down marching with five cell-layer initialization followed by one PS iteration. The maximum error is 0.34 ms over the domain.

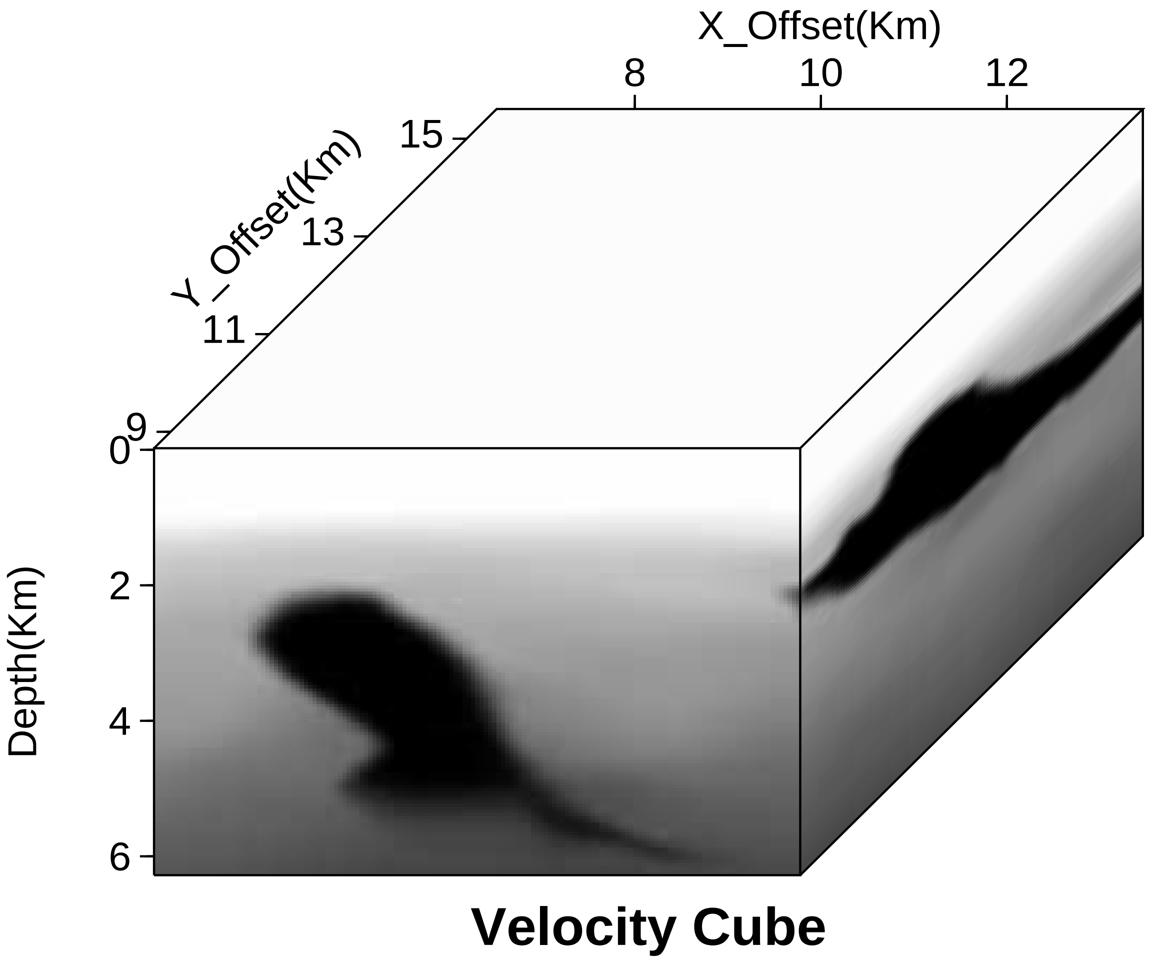

Figure 8: The real velocity cube. It is composed of $150 \times 150 \times 136$ cells; the edge length of the cell is 45.7 m (=150 ft).

