tanulmányok 171/1985

MTA Számitástechnikai és Automatizálási Kutató Intézet Budapest

COMPUTER AND AUTOMATION INSTITUTE, HUNGARIAN ACADEMY OF SCIENCES

INTEGRATION OF FREE-FORM SURFACES INTO A VOLUMETRIC MODELLER

dr Tamas Varady

Studies 17 1/1985

A kiadásért felelős:

Dr Vámos Tibor

Főosztályvezető:

Dr Nemes László

ISBN 963 311 192 7
ISSN 0324–2951

Alfaprint

CONTENT

ACKNOWLEDGEMENT

I am indebted to the Mechanical Engineering Automation Division of the Computer and Automation Institute, especially to Joe Hatvany, Laszlo Nemes and Julius Hermann for their support and encouragement to conduct this geometric modelling research. The creative atmosphere of the FFS surface modelling group at CAI helped me very much in the clarification of the synthesis problem. Many thanks to Malcolm Sabin and Mike Pratt for directing me to the BUILD Geometric Modelling Group, Cambridge University Engineering Department. I am particularly grateful to all members of the BUILD Group for their hospitality and collaboration, which made it possible to become familiar with the fundamentals of BUILD and to overcome many problems. I am particularly indebted to the head of the group, Graham Jared, who invited me to Cambridge, played an important role in the U.K.-Hungarian research collaboration and made significant contributions to the research described in this thesis, including the correction of my Hungarian-English.

NOTATION

$\underline{r} = (x,y,z)$	-	vector quantities will be denoted by underlining them
$\underline{r}_1\ \underline{r}_2$ or $\underline{r}_1 \cdot \underline{r}_2$	-	scalar product of $\underline{r}_1$ and $\underline{r}_2$
$\underline{r}_1 \times \underline{r}_2$	-	vector product of $\underline{r}_1$ and $\underline{r}_2$
$\lvert \underline{r} \rvert$	-	the absolute value of $\underline{r}$
$\underline{r}_0$	-	index 0 always means unitvector
$A_u = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$	-	sign dependent matrix - means depending on the sign of a certain parameter value (u) the third or the fourth row of the matrix will be taken
$\neq$	-	will be used for non equal
$\underline{r}(t)$	-	vector - scalar function
$\dot{\underline{r}}_t(t)$	-	its derivative by parameter t

Chapter I

INTRODUCTION

1.Synthesis of free-form surfaces and solid modelling

Today's computer aided design systems employ a range of different methods for modelling complex engineering objects. The data-structure created by the modelling systems makes it possible to visualize the objects, draw their orthogonal and perspective views, generate tool-paths for NC machining, analyze their mechanical and thermodynamical characteristics, and calculate their geometric and physical properties. Two-dimensional draughting systems have obvious limitations for the description of 3D shapes. While 3D wireframe modellers offer a greater potential in this respect, the most advanced techniques, which are likely to be basic in future CAD/CAM systems, are those of surface modelling and volumetric modelling.

The two latter disciplines have until recently been evolving separately. Sculptured surface modelling originated in the early sixties, when the theoretical basis of piecewise parametric surfaces was developed by several workers including Coons [23], Ferguson [33], and Bezier [4] . Volumetric modelling started in the late sixties by the works of Braid [9], Voelcker [79] , Okino [51] and others. Initially, the only mode of construction was the use of Boolean set operators on simple solid primitives. The defined objects were usually bounded by simple analytical surfaces (planes,and quadrics) described by implicit equations.

Because of the difference in mathematical surface formulation, the two types of system developed largely independently. While parametric equations are generally used to define bounded curve segments and surface regions, implicit

equations suggest the use of infinite half-spaces. Different mathematical, algorithmic and computational methods were applied; different design methods and user interfaces were created.

By the late seventies, both disciplines had significantly advanced and many commercial systems were released. At the same time it was recognised that there are many deficiences in both areas and that many applications require the simultaneous use of both techniques. Sculptured surface design systems had difficulties in handling topology, in defining composite surfaces, incorporating implicit surface types and in creating closed objects. Solid modelling systems were not able to describe free-form shapes, except by means of coarse approximations. They also faced the problem of accurately representing intersection curves, and blending surfaces, etc., which cannot be defined by simple analytic equations.

From the early eighties, intensive research work has been directed towards the integration of free-form and solid geometry. There are many geometric modelling projects, which attack the problem. Just a short list of them, known to the author, and references are given here: ALPHA-1 [21],[22]; BUILD [15], [1] ; CATIA [3] ;COMPAC [66], EUCLID [67]; EUKLID [31]; GEOMOD [44]; GMSOLID [8] , MODIF/GEOMAP [41],[20],[45]; MEDUSA [37];ROMULUS [6] , [58]; REMUS [77]; SYNTHAVISION [38]; TIPS [52]; PROREN [27]; UNIBLOCK [50]. Details can be found in the article of Pratt and Varady [55].

Some significant results have been published. Nevertheless many questions still remain. In most cases to avoid problems a certain compromise must be made, which results in tight limitations on the user of these advanced modellers.

Some examples are given as follows:

- New methods have been developed for creating free-form solids, but these cannot be combined using set operations in a general way.

- Certain free-form facilities are available in the modeller, but these are insufficient to represent geometrically complex objects.

- The used surface type is appropriate for describing complex geometry (high degree polynomials,and rational functions), but there are restrictions for combining them, for example, free-form surface-surface intersection is not permitted, or the computation time is unacceptable.

- The surfaces are represented only approximately in faceted form, consequently the difficult intersection problem is avoided, but the datastructure may become inaccurate, sometimes inconsistent.

- The surfaces are accurately represented, but the model, including the edge-curves are implicitly defined, thus at each evaluation, computationally expensive, mostly numerical procedures must be applied.

The above listed geometric modelling systems, according to the latest information known to the author, suffer from one or more of these "diseases". The BUILD project (Cambridge University Engineering Department) adopted a compromise on the selection of the surface type, but raised high requirements on the generality of the solution.

A new free-form curve and surface representation called double-quadratics (hereafter abbreviated to dq-s) was chosen. The dq-s not only have mathematical and computational benefits, but also sufficient freedom in making the free-form contra solid synthesis flexible.

The basic idea of double-quadratics is that instead of a cubic blending function, two joining parametric quadratics are used. In this way, a sufficient degree of freedom for designing a wide range of engineering objects is preserved, and significant computational efficiency is gained, which compensates for the loss of internal curvature continuity. The geometric interrogations of dq-curves and surfaces can be solved mostly by analytic and simple numerical methods. Comparing this to other surface types, this leads to robust and efficient procedures, appropriate for building solids bounded by free-form elements as well.

BUILD uses a fully evaluated boundary representation. The edge curves, which lie on the surface-surface intersection curves are also explicitly stored. If these cannot be described by simple analytic equations, a very accurate free-form approximating curve, i.e. a dq-curve is generated. It is fitted using a special technique, and its accuracy can be controlled by the tolerance of the model.

The greatest power of the BUILD dq-implementation is that the free-form geometry is handled exactly in the same way as the conventional one. Therefore, once free-form surfaces have been incorporated into solid models, no distinction is made afterwards, for example, arbitrary two solids with free-form faces can be added together. All BUILD operations are extended to solids bounded by double-quadratic elements. This thesis discusses the computational geometry side of the synthesis, what problems need to be solved for ensuring the uniformity of free-form and conventional elements, and how the double-quadratic curves and surfaces were incorporated into BUILD.

2. Free-form surfaces in engineering design

The use of free-form surfaces in practical engineering design can be categorised into four types from the user's point of view, as was suggested by M.J. Pratt and the author in [55]. These are, in order of increasing degree of geometric constraints on the surface being designed:

(i) Aesthetic surfaces

(ii) Generalised duct surfaces

(iii) Blends and fillets

(iv) Functional or fitted surfaces

The distinctions are by no means clear-cut, and in many cases some overlap may occur.

In the case of aesthetic surfaces, appearence is the main criterion for acceptability. Often there are few precise geometric constraints from the point of view of functionality of what is being designed. During the design process, modifications are made until the appearence is acceptable to the designer. Examples include many plastic mouldings such as casings of household electrical appliances.

Generalised duct surfaces are surfaces, which are subject to more geometric constraints, usually at their boundaries. Often the precise specification of the surface is of no great importance, provided that it gives a smooth blend between the specified boundary conditions, and that its gross characteristics are under the designer's control. Examples include many automotive components such as suspension arms, exhaust manifolds and other types of ducting.

Blends and fillets are required to provide smooth transitions between neighbouring previously defined object faces. In this case, the geometric constraints are clearly greater. The designer may wish to have control over the explicit representation of the fillet surface, in which case he will have to provide some small amount of information to supplement the requirements for tangency with the faces being blended. Alternatively, he may wish to model the fillet explicitly, but will simply label an object edge as being blended. Many general engineering objects exhibit blends and fillets of the kind described, and they are particularly useful in the design of mouldings, castings and forgings.

The fourth class of surfaces is that of functional or fitted surfaces. Here a high degree of geometric constraint is applied to the surface as a whole, which is usually fitted to a large number of measured or precomputed points. Examples include turbine blades and the aerodynamic surfaces of aircrafts; in both cases, the surfaces are originally defined by reference to physical laws, and must satisfy certain optimality criteria. Deviations from the constraints imposed will lead to loss of performance. Fitted surfaces are also widely used in the automotive industry for specifying car body shapes. These might be thought to be of 'aesthetic' variety, but in most companies the aesthetic criteria are imposed during the creation of a clay model, from which subsequently digitised points are used in the generation of a 'fitted' surface.

3. The BUILD geometric modeller and its free-form facilities

To be able to fully understand the significance of the following chapters, first the solid modelling background and the design techniques for creating free-form solids must be presented.

The BUILD geometric modeller (Cambridge University Engineering Department) represents one of the basic approaches in solid modelling [1] ,[15]. All topological and geometrical information concerning the boundary of the object is stored in the so-called boundary representation. The links between the vertices, edges and faces are stored, together with the equations of curves and surfaces on which the edges and faces lie.In this case, the "history" of the object cannot be deduced from the data-structure, but all the necessary boundary data is at hand for subsequent applications.

It must be noted that the other approach, the so-called CSG (Constructive Solid Geometry) faces problems in incorporating free-form geometry, which conventionally requires the use of bounded surfaces. CSG modellers represent the objects in nonevaluated form by a tree-structure, whose leaves are volumetric primitives, often expressed in terms of infinite half-spaces and whose nodes are the Boolean operations performed on them. Thus either the basic CSG half-space concept or the concept of the bounded parametric surfaces must be extended.

The objects in BUILD-4 can be bounded by straight lines, conic segments, dq-curves, planes, general quadrics and dq-surfaces. Starting from primitive solids such as blocks, cylinders, cones, and spheres, etc., the objects can be modified by translation, rotation and scaling, and afterwards combined by set-operations - "or" (add), "and","difference" (subtract), and "negate". BUILD-4 has quite a wide range of local operations [42], which provide a means of making directed changes to objects. These preserve topological and geometric consistency, moreover avoid the expense of the global access to the entire part of the model that is necessary in Boolean set operations. Examples of such local operations are the blending or chamfering of edges, the glueing together of two juxtaposed faces, and the setting of a "draught angle" on the vertical faces of an object, which is to be manufactured by a casting process, etc.

The BUILD group have made many significant contributions not only to the principles and basic theory of geometric modelling [9], [11], [12], but also to the practical application side of this technique in the area of finite--element mesh generation [80], automatic NC tool-path generation [53], dimensioning and tolerancing [13], and feature-recognition [46], etc.

There is a subsystem in BUILD-4 called "design dq", for defining free-form elements. Free-form surfaces can be created by dq-interpolation or by quadratic B-spline approximation [69],[75]. (In both cases dq-patches are defined internally.) In many practical applications, surfaces are designed based on curves or curve-networks, simplifying the whole design procedure. Well-known techniques such as "sweeping" (translation) or "swinging" (rotation) a free-form profile or creating a surface blended through a set of

section curves are also available in the course of a "design dq" session. Final local adjustment of the patches can be made by modifying the position and the tangent vectors of the surface grid. The result of the "design dq" session is an open or closed (at most one boundary) free-form surface, ready for further use, that is to be incorporated into a solid model.

A set of recommended operations for synthetizing free-form surfaces and solid geometry are described in the article of Pratt and Varady [55]. Here only the most important categories will be outlined. These operations are under development in BUILD. The first two groups are complete, the "insert a new face" and the "rubber object" operations together with a limited subset of automatic blending are planned to be completed in the near future.

1. Surface -solid operations

Two operands - a free-form surface and a previously designed solid - are given. The result is a new solid with one or more faces lying on the surface. Examples for these operations are shown in Fig.3.1. "SECTIONING" (b) cuts off a certain part of the solid, "SETSURF" (c) replaces the geometry of a selected face by the free-form surface, and "ADDSURF" (d) does the same, but only "above" the selected face.

2. Free-form primitives

An open or closed free-form surface is given. Adding straightforward geometrical and topological entities according to different rules, primitive solids can be created. Typical examples are the following as shown in Fig.3.2. A DQPRISM (a) is created by projecting the free-form surface onto another surface, and a DQCYLINDER (b) by adding end-faces to a closed free-form surface. A DQOFFSET (d) solid is made by thickening the free-form surface.

