JOURNAL OF GEOPHYSICAL RESEARCH, VOL. 103, NO. B1, PAGES 773–781, JANUARY 10, 1998

Tomographic systems of equations with irregular cells

Malcolm Sambridge[1] and Ólafur Guđmundsson,

Research School of Earth Sciences, Institute of Advanced Studies, Australian National University
Canberra, A.C.T., Australia

Abstract. How to implement seismic or any other type of ray-based tomography in a new class of cellular parameterization built from any chosen distribution of nodes in two or three dimensions is shown. The complete flexibility of this novel approach allows one to introduce detail in a tomographic model only where desired, thereby reducing memory and computation time, or to impose complex a priori constraints on the inversion. Full details of powerful new algorithms are given to generate unequally sized tetrahedral or polyhedral cells and to calculate the necessary Frechet derivatives required in linearized tomography. These algorithms are efficient enough in three dimensions, to allow the parameterization to be refined during an inversion. The methods are illustrated with numerical examples. It is concluded that in linear or nonlinear inversion the computational cost of the new algorithms will not be significantly higher than that incurred by using a regular Cartesian grid.

1. Introduction

Many aspects of seismic tomography influence the quality and quantity of information that can be retrieved on Earth structure. One area that is receiving increasing attention is the nature of the parameterization used. Local parameterizations, such as cubic cells, are popular because they are simple to generate and only require trivial bookkeeping tasks to be solved, i.e., locating the cell containing any given point. However, in three dimensions, regular meshes often lead to a very large number of unknowns. For example, in mantle tomography, 10^5-10^6 cells are common [*Zhou and Clayton*, 1990; *Spakman*, 1991], a large proportion of which are often poorly constrained.

Recently, *Sambridge et al.* [1995] presented a new class of parameterization based on unevenly sized tetrahedra. The computational tools developed for that work allow "well shaped" tetrahedra to be automatically built around any set of nodes (points). The complete flexibility allowed in positioning the nodes has considerable potential for use in seismic tomography. For example, one might choose to concentrate nodes in areas of high ray density, thereby improving spatial resolution in well-constrained parts of the model or reducing it in ill-constrained regions. Other possibilities are parameterizations which vary during an inversion (i.e., nodes added or removed) or ones which impose a priori constraint on the inversion through a particular choice of nodal positions.

With an irregular parameterization the bookkeeping tasks are no longer trivial and require sophisticated algorithms for an efficient implementation. The purpose of this paper is to give full details of all algorithms required to make use of these parameterizations in tomographic problems. Furthermore, we extend the class of parameterization from the Delaunay tetrahedra (described previously) to include convex Voronoi polyhedra of arbitrary size or combinations of the two.

The irregular parameterization used in this paper is not built upon an underlying regular grid, as, for example, in the approach developed by *Abers and Roecker* [1991]. In their work the entire region under consideration is divided into a "fine scale" regular Cartesian grid. Variably sized "metablocks" are then constructed by defining a cross-reference table which associates each small block with its metablock. The metablocks can therefore be of any shape defined by grouping the small blocks together. This type of approach has the advantage that it is conceptually simpler than the one we present here, and in effect, the usual cubic block tomography can be used; however, it has two distinct disadvantages. The first is that in many cases it will only be possible to construct the cross-reference table by hand (for example, by visual inspection) rather than in any automatic fashion, which can, in general, be a very tedious and labor-intensive process. The second is that the underlying regular grid places an a priori lower bound on the scale length of allowed structures, which cannot be changed later without complete reconstruc-

[1]Also at Centre for Information Science Research, Australian National University, Canberra, A.C.T., Australia.

Paper number 97JB02602.
0148-0227/98/97JB-02602$09.00

tion of a new cross-reference table. Since Delaunay and Voronoi cells can be calculated about any set of points, there is no a priori limit to the range of cell sizes, and it is trivial to refine the parameterization at any stage by simply adding extra nodes locally. This might be advantageous, for example, during an inversion where the parameterization was updated or refined dynamically [e.g., *Michelini*, 1995]. With the metablock approach one must recalculate the cross-reference table or define a very large one in the first place. In general, however, one may well wish to combine these two ideas and build nonconvex metaelements by joining Delaunay or Voronoi cells together. This is the approach taken by *Guðmundsson and Sambridge* [1998] in constructing a set of three-dimensional models of subduction zones, which are also used in the numerical example in section 4. (In this case, however, it was possible to design a set of criteria to semiautomatically generate the cross-reference table.)

In this paper we only deal with part of the tomographic problem, i.e., the building of the linearized system of equations, the solution of the resulting system may be carried out with standard techniques. In particular, we show how to efficiently calculate the Frechet derivatives required in linearized tomography using these irregular cells. The algorithms are illustrated with numerical examples.

2. Building an Irregular Parameterization

The geometrical constructs known as Delaunay triangulations (in 2-D) and tetrahedralizations (in 3-D) have received much attention, mainly in the field of computational geometry [see *Aurenhammer*, 1991; *Okabe et al.*, 1992]. Recently, they have found a number of applications in geophysics [*Parker et al.*, 1987; *Sambridge et al.*, 1995; *Braun and Sambridge*, 1995, 1997]. Delaunay triangles, or tetrahedra, may be built around any desired set of nodes in two or three dimensions. These nodes define the vertices of the tetrahedra (see Figures 1a and 2a). Alternatively, one may define Voronoi polyhedra about each node as the region which is closest to that node (see Figures 1b and 2b). (From here on we use the generic term "cell" to refer to both types.) Because of the special properties of these cells [see *Aurenhammer*, 1991], their volume will reflect the local nodal density regardless of the distribution of nodes. In our experience the Delaunay cells are more convenient when one wishes to build parameterizations containing particular interfaces, for example, the surface of a subducting plate, because the nodes can be distributed on the surface. In contrast, Voronoi cells are more useful when areal or volumetric coverage is of concern, for example, representing tectonic regions on the surface of the Earth (see below).

