P.V. Prakash
N.M. Patrikalakis

Surface-to-Surface Intersections for Geometric Modeling

MIT Sea Grant
College Program

Massachusetts
Institute of Technology
Cambridge, MA 02139

MITSG 88-8
December 1988

SURFACE-TO-SURFACE INTERSECTIONS FOR GEOMETRIC MODELING

b y

P.V. Prakash

N.M. Patrikalakis

MIT Sea Grant
College Program

Massachusetts Institute
of Technology
77 Massachusetts Ave.
Cambridge, MA 02139

MITSG 88-8
December 1988
NA86AA-D-SG089
R/T 22 $6.00

Table of Contents

List of Figures

List of Tables

Abstract

In this work, we address the problem of intersecting algebraic surfaces with piecewise continuous rational polynomial parametric surface patches, such as B-splines. Such problems can be reduced to tracing a planar algebraic curve within a rectangular domain. Our method combines the advantageous features of analytic representation of the governing equation of the algebraic curve in the Bernstein basis within a rectangular domain, adaptive subdivision and polyhedral faceting techniques, and the computation of turning and singular points, to provide the basis for a reliable solution procedure. This representation transforms the problem of intersection of two curved polynomial surfaces to the intersection of a Bezier surface with a plane. Such transformation also allows computation of turning and singular points of the curve, using a hybrid method based on minimization and Newton techniques as well as Bernstein subdivision. Using turning and singular points, the intersection problem can be partitioned into subdomains that can be processed independently and which involve intersection segments that can be traced with faceting methods. This partitioning and the tracing of individual segments is carried out using an adaptive subdivision algorithm for Bezier/B-spline surfaces followed by Newton correction of the approximation. The method has been successfully tested in tracing complex algebraic curves and in solving actual intersection problems with diverse features.

Acknowledgments

This work was supported in part by the MIT Sea Grant College Program, the General Electric Company and the National Science Foundation under grant numbers DMC-8706592 and DMC-8720720. Mr. G. A. Kriezis assisted us in the preparation of this report.

Authors

Dr. P. V. Prakash is a Senior Software Engineer in the CAD/CAM development department at Prime Computer Inc. This work was performed while he was a doctoral graduate student in the Department of Ocean Engineering at M. I. T. and a member of the Design Laboratory.

Dr. Nicholas M. Patrikalakis is Assistant Professor of Ocean Engineering at M. I. T., Doherty Professor of Ocean Utilization (1988-1990) and a member of the M. I. T. Ocean Engineering Design Laboratory.

Related Sea Grant Reports

Computation of Algebraic and Polynomial Parametric Surface Intersections, by N. M. Patrikalakis and P. V. Prakash, MIT Sea Grant Report No: 87-19, Cambridge, MA, 1987.

Piecewise Continuous Algebraic Surfaces in Terms of B-splines, by N. M. Patrikalakis and G. A. Kriezis, MIT Sea Grant Report No: 88-5, Cambridge, MA, 1988.

1 Introduction

Interrogation of intersections of surfaces is a fundamental problem in geometric modeling of complex shapes in a computer environment. For anything but the simplest artifacts, objects must be combined to increase complexity. Such a combination of objects involves the use of Boolean operations, where the bounding surfaces of the primitive objects are intersected with each other to determine the surface geometry of the new objects. This process, called boundary evaluation, is also used in the conversion of representations of solids using a Constructive Solid Geometry scheme to a Boundary Representation scheme [31]. The fundamental step involved in such a process is to determine if pairs of bounding surfaces from the objects being combined intersect and if so to compute the intersection curve between them. These intersection curves, when properly trimmed by other bounding surfaces, form the boundaries of the new trimmed surface patches, which, in turn, bound the solid being created. An explicit specification of the geometry of the new solid, therefore, requires complete knowledge of intersection curves. Computation of intersections between surfaces is also needed in analysis such as in finite element mesh generation, in simulation of manufacturing processes and in control of machine tools.

In this paper, we address the problem of intersection of low order algebraic surfaces and piecewise continuous rational polynomial surface patches such as Bezier and B-spline patches. Low order algebraics of particular interest in geometric modeling are the classical algebraic surfaces such as planes, quadrics, cyclides as well as conical or cylindrical ruled surfaces and surfaces of revolution with rational polynomial profile curves [9]. Such surfaces also possess a rational polynomial parametric representation that can be obtained directly. This dual representation allows us to compute intersections of pairs of the above surfaces in the same manner as we treat intersections of low order algebraics and rational B-spline patches.

The above dual algebraic-parametric form is, in principle, available for rational polynomial patches such as biquadratic and bicubic Bezier patches through the process of implicitization and inversion [36]. However, as has been pointed out in the literature, the above process is impractical for numerical computation as it involves the expansion of large determinants [35, 9]. Furthermore, rational biquadratic and bicubic patches are high degree algebraics (8 and 18, respectively). This high degree also makes the use of such algebraic representations unattractive for intersection problems. To reiterate, the method proposed here is applicable to cases where one of the two intersecting surfaces has a low degree algebraic representation and the other a

rational polynomial parametric representation.

The intersection of two such surfaces can be very complicated, in general, involving closed loops, segments extending from boundary to boundary, common surface segments and singularities. The ability to detect and describe all such features of the intersection curve and to trace them correctly is an important problem in geometric modeling [12, 34, 40, 8, 4]. As can be seen from Section 2, improvements in the reliability and efficiency of current solution techniques are desirable.

This work summarizes an algorithm allowing automatic interrogation of planar algebraic curves within a rectangular domain, arising in the context of the above intersection problem. The method exploits the advantageous features of analytic representation of the governing equation in the Bernstein basis, adaptive subdivision, the a priori computation of border, turning and singular points of the curve and provides the basis for a reliable and efficient solution procedure. In particular, we address the problem of reliable derivation of the correct connectivity of the curve in the presence of internal loops and singularities arising from tangencies of the intersecting surfaces. Singularities and small internal loops are very common features of algebraic curves arising in a geometric modeling context and significantly affect the reliability and overall performance of current algorithms. Our algorithm has been influenced primarily by the work of [12, 37, 8].

This paper is structured as follows. Section 2 provides a brief review of previous work on the intersection problem. Section 3 formulates the present intersection problem in terms of tracing a planar algebraic curve in the Bernstein basis in a rectangular domain. Section 4 summarizes our research on algebraic geometry methods to compute significant points of the curve (such as extremal and singular points). Section 5 provides a more reliable alternative to the methods of Section 5 based on a hybrid technique involving minimization and Newton methods and Bernstein subdivision and faceting techniques. Section 6 outlines a method of tracing an algebraic Bernstein curve with known significant points in a rectangular domain using adaptive subdivision and Newton method based corrections of the approximation. Section 7 illustrates the application of our method and Section 8 summarizes the results of our work. Finally, Appendix I presents a technique to extract parametric line components of reducible algebraic curves.

2 Summary of Previous Work

Known methods to solve intersection problems are discussed under one of the following four categories :

Methods Relying on Representation : These methods rely on the derivation of a governing equation describing the intersection of two surfaces. For polynomial surfaces, the resulting equation is an algebraic curve which is an implicit polynomial in two variables. This equation can, in principle, be obtained by elimination of one Cartesian coordinate for the case of two implicit surfaces [37] or by elimination of three Cartesian coordinates for the case of an implicit surface intersecting a parametric surface [8]. In the first case, an equation for the projection of the intersection curve on one coordinate plane is obtained and the inversion algorithm of [36] is needed to compute the third coordinate of the three dimensional intersection curve. In the second case, the resulting intersection equation is an implicit polynomial in the parameters of the patch. When the implicit surfaces of interest are actually bounded, (i.e. they are algebraic patches [38, 27] or they resulted from implicitization of parametric patches [36]), points which are found to satisfy the intersection equation within the space of one patch should be tested to verify if they lie on the appropriate portion of the other surface.

The intersection between two algebraic surfaces can also be found by converting the problem to the simpler problem of algebraic-rational polynomial parametric surface intersection when a rational polynomial parametrization of one algebraic surface is possible [1]. In this approach, either one of the algebraic surfaces is converted into its rational polynomial parametric (RPP) form or another parametric polynomial surface which contains the curve of intersection is found. This approach has been used in the intersection of two quadric surfaces [21, 33] where a third parametric polynomial ruled surface, the parametrization surface, is found which also contains the intersection between the quadric surfaces. One of the quadric surfaces is then intersected with the parametrization surface to get the required representation.

Similarly, the intersection between two parametric surfaces can be attempted by first obtaining a representation for their curve of intersection. Since all rational polynomial parametric surfaces have an algebraic surface representation [36], we could, in theory, convert one of the two RPP surfaces to its algebraic representation and treat the problem as an algebraic surface/RPP surface intersection problem. The major problem involved here is that the algebraic representation of a patch with degree m, n in its two parameters is of degree 2mn which can be, relatively, high. The

subsequent substitution of the second parametric surface (say of degree p, q in its parameters u and v) in the algebraic surface will lead to an algebraic curve of very high degree (2mnp in u and 2mnq in v). For the case of intersections between rational biquadratic patches, we have a curve of degree 16 in each variable. Similarly, for rational bicubic patches, the resulting curve is of degree 54 in each variable. This high degree of the representation of the intersection curve discourages the use of methods relying on explicit representation for the computation of general RPP patch intersections.

In practical situations, once the equation describing the intersection curve between low degree algebraics and RPP patches is obtained as above, it must be traced. For special cases, the resulting implicit equations (algebraic curves) can be solved in terms of explicit expressions involving radicals [21] or in terms of rational parametric polynomials for algebraic curves with known singularities and whose genus is zero [2]. Once the range of the independent variable in such cases is determined, the above explicit equations can be used to trace the intersection curve. The local extrema and singular points of the curve can also be used to advantage as explained in [33]. However, for general cases explicit representation of algebraic curves in terms of elementary functions are impossible [36].

Algebraic curves with integer coefficients can be analyzed using the cylindrical algebraic decomposition algorithm [3], to provide a structure graph of the curve that defines pieces of the curve between significant points on the curve. The method separates the curve into a set of simple segments by creating parallel bands bounded by lines passing through singular points and curve extrema in one principal direction. Numerical methods (with or without rational arithmetic) are then used to step along each piece of the curve. This method, as implemented in rational arithmetic, although providing a guarantee that the solution is topologically reliable, is impractical at present, because of its very large memory requirements and poor efficiency. In addition, algebraic curves with integer coefficients are not general enough for geometric modeling, where simple rotation of intersecting primitives creates curves with possibly irrational or transcendental coefficients. The above considerations suggest the need to revert to other methods that trace general algebraic curves and in a more efficient manner. [8] uses the representation of the algebraic curve in order to compute significant points of the curve and to initiate a marching method of tracing the curve as discussed later.

Availability of a representation makes tracing of the curve more reliable and provides a direct

method for accuracy verification. It also facilitates computation of the significant points of the curve, as compared to cases where a representation is not explicitly available. Further, for trimmed patch processing it provides a complete and compact representation of the intersection in comparison to a linear approximation to the curve.

Lattice Evaluation : Lattice methods reduce the dimensionality of surface-to-surface intersection problems by computing intersections of a number of parametric curves of one surface with the other surface followed by connection of the resulting discrete intersection points to form different solution branches [42]. For intersections of parametric patches, the method reduces to the solution of a large number of independent systems of three nonlinear equations in three unknowns. For intersections of parametric polynomial patches with algebraic surfaces it reduces to the computation of the real roots of a large number of independent univariate polynomials within an interval. Although the technique is parallelizable, the solution of each independent univariate polynomial equation or system of equations is difficult. For univariate polynomials, no initial estimate of the solutions are required while for systems of nonlinear equations, available numerical techniques (such as Newton and minimization methods) require good initial approximations for convergence, an important disadvantage. Using elimination techniques [43], systems of nonlinear polynomial equations can, in principle, be reduced to the solution of high degree univariate polynomials and, hence, can be solved without initial estimates. The practical value of such a procedure is reduced by the error accumulation arising from the requisite determinant expansions and large scale computation. By definition of lattice methods, the reduction of the dimensionality of the problem involves an initial choice of grid resolution, which, in turn, may lead the method to miss important features of the solution, such as small loops and isolated points which reflect near tangency or tangency of intersecting surfaces. Finally, the second element of the method involves connection of discrete solution points to form solution branches. This, typically, requires determining adjacency on the basis of minimum mutual distance which may lead to incorrect connectivity particularly at singular points or for near singular situations. Derivative information may be employed to enhance the reliability of the method. In the lattice method of [32], this idea has been used in the approximation of curves of intersection by piecewise circular arcs. There an adaptive subdivision strategy, instead of derivative matching, has been used to prevent failure near closely spaced curve features, except the limiting case of singularities.

Marching Methods : Marching methods involve generation of sequences of points of an intersection curve branch by stepping from a given point on the required curve in a direction prescribed by the local differential geometry. Assuming the knowledge of starting points, marching methods are relatively simple to implement and efficient as they do not require solution of large numbers of nonlinear equations. They normally involve a variable step length move along the tangent of the curve followed by correction of that new position to satisfy a criterion of closeness to the curve. However, such methods require starting points for every branch and a stepping size which is case dependent and difficult to determine. Incorrect step size may lead to erroneous connectivity of solution branches or even to endless looping in the presence of closely spaced features [12]. Simple marching methods, not employing appropriate expansions of implicit polynomials near singular points, may also fail near such points. The desingularization method based on birational transformations outlined in [4] provides an elegant rectification of this type of failure in marching methods. Marching methods significantly benefit from the use of curvature analysis or power series expansions about each point of the solution to control the step size along the tangent [7]. In this reference, for example, an estimate of the radius of convergence of local power series expansions about non-singular points is derived and employed to specify the variable step size. This reduces the chances of incorrect connectivity and endless looping arising from simple marching methods.

Reliability of lattice and marching methods can be substantially improved by the determination of all significant points of the curve [33, 8]. The term significant points denotes the border, turning and singular points of the curve in a given domain. The spacing of these points is beneficial in the selection of, possibly, non-uniform grid size for lattice methods and stepping size and initial points for every branch in marching methods. Knowledge of significant points and the multiplicity of singular points also provides an independent count of the number of monotonic branches between significant points. This count can confirm the number of branches obtained using lattice and marching methods and provides added confidence in the solution. Encouraging results using such a method were reported in plane sectioning of low order parametric patches [8]. As noted in this reference, however, the above method does not establish the proper number of branches in the presence of tacnodes, i.e. cusps with a tangent direction which is a multiple tangent of the curve; and it requires rotation of the curve to handle branches between singular points without intervening turning points, as it cannot start at singular points. Starting at singular points requires additional information which can be found from the

desingularization procedure of [4]. The method to estimate the number of monotonic branches between significant points also requires adjustment when turning and singular points are also border points. We should also point out the extreme importance of accurate computations of all significant points since such marching methods depend on the availability of these points. As stated above, such significant points have also been used in the cylindrical decomposition of algebraic curves, where such points are computed exactly (ie. without numerical approximation) as solutions to algebraic equations with integer coefficients [3].

Barnhill et al [5] also use a marching method to trace the curve of intersection of two general topologically rectangular parametric surface patches. The method only requires a procedure to evaluate the surface position and its first partial derivatives at any point and, therefore, is not limited to polynomial surfaces. The method employs a combination of lattice evaluation, subdivision and Newton methods to determine starting points for different branches of the intersection which is then traced by a marching method. As pointed out in [5], the method does not handle the intersection of partially coincident surfaces and tangent or nearly tangent surfaces leading to singularities and small loops.

Subdivision Methods : The main idea of subdivision methods involves recursive decomposition of the original intersection problem into simpler similar problems until a level of simplicity is reached, which allows simple direct solution, (e.g. plane/ plane intersection). This is followed by a connection phase of the individual solutions to form the complete solution. By definition, the application of the method requires that the problem be subdividable, and, at each stage, there must be some recognizable reduction in the complexity of the problem. Parametric polynomial surfaces used in geometric design applications are amenable to subdivision [6, 18, 22] allowing computation of their intersections. Subdivision methods are convergent in the limit and do not suffer from many of the degeneracies of simple marching methods.

Initially conceived in the context of intersections of polynomial parametric surfaces [18, 41], they can be extended to the computation of algebraic/polynomial parametric and algebraic/algebraic surface intersections. This extension is possible by reformulating the algebraic curves arising from such intersections in terms of the Bernstein basis within a rectangular window [12] and interpreting such curves as intersections of Bezier surfaces and a plane [37]. Subdivision techniques do not require starting points as marching methods, an important advantage from the reliability point of view. Many elements of subdivision techniques

are also parallelizable, which is an important advantage for future large-scale real-time applications. General non-uniform subdivision allows easy selective refinement of the solution providing the basis for an adaptive intersection technique. The major disadvantage of subdivision techniques is that, in actual implementations with finite subdivision steps, correct connectivity of solution branches near singular points is difficult to guarantee, small loops may be missed or extraneous loops may be present in the approximation of the solution. In general, features of the solution smaller than the final subdivision size are not resolved. For example, in the method proposed in [12], the implicit intersection curve, expressed in the Bernstein form within a square, is subdivided using a quadtree approach until each cell contains one monotonic piece of the curve. The presence of such a simple piece of the curve within a cell can be detected by interrogation of the Bernstein coefficients of the curve for that cell. Such simple curve segments can then be traced using marching. However, this simplicity criterion fails, for example, at turning and singular points and the algorithm recurses until the cell size is the limiting precision of the machine, an inefficient feature given that the entire computation is based on subdivision. An important variant of the subdivision-polyhedron faceting methods involves degree reduction and subdivision to reach the simplicity of plane/plane intersections [28, 40]. The efficiency of degree reduction and other intersection techniques is discussed in [39].

3 Bernstein Representation of the Intersection

3.1 Definitions and Preliminary Transformations

A general non-uniform rational B-spline surface patch $\mathbf{Q}_{k,l}(u,v)$ of maximum degrees k and l in the parametric variables u and v is given by

$$\mathbf{Q}_{k,l}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{ij} N_{i,k}(u) N_{j,l}(v) \quad where \quad u \in [u_a,u_b], v \in [v_a,v_b] \tag{1}$$

where $\mathbf{P}_{ij}$ are homogeneous coordinates of the (m+1) x (n+1) control points given in a world coordinate system and $m \geq k$, $n \geq l$. The fourth homogeneous coordinate of each control point $\mathbf{P}_{ij}$ is assumed to be positive. $N_{i,k}(u)$ and $N_{j,l}(v)$ are the B-spline basis functions defined over non-uniform knot vectors in the u and v directions given by $\{u_0,...u_{k+m+1}\}$ and $\{v_0,...v_{l+n+1}\}$ respectively and may be computed by de Boor's recursion.

The algebraic surface, G, is expressed as an implicit polynomial equation in the homogeneous

coordinates $\mathbf{R} = [x, y, z, w]$ as

$$G(x,y,z,w) = \sum_{i=0}^{q} \sum_{j=0}^{q-i} \sum_{k=0}^{q-i-j} D_{ijk} x^i y^j z^k w^{q-i-j-k} = 0 \tag{2}$$

where x/w, y/w, z/w are the three-dimensional Cartesian coordinates in the world coordinate system and the degree of the surface is q.

The intersection of the above two surfaces can be defined as the set of points of patch (1) which lie on the surface represented by (2). Hence the general point on (1) is substituted into (2) to give (3), the general equation of the intersection of the two surfaces

$$G(x,y,z,w) = G(\mathbf{R}) = G(\mathbf{Q}_{k,l}(u,v)) = F(u, v) = 0 \tag{3}$$

Such an equation, in general, represents an algebraic curve in the variables u, v where u, v are restricted to the parametric space of patch (1). The maximum degree in u is kq and in v is lq.

The substitution of (1) in (2), while conceptually simple is very difficult to perform in practice for anything but the simplest algebraic surface, ie. the plane. For degree two algebraic surfaces, the quadrics, the substitution involves the squaring of piecewise polynomial functions and for higher degree algebraic surfaces it involves raising to higher powers and is unattractive for use. Further, the general form of (2) involves products of x, y, z and w making the substitution very complex and, therefore, any reduction in the complexity of (2) by suitable transformations, such as translation and rotation, is useful for implementation. Classical algebraic surfaces when specified in a local (natural) coordinate system assume a compact form. These considerations suggest that it is convenient to perform the intersection in a coordinate system where the algebraic surface equation is most compact and also obtain intersections of each of the component rational polynomial patches separately. This approach requires transformation of the parametric patch equation to the local (natural) system of the algebraic surface and conversion of the B-spline surface patch into its component rational polynomial (Bezier) patches [22]. When the algebraic surface is a plane, such transformations are not needed as the parametric patch equation can be directly substituted in the implicit plane equation. The resulting Bezier patches are defined by equations of the form

$$\mathbf{S}_{pq}(u,v) = \sum_{i=0}^{k} \sum_{j=0}^{l} \mathbf{P}_{ij}^{pq} B_{i,k}(u) B_{j,l}(v) \qquad u,v \in [0,1] \tag{4}$$

where p,q are indices representing a typical component patch, $\mathbf{P}_{ij}^{pq}$ are the homogeneous control points of this patch and $B_{i,k}(u)$ and $B_{j,l}(v)$ are the Bernstein polynomials of degrees k and l

respectively. The algebraic surface is given by

$$H(x,y,z,w) = \sum_{i=0}^{q} \sum_{j=0}^{q-i} \sum_{k=0}^{q-i-j} C_{ijk} x^i y^j z^k w^{q-i-j-k} = 0 \tag{5}$$

where the C_{ijk} are assumed to be normalized using the maximum $|C_{ijk}|$ so that $|C_{ijk}| \leq 1$.

3.2 Representation of Intersection Curve

A representation of the intersection of this rational polynomial parametric patch with the algebraic surface $H(x,y,z,w) = 0$ is obtained by substituting (4) in (5) to get an implicit polynomial equation

$$F_{pq}(u^m, v^n) = 0 \quad \textit{where} \quad u \in [0,1], v \in [0,1] \tag{6}$$

the dependence u^m and v^n signifies that the maximum degree in u is m and in v is n and where the index p and q signifies that each polynomial subpatch of the original B-spline patch gives a separate equation representing the intersection in that region. This implicit equation describes, in general, a planar algebraic curve in the parametric space of the patch of degree $m = kq$ and $n = lq$ in the two variables of interest. Such a curve is a special case of the general planar algebraic curve of degree m+n because no terms of the form u^{m+n} or v^{m+n} appear in (6). Algebraic curves of even modest degree can be very complex [43, 12, 8]. Solution of the intersection problem stated above, therefore, reduces to the discovery and description of all features of a planar algebraic curve in a finite rectangular domain and its subsequent mapping in three-dimensional space.

