J. Indian Inst. Sci., July–Aug. 2005, **85**, 161–182

A tracing method for parametric Bézier triangular surface/ plane intersection over triangular domain

R. SHARMA[1]* AND O. P. SHA[2]
[1]Design Laboratory, Department of Ocean Engineering and Naval Architecture, Indian Institute of Technology, Kharagpur 721 302, WB, India.
Fax no.: [1]*+091-03222-282700, [2]+091-03222-277190; Phone no.: [1]*+091-03222-281601, [2]+091-03222-283788.
email: [1]*rajivatri@yahoo.com; [2]ops@naval.iitkgp.ernet.in

Received on October 3, 2003; Revised on October 3, 2004.

Abstract

Surface/plane intersection problem is a special case of surface/surface intersection, and is an active area of research across many disciplines in computer-aided geometric design. This paper presents, in general the setting of derivational continuities, (i.e. C^0, C^1, and C^2), a surface/plane intersection algorithm for parametric Bézier triangular surface over triangular domain. The present algorithm is a combinatorial algorithm that is based on a *tracing method,* in which the intersection curves are traced out in the direction of tangent vectors at intersection points. The intersection curves are represented by their piecewise polynomial approximations, given by ordered pair of surface domain points. The aim is to obtain the smallest number of ordered points that correctly represent the topology of the true intersection curves. Since the points on the intersection curves are obtained in an ordered manner, no sorting is essential. The refinement of the intersection curve may include fitting an interpolatory cubic spline, or adding and deleting points. The method has been used to compute different planar sections (i.e. intersection curves with X, Y, and Z planes) for surfaces having different derivational continuities (i.e. C^0, C^1, and C^2).

Keywords: Bézier rational triangular patches, rational-triangular parametric surfaces, surface/plane intersection, surface directional tangent vectors, topological subdivision.

1. Introduction

In general context, surface/plane intersection problem is a special case of surface/surface intersection. This has various applications in CAD/CAM systems, computer graphics, surface information systems, geographic information systems, etc. The computation of curve is required for different objectives in many industries, i.e. surface slicing for laminated object manufacturing and stereo lithography, NC tool path generation, intersection of hunting planes to evaluate machining information, and contour lines for geographical and information models. Though it is computationally expensive to treat surface/plane intersection problem as a special case of surface/surface intersection rather than purely a planar cut problem, theoretically it is better because some of the ideas of the algorithm may be utilized to compute surface/surface intersection. The surface/surface intersection problem is considered to be one of the most basic but difficult problems in computer-aided geometric design (CAGD). However, it has been dealt extensively to obtain exact, robust, and efficient solutions in general setting for surfaces.

*Author for correspondence.

Triangular surfaces are important because in areas where the geometry is not similar to rectangular domain, the rectangular surface patch will collapse into a triangular patch. In such a case, one boundary edge may collapse into a boundary vertex of the patch, e.g. Wolter and Tuohy [1]. This may give rise to geometric dissimilarities, (e.g. shape parameters, Gaussian curvature distribution, cross-boundary continuities, etc.) and topological inconsistency. Furthermore, since a triangular patch (i.e. defined as a closed polygon) is a basic 2D figure in algebraic topology, any fairly irregular complex geometry can be efficiently modeled/designed/regenerated with a triangular patch, e.g. Farin [2]. However, triangular surfaces over triangular domain remain relatively unexplored as compared to rectangular surfaces over rectangular domain.

In the present work, we are interested in surface/plane intersection algorithm for parametric Bézier triangular surface over triangular domain. Since, in full generality, the curve tracing in surface/surface intersection is a difficult problem, no infallible method is available either in general context or in general setting of derivational continuities. Instead of working with one special type of derivationally continuous surface (i.e. C^0, or C^1, or C^2 continuous) or concentrating on a specialized area in the general problem of surface/plane intersection, in this work we take a combinatorial approach that utilizes different ideas, and concentrate on general settings of derivational continuities (i.e. C^0, C^1, and C^2 continuity). Our aim is to present a basic surface/plane intersection algorithm for parametric Bézier triangular surface over triangular domain without any *add-on* feature.

We consider the non-self-intersecting surface/plane intersection problem as a special case of non self-intersecting surface/surface intersection. Basically, we compute surface/surface intersection, while defining the plane itself as a fixed domain surface. The planar surface is designed/modeled/defined over the plane as a fixed domain surface, where the domain is user-specified. This will ensure that within the bounds of the domain of the two surfaces, one surface and another planar surface for the plane, an intersection does exist. The algorithm used in this work is based on a *tracing method*, in which the intersection curves are traced out in the direction of tangent vectors at intersection points. The intersection curves are represented by their piecewise polynomial approximations, given by ordered pair of surface domain points. The aim is to obtain the smallest number (i.e. in this work the number is between 2 and 23) of ordered points that correctly represent the topology of the true intersection curves. The algorithm presented is simple, but allows the correct computation of topology of the true intersection curves. This allows any complex and complicated feature (i.e. the computation of intrinsic properties such as unit tangent vector, curvature vector, binormal vector, curvature, torsion, and higher-order transversal and tangential derivatives), required to further analyze the intersection curve to be suitably added. Since the points on the intersection curves are obtained in an ordered manner, no sorting is essential. We consider the computation of multiple intersection curves at one plane with multiple solutions of starting point and for each starting point the intersection curve is traced separately. However, we do not include the computation of intersection curve with multiple branches (i.e. branch points, and bifurcation points), or lower and higher-order singularities in our surface/plane intersection algorithm. The surface/plane intersection algorithm presented in this work is a structured algorithm that allows better insight into the problem, and stepwise computations. This shall allow easy addition of any suitable *add-on* feature for future research.

This paper is organized as follows. In Section 1, a brief description of the work is presented. The current state of the art has been reviewed in Section 2. The theoretical background and implementation of the surface/plane intersection is presented in Section 3. Numerical examples are discussed in Section 4. Section 5 concludes by identifying some future applications and scope of research.

2. Brief review of the state of the art

Normally, the surface/surface intersection approaches are classified into four groups: analytical, lattice evaluation, subdivision, and tracing (i.e. including numerical methods)-based methods. Of these four approaches, subdivision- and tracing-based methods are the most widely used because of their generality. In this section we concentrate on the review of the methods based upon subdivision and tracing.

Bajaj *et al*. [3] discuss a purely numerical approach that used a third-order Taylor approximation by taking steps of reliable lengths and the intersection curves are computed using the Newton iteration method. They also presented implemental techniques to address the problem of *singularity*, but, because of the implicit nature of the method it is suitable to only CAD systems in engineering where implicit surface definitions are used. Montaudouin *et al*. [4] presented a method in limited context for approximation of intersections for algebraic curves and surfaces. They used power series in approximation. Asteasu [5], and Garrity and Warren [6] also worked on similar lines.

Houghton *et al*. [7] discuss a subdivision-based method. It is dependent on C^0 linear subdivided surface approximations, which is a limitation in general context. Filip *et al.* [8] also worked on similar lines. Bürger and Schaback [9] presented a parallel algorithm using the divide-and-conquer (i.e. subdivision based) method with the analysis of its complexity. The method uses progressive refinement in interval search (e.g. Gregory [10]). Barnhill *et al.* [11] discuss a general well-structured surface/surface intersection algorithm for C^1 continuous surfaces in parametric rectangular domain. The algorithm computes the intersection curve stepwise, and hence allows a better insight into the problem. This allows any '*add-on*' feature to be integrated with the algorithm if the geometric constraints so demand. Later, Barnhill and Kersey [12] generalized the algorithm further by excluding explicit surface definitions and including more geometric and topological definitions (i.e. evaluated surface positions and tangents). The method uses oriented bounding boxes (e.g. common with [7]) over the surface domain for selective subdivision and to get starting points. In similar line, Cugini *et al*. [13] present the concept of shrinking bounding boxes over the surface domain, and use conflicting bounding boxes to compute the starting points. Though the concept of bounding boxes is elegant, simple, and effective in many cases, it is pure geometric, and hence may miss the intersection points.