Figure 3.1

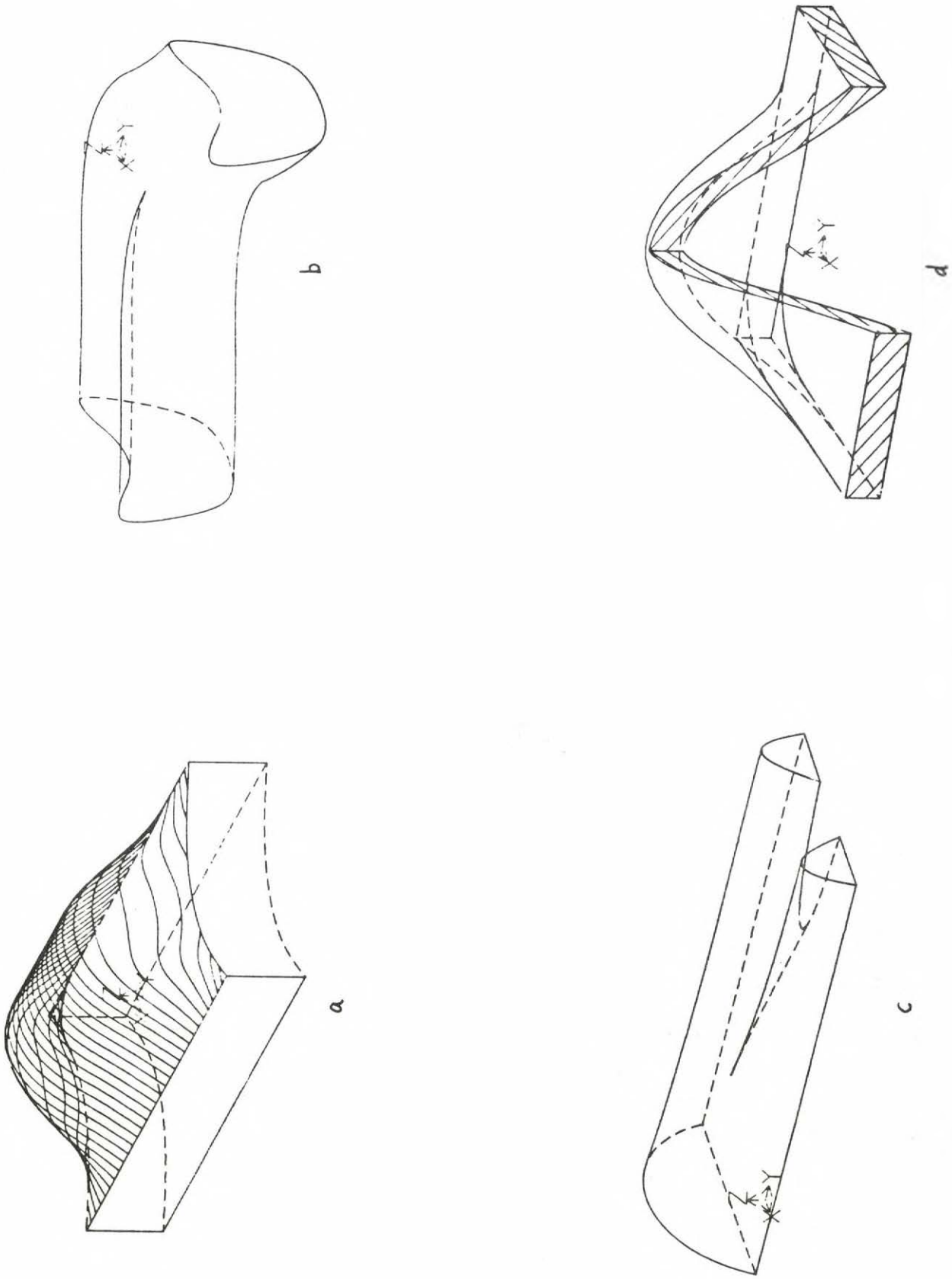


Figure 3.2

3."Insert a new face" operation

A solid is given. Specifying constraints for the corner points and the boundaries of a surface, a new - mostly four sided - face can be added. Either positional or tangential continuity can be specified across the surface boundaries in respect to the existing faces. The interior of the surface can be altered as needed. The topology of the solid is locally changed depending on the position of the boundaries and their tangency. (See Fig.3.3.)

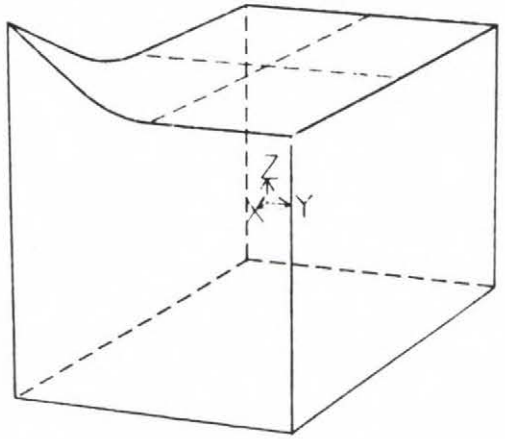
4. "Rubber object" operations

Certain parts of the solid - specified by topological entities, such as vertices, edges, edge-loops, and faces - can be converted to be elastic with or without constraints concerning the degrees of freedom for an elastic deformation. As shown in Fig.3.4, the elastic pieces can be modified or redesigned by using free-form design techniques, while preserving the topology of the original solid.

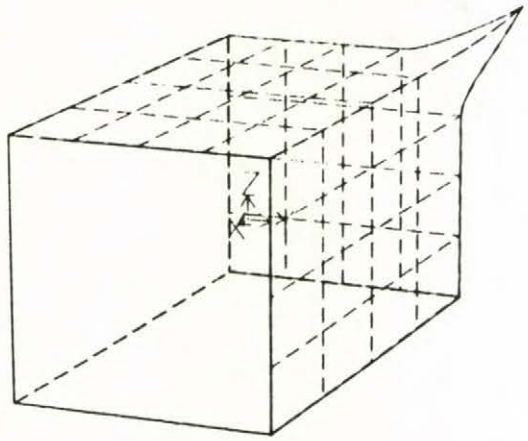
5. Blending operations

This operation is of major importance in engineering design. It may be needed for aesthetic reasons, to ensure machinability along concave edges or to ensure desirable mechanical properties. Blending results in smooth transition surfaces between faces, replacing sharp corners and edges of the given solid. It can be "superficial" named by Braid in [14], i.e. the blends are only glued to the body, or topological, when the blends get evaluated and imply local changes in the boundary datastructure (see Fig.3.5 and 3.6).

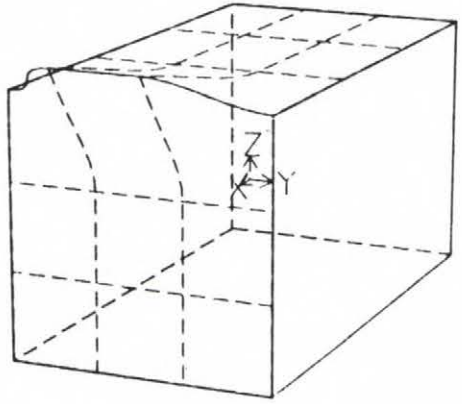
Figure 3.3



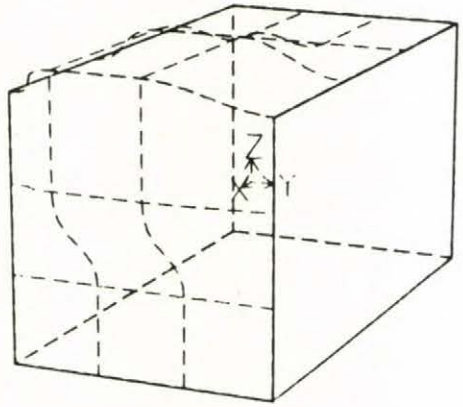
a



b



c



d

Figure 3.4

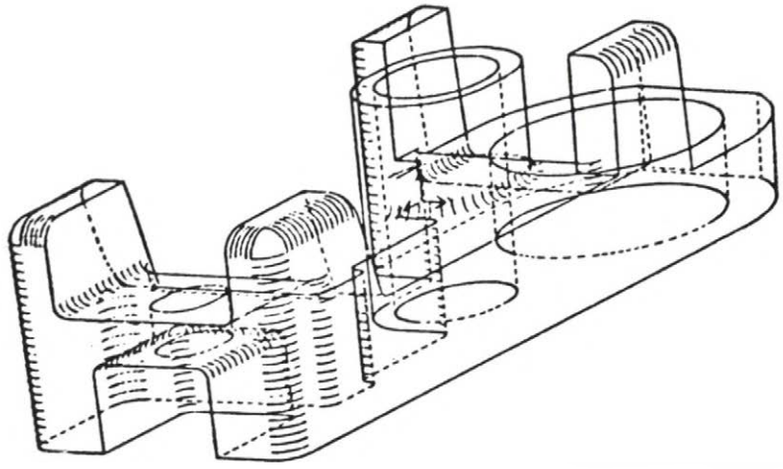


Figure 3.5

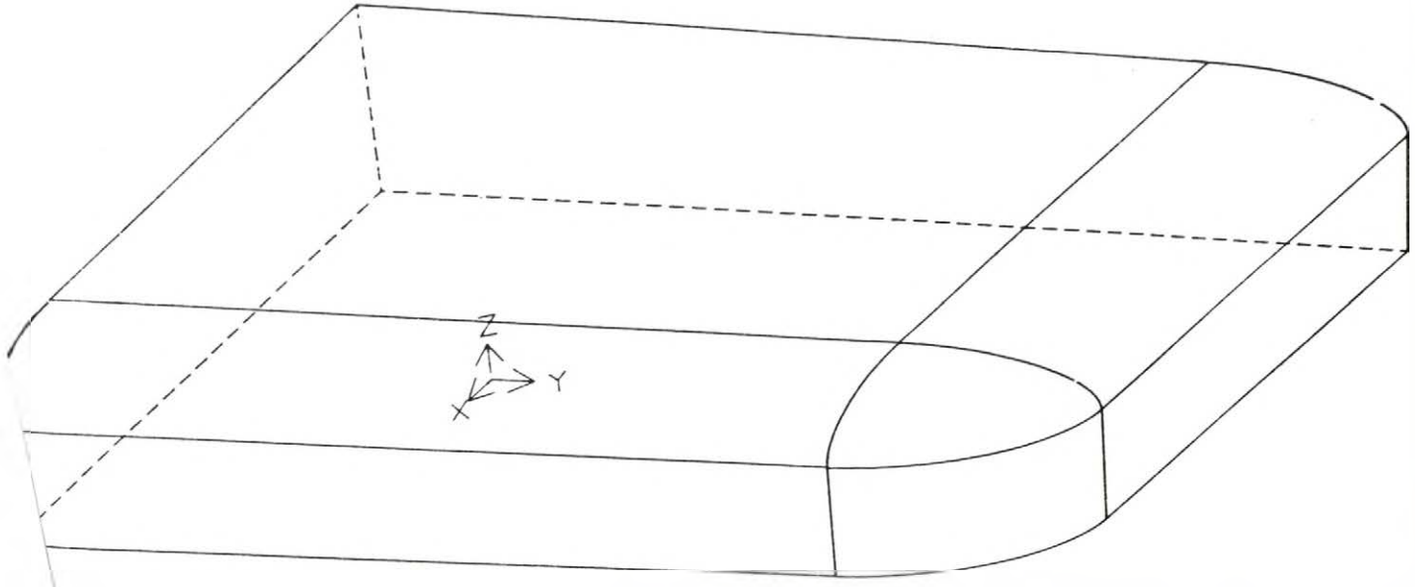


Figure 3.6

There is a general geometric interface in BUILD for all possible geometric interrogations and intersections of all curves and surfaces. All Boolean set operations, local operations and the above special free-form vs. solid operations are implemented purely in terms of generalised points, curves and surfaces. This holds for the graphic subsystem, the NC processor, the finite--element generator, etc., as well. All detailed knowledge of the actual representations used and their properties is confined to a centralised "geometric package". Thus, for example, where a pair of faces are intersected during a set-operation on two objects, the algorithm to process the result deals in a general manner with a set of intersection curves, rather than examining the two surfaces involved, discovering that maybe they are planar and using properties specific to straight line intersection curves.

In this way, new curve and surface types can be gradually introduced by adding the necessary routines to the centralised geometric package, avoiding the necessary comprehensive rewriting of the entire programme at every stage. This applies in the case of the double-quadratic curves and surfaces.

4. Content of the thesis

This thesis is a result of the research work conducted in the Mechanical Engineering Automation Division of the Computer and Automation Institute of the Hungarian Academy of Sciences in the period between 1976 and 1984. The research on the free-form vs. solid synthesis, the interrogation algorithms and the actual implementation of the double-quadratic software in BUILD were carried out during the author's visits to the BUILD Group at the Cambridge University Engineering Department in the spring of 1982 and in the academic year of 1983/84.

This thesis obviously cannot cover all the theoretical and computational problems, which had to be solved to accomplish the project. Only a small part was elaborated by the author, and only a part of this is discussed here. The double-quadratic curve and surface design methods, the algorithms of the free-form vs. solid synthesis and some other topics must be neglected, in order to focus on the main points of this thesis, that is the problems and realization of the geometric algorithms for the previously described centralized geometric package in BUILD.

Basically, this thesis consists of four independent essays (Chapter II, III, IV, V), each of them can stand by themselves and cover one well-defined problem. (This led to some overlaps at certain places. The figures and the mathematical formulas are also numbered by chapters.) The common root is that each of them discusses computational geometry problems related to double--quadratics. After the introductory chapter, which defines and describes double-quadratic curves and surfaces, solid modelling problems are presented.

