

Robust Geometric Computation (2004; Li, Pion, Yap)

Chee K. Yap, New York University, www.cs.nyu.edu/~yap
Vikram Sharma, New York University, www.cs.nyu.edu/~sharma

entry editor: Vikram Sharma

INDEX TERMS: Robust geometric computation, constructive zero bounds, filter techniques, precision-driven computation, exact geometric computation, guaranteed accuracy computation, expression evaluation.

SYNONYMS: Exact Geometric Computation, floating-point filter, dynamic and static filters, topological consistency.

1 PROBLEM DEFINITION

Algorithms in computational geometry are usually designed under the Real RAM model. In implementing these algorithms, however, fixed-precision arithmetic is used in place of exact arithmetic. This substitution introduces numerical errors in the computations that may lead to nonrobust behaviour in the implementation, such as infinite loops or segmentation faults.

There are various approaches in the literature addressing the problem of nonrobustness in geometric computations; see [1] for a survey. These approaches can be classified along two lines: the **arithmetic approach** and the **geometric approach**.

The arithmetic approach tries to address nonrobustness in geometric algorithms by handling the numerical errors arising because of fixed-precision arithmetic; this can be done, for instance, by using multi-precision arithmetic [2], or by using rational arithmetic whenever possible. In general, all the arithmetic operations, including exact comparison, can be performed on algebraic quantities. The drawback of such a general approach is its inefficiency.

The geometric approaches guarantee that certain geometric properties are maintained by the algorithm. For example, if the Voronoi diagram of a planar point set is being computed then it is desirable to ensure that the output is a planar graph as well. Other geometric approaches are finite resolution geometry [3], approximate predicates and fat geometry [4], consistency and topological approaches [5], and topology oriented approach [6]. The common drawback of these approaches is that they are problem or algorithm specific.

In the past decade, a general approach called the **Exact Geometric Computation** (EGC) [7] has become very successful in handling the issue of nonrobustness in geometric computations; strictly speaking, this approach is subsumed in the arithmetic approaches. To understand the EGC approach, it helps to understand the two parts common to all geometric computations: a *combinatorial structure* characterizing the discrete relations between geometric objects, e.g., whether a point is on a hyperplane or not; and a *numerical part* that consists of the numerical representation of the geometric objects, e.g. the coordinates of a point expressed as rational or floating-point numbers. Geometric algorithms characterize the combinatorial structure by numerically computing the discrete relations (that are embodied in geometric predicates) between geometric objects. Nonrobustness arises when numerical errors in the computations yield an incorrect characterization. The EGC approach ensures that all the geometric predicates are evaluated correctly thereby

ensuring the correctness of the computed combinatorial structure and hence the robustness of the algorithm.

Notation: An **expression** E refers to a syntactic object constructed from a given set of operators over the reals \mathbb{R} . For example, the set of expressions on the set of operators $\{\mathbb{Z}, +, -, \times, \sqrt{\cdot}\}$ is the set of division-free radical expressions on the integers; more concretely, expressions can be viewed as directed acyclic graphs (DAG) where the internal nodes are operators with arity at least one, and the leaves are constants, i.e., operators with arity zero. The value of an expression is naturally defined using induction; note that the value may be undefined. Let E represent both the value of the expression and the expression itself.

2 KEY RESULTS

Following are the key results that have led to the feasibility and success of the EGC approach.

Constructive Zero Bounds: The possibility of EGC approach hinges on the computability of the sign of an expression. For determining the sign of algebraic expressions EGC libraries currently use a numerical approach based upon zero bounds. A **zero bound** $b > 0$ for an expression E is such that absolute value $|E|$ of E is greater than b if the value of E is valid and nonzero. To determine the sign of the expression E compute an approximation \tilde{E} to E such that $|\tilde{E} - E| < \frac{b}{2}$ if E is valid, otherwise \tilde{E} is also invalid. Then sign of E is the same as the sign of \tilde{E} if $|\tilde{E}| \geq \frac{b}{2}$, otherwise it is zero. A **constructive zero bound** is an effectively computable function B from the set of expressions to real numbers \mathbb{R} such that $B(E)$ is a zero bound for any expression E . For examples of constructive zero bounds, see [8, 9].

Approximate Expression Evaluation: Another crucial feature in developing the EGC approach is developing algorithms for approximate expression evaluation, i.e., given an expression E and a relative or absolute precision p , compute an approximation to the value of the expression within precision p . The main computational paradigm for such algorithms is the **precision-driven approach** [7]. Intuitively, this is a downward-upward process on the input expression DAG; propagate precision values down to the leaves in the downward direction; at the leaves of the DAG, assume the ability to approximate the value associated with the leaf to any desired precision; finally, propagate the approximations in the upward direction towards the root. Ouchi [10] has given detailed algorithms for the propagation of “composite precision”, a generalization of relative and absolute precision.

Numerical Filters: Implementing approximate expression evaluation requires multi-precision arithmetic. But efficiency can be gained by exploiting machine floating-point arithmetic, which is fast and optimized on current hardware. The basic idea is to check the output of machine evaluation of predicates, and fallback on multi-precision methods if the check fails. These checks are called numerical filters; they certify certain properties of computed numerical values, such as their sign. There are two main classifications of numerical filters: *static filters* are those that can be mostly computed at compile time, but they yield overly pessimistic error bounds and thus are less effective; *dynamic filters* are implemented during run time and even though they have higher costs they are much more effective than static filters, i.e., have better estimate on error bounds. See Fortune and van Wyk [11].

