An efficient and robust tracing method based on matrix representation for surface-surface intersection

Hongyu Chen, Xiao-Diao Chen
School of Computer, Hangzhou Dianzi University
Hangzhou, China
chenhongyu651@gmail.com(Hongyu Chen), xiaodiao@hdu.edu.cn(Xiao-Diao Chen)

Abstract

The surface-surface intersection(SSI) problem is a fundamental one in CAD/CAM. Tracing methods are widely applied for solving the SSI problem, where branch jumping and branch-missing are two of the challenging things. This paper takes the intersection problem between two parametric surfaces as an example and presents an SRS-BFS method based on the Dixon matrix technique, which turns the tracing process into a simple root-solving (SRS) problem and introduces breadth-first searching (BFS) method for solving branch points robustly. Moreover, it provides an Edge-Chain-Tracing (ECT) method to further improve the tracing efficiency. Extensive experiments have been conducted on various surfaces, and their floating-point representations have been used as well. These examples have covered rich intersection curve topology with multiple branches and singular points. All examples show that the SRS-BFS method can avoid branch jumping problems, and achieves higher efficiency and robustness, even with their floating-point representation and a given tolerance.

***Keywords:** parametric surface tracing methodbranch jumping loop detectionSRS-BFS method.*

1. Introduction

The surface/surface intersection (SSI) problem is a fundamental one in CAD/CAM [15] and computer graphics [12]. It is useful for the design of complex objects and NC machining [1]. It is also taken as one of the most challenging technical issues in geometric and solid modeling [11, 17]. As pointed out in [15], the lack of topologically consistent surface-surface intersection algorithms becomes a common consensus on the greatest cause of poor reliability of CAD systems.

In principle, the SSI can be defined as solutions of an equation system with different dimensions, i.e., non–solution, zero–dimension of finite points, one–dimension of curve branches, or even two–dimension of coincident surface patches [11]. A good intersection algorithm should efficiently and correctly compute all of these features [7, 15, 13].

Typical SSI approaches include the algebraic method [15], the tracing method [4], the lattice method [20], the subdivision method [20], the homotopy method [14], as well as hybrid methods [18]. Different methods have their own advantages and disadvantages. For example, the algebraic method can preserve the geometric features such as small loops and singular points on the intersection curve to a great extent, but it usually needs the implicitization of one surface which is a challenging problem [22, 16]. Similarly, the tracing method can be easily implemented and is widely adopted in many CAD systems. However, the starting points for tracing need to be accurately and efficiently solved from a non-linear equation system, and furthermore, branch jumping or loop missing of the intersection curves is not easy to avoid in the tracing process. The lattice method and the subdivision method are an easy common way to compute the discrete points on the intersection curve, but they are generally not efficient enough and tend to miss small loops or isolated points [4, 20, 21, 5].

Hybrid methods are a good trade-off between efficiency and robustness, and they are more widely applied [7], such as the hybrid one of recursive subdivision and curve tracing [3, 12], the hybrid one of the algebraic method and the tracing method [4, 20, 21, 5]. A matrix representation-based method combined with the lattice method is provided in [15]. It significantly improves the efficiency of the corresponding SSI algorithm. Recently, the method in [27] improves the topology of the intersection. However, the topology of the intersection may be confused, which leads to either branch jumping or missing intersection segments, see also the case shown in Fig. 6(g).

This paper takes the intersection problem between two parametric surfaces as an example, and presents a SRS-BFS method based on the Dixon matrix technique, which turns the tracing process into a simple root-solving (SRS) problem, and introduces the breadth-first searching (BFS)

method for solving branch points robustly. Given two parametric surfaces, a bivariate function $M(u, v)$ in Dixon matrix representation is obtained by using the method in [15]. Firstly, given a starting point, an Edge-Chain-Tracing (ECT) method is presented to trace the lattices of the smallest size, which can reduce much of the lattices used in [15], and leads to a much better efficiency. Secondly, we turn the intersection point computation problem into a simple root-solving problem, which can detect whether or not there are two branches in a lattice. Partly because of the high accuracy of the intersection points, numerical experiments show that the SRS-BFS method can satisfy the given tolerance even with a bigger size of the lattices. Finally, a breadth-first searching is introduced to achieve good initial values for solving the branch intersection points, and improves the corresponding robustness. Experimental results verify both the efficiency and the effectiveness of the proposed method.

The main contributions are summarized as follows:

1. The SRS-BFS method is proposed for the SSI problem, which can avoid the branch jumping or loop-missing problem, preserve the topology of the intersection (under the given tolerance), and achieve better efficiency and robustness, even with floating-point arithmetic.

2. It combines the ECT method with the previous matrix representation, which admits the advantages of the matrix representation and further improves the efficiency significantly.

3. Fixing one parameter turns the tracing process into a simple root-solving (SRS) problem; and combining with the BFS method, can efficiently trace the intersection point and robustly solve a branch point or a singular point.

2. Related work

The SSI problem has been extensively explored [6–9,14]. Typical approaches include the algebraic method, the tracing method, the lattice evaluation method, the subdivision method, the hybrid method, coincidence detection methods, and so on.

Algebraic methods. The algebraic methods usually utilize the implicit equation of one of the two parametric surfaces for substituting the parametric form of the other surface, and finally, obtain an algebraic curve in the parametric domain [7,10]. The advantage of algebraic methods is that they preserve the algebraic properties of the intersection curve well. However, the implicitization of rational surfaces of a high degree is not easy in general; moreover, the obtained algebraic curve in parametric domain generally has a high degree, which causes difficulty in sequence computations.

Tracing methods. The tracing methods are the most widely used methods for surface-surface intersection. The strengths of this technique include generality and simplicity. The tracing methods determine a starting point for each intersection branch and then trace out a sequence of intersection points from each starting point using the local differential geometry [1,6,13,15]. The disadvantages of such methods are the requirement of starting points which are not easy to hunt and the determination of tracing directions at singular points, and even worse it may suffer from branch jump problems.

Lattice evaluation methods. The lattice evaluation methods decompose one of the two surfaces to its isoparametric curves, and then compute the intersections of these isoparametric curves with the other surface [12]. The final intersection curve is obtained by connecting the resulting discrete points. The main problem here is that an improper choice of grid size will lose small loops and isolated points.

Subdivision methods. The main idea of subdivision methods is to decompose the problem into simpler ones recursively, which has direct solutions such as plane/plane intersection. Combining all the individual solutions together gives the complete intersection curve. However, in the practical implementation of finite subdivision steps, small loops may be missed and it is difficult to guarantee the correct connectivity near the singular points [8,16].

Hybrid methods. Hybrid methods combine two or more methods to take advantage of them [9,13]. For example, hunting starting points using algebraic methods, and then tracing out the intersection curve using marching methods. The hybrid of the algebraic method and the tracing method is often used to balance the efficiency and the topological correctness of the intersection curve. However, the branch jumping problem or the lack of segments problem still exists.

