# Lecture 3
# FP APPROACHES

We will look at some of the approaches to nonrobustness based on FP computation. Especially interesting is the notion of "finite precision geometry". After all, if nonrobustness is a "geometric phenomenon", it makes sense to modify the geometry to reflect the finite precision we assume in numbers.

## §1. Arithmetic Fixes

We describe an "arithmetic approach" for robustness, proposed by Ratschek and Rokne [10]. They gave algorithm called ESSA to solve the following [1] **Sign of Sum Problem** (SoS Problem). Given a sequence

$$a_1, a_2, \ldots, a_n$$

of numbers, we want to determine the sign of the sum

$$S = \sum_{i=1}^{n} a_i. \tag{1}$$

In the same spirit, an early paper of Ottmann, Thiemt and Ullrich [9] show that the use of an accurate scalar product operator leads to improved robustness in segment intersection algorithms.

An exact solution for this problem (such as provided by ESSA) would avoid all non-robustness for our convex hull algorithms. To see this, observe that the convex hull algorithms we have described ultimately depend on the predicate $LeftTurn(p, q, r)$. This amounts to determining the sign of a certain $2 \times 2$ (or $3 \times 3$) determinant. Moreover, a $2 \times 2$ determinant is an expression of the form $ad - bc$, and this can be reduced to SoS Problem above (with $n = 2$).

ESSA (which stands for "Exact Sign of Sum Algorithm") assumes the following: we are computing in some system $F(2, t)$ and

$$n \leq 2^{t-1}. \tag{2}$$

Note that overflow or underflow in the exponent does not occur in $F(2, t)$. By sorting the inputs in a preprocessing step, we may further assume

$$a_1 \geq a_2 \geq \cdots a_k > 0 > a_{k+1} \geq \cdots \geq a_n.$$

By renaming the negative inputs $a_{k+i}$ as $-b_i$ ($i = 1, \ldots, \ell = n - k$) the problem is now reduced to determining the sum

$$S = \sum_{i=1}^{k} a_i - \sum_{j=1}^{\ell} b_j.$$

Let $E_i$ be the exponent part of $a_i$ and $F_j$ the exponent part of $b_j$. The algorithm goes as follows:

[1] But be aware that "SoS" often refers to a more well-known "SoS Technique" (Simulation of Simplicity Technique) from Edelsbrunner and Muecke for treating data degeneracies.

ESSA
Input: $a_i, b_j$ $(i = 1, \ldots, k, j = 1, \ldots, \ell)$ as above.
Output: The sign of $S$
1. (BASIS) Terminate with the correct output in the following cases:
   1.1 If $k = \ell = 0$ then $S = 0$.
   1.2 If $k > \ell = 0$ then $S > 0$.
   1.3 If $\ell > k = 0$ then $S < 0$.
   1.4 If $a_1 \geq \ell 2^{F_1}$ then $S > 0$.
   1.5 If $b_1 \geq k 2^{E_1}$ then $S < 0$.
2. (AUXILIARY VARIABLES)
   $a' = a'' = b' = b'' = 0$;
3. (PROCESSING THE LEADING SUMMANDS)
   CASE $E_1 = F_1$:
      If $a_1 \geq b_1$ then $a' \leftarrow a_1 - b_1$
      Else $b' \leftarrow b_1 - a_1$;
   CASE $E_1 > F_1$:
      $u \leftarrow 2^{F_1 - 1}$ if $b_1 = 2^{F_1 - 1}$, else $u \leftarrow 2^{F_1}$;
      $a' \leftarrow a_1 - u$, $a'' \leftarrow u - b_1$;
   CASE $E_1 < F_1$:
      $v \leftarrow 2^{E_1 - 1}$ if $b_1 = 2^{E_1 - 1}$, else $v \leftarrow 2^{E_1}$;
      $b' \leftarrow b_1 - u$, $b'' \leftarrow u - a_1$;
4. (UPDATE BOTH LISTS)
   Discard $a_1, b_1$ from list.
   Insert $a', a''$ into the $a$-list provided $a', a'' \neq 0$.
   Insert $b', b''$ into the $b$-list provided $b', b'' \neq 0$.
   Update the values of $k, \ell$ and return to Step 1.

