

Surface Intersections and r-sets: Theory and Practice

T. J. Peters

Department of Computer Science and Engineering
Department of Mathematics
University of Connecticut
Storrs, CT 06269-3155

Abstract

Historically, r-sets are fundamental in the theory of solid modeling. The classical Boolean algebra of regular closed sets provides a rigorous formalism which has guided the implementation of software for executing combinatorial operations on solid models. In constructive solid geometry (CSG) modelers, these Boolean operations are elegantly stored as a tree, where this tree is distinct from any corresponding instantiated geometry. For B-rep models, the approximating geometric data is much more central. In both modeling paradigms, the approximated instance geometry typically deviates from being a regular closed set. Provable bounds on these approximations, particularly for surface intersection algorithms, are essential for solid models to be used reliably in a variety of engineering applications. Our work on two surface intersector tools, one to permit user-specification of model-space error bounds and the other to verify attainment of these bounds, is presented within the broader context of integration of CAGD and engineering simulations.

1 Introduction

The CSG method has an elegant binary tree structure which captures the Boolean operations. As such, it is a data representation for a regular closed set. However, as soon as explicit geometry is assigned to any of the operands, approximation is inevitable and the global topological property of being a regular closed set is usually lost. This regular closed set formalism has great appeal because of the persuasive argument [25] that regular closed sets are the conceptual idealization for the vast majority of objects of design interest. In particular, one early theoretic success of this conceptual view of solid modeling led to the avoidance of the usual binary set operations of intersection, union and difference, because these could result in ‘dangling edges’ and ‘incomplete solids’ [27]. So, the Boolean algebra has been a highly successful metaphor in guiding software development of combinatorial operations, but it is critically limited by two compelling factors

- The operands are rarely regular closed sets themselves.
- The central underlying surface intersection algorithms rarely deliver a boundary that is shared completely by both intersecting sets.

Guided by these factors, practical software tools are presented to facilitate the use of approximated solid models within engineering design and analysis.

Many geometric models are created by operations which join two or more spline surfaces along their intersection and splines are the preferred CAGD geometric representation for engineering applications. The idealized conceptual view of these joining operations is illustrated in Figure 1, where the intersection curve¹ is denoted as c .

For spline geometry, the compelling algorithmic issues of performance, stability and data volume result in this intersection curve being approximated [10]. Any computed intersection curve will typically be approximated twice; once within the parametric domain of one surface and then again within the parametric domain of the other. These approximations are denoted as c_1 and

¹We focus here on the generic case of an intersection curve, but isolated points and co-incident areas can also arise.

c_2 in Figure 1. Mapping from parameter space into the space² containing the geometric model then requires algorithmic evaluation of these approximated intersection curves, as indicated by F and G in Figure 1. It is virtually certain that those evaluations will not be exactly equal in model space.

