Stanford Exploration Project, Report SERGEY, November 9, 2000, pages 455–??

A variational formulation of the fast marching eikonal solver

Sergey Fomel[1]

ABSTRACT

I exploit the theoretical link between the eikonal equation and Fermat's principle to derive a variational interpretation of the recently developed method for fast traveltime computations. This method, known as fast marching, possesses remarkable computational properties. Based originally on the eikonal equation, it can be derived equally well from Fermat's principle. The new variational formulation has two important applications: First, the method can be extended naturally for traveltime computation on unstructured (triangulated) grids. Second, it can be generalized to handle other Hamilton-type equations through their correspondence with variational principles.

> Now we are in the rarefied atmosphere of theories of excessive beauty and we are nearing a high plateau on which geometry, optics, mechanics, and wave mechanics meet on a common ground. Only concentrated thinking, and a considerable amount of re-creation, will reveal the full beauty of our subject in which the last word has not been spoken yet.–Cornelius Lanczos, *The variational principles of mechanics*

INTRODUCTION

Traveltime computation is one of the most important tasks in seismic processing (Kirchhoff depth migration and related methods) and modeling. The traveltime field of a fixed source in a heterogeneous medium is governed by the eikonal equation, derived about 150 years ago by Sir William Rowan Hamilton. A direct numerical solution of the eikonal equation has become a popular method of computing traveltimes on regular grids, commonly used in seismic imaging (Vidale, 1990; van Trier and Symes, 1991; Podvin and Lecomte, 1991). A recent contribution to this field is the *fast marching* level set method, developed by Sethian (1996a) in the general context of level set methods for propagating interfaces (Osher and Sethian, 1988; Sethian, 1996b). Sethian and Popovici (1997) report a successful application of this method in three-dimensional seismic computations. The fast marching method belongs to the family of upwind finite-difference schemes aimed at providing the *viscosity* solution (Lions, 1982), which corresponds to the first-arrival branch of the traveltime field. The remarkable stability of the method results from a specifically chosen order of finite-difference evaluation. The order selection scheme resembles the *expanding wavefronts* method of Qin et al. (1992). The

[1]**email:** sergey@sep.stanford.edu

fast speed of the method is provided by the heap sorting algorithm, commonly used in Dijkstra's shortest path computation methods (Cormen et al., 1990). A similar idea has been used previously in a slightly different context, in the *wavefront tracking* algorithm of Cao and Greenhalgh (1994).

In this paper, I address the question of evaluating the fast marching method's applicability to more general situations. I describe a simple interpretation of the algorithm in terms of variational principles (namely, Fermat's principle in the case of eikonal solvers.) Such an interpretation immediately yields a useful extension of the method for unstructured grids: triangulations in two dimensions and tetrahedron tesselations in three dimensions. It also provides a constructive way of applying similar algorithms to solving other eikonal-like equations: anisotropic eikonal (Dellinger, 1991), "focusing" eikonal (Biondi et al., 1997), kinematic offset continuation (Fomel, 1995), and kinematic velocity continuation (Fomel, 1996). Additionally, the variational formulation can give us hints about higher-order enhancements to the original first-order scheme.

A BRIEF DESCRIPTION OF THE FAST MARCHING METHOD

For a detailed description of level set methods, the reader is referred to Sethian's recently published book (1996b). More details on the fast marching method appear in articles by Sethian (1996a) and Sethian and Popovici (1997). This section serves as a brief introduction to the main bulk of the algorithm.

The key feature of the algorithm is a carefully selected order of traveltime evaluation. At each step of the algorithm, every grid point is marked as either *Alive* (already computed), *NarrowBand* (at the wavefront, pending evaluation), or *FarAway* (not touched yet). Initially, the source points are marked as *Alive*, and the traveltime at these points is set to zero. A continuous band of points around the source are marked as *NarrowBand*, and their traveltime values are computed analytically. All other points in the grid are marked as *FarAway* and have an "infinitely large" traveltime value.

An elementary step of the algorithm consists of the following moves:

1. Among all the *NarrowBand* points, extract the point with the minimum traveltime.
2. Mark this point as *Alive*.
3. Check all the immediate neighbors of the minimum point and update them if necessary.
4. Repeat.

An update procedure is based on an upwind first-order approximation to the eikonal equation. In simple terms, the procedure starts with selecting one or more (up to three) neighboring points around the updated point. The traveltime values at the selected neighboring points need to be smaller than the current value. After the selection, one solves the quadratic equation

$$\sum_j \left(\frac{t_i - t_j}{\triangle x_{ij}}\right)^2 = s_i^2 \qquad (1)$$

for t_i. Here t_i is the updated value, t_j are traveltime values at the neighboring points, s_i is the slowness at the point i, and $\triangle x_{ij}$ is the grid size in the ij direction. As the result of the updating, either a *FarAway* point is marked as *NarrowBand* or a *NarrowBand* point gets assigned a new value.

Except for the updating scheme (1), the fast marching algorithm bears a very close resemblance to the famous shortest path algorithm of Dijkstra (1959). It is important to point out that unlike Moser's method, which uses Dijkstra's algorithm directly (Moser, 1991), the fast marching approach does not construct the ray paths from predefined pieces, but dynamically updates traveltimes according to the first-order difference operator (1). As a result, the computational error of this method goes to zero with the decrease in the grid size in a linear fashion. The proof of validity of the method (omitted here) is also analogous to that of Dijkstra's algorithm (Sethian, 1996a,b). As in most of the shortest-path implementations, the computational cost of extracting the minimum point at each step of the algorithm is greatly reduced [from $O(N)$ to $O(\log N)$ operations] by maintaining a priority-queue structure (heap) for the *NarrowBand* points (Cormen et al., 1990).

Figure 1 shows an example application of the fast marching eikonal solver on the three-dimensional SEG/EAGE salt model. The computation is stable despite the large velocity contrasts in the model. The current implementation takes about 10 seconds for computing a 100x100x100 grid on one node of SGI Origin 200. Alkhalifah and Fomel (1997) discuss the differences between Cartesian and polar coordinate implementations.

The difference equation (1) is a finite-difference approximation to the continuous eikonal equation

$$\left(\frac{\partial t}{\partial x}\right)^2 + \left(\frac{\partial t}{\partial y}\right)^2 + \left(\frac{\partial t}{\partial z}\right)^2 = s^2(x,y,z)\,, \qquad (2)$$

where x, y, and z represent the spatial Cartesian coordinates. In the next two sections, I show how the updating procedure can be derived without referring to the eikonal equation, but with the direct use of Fermat's principle.

THE THEORETIC GROUNDS OF VARIATIONAL PRINCIPLES

This section serves as a brief reminder of the well-known theoretical connection between Fermat's principle and the eikonal equation. The reader, familiar with this theory, can skip safely to the next section.