Abdel-Malek and Yeh [14] present a tracing method to compute the intersection curve for surface/surface intersection. The authors deal with solid surfaces (i.e. closed surface defined over a solid) and represent a solid by a number of parametric surfaces (i.e. ${}^{A}S_{S_i}(u,v): U \subset R^2 \to R^3$, ${}^{A}S_{S_i} = \{{}^{A}S_{S_1}, {}^{A}S_{S_2}, {}^{A}S_{S_3}, \ldots\}$, which is a differentiable map ${}^{A}S_{S_i}$ from an open set $U \subset R^2$ into R^3). The authors define a solid surface over R^3 scattered data, onto different open surfaces over R^2 parametric plane. The implicit condition in the algorithm presented by the authors is that the vectors $\partial^A S_{S_i}/\partial u$ and $\partial^A S_{S_i}/\partial v$ are linearly independent. The linear independency of these partial derivatives is essential to have a nonzero Jacobian (J_C) of the surface $S_S(u, v)$. The requirement of a nonzero Jacobian (J_C) of the surface $S_S(u, v)$ is because if the Jacobian (J_C) is zero then the computation of the inverse of the Jacobian (J_C) is computationally expensive, and numerically unstable too. This implicit condition holds true only for a class of surfaces, which are higher-order continuous (i.e. G^n/C^n continuous for $n \geq 1$), and do not have a tangent crack (i.e. surface which has C^0 continuity overall or in hybrid setting of derivational continuities). Because of this, the formulation presented is restricted to surfaces in engineering sciences that are nonflat, nonhybrid, and without a crack. Thus, the authors have only discussed the examples of analytical surfaces (i.e. nonflat higher-order continuous surfaces such as cylindrical and spherical surfaces). Additionally, the authors trace the curve with a purely numerical approach (i.e. incorporating extra constraints representing local differential geometrical properties such as tangent vector), and hence solve a system of highly nonlinear equations using a professional software (PITCON$^{\text{TM}}$*). The solution of nonlinear equations generally is a complex, complicated, and computationally expensive process, and hence for robustness specialized software is needed. Furthermore, the authors address the problem of singularity/bifurcation (i.e. the intersection curve having multiple branches), by relying upon the change in the sign of Jacobian (J_C) of the surface, and the assumption is used in switching from one intersection curve branch to another of the same curve. This formulation holds true only for the cases when Jacobian (J_C) of the surface is nonzero and nonconstant. In engineering sciences, this is valid only for a class of surfaces that have a fair distribution of curvatures (i.e. cylindrical surfaces). For the objective of switching the authors use low rank deficiency in the Jacobian (J_C) (i.e. if only the first-order Taylor expansion of the surface is zero and not the higher-order ones, then it is low rank). Numerically, rank deficiency of the Jacobian (J_C) can only be tackled efficiently with analytical surfaces, and not with piecewise polynomial surfaces. This is because for analytical functions the degree in the series expansion can be sufficiently large, while for piecewise polynomials the popular degrees are either 3 or 4. As in the present work, piecewise polynomial surfaces are investigated we avoid, in our formulation, rank deficiency in the Jacobian (J_C).

In recent years, research has been concentrated on the '*loop detection*'-based methods, e.g. Cheng [15], Kriezis *et al.* [16], Sederberg *et al.* [17], Sederberg and Meyers [18], and Sinha *et al.* [19]. In those techniques, one seeks the solution to sequential initial value problems. Each initial value problem terminates when the solution encounters one of the possible stopping points. Since it is impossible to estimate the parametric value at which such an encounter might occur, a heuristic approach is taken into the curve tracking/tracing scheme. This procedure in general depends upon prior setting of a few tolerances, a process that is dependent on the geometrical features of the surfaces. Since the choices are not obvious, it is not always easy to handle.

Grandine and Klein [20] approach this problem differently. Instead of working with initial value problem, the authors try to determine, *in advance*, which of the candidate stopping points is the actual one, and set that prior to curve tracking/tracing. Hence, the authors

*Trademark and copyright are with Prof. W. C. Rheinboldt, Department of Mathematics, University of Pittsburg, USA.

work with boundary value problem in place of initial value problem. The approach is restricted heavily to a class of surfaces (i.e. in the author's case tensor product surfaces) to which the computationally robust solution to a nonlinear system of equations exists. Again, the surface has to be C^1 continuous, and if not, it needs to be subdivided into subsurfaces that are C^1 continuous. The method can be extended to piecewise polynomial surfaces defined over triangular domain (i.e. those used in the present work) and irregularly shaped regions, and algebraic surfaces. Though the method computes excellent results in some cases, because of its restrictions and computational expenses, it cannot be applied in general context.

The '*loop detection*' technique ensures that the different branches of the intersection curve will be computed. However, this technique does not ensure the correct topological computation of the intersection curve. Since, in interactive user environment in engineering, the correct topology of the intersection curve is important for interactive design this is a serious restriction. In general context, only the marching, tracing, and curve tracking makes the computation of topological properties of the intersection curve an in-built feature in the procedure. Because of this reason '*loop detection*'-based techniques are considered a suitable add-on, but not a sole methodology in themselves.

In general, numerical algorithms require the computation of a starting point on/or close within specified tolerances to the intersection curve. Müllenheim [21] discusses an iterative procedure to compute a starting point close within specified tolerances to the intersection curve. Dokken [22], and Dokken *et al.* [23] present a surface/surface intersection based on recursive subdivision. Abdel-Malek and Yeh [24] present two iterative numerical algorithms to compute the starting points in surface/surface intersection with their complexity analysis. The first numerical algorithm is an iterative, conjugate gradient-based optimization technique. The second algorithm uses the Moore–Penrose pseudo-inverse of the constraint function to determine the starting point.

The recursive subdivision based on some type of global divide-and-conquer algorithms can continue till the problem reduces to a simple optimization problem within specified tolerances, for example, Dokken *et al.* [23]. Since optimization moves from global to local via subdivision, any error in subdivision at any level will continue and might affect the convergence in the whole solution. Also, because of their global characteristics the subdivision-based algorithms are slow and computationally expensive. The iterative methods consider the intersection problem in local context of the parametric domain. Hence these are fast and computationally efficient, and due to the same reason do not ensure the complete determination of multiple solutions. The Moore–Penrose pseudo-inverse, however, guarantees that if a solution exists, it will be computed within/closest to user-specified tolerance (i.e. initial guess). If multiple solutions exist, then the closest solution to the *pre-specified* initial guess will be found.

The subdivision is important on surfaces that occur normally in engineering sciences (discussed in Section 3.1.2.). The recursive subdivision methods are based on some type of pure geometric primitives. And, with each step of subdivision, the method loses information on the derivational continuities and subsequently the surface's topological features too. It is simple and computationally efficient to continue the subdivision till the problem reduces to a simple optimization problem within specified tolerances. However, this restricts the use of recursive subdivision methods to only a class of surfaces where derivational continuities are not of much importance (i.e. planar, flat, nearly flat surfaces, or surfaces having low Gaussian or principal mean curvatures). Since these types of surfaces constitute only a small type of general surfaces in engineering sciences, this is a serious restriction. The subdivision can be based upon topological features that maintain not only derivational continuities but also convexity and other topological properties. This is desired in engineering sciences, and allows for a wide variety of surfaces to be included in the surface/surface intersection algorithm.

Almost all the methods mentioned above deal with rectangular surfaces defined over rectangular domains. Both the methods (i.e. curve tracing and ‘*loop detection*’ technique) deal with prior setting of a few tolerances, a process that is dependent on the geometrical features of surfaces, and is always difficult to handle in semi-automatic or fully automatic computation process. Because of this, in full generality, the surface/surface intersection problem based on either curve tracing or ‘*loop detection*’ techniques is a difficult problem, and hence no infallible method is available either in general context or in general settings of derivational continuities. Any algorithm involving tolerances can be defeated by sufficiently ill-chosen examples. Thus, the surface/surface intersection problem is user-specified, tricky, and involves sufficient, correct and flexible user inputs. More comprehensive review on iterative and subdivision-based methods can be found in [23, 24] and on surface/surface intersection algorithms in [10–12, 20, 25, 26].