3.2 Real velocity model

ENO-DNO-PS has been tested for a real velocity model from the Gulf of Mexico. Figure 8 shows a part of the model composed of $150 \times 150 \times 136$ cells (=3,123,737 grid points); each cell has its edge length of 45.7 m. A point source is located at the top center. After two PS iterations (with the maximum number of substeps chosen not larger than two), the traveltime becomes stable enough to complete the computation; the two PS iterations takes 37.2 seconds and the whole computation time is 49.7 seconds. The traveltime contours superposed on vertical and horizontal slices are plotted in Figures 9 and 10, respectively.

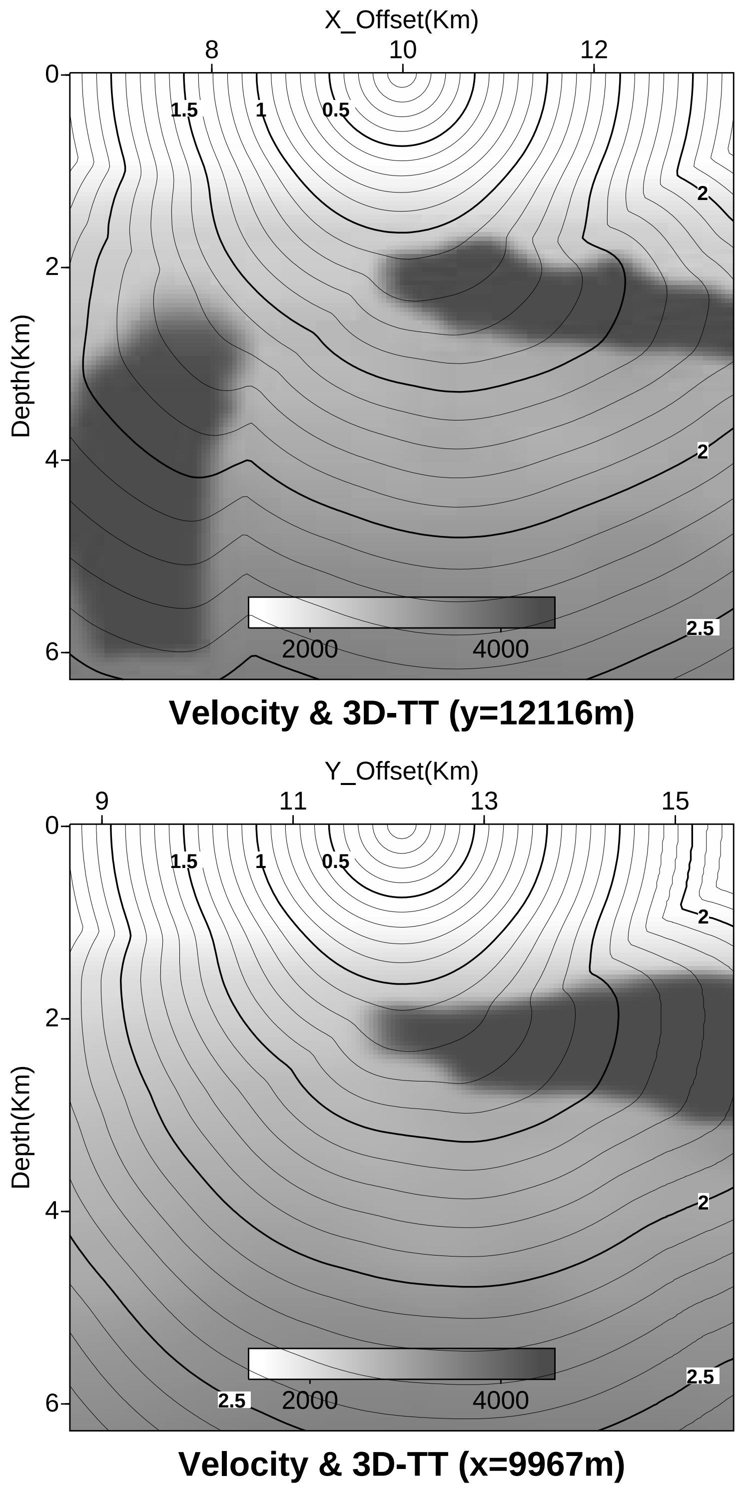

Figure 9: Traveltime contours are superposed on vertical slices of the real velocity model. Top is the slice in x-direction $\{y = 12116 \text{ m}\}$ and the bottom is the slice in y-direction $\{x = 9967 \text{ m}\}$.

