TEXAS STATE UNIVERSITY
SAN MARCOS

Department of Computer Science
San Marcos, TX 78666

Report Number TXSTATE-CS-TR-1995-1

Facet-Based Tetrahedralization Software

Carol Hazlewood

1995-04-12

Facet-Based Tetrahedralization Software

Carol Hazlewood
ch04@swt.edu
Department of Computer Science,
Southwest Texas State University
San Marcos, TX 78666, USA

12 April 1995

Abstract

Software for computing Delaunay tetrahedralizations is described. The software is designed to be used as part of a larger system and has a simple user interface. Numerically stable Householder transformations are used to implement orientation and insphere tests in floating-point arithmetic. A backward error analysis is done for the orientation test.

keywords: geometry software, Delaunay tetrahedralization, Householder transformations, floating point arithmetic

1 Introduction

A Delaunay tetrahedralization is a decomposition of the convex hull of a finite set of points in three-dimensional Euclidean space into tetrahedra. It has an abundance of applications, including surface interpolation and finite-element mesh generation. We describe an implementation of a facet-based algorithm for computing Delaunay tetrahedralizations.

The software uses a simple algorithm that never deletes tetrahedra once they are discovered. Using a modification of Dwyer's[1] algorithm, the software starts with a triangular face of the convex hull and constructs a sequence of tetrahedra by repeatedly finding an apex for an existing Delaunay facet. The apex calculation uses the orientation and insphere predicates.

The software is designed to be used as part of a larger system and has a well-defined user interface, where all communication is through the parameter list and no input or output is produced by the software. The tetrahedralization is represented by two arrays containing vertex and adjacency information. Numerically stable Householder transformations are used in the floating-point implementations of the orientation and insphere tests and the giftwrapping primitive. The software was recently used by Frank Ray, a research scientist at McDonald Observatory, University of Texas at Austin, in the interpolation of thermal volumetric data in steel telescope primary mirror structures[2]. C and Fortran code are available from the author.

A Delaunay tetrahedralization can be computed directly in an incremental fashion (Watson [3], Field[4], Cavendish et al,[5] and Rajan[6]). These methods begin with an initial tetrahedralization of a few points and update the existing tetrahedralization as points are added. Dwyer[1] presents a linear expected time analysis of a facet-based technique for points uniformly distributed in a unit d-ball.

Cline and Renka[7] and Fortune[8] describe implementations of incremental two-dimensional Delaunay triangulation algorithms. Fortune[9] uses a sweep-line technique for computing Delaunay triangulations. Renka's incremental code[10] and Fortune's sweepline code are available from netlib. GEOMPACK by Barry Joe[11] contains code for computing tetrahedralizations.

Many people have studied the problems of implementing geometric computations. Fortune and Van Wyk[12] develop efficient, exact arithmetic for geometric calculations. Edelsbrunner and Mücke[13] and Yap[14] perturb objects to avoid degeneracies.

Ottmann, Thiemt, and Ullrich[15], and Dobkin and Silver[16] use more than one kind of arithmetic to achieve numerical stability for geometric computations. Hoffman, Hopcroft, and Karasick[17] maintain exact incidence information and approximate numerical information about objects to achieve robust computations. Li and Milenkovic use rounded arithmetic to approximate curve arrangements[18] and convex hulls.[19] Guibas, Salesin, and Stolfi[20] use interval arithmetic and backward error analysis to produce the exact answer to a perturbed problem as well as a bound on the size of the perturbation. Fortune[21] analyzes floating point error for basic geometric calculations that are used in computing triangulations. Field[4] details a case study on the problems encountered in implementing Watson's algorithm for triangulations [3]. Hoffmann[22] compares several approaches and calls for more research on the topic of geometric computation.

In Section 2 the basic algorithm is presented; in Section 3 the implementation of floating point functions using Householder transformations is described; in Section 4 experimental results are reported.

2 Facet-Based Tetrahedralization

The three vertices of a triangle uniquely determine a plane. The plane partitions E^3 into two open half spaces and the plane itself. The union of the plane and an open half space is a closed half space. The triangle has two sides, one corresponding to each of the half spaces.

