# Surface to Surface Intersections

N. M. Patrikalakis[1], T. Maekawa[2], K. H. Ko[3] and H. Mukundan[4]

[1]Massachusetts Institute of Technology, nmp@mit.edu
[2]Yokohama National University, maekawa@ynu.ac.jp
[3]Massachusetts Institute of Technology, khko@mit.edu
[4]Massachusetts Institute of Technology, harishm@mit.edu

## ABSTRACT

This paper presents an overview of surface intersection problems and focuses on the rational polynomial parametric/rational polynomial parametric surface intersection case including transversal and tangential intersections. Emphasis is placed on marching methods with a discussion of the problems with conventional tracing algorithms. An approach using a validated interval ordinary differential equation system solver is outlined and illustrated with examples, which offers significant advantages in robustness over conventional marching schemes.

**Keywords:** rounded interval arithmetic, boundary representation, parametric surfaces, singularity, tangency.

## 1. INTRODUCTION

Intersections are fundamental in CAD, CAM, computational geometry, geometric modeling and design, analysis and manufacturing applications [11,29]. Examples of intersection problems include: (1) Contouring of surfaces through intersection with a series of parallel planes or coaxial cylinders for visualization. (2) Numerical control machining (milling) involving intersection of generalized offset surfaces with a series of parallel planes, to create machining paths. (3) Representation of complex geometries in the *Boundary Representation (B-rep)* scheme using a process called *boundary evaluation*, in which the Boundary Representation is created by evaluating a *Constructive Solid Geometry (CSG)* model of the object. In this process, intersections of the surfaces of primitives must be found during Boolean operations (union, intersection, difference) between primitives.

When studying intersection problems, the type of curves and surfaces that we consider can be classified primarily into two types: (1) Rational polynomial parametric (RPP) and (2) Implicit algebraic (IA). Non-Uniform Rational B-Spline (NURBS) curves and surfaces can be subdivided into RPP curves and surfaces, and analyzed in a similar manner. A detailed treatment of intersection problems including general procedural curves and surfaces can be found in [31,30].

Among all types of intersections, the surface to surface (S/S) intersection is the most complicated problem. It can have various intersection components such as curve segments, points, loops and singular points. Therefore, any surface intersection algorithm should satisfy the robustness requirements imposed by solid and geometric modeling in (a) finding all the components of the intersection and (b) computing each component with high accuracy.

In this paper we only deal with S/S intersection problems of RPP/RPP type with emphasis on tracing the intersection curve robustly. General intersection problems are analyzed in [31,30].

This paper is structured as follows: Section 2 presents a classification of surface to surface intersection problems. In Section 3, a marching solution method is explained with formulation of transversal and tangential surface intersections. In Section 4, a robust marching method based on a validated ODE solver is introduced with examples illustrating the method. Section 5 concludes the paper.

## 2. CLASSIFICATION OF SURFACE TO SURFACE INTERSECTION PROBLEMS

An implicit algebraic surface is represented by a polynomial function defined as $f(\mathbf{r})=0$, where $\mathbf{r}$ is the position vector of a point on the surface. The rational polynomial parametric type includes Bézier, rational Bézier, B-spline and NURBS surface patches, which are represented with two parameters $u$ and $v$ as $\mathbf{r}=\mathbf{r}(u,v)$,

$0 \le u, v \le 1$. These surfaces are popular in CAD/CAM and geometric design, and NURBS surfaces are chosen as the standard format in industry. Depending on the surfaces involved in intersection, we have three distinct classes: IA/IA, RPP/IA and RPP/RPP. They are the most frequent surface to surface intersection problems.

### 2.1. IA/IA Surface Intersection

Implicit algebraic surface to implicit algebraic surface intersection is defined as follows:

$$f(\mathbf{r}) = 0 \cap g(\mathbf{r}) = 0, \tag{1}$$

where $f, g$ are polynomial functions. Here we have two equations in three unknowns $\mathbf{r}$.

A method for low order $f, g$ is to eliminate one variable (e.g. $z$ ) to find projection of intersection curves on the plane of other two variables (e.g. $x$, $y$ ), then trace the algebraic curve and use the inversion algorithm to find $z$. Intersections of low degree implicit algebraic surfaces are of special interest in the boundary evaluation of the Constructive Solid Geometry models. A more complete analysis of the special intersections of two quadric surfaces can be found in [25,37,42].

### 2.2. RPP/IA Surface Intersection

Rational polynomial parametric surface to implicit algebraic surface intersection is defined as follows:

$$\mathbf{r}(u,v) \cap f(\mathbf{r}) = 0, \qquad 0 \le u, v \le 1, \tag{2}$$

where $\mathbf{r}(u,v) = \left(\frac{X(u,v)}{W(u,v)}, \frac{Y(u,v)}{W(u,v)}, \frac{Z(u,v)}{W(u,v)}\right)^T$. This leads to four algebraic equations in five unknowns $\mathbf{r} = (x, y, z), u, v$. For the usual low degree surfaces $f(\mathbf{r})$ and low degree patches $\mathbf{r}(u,v)$, we can substitute $\mathbf{r}(u,v)$ into $f(\mathbf{r}) = 0$ to obtain an implicit algebraic curve in $u, v$, see [16,17,32,33] for detailed treatment.