Both Fermat's principle and the eikonal equation can serve as the foundation of traveltime calculations. In fact, either one can be rigorously derived from the other. A simplified derivation of this fact is illustrated in Figure 2. Following the notation of this figure, let us consider a

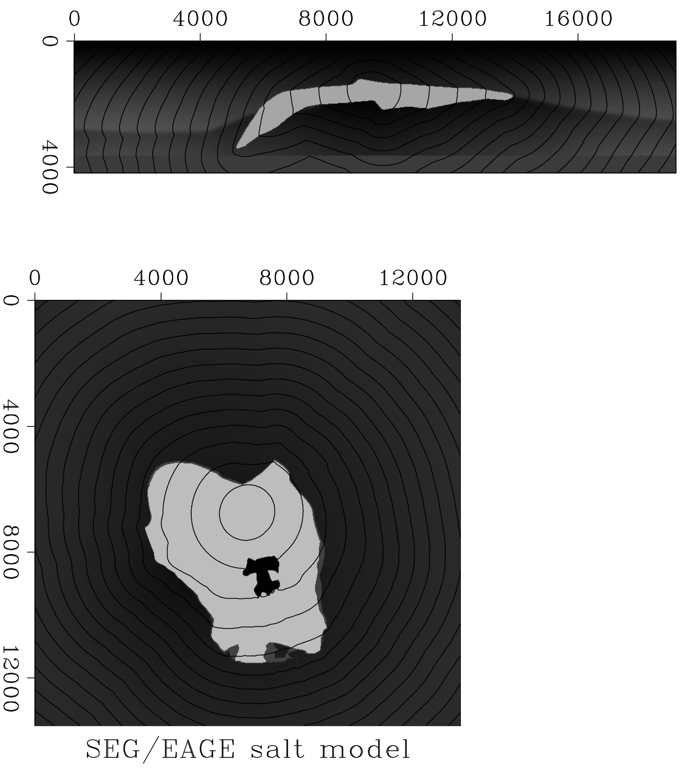

Figure 1: Constant-traveltime contours of the first-arrival traveltime, computed in the SEG/EAGE salt model. A point source is positioned inside the salt body. The top plot is a diagonal slice; the bottom plot, a depth slice. fmeiko-salt [CR]

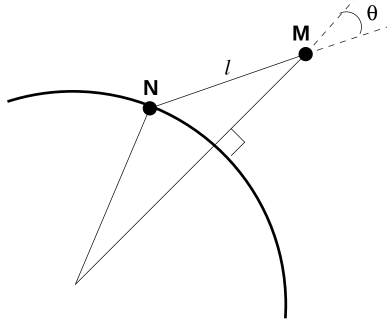

Figure 2: Illustration of the connection between Fermat's principle and the eikonal equation. The shortest distance between a wavefront and a neighboring point M is along the wavefront normal. fmeiko-fermat [NR]

point M in the immediate neighborhood of a wavefront $t(N) = t_N$. Assuming that the source is on the other side of the wavefront, we can express the traveltime at the point M as the sum

$$t_M = t_N + l(N,M) s_M \ , \tag{3}$$

where N is a point on the front, $l(N,M)$ is the length of the ray segment between N and M, and s_M is the local slowness. As follows directly from equation (3),

$$|\nabla t| \cos\theta = \frac{\partial t}{\partial l} = \lim_{M \to N} \frac{t_M - t_N}{l(N,M)} = s_N \ . \tag{4}$$

Here θ denotes the angle between the traveltime gradient (normal to the wavefront surface) and the line from N to M, and $\frac{\partial t}{\partial l}$ is the directional traveltime derivative along that line.

If we accept the local Fermat's principle, which says that the ray from the source to M corresponds to the minimum-arrival time, then, as we can see geometrically from Figure 2, the angle θ in formula (4) should be set to zero to achieve the minimum. This conclusion leads directly to the eikonal equation (2). On the other hand, if we start from the eikonal equation, then it also follows that $\theta = 0$, which corresponds to the minimum traveltime and constitutes the local Fermat's principle. The idea of that simplified proof is taken from Lanczos (1966), though it has obviously appeared in many other publications. The situations in which the wavefront surface has a discontinuous normal (given raise to multiple-arrival traveltimes) require a more elaborate argument, but the above proof does work for first-arrival traveltimes and the corresponding viscosity solutions of the eikonal equation (Lions, 1982).

The connection between variational principles and first-order partial-differential equations has a very general meaning, explained by the classic Hamilton-Jacobi theory. One generalization of the eikonal equation is

$$\sum_{i,j} a_{ij}(\mathbf{x}) \frac{\partial \tau}{\partial x_i} \frac{\partial \tau}{\partial x_j} = 1 \ , \tag{5}$$

where $\mathbf{x} = \{x_1, x_2, \dots\}$ represents the vector of space coordinates, and the coefficients a_{ij} form a positive-definite matrix A. Equation (5) defines the characteristic surfaces $t = \tau(\mathbf{x})$ for a linear hyperbolic second-order differential equation of the form

$$\sum_{i,j} a_{ij}(\mathbf{x}) \frac{\partial^2 u}{\partial x_i \partial x_j} + F(\mathbf{x}, u, \frac{\partial u}{\partial x_i}) = \frac{\partial^2 u}{\partial t^2} \ , \tag{6}$$

where F is an arbitrary function.

A known theorem (Smirnov, 1964) states that the propagation rays [characteristics of equation (5) and, correspondingly, bi-characteristics of equation (6)] are geodesic (extreme-length) curves in the Riemannian metric

$$d\tau = \sqrt{\sum_{i,j} b_{ij}(\mathbf{x})\, dx_i\, dx_j} \ , \tag{7}$$

where b_{ij} are the components of the matrix $B = A^{-1}$. This means that a ray path between two points $\mathbf{x}_1$ and $\mathbf{x}_2$ has to correspond to the extreme value of the curvilinear integral

$$\int_{\mathbf{x}_1}^{\mathbf{x}_2} \sqrt{\sum_{i,j} b_{ij}(\mathbf{x})\, dx_i\, dx_j}\ .$$

For the isotropic eikonal equation (2), $a_{ij} = \delta_{ij}/s^2(\mathbf{x})$, and metric (7) reduces to the familiar traveltime measure

$$d\tau = s(\mathbf{x})\, d\sigma\ , \tag{8}$$

where $d\sigma = \sqrt{\sum_i dx_i^2}$ is the usual Euclidean distance metric. In this case, the geodesic curves are exactly Fermat's extreme-time rays.

From equation (7), we see that Fermat's principle in the general variational formulation applies to a much wider class of situations if we interpret it with the help of non-Euclidean geometries.