A number of methods are available for calculating either the Delaunay or Voronoi cells in two and three dimensions (see *Sambridge et al.* [1995] for a discussion). The approach of *Barber et al.* [1993] is used throughout this paper. Public domain software is available for its implementation (B. Barber and H. Huhdanpaa, Qhull computer program, available via ftp from geom.umn.edu, 1994). The data structure used to represent either parameterization is a list of node indices at the vertices of each Delaunay triangle, or tetrahedron, i.e., $V_{t,j}(t = 1, \ldots, N_t; j = 1, \ldots, D+1)$, where D is the number of dimensions and N_t is the number of triangles, or tetrahedra (determined by the generation algorithm). This list is calculated from the set of nodal coordinates. In two dimensions, N_t is approximately twice the number of nodes, N_p (although it can not be greater than $2N_p$). In three dimensions N_t can be much higher.

3. Calculating Frechet Derivatives

The main computational task in linearized tomography is to calculate the coefficients of the linear system. A local linearization of the travel time equation leads to the well-known relation,

$$\delta t_i = \int_{L_i^o} \delta s(\mathbf{r})\, dl \tag{1}$$

where $\mathbf{r}$ is a position vector and δt_i is the travel time perturbation for the ith ray due to a perturbation in the slowness field δs. The integral is along the ith ray path L_i^o, where the suffix o indicates evaluation in the reference medium. If we expand the slowness perturbation in terms of orthogonal basis functions $\phi_k(\mathbf{r})$, we have

$$\delta s(\mathbf{r}) = \sum_{k=1}^{N} s_k \phi_k(\mathbf{r}) \tag{2}$$

where s_k are the coefficients to be determined. Combining (2) with (1), we get the linear system of equations

$$\delta t_i = \sum_k \mathbf{A}_{i,k} s_k \tag{3}$$

where the elements of the Frechet derivative matrix $\mathbf{A}$, are given by,

$$\mathbf{A}_{i,k} = \int_{L_i^o} \phi_k(\mathbf{r})\, dl \tag{4}$$

i.e., the integral of the kth basis function along the ith ray path. If, instead, we choose to expand the velocity perturbation in terms of orthogonal basis functions, i.e.,

$$\delta v(\mathbf{r}) = \sum_{k=1}^{N} v_k \phi_k(\mathbf{r}) \tag{5}$$

then s_k is replaced with the velocity coefficients v_k in (3), and we get

$$\mathbf{A}_{i,k} = -\int_{L_i^o} \frac{\phi_k(\mathbf{r})}{v_o^2(\mathbf{r})}\, dl \tag{6}$$

