Algorithm 772: STRIPACK: Delaunay Triangulation and Voronoi Diagram on the Surface of a Sphere

ROBERT J. RENKA
University of North Texas

STRIPACK is a Fortran 77 software package that employs an incremental algorithm to construct a Delaunay triangulation and, optionally, a Voronoi diagram of a set of points (nodes) on the surface of the unit sphere. The triangulation covers the convex hull of the nodes, which need not be the entire surface, while the Voronoi diagram covers the entire surface. The package provides a wide range of capabilities including an efficient means of updating the triangulation with nodal additions or deletions. For N nodes, the storage requirement for the triangulation is $13N$ integer storage locations in addition to $3N$ nodal coordinates. Using an off-line algorithm and work space of size $3N$, the triangulation can be constructed with time complexity $O(N\log N)$.

Categories and Subject Descriptors: G.1.1 [**Numerical Analysis**]: Interpolation; G.1.2 [**Numerical Analysis**]: Approximation; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms

Additional Key Words and Phrases: Delaunay triangulation, Dirichlet tessellation, sphere, Thiessen regions, Voronoi diagram

1. INTRODUCTION

The Delaunay triangulation [Delaunay 1934] and its dual, the Voronoi diagram [Voronoi 1908], have a large and growing number of applications, among which are finite-element grid generation, scattered data fitting, and the efficient solution of closest-point problems [Preparata and Shamos 1985]. Most of the literature on these structures describes the planar case, but the applications also frequently arise in $\mathbf{R}^3$ and on the surface of a sphere [Augenbaum and Peskin 1985; Renka 1984a; 1997].

This material is based upon work supported by the National Science Foundation under Award No. CCR-9111671.
Author's address: Department of Computer Sciences, University of North Texas, Denton, TX 76203-1366.

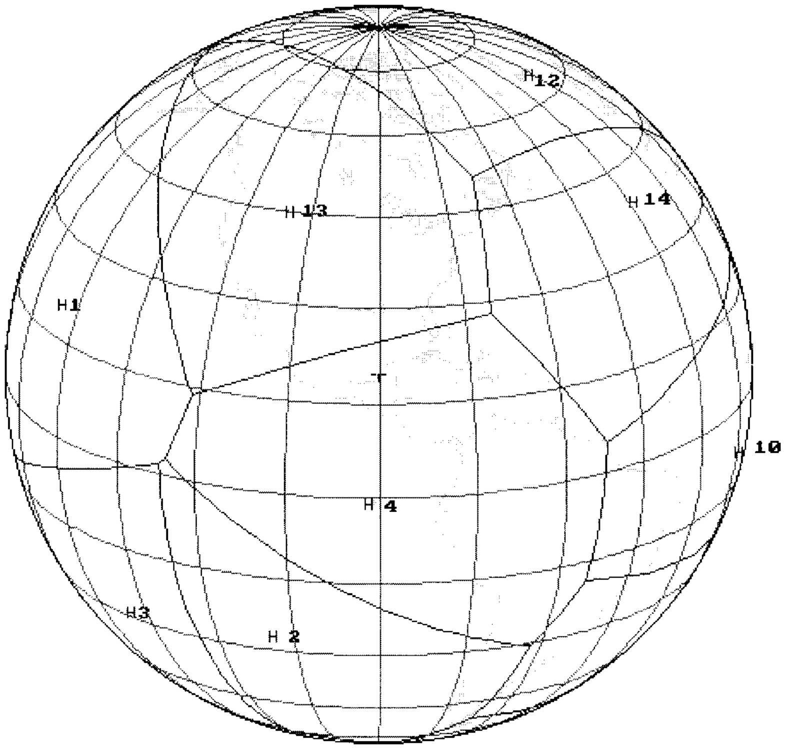

Fig. 1. Hotspot-centered Voronoi regions (convection cells).

STRIPACK is a revision of the triangulation portion of Algorithm 623 [Renka 1984b]. The language has been updated to Fortran 77; the data structure has been changed; the time complexity has been reduced; and a number of capabilities have been added. The additions include a subroutine that constructs the Voronoi diagram associated with a given Delaunay triangulation, and a function that locates a point relative to a polygon on the surface of the sphere. These are further discussed in Section 3. Both were motivated by the development of a new model of plate tectonics in which plate motion is driven by the frictional forces exerted by the underlying mantle [Fohlmeister and Renka 1994; Fohlmeister et al. 1990]. It is assumed that the mantle flow at the surface is directed radially outward from each of a set of hotspots at various locations on the surface of the earth (modeled by the unit sphere). We partition the surface into a set of Voronoi regions (spheres of influence), each centered on a hotspot. Refer to Figure 1.

We present definitions in Section 2, some theoretical background in Section 3, a discussion of algorithms in Section 4, and a description of the software package in Section 5.

2. DEFINITIONS AND TERMINOLOGY

Denote the unit sphere centered at the origin by

$$U = \{p \in \mathbf{R}^3 : \|p\| = 1\},$$

where $\|\cdot\|$ is the Euclidean norm on $\mathbf{R}^3$. A metric on U is defined by $d(p, q) = arccos\langle p, q\rangle \forall p, q \in U$, where $\langle \cdot , \cdot \rangle$ is the inner product associated with $\mathbf{R}^3$, and the angular separation (arclength) between p and q is in the interval $[0, \pi]$. A subset $H \subset U$ is *convex* if, for every pair of points p, $q \in H$, a geodesic joining p to q (uniquely defined unless $d(p, q) = \pi$) is contained in H. The *convex hull* H of a finite set S of points of U is the smallest convex set that contains all the points. Note that, if the elements of S are contained in a single hemisphere, H has an interior (the smaller region) and an exterior (the larger region) with a boundary curve consisting of geodesics joining elements of S. Otherwise, H is the entire sphere surface. The convex hull of three distinct points is a (spherical) *triangle* with the points as vertices. If the vertices are collinear (lie on a common great circle), the triangle is *null* (has zero area).

