Two Methods for Surface/Surface Intersection Problem Comparative Study

Ramadhan Abdo Musleh Alsaidi
China
School of Mathematics and Statistics
Huazhong University of Science and Technology

ABSTRACT

The determination of the intersection curve between two surfaces may be seen as two different and sequential problems (1) determining initial points of the intersection curve and (2) tracing it from these points. Presented in this paper are: one technique for computing the initial point, and two methods for tracing the intersection curve of two parametric surfaces. Algorithms, implementation, and illustrative examples will be discussed as well as a comparative analysis of each method.

Keywords:

Surface/surface intersection, Bézier surface, Continuation method, The Marching Method with Differential Equations

1. INTRODUCTION

For a long time now of computer aided manufacturing [CAM] and computer aided design [CAD], intersection algorithms have always played a central role in making geometric modeling systems work. The first geometric modelers were curve based, and efficient algorithms were developed giving satisfactory intersection results. One example of such a system was the AUTOKON system, a complete batch oriented CAD/CAM-system for ship building which was developed in Norway in the 1960s. However, when surface and volume based systems were developed, intersection algorithms became more complex.
In curve based systems, the result of an intersection is points or segments of the curves being intersected. Thus, describing the intersection is fairly straight forward. In surface based systems the result of an intersection is points, curves, and regions of the surfaces being intersected[14].

1.1 Motivation

The computation of self-intersection for a patch or intersection between two patches is an important problem in Computer Aided Geometric Design [CAGD]; and this problem was the main topic of the European project GAIA II [http//www.sintef.no/static/AM/gaiatwo/].
In fact calculation of intersection curves between surfaces can be applied to various fields, such as computational geometry, solid modeling, geometric processing, computer aided design [CAD], computer aided geometric design [CAGD], manufacturing, numerical-controlled machining, visualization, and robotics.
In many applications, there is need to find the curves where two surfaces intersect, for example

—Constructing a contour map to graphically represent a prescribed surface or hyper-surface.
—Computing silhouettes to improve the graphical display of surfaces.
—Performing Boolean operations on solid bodies.
—Constructing smooth blending curves and surfaces.
—Finding offset curves and surfaces, e.g., Numerical-Controlled machining (NC-construction) (Theoretically defined offsets can have self-intersections)[22].

The main goal concerning the surface to surface intersection problem is to develop robust, accurate and fast algorithms for computing the intersection curve between two surfaces, needing the least user intervention.

1.2 Related Work

Researchers around the world have attempted to solve this problem; however, no perfect algorithm has been introduced. Each algorithm has several problems, specifically with the characteristic points. Bajaj et al.[4] presented a method in which a third-order Taylor a approxiamant is constructed by taking steps of variable lengths and the curve is computed using the general Newton iteration procedure. This method also presents techniques to handle singularities. Montaudouin et al. [29] presented a method where power series are used to approximate plane algebraic curves and surface intersections. General intersection problems have been addressed by [33, 15]. Lattice evaluation methods were used to determine the intersection curve[32]. Reviews of general intersection methods are numerous[5, 20, 30, 13]. Wilf and Manor[35] presented a method using a modification of Levins ruled-surface parameterization scheme, guided by invariant-factors classification and furthermore, by factorization of the parameterization polynomials. Markot and Magedson[28] presented a procedural method to parameterize the intersection curve of two surfaces by computing exact points on the true intersection curve. A parallel algorithm using the divide-and-conquer method was presented by Burger and Schaback[7]. The computational complexity of this algorithm was also analyzed. Search techniques were used to refine the interval progressively[19]. Another method that was applied to the surface-surface intersection problem is a topological and differential-equation method[11]. In this method, the vector field defined as the gradient of the oriented distance function is used to detect critical points in the field such as singularities. Tensorial differential equations are then used to trace intersection segments. Another method that uses unidimensional searches to detect intersection points was presented by Aomura and Uehara[3]. Surface intersection using parallelism was addressed by Chang et al.[10]. A different approach to higher-dimensional formulation including offsets, equal distance surfaces and variable radius blending surfaces was discussed by Hoffman[21]. A higher-dimensional formulation was also used by Chuang[12] to determine a local and global approximant. Marching methods have been extensively used by researchers in this field[11]. The accuracy of marching methods has been improved by proper control of the step size. Singularities were analyzed by Abdel-Malek,K. ,Harn-Jou Yah and Rockwood, A.[1] by locally constructing a second-order approximant to each surface. Parametric surface-surface intersection has also been addressed by Houghton et al.[23] Garrity and Warren[18] and Mullenheim[26]. Singularities along the intersection curve have been identified in the work by Lukacs[27] who used the quadratic of the curves curvature to detect any bifurcation points. Eigen values of this system were then studied to determine the form of the matrix, and therefore its behavior at critical points. Most numerical algorithms require the estimation of a starting point on or close to the solution curve. This topic forms the starting point of this paper. In recent years, there have been many studies concerning the determination of the initial points for tracing intersection curves. Cugini et al.[7] introduced the concept of shrinking bounding boxes that cover the total space. Parts of the two surfaces existing in the same bounding box are identified to calculate a starting point. The curve is then traced by introducing one additional constraint as the tangent to an advancing plane. The problem of loop detection was also addressed using relatively simpler algorithm[22, 34]. Although these algorithms may miss intersection components, they are somewhat simpler, more geometric, and in some cases more effective. Muellenheim [25] presented an iterative method for calculating a starting point that is close to a solution curve. The starting point has also been computed using lattice and subdivision methods[27]. Continuation methods[2, 31, 21, 16] were recently used to trace the curve and to determine tangents at bifurcation points[1]. Currently, the most widely used surface intersection algorithm is recursive subdivision[9, 10]. In most approaches, the determination of initial points faces the task of solving multivariable polynomial systems. Mrio C. F. and Marcos S. G. T. used the Projected Polyhedral Method to solve the multivariable polynomial systems[17]. Finally, in one of the new studies[8], the authors proposed a new method for computing an equation of a plane curve that is in correspondence with the intersection space curve of these two surfaces for very general parameterizations. This paper focuses on the problem of computing the intersection of two parametric surfaces $X(u, v)$ and $Y(s, t)$, since a common representation of surfaces in Solid Modeling and Computer Aided Geometric Design (CAGD) uses parameterized patches.(we address the computation of the intersection curve of two surface patches of bidegree (2,2),i.e., biquadratic patches.).