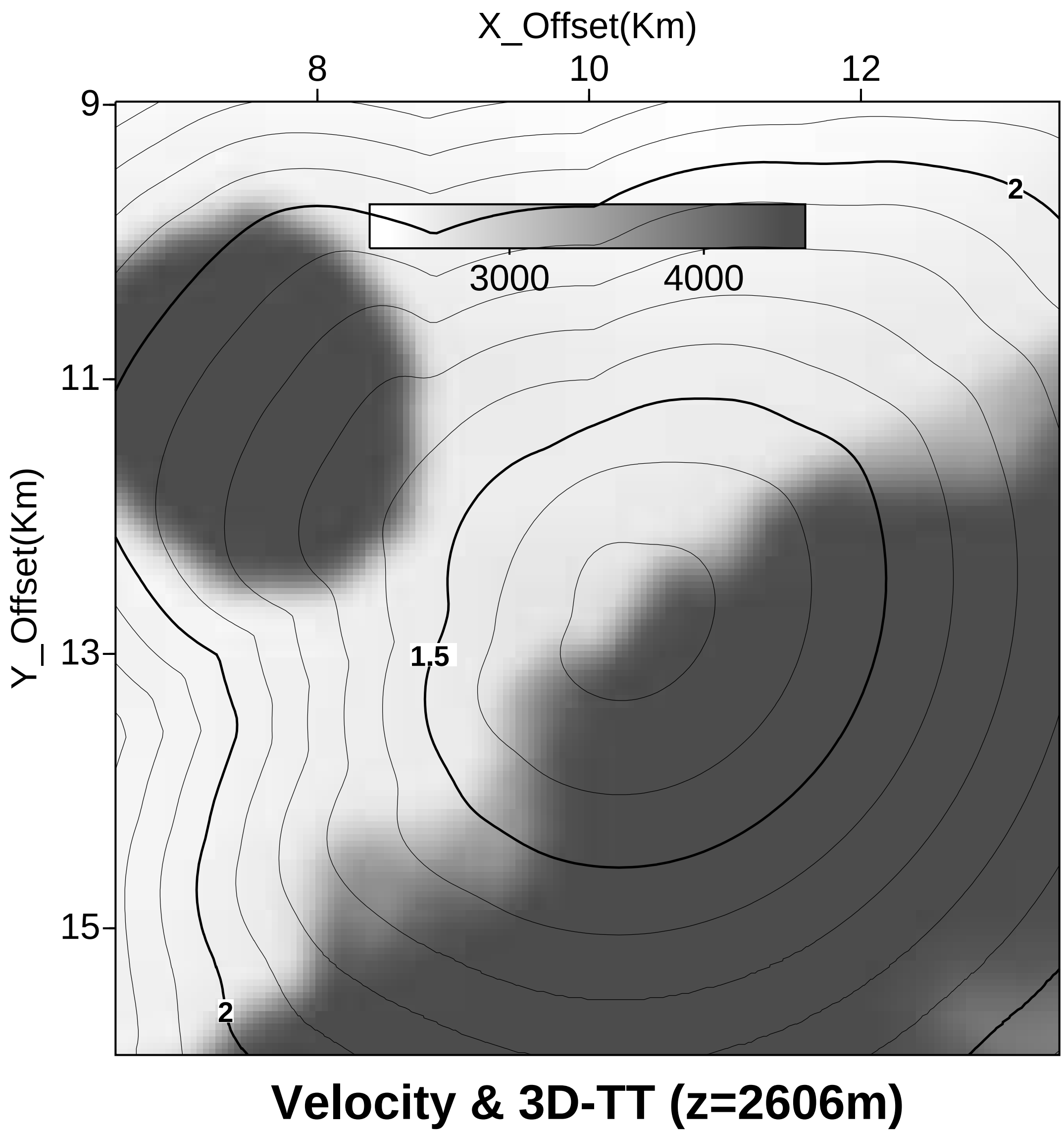

Figure 10: Traveltime contours are superposed on the horizontal slice of the real velocity model $\{z = 2606 \text{ m}\}$.

The PS iteration is not expensive when the maximum number of substeps is chosen to be small. The convergence of PS iteration seems showing a weak sensitivity to the maximum number of substeps; we choose 2 for it! For various velocity models, two PS iterations seems enough for the solution to become stable and accurate; the third iteration improves the traveltimes in a negligible amount. Such a fast convergence can be explained as follows. After the DNO marching, the solution is mostly accurate, in particular, near the source and inside a cone having its vertex on the source and its base on the bottom of the model, since the rays are mostly downgoing. The $(x+)$ and $(x-)$ sweeps correct the solution in areas where the ray angles from the x-axis are not larger than a given angle (e.g. 60 degrees which corresponds to two marching substeps as the upper limit). So the traveltime becomes accurate at most grid points of a vertical slab containing the source, parallel to xz-plane. Then, the $(y+)$ and $(y-)$ sweeps can expand the accuracy to the whole domain in the same manner the $(x+)$ and $(x-)$ sweeps do. The $(z+)$ sweep corrects the solution for the rays that first propagate almost horizontally and then change the direction downward almost vertically. (It is hardly observed that the rays turn upward almost vertically; the $(z-)$ sweep may not be necessary.) These five (or six) directional sweeps finish one PS iteration and correct the traveltime accurately at most grid points.

ENO-DNO-PS computes the traveltimes about 200 times faster than MTA3D for the linear model. Algorithm MTA3D seems unstable for velocity models having large lateral and vertical gradients; for various grid sizes, its traveltime error becomes large (≥ 10 ms) near the top surface and the upper quarter of the