Based on their mathematical background, new algorithms are described together with their advantages and disadvantages compared to the other methods published in literature. The computational aspects are also focused on throughout the thesis.

In Chapter II the basic equations and the most important geometric properties of double-quadratics are presented. The double-quadratic curve and surface class, as it is stated here is my own "invention". The presented geometric invetigations can be similar to those of other sorts of curves and surfaces.

In Chapter III the geometric interrogations and intersections of double--quadratics are discussed. Apart from the simplest global tests and geometric interrogations these algorithms are the result of my own research work.

In Chapter IV a volumetric modeller oriented curve-fitting algorithm is presented, which uses dq-curves. In Chapter V the problem of visualizing free-form solids is tackled. An algorithm is given for generating silhouette curves of double-quadratic surfaces. The curve-fitting and the silhouette algorithms are also my own research developments.

In Chapter VI a conclusion is drawn. After summarizing the results some practical, mechanical engineering objects are shown to illustrate the design facilities of the BUILD free-form modelling. Suggestions for further work are also given.

The list of references and figures conclude the thesis.

The pictures of the solid objects in this dissertation were all drawn with local hidden-line removal by the BUILD geometric modeller.

Chapter II

BASIC EQUATIONS AND SIMPLE GEOMETRIC PROPERTIES OF DOUBLE-QUADRATIC CURVE SEGMENTS AND SURFACE PATCHES

1. Introduction

The use of piecewise parametric equations in computer aided geometric design was first introduced by Ferguson [33]. Since then the mathematical theory of free-form curve segments and surface patches was thoroughly elaborated in the works of Coons [22], Bezier [4] , Forrest [34], Sabin [59],[61] and others. Several different composite design techniques were developed, well known examples are the spline, Bezier, and B-spline, etc., curves and surfaces [24], [49] , [5] , [39]).

A large part of the existing commercial and research sculptured surface systems uses cubic equations. This is mainly due to the fundamental feature of cubics, that they have sufficient freedom to design complex engineering shapes. In the equation of a cubic curve segment, there are four vector coefficients. A convenient way to describe this segment is to supply the two endpoints and the two tangent vectors there, which uniqely define all four vector coefficients. Another characteristic feature of cubics is that curvature continuity is ensured within a segment.

Considering surface definitions,these are most frequently based on the simplified version of Coons's patches, the bicubic tensor product patches [32], since the corner points and the tangent vectors along the boundaries can be blended in a convenient and straightforward way by cubic polynomials. (Four twist vectors must also be added to make the given 16 vector coefficients fully constrained.)

Many researchers also investigated the properties of parametric quadratics, see for example the quadratic Bezier and B-spline curves. Due to the lack of design freedom they were generally neglected in practical design systems. Only a limited application of them has been published, as outlined here. In 1975, Chaikin proposed a fast algorithm for high-speed curve generation in [18]. Later it was proven ([57], [35]), that the algorithm defines quadratic B-splines. That technique was generalised to the recursive subdivision algorithms for smoothing down irregularly shaped polyhedrons as discussed in Sabin and Doo's papers [29], [30]. Starting from the approximating polyhedron, this method generated a set of quadratic B-spline patches. Where non-four-sided regions were formed, the subdivision must have been repeated, which eventually converged to a smooth, C1 continuous surface. (A new theoretical result by Sabin is that instead of performing subdivisions, the 3 and 5 sided regions can be replaced explicitly by 3 and 5 sided patches, which smoothly join the surrounding ones [63].) This subdivision concept was implemented in the REMUS system [77],however, it turned out that the object definition based on this approximating scheme is quite difficult from an engineering point of view [78].

Returning to cubics - inspite of the previously mentioned attractive features, many difficulties arise when the computation of different geometrical properties, intersections with other geometric elements, etc., are needed, however, in fact, these are the most frequent tasks in computer aided geometric design. Generally, computationally expensive, not very robust, iterative methods must be used. And here comes the idea of double-quadratics. Using two joining parametric quadratic equations instead of the cubic one, the complexity, smoothness and the degrees of freedom of cubics can be retained.

Moreover, reducing the degree of the used polynomials, significant computational efficiency can be gained together with other properties, which emerged later when cubics and double-quadratics were compared.

The "loss" is curvature continuity, which is not necessarily needed in the large majority of engineering applications. The simplest example is an "engineering" piece, where a plane and a cylindrical surface meet each other, obviously with curvature discontinuity. Another argument, however, is that cubics or the higher order polynomials ensure internal curvature continuity, in most design systems, only tangential or normal continuity are satisfied, when the neighbouring patches are actually joined together.

Therefore, it was felt that double-quadratics represent a good compromise between design freedom and computational simplicity. Further investigations were initiated in the sense that not quadratics and cubics, but double--quadratics and cubics were compared with each other. This "double-thinking", according to the author's best knowledge, has not been used in computer aided geometric design. The only similar method was the use of biarc curves for curve-fitting [7]. Now, there is a successor of the double-quadratic patches - the so-called double-cyclides, described by dePont in his recent thesis [25].

Double-quadratics (hereafter abbreviated to dq-s) were first tested in January 1981, as a part of the FFS (Free-Form Shapes) system [68],[36]. Since that time, a large project started in the BUILD geometric modelling Group, Cambridge University Engineering Department for integrating free-form surfaces into volumetric modelling. BUILD facilitates the design of complex

free-form solids bounded by double-quadratic surfaces beside the conventional planar and quadric ones. Details can be found in [43].

In this chapter, the well-known methods of creating composite curves and surfaces are not discussed. It is beleived that there is no single best technique for creating free-form shapes, and different applications may need different methods, different interpolating or approximating schemes. A wide variety of them are described in Faux-Pratt's book [32]. Design with dq-s means that the basic constituting pieces are dq-elements. In this introductory chapter we focus on the basic equations of double-quadratic curve segments and surface patches together with some simple properties around them.

2. The equation of a double-quadratic curve segment

Given a curve segment, defined by its endpoints $\underline{r}_A$, $\underline{r}_B$ and tangent vectors $\dot{\underline{r}}_A$, $\dot{\underline{r}}_B$, its parametric description can be interpreted, that the curve represents the path of a moving point, which runs from $\underline{r}_A$ to $\underline{r}_B$, as the parameter value runs along the parametric interval. The tangent vectors determine the direction and also the velocity of the motion at the endpoints.

Figure 2.1.

Since cubics and double-quadratics are very similar to each other, first let us see how a parametric cubic equation can satisfy the above boundary conditions. It is given in the form of:

$$\underline{r} = \underline{r}(u) = \underline{a}_3 u^3 + \underline{a}_2 u^2 + \underline{a}_1 u + \underline{a}_0 \tag{2.1}$$

where $\underline{a}_i$, $i = 0,1,2,3$ are the vector coefficients of the equation, u is the parameter. Usually the parameter values u=0 and u=1 are assigned to the endpoints, with $0 < u < 1$ in between. The shape of the cubic segment is uniquely defined by the above four independent vector quantities, since four degrees of freedom are available.

Satisfying the end conditions, the following system of four equations can be obtained:

$$\begin{aligned} \underline{a}_0 &= \underline{r}_A \\ \underline{a}_0 + \underline{a}_1 + \underline{a}_2 + \underline{a}_3 &= \underline{r}_B \\ \underline{a}_1 &= \dot{\underline{r}}_A \\ \underline{a}_1 + 2\underline{a}_2 + 3\underline{a}_3 &= \dot{\underline{r}}_B \end{aligned} \tag{2.2}$$

Solving this for $\underline{a}_0$, $\underline{a}_1$, $\underline{a}_2$, $\underline{a}_3$ the following expressions are obtained:

$$\begin{aligned} \underline{a}_0 &= \underline{r}_A \\ \underline{a}_1 &= \dot{\underline{r}}_A \\ \underline{a}_2 &= 3 (\underline{r}_B - \underline{r}_A) - 2 \dot{\underline{r}}_A - \dot{\underline{r}}_B \\ \underline{a}_3 &= 2 (\underline{r}_A - \underline{r}_B) + \dot{\underline{r}}_A + \dot{\underline{r}}_B \end{aligned} \tag{2.3}$$

For computational purposes generally the well-known blending function formalism is used [32]. The blending functions ensure the appearence and the disappearence of the given positional and tangential constraints as the parametric variable runs along the parametric interval. If the f_0, f_1, g_0, g_1 blending functions satisfy the following conditions

$$
\begin{aligned}
&f_0(0) = 1, \quad f_0(1) = 0, \quad f_1(0) = 0, \quad f_1(1) = 1, \\
&f_0'(0) = f_0'(1) = f_1'(0) = f_1(1) = 0, \\
&g_0(0) = g_0(1) = g_1(0) = g_1(1) = 0, \\
&g_0'(0) = 1, \; g_0'(1) = 0, \; g_1'(0) = 0, \; g_1'(1) = 1
\end{aligned} \tag{2.4}
$$

then the above specified curve-segment can be written in the following form:

$$\underline{r}(u) = \underline{r}_A f_0(u) + \underline{r}_B f_1(u) + \dot{\underline{r}}_A g(u)_0 + \dot{\underline{r}}_B g_1(u). \tag{2.5}$$

In the cubic case, for example:

$$
\begin{aligned}
f_0(u) &= 1 - 3u^2 + 2u^3 \\
f_1(u) &= 3u^2 - 2u^3 \\
g_0(u) &= u - 2u^2 + u^3 \\
g_1(u) &= -u^2 + u^3
\end{aligned} \tag{2.6}
$$

This leads us to the commonly used matrix-equation of cubic segments:

$$\underline{r} = \underline{U}\,\underline{C}\,\underline{S} \tag{2.7}$$

where

$$\underline{U} = [\, 1 \;\; u \;\; u^2 \;\; u^3 \,],$$

$$\underline{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}, \qquad \underline{S} = \begin{bmatrix} \underline{r}_A \\ \underline{r}_B \\ \dot{\underline{r}}_A \\ \dot{\underline{r}}_B \end{bmatrix}.$$

A parametric double-quadratic curve segment is defined in the form of:

$$(2.8)\qquad \underline{r} = \underline{r}(u) = \begin{cases} \underline{r}^-(u) = \underline{a}_2 u^2 + \underline{a}_1 u + \underline{a}_0 & -0.5 <= u <= 0 \\ \underline{r}^+(u) = \underline{b}_2 u^2 + \underline{b}_1 u + \underline{b}_0 & 0 < u <= 0.5 \end{cases}$$

where $\underline{a}_i, \underline{b}_i$; $i = 0,1,2$ are the vector coefficients of the equation, u is the parameter.

As the equation shows, the segment is composed of two quadratics. Preserving the unit parametric length, the parameter is chosen to run from -0.5 to 0.5 for symmetry reasons. We have gained four constraints as previously and six degrees of freedom, since there are six vector coefficients to be determined in (2.8). The remaining two are for satisfying positional and first derivative continuity at u=0. After substitution the following system of equations is obtained:

$$(2.9)\qquad \begin{aligned} 0.25\underline{a}_2 - 0.5\underline{a}_1 + \underline{a}_0 &= \underline{r}_A, \\ -\underline{a}_2 + \underline{a}_1 &= \dot{\underline{r}}_A, \\ 0.25\underline{b}_2 + 0.5\underline{b}_1 + \underline{b}_0 &= \underline{r}_B, \\ \underline{b}_2 + \underline{b}_1 &= \dot{\underline{r}}_B \end{aligned}$$

At the midpoint

$$(2.10)\qquad \lim_{u \to 0} \underline{r}^-(u) = \lim_{u \to 0} \underline{r}^+(u)\ ,$$

$$\lim_{u \to 0} \dot{\underline{r}}^-(u) = \lim_{u \to 0} \dot{\underline{r}}^+(u)\ ,$$

thus:

$$(2.11)\qquad \underline{a}_0 = \underline{b}_0 \quad \text{and} \quad \underline{a}_1 = \underline{b}_1.$$

Note, that $\underline{a}_0$, $\underline{b}_0$ and $\underline{a}_1$, $\underline{b}_1$ give the midpoint position and tangent vectors respectively. Solving the system of the above six equations for $\underline{a}_i$, $\underline{b}_i$ we gain the solution below:

$$
\begin{aligned}
\underline{a}_0 &= \underline{b}_0 = 0.5(\underline{r}_A + \underline{r}_B) + 0.125(\dot{\underline{r}}_A - \dot{\underline{r}}_B) \\
\underline{a}_1 &= \underline{b}_1 = 2(\underline{r}_B - \underline{r}_A) - 0.5\,(\dot{\underline{r}}_A + \dot{\underline{r}}_B) \\
\underline{a}_2 &= 2(\underline{r}_B - \underline{r}_A) - 0.5(3\dot{\underline{r}}_A + \dot{\underline{r}}_B) = \underline{a}_1 - \dot{\underline{r}}_A \\
\underline{b}_2 &= 2\,(\,\underline{r}_A - \underline{r}_B) + 0.5\,(\,\dot{\underline{r}}_A + 3\,\dot{\underline{r}}_B) = -\underline{b}_1 + \dot{\underline{r}}_B
\end{aligned}
\qquad (2.12)
$$

The blending function equation holds for double-quadratics as in case of cubics, see (2.4). All double-quadratic blending functions are composed of two quadratic pieces, as follows:

$$
\begin{aligned}
f_0^-(u) &= -2u^2 - 2u + 0.5, & f_0^+(u) &= 2u^2 - 2u + 0.5, \\
f_1^-(u) &= 2u^2 + 2u + 0.5, & f_1^+(u) &= -2u^2 + 2u + 0.5, \\
g_0^-(u) &= -1.5u^2 - 0.5u + 0.125, & g_0^+(u) &= 0.5u^2 - 0.5u + 0.125, \\
g_1^-(u) &= -0.5u^2 - 0.5u - 0.125, & g_1^+(u) &= 1.5u^2 - 0.5u - 0.125.
\end{aligned}
\qquad (2.13)
$$

To be able to use the relating matrix formalism, we have to introduce the notation of sign-dependent coefficients. For example,

$$
x_u = \begin{cases} - & 5 \\ + & 8 \end{cases}
\qquad (2.14)
$$

means, that depending on whether another variable, say u is less-equal or greater than zero, x_u equals 5 or 8, respectively. Going further, using signdependent matrices, the third or the fourth row must be taken depending on the sign of another variable. In the case of a double-quadratic curve segment, the equation can be given in the form of :

(2.15) $$\underline{r} = \underline{U}\,\underline{DQ}_u\,\underline{S},$$

where

$$\underline{U} = [\,1 \;\; u \;\; u^2\,],$$

$$\underline{DQ}_u = \begin{bmatrix} 0.5 & 0.5 & 0.125 & -0.125 \\ -2 & 2 & -0.5 & -0.5 \\ -\begin{bmatrix} -2 & 2 & -1.5 & -0.5 \end{bmatrix}- \\ +\begin{bmatrix} 2 & -2 & 0.5 & 1.5 \end{bmatrix}+ \end{bmatrix},$$

$\underline{S}$ is as in (2.7).

