Mecánica Computacional Vol. XXI, pp. 3086–3098
S. R. Idelsohn, V. E. Sonzogni and A. Cardona (Eds.)
Santa Fe–Paraná, Argentina, October 2002

CONTROL OF HERMITE-INTERPOLATION ERROR IN SURFACE INTERSECTION

José M. Olivencia[*] **and Luis E. Quiroz**[†]

[*] Departamento de Ingeniería Matemática
Universidad de Concepción
Casilla 160-C, Concepción, Chile
e-mail: jolivenc@ing-mat.udec.cl, web page: http://www.ing-mat.udec.cl

[†] Departamento de Ingeniería Mecánica
Universidad de Concepción
Casilla 53-C, Concepción, Chile
e-mail: lquiroz@udec.cl, web page: http://www.udec.cl

Key words: surface intersection, marching method, Hermite-interpolation.

Abstract. *In computer aided manufacturing is necessary to have a numerical method capable to obtain an approximation of the surface intersection curve inside a prescribed error tolerance and with the lowest possible number of interpolation points. The marching methods are the most widely used to compute the intersection curve of two surfaces. A marching method builds a sequence of points that belong to the intersection curve. The resulting intersection points are interpolated and an approximation of the intersection is obtained. A method to control the error of a linear interpolation process in order to satisfy the prescribed tolerance has been done, first for a constant step size and then for an adaptive step size, making use of the normal curvatures and the osculating circle of the intersection curve on each surface. Using the local differential geometry of both surfaces and their intersection, we show in this work, the extension of these ideas to the general case of Hermite-interpolation, making use of the subdivision property of polynomial Bézier curves with the de Casteljau algorithm. Thus, we get a marching method with an appropriate adaptive step size and with an effective and efficient control of Hermite-interpolation error. Numerical experiences in practical cases are presented to show the properties of the method.*

1 INTRODUCTION

The problem of describing the intersection between two surfaces arises often in mechanical engineering applications. For example, to model solids bounded by trimmed surfaces is important to determine the surface intersection curves and in manufacturing the tool path is calculated from the boundary representation of the geometric model composed by surfaces and their intersections.

In general, there is no close form representation of surface intersection. Therefore, only approximating methods are able to provide the intersection calculation. Numerical marching methods are the most widely used to compute the intersection curves. The pre-processing step of a marching method must serve to detect points on each one of the components of the intersection. Beginning at these points, the method must build a sequence of points that belong to the intersection curve[1,2]. The resulting intersection points will be interpolated later, so an approximation of the intersection is generated. Manufacturing applicacions make it necesary to specify a tolerance so that the intersection curve will always be closer than the tolerance from the approximation generated, for all points on the intersection curve.

We propose a marching method that provides a Hermite interpolant that approximates the intersection of two surfaces, with an error inside a prescribed tolerance and with few interpolation points. By assuming that the pre-processing step has already been succesfully performed, our method generates at first a polygonal whose vertices are on the intersection and whose maximal deviation from the real intersection is less than a prescribed tolerance. For an optimal machining process it is desirable to reduce the number of interpolation points, the vertices of the polygonal generated before. By using the differential geometry of the two surfaces intersection and taking into account properties of the polynomial Bézier curves, one achieves the desirable reduction to extend the usual linear interpolation and building a Hermite interpolation with adaptive step size and remaining inside the prescribed tolerance.

2 CONTROL OF LINEAR INTERPOLATION ERROR

In this section a method is proposed to estimate and control the error of the linear interpolation process of surface intersection. Without loss generality, we can consider that the intersection curve is parametrized by arc length. Making use of the normal curvatures of the intersection curve on each one of the intersecting surfaces, we can get an upper bound for the curvature of the intersection curve.

Thus, let $\mathbf{c}(s)$ be the 3D intersection curve parametrized by arc length s, defined on $[s_i, s_i + \Delta s_i]$ and $\mathbf{p}(s)$ the 3D linear polynomial, defined on $[s_i, s_i + \Delta s_i]$ such that

$$\mathbf{p}(s_i) = \mathbf{c}(s_i),\ \mathbf{p}(s_i + \Delta s_i) = \mathbf{c}(s_i + \Delta s_i)\,. \tag{1}$$

It is known[3] that the interpolation error of $\mathbf{p}(s)$ in s is given by

$$\mathbf{c}(s) - \mathbf{p}(s) = \frac{1}{2}(s - s_i)(s - s_i - \Delta s_i)\bar{\mathbf{c}}^{(2)} \tag{2}$$

where $\bar{\mathbf{c}}^{(2)} = \left(c_1^{(2)}(\xi_1), c_2^{(2)}(\xi_2), c_3^{(2)}(\xi_3)\right)$ with ξ_j, $j = 1,2,3$ (in general unknown) on $[s_i, s_i + \Delta s_i]$. A straightforward calculation shows that

$$\left|\mathbf{c}(s) - \mathbf{p}(s)\right| \leq \frac{3}{8}\max\left|\mathbf{c}^{(2)}(s)\right|(\Delta s_i)^2 = \frac{3}{8}\max \kappa(s)(\Delta s_i)^2 \tag{3}$$

