Exact Computation of Arrangements of Rotated Conics*

Eric Berberich† Manuel Caroli† Nicola Wolpert‡

Abstract

Transformations of geometric objects, like translation and rotation, are fundamental operations in CAD-systems. Rotations trigger the need to deal with trigonometric functions, which is hard to achieve when aiming for exact and robust implementation.

We show how we efficiently compute the planar arrangement of conics rotated by angles that can be constructed with straightedge and compass. Well-known examples are multiples of 45°, 30°, and 15°. The main problem one has to solve is root-isolation of univariate polynomials $p(x) \in \mathbb{Q}(\sqrt{c_1})\dots(\sqrt{c_d})[x]$, for which we use a modified version of the Descartes method. For $d = 1$, we additionally present a new method that isolates the real roots of p by using root isolation for polynomials $q(x) \in \mathbb{Q}[x]$ only. We show results of our benchmark experiences comparing both methods.

1 Introduction

We construct a set of transformed conics C' by applying an individual sequence of translations and rotations to each conic of a given set C. We aim to compute the subdivision of the plane induced by C' into cells of dimension 0 (*vertices*), of dimension 1 (*edges*), and of dimension 2 (*faces*) — commonly referred to as the *arrangement* of C'. Arrangements are well-studied in the field of computational geometry and serve as a basis for different applications [12]. In the past years, research concentrated on finding exact, robust, and efficient solutions to compute the arrangement of non-linear objects. Emiris et al. [11] concentrated on circles, while arrangements of conics have been considered in CGAL[1] by Wein [16] and in EXACUS[2] by Berberich et al. [3], all using a modified version [14] of the sweep-line algorithm [1].

Performing arbitrary rotations requires evaluation of trigonometric functions. Canny et al. [5] introduced a rotation scheme to approximate rotations by α with angles $\bar{\alpha}$ whose sine and cosine are rational. Such angles $\bar{\alpha}$ are dense in $[0, 2\pi)$, which can be be easily seen by the parameterization $(\frac{2t}{1+t^2}, \frac{1-t^2}{1+t^2})$ of the unit circle. Each desired angle can be arbitrarily approximated by existing implementations. But this method leads to increased bitlengths of the involved coefficients and still does not compute the exact solutions as expected by the *exact geometric computation* paradigm [18]. Each exact solution recalls the question whether trigonometric computations can be made "geometrically exact" – first tackled by [7]. We give a positive answer for rotations by angles that can be constructed with straightedge and compass. Such rotations are discussed in Section 2. We explain in Section 3 how to compute the intersection points of such rotated conics and present details of the new implementation within EXACUS in Section 4. This work concludes in Section 5 with a selection of experiments.

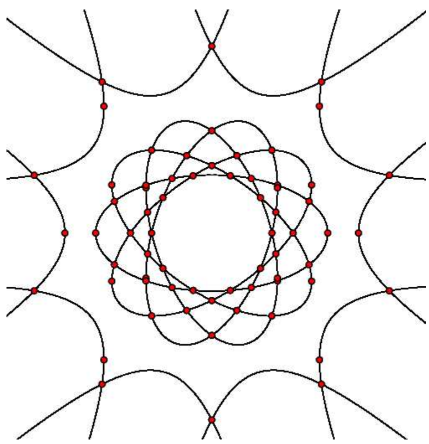

Figure 1: The arrangement of a hyperbola and an ellipse, both rotated by angles of 0°, 36°, 72°, 108°, and 144°.

2 Transformations of Conics

Definition 1 (Conic) *Let $\mathbb{K}$ be a field and $f(x, y) \in \mathbb{K}[x, y]$ a bivariate polynomial. If $\deg(f) \leq 2$ we call the zero set $\mathcal{V}(f) := \{(x, y) \mid f(x, y) = 0\}$ a* conic *and denote it by c.*[3]

A usual choice is $\mathbb{K} = \mathbb{Q}$. Since computations with integers are much faster than using rational arithmetic, we always multiply rational coefficients by their common denominator keeping the curve unchanged.

*This work has been supported in part by the IST Programme of the EU as Shared-cost RTD (FET Open) Project under Contract No IST-006413 (ACS - Algorithms for Complex Shapes)

†Max-Planck-Institut für Informatik, Saarbrücken, Germany, {eric, mcaroli}@mpi-inf.mpg.de

‡University of Applied Sciences, Stuttgart, Germany nicola.wolpert@hft-stuttgart.de

[1] http://www.cgal.org/

[2] http://www.mpi-inf.mpg.de/projects/EXACUS/

[3] For convenience, we also use c to denote the polynomial f.

