Algorithmic Tolerances and Semantics in Data Exchange

T. J. Peters[*]
Computer Science & Eng. U-155
University of Connecticut
Storrs, CT 06269-3155 USA
tpeters@eng2.uconn.edu

N. F. Stewart[†]
Dep. IRO
Univ. de Montréal
Montréal, Qc, Canada H3C 3J7
stewart@iro.umontreal.ca

D. R. Ferguson
Boeing
Seattle, WA 98124-2207 USA
drf@espresso.rt.cs.boeing.com

P. S. Fussell
Alcoa Technical Center
Alcoa Center, PA 15069 USA
fussell_ps@atc.alcoa.com

Abstract

Within industrial contexts, a common view of the level of confidence needed in CAD data transfer processes may be expressed as *"I don't mind if the [CAD data-transfer] algorithm sometimes has difficulties, provided I get a warning message that tells me something has gone wrong."* A particularly annoying, even mystifying, CAD data exchange problem occurs in the so-called 'round-trip' problem. There, an acceptable model is transferred from an originating system to a neutral format and then back to the original system, only to find that serious new flaws now appear in the model within the originating system.

1 Introduction

The question of ensuring the integrity of data exchanges between companies is of great importance in Computer-Aided Design (CAD). Indeed, the problem is serious enough, and important enough, to have spawned a fairly large industry dedicated to its solution [5].

One integrity issue is the desire for consistency of topology and geometry data. Design models contain symbolic and numeric data types expressing, respectively, topology and geometry, with some level of interaction between the two. The symbolic data is exact, whereas the numeric data suffers the errors of floating point arithmetic. The unavoidable nature of this disparity is well recognized; attempts to resolve it are based upon the use of tolerance values. When we say that the geometry and topology are inconsistent, we mean, for example, that the symbolic data indicates that two geometric elements are adjacent, yet the corresponding floating point data implies that they are disjoint.

Particularly vexing (sometimes mystifying, even to seasoned CAD practitioners) is the 'round-trip' data exchange problem. This describes the following exchange pathology. The model has topological and geometric consistency in its native system. It is moved to an exchange representation (like STEP) [6], where the consistency is preserved. However, upon being transferred back to the native system, the model may no longer be consistent. We illustrate how and why this can happen, and indeed, why it is even likely for some class of problems.

2 Related Work

Some evidence [5] indicates that model transfer can be improved when the STEP default tolerances are replaced by ones more closely adapted to those of the input system. In a similar vein [5], several illustrative examples of topology and geometry inconsistency were created, and labeled as 'gap problems', where two entities are adjoint by their topological data, yet their geometric data indicates they are disjoint.

New intersection algorithms [2, 3] incorporate tolerance intervals that are adaptive to the numerical errors compounded by extensive geometric computations.

Backward error analysis [1] has been performed relative to the errors for certain geometric algorithms.

3 Example

Consider the simple illustrative geometry of Figure 1. Vertex V0 serves as a trimming endpoint for edge E0, with V0 being within tolerance of E0.

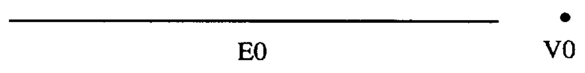

Figure 1: Simple Illustrative Geometry

Now, during model transfer, E0 could be perturbed sufficiently so that VO was no longer within tolerance of E0. For example, suppose the line segment indicated was represented in the native system by an abstract data type having two

*This author acknowledges, with appreciation, partial funding for this work received under National Science Foundation Grant Number MII-9308346 and the hospitality of the University of Montréal during sabbatical leave. The views expressed herein are of this author, not of the Foundation.

†The research of this author was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada

geometric points, with the metric used for tolerance comparisons in the originating system being the Manhattan (also known as the taxi-cab) metric. For points $p_0 = (x_0, y_0)$ and $p_1 = (x_1, y_1)$, the Manhattan metric is given by the equation

$$d_M(p_0, p_1) = |x_0 - x_1| + |y_0 - y_1| \quad .$$

Suppose also that the neutral transfer representation called for lines to be represented in the format of an originating point and a directional unit vector with a separate scalar for its magnitude. In the conversion of the original line representation into the neutral format, some numerical error could occur, thereby perturbing E0.