The so-called fullness of double-quadratic curve-segments can be adjusted in the same way as in case of cubics. With fixed endpoints, the $\dot{r}_A$ and $\dot{r}_B$ tangent vectors can be written in the form below:

(2.16) $$\dot{r}_A = \alpha\, \dot{r}_{A0},$$
$$\dot{r}_B = \beta\, \dot{r}_{B0}.$$

$\dot{r}_{A0}$ and $\dot{r}_{B0}$ defines the unit direction vectors, α and β defines the magnitudes of the tangents. Having fixed the unit vectors, the adjustment of α and β alters the shape of the segment, as shown in Fig.2.2 and Fig.2.3.

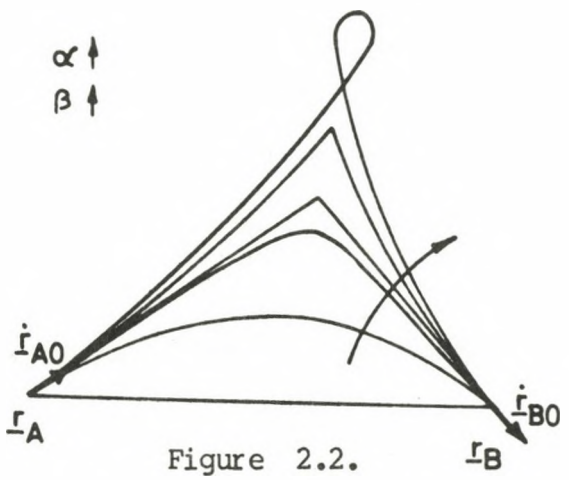


Figure 2.2.

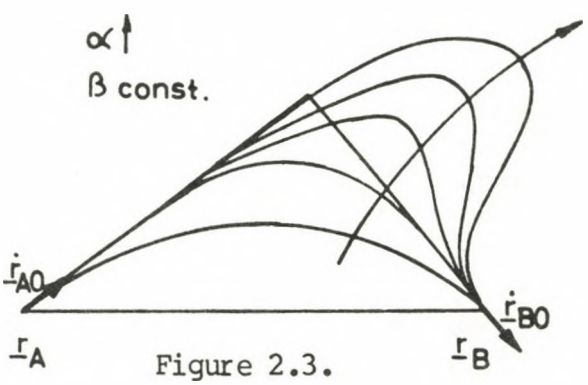


Figure 2.3.

This feature is significant from the design point of view, that is when local shape modification is needed and also for adequate curve fitting, where the best alternative is generated by this sort of adjustment of fullness.

3. The characteristic polygon of a double-quadratic curve segment

Bezier chose a special representation for defining curves [4]. From a mathematical point of view, the cubic Bezier curve is only a rearrangement of the (2.5) equation. In Bezier's form:

$$\underline{r}(u) = (1-u)^3 \underline{r}_0 + 3u(1-u)^2 \underline{r}_1 + 3u^2(1-u)\underline{r}_2 + u^3 \underline{r}_3, \tag{3.1}$$

where

$$\begin{aligned} \underline{r}_0 &= \underline{r}_A, \\ \underline{r}_1 &= \underline{r}_A + \dot{\underline{r}}_A/3, \\ \underline{r}_2 &= \underline{r}_B - \dot{\underline{r}}_B/3, \\ \underline{r}_3 &= \underline{r}_B. \end{aligned} \tag{3.2}$$

The significance of this formulation is that by means of the above vectors, the so-called characteristic polygon can be constructed (Fig.3.1). In fact, the curve is characterized by this polygon, since it passes through the endpoints and it is tangential to the $(\underline{r}_1 - \underline{r}_0)$ and $(\underline{r}_3 - \underline{r}_2)$ vectors, respectively. The characteristic polygon is not only an adequate tool for adjusting curves, but it facilitates the understanding of many relating geometric properties [32]. These include the convex hull property and also the straightforward geometric interpretation of curvature, points of inflections, and loops, etc.

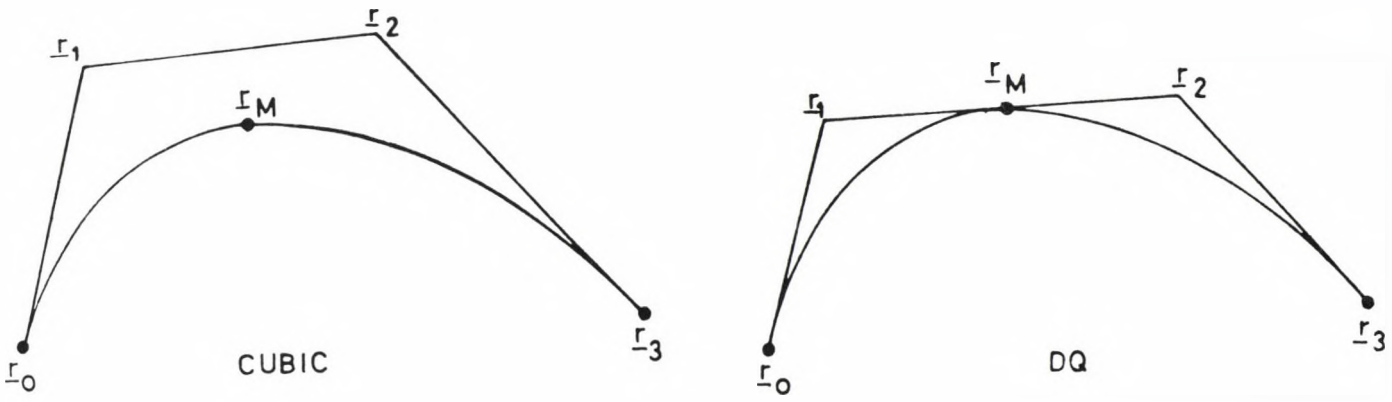


Figure 3.1.

The characteristic polygon of double-quadratics can be derived analogously (based on the quadratic Bezier formulation):

$$
\begin{aligned}
\underline{r}_0 &= \underline{r}_A, \\
\underline{r}_1 &= \underline{r}_A + \dot{\underline{r}}_A/4, \\
\underline{r}_2 &= \underline{r}_B - \dot{\underline{r}}_B/4, \\
\underline{r}_3 &= \underline{r}_B.
\end{aligned}
\tag{3.3}
$$

Unlike in equation (3.2) here we use one-fourth of the tangent vectors. The properties of the cubic Bezier curve can also be generalized for double-quadratics. Moreover, calculating the $\underline{r}(u)$ and $\dot{\underline{r}}(u)$ vectors at the midpoint we obtain:

$$
\begin{aligned}
\underline{r}(0) &= 0.5(\underline{r}_A + \underline{r}_B) + 0.125(\dot{\underline{r}}_A - \dot{\underline{r}}_B), \\
\dot{\underline{r}}(0) &= 2(\underline{r}_B - \underline{r}_A) - 0.5(\dot{\underline{r}}_A + \dot{\underline{r}}_B)
\end{aligned}
\tag{3.4}
$$

This means that the midpoint of a double-quadratic segment is equal to the midpoint of the characteristic polygon, that is to the arithmetic mean of $\underline{r}_1$ and $\underline{r}_2$. Furthermore, the difference vector of $\underline{r}_2$ and $\underline{r}_1$ is tangential to the curve there.

$$
\underline{r}(0) = 0.5(\underline{r}_1 + \underline{r}_2), \qquad \dot{\underline{r}}(0) = 2(\underline{r}_2 - \underline{r}_1). \tag{3.5}
$$

To sum it up , the characteristic polygon of double-quadratics gives a more convenient tool for local adjustment than that of cubics. Constructing the characteristic polygon, one can immediately see the behaviour of the curve, since not only the endpoints, but the midpoint and the corresponding tangent vector become "visible" in this way.

Double-quadratic curve-segments can represent twisted 3D curves as well, in spite of that they are made up of planar pieces, as shown in Fig.3.2 below. The twist is always at the midpoint, unlike cubics, where it changes continuously along the segment.

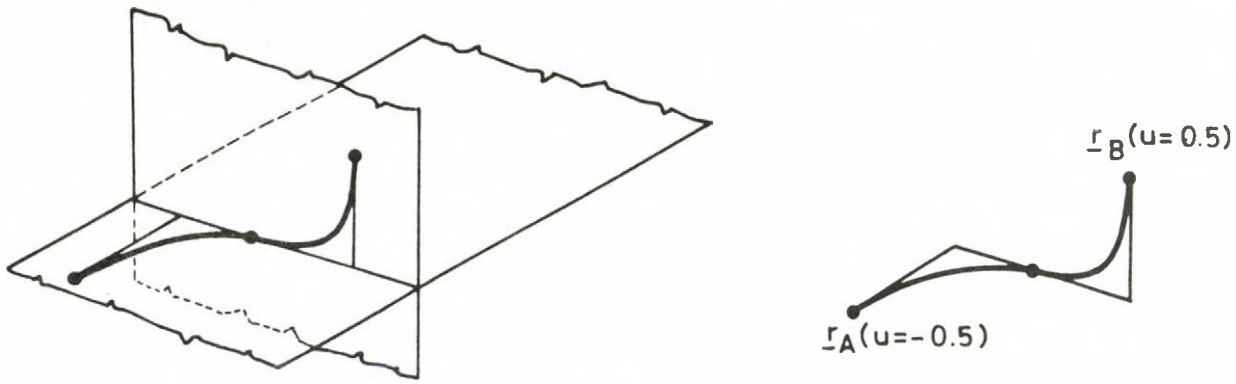


Figure 3.2.

The midpoint of the cubic segment can be expressed by substituting u=0.5 into the (2.5) equation.

$$\underline{r}(0.5) = 0.5(\underline{r}_A + \underline{r}_B) + 0.125(\dot{\underline{r}}_A - \dot{\underline{r}}_B) \tag{3.7}$$

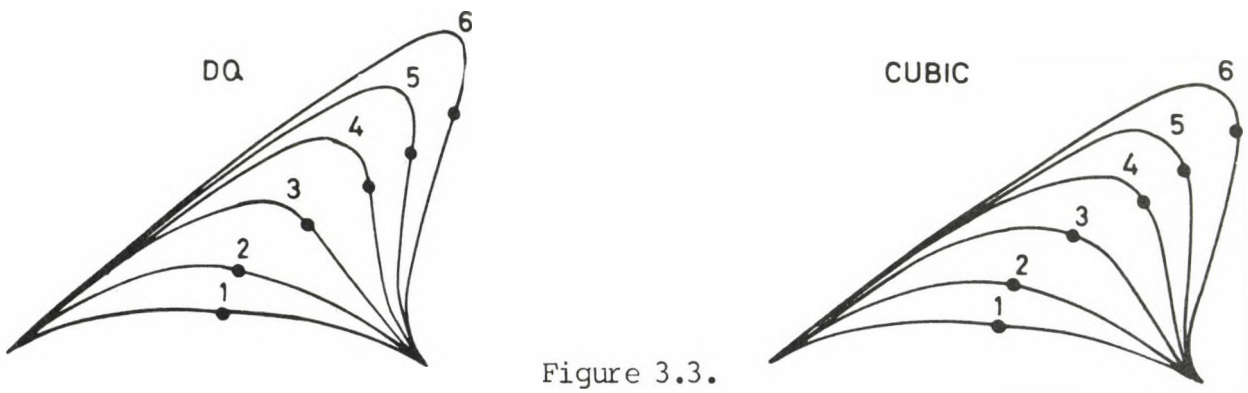


Figure 3.3.

Surprisingly, this is the same for the double-quadratic segment with the same end-conditions. This is one more argument for the previously mentioned similarity (see Fig. 3.3), i.e. the two different types of curve segments beside behaving similarly in the vicinity of the endpoints, go through the same midpoint at the parametric half. It must also be noted that since the characteristic polygon of dq-s is the 3/4-th of cubics, more efficient convex hull tests can be performed on them.