VARIATIONAL PRINCIPLES ON A GRID

In this section, I derive a discrete traveltime computation procedure, based solely on Fermat's principle, and show that on a Cartesian rectangular grid it is precisely equivalent to the update formula (1) of the first-order eikonal solver.

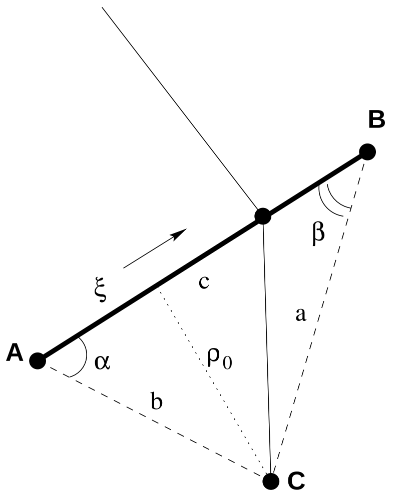

Figure 3: A geometrical scheme for the traveltime updating procedure in two dimensions. fmeiko-triangle [NR]

For simplicity, let us focus on the two-dimensional case[2]. Consider a line segment with the end points A and B, as shown in Figure 3. Let t_A and t_B denote the traveltimes from a fixed distant source to points A and B, respectively. Define a parameter ξ such that $\xi = 0$ at A,

[2]A very similar analysis applies in three dimensions, but requires a slightly more tedious algebra. It is left as an exercise for the reader.

$\xi = 1$ at B, and ξ changes continuously on the line segment between A and B. Then for each point of the segment, we can approximate the traveltime by the linear interpolation formula

$$t(\xi) = (1-\xi)t_A + \xi t_B \; . \tag{9}$$

Now let us consider an arbitrary point C in the vicinity of AB. If we know that the ray from the source to C passes through some point ξ of the segment AB, then the total traveltime at C is approximately

$$t_C = t(\xi) + s_C \sqrt{|AB|^2(\xi-\xi_0)^2 + \rho_0^2} \; , \tag{10}$$

where s_C is the local slowness, ξ_0 corresponds to the projection of C to the line AB (normalized by the length $|AB|$), and ρ_0 is the length of the normal from C to ξ_0.

Fermat's principle states that the actual ray to C corresponds to a local minimum of the traveltime with respect to raypath perturbations. According to our parameterization, it is sufficient to find a local extreme of t_C with respect to the parameter ξ. Equating the ξ derivative to zero, we arrive at the equation

$$t_B - t_A + \frac{s_C\,|AB|^2\,(\xi-\xi_0)}{\sqrt{|AB|^2(\xi-\xi_0)^2 + \rho_0^2}} = 0 \; , \tag{11}$$

which has (as a quadratic equation) the explicit solution for ξ:

$$\xi = \xi_0 \pm \frac{\rho_0\,(t_A - t_B)}{|AB|\sqrt{s_C^2\,|AB|^2 - (t_A-t_B)^2}} \; . \tag{12}$$

Finally, substituting the value of ξ from (12) into equation (10) and selecting the appropriate branch of the square root, we obtain the formula

$$\begin{aligned} c\,t_C &= \rho_0\sqrt{s_C^2c^2 - (t_A-t_B)^2} + c\,t_A\,(1-\xi_0) + c\,t_B\,\xi_0 = \\ & \rho_0\sqrt{s_C^2c^2 - (t_A-t_B)^2} + a\,t_A\,\cos\beta + b\,t_B\,\cos\alpha \; , \end{aligned} \tag{13}$$

where $c = |AB|$, $a = |BC|$, $b = |AC|$, angle α corresponds to $\widehat{BAC}$, and angle β corresponds to $\widehat{ABC}$ in the triangle ABC (Figure 3).

To see the connection of formula (13) with the eikonal difference equation (1), we need to consider the case of a rectangular computation cell with the edge AB being a diagonal segment, as illustrated in Figure 4. In this case, $\cos\alpha = \frac{a}{c}$, $\cos\beta = \frac{b}{c}$, $\rho_0 = \frac{ab}{c}$, and formula (13) reduces to

$$t_C = \frac{ab\sqrt{s_C^2(a^2+b^2) - (t_A-t_B)^2} + a^2\,t_A + b^2\,t_B}{a^2+b^2} \; . \tag{14}$$

We can notice that (14) is precisely equivalent to the solution of the quadratic equation (13), which in our new notation takes the form

$$\left(\frac{t_C - t_A}{b}\right)^2 + \left(\frac{t_C - t_B}{a}\right)^2 = s_C^2 \; . \tag{15}$$

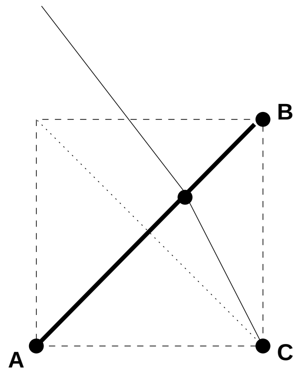

Figure 4: A geometrical scheme for traveltime updating on a rectangular grid. fmeiko-square [NR]

What have we accomplished by this analysis? First, we have derived a local traveltime computation formula for an arbitrary grid. The derivation is based solely on Fermat's principle and a local linear interpolation, which provides the first-order accuracy. Combined with the fast marching evaluation order, which is also based on Fermat's principle, this procedure defines a complete algorithm of first-arrival traveltime calculation. On a rectangular grid, this algorithm is exactly equivalent to the fast marching method of Sethian (1996a) and Sethian and Popovici (1997). Second, the derivation provides a general principle, which can be applied to derive analogous algorithms for other eikonal-type (Hamilton-Jacobi) equations and their corresponding variational principles.

SOLVING THE EIKONAL EQUATION ON A TRIANGULATED GRID

Unstructured (triangulated) grids have computational advantages over rectangular ones in three common situations:

- When the number of grid points can be substantially reduced by putting them on an irregular grid. This situation corresponds to irregular distribution of details in the propagation medium.
- When the computational domain has irregular boundaries. One possible kind of boundary corresponds to geological interfaces and seismic reflector surfaces (Wiggins et al., 1993). Another type of irregular boundary, in application to traveltime computations, is that of seismic rays. The method of bounding the numerical eikonal solution by ray envelopes has been introduced recently by Abgrall and Benamou (1996).
- When the grid itself needs to be dynamically updated to maintain a certain level of accuracy in the computation.

With its computational speed and unconditional stability, the fast marching method provides considerable savings in comparison with alternative, more accurate methods, such as semi-analytical ray tracing (Guiziou et al., 1991; Stankovic and Albertin, 1995) or the general Hamilton-Jacobi solver of Abgrall (1996).