Coincidence detection methods. For the coincidence case where two surfaces coincide with each other, the tracing method and its corresponding hybrid methods may fail because of the following two reasons [19, 24]: (1) The subdivision method may lead to endless subdivisions and the system may crash because of memory depletion; and (2)the equation system for solving the next intersection point is degenerated. In 1997, Berry and Patterson [23] discussed the uniqueness of Bézier control points. In 2011, Wang et.al. [24] proved the coincidence condition between two cubic Bézier curves in a different way from that of [23]. Later, Chen et.al. [8, 10] pointed out that if a Bézier curve is reparameterized in two different polynomials of the same degree, the resulting two Bézier curves of the same degree coincide but have a non-coincident control polygon. On the other hand, if two Bézier curves are properly parameterized which cannot be reparameterized into other Bézier curves of a lower degree, the corresponding coincidence condition

is equivalent to the uniqueness of Bézier control points. The coincidence problem of two rational cubic Bézier curves was discussed in [10]. Recently, Wang, Chen, and Yong discussed proper reparameterization of plane rational Bézier curves [25]. Because of the floating-point arithmetic, coincidence conditions with tolerance control between two curves or surfaces are still challenging.

3. Preliminaries

Given two parametric surfaces $\mathbf{S}_1(u,v) = (\frac{x_1(u,v)}{w(u,v)}, \frac{y_1(u,v)}{w(u,v)}, \frac{z_1(u,v)}{w(u,v)})$ and $\mathbf{S}_2(s,t) = (\frac{x_2(s,t)}{\gamma(s,t)}, \frac{y_2(s,t)}{\gamma(s,t)}, \frac{z_2(s,t)}{\gamma(s,t)})$. The corresponding intersection problem is equivalent to solving the following equation system

$$\begin{cases} f(s,t) = x_1(u,v)\gamma(s,t) - x_2(s,t)w(u,v) = 0, \\ g(s,t) = y_1(u,v)\gamma(s,t) - y_2(s,t)w(u,v) = 0, \\ h(s,t) = z_1(u,v)\gamma(s,t) - z_2(s,t)w(u,v) = 0. \end{cases} \quad (1)$$

3.1. Eliminating s and t by using Dixon matrix

Suppose $f(s,t)$,$g(s,t)$ and $h(s,t)$ are polynomials of degree n and m in s and t, respectively. Let

$$\begin{cases} \rho(s,t,\alpha,\beta) = \begin{vmatrix} f(s,t) & g(s,t) & h(s,t) \\ f(\alpha,t) & g(\alpha,t) & h(\alpha,t) \\ f(\alpha,\beta) & g(\alpha,\beta) & h(\alpha,\beta) \end{vmatrix}, \\ \delta(s,t,\alpha,\beta) = \dfrac{\rho(s,t,\alpha,\beta)}{(s-\alpha)(t-\beta)}, \\ = \begin{pmatrix} \alpha^{2m-1}\beta^{n-1} \\ \vdots \\ \alpha^{2m-1} \\ \vdots \\ \beta^{n-1} \\ \vdots \\ 1 \end{pmatrix}^T D(u,v) \begin{pmatrix} s^{m-1}t^{2n-1} \\ \vdots \\ s^{m-1} \\ \vdots \\ t^{2n-1} \\ \vdots \\ 1 \end{pmatrix} \end{cases} \quad (2)$$

$D(u,v)$, which is a square matrix of size $2mn \times 2mn$, and its determinant $|D(u,v)|$ are defined as the corresponding Dixon matrix and Dixon resultant [2, 26].

Suppose rank(D)=r. It is worth noting that the corresponding matrix representation $\bar{M}(x,y,z,w)$ of $\mathbf{S}_2$ is deduced in [15, 27], which is of size $r \times r$. Let $\mathbf{p}_0 = (u_0, v_0)$. If $\mathbf{S}_1(\mathbf{p}_0) = \mathbf{S}_1(u_0,v_0)$ is an intersection point on $\mathbf{S}_2$, we have that

$$\begin{aligned} &F(u_0,v_0) = F(\mathbf{p}_0) \\ &= \det(\bar{M}(x_1(\mathbf{p}_0), y_1(\mathbf{p}_0), z_1(\mathbf{p}_0), w(\mathbf{p}_0))) = 0. \end{aligned} \quad (3)$$

In this paper, we refer to references [15, 27] for more detailed explanations of the mathematical formulations and specific implementation steps for the Dixon matrix technique.

3.2. Coincidence detection

As shown in Section 3.1, the corresponding intersections should satisfy

$$F(u,v) = 0, \quad (4)$$

where u and v are parameters of $\mathbf{S}_1(u,v)$. In principle, if $\bar{M}(x_1,y_1,z_1,w)$ is in polynomial form of x_1, y_1, z_1, w, and $\mathbf{S}_1(u,v)$ is a NURBS surface, $F(u,v)$ can also be written into NURBS form as

$$F(u,v) = \frac{\sum_{i=0}^{k}\sum_{j=0}^{l} w_{i,j}p_{i,j}B_{n,i}(u)B_{m,j}(v)}{\sum_{i=0}^{k}\sum_{j=0}^{l} w_{i,j}B_{n,i}(u)B_{m,j}(v)}, \quad (5)$$

where $B_{n,i}(u)$ and $B_{m,j}(v)$ are B-spline basis functions of degree n and m, $w_{i,j}$ and $p_{i,j}$ are weight and control point, respectively.

If $F(u,v) \equiv 0$ for $(u,v) \in [a,b]\times[c,d]$, i.e., $p_{i,j} = 0$ for all of $i = 0, 1, \cdots, k$, $j = 0, 1, \cdots, l$, one has that $\mathbf{S}_1(u,v)$ coincides with the other surface [23]. Then, the coincidence regions are detected and the resulting intersection surface patches are returned.

3.3. Solving a simple root case within an interval

Assume that there is a simple root $t^\star \in [a,b]$ of $g(t)$, such that $g(a)\cdot g(b) < 0$. By using the SRS method for solving a simple root in [9], the explicit formula of t_i is iteratively computed as follows without derivatives.

$$t_i = \frac{(\sum_{j=1}^{i-1} A_{i,j}t_j)\cdot(\prod_{k=1}^{i-1} g(t_k))}{(\sum_{j=1}^{i-1} A_{i,j})\cdot(\prod_{k=1}^{i-1} g(t_k))} = \frac{\sum_{j=1}^{i-1} A_{i,j}t_j}{\sum_{j=1}^{i-1} A_{i,j}}, \ i \geq 3, \quad (6)$$

where

$$D_{i,j} = \begin{cases} 1, & i = 3, \\ \prod_{\substack{1 \leq k < r \leq i-1, \\ k,r \neq j,}} (t_k - t_r), & i > 3, \end{cases}$$

and $A_{i,j} = (-1)^j \cdot \dfrac{D_{i,j}}{g(t_j)}$.

4. The SRS-BFS method

As shown in Eq. (4), the intersection curves can be represented by $F(u,v) = 0$, see more details deduced in the methods in [15, 27]. In principle, the size of a grid can be fixed as the given smallest size Δ in ECT, where no redundant grids are traced, which means much better efficiency of the ECT method. By using the SRS method, even given a bit big size grid, the topology of the grid can be verified to be correct in all of the testing cases. By using the BFS method, good initial values are obtained for solving branch points, even for complicated cases.

4.1. The outline of the SRS-BFS algorithm

The outline is as follows.

Algorithm 1. The SRS-BFS algorithm of SSI.
Input: Two parametric surfaces $\mathbf{S}_1$ and $\mathbf{S}_2$.
Output: The intersection curves.

1. **Begin:** Compute the initial points (Section 4.2).
2. By using the Edge-Chain-Tracing method, the intersection is traced branch by branch (Section 4.3).
3. Compute the intersection point on each edge by using the SRS method (Section 4.4).
4. Tracing from a breakpoint pair by using the BFS method (Section 4.5).
5. **End:** Output all of the intersection curves.