The rest of this paper is structured as follows. Section 2 we introduce and review the problem as well as some basic concepts and terminology to be used throughout the paper. For computing, the initial point of the intersection curves a technique is based on extended Newton Method is presented in section 3. Section 4 present two different techniques for tracing the intersection curves. We apply the two techniques to four representative examples and report the results in Section 5. Finally, the conclusions and future work are discussed in Section 6.

2. OVERVIEW OF THE PROBLEM AND SOLUTION

Indeed two surfaces of any kind can have one of the following two relationships with each other

—The two surfaces have no points in common.
—The two surfaces have a set of points in common.

The first relationship indicates that the two surfaces do not intersect. The second relationship is the main interest of this study, because it implies that the two surfaces actually intersect. If the set of points of the intersection of the two surfaces is not empty, then there are two possibilities

—The set contains only one point.
—The set contains more than one point, in which case the intersection could be either a line (a set of lines), or a smooth patch (a set of smooth patches).

In fact the generic algorithm for computing the intersection curve(s), consists of three steps which are:

—Find at least one point on each component of the intersection,
—trace the segments of the intersection curve, and
—collect and convert the segments into a format that is suitable for further processing (depending on the application).

2.1 Mathematical Formulation of the Problem

Let two surfaces be $X(u, v) = \begin{bmatrix} X_1(u, v) \\ X_1(u, v) \\ X_1(u, v) \end{bmatrix}$ and $Y(s, t) = \begin{bmatrix} Y_1(s, t) \\ Y_2(s, t) \\ Y_3(s, t) \end{bmatrix}$
where , $X, Y \in C^1[0, 1]^2$. are given, the problem of intersection is solved by computing the set

$$M = \{(u, v, s, t) \in [0, 1]^4 \,\|\, F(u, v, s, t) = X(u, v) - Y(s, t) = 0\} \Longleftrightarrow$$
$$F(u(\alpha), v(\alpha), s(\alpha), t(\alpha)) = \begin{bmatrix} X_1(u, v) - Y_1(s, t) \\ X_2(u, v) - Y_2(s, t) \\ X_3(u, v) - Y_3(s, t) \end{bmatrix} =$$
$$\begin{bmatrix} f_1(u, v, s, t) \\ f_2(u, v, s, t) \\ f_3(u, v, s, t) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

In the non-degenerate case i.e. the set of intersection points is not empty, the solution manifold M consists of one or several curves, each fulfilling the condition

$$F(u(\alpha), v(\alpha), s(\alpha), t(\alpha)) = 0 \tag{1}$$

where α is any parameter. It can be interpreted as a pair of two curves as:

$$k_1(\alpha) = \big(u(\alpha), v(\alpha)\big)^T \subseteq [0, 1]^2 \; and \; k_2(\alpha) = \big(s(\alpha), t(\alpha)\big)^T \subseteq [0, 1]^2$$

which represent the path of intersection on each of the surfaces in the corresponding parameter space.

2.2 Basic Concepts and Terminology

In this paper we restrict ourselves to parametric surfaces (including Bézier surface), which are assumed to be differentiable at any point since the parametric representation is the most used mathematical representation in industry. The parametric surfaces are described by a vector-valued function of two variables

$$S(u,v) = \big(x(u,v), y(u,v), z(u,v)\big) where u, v \in \Omega \subset R^2 \quad (2)$$

where u and v are the surface parameters. Expression 2 is called a parameterization of the surface S. At regular points, the partial derivatives $S_u(u,v)$ and $S_v(u,v)$ do not vanish simultaneously. These partial derivatives define the unit normal vector N to the surface at $S(u_0, v_0)$ as

$$N = \frac{S_u \times S_v}{\|S_u \times S_v\|} \quad (3)$$

where $\times$ denotes the cross product. A curve in the domain Ω can be described by means of its parametric representation $\{u = u(s), v = v(s)\}$. This expression defines a three-dimensional curve $C(s)$ on the surface S given by $C(s) = S\big(u(s), v(s)\big)$. Applying the chain rule, the tangent vector $\grave{C}(w)$ of this curve at a point $C(s)$ becomes

$$\grave{C}(w) = S_u \cdot \grave{u}(w) + S_v \cdot \grave{v}(w) \quad (4)$$

In this work the curve $C(w)$ will usually be parameterized by the arc-length w . Its geometric interpretation is that a constant step w traces a constant distance along an arc-length parameterized curve. Since some industrial operations require an uniform parameterization, this property has several practical applications. For example, in computer controlled milling operations, the curve path followed by the milling machine must be parameterized such that the cutter neither speeds up nor slows down along the path. Consequently, the optimal path is that parameterized by the arc-length.