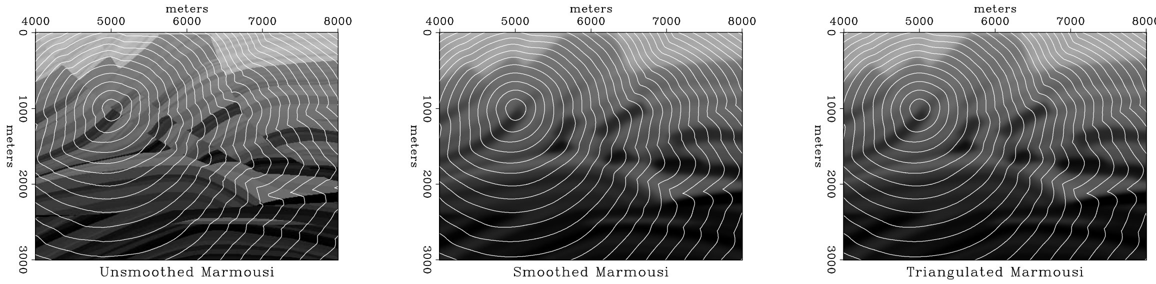

Figure 5: Traveltime contours, computed in the rough Marmousi model (left), the smoothed Marmousi (middle), and the smoothed triangulated Marmousi (right). fmeiko-test [NR]

Figure 5 shows a comparison between first-arrival traveltime computations in regularly gridded and triangulated Marmousi models. The two results match each other within the first-order accuracy of the fast marching method. However, the cost of the triangulated computation has been greatly reduced by constraining the number of nodes.

Computational aspects of triangular grid generation are outlined in Appendix A. A three-dimensional application would follow the same algorithmic patterns.

CONCLUSIONS

Variational principles have played an exceptionally important role in the foundations of mathematical physics. Their potential in numerical algorithms should not be underestimated.

In this paper, I interpret the fast marching eikonal solver with the help of Fermat's principle. Two important generalizations follow immediately from that interpretation. First, it allows us to obtain a fast method of first-arrival traveltime computation on triangulated grids. Furthermore, we can obtain a general principle, which extends the fast marching algorithm to other Hamilton-type equations and their variational principles. More research is required to confirm these promises.

In addition, future research should focus on 3-D implementations and on increasing the approximation order of the method.

ACKNOWLEDGMENTS

I thank Mihai Popovici and Biondo Biondi for drawing my attention to the fast marching level set method. Discussions with Bill Symes, Dan Kosloff, and Tariq Alkhalifah were crucial for developing a general understanding of the method. Jamie Sethian kindly responded to more specific questions. The conforming triangulation program was developed at the suggestion of Leonidas Guibas as a research project for his Geometrical Algorithms class at Stanford.

REFERENCES

Abgrall, R., and Benamou, J.-D., 1996, Multivalued traveltime fields, ray tracing and ekonal solver on unstructured grids: 66th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 1208–1211.

Abgrall, R., 1996, Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes: Comm. on Pure and Applied Math., **XLIX**, 1339–1373.

Albertin, U. K., and Wiggins, W., 1994, Embedding geologic horizon surfaces in tetrahedral meshes for geologic modeling: 64rd Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 502–505.

Alkhalifah, T., and Fomel, S., 1997, Implementing the fast marching eikonal solver: Spherical versus cartesian coordinates: SEP–**95**, 149–171.

Biondi, B., Fomel, S., and Alkhalifah, T., 1997, "Focusing" eikonal equation and global tomography: SEP–**95**, 61–76.

Cao, S., and Greenhalgh, S. A., 1994, Finite-difference solution of the eikonal equation using an efficient, first-arrival wavefront tracking scheme: Geophysics, **59**, no. 4, 632–643.

Chew, L. P., 1989, Constrained Delaunay triangulations: Algorithmica, **4**, 97–108.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L., 1990, Introduction to algorithms: McGraw-Hill.

Delaunay, B. N., 1934, Sur la sphère vide: Izv. Akad. Nauk SSSR, Otdel. Mat. Est. Nauk, **7**, 793–800.

Dellinger, J., 1991, Anisotropic finite-difference traveltimes: 61st Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 1530–1533.

Dijkstra, E. W., 1959, A note on two problems in connection with graphs: Numer. Math., **1**, 269–271.

Edelsbrunner, H., and Tan, T. S., 1993, An upper bound for conforming Delaunay triangulation: Discrete Comput. Geom., **10**, 197–213.

Fomel, S., 1995, Amplitude preserving offset continuation in theory Part 1: The offset continuation equation: SEP–**84**, 179–198.

Fomel, S., 1996, Migration and velocity analysis by velocity continuation: SEP–**92**, 159–188.

Fortune, S., 1987, A sweepline algorithm for Voronoi diagram: Algorithmica, **2**, 153–174.

Garland, M., and Heckbert, P. S., 1996, Fast and flexible polygonization of height fields: SIGGRAPH 96, Visual Proceedings, 143.

Guibas, L., and Stolfi, J., 1985, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams: ACM Trans. Graphics, **4**, 74–123.

Guibas, L. J., Knuth, D., and Sharir, M., 1992, Randomized incremental construction of Delaunay and Voronoi diagrams: Algorithmica, **7**, 381–413.

Guiziou, J. L., Mallet, J. L., Nobili, P., Anandappane, R., and Thisse, P., 1991, 3-D ray-tracing through complex triangulated surfaces: 61st Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 1497–1500.

Hale, D., and Cohen, J. K., 1991, Triangulated models of the Earth's subsurface: Center for Wave Phenomenon Report, **CWP-107**.

Hansen, A. J., and Levin, P. L., 1992, On conforming Delaunay mesh generation: Adv. in Eng. Soft., **14**, 129–135.

Lanczos, C., 1966, The variational principles of mechanics: University of Toronto Press, Toronto.

Lee, D. T., and Lin, A. K., 1986, Generalized Delaunay triangulation for planar graphs: Discrete Comput. Geom., **1**, 201–217.

Lions, P. L., 1982, Generalized solutions of Hamilton-Jacobi equations: Pitman.

Moser, T. J., 1991, Shortest path calculation of seismic rays: Geophysics, **56**, no. 1, 59–67.

Osher, S., and Sethian, J. A., 1988, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulation: Jour. of Comp. Phys., **79**, 12–49.

Podvin, P., and Lecomte, I., 1991, Finite difference computation of traveltimes in very contrasted velocity models: A massively parallel approach and its associated tools: Geophysical Journal International, **105**, 271–284.

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**, no. 3, 478–487.

Rivara, M.-C., 1996, New mathematical tools and techniques for the refinement and/or improvement of unstructured triangulations: 5th International Meshing Roundtable, Proceedings, 77–86.

Ruppert, J., 1995, A Delaunay refinement algorithm for quality two-dimensional mesh generation: Journal of Algorithms, **18**, 548–585.