Let $S = \{p_i\}_{i=1}^N$ denote a set of N distinct elements of U, referred to as *nodes*, not lying on a common great circle, for $N \geq 3$, and let H denote the convex hull of S. A *triangulation* T of S is a set of triangles that satisfy the following properties:

(1) the triangle vertices are nodes,

(2) no triangle contains a node other than its vertices,

(3) the triangle interiors are pairwise disjoint, and

(4) the union of triangles covers H.

It follows from the definition that the set of vertices coincides with S and the triangulation partitions H. If H is contained in a hemisphere, the triangulation has *boundary edges* (triangle sides that are not shared by two triangles), the endpoints of which are referred to as *boundary nodes*. An edge that is not a boundary edge is an *interior edge*, and a node that is not a boundary node is an *interior node*.

A triangle with vertices p_i, p_j, and $p_k \in S$ will be denoted (p_i, p_j, p_k), where the vertices are specified in counterclockwise order, i.e., $det(p_i, p_j, p_k) = \langle p_i \times p_j, p_k\rangle \geq 0$, where $det(p_i, p_j, p_k)$ denotes the determinant of the matrix with rows (columns) p_i, p_j, and p_k in that order, and $p_i \times p_j$ denotes the vector cross product. The *circumcenter* of a triangle (p_i, p_j, p_k) is

$$v_{ijk} = \frac{(p_j - p_i) \times (p_k - p_i)}{\|(p_j - p_i) \times (p_k - p_i)\|}.$$

Note that since the vertices cannot lie on a common line, v_{ijk} is well defined even for a null triangle. Also, since $\langle v_{ijk}, p_l \rangle = det(p_i, p_j, p_k) \, / \, \|(p_j - p_i) \times (p_k - p_i)\|$ for $l \in \{i, j, k\}$, v_{ijk} is equidistant from the vertices (as is $-v_{ijk}$, but no other point of U). v_{ijk} and $-v_{ijk}$ are the intersections of the perpendicular bisectors (great circles) of the triangle sides. The *circumradius* of (p_i, p_j, p_k) is $r_{ijk} = d(v_{ijk}, p_l)$ for $l \in \{i, j, k\}$, and the *circumcircle* of (p_i, p_j, p_k) is

$$C_{ijk} = \{p \epsilon U : d(v_{ijk}, p) = r_{ijk}\}.$$

Note that $0 < r_{ijk} \leq \pi \, / \, 2$, and C_{ijk} contains p_i, p_j, and p_k.

A *Delaunay triangulation* T^* of S is a triangulation of S with the *empty circumcircle interior property*, i.e., no circumcircle of a triangle of T contains a node in its interior. It is uniquely defined up to the neutral case of four or more nodes lying on a common circle (plane). Delaunay triangulations have also been referred to as *Thiessen triangulations*. The circumcircle property implies a reasonably uniform triangulation which provides a good basis for interpolation of data values associated with the nodes [Lawson 1984; Renka 1984a; 1984b].

For each node $p_i \in S$, we define a *Voronoi region* R_i as the closure of the set of points closer to p_i than to any other node, i.e.,

$$R_i = \{p \in U : d(p, p_i) \leq d(p, p_j) \forall p_j \in S\}.$$

The set of Voronoi regions $\{R_i\}_{i=1}^{N}$ partitions U into a *Voronoi diagram*, also termed a *Dirichlet* or *Thiessen tessellation* [Rhynsburger 1973]. A *Voronoi edge* is the intersection of two Voronoi regions that share a point, and a *Voronoi vertex* is the intersection of three or more Voronoi regions that share a point. Hence Voronoi edges are portions of perpendicular bisectors of geodesics joining pairs of nodes, and Voronoi vertices are circumcenters or negative circumcenters of triangles with vertices in S.

3. DUALITY AND EXTENSION OF THE TRIANGULATION

We now illustrate the duality between the Delaunay triangulation and the Voronoi diagram. This provides the basis for an algorithm that constructs one from the other. The theory for the analogous planar tessellations is well known [Lawson 1977; Sibson 1978]. In that case, there is a one-to-one correspondence between Voronoi regions and triangulation vertices (nodes); Voronoi edges are portions of perpendicular bisectors of triangulation edges; and the set of Voronoi vertices coincides with the set of triangle circumcenters. (In the neutral case of four or more nodes lying on a common circle, one or more Voronoi edges degenerates to a point.) Thus, given a Delaunay triangulation, the Voronoi region associated with a node is defined by the (cyclically) ordered sequence of circumcenters of the triangles containing the node. Conversely, a Delaunay triangulation may be constructed from a Voronoi diagram by connecting nodes whose Voronoi regions share more than one point and, in the case of k nodes on a common circle for $k > 3$, choosing an arbitrary set of $k - 3$ nonintersecting edges. Both algorithms have operation counts of $O(N)$.

We will show that the case of nodes on the surface of the sphere is completely analogous to the planar case if the convex hull H is the entire surface. This restriction seems to be an implicit assumption in the literature [Augenbaum and Peskin 1985]. If the nodes are contained in a single hemisphere, however, the set of Voronoi vertices is a superset of the triangulation circumcenters. The question is then how to select the additional Voronoi vertices in an algorithm for constructing the Voronoi diagram from a given Delaunay triangulation. This section is primarily addressed to answering that question.

Consider the simplest case in which $N = 3$. There are two Voronoi vertices: the circumcenter and its negative. In the case of four nodes and two triangles, the Voronoi vertices are the two circumcenters along with the negatives of the circumcenters of the other possible pair of triangles (the two triangles obtained by swapping the shared edge for the other diagonal). More generally, there are $2N - 4$ Voronoi vertices, corresponding to the number of triangles in a triangulation that covers the surface, while the number of triangles is $2N - N_B - 2$ for N_B boundary nodes if $N_B > 0$. Thus the number of additional Voronoi vertices is $N_B - 2$. This is the number of triangles in a triangulation in which all nodes are boundary nodes, and in fact the solution is to construct a triangulation T_B of the boundary nodes using an "empty circumcircle exterior" criterion. The additional Voronoi vertices are then the negative circumcenters of the triangles of T_B. These may be thought of as the circumcenters of a set of pseudotriangles that cover the exterior of H. The following results make these ideas more precise.