3. The tracing method

In this section, we present the surface/plane intersection algorithm considering it as a special case of general surface/surface intersection problem. The algorithm used in this work is based upon three basic ideas: computation of starting point to initiate the algorithm (e.g. [14 and 24]), topological subdivision of the surface to improve upon the aspect ratios of the discretized triangular elements/patches (e.g. Goldman [27]), and tracing along intersection curve in the direction of *tangent vector* in an ordered and structured manner (e.g. [12]). The structured algorithm includes two processing steps for obtaining intersection approximations: getting starting points on intersection curve, and trace along intersection curve. The compository structure of the algorithm used is the following:

1. Getting start points
   - Subdivide the surface patches.
2. Trace along intersection curve
   - Get step vector.
   - Relax point to an intersection.
   - Relax points on to boundaries if desired.
   - Identify branch points and tangent points if desired.
   - Tolerances.
   - Repeat for all start points.

The implementation program of the algorithm is briefly presented in the next section.

3.1. *Getting start points*

Following [14, 24], let the equation of a surface parametrized in independent parameters *u*, and *v* be

$$S_S(u, v) \tag{1}$$

with constrained parameters

$$u_1 \leq u \leq u_2 \tag{2}$$

$$v_1 \leq v \leq v_2 \tag{3}$$

and let the equation of a plane defined as a fixed domain surface be

$$S_{Pl}(l, m) \tag{4}$$

with constrained parameters

$$l_1 \leq l \leq l_2 \tag{5}$$

$$m_1 \leq m \leq m_2. \tag{6}$$

It may be noted here that $S_{Pl}(l, m)$ is a planar and flat surface and has been defined as a C^0 continuous surface. $S_{Pl}(l, m)$ may be defined as C^1 or C^2 continuous surface, but this will not change the nature of the numerical computations since a fixed domain surface for a plane will remain planar irrespective of the continuities imposed. Therefore, a planar surface as C^1 or C^2 continuous surface will only affect a few entries via the incorporation of a *constant valued* column or row in the system matrices. This will not affect computations *per se*, and only increase the computational space.

For numerical computation, let us reparametrize the above constraints by the introduction of new generalized coordinate c_i, in order that an inequality constraint in the form

$$O_i^{\min} \leq O \leq O_i^{\max} \tag{7}$$

can be parametrized as

$$O_i = \left(\frac{O_i^{\max} + O_i^{\min}}{2}\right) + \left(\frac{O_i^{\max} - O_i^{\min}}{2}\right) * \sin c_i \tag{8}$$

where $\left(\frac{O_i^{\max} + O_i^{\min}}{2}\right)$ is the midpoint and $\left(\frac{O_i^{\max} - O_i^{\min}}{2}\right)$ is the half range of the inequality constraint. The intersection problem is considered as solving a system of seven nonlinear equations, in eight variables, and with the constraint function as

$$y(\mathrm{P}) = \begin{bmatrix} S_S - S_{Pl}(l, m) \\ u - [(u_1 + u_2)/2 + \sin(c_1) * (u_2 - u_1)/2] \\ v - [(v_1 + v_2)/2 + \sin(c_2) * (v_2 - v_1)/2] \\ l - [(l_1 + l_2)/2 + \sin(c_3) * (l_2 - l_1)/2] \\ m - [(m_1 + m_2)/2 + \sin(c_4) * (m_2 - m_1)/2] \end{bmatrix} = 0 \tag{9}$$

where P = [$u\ v\ l\ m\ c_1\ c_2\ c_3\ c_4$] and the inequality constraints of eqns (2), (3), (5), and (6) are parametrized as per eqn (8).

3.1.1. *The Moore-Penrose pseudo-inverse method:* The solution is desired for

$$y_{\mathrm{P}}\Delta \mathrm{P} = -y \tag{10}$$

where y_{P} is defined as the Jacobian of the constraint, i.e. $y_{\mathrm{P}} = \dfrac{\partial y}{\partial \mathrm{P}}$.

There exist two different possibilities,

- If the constraint subJacobian matrix is square, then eqns (9) and (10) comprise the conventional Newton–Raphson iteration method (i.e. with quadratic convergence properties, e.g. Markot and Magedson [28]).
- If the constraint equation (9) has more rows than columns, and the constraint subJacobian y_{P} has more columns than rows, and vice versa, then (10) will have more than one solution. Using the principle of *minimum norm*, (Haug *et al.* [29], and Noble and Daniel [30]); i.e. to find the solution of the minimization problem;

$$\min \frac{1}{2}\Delta \mathrm{P}^T \Delta \mathrm{P} \tag{11}$$

$$y_{\mathrm{P}}\Delta \mathrm{P} = -y. \tag{12}$$

Applying Lagrange multiplier approach (i.e. appending a multiplier vector number of times the equation to be satisfied in the minimization function), we define

$$V = \frac{1}{2}\Delta \mathrm{P}^T \nabla \mathrm{P} - j^T(y_{\mathrm{P}}\Delta z + y). \tag{13}$$

For a solution to eqn (13) to exist, the gradient of the function in eqn (13) should be zero, i.e.

$$\Delta \mathrm{P}^T = j^T . y_{\mathrm{P}}. \tag{14}$$

Transposing both sides in eqn (14),

$$\Delta \mathrm{P} = y_{\mathrm{P}}^T . j\,, \tag{15}$$

and substituting in eqn (12),

$$y_{\mathrm{P}} y_{\mathrm{P}}^T j = -y. \tag{16}$$

If the subJacobian y_{P} has full row or column rank, then the coefficient matrix on the left hand side of eqn (10) is positive, definite, and nonsingular, and therefore

$$j = (y_{\mathrm{P}} y_{\mathrm{P}}^T)^{-1}(-y). \tag{17}$$

Substituting in eqn (15)

$$\Delta \mathrm{P} = y_{\mathrm{P}}^T (y_{\mathrm{P}} y_{\mathrm{P}}^T)^{-1}(-y). \tag{18}$$

The coefficients of the right-hand side in eqn (18) are called Moore–Penrose pseudo-inverse of the subJacobian ([30]). The equation can also be written as

$$\Delta \mathrm{P} = y^{*}_{\mathrm{P}}(-y) \tag{19}$$

where y^{*}_{P} is the Moore–Penrose pseudo-inverse of the Jacobian y_{P}, and is defined as

$$y^{*}_{\mathrm{P}} = y^{T}_{\mathrm{P}}(y_{\mathrm{P}} y^{T}_{\mathrm{P}})^{-1}. \tag{20}$$

For numerical computation, let us define,

$$r \equiv (y_{\mathrm{P}} y^{T}_{\mathrm{P}})^{-1}(-y) \tag{21}$$

which is equivalent to the solution

$$(y_{\mathrm{P}} y^{T}_{\mathrm{P}})r = -y. \tag{22}$$

Solving eqn (22) numerically for r, and substituting in eqn (18),

$$\Delta \mathrm{P} = y^{T}_{\mathrm{P}} r. \tag{23}$$

With the specification of initial P_i, the generalized set of coordinates is computed as

$$\mathrm{P}_{i+1} = \mathrm{P}_i + \Delta \mathrm{P}. \tag{24}$$

The method described above has the same quadratic convergence as the standard Newton–Raphson method (e.g. [21]). And, using this method, the starting point P_0 is computed with smaller number of iterations (i.e. 4–6), but it gives only one starting point that is nearest to the user-specified initial point, P_i.

3.1.2. *Subdivide the surface patches*: In engineering sciences (i.e. aircraft, mechanical and ship sciences), a surface is designed/modeled/generated with a set of discrete input points called control points (i.e. *control net* or *offset points*). Derivational continuities do not ensure that the surface will have a fair distribution of aspect ratios of the discretized elements. Hence, there is no practical control over the aspect ratio of the elements generated in the surface. Furthermore, for surfaces of underwater or surface floating bodies, the flow changes drastically in the normal direction to the surface and little in the tangential longitudinal direction. This forces the discretized elements to have extreme side ratios, e.g. Bertram [31]. Again, in the case of some surfaces the production features of the surface also show drastic change from one end to another and that forces discretized elements to have poor aspect ratios. This means that in some cases of surface generation process, the poor aspect ratio of individual domain cannot be avoided.

The poor aspect ratios of long, narrow, and thin elements/patches often affect the stability, efficiency, and robustness of the solution process. It is not only difficult to compute either numerical or computational objective function within a domain or boundary of a long, narrow, and thin element/patch, but even the simple geometric and algebraic function tends to fail to converge for a long, narrow, and thin element/patch. Therefore, in the present work the subdivision is used selectively, and for the following purposes: The long, narrow, thin and patches with poor aspect ratio are subdivided explicitly before computing surface/plane intersection.