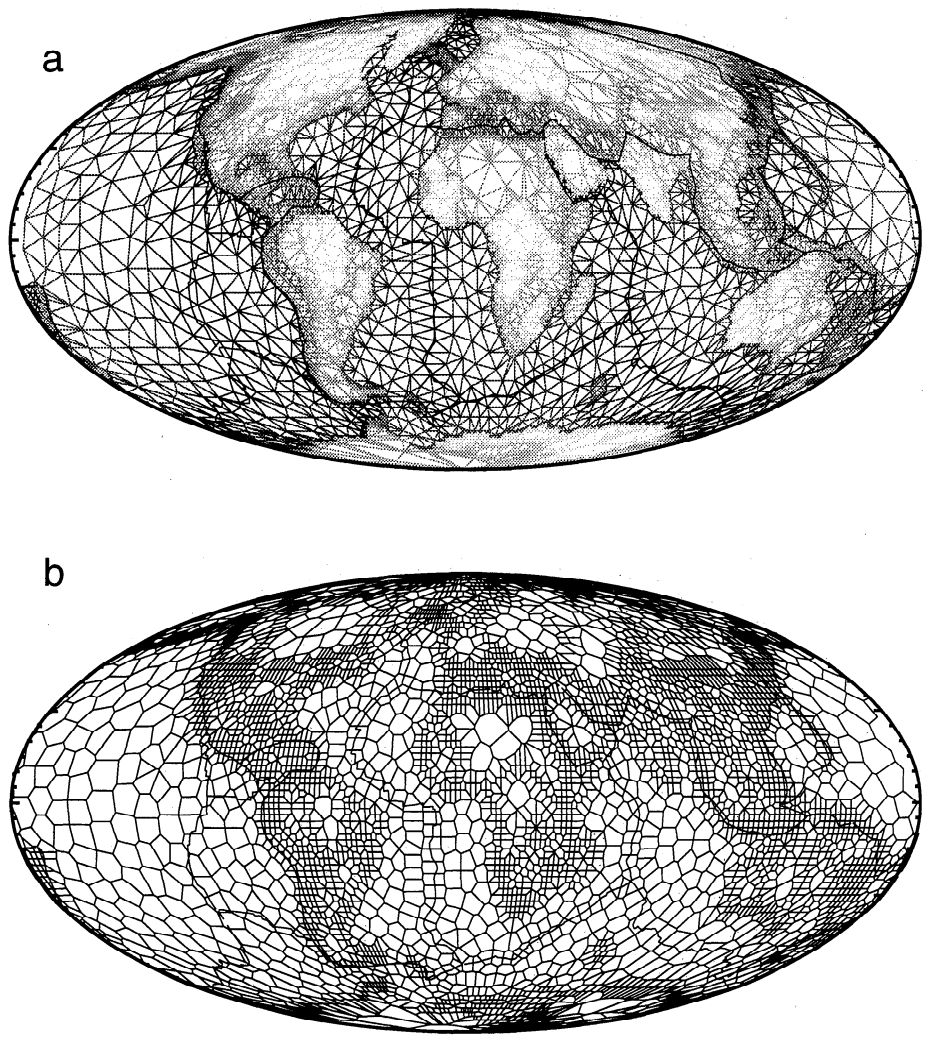

Figure 1. (a) The exterior faces of Delaunay tetrahedra formed around 4274 nodes on the surface of the Earth. The nodal positions were chosen to define boundaries of tectonic regions. (b) The exterior faces of Voronoi polyhedra built around the same set of nodes used in Figure 1a. Note that each node lies at the vertex of a Delaunay tetrahedron and inside a Voronoi polyhedron.

In either case, numerical integrations are required along each ray. The simplest way to proceed is to step down each ray and evaluate the integrands in (4) and (6). If we denote the position of the lth point along the ith ray as $\mathbf{r}_{i,l}$, then the numerical approximation of both integrals (4) and (6) can be written as

$$\mathbf{A}_{i,k} = \sum_l h_k(\mathbf{r}_{i,l})\ \Delta_{i,l} \tag{7}$$

where h_k represents either integrand in (4) or (6) and $\Delta_{i,l}$ is the length of the ray segment with midpoint $\mathbf{r}_{i,l}$. This is a general expression which covers a range of possible parameterizations, depending on the choice of the basis function. The two simplest cases are for the kth basis function to be constant inside the kth Delaunay or Voronoi cell. In this case we solve for a single coefficient representing the slowness perturbation inside each cell. Using (4), this leads to $h_k(\mathbf{r}_{i,l}) = 1$, and so $\mathbf{A}_{i,k}$ is just the length of the ith ray in the kth cell. This choice is adequate if smoothness is not required in the resulting slowness field. If ray tracing is to be performed in a heterogeneous model (e.g., in nonlinear tomography), then it is more convenient to choose a basis function with continuity in the slowness field. This can be achieved using Delaunay cells and specifying the slowness (or velocity) at the vertices of the tetrahedra (i.e., the nodes), rather than inside the tetrahedra. In this case we have one coefficient per node, and the basis functions depend linearly on position, that is we get

$$\phi_k(\mathbf{r}) = \frac{(\mathbf{r}-\mathbf{r}_1)\cdot\{(\mathbf{r}_2-\mathbf{r}_1)\times(\mathbf{r}_3-\mathbf{r}_1)\}}{(\mathbf{r}_k-\mathbf{r}_1)\cdot\{(\mathbf{r}_2-\mathbf{r}_1)\times(\mathbf{r}_3-\mathbf{r}_1)\}} \tag{8}$$

where $\mathbf{r}_k$ is the position vector of node k and $\mathbf{r}_1, \mathbf{r}_2$, and $\mathbf{r}_3$ are the position vectors of the other three nodes in the tetrahedron containing $\mathbf{r}$ and $\mathbf{r}_k$. In this case the slowness field in any tetrahedron only depends on the values of the slowness coefficients at its four vertices, and its gradients are discontinuous across the

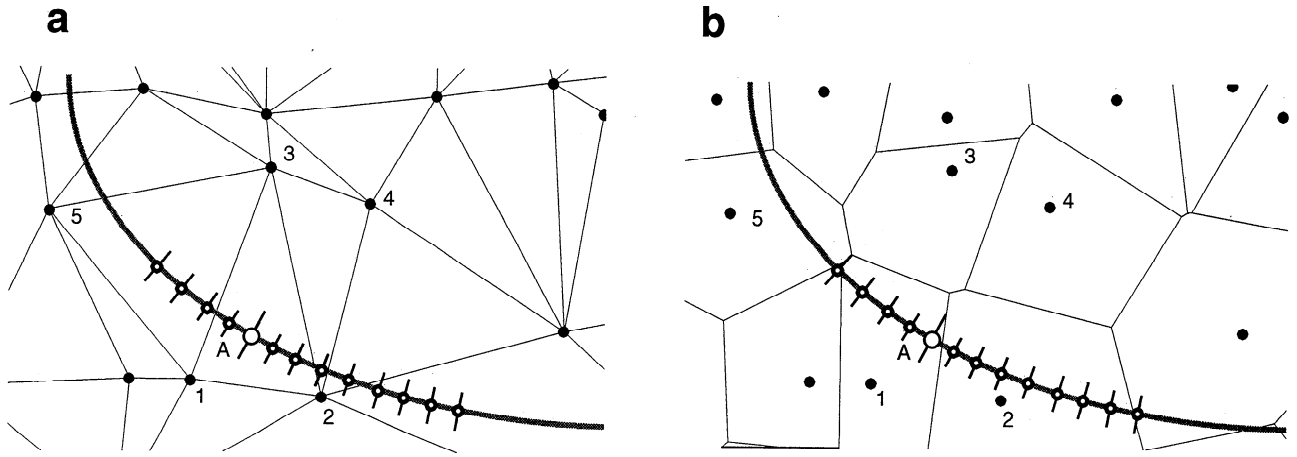

Figure 2. (a) Stepping along a ray path in a Delaunay triangulation and (b) same as Figure 2a but in the corresponding Voronoi cells. The ray path length in either type of cell can be determined by simple numerical integration. The point marked A will be influenced by different nodes, or cells, depending on the type of parameterization used.

