The Convex Hull in a New Model of Computation

Abbas Edalat* **André Lieutier†** **Elham Kashefi***

Abstract

We present a new model of geometric computation which supports the design of robust algorithms for exact real number input as well as for input with uncertainty, i.e. partial input. In this framework, we show that the convex hull of N computable real points in $\mathbb{R}^d$ is indeed computable. We provide a robust algorithm which, given any set of N partial inputs, i.e. N dyadic or rational rectangles, approximating these points, computes the partial convex hull in time $O(N \log N)$ in 2d and 3d. As the rectangles are refined to the N points, the sequence of partial convex hulls converges effectively both in the Hausdorff metric and the Lebesgue measure to the convex hull of the N points.

1 Introduction

Despite a huge number of algorithms and articles published on robustness issues related to the convex hull of a finite number of points in $\mathbb{R}^d$ [4, 2, 3, 7, 25, 26, 28], the question of computability of the convex hull for general exact real number inputs has never been addressed in the literature.

Robustness problems arise from the discrepancy between the unrealistic real RAM machine model [24], used to prove the correctness of algorithms, and real computers which are only able to deal with finite data. This problem is particularly serious in computational geometry in which combinatorial computations usually rely on numerical ones: small numerical inaccuracies turn into fatal inconsistencies in the combinatorial part of the computation. This is related to the fact that, while basic arithmetic operators and analytic functions on real numbers are Turing-computable, comparison of two real numbers is only semi-decidable [30].

Computing with uncertain inputs is unavoidable in physical modeling [9, 22]. In applications such as robotics or solid modeling, actual geometric inputs are measurement of physical objects, and as such, they are inherently uncertain and can only be represented for example using intervals of numbers.

In brief, one can classify existing approaches to robustness into two main categories. The first is the so called exact computation model. It is based on simulating a real RAM by restricting real computations to a countable subfield of $\mathbb{R}$, usually the rational numbers or a subfield of algebraic numbers [8, 6, 5, 13, 1, 14, 31, 20] for which the comparison predicate is computable. The second category, as for example in ϵ-geometry [16] or in interval geometry [29, 18, 17], tries to devise correct predicates from imprecise data and computations [27, 21].

Since the output of an algorithm often may have to be used again as the input of a new one, the depth of computation can be a priori unknown. In such cases, even in the exact methods, some form of *rounding*, with loss of information, to prevent an unrealistic growth in the size of the data is unavoidable [15].

Our new approach, which may be put in the second category, provides a general model of computation in which provably correct algorithms match naturally with feasible programs.

In this paper, aimed at the computational geometry community, we focus on practical algorithms and will only briefly describe the underlying mathematical model given in [11] that relies on recursion theory and domain theory [23, 10, 12]. We illustrate our model of computation for the non-continuous, and hence non-computable, sign predicate $s : \mathbb{R} \to \{-, 0, +\}$ that gives the sign of a real number.

In order to define the best Turing-computable approximation of this predicate, we first introduce the set $\mathbf{I}\mathbb{R}$ of real intervals. The *information order* $\sqsubseteq$ on real intervals is reverse inclusion. $[a, b] \sqsubseteq [c, d] \iff_{\text{def}} [a, b] \supset [c, d]$. Thus, $[a, b] \sqsubseteq [c, d]$ means that $[c, d]$ refines (or has more information than) the information given by $[a, b]$. The partial order $\mathbf{I}\mathbb{R}$ is a *complete partial order* (*cpo* or *domain*), in which every increasing sequence has a supremum, namely the intersection of all the compact intervals in the sequence. The set $\mathbb{D}$ of dyadic numbers is the set of rational numbers whose denumerator is a power of 2, in other words, whose binary expansion is finite. A dyadic interval is a closed interval whose ends are dyadic numbers. The countable set $\mathbf{I}\mathbb{D}$ of dyadic intervals, is said to form a *basis* of $\mathbf{I}\mathbb{R}$. Alternatively, we can work with the basis $\mathbf{I}\mathbb{Q}$ of rational intervals.

The supremum of an increasing[1] sequence of dyadic intervals is a real interval, namely the intersection of all the dyadic intervals in the sequence. The maximal real intervals $[x, x]$ are identified with real numbers. A real interval $[a, b]$ is *computable* if there is an *effective* increasing sequence of basis elements who supremum

*Department of Computing, Imperial College, London, U.K.
†Dassault Systemes Provence, Aix-en-Provence & LMC/IMAG, Grenoble, France

[1] i.e. increasing with respect to the information order $\sqsubseteq$.