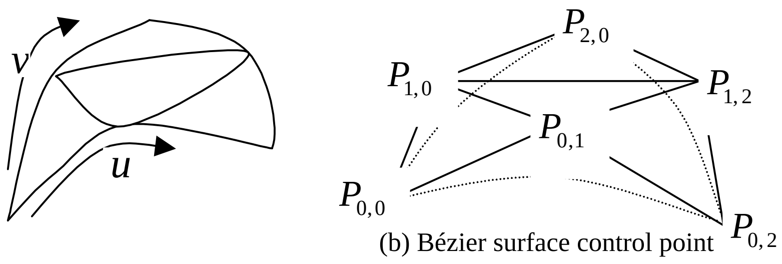


FIG. 1. A simple triangular Bézier surface and corresponding control net.

Following Farin [2, 32], let the surface be parametrized by two parameters u and v and a weight function $B^d_{i,j}(u,v)$ describes the influence of a control point $P_{i,j}$ on the shape of surface, where d is the degree of the surface. The triangular rational Bézier surfaces are defined as follows:

$$P(u,v) = \frac{\sum_{j=0}^{d}\sum_{i=0}^{d-j} P_{i,j} w_{i,j} B^d_{i,j}(u,v)}{\sum_{j=0}^{d}\sum_{i=0}^{d-j} w_{i,j} B^d_{i,j}(u,v)} \tag{25}$$

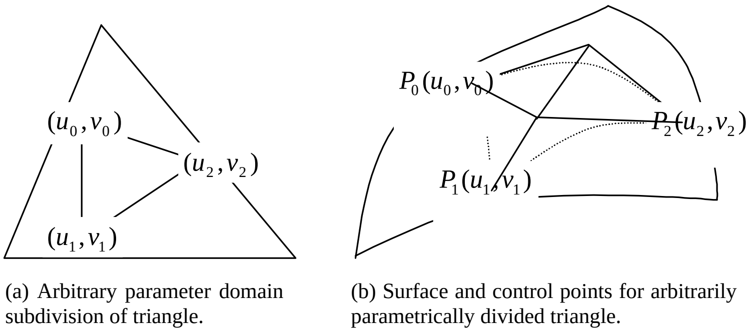
with the corresponding Bézier weight functions:

$$B^d_{i,j}(u,v) = \frac{d!}{i!\,j!\,k!} u^i v^j w^k \text{ with } w = 1 - u - v, \qquad k = d - i - j. \tag{26}$$

The each control point on the surface is assigned a weight $w_{i,j}$ for better control. In case all the weights are constrained to be positive, then the two properties hold: the surface lies completely in the convex hull of its control points, and the surface can be repeatedly subdivided into subsurfaces of the same kind, e.g. [2] and [32]. Figure 1 shows a simple Bézier surface and the corresponding control net.

The triangular surfaces can be subdivided with an extended form of the de Casteljau algorithm, or any of its variant [33]. The triangular control point net is recursively subdivided with trilinear interpolation for a parameter set, and the result being control net for the surfaces and a point on the surface. Furthermore, the geometric and topological properties are computed as for the initial surface. The successive recursion to the resulting subsurfaces, with de Casteljau algorithm leads to long, narrow, and thin triangular patches with poor aspect ratio, often causing numerical and computational problems [34]. Thus, the de Casteljau algorithm is not practical for use in this work.

Goldman [27] presents a general subdivision algorithm, based on arbitrary subdivision that allows explicit control over the aspect ratio. The idea is: each of the three corner points $P_j(u_j, v_j)$, $j = 1, 2, 3$ of an arbitrary triangular subsurface can be computed by evaluating a sequence of d de Casteljau steps $Cd^i_j(u_j, v_j)$, $i = 0, \dots, d$, where d is the degree of the surface. The selection of an arbitrary step from among the three $Cd^i_j(u_j, v_j)$ for each of the d



(u_0, v_0)
(u_2, v_2)
(u_1, v_1)
(a) Arbitrary parameter domain subdivision of triangle.
$P_0(u_0, v_0)$
$P_2(u_2, v_2)$
$P_1(u_1, v_1)$
(b) Surface and control points for arbitrarily parametrically divided triangle.

FIG. 2. Arbitrary parameterization and corresponding subsurface and its control net.

steps of the sequence will compute a control point of the subtriangle. The computation for all possible permutations of this step sequence will give all the control points of the triangular subsurface, and the geometric and topological properties are computed as for the initial surface (Fig. 2).

This process is known as blossoming principle after Ramshaw [35], and was independently developed by de Casteljau [33]. In this work, we have used techniques described in Goldman [27] to arbitrarily subdivide the parametric surface to improve upon the aspect ratios of the discretized triangular elements/patches of the parametric surface $S_S(u, v)$, and correspondingly the geometric and topological properties are computed.

3.2. *Trace along intersection curve*

3.2.1. *Getting step vector*: After getting start point P_0 and selectively subdividing the surface $S_S(u, v)$, the corresponding intersection curve is traced in the direction of tangent vector along intersection curve. For a step direction, it is necessary to determine a step distance at which a new point (i.e. approximate) will be obtained. In general, step length is important because a too small step length decreases the computational efficiency and a too large one may miss the points on the intersection curve. In order to achieve balance in these two conflicting requirements, adaptive step length based on curvature estimates at intersection points is used in this work.

In the case of nonparallel surface tangent planes, step vector direction is computed by the crossproduct of the plane-surface normal. The tracing moves by stepping a distance in the direction (i.e. +ve or –ve) along this vector (Fig. 3). Instead of crossproduct, another calculation that is computationally efficient is that for a given Jacobian matrix (J_C) of the tangent plane on one surface (i.e. S_S), and the plane normal of the other surface (i.e. S_{Pl}) tangent plane N_{Pl}, the intersection vector between these planes is represented by $(J_C\Delta_{(u,v)}S_S)\cdot\mathrm{N}_{Pl} = 0$), where $\Delta_{(u,v)}S_S$ is the domain gradient of the first surface (i.e. S_S) corresponding to the intersection vector. The expansion will give the intersection vector;

$$I_{SV} = S_{Sv} - \frac{S_{Sv}.\mathrm{N}_{Pl}}{S_{Su}.\mathrm{N}_{Pl}}.S_{Su} \tag{27}$$

where S_{Su} and S_{Sv} are the partial derivatives of the surface S_S at the previous point (i.e. P_0).

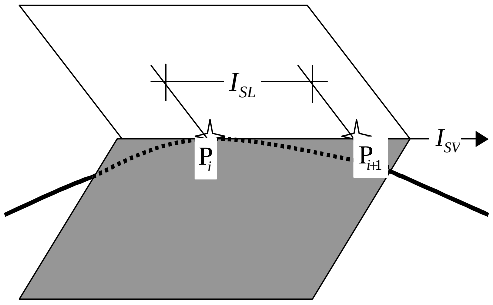


FIG. 3. The stepping vector and stepping length.

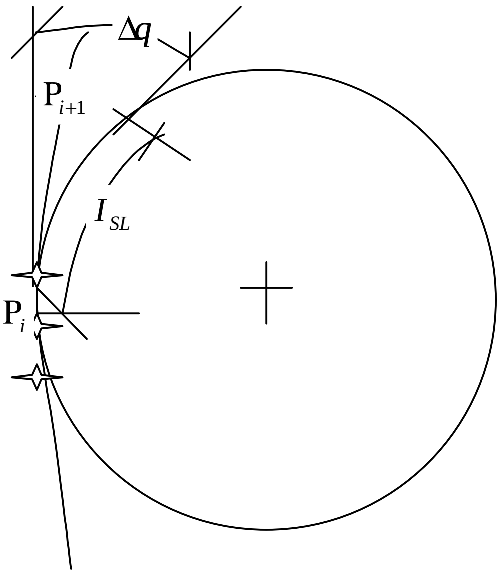


FIG. 4. The approximation of stepping length.

In the case of parallel surface tangent planes (i.e. in the case of surfaces having tangent cracks, or C^0 continuity), the computation of step vector is difficult. One possible approach that has been utilized in the present work is to search coordinate points radially about the starting point. It means that a new point is computed at a small distance e from the start point (i.e. P_0) on the intersection. The search is done radially in each local parametric domain till a new point is computed. The new tangent vector then is computed as the difference between the new and the starting point (i.e. P_0).