Derivation of (6) can be, for example, accomplished by converting the rational polynomial patch from its Bernstein form into its monomial form to allow easy multiplication of a number of bivariate polynomials resulting from the expansion of powers of x, y, z and w in $H(x,y,z,w) = 0$. This process leads to (6) expressed in the monomial form

$$F(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} a_{ij} u^i v^j = 0 \quad \textit{where} \quad u \in [0,1], v \in [0,1] \tag{7}$$

where subscripts p, q were dropped. In the case of the intersection problem at hand, where we are interested only in a finite portion of the curve within a rectangular domain, it is expedient to transform the above monomial expansion into the Bernstein basis

$$F(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} B_{i,m}(u) B_{j,n}(v) = 0 \quad \textit{where} \quad u \in [0,1], v \in [0,1] \tag{8}$$

Curve (8), in general, has (m+1)(n+1)-1 degrees of freedom because one of the coefficients can be chosen arbitrarily without modifying the curve. The new coefficients w_{ij} are obtained by matrix multiplication

$$[\,W\,] = [\,B_m\,]\,[\,A\,]\,[\,B_n\,]^T \qquad T - denotes\ \ transpose \tag{9}$$

$W = [\,w_{ij}\,]$ and $A = [\,a_{ij}\,]$ are (m+1)x(n+1) matrices of the coefficients and $[\,B_k\,]$ is a (k+1)x(k+1) transformation matrix with elements given by [10]

$$\begin{aligned} B_{ij} \quad &= \binom{i}{j}\binom{k}{j}^{-1} \quad for\ i \geq j \\ &= 0 \qquad\qquad for\ \ i < j \qquad i,j = 0,1,\ldots k \end{aligned} \tag{10}$$

The elements of matrices $[\,B_m\,]$ and $[\,B_n\,]$ can be computed exactly as they are rational numbers and is useful in implementing transformation (9) with improved accuracy.

In the exceptional case, where $w_{ij} = 0$ for all i = 0,1...m, j = 0,1...n, (8) does not define a curve as the intersection of two surfaces. It, rather, points out that the rational polynomial patch coincides with a portion of the algebraic surface. When the coefficients of the algebraic surface are normalized as in (5), and floating point arithmetic is employed, the condition $|\,w_{ij}\,| \leq \varepsilon$, where ε is a small positive number, may be used to test for such an exceptional occurrence. The non-dimensional number $\varepsilon << 1$ is related to a distance tolerance, δ, for coincidence of the two surfaces, by the approximate equation $\varepsilon \approx \delta\,|\,\nabla H\,|$, where ∇ denotes gradient and ∇H is evaluated at a point **S** on the parametric surface assumed to be close to the algebraic surface. When the w_{ij} are either all positive or all negative, there is no intersection since the $B_{i,m}(u)$ and $B_{j,n}(v)$ are always non-negative.

In what follows, it is convenient to normalize the coefficients w_{ij} using the maximum $|w_{ij}| \neq 0$, so that $|w_{ij}| \leq 1$ for all i and j. The above normalized w_{ij} will be used as the representation of the intersection curve for all further discussion. The procedure of substitution leading to (7) and the transformation into the Bernstein basis can be carried out by implementing polynomial multiplication to obtain the a_{ij} and then use transformation (9). This process involves extensive numerical computation to obtain the Bernstein coefficients, w_{ij}, which could lead to significant loss of accuracy. The same procedure, can also be carried out symbolically in a symbolic manipulation system [23]. In such an approach, the Bernstein form of the algebraic curve is obtained explicitly in terms of primary data, namely the control points of the patch (4) and the coefficients of the algebraic surface (5). In general, w_{ij} is a function of the form

$$w_{ij} = f_{ij}\,(\,\mathbf{P}^{pq}_{\mu\nu},\, C_{def}) \tag{11}$$

where the $\mathbf{P}^{pq}_{\mu\nu}$ are the control points of surface (4) and C_{def} are the coefficients of the algebraic surface (5). The functions in (11) may be worked out for all pairs of required surfaces ie. for varying degree rational Bezier and algebraic surfaces, and directly incorporated into general intersection programs. It was noted during such a derivation that the number of terms involved in the expressions for w_{ij} were significantly lesser than the corresponding number of terms for the intermediate a_{ij} in terms of the same primary data of the patch in the Bernstein basis. This implies that the a_{ij} could involve less accuracy in their computation compared to the w_{ij}. Further, since we are interested in eventually obtaining the Bernstein representation, these inaccurate a_{ij} have to be used in the monomial to Bernstein conversion according to (9). Such a conversion has, recently, been shown to be inherently ill-conditioned [11] and should be avoided when possible. Availability of explicit expressions for the coefficients w_{ij} (essentially summations of products) can also be used to advantage, particularly in higher order cases where the number of terms may be substantial, to get better numerical accuracy in the computation of w_{ij} in floating point arithmetic. Use can be made of sophisticated methods to sum series that give much less accumulation of round-off error [17]. The capabilities of symbolic manipulation systems can also be employed to factor terms and identify subexpressions in order to enhance efficiency.

3.3 Geometric Interpretation of Algebraic Curves in a Rectangular Domain

It is now convenient to visualize (8) as the intersection of the explicit surface w = F(u,v) with the control plane w = 0. The above explicit surface, can be recast as a tensor product Bezier patch

$$\mathbf{T}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{T}_{ij}\, B_{i,m}(u)\, B_{j,n}(v) \quad where \quad \mathbf{T} = [\,u\ v\ w\,] \qquad \mathbf{T}_{ij} = [u_i'\ \ v_j'\ \ w_{ij}]$$

$$u_i' = \frac{i}{m} \ and \ v_j' = \frac{j}{n} \quad i = 0,1,...m \ and \ j = 0,1,...n \tag{12}$$

The coefficients w_{ij} introduced in (8) are the w coordinates of the control polyhedron vertices of the parametric surface defined by the above equation while the u and v coordinates of control polyhedron vertices are uniformly spaced in the range of the parametric variables. The above reformulation has been used by [12] in a study of surface interrogations, while [37] used a similar method to represent algebraic curve portions within triangles. [25] introduced an extension of (12) to B-splines as a method of shape creation in terms of piecewise continuous

algebraic curves and surfaces within rectangular boxes. Since the control polyhedron, $\mathbf{T}_{ij}$, provides an approximation to the geometry of the surface (12), we can get an approximation of the intersection curve (8) by intersecting a faceting of the control polyhedron with the plane $w = 0$. The control polyhedron can be made piecewise planar by triangulating the rectangular grid of polyhedron vertices (faceting). Each of the triangles can then be intersected with the plane $w = 0$ to get a series of straight line segments which together form a piecewise linear approximation of the curve. The accuracy of this approximate intersection curve depends on the accuracy to which the control polyhedron approximates the surface. As is well known, the control polyhedron becomes an increasingly better approximation of the surface by the use of subdivision.

4 Computation of Significant Points Using Algebraic Geometry Techniques

4.1 Significant Points

The approximate algebraic curve generated using the faceting method outlined in Section 3 does not, in general, exhibit the connectivity of the actual algebraic curve for finite refinement of the polyhedron. In particular, portions of the curve near singularities cannot, in general, be approximated by plane-plane intersections and spurious small loops may also be easily introduced. This suggests partition of the surface (12) at turning and singular points so that the resulting pieces of the control surface (hereafter referred to as subpatches) represent monotonic curve portions that have no singularity or actual internal loops within their respective subdomains. This also ensures that curve complexity associated with singularities resides only at corners of subpatches. Once partition of the surface has been carried out along parametric lines corresponding to these significant points, the procedure of intersecting each facet of the polyhedron of each subpatch can be carried out and the resulting straight line segments can be connected together to reflect the true connectivity of the curve. This reformulation provides subdivision-faceting based intersection solutions with the potential for extracting the true connectivity of the curve. This procedure of splitting along parametric lines of the control surface, the tracing of the curve defined by each of the subpatches and the resolution of constrictions is described in Section 6. Here we concentrate on a method to compute the turning and singular points using elimination methods from algebraic geometry while in Section 5 we propose a new method to compute such points using direct numerical techniques coupled with subdivision. We first define these significant points of the curve.

Singular points of an algebraic curve are defined by vanishing partial parametric derivatives ie. by the following three simultaneous equations

$$F(u,v) = F_u(u,v) = F_v(u,v) = 0 \tag{13}$$

An extensive discussion of singular points may be found in [43, 8, 4]. Singular points are classified according to the number of partial derivatives, beyond the first, vanishing at such points. Points at which (13) is valid and at least one second order partial parametric derivative is non-zero are called double points. In general, points on an algebraic curve at which all partial derivatives up to order m-1 vanish and at least one partial derivative of order m is non-zero are called singular points of multiplicity m.

Turning points are points on an algebraic curve at which the normal vector to the curve is parallel to the parametric u and v axes. For non-singular points, the unit normal vector is uniquely defined by $\mathbf{n} = \nabla F/|\nabla F|$ and, therefore, turning points on an algebraic curve may be defined by the following two sets of simultaneous equations

$$F(u,v) = 0 \qquad F_u(u,v) = 0 \tag{14}$$

and

$$F(u,v) = 0 \qquad F_v(u,v) = 0 \tag{15}$$

Eq. (14) define the so-called v-turning points where the normal vector is parallel to the v-axis and Eq. (15) the u-turning points. According to Eq. (13), points satisfying the requirements for the u and v turning points simultaneously, are not turning but singular points. Therefore, identification of all solutions of (14) and (15) also provides all singular points.

When the curve F(u,v) = 0 can be written as

$$F(u,v) = \prod_{i=1}^{M} F_i^{e_i}(u,v) = 0 \tag{16}$$

the curve is made up of component curves $F_i(u,v)$ with multiplicity e_i [43]. When $e_i \geq 2$ an infinity of points of the curve F(u,v) = 0 satisfy (13) to (15) and a modification of the method presented here is required. Such a situation corresponds to the case of infinite singularities arising from tangency of the two intersecting surfaces along a curve. When $e_i = 1$, the above definitions of turning and singular points remain valid for a finite number of such points. When a parametric line (ie. u - c = 0 or v - d = 0) is a component of the intersection curve, it should be factored out from the representation used to find turning and singular points because such lines form an infinity of turning points, see Appendix I.

Beyond turning and singular points it is also convenient to identify border points a priori in order to allow reliable connection of the intersection curve with other parts of the curve outside the domain of interest. Border points are points of the curve at which at least one of the parametric variables take values equal to the borders of the rectangular parametric domain. The border points are located by solving a univariate polynomial in u for the v = 0 and v = 1 borders and a univariate polynomial in v for the u = 0 and u = 1 borders. Substituting these specific values for each border in the equation of the algebraic curve (8) and using the properties of the Bernstein basis we obtain for the v = 0 border, for example,

$$F(u,0) = \sum_{i=0}^{m} w_{i0}\, B_{i,m}(u) = 0 \tag{17}$$

Border, turning and singular points are collectively called the significant points of the curve. The real roots of the above polynomials of degree m or n in the interval [0,1] can be found using a root-solving technique that directly exploits the properties of the Bernstein basis. Such a technique is outlined in [19] for solving for the real roots of a polynomial in an interval employing recursive binary subdivision and the variation diminishing property of this basis. For use with other standard polynomial root solvers [24, 23], the polynomial must be first converted the monomial basis. However, based on [10] and our numerical experiments we do not recommend such conversions as they adversely affect the computation accuracy.

4.2 Turning and Singular Point Computation

4.2.1 Preliminary Remarks and Derivations

In matrix form, equation (14) in the case of (8) expressed in the Bernstein basis, becomes

$$\begin{aligned} &[\, B_u^m \,]\, [\, W \,]\, [\, B_v^n \,]^T = 0 \\ &[\, B_u^{m-1} \,]\, [\, W^u \,]\, [\, B_v^n \,]^T = 0 \end{aligned} \tag{18}$$

where [W] is the matrix of w_{ij}'s of size (m+1)x(n+1) and $[\, B_u^m \,]$ and $[\, B_v^n \,]$ are the vectors of all Bernstein polynomials of degree m in u and degree n in v, respectively, and W^u is a matrix of size m x (n+1) with elements w_{ij}^u obtained directly by taking the derivative of (18a)

$$w_{ij}^u = m\,(\, w_{i+1\,j} - w_{ij} \,) \quad \textit{for } i = 0,1,\ldots m-1 \textit{ and } j = 0,1,\ldots n \tag{19}$$

All real solutions of the above simultaneous equations within the window of interest [0,1]x[0,1] are required for the determination of v-turning points. The conversion of (15) to matrix form and their solution follows a procedure similar to the derivation and solution of (18). Singular points

are located by solution of simultaneous Eqs. (13), and hence, can be obtained as the common solutions of (14) and (15). Note that singularities of all orders satisfy the above equations and their multiplicities could be identified by additional computation of higher order derivatives [43]. A feature of the present method is that explicit a priori knowledge of the multiplicity of the point, based on higher order derivative evaluations, is not employed.

Eq. (18) represents two simultaneous bivariate polynomial equations, the real solutions of which in the region [0,1]x[0,1] are of interest. First, we consider algebraic geometry techniques for reducing this problem to the solution of a univariate polynomial equation. This process involves elimination of one variable from a pair of bivariate polynomial equations [8]. This gives us a univariate polynomial equation which is, in general, of higher degree in the other variable than the degrees in u and v in the original equations. Such univariate polynomials can be solved without the help of initial estimates, an important advantage with respect to other iterative numerical techniques for the solution of systems of non-linear equations with an unknown number of roots within a given domain. All real solutions of this univariate equation give the set of values of one of the variables, which possibly form one element of solution pairs within this domain of the simultaneous equations. These solutions are then back-substituted in the original equations such as one of Eq. (18) and univariate polynomials (of generally much lesser degree) are solved to get values of the variable originally eliminated. From the solutions of the above univariate polynomial equation, only the real solutions in [0,1] also satisfying both Eqs. (18) are the turning point solutions.

4.2.2 Elimination in the Bernstein Basis

The general theory of elimination deals with finding conditions which the coefficients of two univariate polynomial equations must satisfy in order to have common roots. A good discussion of elimination techniques is given in [43] and their use in computational geometry is discussed in [36, 8]. An extension of this technique can be used to eliminate one of the variables from two bivariate polynomial equations. Elimination theory with the polynomials expressed in the monomial basis is well developed. In what follows, we use these techniques with their appropriate extension in the Bernstein basis. Working in the Bernstein basis is advantageous in geometric modeling applications, not only because of their geometric properties but also because of their computational properties. It has been observed, for example, that operations such as root computations on polynomials expressed in the Bernstein basis are numerically better conditioned

as compared to similar computations in the monomial basis [39, 26]. More recently, it has been shown theoretically that the Bernstein basis is superior to the monomial basis for common operations such as evaluation and root-finding [10]. Hence, we prefer to carry out this elimination step in the Bernstein basis directly. Expressing (18) as polynomials in v we have

$$[A'][B_v^n]^T = 0 \;\; where \;\; [A'] = [B_u^m][W] \tag{20}$$

$$[B'][B_v^n]^T = 0 \;\; where \;\; [B'] = [B_u^{m-1}][W^u] \tag{21}$$

[A'] and [B'] are row vectors of size 1x(n+1) with elements defined by

$$a_j'(u) = \sum_{i=0}^{m} w_{ij} B_{i,m}(u) \qquad b_j'(u) = \sum_{i=0}^{m-1} w_{ij}^u B_{i,m-1}(u) \qquad for\ j = 0,1,\ldots n \tag{22}$$

Next we rewrite Eqs. (20) and (21) so that the vectors $[B_v^n]$ contain only the polynomial part of the Bernstein coefficients without the leading combinatorial factors. In this manner, Eqs. (20) and (21) become

$$[\,A\,][\,L_v^n\,]^T = 0 \qquad [\,B\,][\,L_v^n\,]^T = 0 \tag{23}$$

where the elements of row vectors [A] and [B] (size 1x(n+1)) are defined by

$$a_j(u) = \frac{n!}{j!(n-j)!} \sum_{i=0}^{m} w_{ij} B_{i,m}(u) \qquad where\ j = 0,1,\ldots n$$

$$b_j(u) = \frac{n!}{j!(n-j)!} \sum_{i=0}^{m-1} w_{ij}^u B_{i,m-1}(u) \qquad where\ j = 0,1,\ldots n \tag{24}$$

and the elements of the row vector $[L_v^n]$ (size 1x(n+1)) by

$$l_{j,n}(v) = (1-v)^{n-j} v^j \qquad where\ j = 0,1,\ldots n \tag{25}$$

We can now proceed to eliminate v from (23) using the technique described in [39, 14] leading to the following system of 2n homogeneous equations

$$[\,D(u)\,].[L_v^{2n-1}]^T = [\,0\,] \;\; where$$

$$d_{ij} = a_{j-i}(u) \qquad for\ 0 \le i \le n-1 \;\; and \;\; j \ge i$$

$$= b_{j-i+n}(u) \;\; for \;\; n \le i \le 2n-1 \;\; and \; j \ge i-n$$

$$= 0 \qquad otherwise$$

$$l_{j,2n-1}(v) = (1-v)^{2n-1-j} v^j \qquad j = 0,1,2,\ldots,2n-1 \tag{26}$$

We seek the set of values (u_i, v_i) such that the above system of equations is satisfied. A necessary and sufficient condition that the above system has non-trivial solutions is that $|\,D\,| = 0$, where $|\;|$

denotes determinant. Since all elements of the matrix D are polynomials in u, the determinant in this case is the characteristic polynomial in u of degree 2mn-n [8] which defines possible v-turning points within the domain. This univariate polynomial can be expressed in the monomial basis as follows

$$\sum_{i=0}^{2mn-n} c_i u^i = 0 \tag{27}$$

to be solved to give the solution set u_i. For each of the u_i we can find the corresponding v_i coordinate by substituting in one of the Eqs. (18) and solving a univariate polynomial equation in v of degree n. From these solutions, those which satisfy the other equation (from Eqs. (18)) lead to the final solution set (u_i,v_i). Note that, although the above elimination was carried out in the Bernstein basis, the resulting characteristic polynomial in u is expressed in the monomial basis since polynomial multiplication needed in direct determinant expansion is easier in this basis.

The solution of the univariate polynomial for turning points forms the critical part of the determination of significant points since the quantities 2mn - n and 2mn - m, representing the degree of the turning point characteristic equations, are greater than or equal to m or n and grows rapidly with the degree of the intersecting surfaces ie. k, l and q. As the degree of the equation grows, the error involved in each evaluation of the polynomial using floating point arithmetic during the determination of roots grows linearly with the degree [10]. This implies that, when the coefficients are represented as accurately as the precision of the computer will allow, the accuracy in the evaluation and, therefore, solution of these equations will deteriorate with degree. A second, and possibly more serious, problem is significant root perturbation due to small perturbation of the coefficients. This may occur in the solution of any polynomial whose coefficients are not primary data but are the result of some intermediate computation [44].

In order to appreciate the complexity of various intersection problems encountered in geometric modeling, we include Table 1 illustrating the degrees of polynomials, the solution of which is, *in principle*, needed to allow computation of significant points. The degrees involved in the determination of turning points for the plane-biquadratic, plane-bicubic and quadric-torus (the torus being represented as a rational biquadratic) are of relatively low degree. The degree of the characteristic polynomial for some of the other cases like torus-torus (biquadratic) or torus-bicubic is relatively high.

Table 1: Intersection Problem Complexity

Surf A	Surf B	Curve Degree in Cartesian coordinates	Minimum curve exponents in in parametric form $F(u^m,v^n) = 0$	Polynomial Degree Border	Turning/ Singular
Plane	Torus	4	2,2	2	6
Quadric	Torus	8	4,4	4	28
Torus	Torus	16	8,8	8	120
Plane	E-cubic	3	3,1	3	3
Quadric	E-cubic	6	6,2	6	22
Torus	E-cubic	12	6,6	6	66
Plane	R-cubic	6	3,2	3	10
Quadric	R-cubic	12	6,4	6	44
Torus	R-cubic	24	12,8	12	184
Plane	P-cubic	18	3,3	3	15
Quadric	P-cubic	36	6,6	6	66
Torus	P-cubic	72	12,12	12	276

Note:
E-cubic = Ruled Patch with Rational Cubic Profile
R-cubic = Surface of Revolution with Rational Cubic Profile
P-cubic = Rational Bicubic

4.3 Determination of Characteristic Polynomial

In this section, we discuss two methods of obtaining the coefficients c_i which could lead to better accuracy on the roots of the characteristic polynomial. The first approach uses a symbolic method to obtain an explicit expression for each coefficient and is aimed at reducing the error in the polynomial coefficients themselves. The second approach attempts to reduce the chance of complete loss of turning and singular points by obtaining these coefficients directly in the Bernstein basis, since root computation in this basis is known to be stable compared to the monomial basis.

4.3.1 Symbolic Method

Expansion of determinants needed to obtain the coefficients of the characteristic polynomial is an inherently ill-conditioned problem [44]. It involves the multiplication of various elements of D, each of which is a polynomial in u. In this section, we investigate the use of symbolic computation as a means to reduce the error accumulated in the numerical computation of c_i for two reasons. First, since a number of coefficients of the matrix [D] are equal to other coefficients, we expect a sizable amount of simplification to occur in the computation of these coefficients. If expansion is carried out symbolically, such simplifications can be easily performed leading to numerical error reduction compared to a direct numerical expansion. Second, the symbolic derivation of the characteristic polynomial in a symbolic system like [23] allows complete control over the various terms and symbols that combine to form the polynomial coefficients. That is, it allows each coefficient in the final polynomial (27) to be obtained as one continuous sum of products of the original Bernstein coefficients w_{ij} and w_{ij}^{u}. This control over the terms of the sum allows us to adopt sophisticated methods of summing a series of values which give a much smaller relative error on the sum than direct summation methods. Most importantly, following Kahan's method [17], the relative error on the sum is independent of the number of terms N of the sum and linear in the machine precision ε (when $N \ll 1/\varepsilon$). The above method offers a remarkable improvement over direct summation.

This method allowed the determination of each of the coefficients in symbolic form

$$c_i = \sum (constant \times \prod t_{jk}) \tag{28}$$

where t_{jk} is either one of w_{ij} or w_{ij}^{u} or a linear combination of those. The number of terms in each product is small and at most twice the degree m or n while the number of terms in the summation of these products is large and varies with the degree i of the coefficient c_i being computed. It is a maximum for indices i near the middle and a minimum for the indices i at the ends of the interval [0,2mn-n]. If a simple summation is adopted, the error in these coefficients can be high and it is here that the use of Kahan's summation scheme is used. Experiments using the above procedure including Kahan's enhancements were performed in connection with plane sections of rational biquadratic and bicubic patches.

The growth in the number of terms in the summation rapidly increases with degree and the symbolic scheme of finding expressions for c_i become *infeasible* due to the enormous

computational resources required. An alternate method that is not limited by such large symbolic expansions is the implementation of polynomial arithmetic and performing substantial summations using Kahan's scheme and without recourse to explicit expression derivation. Such a method was not implemented since it only solves the problem of limited computational resources but does not alleviate the loss of accuracy that occurs even with the use of Kahan's summation as was occasionally noticed in the case of plane sections of bicubic patches.