4. Some properties of the planar double-quadratic curve segment

4.1. Curvature, inflection, loops

Examining the shape of a parametric curve segment, one may wonder whether it has got a point of inflection or it is looping. Having the given position and tangent vectors, these questions can be easily answered by means of the characteristic polygon or by algebraic analysis of the curvature. We follow the geometric interpretation.

The curvature of a parametric curve is characterized by the vector $\kappa(u)\ \underline{B}(u)$, where $\kappa(u)$ is the reciprocal value of the radius of the osculating circle, $\underline{B}(u)$ denotes the so-called binormal unit vector.

$$(4.1) \qquad \kappa(u) = \frac{|\dot{\underline{r}}(u) \times \ddot{\underline{r}}(u)|}{|\dot{\underline{r}}(u)|^3} \qquad \underline{B}(u) = \frac{\dot{\underline{r}}(u) \times \ddot{\underline{r}}(u)}{|\dot{\underline{r}}(u) \times \ddot{\underline{r}}(u)}$$

Denoting the tangent and normal unit vectors of the curve by $\underline{T}$ and $\underline{N}$, respectively, we can write:

$$(4.2) \qquad \underline{N} = \underline{B} \times \underline{T}.$$

In the case of planar curves, $\underline{B}$ is always perpendicular to the plane, where the curve and the corresponding $\underline{T}$, $\underline{N}$ vectors lie. $\underline{N}$ always points to the centre of the osculating circle. The sign of the $\dot{\underline{r}}(u) \times \ddot{\underline{r}}(u)$ vector product tells us in which side of the curve this centre lies. Whenever the $\underline{B}$ unit vector changes its sign, a point of inflection occurs, supposing $\dot{\underline{r}}(u) \neq 0$. This last condition holds in the practical cases, otherwise we get cusps as in Fig.4.1.

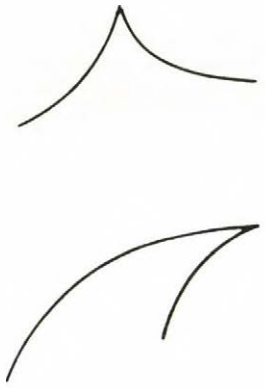


Figure 4.1.

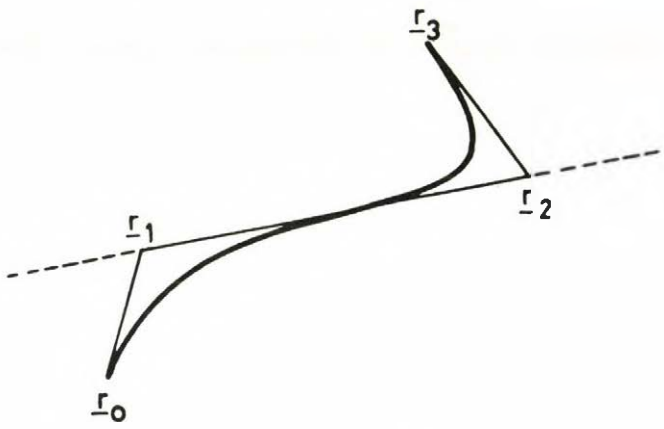


Figure 4.2.

Expressing the curvature by the scalar coefficients of a dq curve-segment, we obtain:

$$(4.3)\quad \bar{K}(u) = \begin{cases} \dfrac{a_{2y}a_{1x} - a_{2x}a_{1y}}{|\dot{\underline{r}}^{-}(u)|^3} & -0.5 \leq u \leq 0 \\ \dfrac{b_{2y}b_{1x} - b_{2x}b_{1y}}{|\dot{\underline{r}}^{+}(u)|^3} & 0 < u \leq 0.5 \end{cases}$$

that is the sign of the curvature is constant within one quadratic piece, and can change only at the midpoint. This means that a double-quadratic segment has a point of inflection if and only if the sign of the above expressions differ. It can be seen that the two pieces either join each other with curvature continuity, if the two numerators are equal, or there is a jump in $\kappa(u)$, if not.

One straightforward way of describing curvatures is using the vectors of the Bezier-like characteristic polygon.Using the equations (2.8) and (3.3) we obtain:

$$
\begin{aligned}
\dot{\underline{r}}(-0.5) &= 4(\underline{r}_1 - \underline{r}_0) \\
\ddot{\underline{r}}(-0.5) &= 8(\underline{r}_0 - \underline{r}_1) + 4(\underline{r}_2 - \underline{r}_1) \\
\dot{\underline{r}}(\ 0.5) &= 4(\underline{r}_3 - \underline{r}_2) \\
\ddot{\underline{r}}(\ 0.5) &= 8(\underline{r}_2 - \underline{r}_3) + 4(\underline{r}_1 - \underline{r}_2)
\end{aligned}
\tag{4.4}
$$

Consequently

$$
\begin{aligned}
\kappa(-0.5) &= [(\underline{r}_1 - \underline{r}_0) \times (\underline{r}_2 - \underline{r}_1)] \ / \ 4|\underline{r}_1 - \underline{r}_0|^3 \\
\kappa(0.5) &= [(\underline{r}_3 - \underline{r}_2) \times (\underline{r}_1 - \underline{r}_2)] \ / \ 4|\underline{r}_3 - \underline{r}_2|^3
\end{aligned}
\tag{4.5}
$$

Since the curvature does not change its sign within one quadratic piece, the centre of the osculating circle in the first half "directs" towards the chord P_0P_2, in the second towards P_1P_3. As was shown, the signs of the curvature at the endpoints are defined by the vectors of the characteristic polygon, and these determine the curvature in the whole segment. Considering the straight line, which goes through P_1 and P_2, if P_0 and P_3 lie in the same side of the line, then:

$$
\underline{B}(-0.5) = \underline{B}(0.5), \tag{4.6}
$$

if P_0 and P_3 lie in different sides, then the signs differ and there is a point of inflection at the midpoint.

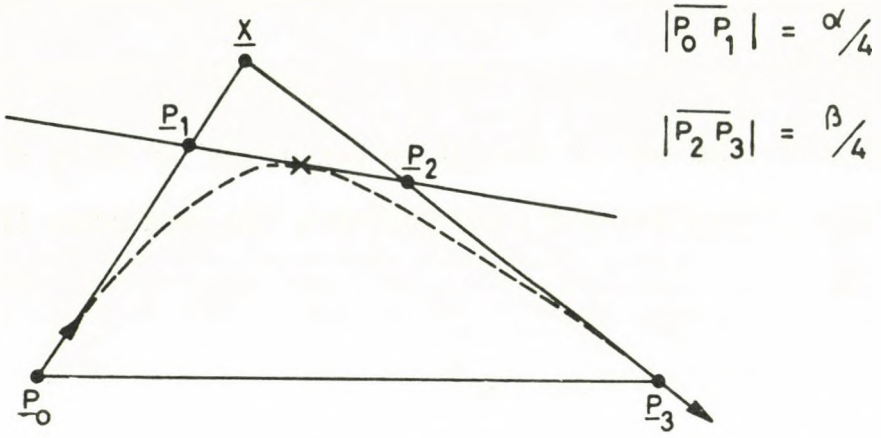


Figure 4.3

Before going further we introduce the notion of the characteristic triangle. It is composed of the endpoints P_0, P_3 and the point X, which is the intersection point of the two tangential lines going through the endpoint with the given slopes. Let α and β be the magnitudes of the tangent vectors, their sense is positive if $(r_1 - r_0)$ directs towards X and $(r_3 - r_2)$ from away X. Fixing the unit tangent vectors, one can vary the α , β magnitudes creating shapes with different fullness. Fig. 4.4 shows some examples, separating the curves according to the sign of α , β into four different classes. (Class III is symmetric to Class II.) The previously mentioned rule on the characteristic polygon and the points of inflections are also illustrated in Fig. 4.4.

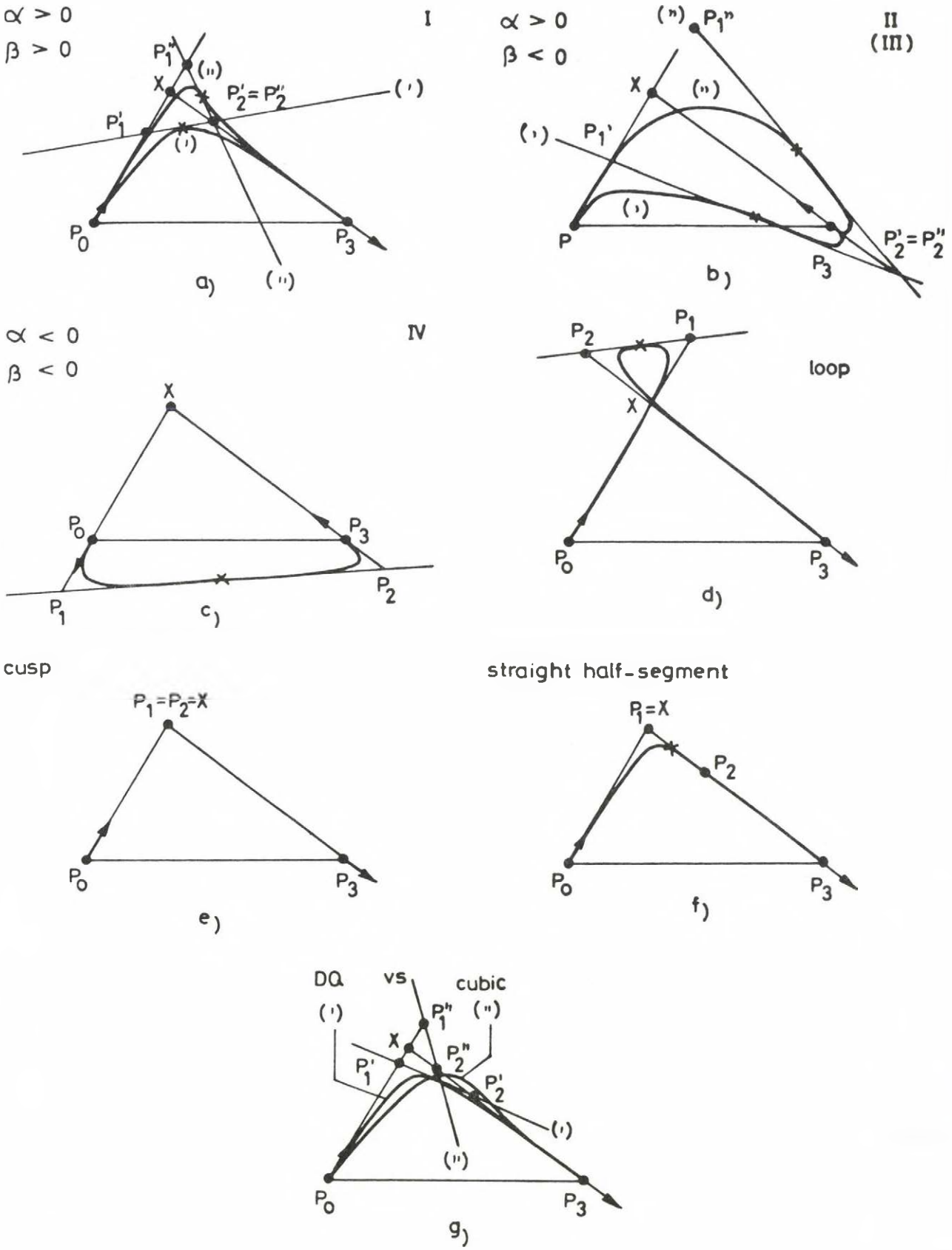

α > 0
β > 0
I
P1"
(")
X
P2' = P2"
(')
P1'
(')
P0
P3
a)
(")
α > 0
β < 0
(")
P1"
II
(III)
X
(")
(')
P1'
(')
P
P2' = P2"
P3
b)
α < 0
β < 0
IV
X
P0
P3
P2
P1
c)
P2
P1
X
loop
P0
P3
d)
cusp
P1 = P2 = X
P0
P3
e)
straight half-segment
P1 = X
P2
P0
P3
f)
DQ
vs
cubic
(')
P1"
(")
X
P2"
P1'
P2'
(')
(")
P0
P3
g)

Figure 4.4.

The condition for a point of inflection can easily be deduced, as mentioned in the previous paragraphs. If the internal points of the polygon - $\underline{r}_1$ and $\underline{r}_2$ are both within the P_0X and XP_3 chord or outside it, there will be no inflection, however, if one of them is inside and the other is outside, a point of inflection occurs. (It is noted that this result can be easily reached in an algebraic way, as well.) Denoting P_0X by α^*, XP_3 by β^* our condition can be expressed as follows:

A point of inflection occurs if

(4.7)

a). $(0 < \alpha/4 < \alpha^*)$ and $(\beta/4 < 0$ or $\beta/4 > \beta^*)$

or

b). $(\alpha/4 < 0$ or $\alpha/4 > \alpha^*)$ and $(0 < \beta/4 < \beta^*)$

The above simple apparatus can help us in examining degenerate cases, keeping in mind that a dq-curve is always tangential to the P_2P_3 chord. Looping occurs if and only if both P_2 and P_3 lie on the positive extensions of the P_0X and P_3X chords, since this is the only possible case, when the direction of the P_2P_3 vector can turn over. With algebraic terms a loop occurs (Fig.4.4), if

(4.8) $\alpha^* < \alpha/4$, $\beta^* < \beta/4$.