If a triangle is a facet of a tetrahedron, the tehrahedron is in exactly one of the closed half spaces defined by the facet, or on one side of the facet. The tetrahedron vertex that is not in the facet is called the apex of the facet.

```
pre:  $P$ is set of $n$ points in $E^3$
post: $T$ is a Delaunay tetrahedralization of $P$

$tess(P,T)$

  $T \leftarrow \emptyset$;                    // known tetrahedra
  $g \leftarrow initial\_facet$;               // in boundary
  $Q \leftarrow g$;                            // pool of facets
  while ($Q \neq \emptyset$) do
    $f \leftarrow front(Q)$;                   // get an open facet
    $a \leftarrow getapex(f)$;                 // find its apex
    $t \leftarrow conv(f \cup \{a\})$;         // new tetrahedron
    $T \leftarrow T \cup \{t\}$;
    for $g$ a facet of $t$ do
      if ($g \notin Q$)
        then $Q \leftarrow Q \cup \{g\}$;      // g is open
        else $Q \leftarrow Q - \{g\}$;         // g now closed
```

Figure 1: Facet-Based Delaunay Tetrahedralization

Lemma 1 provides a basis for the algorithm shown in Figure 1

Lemma 1 *In a tetrahedralization of P, every facet of a tetrahedron is either a facet of exactly two tetrahedra, or in the boundary of the convex hull of P and a facet of exactly one tetrahedron.*

As illustrated in Figure 2, the algorithm first finds a facet g of a tetrahedron s in T that lies in the boundary of the convex hull of P. All the points of P are on one side of g. Then the apex c of g in s is found among the points of P. Let f be a facet of s that contains c, let b be the apex of f in s, and let A, be the set of points of P that are on the opposite side of f from b. If A is empty, f is in the boundary and all of its tetrahedra have been found. Otherwise, there is another tetrahedron t that contains f, and its apex a must be found. The facets of t that contain a will be explored, in turn, to see if they form part of the boundary or yield yet another tetrahedron. As the process continues, the construction may wrap back on itself so that tetrahedron t has already been found and no construction is necessary. The algorithm terminates when no unexplored facets are left.

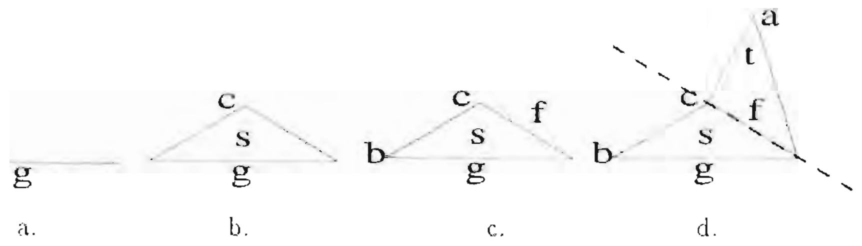


Figure 2: Basic Tetrahedron Construction

Since a Delaunay tetrahedralization induces a Delaunay triangulation on the facets of the convex hull, an initial facet can be obtained by computing a convex hull facet, and, if necessary, a Delaunay triangulation of the facet. The giftwrapping algorithm [23] provides a convenient way to compute a single convex hull facet.

Although every facet in a Delaunay tetrahedralization of P has the property stated in Lemma 1, while the algorithm is executing, the information we have at hand is incomplete. Each facet falls into one of three categories:

- unknown: the facet belongs to no tetrahedron (facet g in Figure 2a)
- open: the facet belongs to one known tetrahedron and no boundary information is known, or the facet belongs to no known tetrahedra and is known to be in the boundary (initial facet only) (facet f in Figure 2c, facet g in Figure 2a)
- closed: the facet belongs to two known tetrahedra, or the facet belongs to one known tetrahedron and is known to be in the boundary (facets f and g in Figure 2d)

Open facets are kept in a queue, which is implemented as a hash table for faster searching.

Apex construction depends on the following properties of intersecting balls.

