Fighting Inaccuracies: Using Perturbation to Make Boolean Operations Robust

The robustness of Boolean operations between solids is crucial for the usability of a solid modeler like HP PE/SolidDesigner. Unfortunately, geometric modeling is like shoveling sand. With every shovel you pick up a bit of dirt. The numerically imperfect nature of geometric algorithms can challenge HP PE/SolidDesigner's Boolean engine with contradictions and inconsistencies. The Boolean engine uses a perturbation method[1,2] to push the frontier of robustness. This article explains the notion of model consistency and demonstrates what can go wrong inside a Boolean operation and what can be done to come up with a correct result anyway.

Consistency of a Solid

Looking at a solid we usually believe that it is mathematically correct, that is, that the edges are exactly on their adjacent faces and the edges meet exactly at their common vertices. In reality, however, the limited floating-point accuracy of a computer introduces errors. On the microscopic level there are gaps and holes everywhere (see Fig. 1).

The tolerable amount of error is specified by the *modeling resolution*. The system will ignore gaps and holes smaller than the resolution. However, some geometric algorithms, such as the various intersection calculations, tend to magnify errors in certain geometric configurations. This means that given an input where all errors are within limits, the result can be inconsistent in the context of the solid and prohibit the successful completion of the requested Boolean operation.

Solving the Numerical Puzzle

One area in the Boolean operation that is particularly vulnerable to numerical inconsistencies is the intersection graph construction. The graph construction assumes that all intersections of curves defined on one of two intersecting surfaces are also on the intersection track (here the term *on* means closer than the resolution). This is no problem if the surfaces are reasonably orthogonal. However, for intersections between tangential or almost tangential surfaces, a small error in the orthogonal direction of a surface implies a larger error in the direction of the surface, and this assumption becomes false.

Fig. 2 shows a shallow intersection between the two surfaces sf1 and sf2 and the intersection with sf2 of a curve (cv) contained in sf1. The curve/surface intersection point (small colored triangle) has, because of the small distance (epsilon) between cv and its containing surface sf1, moved farther away from the surface/surface intersection track (colored line) than the resolution permits. The smaller the angle β the larger the distance d from the intersection track and hence the larger the inconsistency.

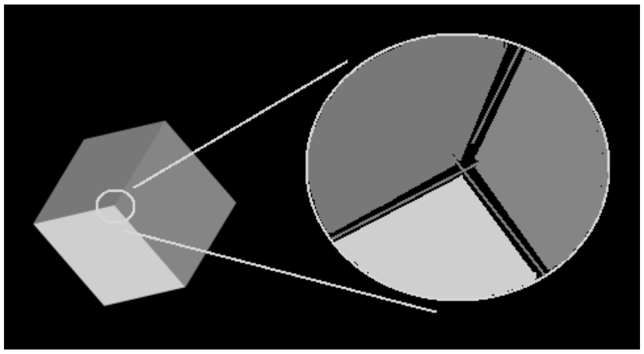

Fig. 1. In solid models edges seem to be exactly on their adjacent faces and meet exactly at their common vertices. In reality, because of the limited floating-point accuracy of a computer, on the microscopic level there are gaps and holes everywhere.

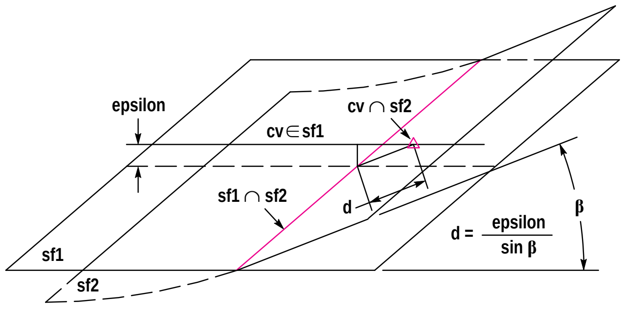

Fig. 2. A shallow intersection between the two surfaces sf1 and sf2 and the intersection with sf2 of a curve (cv) contained in sf1.