4.3.2 Indirect Approach to Obtain the Bernstein Form of the Characteristic Polynomial

The characteristic polynomial equation $|D(u)| = 0$ is known to be a polynomial in u of a given degree, say p, where $p = 2mn-n$ for v-turning points. Hence it is possible to evaluate the value of this polynomial at a discrete set of p+1 values of u in the domain of u and obtain a set of p+1 equations in the unknown coefficients b_i of the univariate polynomial. These evaluations of the determinant of D(u) require p+1 evaluations of each polynomial element of D(u) and p+1 evaluations of the determinant. The corresponding p+1 values of the determinant |D(u)| are the Lagrange coefficients of the characteristic polynomial associated with the above sampling. For $u \in [0,1]$, the discrete values of u can be chosen to be equally spaced in the domain of u so that they are rational numbers. If (27) is expressed in the Bernstein basis as

$$\sum_{j=0}^{p} b_j B_{j,p}(u) = 0 \quad u \in [0,1] \tag{29}$$

then the resulting linear system from the p+1 evaluations is given by

$$\sum_{j=0}^{p} b_j B_{j,p}(u_i) = |D(u_i)|, \quad u_i = \frac{i}{p}, \quad i = 0,1,2 \ldots p \tag{30}$$

Since $b_0 = |D(0)|$ and $b_p = |D(1)|$, the system is reduced to the solution of p-1 equations in the p-1 unknowns b_1 through b_{p-1}. Since the u_i are rational numbers, each of the $B_{j,p}(u_i)$ and the matrix $[B]^{-1} = [B_{j,p}(u_i)]^{-1}$ can be computed exactly using rational arithmetic and obtained to the full floating point precision by a single division at the end. The computation of the coefficients then reduces to matrix multiplication and can be performed with the best possible accuracy. Since the characteristic polynomial is now in the Bernstein basis, the solution method of [19] can be invoked. Successful experiments using this approach were performed for plane sections of rational biquadratic and bicubic patches. Higher degree cases, however, require excessive computational resources in the inversion of the matrices using rational arithmetic and, therefore, the above approach proves to be unattractive for general application.

4.4 Determination of Turning and Singular Points

Eqs. (18) were solved above by eliminating the variable v. We could instead have solved the equations by eliminating u. The degrees of the resulting polynomials in u or v are approximately of the same order and are equal when $m = n$. Eliminating v involves the expansion of a determinant of size 2n while eliminating u involves a size of 2m-1. For the case $m = n$, this involves the expansion of a slightly smaller determinant when we eliminate u. However elimination of u leads to an ill-conditioned problem. This can be seen qualitatively from the fact that $dF = F_u du + F_v dv = 0$ along the curve and, hence, $du/dv = -F_v/F_u$, is very large in the neighborhood of a v-turning point so that a small change of v due to error will lead to a much larger change of u. This implies that solutions for v-turning points should, preferably be performed by first eliminating v and solving for the u coordinate. If the other coordinate u is eliminated instead, then an error δv associated with the root-finding procedure for v will lead to large displacements δu of the corresponding u. Within floating point computation, the coefficients of the characteristic polynomial are perturbed quantities because of the process of expanding determinants of matrices whose elements are themselves polynomials. Accumulation of round-off error in the characteristic polynomial coefficients along with the inherent error accumulation characteristics of general root-solving techniques is bound to displace the values of v. For this reason, it is advantageous to eliminate v and solve in terms of u first. In the case of the u-turning point equation, we eliminate u and solve for v which, by analogous reasoning gives better conditioning in this case.

The solution of Eq. (27) in u will lead to $2mn - n$ solutions $\{u_s\}$. Of these only the real solutions in the range [0,1] are of interest

$$\{u_s\} \quad s = 1,2,\ldots \mu \quad \textit{where} \quad \mu \leq 2mn-n \tag{31}$$

We next substitute each of the above $\{u_s\}$ in the first of (18) to get a univariate polynomial in v of degree n in the Bernstein basis, which can be directly solved [19] to get the set of solutions of corresponding v's $\{v_{st} : t = 1...n\}$. Of these, only the real solutions in the range [0,1] are of interest, ie. $\{v_{st} : t = 1...\nu(u_s)\}$, where $\nu(u_s) \leq n$ is the number of real solutions of v in [0,1] for every solution u_s. These solution pairs are then substituted in the second of (18) to remove those solutions that do not satisfy this equation. This process is repeated for all solutions (31) to get the entire solution set which satisfies both of Eqs. (18) and is denoted by $\{u_i, v_i\}$ $i = 1,2,...p$ where p is the number of v-turning points of the curve F(u,v) in [0,1]x[0,1]. The two bivariate

polynomials $F(u,v) = 0$ and $F_v(u,v) = 0$ may be similarly solved to get the list of u-turning points $\{u_i, v_i\}$ $i = 1,2,...q$ where q is the number of u-turning points in this case. The set of common solutions between the u and v turning points is then the set of singular points $\{u_i, v_i\}$, $i = 1,2,...r$ where $r \leq \min(p,q)$. Separate identification of singular points is not required in our method and the solution for the two types of turning points and the border points are merged and any duplicates removed. Such duplicates occur at singular points where two points appear in the merged list once as a u-turning point and then as a v-turning point and when a point is border, turning and/or singular at the same time. Two u or v values are considered the same if they are the same to the accuracy with which the roots of the characteristic polynomial are computed. The final list of significant points is denoted by $\{su_i, sv_i\}$ $i = 0,1,...s$ where $s \leq 4mn + m + n - 1$ each element of which is at least one of the border, turning or singular points.

5 Computation of Turning and Singular Points using Direct Numerical Techniques

5.1 Introductory Remarks

The computation of turning and singular points according to the method described in Section 4 encounters significant numerical problems in actual implementations. The polynomial equations that have to be solved are of high degree and their coefficients are obtained by the expansion of determinants. Both these characteristics lead to significant perturbations in the values for the turning points when using floating point arithmetic, particularly for high degree cases. These considerations suggest the use of other direct numerical techniques like minimization and Newton techniques for the computation of turning and singular points. These methods depend only on evaluations of F(u,v) and its derivatives, which are of relatively low degree compared to the polynomials arising from elimination and, hence, suffer less inaccuracy in their computation. The above numerical methods, however, require good initial approximations which are not easily available. It is for this reason primarily that the earlier direct elimination method was addressed first in the literature and also investigated in this paper. The objective of this section is to provide a scheme to obtain these starting points and to use them with minimization and Newton techniques to compute singular and turning points directly.

5.2 Computation of Starting Points and Consistency Verification

An approximation to the intersection curve can be obtained by choosing a rectangular grid of points *on* the parametric surface $w = F(u,v)$, and triangulating and intersecting the resulting facets with the plane $w = 0$. Such a scheme is however not very satisfactory because a small loop could be easily missed if its size is small compared to the grid size. A similar comment holds for the limiting case of a small loop, an isolated point.

This suggests the use of a control polyhedron faceting method to approximate the curve. Due to the convex hull property the above condition of missing a single loop could not occur. It is well known that uniform subdivision [6, 22] of the control polyhedron and its subsequent faceting provides a better approximation to the surface as the subdivision is reduced. The approximation to the algebraic curve obtained from the polyhedron does not, in general, reflect the true connectivity of the actual curve under finite uniform subdivision steps. The approximation can, however, be efficiently improved by inserting knots using information about the deviation of this approximation from the true curve and this process of subdivision and consequent improvement can be repeated till some criterion of accuracy is met. In the vicinity of singularities, plane-plane intersections of the type carried out to obtain the above approximate intersection cannot, in general, model the complexity of the intersection curve. Thus the above method does not provide a complete means of tracing the intersection curve. However, this approximate intersection curve (called A in what follows) provides good starting estimates in the form of turning points for both turning and singular points. Since these points are to be used in conjunction with iterative non-linear equation solution techniques they must be close (in distance) to the solution. Therefore, all points on A are first required to lie within a certain small distance of the actual algebraic curve so that any starting points found from A are also close to the solutions of (14) or (15). A scheme of inserting knots and refining the original surface to bring the polyhedron closer to the surface is used to increase the accuracy of the intersection curve approximation. In the final iteration, when curve A satisfies some criterion of accuracy, the intersection curve is obtained as the intersection of a set of triangular facets of the polyhedron with the control plane. The intersection of each triangular facet with the plane is classified into the following three cases by comparing the position of its vertices with respect to the plane

1. No intersection (All vertices on one side)
2. Triangle lying on the plane leading to surface intersection (All vertices on the plane)

3. Proper intersection (Point/Line intersection - all other cases)

Only case 3 is of interest to the present problem and provides two points at either end of the segment. Since neighboring triangles to intersecting triangles of case 2 necessarily intersect the plane as in case 3 we do not find any starting points using case 2 intersections. The case of single point intersection is treated as a linear segment of length zero (ie. when the two end points are identical). Let G(u,v) be one of the parametric derivatives of F(u,v) ie. F_u or F_v to facilitate simultaneous discussion for u and v turning points. The derivative of F ie. the value of G, is evaluated at each of the end points of the segment. Since we are looking for the intersections of the two algebraic curves F(u,v) = 0 and G(u,v) = 0, we classify each linear segment (on the approximate curve for F(u,v) = 0) against the second function G(u,v) and look for a segment that has roots of G(u,v) = 0. The evaluation of G(u,v) at each of the ends of the segment and separation into one of the intervals [∞, -ε), [-ε,ε],(ε, ∞] allows an approximate root to be found. The various cases that could occur for each of the segments are illustrated in Figure 1. Cases a,

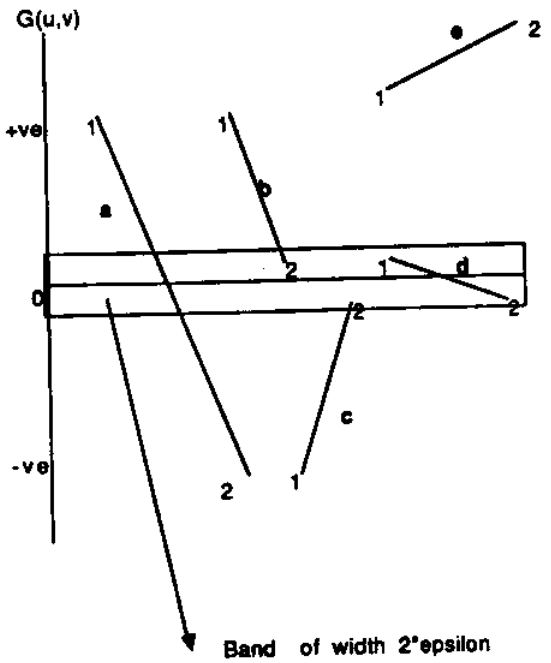


Figure 1: Various cases for starting point approximations

b, c, d represent points close to the intersection of the algebraic curves F = 0 and G = 0 and, thus, close to the turning points of F = 0 while case "e" represents points close to F = 0 but not G = 0. Here ε represents a region around zero that reflects the inaccuracy in the computation of F_u or F_v. In cases b and c point 2 is used as a starting point while in case "d" the mid-point is chosen as

the starting point. In case "a" we find the point (u, v) that corresponds to the zero of G to provide the best estimate of the turning point. A similar procedure is carried out for each of the segments and the collection of starting points for the turning points in the u and v direction are obtained separately.

The set of starting points provided by such a procedure contains some identical redundant points caused by end points of adjacent segments in one of the cases b, c or d. All such redundancies can be removed by simple comparison and, consequently, the set of starting points can be reduced to a minimal set. However, it is still possible that there are two or more starting points, close to each other, but clearly distinct, and which may lead to the same final turning point to the precision of the computation. Let $\{u_i, v_i\}$ $i = 1,2, \ldots n_u$ be the set of starting points for u-turning points and $\{u_j, v_j\}$ $j = 1,2, \ldots n_v$ be the corresponding set of v-turning points.

In our numerical experiments using a very coarse subdivision of the polyhedron, the above procedure followed by the iterative minimization and Newton method outlined in Section 5.3 was in a position to provide all turning and singular points of a variety of algebraic curves chosen for their complexity and diversity [30]. However, no guarantee exists than in general at least one starting point exists for every turning point nor that the iterative Newton technique will converge to the appropriate turning point. The first possibility did not occur in extensive experiments, giving us confidence that such occurrences are infrequent and that, therefore, *most* turning points can be computed efficiently with the rapidly convergent minimization and Newton method of Section 5.3. The second possibility, ie. the divergence likelihood of the iterative technique away from the appropriate point obviously depends on the subdivision size of the polyhedron. In order to provide a theoretical verification of the success of the above process to determine all turning and singular points the following additional computation is required. First splitting of the control surface using all parametric lines passing through all available turning points and all border points is performed. Next, the possibility of turning/singular points not at corners of the resulting subpatches is examined. This can be performed using a method akin to the intersection procedure developed by [12]. The u and v derivative surface control points for each subpatch are created. Using the convex hull property, such polyhedra can verify the absence of turning/singular points not at corners of the subpatches (all control points except corner points are on one side of the control plane). In case of failure, an additional call to the starting point/minimization-Newton technique is invoked within the smaller subdomain and the above

process of splitting at new turning/singular points and checking via the derivative patch is repeated. In case of repeated failure of the derivative patch criterion, the direct subdivision procedure of [12] may be employed to bracket the eluding turning and singular points as outlined in Section 2. This procedure terminates when the size of the enclosing box of each such point is below a prespecified tolerance. The center of this box is used as our estimate of the turning/singular point.

The above method of obtaining starting points and computing turning points is very useful because it does not depend on external information to initiate the procedure. In addition, unlike the method of [12], it allows computation of turning/singular points with very little initial subdivision and rapidly convergent Newton like iteration. Other methods of obtaining starting points for direct numerical solution techniques used to locate singularities have been employed by [4] where a marching method is started from an arbitrary starting point on a segment of the curve. The starting point for a direct solution method to locate singularities is obtained when a point on the marching path has derivatives that are below a certain small positive number. The key difference between such a technique and the method developed here is that no knowledge of initial starting points on every segment of the curve is required by our procedure. Our method to identify turning and singular points could, in principle, be combined with the lattice and marching methods used in [7, 8, 4] to trace an algebraic curve correctly because it provides all the necessary starting points on every segment of the curve and the singularities of the curve.

Before attempting to find the sets of starting points for the turning point computation in the above manner, it is important to exclude situations where all points in a segment are turning points like those on parametric lines. This can be done by first factoring out all such components, see Appendix I.

5.3 Direct Numerical Solution Methods

Each of the starting points for turning point computation is now used to solve the system of Eqs. (14) and (15) separately. Since we are interested in locating singular points as well as turning points, the chosen method must be able to handle the presence of points in the domain where the Jacobian is singular. These situations can be handled by the modified Newton algorithms of [29]. Alternatively, the same problem can be handled as a minimization problem using a modified Gauss-Newton method by minimizing the sum of the residuals of the two non-

linear equations [13]. A brief description of the two methods is given below.

Modified Newton technique [29] : The standard Newton technique to solve non-linear equations obtains a correction to the present position based on the solution of a linear system for the correction vector. The modification used here has two parts. Whenever the above correction has a total step length lesser than an allowed maximum step for this iteration, the method uses the above correction vector. But if it does not, then the method selects a step length and direction controlled by the maximum allowed step length for that iteration, the Newton correction vector and the gradient of the sum of squares of the residuals. This strategy results in an interpolation between classical Newton techniques and steepest descent techniques. This makes the direction of the correction sensible even when the Jacobian is near singular (a condition of interest to the present solution). The parameters of control are the accuracy on the sum of squares of the residuals and the initial choice of maximum step length, namely an upper estimate of the distance between the starting point and the solution point.

Modified Gauss-Newton method for the minimization problem [13] : The problem to be solved here consists of minimizing the following objective function within the domain of the algebraic curve

$$P(u,v) = F^2(u,v) + F_u^2(u,v) \tag{32}$$

The k-th iteration is governed by equations obtained from a Taylor expansion of the conditions for a minimum of a function of two variables, namely $P_u = 0$ and $P_v = 0$.

$$\begin{bmatrix} P_{uu} & P_{uv} \\ P_{uv} & P_{vv} \end{bmatrix} \begin{bmatrix} \delta^u \\ \delta^v \end{bmatrix} = \begin{bmatrix} -P_u \\ -P_v \end{bmatrix} \quad ie. \quad [G^k][\delta^k] = -[\mathbf{g}^k]$$
$$\mathbf{r}^{k+1} = \mathbf{r}^k + \alpha^k \delta^k \tag{33}$$

where $\mathbf{g}^k$ is the vector of first derivatives and G^k is the Hessian matrix of P(u,v) both evaluated at $\mathbf{r}^k$, δ^k is the direction of the step to be taken in the k iteration and α^k is the step length. The vector $\mathbf{r}^k$ is the approximate solution at the k-th stage of the iteration. The solution for the correction δ^k is obtained by a modified Cholesky factorization of G^k where the modification handles the case of G^k being indefinite or singular. Another important modification is the change in the solution procedure whenever the gradient of the function becomes lesser than a certain tolerance and the matrix G is not positive definite as in the case of saddle points. This method of solution implies the use of the second derivatives of the objective function ie. the use of third order derivatives of the algebraic curve and which, in turn, contributes to the reliability of the

solution from the convergence point of view.

In the event of encountering higher order singularities of multiplicities greater than three it may be required to modify the objective function with the known vanishing derivatives near such points. Whenever all derivatives of a certain order vanish, in the converging sequence approaching the solution, the minimization function can be augmented by an additional derivative. For example, near triple point singularities, the objective function should be modified to

$$P(u,v) = F^2(u,v) + F_u^2(u,v) + F_{uu}^2(u,v) \tag{34}$$

In both minimization and Newton methods the required derivatives for the solution are explicitly available at all points in the domain and this contributes to the reliability of these schemes. Because of the formulation of the minimization problem, an additional derivative of the algebraic curve is used at each stage of the sequence and this improves the reliability of its convergence to the solution. On the other hand, the Newton technique which uses information only up to second derivative was found to occasionally diverge unless the initial approximation of the solution is in its region of attraction.

In numerical experimentation with both methods on a large number of examples of planar algebraic curves selected for their complexity and diversity and with a coarse initial subdivision, it was found that the minimization scheme was stable in that it always produced a solution very close to the correct solution but usually with reduced accuracy. The Newton technique on the other hand provided a much better accuracy in the solution compared to the minimization method in all cases where it did not diverge. However, the Newton technique failed to converge in a small number of cases where the starting points were not very close to the required singularities. Hence, in order to exploit both the stability of the minimization method and the accuracy provided by the Newton method, a combination of the two was used [24] and proved successful in all the difficult test cases chosen. The above methods were used with the starting points provided by the procedure of section 5.2. Both methods were also used independently using the above starting points and with the minimization followed by the Newton scheme. The results of some of these experiments are given in Section 7, where minimization followed by the modified Newton scheme was employed.

6 Curve Tracing

6.1 Partition at Significant Points

This section describes a technique of tracing an algebraic curve given its Bernstein representation of the curve and the set of its significant points. The first step of our tracing process, is to split (partition) the control patch in such a manner that the u and v coordinates of all significant points define the borders of the subpatches and the actual significant points lie only at corners of the subpatches (see Figure 2). It is possible to split a patch into its subdomain

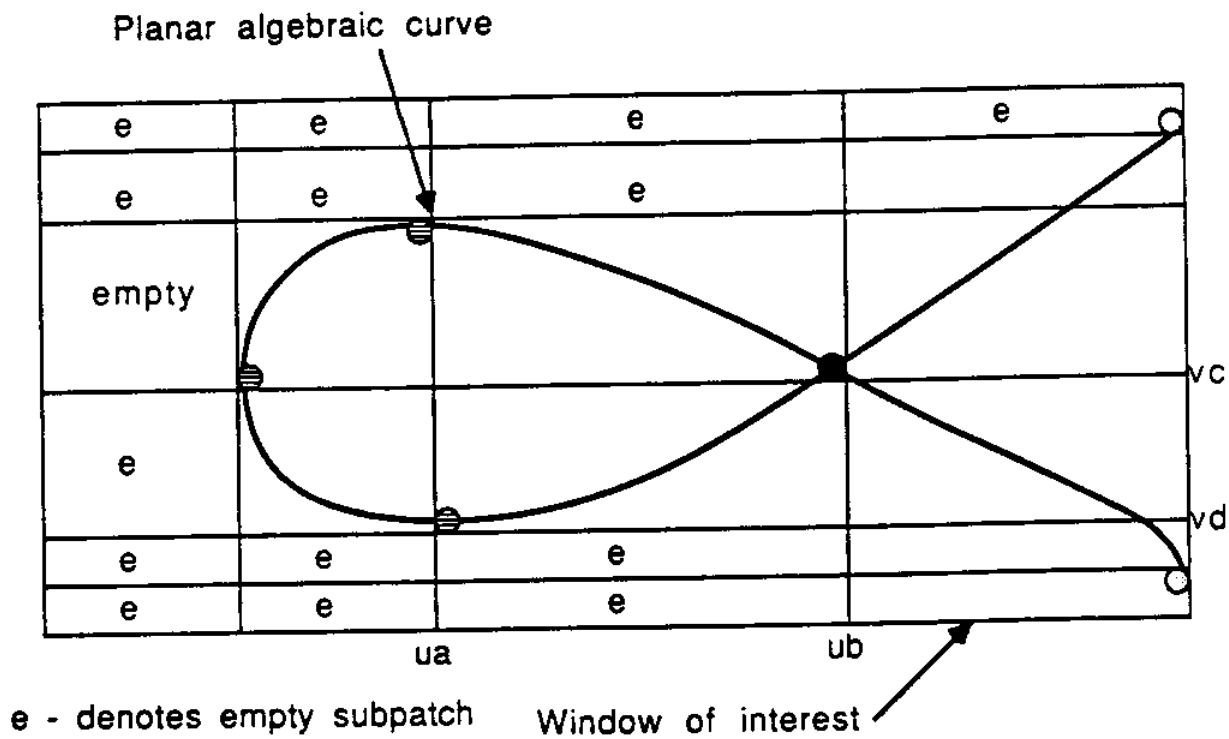


Figure 2: Splitting of polynomial patch at the significant points

patches along parametric lines using a subdivision algorithm for B-spline surfaces [6, 22]. Such algorithms allow non-uniform refinement of the polyhedron by adding new knots to the initial knot vectors of a B-spline surface. By adding knots of multiplicity m+1 and n+1 at specific u and v values it is possible to extract the subpatch corresponding to a given parametric subdomain $[u_a, u_b]x[v_c,v_d]$ hereafter referred to as a subpatch. Each subpatch is separately traced and the solutions from each of the matrix of subpatches are combined to provide the overall solution.

The equation of each subpatch in the Bernstein basis provides the representation of the intersection curve in the subdomain $[u_a,u_b]x[v_c,v_d]$. The curve may contain singular, turning or border points only at corners of this subdomain. However, this subdomain may contain more

than one segment of the algebraic curve (see Figure 2), each of which is monotonic.

6.2 Tracing Curve Segments Represented by One Subpatch

The tracing of the curve segments represented by one subpatch is based on a faceted approximation of the subpatch and representation of the curve segments as intersections of this faceted approximation with the control plane. The particular faceted approximation chosen is based on the polyhedron formed by a triangulation of the control polyhedron of the Bezier subpatch. Before we proceed with the specifics of triangulation and intersection of triangular facets with the control plane, we describe some preliminary computations which should be carried out a priori in order to efficiently eliminate non-intersecting subpatches.

6.2.1 Elimination of Empty Subpatches

It is possible (and, in fact, very frequent) that a given subpatch, produced by the splitting at significant points does not contain a portion of the algebraic curve (see Figure 2). It is also possible that the control polyhedron intersects the control plane, but the underlying control surface itself does not intersect the plane which occurs whenever the control surface is close to the plane. Since we have computed all turning points and split the surface at such points, we are guaranteed that any isolated intersection of polyhedron occurring completely in its interior does not contribute to an intersection segment. Hence we screen each subpatch for proper (monotonic) intersections and proceed with the triangulation and intersection with a plane only if the following conditions hold true

1. There is at least one sign change among the w_{ij}'s. A sign change is identified when w_{ij} exist in more than one of the following intervals of the real w axis $[-1, -\varepsilon)$, $[-\varepsilon,\varepsilon]$,$(\varepsilon,1]$, where ε is a small positive number, denoting the accuracy used to decide whether a w_{ij} coordinate may be considered to lie on the control plane. Henceforth, we refer to this ε as ε_{plane}.
2. There is at least one sign change along the edge (border) control points of the subpatch. This condition is required so as to prevent attempting to intersect subpatches whose control polyhedra at some finite level of subdivision still intersect the plane (usually in small loops) but the underlying surfaces themselves do not intersect. As stated earlier, we are guaranteed not to have small loops within our sub-domains. In fact, curve segments within our sub-domains are monotonic and start and end at distinct borders. This is the reason why we test edge control points for at least one sign change.
3. We also recognize cases where intersection segments are entirely on one or more of the edges of the subpatch.