### 2.3. RPP/RPP Surface Intersection

Rational polynomial parametric surface to rational polynomial parametric surface intersection is defined as follows:

$$\begin{gathered} \mathbf{r}_1(\sigma,t) \ \cap \ \mathbf{r}_2(u,v), \\ (0 \le \sigma, t \le 1, \ 0 \le u, v \le 1) \end{gathered} \tag{3}$$

where $\mathbf{r}_1(\sigma,t) = \left(\frac{X_1(\sigma,t)}{W_1(\sigma,t)}, \frac{Y_1(\sigma,t)}{W_1(\sigma,t)}, \frac{Z_1(\sigma,t)}{W_1(\sigma,t)}\right)^T$ and $\mathbf{r}_2(u,v) = \left(\frac{X_2(u,v)}{W_2(u,v)}, \frac{Y_2(u,v)}{W_2(u,v)}, \frac{Z_2(u,v)}{W_2(u,v)}\right)^T$. Formulation involves setting $\mathbf{r}_1(\sigma,t) = \mathbf{r}_2(u,v)$ which leads to three nonlinear polynomial equations in four unknowns $\sigma, t, u, v$. This is an underconstrained system. This system can in principle be solved by the *Interval Projected Polyhedron (IPP)* algorithm [38]. However, as the solutions are typically not isolated points but curves, such approach is inefficient when small tolerances are used. Another method involves implicitization of $\mathbf{r}_1(\sigma,t)$ to the form $f(\mathbf{r}) = 0$ and substitution of $\mathbf{r} = \mathbf{r}_2(u,v)$ into $f$ to reduce the problem to RPP/IA case for a low degree surface [17]. Heo *et al.* [10] developed an intersection algorithm for two ruled surfaces which performs more efficiently than those for general parametric surfaces.

There are three major techniques for solving RPP/RPP surface intersections: lattice methods, subdivision methods and marching methods. Detailed reviews can be found in [29,31,30]. In this paper, we focus on marching methods which are efficient in most cases and hence attractive if combined with other methods such as adaptive subdivision methods to locate starting points.

## 3. MARCHING METHODS

*Marching methods* involve generation of sequences of points of an intersection curve branch by stepping from a given point on the required curve in a direction prescribed by the local differential geometry [1,2,15,43]. Marching method formulates the surface intersection as an initial value problem (IVP) in the domain $0 \le \sigma, t, u, v \le 1$. However, such methods are by themselves *incomplete* in that they require *starting points* (initial conditions) for every branch of the solution.

### 3.1. Computation of Starting Points

In order to identify all connected components of an intersection curve, a set of characteristic points on the intersection curve can be defined. Such a set may include *border, turning and singular points* of the intersection and provides at least one point on any connected intersection segment and identifies all singularities. For RPP/RPP surface intersections a more convenient set of such points sufficient to discover all connected components of the intersection, includes *border and collinear normal points* between the two surfaces. Collinear normal points provide points inside all intersection loops and all singular points [12]. Border points are points of the intersection at which at least one of the parametric variables $\sigma, t, u, v$ takes a value equal to the border of the $\sigma$ - $t$ or $u$ - $v$ parametric domain. To compute border points, a piecewise rational polynomial curve to piecewise rational polynomial surface intersection capability is required, e.g.,

$\mathbf{r}_1(0,t)=\mathbf{r}_2(u,v)$, which can be robustly solved by the IPP algorithm [31,38].

Sederberg *et al.* [35] first recognized the importance of collinear normal points in detecting the existence of closed intersection loops in intersection problems of two distinct parametric surface patches. These are points on the two parametric surfaces at which the normal vectors are collinear. Collinear normal points are a subset of parallel normal points first used by Sinha *et al.* [39] in surface intersection loop detection methods.

To simplify the notation, we replace $\mathbf{r}_1(\sigma,t)$ by $\mathbf{p}(\sigma,t)$ and $\mathbf{r}_2(u,v)$ by $\mathbf{q}(u,v)$. Then the collinear normal points satisfy the following equations [12]:

$$\begin{aligned}&(\mathbf{p}_\sigma\times\mathbf{p}_t)\cdot\mathbf{q}_u=0,\quad(\mathbf{p}_\sigma\times\mathbf{p}_t)\cdot\mathbf{q}_v=0,\\&(\mathbf{p}-\mathbf{q})\cdot\mathbf{p}_\sigma=0,\quad(\mathbf{p}-\mathbf{q})\cdot\mathbf{p}_t=0.\end{aligned}\tag{4}$$

Equations (4) form a system of four nonlinear polynomial equations that can be solved using the IPP algorithm (also refer to [12] for more details on interval methods coupled with subdivision to solve the system (4)). Now we split the patches in (at least) one parametric direction at these collinear normal points. Consequently, starting points are only border points on the boundaries of all subdomains created. Grandine and Klein [8] follow a systematic approach for topology resolution of B-spline surface intersections. In this process, they determine the structure of the intersection curves including closed loops prior to numerical tracing (followed by a marching method based on numerical integration of a differential algebraic system of equations). Topology resolution in this context relies on an extension of the *Projected Polyhedron (PP)* algorithm [38] to the B-spline case. An alternate way to detect closed intersection loops is to use topological methods [15,3,19,22,23,24,41,40]. Also bounding pyramids [14,36] can be used effectively to assure the nonexistence of closed surface to surface intersection loops. These earlier methods need to be implemented in exact or rounded interval arithmetic (RIA) for robustness [31].

