Symbolic Treatment of Geometric Degeneracies

CHEE-KENG YAP¹

Courant Institute of Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012.

(Received January 8, 1988)

Many descriptions of algorithms in computational geometry exclude degeneracies by fiat. Practitioners are left to their own devices for dealing with degeneracies when implementing such algorithms. Since degeneracies tend to be numerous and hard to enumerate exhaustively, this is often a reason against implementing such algorithms. This paper proposes a symbolic scheme for treating degeneracies. Our method is simple to use, and is applicable to a variety of problems in computational geometry. Implementation, limitations and wider issues are discussed.

1. Introduction

The theoretical study of algorithms in computational geometry is an active area. Besides the inherent beauty arising from the interplay of geometric and algorithmic properties, this area holds the promise of impact on important application areas such as robotics, graphics and VLSI. Unfortunately, the reduction of theoretical algorithms to practice has been relatively slow and this has often been commented upon. In our view, two fundamental issues must be addressed in order to speed up this 'technology transfer'.

¹Supported in part by NSF grants #DCR-84-01898 and #DCR-84-01633. A preliminary version in proceedings, International IFIPS conference on System Modeling and Optimization, Tokyo, Japan, August 1987 (see Lecture Notes in Control and Information Science, No.113, eds. Iri and Yajima).
Fixed precision arithmetic. Theoretical algorithms assume an exact (arbitrary precision) model of numerical computation. Its implications in the world of fixed precision computations are not fully understood.

Data degeneracy. Theoretical algorithms are often described for the ‘non-degenerate’ cases of the inputs. Sometimes, even careful attempts at capturing all degenerate cases leave hard-to-detect gaps.

Both problems are a deep source of frustration for practitioners who often find mysterious failures in implementing such algorithms. We firmly believe that theoreticians are justified in making these two assumptions as long as their goal is the understanding of the global, combinatorial structure of problems. However, it would not justify a continuing neglect of both these issues. Both raise extremely interesting questions in their own right. Partly because of such neglect, there seems to be a credibility gap between theoreticians and implementors: the latter often view theoretical algorithms with suspicion.

Theoreticians have begun to address these questions. For instance, the recent paper of Greene and Yao (1986), addresses the fixed precision issue in the context of computer graphics. It is important to realize that although fixed precision arithmetic and data degeneracy are related, they are distinct issues. In this paper we deal with the latter. The symmetry breaking rules in simplex algorithms (e.g. Chvatal (1983)) are the precursors of symbolic treatment of data degeneracies. In computational geometry, Edelsbrunner and his students were among the first to publish solutions to the problem of degeneracies (Edelsbrunner (1986), Edelsbrunner and Waupotitsch (1986), Edelsbrunner and Mücke (1988), see also chapter 9.4 of Edelsbrunner (1987)). Our independently discovered scheme will turn out to be a generalization and simplification (at least conceptually) of their method. While the above papers focused on evaluating determinants, part of our conceptual simplification comes from realizing that polynomial evaluation is the proper general setting for the method.

Degeneracy in computational geometry is a general phenomenon. So in what sense can we justify its neglect in theoretical algorithms? One justification is that explicit handling of degeneracies obscures the centrality of the non-degenerate cases: degeneracies normally involve an overwhelming number of cases that are disproportionate to their likelihood of occurrence. But an implementor of these algorithms must handle the degeneracies when they do arise. Most authors are correct in suggesting that a random perturbation of data will remove degeneracies with high probability. Such a suggestion is justified if the problem is stable, as the case generally turns
out to be. However, an explicit demonstration that the problem is stable is seldom done.

In contrast to the suggestion of random perturbation, it is occasionally suggested that the algorithmic description ought to carefully work out all the degenerate cases. We take the opposite position that this is, in general, inadvisable: it is neither illuminating for a global understanding of the algorithm nor is it in the interest of the implementor who would then have to implement the numerous degenerate cases. Besides, the increase in the number of cases may lead to other programming errors or incomplete theoretical analysis. The approach advocated in this paper is to work out some general scheme to achieve suitable data perturbation. In this way, theoreticians can continue to focus on the interesting non-degenerate cases while implementors can enjoy the benefits of algorithms that have few cases and yet can handle all conceivable inputs. Paraphrased, we may say that the goal is to admit implementors to the theoreticians' paradise in which degeneracies are (virtually) abolished. This paper hopes to contribute towards such a goal. A companion to the present paper is Yap (1990).

**Overview.** The rest of the paper is organized as follows. Section 2 explores the meaning of degeneracy and related concepts. Section 3 reviews probabilistic perturbation schemes and points out unsatisfactory properties. To illustrate the symbolic approach, we outline Edelsbrunner's method in section 4. The new scheme appears in section 5. In section 6, we explore the concept of ordered rings implied by our scheme. Section 7 investigates the implementation and complexity questions arising from the scheme. We conclude with some directions for future work in section 8.

### 2. What is geometric data degeneracy?

How can a general solution as proposed in the introduction be achieved? We first need an understanding of what we mean by degeneracies. Besides clarifying our view of degeneracy, we hope that the following discussion will expose some of the issues related to degeneracy.

In this paper, we assume that the problem input is a sequence of real numbers called the input parameters \( a = (a_1, \ldots, a_n) \). An input is degenerate for an algorithm if some polynomial \( p_j(x_1, \ldots, x_{k_j}) \) in a fixed set

\[
D = \{ p_j(x_1, \ldots, x_{k_j}) : j = 1, 2, \ldots, m \text{ and } k_j \geq 1 \}, \quad (m \geq 1)
\]

evaluates to zero when an allowable substitution for the variables \((x_1, \ldots, x_{k_j})\) is made using the \(a_i\)'s. Here the set \(D\), called the test polynomials, depends
only on the algorithm and not on the inputs. Although $D$ is a finite set here (the usual case in practice), our method works as well if $D$ were infinite. The 'allowable substitution' of variables in each test polynomial by input parameters is dictated by the problem. We refrain from making this more formal but this will be easy to do (for each particular case) once the following examples are understood.