4.2. Computing the initial points for tracing processes

The initial points mainly consist of two parts: (1) the boundary points such that

$$F(u,i)=0 \text{ or } F(j,v)=0; \tag{7}$$

where $i,j\in\{0,1\}$; and (2) the extreme points such that

$$F(u,v)=F_u(u,v)=0 \text{ or } F(u,v)=F_v(u,v)=0. \tag{8}$$

The methods in [6, 18] are applied for solving the equation systems (7) and (8).

Remark 1. If there are two branches of intersection curves intersected at point $\mathbf{p}$ (named as branch point) of parameters (u,v), one obtains

$$F(u,v)=F_u(u,v)=F_v(u,v)=0. \tag{9}$$

In principle, the parameter (u,v) of an isolated point also satisfies Eq. (9).

4.3. The Edge-Chain-Tracing method for tracing a branch

As shown in Fig. 1, the distribution of $F(u,v)$ at a 2×2 grid can be divided into four classes: (a) no edges; (b) four edges; (c) two non-parallel edges; and (d) two parallel edges.

For cases (c) and (d), there is one inside edge (denoted by a black arrow) and one outside edge (denoted by a red arrow). Firstly, starting from an inside edge e_i of a grid g_i, there is one outside edge e_{i+1} and unique grid g_{i+1} sharing e_{i+1} with grid g_i. And then, by tracing g_{i+1}, the edge e_{i+1} is taken as the inside edge, and we obtain the unique outside edge e_{i+2} of g_{i+1}. The above tracing process continues until either the grid has been used or the grid is a boundary

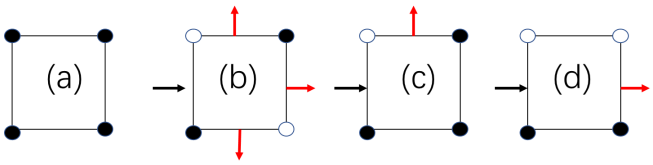


Figure 1: Illustration of four classes of the distribution of $F(u,v)$ at a 2×2 grid.

grid such that there is no grid sharing with the last edge. Finally, the edge chain consisting of $\{\cdots,e_i,e_{i+1},e_{i+2},\cdots\}$ is achieved. Fig. 2 illustrates more details of the ECT process. The process starts at grid $1\#$ from its boundary, the next but unique grid shared the edge in dashed red with the $i\#$ grid are grid $(i+1)\#$, where $i=1,2,\cdots,6$. Note that grid $7\#$ is on the boundary such that there is no grid shared the edge in dashed red with it, the corresponding ECT process stops.

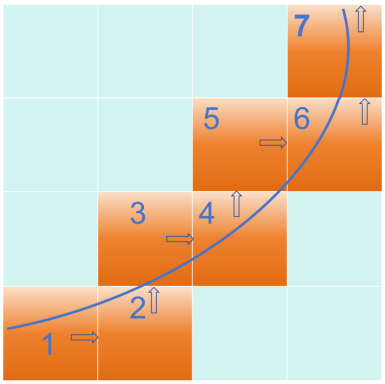


Figure 2: Illustration of ECT process.

Remark 2. In this paper, the branch tracing process stops at the grid of case (b); otherwise, the grid is traced in three different directions instead.

4.4. Computing the intersection point on each edge by using the SRS method

Starting from the intersection point $\mathbf{I}_i$ on e_i, we want to compute the next intersection point $\mathbf{I}_{i+1}$ on e_{i+1}. Suppose that the grid g_{i+1} containing e_i and e_{i+1} has four points $\mathbf{p}_{i,j}, j=1,2,3,4$, where $\mathbf{p}_{i,j}, j=1,2$ are two end points of e_i.

Firstly, by applying the SRS method without derivatives,

$$\bar{F}_i(t)=F(\mathbf{p}_{i,1}\cdot(1-t)+\mathbf{p}_{i,2}\cdot t)=0$$

can be solved and the intersection point on e_i is computed.

Secondly, suppose that the ray $\mathbf{I}_{i-1}(1-t)+\mathbf{I}_i t, t>0$ prior to intersect with L_i, which is one of the three edges

of grid g_{i+1}, if L_i is checked to be the outside edge e_{i+1} of grid g_{i+1} by computing the sign of $F(\mathbf{p}_{i,3})$, then the sign of $F(\mathbf{p}_{i,4})$ is unnecessary to compute, which will speed up the Edge-Chain-Tracing process.

Thirdly, for a single branch tracing case, the SRS method can be applied for tracing the intersection points without grid computation. Given a point $\mathbf{I}_i(u_i, v_i)$ on a branch, we trace the next point $\mathbf{I}_{i+1}(u_{i+1}, v_{i+1})$ as follows. Note that

$$\begin{aligned} &0 = F(\mathbf{I}_{i+1}) \approx \\ &F(\mathbf{I}_i) + F_u(\mathbf{I}_i)(u_{i+1} - u_i) + F_v(\mathbf{I}_i)(v_{i+1} - v_i) \\ &= F_u(\mathbf{I}_i)(u_{i+1} - u_i) + F_v(\mathbf{I}_i)(v_{i+1} - v_i). \end{aligned} \tag{10}$$

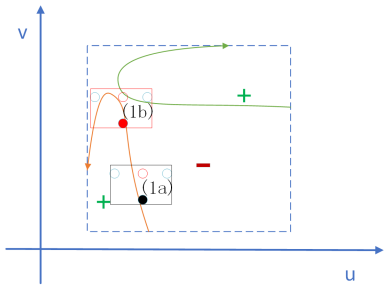


Figure 3: Illustration of two possible cases (1a) and (1b).

Without loss of generality, assume that $|F_u(\mathbf{I}_i)| > |F_v(\mathbf{I}_i)|$, Eq. (10) leads to

$$|\Delta| = |u_{i+1} - u_i| \geq |v_{i+1} - v_i|. \tag{11}$$

As shown in Fig. 3, by setting $u_{i+1} = u_i + \Delta \cdot k$, where $k \in \{1, -1\}$ denotes the tracing direction, there may be three possible cases:(1) if $F(u_i + \Delta, v_i + \Delta) \cdot F(u_i + \Delta, v_i - \Delta) < 0$, there may be two cases: (1a) A simple root v_{i+1} of $F(u_i + \Delta, v) = 0$ which can be successfully solved by the method in Section 3.3, and (1b) multiple roots of $F(u_i + \Delta, v) = 0$ such that the method in Section 3.3 converges slowly or even fails, the tracing process stops at this marching direction, and mark (u_i, v_i) as a breakpoint; (2) if $F(u_i + \Delta, v_i + \Delta) \cdot F(u_i + \Delta, v_i - \Delta) = 0$, the root v_{i+1} is verified to be either $v_i + \Delta$ or $v_i - \Delta$; (3) if $F(u_i + \Delta, v_i + \Delta) \cdot F(u_i + \Delta, v_i - \Delta) > 0$, there may be multiple roots of $F(u_i + \Delta, v)$ within $[v_i - |\Delta|, v_i + |\Delta|]$, the tracing process stops at this marching direction, and mark (u_i, v_i) as a breakpoint.

Remark 3. A breakpoint means that there may be two or more branches locate in a local region or sub-grid.