### 3.2. Formulation of the ODE System

The intersection curve can also be viewed as a curve on the two intersecting surfaces. A curve $\sigma=\sigma(s)$, $t=t(s)$ in the $\sigma t$-plane defines a curve $\mathbf{r}=\mathbf{c}(s)=\mathbf{p}(\sigma(s),t(s))$ on a parametric surface $\mathbf{p}(\sigma,t)$, as well as a curve $u=u(s)$, $v=v(s)$ in the $uv$-plane defines a curve $\mathbf{r}=\mathbf{c}(s)=\mathbf{q}(u(s),v(s))$ on a parametric surface $\mathbf{q}(u,v)$. We can derive the first derivative of the intersection curve, $\mathbf{c}'(s)$, from a curve on the parametric surface using the chain rule:

$$\mathbf{c}'(s)=\mathbf{p}_\sigma\sigma'+\mathbf{p}_t t',\quad\mathbf{c}'(s)=\mathbf{q}_u u'+\mathbf{q}_v v'.\tag{5}$$

After we find the unit tangent vector of the intersection curve, we can find $\sigma'$, $t'$, $u'$ and $v'$ by taking the inner product on both sides of the first equation of (5) with $\mathbf{p}_\sigma$ and $\mathbf{p}_t$ and the second equation with $\mathbf{q}_u$ and $\mathbf{q}_v$, which leads to two linear systems [12]. The solutions are obtained as

$$\begin{aligned}&\sigma'=\frac{det(\mathbf{c}',\mathbf{p}_t,\mathbf{P}(\sigma,t))}{\mathbf{P}(\sigma,t)\cdot\mathbf{P}(\sigma,t)},\quad t'=\frac{det(\mathbf{p}_\sigma,\mathbf{c}',\mathbf{P}(\sigma,t))}{\mathbf{P}(\sigma,t)\cdot\mathbf{P}(\sigma,t)},\\&u'=\frac{det(\mathbf{c}',\mathbf{q}_v,\mathbf{Q}(u,v))}{\mathbf{Q}(u,v)\cdot\mathbf{Q}(u,v)},\quad v'=\frac{det(\mathbf{q}_u,\mathbf{c}',\mathbf{Q}(u,v))}{\mathbf{Q}(u,v)\cdot\mathbf{Q}(u,v)},\end{aligned}\tag{6}$$

where $det$ denotes the determinant (see also [8]) and

$$\mathbf{P}(\sigma,t)=\mathbf{p}_\sigma\times\mathbf{p}_t,\quad\mathbf{Q}(u,v)=\mathbf{q}_u\times\mathbf{q}_v,\tag{7}$$

are the normal vectors of $\mathbf{p}$ and $\mathbf{q}$, respectively.

#### 3.2.1. Transversal Intersection

When two surfaces intersect transversally, the tangential direction $\mathbf{c}'(s)$ of the intersection curve $\mathbf{c}(s)$ is perpendicular to the normal vectors of both surfaces. So, the marching direction can be obtained as follows:

$$\mathbf{c}'(s)=\frac{\mathbf{P}(\sigma,t)\times\mathbf{Q}(u,v)}{|\mathbf{P}(\sigma,t)\times\mathbf{Q}(u,v)|},\tag{8}$$

where the normalization forces $\mathbf{c}(s)$ to be arc length parametrized.

#### 3.2.2. Tangential Intersection

When the two surfaces intersect tangentially, we cannot use Equation (8) since the denominator vanishes. In such cases we must find the marching direction in an alternate way [44].

The unit tangent vector $\mathbf{c}'(s)$ must lie on the common tangent plane of $\mathbf{p}(\sigma,t)$ and $\mathbf{q}(u,v)$. It can be defined using the linear combination of the partial derivatives $\mathbf{p}_\sigma$, $\mathbf{p}_t$, $\mathbf{q}_u$ and $\mathbf{q}_v$ of each of the surfaces as follows:

$$\mathbf{c}'(s)=\mathbf{p}_\sigma\sigma'+\mathbf{p}_t t'=\mathbf{q}_u u'+\mathbf{q}_v v'.\tag{9}$$

Since the normal vectors of the surfaces at a point on the intersection are the same, both surfaces have the same normal curvature at that point in the direction $\mathbf{c}'(s)$ of the intersection curve. This implies that the *second fundamental forms* of both surfaces are equal, which can be expressed as follows:

$$\begin{aligned}&L^p(\sigma')^2+2M^p\sigma't'+N^p(t')^2\\&=L^q(u')^2+2M^q u'v'+N^q(v')^2,\end{aligned}\tag{10}$$

where $L^p, M^p, N^p$ and $L^q, M^q, N^q$ are the second fundamental form coefficients of both surfaces.
This is a quadratic equation in $(\sigma', t', u', v')$. By taking the cross product of both sides of equation (9) with $\mathbf{q}_u$ and $\mathbf{q}_v$, and projecting the resulting equations onto the common surface normal vector $\mathbf{N}$ at a point on the intersection, $u'$ and $v'$ can be represented as the following linear combination of $\sigma'$ and $t'$:

$$u' = a_{11}\sigma' + a_{12}t', \tag{11}$$

$$v' = a_{21}\sigma' + a_{22}t', \tag{12}$$

where the coefficients $a_{11}, a_{12}, a_{21}$ and $a_{22}$ are

$$a_{11} = \frac{(\mathbf{p}_\sigma \times \mathbf{q}_v)\cdot\mathbf{N}}{(\mathbf{q}_u \times \mathbf{q}_v)\cdot\mathbf{N}} = \frac{det(\mathbf{p}_\sigma, \mathbf{q}_v, \mathbf{N})}{\sqrt{E^q G^q - (F^q)^2}},$$

$$a_{12} = \frac{(\mathbf{p}_t \times \mathbf{q}_v)\cdot\mathbf{N}}{(\mathbf{q}_u \times \mathbf{q}_v)\cdot\mathbf{N}} = \frac{det(\mathbf{p}_t, \mathbf{q}_v, \mathbf{N})}{\sqrt{E^q G^q - (F^q)^2}},$$