LEMMA 3.1. *A triangle circumcenter or its negative $q = \pm v_{ijk}$ is a Voronoi vertex if and only if q is not contained in the interior of a Voronoi region, i.e., for every node $p_l \in S$, $d(q, p_l) \geq d(q, p_m)$ for some node $p_m \neq p_l$.*

PROOF. This follows immediately from the definitions. □

Let T^* be a Delaunay triangulation of S.

THEOREM 3.2. *If $(p_i, p_j, p_k) \in T^*$, then v_{ijk} is a Voronoi vertex.*

PROOF. Suppose $(p_i, p_j, p_k) \in T^*$. Then $d(v_{ijk}, p_l) \geq r_{ijk} \forall p_l \in S$ by the empty circumcircle interior criterion. Hence, for every $p_l \in S$, $d(v_{ijk}, p_l) \geq d(v_{ijk}, p_m)$ for some $p_m \neq p_l$ (using $m \in \{i, j, k\} - \{l\}$), i.e., v_{ijk} is a Voronoi vertex by Lemma 3.1. □

LEMMA 3.3. *Suppose p_i is an interior node of T^*. Then, for every triangle (p_i, p_j, p_k) of T^* that contains p_i, $-v_{ijk}$ is closer to some node p_l than to p_i, p_j, or p_k.*

PROOF. If some circumcircle C_{ijk} contains all nodes, then every node is a boundary node. Hence, for an interior node p_i, C_{ijk} has a node in its exterior, i.e., $\exists p_l \in S$ such that $d(v_{ijk}, p_l) > r_{ijk}$, implying that $d(-v_{ijk}, p_l) = \pi - d(v_{ijk}, p_l) < \pi - r_{ijk} = d(-v_{ijk}, p_m)$ for $m \in \{i, j, k\}$. □

Let p_l be the closest node to $-v_{ijk}$. Then either $-v_{ijk}$ is not a Voronoi vertex (it is interior to R_l) or it is a Voronoi vertex (circumcenter or negative circumcenter) associated with some triangle containing p_l. In either case, we have the following.

THEOREM 3.4. *The negative circumcenter $-v_{ijk}$ of a triangle of T^* that contains an interior node need not be included in the set of Voronoi vertices. (If it is a Voronoi vertex, it coincides with the circumcenter of another triangle or the negative circumcenter of a triangle containing only boundary nodes.) Thus, if all nodes are contained in a single hemisphere, the additional Voronoi vertices (other than v_{ijk} for $(p_i, p_j, p_k) \in T^*$) are negative circumcenters of triangles containing only boundary nodes.*

By Theorem 3.4, the additional Voronoi vertices associated with nodes in a single hemisphere are obtained by triangulating the set of N_B boundary nodes S_B. Let T_B^* denote a triangulation of S_B in which all triangle circumcircles have no nodes in their exteriors: the complement of a Delaunay triangulation.

THEOREM 3.5. *For each triangle $(p_i, p_j, p_k) \in T_B^*$, $-v_{ijk}$ is a Voronoi vertex (in the Voronoi diagram associated with S).*

PROOF. For $(p_i, p_j, p_k) \in T_B^*$, all nodes of S_B are on or interior to C_{ijk}, i.e., for every node $p_l \in S_B$, $d(v_{ijk}, p_l) \leq r_{ijk}$. Hence $d(-v_{ijk}, p_l) = \pi - d(v_{ijk}, p_l) \geq \pi - r_{ijk} = d(-v_{ijk}, p_m) \; \forall \, m \in \{i, j, k\}$.
Thus $-v_{ijk}$ is not interior to a Voronoi region and is therefore a Voronoi vertex by Lemma 3.1. □

As in the case of the Delaunay triangulation, T_B^* can be constructed from an arbitrary triangulation T_B of S_B by swapping diagonal arcs in convex quadrilaterals consisting of pairs of adjacent triangles. Note that, in the case of T_B^*, all such quadrilaterals are necessarily convex. The swap test applied to interior edges is a local application of the circumcircle criterion: the shared edge in a pair of adjacent triangles is swapped out if and only if the fourth vertex is exterior to the circumcircle of (either) one of the triangles. In the case of collinear nodes, T_B^* will contain null triangles, but their circumcenters are well defined.

The following theorem provides a simple computational procedure for applying the swap test without computing distances. The operation count is 9 multiplications, 14 additions, and a comparison.

THEOREM 3.6. *Given adjacent triangles* (p_i, p_j, p_k) *and* (p_j, p_i, p_l) *in* T_B, *the diagonals should be swapped if and only if*

$$det(p_j - p_i, p_k - p_i, p_l - p_i) < 0.$$

PROOF.

$$\begin{aligned}
& d(v_{ijk}, p_l) > r_{ijk} \\
& \Leftrightarrow arccos\langle v_{ijk}, p_l\rangle > arccos\langle v_{ijk}, p_i\rangle \\
& \Leftrightarrow \langle v_{ijk}, p_l\rangle < \langle v_{ijk}, p_i\rangle \\
& \Leftrightarrow \langle v_{ijk}, p_l - p_i\rangle = \left\langle \frac{(p_j - p_i) \times (p_k - p_i)}{\|(p_j - p_i) \times (p_k - p_i)\|}, p_l - p_i \right\rangle < 0 \\
& \Leftrightarrow det(p_j - p_i, p_k - p_i, p_l - p_i) < 0
\end{aligned}$$

□

Corollary 3.7. The swap test is well defined, i.e., given adjacent triangles (p_i, p_j, p_k) and (p_j, p_i, p_l) in T_B,

$$\begin{aligned}
& p_l \text{ is exterior to } C_{ijk} \\
& \Leftrightarrow p_k \text{ is exterior to } C_{jil} \\
& \Leftrightarrow p_i \text{ is interior to } C_{klj} \\
& \Leftrightarrow p_j \text{ is interior to } C_{lki}.
\end{aligned}$$