To see the correctness of this algorithm, we note the following properties

- *Each arithmetic operation is error free.* For instance, in case $E_1 = F_1$, the value $a' \leftarrow a_1 - b_1$ is computed exactly. In case $E_1 > F_1$, and $u = 2^{F_1}$, then $a'' \leftarrow u - a_1$ is exact. To see that $a' \leftarrow a_1 - u$ is also exact, we use the fact that the non-execution of Step (1.4) implies $a_1 < \ell 2^{F_1}$ and hence by assumption (2), we have $a_1 < 2^{F_1 + t - 1}$. This then implies $E_1 < F_1 + t$. Thus $a_1 - 2^{F_1}$ is exact.
- *Each updating of the lists does not change the (exact) value of the sum $S$.* Combined with the previous remark, this implies the partial correctness of ESSA: *if it terminates, the answer is correct.*
- Next, in both the $a$- and $b$-lists, their sums are positive and strictly decreasing towards 0.
- Moreover, we note that no number has an exponent less than $\min\{E_k, F_\ell\} - t + 1$. where $E_k, F_\ell$ are the values at the very begining of the computation. This, combined with the previous remark implies termination.

Let us discuss some practical issues: Termination, in practice, tends to be faster than the above analysis would suggest. When we apply it to our $2 \times 2$ determinant $ad - bc$, there are two places where the numbers may overflow or underflow a finite precision number system (not $F(2,t)$ but $F(2, t, e_{\min}, e_{\max})$). First the values $a, b, c, d$ really arises as the difference of two floating point numbers So this may cause an underflow in one of the values $a, b, c, d$. More precisely, the determinant really comes from a special kind of $3 \times 3$ matrix (see equation (1) in Lecture 2). Using the notation from Lecture 2, the determinant to be evaluated is really

$$\Delta = (b - a)(c' - a') - (c - a)(b' - a') = bc' - ac' - ac' - cb' + ab' + ca'.$$

So this underflow problem can be avoided if we assume the input to ESSA has these 6 terms instead of the 2 terms. However, any of the 6 terms may still overflow if the original data are double-precision. There are nevertheless well-known ways to handle this (see the original paper).

**Summation Problem.** Let us briefly consider the more general problem of computing the sum $S$ in (1), not just its sign. This is a very important problem in its own right, and has been investigated in detail. For instance, whenever we compute a scalar product of two vectors, we need to evaluate such a sum. Mathematically, this is a triviality, but within FP computation, this is[2] no longer true: the computational techniques explored tries to answer the question "how can we organize the computation of this sum to minimize the error?" Higham [5, chapter 4] treats this problem. There is in this context a remarkable algorithm called "compensated summation" for this problem. Suppose $a, b$ are floating point numbers with $|a| \geq |b|$ and consider[3] inside the the sum $\widetilde{s} = [a + b]$ (figure 1).

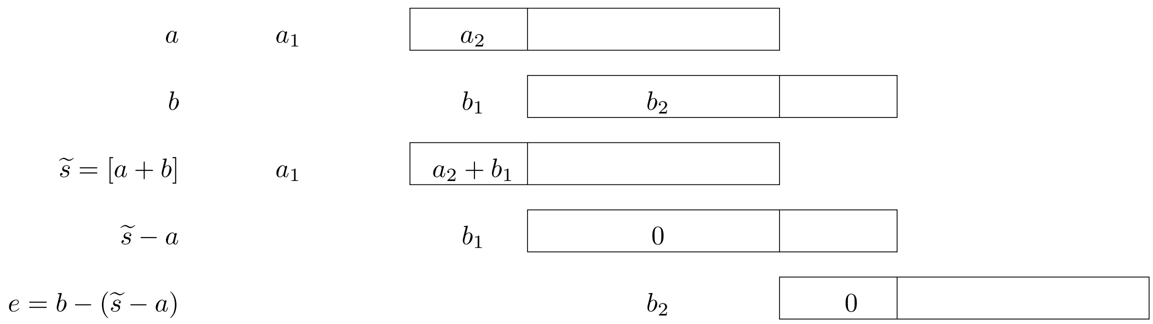



Figure 1: Estimated error, $e = (b - ((a + b) - a))$.