2.2.1 *Bézier Surfaces.*

The Bézier surfaces are widely used in CAD applications. Such parametric surfaces are defined by the following expression[22]

$$S(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m(u) B_j^n(v) \cdot P_{ij}$$

Where P_{ij} are the control points of the surface and $\big(B_i^m(u)$, $B_j^n(v)\big)$ are Bernstein polynomials, defined by

$$B_i^m(u) = \binom{n}{i} \cdot u^i \cdot (1-u)^{n-i} , 0 \le u \le 1 \; e.g. \; u \in [0,1]$$

where $\binom{n}{i}$ is the binomial coefficient which defined as

$$\binom{n}{i} = \frac{n!}{i! \cdot (n-i)!}$$

Nonetheless, the Bernstein polynomials have, among others, the following main properties

$$B_i^m(u) > 0, \; \forall u \in [0,1]$$

$$\sum_{i=0}^{m} B_i^m(u) = 1$$

The control points are disposed in a rectangular net of $n + 1$ by $m + 1$ point. This way, the position of a surface point in the xyz space is given by

$$S(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v)) \end{bmatrix}$$

where each coordinate is defined by a polynomial in u and v. The surface degree (and therefore, the degree of the polynomials that describe each coordinate) is given by m and n.

3. FINDING A START POINT INCLUSION

3.1 Tracing a Branch of an Intersection Curve (continuation method)

As mentioned above the calculating surface intersection point means solving $Y(s,t) = X(u,v) \; for \; (u,v,s,t)$ the algorithms introduced here is a kind of marching method which follows the different branches of the solution performed within the parameter space. For each of these methods, it is necessary to obtain appropriate start points lying within the solution manifold. The curve tracing is initiated at these points.

3.2 Extended Newton Method

In this section, some basic definitions and theorems of a method for determining the roots of an arbitrary system of equation iteratively are given, where the equations can be nonlinear algebraic or transcendental. The number of equations of the system can be greater than, less than or equal to the number of variables. Details of this method can be found in[6].
Let $X = [x_1, x_2, \ldots, x_k]^T$ be a k-dimensional column vector with components $x_1, x_2, \ldots, x_k$ in C and $F(X) = [f_1(x), f_2(x), \ldots, f_h(x)]^T$ an h-dimensional vector valued function. To deal with the norm of a vector, we can choose any one of the norms, for example

$$\|X\|_p = (\textstyle\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}} \; , \; \forall \, 1 \le p < \infty \; and \; \|X\|_\infty = max(|x_1, \ldots, x_n|)$$

We define the Jacobian matrix of f w.r.t. x to be the $h \times k$ matrix as

$$J_f(x) = \left(\tfrac{\partial f_i(x)}{\partial x_j}\right)_{i=1\ldots h, j=1\ldots k}$$

When $h = k$ and with the assumption that $det(J(x)) \neq 0$ for x in a closed and bounded set S, we will denote the inverse Jacobian of f at x by J_f^{-1}. In this case, Newtons method is a wellknown iterative method for determining the roots of the equation $f(x) = 0$. The iteration can be written in the form

$$x_{n+1} = g(x_n), \; where \; g(x_n) = x_n - J_f^{-1}(x_n) \cdot f(x_n)$$

Unfortunately, here $h \neq k$, the Jacobian matrix of is not a square matrix. Therefore, it cannot be inverted. Hence, Newtons method fails to apply to such cases. We extend Newtons method such that the extended method can be applied to these cases as well. To get around the problem of a n-square matrix, we will use what we call the pseudo inverse of the Jacobian matrix.
Definition 1
A pseudo inverse of a matrix A is a matrix $A^\perp$ satisfying

$$AA^\perp A = A, \;\; A^\perp A A^\perp = A^\perp, \;\; (A^\perp A)^\star = A^\perp A, \;\; (AA^\perp)^\star = AA^\perp$$

where $(.)^\star$ denotes the conjugate transpose of the matrix in case of real coefficient functions, we will use $(.)^T$ for $(.)^\star$.

Theorem 1
For any matrix A there exists a pseudo inverse matrix $A^\perp$. Moreover the pseudo inverse is unique.
There are several techniques for obtaining the pseudo inverse of a numerical matrix, for example Hestenes technique or Grevilles technique (see[24]). However, those techniques are pure numerical techniques, i.e., they can only be applied to each individual numerical matrix; and we have to take into the account the problems of rounding error, word length, precision of arithmetic, and so on. Moreover, when we use such techniques in our iterative method for determining the roots of a system of equations, we have to apply the approximate process for obtaining the pseudo inverse matrix $J^\perp$ at each iterate

$$J^-(\{x^{(0)}, x^{(1)}, \ldots, x^{(n)}, \ldots\}) = \left(J(x)\cdot J^T(x)\right)^{-1}\cdot J^T(x)$$

That means that it is time consuming and has accumulated errors. To avoid such problems, we try to find a symbolic formula and use symbolic computation for the method. For example

$$J^+ = J^T\cdot(J\cdot J^T)^{-1} \;\; or \;\; J^- = (J^T\cdot J)^{-1}\cdot J^T$$

Lemma 1
For all x, if $(J\cdot(x)J^T(x))$ is nonsingular then $J^+(x)$ is the pseudo inverse of matrix $J(x)$.
Proof Lemma 1
The matrix$J^+(x)$is defined when $(J(x)\cdot J^T(x))$is nonsingular; and it is easy to check that $J^+(x)$ satisfies the definition 1.
Lemma 2
For all x, if $(J^T(x)\cdot J(x))$ is nonsingular then $J^-(x)$ is the pseudo inverse of matrix $J(x)$.[13]
Lemma 3
For all x, if both $J^+(x)$and $J^-(x)$ exist then they are identical.
Matrix $J^+(x)$or $J^-(x)$ is a matrix of order $k\times h$. Hence, we can define the function g as follows

$$g(x) = x - J^+(x)\cdot f(x) \; or \; g(x) = x - J^-(x)\cdot f(x)$$

where $g(x)$, x are vectors of the same dimension k and f is defined as equ.1. We find a fixed point of the function g w.r.t. the initial point $x^{(0)}$ by the following algorithm
Algorithm 1 (Extended Newton Method).

