International Journal of Computational Geometry & Applications

NUMERICAL STABILITY OF ALGORITHMS FOR 2D DELAUNAY TRIANGULATIONS

Steven Fortune

AT&T Bell Laboratories
Murray Hill, New Jersey, USA 07974

ABSTRACT

We consider the correctness of 2-d Delaunay triangulation algorithms implemented using floating-point arithmetic. The *α-pseudocircle* through points a, b, c consists of three circular arcs connecting ab, bc, and ac, each arc inside the circumcircle of a, b, c and forming angle α with the circumcircle; a triangulation is *α-empty* if the α-pseudocircle through the vertices of each triangle is empty. We show that a simple Delaunay triangulation algorithm—the flipping algorithm—can be implemented to produce $O(n\epsilon)$-empty triangulations, where n is the number of point sites and ϵ is the relative error of floating-point arithmetic; its worst-case running time is $O(n^2)$. We also discuss floating-point implementation of other 2-d Delaunay triangulation algorithms.

Keywords: Delaunay triangulations, numerical stability, robustness, error analysis

1. Introduction

Geometric algorithms are usually described assuming that exact real arithmetic is available to implement geometric predicates. Exact real arithmetic is a useful abstraction, simplifying the discovery, analysis, and presentation of algorithms. However, no computer provides exact real arithmetic, so to obtain an implementation, some substitution must be made. We consider the substitution of floating-point arithmetic in the context of 2-d Delaunay triangulation algorithms.

Many 2-d Delaunay triangulation algorithms are known, including flipping,[15] incremental,[10] random-incremental,[2] divide-and-conquer,[11,20] and sweepline[5] algorithms. For all but the sweepline algorithm, the necessary geometric primitives are the 2-d orientation test and the 2-d incircle test. Given planar points p_1, p_2, p_3, the 2-d orientation test decides if p_3 lies to the left, on, or to the right of the line directed from p_1 to p_2; given planar points p_1, p_2, p_3, p_4, the incircle test decides if p_4 lies outside, on, or inside the circle through p_1, p_2, p_3. Both tests can be expressed as the sign of an appropriate determinant (see figure 1).

If floating-point arithmetic is naively used to evaluate such determinants, the resulting implementation need not be correct. Figure 2 depicts a simple example of potential failure. Because point p_4 is very close to point p_1, the floating-point incircle test might erroneously report that p_4 is outside the circumcircle of p_1, p_2, p_3 (notice that this answer is the correct answer for a point p_4' very slightly perturbed from p_4). The floating-point incircle test might correctly report that p_1 is outside the circumcircle of p_3, p_4, p_5, so the implementation might report the triangulation in figure 2. However, point p_5 is well inside the circumcircle of $p_1p_2p_3$, so the reported triangulation is not even a good approximation to the true Delaunay triangulation of the point set.

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \qquad \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix} \qquad \begin{vmatrix} x_1 & y_1 & x_1^2+y_1^2 & 1 \\ x_2 & y_2 & x_2^2+y_2^2 & 1 \\ x_3 & y_3 & x_3^2+y_3^2 & 1 \\ x_4 & y_4 & x_4^2+y_4^2 & 1 \end{vmatrix}$$

Figure 1: From left to right: 2-d Orientation, 3-d Orientation, 2-d Incircle.

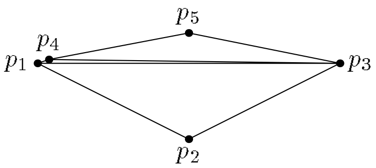


Figure 2: Possible incorrect Delaunay triangulation.

Erroneous results from primitives can be eliminated if arithmetic calculations are carried out exactly, for example with integer arithmetic. Most geometric primitives can be expressed as the sign-evaluation of a multivariate polynomial; evaluating a polynomial of degree d on integers of bit-length b requires intermediate values of approximate bit-length bd. The 2-d orientation test has degree 2 and the 2-d incircle test has degree 4. Thus with the 32-bit integer arithmetic native to many computers, the 2-d incircle test can be evaluated exactly with coordinates of about 7 bits; with 64-bit arithmetic, the coordinates can be about 15 bits.

Software multiprecision integer arithmetic permits input coordinates with longer bit-length. Unfortunately, the overhead is high. Experiments with standard, carefully coded multiprecision arithmetic packages indicate that multiprecision integer evaluation of 2-d incircle determinants is about two orders of magnitude slower than floating-point evaluation, with runtime increasing approximately quadratically with coordinate bit-length. Perhaps further tuning of multiprecision arithmetic is possible: for example, Karasick *et al.*[14] substantially improve the performance of a rational-arithmetic version of the 2-d divide-and-conquer Delaunay triangulation algorithm by using adaptive-precision arithmetic and by algorithmic changes (though in their case the engineering effort to tune arithmetic dominates the effort to implement the algorithm). However, it is easy to find algorithms using geometric predicates of high degree, implying that even carefully-tuned multiprecision arithmetic will be a dominant performance cost. For example, the 2-d incircle test has degree 8 if input points have homogeneous coordinates (perhaps desirable to allow rational coordinates or to avoid special cases); the primitive required by the 2-d sweepline Delaunay triangulation algorithm ("given two circles, each inscribed in three points, which has bigger maximal y-coordinate?") appears to have degree 20; the incircle test on line segments ("given a circle tangent to three line segments, specified by their endpoints, does a fourth segment intersect the circle?") appears to have degree 24. Cascading increases bit-length considerably; for example, the coordinate bit-lengths of the point of intersection of two line segments is about four times longer than the coordinate bit-length of the segment endpoints. If such computed coordinates are later used in other geometric primitives, algebraic degree and arithmetic bit-length go up accordingly.

We consider instead the use of floating-point arithmetic to implement geometric algorithms. Floating-point arithmetic has numerous engineering advantages: it is well-supported by programming languages; it is portable; it has useful features, such as automatic scaling; it has been extensively optimized in current computer hardware. The challenge is to demonstrate that a reliable implementation can result from the use of floating-point arithmetic.

Define the the α-*pseudocircle* of triangle Δabc to consist of circular arcs connecting ab, bc, and ac, with each arc lying inside the circumcircle C_{abc} of Δabc and forming angle α with C_{abc} (see figure 3(a)); define a triangulation to be *α-empty* if the α-pseudocircle of each triangle is empty of other sites. We show that the diagonal-flipping Delaunay triangulation algorithm can be implemented in floating-point arithmetic to produce $O(n\epsilon)$-empty triangulations. Here n is the number of input point sites and ϵ is the relative error in floating-point arithmetic. The sites do not have to be in general position: three sites can be approximately or exactly collinear and four sites can be approximately or exactly cocircular. The worst-case running time is $O(n^2)$, the same as the exact-arithmetic version of the algorithm.

The algorithm requires numerically stable implementations of the orientation test and the incircle test; the error analysis of the incircle test is a principal contribution of this paper. We describe a procedure *incircle* implemented using floating-point arithmetic: given four points a, b, c, d forming a convex quadrilateral, *incircle* reports either d inside, approximately on, or outside C_{abc}. The first and third answers are reliable; the second answer means that the angle between C_{abc} and C_{acd} is small, a constant multiple of ϵ. For example, in figure 2 *incircle* would report that p_4 is inside the $C_{p_1p_2p_3}$, since the angle between $C_{p_1p_2p_3}$ and $C_{p_1p_3p_4}$ is large even though p_1 and p_4 are close.