Definition 2 (Euclidean Transformations) *Let* $\boldsymbol{x}, b \in \mathbb{K}^n$*, and let* $A \in \mathbb{K}^{n\times n}$ *an orthogonal* $n \times n$*-matrix. Transformations of the form* $\mathcal{E}(\boldsymbol{x}) := A\boldsymbol{x} + b$ *are called* Euclidean Transformations*.*

We aim for the coefficients of the new conic $\bar{c}$ when applying $\mathcal{E}$ on c, i. e., for each $(x, y) \in \mathcal{V}(c)$ we want that $\mathcal{E}(x, y) \in \mathcal{V}(\bar{c})$. This condition is fulfilled if we choose $\bar{c} := c \circ \mathcal{E}^{-1}$. Since A is orthogonal we get $\mathcal{E}^{-1}(x, y) = A^T\binom{x}{y} - A^T b$. Let us consider rotations by an angle α. Then

$$A = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

If the entries of A are rational we can use the existing implementations. Otherwise the entries of A are non-rational and in most cases they are even transcendental. For some angles α we have $\sin(\alpha)$ and $\cos(\alpha) \in \mathbb{Q}(\sqrt{c_1})\ldots(\sqrt{c_d})$, for a constant d and all $0 < c_i \in \mathbb{Q}$. This holds for all angles that can be constructed by straightedge and compass, e. g., multiples of $\alpha = \frac{2\pi}{k}$, with $k = 2^n F_1 \ldots F_l$, $l, n \in \mathbb{N}$, where the F_i are Fermat primes. Fermat primes are of the form $F_i = 2^{2^{e_i}} + 1$ for some natural $e_i \neq e_j$. Since the Fermat primes grow very fast, it is clear that we can get very small angles, e. g., $1.5^\circ = \frac{360^\circ}{2^4 F_0 F_1}$. Note that angles can be halved by straightedge and compass (one additional square root). In the following sections we only consider angles for rotations constructible by straightedge and compass. Observe that the angle $\alpha = 1^\circ$ is not constructible this way [15].

3 Intersection Points of Rotated Conics

Newer implementations of the sweep-line algorithm require a set of basic geometric predicates on the curves to be swept [17]. It even can be reduced to the topological analysis of single curves and pairs of curves [2]. A basic step is to find all x-coordinates of intersection points of two curves. These are usually computed by real root isolation of a univariate polynomial p that is obtained by a resultant computation [19]. In case of conics $\deg(p) \leq 4$. Real root isolation means to determine for every real root of p a (rational) interval $[l, r]$, that contains exactly one root of p. A well-known technique for real root isolation is the Descartes method [8, 10] that we adapted for our purpose. Although intended for integral polynomials, it is also applicable to polynomials with non-rational coefficients, as in our case $p \in \mathbb{Q}(\sqrt{c_1})\ldots(\sqrt{c_d})[x]$. If only one root is adjoined, we additionally explore another technique to isolate the real roots:

Theorem 1 *The roots of a polynomial* $p \in \mathbb{Q}(\sqrt{c})[x]$ *with* $\deg(p) \leq 4$ *can be isolated by only using real root isolation on polynomials* $\bar{p} \in \mathbb{Q}[x]$*.*

We sketch a constructive proof by giving the main ideas of the algorithm. Note that for $p \in \mathbb{Q}(\sqrt{c})[x]$ we have

$$\begin{aligned} p(x) &= \sum_{i=0}^{4}(a_i + \sqrt{c}\cdot b_i)x^i \\ &= \sum_{i=0}^{4} a_i x^i + \sqrt{c}\cdot\sum_{i=0}^{4} b_i x^i \\ &= \alpha(x) + \sqrt{c}\cdot\beta(x) \end{aligned}$$

with $\alpha, \beta \in \mathbb{Q}[x]$. We consider the bivariate polynomial $q(x, u) := \alpha(x) + u\beta(x)$ and our goal is to isolate the real roots of $q(x, \sqrt{c}) = p(x)$. Note that q is linear in u with coefficients in $\mathbb{Q}[x]$. If $\mathrm{con}(q) := \gcd(\alpha(x), \beta(x))$ is constant, $\mathcal{V}(q)$ defines the graph of a function in x. Otherwise, for each root x_i of $\mathrm{con}(q)$ we have: $\forall u\ q(x_i, u) = 0$. More intuitively: q defines an algebraic curve of degree 4 in the xu-plane that either comprises of the graph of a function or the graph of a function multiplied with some vertical lines.

