Algebraic methods and arithmetic filtering for exact predicates on circle arcs*

Olivier Devillers Alexandra Fronville Bernard Mourrain Monique Teillaud

Abstract We present a new method to design exact geometric predicates in algorithms dealing with curved objects such as circular arcs [DFMT99]. We focus on the comparison of the abscissae of two intersection points of circle arcs, which is known to be a difficult predicate involved in the computation of arrangements of circle arcs. We give an algorithm for deciding the x-order of intersections from the signs of the coefficients of a polynomial, obtained by a general approach based on resultants. This method allows the use of efficient arithmetic and filtering techniques leading to fast implementation as shown by the experimental results.

Introduction Implementing geometric algorithms is difficult because the decisions made by such algorithms are taken on the basis of simple geometric questions, called *predicates*, solved by the evaluation of *continuous* functions subject to rounding errors, though the algorithms are basically of combinatorial and *discrete* nature. For example, the sweep line paradigm is a combinatorial algorithm relying on predicates such as x-comparisons.

The use of floating point arithmetic to evaluate predicates often produces inconsistencies. For instance, plane sweep algorithms, which are basic tools in computational geometry, are known to be very sensitive to numerical errors: when computing arrangements of curves, a plane sweep algorithm needs to sort intersection points between curves by x coordinates, and if, due to erroneous numerical computations, the x comparison test is not transitive, the algorithm may crash.

To cope with this problem, people may either work on the combinatorial part and design new algorithms that support inconsistencies, or implement the predicates in an exact way so that the combinatorial algorithm may rely on them safely. In this paper, we use the second approach and concentrate our attention on predicates.

A predicate takes a continuous input (points, coefficients) and produces a discrete result; in general the result has three possible values: two main values (e.g. *inside, outside*) correspond to geometric situations where the answer remains the same in a neighborhood of the input, and the third value (e.g. *on the boundary*) is the situation, called *degenerate*, where the answer switches from one to the other main values. The general methodology that we propose here can be sketched as follows:

-1- Determine the geometric configurations for which the input of the predicate is in a degenerate situation (same abscissae, cocircular points, ...).

-2- Apply resultant techniques to compute a polynomial of the input parameters that characterizes these configurations.

-3- Exploit the geometric meaning of this resultant polynomial in order to optimize the computations.

-4- Deduce from the signs of the roots of this resultant polynomial (as a polynomial in one of the parameters) the value of the predicate.

-5- Use efficient arithmetic and filtering techniques to get an efficient implementation of the predicate [Yap97].

In this approach, the value of the predicate depends on a combination of signs of polynomials. The robustness of a geometric algorithm is clearly connected to the algebraic degrees of the polynomials involved in the predicates it uses. The maximum of these degrees has been proposed as a measure for algorithms [LPT99].

We study this degree in the case of algorithms computing the intersections of curve segments.

*INRIA, BP 93, 06902 Sophia Antipolis CEDEX, FRANCE. `Firstname.Lastname@sophia.inria.fr` . `http://www-sop.inria.fr/prisme` . This research was partially supported by the ESPRIT IV LTR Project No. 28155 (GALIA). The full version of this paper is available at `http://www.inria.fr/RRRT/RR-3826`

Such algorithms are based on some of the following predicates [BS99]:

a) $x-$order of endpoints
b) endpoint above or below curve
c) curve intersection test
d) orientation of three endpoints
e) $x-$order of endpoint and intersection point
f) intersect in slab
g) order of intersections on curve
h) $x-$order of intersections.

Predicates a, b, c, e, h, basic for classical plane sweep algorithms, are represented on Figure 1.

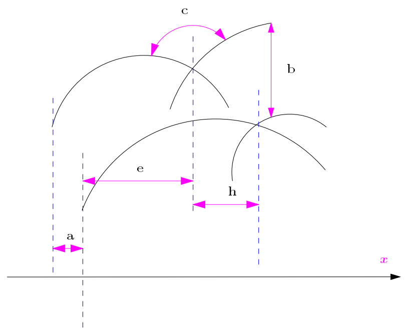

Fig. 1 *Basic predicates for plane sweep.*

Recent researches have been performed to propose algorithms that use a restricted subset of predicates of low degrees [BP, BS99, BV99]. Typically, one of the goals is to avoid predicates e) and h) because their degree is known to be higher than the degree of a). This is true in particular for the most extensively studied case: the case of line segments.

We focus here on the basic predicates a), e) and h). We show that for circle arcs, with an appropriate representation, they are equivalent, and propose a method to answer them exactly and efficiently. All details can be found in the full paper [DFMT99].