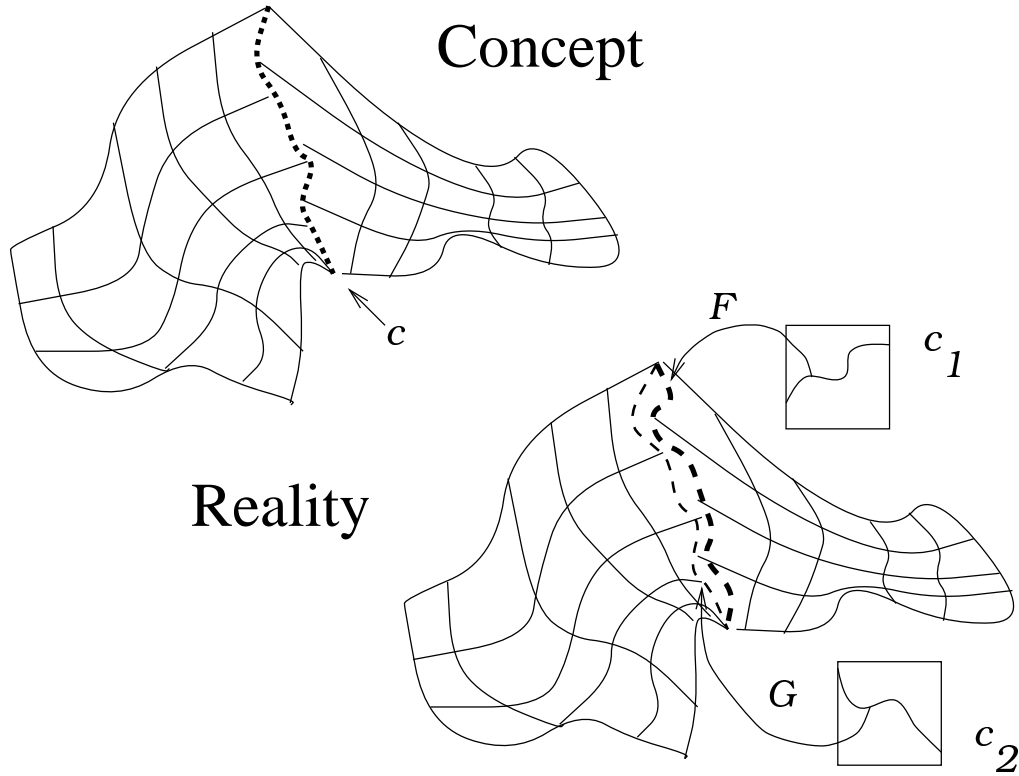


Figure 1: Joining Operations for Geometric Objects

The mismatch between concept and reality depicted in Figure 1 creates ambiguity, as the intersection representation is sometimes considered as a unique set, from the symbolic topological view, and at other times as two approximating sets, from the geometric view. To resolve this ambiguity, we propose a richer representation, which introduces new rigorous upper bounds for intersection approximations.

In practice, gaps and overlaps appear in many models, leading to non-manifold boundaries or incomplete boundaries, both invalidating the regular closed set assumption. The key shift of focus presented here is to narrow attention upon the intersection boundaries. This moves from the global consideration of a model being regular closed³ to local attention to measurements along the intersection boundary. This argues for more emphasis upon the notion of a topological complex. Then, the key terminology is that surfaces are *well-joined*, meaning that composite elements of the model are joined only along lower dimensional boundary elements and that any joining must be complete along any common boundary. The focus upon being well-joined allows formulation of rigorous error bounds for surface intersections and has led to the methods and tools presented here to improve integration of intersected surfaces and engineering analyses.

The rest of this paper is organized into the successive main sections of literature review, the role of spline surfaces and approximations, regular closed sets in engineering, description of the tools developed and conclusions.

²The range for this mapping is known as model space. Throughout this paper, the model space is \mathbb{R}^3 .

³To definitely demonstrate that any particular instantiated geometric model is regular closed is impractical. In theory, one would need to be able to verify the interior and its closure. Typically, one can only inspect a particular model to see if it has some obvious flaw to prevent it from being admissible as a regular closed set.

2 Literature Review

Research into geometric robustness problems dates back over 20 years, as has been documented in the literature [12, 22, 26]. Intersections are fundamental in computational geometry, geometric modeling and design, analysis and manufacturing applications [1, 13, 21]. Among intersection techniques, subdivision methods [17, 20, 24, 28] are generally stable and efficient at finding starting points for tracing simple intersections. The Projected Polyhedron (PP) algorithm is an iterative global root-finding method for n -dimensional systems of nonlinear polynomial equations. It belongs to the subdivision class of methods and was developed by Sherbrooke and Patrikalakis [28]. Maekawa and Patrikalakis [18] extended the PP algorithm to operate in rounded interval arithmetic in order to find all the roots of a polynomial system *robustly*, which is referred to as the Interval Projected Polyhedron (IPP) algorithm.

Grandine and Klein [11] present a spline surface intersection algorithm where implementation of topology resolution relies on an extension of the PP algorithm to piecewise algebraic systems. Code for the Grandine-Klein (G-K) intersector [4] is publicly available, leading to its use in industry and in DoD laboratories, including a prominent role in the Air Force 3D-OPT project [31]. This G-K approach advanced the state of the art in intersector technology over that presented in a 1992 survey on surface intersectors [5].

An industrial survey of CAGD vendors [8] showed little consistency, understanding or sophistication in measuring or implementing error bounds for geometric models, even indicating that critical questions persist about appropriate metrics for measuring these model errors. As models are communicated and distributed across several systems, such as undertaken via STEP, the question of consistent use of differing metrics remains unresolved [23]

The contemporary literature contains reports of many of the problems of CAGD models [6, 8, 12]. Attention is given to the *symptoms* of unwanted gaps along surfaces that are expected to be well-joined, inappropriate self-intersections and small geometric elements that appear as artifacts of the intersection process [6]. Furthermore, vendor software appears to give inadequate attention to the crucial role of error bounds in approximating geometric algorithms [8].

Some commercial software tools are available for ‘healing’ [7, 29, 30] geometric design models for specific purposes (by minimizing some gaps and overlaps), but these products are limited by their reliance upon domain-specific heuristics. Recent estimates by the National Institute of Standards and Technology (NIST) show that approximately \$1 billion is lost per year in the automotive sector due to semantic inconsistencies in CAGD models [2] and similarly large losses would be expected from other major industries. These estimates underscore the potential impact that integrated use of rigorous error bounds could have towards increasing national prosperity.

Fortune advocated exact arithmetic for polyhedral CAGD model intersections [9] and recent work by Manocha’s group has further extended the use of exact arithmetic in modeling [3, 14, 15]. Both of these researchers have emphasized the invocation of exact evaluation only when it is essential. That is, a run-time error estimation determines whether the evaluation of specific predicates can be done dependably in floating-point arithmetic. The algorithm will do an exact evaluation only if the floating-point evaluation of the predicate is uncertain. This provides a significant increase in performance over the exclusive use of exact arithmetic for all calculations. Unfortunately, even this restricted use of exact arithmetic remains costly when models include high-degree spline surfaces. Therefore, these techniques are not yet widely used in engineering applications.