To isolate the roots of $q(x, \sqrt{c})$ we approximate $\sqrt{c}$ by an interval $[s, t]$, $s, t \in \mathbb{Q}$, such that at most $\sqrt{c}$ is a root of $\rho(u) := \mathrm{res}_x(q, \frac{\partial}{\partial x}q)$ in $[s, t]$. We infer the desired isolating intervals from the isolating intervals of $p_s(x) := q(x, s)$ and $p_t(x) := q(x, t)$. Observe that $p_s, p_t \in \mathbb{Q}[x]$. Depending on whether $\rho(\sqrt{c}) = 0$ we have two distinguish two cases:

***p* square-free:** It is easy to see that $\sqrt{c}$ is not a root of $\rho(u)$. The number of roots of p_s and p_t is equal and bounded by 4. Since q is continuous and locally either the graph of a function or a vertical line, and due to the choice of s and t, we have, for each root x_i, $i \leq 4$, $\lim_{s\nearrow\sqrt{c}} q(x_i, s) = q(x_i, \sqrt{c}) = \lim_{t\searrow\sqrt{c}} q(x_i, t)$. The convex hull of properly refined isolating intervals for $q(x_i, s)$ and $q(x_i, t)$ forms the isolating interval of $q(x_i, \sqrt{c})$.

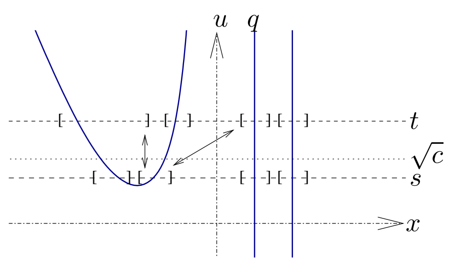

Figure 2: The isolating intervals for $q(x_i, s)$ and $q(x_i, t)$ define the isolating interval for $q(x_i, \sqrt{c})$, after the ones for $q(x_i, s)$ are refined (examples indicated by $\leftrightarrow$) with respect to $q(x_{i-1}, t)$ and $q(x_{i+1}, t)$. Similar for $q(x_i, t)$.

***p* not square-free:** We have $\rho(\sqrt{c}) = 0$ which disables us to infer the desired isolating intervals directly from the isolating intervals of roots of p_s and p_t, since the number of roots may differ. Let

$\sigma(u) := \mathrm{res}_x(\frac{\partial}{\partial x}q, \frac{\partial^2}{\partial x^2}q)$. Both, $\rho(u)'$ and $\sigma(u)$ help to distinguish different cases for the number of roots of p and their multiplicities listed in the given table.

	$\rho'(\sqrt{c})$	$\sigma(\sqrt{c})$	# of multiplicities 1	2	3	4
A	$\neq 0$		≤ 2	1	0	0
B	$= 0$	$\neq 0$	0	2	0	0
C	$= 0$	$= 0$	≤ 1	0	1	0
D			0	0	0	1

Note that in case A and C we still need to assign the correct multiplicity to each root. We omit the full case distinction conducted in [6], e. g., additional criteria that help to distinguish between case C and D, and exemplarily, we present case A.

The double root, denoted by $\bar{\bar{x}}$, may or may not induce roots in p_s and p_t. Consider the case of a vertical line at $\bar{\bar{x}}$. Then the numbers of roots are positive and equal. Let $q_x(x,u) := \frac{\partial}{\partial x}q(x,u)$. Then $(x - \bar{\bar{x}})$ is a double factor of q and a simple factor of q_x that can be computed by means of the greatest common divisor.

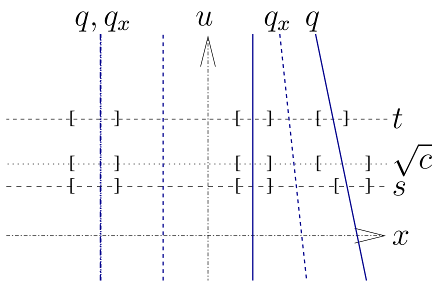

Figure 3: p_s and p_t show 3 roots that do not determine the double root of $p(x, \sqrt{c})$, but $\gcd(q, q_x)$ gives it.