The next step is to compute step length. The method used in this work is adaptive and is based upon approximate curvature and an angle tolerance with curve refinement tolerance (CRT) as a minimum bound width for stepping distance, e.g. [11, 12]. As shown in Fig. 4, for the computation of radius of curvature, an osculating circle is approximated at a given point P.

The circle is defined by P, and two neighboring points at fairly small distances from P in each direction along the tangent vector. The neighboring points are then relaxed to the intersection using the methods described in Section 3.2.2. Within the computational limit, these three points describe the true osculating circle. For these three points as computed, the radius of their circumscribed circle that is assumed to be the approximate radius of curvature is, e.g. [26];

$$f = \frac{|Q-P|.|R-P|.\left\||Q-P|-|R-P|\right\|}{2.\left\||Q-P|\times|R-P|\right\|}. \tag{28}$$

In the case of surfaces with tangent plane parallelity, and possibly in some other cases, when the three points are collinear, or if step length I_{SL} > CRT, then CRT is chosen as the step length. The step length (i.e. arc length I_{SL}) is then approximated to $I_{SL} = \mathrm{f}.\Delta q$. The value of angle tolerance (i.e. Δq) is defined by the user.

3.2.2. *Relax point to an intersection*: The refinement (i.e. relaxing approximate points on to intersections) is computed using linear iterative techniques as described in [12]. The initial approximate point is the tracing point with approximate domain and range values. For an approximate intersection point $\mathrm{P} \in \Re^3$, and coordinate pairs (u, v) and (l, m) in the domain of surfaces $S_S(u, v)$ and $S_{Pl}(l, m) \in \Re^3$ respectively, the aim is to relax P on to a true intersection. The point P may not in general be lying on any surface. The one possible method for refining P is to first compute surface 'near point', and then evaluating the linearized surface equations at these points. The process will continue till convergence is achieved, (i.e. $\|S_S(u, v) - S_{Pl}(l, m)\|_2 \langle$ SPT, where $\|\cdot\|$ is the Euclidean vector norm, and SPT is the same point tolerance ([11]).

The surface near points is defined as points, one on surface, which is at a minimum distance from P. The point on surface $S_S(u, v)$ that is nearest to P is determined by minimizing the norm of P – $S_S(u, v)$. The linearized form is computed as Taylor expansion, as:

$$\begin{bmatrix} \left.\dfrac{\partial S_S}{\partial u}\right|_{u_i,v_i} & \left.\dfrac{\partial S_S}{\partial v}\right|_{u_i,v_i} \end{bmatrix} \begin{bmatrix} u_{i+1}-u_i \\ v_{i+1}-v_i \end{bmatrix} = [\mathrm{P} - S_S(u_i, v_i)]. \tag{29}$$

The matrix (i.e. 3×2) on the left-hand side in eqn (29), which contains column vectors, is the Jacobian transformation J_C. Since the Jacobian matrix is not rank deficient, the least square solution (i.e. 2-norm minimum) can be computed uniquely by inverting the Jacobian using its pseudo-inverse;

$$[u_{i+1} - u_i] = [Jc^T Jc]^{-1} Jc^T [\mathrm{P} - S_S(u_i, v_i)]. \tag{30}$$

The iterations (i.e. after 3–4 iterations in general) converge fast because of the quadratic convergence of Newton methods. When the surfaces near points are not within the SPT then the linearized forms of the surface equations are computed again. And, $S_S - S_{Pl} = 0$, at surface points (u, v) and (l, m) near to P_i, are tackled directly. The tangent planes at the surface points are intersected with a third nonparallel plane to uniquely define a new point P_{i+1}. There exist many methods to address this, e.g. [7, 11, 26]. The computationally efficient method is to use the mid-point of the projection of the surface near points on to the intersection line of the tangent planes to describe the next iterate (Fig. 5).

If N_S and N_{Pl} are the tangent plane normals at the surface near points, and $\mathrm{N}_{SPl} = \mathrm{N}_S \times \mathrm{N}_{Pl}$ is the third tangent plane normal, then the next iterate P_{i+1} is computed from

$$\begin{bmatrix} \mathrm{N}_S^T \\ \mathrm{N}_{Pl}^T \\ \mathrm{N}_{SPl}^T \end{bmatrix} [\mathrm{P}_{i+1}] = \begin{bmatrix} \mathrm{N}_S . S_S \\ \mathrm{N}_{Pl} . S_{Pl} \\ \mathrm{N}_{SPl} . (S_S + S_{Pl})/2 \end{bmatrix}. \tag{31}$$

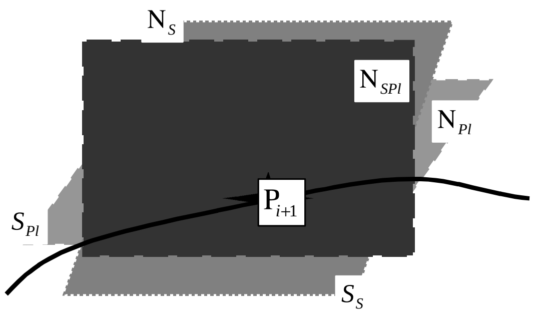


FIG. 5. The tangent planes on the surfaces, S_S and S_{Pl}.

The convex combination of the two projected points is sufficient for convergence (e.g. Mullenheim [36]). This means that the desired element (i.e. third) of the righthand side matrix in eqn (31) can generally be taken as $N_{Spl}\cdot(t\cdot S_S + (1 - t)S_{Pl})$, where $0 \le t \le 1$. In the present work, as given in eqn (31), we have taken $t = 1/2$.

The methods used above may be problematic in the case of nearly or the same parallel tangent planes at surface near points. If the planes are the same at two surfaces near points, then there is an intersection, and computationally it should have been computed with SPT, before refinement. If the tangent planes are parallel, then assuming twice differentiable surface equation will lead to linear convergence. Though, theoretically in this case, the solution is not possible, but within SPT, a root shall always be computed; and that is sufficient for computational purposes.

3.2.3. Relax points on to boundaries: Following [11], let the initial domain values be v_0, l_0, m_0, and the constant isoparametric $u = \bar{u}$ on the surface S_S (i.e. the $u = 0$, or $u = 1$ boundary of the triangular domain), then a new iterate can be computed by minimizing the norm of $S_{Pl}(l, m) - S_S(u, v)$. Computing the linearized form will result in 3×3 Newton iteration on v, l, m, meaning;

$$\begin{bmatrix} -\left.\dfrac{\partial S_S}{\partial v}\right|_{\bar{u},v_i} & \left.\dfrac{\partial S_{Pl}}{\partial l}\right|_{l_i,m_i} & \left.\dfrac{\partial S_{Pl}}{\partial m}\right|_{l_i,m_i} \end{bmatrix} \begin{bmatrix} v_{i+1} - v_i \\ l_{i+1} - l_i \\ m_{i+1} - m_i \end{bmatrix} = [S_S(\bar{u}, v_i) - S_{Pl}(l_i, m_i)] \tag{32}$$

with the termination of iteration at $\| S_S(\bar{u}, v_{i+1}) - S_{Pl}(l_{i+1}, m_{i+1}) \|_2 \langle$ SPT. Alternatively, in a more computationally efficient manner, it is possible to incorporate u = constant constraint, directly into the system, and this results in a 4×4 system matrix, in four unknowns. Let $\Delta u = u_2 - u_1$, and constraint u_2, as $u_2 = \Delta u + u_1 = \mathrm{a}$, where a is a constant, for a boundary curve with a = 0 or a = 1, then $\Delta u = \mathrm{a} - u_1$, and the system matrix is;

$$\begin{bmatrix} -\left.\dfrac{\partial S_S}{\partial u}\right|_{u_i,v_i} & -\left.\dfrac{\partial S_S}{\partial v}\right|_{u_i,v_i} & \left.\dfrac{\partial S_{Pl}}{\partial l}\right|_{l_i,m_i} & \left.\dfrac{\partial S_{Pl}}{\partial m}\right|_{l_i,m_i} \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \\ \Delta l \\ \Delta m \end{bmatrix} = \begin{bmatrix} S_S(u_i, v_i) - S_{Pl}(l_i, m_i) \\ \mathrm{a} - u_i \end{bmatrix} \tag{33}$$