These difficulties have also led to theoretical studies about the effective computability of regular-closed sets. Recent work [32] astutely defines the regular closed sets as an infinite limit of approximating sets. This is a sound theoretical model and allows for arbitrarily close approximation of any regular closed set. However, the practicalities of intersection algorithms demand delicate trade-offs between performance and approximation techniques, meaning, that, in practice, there is a severe truncation of this limiting process and any instance of a regular closed set is only one term in this sequence, often with no explicit error bound.

3 Splines and Approximations for Engineering Design

In any study of the accuracy of solid models, intersection techniques are pivotal and the underlying intersection sets are of critical importance. Yet, the mathematical and algorithmic subtleties of surface intersections are not fully understood. Within computational geometry, investigations of adaptive forms of exact arithmetic to implement geometric predicates have produced some very impressive results. In particular, contemporary research on exact methods holds promise with respect to special cases of low-degree geometry. These computational geometry results rest upon the foundational notion that the given numerical input data can be interpreted as exact or can easily be made exact. However, we note that the solid models which are dominant in complex industrial design domains are customarily given as boundary representations with spline faces. These cannot generally be considered as exact input, which sharply limits the role exact arithmetic can play in CAGD for engineering applications.

The commonly accepted formalisms for a well-formed solid model do not pay sufficient attention to the role of approximation — either its inherent existence in the input data or its introduction within algorithmic computations. Specifically, current definitions of well-formed solid models only treat the idealized view, as a regular-closed set. However, the engineering approximations used mean that the instantiated models will *not* be regular-closed sets because of ill-defined interiors and overlaps along boundaries. So, our attention has been directed towards providing software tools to specify and verify approximation errors along surface intersections.

4 Regular Closed Sets for Engineering Simulations

The critical CSG data representation is a binary tree. The three binary operations are join, meet and subtraction. An example appears in Figure 2, where the operation nodes are indicated as circles and the operand nodes are indicated as boxes. The operations follow the typical Boolean notation of using \wedge , \vee and $-$. The CSG perspective is that the geometry at the leaf nodes would be chosen from a few primitive geometric classes. Typical classes would consist of all possible spheres, cylinders, boxes, tori or wedges. Since, ideally, each of these objects is regular closed, the corresponding CSG trees also represent regular closed objects. In principle, this is true, and remains valid when the CSG tree is considered independent of any geometric instantiation. However, as soon as one attempts to instantiate the geometry, even with these primitive classes, problems arise.

For example, consider the sphere. A compact equational representation is typically given by

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2.$$

The immediate observation is that the indicated variables are permitted to be any real values and those might only be approximated in any computational arithmetic scheme. Furthermore, creating a specific geometric instantiation of a circle typically results in gaps or overlaps. Similar problems occur for geometric representations of the other primitives. Hence, even at the leaf nodes, the instantiated geometry does not satisfy the hypothesis of being a regular closed set, so it is unreasonable to expect that final resultant is regular closed.

While this is well recognized by most CAGD practitioners, this formal distinction is extremely important for engineering applications, where the object of primary interest is *not necessarily* just the pure conceptual model, but is more likely to be the actual instantiated geometric model. This approximate instantiated geometry is the input to a variety of engineering analysis and simulation programs, such as visualization, finite element analysis (FEA), computational fluid dynamics (CFD), computational electromagnetics (CEM) and computational optics (COP). Today, the interface between CAGD and visualization is relatively mature and successful. The interfaces between CAGD and engineering analyses and simulations, though, are less mature and continue to suffer vexing problems of geometric robustness. The critical difference between