The curvature, κ, of the intersection curve, $\mathbf{c}$, of the two surfaces $\mathbf{S}^1$ y $\mathbf{S}^2$, can be expressed in terms of the normal curvatures, $\kappa_{\mathbf{n}1}^1$ y $\kappa_{\mathbf{n}1}^2$ of the curve on $\mathbf{S}^1$ and $\mathbf{S}^2$, respectively, with the unit normals, $\mathbf{N}^1$ y $\mathbf{N}^2$, to the two surfaces. In fact, by considering the Frenet system[4] $(\mathbf{t}, \mathbf{n}, \mathbf{b})$ of the curve, we have

$$\mathbf{t} = \pm\frac{\mathbf{N}^1 \times \mathbf{N}^2}{\left|\mathbf{N}^1 \times \mathbf{N}^2\right|} \tag{4}$$

As $\mathbf{b} = -\mathbf{n} \times \mathbf{t}$, we have, expanding the triple vector product,

$$k\mathbf{b} = \mp\frac{\kappa\mathbf{n} \times (\mathbf{N}^1 \times \mathbf{N}^2)}{\left|\mathbf{N}^1 \times \mathbf{N}^2\right|} = \pm\frac{\kappa_{\mathbf{n}1}^1\mathbf{N}^2 - \kappa_{\mathbf{n}1}^2\mathbf{N}^1}{\left|\mathbf{N}^1 \times \mathbf{N}^2\right|} \tag{5}$$

Therefore,

$$k = \frac{\left|\kappa_{\mathbf{n}1}^1\mathbf{N}^2 - \kappa_{\mathbf{n}1}^2\mathbf{N}^1\right|}{\left|\mathbf{N}^1 \times \mathbf{N}^2\right|} \leq \frac{\left|\kappa_{\mathbf{n}1}^1\right| + \left|\kappa_{\mathbf{n}1}^2\right|}{\left|\mathbf{N}^1 \times \mathbf{N}^2\right|} \leq \frac{\left|\kappa^1\right| + \left|\kappa^2\right|}{\sin\theta}, \tag{6}$$

since each one of the absolute values of the normal curvatures, $\left|\kappa_{\mathbf{n}1}^i\right|$, can be upper bounded by the maximum of the absolute values of the principal curvatures of the surface $\mathbf{S}^i$, $\left|\kappa^i\right|$. If the surfaces are not tangential, i.e. if $\sin\theta \geq c > 0$, for some $c > 0$, we have

$$\left|\mathbf{c}(s) - \mathbf{p}(s)\right| \leq \frac{3}{8}\frac{\left|\kappa^1\right| + \left|\kappa^2\right|}{c}(\Delta s_i)^2 = \frac{3}{8}\bar{\kappa}(\Delta s_i)^2, \tag{7}$$

that provides a safe way to estimate the error of the linear interpolation process of the two surfaces intersection curve.

With the estimation obtained above, we can propose now a safe method to generate points of the intersection, which being interpolated later, will provide a polygonal that approximates the intersection with a given tolerance.

Let ε be an error tolerance, small enough, given, for example, by manufacturing criteria. We find intersection points of the surfaces separated by a step size Δs constant and small enough, such that

$$\Delta s \leq \sqrt{\frac{8}{3}\frac{\varepsilon}{\bar{\kappa}}}. \tag{8}$$

We can imagine a circular segment with radius $1/\bar{\kappa}$ from point $\mathbf{c}(s)$ on the intersection curve $\mathbf{c}$ tangential to $\mathbf{c}$ at $\mathbf{c}(s)$. We take the length of this circular segment as $\bar{l}$, where

$$\bar{l} \leq \sqrt{\frac{8}{3}\frac{\varepsilon}{\bar{\kappa}}} \text{ and } \bar{l} < \frac{\pi}{2\bar{\kappa}}. \tag{9}$$

As $\bar{\kappa}\bar{l} < \pi/2$, we can choose real numbers K and L, such that $K > \bar{\kappa}$, $L \leq \bar{l}$ and $KL = \pi/2$. Thus, we have a circular quadrant with radius $1/K$ and length L from point from point $\mathbf{c}(s)$ in the intersection curve $\mathbf{c}$ tangential to $\mathbf{c}$ at $\mathbf{c}(s)$, as can be seen in Figure 1.

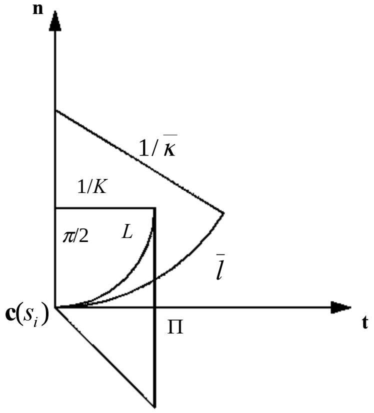

Figure 1: Control of linear interpolation error

If we intersect the plane Π defined by $(\mathbf{P} - \mathbf{c}(s)) \cdot \mathbf{c}^{(1)}(s) = 1/R$ where $R \geq K$, with $\mathbf{c}$, this procedure determines a polygonal which approximates the intersection curve with a safe error tolerance not greater than ε, whose vertices are on the intersection[5].