$$a_{21} = \frac{(\mathbf{q}_u \times \mathbf{p}_\sigma)\cdot\mathbf{N}}{(\mathbf{q}_u \times \mathbf{q}_v)\cdot\mathbf{N}} = \frac{det(\mathbf{q}_u, \mathbf{p}_\sigma, \mathbf{N})}{\sqrt{E^q G^q - (F^q)^2}},$$

$$a_{22} = \frac{(\mathbf{q}_u \times \mathbf{p}_t)\cdot\mathbf{N}}{(\mathbf{q}_u \times \mathbf{q}_v)\cdot\mathbf{N}} = \frac{det(\mathbf{q}_u, \mathbf{p}_t, \mathbf{N})}{\sqrt{E^q G^q - (F^q)^2}}.$$

Here, $E^q, G^q, F^q$ are the *first fundamental form coefficients* of the surface $\mathbf{q}$.
Substituting (11) and (12) into (10), then we obtain a quadratic equation of the form,

$$b_{11}(\sigma')^2 + 2b_{12}(\sigma')(t') + b_{22}(t')^2 = 0, \tag{13}$$

where,

$$b_{11} = a_{11}^2 L^q + 2a_{11}a_{21}M^q + a_{21}^2 N^q - L^p,$$

$$b_{12} = a_{11}a_{12}L^q + (a_{11}a_{22} + a_{21}a_{12})M^q + a_{21}a_{22}N^q - M^p,$$

$$b_{22} = a_{12}^2 L^q + 2a_{12}a_{22}M^q + a_{22}^2 N^q - N^p.$$

There are four distinct cases to the solution of (13) depending upon the discriminant $(d = b_{12}^2 - b_{11}b_{22})$.

- ($d < 0$): The surfaces have an isolated tangential contact point.
- ($d > 0$): We have the phenomenon of branching, i.e. $\mathbf{c}'(s)$ is not uniquely defined.
- ($d = 0$ and $b_{11}, b_{12}, b_{22} = 0$): The intersection of surfaces $\mathbf{p}$ and $\mathbf{q}$ cannot be evaluated by this method or they have a contact of at least second order (i.e., curvature continuous).
- ($d = 0$ and $b_{11}^2 + b_{12}^2 + b_{22}^2 \neq 0$): The marching direction vector is defined. Thus, $\mathbf{p}$ and $\mathbf{q}$ are said to intersect tangentially at the neighborhood.

If $b_{11} \neq 0$, $\frac{\sigma'}{t'} = \nu = -\frac{b_{12}}{b_{11}}$, and the marching direction is given by,

$$\mathbf{c}'(s) = \frac{\nu\mathbf{p}_\sigma + \mathbf{p}_t}{|\nu\mathbf{p}_\sigma + \mathbf{p}_t|}. \tag{14}$$

If $b_{11} = 0$ and $b_{22} \neq 0$, $\frac{t'}{\sigma'} = \mu = -\frac{b_{12}}{b_{22}}$, then the marching direction is given by,

$$\mathbf{c}'(s) = \frac{\mathbf{p}_\sigma + \mu\mathbf{p}_t}{|\mathbf{p}_\sigma + \mu\mathbf{p}_t|}. \tag{15}$$

*3.2.3. Conventional Solution Methods and Issues*

The points of the intersection curves are computed successively by integrating the initial value problem for a system of nonlinear ordinary differential equations (6) using standard numerical techniques such as the Runge-Kutta method, Taylor series method or the Adams-Bashforth method [7]. But when two intersection curves are close to each other, then step size selection becomes complex and incorrect step size may lead to a critical problem, *straying* or *looping* [6], which is illustrated in Figure 1 [27].

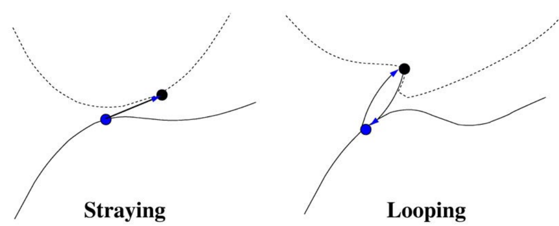


Fig. 1. Conceptual figures for straying and looping

Figure 2 shows the looping phenomenon when the Runge-Kutta method is used to solve an initial value problem corresponding to Figure 6 where we have two intersecting surfaces. The intersection contains a singular point at $[\sigma, t, u, v]^T = [0.5, 0.5, 0.5, 0.5]^T$. With an initial condition $[\frac{1}{3}, 0, 0, \frac{3}{4}]^T$, the system of equations (6) is provided as input to a Matlab ODE solver, *ode45*, which is based on the Runge-Kutta method and adopts an adaptive step size control scheme. As Figure 2 shows, the Matlab ODE solver breaks down near the singular point.