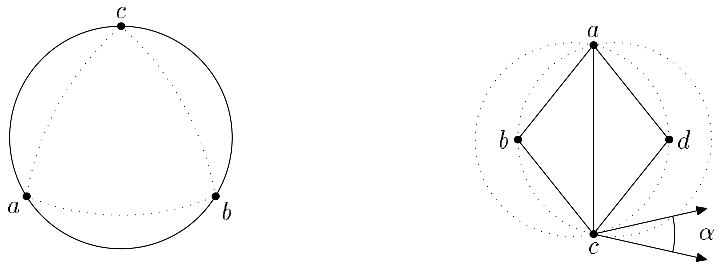
The exact-arithmetic flipping algorithm produces a triangulation so that each interior edge ac is "locally Delaunay": if triangles Δabc and Δacd are incident, then point d must be on or outside C_{abc} (and b must be on or outside C_{acd}). It is easy to see that a "locally Delaunay" triangulation is (a completion of) the Delaunay triangulation. The floating-point version establishes that each edge is "approximately locally Delaunay": d must be outside or approximately on C_{abc}, as reported by *incircle*. The assertion that the triangulation is $O(n\epsilon)$-empty follows from this "approximately locally Delaunay" property.

The other incircle-based Delaunay triangulation algorithms can be implemented using the floating-point orientation test and *incircle*. Unfortunately, the correctness proof does not immediately extend. One difficulty is maintaining a triangulation—some site configurations simply cannot be resolved by the floating-point 2-d orientation test. A second difficulty comes from the "approximately locally Delaunay" property. The exact-arithmetic version of, say, the divide-and-conquer algorithm establishes that a triangulation edge is "locally Delaunay" either by an explicit incircle test or by an inference based on incircle tests on other sites. Unfortunately, the corresponding inference using the "approximately locally Delaunay" property potentially leads to nonconstant local error, jeopardizing the global $O(n\epsilon)$ error bound. Such nasty situations seem unlikely to occur very often, and can be easily detected, suggesting that a floating-point implementation will be reliable in practice. We discuss the other algorithms at greater length in section 6.

Can a different, possibly more natural, error analysis be given for Delaunay triangulations? Numerical analysis traditionally uses either forward or backward error analysis. A forward error analysis would require that the computed triangulation be "near" the true Delaunay triangulation, but it is unclear what it means for one triangulation to be near another. A backward error analysis would require that the computed triangulation be the true Delaunay triangulation of a slightly perturbed set of sites. Clearly, such a computed triangulation must be "combinatorially Delaunay," that is, the Delaunay triangulation of some set of sites. Recent work by Rivin, Smith, and Dillencourt[4,19] suggest the possible existence of simple criteria guaranteeing that a triangulation is combinatorially Delaunay–in time this may lead to a true backward error analysis.

Other work. A number of floating-point analyses of geometric algorithms have been given. Fortune[6] and Li and Milenkovic[16] consider 2-d convex hulls; Fortune and Milenkovic[9] consider 2-d line arrangements; Barber[1] considers 3-d convex hulls; Hoffmann, Hopcroft, and Karasick[13] consider set operations on polyhedral solids. Various references discuss issues of exact vs. approximate arithmetic[6,12,17,18].

Sugihara and Iri[21,22] consider 2-d Voronoi diagrams, the dual of Delaunay triangulations. They give a floating-point algorithm that guarantees that the computed Voronoi diagram is a planar graph, with one region per input site. However they do not give any numerical bounds on its accuracy. This paper can



(a) (b)

Figure 3: (a) Pseudocircle; (b) $\alpha = \mu(a, b, c, d)$.

be viewed as the beginnings of a justification of Sugihara and Iri's claim of empirically observed numerical stability.

2. Definitions

For this section, let a, b, c, d be distinct points in R^2. A *triangle* Δabc is a cyclically ordered triple of points, thus $\Delta abc = \Delta bca = \Delta cab$. Δabc is *positive*, *flat*, or *negative* as c is to the left, on, or to the right of the line directed from a to b. $\theta(a, b)$ is the (unit) direction vector from a to b. The *quadrant* of a direction vector is one of the four possibilities 1, 2, 3, 4 (with a direction parallel to an axis lying in the quadrant to its left).

C_{abc} is the circle through a, b, c and $\phi_a(b, c)$ is the unit tangent vector at a to C_{abc}, where C_{abc} is oriented so that it passes through a, b, c in that order. If Δabc is positive and $\alpha \in [0, \pi/2]$, then the *α-pseudocircle through* Δabc consists of three circular arcs ab, bc, ac, where for example every point p of arc ab satisfies $\phi_a(p, b) = \phi_a(b, c) + \alpha$ (see figure 3(a)). For positive triangles Δabc and Δacd, define

$$\begin{aligned} \mu(a, b, c, d) &= \angle cba + \angle adc - \pi \\ &= \phi_c(d, a) - \phi_c(a, b) \end{aligned}$$

(see figure 3(b)); some elementary geometry is necessary to show the second equality. Notice $\mu(a, b, c, d) > 0$, $= 0$, or < 0 as d is inside, on, or outside C_{abc}; μ is a measure of how much inside or outside. The *angle between* circles C_{abc} and C_{acd} is

$$\min(|\phi_c(d, a) - \phi_c(a, b)|, |\phi_c(d, a) - \phi_c(b, a)|);$$

thus the angle between two circles is always in the range $[0, \pi/2]$.

Euclidean distance is denoted by e, thus $e(a, b)$ is the distance from a to b and $e(a) = e(\mathcal{O}, a)$ is the distance of a from the origin $\mathcal{O}$. Points a and b are *lexicographically ordered*, $a \prec b$, if $a_x < b_x$ or both $a_x = b_x$ and $a_y < b_y$. Mapping ι is inversion about the unit circle centered at the origin, thus

$$\iota(p) = p/e(p)^2.$$

For reals or points a and b, *relative distance* d_r is defined by

$$d_r(a, b) = e(a, b)/\max(e(a), e(b)),$$

as long as $e(a), e(b)$ not both 0; a and b *differ relatively* by α if $d_r(a, b) \leq \alpha$.

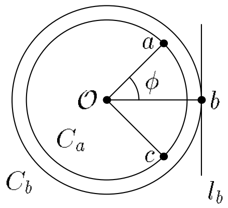


Figure 4: Proof that d_r is a metric.

Lemma 1 *Let $a, b \in R^2$ and $\alpha, \beta \geq 0$.*

(i) d_r is a metric.
(ii) $d_r(\alpha a, \beta b) \leq d_r(\alpha, \beta) + d_r(a, b)$.
(iii) If $a \neq \mathcal{O} \neq b$ then $d_r(a, b) = d_r(\iota(a), \iota(b))$.

Proof.

(i) It suffices to show that d_r satisfies the triangle inequality, that is,

$$\frac{e(a,b)}{\max(e(a), e(b))} + \frac{e(b,c)}{\max(e(b), e(c))} \geq \frac{e(a,c)}{\max(e(a), e(c))}. \tag{1}$$

Inequality (1) is obvious if a or c is furthest from the origin; otherwise by scaling and renaming we can assume $e(b) = 1 \geq e(a) \geq e(c)$, so that (1) simplifies to

$$e(a,b) + e(b,c) \geq \frac{e(a,c)}{e(a)}. \tag{2}$$

We reduce (2) to a special position of a, b, c and then show that the inequality holds. Let C_a and C_b be the circles centered at the origin through a and b, respectively, and l_b the line through b tangent to C_b. See figure 4. First notice that if a and b are fixed, and c is free to vary with $s = e(b, c)$ fixed and $e(c) \leq e(a)$, then $e(a, c)$ is maximized if c lies on C_a on the opposite side of line $\mathcal{O}b$ from a (if $\mathcal{O}, a, b$ are collinear, then c can be chosen as either point on C_a at distance s from b). Now suppose a and c are fixed on C_a, and notice that $e(a, b) + e(b, c)$ is minimized if the angle between l_b and line ab equals the angle between l_b and line bc. Hence we can assume that a and c lie on C_a and $\angle b\mathcal{O}a = \angle c\mathcal{O}b = \phi$, where $\phi \in [0, \pi]$. Then some manipulation yields

$$\begin{aligned} e(a,c) &= 2e(a)\sin\phi \\ e(a,b) &= e(b,c) = \sqrt{(e(a) - \cos\phi)^2 + \sin^2\phi}. \end{aligned}$$

