Analysis of Validated Error Bounds of Surface-to-Surface Intersection

K. H. Ko [a,*,1] N. M. Patrikalakis [b]

[a] *Modeling and Simulation Laboratory, Room 202, Department of Mechatronics, Gwangju Institute of Science and Technology, Gwangju, 500-712, Republic of Korea*

[b] *Design Laboratory, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*

Abstract

In this paper, we address the problem of computing the error bounds of surface to surface intersection and propose a novel procedure to reduce them. We formulate the surface to surface intersection problem as solving a system of ordinary differential equations and using the validated ODE solver we compute the validated *a priori* enclosures of an intersection curve, in which the existence and the uniqueness of a solution are guaranteed. Then we use straight line enclosures to reduce the size of the *a priori* enclosures. These reduced enclosures are again enclosed by bounding curves, which can be used as the reduced error bounds of the intersection curve. We demonstrate our method with tangential and transversal intersections.

Key words: validated ODE solver, validated error bounds, *a priori* enclosures, surface to surface intersection, the tracing method, reduction of enclosures

1 Introduction

Intersection computation has been an active research topic since the inception of computer aided design dating back to over twenty years ago. It is a core step in geometric and solid modeling and there has been substantial literature

* Corresponding author.
Email addresses: khko@gist.ac.kr (K. H. Ko), nmp@mit.edu (N. M. Patrikalakis).
[1] Tel: +82-62-970-3225, FAX: +82-62-970-2384

devoted to this topic [1]. However, among various types of intersections, surface to surface intersection (SSI) still remains a difficult problem in practice.

When two surfaces intersect, we can find the intersection by using various methods. In particular, the tracing method [1] is widely used in various applications. This approach requires a preliminary step, where all the critical points of the intersection are identified and the topological configuration for each intersection curve segment is performed [2,3]. Then we trace each SSI curve segment by solving a set of nonlinear ordinary differential equations formulated for SSI as an initial value problem (IVP) using the critical points as the starting and ending points [1]. The intersection curve segment can also be computed by solving a boundary value problem [2]. To reduce the difficulty involved in handling general free-form surface intersections, special types of surfaces such as ruled surfaces and swept surfaces have also been considered in the literature [4].

In intersection computation, we need to consider two issues: 1) the inherent approximation of intersecting surfaces in floating point arithmetic with limited precision and 2) the robustness of intersection calculation. Even though we define a surface with exact coefficients, unless exact arithmetic is used, the coefficients are inevitably approximated, creating an approximation to the surface. This implies that in intersection computation we are solving a problem which is close to what we are trying to define. In addition, the critical points of intersections which are used as the initial conditions for tracing intersection curves cannot be computed exactly due to the limited precision. Therefore, the numerical methods solving such an initial value problem will generate an approximation to the exact solution, which could result in numerical inaccuracy that may affect the resulting topological structure of the intersection curves. The second issue pertains to the robustness of the numerical methods solving ODEs. As indicated in [5,1], the standard numerical methods such as Runge-Kutta method, suffer from instabilities; when there exists a singular point, the behavior of the methods becomes unpredictable; and where distinct solutions are close to each other, the conventional numerical methods may suffer from *straying* or *looping*.

As a solution to such problems, Mukundan *et al.* [5] proposed to use the validated ordinary differential equation solver to trace intersections. Since it operates in interval arithmetic, the uncertainty involved in the surface definition as well as the critical point computation can be handled robustly. Moreover the validation step of the existence and the uniqueness of the solution can prevent a possible unstable behavior associated with the topological structure of the solution.

The result of the interval based SSI computation is directly related with the error bound analysis of SSI, which is important for geometric modeling and

manufacturing. A geometric model may be defined with a tolerance for considering possible errors introduced during computation and manufacturing. In particular, the SSI intersection part in the model can be represented as an approximate intersection curve with the associated error bounds. Mow *et al.* [6] provided a relation between the model and the parameter space error bounds for SSI curves using Taylor's theorem and compared the result with the Grandine-Klein (G-K) intersector [2]. They presented a convenient user interface for specifying the model space error bounds, which are, then, converted into corresponding parameter space error bounds that are provided as input to an intersection algorithm such as the G-K intersector. This approach, however, does not consider the validation aspect of the solution. In [5], the validation of the intersection is discussed by using the validated ODE solver. Then the enclosures of the solution are used as the validated error bounds of the exact solution. In surface to surface intersection, such enclosures are computed for each parametric variable with respect to the arc length parameter of the intersection curve. As an extension to [5], Mukundan *et al.* [7] analyzed the error bounds in model space and provided a relation between the parametric and the model space error bounds. The extension, however, may overestimate the error bounds of the intersection when the exact solution lies diagonally across each enclosure.

