A certified algorithm for the InCircle predicate among ellipses

Ioannis Z. Emiris* Elias P. Tsigaridas* George M. Tzoumas*

Abstract

This paper examines the InCircle predicate among ellipses in the Euclidean plane, under the exact computation paradigm. The ellipses are non-intersecting and given in parametric representation. We present a subdivision-based algorithm and implement it in Maple and CORE.

1 Introduction

We study the InCircle predicate for the Voronoi diagram of ellipses. This is the hardest predicate for implementing the algorithm of [7] and has not been solved in the exact computation paradigm. The work coming closest to ours is [5]: The authors essentially trace the bisectors in order to compute the Voronoi cells of arbitrary curves up to machine precision. Their algorithm uses floating point arithmetic; they claim that their software works well in practice. Although they argue that their algorithm can be extended to exact arithmetic, they do not explain how. For instance, they do not discuss degenerate configurations. Our implementations are exact but can also run with any prescribed precision.

It seems hard, for the algebraic approach of [3], to yield a fast solution. All four predicates of the incremental algorithm [7] were studied in [4], including a certified subdivision-based algorithm for InCircle, implemented in Maple. In this paper, we offer a better implementtion using the CORE library [6]. The algorithm "moves" on the border of parametrically defined ellipses. This avoids computing the Voronoi circle explicitly.

The Voronoi circle is specified at any desired accuracy. This is achieved by refining the interval expressing its 3 tangency points until the predicate can be decided; in fact, all tangency points are expressed as a function of one of them. Exactness is guaranteed by root separation bounds.

Let the length of the axes be $2\alpha, 2\beta$ and (x_c, y_c) be the center. We use the parametric representation:

$$\begin{aligned} x(t) &= x_c - (\alpha(1-w^2)t^2 + 4\beta w t - \alpha(1-w^2))/d \\ y(t) &= y_c + 2(-\alpha w t^2 + \beta(1-w^2)t + \alpha w)/d, \end{aligned}$$

*Dept. of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece, {emiris,et,geotz}@di.uoa.gr

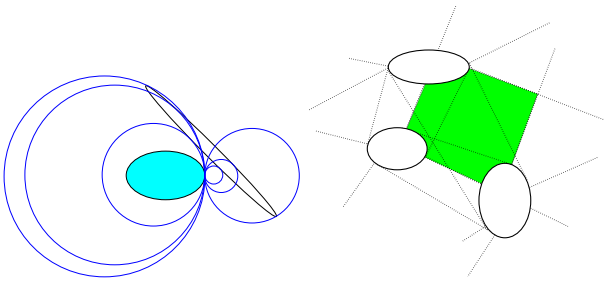

Figure 1: Left: The 6 bitangent circles. The Apollonius circle is the 4^{th} from the left. Right: Starting intervals for t, r, s (and region of the Voronoi vertex)

where $d = (1+w^2)(1+t^2)$, $t = \tan(\theta/2) \in (-\infty, \infty)$, θ is the angle that traces the ellipse, $w = \tan(\frac{\phi}{2})$, and ϕ is the rotation angle between the major and horizontal axes. We denote by E_t an ellipse parameterized by t.

2 The bitangent circle

Lemma 1 *Given 2 ellipses and a point on the first, there may exist up to 6 real bitangent circles, tangent at the specific point. This bound is tight. Only one such circle is* external *to both ellipses.*

We call this unique *external* bitangent circle the *Apollonius* circle of the 2 ellipses, e.g. the third circle from the right in fig. 1 (left).

Given ellipses E_t, E_r, the tangency points of any Apollonius circle lie inside their Convex Hull (CH). This offers a starting point to begin our search for the tangency point of the Voronoi circle within a *continuous* range on the boundary of an ellipse. Now, consider all bitangent circles to E_t, E_r, tangent at point t of E_t.

We shall compute arc (r_1, r_2) on E_r which contains *only* the tangency point of the Apollonius circle, *isolating* it from the tangency points of non-external bitangent circles. Consider all bitangent circles at t. Also, consider the lines from t tangent to E_r at points r_1, r_2. They define two arcs on E_r. Arc (r_1, r_2), whose interior points lie on the same side of line $r_1 r_2$ as t, is called a *visible arc*.