3 ADAPTIVE HERMITE INTERPOLATION

For an optimal machinig process, it is possible to reduce the number of interpolation points that participate in the building of the precedent polygonal, in order to obtain a Hermite interpolant with adaptive step size and inside of the given tolerance. If $\mathbf{p}(s)$ is the 3D Hermite polynomial, defined on $[s_i, s_i + \Delta s_i]$ such that

$$\mathbf{p}^{(j)}(s_i) = \mathbf{c}^{(j)}(s_i),\ \mathbf{p}^{(j)}(s_i + \Delta s_i) = \mathbf{c}^{(j)}(s_i + \Delta s_i),\ j = 0(1)k, \tag{10}$$

where $\rho = 2k+1$ is the degree of the polynomial $\mathbf{p}$. It is known[3], that the interpolation error of $\mathbf{p}(s)$ in s is given by,

$$\mathbf{c}(s) - \mathbf{p}(s) = \frac{1}{(\rho+1)!}(s-s_i)^k (s-s_i-\Delta s_i)^k \bar{\mathbf{c}}^{(\rho+1)}, \tag{11}$$

where $\bar{\mathbf{c}}^{(\rho+1)} = \left(c_1^{(\rho+1)}(\xi_1), c_2^{(\rho+1)}(\xi_2), c_3^{(\rho+1)}(\xi_3)\right)$, with ξ_j, $j = 1,2,3$ (in general unknown) on $[s_i, s_i + \Delta s_i]$. We have that

$$\left|\mathbf{c}(s) - p(s)\right| \le \frac{3}{(\rho+1)!(\rho+1)^2} \max\left|\mathbf{c}^{(\rho+1)}(s)\right|(\Delta s_i)^{\rho+1}. \tag{12}$$

Given ε an error tolerance, taking $\varepsilon_c < \varepsilon$, we find, firstly, a polygonal that approximates the intersection curve with an error tolerance not greater than ε_c, whose vertices are on the intersection. Then, we select, from the intersection points found before, a sequence of points belonging to the intersection according an adaptive step size given by Δs_i, where, for example,

$$\Delta s_i \le \sqrt[\rho+1]{\frac{(\rho+1)!(\rho+1)^2}{3}\frac{\varepsilon-\varepsilon_c}{\left|\mathbf{c}^{(\rho+1)}\right|_i}}. \tag{13}$$

A good approximation of Δs_i is given adding, from the start point $\mathbf{c}(s_i)$, the lengths of consecutive segments of the polygonal found before. Using the subdivision property of the polynomial Bézier curves, which can be carry out efficiently with the de Casteljau algorithm[3], it is calculated, in each step, the distance from the segment of polynomial Bézier curve, $\mathbf{p}(s)$, defined on $[s_i, s_i + \Delta s_i]$, such that $\mathbf{p}^{(j)}(s_i) = \mathbf{c}^{(j)}(s_i)$, $\mathbf{p}^{(j)}(s_i + \Delta s_i) = \mathbf{c}^{(j)}(s_i + \Delta s_i)$, $j = 0(1)k$, to the corresponding portion of the polygonal obtained before. If this distance is greater than $\varepsilon - \varepsilon_c$, which could happen if Δs is not small enough, we take a smaller Δs and we do again the procedure until the distance is not greater than $\varepsilon - \varepsilon_c$. This will be a reliable Δs, in order to control the error by ε and with an adaptive step size. There are several ways to choose this initial step size Δs_i. Since the aim is to have the lowest possible number of interpolation points, could be, to take at first Δs_i as large as the whole segment.

The knowledge of $\mathbf{c}^{(j)}(s_i)$and$\mathbf{c}^{(j)}(s_i + \Delta s_i)$, for $j = 0(1)k$ allows to determine the polynomial Bézier curve interpolant,

$$\mathbf{p}(s) = \sum_{j=0}^{\rho} B_j^{\rho}(s)\mathbf{b}_j, \quad s \in [s_i, s_i + \Delta s_i]. \tag{14}$$

$$B_j^\rho(s) := \binom{\rho}{j}\frac{(s_i+\Delta s_i - s)^{\rho-j}(s-s_i)^j}{(\Delta s_i)^\rho},\ s\in[s_i, s_i+\Delta s_i],\ j=0(1)\rho, \tag{15}$$

are the $\rho+1$ Bernstein polynomials of degree ρ in s on $[s_i, s_i+\Delta s_i]$. The $\rho+1$ coefficients $\mathbf{b}_j$, $j=0(1)\rho$, are the Bézier points, that are found as the solutions of the system of equations, for $j=0(1)k$,

$$\mathbf{c}^{(j)}(s_i) = (\Delta s_i)^{-j}\frac{\rho!}{(\rho-j)!}(E-1)^j\mathbf{b}_0,\ \mathbf{c}^{(j)}(s_i+\Delta s_i) = (\Delta s_i)^{-j}\frac{\rho!}{(\rho-j)!}(E-1)^j\mathbf{b}_{\rho-j}, \tag{16}$$

where $E\mathbf{b}_i = \mathbf{b}_{i+1}$, $E^mE^n = E^{m+n}$, $Ea=aE$, $E^0=1$, $EE^{-1}=1$.