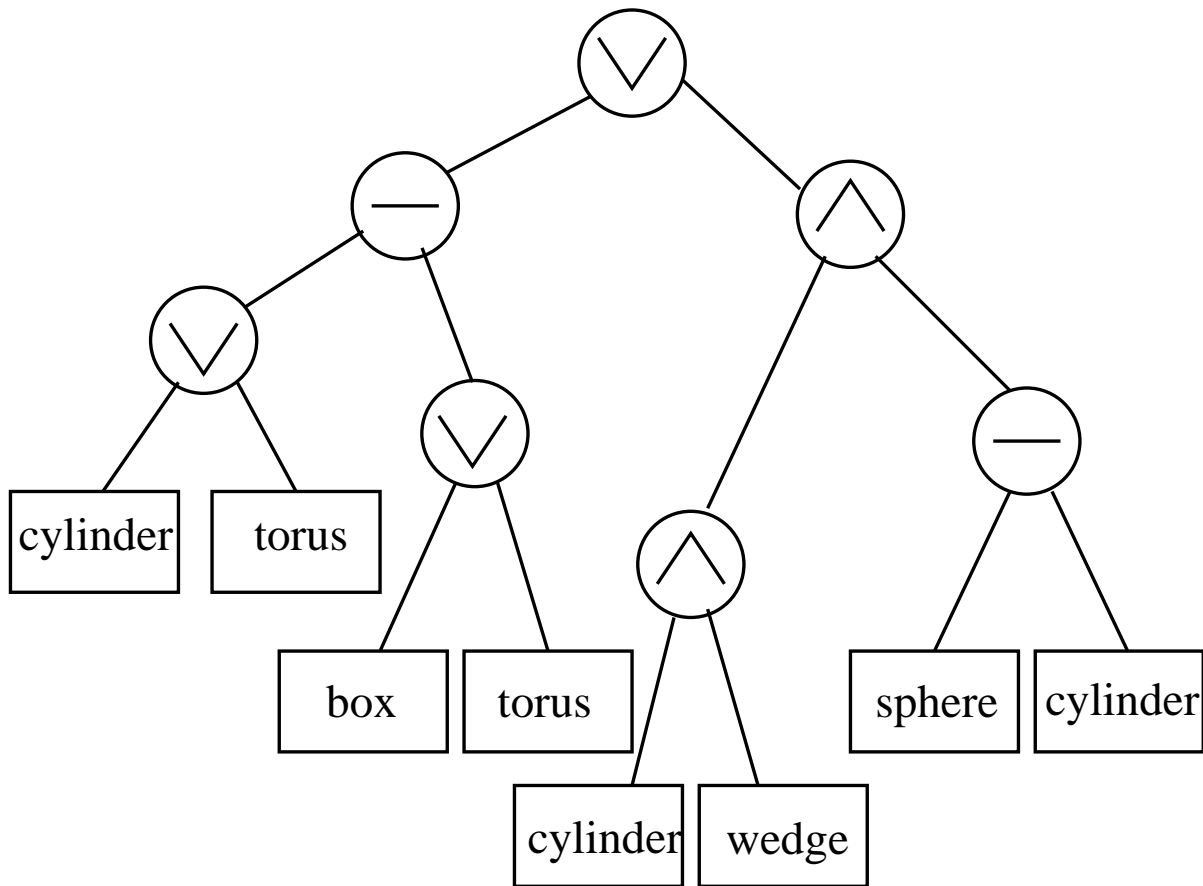


Figure 2: A CSG Tree

visualization and engineering applications is that the engineering applications critically depend upon numerical solutions of systems of partial differential equations defined over the approximated model geometry. The mathematical community has long been aware of the sensitivity of these numerical methods to input errors and this subject remains of current intense research efforts. So, it is at this interface, from the solid modeling system to the engineering simulations that the assumed ideals and the implemented approximations for geometric models become of intense interest.

If these engineering simulations are designed with the expectation that the input geometry sets are regular closed, with no approximation errors, then they will ignore error in this input. However, if it is clearly articulated that these models are approximations having specified error bounds, then these error bounds can be integrated with appropriately robust engineering simulations. The absence of careful error analysis from geometric input through to engineering output will greatly reduce the engineering value of any simulations performed. If the vision of proceeding directly from an electronic model of an airplane to its production, without any intervening physical testing, is to be achieved, then the specification and verification of the error bounds of these geometric approximations are of paramount importance.

5 Tools for Specification & Verification of Error Bounds

We present two complementary software tools for specification and verification of error bounds on surface intersections:

1. conversion of user-specified model-space error bounds into parametric-space error bounds for input to the G-K intersector, and
2. measuring software to test that the specified bounds have been attained on any specific model instance.

5.1 Tool to Convert Bounds

The rigorous definition of approximation bounds is rare for CAGD intersection algorithms [8]. A typical CAGD intersection algorithm merely accepts two surfaces and delivers an approximation to the intersection set, but presents no upper bound on the error. This would be analogous to using a Taylor’s series to approximate a function, while giving no estimate on the global truncation error.

A pragmatic approach has been followed in our development of an error-specification interface and error-measurement techniques. Within that pragmatic approach, this paper focuses upon B-rep models, with specific attention to the error bounds associated with the G-K algorithm. Although the software tools presented here were based directly upon the G-K approach, we expect generalizations to other intersectors, as will be discussed, below.

The current implementations of the G-K intersector rely upon elegant mathematical theory [11] to guarantee error bounds in *parameter space* for spline intersections. We have observed that this parametric bound is probably unintuitive for many end-users who would likely prefer⁴ an error bound in model space. Hence, the first tool is an interface designed to permit the user to express a bound in model space with conversion to the corresponding error bound in parameter space for input to the G-K intersector. Assuming C^2 continuity of each surface⁵, Mow, Peters and Stewart have recently formulated a relational between a model-space error bound λ [19] and its corresponding parameter-space value, ϵ . The proof is based upon Taylor’s Theorem in two dimensions and the full proof is available [19] elsewhere, so only a brief summary is presented here. In the rest of this subsection, the notation of Figure 1 is assumed⁶.