Substituting into (2) and squaring yields the immediate assertion

$$(e(a) - \cos\phi)^2 + \sin^2\phi \geq \sin^2\phi.$$

(ii) Using (1) and elementary properties of d_r,

$$d_r(\alpha a, \beta b) \leq d_r(\alpha a, \alpha b) + d_r(\alpha b, \beta b) = d_r(a, b) + d_r(\alpha, \beta).$$

(iii) Substituting in the definition of d_r and ι and simplifying, we get

$$d_r(\iota(a), \iota(b)) = \frac{e(\frac{a}{e(a)^2}, \frac{b}{e(b)^2})}{\max(\frac{1}{e(a)}, \frac{1}{e(b)})} = \frac{e(a,b)}{e(a)e(b)\max(\frac{1}{e(a)}, \frac{1}{e(b)})} = \frac{e(a,b)}{\max(e(a), e(b))}.$$

/* Procedure $orient(a, b, c)$:
Points a, b, c must be distinct.
Returns -1 if Δabc negative
 0 if $S(a, b, c) \vee S(a, c, b) \vee S(b, a, c)$
 1 if Δabc positive */
Sort so that $a \prec b \prec c$, recording $\sigma = \pm 1$ as
 parity of number of inversions.
$r := (b_y - a_y)(c_x - b_x)$;
$s := (b_x - a_x)(c_y - b_y)$;
if $r > s(1 + 7\epsilon)$ **then** return $(-\sigma)$
if $r < s(1 - 7\epsilon)$ **then** return (σ)
return (0)

Figure 5: Procedure *orient*

□

3. Geometric primitives in approximate arithmetic

Approximate arithmetic is a set of operations $\hat{+}$, $\hat{-}$, $\hat{\times}$, $\hat{\div}$ defined on the reals that make relative error ϵ, e.g. for all $x, y \in R$, there is $|\eta| \leq \epsilon$ so that $x \hat{-} y = (x-y)(1+\eta)$. Operations $=$ and $<$ are exact; thus, for example, sorting is exact. We use approximate arithmetic to implement geometric primitives (approximate arithmetic simplifies true floating-point arithmetic by ignoring underflow and overflow, but these possibilities can be addressed in conceptually simple ways). We assume $\epsilon < 10^{-4}$; typical double-precision floating-point has $\epsilon \approx 10^{-16}$. For the following, operations $+, -, \times, \div$ appearing in program text are approximate; higher-level operations such as dot-product are implemented in the obvious way using approximate arithmetic.

Two-dimensional orientation. Let $\delta_T = 8\epsilon$. Distinct points a, b, c satisfy $S(a, b, c)$ if $\theta(b, c)$ and $-\theta(b, a)$ lie in the same quadrant and differ by at most δ_T. Informally, $S(a, b, c)$ means point b approximately lies on segment ac. Clearly $S(a, b, c)$ iff $S(c, b, a)$.

Figure 5 gives procedure *orient*, an approximate-arithmetic implementation of the two-dimensional orientation test. Given three distinct points a, b, c, it asserts either that Δabc is positive, that Δacb is positive, or $S(a, b, c) \vee S(a, c, b) \vee S(b, a, c)$. (A slight modification of the procedure using extra coordinate comparisons can separate the cases $S(a, b, c)$, $S(a, c, b)$, and $S(b, a, c)$). The proof of the following lemma has appeared elsewhere.[6]

Lemma 2 *Procedure orient is correct.*

Three-dimensional orientation. Let p_1, p_2, p_3, p_4 be distinct points in R^3, with $p_i = (x_i, y_i, z_i)$. The sign of the middle determinant in figure 1 determines the orientation of simplex $p_1p_2p_3p_4$, that is, whether p_4 is in the positive halfspace, on, or in the negative halfspace of the oriented plane through $p_1p_2p_3$. We briefly discuss approximate-arithmetic evaluation of the orientation determinant, in order to motivate the more complicated analysis of the incircle test. (The orientation test in three dimensions is not itself needed in this paper.)

Using standard numerical techniques,[3] it is possible to compute an approximate determinant $\hat{d}$ and an error estimate $\hat{\delta} > 0$ so that the true determinant lies in the interval $[\hat{d}-\hat{\delta}, \hat{d}+\hat{\delta}]$, and so that for any $d \in [\hat{d}-\hat{\delta}, \hat{d}+\hat{\delta}]$, there is a slightly perturbed set of points p'_1, p'_2, p'_3, p'_4 having determinant d. The perturbation is a small relative one, i.e. $d_r(p_i, p'_i)$ is a constant multiple of ϵ. Thus if $|\hat{d}| > \hat{\delta}$, the sign of $\hat{d}$ is the true sign; if $|\hat{d}| \leq \hat{\delta}$, the true sign is unknown but there is a nearby flat simplex $p'_1p'_2p'_3p'_4$.

This computation can be improved. In particular, in the case that the sign is uncertain, it is possible to guarantee that there is a nearby flat simplex, where each point has been moved by an amount that is small relative to the distance to the other points, rather than small relative to distance to the origin. The computation is as follows. Find a minimum spanning tree of the points; approximate distance computation is adequate. If there is a node of degree three, say p_1, compute instead the determinant using points $q_1 = \mathcal{O}$, $q_2 = p_2 - p_1$, $q_3 = p_3 - p_1$, $q_4 = p_4 - p_1$. If the tree is a path, say p_2p_1, p_1p_3, p_3p_4, compute instead the determinant using points $q_1 = \mathcal{O}$, $q_2 = p_2 - p_1$, $q_3 = p_3 - p_1$, $q_4 = p_4 - p_3$. Notice that the difference vectors can be computed accurately, i.e. with relative error a constant times ϵ. If the magnitude of the determinant is greater than the error estimate, the determinant sign is correct. If the magnitude of the determinant is less than the error estimate, there are points q'_1, q'_2, q'_3, q'_4 that form a flat simplex, where $d_r(q'_i, q_i)$ is a constant multiple of ϵ. In fact it is possible to choose $q'_1 = q_1 = \mathcal{O}$. Now define p'_i using the q'_i, for example in the first case set $p'_1 = p_1$, $p'_2 = p_1 + q'_2$, $p'_3 = p_1 + q'_3$ and $p'_4 = p_1 + q'_4$. The claim is that $p'_1p'_2p'_3p'_4$ is a flat simplex so that for each i, j with $1 \leq i < j \leq 4$, $d_r(p_i - p_j, p'_i - p'_j)$ is a (bigger) constant multiple of ϵ. This is immediate if p_ip_j is a spanning tree edge. If p_ip_j is not a spanning tree edge, then there is a path in the tree which together with p_ip_j forms a cycle; the bound comes from adding the perturbations of the edges along the path and observing that p_ip_j is longer than any other edge in the cycle.

The Incircle test. Suppose points $a, b, c, d \in R^2$ form a convex quadrilateral, with vertices in counter-clockwise order. The incircle predicate[11] is defined to be true if d is inside C_{abc}, and false otherwise. Define the *lifting map* $\lambda : R^2 \rightarrow R^3$ by $\lambda((x, y)) = (x, y, x^2 + y^2)$. Then the incircle predicate is true exactly if the simplex $\lambda(a)\lambda(b)\lambda(c)\lambda(d)$ has positive orientation; if the simplex is flat, points a, b, c, d are cocircular.[11]

Figure 6 gives an approximate-arithmetic procedure *incircle*. The call *incircle*(a, b, c, d) gives one of three answers: "d inside C_{abc}", "d outside C_{abc}", or "a, b, c, d approximately cocircular." The meaning of "approximately cocircular" is that $|\mu(a, b, c, d)| < \delta_I$, where δ_I is a constant multiple of ϵ determined by the error analysis of *incircle*. Notice that if, for example, $\mu(a, b, c, d)$ is small but positive, *incircle* is allowed to return either "d inside C_{abc}" or "$abcd$ approximately cocircular."