(i.e. intersection) is $[a,b]$, i.e. there is a program which with input n outputs the n^{th} element of the sequence of basis elements.

For an interval $[a,b]$, define the sign predicate:

$$s([a,b]) = \begin{cases} - & \text{if } b < 0, \\ \perp & \text{if } a \leq 0 \leq b, \\ + & \text{if } a > 0. \end{cases} \qquad (1)$$

Here $\perp$ stands for "any value from $\{-,0,+\}$" and the range of s is the three element partial order $\{-,+,\perp\}$, which is denoted by $\{-,+\}_\perp$ in which $+$ and $-$ are incomparable and $\perp$ is the least element.

In our model, a real number x is given by an infinite increasing sequence of dyadic intervals x_k. Applying (1) to each x_k would give a sequence b_k of combinatorial (boolean) answers which converge in $\{-,+\}_\perp$. The sign predicate is Scott continuous[2] and an implementation of this predicate would consist in computing (1) on dyadic intervals which is possible on an integer RAM machine[3]

In subsequent sections, we apply this model of computation to the problem of computing the convex hull of n points in $\mathbf{I}\mathbb{R}^d$. The set $\mathbf{I}\mathbb{D}^d$ of all rectangles with dyadic vertices, is a basis for the domain $\mathbf{I}\mathbb{R}^d$. Using the same scheme as for the sign predicate, it is enough to compute the convex hull for a set of n dyadic interval points, that is points in $\mathbf{I}\mathbb{D}^d$.

In our framework, geometric objects, i.e. subsets of $\mathbb{R}^d$, are captured by elements of the *solid domain* $\mathbf{S}\mathbb{R}^d$, which are called *partial solids* or *partial geometric objects* [11]. A geometric object is represented by a pair (I,E) of disjoint open sets, representing respectively the interior I and the exterior[4] E of the object. The information ordering is componentwise inclusion, i.e. $(I,E) \sqsubseteq (I',E')$ iff $I \subseteq I'$ and $E \subseteq E'$. The geometric object (I,E) is maximal in $\mathbf{S}\mathbb{R}^d$ iff $I = (E^c)^\circ$ and $E = (I^c)^\circ$, where X° and X^c denote respectively the interior and the complement of a set X. The collection of pairs of interiors of finite unions of dyadic (or rational) polytopes forms a basis for $\mathbf{S}\mathbb{R}^d$. Any geometric object (I,E) can be obtained as the union of these basis elements. And (I,E) is *computable* if there exists an increasing effective sequence of these basis elements with union (I,E).

[2] A monotone (i.e. information order preserving) map f between cpo's is *Scott continuous*, if for any increasing sequence x_n, we have $f(\sup_{n\geq 0}(x_n)) = \sup_{n\geq 0}(f(x_n))$. Thus, it is enough to compute f on basis elements to obtain an algorithm that computes approximations which converge to $f(x)$ for any x.

[3] In this paper, we assume as machine model of computation a RAM (Random Access Memory) machine able to perform usual integer (and hence rational and dyadic) arithmetic. As regards computability, this machine model is equivalent to a universal Turing machine.

[4] The exterior of a set is the interior of its complement.

2 Convex hull on $(\mathbf{I}\mathbb{R}^d)^N$

We define the *partial convex hull map*:

$$\begin{array}{rccc} f: & (\mathbf{I}\mathbb{R}^d)^N & \to & \mathbf{S}\mathbb{R}^d \\ & \overline{R} & \mapsto & (I,E) \end{array}$$

where $\overline{R} = (R_1, \ldots, R_N)$ represents an ordered list of N rectangles. The open sets I and E stand, respectively, for the *interior* and the *exterior* of the partial convex hull of $\overline{R}$. In fact, E is the set of points that are surely in the exterior of the convex hull of any N points selected one from each rectangle, whereas I is the set of points that are surely in the interior of any such convex hull.

Let C be the classical convex hull map taking a set of points to the convex hull of them, considered as a compact subset of $\mathbb{R}^d$.

$$\begin{array}{rc} C: & (\mathbb{R}^d)^N \to \mathcal{C}\mathbb{R}^d \\ & (x_1, \ldots, x_N) \mapsto \\ & \{\sum_{i=1}^N \lambda_i x_i \mid \sum_{i=1}^N \lambda_i = 1 \text{ with } \lambda_i \geq 0\} \end{array}$$

where $\mathcal{C}\mathbb{R}^d$ is the set of all non-empty compact sets. For a given ordered list of N d-rectangles $\overline{R}$ in $(\mathbf{I}\mathbb{R}^d)^N$ define

$$P(\overline{R}) = \{(p_1, \ldots, p_N) \mid p_j \in R_j \text{ for } j = 1, \ldots, N\},$$

to be the set of all possible N-tuples of points of the d-rectangles.