Figure 1 suggests that if we compute $e$ using in the order

$$e = (b - ((a + b) - a))$$

then $|e|$ is roughly the error in $s$ as an approximation to $a + b$. More precisely, if

$$\widetilde{e} = [b - [\widetilde{s} - a]] = [[a - [a + b]] + b]$$

then, in base 2, Dekker (1971) shows that

$$a + b = \widetilde{s} + \widetilde{e}.$$

This gives rise to the following compensated summation algorithm:

Compensated Summation Algorithm
Input: Floating point numbers $a_1, \ldots, a_n$
Output: Approximate sum $s = \sum_{i=1}^{n} a_i$
  $s \leftarrow 0$; $e \leftarrow 0$;
  for $i = 1$ to $n$
    $temp \leftarrow s$;
    $a \leftarrow [a_i + e]$; (compensated summand)
    $s \leftarrow [temp + a]$;
    $e \leftarrow [temp - s] + a]$; (error estimate)

[2]This is true of many issues in FP computation: mathematical trivialities that aren't.

[3]Here, we use Kahan's notation that $[\ldots]$ denotes the floating point approximation of the real number $\ldots$ expressed inside the square brackets. This is just a more compact notation for $fl(\ldots)$ that we had used before.

The idea is that the error is estimated at each step and added back to the partial sum. Note that this error estimate is not exact because Dekker's analysis depends on the inequality $|a| \geq |b|$ which we give up. Nevertheless, it can be shown (see [7, pp. 572–573] and [1]) that the computed value is equal to

$$\widetilde{s} = \sum_{i=1}^{n} (1+\delta_i) a_i, \qquad |\delta_i| \leq 2u + O(nu^2).$$

## §2. Robust Line Segment Intersection

We consider another fundamental problem in computational geometry: given a set $S$ of (line) segments in the plane, we want to compute their intersections. This is a very basic problem that arises in VLSI circuit design, hidden line elimination, clipping and windowing, physical simulations, etc.

We should clarify that we are really computing a **segment arrangement**. That is, we want to compute the **1-skeleton** (or network) determined by the line segments and their intersections: a 1-skeleton is basically a graph that is embedded in the plane: its vertices are the endpoints of segments and the intersection points. Its edges is the open portions of segments bounded by two vertices.

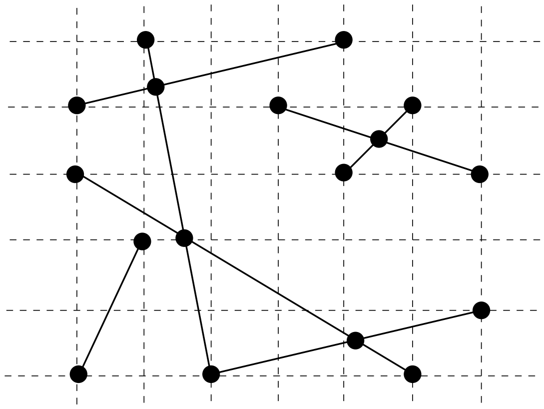


Figure 2: 1-Skeleton of a set of line segments

There are several algorithms for this problem. The asyptotically best algorithm is from Chazelle and Edelsbrunner, with running time $O(k + n \log n)$ where $k$ is the number of intersection points and $n$ is the number of segments. Of course, $k$ is an output parameter and ranges from 0 to $n^2$. More practical algorithms based on randomization have been given Clarkson and also Mulmuley. Algorithms that takes into account robustness issues have been studied by Greene and Yao [3], Milenkovic [8], and Sugihara [11]. The approach of Hobby [6], Guibas and Marimont [4] and Goodrich, et al [2] are based on the concept of "snap rounding", which we next explain.

Recently, Preparata et al considered another issue, namely modifying the primitives using in these algorithms so that the algebraic degree is as small as possible. Although this clearly can help FP computations, it is useful even in AP computation.