A cusp occurs when $\alpha/4 = \alpha^*$ and $\beta/4 = \beta^*$, see Fig.4.4/e.
In this very special case, we still have the first derivative continuity, since approaching X from both end $\dot{\underline{r}}(u)$ becomes zero. (In practical cases we attempt to avoid this.) Another special case, when one of the quadratic pieces is a linear segment, see Fig.4.4/f. This happens when either $\underline{r}_2$ or $\underline{r}_3$ is equivalent to X. (Special care must again be taken to avoid the $\dot{\underline{r}}(u) = 0$ case.)

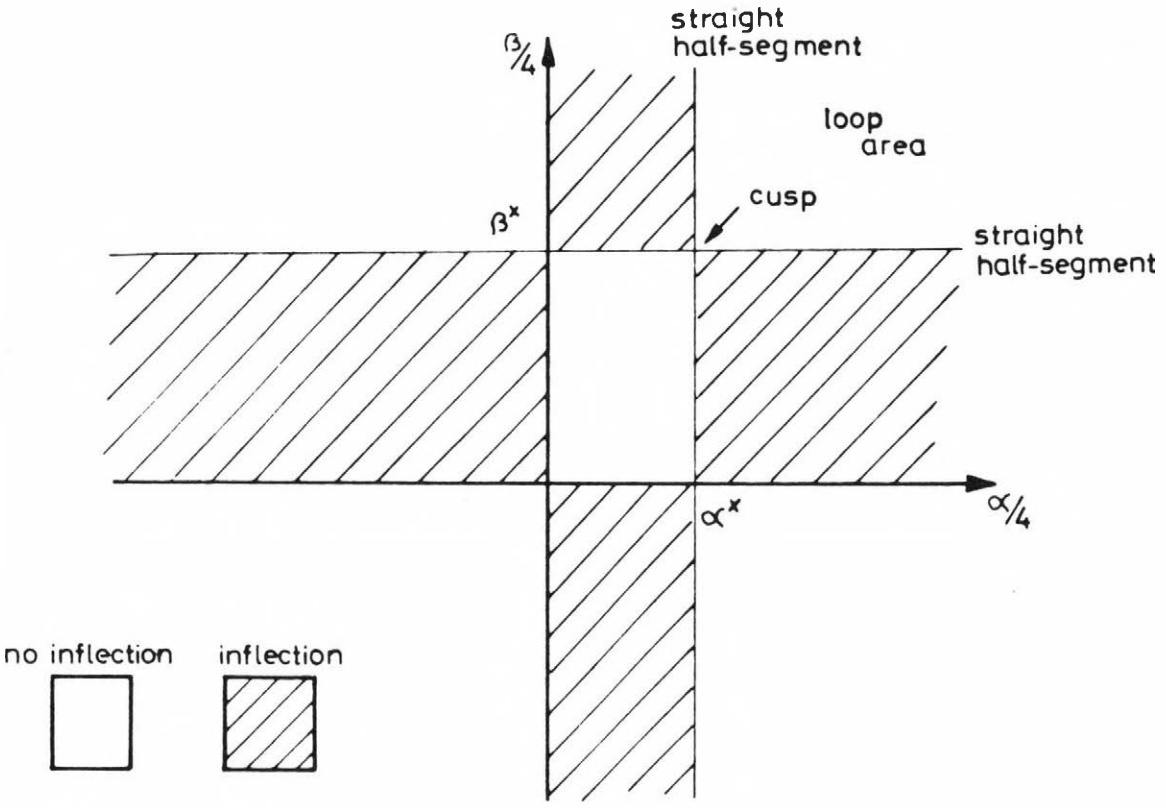


Figure 4.5.

The previous statements can be summarized in the above diagram, where as a function of α and β the different subcases are shown in the $\alpha - \beta$ plane.

It is noted that similar investigations and rules can be established for cubics. Instead of $\alpha/4$ and $\beta/4$, $\alpha/3$ and $\beta/3$ must be used. Cubic curve-segments may have two inflections. Nevertheless there are practical cases, when having the same endpoints and tangent vectors the cubic curve has a point of inflection, while the dq one does not. This happens when:

(4.9) $\quad \alpha/4 < \alpha^* < \alpha/3 \quad$ and $\quad 0 < \beta/3 < \beta^*, \quad$ or

$0 < \alpha/3 < \alpha^* \quad$ and $\quad \beta/4 < \beta^* < \beta/3,$

as is shown in Fig.4.4/g.

4.2. Line and circle approximation

In computer aided geometric design, the conventional elements of engineering drawings must often be used to describe free-form shapes. In the case of planar geometry, often line segments and circular arcs are incorporated into a free-form curve. That is why the question arises: how accurately a double-quadratic segment can represent a line or a circle.

Linesegments can be described accurately by dq-curves, however, special care must be taken to choose the tangent vectors at the endpoints, which determine the parametric distribution of the segment. The $\dot{\underline{r}}(u)=0$ case must be avoided. The half of the dq-curve-segment is either a linear segment, when

(4.9) $$\underline{r}(u) = \underline{a}_1 u + \underline{a}_0,$$

or a quadratic one

(4.10) $$\underline{r}(u) = \underline{a}_2 u^2 + \underline{a}_1 u + \underline{a}_0,$$

with $\underline{a}_2 \times \underline{a}_1 = 0$. In both cases, $\underline{a}_2 = \lambda\, \underline{a}_1$, where λ is a scalar constant.

Denoting the chordlength of the segment by d and the positive tangent magnitudes by α and β, it can be easily proved by substituting into the (2.8) basic equations that if $\alpha + \beta < 4d$ then no zero first derivative occurs. This corresponds to the results obtained in the previous section, i.e. - geometrically interpreting - the $\underline{r}_0$, $\underline{r}_1$, $\underline{r}_2$, $\underline{r}_3$ vectors must lie within the chord in this order, which is equivalent to the above condition in the form of:

(4.11) $$|\underline{r}_0 - \underline{r}_1| + |\underline{r}_2 - \underline{r}_3| < |\underline{r}_0 - \underline{r}_3|.$$

Otherwise the following cases, which are degenerate from engineering point of view may occur:

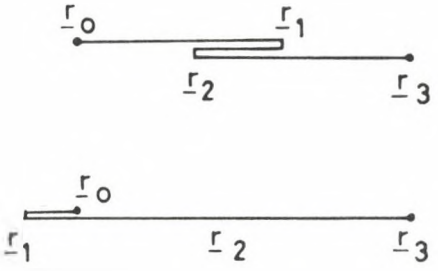


Figure 4.6.

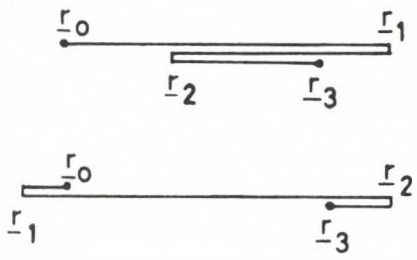


Figure 4.7.

As it is well-known, circular arcs can be exactly represented only by rational functions, though the approximation by double-quadratics and cubics, etc., are particularly good, when the angle of the arc is not too large. For qualititive investigation, let us assume that the arc to be approximated has unit radius and its angle is φ. Let us assume that the approximating dq-curve goes through the arc endpoints and its midpoint and is also tangential to the arc at these points as is shown in Fig. 4.8. (Similar assumptions and results were obtained by Peters for cubic approximations in [54]. It must also be noted that a better approximation can be made by releasing the midpoint constraints - see for example [56].)

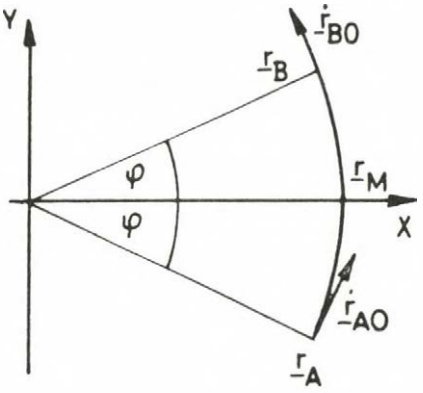


Figure 4.8.

The above conditions lead to the following equations; here k denotes the unknown tangent vector magnitudes.

$$
\begin{aligned}
&\text{(i)} && \underline{r}(-0.5) = (\cos\varphi , -\sin\varphi) \\
&\text{(ii)} && \dot{\underline{r}}(-0.5) = k (\sin\varphi , \cos\varphi) \\
(4.12)\ &\text{(iii)} && \underline{r}(0.5) = (\cos\varphi , \sin\varphi) \\
&\text{(iv)} && \dot{\underline{r}}(0.5) = k (-\sin\varphi , \cos\varphi) \\
&\text{(v)} && \underline{r}(0) = (\cos\varphi + 0.25k \sin\varphi , 0) \\
&\text{(vi)} && \dot{\underline{r}}(0) = (0, 4\sin\varphi - k \cos\varphi)
\end{aligned}
$$

The last two equations came from the midpoint expressions in (3.4). Since $\underline{r}(0)$ is equal to (1,0), k can be directly expressed from (v).

$$(4.13) \qquad k = 4 (1 - \cos\varphi) / \sin\varphi$$

Now k is defined and we look for the extremum values of the approximation. The $h(u) = [\underline{r}(u)]^2$ function is examined instead of the inconvenient $h(u)=|\underline{r}(u)|$. After differentiating, we obtain the following expression:

$$(4.14) \qquad h(u) = \underline{r}(u)\ \underline{r}'(u) = x(u)\frac{dx}{du} + y(u)\frac{dy}{du} = 0$$

This is a third degree equation in u. Let us examine the first quadratic segment. Let us take the scalar coefficients of equation (2.8) to express h(u).

$$
\begin{aligned}
(4.15) \qquad h(u) = &(a_{2x}u^2 + a_{1x}u + a_{0x})\ (2a_{2x}u + a_{1x}) + \\
&+ (a_{2y}u^2 + a_{1y}u + a_{0y})\ (2a_{2y}u + a_{1y}) = 0.
\end{aligned}
$$

$y(u)=0$ and $x'(u)=0$ at $u=0$, thus we obtain $a_{0y}=a_{1x}=0$. Taking (i), (ii), and (v),(vi), it can be seen that $u=-0.5$ and $u=0$ are both roots of $h(u)$, so after dividing by these terms using Horner's method, we obtain:

$$u^* = 0.5 - 3a_{2y}a_{1y}/2(a_{2x}^2 + a_{2y}^2) \tag{4.16}$$

The missing unknown coefficients can also be expressed using (2.12):

$$\begin{aligned} a_{2x} &= a_{1x} - x(-0.5) = - k \sin\varphi, \\ a_{1y} &= 4 \sin\varphi - k \cos\varphi = k, \\ a_{2y} &= a_{1y} - y(-0.5) = k\,(1-\cos\varphi). \end{aligned} \tag{4.17}$$

The above equations can be substituted into (4.15), resulting in $u^* = -0.25$. Therefore we ascertained that independently on the angle of the circular arc, the maximum deviation of the dq-segment from the mathematical arc is at $u=-0.25$ and its symmetric counterpoint $u=0.25$. (It must be noted that in the case of cubics, the value of the critical u^* depends on φ .)

The deviation as a function of φ can be easily calculated:

$$\begin{aligned} x(-0.25) &= a_{2x}/16 + a_{0x} = \cos\varphi/4 + 0.75, \\ y(-0.25) &= a_{2y}/16 - a_{1y}//4 = \\ &= (1 - \cos\varphi)^4/\sin^2\varphi - (1 - \cos\varphi)/\sin\varphi. \end{aligned} \tag{4.18}$$

Some typical values are given in Table 1, to illustrate the rate of convergence as $\varphi \to 0$.

2φ	120°	90°	60°	30°	15°	7.5°
$\lvert\Delta R_{max}\rvert$	4.8e-2	9.2e-3	2.9e-3	1.8e-4	1.2e-5	7.1e-7

Table 1.

The question of how double-quadratics approximate conics can also be posed. It is well-known that rational quadratic functions precisely represent conics - a simple proof can be found in [32] pp. 138. - 144. When the denominator of the rational function equals 1, a parabola is defined. This leads us to the known fact that parametric quadratics are equivalent to parabolas, in our case, double-quadratics are equivalent to two parabolic pieces. Therefore, parabolas can be represented precisely, ellipses and hyperbolas only approximately. Similar qualitative investigations can be carried out as we did for circular arcs in the previous paragraphs.

5. The equation of a double-quadratic patch

The equation of a double-quadratic patch can easily be formulated, once the double-quadratic blending functions are given (see 2.13). Analogously to bicubic patches [32], bivariate blending is used by two parametric variables u and v. Beside the corner points and the tangent vectors there, the so-called twist vectors must be added (Fig. 5.1). The latter ones basically determine the interior of the patch, by defining the changing rate of one parametric variable relating to the other one at the corner points. (To properly determine the twist vectors is a difficult task for the user, that is why most design systems set them automatically based on the surrounding tangent vectors or they are calculated by the used composite interpolating/approximating algorithms.)

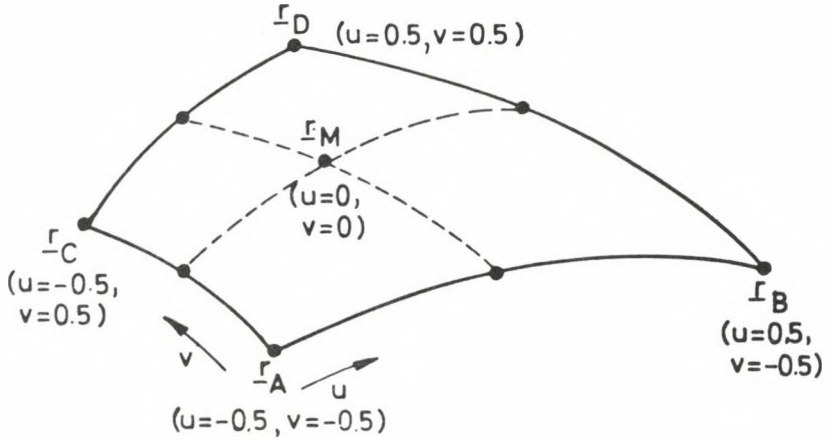


Figure 5.1.