A subpatch is first processed to verify the above conditions and the tracing procedure is

attempted only when these conditions are true.

6.2.2 Triangulation and Computation of Individual Triangle/Plane Intersections

As a way of increasing the intersection accuracy from the first step and isolating the effect of vanishing derivatives at different corners of subpatches we introduce m+2 and n+2 uniformly spaced internal knots in the u and v directions, respectively, and find the corresponding new set of control points of the new, geometrically identical, B-spline surface. After the above preliminary subdivision, the control polyhedron of the B-spline surface is triangulated. The particular scheme used here is to subdivide each three dimensional quadrilateral formed by the control points into two triangles by joining either control vertices $\mathbf{T}_{ij}$ and $\mathbf{T}_{i+1\,j+1}$ or $\mathbf{T}_{i\,j+1}$ and $\mathbf{T}_{i+1\,j}$. An alternate method of a symmetric triangulation scheme of a parametric tensor product surface patch can be found in [41]. The above triangulation may be expressed by the following two sets of coordinate triplets over all the control points

$$\begin{aligned} &\textit{Case } 1 \cdot \quad T_{ij0} = [\,\mathbf{T}_{ij},\ \mathbf{T}_{i+1\,j},\ \mathbf{T}_{i+1\,j+1}\,] \quad T_{ij1} = [\,\mathbf{T}_{ij},\ \mathbf{T}_{i\,j+1},\ \mathbf{T}_{i+1\,j+1}\,] \\ &\textit{Case } 2 \cdot \quad T_{ij0} = [\,\mathbf{T}_{ij},\ \mathbf{T}_{i+1j},\ \mathbf{T}_{i\,j+1}\,] \quad T_{ij1} = [\,\mathbf{T}_{i+1\,j},\ \mathbf{T}_{ij+1},\ \mathbf{T}_{i+1\,j+1}\,] \end{aligned} \tag{35}$$

Each quadrilateral consists of two triangles each denoted $\mathbf{T}_{ijk}$, where k = 0 or 1 and is the index specifying one of the two triangles whose common edge extends from $\mathbf{T}_{ij}$ and $\mathbf{T}_{i+1\,j+1}$ or $\mathbf{T}_{i+1\,j}$ and $\mathbf{T}_{i\,j+1}$. In this work, for simplicity we adopt the first of the two possible triangulations for each of the quadrilaterals formed by the polyhedron. This choice is not appropriate in cases where all of the four edges of the quadrilateral formed by the two triangles have intersections, a situation occurring in the presence of small curve features such as constrictions (see Figure 3). When such a situation occurs, knots are introduced at the parameter values of the quadrilateral's mid point in order to improve the approximation of the polyhedron in this region. Such a procedure will eventually provide quadrilaterals that have only two intersections on the edges because all singularities are computed a priori and are located at borders of the subpatches. This approach improves on the capabilities of available techniques which do not compute such points a priori. Each of the triangles $\mathbf{T}_{ijk}$ is then intersected with the plane to get a piece of the piecewise linear approximation of the algebraic curve. The results of this intersection are retained to a high level of detail so that they may be used later to directly form a connected sequence of these intersection segments.

The procedure of intersecting the triangles with the plane is one of the most repeated parts of the computation and grows as O(pq) where p, q are the number of control point rows of the

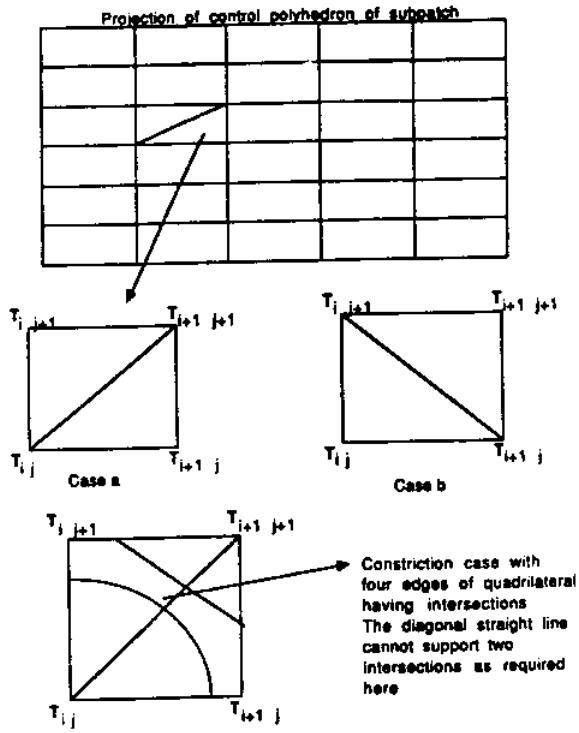


Figure 3: Choice of Triangulation

subpatch in the u and v directions. Since the polyhedron is iteratively refined in order to satisfy a certain level of accuracy, this step is also repeated in every iteration and along with the refinement of the polyhedron using a subdivision algorithm [6, 22] forms the most computation intensive intersection stage. Improvement in the efficiency of this stage will consequently drastically affect the performance of the algorithm. Each triangle-plane intersection for one iteration can be thought of as entirely independent of other triangle-plane intersections and could, in the future, be parallelized.

The intersection of each triangle with the plane is a straightforward but important segment of the process. In what follows, we outline the information obtained from this intersection. The three vertices of each triangle are obtained from (35) and the number of sign changes of their w_{ij} coordinates with respect to the plane $w = 0$ are recorded. Sign changes are identified according to whether each w_{ij} is in $[-1,-\varepsilon),[-\varepsilon,\varepsilon],(\varepsilon,1]$, where $\varepsilon = \varepsilon_{plane}$. Based on the number of vertices that lie on the plane, ie. in $[-\varepsilon,\varepsilon]$, and the sign changes among the three vertices we distinguish the following cases

1. No intersection (all vertices on one side)
2. Triangle on plane (all vertices on plane, degenerate intersection)
3. Edge intersection (two vertices on plane, third vertex not on plane)

4. Single vertex (one vertex on plane, two vertices on same side of plane)
5. Line intersection with one end on vertex (one vertex on plane, two vertices on either side of the plane)
6. General intersection (no vertices on plane, one vertex on one side of the plane and two on the other side)

Cases 3 through 6 are hereafter collectively called proper intersections. Such information can be used to connect the segments together in an unambiguous manner without the need to exhaustively compare ends with every other segment [30].

The above phase gives us a number of disconnected linear segments whose end points are close to the curve. The quality of approximation of each of these points may be verified by evaluating the intersection curve at these points to see if an accuracy criterion is met. The exact nature of this criterion is not important for now but only that a certain point satisfies this criterion or not. Each point appears twice as the start and end point of adjoining segments. Since there may be more than one branch in the intersection, these disconnected segments are not, generally, in order and the check for accuracy would be repeated for each point twice. In order to avoid this, we connect all these disconnected linear segments into piecewise linear splines extending from border to border before evaluation of each of the points for accuracy. Further, as pointed out earlier, in the initial stages of the iteration it is possible to have spurious loops appearing in the polyhedral intersection which do not form part of the real intersection. This procedure of connecting segments before their evaluation for accuracy also allows us to remove any such spurious intersections by using the monotonicity property of complete intersection segments within subpatches, ie. that they start and end at distinct borders.

6.2.3 Connection Phase Within a Subpatch

The intersections corresponding to each triangle-plane intersection together form a disconnected set of intersections which possibly contain redundancies and insufficient information because planar facets have been used to approximate a curved surface. The intersection information for all the triangles is scanned and triangles having redundant information are marked as having no intersection and triangles with "incomplete information" are treated as triangles that lie on the plane [30].

The connection procedure depends upon

- the detailed information of the triangle-plane intersection

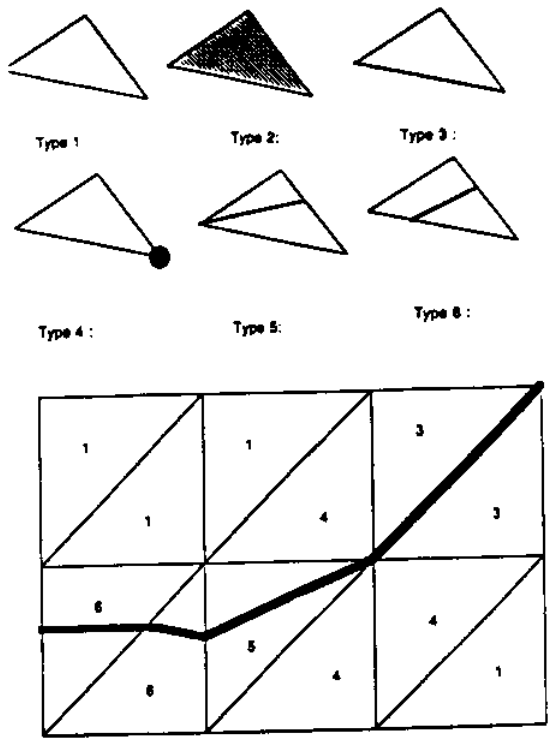


Figure 4: Types of triangle-plane intersections

- the monotonic form of segments, and,
- the property that all branches start and end at borders of the subpatches.

The basic procedure involves searching for starting points of intersection branches along the borders of the subpatch or of triangles with degenerate intersections. Once such an intersecting triangle has been found we proceed to extend this initial intersection through neighboring triangles till it meets another border. The above process is then repeated till no starting intersection can be found. The major components of the above scheme of connecting the disconnected segments involves the finding of a starting triangle and the finding of neighboring triangles across edges and vertices. The starting triangle is found by searching at all corners in sequence and then the borders of the patches. For each corner the search is carried out along the border of triangles marked as degenerate if the corner is either a turning or a singular point. A starting triangle is found by first looking along such corners and borders till a triangle with a proper intersecting segment is found. Of those, we then select one which has one neighboring triangle with a proper intersection on one side and none on the other side. When such a starting triangle has been found, the search for more starting triangles is stopped until this segment is completed. If the start triangle was found while searching around a corner with a turning or singular point a small number of collinear segments are added between the appropriate corner

and the end of the intersecting segment. This segment is then extended by locating neighbor segments across edges and vertices to form a connected intersection segment. Neighbors across edges are straightforward to find using the information about the edges on which each intersecting segment ends as detailed in the previous section. The neighbors across vertices require the scanning of all neighbors across that vertex to locate the correct neighboring intersection. Finally this growth of a segment across edges and vertices ends at a border or a degenerate triangle. A border ending signifies that the segment does not pass through any significant point and will be later connected together with segments in adjacent subpatches as outlined in section 6.5. If it ends at a degenerate triangle, the closest corner is located using the index of the segment end point and the corner indices to unambiguously locate the corner associated with such a triangle.

After completing one such branch, we attempt to find more starting triangles and locate more branches until no starting triangles can be found and we obtain a list of branches each of which is a connected sequence of points. These points are given in the u,v space of the entire patch being intersected.

6.3 The Accuracy of the Algebraic Curve Approximation

The connected list of points are approximate points that are close to the intersection curve in the parametric space. Their mappings through the parametric surface (4) into three-dimensional space will give points that lie on the parametric surface but do not necessarily lie close enough to the algebraic surface.

Let $\mathbf{P} = (x_i, y_i, z_i, w_i)$ be the point of the parametric patch corresponding to a parametric point $Q_i(u_i, v_i)$ on the approximate intersection curve. The value of the algebraic surface equation at point **P** given by $H(x_i, y_i, z_i, w_i)$, where the coefficients of H are normalized as in (5) gives an approximate criterion by which the proximity to the algebraic surface can be judged and, in the limit, as the absolute value of this function becomes smaller than a small positive number, ε, point **P** may be considered to lie on the algebraic surface as well. However the non-dimensional number ε is not, by itself, an indication of the distance between the algebraic surface and the point **P**. An estimate of the distance δ between point **P** and the algebraic surface can be computed from $\delta \approx |H| / |\nabla H|$ where the right hand side is evaluated at point **P** and $|\nabla H|$ is assumed to be non-zero, i.e. **P** is not close to singularities of the algebraic surface $H = 0$. This

linearized estimate is a good approximation only after $|H| < \mu$ where μ is a small positive number and the distance estimate used only after the criteria $|H| < \mu$ is satisfied. Distance considerations are of practical importance in solid modeling applications and the accuracy requirements for these intersection curves are normally given in terms of distance. The algorithm developed here employs the above criterion by specifying ($\delta \leq \varepsilon_{3D}$) to judge the accuracy of points in three-dimensional space.

6.4 Iteration

If every point Q_i from the list of branches in the domain of each subpatch and the mid points of every pair (Q_i, Q_{i+1}) is considered acceptable according to criteria established in the Section 6.3, the iteration is complete and the points provide a piecewise linear approximation of the intersection in the parametric space of the patch. It is important to note here that the chordal approximation between successive points Q_i must also satisfy the closeness criteria. This verification may be carried out by solving a maximization problem for the distance between points on the chordal segment and the algebraic curve. Such a procedure would be computationally expensive and, for our present implementation, we assumed that the maximum deviation happens at the mid-chord point ie. the chordal approximation is good if the mid point is within the same distance criteria used for the end points. The more expensive procedure may be, if necessary, used at the end after the above simpler criterion is satisfied.

If a point Q (either ends or mid-points of each linear segment) does not satisfy the accuracy requirement, then its coordinates u and v are used to insert a knot in the u and v knot vectors, respectively, of the patch which generated these intersections by faceting. This procedure is carried out for all points. In order to avoid introducing multiplicities at a certain value of u or v because two distinct failing points have a common u or v, we first obtain a list of the failing points, separate the (u,v) pairs into two sorted lists of u and v values and remove any duplications in the u or v to given tolerance ε_{sort}. Once the new knot vectors are created, a subdivision procedure is invoked [22]. This forms a computationally intensive part of the iteration step and may be suitable for parallel implementation. The above subdivision procedure allows the simultaneous insertion of an arbitrary number of non-uniformly spaced knots. Non-uniform insertion of knots allows us to introduce a finer approximation at regions where the curve is not well approximated, providing the basis for an adaptive algorithm. The finer control polyhedron will now be closer to the surface and, hence, the repetition of the intersection of

triangles, connection and testing for accuracy will yield a new intersection approximation that is better than that obtained in the preceding stage. This process of iteration may be continued till all points in all intersection branches of the subpatch are of sufficient accuracy.

Once the above procedure is executed to some coarse level of subdivision, an efficiency enhancement to obtain high accuracy can be achieved as follows. The u or v coordinate of approximate intersection points Q_i and the midpoints of segments Q_i,Q_{i+1} in the parametric space can be fixed and a Newton iteration adopted to minimize the value of F(u,v). The choice of fixed parameter can be made according to the slope of the algebraic curve at the point. Since the maximum of change of slope of any segment of the algebraic curve is a right angle, the parameter u is kept constant for slopes lesser than 45^0 with respect to the u-axis and v constant for slopes exceeding 45^0 with respect to the u-axis. Since we are interested only in bringing the distance $|H| / |\nabla H|$ below ε_{3D}, the iterations need to be carried out only till this criterion is met.

6.5 Connection Phase Across Subpatches and Overall Solution

The above tracing procedure is repeated for each one of the subpatches generated by splitting of the surface at all significant points. It is possible (and, in fact, frequent) that such partition produces splitting of segments at regular points at which no special reasons exist for discontinuity. This is a by-product of the particular splitting procedure employed. A method of correcting this artificial discontinuity, present when a finite amount of subpatch subdivision is carried out, and combining the solutions from each of the subpatches into a complete intersection curve has been devised and is outlined in [30]. Once this procedure is completed, a list of separate segments of the curve starting and ending at all significant points is produced. The tracing procedure used above does not produce lists which contain isolated points as solution points. In order to rectify the final solution we add one-point segments corresponding to all isolated points. These can be determined by comparing the ends of all segments with the list of significant points. The segments ending at turning points may be connected across such points without difficulty because these points have been computed a priori. In the case of segments ending at singular points the desingularization method studied in [4] may be employed to connect the proper intersection branches across such points.

As mentioned in Section 3.2, we trace the intersection curve between the B-spline patch and the algebraic surface by splitting the B-spline patch into its constituent rational polynomial

patches and interrogating each one of them separately. The above method describes the tracing of the intersection of one such rational polynomial patch with a given algebraic surface. Similar intersection problems are solved for all other patches to get similar solutions of connected lists of points that all end at significant points. It must be noted here that each of the above intersection problems is completely independent of the solutions of the other problems and, hence, the above process is suitable for a parallel implementation. The solutions may then be combined into larger segments across border points by simply matching end points between the segments of neighboring polynomial patches. Finally, the points in each one of the segments is mapped into three-dimensional space to obtain the actual intersection curve. The points may also be retained in the parameter space for some applications, such as processing of trimmed patches.

7 Numerical Experiments

7.1 Preliminary Remarks

The method outlined in earlier sections has been implemented and tested to verify its reliability and to evaluate the accuracy achieved in the various stages of computation. The implementation is written in C on a DEC Vax Station II GPX machine operating under the Unix system and employing the Silicon Graphics Remote Graphics Library [15] to drive a Silicon Graphics IRIS 3030 workstation.

The method was implemented and tested in three stages :

1. The tracing of the intersection curve given its representation in the monomial basis and its significant points ie. the process of obtaining various branches of the curve that are linear spline approximations of the curve which run from significant point to significant point as explained in Section 6.
2. The computation of the turning and singular points of an algebraic curve given its monomial coefficients using the methods of Section 5. The method of Section 4 was also used to compute turning and singular points of only low degree intersections such as planar sections of rational biquadratic and bicubic patches. Border points were computed based on the method of Section 4 using the Bernstein representation.
3. The overall solution of the intersection problem between algebraic surfaces up to degree two (ie. planes and quadrics) and parametric rational B-spline surfaces up to degree three.

The first two stages of the above testing require conversion of the algebraic curve equation from the monomial to the Bernstein basis which was performed according to [11]. The above separation into stages was used so as to be able to evaluate each element of the method independently and to the fullest potential of that part of the solution. In our current implementation, for example, the tracing portion is not limited by the degree of the curve, so that curves of arbitrary degree can, in principle, be traced. Similarly, given the representation of any algebraic curve it is possible to find its significant points using the procedure of Section 5 which is not affected as seriously by degree as the procedure of Section 4.

7.2 Computation of Significant Points and Tracing of Known Curves with Varied Features

Extensive examples of planar algebraic curves can be found in Walker [43] and Lawrence [20] and more recently in Geisow [12] who used them to test methods to trace algebraic curves. A number of examples drawn from these references have been traced using the method described above. The standard form of these curves given in the literature is the monomial form with integer coefficients. The characteristic polynomials of these curves were first derived using elimination techniques in rational arithmetic. In most instances exact solutions of these equations in terms of rational numbers or numbers involving radicals of rationals are possible and, hence, the significant points of these curves were obtained with sixteen decimal digit accuracy, the floating point precision used throughout all the examples discussed. The remaining real solutions within the window of interest were obtained using Jenkins' algorithm [16] implemented within [23] to the best possible accuracy employing 16 decimal digit floating point arithmetic. A similar method was employed in the derivation of border points for such curves.

The range of curves chosen involve a maximum of degree six in each of the two variables and exhibit double, triple or quadruple singular points. For double points, special cases involving an isolated point (hermit point or acnode); node (crunode or self-intersection) and cusp (spinode) have been studied. Cusps of the first and second kind have been studied, i.e. when curve arcs lie on either or one side of the double tangent. Finally, double cusps (tacnodes or points of osculation) at which the two arcs extend in both directions of tangents have also been tested. Some of these examples have, also, a number of turning and border points depending on the window of interest.

The tolerances introduced in Section 6 were held constant and equal to the following values

- $\varepsilon_{sort} = 1.0 \times 10^{-6}$, used to sort the u and v coordinates of significant points and remove duplicate u or v values
- $\varepsilon_{plane} = 1.0 \times 10^{-12}$, used to evaluate sign changes of coefficients w_{ij} of each of the subpatches obtained after splitting (a number that must be chosen to reflect the estimated error in their computation).
- $\varepsilon_{2D} = 1.0 \times 10^{-3}$, used to check the proximity of each point from the algebraic curve.

Note that the weights w_{ij} in the Bernstein basis have been normalized so that $|w_{ij}| \leq 1$ and, therefore, the surface $w = F(u,v)$ lies in [-1,1] within the window of interest. When only the equation of the algebraic curve is given and there is no generating three-dimensional intersection

problem, the accuracy criterion of subsection 6.3 cannot be implemented as stated. However, when $|F(u,v)| \leq \varepsilon << 1$ is obtained, a distance estimate from an approximate point u, v close to the curve to the curve $F = 0$ can be approximated by $\delta = |F|/|\nabla F|$ evaluated at the approximate point, valid when $|\nabla F| \neq 0$. The accuracy criterion used to trace the algebraic curve examples of this Section is $\delta \leq \varepsilon_{2D}$, where ε_{2D} is as stated above.

The same set of examples, as used to study the tracing method, was also used to test the computation scheme for turning and singular points. The starting points were computed using the scheme of subsection 5.2. These were then used to start the minimization method followed by a Newton-Raphson scheme as explained in Section 5.

The tracing method was tested with full sixteen significant digit accuracy on both the monomial coefficients and on the significant points while the computation of the significant points from the exact monomial coefficients was done separately. The results of the two independent tests on a number of test examples are presented in the following pages showing the details specified below

- The monomial representation of the curve and the window of interest (primary data)
- The correct (to fourteen significant digits) and computed significant points of the curve using the numerical techniques of Section 5 for the turning and singular points and the Bernstein subdivision method for border points.
- A picture of the actual trace of the curve showing the boundaries of the overall window of interest and of the subwindows created by splitting at all significant points (thick and fine lines, respectively).

The following is a list of the examples that we have included in this report

1. Double point - (acnode, isolated point [43])
2. Double point - (crunode, Tschirnhausen's cubic [43])
3. Double point - (crunode, folium of Descartes [43])
4. Double point - (crunode, torus-plane intersection at a saddle point)
5. Double point - (cusp of the first kind, Isochrone [20])
6. Double point - (cusp of the first kind, Cardioid [20])
7. Double point - (cusp of the second kind, Ramphoid [20])
8. Double point - (Double cusp ie. tacnode, Hippopede [20])
9. Double point - (Geisow's tacnode and crunode [12])
10. Double point - (Geisow's multiple crunode case [12])

11. Triple point [43]
12. Quadruple point [43]
13. Reducible curve - (crunode)
14. Reducible curve with parametric line component
15. Double point - (Cusps in a Bicorn [20])
16. Constriction resolution

As can be seen from Figures 5 through 20 summarizing the results of the above examples, the accuracy obtained on border points is very good and typically involves more than twelve significant decimal digits of accuracy. These were obtained using the Bernstein root solving technique of [19]. The same accuracy was also obtained for all the turning points in the above cases when the numerical methods of Section 5 were employed. In the case of singular points, the accuracy obtained varied from case to case depending both on the type and order of the singularity. The accuracy deteriorated with the order of the singularity because such curves must be of a higher degree to support such a singularity. The higher degree of the curve contributes to greater error in the evaluation of the functions and their derivatives thereby contributing to the reduced accuracy. Asymmetric singularities also reduced the accuracy in the position of the singularities because points farther satisfy the defining equations. In the worst case, only three decimal digit accuracy was obtained while in most cases about six to nine decimal digits are accurate. Further in some cases, the accuracy of singular point is substantially different between the u and v coordinates. The following paragraph discusses the origin of this discrepancy and possible ways to correct it.