Visible arc (r_1, r_2) contains only tangency points of bitangent circles at t, which are externally tangent to E_r, but may be internally tangent to E_t. They include the Apollonius circle of E_t, E_r, tangent at $t \in E_t$.

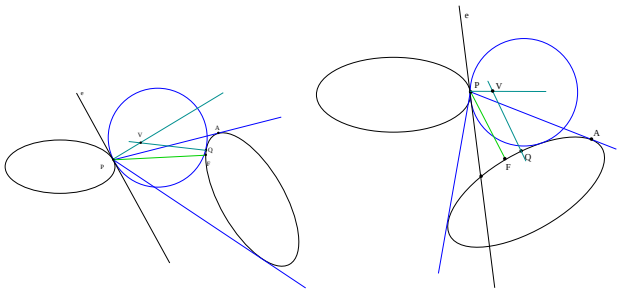

Figure 2: The visible arc and the Apollonius circle

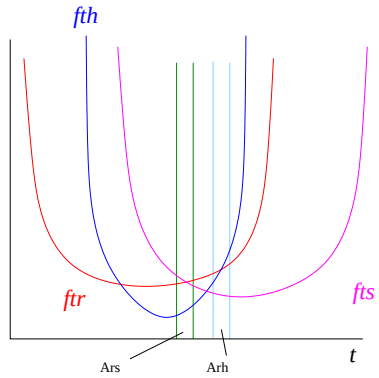

Figure 3: Deciding the predicate

Lemma 2 *Given is a point $P = (x(t), y(t))$ on E_t. Consider line ϵ, tangent at P (cf. fig. 2). If ϵ does not intersect E_r, then the visible arc contains a unique Apollonius circle. Otherwise, the endpoints of such an arc are: the intersection of ϵ with E_r and the endpoint of the visible arc which lies on the opposite side of E_t with respect to ϵ.*

Given ellipses E_t, E_r their bisector $B(t, r)$ is a bivariate polynomial of degree 6 in t and 6 in r. The above lemma provides an isolating interval for the unique root $\hat{r}$ of $B(t, r)$, which lies on the visible arc of E_r with respect to some fixed t. In other words, $\hat{r}$ corresponds to the Apollonius circle.

Given a point $(x(t), y(t))$ on E_t, the *squared radius* of the Apollonius circle of E_t, E_r tangent to E_t at that point is denoted by $f_{tr}(t)$. It follows that

$$f_{tr}(t) := \big(v_1(t, \hat{r}) - x(t)\big)^2 + \big(v_2(t, \hat{r}) - y(t)\big)^2,$$

where $\hat{r}$ is the root of the bisector that corresponds to the Apollonius circle, when we fix t, and (v_1, v_2) is the intersection of the normals at t and $\hat{r}$.

In the sequel, we assume that $f_{tr}(t)$ is defined on a continuous interval (a, b). The interval can also be of the form $(-\infty, a) \cup (b, \infty)$, but in this case the problem is identical or easier.

Lemma 3 *Function $f_{tr}(t)$ consists of two strictly monotone parts, one decreasing and one increasing.*

We have not proved the function's convexity, though this is implied by numerical examples.

Our overall algorithm maintains an interval that contains the tangency on E_t. At every iteration, it picks some value for t within the interval and solves $B(t, r)$ for finding $\hat{r}$, which was defined above. These values are used to determine the sign of $\mathcal{S}$, to be introduced below.

3 Deciding the predicate

The Voronoi circle is the circle which is externally bitangent to E_t, E_r, and E_t, E_s at the same time. Its tangency point on E_t is defined by the condition:

$$\mathcal{S}_{trs}(t) = 0, \text{ where } \mathcal{S}_{trs}(t) = f_{tr}(t) - f_{ts}(t),$$

where f_{tr} (and similarly f_{ts}) in the case of ellipses becomes

$$f_{tr}(t) = \frac{1}{4} P_t(t) \left(\frac{A_{tr}(t, \hat{r})}{(1+t^2)(1+\hat{r}^2) D_{tr}(t, \hat{r})} \right)^2.$$

In the above equation, $P_t(t)$ has no real roots, A_{tr} is a bivariate polynomial of degree 2 in t and 4 in r and $D_{tr} \neq 0$, unless the normals at $t, \hat{r}$ are parallel.