The G-K intersector accepts a user-specified value for the maximum acceptable error in parameter space for the output pre-images, c_1 and c_2 . The true intersection curve in model space is denoted as $c(t)$, $t \in [0, 1]$. Let $F : [0, 1]^2 \rightarrow \mathbb{R}^3$ define one surface, with the parametric variables being $(u, v) \in [0, 1]^2$. Let $\hat{c}_1 : [0, 1] \rightarrow [0, 1]^2$ be the unknown, exact, pre-image of c such that $F(\hat{c}_1(t)) = c(t)$, for all $t \in [0, 1]$, and assume that $c_1(t)$ is an approximation of \hat{c}_1 . With ϵ as the error bound for the G-K intersector, and assuming that \hat{c}_1 and c_1 are C^2 NURBS functions, the G-K intersector guarantees that $\|c_1(t) - \hat{c}_1(t)\| < \epsilon$ for all $t \in [0, 1]$.

If ϵ is the error bound in parameter space, then the model-space error bound for surface F is given by

$$\epsilon M_1(F) + \epsilon^2 M_2(F), \tag{1}$$

where $M_1(F)$ is an upper bound on the sums of the maximal absolute values of the first partial derivatives of F and $M_2(F)$ is an upper bound on the sums of the maximal absolute values of the second partial derivatives of F . The notation of $M_1(G)$ and $M_2(G)$ is defined similarly, for the second surface G . Application of the triangle inequality leads to an upper bound in model space on the error between the approximated boundary curves as

$$\epsilon(M_1(F) + M_1(G)) + \epsilon^2(M_2(F) + M_2(G)).$$

⁴The likelihood for this preference has been corroborated by a Boeing collaborator [16].

⁵While C^2 continuity may seem relatively restrictive from a theoretical view, such an assumption is typically not problematical in a wide variety of CAGD applications.

⁶We follow common abuse of notation in using c_1 both to represent a function and the geometric curve in \mathbb{R}^3 . Similar remarks apply to the notation used for other functions and their geometric representations.

Hence, for any user-chosen λ in model-space, it is only necessary to find an ϵ to ensure that the previously displayed sum is less than λ .

5.2 Improving Surface Intersector Interfaces

To illustrate the importance of error bounds for intersectors, we list three interfaces, below. The notation is that S_1 and S_2 are input surfaces and that a semi-colon separates the input from the output. Those interfaces have progressed, historically, as follows:

1. a naive interface, where only the surfaces are input, and no error bounds are given,
2. a direct interface to the G-K intersector, where the user must choose the value of ϵ , in parameter space, typically by means of some heuristic, and
3. an interface defined here that accepts the user-specified model-space error bound of λ , then converts that into the corresponding ϵ prior to calling the G-K intersector.

The naive interface is shown as

$$\textit{intersect}(S_1, S_2; S_1 \cap S_2).$$

The direct interface for the G-K intersector is given as

$$\textit{intersect}(S_1, S_2, \epsilon; S_1 \cap S_2).$$

Our newly defined interface has two steps, given as

$$\textit{convert}(S_1, S_2, \lambda; \epsilon(\lambda)),$$

which converts the user-specified model-space error bound λ into its corresponding parameter-space value $\epsilon(\lambda)$. This conversion is then followed by

$$\textit{intersect}(S_1, S_2, \epsilon(\lambda); S_1 \cap S_2).$$

Of these three presented options, only this last, two-step approach guarantees provable error bounds in model-space for the output intersection set.

The differing software flows are depicted in Figure 3, where the use of heuristics has previously been common to define the input to the G-K intersector, in contrast to the mathematically based approach now available.

The convert interface has been developed in prototype code that has been integrated with the DT_NURBS library⁷. It should be noted that the presented analysis assumes that approximations undertaken in the G-K algorithm are of primary importance and, specifically that these approximation errors dominate any errors that may be introduced at run time through inexact input or from software implemented in floating point arithmetic. This is a reasonable assumption for implementation of this user-interface, but it remains of interest to investigate the full run-time effect of inexact input and floating point errors. In particular, the measuring tool discussed in the next subsection can verify whether the specified bounds have been met and it will be of interest to see if any discrepancies discovered might be attributed to any approximations that have not yet been formally analyzed. These bounds proven have been the basis for our developing software tools for model space error bounds for realistic engineering models. These model-space error bounds are only directly applicable to the case of pairwise surface intersection. The broader question for CAGD models is the consideration of multiple surfaces intersecting simultaneously, where study of this broader question remains open.

⁷During that process, the next generation of geometric tools, known as GEML has appeared. Ongoing efforts are being devoted to further testing and integration with this new GEML software.