4.5. Tracing from a breakpoint pair by using the BFS method

Firstly, we illustrate how to trace a branch starting from a breakpoint pair. For the breakpoints obtained in the above tracing process in Section 4.4, we make $(\mathbf{b}_i, \mathbf{b}_j)$ as a pair if $||\mathbf{b}_i - \mathbf{b}_j|| \leq 2\sqrt{2}\Delta$. As shown in Fig. 4, it is divided into two cases: (a) there are branches to be traced; and (b) there is no branch and the tracing process stops. These two cases can be distinguished by using the signs of the derivative $F_u(u, v)$ or $F_v(u, v)$, e.g., case (a) has different signs of the derivatives at points $\mathbf{b}_3$ and $\mathbf{b}_4$, while case (b) usually has the same signs as the derivatives at $\mathbf{b}_3$ and $\mathbf{b}_4$. Thus, we can isolate the two roots by solving the root of the derivative $F_u(u, v)$ or $F_v(u, v)$, and then obtain two single-root-solving problems.

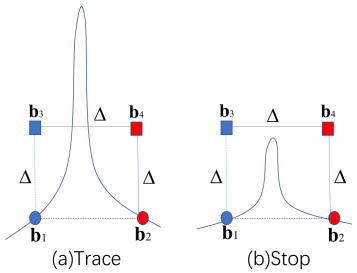


Figure 4: Illustrating two cases of breakpoint tracing:(a) Two intersection points; and (b) no intersection points.

Secondly, we show how to solve or refine the branch point by using the BFS method. As shown in Fig. 5, there are four starting points and four breakpoints denoted by solid circles in red and in black, respectively. Different from the Depth-First Searching method which traces branch by branch, the BFS method simultaneously traces the branches from all of the four starting points, which stops at the breakpoints. Combining the SRS technique with the information of the breakpoints, one can solve the possible branch point, e.g., one traces the four breakpoints with half-step size Δ/k one by one, where $k = 2$, these breakpoints synchronously approach and get closer and closer to the branch point. And then, the topology of the local region can be distinguished with the given tolerance, even for the case in Example 5.7 which is also shown in Fig. 6(g). In principle, once a smaller k is used, these breakpoints get closer to the branch point. Moreover, if necessary, the barycentre of these breakpoints is taken as the initial value for solving the possible branch point, as shown in Fig. 5(b–c).

5. Examples and discussions

This section provides some examples that show the effect of our technology in a range of challenging circumstances. Our examples are divided into two parts: (1) An algebraic surface and a parametric surface which needs no matrix representation, including the intersection of quadratic surfaces with classic surfaces that have complex self-intersection properties, such as parametric surface and octahedron surfaces, and the intersection between quadratic surfaces and a

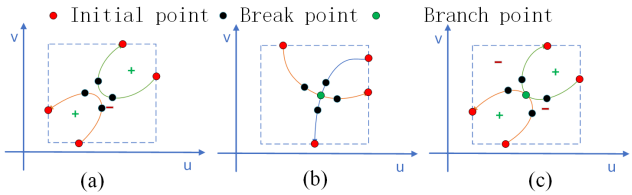


Figure 5: Illustration of tracing from breakpoint pairs (denoted as a solid circle in black): (a) No branch point; (b) a transverse branch point; and (c) a contact branch point.

NURBS surface; and (2) two parametric surfaces from the examples in [15] where the matrix representation is utilized. All examples shown are run in MAPLE, with an Intel(R) Core(TM) i5-13600K @ 3.50GHz Windows PC.

5.1. Examples

This section lists the following six examples. It shows that the SRS-BFS method works well on all of these examples, e.g., it avoids branch jumping problem and computes the branch points robustly, both with and without floating-point arithmetic.

Example 5.1(Two Separate Branches). Consider the intersection of a cylindrical surface patch $P_1(x,y,z) = x^2 + (y-1)^2 - 4 = 0$ and a rational Bézier surface $\mathbf{Q}_1(u,v) = (\frac{x(u,v)}{w(u,v)}, \frac{y(u,v)}{w(u,v)}, \frac{z(u,v)}{w(u,v)})$,$u \in [0,1]$, $v \in [0,1]$, where

$$\begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \\ w(u,v) \end{bmatrix} = \begin{bmatrix} 2u^2 - 2v^2 - 2u + 2v + 1 \\ -1 + v^2(4u^2 - 4u + 2) + 2v \\ 2u(2v^2 - 2v + 1)^2 \\ (2u^2 - 2u + 1)(2v^2 - 2v + 1) \end{bmatrix} \tag{12}$$

The intersection can be categorized into two separate branches, which is shown in Fig. 6(a). In this case, there is no breakpoint pair, and the tracing method works well.

Example 5.2(Two Branches that Intersect at a Common Tangent Point). Consider the intersection of a sphere $P_2(x,y,z) = 4x^2 + (2y-1)^2 + 4(z-1)^2 - 16 = 0$ and the surface $\mathbf{Q}_1(u,v)$. The intersection of the two surfaces is two tangent curves, which have a common tangent point, as shown in Fig. 6(b). There is a singular point on the intersection curve which is of parameter $(u,v) = (0.5, 0.5)$. The resulting intersection curves from the SRS-BFS method and the intersection in the parameter domain are shown in Fig. 6(b) and Fig. 7(b), respectively.

Example 5.3(Two Branches that Intersect Transversely at a Point). Consider the intersection of a cylindrical surface patch $P_3(x,y,z) = 4x^2 + (2y-1)^2 - 16 = 0$ and surface $\mathbf{Q}_1(u,v)$. The intersection of the two surfaces is two intersecting curves, which have a singular point of parameter $(u,v) = (0.5, 0.5)$, as shown in Fig. 6(c). The resulting intersection curves from the SRS-BFS method and the intersection in the parameter domain are shown in Fig. 6(c) and Fig. 7(c).

Example 5.4(Six Separate Branches Including a Few Loops). Consider the intersection of a cylindrical surface patch $P_4(x,y,z) = y^2 + z^2 - 100 = 0$ and rational Bézier surface $\mathbf{Q}_2(u,v) = (\frac{x_2(u,v)}{w_2(u,v)}, \frac{y_2(u,v)}{w_2(u,v)}, \frac{z_2(u,v)}{w_2(u,v)})$,$u \in [-5,9]$,$v \in [-4.5, 4.5]$, where

$$\begin{bmatrix} x_2(u,v) \\ y_2(u,v) \\ z_2(u,v) \\ w_2(u,v) \end{bmatrix} = \begin{bmatrix} u \\ v \\ 7 + 3\sin(u)\cos(v) \\ 1 \end{bmatrix} \tag{13}$$

The intersection of the two surfaces consists of seven loops including three minimal loops, see also Fig. 6(d). In this case, one has

$$F(u,v) = v^2 + (7 + 3\sin(u)\cos(v))^2 - 100, \tag{14}$$
$$F_u(u,v) = 6(7 + 3\sin(u)\cos(v))\cos(u)\cos(v), \tag{15}$$
$$F_v(u,v) = 2v - 6(7 + 3\sin(u)\cos(v))\sin(u)\sin(v). \tag{16}$$

The solutions of Eq. (14)$\sim$ Eq.(16) can be verified to be

$$(u,v) = (\pi/2 + 2k\pi, 0), \quad k = -1, 0, 1,$$

within $[-5,9] \times [-4.5, 4.5]$. The resulting intersection curves from the SRS-BFS method and the intersection in the parameter domain are shown in Fig. 6(d) and Fig. 7(d), respectively.