faces. If higher-order smoothness is required, then natural neighbor basis functions can be used about each node [see *Sibson* 1980; *Watson*, 1992; *Sambridge et al.*, 1995].

Regardless of the choice of basis function, the same bookkeeping problem needs to be solved in order to calculate Frechet derivatives, i.e. to find the cell containing a given point $\mathbf{r}_{i,l}$ along the ray. The only difference is in the number of basis functions that are nonzero at any point and the value of the integrand in (7). Therefore it is only necessary to consider the two simplest cases of constant perturbations per Delaunay or Voronoi cell. The extension to more complex basis functions is straightforward.

3.1. Ray Lengths in Irregular Cells

The ray lengths in each Delaunay tetrahedron or Voronoi polyhedron can be found in the usual way by sampling along each ray at a predetermined step length and finding the cell containing the midpoint of the current ray segment. This is illustrated in Figure 2. Since mantle tomography can involve thousands of teleseismic rays and thousands of steps down each ray, the cell location will usually need to be found millions of times, and therefore an efficient location algorithm is essential in both types of mesh.

3.2. Delaunay Tetrahedra

An algorithm has been developed for point location in Delaunay tetrahedra which is a generalization of the 2-D method described by *Sloan* [1987] and also discussed by *Sambridge et al.* [1995]. The algorithm requires a preprocessing step to be carried out only once, regardless of the number of locations performed. The object of this stage is to calculate an adjacency matrix $\Omega_{t,j}(t = 1, \ldots, N_t; j = 1, \ldots, 4)$ containing the index of the tetrahedron which shares the jth face of tetrahedron t. The jth face of tetrahedron t is defined as the one (of the four) which does not contain the node $V_{t,j}$. The adjacency matrix indicates which four tetrahedra are neighbors to any given tetrahedron t, i.e., those which share a face with t. A zero entry in the adjacency matrix indicates that the corresponding face of the tetrahedron lies on the boundary of the model (the convex hull) and hence has no neighbor attached to that face.

A straightforward method of calculating the adjacency matrix is simply to use the vertices list to check all pairs of tetrahedra for possible neighbors. The computational cost of this approach scales with the square of the number of tetrahedra, and so it can be expensive as N_t becomes large. More efficient methods are available which scale linearly with N_t. For the 2-D case a method of this type was presented by *Sambridge et al.* [1995]. Also, many 2-D Delaunay algorithms calculate the adjacency matrix as a by-product [e.g., *Sloan*, 1987]. For the 3-D case we present a new algorithm in the appendix which also scales linearly with N_t.

Once the adjacency matrix has been calculated, the tetrahedron location algorithm is straightforward. To find the tetrahedron containing the point $\mathbf{r}$, we start with any initial guess tetrahedron t. For each face $j (j = 1, \ldots, 4)$ of tetrahedron t, we test if $\mathbf{r}$ is on the same side of the face as node $V_{t,j}$. This is true if

$$\frac{\mathbf{n}_j \cdot (\mathbf{r} - \mathbf{r}_f)}{\mathbf{n}_j \cdot (\mathbf{r}_j - \mathbf{r}_f)} \geq 0 \qquad (9)$$

where $\mathbf{n}_j$ is the normal to face j, $\mathbf{r}_j$ is the position vector for node j, and $\mathbf{r}_f$ is the position vector of any node in the face j. If this condition is true, then the next face of t is tested (i.e., j is set to $j + 1$). If (9) is not true, then t is set to the neighboring tetrahedron attached to this face (i.e., t is set equal to $\Omega_{t,j}$), and the four faces of the new tetrahedron are tested in the same way. If

$\mathbf{r}$ passes the test for all four faces, then it must be inside the current tetrahedron. If (9) is not satisfied for a face and the corresponding adjacency matrix entry $(\Omega_{t,j})$ is zero, then $\mathbf{r}$ must be outside of all tetrahedra, and the procedure is stopped.

Using this approach, the algorithm is guaranteed to locate the tetrahedron containing $\mathbf{r}$ by "walking" in a nearly direct path from the starting guess. (For a 2-D example, see *Sambridge et al.* [1995]) If the starting guess is close to the solution, then it will be found rapidly. Fortunately, for a sequence of points along a ray the tetrahedron containing the previous point is an excellent starting guess for the next point. If two neighbor points are not located in the same or neighboring cells, then we can simply reduce the step size to ensure that no cell intersected by the ray is missed. In most tomographic problems, there are usually many rays per receiver, and so it is worthwhile to locate each receiver in advance and start stepping from the known receiver each time, thereby avoiding many relocations of the same endpoint.