**Bentley-Ottman Algorithm.** The classic algorithm for this problem is from Bentley and Ottmann, and this seems to be the most practical [6]. This algorithm takes time $O((n+k)\log n)$, and is quite easy to describe. Later we will describe Hobby's snap rounding in the context of this algorithm.

We assume a vertical sweepline $L$ with equation $y = y_0$. The line sweeps from left to right (so $y_0$ is increasing. The end points of every input segment are first placed in a priority queue $Q$: an point $(x, y)$ in $Q$ has priority $x$. As $L$ sweeps from left to right, we pull events from $Q$. At any moment, we keep track of the set of segments that intersect $L$. These segments are sorted by their the $y$-coordinate of their intersection with $L$. We store them in a balanced binary tree $T$ (e.g., a red-black tree, etc). Moreover, for each pair of adjacent segments in $T$, if they intersect at a point $q$ to the right of $L$, then we insert $q$ into $Q$.

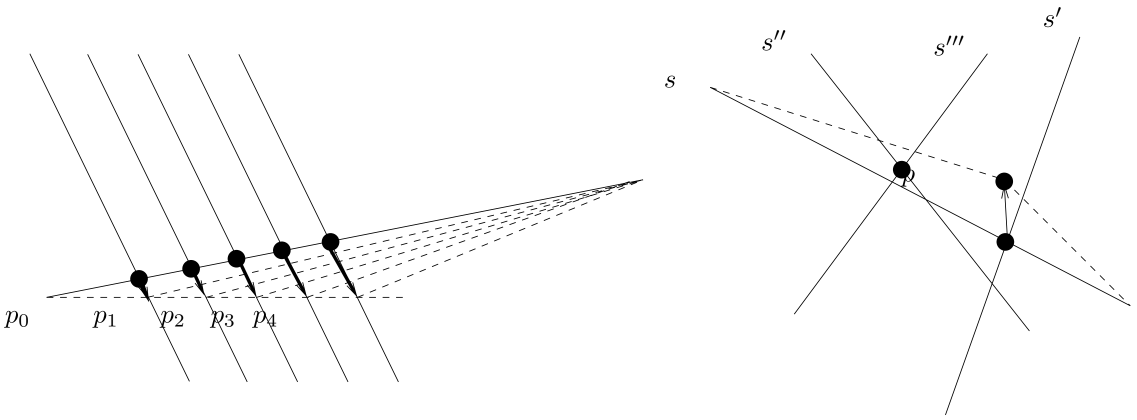
Here is how we update $T$ with each event. If an end point, we delete the corresponding segment from $T$ If a start point, we insert the corresponding segment into $T$ If an intersection point, we exchange the relative order of the two segments in $T$.

**Yao-Greene Grid Model.** We consider the following finite-precision model of plane geometry: let $G_2 = \mathbb{Z} \times \mathbb{Z}$ be the **(unit) grid**. All representable points must come from $G_2$. Suppose $s, s'$ are two representable line segments (i.e., their endpoints are in $G_2$). If they intersect at a point $p$, this point is generally not representable. So we must round it to some nearby grid point $p'$. If $s = [a, b]$, then $p'$ no longer lie on $[a, b]$ – failing a fundamental identity we discussed in Lecture 1:

$$OnLine(Intersect(s, s'), s)$$

must be identically true if $Intersect(s, s')$ is defined. To restore this identity, we now replace line segments with **polylines** which is just a finite polygonal path $[a_1, a_2, \ldots, a_n]$. Thus $s$ is now the polyline $[a, p', b]$ and there is a similar modification for $s'$. In general, for a polyline $s$, we can "snap" a point $p$ in $s$ to a grid point $p'$ and change $s$ accordingly.

This defines our basic model, first studied by Greene and Yao for this problem. What are the issues?