An alternative definition is this: assume that the algorithm only makes decision steps by evaluating polynomials from a set $D$, and makes a three-way branch (zero, positive or negative) depending on the sign of the evaluation. Then an input $a$ is degenerate if the algorithm on input $a$ takes the zero-branch during a decision step.

**Examples of degeneracy.** When input $a$ represents a set of points in the Euclidean plane, degeneracies that has appeared in the literature include (i) two coincident points, (ii) three collinear points, or (iii) four cocircular points. If the input represents a set of lines, common notions of degeneracy include (iv) a vertical line, (v) two parallel lines, (vi) two perpendicular lines, or (vii) three concurrent lines. If the input represents points and lines, degeneracy may include (viii) a point lying in a line, or (ix) a line parallel to the line through two points. This list can go on. In fact, many papers do not make an explicit list of the degeneracy conditions but uses the blanket term 'general position' to describe the non-degenerate input.

The 'allowable substitution' above may only amount to typing the variables in $D$ and the input parameters so that substitutions must respect the type distinctions. For instance, some variables and parameters correspond to the first coordinate of points and others correspond to the slope of lines, etc. More concretely, suppose the input parameters are $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$ representing $n$ points in the plane, and let the set of test polynomials consists of only one polynomial,

$$\Delta(x_1, y_1, x_2, y_2, x_3, y_3)$$

which is just the 3 by 3 determinant that tests if the points $(x_1, y_1), (x_2, y_2)$ and $(x_3, y_3)$ are collinear (cf. section 4). Allowable substitution here means that (1) the substitution of the input parameters must be in pairs (i.e. for all $i, j$, the pair $(a_i, b_i)$ must be substituted simultaneously for $(x_j, y_j)$), and that (2) the three input points to be substituted must have distinct subscripts.

**Exact model of numerical computation.** It is important to realize that this paper assumes the exact model of numerical computations. All
numbers are represented exactly: for example, to represent any real algebraic number \( \alpha \) exactly, it suffices to specify a polynomial \( p(x) \) with integer coefficients together with an interval \( I \) containing \( \alpha \) but no other distinct root of \( p(x) \). The end points of \( I \) are exact, say represented by rational numbers. Our development is not restricted to any particular exact representation. Note that since the integers involved (as in the coefficients of \( p(x) \), or in representing \( I \)) can be arbitrarily long, this is sometimes called the arbitrary precision model. However, the terminology 'exact model' is more to the point since although exact computation imply arbitrary precision, the converse is not certainly true. In any case, the exact model stands in sharp contrast to the fixed precision model in much of numerical analysis. With the advent of computer algebra, the exact models are becoming more important and indeed unavoidable for some applications.

Moreover, we think that computers can realistically incorporate hardware arithmetic units to do arbitrary precision arithmetic whose performance degrade gracefully with the precision. Such machines would be a boon to computational geometry since exact computation for many algorithms in computational geometry requires only bounded precision in the sense that for inputs that are accurate to \( n \)-bits, the computation can be carried out exactly using \( kn \)-bits numbers, where \( k \) depends only on the problem (\( k \) is usually small). This would eliminate some of the well-known problems with floating point arithmetic in current hardware.

We have stated that the issues of fixed precision and data degeneracy are distinct. Indeed, any attempt to consider data degeneracy in the fixed precision model faces some very difficult problems: for instance, there are many puzzles in just trying to define what it means for three points to be collinear in the fixed precision model (actually, the world of pixels). In any case, one should begin by understanding degeneracy in exact models.

**Degree of derivation and derived degeneracies.** In the course of executing algorithms, derived values \( b = (b_1, b_2, \ldots) \) may be generated from the input parameters \( a \). Let us define the degree of derivation of \( b \) be the least integer \( d \geq 1 \) such that for each \( b_i \) in \( b \) there is a \((m + 1)\)-variate polynomial \( p(x, x_1, \ldots, x_m) \) (\( m \) depends on \( b_i \)) with integer coefficients, whose (algebraic) degree is at most \( d \) such that if each \( x_j \) (\( j = 1, \ldots, m \)) is substituted by suitable values \( \gamma_j \) from \( a \), then \( b_i \) is a root of \( p(x, \gamma_1, \ldots, \gamma_m) \). For instance, if the input represents points, then \( b \) may be the computed distances between pairs of points, and the degree of derivation is 2. If this

---

It is also called the 'infinite precision model' but this is even more misleading since the precision is finite at any moment.
degree $d$ of derivation does not grow with the input size $n$, we say that the algorithm has bounded degree of derivation. Examples of bounded degree algorithms include the usual algorithms for computing the convex hulls or the Voronoi diagrams. Examples of algorithms with unbounded degree include shortest path problems and root isolation (or more generally, cell decomposition). Interestingly, the Simplex Algorithm is bounded degree while the newer polynomial time methods for linear programming are of unbounded degree. Our method to be described does not allow the substitution of derived values with derivation degree more than 1 (this is really the same as allowing substitution by input parameters only), but it is not hard to see that bounded degree problems can be handled by a simple modification.

**Alternative notions of degeneracy.** In a provocative discussion Kender and Freudenstein (1986) point out that there are several, not necessarily mutually-consistent, notions of degeneracy in the literature. They propose to unify these disparate notions by describing degeneracies to be a ‘system relative’ concept. Of course, this is a very general formulation and our particular notion here is ‘system relative’ to the extend that the particular set of test polynomials $D$ depend on the algorithm.

Degeneracies are often depicted as rare events. In computer vision, there are notions of degeneracy that belie this view, as shown by this example from Kender and Freudenstein (1986). Imagine a very squat pyramid and a view of the pyramid from ‘below’. This view shows only the base of the pyramid and would be considered a ‘degenerate view’ in certain contexts of vision research. Yet, this bottom view is hardly ‘rare’ by any reasonable definition (in the sense of geometric measures). Perhaps it is better to call such views ‘deficient’ (since they give inadequate information) rather than degenerate.