3.3. Voronoi Polyhedra

The algorithm for locating the Voronoi polyhedron containing a given point has a similar structure to the one above. Again a preprocessing stage is required to generate a natural neighbor matrix, which contains the natural neighbors of each Voronoi cell; these are the ones which share a face. For example, in Figure 2b, cells 3 and 4 are neighbors, but 4 and 5 are not neighbors. Since polyhedra can have any number of faces, they can also have any number of neighboring cells (not just 4 as in the tetrahedra case). The natural neighbor list is therefore written as $\Theta_{k,j}(k = 1, \ldots, N_p; j = 1, \ldots, N_k)$, where N_p is the number of Voronoi cells (or nodes) and N_k is the number of cells sharing a face with cell k. A defining feature of Voronoi cells is that their neighbors are also the ones to which their node is connected in the Delaunay mesh (see Figure 2). As before, it is possible to calculate Θ using a "brute force" (order N_p^2) method, but this will become impractical for large N_p. A new more efficient (order N_p) method is described in the appendix.

Once Θ is known, the procedure to locate the Voronoi cell containing any point $\mathbf{r}$, is similar to the tetrahedra case. Again, we start with a first guess cell k and test whether $\mathbf{r}$ is inside cell k. This is only true if all of the following inequalities hold,

$$(\mathbf{r} - \mathbf{r}_k)\cdot(\mathbf{r} - \mathbf{r}_k) \leq (\mathbf{r} - \mathbf{r}_{k,j})\cdot(\mathbf{r} - \mathbf{r}_{k,j}) \quad j = 1, \ldots, N_k \tag{10}$$

where $\mathbf{r}_k$ and $\mathbf{r}_{k,j}$ are the position vectors of node k and its jth neighbor, respectively. These conditions simply state that the point $\mathbf{r}$ is closer to node k than any of its neighbors, which is the definition of being inside a Voronoi cell. We therefore test each condition in turn, and as soon as one of them fails, we move into the cell whose node is closer to $\mathbf{r}$ and start again; that is we set k equal to the new cell $\Theta_{k,j_{\text{fail}}}$. Again, the algorithm is guaranteed to find the cell containing $\mathbf{r}$ no matter where it starts from. All comments regarding the efficiency of the method carry over from the Delaunay case.

4. Numerical Examples of Calculating Frechet Derivatives in Delaunay and Voronoi Cells

To demonstrate the efficiency of these algorithms for calculating Frechet derivatives, we use a whole Earth parameterization devised by *Guðmundsson and Sambridge* [1998] and examine computational times in detail. Figures 1a and 1b show the outside faces of 3-D Delaunay and Voronoi cells, respectively, from nodes distributed in the upper mantle. The nodes on the surface (Figure 1a) were chosen to lie on the boundaries between tectonic provinces. (Notice how the outline of the continents and subprovinces can be seen within the Delaunay mesh.) This layer of 4274 nodes is repeated at a depth of 700 km, forming a 3-D upper mantle mesh. In addition to these "tectonic" nodes a much larger number were added to represent the 3-D shape of 24 subducted slabs. The morphology of the slabs was determined by the contouring of the Engdahl, van der Hilst and Bullen (EHB) relocated earthquake catalogue *Engdahl et al.* [1998]. Nodes were positioned along upper and lower faces of each slab with a spacing of between 10 and 50 km. This resulted in a total of 68,173 nodes in the upper mantle. *Guðmundsson and Sambridge* [1998] used this 3-D model to impose a priori constraints on an inversion of teleseismic arrival time data for regionalized upper mantle structure.

The model is an example of the type of complex parameterization that can be easily built with a Delaunay mesh. A total of 410,220 Delaunay tetrahedra were calculated using the quickhull method of *Barber et al.* [1993]. The volumes of the tetrahedra varied by 3 or 4 orders of magnitude within the model. To demonstrate the efficiency of the Frechet derivative calculation, we traced rays in a 1-D Earth model from 105 events to 3033 receivers and calculated the ray length in each Voronoi cell encountered by each ray. The reference velocity model was divided into 2 km layers, and rays were traced using standard analytical expressions based on the Mohorovičić velocity distribution, $v(r) = ar^b$ [*Bullen and Bolt*, 1985]. Every ray had an upgoing and downgoing leg within the upper mantle, and ~350 points along each leg were used to perform the numerical integration in (7). For the 30,638 P rays this corresponds to 10.7 million repeated cell locations. In this type of calculation, there is always a trade-off between the accuracy of the ray length calculation (improved by increasing the number of nodes placed along the ray) and the computational cost of the calculation (reduced by decreasing the number of nodes along the

ray). In these examples we sample the ray quite finely (2 km vertical separation) to ensure accuracy in the computed ray lengths rather than speed of calculation.

In Table 1a we compare the computation time taken up by the Voronoi location algorithm relative to the "1-D ray tracing" time. The latter includes stepping along the rays in a depth-varying velocity model and adding ray segment lengths together. (Note that this would effectively be equal to the time for calculating Frechet derivatives in a cubic cell model, assuming that the cost of locating a point in the cubic grid were zero.) One can see immediately that in this example the cost of the location algorithm is $< 25\%$ of the ray tracing/Frechet derivative calculation in a 1-D Earth model (AK135 of *Kennett et al.* [1995]). In most real tomographic problems, additional work is required to solve the resulting linear system of equations (possibly several times), and so the relative cost of the location algorithm to the total computation will be minor and certainly not much greater than in a regular Cartesian grid. The "preprocessing" tasks of calculating the Delaunay vertices and building the natural neighbor matrices (using the algorithm in the appendix) add little to the overall cost. These steps are independent of the number of rays and took 152 and 6 s, respectively (on a Sun Sparc 10/40 workstation), which corresponds to 6% and 0.01%, respectively, of the P phase ray tracing.