While solving for turning points actually coinciding with singular points using the direct numerical techniques of Section 5, more than one solution is obtained since such points are the limiting case of coincident u and v turning points. This is because two or more different starting points are found for each singular point. The different starting points lead to different solutions which are within a small disc covering a neighborhood of the singular point. The solution that has the minimum sum of the squares according to equation 32 for v-turning points is selected to represent the point. If the point representing the solution was started from a v-turning point then the u coordinate is more accurate than the v coordinate. Similarly if it was started from a u-turning point then the v coordinate is more accurate than the u coordinate. If two or more turning points lead to a common solution within a neighborhood, the one with the minimum sum of

squares $F^2 + F_u^2 + F_v^2$ is chosen. This accounts for the difference in accuracy between the u and v coordinates of the singular points since it is only a turning point at which the corresponding other derivative is very small.

One method of improving such a solution is to treat such points obtained from v-turning points to be used as starting points for solutions as u-turning points whenever the corresponding derivative falls below a threshold value. Another method is to solve the minimization problem directly for the singular point instead of a u-turning point or a v-turning point ie. by directly minimizing $F^2 + F_u^2 + F_v^2$. However, in this case although the symmetry of the accuracy is better, the actual improvement of the solution in terms of the above residual is the same. There is a limited scope for improvement of the accuracy of these singular points using the ideas expressed above. For any substantial increase in the accuracy of these points, floating point arithmetic with extended precision may be required to obtain the precision required in the final results.

7.3 Intersections of Algebraic Surfaces with Rational Biquadratic and Bicubic Patches

Although the computations of significant points of the curve or the tracing scheme is not limited by degree of the curve, the computation of the representation ie. the computation of w_{ij} from primary data according to equation 11 using a direct symbolic method has been implemented only for intersections of planes and quadrics with rational biquadratic and bicubic B-spline surfaces. A number of examples showing intersections between various combinations of these surfaces are presented to illustrate the capabilities of the methods developed in this report. For each of these cases, the input consists of the position and orientation of local coordinate systems of the algebraic surface and the patch within a world coordinate system and a representation of each of the surfaces in their respective local coordinate systems. The algebraic surface is represented by an implicit equation describing the surface in its local coordinate system while the rational B-spline patch is represented by the specification of degree, the knots and the location of the control points in its local coordinate system.

The tolerances introduced in this report were held constant and equal to the following values

- $\varepsilon_{sort} = 1.0 \times 10^{-6}$, used to sort significant points (reflecting of the accuracy to which the significant points have been computed).
- $\varepsilon_{plane} = 1.0 \times 10^{-8}$, used to evaluate sign changes of weights w_{ij} of the subpatches obtained by splitting (a number reflecting the estimated error in their computation)

- $\epsilon_{3D} = 1.0 \times 10^{-3}$, used to check the distance accuracy of an approximate intersection point

We have chosen the following examples of rational polynomial parametric surface patches to illustrate the handling of various features of the curve of intersection by our algorithm. In particular, the following examples illustrate the ability of the method to discover and describe all segments of the curve irrespective of the size of the segment/feature/loop and to handle tangencies and near tangencies of the intersecting surfaces. The numerical stability of the method for some of these examples was tested by positioning two surfaces at a tangency position and at a very small distance (0.1% of the characteristic dimension) away from the tangency position in two opposite directions. The intersection curve reflected the different topology in each of the three cases tested. Isolated points at the center of a Bezier patch were also obtained by creating tangencies with quadric surfaces at that point and were identified as a single point segment. Such an example is not shown in the figures because it is difficult to show such an entity in the graphics. The specific examples illustrated below are

1. A biquadratic Bezier patch and a plane leading to an algebraic curve with four turning points
2. A biquadratic Bezier patch and a plane leading to a parametric line intersection component, an example where factorization is useful
3. A torus represented as a rational biquadratic B-spline patch intersected with a plane tangent at an inner saddle point, illustrating a self-intersection.
4. A torus represented as above with a nearly tangent plane leading to a very small loop
5. A torus represented as above with a plane tangent to it at two points on either side of the equatorial plane, leading to a double crescent-like curve with two self-intersections.
6. A bicubic patch with a plane leading to an algebraic curve with one loop
7. A bicubic patch with a plane leading to an algebraic curve with one self-intersection
8. A torus intersecting with a cylinder of radius equal to the smaller torus radius leading to a curve with two self-intersection points
9. The radius of the cylinder in example 8 diminished by 0.1% to illustrate the sudden change in the topology of the intersection curve leading to an algebraic curve with two loops.
10. The radius of the cylinder in example 8 increased by 0.1% to illustrate the sudden change in the topology of the intersection curve leading to an algebraic curve with two loops of different form than in example 9.

11. A torus with a cylinder of radius smaller than the smaller of the radii of the torus producing four independent loops
12. A torus with a sphere with inner tangency leading to an algebraic curve with self-intersection
13. Cylinder - Elliptic paraboloid intersection to produce a cusp
14. Sphere - cone intersection producing two loops
15. Sphere - cylinder intersection producing a double point when the sphere is positioned just touching the inner surface of the cylinder
16. Cone - cylinder intersection leading to two double points
17. Cylinder - cylinder intersection with both cylinders of same radii and position to produce two double points
18. Bicubic patch with a cylinder leading to two loops
19. Bicubic patch with a cone leading to two loops
20. Bicubic patch with a sphere positioned in its concave side to produce four independent loops
21. Bicubic patch with a sphere of slightly larger radius positioned at the same point as in the previous example

8 Summary and Conclusions

This paper presents the basic components of a new method for handling the intersections of algebraic surfaces and piecewise continuous rational polynomial parametric surface patches. Such problems, in general, reduce to the more fundamental problem of tracing a planar algebraic curve within a rectangular domain. The method combines the advantageous features of an explicit representation of the intersection curve in the Bernstein basis, the computation of border, turning and singular points and adaptive subdivision techniques to provide the basis for a reliable solution procedure. Theoretically, the algorithm is not restricted by the degree of the intersecting surfaces and, in practice, has been used and tested for intersections of up to degree two algebraic surfaces with up to rational bicubic patches.

This research was pursued mainly to address the issue of handling a common problem in presently available algorithms namely that caused by singularities and the presence of small isolated features in the intersection curve such as small loops. Most marching algorithms have difficulty in handling singularities or closely spaced features. Recent work in the desingularisation of curves and in the use of power series expansions about a point and the computation of significant points have improved these techniques but a common problem still

Figure 5: Double point - (Self intersection in Tschirnhausen's cubic, [43])

$F(u,v) = 15v^2 - 5u^2 - u^3 = 0$ in [-5.0 2.0] x [-2.0 2.0]

Significant Points :

Correct (u, v)		Computed (u, v)	
Border points :			
2.0000000000000	1.3662601021279	2.0000000000000	1.3662601021279
2.0000000000000	-1.3662601021279	2.0000000000000	-1.3662601021279
Turning points :			
-5.0000000000000	0.0000000000000	-5.0000000000000e+00	+1.1035308832600e-16
-3.3333333333333	1.1111111111111	-3.3333333333333e+00	+1.1111111111111e+00
-3.3333333333333	-1.1111111111111	-3.3333333333333e+00	-1.1111111111111e+00
Singular points :			
0.0000000000000	0.0000000000000	-3.4441669592500e-09	+1.3389643917060e-16

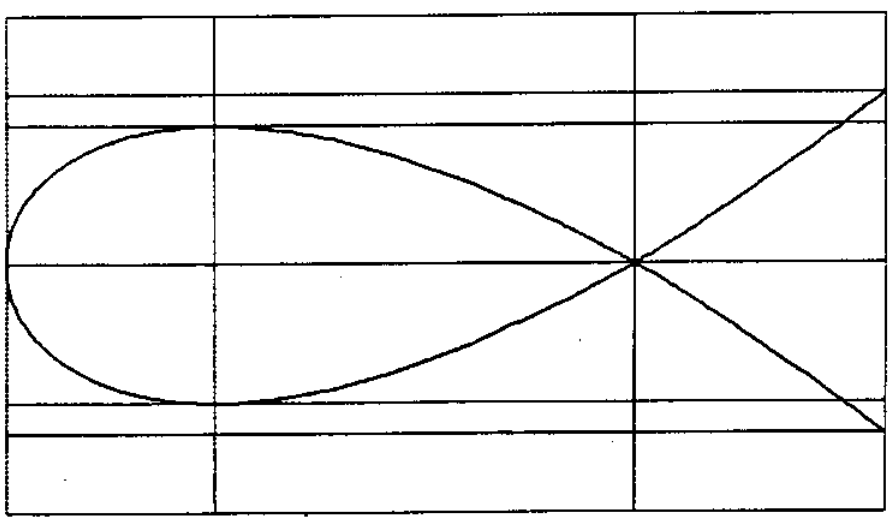


Figure 6: Double point - (acnode, isolated point)

$F(u,v) = u^3 + u^2 + v^2 = 0$

Window of interest : [-2.0 1.0] x [-1.0 1.0]

Significant Points :

	Correct (u, v)	Computed (u, v)	
Border points :			
-1.4655712318767	1.0000000000000	-1.4655712318767	1.0000000000000
-1.4655712318767	-1.0000000000000	-1.4655712318767	-1.0000000000000
Turning points :			
-1.0000000000000	0.0000000000000	-1.0000000000000e+00	+4.2638000362449e-21
Singular points :			
0.0000000000000	0.0000000000000	-3.0065687812916e-08	+6.2908533115179e-18

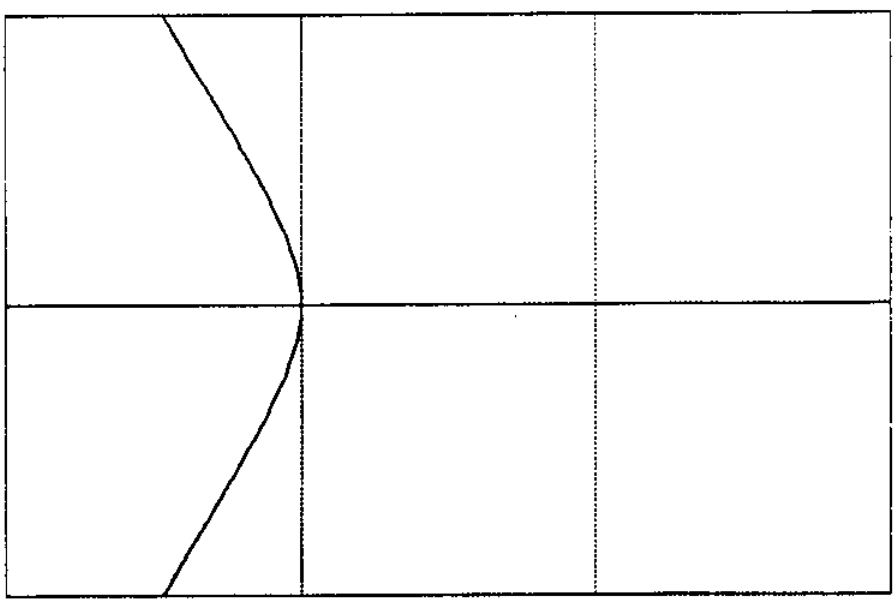


Figure 7: Double point - (crunode, folium of Descartes)

$F(u,v) = u^3 - 3uv + v^3 = 0$

Window of interest : [-3.0 2.0]x[-2.0 2.0]

Significant Points :

	Correct (u, v)	Computed (u, v)	
Border points :			
1.1071475644353	-2.0000000000000	1.1071475644353	-2.0000000000000
-2.9513730355914	2.0000000000000	-2.9513730355914	2.0000000000000
Turning points :			
1.2599210498948	1.5874010519681	+1.2599210498948e+00	+1.5874010519682e+00
1.5874010519681	1.2599210498948	+1.5874010519682e+00	+1.2599210498948e+00
Singular points :			
0.0000000000000	0.0000000000000	+1.8790795284973e-04	+3.5310367731043e-08

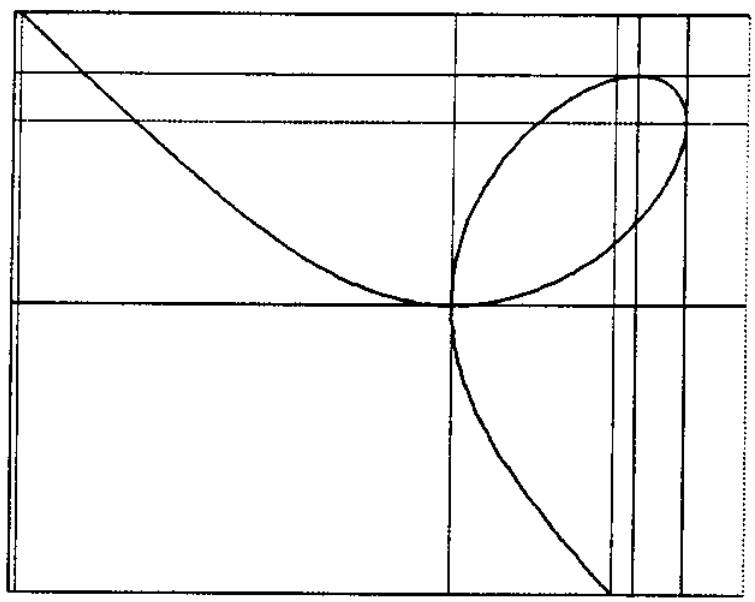


Figure 8: Double point - (crunode, torus plane intersection at a saddle point)

$F(u,v) = u^4 - 7200u^2 + 2u^2v^2 + 7200v^2 + v^4 = 0$

Window of interest : [-90.0 90.0] x [-30.0 30.0]

Significant Points :

Correct (u, v)		Computed (u, v)	
Turning points :			
-84.852813742385	0.0000000000000	-8.4852813742385e+01	-2.5950233745632e-15
-51.961524227066	30.000000000000	-5.1961524227066e+01	+2.9999999999999e+01
51.961524227066	-30.000000000000	+5.1961524227066e+01	-3.0000000000001e+01
84.852813742385	0.0000000000000	+8.4852813742387e+01	-9.0220709858554e-16
51.961524227066	30.000000000000	+5.1961524227066e+01	+3.0000000000001e+01
-51.961524227066	-30.000000000000	-5.1961524227066e+01	-2.9999999999999e+01
Singular points :			
0.0000000000000	0.0000000000000	-4.7686776705208e-07	-2.1352336639535e-14

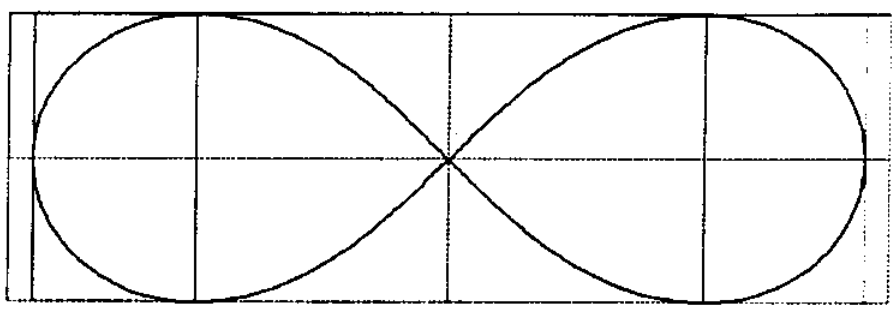


Figure 9: Double point - (cusp of the first kind, Isochrone)

$F(u,v) = v^2 - u^3 = 0$

Window of interest : [-1.0 1.0] x [-1.1 1.1]

Significant Points :

Correct (u, v)		Computed (u, v)	
Border points :			
1.0000000000000	1.0000000000000	1.0000000000000	1.0000000000000
1.0000000000000	-1.0000000000000	1.0000000000000	-1.0000000000000
Singular points :			
0.0000000000000	0.0000000000000	-6.1462656620917e-06	+2.9244303931972e-17

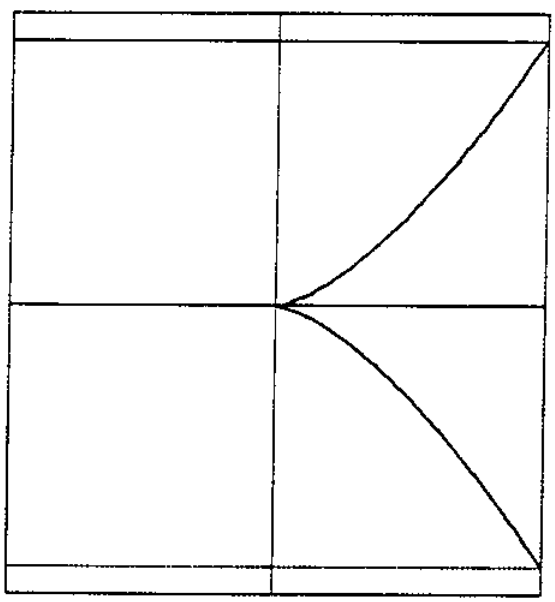


Figure 10: Double point - (cusp of the first kind, Cardioid)

$F(u,v) = u^4 - 4u^3 + 2u^2v^2 - 4uv^2 - 4v^2 + v^4 = 0$

Window of interest : [-0.5 4.0] x [-3.0 3.0]

Significant Points :

Correct (u, v)		Computed (u, v)	
Turning points :			
1.5000000000000	2.5980762113533	+1.5000000000000e+00	+2.5980762113533e+00
1.5000000000000	-2.5980762113533	+1.5000000000000e+00	-2.5980762113533e+00
4.0000000000000	0.0000000000000	+4.0000000000000e+00	+1.8911327401530e-15
-0.5000000000000	-0.8660254037844	-5.0000000000001e-01	-8.6602540378441e-01
-0.5000000000000	0.8660254037844	-5.0000000000002e-01	+8.6602540378441e-01
Singular points :			
0.0000000000000	0.0000000000000	-3.2315366105781e-05	-5.82184768181896e-15

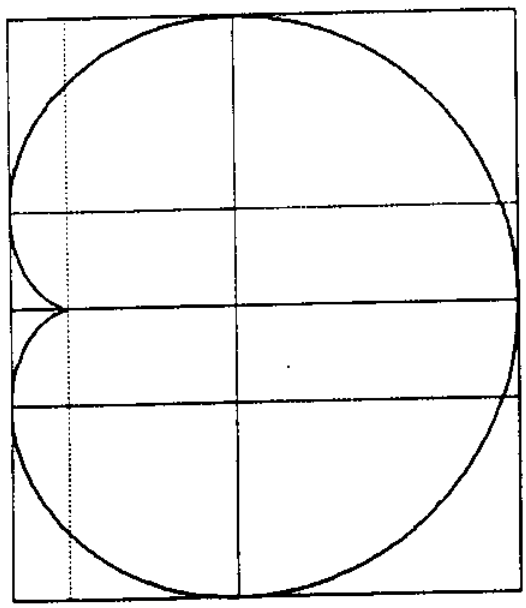


Figure 11: Double point - (cusp of the second kind, Ramphoid)

$F(u,v) = u^4 - 2u^2v + u^2v^2 - uv^2 + v^2 = 0$

Window of interest : [-2.0 2.0]x[-2.0 2.0]

Significant Points :

	Correct (u, v)	Computed (u, v)
Turning points :		
0.9232162147621	1.1616026426114	+9.2321621476213e-01 +1.1616026426115e+00
1.0000000000000	1.0000000000000	+1.0000000000000e+00+1.0000000000000e+00
Singular points :		
0.0000000000000	0.0000000000000	-5.1724452316170e-08 +1.0094194454210e-07

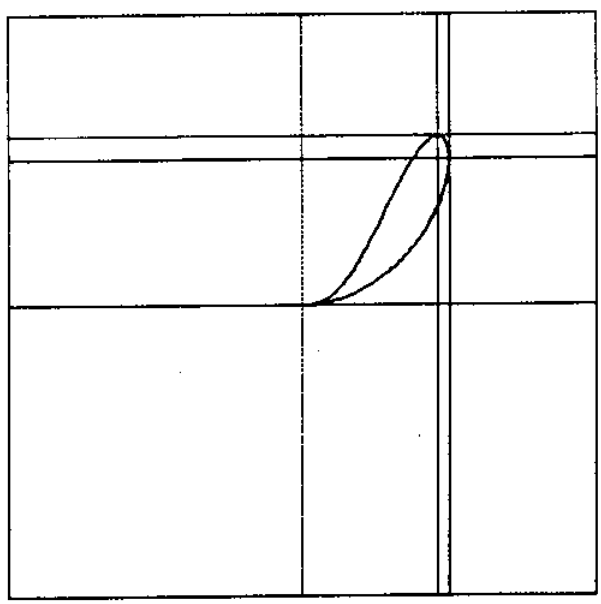


Figure 12: Double point - (Double cusp ie. tacnode, Hippopede)

$F(u,v) = u^4 - 4u^2 + 2u^2v^2 + v^4 = 0$

Window of interest : [-2.0 2.0] x [-2.0 2.0]

Significant Points :

Correct (u, v)		Computed (u, v)	
Turning points :			
1.0000000000000	1.0000000000000	+9.9999999999999e-01	+1.0000000000000e+00
-1.0000000000000	1.0000000000000	-9.9999999999999e-01	+1.0000000000000e+00
1.0000000000000	-1.0000000000000	+9.9999999999999e-01	-1.0000000000000e+00
-1.0000000000000	-1.0000000000000	-9.9999999999999e-01	-1.0000000000000e+00
2.0000000000000	0.0000000000000	+2.0000000000000e+00	+8.5626049309039e-16
-2.0000000000000	0.0000000000000	-2.0000000000000e+00	+8.1223569132370e-16
Singular points :			
0.0000000000000	0.0000000000000	-4.1290175986993e-09	-7.6852166547902e-07

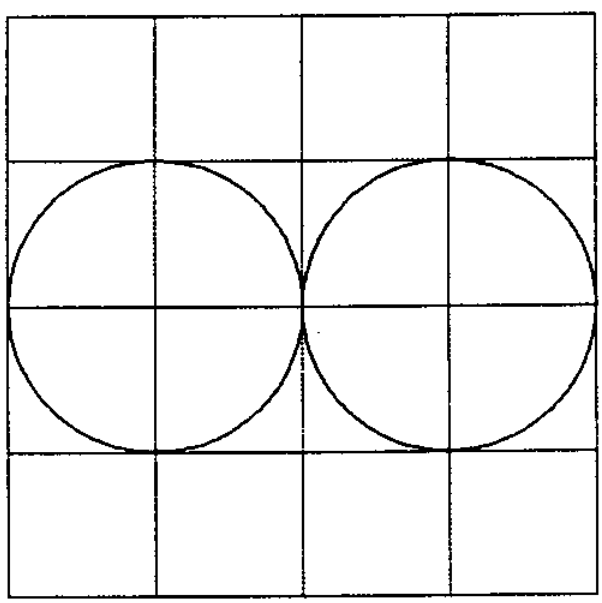


Figure 13: Double point - (tacnode and self-intersection) [43]