We now define:

$$(I,E) = \left([\bigcap_{p \in P(\overline{R})} C(p)]^\circ, [\bigcup_{p \in P(\overline{R})} C(p)]^c \right).$$

For a rectangle $R \in \mathbf{I}\mathbb{R}^d$, and an index $(j_1, \ldots, j_d) \in \{-,+\}^d$ let $R^{j_1 j_2 \cdots j_d} \in \mathbb{D}^d$ be the *corner* of R whose k^{th} coordinate is the lower (respectively upper) end of the k^{th} coordinate of R if $j_k = -$ (respectively $j_k = +$). For example in the plane ($d=2$), R^{--} is the lower left corner of the rectangle R.

The following expression gives finite algorithms for the computation of I and E when the input is a list of N dyadic (or rational) rectangles, i.e. when we restrict f to the basis $(\mathbf{I}\mathbb{D}^d)^N$ (or $(\mathbf{I}\mathbb{Q}^d)^N$):

$$I = [\bigcap_{(j_1,\ldots,j_d) \in \{-,+\}^d} C(\{R_k^{j_1,\ldots,j_d} | k = 1, \ldots, N\})]^\circ$$

$$E = [C(\{R_k^{j_1,\ldots,j_d} | (j_1, \ldots, j_d) \in \{-,+\}^d, \\ k = 1, \ldots, N\})]^c$$

The expression for the exterior E above is similar to what Ratschek and Rokne call "the smallest machine representable convex hull that includes the exact convex hull" [26]. To our knowledge, however, the concept of interior I and the above algorithm for it is new.

While the complexity of computing the exterior part E is clearly the same as for the convex hull in the standard model, the optimal complexity of computing the interior part I for dimensions $d > 3$ is still open. We, however, have the following bounds.

Theorem 2.1 *For dimension $d \leq 3$, I and E can be computed in time $O(Nlog(N))$. For dimension $d > 3$, E can be computed in time $O(N^{\lfloor d/2 \rfloor})$. For $d > 3$, a set of $O(N^{\lfloor d/2 \rfloor})$ linear half-spaces with intersection I can be computed in time $O(N^{\lfloor d/2 \rfloor})$; whereas the set of $O(N^{\lfloor d/2 \rfloor})$ non-redundant linear half-spaces with intersection I can be computed in time $O(N^{2(1-\frac{1}{\lfloor d/2 \rfloor+1})\lfloor d/2 \rfloor} log^{O(1)}(N))$.*

The partial convex hull map f is *computable*; this means, in particular, that if the input is a list of N computable rectangles then the output (I, E) will be a computable geometric object.

We now examine the computation of combinatorial predicates. Given N points $x_1, \ldots, x_N$ in $\mathbb{R}^d$, consider, for $1 \leq k \leq N$, the predicate: "Does x_k lie on the boundary of the convex hull of these N points?" With uncertain input, i.e. for N input rectangles, the answer is either "surely yes" (tt), or "surely no" (ff) or "no idea"($\perp$). More formally, we have a Scott continuous predicate $P_k : (\mathbf{I}\mathbb{R}^d)^N \to \{\mathsf{tt}, \mathsf{ff}\}_\perp$. For $\overline{R} = (R_1, \ldots, R_N) \in (\mathbf{I}\mathbb{R}^d)^N$, let $\overline{R}(k) \in (\mathbf{I}\mathbb{R}^d)^{N-1}$ be the ordered list of the $N-1$ dyadic interval vertices: $\overline{R}(k) = (R_1, \ldots, R_{k-1}, R_{k+1}, \ldots, R_N)$. We have:

$$P_k(\overline{R}) = \begin{cases} \mathsf{tt} & \text{if } \; R_k \subseteq E(\overline{R}(k)), \\ \mathsf{ff} & \text{if } \; R_k \subseteq I(\overline{R}), \\ \perp & \text{otherwise.} \end{cases} \tag{2}$$

For the two-dimensional case ($d = 2$), the combinatorial structure of the polytope is the cyclic order of the contributing vertices which can be computed by finding the vertices R_j with $P_j(\overline{R}) = \mathsf{tt}$.