Table 1b shows results of a second example, which uses both the Voronoi and Delaunay location algorithm. In this case the model is more complex and consists of two independent meshes. The Delaunay tetrahedra are used to represent the 3-D subduction zones, and the Voronoi polyhedra are used in the remainder of the upper mantle. The Voronoi cells are built about the surface and 700 km deep nodes, shown in Figure 1. This is the same as in the first model. The Delaunay tetrahedra were built from the subduction zone nodes only (after some refinement). In the Delaunay mesh, tetrahedra occur both within each subduction zone and between zones. We therefore label each tetrahedron as either type A, indicating that they are inside a slab, or type B, indicating that they are outside a slab. The Voronoi and Delaunay meshes are then combined by assigning a point $\mathbf{r}$ to be inside the Delaunay mesh only if it is inside a type A tetrahedron. If it is inside a type B tetrahedron, then the Delaunay mesh is ignored and the Voronoi cell is found which contains the point. Therefore, in this combined parameterization a "cell location" consists of a location in the Delaunay mesh, possibly followed by a location in the Voronoi mesh. By combining the two meshes in this way we are able to control more easily the position of cell boundaries (because the tetrahedra directly connect nodes, whereas the Voronoi cell boundaries always lie between nodes). For example, this feature makes it easier to represent the morphology of the slabs. In this case the time taken to perform the cell location is dominated by the Delaunay location, rather than the Voronoi cell location in the first example.

The combined slab mesh consists of 27,970 nodes and results in 126,472 tetrahedra. The Voronoi mesh is as before with 8548 polyhedra. Table 1b shows timing results of Frechet derivative calculations and cell location. In this case the cell location algorithm is more expensive but still adds only $\sim$40% extra cost to the combined ray tracing/Frechet derivative calculation. From Table 1b one can see that for the P phase the total number of locations performed over all rays is close to 30 million, and this requires only 1436 s of CPU. As with the previous example, we step down each ray from the receivers, which were located in advance. These results show that it is possible to construct very complex irregular parameterizations with Delaunay and Voronoi cells, and by using the algorithms presented here, there is no significant increase in the computational burden relative to using simple cubic cells.

5. Choosing Nodes in Tomographic Problems

The algorithms described here allow tomography to be performed in any parameterization based on Delaunay or Voronoi polyhedra. One of the advantages of

Table 1a. Relative Times for Computing Ray Lengths in the Voronoi Polyhedra and Ray Tracing in a 1-D Reference Model.

Phase	No. of Rays	Ray Tracing	Cell Location	Mesh[a]	List[a]
P	30638	71.3	23.1	5.4	0.2
PP	1941	76.1	23.9		
PcP	857	76.5	23.5		

All values are expressed as percentages. Cell location, Mesh and List refer to relative times taken by the cell location algorithm, Delaunay mesh generator, and neighbor list calculation, respectively.

[a]Preprocessing step, independent of the number of data.

Table 1b. Time Taken for Ray Tracing in Seconds Per 1000 Rays Compared to the Joint Voronoi/Delaunay Cell Location Algorithm in the Second 3-D Model.

Phase	No. of Rays	Ray Tracing	Cell Location	No. of Locations.[a]	Average Walk[b]	Maximum Walk[c]
P	44973	84.0	38.0	645429	1.011	10
PP	4352	171.1	40.7	1301822	1.039	53
PcP	638	80.6	36.4	645169	1.007	8

The cell location time is expressed as a percentage of the ray tracing time. The average length of the walk is close to unity, indicating nearly optimal efficiency. The time taken for mesh generation and list calculation was 64 s. All calculations were performed on a Sun Sparcstation 10/40 with 96 Mb of memory.

[a]Number of cell locations performed per 1000 rays.

[b]Average number of cells encountered during all locations.

[c]Maximum number of cells encountered during a single location.

these tools is that complete freedom is allowed in choosing the positions of the nodes. Inevitably, the most suitable choice will be problem dependent. Here we discuss some possibilities.

The first is to use the mesh design to impose a priori constraints on the inversion (as in the examples above). In this way one might incorporate information from other geological or geophysical studies on the expected character of the region. A priori information can be a useful aid in retrieving information from underdetermined problems (i.e., most real problems). This is the approach taken by *Gudmundsson and Sambridge* [1998]. Alternatively, one could use the flexibility of the parameterization to test a particular hypothesis, for example, whether the existence of certain structures in the model (perhaps suggested by independent evidence) is necessary to fit the data. This could be done by distributing the nodes so that the outline of the particular type of structure is represented by a subset of cells or nodal connections. The inversion would then be geared to testing out models within that class of structures.

Another strategy is to gear the density of nodes directly to the data density. In many tomographic studies, especially in three dimensions, the data density can vary greatly across a model, with some areas obviously undersampled. By gearing the nodal density to the ray density we can improve resolution in well-constrained parts of the model and reduce the number of model parameters in poorly constrained regions. It is conceivable that this could be extended to include densities based on ray "tubes" whose widths are estimated from Fresnel volumes [e.g., *Stark and Nikolayev*, 1993; *Vasco and Mayer*, 1993; *Gudmundsson*, 1996].