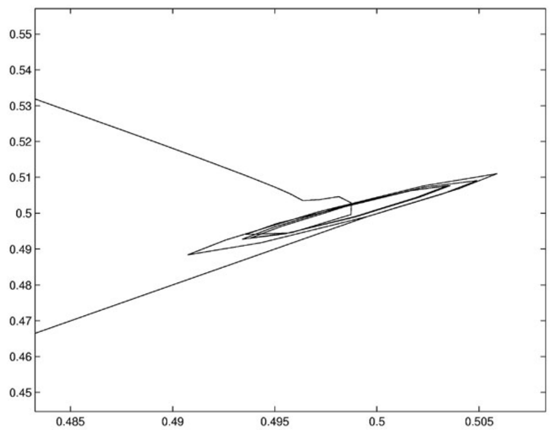


Fig. 2. An example of looping by Runge-Kutta (ode45) method

## 4. ROBUST MARCHING METHOD

In order to avoid the problems inherent to the conventional numerical methods, we have to rely on a different concept to solve the initial value problem of an ODE system. To ensure robustness in finding roots of the ODE system, researchers have focused on validated schemes using interval arithmetic [26]. The validated ODE solution scheme traces a solution after verifying the existence and uniqueness of the solution at every step. This idea is formulated and implemented in various forms by Moore [26], Krückeberg [9], Eijgenraam [5], Löhner [20] and Nedialkov [28]. After validation, a bound is computed which encloses errors in initial values, truncation errors and round off errors [4].

### 4.1. Concept of Validated ODE Solver

A validated ODE solving scheme consists of two phases [28]: *Algorithm I* and *Algorithm II*. Algorithm I finds an *a priori* enclosure and a step size (based on validation) such that the existence and uniqueness within the *a priori* enclosure for the step size is verified. This validation is achieved by applying *Picard-Lindelöf* operator and *Banach's fixed point theorem* [28]. A few methods for validation have been proposed such as the *constant enclosure method* [5], the *polynomial enclosure method* [21] or the *Taylor series method* [4].

Algorithm II deals with the propagation of the solution, reduction of *wrapping* and further prediction of a new step size for the next step. Wrapping is defined as *undesirable overestimation of a solution set of an iteration or recurrence which occurs if this solution set is replaced by a superset of some simpler structure and this super set is then used to compute the enclosures for the next step which may eventually lead to an exponential growth of overestimation* [18]. The control of the wrapping effect is a critical issue in this phase and several methods such as a local coordinate transformation method [26], a parallelepiped method and a *QR factorization* method [20] have been proposed.

### 4.2. Application to Surface Intersection Problem

Since the marching scheme requires to solve a system of equations (6), we can use a validated ODE solver by formulating the equations presented in Section 3.2 in interval arithmetic with interval initial conditions [27]. The solver produces an *a priori* enclosure at a step and a corresponding step size, which form a region, called *a priori box*, where the existence and uniqueness of the solution is verified. The union of such *a priori* boxes constructs a continuous bound enclosing the exact solution curve in the parametric space, which can be mapped into the model space to provide a gap-free bound in 3D model space [27]. The intersection of bounds in the model space mapped from each surface may further reduce the bound containing the intersection curve [27]. The result of this process can serve as one of the basic building blocks of interval solids introduced in [13,34].

One prominent advantage of the application of the validated ODE solver to the surface to surface intersection problem is the capability of coping with singular points, straying and looping [27]. When the solver approaches singular points or points where two intersection curves get close to each other, a validation condition in Algorithm I of the validated ODE solver gets violated so that the *a priori* enclosure as well as the step size is adjusted. This adjustment is repeated iteratively until the validation condition is satisfied, which leads the solver to trace the correct solution [27]. This iteration will resolve straying or looping in tracing an intersection curve. If this iteration continues to make the step size less than a certain minimum value, then the iteration stops and the solver reports a singular point, see [27].

### 4.3. Examples

Figure 3 shows a torus and a cylinder intersecting. We trace one of the four loops of the curves of intersection. We apply the validated ODE solver and map the error bounds in parametric space to obtain strict bounds in the 3D model space. The maximum relative model space error = 0.0187.

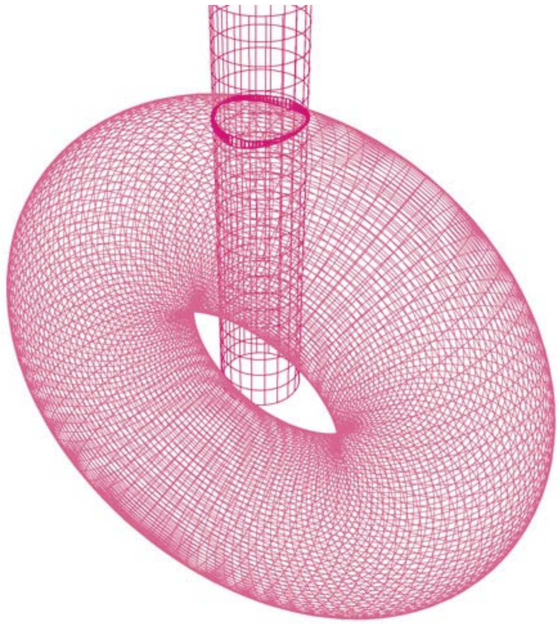

Fig. 3. Transversal intersection of a torus and a cylinder.

Transversal intersection of two tensor product Bézier patches is depicted in Figure 4. Like the previous example we solve the IVP for ODEs using a validated ODE solver and subsequently obtain the model space error bounds. The Figure 4 shows how the 3D model space error bound converges to the true intersection for small values of the error.

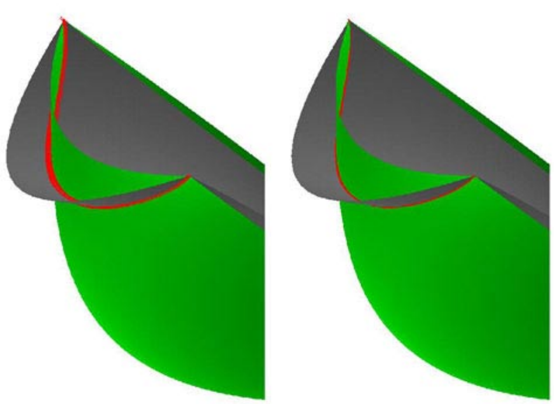

Fig. 4. Transversal intersection of tensor product Bézier surface patches, and the convergence of error bounds.