Lemma 2 *If two balls B and C intersect, their intersection is a disk D. The disk determines a plane and open half spaces H^+ and H^-. Let $B^\pm = B \cap H^\pm$ and $C^\pm = C \cap H^\pm$. Then exactly one of the assertions $B^+ \subset C^+$ and $C^+ \subset B^+$ is true, and the inclusion is reversed in H^-. That is, if $B^+ \subset C^+$, then $C^- \subset B^-$, and if $C^+ \subset B^+$, then $B^- \subset C^-$.*

Lemma 3 *Let f be a triangle in E^3, H^+ an open halfspace determined by f, and let $\mathcal{L}$ be the line through the circumcenter of and normal to f. For point a in H^+, let C_a be the ball defined by a and the vertices of f with center c_a and radius r_a and let $C_a^+ = C_a \cap H^+$. Impose a one-dimensional coordinate system on $\mathcal{L}$ with 0 in the plane of f and positive direction in H^+, and let L_a be the $\mathcal{L}$-coordinate of c_a.*

If a and b are points in H^+, then $L_a < L_b$ if and only if $C_a^+ \subset C_b^+$

Proof: Let $\ell_a^\pm$ ($\ell_b^\pm$) be the intersection of the boundary of C_a (C_b), line $\mathcal{L}$, and halfspace $H^\pm$. By Lemma 2 it suffices to show that ℓ_a^+ is closer than ℓ_b^+ to f, or that ℓ_b^- is closer than ℓ_a^- to f. The distance from $\ell_a^\pm$ to f is $r_a \pm \ell_a$, and the distance from $\ell_b^\pm$ to f is $r_b \pm L_b$. If $r_a < r_b$, then $r_a + L_a < r_b + L_b$, so ℓ_a^+ is closer than ℓ_b^+ to f . If $r_b < r_a$, then $r_b - L_b < r_a - L_a$, so ℓ_b^- is closer than ℓ_a^- to f . □

This result yields in a straightforward manner a characterization of the apex of a facet in a Delaunay tetrahedralization.

Theorem 1 *Let f be a facet in a Delaunay tetrahedralization, and let P^+ be the subset of P on one side of f. If P^+ is empty, then f is a boundary facet. Otherwise a point of P^+ that generates the center of minimum $\mathcal{L}$-coordinate is the apex of facet f in a Delaunay tetrahedralization.*

Theorem 2 *Algorithm tess shown in Figure 1 produces a Delaunay tetrahedralization of P in $O(n^2)$ space and $O(n^3)$ time in the worst case.*

Proof: By Theorem 1 each tetrahedron producd by algorithm *tess* is a Delaunay tetrahedron. These tetrahedra intersect in mutual faces and have circumballs, the interiors of which are free of points of P. If the convex hull of P is not covered by these tetrahedra, by Lemma 1 there will be at least one facet interior to the convex hull of P that belongs to exactly one tetrahedron. This is impossible, since open facets remain in the facet pool and prevent termination of the algorithm, so the convex hull of P is covered by tetrahedra and that P is the set of vertices of tetrahedra.

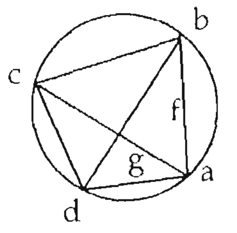
Since no facet is added to the pool more than once, and a facet is removed in each iteration, the while loop, and therefore the algorithm, terminate.

Construction of the initial facet requires $O(n)$ time. Finding the apex of a single tetrahedron requires $O(n)$ time. The number of Delaunay tetrahedra in a tetrahedralization of n points can be as great as $O(n^2)$ [24]. □

For many point sets the Delaunay tetrahedralization has size $O(n)$.

Corollary 1 *If the size of the tetrahedralization is $O(n)$, algorithm tess requires $O(n)$ space and $O(n^2)$ time.*

A Delaunay tetrahedralization is unique unless five or more points of P lie on the circumball of a tetrahedron. The convex hull of the points on the circumball define a unique polytope, any tetrahedralization of which is a Delaunay tetrahedralization. It is possible for several points to generate the minimum $\mathcal{L}$-coordinate, and therefore lie on a common sphere. Any of the points can be used as the apex, since this is exactly the situation where the Delaunay tetrahedralization is not unique. However, our algorithm can cover the convex hull with multiple sets of tetrahedra. The two-dimensional example in Figure 3 illustrates the problem. In the two-dimensional case, facets are edges of triangles. Eventually $\triangle abc$, $\triangle acd$, $\triangle abd$, and

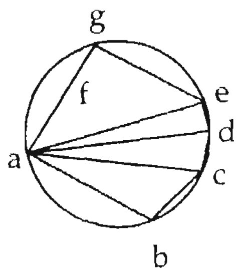


open facets in the queue	$Q = \{\overline{ab}, \overline{ad}\}$
c is found as an apex for $\overline{ab}$	$Q = \{\overline{ad}, \overline{bc}, \overline{ac}\}$
b is found as an apex for $\overline{ad}$	$Q = \{\overline{bc}, \overline{ac}, \overline{ab}, \overline{bd}\}$

Figure 3: Problem with Cospherical Points

$\triangle bcd$ will be added to the tetrahedralization. The convex hull of the cospherical points is covered with two sets of triangles, and facet $\overline{ab}$ is incorrectly added to the queue for a second time.