Input: a function $F: C^4 \to C^3$, an initial point $x^{(0)}$, a measured value (a tolerance ε) and a maximum number of iterates M .
Output: A starting point.
Method steps

```
(1) Let initial number of iteration ($L = 1$) and
    $F(u(\alpha), v(\alpha), s(\alpha), t(\alpha)) = \begin{bmatrix} X_1(u,v)-Y_1(s,t) \\ X_2(u,v)-Y_2(s,t) \\ X_3(u,v)-Y_3(s,t) \end{bmatrix} = \begin{bmatrix} f_1(u,v,s,t) \\ f_2(u,v,s,t) \\ f_3(u,v,s,t) \end{bmatrix}$
(2) Compute Jacobian matrix $J$ at $x^{(0)}$ as:
    $J_f(x^{(0)}) = \left(\frac{\partial f_i(x^{(0)})}{\partial x_j}\right)_{i=1\ldots3, j=1\ldots4}$
(3) Evaluate a function $F$ at $x^{(0)}$.
(4) Extract new point via [expr]
    $x_{new} = x^{(0)} - pinv(J(x^{(0)})) \times F(x^{(0)})$.
    where (pinv) is pseudo inverse.
(5) Compute $t = \|x_{new} - x^{(0)}\|$.
(6) Do test $if\ t \le \varepsilon$
    $then\ Start - point = x_{new},\ exit.$
    $else\ if\ t>\varepsilon\ and\ N \le M$
    $set\ x^{(0)} = x_{new},\ L = L + 1;$
    $goto\ step$ 2.
    $else\ return\ (failure\ non - convergence\ try\ for\ new\ initial\ point).$
```

4. ALGORITHMS FOR TRACING THE INTERSECTION CURVES

Once the starting point is found, the subsequent points on the intersection curve can be traced along the tangent direction. Suppose the parametric coordinates of the starting point q are (u_0, v_0) and (s_0, t_0) .The constraint function $H(q)$ is defined by

$$F(q) = \begin{bmatrix} X_1(u_0,v_0)-Y_1(s_0,t_0) \\ X_2(u_0,v_0)-Y_2(s_0,t_0) \\ X_3(u_0,v_0)-Y_3(s_0,t_0) \end{bmatrix} \tag{5}$$

The Jacobian of the constraint function $F(q)$ for a certain configuration q is the 3×4 matrix as:

$$J_f(q) = \left(\frac{\partial f_i(q)}{\partial x_j}\right)_{i=1\ldots3, j=1\ldots4}$$

Here, the tangent vector λ to the set defined by $H(q) = 0$ is uniquely defined by

$$J_f(q)\cdot\lambda = 0 \tag{6}$$

$$\|\lambda\|_2 = 1 \;\; (normalization). \tag{7}$$

$$det\begin{bmatrix} J_f(q) \\ \lambda^T \end{bmatrix} > 0 \tag{8}$$

Where Equations 6,7 and 8 were set forth to define a unique tangent in the Predictor-Corrector method [17]. Combining equation 5 and equation 6 yields a system of four equations in four variables, such that the Newton-Raphson iteration method can be used to calculate the new point x. By setting the new point as the current point, the tracing is continued until one of three cases occurs

(1) the boundary is reached,
(2) returning to the starting point, or
(3) reaches to the non-convergence point.

The Complete Algorithm

(1) Find an inclusion of one start point of an intersection curve denote by q.
(2) Solve equation 6 then set it e.
(3) Using relation $x_{new} = q \pm (e\times d)$ to compute a new initial guess point.(where d is the desire distance)
(4) Compute the next point using Extended Newton Methods where x_{new} is a new initial guess point.

4.1 The Marching Method with Differential Equations

Let $F(u,v)$ and $G(s,t)$ be two surfaces whose unit normal vectors at a point on the intersection curve $C(w)$ between $F(u,v)$ and $G(s,t)$ are N_1 and N_2 respectively.

$$N_1 = \frac{F_u \times F_v}{\|F_u \times F_v\|},\ N_2 = \frac{G_s \times G_t}{\|G_s \times G_t\|}$$

Once the initial points of the intersection curves are obtained, each curve $C(w)$,parameterized by the arc length w, is traced using the Marching Method proposed by [9]. The marching direction is given by the tangent vector of $C(w)$, which is perpendicular to the normal vector of both surfaces at the given intersection point

$$\grave{C} = \frac{dC}{dw} = \frac{N_1 \times N_2}{|N_1 \times N_2|}$$

Applying the chain rule, we obtain the unit tangent vector $T_1(w)$ ($T_2(w)$ resp.) to the curve $C(w)$ considered as belonging to the surface F (G resp.) is given by

$$\frac{dC}{dw} = \frac{du}{dw} \cdot F_u + \frac{dv}{dw} \cdot F_v.$$
$$\frac{dC}{dw} = \frac{ds}{dw} \cdot G_s + \frac{dt}{dw} \cdot G_t.$$

Because T_1 and N_2 are orthogonal and so are T_2 and N_1 we have

$$\begin{aligned} \left(\frac{\partial F}{\partial u} \cdot N_2\right) \cdot \left(\frac{du}{dw}\right) + \left(\frac{\partial F}{\partial v} \cdot N_2\right) \cdot \left(\frac{dv}{dw}\right) &= 0, \\ \left(\frac{\partial G}{\partial s} \cdot N_1\right) \cdot \left(\frac{ds}{dw}\right) + \left(\frac{\partial G}{\partial t} \cdot N_1\right) \cdot \left(\frac{dt}{dw}\right) &= 0 \end{aligned} \tag{9}$$