Drifting

Topological Inversion relative to point $p$

Figure 3: Issues in Snap Rounding

- When we snap a point of $s$, we may generate new intersections. If we now snap these intersections points, we may obtain other new ones. Does this terminate? How often do we need to do this?
- Even if we do not generate new intersections, the old intersection point may move by some distance. It is not hard to see that this distance is not bounded.
- If we keep modifying a polyline, the result may keep drifting away from its original position. There may be no constant bound on this drift.

- Topologically, we may create a different topological configuration (imagine a segment moving past an intersection point).

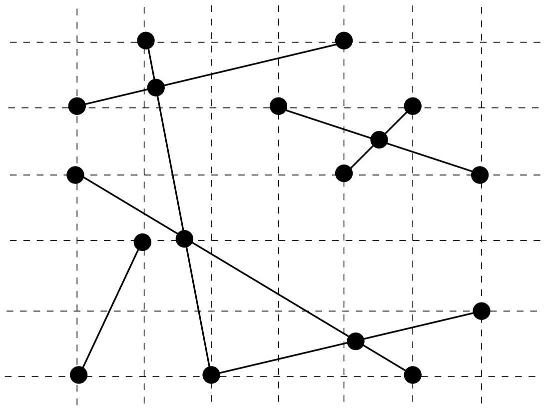
This list of problems [3] seems formidable. So it is somewhat surprising to see the simple solution described next.

**What is an acceptable topological change?** Topological change is inevitable in finite precision computation. But we distinguish between "degeneration of topology" (which is acceptable) versus "inversion of topology" (which is not). For instance, if an edge in a planar arrangement of segments is collapsed to a point and it does not affect other relations (except for the obvious ones forced by this collapse), this is called a "degeneration". On the other hand, if a vertex inside a cell is modify to lie outside the cell, this is an "topological inversion". Of course, there is a trivial solution for any segment arrangement: collapse everything to a point. So, clearly these remarks needs to be supplemented by metric properties such as a bound of the amount of degeneration.

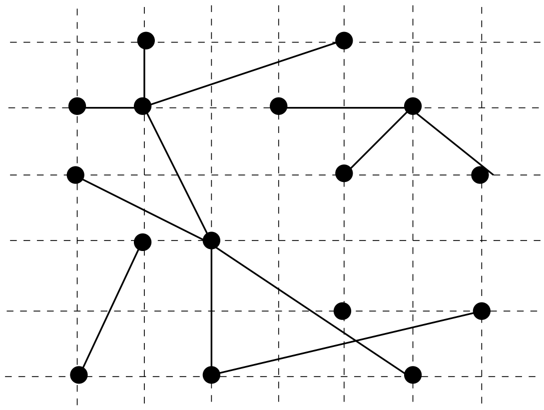
**Snap Rounding.** It is convenient to introduce the half-open interval $R_1 = (-\frac{1}{2}, \frac{1}{2}]$. Also let $R_2 = R_1 \times R_1$. Call $R_1$ and $R_2$ the **rounding interval** or **square**, respectively. It defines the rounding model for us: a number $x$ is rounded to the unique integer $(x + R_1) \cap \mathbb{Z}$, sometimes denoted $\lfloor x \rceil$. In other words, we round up numbers: 2.5 rounds to 3. Similarly, a point $p$ is rounded to the unique point $(p + R_2) \cap G_2$, also denoted $\lfloor p \rceil$. Note that "$+$" here is the Minkowski sum.

We can view rounding from the view point of the grid: let $\overline{R}_1 = [-\frac{1}{2}, \frac{1}{2})$ (mnemonic: $\overline{R}_1 = -R_1$). If $i \in \mathbb{Z}$, then $\lfloor x \rceil = i$ iff $x \in i + \overline{R}_1$. Similarly, we may define $\overline{R}_2 = \overline{R}_1 \times \overline{R}_1$ and define the square of influence of a grid point $g \in G_2$ to be $g + \overline{R}_2$.

Let $S$ be a set of segments, where each pair of segments intersect transversally (we do not allow overlap). Define $H$ to be the set comprising all the end points of segments in $S$ and also all intersection points. So $H$ is finite. Let the rounded point set $\lfloor H \rceil$: these are called the "hot points".