Procedure *incircle* follows the sketch outlined before for finding the orientation of a simplex in R^3. One difference is that it is not necessary to find the true minimum spanning tree. Due to the specific nature of the incircle test, one diagonal of $abcd$ can be chosen arbitrarily and the two other vertices connected to the end points of the diagonal; it does not matter if the two other vertices are close together. The lifting is done after difference vectors are computed. This is straightforward if all spanning tree edges are incident to one point, (figure 7(a)). If the tree is in fact a path, say ab, bd, dc (figure 7(b)), the lifted values are $\mathcal{O}$, $\lambda(a - b)$, $\lambda(d - b)$, and Λ, where $\Lambda = \lambda(c - b) - \lambda(d - b) = (c_x - d_x, c_y - d_y, e(c - d)^2 + 2(c - d) \cdot (d - b))$. Some simplification of the determinant calculation is possible since one of the points is the origin.

If the magnitude of the computed determinant is less than the uncertainty bound, we must show that $|\mu(a, b, c, d)|$ is small. To do this, we show that there are slightly perturbed points a', b', c', d' that are cocircular. The perturbation is small in the sense that any difference vector between two points changes by a small relative amount.

A slight difficulty is that the analysis of the orientation test guarantees that there are slight perturbations of the lifted points that are coplanar. However, the perturbed points may not lie on the paraboloid $\lambda(R^2)$, hence the projection onto R^2 of the perturbed points may not be cocircular. There is a canonical way of obtaining cocircular points in R^2 from coplanar points in R^3. Define the *scaling map* $\sigma : R^3 \rightarrow R^2$ by

/* *incircle*(a, b, c, d):
Points a, b, c, d must form a convex quadrilateral.
Returns: -1 if d outside C_{abc}
    0 if $-\delta_I \leq \mu(a,b,c,d) \leq \delta_I$
    1 if d inside C_{abc}
α bounds the absolute error in z_3.
β bounds the absolute error in det. */
var q_1, q_2, q_3 : 2dPoint, p_1, p_2, p_3 : 3dPoint;
Let T consist of edge bd, the shorter of ab and bd, and the shorter of bd and cd.
Up to symmetry there are two cases:
case $T = \{ab, bd, bc\}$: /* Figure 7(a) */
    $q_1 := a - b$; $q_2 := d - b$; $q_3 := c - b$;
    $p_1 := \lambda(q_1)$; $p_2 := \lambda(q_2)$; $p_3 := \lambda(q_3)$;
    $\alpha := 5\epsilon e(q_3)^2$;
case $T = \{ab, bd, dc\}$: /* Figure 7(b) */
    $q_1 := a - b$; $q_2 := d - b$; $q_3 := c - d$;
    $p_1 := \lambda(q_1)$; $p_2 := \lambda(q_2)$; $p_3 := \lambda(q_3) + (0, 0, 2q_2 \cdot q_3)$;
    $\alpha := 5\epsilon e(q_3)^2 + 15\epsilon e(q_2)e(q_3)$;
Let $p_i = (x_i, y_i, z_i)$, $i = 1, 2, 3$.
$det := z_1(x_2y_3 - y_2x_3) - z_2(x_1y_3 - x_3y_1) + z_3(x_1y_2 - x_2y_1)$;
$\beta := 12\epsilon(|z_1|(|x_2y_3| + |x_3y_2|) + |z_2|(|x_1y_3| - |x_3y_1|)) + 6\epsilon(|z_3| + \alpha)(|x_2y_1| + |x_1y_2|)$;
if $det < -\beta$ **then** return(-1)
else if $det > \beta$ **then** return(1)
else $dot1 := (a-b)\cdot(c-b)/(e(a-b)e(c-b))$;
    $dot2 := (a-d)\cdot(c-d)/(e(a-d)e(c-d))$;
    if $dot1 < -.9$ and $dot2 < -.9$ **then** return(1)
    else if $dot1 > .9$ and $dot2 > .9$ **then** return(-1)
    else return(0)

Figure 6: Approximate-arithmetic implementation of *incircle*

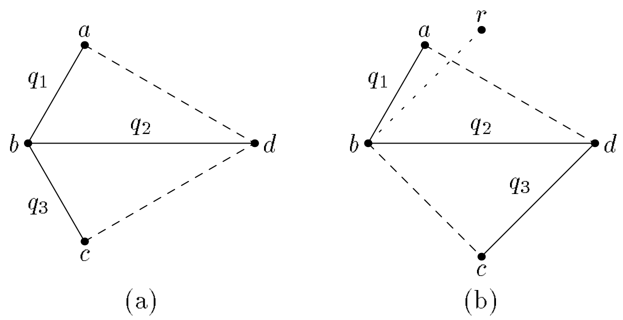


Figure 7: Two cases for spanning tree T.

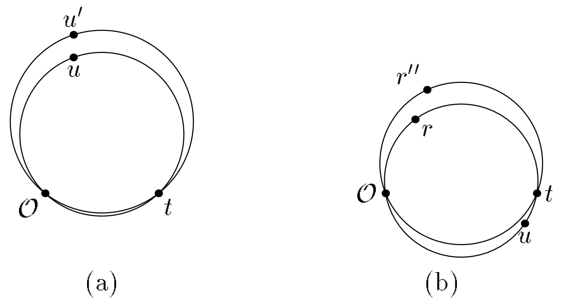


Figure 8: Angle between circles.

$\sigma((x,y,z)) = (x/z, y/z)$, $z \neq 0$. Clearly $\sigma(p) = \sigma(p')$ iff p and p' lie on a common line through the origin. An easy calculation shows that both $\iota \circ \sigma \circ \lambda$ and $\sigma \circ \lambda \circ \iota$ are the identity. If $p \in R^3$, let $q = \iota(\sigma(p)) \in R^2$, then $\sigma(\lambda(q)) = \sigma(\lambda(\iota(\sigma(p)))) = \sigma(p)$, and in fact $\lambda(q)$ is the intersection of the line $\mathcal{O}p$ with the paraboloid. Hence if points $\mathcal{O}, p_1, p_2, p_3$ are coplanar, then $\mathcal{O}, q_1, q_2, q_3$ are all cocircular, where $q_i = \iota(\sigma(p_i))$. A useful calculation is $\iota(\sigma((x,y,z))) = z\iota((x,y))$; this shows for example that if $z \approx x^2 + y^2$, then $\iota(\sigma((x,y,z))) \approx (x,y)$.

The following lemma shows how relative perturbation of points corresponds to angular difference between circles.

Lemma 3 *Suppose t, u, u' are distinct points with $d_r(u,u') = \tau < \frac{1}{10}$ and let $\kappa \geq 1$ satisfy $e(u,t) \geq \frac{1}{\kappa} e(u,\mathcal{O})$. Let α be the angle between $C_{\mathcal{O}tu}$ and $C_{\mathcal{O}tu'}$. Then $\sin\alpha \leq 3\kappa\tau$. (See figure 8(a)).*

Proof. Assume that t lies on the x-axis, so the perpendicular bisector l of $\mathcal{O}$ and t is vertical; the centers z and z' of $C_{\mathcal{O}tu}$ and $C_{\mathcal{O}tu'}$, respectively, lie on l. By interchanging $\mathcal{O}$ and t if necessary, we can assume that u lies on or to the left of l and that u' is on or inside the circle D of radius $\frac{10\kappa\tau}{9} e(u)$ (the factor $\frac{10}{9}$ comes from using absolute distance; the factor κ comes from possibly interchanging $\mathcal{O}$ and t). See figure 9(a).