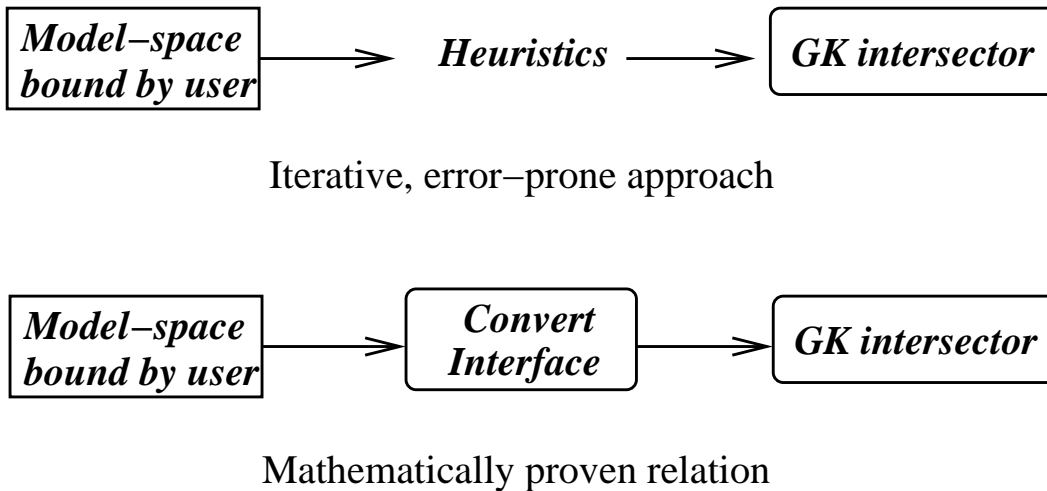


Figure 3: Heuristics Versus Formalism

5.3 Model Instance Error Verification Tool

To verify that a user-given bound has been attained we have also created software that accepts a geometric model as input and then computes model-space approximation errors as sampled along an intersection curve. This verification is now needed because the present model space bound conversion interface does not address any errors from inexact input or run-time errors arising from the use of floating point arithmetic. Although these additional sources of error surely deserve consideration, this de-coupling was important for efficient implementation of these software tools. Furthermore, it is reasonably justified in many cases, as the errors of approximation within surface intersection algorithms are often likely to be orders of magnitude greater than the accumulated floating point errors. However, it is important to verify this assumption and it is expected, that as intersector technology continues to improve and users demand greater accuracy, then both input errors and floating point errors will become more significant factors. The emerging software of all these error-specification and error-verification tools will then serve as a library to manage errors in CAGD models to ensure their robust use in engineering applications.

It is, of course, of interest to examine the relation between user-input model-space upper bounds and actual errors observed in specific model instances. Our studies have measured errors on specific models to be an order of magnitude less than the user-input upper bounds (Please see Table 1 for a representative summary, as explained in the next subsection.)

A software tool was created to measure the distance between the approximations to the boundary curves on the two surfaces in model space, where this distance is measured at corresponding parametric values. Using the triangle inequality and the given error bound of Expression (1), as applied to each surface, it is then easy to derive an upper bound for this distance between corresponding points on the approximated boundary curves in model space, as

$$\|F(\tilde{u}(\tau), \tilde{v}(\tau)) - G(\tilde{s}(\tau), \tilde{t}(\tau))\| \leq \gamma(F) + \gamma(G), \quad (2)$$

where $\gamma(F)$ denotes an upper bound for Expression (1) for surface F and $\gamma(G)$ is defined similarly for surface G .

5.4 A Representative Example

An illustrative example is presented. This example is based on intersecting a plane with the extrusion of a spiral curve having varying curvature, as summarized in Table 1. The generator curve for that surface is shown in Figure 4. This spiral surface was chosen to show the dependence

of the model space bounds upon the changing curvature of the model, even while the linearly extruded surface and the planar cut made obvious the nature of the true intersection set.