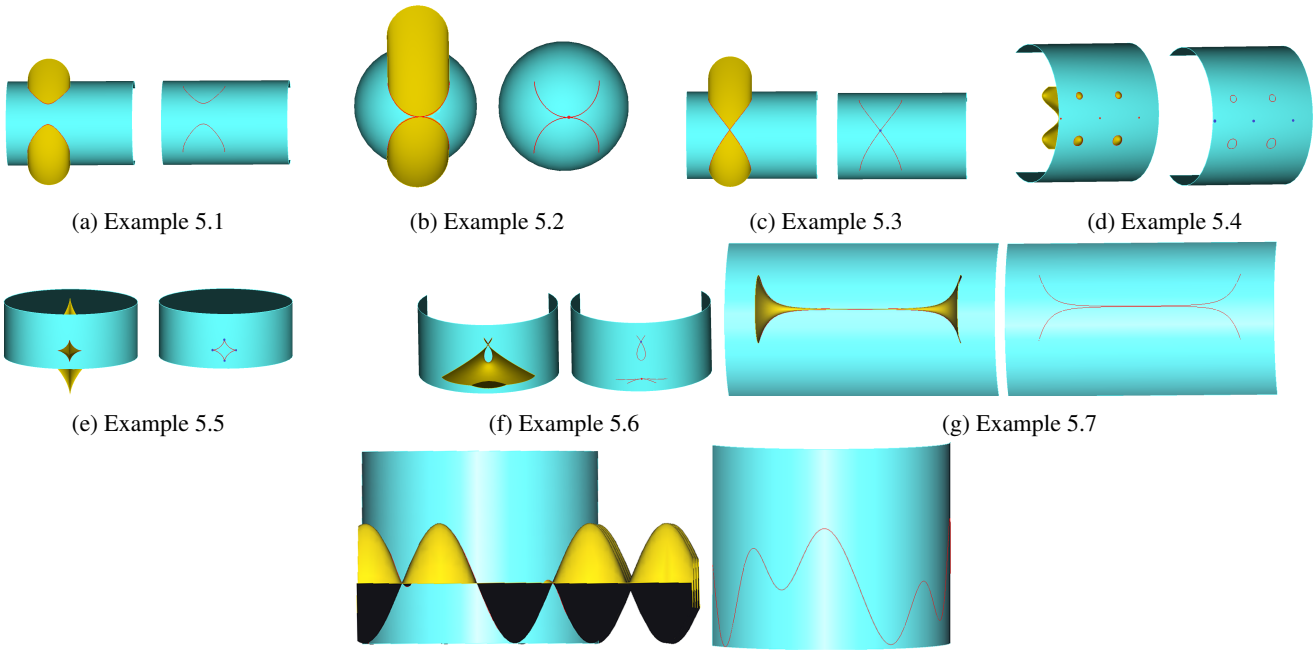
Example 5.5(Intersection Curve with Several Cusps). Consider the intersection of a cylindrical surface $P_5(x,y,z) = z^2 + (x+7)^2 - 121 = 0$ and Octahedron surface $\mathbf{Q}_3(u,v) = (\frac{x_3(u,v)}{w_3(u,v)}, \frac{y_3(u,v)}{w_3(u,v)}, \frac{z_3(u,v)}{w_3(u,v)})$, $u,v \in [-1,1]$, where

$$\begin{bmatrix} x_3(u,v) \\ y_3(u,v) \\ z_3(u,v) \\ w_3(u,v) \end{bmatrix} = \begin{bmatrix} 8(v^2-1)^3(u^2-1)^3 \\ -64v^3(u^2-1)^3 \\ 64u^3(v^2+1)^3 \\ (v^2+1)^3(u^2+1)^3 \end{bmatrix} \tag{17}$$

and $\mathbf{Q}_3(u,v)$ is a NURBS with a few apex edges. The intersection of the two surfaces is a quadrangle, which has four cusps, as shown in Fig. 7(e). In this case, there are four initial points, two $(\pm\lambda_1, 0)$ satisfy $F(u,v) = F_u(u,v) = 0$ while the other two $(0, \pm\lambda_2)$ satisfy $F(u,v) = F_v(u,v) = 0$, where $\lambda_1 \approx 0.3391361554$ and $\lambda_2 \approx 0.3495907910$. The resulting intersection curves are then traced, as shown in Fig. 6(e).

Example 5.6(Three Branches Including a Self-intersection Point and a Common Tangent Point). Consider the intersection of a cylindrical surface patch

$$P_6(x,y,z) = (y-11)^2 + x^2 - 121 \tag{18}$$

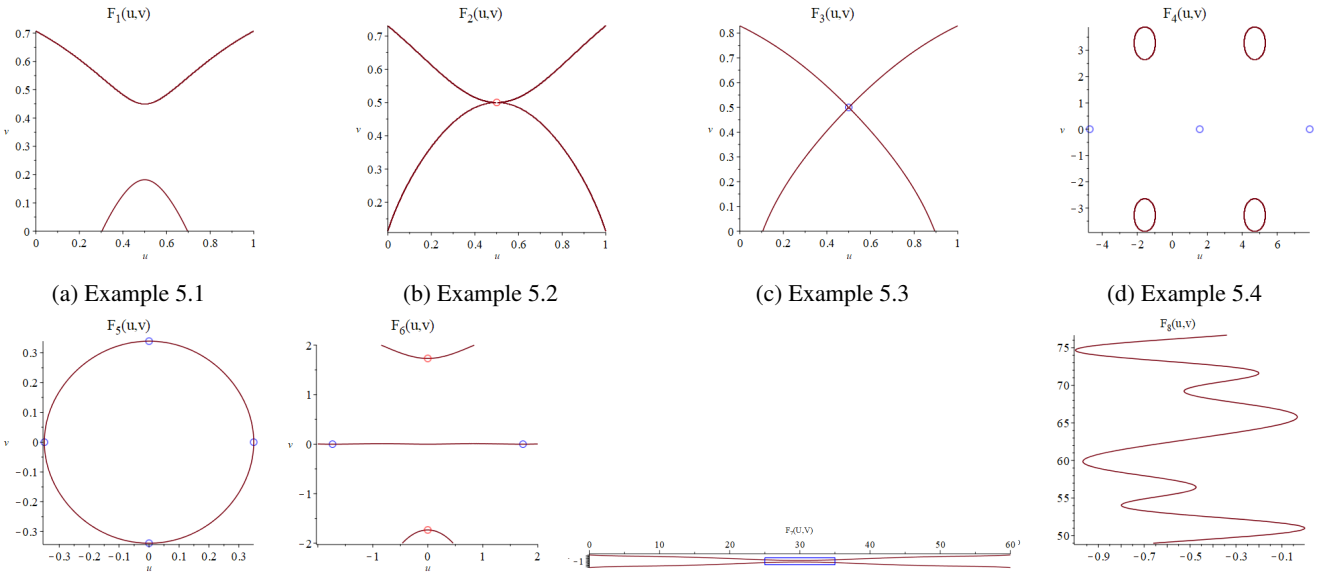


(a) Example 5.1 (b) Example 5.2 (c) Example 5.3 (d) Example 5.4

(e) Example 5.5 (f) Example 5.6 (g) Example 5.7

(h) Example 5.8

Figure 6: Examples of the intersections of two surfaces. For each example, the left column shows two intersecting surfaces and their intersection (in red) given by our algorithm; the right column shows the intersection which is shown on the left column more clearly, and the singular points are marked by blue dots.