Let α be the measure of $\angle zaz'$; clearly $|\alpha|$ is the angle between $C_{\mathcal{O}tu}$ and $C_{\mathcal{O}tu'}$. Clearly α is maximized if z' has as large a y-coordinate as possible. For $u' \in D$, the maximal α is obtained if u' lies on D with z', u, u' collinear in that order: if u' is any other point of D, a motion of u' preserving $e(u',z') \geq e(u,u')$ into the interior of D is possible, and then another motion of u' would raise z'.

Let $\beta = \angle u\mathcal{O}u'$, $\gamma = \angle z'\mathcal{O}u$ and $\delta = \angle zuz'$. Certainly $\beta \leq \arcsin e(u,u')/e(u) \leq \pi/4$. Add up the angles at u, using the fact that both $\Delta\mathcal{O}zu$ and $\Delta\mathcal{O}z'u'$ are isosceles:

$$\pi = \delta + (\gamma + \alpha) + (\pi - (\beta + \beta + \gamma)).$$

Simplifying yields $2\beta = \alpha + \delta$; since $\delta \geq 0$, $\alpha \leq 2\beta \leq \pi/2$. Thus

$$\sin\alpha \leq \sin 2\beta \leq 2\sin\beta \leq 2\frac{e(u,u')}{e(u)} \leq 2 \cdot \frac{10\kappa\tau}{9} \leq 3\kappa\tau.$$

The argument for minimizing α is similar. We get $|\sin\alpha| \leq e(u,u')/e(u')$, and the claimed bound follows using $e(u') \geq \frac{9}{10} e(u)$. □

Consider the case when the spanning tree is a path (figure 7(b)). When c and d are approximately equidistant from b, the relative error in the computed value of z_3, the z-component of Λ, can be large. (Notice that α estimates the absolute error of z_3). To estimate the effect of the error, it is useful to give another interpretation of Λ. Define $\rho(b,c,d)$ to be the other point of intersection of C_{bcd} with the line through b parallel to cd. Points $b, c, d, r = \rho(b,c,d)$ form a trapezoid inscribed in C_{bcd} (see figure 7(b)). It is possible

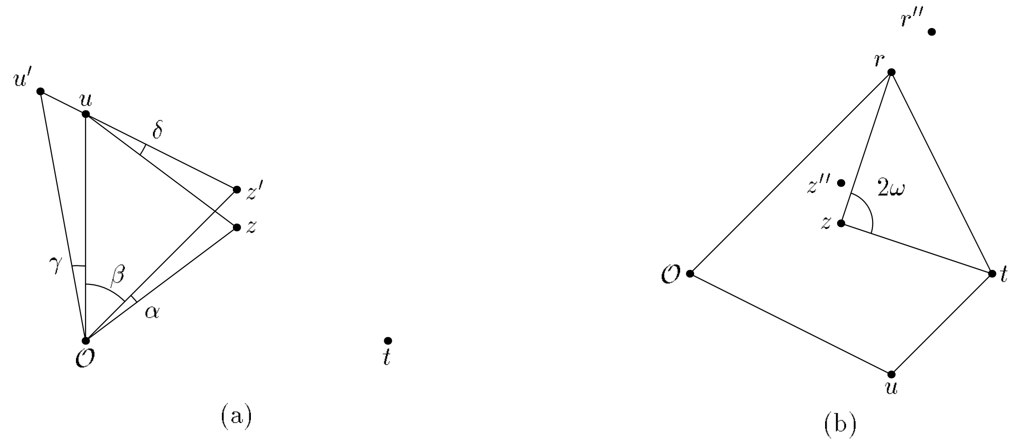


Figure 9: Lemma proofs.

to verify that r is $b + \alpha(c-d)$, where $\alpha = 1 + \frac{2(c-d)\cdot(d-b)}{e(c-d)^2}$. Notice that r may be on either side of line bd, depending upon the sign of α. Some arithmetic shows that $\alpha\lambda(r-b) = \Lambda$; thus both $\lambda(r-b)$ and Λ lie on a common line through the origin, and $\iota(\sigma(\Lambda)) = \iota(\sigma(r-b))$.

Lemma 4 *Suppose $\mathcal{O}, t, u$ are distinct points; set $r = \rho(\mathcal{O}, t, u)$ and let r'' be collinear with r and $\mathcal{O}$. (See figure 8(b).)*

(i) The angle α between $C_{\mathcal{O}tr}$ and $C_{\mathcal{O}tr''}$ satisfies $\sin\alpha \le e(r, r'')/e(u)$.
(ii) As long as $e(t, u) \le 2e(u)$ and $e(r, r'') \le e(u)/10$, $e(r'') \le 5e(r'', t)$.

Proof.

(i) Assume that u and r are on opposite sides of line $\mathcal{O}t$ (otherwise a symmetric argument suffices) and that $\mathcal{O}, u, t, r$ are in that counterclockwise order. Let z and z'' be the centers of circles $C_{\mathcal{O}tr}$ and $C_{\mathcal{O}tr''}$, respectively, let $\angle t\mathcal{O}r = \omega$, so $\angle tzr = 2\omega$, and let R be the radius $C_{\mathcal{O}tr}$. See figure 9(b). We have

$$e(u) = e(r, t) = 2R\sin\omega$$

and since the distance between the projections of z and z'' on line $\mathcal{O}r$ is $e(r, r'')/2$,

$$e(z, z'') = e(r, r'')/(2\sin\omega).$$

Hence

$$\sin\alpha \le e(z, z'')/R = e(r, r'')/e(u).$$

(ii) We have $e(r'', t) \ge e(r, t) - e(r, r'')$; using $e(r, t) = e(u)$ and $e(r, r'') < e(u)/10$, we get $e(r'', t) \ge \frac{9}{10}e(u)$. Hence

$$e(r'') \le e(u) + e(t, u) + e(t, r) + e(r, r'') \le \frac{41}{10}e(u) \le \frac{41}{9}e(r'', t) \le 5e(r'', t).$$

□

Theorem 1 *Procedure incircle is correct.*

Proof. We write for example p_1 for a true value (computed using exact arithmetic), $\hat{p}_1$ for the value computed in approximate arithmetic, and p_1' for a perturbed value.

Straightforward but tedious manipulations of error bounds show the following:

(i) If $|\hat{det}| > \hat{\beta}$ then the signs of det and $\hat{det}$ are the same.

(ii) If $|\hat{det}| < \hat{\beta}$, then there are perturbed values p'_1, p'_2, p'_3 so that

(a) $\mathcal{O}, p'_1, p'_2, p'_3$ are coplanar.

(b) $p'_i = (x'_i, y'_i, z'_i)$ and $p_i = (x_i, y_i, z_i)$ differ in each coordinate relatively by $O(\epsilon)$, with the exception that z_3 and z'_3 differ absolutely by at most 2α.

(c) $z'_i \neq 0$, $i = 1, 2, 3$.

The correctness of *incircle* in case (i) is immediate. Assume case (ii) holds; we first show that the angle between C_{abc} and C_{acd} is $O(\epsilon)$.