In order to deal with the two goals, the validation of the intersection and the reduction of the error bounds, we propose to use the validated ODE solver and reduce the validated regions associated with the error bounds of the intersection. The first simple method is to use the smaller stepsize for the validated ODE solver. This approach, however, increases the number of steps, leading to performance degradation. In this paper, we address the error bound issues and propose a novel method for reducing the validated error bounds of the exact intersection curve without using a smaller stepsize. The method takes constant *a priori* enclosures as input and reduces the region of each *a priori* enclosure by using higher order enclosure methods. It is proposed as a postprocess of the validated ODE solver and can reduce the enclosure size significantly in comparison to the enclosures obtained by the constant enclosure method. Then we discuss the error bound relation between the model and the parametric spaces.

This paper is structured as follows: In Section 2, we briefly review the validated ODE solver, which serves the foundation for the presentation of this paper. In Section 3, the proposed method is explained in detail with the application to surface to surface intersection. Section 4 discusses the model and the parametric space error bounds and Section 5 concludes this paper.

2 Computation of Error Bounds

In this section we briefly summarize the validated ODE solver and introduce various notations used in Section 3.

Given a continuous function $\mathbf{f} : \mathcal{D} \rightarrow \mathbf{R}^n$, $\mathbf{f} \in C^{k-1}(\mathcal{D})$ for an open set $\mathcal{D} \subset \mathbf{R}^n$, where k is a positive integer with $k > 1$, we define an autonomous initial value problem with an initial value $\mathbf{y}_0 \in \mathbf{R}^n$ [8,9]:

$$\mathbf{y}'(s) = \mathbf{f}(\mathbf{y}), \quad \mathbf{y}(s_0) = \mathbf{y}_0, \tag{1}$$

where $s \in [s_0, s_m]$ for some $s_m > s_0$ and $s_0, s_m \in \mathbf{R}$. Suppose we have a grid $s_0 < s_1 < \cdots < s_m$, $s_i \in \mathbf{R}$. The step from s_j to s_{j+1} is called the $(j+1)$ step with a stepsize $h_j = s_{j+1} - s_j$. We denote the solution of Equation (1) with an initial condition $\mathbf{y}_j$ at s_j by $\mathbf{y}(s; s_j, \mathbf{y}_j)$ [9,8].

In interval arithmetic, for an initial condition $[\mathbf{y}_j]$ at s_j, we have a set of solutions of Equation (1) [9]:

$$\mathbf{y}\left(s; s_j, [\mathbf{y}_j]\right) = \left\{\mathbf{y}\left(s; s_j, \mathbf{y}_j\right) | \mathbf{y}_j \in [\mathbf{y}_j]\right\}. \tag{2}$$

Here, we define an interval $[y]$ by a set of real values in $\mathbf{R}$, and $[y] = \left[\underline{y}, \overline{y}\right] = \left\{y | \underline{y} \leq y \leq \overline{y}\right\}$. An interval vector is defined as the vector whose components are intervals.

In general, the validated scheme for solving an IVP of an ordinary differential equation (ODE) consists of two phases [9]: 1) the stepsize selection and *a priori* enclosure computation, and 2) the tight enclosure computation.

Phase I: Stepsize Selection and a Priori Enclosure Computation

In this phase, a stepsize h_j and an *a priori* enclosure $[\tilde{\mathbf{y}}_j]$ of the solution to Equation (1) are computed such that $[\mathbf{y}_j] \subseteq [\tilde{\mathbf{y}}_j]$ and $h_j = s_{j+1} - s_j$. So $[\tilde{\mathbf{y}}_j]$ and h_j form a domain within which the existence and the uniqueness of $\mathbf{y}(s; s_j, \mathbf{y}_j)$ for all $s \in [s_j, s_{j+1}]$ and $\mathbf{y}_j \in [\mathbf{y}_j]$ are validated. Since $[\tilde{\mathbf{y}}_j]$ is determined from $[\mathbf{y}_j]$ at s_j, we have used j instead of $j+1$ for the index of the *a priori* enclosure $[\tilde{\mathbf{y}}_{j+1}]$. This step finds as large a step h_j as possible to reduce the number of discretization points in the final solution [9]. There exist several different approaches to find such a step [10,11,9,8] and we use the constant enclosure method [9,8] for computing *a priori* enclosures in this phase.

Phase II: Tight Enclosure Computation [8,9]

In this step, given $[\tilde{\mathbf{y}}_j]$ and h_j at s_j obtained from Phase I, a tight enclosure $[\mathbf{y}_{j+1}] \subseteq [\tilde{\mathbf{y}}_j]$ at s_{j+1} is computed such that

$$\mathbf{y}\left(s_{j+1}; s_j, [\mathbf{y}_j]\right) \subseteq [\mathbf{y}_{j+1}]\,. \tag{3}$$

This means that $[\tilde{\mathbf{y}}_j]$ is the *a priori* enclosure for a step $[s_j, s_{j+1}]$, and $[\mathbf{y}_j]$ and $[\mathbf{y}_{j+1}]$ where $[\mathbf{y}_j] \subseteq [\tilde{\mathbf{y}}_j]$ and $[\mathbf{y}_{j+1}] \subseteq [\tilde{\mathbf{y}}_j]$. The interval $[\mathbf{y}_{j+1}]$ at s_{j+1} is provided as the initial condition for the next step, which is used to determine the stepsize h_{j+1} and an *a priori* enclosure $[\tilde{\mathbf{y}}_{j+1}]$ at s_{j+1}. The naive computation of the enclosure $[\mathbf{y}_{j+1}]$ at s_{j+1} may result in the explosion of the *a priori* enclosure size in the subsequent integration step. This phenomenon is called *the wrapping effect* [11,12,9] and controlling such an effect is the primary goal of this phase. In order to control the size of the enclosure, the local coordinate transformation [12], the QR-factorization method [11], the interval Hermite-Obreschkoff method [8], etc. were proposed. In this work, we use the QR-factorization method by Löhner [11] for Phase II.