Perhaps a more pertinent illustration which suggests that the ‘degeneracies as rare events’ view require careful interpretation is this: when dealing with highly structured scenes or robot environments, certain ‘degeneracies’ such as parallel lines or collinear points are features rather than accidents of the input space. For instance, in descriptions of a robot environment in a factory, we expect parallel lines to be a common feature. The explanation lies in realizing that we normally assume that the input space (for a fixed input size of $n$) is the set of all $n$-tuples of (exactly representable) real numbers. It may happen that the input space is a proper subset of all possible $n$-tuples (usually, a submanifold); ‘rarity of degeneracies’ must be relative to this manifold. However, our method as it stands assumes the full input

---

\(^3\)Indeed, the talk of Kender prompted the present work.
Problem stability. The fundamental assumption in this paper is that with the 'rare event' view of degeneracies, we can perturb away the degeneracy. A crucial but often implicit requirement is that the problem at hand defines a function from the input space to the output space that is 'continuous' in this sense: a solution to a perturbed version of the input data is a reasonable approximate solution for the original data. Then we are indeed justified in using a solution to the perturbed input as the final output. We call such problems stable. Since continuity arguments are involved, stability is relative to the choice of topologies on the spaces concerned.

Example of instability/stability. Consider the problem of constructing the Voronoi diagram of a set of planar points (sites). We would like to show that this problem is stable. Let us assume that certain four cocircular sites cause the Voronoi diagram to have a Voronoi vertex of degree 4. Any perturbation of the input that removes the cocircularity of these four sites causes the said Voronoi vertex to split into two very close Voronoi vertices of degree 3 each. There are two combinatorially distinct ways in which this split can occur. Clearly the combinatorial structure of the perturbed Voronoi diagram is different from the original Voronoi diagram. So, in the combinatorial sense, the problem would not seem stable. Another attempt at trying to show that the problem is stable is this: use the Hausdorff metric on closed point sets. Unfortunately a small perturbation may introduce an infinite Hausdorff distance between the two Voronoi diagrams. [Consider the diagram of two points \( p = (-1, 0) \) and \( q = (+1, 0) \) and then perturb one of the points to \( p' = (-1, \delta) \) for all \( \delta > 0 \).]

Now suppose our goal is to use this Voronoi diagram to compute an obstacle-avoiding path for a unit disc between two specified points \( P \) and \( Q \), viewing these sites as obstacles. (It follows from O'Dunlaing and Yap (1985) that to find such a path, it is sufficient to look for one path in which the center of the disc lies in the Voronoi diagram.) Let the set of \( n \) sites be represented by \( a \in \mathcal{E}^d \) (so \( d = 2n \) and \( \mathcal{E}^d \) is the Euclidean \( d \)-dimensional space). It is natural to measure the 'connection width' between any two points \( P \) and \( Q \) in the plane by the quantity \( C(P, Q; a) \geq 0 \) defined as the maximum clearance attained by some path from \( P \) to \( Q \) in the presence of obstacles defined by \( a \). Clearly, if \( C(P, Q; a) \leq 1 \) then the unit disc has no obstacle-avoiding path connecting \( P \) and \( Q \). Then it is easy to show:

\[ C(P, Q; a) = \max_{x \in [0,1]} \min_{y \in \mathcal{E}^d} \text{clearance of } x \text{ to } y \]

4The clearance of a point \( x \) is its distance from the closest site; the clearance of a path is the minimum clearance among points along the path.
Proposition 1. For any \( \delta > 0 \) there is an \( \epsilon > 0 \) such that for all \( a, b \in \mathcal{E}^d \), and for all points \( P \) and \( Q \), if \( ||a - b|| < \epsilon \) then \( |C(P, Q; a) - C(P, Q; b)| < \delta \). Here \( ||a|| \) denotes the Euclidean norm.

Such a stability or continuity property justifies the perturbation of input \( a \) for this application. This illustrates the kind of justification that must logically precede any application of perturbation methods.

**Induced and inherent degeneracy.** We distinguish between inherent degeneracy of the input data versus an algorithm-induced degeneracy. For example, if the input represent points to a convex hull algorithm, it is apparent that three consecutive collinear vertices on the convex hull ought to be an inherent degeneracy of the convex hull problem. Now if the algorithm uses some kind of vertical partitioning of the input set of points, or uses the vertical sweepline paradigm, then two co-vertical points may be regarded as a degeneracy. These degeneracies are easily removed in this case, either by modifying the algorithm or by perturbing the data deterministically. In any case, they are degeneracies induced by the algorithm. This paper is concerned with algorithm-induced degeneracy. It seems that induced degeneracies subsume inherent degeneracies.

3. Probabilistic schemes

In this paper, we assume a computational model with the following property. Recall that the input is a sequence of real numbers \( a = (a_1, \ldots, a_n) \). There is a fixed set \( D \) of polynomials, independent of the input such that

\[ (*) \text{ All decision steps of the algorithm are based on evaluating some polynomial } p(x) \in D \text{ at } x = (x_1, \ldots, x_k) := (a_{j_1}, \ldots, a_{j_k}) = b \]

where \( b \) is an (allowable) substitution (elements of \( b \) are taken from \( a = (a_1, \ldots, a_n) \)). The algorithm then makes a 3-way branch depending on the sign of \( p(b) \), with the case \( p(b) = 0 \) considered to be degenerate.

The goal is to devise a data-perturbation method so that the algorithm never takes a degenerate branch. It is instructive to first review probabilistic methods for data perturbation often alluded to in the literature.

Perhaps the simplest solution is to arbitrarily choose either \( p(b) < 0 \) or \( p(b) > 0 \) whenever \( p(b) = 0 \) is encountered. This method suffers from
the problem of global consistency: how can transitivity (i.e., \( p(b) > q(b) \) and \( q(b) > r(b) \) implies \( p(b) > r(b) \)) be maintained without expensive bookkeeping?