This problem can be avoided by enforcing some consistent method for choosing an apex. One method is illustrated in Figure 4. Let A be the set of points that generate the minimum $\mathcal{L}$-coordinate with respect to facet f, and let $B = A \cup \{\text{vertices of } f\}$. Let a be the point of minimum index in P among the points of B. We arbitrarily choose the apex so that point a is a vertex of every tetrahedron with vertices in B. If a is



$A = \{a,b,c,d\}$	$A = \{b,c,d,e\}$
$B = \{a,b,c,d,e,g\}$	$B = \{a,b,c,d,e,g\}$
$a \notin f$ so choose a	$a \in f$ so choose e

Figure 4: Solution to Cospherical Points

in A, choose a as the apex of f; if a is a vertex of f, choose the point of minimum index that giftwraps the edge of f that does not contain a.

Further investigation is needed to see if other selection criteria may be more desirable.

3 Using Householder Transformations

Geometric problems are often poorly conditioned in the sense that a small change in problem parameters may result in a large difference in the answer [25]. As an example, consider the computation which, given a line and a point, returns the value *true* if the point is on the line and *false* otherwise. A slight perturbation of the coordinates of the point or the coefficients of the equation of the line may change the result from false to true.

Floating point computations in our algorithm are confined to a variant of the orientation test, in which we test if two points lie on opposite sides of a plane (opp test); a variant of the insphere test, namely the $\mathcal{L}$-coordinate calculation for determining an apex of a facet; and the giftwrapping primitive. Here we present an implementation of the opp test and apex computation using numerically stable Householder transformations (Stewart [26], Golub and Van Loan [27]), and based on a development by Alan Cline [28]. The giftwrapping primitive is similar to the apex computation.

Householder transformations, also known as elementary reflectors or elementary Hermitian matrices, have the form $Q = I - 2uu^T$, where I is the identity matrix and $u^Tu = 1$, and are numerically stable, orthogonal ($Q^TQ = I$), symmetric, and isometric (for any vector y, $||y|| = ||Qy||$) in the Euclidean norm. Geometrically, when Q is applied to a vector v, v is reflected in the plane normal to u.

3.1 Opp Test

Given points z_1 and z_2, and a facet with non-collinear vertices p_1, p_2, and p_3, the opp test returns the value *true* if z_1 and z_2 are on the same side of the plane defined by the facet and *false* otherwise. Let P be the matrix $\left[\begin{array}{cc} p_2 - p_1 & p_3 - p_1 \end{array}\right]$, and compute the Householder QR factorization [27], [26] to obtain $P = Q_1Q_2R$, where Q_1, Q_2 are Householder transformations and R is upper triangular. If $v = Q_1Q_2e_3$, where $e_3 = (0,0,1)^T$, then v has unit length, since Q_1 and Q_2 are isometric. In addition,

$$P^Tv = R^TQ_2Q_1v = R^Te_3 = 0,$$

since the third row of R is 0. That is, v is a unit normal to the facet. Therefore, points z_1 and z_2 are on the same side of the plane defined by p_1, p_2, and p_3 if and only if the signs of $(z_1 - p_1)^Tv$ and $(z_2 - p_1)^Tv$ are the same.

For backward error analysis of the opp test, we want to show that the computed answer is the exact answer to a slightly perturbed problem. The opp test requires the following computations: compute Q_1; compute Q_2; $v \leftarrow Q_1 Q_2 e_3$; compare $(z_1 - p_1)^T v$ and $(z_2 - p_1)^T v$. Following Fortune,[21] we use approximate arithmetic, or floating-point arithmetic without underflow or overflow, with error bound ϵ_0 on a single arithmetic operation. The comparison can be done exactly, so we need only analyze a computation of the form $(z - p)^T Q_1 Q_2 e_3$.

We first examine separately the errors incurred by approximate arithmetic in computing dot products, vector difference, Householder transformations, and vector multiplication by a product of Householder transformations. Where there is a choice, we amortize the error uniformly over the appropriate values.