(a) Example 5.1 (b) Example 5.2 (c) Example 5.3 (d) Example 5.4

(e) Example 5.5 (f) Example 5.6 (g) Example 5.7 (h) Example 5.8

Figure 7: Examples of intersection of two surfaces in the parameter domain.

and a parametric Surfaces $\mathbf{Q}_4(u,v) = (\frac{x_4(u,v)}{w_4(u,v)}, \frac{y_4(u,v)}{w_4(u,v)}, \frac{z_4(u,v)}{w_4(u,v)})$, $u \in [-2,2]$,$v \in [-2,2]$, where

$$\begin{bmatrix} x_4(u,v) \\ y_4(u,v) \\ z_4(u,v) \\ w_4(u,v) \end{bmatrix} = \begin{bmatrix} -u^3 + 3uv^2 + 3u \\ -v^3 + 3u^2v + v \\ 3u^2 - 3v^2 \\ 3 \end{bmatrix} \tag{19}$$

In this case, $\mathbf{Q}_4(u,v)$ is a NURBS with complicated features such as self-intersection curves, and the corresponding intersection of the two surfaces consists of three branches that have a self-intersected point and a common tangent point, as shown in Fig. 6(f). Though the parameters of the self- intersected point and the tangent point satisfy Eq.(8), there is no self-intersected point in the parameter domain, as shown in Fig. 7(f). So one needs to verify whether or not an initial intersection point is a self-intersected point.

Example 5.7(Two Branches Including Two Separate Intersecting Lines). Consider the intersection of a parametric surfaces patch $P_7(x,y,z) = (x-7)^2 + (y-25)^2 - 676 = 0$ and a parametric surfaces $\mathbf{Q}_5(u,v) = (x_5(u,v), y_5(u,v), z_5(u,v))$, $u,v \in [-2,2]$, where

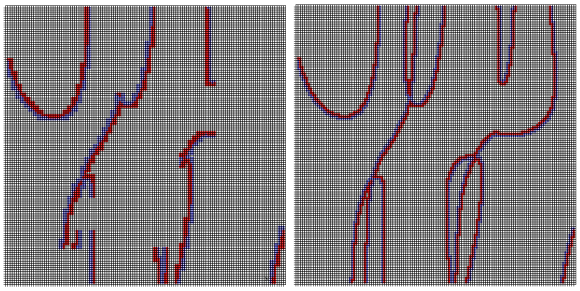
$$\begin{bmatrix} x_5(u,v) \\ y_5(u,v) \\ z_5(u,v) \end{bmatrix} = \begin{bmatrix} 10e^{-5+(u-30)^2/180}\cos(v) \\ 10e^{-5+(u-30)^2/180}\sin(v) \\ u \end{bmatrix} \tag{20}$$

In this case, the corresponding intersection of the two surfaces consists of two branches which have two separate intersection lines, as shown in Fig. 6(g). On the other hand, as shown in Fig. 7(g), the distance between the intersection lines in the parameter domain is within the given smallest size of a grid.

5.2. Comparing with the method M_1 in [15]

The SRS-BFS method (also denoted as M_{new}) applies the novel matrix representation methods in [15, 27], and the difference between M_1 and M_{new} is the marching process. In M_1, if the values $F(u,v)$ at the four corner points of a grid have the same sign, the grid is neglected, see also the grids in blue in Fig. 9; otherwise, the grid is divided into four sub-grids for further checking. The above checking processes iteratively continue until the size of the sub-grids is within the given tolerance. As shown in Fig. 9, as a result, the four corner points of the sub-grids traced in M_1 are denoted by circles in black or white, and M_1 neglects the grids in blue; while all of the traced grids in M_{new} are of the smallest size, including the grids in green and blue. Four more examples, i.e., 5.8–5.11, are added, where 5.9–5.11 are also the examples in [15]. The number of the corner points of the traced grids from M_1 and M_{new} are listed in Table 1, where "/" means that M_1 missed the branches of the intersection. Note that the numbers of M_1 and M_{new} in Example 5.9 are close to each other, the reason is that M_1 neglects part of the grids which contain intersection segments, as shown in Fig. 8. Compared with M_1, M_{new} requires about 51% corner points where the value of $F(u,v)$ is needed to be computed.

Example 5.8(One Branch) Consider the intersection of a surface patch $P_8(x,y,z) = x^2+y^2-100 = 0$ and a parametric surface $\mathbf{Q}_6(u,v) = (x_6(u,v), y_6(u,v), z_6(u,v))$,



(a) M_1 (b) M_{new}

Figure 8: The resulting marching grids from (a) M_1; and (b) M_{new}.

Table 1: The number of corner points from M_1 and M_{new}

Exam	5.1	5.2	5.3	5.4	5.5	5.6
M_1	1151	/	1696	/	/	1028
M_{new}	588	807	879	464	352	501
Exam	5.7	5.8	5.9	5.10	5.11	M_{new}/M_1
M_1	2038	782	1996	785	782	51.08%
M_{new}	1054	398	1824	410	398	

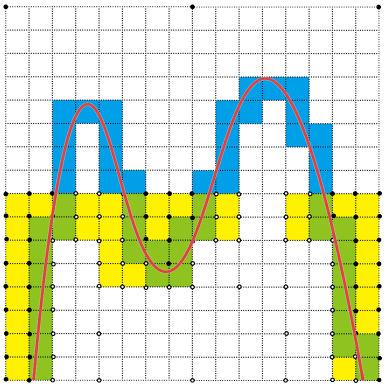


Figure 9: Comparison results between M_1 in [15] and M_{new} (partial intersection of Example 5.8): (a) grids from M_1: the four corner points are denoted as circles in black or white; and (b) grids from M_{new}: the grids in green or blue. M_1 missed the branch segments contained in blue sub-grids.

$u,v \in [-2,2]$, where

$$\begin{bmatrix} x_6(u,v) \\ y_6(u,v) \\ z_6(u,v) \end{bmatrix} = \begin{bmatrix} 2u \\ v \\ 5\sin(u)\cos(v) \end{bmatrix} \tag{21}$$

Example 5.9 (Example 4.1 in [15]) Given two parametric surfaces $\mathbf{P}_{7,1}(s,t)$ is within$(s,t)in[-1,0]\times[-1,0]$ and

$\mathbf{Q}_{7,2}(u,v)$ is within$(s,t) in [-1,1]\times[-1,1]$.

$$\begin{bmatrix} x_{7,1}(s,t) \\ y_{7,1}(s,t) \\ z_{7,1}(s,t) \\ w_{7,1}(s,t) \end{bmatrix} = \begin{bmatrix} -(3t+1)s^2-(3t+3)s+t+9 \\ -(5t+5)s^2+(4-3t)s+2t+7 \\ (s-1)(t+1)(4s+3) \\ (2-2t)s^2+(5-4t)s+4t-5 \end{bmatrix} \tag{22}$$

$$\begin{cases} x_{7,2}(u,v) = & -4v(u^4v^2-14u^2v^2-12u^2+v^2) \\ & \cdot(u^2-1), \\ y_{7,2}(u,v) = & 8uv(3u^4v^2+3u^4-10u^2v^2 \\ & -6u^2+3v^2+3), \\ z_{7,2}(u,v) = & 3(u^4-6u^2+1)(2v^2+1)(u^2+1), \\ w_{7,2}(u,v) = & (u^2+1)^3. \end{cases} \tag{23}$$