The initial perturbation scheme overcomes the consistency problem by preceding the entire computation with an initial random perturbation of the input data. This perturbation ought to guarantee

1. No new degeneracies arise as a result of the perturbation.
2. The original degeneracies are all removed.

Property (1) can be satisfied by computing some a priori upper bound on the size of the perturbation. Part (2) seems more difficult to ensure. Granted that with very high probability no degeneracy remains, the issue remains as to what the algorithm must do when a degeneracy does arise? The simplest recovery is to restart the algorithm with another random perturbation. This, in principle, can repeat indefinitely. Although the probabilistic overhead complexity of the initial perturbation scheme is small, it would be nice to obtain a deterministic perturbation method.

### 4. Simulation of Simplicity

We now turn to symbolic perturbation schemes. To illustrate, we briefly review a version of a scheme (called simulation of simplicity, or SoS for short) described in Edelsbrunner and Mücke (1988) (see also Edelsbrunner (1986) and Edelsbrunner and Waupotitsch (1986) for other applications of SoS). The setting is that of computing hyperplane arrangements where the test polynomials are determinants. For illustration, say a set \( H \) of lines in the plane is simple if (i) no three lines are concurrent, (ii) no two lines are parallel, (iii) no two pairs of lines intersect on a common vertical line, and (iv) no two pairs of lines intersect on a line parallel to a line in \( H \). Each of these conditions corresponds to the vanishing of a suitable determinant over the input parameters. Let a line \( h_i \in H \) be given by \( y = a_i x + b_i \). Now replace each input line \( h_i \) by \( h_i(\epsilon) \) with equation \( y = a_i(\epsilon)x + b_i(\epsilon) \) where

\[
a_i(\epsilon) = a_i + \epsilon^{2^i}, \quad b_i(\epsilon) = b_i + \epsilon^{2^{i-1}}
\]

It is then shown that if \( \epsilon > 0 \) is sufficiently small, then the new arrangement \( H(\epsilon) \) is simple. However, instead of substituting actual values for \( \epsilon \), we carry
out the calculation symbolically. For example, to decide if the intersection of \( h_i(\epsilon) \cap h_j(\epsilon) \) lies above the line \( h_k(\epsilon) \), we must determine the sign of

\[
\Delta(\epsilon) = \det \begin{pmatrix}
a_i(\epsilon) & b_i(\epsilon) & 1 \\
a_j(\epsilon) & b_j(\epsilon) & 1 \\
a_k(\epsilon) & b_k(\epsilon) & 1
\end{pmatrix}
\]

It is then observed that evaluating the sign of \( \Delta(\epsilon) \) amounts to evaluating a sequence of subdeterminants of the original matrix until the first non-zero entry:

\[
\Delta(\epsilon) = \det \begin{pmatrix}
a_i & b_i & 1 \\
a_j & b_j & 1 \\
a_k & b_k & 1
\end{pmatrix}
- \epsilon^{2i-1} \det \begin{pmatrix}
a_j & 1 \\
a_k & 1
\end{pmatrix}
+ \epsilon^{2i} \det \begin{pmatrix}
b_j & 1 \\
b_k & 1
\end{pmatrix}
+ \epsilon^{2j-1} \det \begin{pmatrix}
a_i & 1 \\
_k & 1
\end{pmatrix}
\]

+ \ldots - \epsilon^{2i-1+2j} + \ldots

We will show that this is actually a general phenomenon. One may regard each input parameter \( a_i \) to be perturbed by an infinitesimal amount \( \delta_i (\delta_i = \epsilon^{2k} \) in the preceding illustration). Furthermore there is a suitable fixed total ordering on the set of infinitesimals and their products. For instance, without loss of generality, we may assume

\[
\delta_1 \ll \delta_2 \ll \ldots \ll \delta_i \ll \ldots \ll \delta_n.
\]

But what, for instance, is the relation between \( \delta_1 \delta_3 \) and \( \delta_2^3 \)? In the next section, we will give a systematic framework for making such comparisons.

5. A general scheme to avoid zeroes

We describe a procedure to evaluate any polynomial \( p(x) \) at any value \( x := a \). The procedure outputs the value \( p(a) \). However, in case \( p(x) \) is a non-zero polynomial and \( p(a) = 0 \), then the output is one of two types of zeroes: 0− or 0+. This sign information will be globally consistent in some natural sense. Such a procedure can be used as a black-box by any algorithm satisfying assumption (*) above to always avoid the degenerate branch of a decision step.

Observe that this scheme also gives us a method of comparing the values of two distinct polynomials \( p, q \) at any fixed point \( x = a \). More precisely, if
the sign of the difference polynomial $p - q$ at $x = a$ is positive then we say $p(a) > q(a)$; otherwise $p(a) < q(a)$. Extending this, it means that we can strictly order any set of distinct polynomials $\{p_1, \ldots, p_i\}$ by their ‘values’ at any point $x := a$.

As in Mishra and Yap (1989), let $PP = PP(x_1, \ldots, x_n)$ denote the set of all power products

$$w = \prod_{i=1}^{n} x_i^{e_i} \quad (e_i \geq 0).$$

Let $\text{deg}(w)$ denote $\sum_{i=1}^{n} e_i$. A total ordering $\leq$ on $PP$ is admissible if for any $w, w', w'' \in PP$,

1. $1 \leq w$
2. $w \leq w'$ implies $ww'' \leq w'w''$.

The two most important examples of admissible orderings are the total degree ordering, denoted $\leq_{\text{TOD}}$, and the (pure) lexicographical ordering, denoted $\leq_{\text{LEX}}$. Let $w, v$ be two power products with degree vectors $(e_1, e_2, \ldots, e_n)$ and $(d_1, d_2, \ldots, d_n)$. Then we define $w \leq_{\text{LEX}} v$ if $w = v$ or else $e_i < d_i$ at the smallest index $i$ where the $e_i$'s and $d_i$'s differ. We also define $w \leq_{\text{LEX}} v$ if $\text{deg}(w) < \text{deg}(v)$ or else $(\text{deg}(w) = \text{deg}(v)$ and) $w \leq_{\text{TOT}} v$. In computer algebra work, the reverse lexicographical ordering has many important properties. In fact, there are uncountably many admissible orderings.