Sethian, J. A., and Popovici, A. M., 1997, Three-dimensional traveltime computation using the fast marching method: submitted to Geophysics.

Sethian, J. A., 1996a, A fast marching level set method for monotonically advancing fronts: Proc. Nat. Acad. Sci., **93**, no. 4, 1591–1595.

Sethian, J. A., 1996b, Level set methods: Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science: Cambridge University Press.

Shamos, M., and Hoey, D., 1975, Closest point problems: 16th Annual IEEE Sympos. Found. Comput. Sci., Proceedings, 151–162.

Shewchuk, J. R., 1996, Robust adaptive floating-point geometric predicates: 12th Annual Symposium on Computational Geometry, 141–150.

Sibson, R., 1978, Locally equiangular triangulations: Comput. J., **21**, 243–245.

Smirnov, V. I., 1964, A course on higher mathematics: Pergamon Press.

Stankovic, G. M., and Albertin, U. K., 1995, Raytracing in topological tetrahedral models: 65th Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 1247–1250.

van Trier, J., and Symes, W. W., 1991, Upwind finite-difference calculation of traveltimes: Geophysics, **56**, no. 6, 812–821.

Vidale, J. E., 1990, Finite-difference calculation of traveltimes in three dimensions: Geophysics, **55**, no. 5, 521–526.

Wiggins, W., Albertin, U. K., and Stankovic, C., 1993, Building 3-D depth migration velocity models with topological objects: 63rd Annual Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 170–173.

Woo, M., Neider, J., and Davis, T., 1997, OpenGL programming guide: Addison-Wesley.

APPENDIX A

INCREMENTAL DELAUNAY TRIANGULATION AND RELATED PROBLEMS

Delaunay triangulation (Delaunay, 1934; Sibson, 1978; Guibas and Stolfi, 1985) is a fundamental geometric construction, which has numerous applications in different computational problems. For a given set of nodes (points on the plane), Delaunay triangulation constructs a triangle tessellation of the plane with the initial nodes as vertices. Among all possible triangulations, the Delaunay triangulation possesses optimal properties, which make it very attractive for practical applications, such as computational mesh generation. One of the most well-known properties is maximizing the minimum triangulation angle. In three dimensions, Delaunay triangulation generalizes naturally to a tetrahedron tessellation.

Several optimal-time algorithms of Delaunay triangulation (and its counterpart–Voronoi diagram) have been proposed in the literature. The divide-and-conquer algorithm (Shamos and Hoey, 1975; Guibas and Stolfi, 1985) and the sweep-line algorithm (Fortune, 1987) both achieve the optimal $O(N \log N)$ worst-case time complexity. Alternatively, a family of incremental algorithms has been used in practice because of their simplicity and robustness.

Though the incremental algorithm can take $O(N^2)$ time in the worst case, the expectation time can still be $O(N \log N)$, provided that the nodes are inserted in a random order (Guibas et al., 1992).

The incremental algorithm consists of two main parts:

1. Locate a triangle (or an edge), containing the inserted point.

2. Insert the point into the current triangulation, making the necessary adjustments.

The Delaunay criterion can be reduced in the second step to a simple *InCircle* test (Guibas and Stolfi, 1985): if a circumcircle of a triangle contains another triangulation vertex in its circumcenter, then the edge between those two triangles should be "flipped" so that two new triangles are produced. The testing is done in a recursive fashion consistent with the incremental nature of the algorithm. When a new node is inserted inside a triangle, three new triangles are created, and three edges need to be tested. When the node falls on an edge, four triangles are created, and four edges are tested. In the case of test failure, a pair of triangles is replaced by the flip operation with another pair, producing two more edges to test. Under the randomization assumption, the expected total time of point insertion is $O(N)$. Randomization can be considered as an external part of the algorithm, provided by preprocessing.

Guibas et al. (1992) reduce the point location step to an efficient $O(N \log N)$ procedure by maintaining a hierarchical tree structure: all triangles, occurring in the incremental triangulation process, are kept in memory, associated with their "parents." One or two point location tests (*CCW* tests) are sufficient to move to a lower level of the tree. The search terminates with a current Delaunay triangle.

To test the algorithmic performance of the incremental construction, I have profiled the execution time of my incremental triangulation program with the Unix `pixie` utility. The profiling result, shown in Figures A-1 and A-2, complies remarkably with the theory: $O(N \log N)$ operations for the point location step, and $O(N)$ operations for the point insertion step. The experimental constant for the insertion step time is about 8.6. The experimental constant for the point location step is 4. The CPU time, depicted in Figure A-3, also shows the expected $O(N \log N)$ behavior.

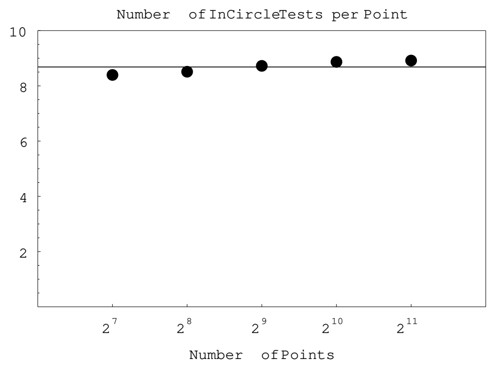

Figure A-1: The number of point insertion operations (*InCircle* test) plotted against the number of points. fmeiko-itime [NR]

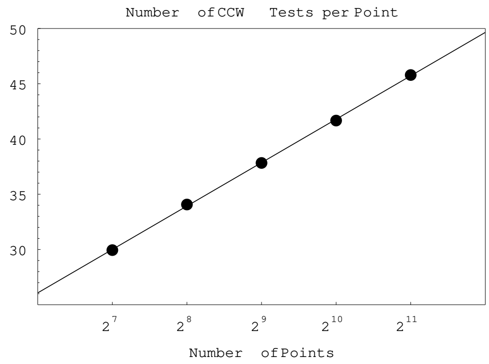

Figure A-2: Number of point location operations (*CCW* test) plotted against the number of points. fmeiko-ctime [NR]

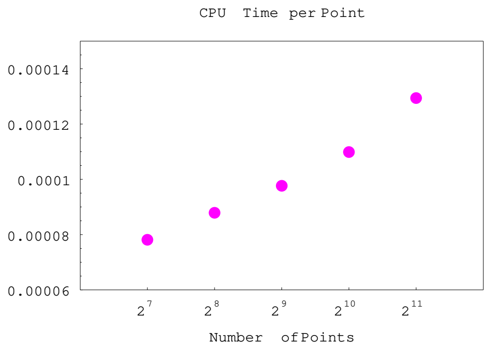

Figure A-3: CPU time (in seconds per point) plotted against the number of points. fmeiko-time [NR]

A straightforward implementation of Delaunay triangulation would provide an optimal triangulation for any given set of nodes. However, the quality of the result for unfortunate geometrical distributions of the nodes can be unsatisfactory. In the rest of this appendix, I describe three problems, aimed at improving the triangulation quality: conforming triangulation, triangulation of height fields, and mesh refinement. Each of these problems can be solved with a variation of the incremental algorithm.