Representation We define an arc to be supported by a circle, given by the Cartesian coordinates of its center and its squared radius, and limited by two lines, given by their Cartesian equations (with some additional orientation conditions). An endpoint of a data arc is thus implicitly represented as an intersection between a circle and a line. Then, the resulting intersection point between two arcs is represented by the intersection between one of the two associated circles and the radical axis of the two circles. Notice that this does not increase the degree of the coefficients. So, intersection points are represented in the same way as endpoints, and predicates a), e) and h) are equivalent.

Using resultants We introduce a new parameter of translation t along the x-axis (Figure 2). Two of the abscissae of the vertices of $\mathcal{A}_1 = \mathcal{C}_1 \cap \mathcal{L}_1$ and $\mathcal{A}_2 = \mathcal{C}_2 \cap \mathcal{L}_2$ coincide if and only if the system

$$\begin{cases} \mathcal{C}_1(x+t, y) = 0 \\ \mathcal{L}_1(x+t, y) = 0 \\ \mathcal{C}_2(x, z) = 0 \\ \mathcal{L}_2(x, z) = 0 \end{cases}$$

has a solution in x, y, z.

$$\begin{aligned} \mathcal{C}_i(x, y) &= (x-\alpha_i)^2 + (y-\beta_i)^2 - \gamma_i, \\ \mathcal{L}_i(x, y) &= p_i\, x + q_i\, y + s_i. \end{aligned}$$

The input parameters are α_i, β_i, p_i, q_i (of degree one), γ_i and s_i (considered of degree two).

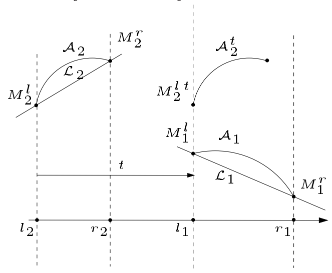

Fig. 2 *Translation*

We compute the resultant of these equations using the Bezoutian formulation [EM99, BEM00] with the MAPLE package multires[1]. We obtain a polynomial $P(t)$ whose coefficients are polynomials in the parameters of the circle arcs. P is of degree 4 in t, its expansion contains 659 monomials and the degrees of its coefficients range from 8 to 12. P has 4 real roots, corresponding to the 4 translations such that one of the abscissae of one arc coincides with one of the abscissae of the second arc. We will use the signs of the coefficients of this polynomial in t to determine the signs of the roots and thus the configuration of these two circle arcs.

Reducing the degrees The resultant polynomial that we obtain corresponds to a geometric condition, which should be a function of basic intrinsic quantities (as asserted by the first fundamental theorem of invariants [Wey39]). Indeed, moving back from algebra to geometry and using invariant theory [KR84, Stu93], we express the coefficients $P_i, i = 0, \ldots, 4$ of P in terms of basic *invariants*. We obtain a compact representation that reduces the

[1] `http://www.inria.fr/saga/logiciels/multires.html`

arithmetic complexity of the expressions whose signs must be evaluated and speeds up their numerical evaluation:

Here, we consider the two binary forms $Q_1(v,w)$ and $Q_2(v,w)$ obtained by elimination of y in the equations of circle $\mathcal{C}_i(x,y)=0$ and line $\mathcal{L}_i(x,y)=0$, using the classical Sylvester resultant:

$$Q_i(v,w) = A_i\,v^2 - 2\,B_i\,v\,w + C_i\,w^2, \quad i=1,2.$$

$$\begin{aligned} P_0 &= A_1^2\,A_2^2 \\ P_1 &= 4\,A_1\,A_2\,J \\ P_2 &= 4\,J^2 + 2\,A_1\,A_2\,K \\ P_3 &= 4\,J\,K \\ P_4 &= K^2 - 4\,I_1\,I_2 \;=\; -4JJ' + J''^2. \end{aligned}$$

The polynomials I_1, I_2 and K are the classical invariants [Dix90, MS93, Stu93] by the action of $SL_2(\mathbb{C})$ (subgroup of $GL(\mathbb{C}^2)$ of matrices of determinant 1) of $Q_1(v,w)$ and $Q_2(v,w)$. The polynomial J (resp. J') is an invariant of the same forms by translations $(v,w)\mapsto(v+aw,w)$ (resp. $(v,w)\mapsto(v,w+bv)$).