vertical boundaries of the computational domain. For the real velocity model (that includes large contrasts but has small lateral gradients in most of the computational domain), MTA3D converges *relatively* faster; ENO-DNO-PS is still 25 times faster than MTA3D. For the real model, MTA3D is comparable to ray-tracers in computation time. ENO-DNO-PS has been at least 20 times faster than MTA3D for all velocity models we have tried.

4 Conclusions

We have considered a second-order upwind ENO scheme for the first-arrival traveltimes by directly solving the eikonal equation. The ENO scheme is implemented with a dynamic Down 'N' Out (DNO) marching and Post Sweeping (PS) iteration. Through the DNO-PS approach, we have been able to update the traveltimes accurately and efficiently, in particular, in the areas of sharply turning rays. ENO-DNO-PS turns out to be unconditionally stable, second-order accurate, and efficient; it shows accuracy/efficiency advantages over ray-tracing methods, expanding-wavefront schemes, and other expanding-box schemes. It has been successfully applied to real velocity models having large contrasts, anisotropic traveltime computation, and prestack Kirchhoff depth migration. These would be subjects for a subsequent article.

Acknowledgement

We thank Dr. T. Padhi for his useful comments and encouragement, and appreciate research management of Shell Offshore Inc. for allowing us to use a real model and publish related results. Some of the ideas in this article were developed when SK was a postdoctoral research associate at Rice University, under guidance of Prof. W.W. Symes. SK appreciates WWS's guidance and wisdom.

References

Berger, M. J. and Oliger, J. (1984). Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512.

Bramble, J. H., Ewing, R. E., Pasciak, J. E., and Schatz, A. H. (1988). A preconditioning technique for the efficient solution of problems with local grid refinement. *Comp. Meths. Appl. Mech. Eng.*, 67:149–159.