Conforming Triangulation

In the practice of mesh generation, the input nodes are often supplemented by boundary edges: geologic interfaces, seismic rays, and so on. It is often desirable to preserve the edges so that they appear as edges of the triangulation (Albertin and Wiggins, 1994). One possible approach is *constrained* triangulation, which preserves the edges, but only approximately satisfies the Delaunay criterion (Lee and Lin, 1986; Chew, 1989). An alternative, less investigated, approach is *conforming* triangulation, which preserves the "Delaunayhood" of the triangulation by adding additional nodes (Hansen and Levin, 1992) (Figure A-4). Conforming Delaunay triangulations are difficult to analyze because of the variable number of additional nodes. This problem was attacked by Edelsbrunner and Tan (1993), who suggested an algorithm with a defined upper bound on added points. Unfortunately, Edelsbrunner's algorithm is slow in practice because the number of added points is largely overestimated. I chose to implement a modification of the simple incremental algorithm of Hansen and Levin. Although Hansen's algorithm has only a heuristic justification and sets no upper bound on the number of inserted nodes, its simplicity is attractive for practical implementations, where it can be easily linked with the incremental algorithm of Delaunay triangulation.

The incremental solution to the problem of conforming triangulation can be described by the following scheme:

- First, the boundary nodes are triangulated.
- Boundary edges are inserted incrementally.
- If a boundary edge is not present in the triangulations, it is split in half, and the middle node is inserted into the triangulation. This operation is repeated for the two parts of the original boundary edge and continues recursively until all the edge parts conform.
- If at some point during the incremental process, a boundary edge violates the Delaunay criterion (the *InCircle* test), it is split to assure the conformity.

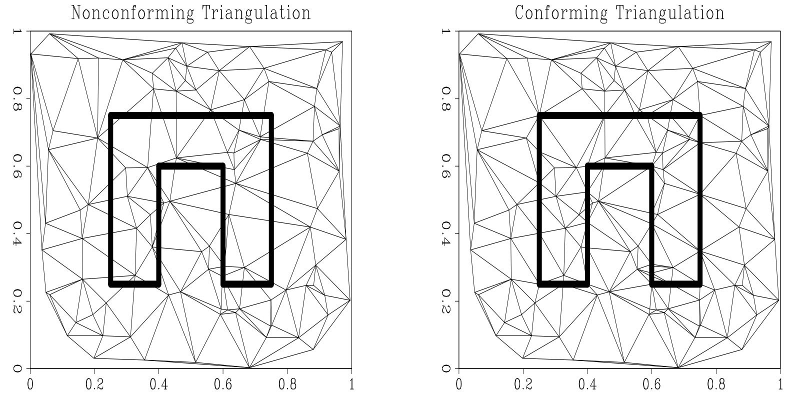

Figure A-4: An illustration of conforming triangulation. The left plot shows a triangulation of 500 random points; the triangulation in the right plot is conforming to the embedded boundary. Conforming triangulation is a genuine Delaunay triangulation, created by adding additional nodes to the original distribution. fmeiko-conform [ER]

To insert an edge AB into the current triangulation, I use the following recursive algorithm:

Function **InsertEdge** (AB)

1. Define C to be the midpoint of AB.
2. Using the triangle tree structure, locate triangle $\mathcal{T} = DEF$ that contains C in the current triangulation.
3. **If** AB is an edge of $\mathcal{T}$ **then return**.
4. **If** A (or B) is a vertex of $\mathcal{T}$ (for example, $A = D$) **then** define C as an intersection of AB and EF.
5. **Else** define C as an intersection of AB and an arbitrary edge of $\mathcal{T}$ (if such an intersection exists).

6. Insert C into the triangulation.
7. **InsertEdge** (CA).
8. **InsertEdge** (CB).

The intersection point of edges AB and EF is given by the formula

$$C = A + \lambda(B - A)\,, \tag{A-1}$$

where

$$\lambda = \frac{(F_y - E_y)(E_x - A_x) - (F_x - E_x)(E_y - A_y)}{\det \begin{vmatrix} B_x - A_x & B_y - A_y \\ F_x - E_x & F_y - E_y \end{vmatrix}}\,. \tag{A-2}$$

The value of λ should range between 0 and 1.

If, at some stage of the incremental construction, a boundary edge AB fails the Delaunay *InCircle* test for the circle $CABD$, then I simply split it into two edges by adding the point of intersection into the triangulation. The rest of the process is very much like the process of edge validation in the original incremental algorithm.

Triangulation of Height Fields

Often, a velocity field (or other object that we want to triangulate) is defined on a regular Cartesian grid. One way to perform a triangulation in this case is to select a smaller subset of the initial grid points, using them as the input to a triangulation program. We need to select the points in a way that preserves the main features of the original image, while removing some unnecessary redundancy in the regular grid description.

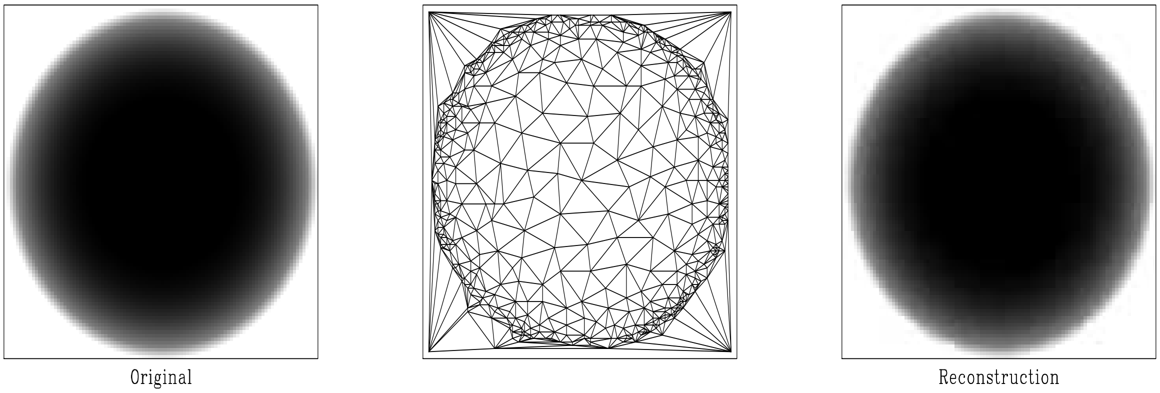

Figure A-5: Illustration of Garland's algorithm for triangulation of height fields. The left plot shows the input image of a sphere, containing 100 by 100 pixels. The middle plot shows 500 pixels, selected by the algorithm and triangulated. The right plot is the result of local plane interpolation of the triangulated surface. fmeiko-sphere [ER]

Garland and Heckbert (1996) surveyed different approaches to this problem and proposed a fast version of the incremental *greedy insertion* algorithm. Their algorithm adds points incrementally, selecting at each step the point with the maximum interpolation error with respect to the current triangulation. Though a straightforward implementation of this idea would lead to an unacceptably slow algorithm, Garland and Heckbert have discovered several sources for speeding it up. First, we can take advantage of the fact that only a small area of the current triangulation gets updated at each step. Therefore, it is sufficient to recompute the interpolation error only inside this area. Second, the maximum extraction procedure can be implemented very efficiently with a priority queue data structure.