Otherwise, $\bar{\bar{x}}$ induces no root in p_s and two in p_t, or vice versa. It suffices to isolate all roots of p_s with respect to the roots of p_t until we can locate $\bar{\bar{x}}$.

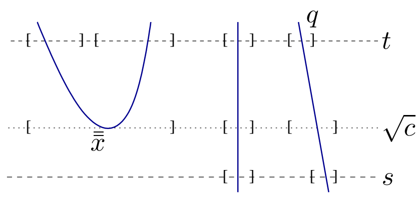

Figure 4: The isolating and cross-refined intervals for roots of p_s and p_t determine the simple roots and the double root $\bar{\bar{x}}$ of $p(x)$.

When analyzing the topology of a pair of conics, we also require to compare y-coordinates of points on conics (x, y) whose x-coordinates are equal. This has to be done only in cases where the coordinates can be represented and exactly as well as efficiently compared by number types like `leda::real` [4] or `CORE::Expr` [13]. This also holds for the case of rotated conics.

4 Implementation

To compute arrangements of rotated conics we extend the EXACUS libraries by the two real root isolators mentioned in Section 3: A modified version of the Descartes method that works on polynomials $p \in \mathbb{Q}(\sqrt{c_1})\ldots(\sqrt{c_d})[x]$, and the isolator that infers the isolating intervals from roots of integral polynomials. We also introduce a new representation class template for transformed conics (`CnX::Rotated_conic_2`), that provides access to a *transformation history*, i. e., a sequence of rotations and translations applied to the original integral polynomial. Its actual behavior is determined by a model of the *RotatedConicTraits* concept. It especially defines the allowed rotation angles, the number type used for the coefficients of the bivariate polynomial, and the method to isolate the real roots of its univariate counterparts. For the number type of the coefficients we rely on `NiX::Sqrt_extension`. It represents (nested) square root extensions, i. e., numbers of the form $a + b \cdot \sqrt{c}$, where a, b, and c are of type `Integer` or, if nested, even another instance of `NiX::Sqrt_extension` again. A such equipped version of `Rotated_conic_2` inherits all the functionality from the generic `Conic_2` class which is required for arrangement computations (e. g., with CGAL's `Arrangement_2` package) or even to compute boolean set operations on polygons bounded by arcs of rotated conics. EXACUS 1.0 contains traits classes to rotate conics by multiples of 45°, 30°, and 15°, dealing with all degenerate cases. Rotations by other angles constructible with straightedge and compass can be implemented straightforward, as we recently did for 36°.

5 Benchmark Results

We provide a selected excerpt of the benchmark results from [6]. We compute the arrangement of n (varying from 10 to 200) randomly chosen conics with the generic implementation of the sweep-line algorithm in EXACUS. Three approaches have been tested:

`CnX`: non-rotated conics using the Descartes method for real root isolation

`s-t`: rotated conics using the Descartes method twice for integral p_s and p_t to isolate the real roots of $p \in \mathbb{Q}(\sqrt{c})[x]$

`Des`: rotated conics using the Descartes method directly on $p \in \mathbb{Q}(\sqrt{c})[x]$

All conics are rotated by 45°. The running times are taken from runs on an Intel Pentium 4 CPU, clocked at 2.8 GHz with 512 kB cache. We list and illustrate the results in Figure 5 and Table 1. As one expects, the computation of arrangements of non-rotated conics is fastest. Running times roughly double when we switch to rotated conics. The comparison of `Des` versus `s-t` shows that `Des` usually is

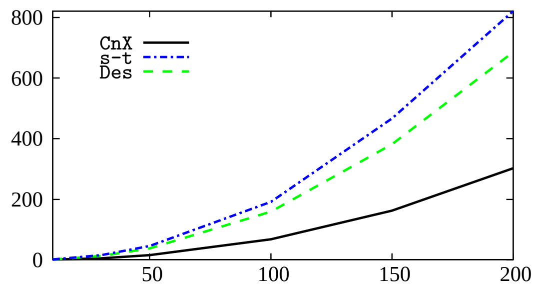

Figure 5: Comparison of running times for arrangement computation of non-rotated and rotated conics

No.	CnX	s-t	Des
10	0.6	1.5	1.3
100	67.8	190.9	158.8
200	302.5	821.2	686.3

Table 1: Running times in seconds.

measurably faster, such that we recommend to use the extended Descartes method when computing arrangements of rotated conics. The running times in general increase remarkable if more sophisticated rotation angles are involved, like rotations by 36°.