In combining Phases I and II, the validated ODE scheme computes the solution of a system of differential equations with the wrapping effect controlled. It generates constant *a priori* enclosures $[\tilde{\mathbf{y}}_j]$ and tight intervals $[\mathbf{y}_j]$ at s_j and those *a priori* enclosures can be used as the validated error bounds of the solution.

3 Reduction of Error Bounds

The algorithm given in Section 2 focuses on how to compute constant *a priori* enclosures and tight bounds avoiding the wrapping effect. However, the size of the constant *a priori* enclosures can be much larger than necessary so the associated error bounds are over-estimated. In this section we propose a novel method to reduce the size of the error bounds without taking a shorter stepsize h_j. We explain our method in detail and expand our discussion to surface to surface intersection.

3.1 Proposed Method

Assume that we have a system of nonlinear ODEs whose dimension is at least one as given in Equation (1), where $\mathbf{y}_j = ({}^1y_j, {}^2y_j, \cdots, {}^\gamma y_j)$, $\gamma \geq 1$. Then using the validated ODE method, we obtain $[\tilde{\mathbf{y}}_j]$ for $[s_j, s_{j+1}]$ and $[\mathbf{y}_{j+1}]$ at s_{j+1}. Consider one component ${}^\beta y_j$, $(1 \leq \beta \leq \gamma)$. Then the bounds at s_j and s_{j+1} and

the *a priori* enclosure for $[s_j, s_{j+1}]$ can be computed, which are schematically shown in Fig. 1. As shown in the figure, the constant enclosure is given in rectangular form in the ${}^{\beta}y$ and s domain. This rectangular shape, however, over-estimates the validated *a priori* enclosure, leading to the over-estimation of the error bound.

In order to reduce the constant *a priori* enclosure, we propose to use enclosures based on degree one polynomials. Given the constant *a priori* enclosure $\left[{}^{\beta}\tilde{y}_j\right]$, we compute two straight lines enclosing the exact solution starting at $(s_j, \overline{{}^{\beta}y_j})$ and $(s_j, \underline{{}^{\beta}y_j})$ towards s_{j+1} as given in two dotted lines in Fig. 1. Here, $\overline{{}^{\beta}y_j}$ and $\underline{{}^{\beta}y_j}$ are the upper and the lower values of the bound $\left[{}^{\beta}y_j\right]$. Similarly, we compute two lines enclosing the exact solution starting at $(s_{j+1}, \overline{{}^{\beta}y_{j+1}})$ and $(s_{j+1}, \underline{{}^{\beta}y_{j+1}})$ towards s_j. For this computation, we need to consider three different cases as shown in Fig. 2: 1) when the straight lines intersect within $\left[{}^{\beta}\tilde{y}_j\right]$ as shown in Fig. 2(a), 2) when they intersect the boundary of $\left[{}^{\beta}\tilde{y}_j\right]$ as shown in Fig. 2(b) and 2(c), and 3) when the two cases 1) and 2) happen at the same time.

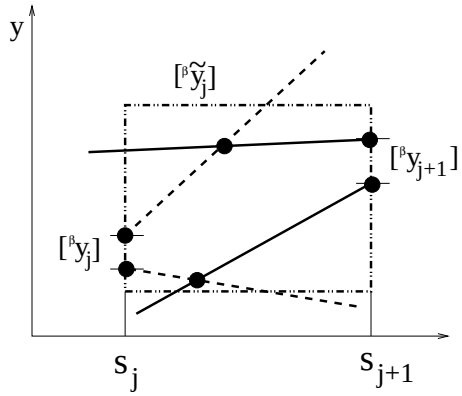

Fig. 1. The two straight lines are intersecting within the *a priori* enclosure at $[s_j, s_{j+1}]$ of the β component

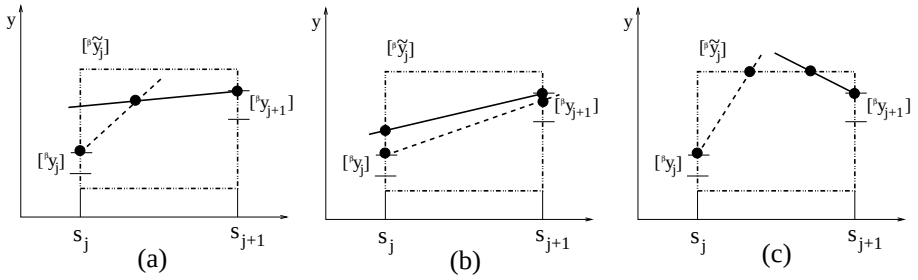

Fig. 2. The three different cases of intersections between two straight lines, and between the straight lines and the boundary of the *a priori* enclosure. The same cases apply for the lower bounding lines.