It is perceived the necessity to know the local differential geometry of the intersección of two surfaces. Specificaly, if one wishes a cubic interpolant, one must know to calculate the fourth-order derivative, at points of the intersection curve. This has been discussed by Ye et al.[6]. For the sake of completeness, in the next secction it is showed how to proceed for calculating these local differential geometry properties of the intersection between two surfaces.

4 DIFFERENTIAL GEOMETRY OF INTERSECTION CURVE

Given a 3D curve $\mathbf{c}=\mathbf{c}(s)$ parametrized by arc length s, from differential geometry of curves[2], we have that $\mathbf{c}^{(1)}=\mathbf{t}$ is the unit tangent vector of $\mathbf{c}(s)$ in s and $\mathbf{c}^{(2)}=\kappa\mathbf{n}$, where $\mathbf{n}$ and κ are the unit normal vector and the curvature, respectively, of $\mathbf{c}(s)$ in s. The unit binormal vector $\mathbf{b}=\mathbf{t}\times\mathbf{n}$ is such that $\mathbf{b}^{(1)}=\tau\mathbf{n}$, being τ the torsion of $\mathbf{c}(s)$ in s. The trihedron $(\mathbf{t},\mathbf{n},\mathbf{b})$, constitutes an orthonormal ordered basis positively oriented, the Frenet basis, which satisfy the system of differential equations, the Frenet equations,

$$\begin{cases}\mathbf{t}^{(1)} = \kappa\mathbf{n}\\ \mathbf{n}^{(1)} = -\kappa t - \tau\mathbf{b}\\ \mathbf{b}^{(1)} = \tau\mathbf{n}\end{cases}. \tag{17}$$

By differentiation with respect to s and expressing in terms of the Frenet basis we have

$$\mathbf{c}^{(3)} = -\kappa^2\mathbf{t} + \kappa^{(1)}\mathbf{n} - \kappa\tau\mathbf{b}, \tag{18}$$

$$\mathbf{c}^{(4)} = -3\kappa\kappa^{(1)}\mathbf{t} + \left(\kappa^{(2)} - \kappa\tau^2 - \kappa^3\right)\mathbf{n} + \left(-2\kappa^{(1)}\tau - \kappa\tau^{(1)}\right)\mathbf{b}, \tag{19}$$

and so on. Ye et al.[6] propose the following procedure to calculate the differential geometry of the intersection curve of two surfaces: Given the intersecting surfaces $\mathbf{S}^1(u_1,v_1)$ y $\mathbf{S}^2(u_2,v_2)$,

$$\mathbf{t} = \pm \frac{\mathbf{N}^1 \times \mathbf{N}^2}{\left|\mathbf{N}^1 \times \mathbf{N}^2\right|} \tag{20}$$

is the unit tangent vector of the intersection curve, the sign depends of the tracing sense. Naturally $\mathbf{N}^1 \cdot \mathbf{N}^2 = \cos\theta$, being θ the angle that form the unit normal vectors $\mathbf{N}^1$, $\mathbf{N}^2$, of the surfaces $\mathbf{S}^1$, $\mathbf{S}^2$, respectively. Moreover, $b\mathbf{n} + c\mathbf{b} = \alpha\mathbf{N}^1 + \beta\mathbf{N}^2$. Using the scalar product of the first derivative of $\mathbf{c}$ as a curve on the surface $\mathbf{S}^i$ by $\mathbf{S}^i_u$, $\mathbf{S}^i_v$ respectively, we obtain

$$\begin{cases} E^i u_i^{(1)} + F^i v_i^{(1)} = \mathbf{S}^i_u \cdot \mathbf{c}^{(1)} \\ F^i u_i^{(1)} + G^i v_i^{(1)} = \mathbf{S}^i_v \cdot \mathbf{c}^{(1)} \end{cases}, \tag{21}$$

thus, we have $u_i^{(1)}$, $v_i^{(1)}$. Using the scalar product of the second derivative of $\mathbf{c}$ as a curve on the surface $\mathbf{S}^i$ by $\mathbf{N}^i$, we obtain

$$\kappa^i_{\mathbf{n}1} = \mathbf{c}^{(2)} \cdot \mathbf{N}^i = e^i (u_i^{(1)})^2 + 2f^i u_i^{(1)} v_i^{(1)} + g^i (v_i^{(1)})^2. \tag{22}$$