Suppose the spanning tree satisfies the first case. Then $\alpha/\hat{z}_3$ is $O(\epsilon)$, so each coordinate of p_i and p'_i differ relatively by $O(\epsilon)$, $i = 1, 2, 3$. Define

$$q'_i = \iota(\sigma(p'_i)) = z'_i \iota((x'_i, y'_i)),$$

$i = 1, 2, 3$; since $\mathcal{O}$, p'_1, p'_2, p'_3 are coplanar, $\mathcal{O}$, q'_1, q'_2, q'_3 are cocircular. Notice that by definition,

$$q_i = z\iota((x_i, y_i)).$$

Since inversion ι preserves relative distance, we have that q_i and q'_i differ relatively by $O(\epsilon)$. Now choose a similarity transformation $\psi : R^2 \rightarrow R^2$ that fixes $\mathcal{O}$ and rotates, then dilates, the plane so that $\psi(q'_2) = q_2$. Clearly $d_r(t, \psi(t))$ is independent of t, so for any t, $d_r(t, \psi(t)) = d_r(q'_2, \psi(q'_2)) = d_r(q'_2, q_2)$, which is $O(\epsilon)$. Consider the sequence of circles

$$C_{\mathcal{O}q_2q_3}, C_{\mathcal{O}q_2q'_3}, C_{\mathcal{O}q_2\psi(q'_3)}, C_{\mathcal{O}q_2\psi(q'_1)}, C_{\mathcal{O}q_2q'_1}, C_{\mathcal{O}q_2q_1};$$

the third and fourth circles are equal because $\mathcal{O}$, $\psi(q'_1)$, $\psi(q'_2) = q_2$, $\psi(q'_3)$ are cocircular. We can use lemma 3 to show that the angle between successive pairs of circles is $O(\epsilon)$: t from lemma 3 is q_2; u, u' are consecutive pairs from q_3, q'_3, $\psi(q'_3)$ or $\psi(q'_1)$, q'_1, q_1; α is $O(\epsilon)$; choosing $\kappa = 1 + O(\epsilon)$ is sufficient, since $e(u,t) \geq \frac{1}{\kappa} e(u, \mathcal{O})$ follows from the choice of the spanning tree T and the fact that the perturbations are not large. Hence the angle between the first and the last is $O(\epsilon)$; so the angle between C_{abc} and C_{acd} is $O(\epsilon)$.

Now suppose that the spanning tree satisfies the second case. Define q'_1, q'_2 and ψ as before and define

$$\begin{aligned} r' &= z'_3 \iota((x'_3, y'_3)) = \iota(\sigma(p'_3)) \\ r'' &= z'_3 \iota((x_3, y_3)) \\ r &= z_3 \iota((x_3, y_3)) = \rho(\mathcal{O}, q_2, q_2 + q_3). \end{aligned}$$

We wish to show that the angle between circles C_{abc} and C_{acd} is $O(\epsilon)$; notice that C_{abc} and C_{acd} are translates of the first and last circles of the sequence

$$C_{\mathcal{O}q_2q_1}, C_{\mathcal{O}q_2\psi(q'_1)}, C_{\mathcal{O}q_2\psi(r')}, C_{\mathcal{O}q_2r}.$$

The angle between the first two is $O(\epsilon)$ by the same argument as the previous case; the second two circles are the same since $\mathcal{O}$, $\psi(q'_1)$, q_2, $\psi(r')$ are cocircular. For the angle between the last two circles, consider the sequence of circles

$$C_{\mathcal{O}q_2r}, C_{\mathcal{O}q_2r''}, C_{\mathcal{O}q_2r'}, C_{\mathcal{O}q_2\psi(r')}.$$

Recall that

$$\alpha \leq \epsilon e(q_3)(5e(q_3) + 15e(q_2));$$

by the definition of the spanning tree, we have

$$e(q_3) \leq e(q_2 + q_3)(1 + O(\epsilon))$$

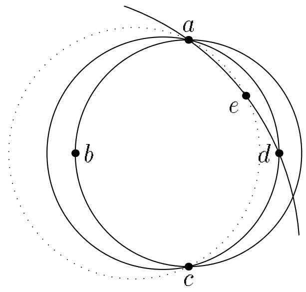


Figure 10: Bounding μ.

$$e(q_2) \le e(q_2 + q_3) + e(q_3) \le (2 + O(\epsilon))e(q_2 + q_3).$$

Hence

$$e(r, r'') = \frac{|z_3' - z_3|}{e(q_3)} \le \frac{2\alpha}{e(q_3)} \le \epsilon(10e(q_3) + 30e(q_2)) \le 71\epsilon e(q_2 + q_3).$$

Combining we get that $e(r, r'')/e(q_2+q_3)$ is $O(\epsilon)$; hence we can apply lemma 4(1) with $t = q_2$ and $u = q_2+q_3$ to obtain the bound of $O(\epsilon)$ for the angle between the first pair of circles in the sequence. The angles between the other consecutive pairs can be bounded by $O(\epsilon)$ using lemma 3: t from lemma 3 is q_2, u, u' are consecutive pairs from r'', r', $\psi(r')$; α is $O(\epsilon)$; and the choice $\kappa = 5 + O(\epsilon)$ is sufficient using lemma 4(2) and the fact that perturbations are small.

To finish the proof, we have that $abcd$ is a convex quadrilateral and that the angle between C_{abc} and C_{acd} is $O(\epsilon)$. Hence $\angle cba + \angle adc$ is within $O(\epsilon)$ of 0, π, or 2π. In the first case d is outside C_{abc}; in the second case $|\mu(a, b, c, d)| < O(\epsilon)$; and in the third d is inside C_{abc}. It is easy to check that the computation of $dot1$ and $dot2$ and the final **if** statement resolves the three cases. □

4. Approximate Delaunay triangulations.

A *triangulation* of a set P of points is a plane graph with vertex set P so that all bounded faces are triangles; a triangulation is *convex* if the complement of the external face is convex. A triangulation is *almost convex* if for any three consecutive vertices a, b, c on the boundary of the triangulation in counterclockwise order, either $S(a, b, c)$ or Δabc positive; thus if external angle $\angle abc$ is not convex, then it is within δ_T of being convex, and $\theta(a, b)$ and $\theta(b, c)$ lie in the same quadrant. In an almost-convex triangulation, both sides of an edge may be incident to the external face; for example a simple path could be an almost-convex triangulation.

Suppose Δabc and Δacd are positive. Predicate $D(a, b, c, d)$ is $\mu(a, b, c, d) < \delta_I$ (equivalently $\angle cba + \angle adc < \pi + \delta_I$). An exercise is to show that $D(a, b, c, d)$ holds if Δbcd is negative, Δabd is negative, $S(b, c, d)$, or $S(b, a, d)$.

$\mathcal{T}$ is an *approximate Delaunay triangulation* if it is almost convex and for each internal edge ac with incident triangles Δabc and Δacd, $D(a, b, c, d)$ holds; $D(a, b, c, d)$ is the *(approximate Delaunay) edge predicate* for edge ac. A fixed set of sites may have many approximate Delaunay triangulations; for example, if the sites are sufficiently close to cocircular, then any triangulation is an approximate Delaunay triangulation.

Lemma 5 *Suppose a, b, c, d, e are distinct points and Δabc, Δacd, Δade and Δace are all positive. Then*

$$\mu(a, b, c, e) \le \max(0, \mu(a, b, c, d)) + \max(0, \mu(a, c, d, e)).$$

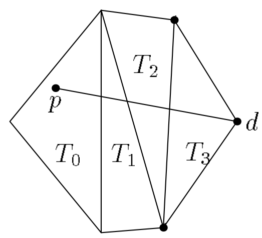


Figure 11: Induction order.

Proof. Suppose, as drawn in figure 10, d is inside C_{abc} and e is inside C_{adc}, i.e. both $\mu(a,b,c,d) > 0$ and $\mu(a,c,d,e) > 0$. Then circles C_{acb}, C_{adc}, C_{aec}, C_{aed} appear in that order if ordered by the direction of the clockwise tangent at a; the lemma follows easily. Other cases are similar. □

A triangulation $\mathcal{T}$ is *α-empty* if for each triangle of $\mathcal{T}$, the interior of the α-pseudocircle of the triangle contains no vertices of $\mathcal{T}$.