3 APPLICATIONS

The EGC approach has led to the development of libraries, such as LEDA Real and CORE, that provide EGC number types, i.e., a class of expressions whose signs are guaranteed. CGAL, another major EGC Library that provides robust implementation of algorithms in computational geometry, offers various specialized EGC number types, but for general algebraic numbers it can also use LEDA Real or CORE.

4 OPEN PROBLEMS

1. An important challenge from the perspective of efficiency for EGC approach is high degree algebraic computation, such as those found in Computer Aided Design. These issues are beginning to be addressed, for instance [12].
2. The *fundamental problem of EGC* is the **zero problem**: given any set of real algebraic operators, decide whether any expression over this set is zero or not. The main focus here is on the decidability of the zero problem for non-algebraic expressions. The importance of this problem has been highlighted by Richardson [13]; recently some progress has been made for special non-algebraic problems [14].
3. When algorithms in EGC approach are embedded in larger application systems (such as mesh generation systems), the output of one algorithm needs to be cascaded as input to another; the output of such algorithms may be in high precision, so it is desirable to reduce the precision in the cascade. The geometric version of this problem is called the **geometric rounding problem**: given a consistent geometric object in high precision, “round” it to a consistent geometric object at a lower precision.
4. Recently a computational model for the EGC approach has been proposed [15]. The corresponding complexity model needs to be developed. Standard complexity analysis based on input size is inadequate for evaluating the complexity of real computation; the complexity should be expressed in terms of the output precision.

5 EXPERIMENTAL RESULTS

None is reported.

6 DATA SETS

None is reported.

7 URL to CODE

1. Core Library: www.cs.nyu.edu/exact.
2. LEDA: www.mpi-sb.mpg.de/LEDA/.
3. CGAL: www.cgal.org.

8 CROSS REFERENCES

None is reported.

9 RECOMMENDED READING

- [1] Chen Li, Sylvain Pion and Chee K. Yap. Recent progress in Exact Geometric Computation. *J. of Logic and Algebraic Programming*, 64:1(2004) 85–111. Special issue on “Practical Development of Exact Real Number Computation”, Eds. N. Mueller and M. Escardo and P. Zimmermann.
- [2] P. Gowland and D. Lester. A survey of exact arithmetic implementations. In J. Blank, V. Brattka, and P. Hertling, editors, *Computability and Complexity in Analysis*, pages 30–47. Springer, 2000. 4th International Workshop, CCA 2000, Swansea, UK, September 17-19, 2000, Selected Papers, Lecture Notes in Computer Science, No. 2064.
- [3] D. H. Greene and F. F. Yao. Finite-resolution computational geometry. *IEEE Foundations of Computer Sci.*, 27:143–152, 1986.
- [4] L. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: building robust algorithms from imprecise computations. *ACM Symposium on Computational Geometry*, 5:208–217, 1989.
- [5] S. J. Fortune. Stable maintenance of point-set triangulations in two dimensions. *IEEE Foundations of Computer Sci.*, 30:494–499, 1989.
- [6] K. Sugihara, M. Iri, H. Inagaki, and T. Imai. Topology-oriented implementation an approach to robust geometric algorithms. *Algorithmica*, 27:5–20, 2000.
- [7] C. K. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, pages 452–486. World Scientific Press, 1995. 2nd edition.
- [8] C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, and S. Schmitt. A separation bound for real algebraic expressions. In *Lecture Notes in Computer Science*, pages 254–265, 2001.
- [9] S. Pion and C. Yap. Constructive root bound method for k-ary rational input numbers, September, 2002. Extended Abstract. Submitted, 2003 ACM Symposium on Computational Geometry.
- [10] K. Ouchi. Real/Expr: Implementation of an exact computation package. Masters thesis, New York University, Department of Computer Science, Courant Institute, January 1997. URL <http://cs.nyu.edu/exact/doc/>.
- [11] Steven J. Fortune and Christopher J. van Wyk. Efficient exact arithmetic for computational geometry. In *Proc. 9th ACM Symposium on Computational Geometry*, 163–172, 1993.
- [12] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, and K. M. und E. Schmer. A computational basis for conic arcs and boolean operations on conic polygons. In *10th European Symposium on Algorithms (ESA02)*, pages 174–186, 2002. *Lecture Notes in CS*, No. 2461.
- [13] D. Richardson. How to recognize zero. *J. of Symbolic Computation*, 24:627–645, 1997.
- [14] Ee-Chien Chang, Sung Woo Choi, DoYong Kwon, Hyungju Park and Chee Yap. Shortest Paths for Disc Obstacles is Computable. *International J. Computational Geometry and Applications – Special Issue on Geometric Constraints*. Eds. X.S. Gao and D. Michelucci. Vol. 16, number 5-6, 567–590, 2006. Also appeared in *Proc. 21st ACM Symp. Comp. Geom.*, pp.116–125, 2005.
- [15] Chee K. Yap. Theory of Real Computation according to EGC. To appear in LNCS Volume based on talks at a Dagstuhl Seminar “Reliable Implementation of Real Number Algorithms: Theory and Practice”, Jan 8-13, 2006.