Since, $\mathbf{c}^{(2)} = \kappa\mathbf{n} = \alpha_1 \mathbf{N}^1 + \beta_1 \mathbf{N}^2$, we have that

$$\begin{cases} \alpha_1 + (\cos\theta)\beta_1 = \mathbf{c}^{(2)} \cdot \mathbf{N}^{(1)} = \kappa^1_{\mathbf{n}1} \\ (\cos\theta)\alpha_1 + \beta_1 = \mathbf{c}^{(2)} \cdot \mathbf{N}^{(2)} = \kappa^2_{\mathbf{n}1} \end{cases}. \tag{23}$$

So,

$$\mathbf{c}^{(2)} = \frac{\kappa^1_{\mathbf{n}1} - \kappa^2_{\mathbf{n}1}\cos\theta}{\sin^2\theta}\mathbf{N}^1 + \frac{\kappa^2_{\mathbf{n}1} - \kappa^1_{\mathbf{n}1}\cos\theta}{\sin^2\theta}\mathbf{N}^2 \text{ and } \kappa = \left|\mathbf{c}^{(2)}\right|. \tag{24}$$

Having calculated $\mathbf{t}$, $\mathbf{c}^{(2)}$ and κ, it is now possible to calculate

$$\mathbf{n} = \frac{\mathbf{c}^{(2)}}{\kappa} \text{ and } \mathbf{b} = \mathbf{t} \times \mathbf{n}. \tag{25}$$

Using the scalar product of the second derivative of $\mathbf{c}$ as a curve on the surface $\mathbf{S}^i$ by $\mathbf{S}^i_u$, $\mathbf{S}^i_v$ respectively, we obtain

$$\begin{cases} E^i u_i^{(2)} + F^i v_i^{(2)} = \mathbf{S}^i_u \cdot \mathbf{c}^{(2)} - \dfrac{E^i_u}{2}(u_i^{(1)})^2 - E^i_v u_i^{(1)} v_i^{(1)} - \left(F^i_v - \dfrac{G^i_u}{2}\right)(v_i^{(1)})^2 \\ F^i u_i^{(2)} + G^i v_i^{(2)} = \mathbf{S}^i_v \cdot \mathbf{c}^{(2)} - \left(F^i_u - \dfrac{E^i_v}{2}\right)(u_i^{(1)})^2 - G^i_u u_i^{(1)} v_i^{(1)} - \dfrac{G^i_v}{2}(v_i^{(1)})^2 \end{cases}, \tag{26}$$

thus, we have $u_i^{(2)}$, $v_i^{(2)}$. Using the scalar product of the third derivative of $\mathbf{c}$ as a curve on the surface $\mathbf{S}^i$ by $\mathbf{N}^i$, we obtain

$$\kappa_{\mathbf{n}2}^i = \mathbf{c}^{(3)} \cdot \mathbf{N}^i = 3\left(e^i u_i^{(1)} u_i^{(2)} + f^i (u_i^{(2)} v_i^{(1)} + u_i^{(1)} v_i^{(2)}) + g^i v_i^{(1)} v_i^{(2)}\right) + \mathbf{S}_{uuu}^i \cdot \mathbf{N}^i (u_i^{(1)})^3 + 3\mathbf{S}_{uuv}^i \cdot \mathbf{N}^i (u_i^{(1)})^2 v_i^{(1)} + 3\mathbf{S}_{uvv}^i \cdot \mathbf{N}^i u_i^{(1)} (v_i^{(1)})^2 + \mathbf{S}_{vvv}^i \cdot \mathbf{N}^i (v_i^{(1)})^3. \tag{27}$$

Since, $\mathbf{c}^{(3)} = -\kappa^2 \mathbf{t} + \kappa^{(1)} \mathbf{n} + \kappa\tau\mathbf{b} = -\kappa^2 \mathbf{t} + \alpha_2 \mathbf{N}^1 + \beta_2 \mathbf{N}^2$, we have that

$$\begin{cases} \alpha_2 + (\cos\theta)\beta_2 = \mathbf{c}^{(3)} \cdot \mathbf{N}^{(1)} = \kappa_{\mathbf{n}2}^1 \\ (\cos\theta)\alpha_2 + \beta_2 = \mathbf{c}^{(3)} \cdot \mathbf{N}^{(2)} = \kappa_{\mathbf{n}2}^2 \end{cases}. \tag{28}$$

So,

$$\mathbf{c}^{(3)} = -\kappa^2 \mathbf{t} + \frac{\kappa_{\mathbf{n}2}^1 - \kappa_{\mathbf{n}2}^2 \cos\theta}{\sin^2\theta} \mathbf{N}^1 + \frac{\kappa_{\mathbf{n}2}^2 - \kappa_{\mathbf{n}2}^1 \cos\theta}{\sin^2\theta} \mathbf{N}^2. \tag{29}$$

Now, it is possible to calculate

$$\tau = \frac{\mathbf{c}^{(3)} \cdot \mathbf{b}}{\kappa} \text{ and } \kappa^{(1)} = \mathbf{c}^{(3)} \cdot \mathbf{n}. \tag{30}$$

Thus, we can continue the recursive process in this way.