Consider the dot product computation $\sum_{i=1}^{n} a_i b_i$. Let s be the result computed in approximate arithmetic. Using standard techniques it can be shown that

$$s = \sum_{i=1}^{n} a_i(1+\theta_i) b_i(1+\theta_i),$$

where $|\theta_i| \le (n - i + 3)\epsilon_0/2$ for $i = 1, \ldots, n$. In this context, $n = 3$, so $|\theta_i| \le 5\epsilon_0/2$ for $i = 1, \ldots, 3$.

Let d be the difference between two vectors, $a - b$, computed in approximate arithmetic. From the bound on approximate arithmetic $d = a(1 + u_a) - b(1 + u_b)$, where the components of error vectors u_a and u_b have absolute value less than ϵ_0.

Suppose $\hat{Q}$ is a computed Householder transformation of order three. According to Golub and Van Loan,[27], if m is the computed result of multiplying 3-vector v by $\hat{Q}$, then $m = Q(v + e)$, where $||e|| \le 9c\epsilon_0||v||$, and c is a constant of order unity. Applying this to the computed value t of $Q_1 Q_2 x$, we have $t = Q_1(Q_2(x+f_2)+f_1)$, where $||f_2|| \le 9c\epsilon_0||x||$ and $||f_1|| \le 9c\epsilon_0||Q_2(x+f_2)|| \le 9c\epsilon_0||x+f_2|| \le 9c\epsilon_0||x||(1+9c\epsilon_0)$, since Q_2 is isometric. Finally, $t = Q_1(Q_2(x + f_2) + f_1) = Q_1 Q_2(x + h)$, where $h = f_2 + Q_2^T f_1$, and $||h|| \le 9c\epsilon_0||x||(2 + 9c\epsilon_0) \le 27c\epsilon_0||x||$.

Let r be the computed result of $(z-p)^T Q e_3$, where $Q = Q_1 Q_2$. Using the above results, $r = \sum_{i=1}^{3}[(z_i(1+u_z) - p_i(1+u_p))(1+\theta)] \cdot [(Q(e_3+f))_i(1+\theta)]$, where $|u_z| \le \epsilon_0$, $|u_p| \le \epsilon_0$, $|\theta| \le 5\epsilon_0/2$, and $||f|| \le 27c\epsilon_0||e_3|| = 27c\epsilon_0$.

Now $z_i(1 + u_z)(1 + \theta) = z_i(1 + \delta)$, and $p_i(1 + u_p)(1 + \theta) = p_i(1 + \delta)$, where $|\delta| \le 4\epsilon_0$.

Furthermore, $(Q(e_3 + f))_i(1 + \theta) = (Qe_3)i + (Qe_3)_i\theta + (Qf)_i + (Qf)_i\theta = (Qe_3)_i + d_i$, where $||d|| \le \epsilon_0 + 27c\epsilon_0 + 27c\epsilon_0^2 \le 29c\epsilon_0$. Reassembling vectors from components, $(Q(e_3 + f))(1 + \theta) = Q(e_3 + f) + d = Q(e_3 + f) + Qg = Q(e_3 + h)$, where $||h|| = ||f|| + ||g|| \le 27c\epsilon_0 + ||d|| \le 56c\epsilon_0$.

Finally,

$$r = \sum_{i=1}^{3}[z_i(1+\delta) - p_i(1+\delta)] \cdot [(Q(e_3 + h))_i],$$

where $|\delta| \le 4\epsilon_0$ and $||h|| \le 56c\epsilon_0$. That is, the computed value of the opp test is the exact value of a slightly perturbed problem.

3.2 $\mathcal{L}$-coordinates

As discussed previously, the apex of a facet is the point that generates a center of minimum $\mathcal{L}$-coordinate among all the points on one side of the facet, say in H^+. If we fix a facet with three non-collinear vertices p_1, p_2, and p_3, then we can define a function L that associates a point p in P^+ with the $\mathcal{L}$-coordinate of the center of the circumball defined by p, p_1, p_2, and p_3. The function L can also be implemented with Householder transformations. We compute v, a unit normal to the facet, c, the circumcenter of the facet, and finally the $\mathcal{L}$-coordinate of the center of the ball. In practice v and c are computed once for each facet. Factor $P = \begin{bmatrix} p_2 - p_1 & p_3 - p_1 \end{bmatrix}$, so $P = Q_1 Q_2 R$, where Q_1, Q_2 are Householder transformations and R is upper triangular, and let $Q = Q_1 Q_2$. As in the opp test, the unit normal is $Q_1 Q_2 e_3 = Qe_3$. Since c is in the plane of the facet, $c - p_1$ can be written as Qy, where the third component of y is 0. Then