PROOF. By Theorem 3.6, p_l is exterior to C_{ijk} if and only if $det(p_j - p_i, p_k - p_i, p_l - p_i) = det(p_j, p_k, p_l) + det(p_i, p_k, p_j) + det(p_i, p_j, p_l) + det(p_i, p_l, p_k) < 0$. Permuting subscripts, p_k is exterior to C_{jil} if $det(p_i - p_j, p_l - p_j, p_k - p_j) = det(p_i, p_l, p_k) + det(p_j, p_l, p_i) + det(p_j, p_i, p_k) + det(p_j, p_k, p_l) < 0$. Equivalence of the first two criteria follows from equality of the determinants. The other results are obtained by a similar computation: p_i is interior to C_{klj} if and only if $det(p_l - p_k, p_j - p_k, p_i - p_k) < 0$, and p_j is interior to C_{lki} if and only if $det(p_k - p_l, p_i - p_l, p_j - p_l) > 0$, where $det(p_l - p_k, p_j - p_k, p_i - p_k) = det(p_k - p_l, p_i - p_l, p_j - p_l) = - det(p_j - p_i, p_k - p_i, p_l - p_i)$. □

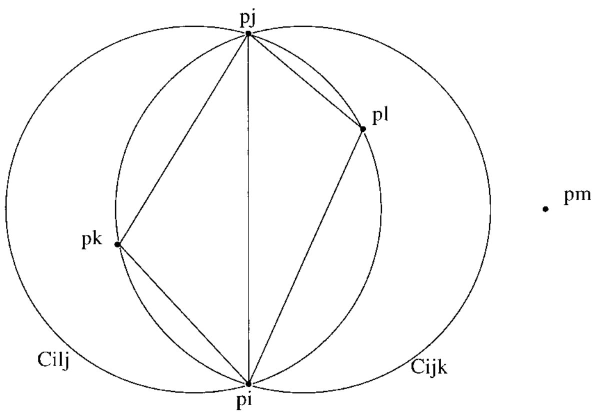

Fig. 2. Theorem 3.8.

The following theorem relates the local circumcircle criterion to the global property defining T_B^*. We define an edge of T_B to be *locally optimal* if and only if it is a boundary edge or it is an interior edge that satisfies the local circumcircle criterion.

THEOREM 3.8. *All circumcircles of T_B have empty exteriors ($T_B = T_B^*$) if and only if all edges of T_B are locally optimal.*

PROOF. The "only if" part follows immediately from the definitions. For the other part, suppose the theorem is false, i.e., that all edges are locally optimal and that there is a circumcircle C_{ijk} with some node p_m in its exterior. Refer to Figure 2. We may assume without loss of generality that p_k and p_m lie on opposite sides of the edge e_{ij} joining p_i and p_j, i.e., e_{ij} is an interior edge, and $det(p_i, p_j, p_m) < 0$. Among all triangles of T_B whose circumcircles have p_m as an exterior point, assume without loss of generality that none is closer to p_m in the sense that they share a point with the geodesic joining p_m to an interior point of e_{ij}. Let (p_i, p_l, p_j) be the triangle that shares edge e_{ij} with (p_i, p_j, p_k). Since e_{ij} is locally optimal, p_l is not exterior to C_{ijk}, and $p_l \neq p_m$. If both e_{il} and e_{lj} were boundary edges, then p_m would be contained in (p_i, p_l, p_j). Hence either e_{il} is an interior edge with $det(p_i, p_l, p_m) < 0$, or e_{lj} is an interior edge with $det(p_l, p_j, p_m) < 0$. Hence p_m is closer to (p_i, p_l, p_j) than (p_i, p_j, p_k), and a contradiction will be reached if it is shown that p_m is exterior to C_{ilj}. The great circle containing p_i and p_j defines two hemispheres, one of which contains p_l, and the portion of C_{ilj} that lies in this hemisphere is on or interior to C_{ijk}. It follows that p_m is exterior to C_{ilj}. □

The following algorithm constructs T_B^* incrementally.

(1) Cyclically order the nodes of S_B in counterclockwise order. Denote the sequence by $\{p_i\}_{i=1}^{N_B}$.

(2) Initialize the triangulation to $T_3 = \{(p_1, p_2, p_3)\}$.

(3) For $k = 4$ to N_B
   (a) construct T_k by connecting p_k to p_1 and p_{k-1}, i.e., $T_k = T^*_{k-1} \cup \{(p_1, p_{k-1}, p_k)\}$
   (b) construct T^*_k from T_k by applying the swap test and appropriate swaps to the interior edges opposite p_k beginning with the edge joining p_1 to p_{k-1}. Following each swap there are two new edges opposite p_k which, if not boundary edges, must be tested for swaps.

We will show that the algorithm results in $T^*_{N_B} = T^*_B$. First note that any reasonable data structure for a triangulation T of S includes the ordered sequence of boundary nodes implicitly, if not explicitly. The operation count for step (1) is therefore at most $O(N_B)$ given the index of some boundary node. The worst-case operation count for step (3b) is $O(k)$, resulting in $O(N_B^2)$ for step (3) but, for randomly generated nodes, step (3b) runs in constant expected time, resulting in an operation count linear in N_B.

Since the nodes lie on the boundary of a convex region, p_1 and p_{k-1} are the only nodes visible from p_k at step (3a), and execution of the step results in a triangulation T_k of the first k nodes. To show that all circumcircles of T^*_k have empty exteriors, it suffices to show that all edges are locally optimal following step (3b). This follows from Theorem 3.8. It is sufficient to show that, following each swap, the new edge containing p_k will remain locally optimal regardless of subsequent swaps (but may be swapped out when additional nodes are added).