Theorem 2 *Let $\mathcal{T}$ be an approximate Delaunay triangulation of a set of n point sites. Then $\mathcal{T}$ is $n\delta_I$-empty.*

Proof. First we augment $\mathcal{T}$ to a convex approximate Delaunay triangulation $\mathcal{T}'$. Consider the *left boundary chain* of $\mathcal{T}$, i.e. the maximal set of boundary edges b_1b_2, b_2b_3, $\ldots$, $b_{m-1}b_m$ in clockwise order so that $b_1, \ldots, b_m$ have strictly increasing y-coordinate. Choose a point s_l with y-coordinate in $[(b_1)_y, (b_m)_y]$ and with x-coordinate sufficiently small to guarantee that, first, for each i, $\Delta s_l b_i b_{i+1}$ is positive and, second, for each i, if there is a site c_i opposite edge $b_i b_{i+1}$, then $D(s_l, b_i, c_i, b_{i+1})$. Add to $\mathcal{T}$ point s_l as a new vertex with edges to $b_1, \ldots, b_m$; similarly add a new site s_r with edges to the vertices of the right boundary chain. Then clearly the resulting triangulation $\mathcal{T}'$ is convex. $\mathcal{T}'$ is also an approximate Delaunay triangulation: the approximate Delaunay predicate for edge $s_l b_i$, $D(s_l, b_{i-1}, b_i, b_{i+1})$, follows from $S(b_{i-1}, b_i, b_{i+1})$ or $\Delta b_{i-1} b_i b_{i+1}$ positive.

Choose a triangle T_0 in $\mathcal{T}'$; we show T_0 is $n\delta_I$-empty. Let d be another site, and choose an interior point p of triangle T_0 so that segment pd intersects no site besides d. See figure 11. Segment pd is contained in the triangulation, since the triangulation is convex. Consider the triangles $T_0, T_1, \ldots, T_k$ intersected by segment pd in intersection order; it is easy to check $k \leq n$. For $i<k$ label the vertices of T_i as a_i, b_i, c_i so that $\Delta a_i b_i c_i$ is positive and $a_i c_i$ is the edge common with T_{i+1}. The lemma follows from the claim that $\mu(a_i, b_i, c_i, d) < (k-i)\delta_I$. The proof is by downwards induction on i, $i{=}k{-}1, k{-}2, \ldots, 0$: the base case follows from the fact that $\mathcal{T}'$ is approximately Delaunay; the induction step is an immediate application of lemma 5. □

5. Flipping

Let $\mathcal{T}$ be a triangulation of a set of n sites. Suppose triangles Δabc and Δacd of $\mathcal{T}$ together form a convex quadrilateral. A *diagonal flip* replaces Δabc and Δacd with Δabd and Δbcd, in effect replacing diagonal ac with diagonal bd. The diagonal flip is a *Delaunay diagonal flip* if d is inside C_{abc}. At most $O(n^2)$ Delaunay diagonal flips can be performed before triangulation $\mathcal{T}$ becomes the Delaunay triangulation, and no flip is possible. One way to see this is to define the weight of triangulation $\mathcal{T}$ as the sum over all triangle circumcircles of the number of sites in the interior of the circumcircle. The weight of $\mathcal{T}$ can be at most $O(n^2)$, and it is not hard to see that the weight of $\mathcal{T}$ decreases by 2 with each Delaunay diagonal flip.

```
Let {s_1, ..., s_n} be sites in lexicographic order.
Build an initial triangulation T:
   Add edge s_1 s_2.
   for i = 3 to n
      Add an edge from s_i to s_{i-1}.
      Walk clockwise from s_{i-1} along bounding edges of T, adding the
         triangle formed by an edge and s_i as long as orient guarantees that
         the triangle is positive.
      Repeat with counterclockwise walk from s_{i-1}.
Perform diagonal flips in T:
   Initialize a candidate edge queue Q
      with all internal edges of T.
   while Q is not empty
      Remove edge ac from Q.
      Let Δabc and Δacd be incident to edge ac.
      if orient(b, d, a) = 1 and orient(b, c, d) = 1
        and incircle(a, b, c, d) = 1
        then Flip ac to bd.
              Add edges ab, bc, ad, bc to Q if not already present.
```

Figure 12: Diagonal flipping algorithm

Figure 12 contains a diagonal flipping algorithm implemented with the geometric primitives *orient* and *incircle*.

Theorem 3 *The diagonal flipping algorithm produces an approximate Delaunay triangulation of n point sites in time* $O(n^2)$.

Proof. Let $\mathcal{T}$ be the triangulation produced by the algorithm. $\mathcal{T}$ is an almost-convex triangulation, since it is after initialization and the only update is to flip a diagonal. Since only Delaunay diagonal flips are performed, the number of flips is $O(n^2)$. To see that the final triangulation is approximately Delaunay, suppose ac is an internal edge with incident triangles Δabc and Δacd. Then either ac results from a previous Delaunay diagonal flip of quadrilateral $abcd$ or at some point edge ac was chosen from the candidate edge queue but the diagonal of $abcd$ was not flipped to bd. In the first case d is outside C_{abc} from the analysis of *incircle*; in the second case either $orient(b, d, a)$, $orient(b, c, d)$ or $incircle(a, b, c, d)$ returned 0 or -1; in all cases $D(a, b, c, d)$ follows. □

6. Other algorithms

Other 2-d exact-arithmetic Delaunay triangulation algorithms—divide-and-conquer, incremental, random-incremental—also use the orientation test and the incircle test as geometric primitives. They can be implemented in approximate arithmetic using *orient* and *incircle*. However, the correctness proof does not extend directly to these algorithms: the approximate nature of *orient* and *incircle* causes subtle difficulties in the maintenance of a triangulation and in the maintenance of the D predicate; illustrative examples appear below.

Even without a complete correctness proof, approximate-arithmetic implementations of the other algorithms are likely to be reliable enough to be useful. It is easy for an implementation to certify that its output is an approximate Delaunay triangulation: use *orient* to certify that each triangle is positive and that the boundary chain is almost-convex, and use *incircle* to certify that the D predicate holds for each pair of adjacent triangles (indeed, the same primitive tests can certify that the output is the true Delaunay triangulation). In some instances, the certification may fail; the discussion below suggests that such instances are rare and suggests possible heuristic corrections.

Adding a new site. An apparently simple extension to the flipping algorithm is to make it online, so that an arbitrary site can be added to an existing approximate Delaunay triangulation. However, it is not always clear how to refine an existing triangulation with a new vertex. Consider adding a new site s to the partial triangulation in figure 13. It is possible that *orient* reports that s is below ac and above ad and db, but *orient* cannot resolve whether s is above or below ab or bc, i.e. it reports both $S(a, s, b)$ and $S(s, b, c)$. An insertion algorithm might choose to treat s as in Δadb, in Δabc, or on segment ab; in any case segments as and sb are created. However, it is not possible to add either diagonal ab or sc to quadrilateral $asbc$ and guarantee that only positive triangles result. (The flipping algorithm avoids this difficulty using the initial sort: in this case, the triangulation boundary would contain edges as, sb and bc; edge ac would never get added.)

One solution to this problem is to use an exact 2-d orientation test. Given a new site s, the triangle containing s can be found and split into three using s. Then the diagonal-flipping procedure can produce an approximate Delaunay triangulation. The exact orientation test requires arithmetic with precision approximately double the input precision, but certainly less than the approximately quadruple precision required for an exact incircle test.

A second solution is possible, in the spirit of using approximate arithmetic to produce an approximate answer. We allow an "approximately embedded triangulation": we maintain the graph structure of the triangulation by purely combinatorial means and allow a nonplane embedding of its vertices and edges (embedded edges may cross or the embedding may not preserve the orientation of triangles). Thus in the the situation in figure 13, we could allow either diagonal sc or ab, even though we cannot guarantee that all resulting embedded triangles are positive. This solution is sketched elsewhere[6,7]; technical issues to solve include how to constrain the embedding so that point location is still possible. The resulting Delaunay triangulation algorithm has worst-case running time $O(n^2)$ to add a total of n sites. Unfortunately, the algorithm is intricate–especially the details of the geometric primitives–so we do not include it here.