Figure 5 represents the intersection of two tensor product Bézier patches. The patches are positioned in such a way that they are tangential to each other and their curve of intersection is a 3D curve. The surface control points are represented as degenerate intervals and are provided as input to a validated ODE solver. The enclosure containing the curve of intersection is mapped from the parameter space to the 3D model space and we obtain rigorous bounds in the 3D model space, which guarantee to contain the true curve of intersection with a maximum relative model space error of 0.002.

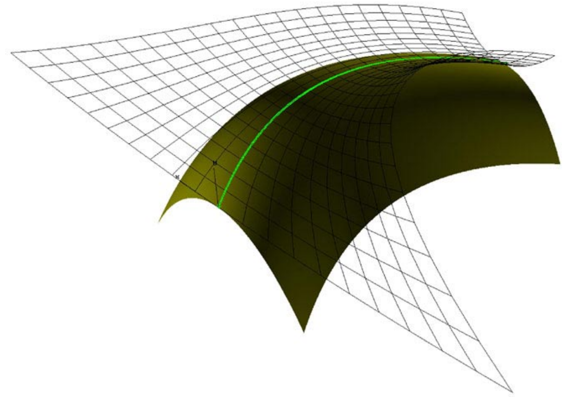

Fig. 5. Tangential intersection of tensor product Bézier surface patches.

Figure 6 shows an example constructed in such a way that there is a singular point in the surface intersection curve segment. Tracing the surface intersection in this example would involve separately tracing the four intersection curve segments, given appropriate starting points.

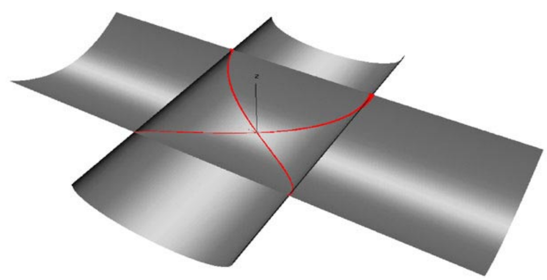

Fig. 6. An example of surface intersection with a singular point involving tracing four separate intersection curve segments.

Application of a conventional ODE system solver, such as the Runge-Kutta or Adams-Bashforth methods would involve the following pathologies:

1. Specifying a starting point which is approximate would mean that the curve traced would not have the singularity or bifurcation. The B-rep model generated would lose topological information and the result may further cause failure in CAD model processing.

2. Straying or looping near the singular region, which are essentially related to the uncertainty of the solver in taking a specific step.

Ideally given a starting point $[\sigma_0, t_0, u_0, v_0]^T = [\frac{1}{3}, 0, 0, \frac{3}{4}]^T$, we expect a solver to notify us as it approaches a region close to the singularity. The use of the recommended solvers in *Matlab* such as *ode45* (an implementation of Runge-Kutta method) and *ode113* (implementation of Adam's method) would result in behavior as erratic as shown in Figures 2 and 7. We show in Figure 8 the behavior of a validated ODE solver which does not march across the singularity. Thus the intersection is traced by separately tracing all the four intersection curve segments.

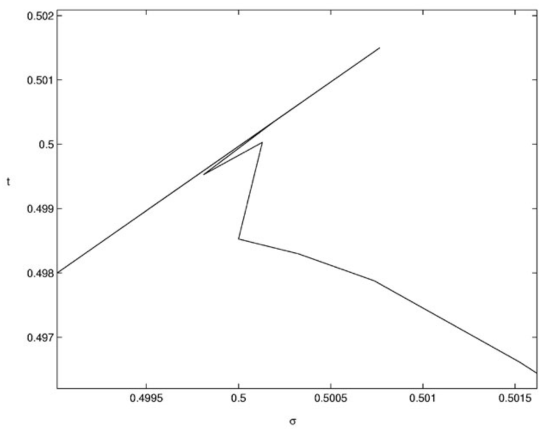


Fig. 7. Integration using *ode113* in Matlab. Straying and looping is seen at the region close to the singularity in the $\sigma, t$ parameter space.

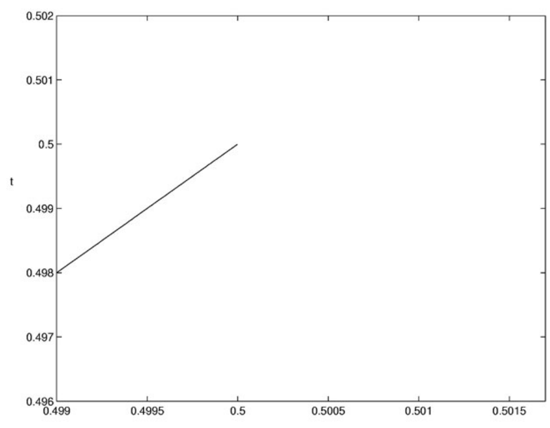


Fig. 8. Integration using a validated ODE solver, not crossing the singular region in $\sigma, t$ parameter space.

Now consider the case when one of the surfaces in Figure 6 is perturbed by a small amount in z-direction such that the intersection curves have different topological configuration. The intersection is now just two separate curve segments, even though they lie very close to each other near the previously singular region.