Similarly, for $u + v = 1$ boundary of the triangular domain of surface S_S, the system is altered slightly for computational easiness. In that case, the sum of equations $u_2 = \Delta u + u_1$ and $v_2 = \Delta v + v_1$, is used to get $\Delta u + u_1 + \Delta v + v_1 = 1$. In other words, $\Delta u + \Delta v = 1 - u_1 - v_1$, and the corresponding system matrix is;

$$\begin{bmatrix} -\left.\dfrac{\partial S_S}{\partial u}\right|_{u_i,v_i} & -\left.\dfrac{\partial S_S}{\partial v}\right|_{u_i,v_i} & \left.\dfrac{\partial S_{Pl}}{\partial l}\right|_{l_i,m_i} & \left.\dfrac{\partial S_{Pl}}{\partial m}\right|_{l_i,m_i} \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \\ \Delta l \\ \Delta m \end{bmatrix} = \begin{bmatrix} S_S(u_i,v_i) - S_{Pl}(l_i,m_i) \\ 1 - u_i - v_i \end{bmatrix}. \tag{34}$$

The methods used above may be problematic in the case of nearly or the same parallel tangent planes at surface near points. In that case, similar techniques as mentioned before are used.

3.2.4. *Termination of intersection curve*: In this work, the intersection curve terminates, when,

a) The intersection curve being traced meets a boundary. In this case, the methods described in Section 3.2.3 are applied.
b) The intersection curve being traced meets an earlier intersection curve.

The second situation in general points to the existence of a 'branch point', where more than one intersection curve meet. In general, branch points are the points where surfaces share tangent plane (i.e. tangent points) [12]. In the present work, we only detect this problem, but have not addressed this. Hence, the intersection curve terminates even when it meets a branch point.

3.2.5. *Tolerances*: The basic tolerance used in this work is SPT. Barnhill and Kersey [12] examine whether the two points (i.e. P_1 and P_2) are the same or not and base the answer on the Euclidean distance between the points. In this work, SPT has been chosen to be SPT = 1.0e – 8. However, for engineering applications, SPT can be chosen anywhere between 1.0e – 4 and 1.0e – 8. Another tolerance used in this work is CRT, which is used as a prescribed stepping distance. This distance measurement is dependent on the relative magnitude of surfaces. As mentioned in Section 3.2.1, angle tolerance has been used, along with the estimation of curvature of intersection curves, and is incorporated in adaptive step length estimations. An adaptive step length provides better control over efficiency and robustness, and the angular tolerance positively affects the dependency on relative surface magnitudes and units of measurements. Hence, this combination is better suited for general surface/plane intersection problems.

3.2.6. *Repeat for all start points*: Once the intersection curve corresponding to the starting point (i.e. P_0) has been traced exhaustively (using steps from Section 3.1–3.2.1–3.2.4), then within different settings, another starting point is computed as mentioned in Section 3.1, and the whole process is run again.

In this work, the intersection curves have been computed at different X = constant, Y = constant, and Z = constant planes for different surfaces (i.e. surfaces having C^0, C^1, and C^2 continuities). The aim is to obtain the smallest number of ordered points that correctly represent the topology of the true intersection curve. In this work, we compute two points (minimum) that are desired even for an intersection curve that is linear. Since we have used cubic surfaces (i.e. $d = 3$ the most popular degree of surfaces in engineering sciences), the generated surfaces consist of 100 elements within a patch (i.e. clearly shown in Fig. 7(a), 10(a), and 12(a)). So, an arbitrary plane may pass through all the ten triangular elements inside one patch, though ideally one must compute three points on each triangle (i.e. two at the local boundary and one inside/at the triangle). This is computationally expensive, and reduces the speed of computation, and *in practice* really does not improve the quality or topology of the intersection curve. Since in this work the intersection curves are represented by their piecewise polynomial approximations, the computation of one point inside/at the triangle is sufficient for applications in engineering sciences. Therefore, we compute one intersection point on each triangle of each patch. Again, since the exact computation of the interior edge or line of derivational continuity, where it is imposed, is essential in engineering sciences for surface-related analysis, we compute one point on the interior edge, or line of derivational continuity and re-check via backtracking. Hence, for a two-patch surface we compute total 23 numbers of points (i.e. one starting point on the surface's exterior boundary + 2*10 + one point on the interior edge/line of derivational continuity + one end point of the surface at the surface's exterior boundary = 23) for an intersection curve.

The algorithm has been implemented in C++ using its object-oriented features on a Silicon Graphics$^{\text{TM}}$** Origin$^{\text{TM}}$** 200 workstation. The surface views and intersection curves are represented explicitly in AutoCAD$^{\text{TM}}$*** R2000.

4. Numerical examples and discussion

To illustrate the theoretical model of this work, we have considered different problems. The first problem deals with the intersection curve of $C^0 – C^0$ continuities combination. The second, under the cases (1) and (2), deals with the intersection curve of $C^1 – C^0$ and $C^2 – C^0$ continuities combinations. For the sake of reproducibility, we give the explicit coordinates and geometric definition with continuities of the point set.

Problem 1. This is basically a $Y = 5$ plane, modeled as a flat plate with five points and four patches. The point set consists of $P_1 = (5, 5, 5)$, $P_2 = (–5, 5, –5)$, $P_3 = (–15, 5, 5)$, $P_4 = (–5, 5, 15)$, and $P_5 = (–5, 5 ,5)$. The indices of the vertices of the triangles in geometry definition F are given by (1, 2, 5), (2, 3, 5), (3, 4, 5), and (4, 1, 5).

Intersection curves: The triangular domain is defined as in Fig. 6. Here C^0 continuity has been imposed across all the four patches (i.e. pa 1–pa 4). In this case, Fig. 7(a) shows the front view in which the connectivity and triangulation pattern is clear. Figure 7(b) shows the isometric view.

**Trademark and copyright with Silicon Graphics Corporation, USA.
***Trademark and copyright with AutoDesk Corporation, USA.

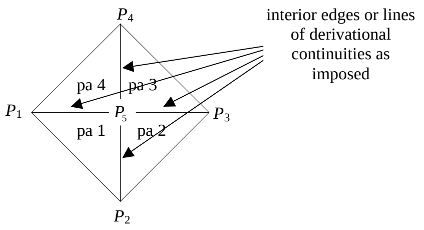

P_4
P_1
P_5
P_3
P_2
pa 4
pa 3
pa 1
pa 2
interior edges or lines of derivational continuities as imposed

FIG. 6. Definition of the triangular domain for flat rectangular plate.

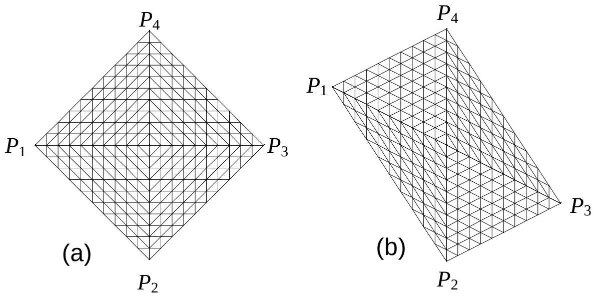


FIG. 7. Surface for a flat rectangular plate, C^0 continuous.

Since the surface is C^0 continuous, it is purely flat. We plot the intersection curves of the surface with constant X planes at the interval of 1.0, from (X = –14.0 to X = 4.0). Figure 8 shows the intersection curves at X planes. The surface is purely flat, so the intersection curves are constantly linear (i.e. this follows the obvious; any continuous surface for a flat plate will be a flat surface with linear curves of intersection).

Problem 2. This is a simple problem of a nonflat plate where one corner has been raised with respect to the plane passing through the three points; and it has been modeled with two patches. The point set consist of $P_1 = (0, 1, 0)$, $P_2 = (1, 0, 0)$, $P_3 = (1, 1, 1)$ and $P_4 = (0, 0, 1)$. The indices of the vertices of the triangles in geometry definition F are given by (1, 2, 4) and (2, 3, 4).