The choice of parameterization is only one way of regularizing an inverse problem; explicit tools of regularization, for example, damping of the resulting linear systems of equations, will still be important. Nevertheless, the freedom to place nodes anywhere in the volume of interest is powerful. One possibility, currently under investigation, is to create a self-adaptive parameterization. An example would be where one solves an initial tomographic system based on a sparsely distributed set of nodes and then places new nodes in the parts of the model where structural features where resolved in the first solution. A second tomographic system would then be more responsive to parts of the model where structure was required to fit the data. Of course, this procedure could be repeated more than once and could lead to a parameterization which is geared to both the density and the signal in the original data.

6. Conclusions

Our objective in this paper is to describe how to perform seismic or any other "ray based" tomography in the class of irregular mesh built from Delaunay or Voronoi cells. We have attempted to include enough detail to enable the reader to develop the necessary computational tools themselves. This is necessary because much of the work is based on concepts from the field of computational geometry and will be unfamiliar to many geophysicists. (The software used in this paper is available from the web site `http://rses.anu.edu.au/seismology/projects/tireg/tomo.html`).

We have considered several types of basis functions built around the irregular parameterizations and given details of efficient algorithms for Frechet derivative calculation. This paper deals only with linearized tomography. All ray tracing is performed in simple depth-varying velocity models. To extend the approach to nonlinear tomography would require two-point ray tracing to be performed in laterally heterogeneous velocity models represented by these irregular cellular parameterizations. This is the subject of ongoing work [*Davies and Sambridge*, 1996].

The numerical examples demonstrate that the new algorithms are efficient and will add only a small overhead to the computational cost of building a tomographic system of equations, compared to a regular Cartesian mesh. We have also offered some suggestions on strategies for choosing the positions of nodes and how this may be exploited in tomography. It is now possible to

take advantage of this highly flexible class of parameterization in tomographic problems.

Appendix: Adjacency and Natural Neighbor Matrices in Three Dimensions

A1. The Adjacency Matrix

The adjacency matrix gives the neighbors of each tetrahedron. $\Omega_{t,j}(t = 1,\ldots,N_t; j = 1,\ldots,4)$ is the index of the tetrahedron attached to the face opposite the jth vertex of tetrahedron t. It can be calculated efficiently using a single loop over tetrahedra (rather than a double loop). The structure of the algorithm is to consider each tetrahedron in turn and store its index, say t_1, at each of its four faces, which must be uniquely identified in some way. If for tetrahedron t_2 an index is already stored at one of its faces, say t_1, then t_1 and t_2 must be neighbors through this face. In this way all pairs of neighboring tetrahedra can be found with a single pass. Since the data structures used here are arranged in terms of nodes (through the vertices list $V_{t,j}$) and not faces, the "storing of a t index at a face" is not straightforward. It is done by associating each face of a tetrahedron, t, with a node in the face, say node v_1, and storing the triplet of indices (v_2, v_3, and t) at that node, where v_2 and v_3 are the other two nodes in that face. Provided each face is stored at only one node, then through the complete loop all interior faces will be sampled twice, and so all neighboring pairs of tetrahedra will be found. More precisely, for $t_1 = 1,\ldots,N_t$ and for $j_1 = 1,\ldots,4$ we set

$$j_k = \mathrm{mod}(j_{k-1}, 4) + 1 \qquad k = 2, 3, \text{ and } 4 \qquad \text{(A1a)}$$

$$i_k = V_{t_1,j_k} \qquad k = 1, 2, \text{ and } 3 \qquad \text{(A1b)}$$

If the triplet of indices (i_1, i_3 and t_1) is not stored at node i_k then we add the triplet (i_1, i_3, and t_1) to the triplets already stored at node i_k. Otherwise set

$$t_2 = S^c_{i_1} \qquad \text{(A1c)}$$

$$\Omega_{t_1,j_4} = t_2 \qquad \text{(A1d)}$$

$$\Omega_{t_2,j_*} = t_1 \qquad \text{(A1e)}$$

where $\mathrm{mod}(i, j)$ is the modulus function and j_* is the index of the face of tetrahedron t_2 containing nodes i_1, i_2, and i_3. After this loop has been completed the adjacency matrix $\Omega_{t,j}$ will have been determined.

A2. The Natural Neighbor Matrix

The algorithm for calculating the list of natural neighbors of each node is straightforward. The first stage is to calculate the list of tetrahedra attached to each node, which is represented as $L^t_{i,j}$, i.e., the index of the jth tetrahedron ($j = 1,\ldots,n^t_i$) attached to node i ($i = 1,\ldots,N_p$). This can also be found with a single loop over the tetrahedra, i.e., for $t = 1,\ldots,N_t$ and for $j = 1,\ldots,6$ we set

$$i = V_{t,j} \qquad \text{(A2a)}$$

$$n^t_i = n^t_i + 1 \qquad \text{(A2b)}$$

$$L^t_{i,n^t_i} = t \qquad \text{(A2c)}$$

At the end of this loop, n^t_i is the number of tetrahedra which have node i at a vertex. In the next stage the natural neighbor list of each node can be built by simply recording those nodes with which it shares a tetrahedron. This is done by performing a loop over the tetrahedra attached to each node; that is, for $i = 1,\ldots,N_p$ and for $j = 1,\ldots,n^t_i$ we set

$$t = L^t_{i,j} \qquad \text{(A3a)}$$

and for each node $V_{t,k}$ ($k = 1,\ldots,4$) which is not equal to i and not already recorded in the list $\Theta_{i,l}$ ($l = 1,\ldots,n^n_i$) we set

$$n^n_i = n^n_i + 1 \qquad \text{(A3b)}$$

$$\Theta_{i,n^n_i} = V_{t,k} \qquad \text{(A3c)}$$

After this loop is completed, $\Theta_{i,l}$ contains the index of the lth node, which is attached to node i as required. Note that this approach requires only one loop over the tetrahedra and one loop over the nodes. Its computational cost therefore scales linearly with the number of nodes and tetrahedra.