$F(u,v) = 2u^4 - 3u^2v + v^2 - 2v^3 + v^4 = 0$ in [-2.0 2.0] x [-1.0 3.0]

Significant Points :

Correct (u, v)		Computed (u, v)	
Turning points :			
-1.2431794435377	2.0606601717798	-1.2431794435377e+00	+2.0606601717798e+00
1.2431794435377	2.0606601717798	+1.2431794435377e+00	+2.0606601717798e+00
0.2365557162041	0.3002385440000	+2.3655571620402e-01	+3.0023854400023e-01
-0.2365557162041	0.3002385440000	-2.3655571620405e-01	+3.0023854400015e-01
1.4969203224061	1.7589358179272	+1.4969203224061e+00	+1.7589358179272e+00
-1.4969203224061	1.7589358179272	-1.4969203224061e+00	+1.7589358179272e+00
Singular points :			
0.0000000000000	0.0000000000000	+2.5153919877490e-08	-1.4254246786341e-07
0.0000000000000	1.0000000000000	-1.4101652684880e-06	+9.9999999999999e-01

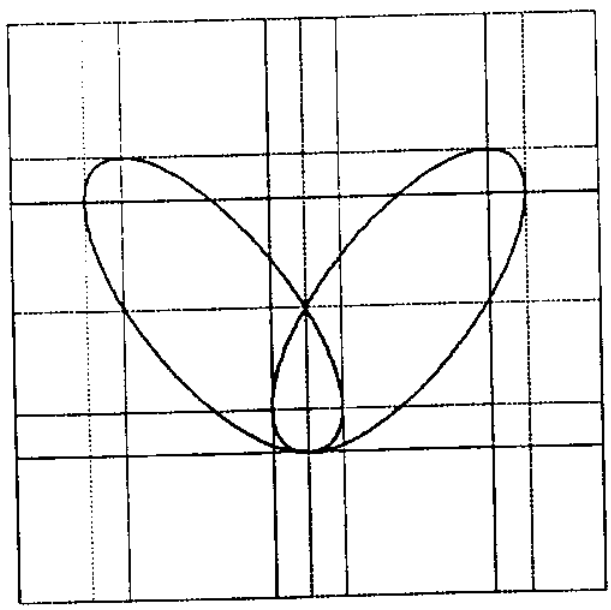


Figure 14: Double point - (Geisow's multiple self-intersection case)

$F(u,v) = -6u^4 + 21u^3 - 19u^2 - 6u^2v^2 + 11uv^2 + 3v^2 - 4v^4 = 0$ in [-0.5 2.0] x [-1.5 1.5]

Significant Points :

Correct (u, v)		Computed (u, v)	
Turning points :			
0.5000000000000	-1.1180339887498	+5.0000000000000e-01	-1.1180339887498e+00
0.5000000000000	1.1180339887498	+4.9999999999999e-01	+1.1180339887498e+00
1.2068546093436	1.0324817331794	+1.2068546093437e+00	+1.0324817331794e+00
1.2068546093436	-1.0324817331794	+1.2068546093436e+00	-1.0324817331794e+00
1.4431453906563	0.8186766581811	+1.4431453906563e+00	+8.1867665818117e-01
1.4431453906563	-0.8186766581811	+1.4431453906563e+00	-8.1867665818116e-01
1.5000000000000	-0.8660254037844	+1.5000000000000e+00	-8.6602540378444e-01
1.5000000000000	0.8660254037844	+1.4999999999999e+00	+8.6602540378444e-01
-0.1000000000000	-0.4795831523312	-9.9999999999998e-02	-4.7958315233127e-01
-0.1000000000000	0.4795831523312	-9.9999999999996e-02	+4.7958315233127e-01
Singular points :			
0.0000000000000	0.0000000000000	+1.5701775202177e-15	-7.3217495753655e-08
1.0000000000000	1.0000000000000	+1.0000000090928e+00	+1.0000000181857e+00
1.0000000000000	-1.0000000000000	+9.9999999764761e-01	-9.9999999529522e-01

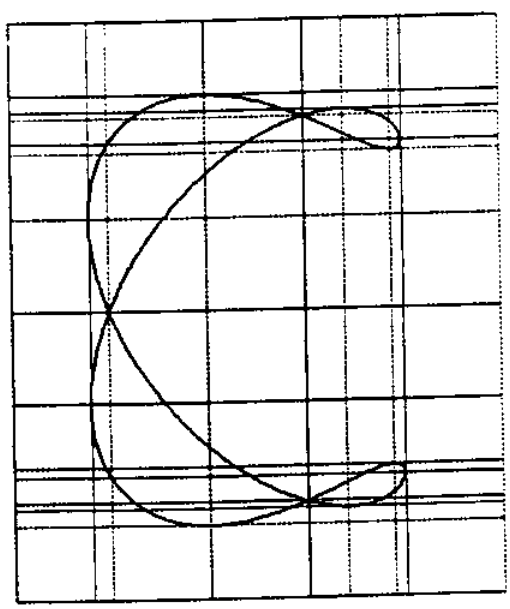


Figure 15: Triple point

$F(u,v) = u^4 + 3u^2v + 2u^2v^2 - v^3 + v^4 = 0$

Window of interest : [-2.0 2.0] x [-2.0 2.0]

Significant Points :

Correct (u, v)		Computed (u, v)	
Turning points :			
0.0000000000000	1.0000000000000	+1.2516102400412e-15	+1.0000000000000e+00
-0.7261843774138	-0.5625000000000	-7.2618437741389e-01	-5.6250000000000e-01
0.7261843774138	-0.5625000000000	+7.2618437741390e-01	-5.6250000000002e-01
-0.8800862965230	-0.4448027481129	-8.8008629652305e-01	-4.4480274811294e-01
0.8800862965230	-0.4448027481129	+8.8008629652307e-01	-4.4480274811294e-01
-0.1845043649140	0.6323027481129	-1.8450436491412e-01	+6.3230274811290e-01
0.1845043649140	0.6323027481129	+1.8450436491412e-01	+6.3230274811290e-01
Singular points :			
0.0000000000000	0.0000000000000	-6.8905407486226e-11	-2.7261333783872e-05

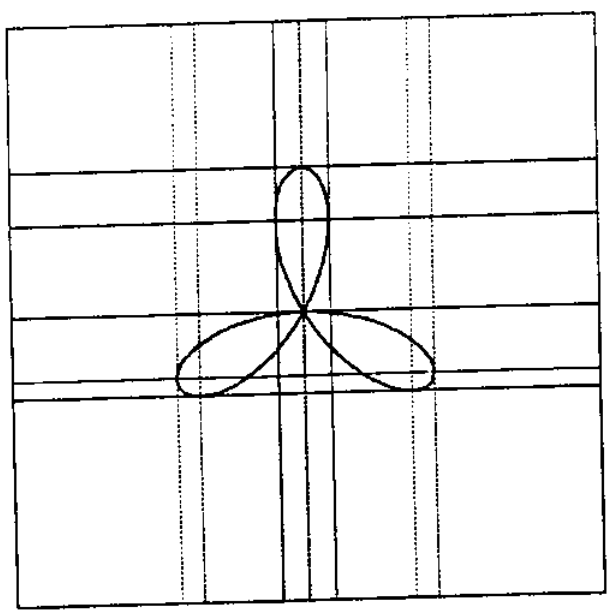


Figure 16: Quadruple point

$F(u,v) = u^6 + 3u^4v^2 - 4u^2v^2 + 3u^2v^4 + v^6 = 0$

Window of interest : [-1.0 1.0] x [-1.0 1.0]

Significant Points :

Correct (u, v)		Computed (u, v)	
Turning points :			
-0.5443310539518	-0.7698003589195	-5.4433105395181e-01	-7.69800358919501e-01
-0.5443310539518	0.7698003589195	-5.4433105395181e-01	+7.69800358919507e-01
0.5443310539518	-0.7698003589195	+5.4433105395181e-01	-7.69800358919504e-01
0.5443310539518	0.7698003589195	+5.4433105395181e-01	+7.69800358919511e-01
-0.7698003589195	-0.5443310539518	-7.6980035891950e-01	-5.44331053951817e-01
-0.7698003589195	0.5443310539518	-7.6980035891950e-01	+5.44331053951818e-01
0.7698003589195	-0.5443310539518	+7.6980035891950e-01	-5.44331053951815e-01
0.7698003589195	0.5443310539518	+7.6980035891951e-01	+5.44331053951817e-01
Singular points :			
0.0000000000000	0.0000000000000	-4.0828282815007e-11	-3.68081202784540e-03

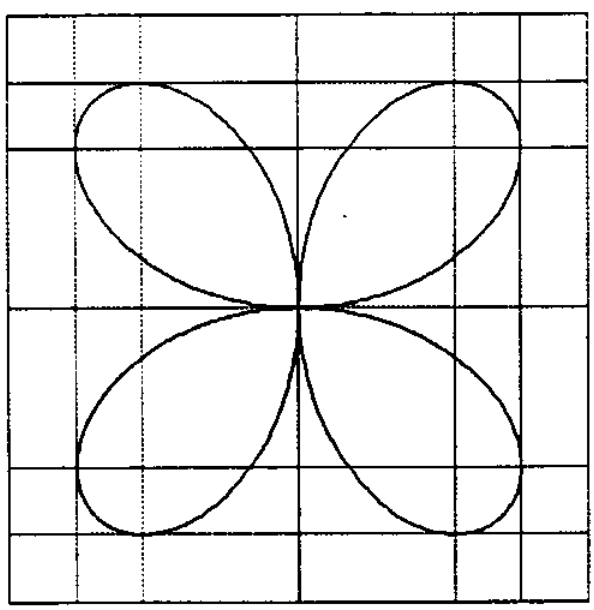


Figure 17: Reducible curve - (crunode)

$F(u,v) = (u - v)(u^2 + v^2 - 1) = 0$

Window of interest : [-1.0 1.0] x [-1.0 1.0]

Significant Points :

Correct (u, v)		Computed (u, v)	
Border points :			
1.0000000000000	1.0000000000000	1.0000000000000	1.0000000000000
-1.0000000000000	-1.0000000000000	-1.0000000000000	-1.0000000000000
0.0000000000000	1.0000000000000	0.0000000000000	1.0000000000000
0.0000000000000	-1.0000000000000	0.0000000000000	-1.0000000000000
1.0000000000000	0.0000000000000	1.0000000000000	0.0000000000000
-1.0000000000000	0.0000000000000	-1.0000000000000	0.0000000000000
Turning points :			
0.0000000000000	1.0000000000000	+8.4594300673926e-16	+1.0000000000000e+00
-1.0000000000000	0.0000000000000	-1.0000000000000e+00	-1.1201587531755e-15
0.0000000000000	-1.0000000000000	+4.6770969741493e-16	-1.0000000000000e+00
1.0000000000000	0.0000000000000	+1.0000000000000e+00	-5.0449157374093e-16
Singular points :			
0.7071067811865	0.7071067811865	+7.0710486887662e-01	+7.0710678118655e-01
-0.7071067811865	-0.7071067811865	-7.0710678118655e-01	-7.0710486920805e-01

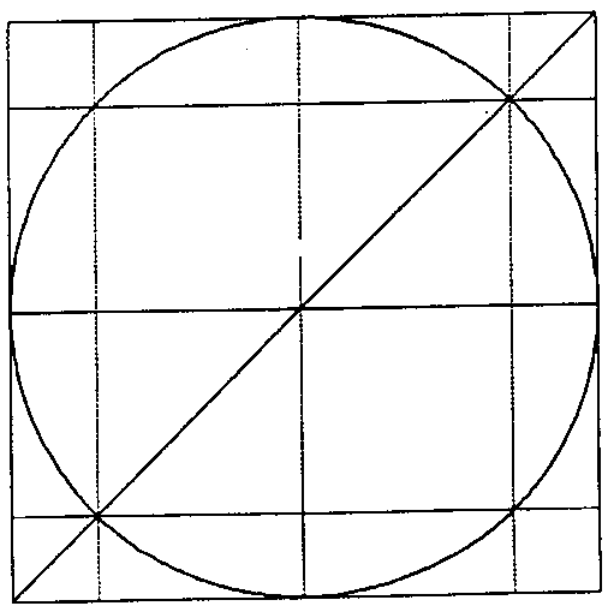


Figure 18: Reducible curve with parametric line component

$F(u, v) = (v - 0.42644777732)(v - 0.9u^2) = 0$

Window of interest : [0.0 1.0] x [0.0 1.0]

Significant Points :

Computed (u,v)
Border points :

0.0000000000000	0.0000000000000
0.0000000000000	0.4264424777732
1.0000000000000	0.4264424777732
1.0000000000000	0.8999999999999

Singular points :

0.6883494572552	0.4264424777732

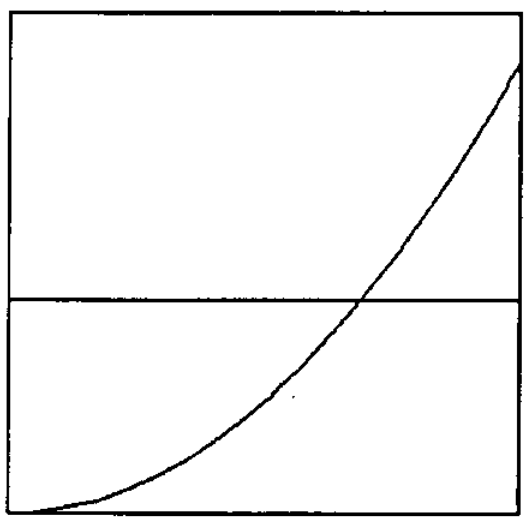


Figure 19: Double point - Cusps in a Bicorn [20]

$F(u,v) = u^4 - 128u^2 + u^2v^2 + 32u^2v - 2048v + 192v^2 + 4096 = 0$

Window of interest : [-8.0 8.0] x [0.0 8.0]

Significant Points :

Correct (u,v)

Border points :

-8.0000000000000 0.000000000000000
8.0000000000000 0.000000000000000
0.0000000000000 8.000000000000000

Turning points :

0.0000000000000 8.000000000000000
0.0000000000000 2.666666666666666

Singular points :

-8.0000000000000 0.000000000000000
8.0000000000000 0.000000000000000

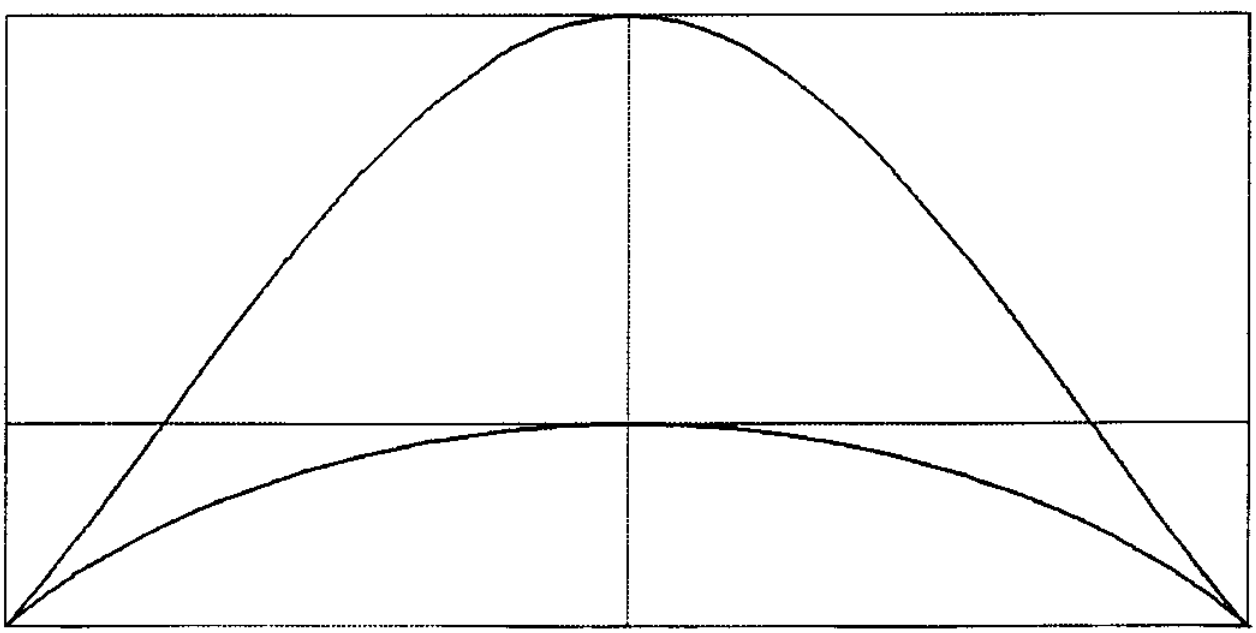


Figure 20: Constriction resolution, constriction size = 7.1×10^{-3}

$F(u,v) = [u^2 + (v - 1)^2 - 0.5][(u - 1)^2 + v^2 - 0.49] = 0$

Window of interest : [0.0 1.0] x [0.0 1.0]

Significant Points :

Correct (u, v)		Computed (u, v)
Border and Turning points :		
0.7071067811865	1.0000000000000	+0.7071067811865e+00+1.0000000000000e+00
0.0000000000000	0.2928932188134	+0.0000000000000e+00+0.2928932188134e+00
0.3000000000000	0.0000000000000	+0.3000000000000e+00+0.0000000000000e+00
1.0000000000000	0.7000000000000	+1.0000000000000e+00+0.6999999999999e+00

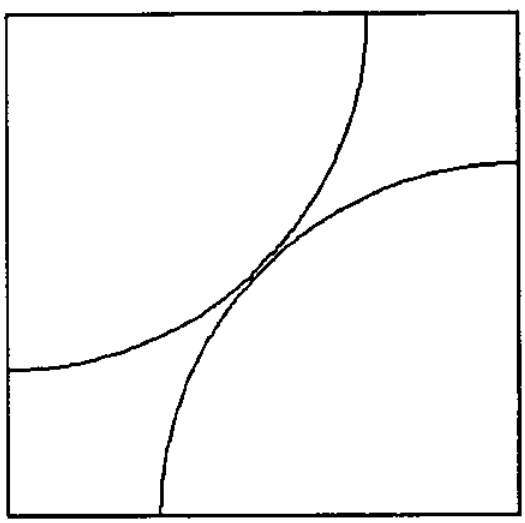


Figure 21:
A biquadratic Bezier patch and a plane leading to an algebraic curve with four turning points

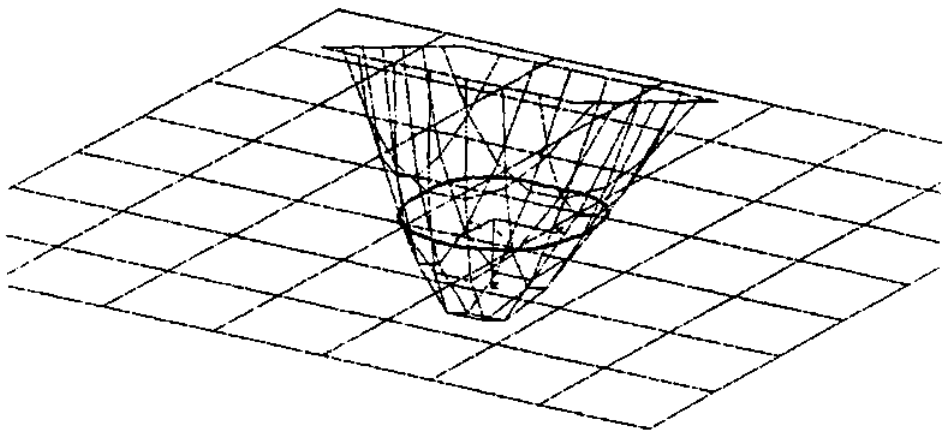


Figure 22:
A biquadratic Bezier patch with a parametric line - example where factorization is useful

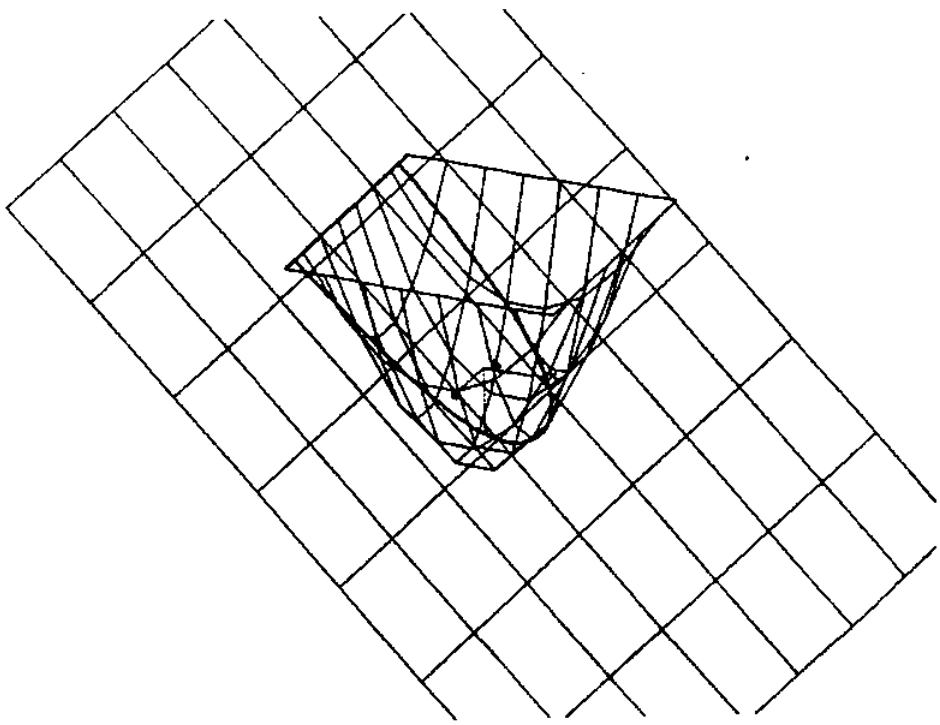


Figure 23:
A torus represented as a rational biquadratic B-spline patch intersected with a plane tangent at an inner saddle point of the torus