Example 5.10 (Example 4.2 in [15]) $\mathbf{P}_{8,1}(s,t)$ is within$(s,t) in [0,1]\times[0,1]$ and $\mathbf{Q}_{8,2}(u,v)$ is within$(s,t) in [-\pi,\pi]\times[-\pi,\pi]$

$$\begin{bmatrix} x_{8,1}(s,t) \\ y_{8,1}(s,t) \\ z_{8,1}(s,t) \\ w_{8,1}(s,t) \end{bmatrix} = \begin{bmatrix} (2s^2+4)(-t^2+1) \\ 2(2s^2+4)t \\ 2s(t^2+1) \\ (s^2+1)(t^2+1) \end{bmatrix} \tag{24}$$

$$\begin{bmatrix} x_{8,2}(u,v) \\ y_{8,2}(u,v) \\ z_{8,2}(u,v) \\ w_{8,2}(u,v) \end{bmatrix} = \begin{bmatrix} 8u^2-8v^2-8u+8v+4 \\ (16u^2-16u+8)v^2+8v-4 \\ 8u^2(2v^2-2v+1) \\ 4(2u^2-2u+1)(2v^2-2v+1) \end{bmatrix} \tag{25}$$

Example 5.11 (Example 6.1 in [15]) $\mathbf{P}_{9,1}(s,t)$ is within$(s,t) in [0,1]\times[0,1]$ and $\mathbf{Q}_{9,2}(u,v)$ is within$(s,t) in [0,1]\times[0,1]$

$$\begin{bmatrix} x_{9,1}(s,t) \\ y_{9,1}(s,t) \\ z_{9,1}(s,t) \\ w_{9,1}(s,t) \end{bmatrix} = \begin{bmatrix} -2(4s^2-4s+3)t(t-1) \\ (4s^2-4s+3)(2t-1) \\ (2s-1)(2t^2-2t+1) \\ (2s^2-2s+1)(2t^2-2t+1) \end{bmatrix} \tag{26}$$

$$\begin{cases} x_{9,2}(u,v) = & (2v^2-2v+3)u^2-v^2+v \\ & +\frac{3}{2}+(-2v^2+2v-3)u, \\ y_{9,2}(u,v) = & 2(u^2-u+1)(2v-1), \\ z_{9,2}(u,v) = & 2u^2(2v^2-2v+1), \\ w_{9,2}(u,v) = & (2u^2-2u+1)(2v^2-2v+1). \end{cases} \tag{27}$$

Furthermore, we have tested ten more examples (Example 7.1–Example 7.10) in [15], where Example 7.i contains two parametric surfaces, $i = 1, 2, \cdots, 10$. The computation time has been listed in Table 2. As shown in Table 2, M_{new} achieves the best efficiency among M_1, M_2 and M_{new}, where M_2 denotes the Dixon-representation based method mentioned in [27].

Table 2: Computational time of M_1, M_2 and M_{new}.

Case in [15]	M_1	M_2	M_{new}	M_{new}/M_2
Example 7.1	1.473	0.520	0.307	0.59
Example 7.2	1.220	0.506	0.343	0.68
Example 7.3	1.608	0.863	0.561	0.65
Example 7.4	6.011	1.369	0.931	0.68
Example 7.5	62.396	5.551	3.608	0.65
Example 7.6	24.926	3.794	2.581	0.68
Example 7.7	29.748	2.589	1.760	0.68
Example 7.8	77.769	7.560	4.460	0.59
Example 7.9	139.854	18.809	11.097	0.59
Example 7.10	143.069	20.655	12.186	0.59

5.3. Comparing with the numerical methods for computing the branch point

Let M_2 be the method in [27], and M_3 denotes the method of the Maple software without good initial values, while M_{new} denotes the SRS-BFS method which can achieve good initial values. M_2 applies exact computations for computing the critical points, including left and right helping points, whose precision is dependent on the tolerance used in the exact computation. The smaller tolerance, the more computational time. Three examples that contain branch points, i.e., 5.2, 5.3 and 5.9, are used for testing M_3 and M_{new}. Table 3 shows the comparison results. M_3 fails to compute two of the three branch points for lacking good initial values, while M_{new} can compute all of the three branch points.

Table 3: Branch point computation from M_3 and M_{new}

Exam	M_3	M_{new}
5.2	($u_1^\star$ - 1.142, $v_1^\star$ - 1.781)	($u_1^\star$ + 1e-10, $v_1^\star$ +1e-14)
5.3	($u_2^\star$ + 1e-16, $v_2^\star$ - 1e-13)	($u_2^\star$ + 1e-16, $v_2^\star$ - 1e-13)
5.9	($u_3^\star$ + 0.488, $v_3^\star$ + 0.654)	($u_3^\star$ + 1e-12, $v_3^\star$ + 1e-13)

$(u_1^\star, v_1^\star) = (\frac{1}{2}, \frac{1}{2})$,$(u_2^\star, v_2^\star) = (\frac{1}{2}, \frac{1}{2})$, $(u_3^\star, v_3^\star)$ = (-0.488, -0.170)

5.4. Comparisons on the topology correctness

As shown in Fig. 8 and Fig. 9, M_1 in [15] may lack some segment of intersection curves. M_2 in [27] works well in most cases, and it is suitable for a smaller tolerance 1e-6, while the used tolerance in M_1 is 1e-3. Moreover, by using the exact computation, M_2 achieves good stability for computing the critical points. Fig. 6(g) shows such a case, there are two branches, part of them (in a framed region) are very close to each other whose distance is smaller than the given tolerance; however, there are no critical points in the framed region. The proper topology consists of two segments $\overset{\frown}{AB}$ and $\overset{\frown}{CD}$. Without exceptional handling in M_2, part of the segment in a framed region may missed, that is to say, the tracing process may stop at the point denoted by a circle in solid green, by merging the branches, one finally

obtains the resulting topology consisting of two segments $\overset{\frown}{AC}$ and $\overset{\frown}{BD}$, which is shown Fig. 10. As shown in Fig. 6(g), the SRS-BFS method can ensure achieving the multiple intersection lines that are adjacent even within the given tolerance.

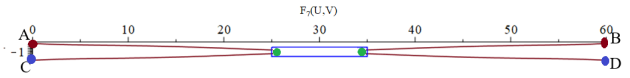


Figure 10: Illustration of possible topology error in M_2.

6. More discussions

By applying the Dixon matrix technique in [15, 27], M_{new} obtains a good computational efficiency in the tracing process. In principle, combining the Dixon matrix technique with the lattice method, M_{new} can rapidly achieve starting points for the tracing process, which is much more efficient than prevailing methods. Then, combining the SRS-BFS method with the ECT technique, M_{new} can trace a branch until a grid may contain two or more branches. In principle, the SRS-BFS method can detect the proper topology in a grid, even if there are two branches whose distance is within a given tolerance.

However, in the case that there is a small loop or isolated intersection point, the lattice method in M_{new} may miss the loop or the isolated point, while the exact computation method can be applied to overcome the above problem, which ensures the computational stability for solving the critical points, as done in M_2.

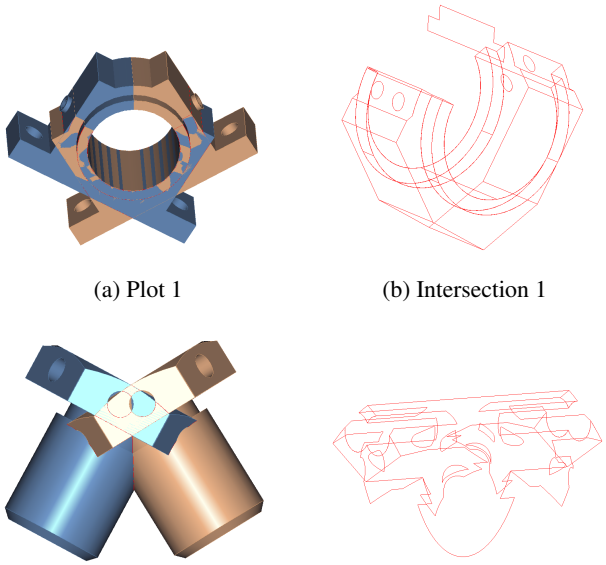
Next, we discuss the advantages and limitations of our algorithm from the aspects of robustness, efficiency, and numerical stability.