Henceforth we assume some arbitrary but fixed admissible ordering $\leq$ on $PP$.

For any $w = x_1^{e_1} \cdots x_n^{e_n}$ and polynomial $p(x), x = (x_1, \ldots, x_n)$, let $p_w$ refer to the $\text{deg}(w)^{th}$ partial differential of $p$, where $p$ is differentiated $e_k$ times with respect to each variable $x_k (k = 1, \ldots, n)$. For example, if $w = x^2 y$ then $p_w = \frac{\partial^2 p}{\partial x^2 \partial y}$. Let $S(p)$ denote the infinite list of polynomials

$$S(p) = (p_{w_0}, p_{w_1}, p_{w_2} \cdots)$$

where $w_0, w_1, w_2, \cdots$ is the list of power products in $PP$ in increasing $\leq$-order.

In particular $w_0 = 1$ and $p_{w_0} = p$. This infinite sequence has only a finite number of non-zero entries and assuming that that $p$ is not identically zero, the last non-zero entry is a constant. Our polynomial evaluation procedure
proceeds as follows: given a non-zero polynomial \( p(x) \) and a point \( a = (a_1, \ldots, a_n) \), we evaluate successive polynomials in \( S(p) \) at \( x := a \) until the first non-zero result; by above remarks, termination of this procedure is guaranteed. In case \( p(a) \neq 0 \), then we return \( p(a) \); otherwise we return either \( 0^+ \) or \( 0^- \), where the sign of the zero is that of the first non-zero result evaluated in the sequence \( S(p) \). It is convenient to introduce the notation
\[
S(p; a) = (p_{w_0}(a), p_{w_1}(a), p_{w_2}(a), \ldots).
\]

Lemma 2
(a) \( S(p; a) \) is the sequence of all zeroes iff \( p \) is identically zero.
(b) For any \( p \) and \( a \), the polynomial \( p \) is uniquely determined by the sequence \( S(p; a) \).

Proof. (a) follows from the above remark that any non-zero polynomial has a derivative that is a non-zero constant. (b) Suppose that \( p, q \) are two polynomials such that \( S(p; a) = S(q; a) \). Then \( S'(p-q; a) = 0 \) (the sequence of all zeroes). Then part (a) implies that \( p - q \) is identically zero, i.e. \( p = q \). Q.E.D.

This proof does not tell us how to reconstruct the polynomial \( p \) from the sequence \( S(p; a) \) but it is easy to give a formula to reconstruct \( S(p; a) \) from \( S(p_{x_i}; a) \) for all \( i = 1, \ldots, n \), where \( p_{x_i} \), of course, is partial differentiation with respect to \( x_i \).

The evaluation procedure amounts to a function that assigns a sign (zero, positive or negative) to every polynomial, where each non-zero polynomial is either positive or negative.

We define \( p(x) > q(x) \) to mean that the polynomial \( p(x) - q(x) \) has positive sign at \( x := a \).

Corollary 3 For all polynomials \( p \) and \( q \), exactly one of the following relation holds: \( p = q \) or \( p > q \) or \( q > p \).

6. Ordered Rings

We now show that the relation \( > \) is a total ordering and has other algebraic structure as well. There is a beautiful theory of ordered fields due to Artin and Schreier (see van der Waerden (1970)). One sees that the definition of ordered fields (van der Waerden (1970)) only uses the ring properties
of fields. Accordingly, we may adapt that definition to rings. Our ring $R$ will be assumed to be commutative with a unit 1.

A partially ordered ring $R$ is a ring with an associated sign function $\sigma : R \rightarrow \{-1, 0, +1\}$ such that for all $a, b \in R$:

A1 $\sigma(a) = -\sigma(-a)$. In particular, $\sigma(a) = \sigma(-a)$ iff $\sigma(a) = 0$.

A2 $\sigma(ab) = \sigma(a)\sigma(b)$

A3 $\sigma(a) \geq 0$ and $\sigma(b) \geq 0$ implies $\sigma(a + b) \geq 0$, with equality iff $\sigma(a) = \sigma(b) = 0$.

Call a ring element $a$ positive, negative or nullary according to the sign $\sigma(a)$ of $a$. Note that axiom (A1) implies that $\sigma(0) = 0$, and (A2) implies $\sigma(1) = +1$. If the sign function, in addition, satisfies the property

A4 $\sigma(a) = 0$ if and only if $a = 0$

then we call $R$ an ordered ring.

We define a relation $>_o$ on $R$ where

$$a >_o b \text{ if and only if } \sigma(a - b) = +1.$$ 

Also, define $a \leq_o b$ if either $a = b$ or $b >_o a$. We get the expected properties as in van der Waerden (1970): transitivity follows from axiom (A3); $a >_o b$ implies $a + c >_o b + c$, and if $c$ is positive, then $ac >_o bc$ holds as well. And if $R$ is a field, $a >_o b$ if and only if $b^{-1} >_o a^{-1}$. Of course, $\leq_o$ is a partial ordering on $R$, and if $R$ is an ordered ring then $\leq_o$ becomes a total ordering. The following two consequences are less obvious:

Lemma 4

1. In a partially ordered ring, $a + b >_o a' + b'$ implies $a >_o a'$ or $b >_o b'$.

2. In an ordered ring, if $a$ and $b$ are both positive and $ab >_o a'b'$, then $a >_o a'$ or $b >_o b'$.

Proof. 1. $\sigma(a + b - a' - b') = +1$ implies that $\sigma(a - a') = +1$ or $\sigma(b - b') = +1$ (otherwise $\sigma(a' - a) \geq 0$ and $\sigma(b' - b) \geq 0$ and axiom (A3) implies $\sigma(a' + b' - a - b) \geq 0$, contradiction).