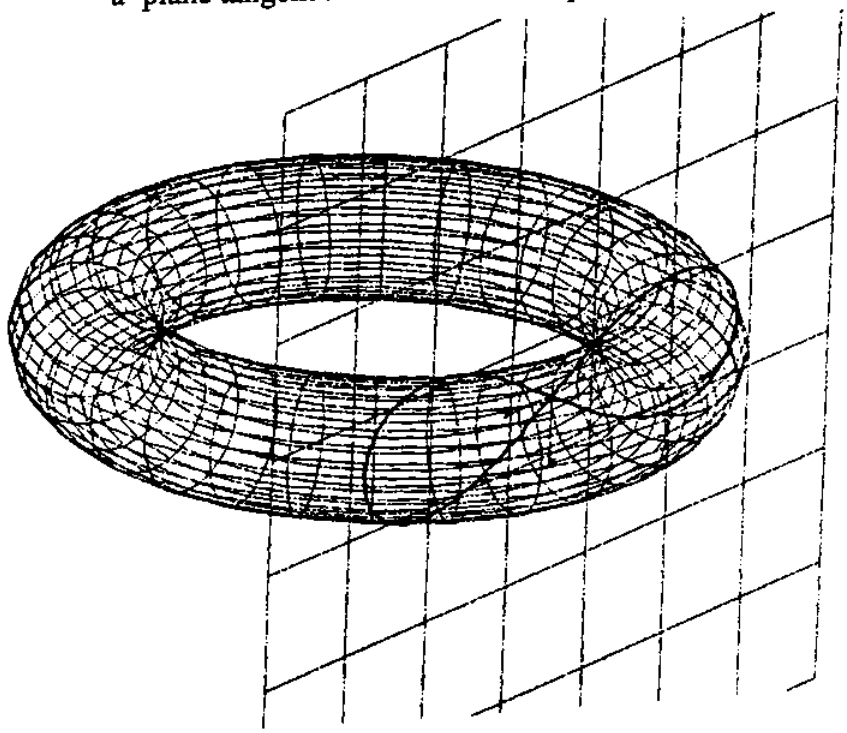


Figure 24:
A torus represented as above with a nearly tangent plane leading to a small isolated loop

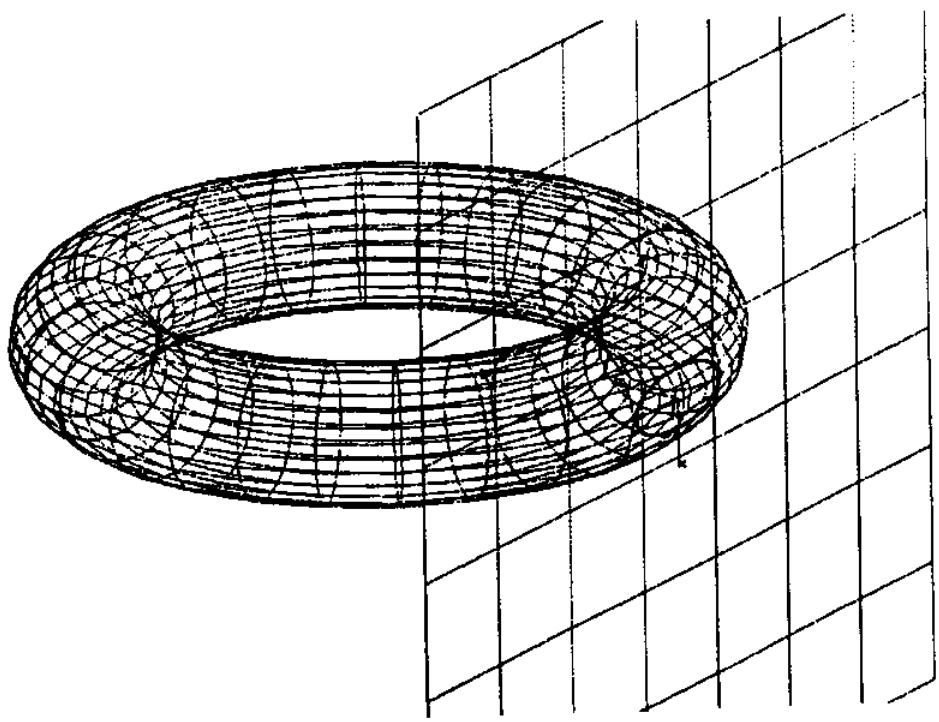


Figure 25:
A torus represented as above with a plane tangent to it at two points on either side of the equatorial plane

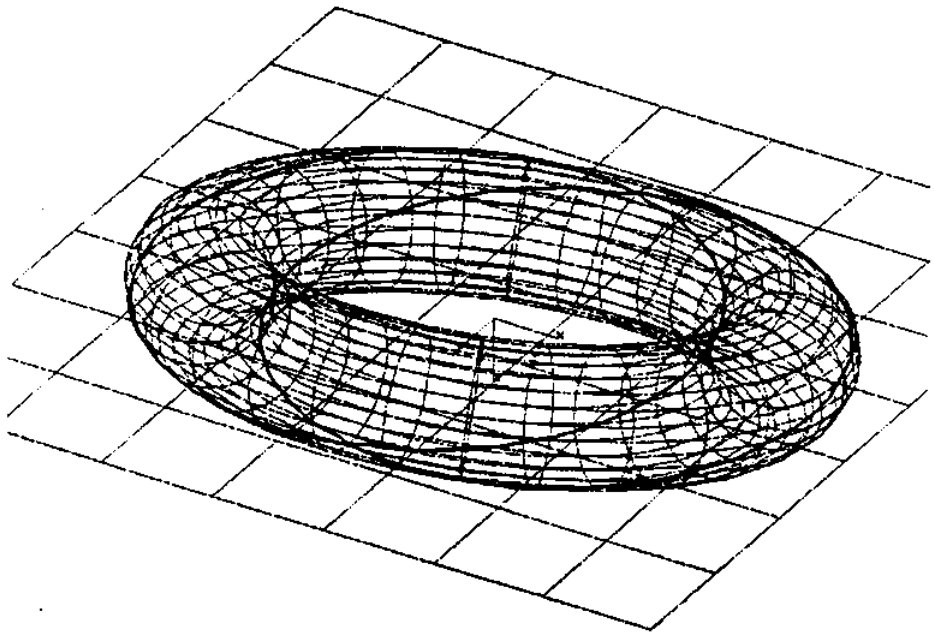


Figure 26:
A bicubic patch with a plane leading to an algebraic curve with one loop

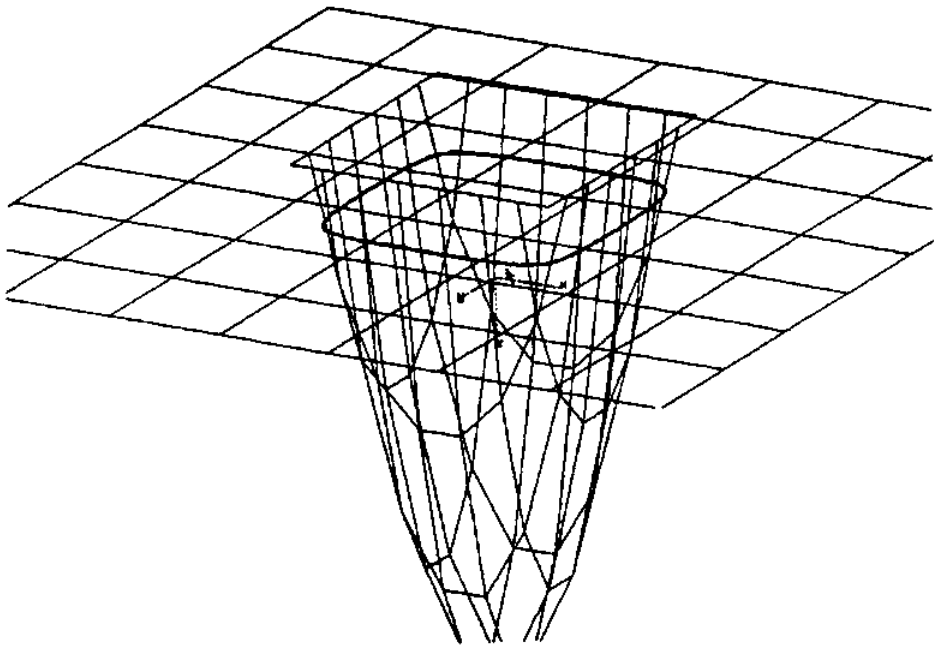


Figure 27:
A bicubic patch with a plane - algebraic curve with one self-intersection

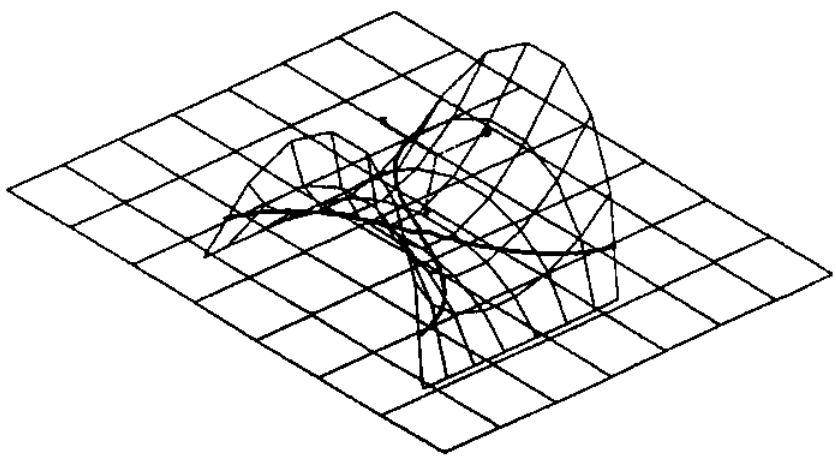


Figure 28:
Torus intersecting with a cylinder of radius equal to smaller torus radius - curve with two double points

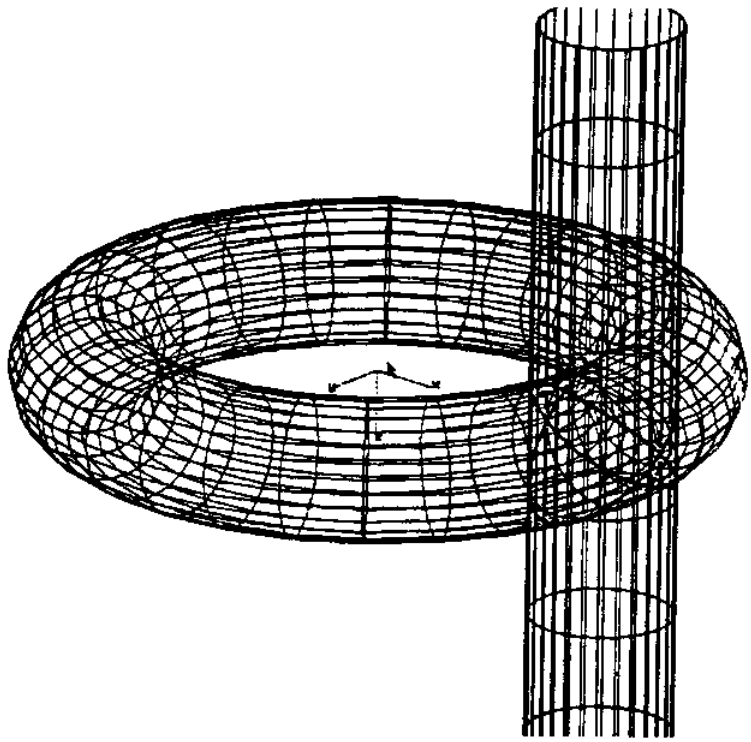


Figure 29:
The radius of the cylinder diminished by 0.1% in above example to illustrate the sudden change in the topology of the intersection curve - algebraic curve with two loops.

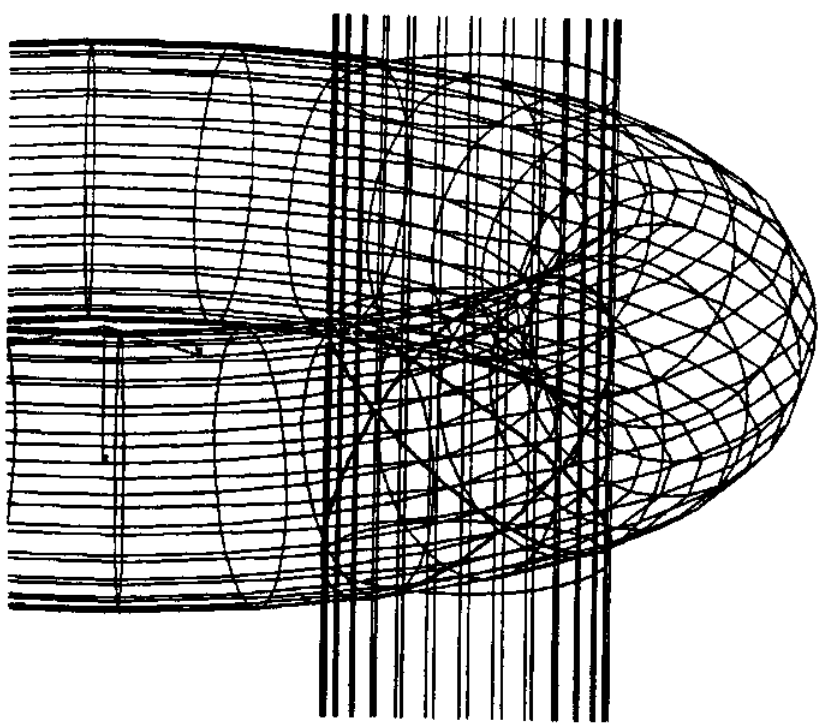


Figure 30:
The radius of the cylinder increased by 0.1% in above example to illustrate the sudden change in the topology of the intersection curve - algebraic curve with two loops.

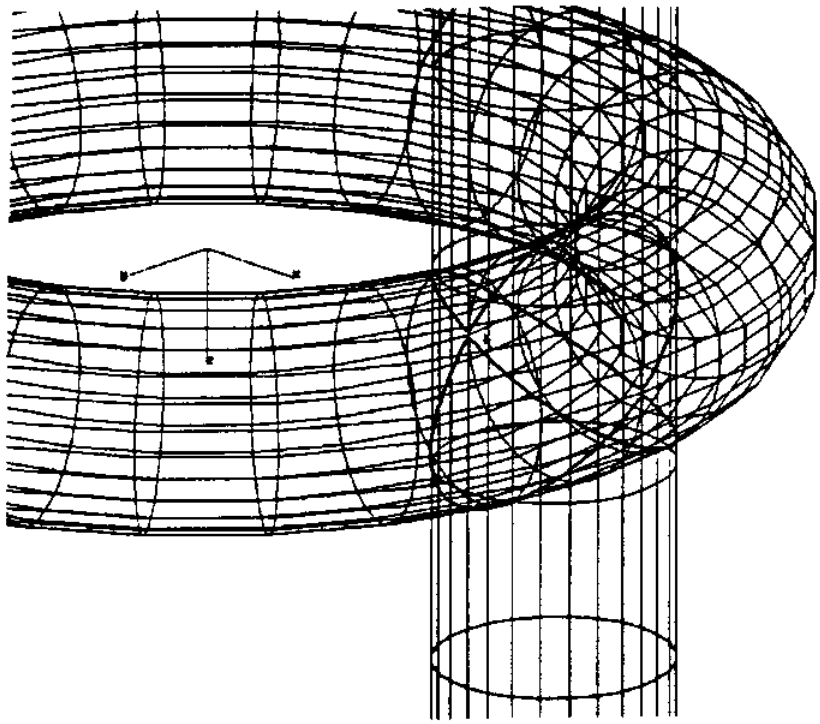


Figure 31: A torus with a cylinder of radius smaller than the smaller of the radii of the torus producing four independent loops

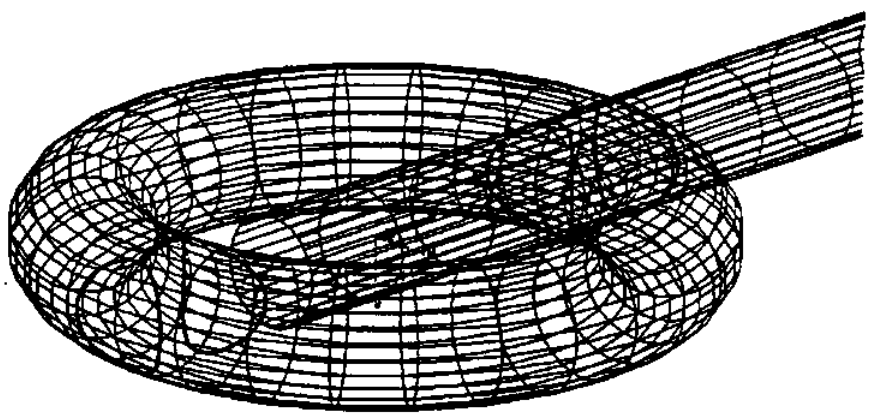


Figure 32:
Torus with a sphere with inner tangency - produces a double point on the algebraic curve

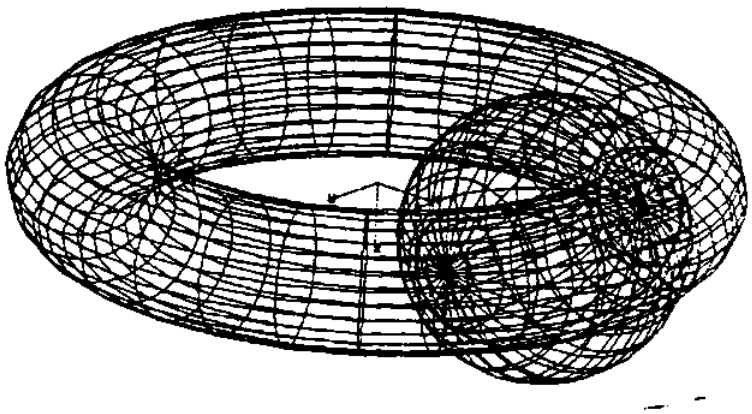


Figure 33: Cylinder $x^2 + (z-1)^2 - 1 = 0$ with elliptic paraboloid $x = u$, $y = v$, $z = 0.5u^2 + 0.25v^2$

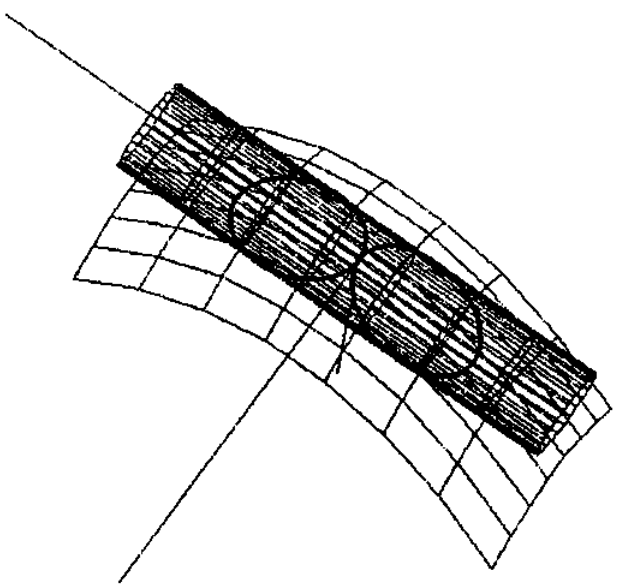


Figure 34: Sphere - cone intersection producing two loops

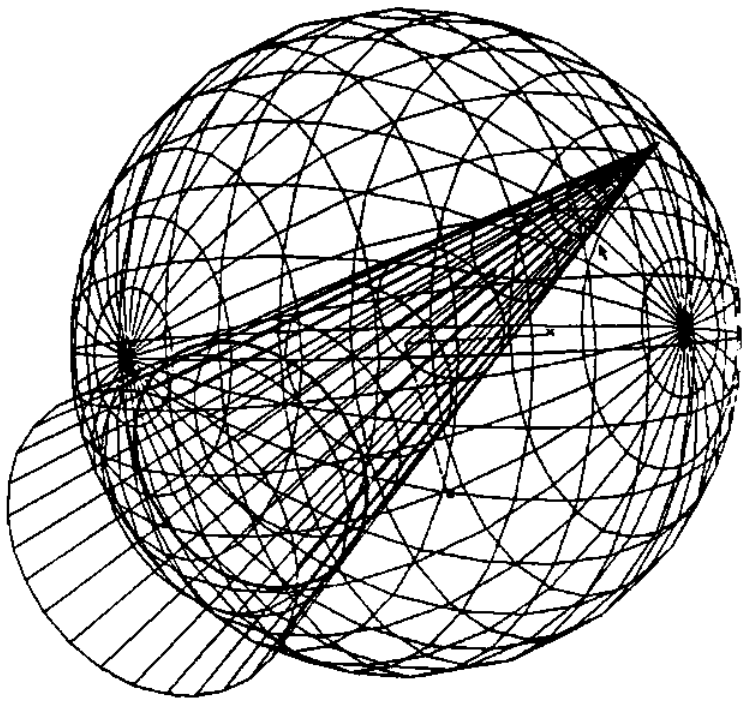


Figure 35: Sphere - cylinder intersection producing a double point when sphere is positioned just touching the inner surface of the cylinder

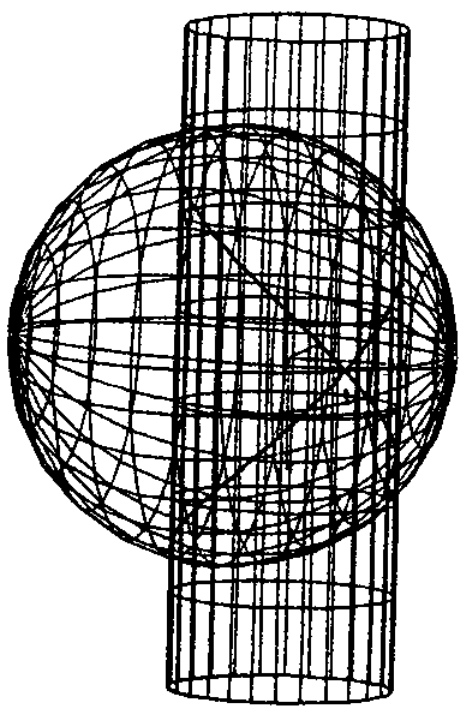


Figure 36: Cone - cylinder intersection leading to two double points

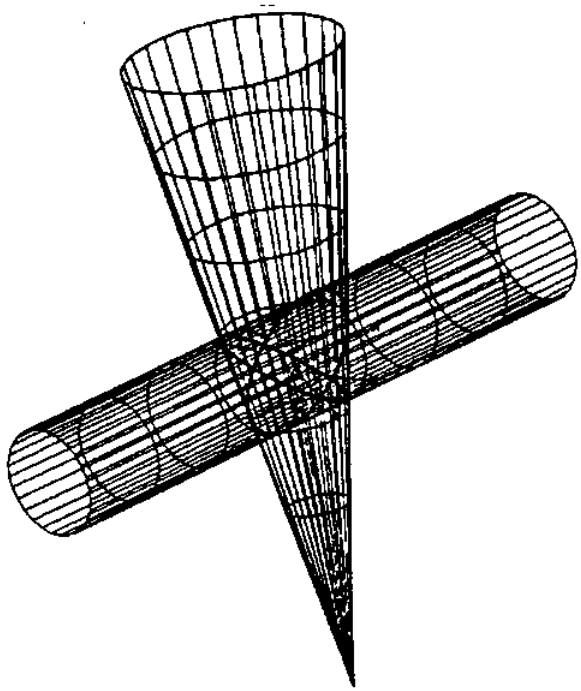


Figure 37: Cylinder - cylinder intersection with both cylinders of same radii and positioned to produce two double points