$$||c - p_1|| = ||Qy|| = ||y||,$$

since Q is isometric. For $j = 1, 2$, the j-th column of P can be written as $p_{j+1} - p_1 = Pe_j = QRe_j = Qr_j$, where r_j is the j-th column of R. Therefore,

$$||c - p_{j+1}|| = ||(c - p_1) - (p_{j+1} - p_1)|| = ||Qy - Qr_j|| = ||y - r_j||. \tag{1}$$

Since c is the circumcenter of the facet, it is equidistant from the vertices, and, for $j = 1, 2$, $||y|| = ||y - r_j||$. Since R is upper triangular, when $j = 1$,

$$y_1^2 + y_2^2 = (y_1 - r_{11})^2 + y_2^2,$$

yielding

$$y_1 = r_{11}/2,$$

and, when $j = 2$,

$$y_1^2 + y_2^2 = (y_1 - r_{12})^2 + (y_2 - r_{22})^2,$$

yielding

$$y_2 = (r_{22} + (r_{12}/r_{22})(r_{12} - 2y_1))/2.$$

Now we are able to compute Qy to obtain c, the center of the facet

The center s of the ball can be written as $c + \mathcal{L}v$, where $\mathcal{L}$ is the $\mathcal{L}$-coordinate of s. Since s is equidistant from p_1 and p,

$$\begin{aligned} ||s - p_1|| &= ||s - p|| \\ ||(c + \mathcal{L}v) - p_1|| &= ||(c + \mathcal{L}v) - p|| \\ ||(c - p_1) + \mathcal{L}v|| &= ||(c - p_1) - (p - p_1) + \mathcal{L}v||. \end{aligned}$$

Writing $Q^T(p - p_1)$ as r, we have $Q^T(p - p_1) = r = Q^TQr$, and $(p - p_1) = Qr$. Recalling that $(c - p_1) = Qy$ and $v = Qe_3$, we get

$$||Qy + \mathcal{L}Qe_3|| = ||Qy + \mathcal{L}Qe_3 - Qr||,$$

and

$$||Q(y + \mathcal{L}e_3)|| = ||Q(y + \mathcal{L}e_3 + r)||.$$

Since Q is orthogonal,

$$||y + \mathcal{L}e_3|| = ||y + \mathcal{L}e_3 + r||.$$

Squaring both sides we get

$$(y + \mathcal{L}e_3)^T(y + \mathcal{L}e_3) = (y + \mathcal{L}e_3 - r)^T(y + \mathcal{L}e_3 - r),$$

which reduces to

$$0 = r^Tr - 2r^Ty - 2\mathcal{L}r^Te_3.$$

Hence

$$\mathcal{L} = \frac{||r||^2 - 2y^Tr}{2e_3^Tr}.$$

This can be simplified using

$$||r||^2 = ||p - p_1||^2 = (p - p_1)^T(p - p_1),$$

$$e_3^Tr = e_3^TQ^T(p - p_1) = (Qe_3)^T(p - p_1) = v^T(p - p_1),$$

and

$$y^Tr = y^TQ^T(p - p_1) = (Qy)^T(p - p_1) = (c - p_1)^T(p - p_1)$$

to the final form

$$\mathcal{L} = \frac{(p - p_1)^T(p - p_1) - 2(c - p_1)^T(p - p_1)}{v^T(p - p_1)}$$

4 Experimental Results

The insphere test was implemented with and without Basic Linear Algebra Subroutines (BLAS),[29] which abstract vector operations. The timings in Table 4 for the insphere test were done on one node of an nCUBE model 2E computer, which has a BLAS library available. The numbers represent average cost per function evaluation, amortizing the one-time setup over all trials.