2. If $\sigma(a - a') = 0$ then $a = a'$ (since $R$ is an ordered ring) and $ab >_o a'b'$.
implies \( ab > ab' \). This means \( \sigma(a(b - b')) = +1 \) and since \( \sigma(a) = +1 \), we get \( \sigma(b - b') = +1 \), proving the desired result. Similarly if \( \sigma(b - b') = 0 \).

Therefore, if we assume the result is false, we must have \( \sigma(a - a') = \sigma(b - b') = -1 \). We get a contradiction as follows. First, these assumptions imply \( \sigma((a' - a)(b' - b)) = +1 \). It follows that \( \sigma((a' - a)(b' - b) + (ab - a'b')) = +1 \). Rearranging terms, we get \( \sigma(a(b - b') + b(a - a')) = +1 \). Using (A3), we infer that \( \sigma(a(b - b')) = +1 \) or \( \sigma(b(a - a')) = +1 \). If \( \sigma(a(b - b')) = +1 \) then applying (A2) with \( \sigma(a) = +1 \), we get \( \sigma(b - b') = +1 \) which is a contradiction. A similar contradiction arises if \( \sigma(b(a - a')) = +1 \). Q.E.D.

The main examples of ordered fields are the rational numbers and the reals. In van der Waerden (1970) (exercises), it is pointed out that we can also order the set of univariate polynomials with coefficients over an ordered field by regarding the sign of a polynomial to be the sign of the leading coefficient. This can be generalized to multivariate polynomials once we have fixed an admissible ordering on the power products: then we may speak of the 'head monomial' and take the sign of a polynomial to be the sign of the coefficient of the head monomial. We now show a new family of total orderings based on our polynomial evaluation scheme.

Let \( Q \) be any ordered ring and let \( R = Q[x_1, \ldots, x_n] \) be the ring of \( n \)-variate polynomials with coefficients from \( Q \). Let \( \leq \) be any fixed admissible ordering on the power products \( PP = PP(x_1, \ldots, x_n) \).

**Lemma 5** Let \( u, v, u', v' \in PP \) such that either \( u \neq u' \) or \( v \neq v' \). If \( uv \leq u'v' \) then \( u \leq u' \) or \( v \leq v' \).

**Proof.** Let the exponents of \( x_1, \ldots, x_n \) in \( u \) and \( u' \) be \( u = (u_1, \ldots, u_n) \) and \( u' = (u'_1, \ldots, u'_n) \), respectively. Similarly for let \( v, v' \) denote the exponents with respect to \( v, v' \). By a characterization of admissible orderings (see Dubé, Mishra and Yap (1986)), there corresponds to \( \leq \) an \( n \) by \( n \) matrix \( W \) with non-negative real entries such that \( uv \leq u'v' \) iff

\[
(u + v)W \leq (u' + v')W.
\]

Let \( y = uW, z = vW, y' = u'W \) and \( z' = v'W \). Let the \( i \)th component of \( y \) (resp. \( y', z, z' \)) be \( y_i \) (resp. \( y'_i, z_i, z'_i \)). If \( i \) is the first index where either \( y_i \neq y'_i \) or \( z_i \neq z'_i \) then since

\[
y_i + y'_i \leq z_i + z'_i
\]
we must have either $y_i < y'_i$ or $z_i < z'_i$. This means either $u < u'$ or $v < v'$. Q.E.D.

We now extend the sign function on $Q$ to a sign function on $R = Q[x_1, \ldots, x_n]$ according to our evaluation scheme. Fix any point $a \in Q^n$ and any admissible ordering $\leq$ on $PP$. For any $p \in R$, we again have the sequence $S(p)$ of its partial derivatives ordered according to $\leq$, and also the sequence $S(p; a)$ obtained by evaluating $S(p)$ at $a$. Define the sign $\sigma(p)$ of $p$ to be the sign (since $Q$ is ordered) of the first non-zero entry in $S(p; a)$. If all entries in $S(p; a)$ are zero, then $\sigma(p)$ is defined to be 0.

**Theorem 6** $R = Q[x_1, \ldots, x_n]$ with the sign function $\sigma$ as determined by the evaluation scheme is an ordered ring. Note that $\sigma$ depends on the choice of admissible ordering $\leq$, on the choice of $a \in Q^n$, and on the sign function on $Q$.

**Proof.** We must verify the four axioms (A1-A4). Axiom (A1) follows from the fact that $S(p; a) = -S(-p; a)$, and (A4) comes from the fact that $S(p; a)$ has some non-zero constant entry unless $p$ is identically zero. (A3) comes from the fact that $S(p + q; a) = S(p; a) + S(q; a)$ where we have component-wise addition of the sequences. To show (A2), suppose that the first non-zero entry in $S(p)$ (resp. $S(q)$) is $pu$ (resp. $qv$) where $u, v \in PP$. We claim that the first non-zero entry in $S(pq)$ is $(pq)_{uv}$ (i.e. the partial derivative of $pq$ with respect to $uv$). We see that

$$(pq)_{uv} = \sum_{u'} p_{u'v'} q_{v'}$$

where $u' \in PP$ range over all divisors of $uv$ and $v'$ is given by $u'v' = uv$. By the previous lemma, unless $u = u'$ and $v = v'$, $u'v' < uv$ implies $u > u'$ or $v > v'$. Hence $p_{u'} = 0$ or $q_{v'} = 0$. We conclude that $(pq)_{uv} = p_u q_v$.

Suppose $uv > u'v'$. The same argument as above shows $(pq)_{u'v'} = 0$. It follows that the sign of $pq$ is equal to the sign of $p_u(a)q_v(a)$, which is equal to $\sigma(p)\sigma(q)$. This proves (A2). Q.E.D.