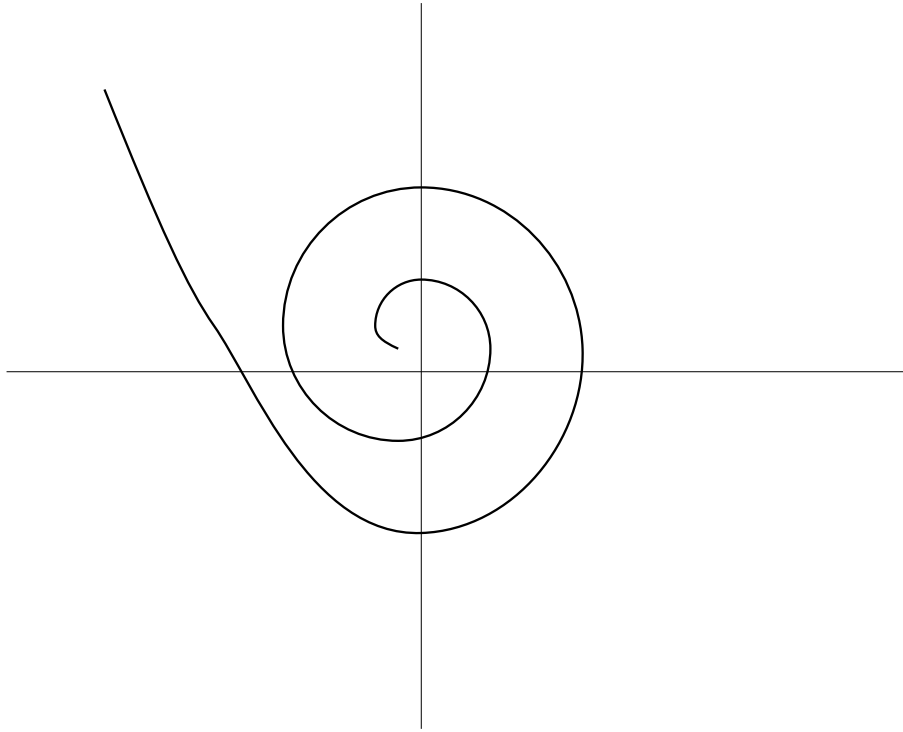


Figure 4: Generator Curve for Spiral Surface

Let F be the linearly extruded spiral surface, where the notation used below for the norms indicates use of the supremum norm. We conducted a numerical study of the partials of F to obtain the following upper bounds, while also concluding that other partials of F were negligible:

$$\left\| \frac{\partial F}{\partial u} \right\|_{\infty} \leq 300 \quad \left\| \frac{\partial F}{\partial v} \right\|_{\infty} \leq 3 \quad \text{and} \quad \left\| \frac{\partial^2 F}{\partial u^2} \right\|_{\infty} \leq 9000.$$

Let G be a planar surface that is just a square, 22 units on a side. Then, the norm of the two first-order partials for G must be bounded above by 22 and the other partials of G will be zero. These numerical upper bounds for the partials of F and G were then used to compute, for each value of ϵ , the right-hand side of the inequality in Expression (2), leading to the upper bounds indicated in the middle column of Table 1. The notation GK_m is the maximum of the actual distances measured using our software tool.

ϵ	$\gamma(F) + \gamma(G)$	GK_m
10^{-3}	$3.52 * 10^{-1}$	$1.64 * 10^{-2}$
10^{-4}	$3.48 * 10^{-2}$	$2.34 * 10^{-3}$
10^{-5}	$3.48 * 10^{-3}$	$3.31 * 10^{-4}$
10^{-6}	$3.48 * 10^{-4}$	$3.67 * 10^{-5}$

Table 1: Upper Bounds & Maximum Measured Values

The implementation of the *convert* interface to the G-K intersector, as described in Section 5.2 has been completed in our laboratory, based upon the mathematical relations expressed here.

5.5 Summary and Extensions of these Tools

This article discusses one algorithmic approach to error values, but it is important to note that the appropriateness of the chosen method and metrics are specific to the techniques implemented in the given intersector discussed. While the present model-space error bounds were formulated specifically for the G-K method, the primary criteria used were that each of the intersecting surfaces was C^2 . Hence, we expect our model-space error bounding techniques to generalize to other surface intersectors that satisfy these hypotheses.

In particular, it can reasonably be argued that the appropriate metric for measuring the difference between any two compact sets is the Hausdorff metric. However, implementing the Hausdorff metric is by no means trivial. Alternatively, the metric underlying our numerical verification tool is given by

$$\sup_{t \in [0,1]} \|c_1(t) - c_2(t)\|_\infty.$$

In general, there are trivial examples which show that such a parametric-based metric is likely to be problematical⁸, but the particular method invoked in the G-K intersector supports that this is a good choice of metric for the G-K output. Specifically, underlying mathematical methods [11] used in the G-K intersector generate the two pre-images based upon consideration of corresponding parametric values in $[0,1]$. As such, the metric in our verification tool is both a reasonable metric for the G-K intersector and also serves as a reasonable approximation to the Hausdorff metric. The critical observation is that there is an importance dependence of the choice of a reasonable metric upon the numerical methods chosen for the intersector.

6 Conclusions

We present a new method for relating user-chosen model-space error bounds to parametric-space error bounds upon surface intersection sets. This relation depends upon Taylor's Theorem in two dimensions, with the implication that pair-wise surface intersections for solid modeling can be delivered with guaranteed upper bounds for their errors (when errors of inexact input and floating point errors are of negligible importance). We can verify attainment of these bounds of any particular model instance with another implemented software tool. This allows an important change of emphasis from *repairing* inadmissible models to *a priori* specification of the inputs necessary to ensure that a created model will be acceptable for a particular engineering application.

References

- [1] Barnhill, R. E., Farin, G., Jordan, M. and Piper, B. R., *Surface/surface intersection*, *Computer Aided Geometric Design*, 4(1-2):3-16, July 1987.
- [2] Brunnermeier, S. B. and Martin, S. A. , NIST economic report, Interoperability Cost Analysis of the U.S. Automotive Supply Chain. Final Report, RTI Project Number 7007-03. Research Triangle Institute, Research Triangle Park, NC, 1999, <http://www.rti.org/publications>
- [3] Culber, T., Keyser, J. and Manocha, D., *Accurate computation for the medial axis of a polyhedron*, Proceedings of the ACM Symposium on Computational Geometry, Miami, FL 1999, 179 - 190.

⁸For instance, consider the interval $[0,1]$, which in one case is parametrized from 0 to 1 and in the other case parametrized from 1 to 0. Although the sets are equivalent and the Hausdorff metric would return 0, the metric based upon comparing points with corresponding parametric values would return 1.