Cerveny, V. (1985). Ray synthetic seismograms for complex two- and three-dimensional structures. *J. Geophys.*, 58:2–26.

Cerveny, V., Molotkov, I. A., and Psencik, I. (1977). *Ray Methods in Seismology.* University of Karlova Press, Prague.

Dellinger, J. (1991). Anisotropic finite–difference traveltimes. In *The 61st Annual International SEG Meeting*, pages 1530–1533. Society of Exploration Geophysicists. Expanded abstract.

Dellinger, J. and Symes, W. W. (1997). Anisotropic finite–difference traveltimes using a Hamilton–Jacobi solver. In *The 67th Annual International SEG Meeting*, pages 1786–1789. Society of Exploration Geophysicists. Expanded abstract.

El-Mageed, M. A. (1996). *3D first arrival traveltimes and amplitudes via Eikonal and Transport finite difference solvers.* Ph.D. Thesis, Department of Computational and Applied Mathematics, Rice University, 6100 Main St., Houston Texas 77005, USA.

Friedlander, F. (1958). *Sound Pulses.* Cambridge University Press.

Gray, S. and May, W. (1994). Kirchhoff migration using eikonal equation traveltimes. *Geophysics*, 59:810–817.

Keys, D. E. and Xu, J., editors (1994). *Domain Decomposition Methods in Scientific and Engineering Computing*, volume 180 of *Contemporary Mathematics*. American Mathematical Society, Providence, RI. Proceedings of the Seventh International Conference on Domain Decomposition Methods, October 27–30, 1993, The Pennsylvania State University.

Kim, S., Symes, W. W., and El-Mageed, M. A. (1997). 3D Kirchhoff migration using finite difference traveltimes and amplitudes. The Rice Inversion Project annual report, article #5, Department of Computational and Applied Mathematics, Rice University, 6100 Main St., Houston, TX 77005, USA.

Kim, S., Symes, W. W., and El-Mageed, M. A. (1998). Superconvergent difference formulas for traveltimes and amplitudes. In DeSanto, J. A., editor, *Fourth International Conference on Mathematical and Numerical Aspects of Wave Propagation*, pages 591–593, Philadelphia. SIAM.

McCormick, S. and Thomas, J. (1986). The fast adaptive composite grid (FAC) method for elliptic equations. *Math. Comp.*, 45:439–456.

Osher, S. and Sethian, J. (1988). Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comp. Phys.*, 79:12–49.

Osher, S. and Shu, C.-W. (1991). High–order essentially nonoscillatory schemes for Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 28:907–922.

Popovici, A. M. and Sethian, J. (1997). Three dimensional traveltime computation using the fast marching method. In *Proc. 67th Annual Meeting, SEG*, pages 1778–1781, Dallas, TX, USA. Society of Exploration Geophysicists, Tulsa, OK.

Qin, F., Luo, Y., Olsen, K. B., Cai, W., and Schuster, G. T. (1992). Finite–difference solution of the eikonal equation along expanding wavefronts. *Geophysics*, 57:478–487.

Reshef, M. and Kosloff, D. (1986). Migration of common shot gathers. *Geophysics*, 51:324–331.

Schneider, W. A. J., Ranzinger, K. A., Balch, A. H., and Kruse, C. (1992). A dynamic programming approach to first arrival traveltime computation in media with arbitrarily distributed velocities. *Geophysics*, 57:39–50.

Strikwerda, J. C. (1989). *Finite Difference Schemes and Partial Differential Equations*. Wadsworth & Brooks/Cole, Pacific Grove, California.

Symes, W. W., Versteeg, R., Sei, A., and Tran, Q.-H. (1994). Kirchhoff simulation migration and inversion using finite–difference traveltimes and amplitudes. TRIP Annual Report, The Rice Inversion Project, Rice University, Houston, TX 77005.