(a) Original Skeleton



(b) Skeleton based on Hot Grid Points

Figure 4: Snap Rounding Based on Hot Points

A polyline $s$ is said to be **near** a point $p$ if $p+\overline{R}_2 \cap s$ is non-empty. If a polyline $s$ that passes near a hot point $p$, we snap $s$ towards $p$. More precisely, If $q \in p + \overline{R}_2 \cap s$, then we snap $q$ to $p$ (recall the definition of this snap operation above). It is not hard to see that this does not depend on our choice of $q$. But in the sequel, we will assume that $q$ is the point in $s$ closest to $p$. The number of vertices in $s$ increases by 1.

**Hobby's Theorem on Snap Rounding.** Given a segment $s \subseteq \mathbb{R}^2$, define the map $D_T(s)$ to comprise all the segments $[a, b]$ such that there exists $[a', b'] \subseteq s$ such that

$$\lfloor H \rceil \cap ([[a', b'] + R_2) = \{a, b\}.$$

If $[a', b']$ is a maximal segment with this property, we call $[a', b']$ the "preimage" of $[a, b]$. Hence, each segment in $D_T(s)$ has at least one preimage in $s$. Below we will prove that these preimages are very well-behaved (in particular, there is a unique preimage for each segment in $D_T(s)$).

**Lemma 1** *For any segment $s$ there is a direction $(1, \delta)$ (where $\delta = \pm 1$) such that no two points in $G_2 \cap (s+R_2)$ have the same $x + \delta y$ value.*

*Proof.* If $s$ has positive slope, let $\delta = +1$ else $\delta = -1$. Let $(x_i, y_i) \in s$ for $i = 1, 2$, and $(\lfloor x_i \rceil, \lfloor y_i \rceil) \in (s + R_2) \cap G_2$. Then

$$\lfloor x_1 \rceil + \delta \lfloor y_1 \rceil \neq \lfloor x_2 \rceil + \delta \lfloor y_2 \rceil$$

unless the two points are identical. To see this, suppose the slope is positive and $\lfloor x_1 \rceil + \lfloor y_1 \rceil = c$ for some $c$. If $\lfloor x_2 \rceil + \lfloor y_2 \rceil = c$, then it is easy to see that they must be diagonal points of a unit square. Indeed, they must be the NW and SE corners of some unit square. But this is impossible. **Q.E.D.**

A collection $S$ of segments are **essentially disjoint** if any pair are disjoint or intersect at their end points only. We also write $D_T(S)$ for the set union of $D_T(s)$, ranging over all $s \in S$.

**Lemma 2** *If $S$ have endpoints in $T$ and are essentially disjoint, then the set $D_T(S)$ are also essentially disjoint.*

*Proof.* Let $s \in S$. The previous lemma ensures that there is a coordinate system

$$(\xi, \eta) = \left( \frac{x + \delta y}{\sqrt{2}}, \frac{-\delta x + y}{\sqrt{2}} \right)$$

such that the endpoints of segments in $D_T(s)$ all have all different $\xi$ coordinates. The transformation $(x, y) \mapsto (\xi, \eta)$ is given by

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix} = M \begin{pmatrix} x \\ y \end{pmatrix}, \qquad \text{where} \quad M = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & \delta \\ -\delta & 1 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

The transformation is a rotation, *i.e.*, $\det M = 1$ and the inverse $M^{-1}$ is given by its transpose $M^T$. E.g., if $\delta = 1$, the points $(\pm 1, 1)$ map to $(1/\sqrt{2}, 0)$ and $(0, 1/\sqrt{2})$.

**Lemma 3** *The set of these preimages forms a cover of $s$. Each segment $D_T(s)$ has a unique preimage.*

*Proof.* The fact that they form a cover of $s$ follows from the fact that if any maximal open subsegment $(a, b) \subseteq s$ that is dijoint from any preimages, then there must be a preimage of the form $[b, c] \subseteq s$ (for some $c$). This implies that $[a, c]$ is contained in a preimage, contradicting the assumption that $[b, c]$ is maximal.