As an example, we see that the scheme in Edelsbrunner and Mücke (1988) is an ordering on the ring of polynomials since it can been shown that their choice of perturbation leads to the lexicographical ordering on $PP$; this observation does not immediately come out in the original paper and is (we feel) one of the advantages of our more general viewpoint.
7. Practical issues and complexity

On evaluating sparse polynomials. Since all evaluation of polynomials are to be done through our 'black-box', it is important for this black-box to be efficient. We now consider this issue. Let \( L(n, s, d) \) denote the minimum number of arithmetic operations (+, -, \( \times \), \( \div \)) sufficient to evaluate all polynomials with \( s \) monomials (terms) on \( n \) variables, where each monomial has degree at most \( d \). Hence

\[
s \leq \binom{n + d}{n}
\]

is a measure of the sparsity of the polynomial and we call this the sparse complexity model of polynomial evaluation. Little work has been done on this model except in special cases: basically two cases are understood: dense univariate polynomials \((n = 1, s = d+1)\) and multivariate monomials \((s = 1\) but \(n, d\) arbitrary). Nevertheless, the model is important especially in the multivariate setting since it is too expensive in general not to exploit sparsity of multivariate polynomials. Notice that even the determinantal function (on \( n^2 \) variables and degree \( n \)) is not 'dense' in the usual sense of having all (or a constant fraction) its possible terms present.

Note that if an algorithm only evaluates polynomials of a bounded degree then the worst case cost of each black-box evaluation is a constant that does not depend on the input size (= \( n \) if the input is \((a_1, \ldots, a_n)\)). This situation is quite common in computational geometry (e.g., convex hulls, Voronoi diagrams, line arrangements). But even with the very large worst case bounds for multivariate polynomials of unbounded degree, one is unlikely to need to evaluate more than the first few polynomials before turning up a non-zero result. With high probability, the black-box evaluation time is exactly the time needed to evaluate the polynomial in the usual sense - so there is no overhead time, with high probability.

To implement the black-box, we observe that we need not precompute the (non-zero entries of the) sequence \( S(p) \). Instead we can compute non-zero successive entries on the fly. This is because, using common admissible orderings such as total degree or lexicographical orderings, it is relatively simple to generate successive polynomials in the sequence. Otherwise, we know that each admissible ordering is characterized by a non-negative real square matrix \( W \) and if \( W \) has computable (sic) elements, we can also compute successive non-zero entries of \( S(p) \). In practice, if one has an a priori bound on the degree of the polynomials, it is known in computer algebra
work that one can always assign integer values to $W$. In the work on Simulation of Simplicity, it had been that the partial derivatives of determinantal functions reach a constant value after only a few partial derivation. The flexibility in choosing admissible ordering could conceivably allow us to exploit such observations systematically. It seems that total degree orderings converge to constant functions the slowest. However, we know very little about such choices.

For the evaluation of univariate polynomials, we have relatively complete knowledge about the worst case complexity of the two extremes of sparsity: the dense polynomial (i.e., a polynomial of degree $d$ has $s = d + 1$ non-zero coefficients) and the totally sparse polynomial which consists of only one monomial (i.e., the addition chain problem). From Yao (1976) and Pippenger (1980), we may deduce that evaluating an $n$-variate polynomial with $s$ monomials, where the degree in each variables is less than $d$ has complexity at most

$$w + v \log d + H/\log H + o(H/\log H)$$

where $H = ns \log d$ and $v = \min\{s, n\}$, $w = \max\{s, n\}$. This bound is tight if we insist on evaluating the monomials as separate entities. For evaluating polynomials, this is a real restriction. For instance, evaluating $(x_1 + x_2)(x_3 + x_4)$ as a polynomial requires strictly fewer arithmetic operations than evaluating the four component monomials separately.

In our applications, we not only want to evaluate a polynomial but may need to evaluate some of its derivatives as well. There is literature on evaluating a dense univariate polynomial and all of its derivatives (e.g., Aho, Steiglitz and Ullman (1975)). They are not directly suitable for our application since we need sparse multivariate polynomials, and generally do not evaluate derivatives of every order. (See also Iri (1986) and Iri and Kubota (1987).) The sparse model of polynomial evaluation lends itself nicely to the derivative evaluation problem since a derivative polynomial is at least as sparse as the original polynomial.

Instead of using the asymptotically optimal method of Pippenger, we use a simpler scheme based on Yao’s method. The method basically says that given $x$ raised to each power of 2 less than $d$, we can evaluate $x^d$ in $\log d [1 + o(1)]$ steps. Hence any $s$ term $n$-variate polynomial can be evaluated in $\frac{s \log d}{\log \log d} [1 + o(1)]$ steps, (assuming an initial cost of $n \log d$ to evaluate each variable raised to powers of 2 less than $d$.) For our application, the advantage of this simple scheme comes from the potential need to
evaluate derivatives: each subsequent derivative can be evaluated using the same method. The space usage is $O(n \log d)$ to store the variables raised to powers of 2. For moderate values of $s$, this space can be improved without loss of speed: by evaluating all the necessary powers of one variable before proceeding to the next variable, and accumulating them in each of the $s$ 'partially formed' monomials, we reduce the space to $O(s + \log d)$.

In the univariate case, we can actually get a uniform method of polynomial evaluation whose complexity is optimal (up to smaller order terms) for the entire range of $s$:

$$L(1, s, d) \leq \log d + \frac{s \log(d/s)}{\log \log(d/s)} \left[1 + o(1)\right]$$

To see this, we use the 'generalized Horner factorization' of a polynomial:

$$p(x) = a_1 x^{d_1} (1 + a_2 x^{d_2} (1 + \ldots (1 + a_s x^{d_s}) \ldots))$$

where $\sum_{i=1}^s d_i = d$. We first evaluate $x^k$ where $k \leq d$ range over powers of 2 in $\log d$ steps. Now evaluate each $x^{d_i}$ in $\frac{\log(d_i)}{\log \log(d_i)} \left[1 + o(1)\right]$ steps. Summing over all $i = 1, \ldots, s$, and observing that the complexity is maximized when each $d_i = d/s$, we get our result. Unfortunately, this method does not seem to generalize to multivariate polynomials.