Fortunately, there is a method called perturbation than can come to the rescue in situations like this. It solves the inconsistency by moving the curve/surface intersection point along the curve until it is closer than the resolution to the surface/surface intersection track. In Fig. 2 the point will be moved to the left. When the intersection point is moved, a new error is introduced because the point is moved away from sf2. However, the overall error is reduced so that it no longer exceeds the resolution.

The perturbation method can be applied to similar situations in which even the number of intersections has to be corrected. The difference in number is a result the freedom algorithms have below the resolution. They may return *anything* in the range of the resolution.

Two Curve/Surface Intersection Points with One Surface/Surface Intersection Track. Fig. 3 shows a geometric configuration in which the intersection between sf1 and sf2 yields one intersection track (colored line) but the intersection of the curve contained in sf1 with sf2 gives two intersection points (colored triangles) which are farther than the resolution away from the track. The perturbation algorithm moves both points inwards (horizontal arrows) and contracts them into a single point (black triangle) which is closer than the resolution to the intersection track (colored line).

Two Surface/Surface Intersection Tracks with One Curve/Surface Intersection Point. Fig. 4 shows a geometric configuration in which the intersection between sf1 and sf2 yields two intersection tracks (colored lines) but the intersection of the curve contained in sf1 with sf2 gives one intersection point (colored triangle) which is farther than the resolution away from the tracks. The perturbation algorithm splits the single intersection into two and moves them outwards (horizontal arrows) until both are closer than the resolution to an intersection track (colored line).

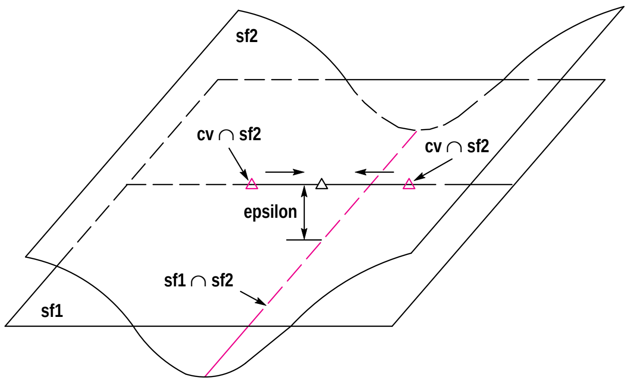

Fig. 3. A geometric configuration in which the intersection between sf1 and sf2 yields one intersection track (colored line) but the intersection of the curve contained in sf1 with sf2 gives two intersection points (colored triangles) which are farther than the resolution away from the track.

References

1. H. Edelsbrunner and E. Mucke, "Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms," *Proceedings of the 4th ACM Symposium on Computational Geometry*, June 1988, pp. 118-133.

2. C. K. Yap, "A geometric consistency theorem for a symbolic perturbation theorem," *ibid*, pp. 134-142.

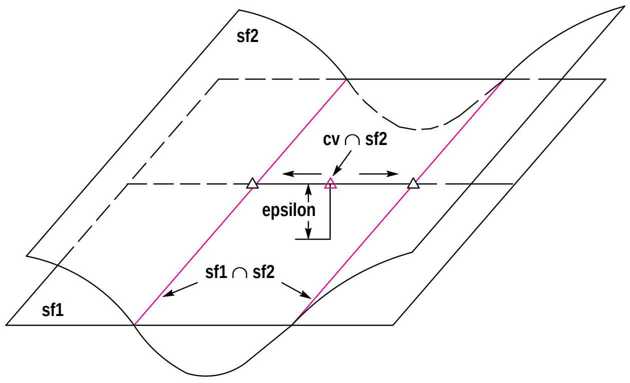

Fig. 4. A geometric configuration in which the intersection between sf1 and sf2 yields two intersection tracks (colored lines) but the intersection of the curve contained in sf1 with sf2 gives one intersection point (colored triangle) which is farther than the resolution away from the tracks.