Trompert, R. A. and Verwer, J. G. (1993). Analysis of the implicit Euler local uniform grid refinement method. *SIAM J. Sci. Comput.*, 14:259–278.

van Trier, J. and Symes, W. W. (1991). Upwind finite-difference calculation of traveltimes. *Geophysics*, 56(6):812–821.

Vidale, J. E. (1988). Finite-difference calculation of travel times. *Bull., Seis. Soc. Am.*, 78:2062–2076.

Vidale, J. E. (1990). Finite difference calculation of traveltimes in three dimensions. *Geophysics*, 55:521–526.

Appendix A The upwind ENO scheme

In this appendix, we present the second-order finite difference scheme for solving the eikonal equation in three-dimensional space, suggested in (Kim et al., 1997); see also (Kim et al., 1998). The scheme is a member of *Essentially Non-Oscillatory* (ENO) schemes introduced by (Osher and Sethian, 1988) for the solution of Hamilton-Jacobi initial-boundary value problems.

We begin with introducing a second-order ENO scheme for the point source at $\mathbf{x}_s$. For a function f of the space variable $\mathbf{x}$ in the computational domain, write

$$f_{i,j}^k = f(\mathbf{x}_{i,j}^k),$$

$$\mathbf{x}_{i,j}^k = (x_i, y_j, z_k) = (x_{\min} + (i-1)\Delta x, y_{\min} + (j-1)\Delta y, z_{\min} + (k-1)\Delta z).$$

Let $\tau_{i,j}^k = \tau(\mathbf{x}_{i,j}^k, \mathbf{x}_s)$ and define the forward ($+$) and backward ($-$) finite difference operators:

$$D_x^{\pm}\tau_{i,j}^k = \pm\frac{\tau_{i\pm1,j}^k - \tau_{i,j}^k}{\Delta x}.$$

Then, from Taylor expansion, we have

$$\frac{\partial \tau}{\partial x} = D_x^{\pm}\tau \mp \frac{1}{2}\Delta x \cdot \frac{\partial^2 \tau}{\partial x^2} + \mathcal{O}(\Delta x^2).$$

For the shock-capturing second-order finite difference scheme, we choose the ENO schemes of (Osher and Sethian, 1988) as particularly appropriate. First we approximate the second derivative in the right hand side of the above equation by

$$\frac{\partial^2 \tau}{\partial x^2} \approx \mathrm{m}\Big(D_x^- D_x^+ \tau,\ D_x^{\pm} D_x^{\pm} \tau\Big),$$

where

$$\begin{aligned} \mathrm{m}(x,y) &= \min\Big(\max(x,0), \max(y,0)\Big) + \max\Big(\min(x,0), \min(y,0)\Big) \\ &= \begin{cases} 0, & \text{if } xy \le 0, \\ x, & \text{if } |x| \le |y| \text{ and } xy > 0, \\ y, & \text{if } |x| > |y| \text{ and } xy > 0. \end{cases} \end{aligned}$$

Then, the second-order ENO finite differences for $\partial\tau/\partial x$ read

$$\frac{\partial \tau}{\partial x} \approx D_x^{\pm,2}\tau = D_x^{\pm}\tau \mp \frac{1}{2}\Delta x \cdot \mathrm{m}\Big(D_x^- D_x^+ \tau,\ D_x^{\pm} D_x^{\pm} \tau\Big).$$

Note that

$$\begin{aligned} D_x^- D_x^+ \tau_{i,j}^k &= \frac{\tau_{i-1,j}^k - 2\tau_{i,j}^k + \tau_{i+1,j}^k}{\Delta x^2}, \\ D_x^{\pm} D_x^{\pm} \tau_{i,j}^k &= \frac{\tau_{i,j}^k - 2\tau_{i\pm1,j}^k + \tau_{i\pm2,j}^k}{\Delta x^2}. \end{aligned}$$