Then we compute the intersections either between the straight lines or between the boundary of the *a priori* enclosure and the straight lines depending on

the cases. After this we will have a polygon Π_j enclosed in the constant *a priori* enclosure $\left[{}^{\beta}\tilde{y}_j\right]$. One example of such a polygon is shown in Fig. 3. This

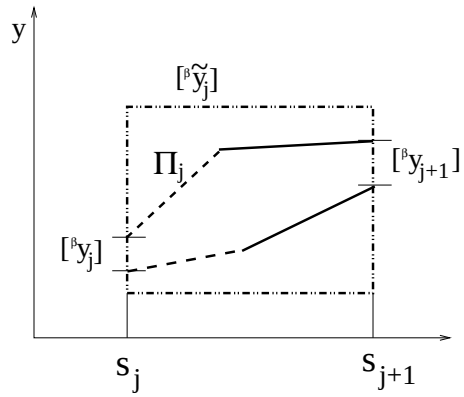

Fig. 3. The *a priori* enclosure with straight line enclosures from s_{j+1} and the final polygonal enclosure

procedure is similarly applied to the other y components, reducing the size of all the constant *a priori* enclosures.

In this procedure, the computation of the bounding straight lines that enclose the exact root is the key step and we propose to use the Taylor series [13] and the polynomial enclosure method [9] for this purpose. In this exposition, we omit the component index β and use y for simplicity instead of ${}^{\beta}y$ to indicate the β-th component of $\mathbf{y}$.

Taylor Series Enclosure Method

The validation given in Section 2 can be done by using the Taylor Series Enclosure method [13,9]. Given an *a priori* enclosure $[\tilde{y}_j]$, intervals $[y_j]$ and $[y_{j+1}]$ at s_j and s_{j+1}, we take the term of degree one to find

$$[y](s) = [y_j] + (s - s_j)\,[y_j]_1\,, \tag{4}$$

where the enclosure for the remainder is $[y_j]_1 = f\,([\tilde{y}_j])$. Equation (4) represents the upper and the lower bounds of the exact root in straight lines, starting from the upper and the lower values of $[y_j]$, respectively.

Similarly, we can compute another upper and lower bounds in straight lines, starting from the upper and the lower values of $[y_{j+1}]$ at s_{j+1} towards s_j.

Polynomial Enclosure Method

Based on the polynomial enclosure method introduced in [14,9], we use a polynomial of degree one as a bounding straight line for verification. We propose

the following three steps for computation:

(1) Let $[y](s) = [y_j] + (s - s_j)[v_{j,1}]$. Here, we set $[v_{j,1}] = \left[\tilde{y}^0_{j,1}\right] = f([\tilde{y}_j])$.
(2) Compute

$$\begin{aligned}[v](s) &= [y_j] + (s - s_j)[v_{j,1}] \\ &\subseteq [y_j] + [0, h_j][v_{j,1}] = [v^*].\end{aligned}$$

Then we set $[f_{j,1}] = f([v^*])$.
(3) Compute

$$\left[\tilde{y}^1_j\right] = [y_j]_1 + \frac{[f_{j,1}]}{2}[0, h_j]. \tag{5}$$

Then check if

$$\left[\tilde{y}^1_{j,1}\right] \subseteq [v_{j,1}] \left(= \left[\tilde{y}^0_{j,1}\right]\right). \tag{6}$$

If condition (6) is true, then the bounding enclosure in straight lines is given by $[y](s) = [y_j] + (s - s_j)\left[\tilde{y}^1_{j,1}\right]$. If not, set $[v_{j,1}] = \left[\tilde{y}^0_{j,1}\right] = f\left(\left[\tilde{y}^1_{j,1}\right] \cup [v_{j,1}]\right)$ and go to Step 2.

The goal of this iterative procedure is to determine $\left[\tilde{y}^0_{j,1}\right]$ such that condition (6) is satisfied. Under this condition, the existence and the uniqueness of the solution within the straight line enclosures at each step are guaranteed. If the condition does not hold, meaning that $\left[\tilde{y}^0_{j,1}\right]$ is not properly selected, we increase $\left[\tilde{y}^0_{j,1}\right]$ by taking the union of $\left[\tilde{y}^0_{j,1}\right]$ and $\left[\tilde{y}^1_{j,1}\right]$ and repeat the process until condition (6) is satisfied. We can always find $\left[\tilde{y}^0_{j,1}\right]$ such that condition (6) holds since the *a priori* enclosure $[\tilde{y}_j]$ has already been validated for the solution's existence and uniqueness. The same process can be applied to the computation of the bounding straight lines from $[y_{j+1}]$ at s_{j+1}.