On the other hand, since the curve $C(w)$ belongs to both F and G, we have

$$\begin{aligned} E_1 \left(\frac{du}{dw}\right)^2 + 2P_1 \left(\frac{du}{dw}\right)\left(\frac{dv}{dw}\right) + Q_1 \left(\frac{dv}{dw}\right)^2 &= 1, \\ E_2 \left(\frac{ds}{dw}\right)^2 + 2P_2 \left(\frac{ds}{dw}\right)\left(\frac{dt}{dw}\right) + Q_2 \left(\frac{dt}{dw}\right)^2 &= 1 \end{aligned} \tag{10}$$

where E,P and Q are the coefficients of the First Fundamental Form of the surface given by:

$$E_1 = F_u \cdot F_u,\ P_1 = F_u \cdot F_v,\ Q_1 = F_v \cdot F_v$$
$$E_2 = G_s \cdot G_s,\ P_2 = G_s \cdot G_t,\ Q_2 = G_t \cdot G_t$$

Solving equation 9 and 10 for $\frac{du}{dw}, \frac{dv}{dw}, \frac{ds}{dw}$ *and* $\frac{dt}{dw}$ we obtain

$$\left.\begin{aligned} \frac{du}{dw} &= \pm \frac{\left(\frac{\partial F}{\partial v} \cdot N_2\right)}{\sqrt{E_1\left(\frac{\partial F}{\partial v} \cdot N_2\right)^2 - 2P_1\left(\frac{\partial F}{\partial v} \cdot N_2\right)\left(\frac{\partial F}{\partial u} \cdot N_2\right) + Q_1\left(\frac{\partial F}{\partial u} \cdot N_2\right)^2}} \\ \frac{dv}{dw} &= \mp \frac{\left(\frac{\partial F}{\partial u} \cdot N_2\right)}{\sqrt{E_1\left(\frac{\partial F}{\partial v} \cdot N_2\right)^2 - 2P_1\left(\frac{\partial F}{\partial v} \cdot N_2\right)\left(\frac{\partial F}{\partial u} \cdot N_2\right) + Q_1\left(\frac{\partial F}{\partial u} \cdot N_2\right)^2}} \\ \frac{ds}{dw} &= \pm \frac{\left(\frac{\partial G}{\partial t} \cdot N_1\right)}{\sqrt{E_2\left(\frac{\partial G}{\partial t} \cdot N_1\right)^2 - 2P_2\left(\frac{\partial G}{\partial t} \cdot N_1\right)\left(\frac{\partial G}{\partial s} \cdot N_1\right) + Q_2\left(\frac{\partial G}{\partial s} \cdot N_1\right)^2}} \\ \frac{dt}{dw} &= \mp \frac{\left(\frac{\partial G}{\partial s} \cdot N_1\right)}{\sqrt{E_2\left(\frac{\partial G}{\partial t} \cdot N_1\right)^2 - 2P_2\left(\frac{\partial G}{\partial t} \cdot N_1\right)\left(\frac{\partial G}{\partial s} \cdot N_1\right) + Q_2\left(\frac{\partial G}{\partial s} \cdot N_1\right)^2}} \end{aligned}\right\} \tag{11}$$

which together with an initial point of the intersection curve$(u(0), v(0), s(0), t(0)) = (u_0, v_0, s_0, t_0)$ constitutes an initial value problem for this system of four explicit first-order ordinary differential equations. The signs $\pm$ and $\mp$ in 11 mean that there are two arcs of curve starting at (u_0, v_0, s_0, t_0) associated with the two possible opposite directions of the tangent vectors $T_1(w)$ and $T_2(w)$[17]. Above are the equations from which the points of the intersection curves may be successively determined using a numerical integration method for the problem of the initial value with the given system of ordinary differential equations of 11 [5] Hu et al. did not explain which method was used to integrate the ordinary differential equations, as a first implementation we decided to use the $4th$ order Runge-Kutta Method to accomplish this task. So, the Marching Algorithm has the following steps:

Step1: Obtaining initial points for the algorithm, which means at least one point for each intersection curve. This is done using the procedures presented in the previous sections.

Step2: For each of these points, are obtained the derived equations of the surface parameters with respect to the arc length of the intersection curve using Eq.11.

Step3: Integrate this system of ordinary differential equations using a numerical method ($4th$ order Runge-Kutta, for instance), providing this way the next point of the intersection curve.

Steps 2 and 3 are repeated until the traced curve reaches a turning point or an initial border points (previously determined).

5. SOME ILLUSTRATIVE EXAMPLES

5.1 Example

To illustrate the determination of surfaces intersection using the continuation method that outlined above. Consider the two surfaces

$$S_1(u,v) = \begin{bmatrix} u \\ v \\ u^2 + v^2 \end{bmatrix}, S_2(s,t) = \begin{bmatrix} s \\ t \\ 9 - \frac{(s^2+t^2)}{5} \end{bmatrix}$$

where the two surfaces are shown in figure 1. Firstly to demon-

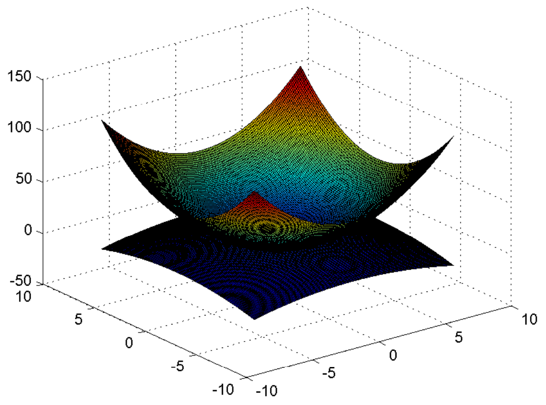

Fig. 1. Two parametric surfaces intersecting

strate the use of the Extended Newton Method set the initial guess is specified as $x_0 = (1,1,1,1)$. At the end of the $4th$ iteration the starting point on the curve is computed as $q = (1.9365, 1.9365, 1.9365, 1.9365)$. Secondly using the continuation method the resulting intersection curve in Figure 2. We can drive 450 points using Continuation method, and we list some of them as follows.