3 Convex Hull on $(\mathbb{R}^d)^N$

We now explain the behaviour of the output of the previous computations when the input dyadic rectangles are all refined to real points in $\mathbb{R}^d$. Let $\overline{R_j} = (R_{j1}, \ldots, R_{jN})$ ($j \geq 0$) be an increasing (i.e. refining) sequence of N-tuples of dyadic interval vertices defining an N-tuple $\overline{x} = (x_1, \ldots, x_N)$ of points in $\mathbb{R}^d$, i.e. $\bigcap_{j \geq 0} R_{jk} = \{x_k\}$.

We have the following results. The two increasing sequences $(I(\overline{R_j}))_{j\geq 0}$ and $(E(\overline{R_j}))_{j \geq 0}$ converge respectively to the interior and the exterior of the convex hull $C(\overline{x})$, which form a maximal element of $\mathbf{S}\mathbb{R}^d$:

$$\bigcup_{j\geq 0} I(\overline{R_j}) = (C(\overline{x}))^\circ \qquad \bigcup_{j\geq 0} E(\overline{R_j}) = (C(\overline{x}))^c.$$

Moreover, this convergence is *effective* in the Hausdorff distance and Lebesgue measure, which means, in particular, that there is an algorithm that, given an integer m as input, can compute n such that whenever $j \geq n$ the Hausdorff distance (respectively the volume, or Lebesgue measure of the set-theoretic difference) between $C(\overline{x})$ and $I(\overline{R_j})$ is bounded by 2^{-m} (and a similar relation on the exterior part E).

Consider now the combinatorial predicate P_k introduced in the previous section. If a point x_k does not lie on the convex hull boundary, there is a value n such that whenever $j \geq n$, the j^{th} approximation confirms it: $P_k(\overline{R_j}) = \mathsf{ff}$. If a point x_k is a non-degenerate vertex of the convex hull, i.e. it is strictly extremum on one edge of the boundary[5], then there is a value n such that whenever $j \geq n$, $P_k(\overline{R_j}) = \mathsf{tt}$. For degenerate boundary points the answer will be $\perp$ forever (undecidable).

4 Implementation

A full implementation of this model of computation with dyadic numbers would require an integer data type allowing arbitrary long binary expansions. However one can implement a more efficient version restricted to bounded binary expansions (for example 32 bit integers or floating point arithmetic), which would be less accurate but still sound with respect to the basic predicates. For this purpose, one has to carry out the analysis of an important step: *geometric rounding*. If input coordinates are approximated by b-bit dyadic numbers, the coordinates of the coefficients of the contributing half-spaces would require $\lceil db + \frac{d}{2} log_2(d) \rceil$-bit expansions for an exact integer representation. In order to keep a reasonable bit expansion, one needs a rounding scheme on half-space coefficients such that the polytopes $round(I)$ and $round(E)$ defined by the rounded coefficients give an object with less information, i.e. $round(I) \subseteq I$ and $round(E) \subseteq E$. For arbitrary accurate implementation, one needs to check further that the chosen rounding scheme preserves the convergence when the input bit size increases.

Another kind of geometric rounding consists of removing some half-spaces in the representation. Within a given rounding scheme, it is possible to analyse the bit-complexity of the algorithms.

5 Conclusion

This new model of computation addresses the problem of robust computation with uncertain inputs in a framework that is mathematically sound and supports a realistic modeling of the physical world. Instead of

[5] In other words, there exists $y \in \mathbb{R}^d$ such that x_k is a strict maximum of $x \mapsto y \cdot x$ on the convex hull.

designing a real RAM algorithm and considering separately the problem of number data types, we address directly the geometric computation within a realistic machine model. As we have shown, this point of view gives rise to interesting new issues in algorithm analysis and implementations.

Future work will focus on devising efficient rounding scheme and considering more elaborate geometric operators such as Boolean operations or Minkowski sum on polyhedra or curved objects in this new context.

References

[1] F. Avnaim, J. D. Boissonnat, O. Devillers, F. Preparata and M. Yvinec, Evaluation of a new method to compute sign of determinants In *Proc. Eleventh ACM Symposium on Computational Geometry*, June 1995.

[2] D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms? *Compt. Geom. Theory Appl.*, 7:265–302, 1997.

[3] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Disc. Comput. Geom.*, 8:295–313, 1992.

[4] H. Brönnimann. Degenerate convex hulls on-line in any fixed dimension. *Disc. Comput. Geom.*, pages 249–258, 1999.