In this paper we do not propose a method of how to compute $[\mathbf{y}_{i+1}]$ at s_{j+1} when $[\mathbf{y}_i]$ at s_j is given. Instead we use $[\mathbf{y}_j]$ at s_j, $[\mathbf{y}_{j+1}]$ at s_{j+1}, and $[\tilde{\mathbf{y}}_j]$ for $[s_j, s_{j+1}]$, respectively, which have already been obtained by using the validated ODE solver. Then we process the *a priori* enclosure to reduce its size. Since we use the computed bounds $[\mathbf{y}_i]$ and $[\mathbf{y}_{i+1}]$ at s_j and s_{j+1}, the wrapping effect does not have to be considered in the proposed procedure .

3.2 Approximation of Error Bounds

In this section, we provide a method to give the upper and the lower bounds of the exact root in polygonal form using the concept of the offset curve. With this method, we can represent both of the bounds compactly [1].

Suppose we have a curve $\alpha(s)$. Then the offset curve $\alpha^*(s)$ is given by

$$\alpha^*(s) = \alpha(s) \pm d\mathbf{n}(s), \tag{7}$$

where α is the progenitor, d the offset distance, s the independent variable and $\mathbf{n}$ the unit normal vector of α. The curve α can be approximated from the list of points $(s_j, \frac{(\overline{y}_j + \underline{y}_j)}{2})$ in the B-spline curve form. The choice of α could be arbitrary as long as it can be used as the progenitor of an offset curve. Then we compute the distances from the vertices of the upper and the lower bounding polylines to the curve α and compute the maximum distance which is set for d in Equation (7).

3.3 Analysis

Suppose that we have n *a priori* enclosures. Then the time complexity of the computation of bounding straight lines by either the Taylor series or the polynomial enclosure methods is $O(n)$ since we compute four straight lines for each *a priori* enclosure. However, an iteration may be needed for the polynomial enclosure method for validation. In such a case, we require more computation time but in most cases the number of iterations is small.

The computation of the maximum distance from each bounding vertex to the progenitor curve is implemented using Newton's method. It is not possible to estimate the time for this computation but we can denote it by t_ϵ. This computation time is proportional to the number of bounding vertices and for each range $[s_j, s_{j+1}]$ we have at most four bounding vertices including the ones at each s_j and s_{j+1}. Therefore, we have $O(4(n + t_\epsilon n))$ and since in most cases the actual computation requires less than three iterations, making $t_\epsilon << 1$, the worst case time complexity becomes $O(n)$.

We compute the progenitor planar curve $\alpha(s)$ which approximates the center points at each s_j in the least squares sense. If we use m control points for the approximation, then the time complexity for this step is $O(m^3 + mn)$ including matrix multiplication operations.

In conclusion, the time complexity of the proposed method mostly depends on the number of the *a priori* enclosures and the number of the control points of the progenitor. Most of the time is spent in the approximation of the progenitor curve.

3.4 Tracing Surface to Surface Intersection Curves Using a Validated ODE Scheme

In this section, we demonstrate how to compute the reduced validated error bounds of an intersection curve using the proposed procedure.

We have chosen two typical examples for demonstrating our method: transversal and tangential intersections. These two examples are enough for illustration since any complicated intersections can be decomposed into a set of monotonic components of transversal and tangential intersections.

Each intersection is formulated as a system of nonlinear ordinary differential equations with initial conditions [1]. In order to solve the system using the validated ODE solver we first reformulate it in terms of interval arithmetic. As indicated in [5], we replace each variable by a corresponding interval variable without changing the equations.

For implementation, we used the existing packages for interval arithmetic [15,16] and the validated ODE solver [8,9]. All programs were compiled using g++ 4.0.3 and run on a Linux computer with a 3.2GHz CPU and 2GB RAM.

3.4.1 Transversal Intersection

Figure 4 shows two transversally intersecting bicubic Bézier surfaces $\mathbf{P}(\sigma, t)$ and $\mathbf{Q}(u, v)$ with parameter domains $(\sigma, t) \in [0, 1]^2$ and $(u, v) \in [0, 1]^2$, which are taken from Mukundan *et al.* [5]. The initial conditions $(0.00000, 0.00000)$,

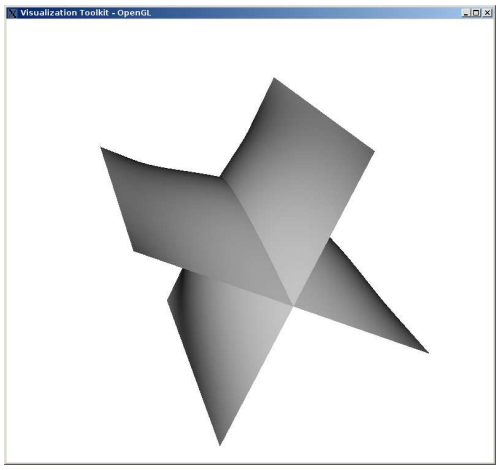

Fig. 4. Transversal intersection of two bicubic surfaces [5]

$(0.49999, 0.50001)$, $(0.00000, 0.00000)$ and $(0.49999, 0.50001)$ for σ, t, u and v, respectively, are used for computation. The validated ODE solver, then, traces the solutions for each parameter and generates a series of enclosures as given

in Fig. 5, where the constant *a priori* enclosures are shown with respect to the arc length s. The zoomed-in views of part of the *a priori* enclosures of each parameter with respect to s are superimposed in each image in Fig. 5 to show the detailed shapes of the enclosures. The widths of each initial value are 0 for σ and u, and 1.0×10^{-5} for t and u. The number of the *a priori* enclosures is 1810.