Case 1–Intersection curves: The triangular domain is defined as in Fig. 9. Here, C^1 continuity has been imposed across the patches (i.e. pa 1–pa 2). Figure 10(a) shows the front view in which the connectivity and triangulation pattern is clear. Figure 10(b) shows a typical isometric view. We have computed the intersection curves of the surface with constant Y planes at the interval of 0.1, from (Y = 0.1 to Y = 0.9). Figure 11 shows the intersection curves at constant Y planes. As stated earlier, since the surface is nonflat but is C^1 continu-

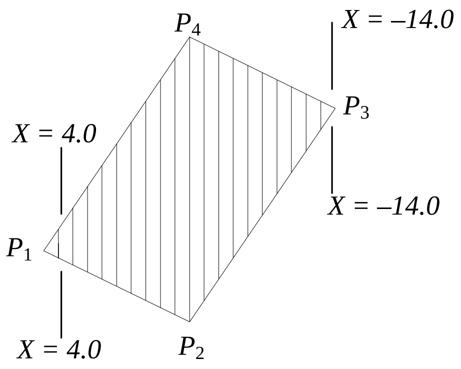


FIG. 8. Intersection curves of constant X planes with surface for flat rectangular plate, C^0 continuous.

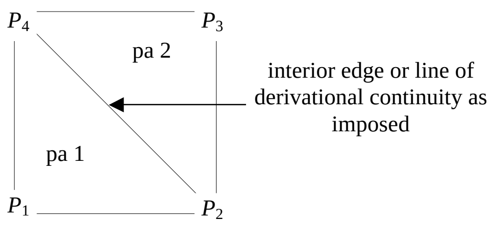


FIG. 9. Definition of the triangular domain for a nonflat rectangular plate.

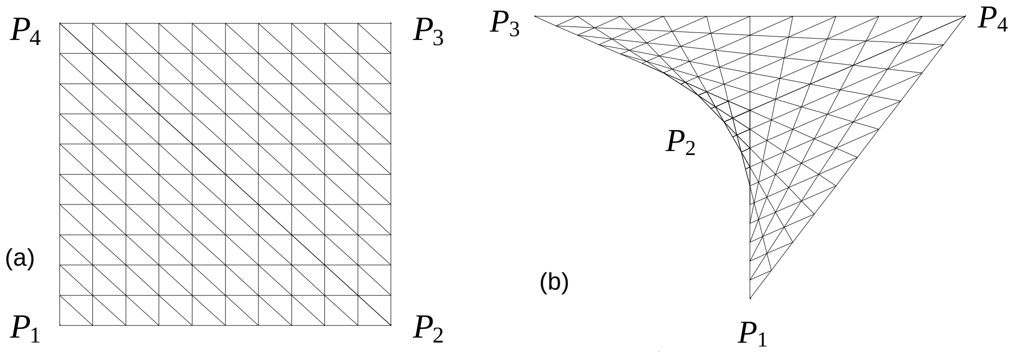


FIG. 10. Surface for a nonflat rectangular plate with one corner raised, C^1 continuous.

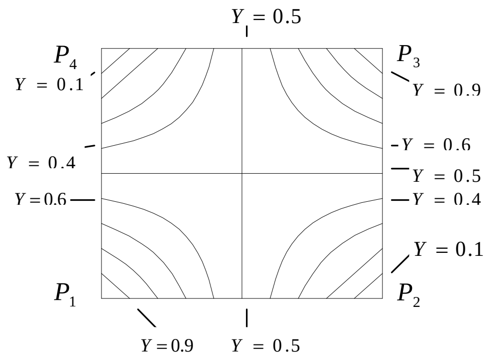


Fig. 11. Intersection curves of constant Y planes with surface for a nonflat rectangular plate with one corner raised, C^1 continuous.

ous, the intersection curves are continuous (i.e. nonlinear variation) without any break in continuity.

Case 2–Intersection curves: Here C^2 continuity has been imposed across all the patches (geometric definition as in Fig. 9). Figure 12(a) shows the front view in which the slightly curved sides, twist in space, higher curvatures, connectivity and triangulation pattern are visible and clear. Figure 12(b) shows a typical isometric view. As previously, we have computed the intersection curves of the surface with constant Y planes at the interval of 0.1, from ($Y = 0.1$ to $Y = 0.9$). Figure 13 shows the intersection curves at Y planes. The surface is nonflat but is C^2 continuous, so the intersection curves are highly nonlinear but continuous.

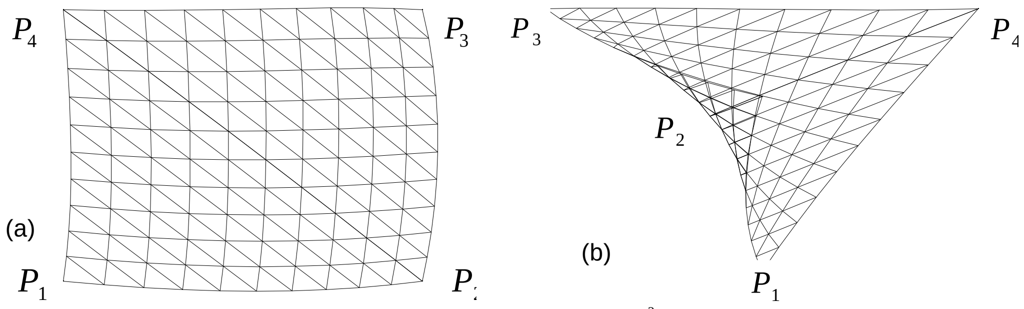


Fig. 12. Surface for a nonflat rectangular plate with one corner raised, C^2 continuous.

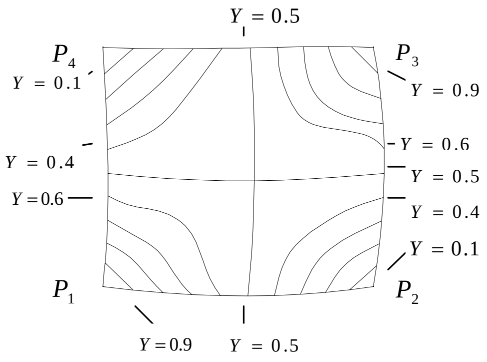


Fig. 13. Intersection curves of constant Y planes with surface for a nonflat rectangular plate with one corner raised, C^2 continuous.

5. Conclusions

This work has presented in general setting of derivational continuities a surface/plane intersection algorithm for non self-intersecting rational triangular Bezier surfaces defined over triangular domain using combination of ideas (i.e. [11, 12, 14, 24, 27]). The algorithm presented is a stepwise, structured, and combinatorial one. Hence, it allows better insight into the problem of the computation of surface/plane intersection curve. The method has been used to compute different planar sections (i.e. intersection curves with X, Y, and Z planes) for surfaces having different combination of continuities (i.e. C^0–C^0, C^1–C^0, and C^2–C^0 continuous surface/plane). The algorithm includes the computation of multiple intersection curves at one plane but does not include the computation of intersection curve with multiple branches (i.e. branch points), bifurcation points, and lower and higher-order singularities. A more complex algorithm in general settings of surfaces in engineering sciences will preferably include these features. Again, the detailed analysis of efficiency and complexity of the present algorithm has not been addressed. Though we have also computed the intersection curves for parallel planes, the *spatial coherency*, (e.g. Kulkarni and Dutta [37], Jun *et al*. [38]), among the planes, have not been incorporated in the algorithm. Again, the surface subdivision used in the present work is explicit, and theoretically based upon Goldman [27] that uses a *5-simplex* of polar value numerical computations. However, if the explicit control over the aspect ratio is not desired, then the same can be achieved in four calls (e.g. Prautzsch [39] and DeCarlo and Gallier [40]). The incorporation of the *four-call* procedure in the surface subdivision algorithm would mean better robustness and computational efficiency. Since in the present algorithm the surface/plane intersection problem has been considered as a special case of the surface/surface intersection problem in general setting of derivational continuities, theoretically some ideas of the algorithm may be utilized to compute intersection curves in surface/surface intersection. Furthermore, in the present algorithm, the correct computation of the topological properties of the intersection curve is an *in-built* feature, hence a suitable '*loop detection*' technique can be implemented as an *add-on* feature. This is under investigation.

Acknowledgements

The authors thank Prof. R. P. Gokarn, formerly faculty at the Department of Ocean Engineering and Naval Architecture, Indian Institute of Technology, Kharagpur, India, for his critical comments and suggestions on the manuscript, which have helped us immensely in revising the manuscript. They also thank Prof. S. P. Pal, Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India, for discussions, which have contributed to improving the clarity of the paper. The authors gratefully acknowledge their support and encouragement.