5 NUMERICAL EXPERIENCES

In this section we show the application of the proposed method, first in the calculation of the intersection curve of two tensor product Bézier surfaces with a prescribed error tolerance. One of the surfaces is given by the control points

$\mathbf{b}_{0,0}$=(0,0,0),	$\mathbf{b}_{1,0}$=(1,0,1),	$\mathbf{b}_{2,0}$=(2,0,1),	$\mathbf{b}_{3,0}$=(3,0,0),
$\mathbf{b}_{0,1}$=(0,1,1),	$\mathbf{b}_{1,1}$=(1,1,0),	$\mathbf{b}_{2,1}$=(2,1,0),	$\mathbf{b}_{3,1}$=(3,1,1),
$\mathbf{b}_{0,2}$=(0,2,1),	$\mathbf{b}_{1,2}$=(1,2,0),	$\mathbf{b}_{2,2}$=(2,2,0),	$\mathbf{b}_{3,2}$=(3,2,1),
$\mathbf{b}_{0,3}$=(0,3,0),	$\mathbf{b}_{1,3}$=(1,3,1),	$\mathbf{b}_{2,3}$=(2,3,1),	$\mathbf{b}_{3,3}$=(3,3,0),

and the another Bézier surface is given by the control points

$\mathbf{b}_{0,0}$=(-1,-1,-1),	$\mathbf{b}_{1,0}$=(1,-1,-1),	$\mathbf{b}_{2,0}$=(2,-1,-1),	$\mathbf{b}_{3,0}$=(4,-1, 1),
$\mathbf{b}_{0,1}$=(-1, 1,-1),	$\mathbf{b}_{1,1}$=(1, 1, 2),	$\mathbf{b}_{2,1}$=(2, 1, 2),	$\mathbf{b}_{3,1}$=(4, 1,-1),
$\mathbf{b}_{0,2}$=(-1, 2,-1),	$\mathbf{b}_{1,2}$=(1, 2, 2),	$\mathbf{b}_{2,2}$=(2, 2, 2),	$\mathbf{b}_{3,2}$=(4, 2,-1),
$\mathbf{b}_{0,3}$=(-1, 4, 1),	$\mathbf{b}_{1,3}$=(1, 4,-1),	$\mathbf{b}_{2,3}$=(2, 4,-1),	$\mathbf{b}_{3,3}$=(4, 4, 1).

The intersection of the considered surfaces will be composed by five components, a closed curve and four open curves in the corners of one of the surfaces, as is rendered in the Figure 2.

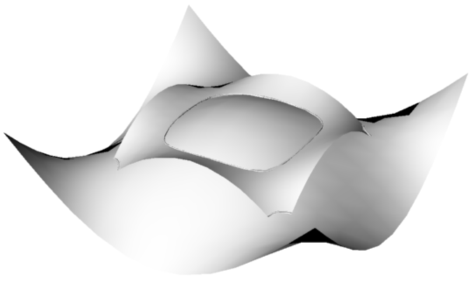

Figure 2: Intersection of two Bézier surfaces

To approximate the intersection, the interpolation process is performed, with the proposed method, as follows. The maximum value of the absolute value of the principal curvatures of the surfaces $\mathbf{S}^1$ and $\mathbf{S}^2$ is $|k^1| = 1.1233$ and $|k^2| = 0.8779$ respectively. The intersection angle between the surfaces is not less than 20^o. Since the surface intersection is not tangential, this can be locally achieved by recursively subdivision of the intersecting surfaces. Now, let $\varepsilon = 0.05$ the allowed error tolerance. Taking, $\varepsilon_c \ll \varepsilon$, we have, for the first step of the proposed method, a constant step size and small enough as to guarantee the convergence of the involved Newton-Raphson iteration. This first step yields a closed polygonal and four congruent open polygonals, that approximate the intersection with a safe error tolerance not greater than ε_c.

For adaptive cubic interpolation, the second step of the proposed method can use naturally an adaptive step size given by Δs_i as large as the remainder of the actual segment. This second step provides 8 interpolation points for the closed component of the intersection and 2 interpolation points for each one of the four open components (see Figure 3.a). By joining with cubic Bézier curves each pair of consecutive points in each one of the components of the intersection (see magnifications in Figures 3.b and 3.c), we obtain a closed cubic piecewise interpolant and four open cubic piecewise interpolants whose distance to the composed polygonal found in the first step is not greater than ε - ε_c. Thus, we obtain a confident adaptive cubic interpolant (see Figure 3.d), that remains inside the prescribed error tolerance, $\varepsilon = 0.05$. The Figure 4 shows the behavior of the error of the open interpolants and the Figure 5, the behavior of the error of the closed interpolant.