Suppose V0 was transferred exactly (via faithful bit transfers) and the tolerance values for the neutral format were identical to that of the originating system, but that the metric used to test for a point being on an edge within the neutral format was the Euclidean metric, whose convex unit sphere contains that of the Manhattan metric. For points p_0 and p_1, the Euclidean metric is given by the equation

$$d_E(p_0, p_1) = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2} \quad .$$

The Euclidean metric, by virtue of its larger unit sphere, could still have the perturbed edge within tolerance of V0, as is indicated in more detail in Section 4 and Figure 2. Hence, the model could be topologically consistent in both its original form and in its neutral transferred format. However, since E0 has been perturbed, when the model is transferred back to the originating system, the more sensitive Manhattan metric could determine that the tolerances were violated, as shown in Figure 2.

This illustrative example appears to rely crucially upon the differences between the Manhattan and Euclidean metrics. However, a moment's reflection shows that even if both metrics were Euclidean, then merely slight differences in their coded implementations (for instance, use a squaring operation versus explicitly multiplying an argument by itself) could lead to similar inconsistencies. Furthermore, the first author has reviewed many instances of commercial CAD/CAM implementations where the two cited metrics were used interchangeably, with no provision for their differing characteristics. These differences are symptomatic of semantic inconsistency across module interfaces. In this example, the expression of each individual metric is mathematically valid, yet the ensuing communication does not preserve the intended meaning.

A simplistic approach to resolution might be just to perturb the co-ordinates of V0 slightly. Alternatively, for the simple geometry depicted, a slight linear extrapolation of E0 might be appropriate. While these approaches might resolve this particularly simple geometry, it is clear that neither approach is extensible to general situations. Note, that for more complex models, the point perturbation 'solution' could be fraught with all the difficulties of changing one item in a highly interrelated constraint network, and the linear extrapolation 'solution' would be totally inapplicable for curvilinear geometry.

4 More Complex Geometry

Figure 2 depicts the original curve as a dotted line, shown to be within the tolerance neighborhood about a point relative to the Manhattan metric, shown as a dashed diamond. The perturbed curve and the Euclidean tolerance neighborhood are shown with solid display.

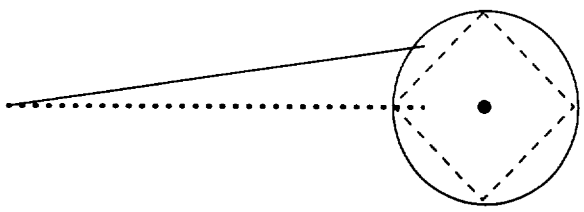
Figure 2: Two Tolerance Neighborhoods

The conversion of the line representation could be implemented in various ways, but the floating point operations invoked contribute to the error incurred. The representation conversion for the line segment is particularly simple. Yet the ideas expressed are indicative of more complex geometry. Consider a trimmed Non-Uniform Rational B-Spline (NURBS) curve represented in one system, with the intent to transfer it to another system. The originating data would consist of the trimming point and an abstract data type for the NURBS curve. The condition that the trimming point was 'on' the NURBS curve would typically be expressed by the distance between the point and the curve being less than some tolerance. Even if one were to suppose that the two communicating systems had identical tolerances and metrics, problems could occur. For instance, suppose the receiving system could only represent Bézier curves. Then, the exchange would require a conversion of the NURBS representation into its best Bézier approximant. The errors induced with that conversion, both due to the curve approximation and to floating point arithmetic, could be such that the distance from the trimming point to the curve would exceed the tolerance, implying that the trimming point was no longer 'on' the curve. The situation is further compounded when the communicating systems invoke differing tolerance values. Furthermore, this illustrative example considered only a dedicated transfer from one system to another, whereas, in practice usually additional conversions are necessary to and from a neutral intervening format.

5 Software Design Issues

The tolerance neighborhoods (Manhattan and Euclidean) depicted in Figure 2 model the situation where geometric co-ordinates are known with sufficient uncertainty that they could lie anywhere within the depicted tolerance neighborhoods. For ease of discussion, consider the location of the individual point to be fixed, as is consistent with our discussion of the transfer example. Then the neighborhood about the point can be considered the region in which there exists non-zero probability for non-void intersection with the line segment. To fully model the underlying semantics would require assigning a probability distribution over this neighborhood. If one were to assume simplistically that the probability distribution were uniform, the Euclidean neighborhood's greater area reflects a more pessimistic view of the uncertainty of the data than does the smaller Manhattan neighborhood.

Our critical issue is to emphasize that these 'hidden assumptions' must be explicated and their choices justified, as these seemingly innocent choices may have unintended implications. Resolving all such concerns is essential for establishing fully consistent, logically correct, formally verifiable semantics for geometric modeling. The cost to do so will be high. But it is our judgement that the failure to do so will be much higher, where the fourth author provides perspective on recent supporting evidence from Alcoa:

> About 18% of our best CAD engineers' and designers' time is spent on re-asserting the consistency of our models (these problems require too much insight for our medium skilled designers to quickly solve). At Alcoa, we expect that proper semantics could help us reclaim a substantial portion of time of our best people, but, more importantly, we will more effectively communicate with our suppliers and customers.