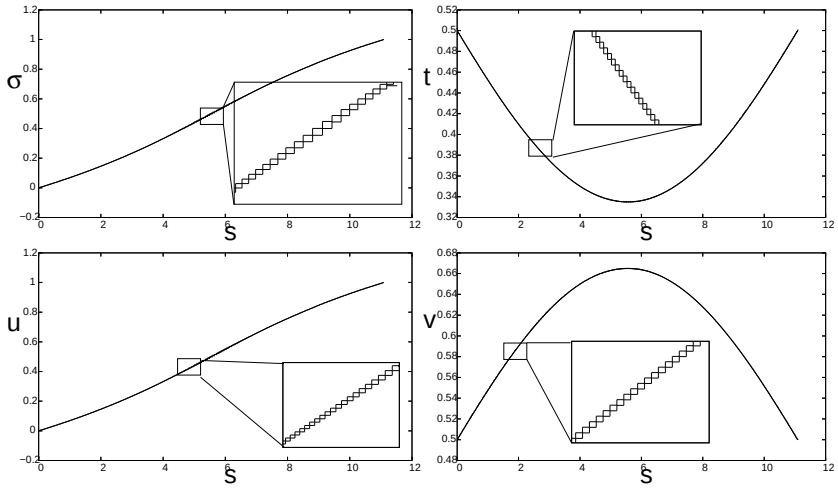

Fig. 5. The *a priori* enclosures for each parameter with respect to the arc length s

The two proposed reduction methods are applied to these constant enclosures. The amount of reduction against the constant enclosures is summarized in Table 1. The ratio means that the enclosed area from the proposed methods has been reduced by that amount compared to that of the constant enclosures.

Parameter	Taylor Enclosure	Polynomial Enclosure
σ	93.70%	99.60%
t	85.95%	95.65%
u	93.08%	99.58%
v	85.97%	96.59%

Table 1
Reduction ratios for each variable

The reduced enclosure for the parameter u using Taylor enclosure method is shown in Fig. 6 for illustration. The left image of Fig. 6 displays the constant and the reduced enclosures over the entire range of s and the right one shows the magnified figure of the portion circled in the left image, where the solid lines are the rectangular enclosures from the constant enclosure method and the dotted ones indicate the reduced enclosures by Taylor enclosure method.

The center points at each s_j are approximated using a cubic B-spline curve

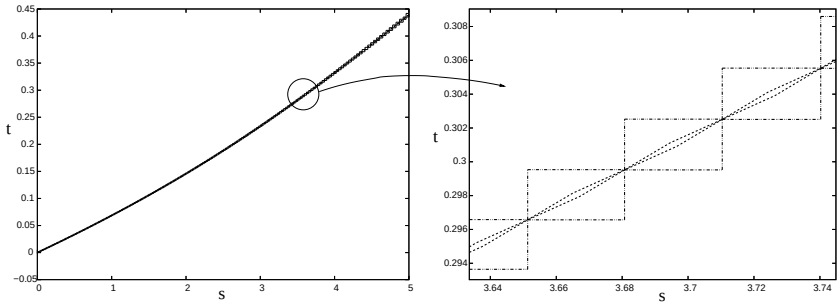

Fig. 6. The constant and reduced enclosures for the parameter u.

with 100 control points, which is used as a progenitor curve for the error bound computation. The maximum distances of the reduced enclosures and of the rectangular *a priori* enclosures from the progenitor curve are summarized in Table 2. The table contains maximum distances from the progenitor curve. So, the maximum widths of the bounding curves are twice the values in the table.

Domains	Taylor Enclosure	Polynomial Enclosure	Rectangular Enclosure
$\sigma - s$	1.4151×10^{-4}	9.0123×10^{-6}	8.7119×10^{-3}
$t - s$	1.1783×10^{-4}	1.7720×10^{-5}	1.4334×10^{-3}
$u - s$	1.6311×10^{-4}	9.7552×10^{-6}	8.7159×10^{-3}
$v - s$	1.1829×10^{-4}	1.5877×10^{-5}	1.1617×10^{-3}

Table 2
Maximum distances from the progenitor curve

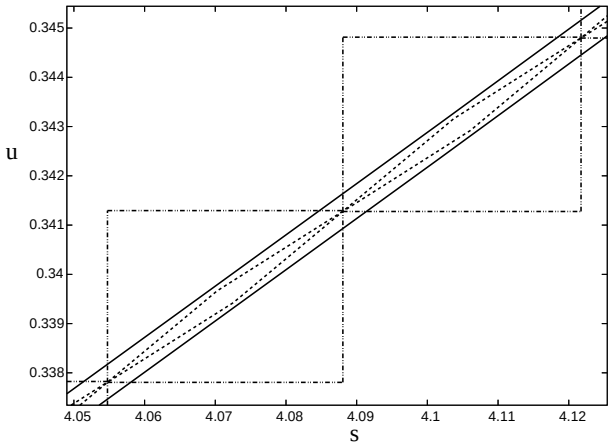

Fig. 7. The bounding curves of the reduced enclosures from Taylor enclosure method for the parameter u.