We factor $\mathcal{S}_{trs}(t)$ as follows:

$$\frac{P_t(t) \cdot \big[Q_1(t, \hat{r}, \hat{s}) - Q_2(t, \hat{r}, \hat{s})\big] \cdot \big[Q_1(t, \hat{r}, \hat{s}) + Q_2(t, \hat{r}, \hat{s})\big]}{4\big[(1+t^2)(1+\hat{r}^2)(1+\hat{s}^2) D_{tr}(t, \hat{r}) D_{ts}(t, \hat{s})\big]^2} \quad (1)$$

We use a customized bisection to approximate a root of $\mathcal{S}_{trs}(t)$. We only need to determine the sign $Q_1 - Q_2$ and $Q_1 + Q_2$, since the rest of the terms in (1) are always positive. This way we avoid computing f explicitly.

We express the Voronoi circle of E_t, E_r, E_s by an interval containing t, such that $(x(t), y(t))$ is the tangency point on E_t.[1] We start by the initial interval $[a, b]$ that contains the tangency point and subdivide it by bisection. The subdivision operator yields

$$\begin{array}{ll} [\frac{a+b}{2}, \frac{a+b}{2}], & \text{if } \mathcal{S}_{trs}(\frac{a+b}{2}) = 0, \\ [a, \frac{a+b}{2}], & \text{if } \mathcal{S}_{trs}(a)\mathcal{S}_{trs}(\frac{a+b}{2}) < 0, \\ [\frac{a+b}{2}, b], & \text{otherwise.} \end{array}$$

Theorem 4 *Let $x \in [a, b]$ be the root of $\mathcal{S}_{trs}(x)$. If $\mathcal{S}_{trh}(x) > 0$, then E_h intersects the Voronoi circle of E_t, E_r, E_s. If $\mathcal{S}_{trh}(x) < 0$, then E_h lies outside the Voronoi circle. Otherwise, E_h is externally tangent to this circle.*

[1] This interval might contain tangency points of other non-external tritangent circles, but they don't interfere with our approach, since it deals only with externally bitangent circles.

Note that there is no such case such as internal tangency. This is due to the fact that we deal only with externally tangent circles.

Clearly, there is a neighborhood $\mathcal{U}$ of x where $\text{sgn}(S_{trh}(u)) = \text{sgn}(S_{trh}(x)),\ \forall u \in \mathcal{U}$. In our implementation, to find $\mathcal{U}$, it will suffice to separate the roots of $\mathcal{S}_{trs}, \mathcal{S}_{trh}$.

We now establish the *exactness* of our algorithm. Consider system $\Delta_1(v_1, v_2, q) = \Delta_2(v_1, v_2, q) = \Delta_3(v_1, v_2, q) = q - v_1^2 - v_2^2 + s = 0$ [4], where (v_1, v_2) is the center and s the squared radius of the Voronoi circle. Let us eliminate v_1, v_2, q; the resultant $R(s)$ is of degree 184 in s and has coefficient bit size $3 \cdot 56 \cdot \tau_\Delta = 168\tau_\Delta$. Here 56 equals the mixed volume of the system $\Delta_i, \Delta_j, q - v_1^2 - v_2^2 + s$, if we consider s as a parameter, and τ_Δ denotes the bit size of the coefficients of Δ_i, where $1 \leq i, j \leq 3$ and $i \neq j$.

The minimum distance between two real roots of a polynomial P of degree d and bit size τ is $sep(P) \geq d^{-(d+2)/2}(d+1)^{(1-d)/2}2^{\tau(1-d)}$ [9], thus the number of bits that we need in order to compute s is no more than $1389 + 30744\,\tau_\Delta$.

In order to compare two radii s_1 and s_2, which are roots of polynomials R_1 and R_2 respectively, we need a bound for $|s_1 - s_2|$. Since $|s_1 - s_2| \geq sep(R_1R_2)$, where the polynomial R_1R_2 has degree 368 and coefficient bit size $8+336\tau_\Delta$, it follows that the number of bits needed is $1508 + 30324\tau_\Delta$. This is tight, because the system has optimal mixed volume [3].

In computing the implicit representation the bit size increases by a factor of 6. If the parametric input coefficients have τ bits, then $\tau_\Delta = 6\tau$. If the order of convergence of our method is ϕ, then the number of iterations needed is $\log_\phi(1508 + 181944\tau)$.