Firstly, our ECT method (also M_{new}) admits all of the advantages of M_1 on robustness and numerical stability, which have been discussed in [15]. Noting that the number of traced grids in the ECT method is about half of that of M_1, as shown in Table 2, the computational time of M_{new} is about 55%–70% of M_1.

Secondly, M_1, M_2, and our ECT method may have a leak such that it missed a branch segment that has two intersection points on the edge of the smallest sub-grid, as shown in Fig. 8(a) and Fig.7g. The M_{new} method uses the SRS method for accurately tracing the intersection point on each edge of the edge chain, which can detect two adjacent branches which are close to each other, even within the tolerance in the parametric domain.

Thirdly, the M_{new} provides the BFS method to achieve good initial values for computing a branch point or singular point. Numerical examples show good stability of the M_{new}.

Finally, M_1, M_2 and M_{new} need to deal with the coincidence case or the case that two surfaces are almost coincident with each other, as partly shown in Fig. 11, which is the common limitation of the tracing-based methods.



(a) Plot 1 (b) Intersection 1

(c) Plot 2 (d) Intersection 2

Figure 11: Plots and intersections of two part pairs.

7. Conclusions

This paper presents an efficient and hybrid method (denoted as the SRS-BFS method) for computing the intersection of two parametric surfaces which can also work well for computing the intersection problem between two algebraic surfaces, or between an algebraic surface and a parametric one. Firstly, the SRS-BFS method provides an ECT method combined with the matrix representation in [15], which admits the advantages of the matrix representation, but reduces the number of traced sub-grids by 40% –50%, comparing with the method M_1 in [15]. The SRS-BFS method can achieve much better efficiency than prevailing methods. Secondly, it also fixes the leak of branch missing (shown in Fig.9(a)) or branch jumping (shown in Fig.7g) by using the SRS technique. Finally, it provides the BFS method to achieve good initial values for computing the branch point or singular point, which leads to better numerical stability.

There is still ample room for further improvement. Firstly, efficiently verifying whether an edge contains two or four intersection points could significantly enhance our algorithm. Secondly, finding a robust solution for all the initial tracing points remains a challenge for future work.

Acknowledgement

This research work was partially supported by the National Science Foundation of China (61972120) and the

Haihe Lab of ITAI Project(XCHK20210102).

Declaration of competing interest

The authors declare that they have no competing interests.

References

[1] N. H. Abdel-All, S. A.-N. Badr, M. Soliman, and S. A. Hassan. Intersection curves of hypersurfaces in r4. *Computer Aided Geometric Design*, 29(2):99–108, 2012. 1

[2] A. G. Akritas. Sylvester's forgotten form of the resultant. *The Fibonacci Quarterly*, 31(4):325–332, 1993. 3

[3] N. M. Aziz, R. Bata, and S. Bhat. Bezier surface/surface intersection. *IEEE computer graphics and applications*, 10(1):50–58, 1990. 1

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

[5] R. E. Barnhill and S. N. Kersey. A marching method for parametric surface/surface intersection. *Computer aided geometric design*, 7(1-4):257–280, 1990. 1

[6] M. Bartoň. Solving polynomial systems using no-root elimination blending schemes. *Computer-Aided Design*, 43(12):1870–1878, 2011. 4

[7] E. Boender. A survey of intersection algorithms for curved surfaces. *Computers & graphics*, 15(1):109–115, 1991. 1

[8] X.-D. Chen, W. Ma, and C. Deng. Conditions for the coincidence of two quartic bézier curves. *Applied Mathematics and Computation*, 225(1):731–736, 2013. 2

[9] X.-D. Chen, J. Shi, and W. Ma. A fast and robust method for computing real roots of nonlinear equations. *Applied Mathematics Letters*, 68(1):27–32, 2017. 3

[10] X.-D. Chen, C. Yang, and W. Ma. Coincidence condition of two bézier curves of an arbitrary degree. *Computers & Graphics*, 54(1):121–126, 2016. 2, 3

[11] J.-S. Cheng, B. Zhang, Y. Xiao, and M. Li. Topology driven approximation to rational surface-surface intersection via interval algebraic topology analysis. *ACM Transactions on Graphics (TOG)*, 42(4):1–16, 2023. 1

[12] J. Gao, F. Sarfraz, M. Irshad, and J.-B. Liu. Optimal intersection curves for surfaces. *Journal of Mathematics*, 2021(1):9945984, 2021. 1

[13] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM computing surveys (CSUR)*, 23(1):5–48, 1991. 1

[14] C. M. Hoffmann. *Geometric and solid modeling: an introduction*. Morgan Kaufmann Publishers Inc., 1989. 1

[15] X. Jia, K. Li, and J. Cheng. Computing the intersection of two rational surfaces using matrix representations. *Computer-Aided Design*, 150(1):103303, 2022. 1, 2, 3, 6, 8, 9, 10

[16] X. Jia, X. Shi, and F. Chen. Survey on the theory and applications of μ-bases for rational curves and surfaces. *Journal of Computational and Applied Mathematics*, 329(1):2–23, 2018. 1

[17] K. H. Ko, T. Maekawa, and N. M. Patrikalakis. An algorithm for optimal free-form object matching. *Computer-Aided Design*, 35(10):913–923, 2003. 1

[18] S. Krishnan and D. Manocha. An efficient surface intersection algorithm based on lower-dimensional formulation. *ACM Transactions on Graphics (TOG)*, 16(1):74–106, 1997. 1, 4

[19] H. Lin, Y. Qin, H. Liao, and Y. Xiong. Affine arithmetic-based b-spline surface intersection with gpu acceleration. *IEEE transactions on visualization and computer graphics*, 20(2):172–181, 2013. 2

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

[21] J. R. Rossignac and A. A. Requicha. Piecewise-circular curves for geometric modeling. *IBM Journal of Research and Development*, 31(3):296–313, 1987. 1

[22] R. F. Sarraga. Algebraic methods for intersections of quadric surfaces in gmsolid. *Computer Vision, Graphics, and Image Processing*, 22(2):222–238, 1983. 1

[23] Thomas, G., Berry, , , Richard, R., and Patterson. The uniqueness of bézier control points. *Computer Aided Geometric Design*, 14(9):877–879, 1997. 2, 3

[24] W.-K. Wang, H. Zhang, X.-M. Liu, and J.-C. Paul. Conditions for coincidence of two cubic bézier curves. *Journal of computational and applied mathematics*, 235(17):5198–5202, 2011. 2

[25] Z.-F. Wang, X.-D. Chen, and J.-H. Yong. New proper reparameterization of plane rational bézier curves. *Journal of Computer Science and Technology*, 39(5):1193–1206, 2024. 3

[26] C. E. Wee and R. N. Goldman. Elimination and resultants. 2. multivariate resultants. *IEEE Computer Graphics and Applications*, 15(2):60–69, 1995. 3

[27] J. Yang, X. Jia, and D.-M. Yan. Topology guaranteed b-spline surface/surface intersection. *ACM Transactions on Graphics (TOG)*, 42(6):1–16, 2023. 1, 3, 8, 9, 10