trials	Householder no blas	Householder library blas	Householder fortran blas
10000	0.2547753E-04	0.4506951E-04	0.7514621E-04
20000	0.2530957E-04	0.4506079E-04	0.7512725E-04
30000	0.2342186E-04	0.4505925E-04	0.7512100E-04
40000	0.2423604E-04	0.4506010E-04	0.7511742E-04
50000	0.2363036E-04	0.4505702E-04	0.7511496E-04
60000	0.2439852E-04	0.4505553E-04	0.7511338E-04
70000	0.2416601E-04	0.4505539E-04	0.7511234E-04
80000	0.2408908E-04	0.4505610E-04	0.7511202E-04
90000	0.2411615E-04	0.4505465E-04	0.7511133E-04
100000	0.2421591E-04	0.4505658E-04	0.7511085E-04

BLAS Timings

Table 4 contains runtime data on one node of the nCUBE 2E for the insphere test implemented with Householder transformations without BLAS and with determinants. The numbers again represent average cost per function evaluation, amortizing the one-time setup over all trials. Set-up for the Householder version includes computing the transformations; for the determininants, the one-time cost is for computing the minors. The runtimes are about the same over a range of trial sizes.

To empirically test accuracy, we set up an insphere problem with coordinates less than two in absolute value with the origin on the sphere, then let the test point approach the origin. Tests were run with 32-bit arithmetic. The Householder version returned a result of zero for points within 10^{-7} of the origin,which is expected, since it uses differences between points: if point $p1$ has coordinates around size one, and $p2$ is within 10^{-7} of origin $p3$, then $p2 - p1$ and $p3 - p1$ will have the same value in 32-bit arithmetic. The determinants performed better than the Householder transformations in this case, giving correct signs for points within 10^{-14} of the origin.

As Fortune[21] observed, points that are close together but far from the origin pose numerical difficulties. We constructed a sequence of insphere problems by starting with a set of point coordinates less than two in absolute value and translating the same coordinate of each point by increasing powers of 10. The problems are solved using Householder transformations with and without BLAS and determinants using an Alpha running VMS with default compiler settings.

There is no difference in accuracy in these tests between the versions of the Householder implementation in 32-bit arithmetic. The Householder versions fail at 10^8 with a zero divide, which is the point at which the points become indistinguishable in this floating-point system. The determinant version fails at 10^4 with an incorrect result.

The test was also run in 64-bit arithmetic with no BLAS. The Householder version has a zero divide at 10^{16}, while determinants fail at 10^6 with an incorrect answer.

Acknowledgments

The nCUBE was partially funded by NSF grants USE-9052040 and DUE-9352798.

	Houscholder	Determinants
trials	no blas	
100	0.2443675E-04	0.2391187E-04
200	0.2612430E-04	0.2939204E-04
300	0.2310050E-04	0.2755631E-04
400	0.2431148E-04	0.2691151E-04
500	0.2335035E-04	0.2684309E-04
600	0.2452692E-04	0.2694426E-04
700	0.2419856E-04	0.2724844E-04
800	0.2406637E-04	0.2639701E-04
900	0.2413915E-04	0.2696446E-04
1000	0.2430852E-04	0.2656197E-04
10000	0.2547753E-04	0.2705126E-04
20000	0.2530957E-04	0.2686962E-04
30000	0.2342186E-04	0.2706156E-04
40000	0.2423604E-04	0.2579072E-04
50000	0.2363036E-04	0.2643391E-04
60000	0.2439852E-04	0.2699300E-04
70000	0.2416601E-04	0.2661089E-04
80000	0.2408908E-04	0.2642198E-04
90000	0.2411615E-04	0.2636195E-04
100000	0.2421591E-04	0.2639183E-04

Insphere Timings on nCUBE

References

[1] Rex A. Dwyer, "Higher Dimensional Voronoi Diagrams in Linear Expected Time", *Proceedings of the Fifth Annual Symposium on Computational Geometry*, ACM Press (1989) 326-333.

[2] F.B. Ray, *HET primary mirror thermal compensation I: mathematical foundation*, TR-941206, HET Technical Library, McDonald Observatory, The University of Texas at Austin, Austin, TX 78712.

[3] D.F. Watson, "Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes", *The Computer Journal* **24**:2 (1981) 167-172.

[4] David A. Field, "Implementing Watson's Algorithm in Three Dimensions", *Proceedings of the Second Annual Symposium on Computational Geometry*, ACM Press (1986) 246 259

[5] James C. Cavendish, David A. Field and William H. Frey, "An Approach to Automatic Three-Dimensional Finite Element Mesh Generation", *International Journal for Numerical Methods in Engineering* **21** (1985) 329-347.