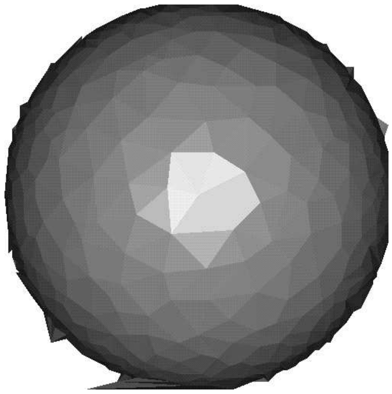

Figure A-6: An image from the previous example, as rendered by the OpenGL library. The shades on polygonal (triangulated) sides are exaggerated by a simulation of the direct light source. fmeiko-opengl [NR]

Figure A-5 illustrates this algorithm with a simple example. The original image (the left plot) contained 10,000 points, laid out on a regular rectangular grid. The algorithm selects a smaller number of points and immediately triangulates them (the middle plot). The image can be reconstructed by local plane interpolation (the right plot.) The reconstruction quality can be further improved by increasing the number of triangles. Figure A-6 shows the same image as rendered by the OpenGL graphics library (Woo et al., 1997).

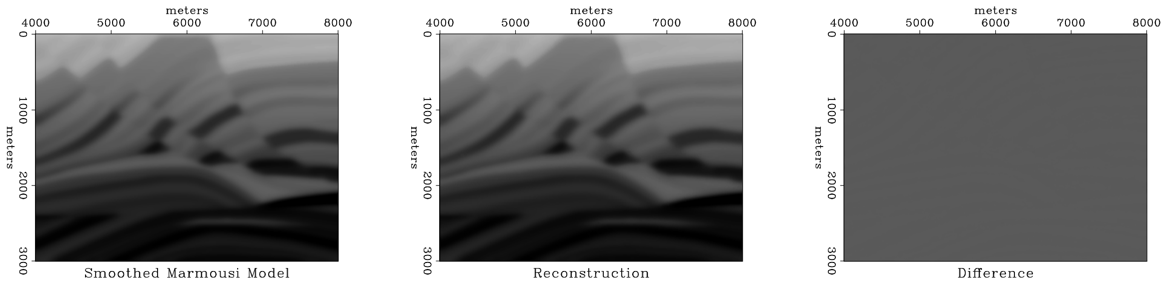

Figure A-7: Applying the height triangulation algorithm to the Marmousi model. The left plot shows a smoothed and windowed version of the Marmousi model. The middle plot is a result of 10,000-point triangulation, followed by linear interpolation. The right plot shows the difference between the two images. fmeiko-marmousi [ER]

Figure A-7 shows an application of the height triangulation algorithm to the famous Marmousi model. The left plot shows a smoothed and windowed version of the Marmousi, plotted on a 501 by 376 computational grid. In the middle plot, 10,000 points from the original

188,376 were selected for triangulation and interpolated back to the rectangular grid. The right plot demonstates the small difference between the two images.

Mesh Refinement

One the main properties of Delaunay triangulation is that, for a given set of nodes, it provides the maximum smallest angle among all possible triangulations. It is this property that supports the wide usage of Delaunay algorithms in the mesh generation problems. However, it doesn't guarantee that the smallest angle will always be small. In fact, for some point distributions, it is impossible to avoid skinny small-angle triangles. The remedy is to add additional nodes to the triangulation so that the quality of the triangles is globally improved. This problem has become known as *mesh refinement* (Ruppert, 1995).

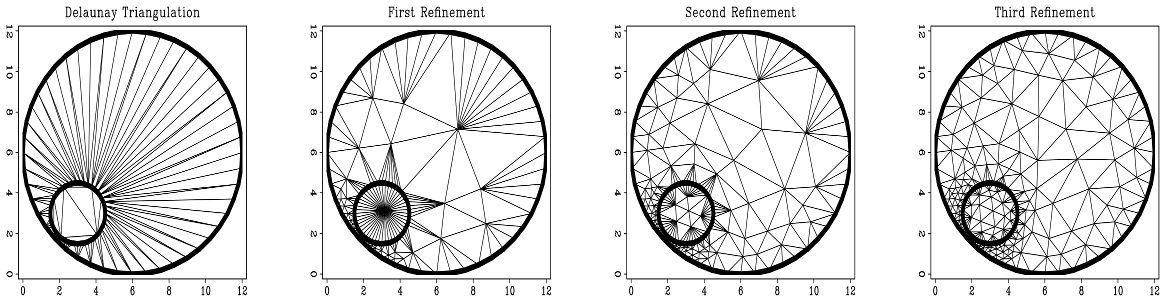

Figure A-8: An illustration of Rivara's refinement algorithm. The left plot shows an input to the algorithm: a valid Delaunay triangulation with "skinny" triangles. Three other plots show successive applications of the refinement algorithm. fmeiko-hole [ER]

One of the recently proposed mesh refinement algorithms is Rivara's *backward longest-side refinement* technique (Rivara, 1996). The main idea of the algorithm is to trace the LSPP (longest-side propagation path) for each refined triangle. The LSPP is an ordered list of triangles, connected by a common edge, such that the longest triangle edge is strictly increasing. After tracing the LSPP, we bisect the longest edge and insert its midpoint into the triangulation. Rivara's algorithm is remarkably efficient and easy to implement. In comparison with alternative methods, it has the additional advantage of being applicable in three dimensions.

Figure A-9 demonstrates an application of different triangulation techniques to a simple layered model, borrowed from the Seismic Unix demos (where it is attributed it to V.Červený.) Another model from Hale and Cohen (1991) is used in Figure A-10.