At a small extra cost in space, we can speed up the evaluation of subsequent entries in $S(p)$. This is based on the observation that evaluating successive polynomials in the sequence $S(p)$ are closely related. The idea is that in the above method for evaluating the monomial $x_1^{d_1} x_2^{d_2} \ldots x_n^{d_n}$, we keep around the values of $x_i^{d_i}$ for each $i$. Then in the derivative (say with respect to $x_1$), we must evaluate the monomial $x_1^{d_1-1} x_2^{d_2} \ldots x_n^{d_n}$ and this can be obtained in $n + \log d_1 / \log \log d_1$ multiplications (instead of $n \log d_1 / \log \log d_1$). Other methods to trade-off time for space can also be devised.

While the discussion on evaluating polynomials is mainly of theoretical interest, if one were interested in a fixed family of polynomials (such as a hardware implementation of an algorithm for Voronoi diagrams) then it pays to achieve the minimum number of arithmetic operations.

8. Directions for future work

1. Let us first make a remark about the scheme in practice. Although the use of admissible orderings point out a vast range of possible symbolic perturbations, there has been no practical experience with any of them, save the lexicographical ordering perturbation for determinants that has
been implemented by Edelsbrunner-Mücke. Of course, they arrive at the pure lexicographical scheme by direct arguments concerning perturbing by a symbolic $\epsilon$-amount. (This is also the method by which some symmetry breaking schemes for the Simplex method are justified.) Currently, all that our new method offers is a simplified approach to a large class of symbolic perturbation techniques (avoiding the direct $\epsilon$-justifications), together with the 'possibility' of some still-undiscovered polynomial evaluation black-box (based choosing some admissible ordering) that has better properties than the pure lexicographical one.

2. The black-box we provide is not invariant under change of coordinate axes. For example, with $f(x, y) = x$ and $g(x, y) = y$ our test yields the inequality $f(0, 0) > g(0, 0)$ (assuming the ordering $x > y$). But if we rotate the coordinate axes of the plane by angle $\theta$, the transformed functions become $F(x, y) = x \cos \theta - y \sin \theta$ and $G(x, y) = x \sin \theta + y \cos \theta$. If $|\theta| > \pi/2$ then we get $F(0, 0) < G(0, 0)$, which is opposite to the unrotated case.

3. We have defined our scheme for evaluation of polynomials. It is easy to extend this to rational polynomial functions, $f(x) = p(x)/q(x)$. Such rational functions might arise in, say the solution of linear systems (using Cramer's rule). We form the sequence $S(f) = (f_1, f_2, \ldots)$ as usual. It is not hard to check that again, this sequence has a finite number of non-zero entries, and there are constant entries among them. Hence the 'first non-zero element' in $S(f; a)$ is well-defined. Now the ordered rings become ordered fields.\footnote{These remarks are made explicit in Yap (1990).}

4. What about evaluation of real algebraic functions? By a real algebraic function we understand a continuous real function $r(x)$ (valid for some range of $x$) such that for some polynomial $p(z; x)$,

$$p(r(x); x) = 0.$$  

For each $x := a$, we get an algebraic number $r(a)$. (As discussed in section 2, such algebraic numbers are represented exactly.) Perhaps the most common instance is the square-root function $r(x) = \sqrt{x_1}$, where $p(z; x) = z^2 - x_1$. If we want to compare the distance $d(p, q)$ between two points $p, q$ with some value $v$, then we could clear square-roots and set up the test as $d(p, q)^2 - v^2 = 0$. One cannot do such transformation of the test polynomial with impunity because they affect the infinitesimals. For example, if $f(x) = x^2$, $F(x) = f(x)^2 = x^4$ and $g(x) = x^3$ then we have $f(0) > g(0) > F(0)$ assuming total degree admissible ordering. Hence, the comparison $f(0) : g(0)$ is not
equivalent to $F(0) : g(0)$ (though it would be equivalent to $F(0) : G(0)$ where $G = g^2$). To treat algebraic functions in general, we can proceed as follows: suppose $r(x)$ is an algebraic function satisfying $p(r(x); x) = 0$. Then the partial derivatives of $r(x)$ is obtained by the chain rule, and we can define the sequences $S(r)$ and $S(r; a)$ in the usual way. For instance, in the case of the square root function, $\frac{dr(x)}{dx} = 1/2r(x)$ and $\frac{d^2r(x)}{dx^2} = -1/r(x)^3$. Note that in general, it is not obvious that we are guaranteed to get a non-zero result after a finite number of evaluations. The computational details of carrying out such a scheme seems nontrivial.

5. In some applications, there are restrictions on the perturbation. In some hidden surface removal algorithms, there is a basic test to see if two triangles $A, B$ obscure one another. Let the polygon $C$ be the intersection of the projections $A', B'$ of the triangles onto the viewplane ($xy$-plane). So $C$ has at most 6 sides. Take any vertex $P$ in $C$. By projecting $P$ back to $A$ and $B$, we get $P_A$ and $P_B$. The test reduces to comparing the $z$-coordinate of $P_A$ and of $P_B$. If their $z$-coordinates are equal, we want to do the same test on a perturbed version of $P$. We must ensure that perturbed $P$ does not go outside $C$: conceptually, we add to $P$ an infinitesimal vector directed to the interior of $C$. Unfortunately, our present perturbation scheme is not helpful here.

6. Another more far reaching direction is to handling uncertainty in the input data. This is related to the issue of treating finite precision arithmetic. A basic question to be faced is this: what is the meaning of degeneracy in the world of imprecise input or in the world of finite precision? This is a difficult question, related, for instance, to the problem defining the digital analogues of lines and circles.

Acknowledgement

This work was first conceived at the Workshop on Geometric Reasoning, Oxford University, June 30-July 3, 1986. The workshop was supported by grants from the NSF and the Science Research Council of Great Britain. I am indebted to members of the workshop (in particular, Micha Sharir and Christoph Hoffman) who commented on preliminary ideas of this scheme.

References