[6] V.T. Rajan, "Optimality of the Delaunay triangulation in R^d", *Proceedings of the Seventh Annual Symposium on Computational Geometry* (1991) 357 363

[7] A.K. Cline and R.L. Renka, "A storage-efficient method for construction of a Thiessen triangulation", *The Rocky Mountain Journal of Mathematics* **14**:1 (1984) 119-139

[8] Steven Fortune, "Voronoi Diagrams and Delaunay Triangulations", *Computing in Euclidean Geometry*, Ding-Zhu Du and Frank Hwang, eds., World Scientific Press (1992) 193 233

[9] S. Fortune, "Sweepline Algorithms for Voronoi diagrams", *Algorithmica* **2** (1987) 153-174

[10] Robert J. Renka, "ALGORITHM 624 Triangulation and Interpolation at Arbitrarily Distributed Points in the Plane", *ACM Transactions on Mathematical Software* **10**:4 (1984) 440-442.

[11] B. Joe, "GEOMPACK - a software package for the generation of meshes using geometric algorithms", *Adv. Eng. Software* **13**(1991) 325-331.

[12] Steven Fortune and Christopher J. Van Wyk, "Efficient Exact Arithmetic for Computational Geometry" *Proceedings of the Ninth Annual Symposium on Computational Geometry*, ACM Press (1993) 163 172.

[13] H. Edelsbrunner and E. Mücke, "Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms", *Proceedings of the Fourth Annual Symposium on Computational Geometry*, ACM Press (1988) 118-133.

[14] Chee-Keng Yap, "A Geometric Consistency Theorem for a Symbolic Perturbation Scheme", *Proceedings of the Fourth Annual Symposium on Computational Geometry*, ACM Press (1988) 134 142

[15] Thomas Ottmann, Gerald Thiemt, and Christian Ullrich, "Numerical Stability of Geometric Algorithms", *Proceedings of the Third Annual Symposium on Computational Geometry*, ACM Press (1987) 119-125.

[16] D. Dobkin and D. Silver, "Recipes for Geometry and Numerical Analysis, Part I: An Empirical Study", *Proceedings of the Fourth Annual Symposium on Computational Geometry*, ACM Press (1988) 93-105.

[17] Christoph Hoffmann, John Hopcroft, and Michael S. Karasick, "Towards Implementing Robust Geometric Computation", *Proceedings of the Fourth Annual Symposium on Computational Geometry*, ACM Press (1988) 106-117.

[18] V. Milenkovic, "Calculating Approximate Curve Arrangements Using Rounded Arithmetic", *Proceedings of the Fifth Annual Symposium on Computational Geometry*, ACM Press (1989) 197 207.

[19] Zhenyu Li and Victor Mikenkovic, "Calculating Approximate Curve Arrangements Using Rounded Arithmetic", *Proceedings of the Sixth Annual Symposium on Computational Geometry*, ACM Press (1990) 235–243.

[20] Leonidas Guibas, David Salesin and Jorge Stolfi, "Epsilon Geometry: Building Robust Algorithms from Imprecise Computations", *Proceedings of the Fifth Annual Symposium on Computational Geometry*, ACM Press (1989) 208–217.

[21] Steven Fortune, "Numerical Stability of Algorithms for 2D Delaunay Triangulations", *Proceedings of the Eighth Annual Symposium on Computational Geometry*, ACM Press, (1992), 83–92.

[22] Christoph Hoffmann, "The Problems of Accuracy and Robustness in Geometric Computation", *IEEE Computer***22** (March 1989) 31–41.

[23] Donald R. Chand and Sham S. Kapur, "An Algorithm for Convex Polytopes", *Journal of the Association for Computing Machinery*, **17**:1 (1970) 78–86.

[24] Franco P. Preparata and Michael Ian Shamos, *Computational Geometry: an Introduction*, Springer-Verlag, New York, (1985).

[25] Anthony Ralston and Philip Rabinowitz, *A First Course in Numerical Analysis*, second edition, McGraw-Hill, New York, (1978) 22–23.

[26] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, (1973)

[27] Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, (1983).

[28] Alan Cline, manuscript, Summer 1987.

[29] C.L. Lawson, R.J. Hanson, D.R. Kincaid, and F.T. Krogh, "'Basic Linear Algebra Subprograms for FORTRAN Usage", *ACM Transactions on Math. Software* 5 (1979) 308–323.