Implementation Details

Edge operations form the basis of the incremental algorithm. Therefore, it is convenient to describe triangulation with edge-oriented data structures (Guibas and Stolfi, 1985). I have followed the idea of Hansen and Levin (1992) of associating with each edge two pointers to the end points and two pointers to the adjacent triangles. The triangle structure is defined by three

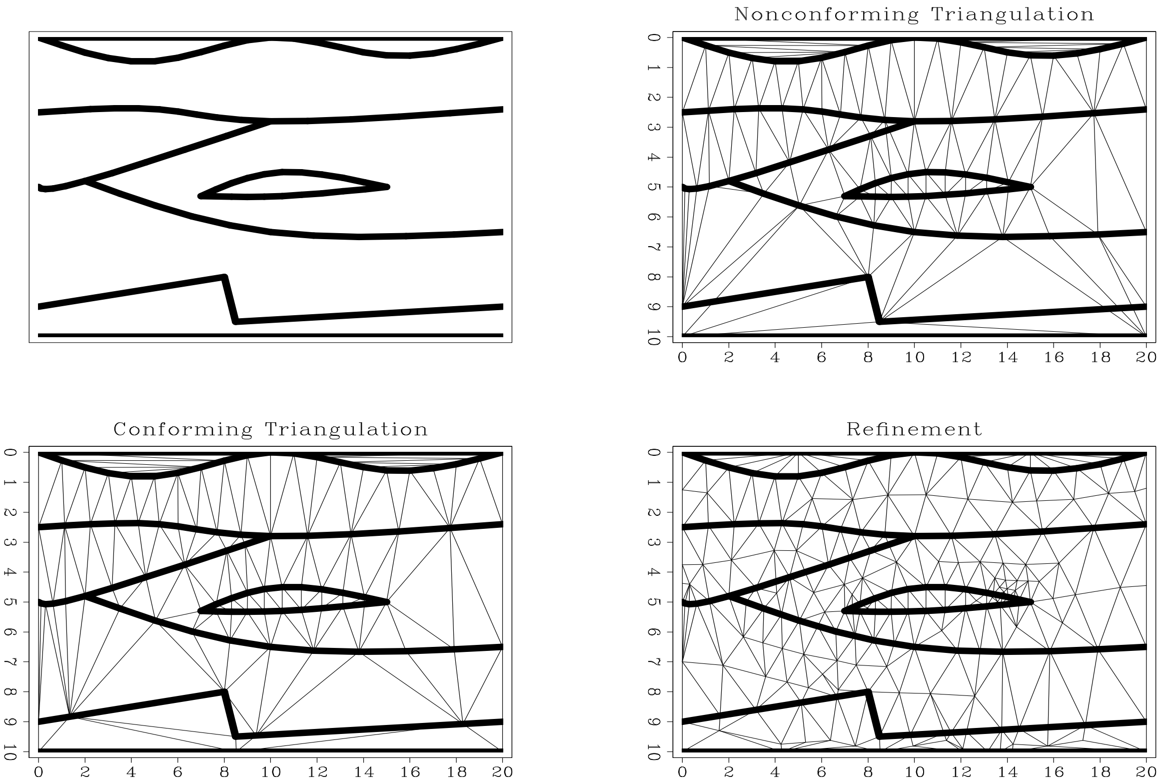

Figure A-9: A comparison of different triangulation techniques on a simple layered model. The top left plot shows the original model; the top right plot, the result of noncomforming triangulation; the two bottom plots, conforming triangulation and an additional mesh refinement. fmeiko-cerveny [ER]

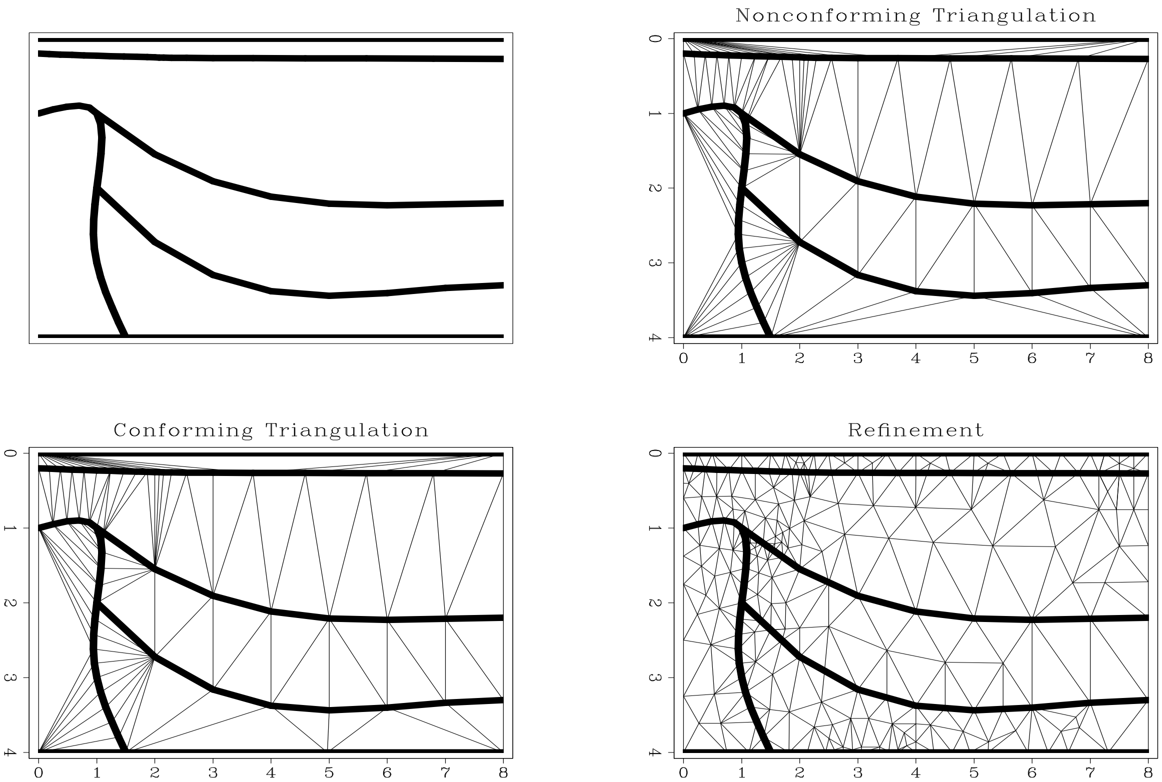

Figure A-10: A comparison of different triangulation techniques on a simple salt-type model. The top left plot shows the original model; the top right plot, the result of noncomforming triangulation; the two bottom plots, conforming triangulation and an additional two-step mesh refinement. fmeiko-susalt [ER] pointers to the edges of a triangle. Additionally, as required by the point location algorithm, each triangle has a pointer to its "children." This pointer is NULL when the triangle belongs to the current Delaunay triangulation.

Many researchers have pointed out that the geometric primitives used in triangulation must be robust with respect to round-off errors of finite-precision calculation. To assure the robustness of the code, I used the adaptive-precision predicates of Shewchuk (1996), available as a separate package from the `netlib` public-domain archive.