$$\begin{aligned} A_1 &= p_1^2+q_1^2 & I_1 &= B_1^2 - A_1\,C_1 \\ A_2 &= p_2^2+q_2^2 & I_2 &= B_2^2 - A_2\,C_2 \\ B_1 &= q_1^2\,\alpha_1 - p_1\,s_1 - p_1\,q_1\,\beta_1 & J &= A_1\,B_2 - A_2\,B_1 \\ B_2 &= q_2^2\,\alpha_2 - p_2\,s_2 - p_2\,q_2\,\beta_2 & J' &= B_1\,C_2 - B_2\,C_1 \\ C_1 &= s_1^2 + 2\,q_1\,s1\,\beta_1 + q_1^2\,\alpha_1^2 + q_1^2\,\beta_1^2 - q_1^2\,\gamma_1 & J'' &= A_1\,C_2 - A_2\,C_1 \\ C_2 &= s_2^2 + 2\,q_2\,s_2\,\beta_2 + q_2^2\,\alpha_2^2 + q_2^2\,\beta_2^2 - q_2^2\,\gamma_2 & K &= C_1\,A_2 + A_1\,C_2 - 2\,B_1\,B_2. \end{aligned}$$

All these polynomials can be interpreted geometrically.

Comparison of abscissae We develop a strategy to answer our predicate. The different possible configurations of the two arcs are related with the number of positive roots of P which is also related by Descartes rule on the number of sign changes in the sequence of coefficients P_0, P_1, P_2, P_3, P_4 of P [Usp48].

Case 1	$l_1 < r_1 < l_2 < r_2$	4 positive roots
Case 2	$l_1 < l_2 < r_1 < r_2$	3 positive roots
Case 3a	$l_1 < l_2 < r_2 < r_1$	2 positive roots
Case 3b	$l_2 < l_1 < r_1 < r_2$	2 positive roots
Case 4	$l_2 < l_1 < r_2 < r_1$	1 positive root
Case 5	$l_2 < r_2 < l_1 < r_1$	0 positive root

Depending on the configuration of the data, computing the signs of polynomials of degrees 5, 6 or more, but never greater than 12 in the input parameters, is sufficient. The comparison of l_1 and l_2 or r_1 and r_2 is a little more complicated than the comparison of r_1 and l_2 (see Figure 3) or r_2 and l_1 but the maximum degree is the same.

P_0	P_1	P_2	P_3	P_4	sign changes	config
	J	$K+(\ldots)^2$	JK	$-JJ'+(\ldots)^2$		
+	+	+	+	+	0	1
+	+	+	+	−	1	2
+	+	+	−	−	1	2
+	+	−	−	−	1	2
+	+	+	−	+	2	3
+	+	−	−	+	2	3
+	−	+	+	+	2	3
+	−	−	+	+	2	3
+	−	+	+	−	3	4
+	−	+	−	−	3	4
+	−	−	+	−	3	4
+	−	+	−	+	4	5

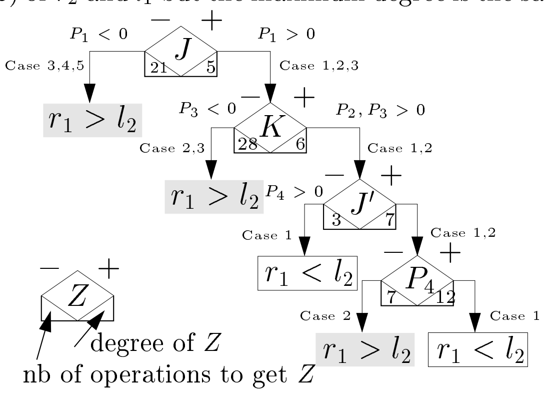

Fig. 3 *Strategy for comparing r_1 and l_2*

Benchmarks The practical impacts of this approach are twofold. First, we show experimentally that our method, though more complicated to describe, substantially improves, using the same kind of arithmetic, on the usual method of evaluation of the predicate, that consists in computing the abscissae of points and comparing them. Second, this method allows us to use the most efficient filtering techniques, which does not apply with the other method since it uses divisions and square roots.

The different possibilities that we consider have been implemented in C++ with CGAL [CGA00]. Times are given in μs per predicate evaluation[2]. We also give the percentage of success of the different kinds of filters. Filters

[2]Time performances have been measured with the Unix command `clock`. We have used a PC-Linux with Pentium-III 500MHz