The equation of the patch can be given in the following form:

$$\underline{r}(u,v) = \underline{u}\ \underline{DQ}_u\ \underline{R}\ \underline{DQ}_v^T\ \underline{v}^T \tag{5.1}$$

where:

$$\underline{u} = [\ 1\ u\ u^2\ ], \qquad \underline{v} = [\ 1\ v\ v^2\ ],$$

$\underline{DQ}_u$ and $\underline{DQ}_v$ are equivalent to the DQ matrix in (2.15),

$$\underline{R} = \begin{bmatrix} \underline{r}_A & \underline{r}_C & \underline{r}_{vA} & \underline{r}_{vC} \\ \underline{r}_B & \underline{r}_D & \underline{r}_{vB} & \underline{r}_{vD} \\ \underline{r}_{uA} & \underline{r}_{uB} & \underline{r}_{uvA} & \underline{r}_{uvC} \\ \underline{r}_{uB} & \underline{r}_{uD} & \underline{r}_{uvB} & \underline{r}_{uvD} \end{bmatrix}$$

The parametric area which is mapped to the 3D space is $-0.5 <= u,v <= 0.5$. The vectors in the first two rows and columns, determine the four boundary curves of the patch, while the twists indirectly indicate the change of the cross-tangent vectors along the boundaries. It can be seen from equation (5.1) that in fact a double-quadratic patch is composed of four biquadratic patches, depending on whether ($u<=0, v<=0$) ; ($u<=0, v>0$); ($u>0$, $v<=0$); ($u>0, v>0$). As expected, due to the dq blending functions, internally first derivative continuity is ensured. The consequence of the sign-dependent matrix notation is that the number of the vector quantities needed to define a dq-patch is the

same as that of cubics. The only supplement is that the sign of the u and v parameters must be evaluated, when the blending polynomials are used.

6. The characteristic polyhedron of a double-quadratic patch

The Bezier-like characteristic polyhedron of bicubic and double-quadratic patches can be derived as the consequence of the used bivariate blending technique. The basic advantage of this formulation is - as in case of curves - that it provides a convenient tool for controlling the shape of the patch. Instead of using tangent vectors and twist vectors, the patch can be adjusted by means of the control points of the polyhedron.

The analogy between the cubic Bezier patch and the double-quadratic one can be drawn easily, as it was done in Section 3. (The tangent vectors are multiplied by a weight 1/4 instead of 1/3, the twist vectors by 1/16 instead of 1/9.)

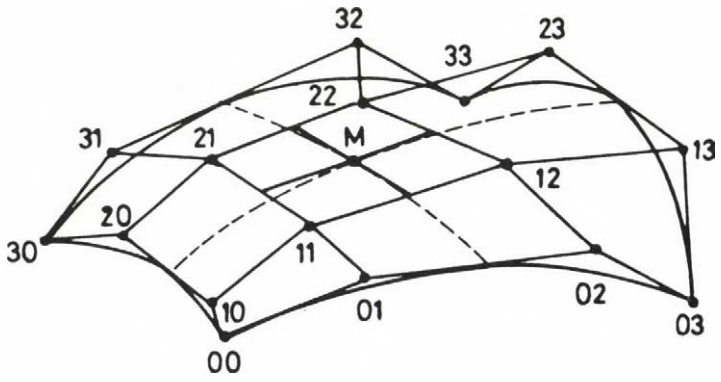


Figure 6.1.

$$
\begin{aligned}
r_{00} &= r_A \\
r_{01} &= r_A + \dot{r}_{uA} / 4 \\
r_{10} &= r_A + \dot{r}_{vA} / 4 \\
r_{11} &= r_A + \dot{r}_{uA} / 4 + \dot{r}_{vA} / 4 + \ddot{r}_{uvA} / 16
\end{aligned}
\qquad (6.1)
$$

(6.1 cont.)

$$\underline{r}_{03} = \underline{r}_B$$
$$\underline{r}_{02} = \underline{r}_B - \dot{\underline{r}}_{uB} / 4$$
$$\underline{r}_{31} = \underline{r}_B + \dot{\underline{r}}_{vB} / 4$$
$$\underline{r}_{12} = \underline{r}_B - \dot{\underline{r}}_{uB} / 4 + \dot{\underline{r}}_{vB} / 4 - \ddot{\underline{r}}_{uvB} / 16$$

$$\underline{r}_{30} = \underline{r}_C$$
$$\underline{r}_{31} = \underline{r}_C + \dot{\underline{r}}_{uC} / 4$$
$$\underline{r}_{20} = \underline{r}_C - \dot{\underline{r}}_{vC} / 4$$
$$\underline{r}_{21} = \underline{r}_C + \dot{\underline{r}}_{uC} / 4 - \dot{\underline{r}}_{vC} / 4 - \ddot{\underline{r}}_{uvC} / 16$$

$$\underline{r}_{33} = \underline{r}_D$$
$$\underline{r}_{32} = \underline{r}_D - \dot{\underline{r}}_{uD} / 4$$
$$\underline{r}_{23} = \underline{r}_D - \dot{\underline{r}}_{vD} / 4$$
$$\underline{r}_{33} = \underline{r}_D - \dot{\underline{r}}_{uD} / 4 - \dot{\underline{r}}_{vD} / 4 - \ddot{\underline{r}}_{uvD} / 16$$

It is also true that the double-quadratic polyhedron gives a more closely related description of the patch than that of bicubics. This follows from the property, that the boundary curve midpoints lie on the polyhedron edges, i.e. at $0.5(\underline{r}_{01}+\underline{r}_{02})$, $0.5(\underline{r}_{31}+\underline{r}_{32})$, $0.5(\underline{r}_{10}+\underline{r}_{20})$ and $0.5(\underline{r}_{13}+\underline{r}_{23})$. This was expected from (3.4). It holds for the midpoint of the patch, that it is equivalent to the midpoint of the middle quadrangle of the polyhedron. Moreover, the cross, which connects the midpoints of the facing edges of the middle quadrangle is tangential to the surface patch at the midpoint, i.e. at the u=0, v=0 values.

$$
\begin{aligned}
\underline{r}(0,0) &= 0.25(\underline{r}_{11} + \underline{r}_{12} + \underline{r}_{21} + \underline{r}_{22}) \\
\dot{\underline{r}}_u(0,0) &= 0.5[-0.5(\underline{r}_{11} + \underline{r}_{21}) + 0.5(\underline{r}_{21} + \underline{r}_{22})] \\
\dot{\underline{r}}_v(0,0) &= 0.5[-0.5(\underline{r}_{11} + \underline{r}_{12}) + 0.5(\underline{r}_{21} + \underline{r}_{22})]
\end{aligned} \tag{6.2}
$$

The above statements can be easily proved by substituting u=0,v=0 into the (5.1) equation and by differentiating by u and v, where needed. Bicubic and double-quadratic patches are very similar to each other based on the arguments in Section 2. The midpoint of a cubic patch is equivalent to the midpoint of its double-quadratic counterpart. In both cases, with given positional and tangential constraints:

$$
\begin{aligned}
\underline{r}(0,0) = {} & (\underline{r}_A + \underline{r}_B + \underline{r}_C + \underline{r}_D)/4 + \\
& (\dot{\underline{r}}_{uA} - \dot{\underline{r}}_{uB} + \dot{\underline{r}}_{uC} - \dot{\underline{r}}_{uD})/16 + \\
& (\dot{\underline{r}}_{vA} + \dot{\underline{r}}_{vB} - \dot{\underline{r}}_{vC} - \dot{\underline{r}}_{vD})/16 + \\
& (\ddot{\underline{r}}_{uvA} - \ddot{\underline{r}}_{uvB} - \ddot{\underline{r}}_{uvC} + \ddot{\underline{r}}_{uvD})/64.
\end{aligned} \tag{6.3}
$$

Provided that the defining tangent magnitudes are not too large, the bicubic and the dq-surfaces result in very similar shapes. The reader should distinguish between them in Fig. 6.2, however, the smaller characteristic polyhedron shows which one is the dq-patch.

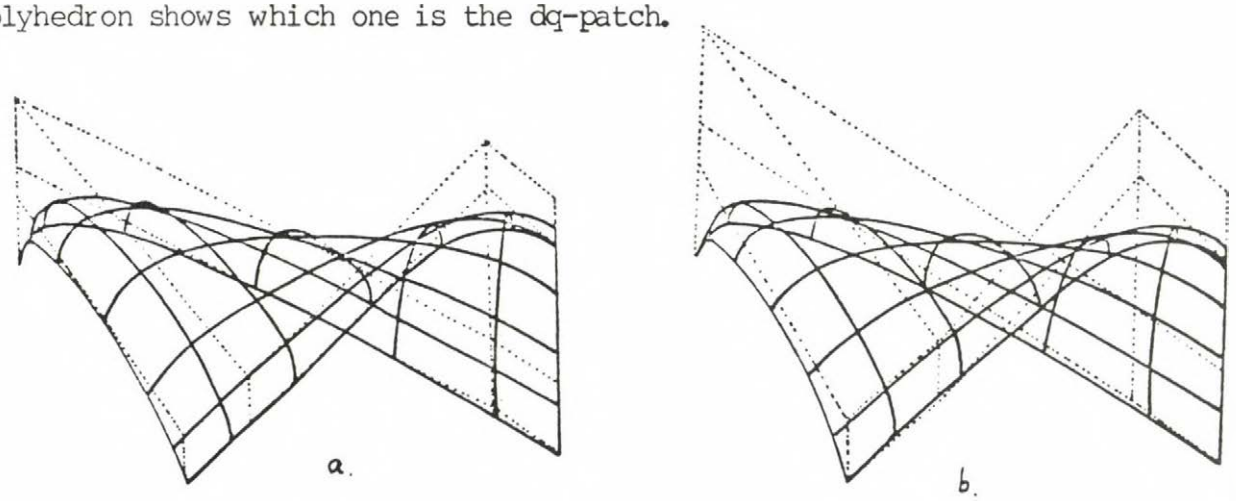


Figure 6.2.

7. Minimum - maximum values of double-quadratic curves and surfaces

It is often necessary to determine the extreme values of a double-quadratic segment. Since it is given in parametric form, this can be done separately for each principal direction. First let us see the z-coordinate of a quadratic segment defined in the form below:

$$z(u) = a\,u^2 + b\,u + c\,, \quad 0 <= u <= 0.5, \quad a \neq 0. \tag{7.1}$$

The extreme values can be either in the endpoints, i.e. at u=0 or u=0.5 or in the point where $dz/du = 0$, as shown in Fig. 7.1. In the latter case

$$u_0 = -b/2a \quad \text{and} \quad z(u_0) = -b^2/4a\,. \tag{7.2}$$

u_0 is the extreme value of the segment if $0 <= u_0 <= 0.5$.

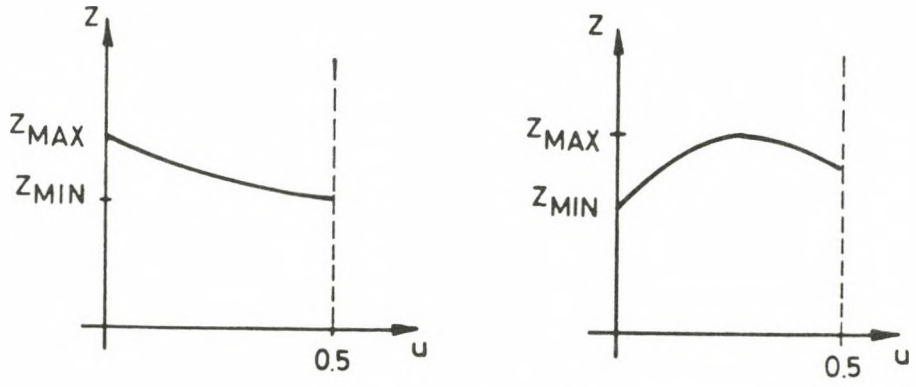


Figure 7.1

For a double-quadratic segment the z-extreme values can be obtained by taking the smallest of the minimum and the largest of the maximum values of the quadratic pieces. The same holds for composite double-quadratic curves as well. Similar procedure can be applied for the x and y coordinates.

For (double-)quadratic patches an analogous algorithm can be used. Let us examine the z coordinates again, given in the following form:

$$(7.3) \qquad z(u,v) = a(v)\ u^2 + b(v)\ u + c(v), \qquad 0 <= u,v <= 0.5$$

The extreme values can be either on the u=0 or u=0.5 boundaries of the patch or on the curve, which contains the extreme values of the individual v constant parameter lines. Fortunately, these extremes can be expressed in the case of quadratics as a function of v. (It is assumed, that $a(v) \neq 0$, for all v. We can also ignore the discrete v values, where a(v)=0, since this means that we have a linear segment, for which the u constant boundaries matters.)