THEOREM 3.9. *Let T^*_{k-1} be a triangulation of $\{p_i\}_{i=1}^{k-1}$ in which no circumcircle has a node in its exterior, and let T_k be a triangulation of $\{p_i\}_{i=1}^{k}$. Let (p_i, p_j, p_k) and (p_j, p_i, p_l) be adjacent triangles of T_k, where (p_j, p_i, p_l) is also a triangle of T^*_{k-1}. Suppose the swap test results in edge e_{ij} being replaced by edge e_{kl}. Then no sequence of swap tests and appropriate swaps will result in the removal of e_{kl}. Note that p_j, p_i, and p_l are not necessarily the first vertices to which p_k was connected.*

PROOF. By the assumptions, p_k is the only node exterior to C_{jil}. Suppose that at some point e_{kl} is to be swapped for some edge e_{rs} joining nodes p_r and p_s (either of which might coincide with p_i or p_j). Refer to Figure 3. Then p_r and p_s must lie on opposite sides of e_{kl}; p_k and p_l must lie on opposite sides of e_{rs}; and p_l must be interior to C_{rks} (by Theorem 3.6). Partition C_{rks} into three arcs $\widehat{p_r p_k}$, $\widehat{p_k p_s}$, and $\widehat{p_s p_r}$. Since p_k is exterior to C_{jil} and p_r is not, $\widehat{p_r p_k}$ must intersect C_{jil} at some point q_r (which may coincide with p_r). Similarly, $\widehat{p_k p_s}$ intersects C_{jil} at some point q_s. Thus, C_{rks} is partitioned into two portions: $\widehat{q_r p_k q_s}$ outside of C_{jil} and $\widehat{q_s p_s p_r q_r}$

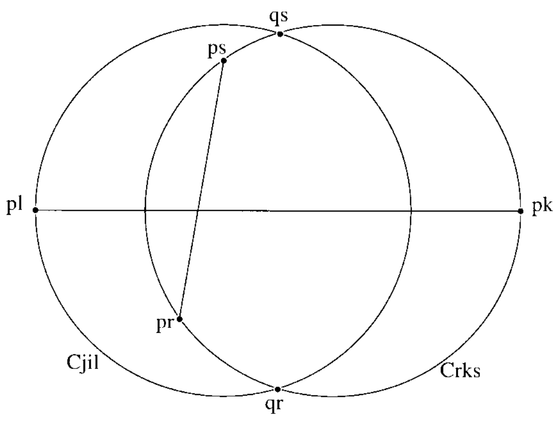

Fig. 3. Theorem 3.9.

inside C_{jil}. The arc of C_{jil} between q_s and q_r that lies outside C_{rks} contains p_l, contradicting the assumption that p_l is interior to C_{rks}. □

4. COMPUTATIONAL PROCEDURES

With the exception of the floating-point computations, the triangulation is constructed by the same incremental algorithm that we have used in the planar case [Cline and Renka 1984; Lawson 1977]. The incremental algorithm has two advantages over an off-line algorithm such as divide-and-conquer: it is not necessary that all nodes be present before processing begins, and it allows for efficiently updating the triangulation with nodal additions and deletions. As described in Section 4.2, the optimal $O(N\log N)$ run time can be achieved in the case that all nodes are immediately available. The package also allows a set of prespecified triangulation arcs to be included, resulting in a constrained Delaunay triangulation [Cline and Renka 1990].

4.1 Floating-Point Computations

A new node to be added to a triangulation is located relative to the triangulation and connected to all visible nodes, resulting in a new triangulation which is then optimized (converted to a Delaunay triangulation) by systematically applying swap tests and the appropriate swaps to the interior arcs opposite the new node. (Following each swap, there are two new arcs opposite the new node, and these must be tested for swaps.) The algorithm that locates a point $p \in U$ relative to a triangulation employs a *left test* which locates p relative to a directed arc:

$$p \text{ left } p_i \rightarrow p_j \Leftrightarrow det(p, p_i, p_j) \geq 0.$$

Note that $det(p, p_i, p_j) = \langle p, p_i \times p_j \rangle \geq 0$ if and only if p lies in the left hemisphere as viewed from p_i toward p_j.

The swap test applied to an interior arc $p_i \rightarrow p_j$ is a local application of the circumcircle criterion: adjacent triangles (p_i, p_j, p_k) and (p_j, p_i, p_l) are replaced by triangles (p_k, p_l, p_j) and (p_l, p_k, p_i) if and only if

$$det(p_j - p_i, p_k - p_i, p_l - p_i) > 0,$$

or equivalently, p_l lies above (on the side opposite the origin) the plane of p_i, p_j, and p_k and is therefore interior to the circumcircle of (p_i, p_j, p_k). As in the case of the reverse swap test (Theorem 3.6), the swap test and the triangles resulting from a swap are well defined, i.e., a swap is only applied to a convex quadrilateral, and the new triangles satisfy the circumcircle criterion with respect to the fourth vertex. An alternative way of viewing the Delaunay triangulation is as follows: the planar triangle corresponding to each spherical triangle lies on the boundary of the convex hull of the nodes (as elements of $\mathbf{R}^3$). An anonymous referee pointed out that the triangulation can be efficiently constructed by a standard convex hull algorithm with the possible addition of an auxiliary point (in case the convex hull is not the entire sphere surface) and some postprocessing.

The floating-point computations required to implicitly extend the Delaunay triangulation to the entire surface when the nodes are contained in a single hemisphere are the distance computation,

$$d(p, q) = arccos\langle p, q \rangle,$$

and the circumcenter of triangle (p_i, p_j, p_k),

$$v_{ijk} = \frac{(p_j - p_i) \times (p_k - p_i)}{\|(p_j - p_i) \times (p_k - p_i)\|}.$$

Although not necessary for constructing the Delaunay triangulation or Voronoi diagram, many applications require that a surface point p be located relative to a polygonal region R of U (such as a tectonic plate in the application described in Section 1). Let the boundary of R be defined by a cyclically ordered sequence of vertices which form a simple closed curve. Our procedure consists of selecting a point q interior to R (by a heuristic method that attempts to avoid ill-conditioning) and traversing the boundary sequence to find all intersections of arc $p \rightarrow q$ with the boundary. Then p is contained in R if and only if the number of intersections is even.