Figure 7 shows two bounding curves of the reduced enclosures generated by Taylor enclosure method. In the figure, the two thick solid curves are the bounding curves, the rectangles the constant enclosures and the dotted lines the reduced straight line enclosures.

In Fig. 8 the bounding curves obtained by Taylor and the polynomial enclosure methods are plotted in uv parametric domain. The right image contains part

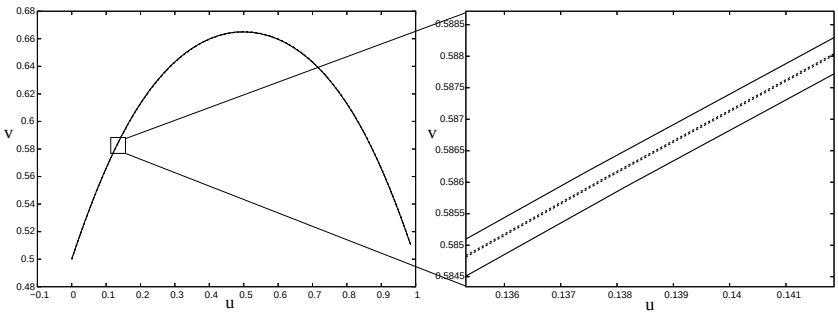

Fig. 8. The bounding curves by Taylor and the polynomial enclosure methods in uv parametric space

of the entire curves within the rectangle for better visualization. The outer solid curves are the bounding curves by Taylor enclosure method and the inner dotted ones by the polynomial enclosure method.

3.4.2 Tangential Intersection

In this section, we demonstrate our method to a tangential intersection case. Figure 9 shows two bicubic Bézier surfaces intersecting tangentially, which were taken from [17]. The initial interval values are given by $\sigma, u = [0.0000000, 0.0000000]$ and $t, v = [0.4999999, 0.5000001]$, respectively.

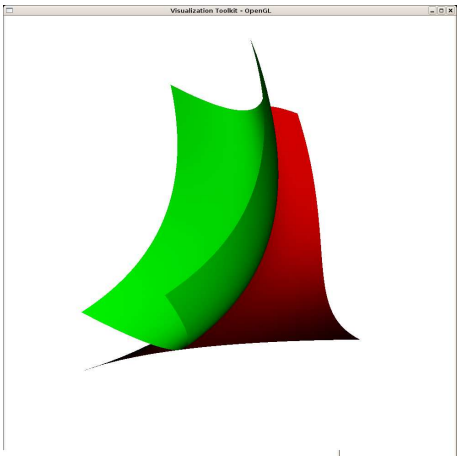

Fig. 9. Two tangentially intersecting surfaces [17]

The validated ODE solver generates 2789 *a priori* enclosures with an arc length of 207.4. Table 3 summarizes the reduction ratios compared to the constant enclosures. We use a cubic B-spline with 200 control points to compute the progenitor. The maximum distances of the reduced enclosures and of the constant *a priori* enclosures from the progenitor are summarized in Table 4. The

Parameter	Taylor Enclosure	Polynomial Enclosure
σ	98.51%	99.99%
t	89.35%	99.38%
u	99.22%	99.99%
v	88.08%	99.71%

Table 3
Reduction ratios for each variable

Domains	Taylor Enclosure	Polynomial Enclosure	Constant Enclosure
$\sigma - s$	2.1577×10^{-5}	1.2740×10^{-6}	2.0986×10^{-3}
$t - s$	2.8052×10^{-5}	2.3918×10^{-6}	2.6951×10^{-4}
$u - s$	1.3364×10^{-5}	1.1331×10^{-6}	2.0992×10^{-3}
$v - s$	5.2834×10^{-5}	2.6521×10^{-6}	6.8490×10^{-4}

Table 4
Maximum distances from the progenitor curve

maximum widths of the bounding curves are twice the values in the table.

Figure 10 shows the bounding curves of v with respect to s. A blown-up image of part of the bounding curves is provided in the right for illustration where the solid curves are the bounding curves by Taylor enclosure method and the dotted ones are those by the polynomial enclosure method.