Table 1. list some of intersection points for example 5.1

kthPoint	Coordinates	kthPoint	Coordinates
0	(1.9365,1.9365,1.9365,1.9365)	240	(-1.9278,1.9451,-1.9278,1.9451)
4	(1.7811,2.0803,1.7811,2.0803)	276	(-1.9330,1.9400,-1.9330,1.9400)
20	(0.8477,2.6041,0.8477,2.6041)	301	(-1.9347,1.9383,-1.9347,1.9383)
33	(0.1167,2.7361,0.1167,2.7361)	367	(-1.9362,1.9368,-1.9362,1.9368)
42	(0.48420.1249,0.4910,0.1454)	89	(0.3823,0.1552,0.3865,0.1651)
45	(0.4499,0.1324,0.4552,0.1487)	49	(0.4205,0.1410,0.4250,0.1542)

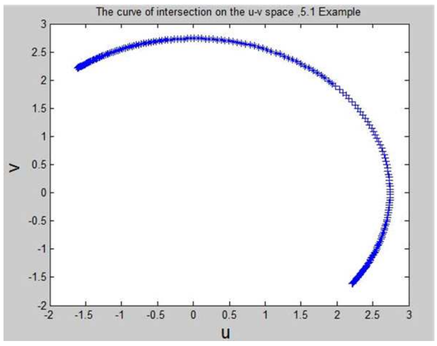

Fig. 2. The result curve of intersection on the$u - v$ space in Ex. 5.1

5.2 Example

Given two Bezier surfaces (S_1 *and* S_2) have the control points as:

$$S_1(u,v) = \left\{ \begin{matrix} (\frac{1}{7},0,\frac{3}{5}) & (\frac{3}{5},\frac{1}{5},\frac{3}{4}) & (1,0,\frac{7}{10}) \\ (\frac{3}{8},\frac{4}{9},\frac{2}{3}) & (\frac{2}{3},\frac{3}{4},\frac{1}{3}) & (\frac{6}{7},\frac{3}{8},\frac{5}{7}) \\ (\frac{1}{5},\frac{6}{7},\frac{4}{7}) & (\frac{3}{4},\frac{7}{8},\frac{3}{4}) & (\frac{7}{8},\frac{7}{9},\frac{5}{8}) \end{matrix} \right\}$$

and

$$S_2(s,t) = \left\{ \begin{matrix} (\frac{2}{7},\frac{1}{7},\frac{2}{5}) & (\frac{3}{5},\frac{1}{10},\frac{2}{3}) & (1,0,\frac{4}{5}) \\ (\frac{3}{8},\frac{4}{9},\frac{2}{3}) & (\frac{1}{3},\frac{1}{2},1) & (\frac{5}{7},\frac{3}{8},\frac{2}{7}) \\ (\frac{1}{5},\frac{6}{7},\frac{3}{7}) & (\frac{3}{4},\frac{7}{8},\frac{5}{8}) & (\frac{7}{8},\frac{4}{7},\frac{1}{2}) \end{matrix} \right\}$$

The two Bezier surfaces are shown in Figure3. The initial guess

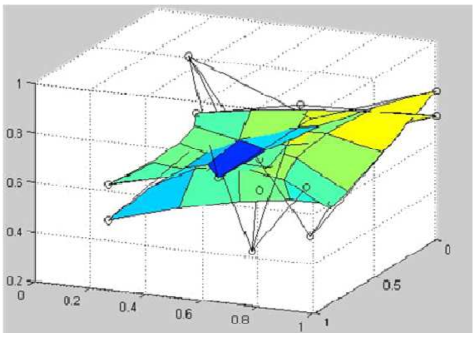

Fig. 3. Two quadratic Bezier surfaces.

is specified as $x_0 = (0.99940.25010.99940.2502)$. At the end of the four iterations the starting point on the curve is computed as $q = (0.72740.23270.73490.2990)$. The resulting intersection curve using the continuation method after compute 200 points at both sides of start point are shown in Figure4. We list some of

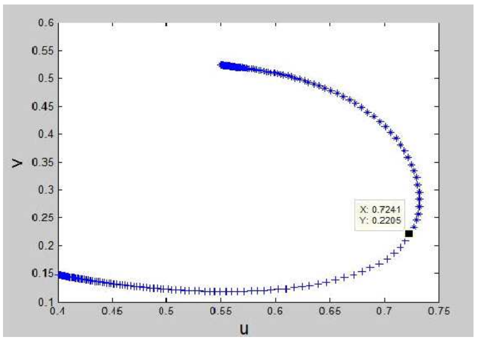

Fig. 4. The result curve of intersection two Bezier surfaces on the u, v space in Ex. 5.2.

points as follows.

5.3 Example

We will use the two surfaces in example (5.1) but here we use the Marching Method with Differential Equations the result of intersection curve compute 100 points and choose $d = 0.2$ are shown in figure 5. We list some of points as follows.