Let $n = p \times q$ be the (nonzero) normal to the plane of p and q, and let $v_1 \rightarrow v_2$ denote a segment of the boundary curve such that v_1 and v_2 are on opposite sides of the plane (so that $v_1 \rightarrow v_2$ intersects the great circle containing p and q). The point of intersection r must be computed to determine if it lies on arc $p \rightarrow q$:

$$r = \frac{v_1 + t(v_2 - v_1)}{\|v_1 + t(v_2 - v_1)\|} \text{ for } t = \frac{\langle n, v_1\rangle}{\langle n, v_1 - v_2\rangle}.$$

Since $\langle n, v_1\rangle$ and $\langle n, v_2\rangle$ have opposite signs (or one of them is zero), t is well defined and lies in the interval $[0,1]$. Hence r is a normalized convex combination of v_1 and v_2 (assumed to be linearly independent) and is orthogonal to n, i.e., r is the required intersection point. Finally, r lies in the interior of arc $p \rightarrow q$ if and only if $\langle r, n \times p\rangle$ and $\langle r, q \times n\rangle$ are both positive.

Note that all of the floating-point computations involve vector cross products and dot products for which Cartesian coordinates are required. It is thus convenient to store surface points as triples of Cartesian coordinates rather than spherical coordinates or latitude/longitude pairs.

4.2 An $O(N\log N)$ Point-Location Method

It has been shown that, regardless of the nodal distribution, if the nodes are inserted in random order, the expected number of structural changes (swaps and swap tests) is $O(N)$ [Guibas et al. 1992]. The time complexity of an incremental algorithm is therefore determined by the cost of locating each new node relative to the triangulation of the previously added nodes. Guibas et al. describe an $O(N\log N)$ algorithm based on a tree structure containing all the intermediate triangulations along with links between overlapping triangles. A new node is located by tracing, in the order of their creation, all triangles that contain it [Guibas et al. 1992]. Barber et al. [1997] present an off-line algorithm (for constructing convex hulls and Delaunay triangulations of nodes in $\mathbf{R}^d$) based on an alternative data structure in which the unprocessed nodes are partitioned into subsets associated with triangles. We take a similar approach, but instead of storing the containing triangle of each unprocessed node, we store the nearest triangulation node. This results in both less storage and improved efficiency.

Since our search procedure begins with an arbitrarily specified triangulation node, we obtain constant search time by maintaining the nearest triangulation node to each unprocessed node. The data structure consists of three length-N arrays: `NEAR`, `NEXT`, and `DIST`. For each unprocessed node `I`, the index of the nearest triangulation node and its distance are stored in `NEAR(I)` and `DIST(I)`. For each triangulation node `J`, the set of unprocessed nodes in `J`'s Voronoi region (those having `J` as nearest triangulation node) are stored as a linked list in `NEAR` and `NEXT`. The linked list uses the nodal indexes as links and includes a zero terminator. The sequence is thus `I1 = NEAR(J)`, `I2 = NEXT(I1)`, `I3 = NEXT(I2)`,..., `0 = NEXT(Ii)`. The triangulation and nearest-node data structures are initialized with a triangulation consisting of a single triangle, and the remaining $N - 3$ nodes are added one at a time. Following the insertion of node `K`, only the unprocessed nodes associated with a neighbor `J` of node `K` need to be considered for a move from J's set to K's set of unprocessed nodes (since the nearest node to K is a neighbor of K in the Delaunay triangulation). Since the $N - k$ unprocessed nodes are distributed over k sets, the average set size is $(N - k) / k$, and the expected total time complexity is

$$O\left(\sum_{k=1}^{N} \frac{N - k}{k}\right) = O\left(N \sum_{k=1}^{N} \frac{1}{k} - N\right) = O(N \log N).$$

Since only relative distances between nodes are needed, we obtain an improvement in efficiency (a smaller constant in the $N\log N$ term) by using an increasing function of distance: the distance α between nodes p_i and p_j is replaced by $-\cos(\alpha) = -\langle p_i, p_j \rangle$.

In order to empirically verify the expected time complexity, we tabulated triangulation times $T(N)$ for N randomly generated nodes with $N = 5000, 10{,}000, 15{,}000, \ldots, 80{,}000$, and we plotted the sequence of 16 points $(N, T(N) / (N\log N))$. The sequence was slowly increasing but clearly asymptotic to a constant. The tests were run on a 200MHz Pentium Pro processor using the Lahey F77L3 Fortran compiler (with a 32-bit DOS extender). The time required for $N = 80{,}000$ was 9.26 seconds. (The same test on a 133MHz Pentium machine required 28.4 seconds.) The nodes were obtained by using a pseudorandom-number generator to get points uniformly distributed in the cube $[-1,1]^3$ and then normalizing the points to unit vectors.

5. SOFTWARE

The software package is described in the following four subsections: Usage, Data Structures, Organization of the Code, and Subprogram Descriptions.

5.1 Usage

A driver named STRITEST is provided with the package. That program reserves storage, reads a set of nodal coordinates, calls subroutine TRMESH to construct the Delaunay triangulation and subroutine CRLIST to construct the Voronoi diagram, and prints the data structures. The call to CRLIST, along with the statements that reserve storage for the Voronoi diagram, may be removed if only the triangulation is needed.

All information needed to use the package is contained in the header comments in the source code. Subroutine TRMESH includes brief descriptions of the additional subprograms that a user might wish to call directly.

5.2 Data Structures

The triangulation data structure is essentially identical to that used by the planar triangulation package TRIPACK [Renka 1996]. It consists of the number of nodes N, type REAL arrays X, Y, and Z of length N containing the Cartesian coordinates of the nodes, and a linked list requiring approximately $13N$ storage locations and which contains the *adjacency list* (ordered sequence of neighbors) for each node:

`LIST`	Set of nodal indexes which, along with `LPTR`, `LEND`, and `LNEW`, define the triangulation as a set of N adjacency lists: counterclockwise-ordered sequences of neighboring nodes such that the first and last neighbors of a boundary node are boundary nodes (the first neighbor of an interior node is arbitrary). In order to distinguish between interior and boundary nodes, the last neighbor of each boundary node is represented by the negative of its index.
`LPTR`	Set of pointers (`LIST` indexes) in one-to-one correspondence with the elements of `LIST`. `LIST(LPTR(I))` indexes the node that follows `LIST(I)` in cyclical counterclockwise order (the first neighbor follows the last neighbor).
`LEND`	Set of pointers to adjacency lists. `LEND(K)` points to the last neighbor of node `K` for `K` $= 1$ to N. Thus, `LIST(LEND(K))` < 0 if and only if `K` is a boundary node.
`LNEW`	Pointer to the first empty location in `LIST` and `LPTR` (list length plus one).