- [4] DT_NURBS library, <http://ocean.dt.navy.mil/dtnurbs/>
- [5] Farin, G., *An SSI bibliography*, in Barnhill, R., ed., *Geometry Processing for Design and Manufacturing*, SIAM, 205 - 207, 1992.
- [6] Farouki, R., *Closing the gap between CAD model and downstream application*, SIAM News, (32) 5, June 1999.
- [7] FEGS Ltd., home page, CADfix product description, <http://www.fegs.co.uk>.
- [8] Ferguson, D. R., Lucian, M. L., and Seitleman, L. H., PDES Inc. Geometric Accuracy Team, Interim Report, PDES, Inc., SCRA, Charleston, SC, 1996
- [9] Fortune, S., *Polyhedral modelling with exact arithmetic*, Proceedings of the Third ACM Symposium on Solid Modeling and Applications. Salt Lake City, UT, 225-233, May 17-19, 1995.
- [10] Grandine, T. A., *Applications of contouring*, SIAM Review, 42(2), 297 - 316, 2000.
- [11] Grandine, T. A. and Klein, F. W. IV, *A new approach to the surface intersection problem*, Computer Aided Geometric Design, 14, 111-134, 1997.
- [12] Hoffmann, C. M., *How solid is solid modeling?*, ed M. Lin and D. Manocha, LNCS State-of-the-Art-Survey, Springer Verlag, 1-8, 1996.
- [13] Hoschek, J. and Lasser, D., *Fundamentals of Computer Aided Geometric Design*, A. K. Peters, Wellesley, MA, 1993, Translated by L. L. Schumaker.
- [14] Keyser, J., Krishnan, S., and Manocha, D., *Efficient and accurate B-rep generation of low degree sculptured solids using exact arithmetic: I - representations*, Computer Aided Geometric Design, 16(9), 841-859, Oct. 1999.
- [15] Keyser, J., Krishnan, S., and Manocha, D., *Efficient and accurate B-rep generation of low degree sculptured solids using exact arithmetic: II - computations*, Computer Aided Geometric Design, 16(9), 861-882, Oct. 1999.
- [16] Klein, F. W. IV, personal communication, July 25, 2001.
- [17] Lane, J. M. and Riesenfeld, R. F., *Bounds on a polynomial*, BIT: Nordisk Tidskrift for Informations-Behandling, 21(1), 112-117, 1981.
- [18] Maekawa, T. and Patrikalakis, N. M., *Computation of singularities and intersections of offsets of planar curves*, Computer Aided Geometric Design, 10(5):407-429, October 1993.
- [19] Mow, C., Peters, T. J. and Stewart, N. F., *Model-space bounds for the Grandine-Klein intersector*, pre-print, <http://www.eng2.uconn.edu/~tpeters/Peters.html>
- [20] Nishita, T., Sederberg, T. W. and Kakimoto, M., *Ray tracing trimmed rational surface patches*, ACM Computer Graphics, 24(4), 337-345, 1990.
- [21] Patrikalakis, N. M., *Surface-to-surface intersections*, IEEE Computer Graphics and Applications, 13(1):89-95, January 1993.
- [22] Peters, T. J., Rosen, D. W., and Dorney, S. M., *The diversity of topological applications within computer aided geometric design*, Annals of the New York Academy of Science, 728, 198-209, 1994.
- [23] Peters, T. J., Stewart, N. F., Ferguson, D. R., Fussell, P. S., *Algorithmic tolerances and semantics in data exchange*, Proceedings of the 13th ACM Symposium on Computational Geometry, June 4 - 6, 1997, Nice, France, 403 - 405.

- [24] Pratt, M. J. and Geisow, A. D., *Surface/surface intersection problems*, In J. A. Gregory, editor, *The Mathematics of Surfaces*, 117–142, Clarendon Press, 1986.
- [25] Requicha, A. A. G., old engineering article about representing machine parts via CSG
- [26] Requicha, A. A. G., *Representations for rigid solids: theory, method and systems*, Lecture Notes in Computer Science, ACM Computing Surveys, 12(4), Springer-Verlag, 437 - 464, 1980.
- [27] Requicha, A. A. G., Tilove, on r-sets
- [28] Sherbrooke, E. C. and Patrikalakis, N. M., *Computation of the solutions of nonlinear polynomial systems*, *Computer Aided Geometric Design*, 10(5), 379–405, 1993.
- [29] Theorem Solutions Lts., home page, CADhealer product description, <http://www.theorem.co.uk>.
- [30] TranscenData, home page, CAD/IQ product description, <http://www.cadiq.com>.
- [31] Wright-Patterson Air Force Base, AFRL/VAAC, Computational Sciences Branch, home page, <http://www.va.af.mil/VAA/vaac/projects.html>.
- [32] Ziegler, M. and Brattka, V., *Turing computability of (non-)linear optimization*, Proceedings of the Canadian Conference on Computational Geometry, 2001.