Conventional methods shows poor behavior near the region where two intersection curves are very close to each other. This is shown by Figure 9 obtained using the Adams-Bashforth method. Note the inconsistency in topology of the intersection curves obtained from conventional methods. The validated ODE solver uses an adaptive step size strategy, easily resolves this case, and behaves well locally close to the near-singular region as shown by Figure 10.

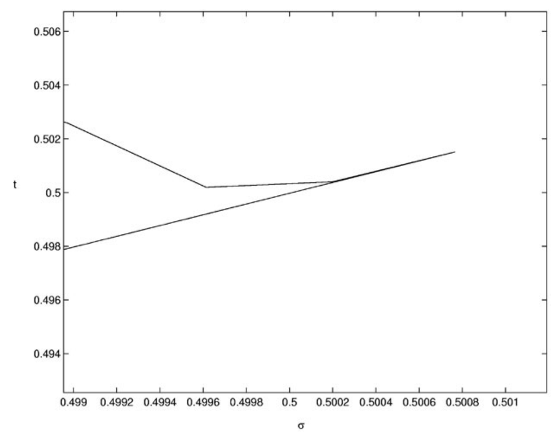


Fig. 9. Result from *ode113* in Matlab.

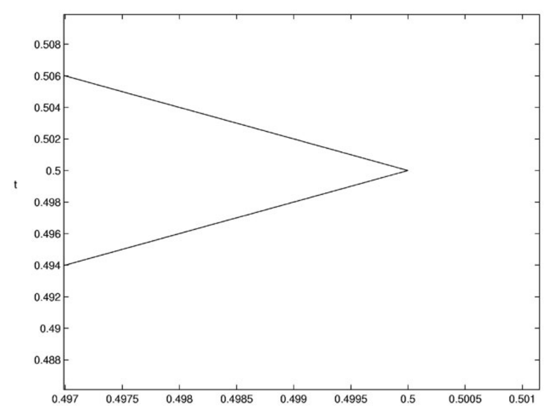


Fig. 10. Result from a validated ODE solver

## 5. CONCLUSIONS

Investigating the effects of floating point arithmetic on the implementation of intersection algorithms has been an important area for basic research during the last decade [31]. Methods to enhance the precision of intersection computation, to monitor numerical error contamination and alternate means of performing arithmetic, not relying on imprecise floating point computation alone, have been explored in some detail. Researchers in surface intersection problems during the last decade have already obtained a good understanding of robustness problems when employing floating point arithmetic and of methods to mitigate these problems based on rounded interval arithmetic [12].

As a result of the deficiencies of the conventional numerical methods, recent research tends to focus on exact methods involving rational arithmetic. Much research remains to be done in bringing such methods to the CAD practice, generalizing the arithmetic to go beyond rational and algebraic numbers (eg. involving transcendental numbers of trigonometric form), and to explore more efficient alternatives that are generally applicable in low and high degree problems alike. A different direction of research involves the use of non-conventional interval methods like a validated ODE solver [27], which considers errors arising in the computation as well as initial conditions. It provides a guaranteed bound which encloses the exact solution, and fits well with the concept of robust interval solid modeling [13,34].

Extension of current intersection methods applied on rational B-spline surfaces to more general and complex surfaces requires further study. Such surfaces include offset, generalized cylinder, blending and medial surfaces, and surfaces arising from the solution of partial differential equations or via recursion techniques.

## ACKNOWLEDGEMENTS

This work was funded in part by the NSF (grants No. DMS-0138098 and CCR-0231511).

## 6. REFERENCES


[1] C. L. Bajaj, C. M. Hoffmann, J. E. Hopcroft, and R. E. Lynch. Tracing surface intersections. *Computer Aided Geometric Design*, 5(4):285–307, November 1988.

[2] R. E. Barnhill and S. N. Kersey. A marching method for parametric surface / surface intersection. *Computer Aided Geometric Design*, 7(1-4):257–280, June 1990.

[3] K.-P. Cheng. Using plane vector fields to obtain all the intersection curves of two general surfaces. In W. Strasser and H. Seidel, editors, *Theory and Practice of Geometric Modeling*, pages 187–204. Springer-Verlag, New York, 1989.

[4] G. F. Corliss and R. Rihm. Validating an a priori enclosure using high-order Taylor series. In G. Alefeld, A. Frommer, and B. Lang, editors, *Scientific Computing and Validated Numerics: Proceedings of the International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics - SCAN ’95*, pages 228–238. Akademie Verlag, Berlin, 1996.

[5] P. Eijgenraam. *The Solution of Initial Value Problems Using Interval Arithmetic.* Mathematical Centre Tracts No. 144., Stichting Mathematisch Centrum, Amsterdam, 1981.

[6] A. Geisow. *Surface Interrogations*. PhD thesis, School of Computing Studies and Accountancy, University of East Anglia, Norwich NR47TJ, U. K., July 1983.

[7] C. F. Gerald and P. O. Wheatley. *Applied Numerical Analysis*. Addison-Wesley, Reading, MA, 4th edition, 1990.

[8] T. A. Grandine and F. W. Klein. A new approach to the surface intersection problem. *Computer Aided Geometric Design*, 14(2):111–134, 1997.

[9] E. Hansen, editor. *Topics in Interval Analysis*. Oxford University Press, 1969.

[10] H.-S. Heo, M.-S. Kim, and G. Elber. The intersection of two ruled surfaces. *Computer-Aided Design*, 31(1):33–50, January 1999.

[11] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A. K. Peters, Wellesley, MA, 1993. Translated by L. L. Schumaker.