Since `LIST` and `LPTR` contain two entries for each triangulation arc (each endpoint stored as a neighbor of the other), their length is $L = 2N_a$, and more generally

$$N_t = \left\{ \begin{array}{ll} 2N - 4 & \text{if } N_b = 0 \\ 2N - N_b - 2 & \text{if } N_b \geq 3 \end{array} \right\},$$

$$N_a = \left\{ \begin{array}{ll} 3N - 6 & \text{if } N_b = 0 \\ 3N - N_b - 3 & \text{if } N_b \geq 3 \end{array} \right\},$$

$$L = \left\{ \begin{array}{ll} 6N - 12 & \text{if } N_b = 0 \\ 6N - 2N_b - 6 & \text{if } N_b \geq 3 \end{array} \right\},$$

where

$N =$	Number of nodes,
$N_b =$	Number of boundary nodes,
$N_t =$	Number of triangles,
$N_a =$	Number of arcs, and
$L =$	Length of `LIST`/`LPTR`.

Thus $L \leq 6N - 12$ and the storage required for the linked list is less than $13N$.

The Voronoi diagram is obtained by implicitly extending the triangulation to the entire surface (constructing a triangulation of the set of boundary nodes) if $N_b \neq 0$, computing the $2N - 4$ triangle circumcenters (Voronoi vertices) and circumradii, and storing a set of adjacency lists similar to `LIST`, but containing triangle indexes instead of nodal indexes. Subroutine `CRLIST` takes the triangulation data structure as input and returns the following arrays:

`XC, YC, ZC`	Type `REAL` arrays of length $2N - 4$ containing the Cartesian coordinates of the triangle circumcenters (including those associated with pseudotriangles as the first $N_b - 2$ entries).
`RC`	Type `REAL` array of length $2N - 4$ containing circumradii in one-to-one correspondence with circumcenters.
`LTRI`	Type `INTEGER` work space array of length $6N_b - 12$ if $N_b \neq 0$. `LTRI` is used to store the triangulation of the set of boundary nodes as a 6-by-($N_B - 2$) triangle list, each column of which represents a triangle by its vertex indexes and neighboring triangle indexes. This is used internally to construct `LISTC`.
`LISTC`	Array containing triangle indexes (indexes to `XC`, `YC`, `ZC`, and `RC`) stored in one-to-one correspondence with `LIST/LPTR` entries (or entries that would be stored in `LIST` for the extended triangulation): the index of triangle (p_i, p_j, p_k) is stored in `LISTC(K)`, `LISTC(L)`, and `LISTC(M)`, where `LIST(K)`, `LIST(L)`, and `LIST(M)` are the indexes of p_j as a neighbor of p_i, p_k as a neighbor of p_j, and p_i as a neighbor of p_k. The Voronoi region associated with a node is defined by the counterclockwise-ordered sequence of circumcenters in one-to-one correspondence with its adjacency list (in the extended triangulation).
`LPTR, LEND, LNEW`	Arrays of pointers updated for the addition of pseudotriangles if the original triangulation contains boundary nodes ($N_b \neq 0$). If $N_b \neq 0$ and both the triangulation and Voronoi diagram are needed, copies of `LPTR`, `LEND`, and `LNEW` must be saved before calling `CRLIST`.

5.3 Organization of the Code

The code is written in 1977 ANSI Standard Fortran. All variable and array names conform to the default typing convention: `I-N` for type `INTEGER` and

A–H or O–Z for type REAL. (There are no DOUBLE PRECISION variables.) There are 35 modules, and they are ordered alphabetically in the package. (The term module here refers to a subprogram rather than a Fortran 90 module.) Each consists of the following sections:

—the module name and parameter list with spaces separating the parameters into one to three subsets: input parameters, I/O parameters, and output parameters (in that order);

—type statements in which all parameters are typed and in which arrays are dimensioned;

—a heading with the name of the package, identification of the author, and date of the most recent modification to the source code;

—a description of the module's purpose and other relevant information for the user;

—input parameter descriptions and output parameter descriptions in the same order as the parameter list;

—a list of other modules required (called either directly or indirectly);

—a list of intrinsic functions called, if any; and

—the code, beginning with type statements and descriptions of all local variables.

5.4 Subprogram Descriptions

The package is comprised of the following modules. An asterisk appears before each module that the calling program (user) is expected to call directly. However, some of these modules are also called by other modules. Also, subprograms AREAS, CIRCLE, INSIDE, INTRSC, IRAND, PROJCT, SCOORD, STORE, and TRANS have applications independent of the triangulation package.

* ADDNOD	Subroutine which updates the triangulation data structure with the addition of a new node.
* AREAS	Real function which returns the area of a spherical triangle on the unit sphere.
BDYADD	Subroutine called by ADDNOD to add a boundary node.
* BNODES	Subroutine which returns an array containing the indexes of the boundary nodes in counterclockwise order. Counts of boundary nodes, triangles, and arcs are also returned.
* CIRCLE	Subroutine which returns the Cartesian coordinates of a sequence of uniformly spaced points on the unit circle centered at (0,0).