To show uniqueness, suppose $[a', b']$ and $[a'', b'']$ are preimages of $[a, b]$. Note that the transformation $(x, y) \mapsto \tau(x, y) = (\xi, \eta)$ is monotonic on $s$ in the sense that for points $(x, y) \in s$, the function $\xi(x, y)$ is increasing with $x$. To see this, let $y = mx + c$ for some $m, c$. Then $\xi(x, y) = x + \delta y = x + \delta(mx + c) = x(1 + \delta m) + \delta c$. But $\delta m > 0$ be choice of $\delta$. Writing $a' = (a'_x, a'_y)$ and $b' = (b'_x, b'_y)$ and assuming $a'_x \le b'_x$, we get $\xi(a') \le \xi(b')$. By maximality of preimages, these facts implies $a' = a''$. Similarly, $b' = b''$. So these preimages are the same. **Q.E.D.**

**Lemma 4** *If the endpoints in $D_T(S)$ maps to $q_1 = (\xi_1, \eta_1), \ldots, q_m = (\xi_m, \eta_m)$, and $\xi_1 < \xi_2 < \cdots < \xi_m$, then they describe a piecewise linear function $F$ on the interval $[\xi_1, \xi_m]$ given by*

$$F(\xi) = \frac{\eta_i(\xi_{i+1} - \xi) + \eta_{i+1}(\xi - \xi_i)}{\xi_{i+1} - \xi_i}$$

*when $\xi_i \le \xi \le \xi_{i+1}$.*

*Proof.* This follows from the fact that the preimages of $[q_i, q_{i+1}]$ are ordered along the segment $s$ in the expected way. [Incomplete] **Q.E.D.**

If $s' \in S$ is a distinct segment, then a similar function $F'$ based on $D_T(s')$ can be defined.

Note that the slope of $s'$ may be different from that of $s$. This will lead to different coordinate systems for $F$ and $F'$. To fix this, we may first rotate the coordinate system so that $s$ and $s'$ have both positive slopes or both negative slopes. It is easy to see this is always possible.

These functions approximate the lines $\ell$ and $\ell'$ through $s$ and $s'$ (respectively).

Since $s, s'$ are essentially disjoint, we may assume THAT IN THE $(\xi, \eta)$-coordinate system, wlog that $s$ is below $s'$ whenever they intersect a common vertical line $\xi = \xi_0$. So it suffices to show that

$$F(\xi) \le F'(\xi) \tag{3}$$

for all $\xi \in \{\xi_1, \ldots, \xi_m\} \cup \{\xi'_1, \ldots, \xi'_{m'}\}$.

Parametrize the lines $\ell$ so that $(\xi, \ell(\xi)) \in \ell$ for all $\xi$. Similarly, assume $(\xi, \ell'(\xi)) \in \ell'$. Since the segment in $D_T(s) \subseteq s_j + R_2$, and similar for $s'$, the difference $F(\xi) - \ell(\xi)$ is limited to the range of the $\eta$ coordinates in $R_2$. This ranges over the intervals $(-\frac{1}{2}, \frac{1}{2})$ or $[-\frac{1}{2}, \frac{1}{2})$. [CHECK]

Then our assumptions that $\ell(\xi) \le \ell'(\xi)$ for all $\xi$ is the range of interest implies that

$$F'(\xi_i) \ge \eta_i, \qquad \text{if } \ell'(\xi_i) \ge \eta_i + \frac{1}{2}.$$

Otherwise, $\ell'(\xi_i) \in \eta_i + R_1$ and the definition of $D_T$ forces $F'(\xi_i) \ge \eta_i$. incomplete

Similarly, we argue that $F(\xi'_j) \le \eta'_j$. This proves equation (3).

**Q.E.D.**

**Snap Rounding Based on Bentley-Ottmann's Algorithm.** For simplicity, we may assume that the input segments are already representable.

STEP 1. We modify the Bentley-Ottman algorithm as follows. We first run the original algorithm as a "first pass". The intersection points are symbolically represented (by the pair of defining segments). Let $H$ be the set of all endpoints and the intersection points found in this way.