Data		Times					Exactness		Success			
		Usual method		New method			Usual	New	New			Usual
		`double`	Interval + `leda_real`	`double`	GMP	static + semi-static + Usual & Int. + GMP	`double`	`double`	static	semi-static	Interval	Interval
		μs	μs	μs	μs	μs	%	%	%	%	%	%
$l_1 \leq r_2$?	rnd22	0.60	2.48	0.25	82	0.36	100	100	99	100	100	100
	rnd16	0.60	2.48	0.25	78	0.41	100	100	40	100	100	100
	almost rnd	0.60	67.	0.25	115	6.80	99	78	0	60	82	96
	degenerate	0.60	2170.	0.25	115	128.	8	17	0	0	0	0
$l_1 \leq l_2$?	rnd22	0.60	2.45	0.28	107	0.44	100	100	84	100	100	100
	rnd16	0.60	2.45	0.28	102	0.56	100	100	7	100	100	100
	almost rnd	0.60	38.1	0.28	130	5.64	99	99	0	75	93	98
	degenerate	0.60	2180.	0.28	130	144.	7	5	0	0	0	0

consist in a rounding computation whose sign is possibly certified by an error computation, either independant of the data (static filter) or dependant of the data. "Interval" is the interval arithmetic package available in CGAL [Pio99a, Pio99b]. The time for computing with `double` numbers is given as a reference, but the results are false in the case of difficult input. We give the percentage of exactness of the predicate with `double` (determined by using an exact version of the predicate for comparison).

If we compare the static + semi-static + Interval + GMP scheme in our method with the Interval + `leda_real` combination in the usual method, we are 6 times faster in general situations and 15 times faster in degenerate situations. If we compare our strategy to the hazardous `double` evaluation, the time penalty is really small and even an exact evaluation with our strategy is cheaper than an unsafe one using the usual method. In almost degenerate cases, the dynamic filter using interval arithmetic has a better success rate in the usual method than in ours. Thus the **best strategy** is to use static and semi-static filtering to filter out easy cases using the new method, then to use interval arithmetic together with the usual method, and in really difficult cases to use the new method with exact integer arithmetic.

Acknowledgments. The authors would like to thank Sylvain Pion for helpful discussions on arithmetic filters.

References

[BEM00] L. Busé, M. Elkadi, and B. Mourrain. Generalized resultant over unirational algebraic varieties. *J. of Symbolic Computation*, 2000. to appear.

[BP] J-D. Boissonnat and F. P. Preparata. Robust plane sweep for intersecting segments. *SIAM J. Comput.* to appear. Research Report 3270, INRIA, 1997. http://www.inria.fr/RRRT/RR-3270.

[BS99] J-D. Boissonnat and J. Snoeyink. Efficient algorithms for line and curve segment intersection using restricted predicates. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 370–379, 1999.

[BV99] J-D. Boissonnat and A. Vigneron. Elementary algorithms for reporting intersections of curve segments. Research Report 3825, INRIA, 1999. http://www.inria.fr/RRRT/RR-3825.

[CGA00] `http://www.cs.uu.nl/CGAL/` . *The CGAL Reference Manual*, 2000. Release 2.1.

[DFMT99] O. Devillers, A. Fronville, B. Mourrain, and M. Teillaud. Exact predicates for circle arcs arrangements. Research Report 3826, INRIA, 1999. http://www.inria.fr/RRRT/RR-3826.

[Dix90] J. Dixmier. Quelques aspects de la théorie des invariants. *Gazette des mathématiques*, 43:39–64, 1990.

[EM99] I.Z. Emiris and B. Mourrain. Matrices in Elimination Theory. *J. of Symbolic Computation*, 28(1&2):3–44, 1999.

[KR84] J. P.S. Kung and G-C. Rota. The invariant theory of binary forms. *Bulletin (New Series) of the American Mathematical Society*, 10(1):27–85, 1984.

[LPT99] G. Liotta, F. P. Preparata, and R. Tamassia. Robust proximity queries: An illustration of degree-driven algorithm design. *SIAM J. Comput.*, 28(3):864–889, 1999.

[MS93] B. Mourrain and N. Stolfi. The Hilbert series of invariants of Sl_n. In G. Jacob, N.E. Oussous, and S. Steinberg, editors, *IMACS SC'93*, pages 89–96, Lille (France), June 1993.

[Pio99a] S. Pion. *De la géométrie algorithmique au calcul géométrique.* PhD, Nice-Sophia Antipolis, France, 1999.

[Pio99b] S. Pion. Interval arithmetic: an efficient implementation and an application to computational geometry. In *Workshop on Applications of Interval Analysis to systems and Control*, pages 99–110, 1999.

[Stu93] B. Sturmfels. *Algorithms in Invariants Theory.* RISC Series on Symbolic Computation. Springer Verlag, 1993.

[Usp48] J.Y. Uspensky. *Theory of equations.* Mac Graw Hill, 1948.

[Wey39] H. Weyl. *The Classical Groups, their invariants and representations.* Princeton University Press, 1939.

[Yap97] C. Yap. Towards exact geometric computation. *Comput. Geom. Theory Appl.*, 7(1):3–23, 1997.

(the compiler is `g++ 2.95.1` with option `-O2 -mcpu=pentiumpro -march=pentiumpro`).