* CIRCUM	Subroutine which computes the circumcenter of a spherical triangle defined by user-specified vertices on the unit sphere.
COVSPH	Subroutine called by ADDNOD to connect a new node to all boundary nodes when all boundary nodes are visible from the new node. The nodal addition thus extends the convex hull of the nodes to the entire surface.
* CRLIST	Subroutine which constructs a Voronoi diagram from a Delaunay triangulation.
* DELARC	Subroutine which removes a boundary arc from a triangulation resulting in a nonconvex triangulation.
DELNB	Subroutine called by DELARC and DELNOD to delete a neighbor from an adjacency list.
* DELNOD	Subroutine which deletes a node from a triangulation, preserving the circumcircle property.
* EDGE	Subroutine which forces a pair of nodes to be adjacent by applying any necessary swaps. A sequence of calls to EDGE may be used to force the presence of arcs defining the boundary of a nonconvex and/or multiply connected region, or to introduce barriers, resulting in a constrained Delaunay triangulation. Note, however, that subsequent nodal additions may remove some of the arcs.
* GETNP	Subroutine which determines an ordered sequence of closest nodes to a given node, along with the associated distances.
INSERT	Subroutine called by BDYADD, COVSPH, and INTADD to insert a neighbor into an adjacency list.
* INSIDE	Logical function which locates a point relative to a polygonal region on the surface of the unit sphere.
INTADD	Subroutine called by ADDNOD to add an interior node.
* INTRSC	Subroutine called by INSIDE to find a point of intersection between a pair of great circles.
* IRAND	Integer function which returns a uniformly distributed pseudorandom integer. This is used by TRFIND to select a starting node for the search in the unlikely event that the default or previously selected starting node results in failure.
LEFT	Logical function called by DELNOD, EDGE, and TRMESH to locate a point relative to a directed geodesic (which defines a left and right hemisphere).
LSTPTR	Integer function called by ADDNOD, CRLIST, DELARC, DELNOD, INTADD, NEARND, SWAP, and TRFIND to determine the LIST index of a specified neighbor of a specified node.

NBCNT	Integer function called by DELNOD to determine the number of neighbors of a specified node.
* NEARND	Integer function which returns the index of the nearest node to an arbitrary point, along with its angular distance.
OPTIM	Subroutine called by DELNOD and EDGE to optimize a portion of a triangulation defined as the triangles that share a set of specified interior arcs.
* PROJCT	Subroutine which applies a depth-perspective transformation to a point in 3-space.
* SCOORD	Subroutine which converts a point from Cartesian coordinates to spherical coordinates.
* STORE	Real function used by TRFIND in computing the machine precision. It forces a value to be stored in main memory so that the precision of floating-point numbers in memory locations rather than registers is computed.
SWAP	Subroutine called by ADDNOD, DELNOD, EDGE, and OPTIM to swap diagonal arcs in a convex quadrilateral defined by a pair of adjacent triangles.
SWPTST	Logical function called by ADDNOD, CRLIST, and OPTIM to apply the swap test (local circumcircle test) to an interior arc.
* TRANS	Subroutine which transforms spherical coordinates into Cartesian coordinates on the unit sphere.
* TRFIND	Subroutine called by ADDNOD and NEARND to locate a point relative to the triangulation.
* TRLIST	Subroutine which converts the linked list triangulation data structure to a triangle list.
* TRLPRT	Subroutine which prints the triangle list created by Subroutine TRLIST.
* TRMESH	Subroutine which creates a Delaunay triangulation.
* TRPLOT	Subroutine which, when linked to a user-supplied graphics package, creates a graphical display of a triangulation.
* TRPRNT	Subroutine which prints the triangulation data structure.

REFERENCES

AUGENBAUM, J. M. AND PESKIN, C. S. 1985. On the construction of the Voronoi mesh on a sphere. *J. Comput. Phys. 59*, 177–192.

BARBER, C. B., DOBKIN, D. P., AND HUHDANPAA, H. 1996. The Quickhull algorithm for convex hulls. *ACM Trans. Math. Softw. 22*, 4 (Dec.), 469–483.

CLINE, A. K. AND RENKA, R. J. 1984. A storage-efficient method for construction of a Thiessen triangulation. *Rocky Mt. J. Math. 14*, 1, 119–139.

CLINE, A. K. AND RENKA, R. J. 1990. A constrained two-dimensional triangulation and the solution of closest node problems in the presence of barriers. *SIAM J. Numer. Anal. 27*, 5 (Oct.), 1305–1321.

DELAUNAY, B. 1934. Sur la sphère vide. *Bull. Acad. Sci. USSR 7*, 793–800.

FOHLMEISTER, J. F. AND RENKA, R. J. 1994. Hotspots, mantle convection and plate tectonics: A synthetic calculation. *Pure Appl. Geophys. 143*, 673–695.

FOHLMEISTER, J. F., RENKA, R. J., AND STOUT, J. H. 1990. Lithospheric plate motions predicted by a quantitative theory. *Trans. Am. Geophys. Union 71*, 43 (Oct.).

GUIBAS, L. J., KNUTH, D. E., AND SHARIR, M. 1992. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica 7*, 381–413.

LAWSON, C. L. 1977. Software for C^1 surface interpolation. In *Mathematical Software III,* J. R. Rice, Ed. Academic Press, Inc., Orlando, FL, 161–194.

LAWSON, C. L. 1984. C^1 surface interpolation for scattered data on a sphere. *Rocky Mt. J. Math. 14*, 1, 177–202.

PREPARATA, F. P. AND SHAMOS, M. I. 1985. *Computational Geometry: An Introduction*. Springer texts and monographs in computer science. Springer-Verlag New York, Inc., New York, NY.

RENKA, R. J. 1984a. Interpolation of data on the surface of a sphere. *ACM Trans. Math. Softw. 10*, 4 (Dec.), 417–436.

RENKA, R. J. 1984b. Algorithm 623: Interpolation on the surface of a sphere. *ACM Trans. Math. Softw. 10*, 4 (Dec.), 437–439.

RENKA, R. J. 1996. Algorithm 751: TRIPACK: Constrained two-dimensional Delaunay triangulation package. *ACM Trans. Math. Softw. 22*, 1 (Mar.), 1–8.

RENKA, R. J. 1997. Algorithm 773: SSRFPACK: Interpolation of scattered data on the surface of a sphere with a surface under tension. *ACM Trans. Math. Softw. 23*, 3 (Sept.). This issue.

RHYNSBURGER, D. 1973. Analytic delineation of Thiessen polygons. *Geograph. Anal. 5*, 133–144.

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

VORONOI, G. 1908. Nouvelles applications des parametres continuis à la theorie des formes quadratiques: Deuxième mémorie: Recherches sur les paralléloèdres primitifs. *J. Reine Angew Math. 134*, 198–287.

Received: July 1995; revised: August 1996; accepted: March 1997