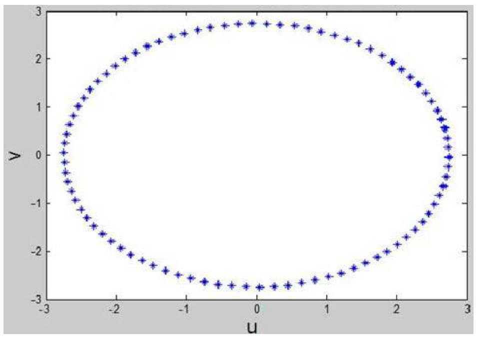

Fig. 5. Result curve obtained by using Marching Method Ex.5.3

Table 2. List some of intersection points for example 5.2

kth point	Coordinates	kth point	Coordinates
0	(0.7274,0.2327,0.7849,0.2990)	57	(0.3939,0.1505,0.3981,0.1613)
5	(0.6506,0.4785,0.7954,0.5829)	64	(0.3862,0.1536,0.3904,0.1637)
15	(0.5868,0.5138,0.7475,0.6408)	77	(0.3827,0.1550,0.3870,0.1649)
25	(0.5839,0.5149,0.7450,0.6428)	80	(0.3825,0.1551,0.3868,0.1650)
33	(0.7031,0.1782,0.7436,0.2362)	81	(0.3825,0.1551,0.3867,0.1650)
78	(-1.3338,2.3919,-1.3338,2.3919)	409	(-1.9364,1.9366,-1.9364,1.9366)
100	(-1.6064,2.2180,-1.6064,2.2180)	420	(-1.9364,1.9366,-1.9364,1.9366)
130	(-1.7882,2.0742,-1.7882,2.0742)	433	(-1.9364,1.9366,-1.9364,1.9366)
200	(-1.9123,1.9604,-1.9123,1.9604)	450	(-1.9365,1.9365,-1.9365,1.9365)

Table 3. List some of intersection points for example 5.3

kth point	Coordinates	kth point	Coordinates
0	(1.9365,1.9365,1.9365,1.9365)	54	(-2.7343,-0.1545,-2.7343, 0.1545)
7	(2.5752,0.9319,2.5752, 0.9319)	60	(-2.5415,1.0202,-2.5415,1.0202)
15	(2.6621,-0.6430,2.662,-0.6430)	67	(-1.7174 ,2.1332 ,-1.7174,2.1332)
23	(1.8658,-2.0047,1.8658,-2.0047)	73	(-0.6501,2.6603,-0.6501,2.6603)
33	(0.0527,-2.7381,0.0527,-2.7381)	80	(0.7345,2.6383,0.7345,2.6383)
37	(-0.7380,-2.6373,-0.7380,-2.6373)	89	(2.1934,1.6399,2.19341.6399)
40	(-1.2936,-2.4139,-1.2936-2.4139)	93	(2.5727,0.9388,2.5727,0.9388)
43	(-1.7873,-2.0750,-1.7873,-2.0750)	97	(2.7341,0.1581,2.7341,0.1581)
50	(-2.5739,-0.9353,-2.5739 ,0.9353)	100	(2.7031,-0.4399 ,2.7031,-0.4399)

5.4 Example

We will use the two surfaces in example (5.2) but here we use the Marching Method with Differential Equations the results of intersection curve are shown in figure 6. We can drive 60 points using

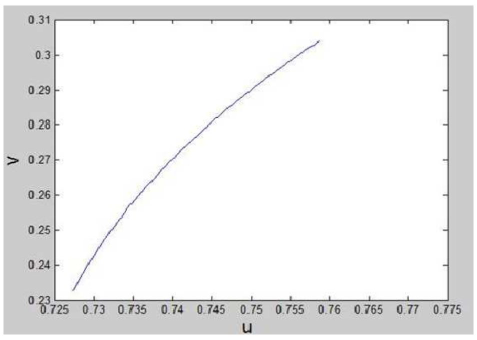

Fig. 6. Result curve obtained by using Marching Method for Bezier surfaces.

the Marching Method with Differential Equations, and we list some of them as follows.

6. CONCLUSION

I presented two different algorithms for computing the intersection curve between two parametric surfaces. I implemented the methods using Matlab software and applied them to many test cases include Bezier surface. The continuation method was able to deal with all test cases. It may produce subsequent points on the intersection curve along the tangent direction but not all the points. We have noticed using this practical method that the processing time taking is short which make it relatively fast meanwhile it keeps its accuracy rate relatively high. After experimenting with Marching Method with Differential Equations we arrived at the conclusion this method is very general, it can be applied to any pair of parametric surfaces and it has shown a very good performance in the examples described in this paper but it takes longer time specially in Bezier surfaces and the accuracy is not very sufficient, so these methods for Bezier surface have not yet been developed enough to come up with practical results and warrant further research.

7. REFERENCES

[1] Karim Abdel-Malek and Harn-Jou Yeh. Determining intersection curves between surfaces of two solids. *Computer-Aided Design*, 28(6):539–549, 1996.

[2] Eugene L Allgower and Kurt Georg. *Numerical continuation methods*, volume 13. Springer-Verlag Berlin, 1990.

[3] S. Aomura and T. Uehara. Intersection of arbitrary surfaces. *Journal of the Japan Society of Precision Engineering*, 57(9):1673–1679, 1991.

[4] Chandrajit L Bajaj, Christoph M Hoffmann, Robert E Lynch, and JEH Hopcroft. Tracing surface intersections. *Computer aided geometric design*, 5(4):285–307, 1988.

[5] Robert E Barnhill. *Geometry processing for design and manufacturing*. SIAM, 1992.

[6] Thomas L Boullion and Patrick L Odell. *Generalized inverse matrices*. Wiley-interscience New York, 1971.

[7] Heiko Bürger and Robert Schaback. A parallel multistage method for surface/surface intersection. *Computer aided geometric design*, 10(3):277–291, 1993.