References

1. F. E. Wolter and S. T. Tuohy, Curvature computations for degenerate surface patches, *Computer Aided Geometric Des.*, **9**, 241–270 (1992).
2. G. Farin, *Curves and surfaces for computer aided geometric design. A practical guide*, 3rd edn, Academic Press (1993).
3. C. L. Bajaj, C. M. Hoffmann, R. E. Lynch and J. E. H. Hopcroft, Tracing surface intersections, *Computer Aided Geometric Des.*, **5**, 285–307 (1988).
4. W. T. Montaudouin, W. Tiller and H. Vold, Applications of power series in computational geometry, *Computer-Aided Des.*, **18**, 514–524 (1986).
5. C. Asteasu, Intersection of arbitrary surfaces, *Computer-Aided Des.*, **20**, 533–538 (1988).
6. T. Garrity and J. Warren, On computing the intersection of a pair of algebraic surfaces, *Computer Aided Geometric Des.*, **6**, 137–153 (1989).
7. E. G. Houghton, R. F. Emnett, J. D. Factor, and C. L. Sabharwal, Implementation of a divide-and-conquer method for intersection of parametric surfaces, *Computer Aided Geometric Des.*, **2**, 173–183 (1985).
8. D. Filip, R. L. Magedson, and R. P. Markot, Surface algorithms using bounds on derivatives, *Computer Aided Geometric Des.*, **3**, 295–311 (1986).
9. H. Bürger and R. Schaback, A parallel multistage method for surface/surface intersection, *Computer Aided Geometric Des.*, **10**, 277–291 (1993).
10. J. A. Gregory, *The mathematics of surfaces* (J. A. Gregory, ed.), Clarendon Press (1986).
11. R. E. Barnhill, G. Farin, M. Jordan, and B. R. Piper, Surface/surface intersection, *Computer Aided Geometric Des.*, **4**, 3–16 (1987).
12. R. E. Barnhill and S. N. Kersey, A marching method for parametric surface/surface intersection, *Computer Aided Geometric Des.*, **7**, 257–280 (1990).
13. A. Cugini, S. Radi, and C. Rizzi, A system of parametric surface intersection, in *Computer-aided surface geometry and design. The mathematics of surfaces* IV (A. Bowyer, ed.), Oxford University Press, pp. 231–242 (1994).
14. K. Abdel-Malek and H. J. Yeh, Determining intersection curves between surfaces of two solids, *Computer-Aided Des.*, **28**, 539–549 (1996).
15. K.-P. Cheng, Using plane vector fields to obtain all the intersection curves of two general surfaces, in *Theory and practice of geometric modeling* (W. Strasser and H.-P. Seidel, eds), Springer-Verlag, pp. 187–204 (1989).
16. G. A. Kriezis, N. M. Patrikalakis, and F. E. Wolter, Topological and differential equation methods for surface intersections, *Computer-Aided Des.*, **24**, 41–55 (1992).
17. T. W. Sederberg, H. N. Christriansen, and S. Katz, An improved test for closed loops in surface intersections, *Computer-Aided Des.*, **21**, 505–508 (1989).
18. T. W. Sederberg and R. J. Meyers, Loop detection in surface patch intersections, *Computer-Aided Des.*, **5**, 161–171 (1988).
19. P. Sinha, E. Klassen, and K. K. Wang, Exploiting topological and geometric properties for selective subdivision, in *Proc. ACM Symp. on Computational Geometry*, pp. 39–45 (1985).
20. T. A. Grandine and F. W.-Klein, IV A new approach to the surface intersection problem, *Computer Aided Geometric Des.*, **14**, 111–134 (1997).

21. G. Müllenheim, On determining start points for a surface/surface intersection algorithm, *Computer Aided Geometric Des.*, **8**, 401–408 (1991).

22. T. Dokken, Finding intersections of B-spline represented geometries using recursive subdivision techniques, *Computer Aided Geometric Des.*, **2**, 189–195 (1985).

23. T. Dokken, V. Skytt, and A. M. Ytrehus, Recursive subdivision and iteration in intersections and related problems, in *Mathematical methods in computer aided geometric design* (T. Lyche and L. L. Schumaker, eds), Academic Press, pp. 207–214 (1989).

24. K. Abdel-Malek and H. J. Yeh, On the determination of starting points for parametric surface intersections, *Computer-Aided Des.*, **29**, 21–35 (1997).

25. G. Farin, An SSI bibliography, in *Geometry processing for design and manufacturing* (R. E. Barnhill, ed.), SIAM, pp. 205–207 (1992).

26. I. D. Faux and M. J. Pratt, *Computational geometry for design and manufacture*, Ellis Horwood (1985).

27. R. N. Goldman, Subdivision algorithms for Bézier triangles, *Computer-Aided Des.*, **15**, 159–166 (1983).

28. R. P. Markot and R. L. Magedson, Procedural method for evaluating the intersection curves of two parametric surfaces, *Computer-Aided Des.*, **23**, 395–404 (1991).

29. E. J. Haug, C. M. Luh, F. Adkins, and J. Y. Wang, Numerical algorithms for mapping boundaries of manipulator work-spaces. *Proc. 24th ASME Mechanisms Conf.*, USA (1994).

30. B. Noble and J. W. Daniel, *Applied linear algebra*, Prentice-Hall (1988).

31. V. Bertram, *Practical ship hydrodynamics*, 2nd edn, Butterworth-Heineman, pp. 34–37 (2000).

32. G. Farin, Triangular Bernstein-Bézier patches, *Computer Aided Geometric Des.*, **3**, 83–127 (1986).

33. P. de Casteljau, *Shape mathematics and CAD*, Kogan Page (1986).

34. W. Stürzlinger, Ray-tracing triangular trimmed free-form surfaces, *IEEE Trans. Visualization Computer Graphics*, **4**, 202–214 (1998). web site address: http://citeseer.nj.nec.com/92665.html (2001).

35. L. Ramshaw, Bézier and B-splines as multiaffine maps, in *Theoretical foundations of computer graphics and CAD* (R. Earnshaw, ed.), Springer-Verlag, pp. 757–776 (1988).

36. G. Müllenheim, Convergence of a surface/surface intersection algorithm, *Computer Aided Geometric Des.*, **7**, 415–423 (1990).

37. P. Kulkarni and D. Dutta, Adaptive slicing of parameterizable algebraic surfaces for layered manufacturing, *Proc. ASME Design Technical Conf.*, Boston, MA, September 1995 (1995).

38. Cha-Soo Jun, Dong-Soo Kim, Deok-Soo Kim, Hyun-Chan Lee, Jiseon Hwang, and Tien-Chien Chang, Surface slicing algorithm based on topology transition, *Computer-Aided Des.*, **33**, 825–838 (2001).

39. Helmut Prautzsch, *Unterteilungsalgorithmen für multivariate splines*, Ph D thesis, TU, Braunschweig, Braunschweig, Germany (in German), (1984).

40. D. DeCarlo and J. Gallier, *On the efficiency of strategies for subdividing polynomial triangular surface patches*, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, USA, pp. 1–19, web site address: http://www.cis.upenn.edu/~jean/home.html (1999).

Nomenclature

J_C	=	Jacobian matrix
e	=	small distance in Euclidean or parametric space
$S_S(u, v)$	=	surface parametrized in independent parameters (u, v)
$S_{Pl}(l, m)$	=	plane defined as a fixed domain surface parametrized in parameters (l, m)
P_0	=	starting point on the intersection curve of surface/plane intersection
P_i	=	point on the intersection curve of surface/plane intersection

I_{SV}	=	intersection stepping vector
I_{SL}	=	intersection stepping length
Δq	=	user-specified angle tolerance
$y*$	=	Moore-Penrose pseudo-inverse of the Jacobian
N_{Pl}	=	tangent plane normal of surface $S_{Pl}(l, m)$
N_S	=	tangent plane normal of surface $S_S(u, v)$
N_i^S	=	surface normal for surface $S_{Sl}(u, v)$ at point P_i
N_i^{Pl}	=	surface normal for surface $S_{Pl}(l, m)$ at point P_i
SPT	=	user-specified same point tolerance
CRT	=	user-specified curve refinement tolerance