STEP 2. Compute the set of hot points: $\lfloor H \rceil$. Now we want to snap each segment $s \in S$ to all the hot points $p$ that it is near to. More precisely, for each $s$ we compute a set of pairs $\{(q_i, p_i) : i = 1, \ldots, k\}$ (for some $k$) such that $p_i$ is a hot point that $s$ is near to, and $q_i$ is the point in $s$ closest to $p_i$. Then we snap $q_i$ to $p_i$, in an natural order determined by the $q_i$'s along $s$. But how do we detect these $p_i$'s? We can do this by modifying the same sweepline algorithm of Bentley-Ottman: for each hot point $p$, we add the four unit segments corresponding to the boundary of $p + \overline{R}_2$ to the input set. Thus the segment $s$ is near to $p$ iff it intersects one of these unit segments of $p$.

We can take advantage of the fact that these unit segments are horizontal or vertical and in special positions relative to the grid $G_2$. An efficient implementation (using the idea of batching, and keeping the unit segments separate from the original segments) is given in the original paper of Hobby.

**Assessment.** Although the snap rounding solution solves the list of problems noted by Yao and Greene, we note another potential problem called "braiding". It is a problem that is intermediate between as topological inversion and topological degeneration:

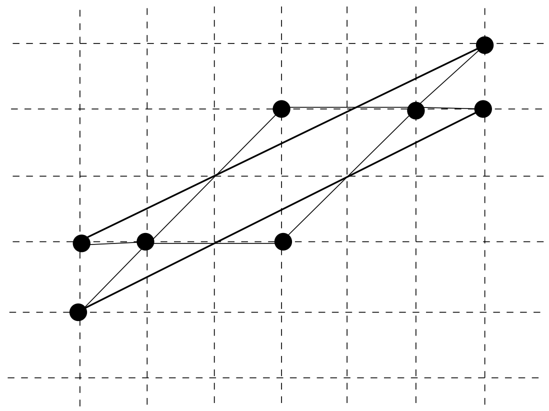


Figure 5: Braiding Behavior in Snap Rounding

**OTHER APPROACHES** Yao and Greene's.

Milenkovic's

Sugihara's.

Exercises

**Exercise 2.1:**
(i) Complete the proof that the ESSA algorithm is correct.
(ii) Describe modifications to the algorithm when exponents are limited in the range $[e_{\min}, e_{\max}]$. □

**Exercise 2.2:** Fix a precision $t \geq 1$. Consider snap rounding in the following "floating point grid" $G_2^t = F(2,t) \times F(2,t)$. Show an example that shows that the snap rounding theorem of Hobby fails in $GF_2$. $\square$

End Exercises

# References


[1] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

[2] M. Goodrich, L. J. Guibas, J. Hershberger, and P. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 284–293, 1997.

[3] D. H. Greene and F. F. Yao. Finite-resolution computational geometry. *IEEE Foundations of Computer Science*, 27:143–152, 1986.

[4] L. Guibas and D. Marimont. Rounding arrangements dynamically. In *Proc. 11th ACM Symp. Computational Geom.*, pages 190–199, 1995.

[5] N. J. Higham. *Accuracy and stability of numerical algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[6] J. D. Hobby. Practical segment intersection with finite precision output. *Computational Geometry: Theory and Applications*, 13:199–214, 1999.

[7] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley, Boston, 2nd edition edition, 1981.

[8] V. Milenkovic. Double precision geometry: a general technique for calculating line and segment intersections using rounded arithmetic. In *Proc. 30th Annual Symp. on Foundations of Comp.Sci.*, pages 500–506. IEEE Computer Society, 1989.

[9] T. Ottmann, G. Thiemt, and C. Ullrich. Numerical stability of geometric algorithms. In *Proc. 3rd ACM Sympos. Comput. Geom.*, pages 119–125, 1987.

[10] H. Ratschek and J. Rokne. Exact and optimal convex hulls in 2d. *Int'l. J. Computational Geometry and Applications*, 1998. To Appear.

[11] K. Sugihara. An intersection algorithm based on Delaunay triangulation. *IEEE Computer Graphics App.*, 12(2):59–67, 1992.