A third solution is possible that maintains the usual notion of a triangulation and uses just the approximate-arithmetic test *orient*. Observe that the difficulty arises when the new site s to be added lies approximately on some edge of a triangle Δ that is close to flat, i.e., has largest angle very close to π. Because the triangulation is an approximate Delaunay triangulation, the triangle Δ' opposite the longest edge of Δ is also close to flat, as is the triangle Δ'' opposite the longest edge of Δ', and so forth. The deletion of a sequence of longest edges results either in a region that can be retriangulated from the vertex opposite the last-deleted edge, or in the deletion of a boundary edge, in which case a chain becomes the new triangulation boundary. Thus in figure 13(a), it suffices to delete boundary edge ac, leaving as, sb, and bc on the boundary; possibly there might have been a triangle Δacd opposite edge ac so that after deleting ac, region $asbcd$ could be triangulated into Δasd, Δsbd, and Δbcd. A careful argument is necessary to show that such retriangulation is always possible. After retriangulating, the Delaunay triangulation can be obtained by diagonal flipping, as before. One difficulty with this approach is that the only apparent bound on the worst-case time required to add n sites is $O(n^3)$: while the retriangulation takes only linear time, possibly $O(n^2)$ diagonal flips are necessary after the retriangulation. (The retriangulation to add a new site may increase the weight of the triangulation, as defined in section 5, by $\Omega(n^2)$–with exact arithmetic adding a new site would only increase weight by $\Omega(n)$.)

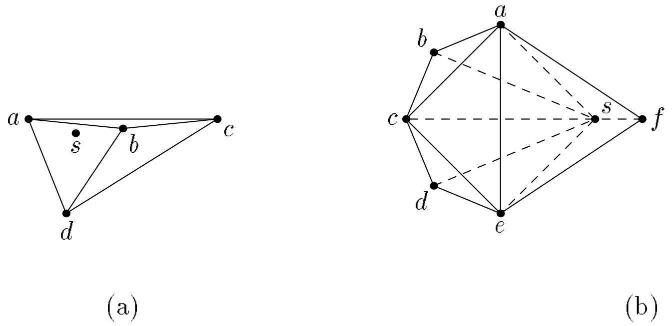


Figure 13: Nasty cases.

Maintaining the D predicate. Let $\mathcal{T}$ be a Delaunay triangulation, and let $\mathcal{T}'$ be the Delaunay triangulation that results by adding a new site s. It is easy to see that all triangles in $\mathcal{T}' - \mathcal{T}$ are incident to s. However, a corresponding property does not hold for approximate Delaunay triangulations. Consider the configuration in figure 13(b). Points a, c, e, s are very close to cocircular, with s very slightly inside C_{ace}; points b and d are also inside C_{ace} but with $\mu(a, b, c, e) = \delta_I = \mu(a, c, d, e)$, so the solid edges form an approximate Delaunay triangulation of a, b, c, d, e, f. Naively, the dashed edges form an approximate Delaunay triangulation that includes site s. However, it can happen that $\mu(s, b, c, d) \approx 2\delta_I$, so $D(s, b, c, d)$ does not hold.

A naively-coded incremental or random-incremental algorithm might produce the dashed configuration in figure 13(b); a similar configuration can arise from the merge step of the divide-and-conquer algorithm. A simple way to guarantee an approximate Delaunay triangulation is to use the flipping algorithm as a post-processing step; in figure 13(b) it suffices to flip sc to bd. However, in the worst case, flipping might add $O(n^2)$ to the running time.

It is possible that such a post-processing step is unnecessary, and the global $O(n\epsilon)$ error bound is achieved without it. In a sense the global error of the dashed triangulation in figure 13(b) is not much bigger than the global error of the solid triangulation. Notice that in the solid triangulation without s, we already have $\mu(a, b, c, e) \approx 2\delta_I$; adding site s in effect just concentrated this error on edge sc. Perhaps there is some amortization argument that shows a good global bound on error, even though local error may not be constant.

7. Discussion

The challenge is to develop floating-point analyses of a wide variety of geometric algorithms. Such analyses should lead to efficient, reliable, easily-implemented geometric algorithms.

A general theory, easily applicable to an arbitrary algorithm, appears to be a distant goal. An important component of a general theory would be a way of transforming exact predicates into approximate predicates. Such a transformation must have two properties: first, the truth of approximate predicates has to be testable using floating-point arithmetic; second, reasoning using approximate predicates has to adequately mimic reasoning using exact predicates. An example is the transformation of the 2-d incircle predicate into the D predicate. This transformation satisfies the first property, since *incircle* can be used to test the D predicate; it does not entirely satisfy the second property, as the discussion in section 6 indicates.

References

1. C.B. Barber, Computational geometry with imprecise data and arithmetic, Ph.D. thesis, Princeton U. 1992, available as Tech. Report. CS-TR-377-92.
2. K.L. Clarkson, P.W. Shor, Applications of random sampling in computational geometry, II, *Disc. Comp. Geom.* **4**:387-421, 1989.
3. G. Dahlquist, Å. Björck, N. Anderson, *Numerical Methods*, Prentice-Hall, 1974.
4. M. Dillencourt, W. Smith, A linear-time algorithm for testing the inscribability of trivalent polyhedra, *Proc. Eighth ACM Symp. Comp. Geom*, 177–185, 1992.
5. S. Fortune, A sweepline algorithm for Voronoi diagrams. *Algorithmica* 2:153-174 (1987).
6. S. Fortune, Stable maintenance of point-set triangulation in two dimensions. An abbreviated version appeared in *30th FOCS*, pp. 494–499, 1989.
7. S. Fortune, Computational Geometry, *Directions in Geometric Computing*, Information Geometers, R. Martin, ed., to appear, 1992.
8. S. Fortune, Numerical Stability of algorithms for 2D Delaunay triangulations, *Proc. Eighth Ann. Symp. Comp. Geom.*, 83–92, 1992.
9. S. Fortune, V. Milenkovic. Numerical stability of algorithms for line arrangements. *Proc. Seventh Ann. Symp. Comp. Geom.*, 334–341, 1991.
10. P. Green, R. Sibson. Computing Dirichlet tessellations in the plane. *Computing J.* 21:168–173, 1977.
11. L. Guibas, J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4:74–123, 1985.
12. C. Hoffmann. The problems of accuracy and robustness in geometric computation. *Computer* 22:31–42, 1989.
13. C. Hoffmann, J. Hopcroft, M. Karasick, Robust set operations on polyhedral solids, *IEEE Comp. Graph. Appl.* **9**(6):50–59, 1989.
14. M. Karasick, D. Lieber, L. R. Nackman, Efficient Delaunay triangulation using rational arithmetic, *ACM Trans. Graphics* **10**(1):71–91, 1990.
15. C.L. Lawson, Software for C^1 surface interpolation, *Mathematical Software III* 161-194, J.R. Rice, ed., Academic Press, 1977.
16. Z. Li, V. Milenkovic, Constructing strongly convex hulls using exact or rounded arithmetic, *Proc. Sixth Ann. Symp. Comp. Geom.* 235–243, 1990.
17. V. Milenkovic. Verifiable implementations of geometric algorithms using finite precision arithmetic. *Artificial Intelligence*, 37:377–401, 1988.
18. V. Milenkovic. *Verifiable implementations of geometric algorithms using finite precision arithmetic.* Ph.D. thesis, Carnegie-Mellon, 1988.
19. I. Rivin, W. Smith, Inscribable graphs, manuscript, NEC Research Institute, 1991.
20. M. Shamos, D. Hoey, Closest-point problems, *Proc. 16th Ann. Symp. Found. Comp. Sci.* 151-162, 1975.
21. K. Sugihara, M. Iri, Geometric algorithms in finite-precision arithmetic, Research Memorandum RMI 88-10, University of Tokyo, September, 1988.
22. K. Sugihara, M. Iri, Construction of the Voronoi diagram for one million generators in single precision arithmetic, First Canadian Conference on Computational Geometry, Montreal, Canada, 1989.