4 Implementation and experiments

A reference implementation with parametric ellipses has been done in Maple 9. We have implemented a small algebraic number package that performs exact univariate real root isolation, comparison and sign evaluation of univariate (bivariate) expressions over one (two) algebraic number(s), using Sturm sequences and interval arithmetic over $\mathbb{Q}$.

We speed up the subdivision by noticing that $\mathcal{S}_{trs}$ is strictly monotone in the starting interval $[a, b]$ and has a unique simple real root in it. So we use Brent's method with theoretical convergence rate $\phi = 1.618$ [1]. Let $[a, b]$ be any interval and $m = \frac{a+b}{2}$. The new endpoint is $x = m + \frac{P}{Q}$, where $R = \frac{\mathcal{S}_{trs}(m)}{\mathcal{S}_{trs}(b)}$, $S = \frac{\mathcal{S}_{trs}(m)}{\mathcal{S}_{trs}(a)}$, $T = \frac{\mathcal{S}_{trs}(a)}{\mathcal{S}_{trs}(b)}$, $P = S(T(R-T)(b-m) - (1-R)(m-a))$ and $Q = (T-1)(R-1)(S-1)$. If $x \notin [a, b]$ then the new estimation is m. However, in practice we observe a fast growth of bitsize, if we use this method with an exact number type, i.e. rationals.

Based on this reference implementation in Maple, we implemented the algorithm in C++ using CORE. While still in a preliminary stage, it is faster than the Maple implementation. In this case it is possible to certify our algorithm based on constructive root separation bounds e.g. [2, 10], which should be tighter than the static bounds now used. In this implementation, the input quantities are of type BigRat (rational numbers), while the endpoints of the interval that expresses the Voronoi circle are BigFloats (multiprecision floating point). Evaluation of $Q_1 \pm Q_2$ in (1) is performed using CORE::Expr constructs after converting t from BigFloat to BigRat. This, along with $B(t, r)$ seem to be the most heavy computations, due to growth of bitsize. In a future improvement we will use guaranteed precision arithmetic with BigFloats. Another improvement will be to use information from previous iterations (where t has fewer correct bits) in order to solve $B(t, r)$ and determine the sign of Q_1 and Q_2.

We performed several preliminary experiments with different triplets of ellipses and circles. We consider a query ellipse (circle) with its centre moving along a line and measure the time taken to decide its relative position wrt the Voronoi circle. Among the various configurations, there were both degenerate and non-degenerate cases.

We did our experiments on a P4 2.6GHz. In fig. 4 we present the times for 2 test suites, where the ellipses have 10-bit coefficients in their parametric form. The first graph involves ellipses that do not share a common Voronoi circle with the query one, whose center moves along the line $y = -x$ (fig. 5 left). Notice that the time increases as we approach a degenerate configuration. Although the hardest cases took about 5s, in 90% of the cases we can decide in less than 2.5s. The C++ implementation without any compiler optimizations is 2 times faster than Maple. The second graph involves circles (fig. 5 right). The peak corresponds to nearly degenerate configurations, running in 38s and 200 iterations. In all other cases the timings are less than 2.5s. Again, the C++ implementation is 3-4 times faster.

Our implementations are exact but can also run with any prescribed precision, e.g. for rendering purposes. In particular, a much faster execution is possible for the above algorithms if we restrict ourselves to machine precision, as in [5].

Solving the algebraic system $B(t, r) = B(t, s) = B(r, s) = 0$ with the SYNAPS [8] package of multivariate subdivision in 3 msec to 1 min, depending on how large the initial domain was. Moreover, we used PHCpack, to solve the system of Δ_i's, in about 36 seconds. These running times indicate that our dedicated solver sometimes outperforms generic solvers on the algebraic system. Moreover, solving the algebraic system alone does not suffice to completely decide the predicate, contrary to our algorithm.