These are the centered differences for $\partial^2\tau/\partial x^2$ for the points $\mathbf{x}_{i,j}^k$ and $\mathbf{x}_{i\pm1,j}^k$, respectively, and are related to the curvature of the traveltime field on the corresponding points. So, the function m chooses the least oscillatory component of the differences. The forward second-order ENO difference can be written as

$$D_x^{+,2}\tau_{i,j}^k = \begin{cases} D_x^+ \tau_{i,j}^k, & \\ D_x^+ \tau_{i,j}^k - \frac{1}{2}\Delta x \cdot D_x^- D_x^+ \tau_{i,j}^k = \dfrac{\tau_{i+1,j}^k - \tau_{i-1,j}^k}{2\Delta x}, & \text{or} \\ D_x^+ \tau_{i,j}^k - \frac{1}{2}\Delta x \cdot D_x^+ D_x^+ \tau_{i,j}^k = \dfrac{1}{\Delta x}\Big(-\dfrac{3}{2}\tau_{i,j}^k + 2\tau_{i+1,j}^k - \dfrac{1}{2}\tau_{i+2,j}^k\Big). & \end{cases}$$

The last two cases of the above are the centered and one-sided second-order finite differences for $\partial\tau/\partial x$, respectively. Also the first seems to be a second-order approximation for a smooth solution, since it appears at inflection points where $\partial^2\tau/\partial x^2 = 0$. The same can be addressed for $D_x^{-,2}\tau_{i,j}^k$.

Now, define the second-order *upwind* ENO approximation for $\partial\tau/\partial x$ incorporating the first-arrivals:

$$\left(\frac{\partial\tau}{\partial x}\right)_{i,j}^{k} \approx \widehat{D}_x\tau_{i,j}^k = \text{mod\_max}\Big(\max\left(D_x^{-,2}\tau_{i,j}^k, 0\right), \min\left(D_x^{+,2}\tau_{i,j}^k, 0\right)\Big), \tag{A.1}$$

where mod_max returns the larger value in modulus.

After introducing the analogues for $\partial\tau/\partial y$, $D_y^{\pm}\tau_{i,j}^k$, $D_y^{\pm,2}\tau_{i,j}^k$, and $\widehat{D}_y\tau_{i,j}^k$, we can formulate the second-order ENO Runge-Kutta algorithm as follows:

$$\begin{aligned} \tau_{i,j}^{k+\frac{1}{2}} &= \tau_{i,j}^k + \Delta z_{\text{CFL}} \cdot H[\tau]_{i,j}^k, \\ \tau_{i,j}^{k+1} &= \frac{1}{2}\left(\tau_{i,j}^k + \tau_{i,j}^{k+\frac{1}{2}} + \Delta z_{\text{CFL}} \cdot H[\tau]_{i,j}^{k+\frac{1}{2}}\right), \end{aligned} \tag{A.2}$$

where

$$H[\tau]_{i,j}^k = \sqrt{\left(s_{i,j}^k\right)^2 - \left(\widehat{D}_x\tau_{i,j}^k\right)^2 - \left(\widehat{D}_y\tau_{i,j}^k\right)^2},$$

and Δz_{CFL} is to be determined to satisfy the CFL condition.

Remark. The finite difference scheme in (A.1) corresponds to the first-arrivals and it turns out to become a second-order ENO modification of Godunov-type flux. For a description of the Godunov scheme, see e.g. (Osher and Shu, 1991). Symes et al. (1994) and El-Mageed (1996) adopted the approximation

$$\left(\frac{\partial\tau}{\partial x}\right)_{i,j}^{k} \approx \sqrt{\left(\max\left(D_x^{-,2}\tau_{i,j}^k, 0\right)\right)^2 + \left(\min\left(D_x^{+,2}\tau_{i,j}^k, 0\right)\right)^2} \tag{A.3}$$

which was formerly suggested in (Osher and Sethian, 1988) for solving fluid flow problems. It is easy to see that for shock points, (A.3) may compute the traveltimes faster than the first-arrivals, i.e. under-estimated traveltimes. Such under-estimations can affect the solution in a large area. Popovici and Sethian (1997) utilized an analogous scheme to (A.3) for their Fast Marching Method, a first-order accurate, expanding-wavefront traveltime solver.