Acknowledgments. We are grateful to J. Braun for useful discussions on the algorithms presented in this paper. We also thank C. Thurber, who provided useful comments, and an anonymous referee for constructive criticisms on an earlier version of the manuscript. This work was supported in part by grants F49620-94-1-0022 and F49620-94-1-0110 from the U.S. Air Force Office of Scientific Research.

References

Abers, G. G., and S. W. Roecker, Deep structure of an arc-continent collision: Earthquake relocation and inversion for upper mantle P and S wave velocities beneath Papua New Guinea, *J. Geophys. Res., 96*, 6379-6401, 1991.

Aurenhammer, F., Voronoi Diagrams: A survey of a fundamental geometric data structure, *Assoc. Comput. Mach. Comput. Surv., 23* (3), 345-405, 1991.

Barber, B., D. P. Dobkin, and H. Huhdanpaa, The quickhull algorithm for convex hull, Tech. Rep. GCG53, The Geometry Cent., Univ. of Minn., Minneapolis, 1993.

Braun, J., and M. Sambridge, A numerical method for solving partial differential equations on highly irregular evolving grids, *Nature, 376*, 655-660, 1995.

Braun, J., and M. Sambridge, Modeling landscape evolution on geologic time scales: A new method based on irregular spatial discretization, *Basin Res., 9*, 27-52, 1997.

Bullen, K. E., and B. A. Bolt, *An Introduction to the Theory of Seismology*, 4th Ed., Cambridge Univ. Press, New York, 1985.

Davies, J. H., and M. Sambridge, 3-D ray tracing in Delaunay tetrahedra using ray perturbation theory, *Eos Trans. AGU, 77* (22), West Pac. Geophys. Meet. Suppl., W86, 1996.

Engdahl, E. R., R. D. van der Hilst, and R. Buland, Global teleseismic earthquake relocation with improved travel

times and procedures for depth determination, *Bull. Seismol. Soc. Am.*, in press, 1998.

Guðmundsson, Ó., On the effect of diffraction on traveltime measurements, *Geophys. J. Int.*, *124*, 304-314, 1996.

Guðmundsson, Ó., and M. Sambridge, A regionalized upper-mantle (RUM) seismic model, *J. Geophys. Res., in press*, 1998.

Kennett, B. L. N., E. R. Engdahl, and R. Buland, Travel times for global earthquake location and phase association, *Geophys. J. Int.*, *122*, 108-124, 1995.

Michelini, A., An adaptive-grid formalism for traveltime tomography, *Geophys. J. Int.*, *121*, 489-510, 1995.

Okabe, A., B. Boots, and K. Sugihara, *Spatial Tessellations Concepts and Applications of Voronoi Diagrams*, John Wiley, New York, 1992.

Parker, R. L., L. Shure, and J. A. Hildebrand, The application of inverse theory to seamount magnetism, *Rev. Geophys.*, *25*, 17-40, 1987.

Sambridge, M., J. Braun, and H. McQueen, Geophysical parametrization and interpolation of irregular data using natural neighbours, *Geophys. J. Int.*, *122*, 837-857, 1995.

Sibson, R., A vector identity for the Dirichlet tessellation, *Math. Proc. Cambridge Philos. Soc.*, *87*, 151-155, 1980.

Sloan, S. W., A fast algorithm for constructing Delaunay triangulations in the plane, *Adv. Eng. Software*, *9* (1), 34-55, 1987.

Spakman, W., Delay-time tomography of the upper mantle below Europe, the Mediterranean, and Asia Minor, *Geophys. J. Int.*, *107*, 309-332, 1991.

Stark, P., and D. L. Nikolayev, Toward tubular tomography, *J. Geophys. Res.*, *98*, 8095-8106, 1993.

Vasco, D. W., and E. L. Mayer, Wavepath travel time tomography, *Geophys. J. Int.*, *115*, 1055-1069, 1993.

Watson, D. F., *Contouring: A Guide to the Analysis and Display of Spatial Data*, Pergamon, Tarrytown, N. Y., 1992.

Zhou, H.-W., and R. W. Clayton, *P* and *S* wave travel time inversions for subducting slab under the island arcs of the northwest Pacific, *J. Geophys. Res.*, *95*, 6829-6851, 1990.

Ó. Gudmundsson and M. Sambridge, Research School of Earth Sciences, Institute of Advanced Studies, Australian National University, Canberra, A.C.T. 0200, Australia. (e-mail: oli@rses.anu.edu.au; malcolm@rses.anu.edu.au

(Received January 16, 1997; revised September 2, 1997; accepted September 11, 1997.)