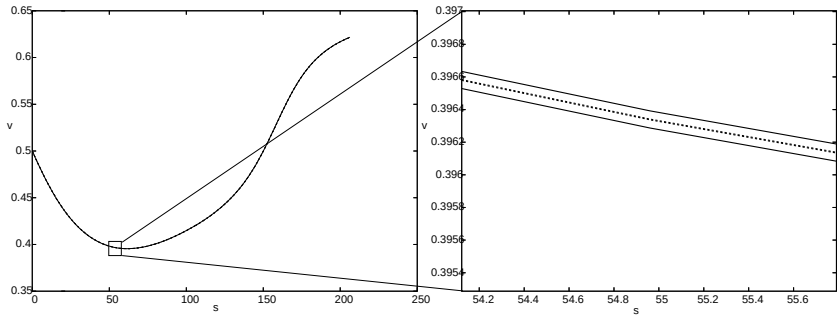

Fig. 10. The bounding curves of the Taylor and polynomial enclosure methods with respect to s

4 Model and Parametric Space Error Bounds

As pointed out in [6], when a model is defined by parametric surfaces, tolerances given for parametric space are not intuitively understood in practice since in most cases it is in model space where the tolerances are provided for design and manufacturing. Therefore, we need a method to relate a tolerance

in model space to that in parametric space. In particular, for an SSI problem, given a tolerance in model space, we compute the corresponding tolerances for each parametric variable using the methods by Mow *et al.* [6] or Mukundan *et al.* [7]. The former can be used for intersections of general surfaces, whereas the latter presents a limited case of two bicubic surface intersection based on interval arithmetic. Once we have the tolerances of each parametric variable, we can iteratively reduce the validated error bounds whose size is less than the tolerances for each parameter.

5 Conclusions

In this paper, we propose a novel method to compute the validated error bounds of surface to surface intersection. We compute the validated constant *a priori* enclosures using the validated ODE solver and reduce them through the straight line enclosure methods without decreasing the step size. Then we compute the bounding error curves for compact representation of those reduced enclosures.

We believe that the validation of the reduced error bounds generated by the proposed method opens a way of guaranteeing the validity of a geometric model near the intersection by considering both the topological and numerical aspects in intersection computation at the same time. With this method, we can possibly achieve a topologically consistent model. Moreover, the proposed method can be interfaced with interval solid modeling method with no extra effort. Further investigation of these topics is a subject of future research.

Acknowledgments

This work was supported in part by the US NSF (grant No. DMS-0138098, CCR-0231511, and DMI-062933) and by the Korea Research Foundation grant funded by the Korean Government (MOEHRD) (KRF-2006-331-D00036). Any findings, opinions or recommendations provided in this paper are those of the authors and do not represent the official views of the US NSF or the MOEHRD.

References

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

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

[3] T. Sakkalis, The topological configuration of a real algebraic curve, Bulletin of the Australian Mathematical Society 43 (1991) 37–50.

[4] J.-K. Seong, K.-J. Kim, M.-S. Kim, G. Elber, R. R. Martin, Intersecting a freeform surface with a general swept surface, Computer-Aided Design 37 (5) (2005) 473–483.

[5] H. Mukundan, K. H. Ko, T. Maekawa, T. Sakkalis, N. M. Patrikalakis, Tracing surface intersections with a validated ODE system solver, in: G. Elber, N. M. Patrikalakis, P. Brunet (Eds.), Proceedings of the Ninth EG/ACM Symposium on Solid Modeling and Applications, EG/ACM, Eurographics Press, Genoa, Italy, 2004, pp. 249–254.

[6] C. Mow, T. J. Peters, N. F. Stewart, Specifying useful error bounds for geometry tools: an intersector exemplar, Computer Aided Geometric Design 20 (5) (2003) 247–251.

[7] H. Mukundan, K. H. Ko, N. M. Patrikalakis, Intersections with validated error bounds for building interval solid models, in: Proceedings of the IDETC/CIE 2005, ASME Design Engineering Technical Conferences, ASME, Long Beach, CA, USA, 2005.

[8] N. S. Nedialkov, Computing the rigorous bounds on the solution of an initial value problem for an ordinary differential equation, Ph.D. thesis, University of Toronto, Toronto, Canada (1999).

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

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

[11] R. J. Löhner, Computation of guaranteed enclosures for the solutions of ordinary initial and boundary value problems, in: J. Cash, I. Gladwell (Eds.), Computational Ordinary Differential Equations, Clarendon Press, Oxford, 1992, pp. 425–435.

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

[13] G. F. Corliss, R. Rihm, Validating an a priori enclosure using high-order Taylor series, in: G. Alefeld, A. Frommer, B. Lang (Eds.), Scientific Computing and Validated Numerics: Proceedings of the International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics - SCAN '95, Akademie Verlag, Berlin, 1996, pp. 228–238.

[14] R. J. Löhner, 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, CA, 1995.

[15] O. Knuppel, Bias-basic interval arithmetic subroutines, Technical Report 93.3, Technical University of Hamburg-Harburg, Harburg, Germany (1993).

[16] O. Knuppel, Profil-programmers runtime optimized fast interval library, Technical Report 93.4, Technical University of Hamburg-Harburg, Harburg, Germany (1993).

[17] H. Mukundan, Surface-surface intersection with validated error bounds, Master's thesis, Massachusetts Institute of Technology, Cambridge, MA (February 2005).