Appendix B Stability condition

For given consistent finite difference schemes solving evolutionary partial differential equations, the Courant-Friedrichs-Levy (CFL) stability condition can be viewed as the requirement that *the numerical domain of dependence contain the physical domain*

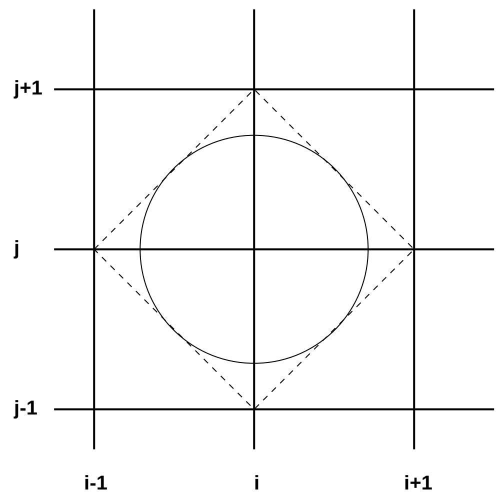

Figure 11: The numerical domain of dependence depicted in the xy-plane $\{z = z_k\}$. The thick straight lines indicate the grid lines and the dashed box is the numerical domain of dependence for the grid point (x_i, y_j, z_k), that is, $\{(x, y, z_k) : |x - x_i|/\Delta x + |y - y_j|/\Delta y \leq 1\}$. The circle is inscribed in the box.

of dependence. This physical interpretation of the CFL condition is advantageous in simplicity, i.e. it is easy to check, while the von Neumann condition can be too complicated to check for most numerical schemes solving general partial differential equations.

One should avoid choosing very small step size in the z-direction. Why? First of all, a small step size increases the computational cost. Secondly (with more importance), it may cause a severe numerical diffusion/dispersion (Strikwerda, 1989, Ch.5), since it diminishes the physical domain of dependence and therefore the numerical domain of dependence is relatively enlarged.

Obviously, the numerical domain of dependence for algorithms (A.2), evolving from the k-th level to the $(k+1)$-th level at the grid point (x_i, y_j, z_k), is

$$\left\{(x, y, z_k) : \frac{|x - x_i|}{\Delta x} + \frac{|y - y_j|}{\Delta y} \leq 1\right\}$$

and the radius of the inscribed circle is $\Delta x \Delta y / \sqrt{\Delta x^2 + \Delta y^2}$; see Figure 11. The physical domain of dependence for algorithm (A.2) is

$$\Delta z_{\text{CFL}} \cdot \frac{\text{change of } \tau \text{ on the } xy\text{-plane}}{\text{change of } \tau \text{ in the } z\text{-direction}}.$$

So, the CFL condition for algorithm (A.2) reads

$$\Delta z_{\mathrm{CFL}} \cdot \max_{i,j} \left[\frac{\sqrt{(\widehat{D}_x \tau_{i,j}^k)^2 + (\widehat{D}_y \tau_{i,j}^k)^2}}{H[\tau]_{i,j}^k} \right] \leq \frac{\Delta x \Delta y}{\sqrt{\Delta x^2 + \Delta y^2}}.$$

That is, Δz_{CFL} can be chosen to satisfy

$$\Delta z_{\mathrm{CFL}} \ \leq \ \min_{i,j} \left[\frac{H[\tau]_{i,j}^k}{\sqrt{(\widehat{D}_x \tau_{i,j}^k)^2 + (\widehat{D}_y \tau_{i,j}^k)^2}} \right] \cdot \frac{\Delta x \Delta y}{\sqrt{\Delta x^2 + \Delta y^2}}. \tag{B.1}$$

The practical efficiency of the ENO scheme seems to strongly depend on the ability of the numerical code to control the step size Δz_{CFL} dynamically. One has to choose Δz_{CFL} large enough to reduce the computational cost and undesirable numerical diffusion and, on the other hand, small enough to satisfy the CFL condition which possibly varies on each step.