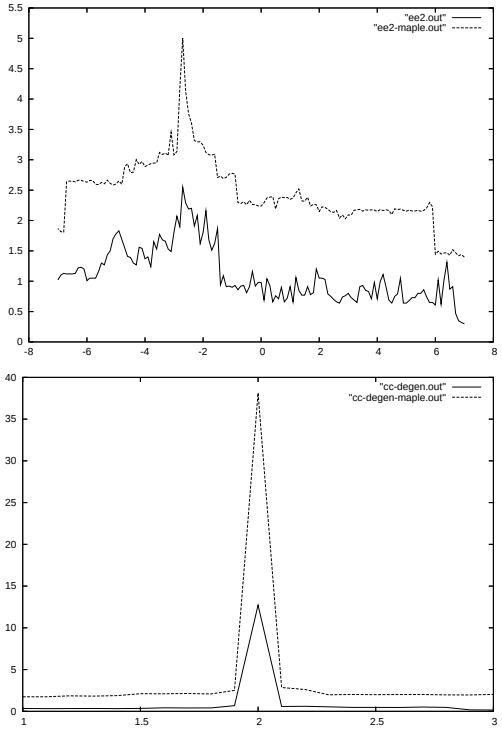

Figure 4: Execution time as function of the position of the query ellipse's (circle's) center. The solid line corresponds to C++, the dotted one to Maple.

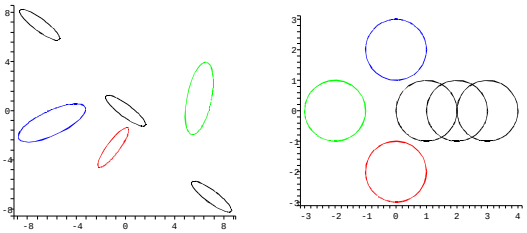

Figure 5: Test suites

5 Future work

Our final goal is a CGAL implementation of the Voronoi diagram of ellipses. Working in C++ may allow us to use one of the powerful interval arithmetic packages. We tried the iCOs interval-arithmetic solver [2] on the system of Δ_i's with bitsize 60. It detects a degeneracy in about 213 sec on a 1GHz P3.

The most difficult part of the implementations is the detection of a degeneracy. Although near-degenerate inputs can be handled quite efficiently, real degeneracies exploit the separation bound and need a large number of iterations. Then, the computed quantities grow too large. We are currently trying to optimize the inner loop. In this direction, we are looking for better ways to perform certified sign computation of $Q_1 \pm Q_2$, as well as better separation bounds using geometric arguments and exploiting the algebraic approach.

[2] http://www-sop.inria.fr/coprin/ylebbah/icos/

Acknowledgments All authors acknowledge partial support by IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-006413-2 (ACS - Algorithms for Complex Shapes) and by PYTHAGORAS, project 70/3/7392 under the EPEAEK program funded by the Greek Ministry of Educational Affairs and EU. GT is partially supported by State Scholarship Foundation of Greece, Grant No. 4631.

References

[1] R. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, N.J., 1973.

[2] C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, and S. Schmitt. A Separation Bound for Real Algebraic Expressions. In *ESA*, volume 2161 of *LNCS*, pages 254–265. Springer, 2001.

[3] I. Emiris and G. Tzoumas. Algebraic study of the Apollonius circle of three ellipses. In *Proc. Europ. Works. Comp. Geom.*, pages 147–150, Holland, 2005. Also: Poster session, CASC'05, Greece. To appear in *SIGSAM Bulletin*.

[4] I. Emiris, G. Tzoumas, and E. Tsigaridas. The predicates of the Voronoi diagram of ellipses. *Symp. of Comp. Geom.*, 2006. To appear. Available from http://www.di.uoa.gr/∼geotz/.

[5] I. Hanniel, R. Muthuganapathy, G. Elber, and M.-S. Kim. Precise Voronoi cell extraction of free-form rational planar closed curves. In *Proc. 2005 ACM Symp. Solid and phys. modeling*, pages 51–59, Cambridge, Massachusetts, 2005. Best paper award.

[6] V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A CORE library for robust numeric and geometric computation. In *15th ACM Symp. on Computational Geometry*, 1999.

[7] M. Karavelas and M. Yvinec. Voronoi diagram of convex objects in the plane. In *Proc. ESA*, pages 337–348, 2003.

[8] B. Mourrain, J. P. Pavone, P. Trébuchet, and E. Tsigaridas. SYNAPS, a library for symbolic-numeric computation. In *8th Int. Symposium on Effective Methods in Algebraic Geometry, MEGA*, Sardinia, Italy, May 2005. to appear.

[9] C. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, New York, 2000.

[10] C. Yap. On guaranteed accuracy computation. In F. Chen and D. Wang, editors, *Geometric Computation*, volume 11 of *Lect. Notes Series Comp.* World Scientific, 2004.