We next plan to apply the Descartes method for bitstream coefficients by Eigenwillig et al. [9] to compute intersection points of rotated conics.

Acknowledgments

We thank all Exacus developers, especially Arno Eigenwillig, Michael Hemmer, and Tobias Reithmann, and we thank Michael Sagraloff for fruitful discussions.

References

[1] J. L. Bentley and T. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, **28**(9):643–647, 1979.

[2] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, L. Kettner, K. Mehlhorn, J. Reichel, S. Schmitt, E. Schömer, and N. Wolpert. Exacus: Efficient and exact algorithms for curves and surfaces. In *Proc. 13th Europ. Sympos. Alg. (ESA)*, volume **3669** of *LNCS*, pages 155–166, 2005.

[3] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, K. Mehlhorn, and E. Schömer. A computational basis for conic arcs and Boolean operations on conic polygons. In *Proc. 10th Europ. Sympos. Alg. (ESA)*, volume **2461** of *LNCS*, pages 174–186, 2002.

[4] C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra. Efficient exact geometric computation made easy. In *15th ACM Symposium on Computational Geometry (SCG'99)*, pages 341–350, New York, NY, USA, 1999. ACM Press.

[5] J. Canny, B. Donald, and E. K. Ressler. A rational rotation method for robust geometric algorithms. In *SCG '92: Proceedings of the eighth annual symposium on Computational geometry*, pages 251–260, New York, NY, USA, 1992. ACM Press.

[6] M. Caroli. Exakte Arrangement-Berechnung gedrehter quadratischer Kurven. Bachelor's thesis, Universität des Saarlandes, Saarbrücken, Germany, 2005.

[7] E.-C. Chang, S. W. Choi, D. Kwon, H. Park, and C. K. Yap. Shortest path amidst disc obstacles is computable. In *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry*, pages 116–125, New York, NY, USA, 2005. ACM Press.

[8] G. E. Collins and A. G. Akritas. Polynomial real root isolation using descarte's rule of signs. In *SYMSAC '76: Proceedings of the third ACM symposium on Symbolic and algebraic computation*, pages 272–275, New York, NY, USA, 1976. ACM Press.

[9] A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, and N. Wolpert. A descartes algorithm for polynomials with bit-stream coefficients. In *CASC*, pages 138–149, 2005.

[10] A. Eigenwillig, V. Sharma, and C. K. Yap. Almost tight recursion tree bounds for the descartes method. In *ISSAC '06: Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 71–78, New York, NY, USA, 2006. ACM Press.

[11] I. Z. Emiris, A. Kakargias, S. Pion, M. Teillaud, and E. P. Tsigaridas. Towards an open curved kernel. In *Proc. 20th Annu. ACM Sympos. Comput. Geom. (SCG)*, pages 438–446, 2004.

[12] E. Fogel, D. Halperin, L. Kettner, M. Teillaud, R. Wein, and N. Wolpert. Arrangements. In J.-D. Boissonnat and M. Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*, chapter **1**, pages 1–66. Springer, 2006.

[13] V. Karamcheti, C. Li, I. Pechtchanski, and C. K. Yap. A core library for robust numeric and geometric computation. In *15th ACM Symposium on Computational Geometry (SCG'99)*, pages 351–359, 1999.

[14] J. Snoeyink and J. Hershberger. Sweeping arrangements of curves. In *Proc. 5th Annu. ACM Sympos. Comput. Geom. (SCG)*, pages 354–363, 1989.

[15] I. Stewart. *Galois theory*. Chapman and Hall, 2nd ed. edition, 1992.

[16] R. Wein. High-level filtering for arrangements of conic arcs. In *Proc. 10th Europ. Sympos. Alg. (ESA)*, volume **2461** of *LNCS*, pages 884–895, 2002.

[17] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. Advanced programming techniques applied to Cgal's arrangement package. In *1st Wrkshp. Library-Centric Software Design. (LCSD'06)*, 2005.

[18] C. K. Yap. Towards exact geometric computation. *CGTA: Computational Geometry: Theory and Applications*, 7, 1997.

[19] C. K. Yap. *Fundamental problems of algorithmic algebra*. Oxford University Press, Inc., New York, NY, USA, 2000.