[8] Laurent Busé and Thang Luu Ba. The surface/surface intersection problem by means of matrix based representations. *Computer Aided Geometric Design*, 29(8):579–598, 2012.

[9] Vijaya Chandru, Debasish Dutta, and Christoph M. Hoffmann. On the geometry of dupin cyclides. *The Visual Computer*, 5(5):277–290, 1989.

!

Table 4. List some of intersection points for example 5.4

kth point	Coordinates	kth point	Coordinates
1	(0.7274,0.2327,0.7849,0.2990)	23	(0.7357,0.2597,0.7693,0.2685)
8	(0.7296,0.2414,0.7801,0.2893)	27	(0.7376,0.2646,0.7663,0.2631)
11	(0.7307,0.2451,0.7780,0.2851)	31	(0.7396,0.2694,0.7632,0.2576)
15	(0.7322,0.2500,0.7751,0.2796)	34	(0.7412,0.2730,0.7609,0.2535)
19	(0.7339,0.2549,0.7722,0.2740)	52	(0.7523,0.2940,0.7464,0.2292)
20	(0.7343,0.2561,0.7715,0.2727)	60	(0.7581,0.3032,0.7396,0.2185)

[10] Long Chyr Chang, Wolfgang W Bein, and Edward Angel. Surface intersection using parallelism. *Computer aided geometric design*, 11(1):39–69, 1994.

[11] Koun-Ping Cheng. Using plane vector fields to obtain all the intersection curves of two general surfaces. In *Theory and practice of Geometric Modeling*, pages 187–204. Springer, 1989.

[12] Jung-Hong Chuang and Christoph M Hoffmann. On local implicit approximation and its applications. *ACM Transactions on Graphics (TOG)*, 8(4):298–324, 1989.

[13] Qiulin Ding and Beaumont John Davies. *Surface engineering geometry for computer-aided design and manufacture*. Prentice Hall Professional Technical Reference, 1988.

[14] Tor Dokken. Aspects of intersection algorithms and approximation. *Doctor thesis*, 1997.

[15] Rida T Farouki. The characterization of parametric surface sections. *Computer Vision, Graphics, and Image Processing*, 33(2):209–236, 1986.

[16] Mário Carneiro Faustini and Marcos Sales Guerra Tsuzuki. Algorithm to determine the intersection curves between bezier surfaces by the solution of multivariable polynomial system and the differential marching method. *Journal of the Brazilian Society of Mechanical Sciences*, 22(2):259–271, 2000.

[17] Akemi Gálvez, Jaime Puig-Pey, and Andrés Iglesias. A differential method for parametric surface intersection. In *Computational Science and Its Applications–ICCSA 2004*, pages 651–660. Springer, 2004.

[18] Thomas Garrity and Joe Warren. On computing the intersection of a pair of algebraic surfaces. *Computer Aided Geometric Design*, 6(2):137–153, 1989.

[19] Michael Gleicher and Michael Kass. An interval refinement technique for surface intersection. In *Proceedings of the conference on Graphics interface'92*, pages 242–249. Morgan Kaufmann Publishers Inc., 1992.

[20] Christoph M Hoffmann. *Geometric and solid modeling: an introduction*. Morgan Kaufmann Publishers Inc., 1989.

[21] Christoph M Hoffmann. A dimensionality paradigm for surface interrogations. *Computer Aided Geometric Design*, 7(6):517–532, 1990.

[22] Josef Hoschek, Dieter Lasser, and Larry L Schumaker. *Fundamentals of computer aided geometric design*. AK Peters, Ltd., 1993.

[23] Elizabeth G Houghton, Robert F Emnett, James D Factor, and Chaman L Sabharwal. Implementation of a divide-and-conquer method for intersection of parametric surfaces. *Computer Aided Geometric Design*, 2(1):173–183, 1985.

[24] SM Hu, JG Sun, TG Jin, and GZ Wang. Computing the parameters of points on nurbs curves and surfaces via moving affine frame method. *Journal of Software*, 11(1):49–53, 2000.

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

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

[27] Gabor Lukacs. Simple singularities in surface-surface intersections. In *INSTITUTE OF MATHEMATICS AND ITS APPLICATIONS CONFERENCE SERIES*, volume 48, pages 213–213. OXFORD UNIVERSITY PRESS, 1994.

[28] Robert P Markot and Robert L Magedson. Procedural method for evaluating the intersection curves of two parametric surfaces. *Computer-Aided Design*, 23(6):395–404, 1991.

[29] Yves de Montaudouin, Wayne Tiller, and Havard Vold. Applications of power series in computational geometry. *Computer-Aided Design*, 18(10):514–524, 1986.

[30] Michael J Pratt and AD Geisow. Surface/surface intersection problems. *The mathematics of surfaces*, 6:117–142, 1986.

[31] Rizzi C Radi S. A system for parametric surface intersection. In *INSTITUTE OF MATHEMATICS AND ITS APPLICATIONS CONFERENCE SERIES*, volume 48, pages 231–231. OXFORD UNIVERSITY PRESS, 1994.

[32] Jaroslaw R Rossignac and Aristides AG Requicha. Piecewise-circular curves for geometric modeling. *IBM Journal of Research and Development*, 31(3):296–313, 1987.

[33] Thomas W Sederberg. Planar piecewise algebraic curves. *Computer Aided Geometric Design*, 1(3):241–255, 1984.

[34] Jianrong Tan, Jianmin Zheng, and Qunsheng Peng. A unified algorithm for finding the intersection curve of surfaces. *Journal of Computer Science and Technology*, 9(2):107–116, 1994.

[35] Itzhak Wilf and Yehuda Manor. Quadric-surface intersection curves: shape and structure. *Computer-Aided Design*, 25(10):633–643, 1993.