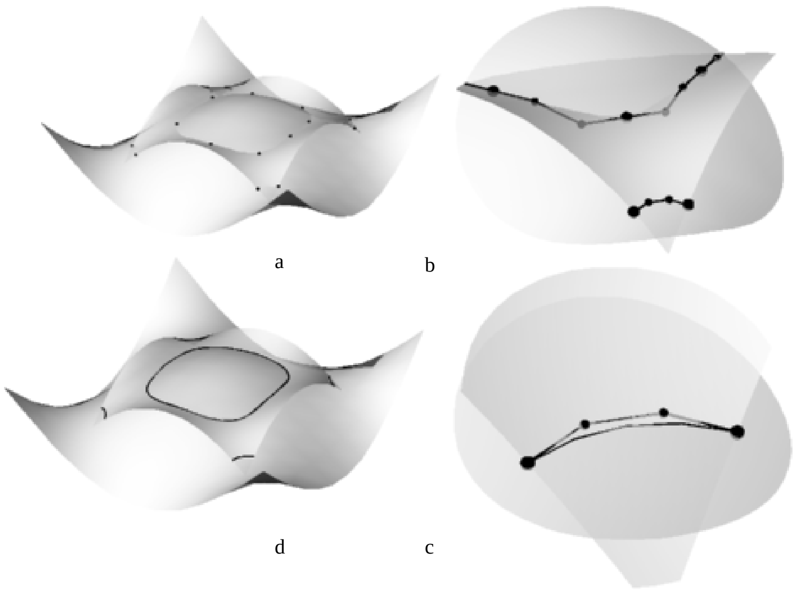

Figure 3: Adaptive cubic interpolation

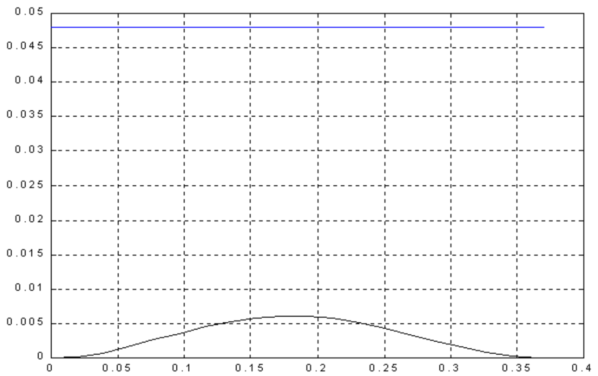

Figure 4: Graphic of the errors of the adaptive cubic interpolation (one of the open components)

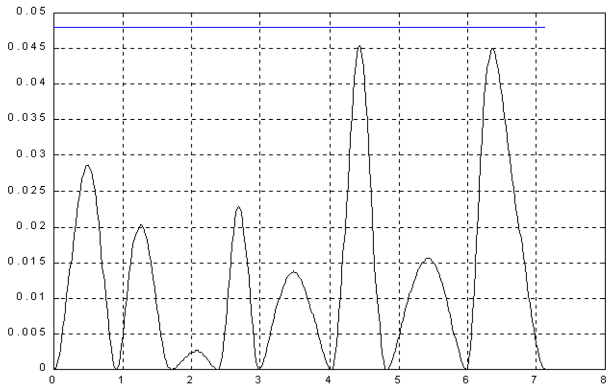

Figure 5: Graphic of the errors of the adaptive cubic interpolation (the closed component)

We show now the application of the proposed method in the calculation of the intersection curve of two tensor product Bézier surfaces with a prescribed error tolerance.
Now, one of the surfaces is given by the control points

$\mathbf{b}_{0,0}$=(-1, 1,0), $\mathbf{b}_{1,0}$=(-3, 1,10), $\mathbf{b}_{2,0}$=(3, 1,10), $\mathbf{b}_{3,0}$=(1, 1, 0),
$\mathbf{b}_{0,1}$=(-1, 6,0), $\mathbf{b}_{1,1}$=(-3, 6,10), $\mathbf{b}_{2,1}$=(3, 6,10), $\mathbf{b}_{3,1}$=(1, 6, 0),
$\mathbf{b}_{0,2}$=(-1,11,0), $\mathbf{b}_{1,2}$=(-3,11,10), $\mathbf{b}_{2,2}$=(3,11,10), $\mathbf{b}_{3,2}$=(1,11,10),
$\mathbf{b}_{0,3}$=(-1,16,0), $\mathbf{b}_{1,3}$=(-3,16,10), $\mathbf{b}_{2,3}$=(3,16,10), $\mathbf{b}_{3,3}$=(1,16, 0),

and the another Bézier surface is given by the control points

$\mathbf{b}_{0,0}$=(-1,0, -4), $\mathbf{b}_{1,0}$=(-6,6,-4), $\mathbf{b}_{2,0}$=(6,6, -4), $\mathbf{b}_{3,0}$=(6,0, -4),
$\mathbf{b}_{0,1}$=(-6,0, 0), $\mathbf{b}_{1,1}$=(-6,6, 0), $\mathbf{b}_{2,1}$=(6,6, 0), $\mathbf{b}_{3,1}$=(6,0, 0),
$\mathbf{b}_{0,2}$=(-6,0, 5), $\mathbf{b}_{1,2}$=(-6,6, 5), $\mathbf{b}_{2,2}$=(6,6, 5), $\mathbf{b}_{3,2}$=(6,0, 5),
$\mathbf{b}_{0,3}$=(-6,0,10), $\mathbf{b}_{1,3}$=(-6,6,10), $\mathbf{b}_{2,3}$=(6,6,10), $\mathbf{b}_{3,3}$=(6,0,10).

