Numerical stability of a convex hull algorithm for simple polygons

by

J. W. Jaromczyk and G. W. Wasilkowski*

Department of Computer Science
University of Kentucky
Lexington, KY 40506

Abstract: The paper presents a numerically stable implementation of an algorithm for finding the convex hull of a simple polygon. We introduce a concept of numerical regions and provide primitives to locate points in such regions. This includes "uncertain" predicates which are inherent to a floating-point arithmetic that is used in the algorithm. An implementation and detailed numerical analysis of all primitives is given. Additionally, a concept of well-conditioned simple polygons and its impact on the performance of the algorithm is discussed.

*Partially supported by the National Science Foundation under Grant CCR-89-05371

1 EXTENDED ABSTRACT

We present a numerically stable implementation of an optimal algorithm for finding the convex hull of a simple polygon. Specifically, we discuss an implementation of the algorithm presented in Preparata and Shamos [6] which is a variant of Lee's algorithm [5].

Let $S = \{q_0, \ldots, q_{n-1}\}$ be a set of points in $\mathbf{R}^2$. The convex hull of S, briefly $CH(S)$, is the intersection of all convex sets containing S. Since S is a finite set, its convex hull is a convex polygon with vertices in S. Consequently, the problem of finding $CH(S)$ can be viewed as a process of identifying these vertices in S.

The convex hull problem is well understood; see e.g., Edelsbrunner [1], Preparata and Shamos [6]. In particular, for the point sets is $\mathbf{R}^2$, optimal $O(n \log n)$ time algorithms are known. It is also known that an additional structure on S, such as that the straight line segments $q_0q_1, \ldots, q_{n-2}q_{n-1}, q_{n-1}q_0$ forming a simple polygon, enables to reduce the time complexity to an optimal $O(n)$; see Graham and Yao [4], Lee [5]. Recall that a polygon is called simple (Jordan curve) if the intersection of any two non-consecutive edges in this polygon is empty.

A recent result in Fortune [2] states that a version of the Graham's [3] algorithm, called further SET HULL, for constructing the convex hull of a point set is numerically stable. That is, the result computed in a floating point arithmetic is a true convex hull for slightly perturbed points. The combinatorial cost of this algorithm is $\Theta(n \ln n)$.

In this paper we prove stability of an algorithm POLYGON HULL for finding the convex hull of a simple polygon; its combinatorial cost is $\Theta(n)$. In particular, we show the following theorem:

Theorem 1 *The algorithm* POLYGON HULL *is numerically stable. That is, for a well-conditioned problem, the computed $\tilde{P}$ is the convex hull for slightly perturbed $\tilde{S} = \{\tilde{q}_0, \ldots, \tilde{q}_{n-1}\}$ with the relative perturbation not exceeding $12\epsilon + O(\epsilon^2)$. Moreover, for $q_i \in \tilde{P}$, $\tilde{q}_i = q_i$, i.e., the vertices of $\tilde{P}$ are not perturbed.*

The stability of POLYGON HULL relies on a numerically stable implementation of a set of primitives used by the algorithm. In addition to numerical stability, our implementation provides information whether the computed (in *fl*) values equal the true values of the primitives. This important information can be utilized in a number of ways. For instance, POLYGON HULL can be equipped with a *flag* indicating if the computed hull $\tilde{P}$ equals the true $CH(S)$. Otherwise, it can provide a list of input points where uncertainty occurred. This, in turn, allows to remove (or decrease) the uncertainty, if desired, by employing a higher precision arithmetic **only** to those uncertain points.

We stress that despite numerical stability, POLYGON HULL algorithm may fail to provide an adequate solution. However should a failure happen, it is because of an inherent difficulty of a particular data, which in numerical analysis is termed as an *ill-conditioned* problem.

To be more specific, recall that SET HULL takes as an input a set of points, whereas POLYGON HULL takes as an input a sequence of points ordered in such a way that they are the consecutive vertices of a simple polygon (equivalently, the corresponding piecewise linear curve is a Jordan curve). Hence, although both algorithms are to provide the same solutions, the different structure of the inputs makes the two problems different. The first problem is not sensitive to small data perturbations, and that is why the first algorithm always produces (numerically) correct solutions. The second problem is, in general, very sensitive since even very small changes in the vertices of a Jordan curve may lead to a non-Jordan curve. Since the second algorithm heavily relies on the assumption of a Jordan curve, this explains why a failure of the algorithm may happen.

Thus, even though POLYGON HULL algorithm has smaller cost, in floating point arithmetic it does not guarantee acceptable solution unless the problem currently being solved is *well-conditioned.*

Finally we add that a small modification of the algorithm enables us to find out whether particular problem is well-conditioned. If this is the case, we are guaranteed to have a (numerically) correct answer in cost proportional to n. Otherwise, should an instance of ill-conditioning be encountered, the algorithm might continue computations by using the SET HULL algorithm. Of course, this modified algorithm has the worst case cost proportional to $n \log n$. However, even for ill-conditioned problems (with large sensitivity at some final points on the curve) its cost can be as small as $\Theta(n)$.

A description of the algorithm followed by a numerical analysis of the primitives, and a proof of the numerical stability of the algorithm will be given in a full version of the paper.

References

[1] H. Edelsbrunner, *Algorithms in Combinatorial Geometry,* Springer-Verlag, 1987.

[2] S. Fortune, "Stable maintenance of point-set triangulations in two dimension," *Proceedings of FOCS,* 1989.

[3] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information processing Letters* **1**, 132-133, 1972.

[4] R. L. Graham and F. F. Yao, "Finding the convex hull of a simple polygon," *Journal of Algorithms* **4**(4), 324-331, 1983.

[5] D. T. Lee, "On finding the convex hull of a simple polygon," *International Journal on Computing and Information Sciences,* **12**(2), 87-98, 1983.

[6] F. P. Preparata and M. I. Shamos, *Computational Geometry - An Introduction*, Springer-Verlag, New York, 1985.