The fundamental difference for the level of consistency seen for the original, neutral and final models arises from two causes

- the differing use of 'metric', and
- the separation distance being near the tolerance limit.

Within *a single* system, adherence to contemporary software engineering principles can be effective as a disciplined means to enforce consistent semantics. For instance, the use of a metric should be standardized within one system. It should be extracted as a separate mathematical utility. All software engineers should then access this code, when needing such a utility. Even in the presence of such effective technical steps, it remains imperative to have ongoing internal education amongst software engineers to inform them of the importance of adhering to these standards while implementing deviating code only under carefully controlled, approved and documented circumstances. This does nothing to help legacy code which may fail to conform to these standards and the re-engineering of such code can be an expensive undertaking.

The demands for ensuring such semantic consistency become significantly more complex when one considers the issue outside the controlled bounds of writing software for only one system. Those more complex issues will be discussed in the next section.

6 Next Steps

The round-trip problem, as illustrated, was due to semantic differences across the exchange interface and to the separation distance being near the tolerance limit. The round-trip problem was chosen as one of the simplest transfer problems to illustrate the need for consistent semantics. Clearly, the problems become even more complex for exchange amongst multiple CAD systems.

For ease of exposition, our example deliberately focused upon the use of the same tolerance values. The problem becomes more complex when many different tolerance values are used, as is reasonable in a large system, where the tolerance values are locally adapted to specific algorithmic needs. It then seems appropriate to rationalize the interplay amongst all such values. An assessment of the success of past practice in isolating all such subtle dependencies has been pessimistically expressed [4]:

> System tuning by adjusting tolerances creates a subtle coupling of otherwise disjointed subprocesses within the system. Tolerance matching is then required. This is difficult or impossible to document. To make matters worse, tolerances are the most tempting control handle available for software maintenance. A programmer may fix a problem by changing a tolerance slightly without fully understanding the implications of the change.
>
> Among users, there is likely to be little guidance beyond a few rules of thumb.

However, we do not view this pessimism as reason to despair. Rather, we expect that the failures to date can be somewhat overcome by a more abstract mathematical approach to these problems. The combinatorial complexity certainly argues that simplistic methods will not suffice. The history of developing successful compilers for programming languages should prove instructive. Compiler theory must address issues of significant combinatorial complexity and it was necessary to develop correspondingly rich mathematical semantics for formally verifiable compilers. Such a development of new semantical theory for CAD tolerances is likely to be a long term project, but there are some helpful steps that can be taken in the interim.

Different conceptual approaches to algorithms (such as Manhattan versus Euclidean) could be unified. This might be particularly helpful, if even differences of implementation (squaring function versus repeated multiplication) could be avoided. This might suggest, (at least in the CAD arena) creation of some standardized geometrical libraries. This would be offering a solution by means of 'shared code', only a slightly more sophisticated approach than requiring homogeneous design environments for data exchange. Even shared code would not address the question as to slightly different results on different architectures. The use of such libraries would require extensive re-writing of existing commercial CAD systems and such effort may be economically prohibitive. While such retroactive fixes may not be feasible, that software architecture ideal should serve as a guide to future CAD software design and development.

A more immediate approach may provide a partial response to the issue raised in the abstract, above, relative to user notification. When computed values come 'close' to the tolerance limits of the communicating CAD system and/or of the transfer software, this situation should be flagged *before any transfer* is executed. It would require an interface to internal tolerance values used in both the communicating CAD systems and the transfer software. The result could be a warning, 'near tolerance–may require user intervention'. So warned, the user could then decide to take responsive action.

References

[1] Desaulniers, H., and Stewart, N. F., *Robustness of numerical methods in geometric computation when problem data is uncertain*, CAD, 25 (9), pp. 539 - 545.

[2] Hu, C.-Y., Patrikalakis, N. M., and Ye, X., *Robust interval solid modeling, Part I: representations*, CAD, 28, 1996, pp. 807–818.

[3] Hu, C.-Y., Patrikalakis, N. M., and Ye, X., *Robust interval solid modeling, Part II: boundary evaluation*, CAD, 28, 1996, pp. 819–830.

[4] NURBS White paper, Automotive Industry Action Group, Southfield, MI, 1991.

[5] PDES, Inc. Geometric Accuracy Team, Interim Report, 24 July, 1996.

[6] 'STEP on a page', http://www.nist.gov/sc5/soap/