The intersection of the considered surfaces will be composed by one open component.

To approximate the intersection, the interpolation process is performed, with the proposed method, as follows. The maximum value of the absolute value of the principal curvatures of the surfaces $\mathbf{S}^1$ and $\mathbf{S}^2$ can be estimated as $\left|k^1\right| = 1.667$ and $\left|k^2\right| = 0.271$ respectively. The intersection angle between the surfaces is not less than 80^{o}. Now, let $\varepsilon = 0.1$ the allowed error tolerance. Taking, $\varepsilon_c \ll \varepsilon$, we have, for the first step of the proposed method, a constant step size and small enough as to guarantee the convergence of the involved Newton-Raphson iteration. This first step yields a closed polygonal and four congruent open polygonals, that approximate the intersection with a safe error tolerance not greater than ε_c.

For adaptive cubic interpolation, the second step of the proposed method can use naturally an adaptive step size given by Δs_i as large as the remainder of the actual segment. This second step provides 5 interpolation points (see Figure 6). By joining with cubic Bézier curves each pair of consecutive points (see Figures 6), we obtain a cubic piecewise interpolant whose distance to the composed polygonal found in the first step is not greater than ε - ε_c. Thus, we obtain a confident adaptive cubic interpolant (see Figure 6), that remains inside the prescribed error tolerance, $\varepsilon = 0.1$. The Figure 7 shows the behavior of the error of the interpolant.

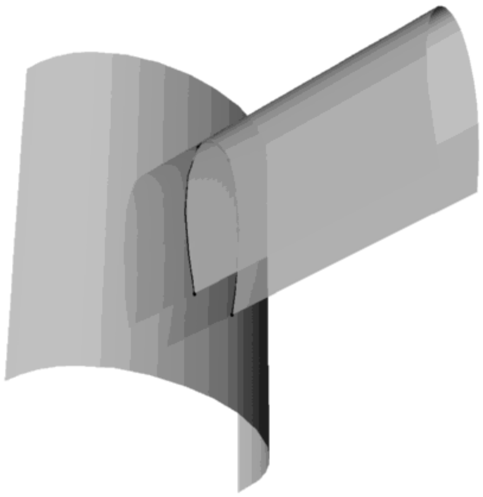

Figure 6: Adaptive cubic interpolation

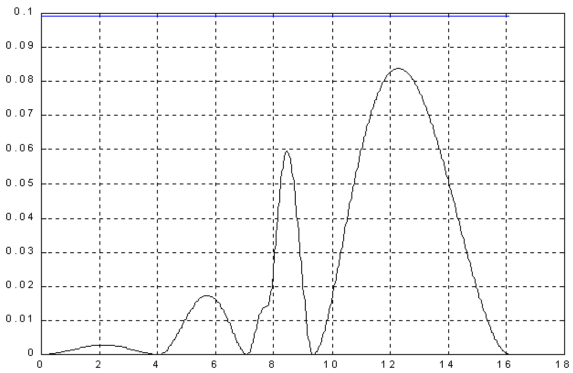

Figure 7: Graphic of the errors of the adaptive cubic interpolation

6 CONCLUSIONS

The proposed method allows to control and adapt the step size, to get a prescribed error tolerance, distributing the interpolation points in a suitable way. Since the method contemplates an a posteriori control of the approximation error, it is possible to choose the initial step size in several ways. For example relaxing, by a factor 3, the step size in the second step of the algorithm, we obtain good results, diminishing the number of interpolation points. The required information to control the approximation error is the local differential geometry of the intersection curve. This requires that the involved surfaces are differentiable enough. Although in the numerical experiences we used cubic interpolants, it is clear that higher degree interpolants can be used.

7 REFERENCES

[1] J. J. Chen and T. M. Ozsoy, “Predictor-corrector type of intersection algorithm for C^2 parametric surfaces”, *Computer-Aided Design*, **20**, (6), 347-352 (1988).

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

[3] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A. K. Peters Ltd., Wellesley, Massachusetts (1993).

[4] M. do Carmo, *Differential Geometry of Curves and Surfaces,* Prentice Hall Inc., Englewood Cliffs, New Jersey (1976).

[5] J. Olivencia, L. Quiroz, and P. Beckers, “Estimation and control of linear interpolation error of free-form surface intersection”, Preprint, (2001).

[6] X. Ye and T. Maekawa, “Differential geometry of intersection curves of two surfaces”, *Computer Aided Geometric Design*, **16**, 767-788 (1999).