[12] C. Y. Hu, T. Maekawa, N. M. Patrikalakis, and X. Ye. Robust interval algorithm for surface intersections. *Computer-Aided Design*, 29(9):617–627, September 1997.

[13] C. Y. Hu, N. M. Patrikalakis, and X. Ye. Robust interval solid modeling: Part I, Representations. *Computer-Aided Design*, 28(10):807–817, October 1996.

[14] G. A. Kriezis and N. M. Patrikalakis. Rational polynomial surface intersections. In G. A. Gabriele, editor, *Proceedings of the 17th ASME Design Automation Conference, Vol. II*, pages 43–53, Miami, September 1991. ASME, New York, 1991.

[15] G. A. Kriezis, N. M. Patrikalakis, and F.-E. Wolter. Topological and differential-equation methods for surface intersections. *Computer-Aided Design*, 24(1):41–55, January 1992.

[16] G. A. Kriezis, P. V. Prakash, and N. M. Patrikalakis. Method for intersecting algebraic surfaces with rational polynomial patches. *Computer-Aided Design*, 22(10):645–654, December 1990.

[17] S. Krishnan and D. Manocha. Efficient surface intersection algorithm based on lower-dimensional

formulation. *ACM Transactions on Graphics*, 16(1):74–106, January 1997.

[18] U. Kulisch, R. J. Lohner, and A. Facius. *Prespectives on Enclosure Methods*. Springer, New York, 2001.

[19] N. G. Lloyd. *Degree Theory*. Cambridge University Press, Cambridge, 1978.

[20] R. J. Lohner. Computation of guaranteed enclosures for the solutions of ordinary initial and boundary value problems. In J.R. Cash and I. Gladwell, editors, *Computational Ordinary Differential Equations*, pages 425–435. Clarendon Press, Oxford, 1992.

[21] R. J. Lohner. Step size and order control in the verified solution of IVP with ODEs. In *SciCADE'95 International conference on scientific computation and differential equations*, Stanford, C.A., March 1995.

[22] Y. Ma and Y.-S. Lee. Detection of loops and singularities of surface intersections. *Computer-Aided Design*, 30(14):1059–1067, December 1998.

[23] Y. Ma and R. C. Luo. Topological method for loop detection of surface intersection problems. *Computer-Aided Design*, 27(11):811–820, November 1995.

[24] R. P. Markot and R. L. Magedson. Solutions of tangential surface and curve intersections. *Computer-Aided Design*, 21(7):421–429, September 1989.

[25] J. R. Miller and R. N. Goldman. Geometric algorithms for detecting and calculating all conic sections in the intersection of any two natural quadratic surfaces. *Graphical Models and Image Processing*, 57(1):55–66, January 1995.

[26] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[27] H. Mukundan, K. H. Ko, T. Maekawa, T. Sakkalis, and N. M. Patrikalakis. Surface intersections with validated error bounds. Technical Report 2003-6, Design Laboratory, MIT, 2003.

[28] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1):21–68, 1999.

[29] N. M. Patrikalakis. Surface-to-surface intersections. *IEEE Computer Graphics and Applications*, 13(1):89–95, January 1993.

[30] N. M. Patrikalakis and T. Maekawa. Intersection problems. In G. Farin, J. Hoschek, and M. S. Kim, editors, *Handbook of Computer Aided Geometric Design, Chapter 25*, pages 623–650, Amsterdam, July 2002. Elsevier.

[31] N. M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer-Verlag, Heidelberg, February 2002.

[32] N. M. Patrikalakis and P. V. Prakash. Surface intersections for geometric modeling. *Journal of Mechanical Design, Transactions of the ASME*, 112(1):100–107, March 1990.

[33] M. J. Pratt and A. D. Geisow. Surface/surface intersection problems. In J. A. Gregory, editor, *The Mathematics of Surfaces*, pages 117–142. Clarendon Press, 1986.

[34] T. Sakkalis, G. Shen, and N. M. Patrikalakis. Topological and geometric properties of interval solid models. *Graphical Models*, 63(3):163–175, 2001.

[35] T. W. Sederberg, H. N. Christiansen, and S. Katz. Improved test for closed loops in surface intersections. *Computer-Aided Design*, 21(8):505–508, October 1989.

[36] T. W. Sederberg and A. K. Zundel. Pyramids that bound surface patches. *Graphical Models and Image Processing*, 58(1):75–81, January 1996.

[37] C.-K. Shene and J. K. Johnstone. On the lower degree intersections of two natural quadrics. *ACM Transactions on Graphics*, 13(4):400–424, October 1994.

[38] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, 10(5):379–405, October 1993.

[39] P. Sinha, E. Klassen, and K. K. Wang. Exploiting topological and geometric properties for selective subdivision. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 39–45. New York: ACM, 1985.

[40] M. N. Vrahatis. CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations. *ACM Transactions on Mathematical Software*, 14(4):330–336, December 1988.

[41] M. N. Vrahatis. Solving systems of nonlinear equations using the nonzero value of the topological degree. *ACM Transactions on Mathematical Software*, 14(4):312–329, December 1988.

[42] I. Wilf and Y. Manor. Quadric-surface intersection curves: shape and structure. *Computer-Aided Design*, 25(10):633–643, October 1993.

[43] S.-T. Wu and L. N. Andrade. Marching along a regular surface/surface intersection with circular steps. *Computer Aided Geometric Design*, 16(4):249–268, May 1999.

[44] X. Ye and T. Maekawa. Differential geometry of intersection curves of two surfaces. *Computer Aided Geometric Design*, 16(8):767–788, September 1999.