[5] H. Brönnimann and M. Yvinec, Efficient exact evaluation of sign of determinants In *Proc. Thirteenth ACM Symposium on Computational Geometry*, pages 136–173, June 1997.

[6] H. Brönnimann, I. Emiris, V. Pan and S. Pion, Computing exact geometric predicates using modular arithmetic with single precision In *Proc. Thirteenth ACM Symposium on Computational Geometry*,pages 174–182, June 1997.

[7] D. R. Chand and S. S. Kapur. An algorithm for convex polytopes. *ACM*, 17:78–86, 1970.

[8] O. Devillers, A. Fronville, B. Mourrain and M. Teillaud, Algebraic methods and Arithmetic Filtering for Exact Predicates on Circle Arcs In *Proc. Sixteenth ACM Symposium on Computational Geometry*, pages 139–147, June 2000.

[9] H. Desaulniers and N. F. Stewart. Robustness of numerical methods in geometric computation when problem data is uncertain. *Computer-Aided Design* Special issue - Uncertainties in geometric design, 25(9):539–545, 1993.

[10] A. Edalat. Domains for computation in mathematics, physics and exact real arithmetic. *Bulletin of Symbolic Logic*, 3(4):401–452, 1997.

[11] A. Edalat and A. Lieutier. Foundation of a computable solid modeling. In *Proceedings of the fifth symposium on Solid modeling and applications*, ACM Symposium on Solid Modeling and Applications, pages 278 – 284, 1999. Full paper is to appear in *Theoretical Computer Science*. Available from `http://www.doc.ic.ac.uk/~ae/papers.html`.

[12] A. Edalat and P. Sünderhauf. A domain theoretic approach to computability on the real line. *Theoretical Computer Science*, 210:73–98, 1999.

[13] S. Fortune. Polyhedral modeling with multi-precision integer arithmetic. *Computer-Aided Design*, 29(2):123–133, 1997.

[14] S. Fortune and C. von Wyk. Efficient exact arithmetic for computational geometry. In *Proceeding of the 9th ACM Annual Symposium on Computational Geometry*, pages 163–172, 1993.

[15] M. Goodrich, L. Guibas, J. Hershberger and P. TanenBaum , Snap Rounding Line Segments Efficiently in Two and Three Dimensions In *Proc. Thirteenth ACM Symposium on Computational Geometry*,pages 284–293, June 1997.

[16] L. J. Guibas, D. Salesin, and J. Stolfi. Epsilon Geometry : Building robust algorithms from imprecise computations. *ACM Symposium on Computational Geometry*, pages 208–217, 1989.

[17] C. Y. Hu, T. Maekawa, E. C. Shebrooke, and N. M. Patrikalakis. Robust interval algorithm for curve intersections. *Computer-Aided Design*, 28(6/7):495–506, 1996.

[18] C. Y. Hu, N. M. Patrikalakis, and X. Ye. Robust interval solid modeling, part ii: boundary evaluation. *CAD*, 28:819–830, 1996.

[19] David J. Jackson. Boundary Representation Modelling with local Tolerances. *ACM Conference on Solid Modelling*, 1995.

[20] LEDA http://www.mpi-sb.mpg.de/LEDA/leda.html

[21] V. Milenkovic. Robust polygon modelling. *Computer-Aided Design*, 25(9), September 1993.

[22] T. J. Peters, D. R. Ferguson, N. F. Stewart, and P. S. Fussell. Algorithmic tolerances and semantics in data exchange. *ACM Symposium on Computational Geometry*, 1997.

[23] M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics*. Springer-Verlag, 1988.

[24] F. Preparata and M. Shamos. *Computational Geometry: an introduction*. Springer-Verlag, 1985.

[25] G. Rote. Proceedings of the 8th annual ACM on comput. geom. pages 26–32, 1992.

[26] H. Ratschek and J. Rokne *Exact and Optimal Convex Hulls in 2D* International Journal of Computational Geometry & Applications Vol. 10,No. 2 (2000)

[27] M. Segal. Using tolerances to guarantee valid polyhedral modeling results. *Computer Graphics*, 24, 1990.

[28] G. F. Swart. Finding the convex hull facet by facet. *Algorithms*, 6:17–48, 1985.

[29] T. W. Sederberg and R. T. Farouki. Approximation by interval Bezier curves. *IEEE Comput. Graph. Appl.*, 15(2):87–95, 1992.

[30] K. Weihrauch. *Computability*, volume 9 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, 1987.

[31] C. K. Yap. The exact computation paradigm. In D. Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*. World Scientific, 1995.