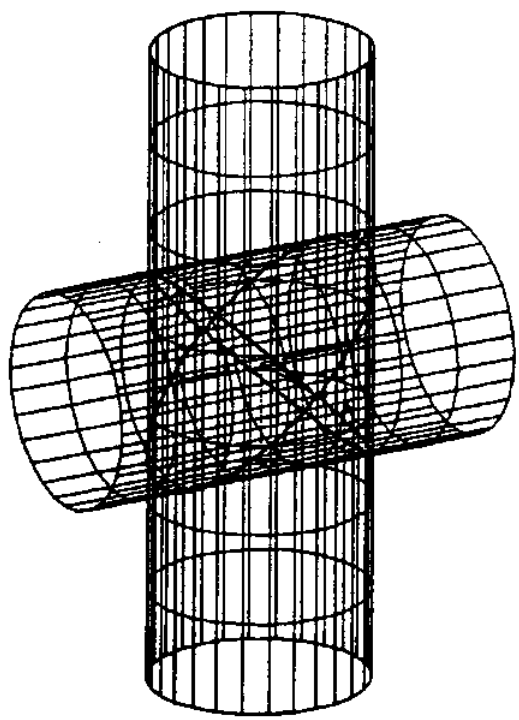


Figure 38: Bicubic patch with a cylinder leading to two loops

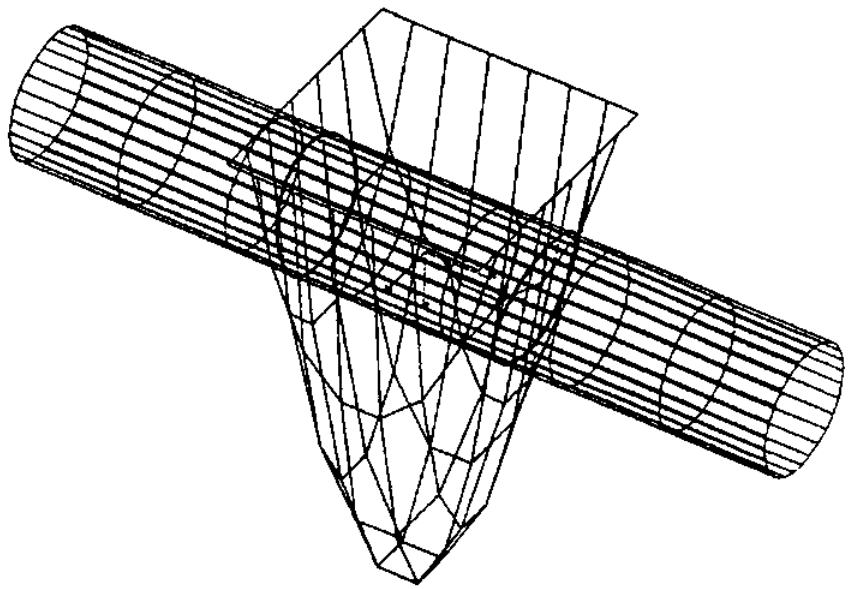


Figure 39: Bicubic patch with a cone leading to two loops

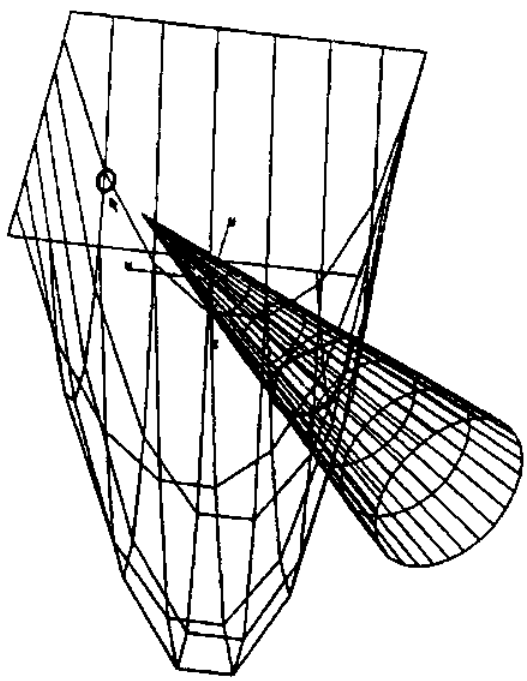


Figure 40: Bicubic patch with a sphere positioned in its concave side it to produce four independent loops

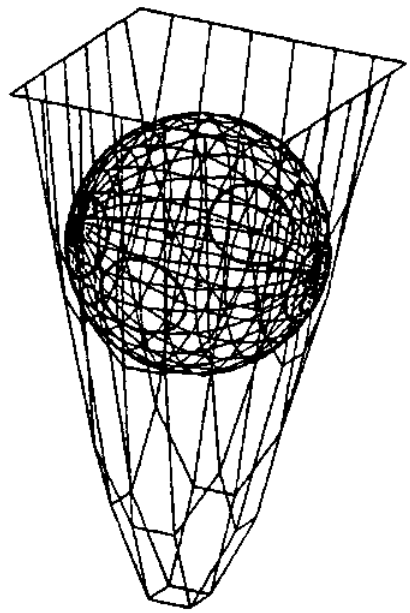
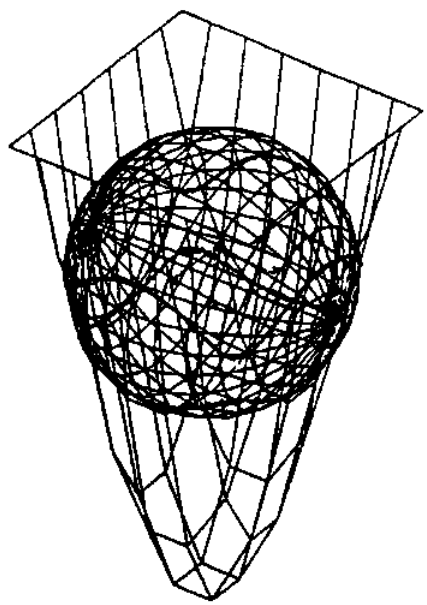


Figure 41: Bicubic patch with a sphere of slightly larger radius positioned at the same point as in the previous example



prevailing is that of obtaining one point on every segment of the curve. Lattice evaluation methods may miss small features of the curve because of the need to choose a grid size a priori. Published methods from this class also do not have an entirely satisfactory means of resolving situations involving singularities in the curve because of the discrete nature of the solution scheme. Subdivision methods have also been very useful in solving intersection problems with the possible exception of cases involving singular points and small loops.

The computation of the analytic representation of the curve provides the capability to transform the problem at hand to the intersection of an auxiliary integral Bezier surface patch and an auxiliary plane and is a key feature of the method. This transformation allows the use of subdivision methods in the solution of algebraic surface and rational polynomial patch intersection problems and provides an effective method of obtaining points on every segment of the original curve. This procedure of finding the representation of the curve has not been adopted in the literature for any intersection case other than with the plane as the algebraic surface because of possible loss of accuracy in the coefficients of the algebraic curve. In this work, we investigated symbolic substitution and expansion to provide explicit expressions for each coefficient of the algebraic curve (in the Bernstein basis) in terms of primary data. Since the primary data and the algebraic curve are both expressed in the Bernstein basis, the resulting expressions for the coefficients turn out to be compact and, therefore, possible to compute with enhanced accuracy. The availability of explicit expressions for each of the coefficients also allows the use of sophisticated addition schemes for large scale series.

The computation and use of border, turning and singular points of the curve allows subdivision and faceting techniques to provide approximations of the curve with correct connectivity. It was found that border points could be computed reliably using subdivision and the variation diminishing property of the Bernstein basis. Two methods of identifying turning and singular points were attempted. The first method involved the solution of univariate polynomials obtained by elimination techniques of algebraic geometry and the coefficients of which are obtained by expansion of determinants. It has been observed that the accuracy and efficiency obtainable by solution of such equations may be below the requirements of geometric modeling applications and, hence, this method is limited to very low degree cases. The second method involved the use of a relatively coarse linear approximation of the curve obtained by subdivision and faceting of the control polyhedron to generate initial approximations for turning and singular

points to be used with minimization and Newton techniques for solving systems of non-linear equations. The Bernstein representation is used to verify the computation of all such points and in case of failure, to provide alternate means of computing such points by subdivision. These methods show promise as they do not involve expansion of determinants and the solution of high degree polynomials and as a result provide improved accuracy on these significant points. This method was used in the complete solution of the general intersection problem stated above. It must also be noted that this method of computing turning and singular points is not as seriously restricted by degree as the elimination method of algebraic geometry since only evaluations of the low degree representation are required for the search solution procedures. In this work, curves with only a finite number of singular points (points of tangency) have been considered.

The tracing of the curve is accomplished by first splitting the Bezier patch describing the intersection curve into a matrix of smaller rectangular patches such that each of the subpatches describes portions of the intersection curve that are simpler to trace using subdivision methods. This splitting is an important addition to existing subdivision and faceting algorithms and provides subdivision-based intersection solutions with the potential for extracting the true connectivity of the curve. This tracing procedure is relatively independent of problems of high degree and performs well in test examples from the literature. The method of triangulation of the control polyhedron is important in the presence of closely spaced features. A scheme of using subdivision as a means to avoid ambiguous and, potentially, incorrect connectivity in the presence of closely spaced features in the curve is developed. Distance based measures of evaluating the algebraic curve approximation for accuracy both in the planar parametric space and in three-dimensional space are developed and used. Linear segments generated by the triangular facets are connected into complete segments by the use of control point indices to provide direct unambiguous matching of adjacent linear segments. Thus the tracing scheme provides connected segments between significant points of the curve which may then be further connected together based on the application. A variety of complex algebraic curves were traced successfully using the above method.

A consistent effort has been made to carry out all evaluations and solutions for the roots of polynomials using the Bernstein basis without any intermediate conversions to the monomial basis to exploit the relative numerical stability of the Bernstein basis. The numerical reliability of the methods developed depends on the accuracy with which significant points are computed.

Border and turning points are computed with very good accuracy while the singular points are computed to lesser accuracy that varies with the type and order of singularity. These inaccuracies in the computation caused by imprecise floating point arithmetic should be reflected in the way the small constant parameters of the method are chosen and procedures to choose such constants in a dynamic manner ought be explored. Because of the inherent ill-conditioned nature of singularities, it may be required to compute singular points in extended precision. Both the computation of significant points using the numerical solution methods and the tracing of the intersection curve are inherently parallel processes. This can be seen at three stages of the intersection process - at the level of splitting the rational B-spline patch into its rational polynomial components, the splitting of each polynomial patch into smaller subpatches that include a simple portion of the curve within their domain, in the subdivision of the control polyhedron and the intersection of each facet of the control polyhedron of the subpatch with the control plane. The parallel nature of the method could become important in future large-scale real-time applications.

The methods used in this work depend to a large extent on the availability of an explicit polynomial representation of the intersection curve. Such an explicit representation, although theoretically available for all intersections between polynomial surfaces, is impractical to obtain in some cases. For example, for intersections between rational biquadratic and bicubic patches, such a representation is very difficult to compute and alternate methods of obtaining the intersection should be developed.

An alternate method would be to study modeling sculptured shape with surface representations that would lead to intersection curves of lesser degree compared to the intersections of the widely used rational biquadratics and bicubics. Modeling with low order algebraic patches within tetrahedra and rectangular boxes studied by [37, 27] provide such examples.

I. Factoring Parametric Line Components from an Algebraic Curve

When there are parametric line components along with other general segments containing both u and v turning points in an algebraic curve, the methods presented in Section 4 and Section 5 encounter difficulties. In particular, let us address the case of an algebraic curve which contains the parametric line u = a as one component ie. the algebraic curve F(u,v) = 0 can be written as

$$F(u,v) = (u - a)\, E(u,v) \tag{36}$$

Such a parametric line represents an infinity of u-turning points and such points satisfy $F_v = (u\text{-}a)$ $E_v(u,v) = 0$ at all points on the parametric line.

In the case of the direct elimination method followed in Section 4, the univariate polynomial equation in v that would be solved for the u-turning points must be identically valid for all v since for *every* value of v there is at least one corresponding value of u which together form a u-turning point namely that on the parametric line. This implies that all coefficients of the univariate polynomial must identically vanish in the presence of such parametric lines. Hence an actual u-turning point of the algebraic curve component E(u,v) cannot be identified with the above method in the presence of such parametric lines. In the case of the method described in Section 5, a large number of segments approximating the parametric line provide starting points to the infinity of turning points that are on a parametric line. These starting points lead to an unnecessary amount of redundant computation of turning points which are more easily computed from the border points. It is thus useful to factor such parametric lines from the algebraic curve in both cases before attempting to obtain the finite set of turning points.

The computation of border points as explained in Section 4 allows us identify these lines and factor them out as follows : Compute border points according to the method of Section 4 and find all pairs of border points that have one common coordinate and a vanishing parametric derivative corresponding to the other coordinate (ie. the coordinate that is distinct in the pair). For each such pair, split the patch corresponding to F(u,v) at the u or v point corresponding to the common coordinate, say u = a, and obtain the control points corresponding to the two sub-patches. The row of control points corresponding to u = a in both the sub-patches must have vanishing weights w_{ij} in order to signify a parametric line at u = a. If this condition is true, these two sub-patches are then used to factor out the parametric line and to determine the turning points in each sub-domain separately.

We now illustrate the above procedure for the case of factoring a parametric line u = a $\in [\alpha,\beta]$. We are interested in finding the descriptions of the Bezier sub-patches E_1 and E_2 obtained after factoring out the parametric line from the sub-patches F_1 and F_2 on either side of u = a ie.

$$\begin{aligned} F_1(u,v) &= (u-a)\,E_1(u,v) \quad u \in [\alpha,a] \text{ and } v \in [\gamma,\delta] \\ F_2(u,v) &= (u-a)\,E_2(u,v) \quad u \in [a,\beta] \text{ and } v \in [\gamma,\delta] \end{aligned} \tag{37}$$

The factorization for sub-patch F_1 can be performed as follows

$$\begin{aligned} F_1(u,v) &= \sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij} \frac{\binom{m}{i}(u-\alpha)^i(a-u)^{m-i}}{(a-\alpha)^m} B_{j,n}(v) \\ &= \sum_{i=0}^{m-1}\sum_{j=0}^{n} w_{ij} \frac{\frac{m}{m-i}\binom{m-1}{i}(u-\alpha)^i(a-u)^{m-1-i}(a-u)}{(a-\alpha)(a-\alpha)^{m-1}} B_{j,n}(v) \\ &= (a-u)\sum_{i=0}^{m-1}\sum_{j=0}^{n} w'_{ij}\, B_{i,m-1}(u)\, B_{j,n}(v) \\ &= (a-u)\,E_1(u,v) \end{aligned}$$

and where

$$w'_{ij} = w_{ij}\frac{m}{(m-i)(a-\alpha)} \qquad i = 0,1,\ldots m-1,\ j = 0,1,\ldots n \tag{38}$$

The new coefficients w_{ij}' along with equally spaced u and v values in the interval $[\alpha,a]x[\gamma,\delta]$ form the coordinates of the control points for the sub-patch E_1. A similar factorization when the parametric line is on the left as in the case of the surface $F_2(u,v)$ gives the factorized sub-patch $E_2(u,v)$ as follows where $u \in [a, \beta]$

$$\begin{aligned} F_2(u,v) &= \sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij} \frac{\binom{m}{i}(u-a)^i(\beta-u)^{m-i}}{(\beta-a)^m} B_{j,n}(v) \\ &= \sum_{k=0}^{m-1}\sum_{j=0}^{n} w_{k+1\,j} \frac{\binom{m}{k+1}(u-a)^k(u-a)(\beta-u)^{m-k-1}}{(\beta-a)(\beta-a)^{m-1}} B_{j,n}(v) \\ &= (u-a)\sum_{k=0}^{m-1}\sum_{j=0}^{n} w'_{kj}\, B_{k,m-1}(u)\, B_{j,n}(v) = (u-a)\,H_2(u,v) \end{aligned}$$

$$\text{and where } w'_{kj} = w_{k+1\,j}\frac{m}{(k+1)(\beta-a)} \qquad k = 0,1,\ldots m-1,\ j = 0,1,\ldots n \tag{39}$$

The new $E_1(u,v)$ and $E_2(u,v)$ can now be used to generate approximations to the curve and the

required starting points for direct solution methods used in the computation of turning points. The same procedure can also be used to factor out v = constant parametric lines. The general case of multiple parametric lines in both directions can be tackled in a similar manner by splitting along all such parametric lines and factoring out the parametric lines in each small domain separately. Finally, if these factors exist with higher multiplicities, the same procedure can be used after determining the multiplicity by observing the number of rows of w_{ij} that vanish parallel and adjacent to the relevant border.

References

[1] Abhyankar, S. S., Bajaj, C.
Automatic Rational Parametrization of Curves and Surfaces I : Conics and Conicoids.
Report No: CSD-TR-583, Computer Sciences Department, Purdue University , March, 1986.

[2] Abhyankar, S. S., Bajaj, C.
Automatic Parametrization of Rational Curves and Surfaces III: Algebraic Plane Curves.
Report No. CSD-TR-619, Department of Computer Sciences, Purdue University , August, 1986.

[3] Arnon, D. S.
Topologically Reliable Display of Algebraic Curves.
ACM Computer Graphics 17(3):219-227, July, 1983.

[4] Bajaj, C., Hoffmann, C. M., Hopcroft, J.
Tracing Planar Algebraic Curves.
Report No. CSD-TR-637, Department of Computer Sciences, Purdue University , September, 1987.

[5] Barnhill, R. E., Farin, G., Jordan, M., Piper, B. R.
Surface/Surface Intersection.
Computer Aided Geometric Design 4:3-16, 1987.

[6] Boehm W.
Inserting New Knots Into B-Spline Curves.
Computer Aided Design (12):99-102, 1980.

[7] De Montaudouin, Y., Tiller, W., Vold, H.
Applications of Power Series in Computational Geometry.
Computer Aided Design 18(10):514-524, 1986.

[8] Farouki, R. T.
The Characterization of Parametric Surface Sections.
Computer Vision, Graphics and Image Processing 33:209-236, 1986.

[9] Farouki, R. T.
Trimmed Surface Algorithms for the Evaluation and Interrogation of Solid Boundary Representations.
IBM Journal of Research and Development 31(3):314-334, May, 1987.

[10] Farouki, R. T., Rajan, V. T.
On the Numerical Condition of Polynomials in Bernstein Form.
Computer Aided Geometric Design (4):191-216, March, 1987.

[11] Farouki, R. T., Rajan, V. T.
Algorithms for Polynomials in Bernstein Form.
IBM Research ReportOB RC 12873 , 1987.

[12] Geisow, A.
Surface Interrogations.
PhD thesis, School of Computing Studies and Accountancy, University of East Anglia, Norwich NR47TJ, U. K., July, 1983.

[13] Gill, P. E., Murray, W.
Newton-Type Methods for Unconstrained and Linearly Constrained Optimization.
Mathematical Programming 7:311-350, 1974.

[14] Goldman, R. N., Sederberg, T. W., Anderson, D. C.
Vector Elimination : A Technique for the Implicitization, Inversion, and Intersection of Planar Parametric Rational Polynomial Curves.
Computer Aided Geometric Design 1(4):327-356, 1984.

[15] *IRIS User's Guide, Volumes I and II, Programming Guide, Version 4.0*
1987.

[16] Jenkins, M. A.
Algorithm 493.
Transactions on Mathematical Software 1:178, 1975.

[17] Kahan, W.
A Survey of Error Analysis.
Proceedings of the International Federation for Information Processing Congress 1971 2:1214-1239, August, 1971.

[18] Lane, J. M., Riesenfeld, R. F.
A Theoretical Development for the Computer Display and Generation of Piecewise Polynomial Surfaces.
IEEE Transactions PAMI 2 , 1980.

[19] Lane, J. M., Riesenfeld, R. F.
Bounds on a polynomial.
BIT: Nordisk Tidskrift for Informations-Behandling 21(1):112-117, 1981.

[20] Lawrence, J. D.
A Catalogue of Special Plane Curves.
Dover Publications, Inc., New York, 1972.

[21] Levin, J. Z.
Mathematical Models for Determining the Intersections of Quadric Surfaces.
Computer Vision, Graphics and Image Processing 11:73-87, 1979.

[22] Lyche, T., Cohen, E., Morken, K.
Knot Line Refinement Algorithms for Tensor Product B-Spline Surfaces.
Computer Aided Geometric Design 2:133-139, 1985.

[23] Symbolics Inc.
VAX UNIX MACSYMA Reference Manual.
Symbolics Inc., 1985.

[24] NAG.
Numerical Algorithms Group FORTRAN Library.
NAG, Oxford, England, 1985.

[25] Patrikalakis, N. M.
Piecewise Continuous Algebraic Curves in Terms of B-Splines.
Submitted for Publication , February, 1987.

[26] Patrikalakis, N. M., Prakash, P. V.
Computation of Algebraic and Polynomial Parametric Surface Intersections.
MIT Sea Grant Report No. 87-19 , 1987.

[27] Patrikalakis, N. M. and Kriezis, G. A.
Representation of Piecewise Continuous Algebraic Surfaces in Terms of B-Splines.
Submitted for Publication , July, 1988.

[28] Petersen, C. S.
Adaptive Contouring of Three-Dimensional Surfaces.
Computer Aided Geometric Design 1:61-74, January, 1984.

[29] Powell, M. J. D.
A Hybrid Method for Nonlinear Equations.
Numerical Methods for Nonlinear Equations, Edited by Rabinowitz, P. :87-114, 1970.

[30] Prakash, P. V.
Computation of Surface-Surface Intersections for Geometric Modeling.
PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, May, 1988.

[31] Requicha, A. A. G, Voelcker, H. B.
Solid Modeling : A Historical Summary and Contemporary Assessment.
IEEE Computer Graphics and Applications 2(2):9-24, March, 1982.

[32] Rossignac J. R., Requicha A. A. G.
Piecewise Circular Curves for Geometric Modeling.
IBM Journal of Research and Development 31(3):296-313, May, 1987.

[33] Sarraga, R. F.
Algebraic Methods for Intersections of Quadric Surfaces in GMSOLID.
Computer Vision, Graphics and Image Processing 22:222-238, 1983.

[34] Sarraga R.F., Waters W.C.
Free-Form Surfaces in GMSOLID: Goals and Issues.
In *Solid Modeling by Computers*, pages 187-204. General Motors, 1984.

[35] Sederberg, T. W.
Implicit and Parametric Curves and Surfaces for Computer Aided Geometric Design.
PhD thesis, Purdue University, August, 1983.

[36] Sederberg T. W., Anderson D. C., Goldman R. N.
Implicit Representation of Parametric Curves and Surfaces.
Computer Vision, Graphics and Image Processing 28(1):72-84, 1984.

[37] Sederberg, T. W.
Planar Piecewise Algebraic Curves.
Computer Aided Geometric Design 1:241-255, 1984.

[38] Sederberg, T. W.
Piecewise Algebraic Surface Patches.
Computer Aided Geometric Design 2:53-59, 1985.

[39] Sederberg, T. W., Parry, S. R.
Comparison of Three Curve Intersection Algorithms.
Computer Aided Design 18(1):58-63, January, 1986.

[40] Solomon, B. J.
Surface Intersection for Solid Modelling.
PhD thesis, Clare College, University of Cambridge, Cambridge, England, 1985.

[41] Thomas, S. W.
Modeling Volumes Bounded by B-Spline Surfaces.
PhD thesis, Department of Computer Science, University of Utah, Salt Lake City, Utah, June, 1984.

[42] Varady, T.
Surface-Surface Intersections for Double-Quadratic Parametric Patches in a Solid Modeller .
Proceedings of the U.K. - Hungarian Seminar on Computational Geometry for CAD/CAM, Cambridge University , 1983.

[43] Walker, R. J.
Algebraic Curves.
Princeton University Press, Princeton, New Jersey, 1950.

[44] Wilkinson, J. H.
Rounding Errors in Algebraic Processes.
H. M. Stationery Office, London, England, 1963.