Similarly to the curve segments, for arbitrary v constant curve the extreme value can be expressed as:

$$(7.4) \qquad u_0(v) = -b(v)/2a(v), \text{ and } z_0(v) = -b^2(v)/4a(v)\ .$$

The extremum value of this $z_0(v)$ function is at the points, where:
$\partial z_0/\partial v = 0$, i.e.:

$$(7.5) \qquad b(v)\cdot[b(v)\ a'(v) - 2\ b'(v)\ a(v)\ ] = 0.$$

This fifth degree equation falls apart to a second and a third degree one - both solvable directly. If for any v_0 solution $0 <= v_0 <= 0.5$ holds, it must be also checked whether $u_0(v_0) = -b(v_0)/2\ a(v_0)$ lies in the given parametric interval as well. If the answer is yes, the (u_0,v_0) pair goes into the set of extreme values, from which the smallest and the largest values give the desired z-coordinates. In case of double-quadratics the set of the extreme values include those of all u-constant boundaries and all four quadrants.

If necessary, the same procedure can be applied for composite surfaces made up of dq-patches. It must be noted, that this method cannot be applied for bicubic or higher order patches, since the "extremum" curve cannot be expressed explicitly.

8. Conclusion

In the previous sections the basic equations of double-quadratic curve-segments and surface-patches were presented. It was proved that dq-s are very close relatives of cubics and retain the benefical properties of them. The degree of design freedom of dq-s is sufficient to create complex engineering shapes as in case of cubics. Moreover, dq-s are particularly beneficial, when the computational evaluation of free-form shapes is needed.

Dq-s are blended by second degree polynomials, furthermore, quadratic curve segments and the constant parameter lines of biquadratic patches represent planar curves. Due to these facts, the geometric interrogations and intersections, which are fundamental in computer aided geometric modelling can be performed by non-iterative methods or apparently 3D operations can be reduced to planar ones. In this chapter, only some illustrations for the simplicity of dq-s were presented. A detailed volumetric modelling exposition of the dq-interrogation problem can be found in the following chapters and in the article of Pratt and the author [55].

Chapter III

GEOMETRIC INTERROGATIONS FOR DOUBLE-QUADRATIC CURVES AND DOUBLE-QUADRATIC SURFACES

1. INTRODUCTION

Geometric interrogations play a crucial role in computer aided geometric modelling. In a general sense every question related to the geometry of an object or a group of objects can be considered as a geometric interrogation. Typical examples would be deciding whether two objects interfere with each other or calculating the centre of momentum of an object. Practically all application programs interrogate in some way the computerised model of the object involved. Frequently new data structures are generated such as a finite-element mesh or an NC program using complex interrogations. This is the main advantage of computer representation, that once they are created, a huge amount of information can be generated automatically from them.

The range of possible geometric interrogations can vary greatly depending on the data-structures and applications involved. Even limiting our scope of interest only to the actual building of computer models of solid objects, a complicated hierarchy of interrogations can be formed. However, the lowest level of this hierarchy can be quite well formulated. This is particularly the case for boundary representation modellers, where the topological and geometrical information concerning the boundary of the object is stored, that is the links between the vertices, edges and faces with the equations of curves and surfaces on which the edges and faces lie.

As it was discussed previously in Chapter I., BUILD is a classical example of a boundary modeller. The designers of BUILD paid special attention to the geometric interrogation problem. At the very beginning in 1974, a low level interface called INTJ was defined for examining the spatial relationship of arbitrary two geometric elements (termed geometricons in BUILD). This also

includes the intersection of two elements, which may result in intersection points or intersection curves. The whole modeller was built up in such a way that once geometry was involved, INTJ was called independently of the actual type of the curves and surfaces. The set-operations, local operations, drawing-routines, etc., all use this general interface to determine any new geometric information. That is why new curve and surface types can be easily introduced into BUILD, without touching the remaining parts.

The problems and attempts to introduce free-form geometry into solid modellers are reviewed in another paper [43]. Here only one point is quoted. Design flexibility and simple, robust interrogations are contradictory conditions. Examining simple parametric patches it turned out that though their interrogations are relatively simple, they do not have enough freedom for creating practical engineering objects. Using complicated mathematical equations like high-degree polynomials or rational functions, the desired design flexibility can be achieved, but the interrogations and intersections become very tedious, mostly soluble only by expensive numerical methods.

As turned out from the previous chapter, the double-quadratic curves and surfaces represent one reasonable compromise between complexity and simplicity. The "double" structure increases the design freedom, but at the same time, all geometric interrogations, which are based on parametric quadratic equations, can be treated with relatively low computational expenses, mostly with direct methods. This chapter is devoted to prove this latter statement.

The BUILD INTJ concept meant that a relatively small curve and surface design package and only the dq-curve and dq-surface interrogation procedures had to be implemented to be able to create and combine objects bounded by free-form geometry. In this chapter only the two-operand dq-interrogations are discussed. (Trivial questions such as given a (u,v) parameter pair, what is the surface normal, will not be described here.)

First, the most important theoretical methods in the literature for surface interrogations, their evaluation and comparison will be described. After introducing the basic equations of all geometricons, the global test concept and all necessary dq-interrogations will be presented. In most cases, the basic concept only will be described, due to space limitations and the assumption that the reader is more or less familiar with this field. Mathematical precision, in the sense that all special cases should be discussed in detail would simply multiply the length of this chapter without adding too much. The reader may easily prove or handle these. However, to compensate for this, many practical computational aspects will be mentioned.

2. NOTES ON VECTOR-SURFACE AND CURVE-SURFACE INTERROGATIONS

Before going into detailed discussion of the dq geometric interrogations, we present some alternative general concepts used in interrogating parametric surfaces.

To give a surface point once the (u,v) values are known is trivial, it can be obtained by substituting into the surface equation. However the inversion problem, that is how to determine the (u,v) parameter values, if we have a 3D point on the surface is more complicated.

(2.1) $$\underline{r}_0 = \underline{r}(u,v)$$

Here we have three nonlinear scalar equations with two unknowns. Selecting two of them, the (u_0,v_0) solution obtained must satisfy the third equation. There are many existing numerical methods for solving such a system of equations. To avoid two-dimensional Newton-Raphson like iterative proccesses for solving the system of equations the problem can be converted to finding the roots of a high-degree polynomial (resultant method) or to solve a set of many linear equations using the method suggested by Sederberg et al [64].

In case of parametric biquadratic surfaces the resultant method will give an 8th degree polinomial for one of the (u,v) parameters, while Sederberg's method leads to a linear system of equations with ten unknowns. Both are reasonable alternatives. Another possibility is eliminating one of the parameters from (2.2), expressing the right side as polynomial coefficients of u.

(2.2) $$\underline{r}_0 = \underline{A}(v) + \underline{B}(v)\, u + \underline{C}(v)\, u^2$$

Assuming that $\underline{B}(v) \times \underline{C}(v) \neq 0$ in the given parametric interval, the equation can be multiplied by this term resulting in a 6th degree equation in v.

(2.3) $$(\underline{A}(v) - \underline{r}_0)\ (\underline{B}(v) \times \underline{C}(v)) = 0$$

Using this algebraic method particular care must be paid to handle all special cases and all potential roots.

Recursive subdivision methods can also be applied to the inversion problem [17],[30],[21]. Using the convex hull property of most parametric patches, the subdivision goes until the given point lies within the convex hull of a subpatch, whose size is smaller than the given tolerance. Subpatches, where the convex hull test fails, are discarded and the process converges relatively quickly.

Unfortunately, in many cases the inversion problem arises in a different way. Since the given point may be a result of previous calculations, it often cannot be guaranteed that the point lies on the surface where we expect, only very close to it. This means that mathematically the equation (2.1) will not have any (u,v) solution. Instead of that we have to look for the nearest point of the surface, that is

(2.4) $$| \underline{r}(u,v) - \underline{r}_0 | = \min.$$

which destroys the previous techniques or at least makes them more complicated. The degree of the nonlinear equations may be doubled. The recursive subdivision must proceed more carefully not to miss the point, etc. Substitution into an implicitised equation also may give false result in these cases.

In Section 12 we propose an alternative method for dq-patches, which is based on the property that every parameter line of a biquadratic patch determines a plane. Apart from the degenerate cases, where only iterative methods can help, this means the solution of a six degree polynomial.

Considering straight line - parametric patch intersection there is a strong similarity to the inversion problem.

(2.5) $$\underline{q}(t) = \underline{r}(u,v)$$

The above vector equation means three scalar equations with three unknowns. Making a special transformation on both sides, $\underline{q}(t)$ can be written as

(2.6) $$\underline{q}(t) = \{ x = x_0,\ y = y_0,\ z = z_0 + z_1\ t \}$$

This always can be done. Taking the x,y equations we obtain again nonlinear equations with two unknowns u and v, leading us to the inversion problem and the same techniques. After getting the (u,v)-s, the z-equation gives the desired t value.

For quadratic patches we can apply again the previous algebraic elimination method.

(2.7) $$\underline{q}_0 + \underline{q}_1\ t = \underline{A}(v) + \underline{B}(v)\ u + \underline{C}(v)\ u^2$$

Assuming that $\underline{q}_1 \times \underline{A}(v) \neq 0$ and $\underline{q}_1 \times \underline{C}(v) \neq 0$ the equation can be multiplied by these terms resulting

(2.8) $$(\underline{A}(v) - \underline{q}_0)\ (\underline{q}_1 \times \underline{C}(v)) + \underline{B}(v)\ (\underline{q}_1 \times \underline{C}(v))\ u = 0$$

(2.9) $$\underline{B}(v)\ (\underline{q}_1 \times \underline{A}(v))\ u + \underline{C}(v)\ (\underline{q}_1 \times \underline{A}(v))\ u^2 = 0 \quad .$$

Again, the many special cases must be treated carefully. The general case as is expected leads to an 8th degree polynomial in v.

Going further in complexity of parametric curves, the next would be the parametric quadratic segment equivalent to a parabola, or bit more generally the rational quadratic segment which covers all conic curves. In these cases the

(2.10) $$\underline{q}(t) = \underline{r}(u,v)$$

equation becomes a difficult nonlinear system of equations in three unknowns

t,u,v. The best alternatives seem to be here implicitization of the patch or recursive subdivision.

In case of biquadratic patches the algebraic surface will be described by an 8th degree equation in x,y,z. It will have (n+1)(n+2)(n+3)/6 = 165 terms. Substituting $\underline{q}(t)$ into the implicit equation gives only a second degree equation for t, but the calculation of the coefficients is computationally very expensive.

Using similar algebraic method as previously for eliminating t and u we start from the equation below.

(2.11) $$(\underline{q}_0 + \underline{q}_1 t + \underline{q}_2 t^2)/(w_0+w_1 t+w_2 t^2) = \underline{A}(v) + \underline{B}(v)u + \underline{C}(v)u^2$$

Here we obtain a 20th degree equation in v at very high computational cost. (The author's intuition is that it should lead to a 16th degree one, however that way is not known.) Otherwise to solve high degree polynomials some quite reliable iterative methods are known in the literature.

The recursive subdivision methods work quite smoothly in this case as well, using the property, that if a subpatch intersects a curve, its convex hull also must be intersected. As in case of all surface subdivisions this means two dimensional iterations until the size of the convex hull falls below the given tolerance or until the subpatch becomes practically planar, which makes the intersection simple.

In Section 13 we propose an alternative method for parametric curve - biquadratic intersection, based again on the planar parameter line property. In the general case, interval-halving iteration in respect to only one of the u,v parameter values is performed. At each step only vector and scalar products

are calculated and second degree polynomials solved, these are relatively cheap operations. The algorithm is valid for arbitrary higher degree non-self-intersecting parametric curves as well.

3. NOTES ON SURFACE - SURFACE INTERSECTIONS

Having two surfaces S_1 and S_2 one would like a curve with the equation $\underline{r} = \underline{r}(t)$, for which $\underline{r}$ lies on both S_1 and S_2 for all t. The problem is that apart from the simplest plane-plane and plane-quadric intersections no explicit parametric equation can be found in most cases including the parametric-parametric case. Using the implicitization technique, referred to in a recent paper by Sederberg et al.[64], the degree of these intersection curves can be determined, however no method is known how to calculate the coefficients of these extremely high degree polynomials. (The degree of the intersection curve of two bicubic patches is for example 324.) That is why, generally instead of this, an approximating curve is determined, based on successions of points. This is the case even at quadric-quadric intersections, thoroughly elaborated in the work of Levin and of Solomon [47],[65].

In general, the process consists of three phases. Firstly, a set of points has to be calculated, for which $\underline{r}_i$ lies on both S_1 and S_2 or at least the distances of $\underline{r}_i$ from S_1 and S_2 are within a given tolerance. In the second phase, these discrete points need to be sorted into an appropriate order, supposing that this order does not obviously come from the previous step. Finally, a good (maybe piecewise linear) interpolating curve must be fitted through the consecutive points.

Basically we will define five different approaches to perform surface-surface intersections, based on Sabin's classification in his review on contouring [62]. However, the special aspects of curve-fitting are not discussed there. In this paper, the intersection of plane-parametric, quadric-parametric and parametric-parametric surfaces are tackled, thus the first "direct contouring" approach, which uses directly curve equations, is excluded.

The second approach is the "conversion to a regular grid" - or in other words - the use of interpolating surfaces. This means that instead of the original surface a low-degree interpolation through its vertices will be used and small sub-patches will be intersected with each other. In case of small facets this gives a reasonable approximation of the intersection curve made of small straight line segments.

Recursive subdivision methods[17],[30] are also based on this approach, basically using the convex hull property of most parametric patches. The subdivision goes until the little subpatch becomes almost planar or very small with respect to a given tolerance. These methods are algorithmically simple, however in the parametric-parametric case they may become very expensive. Using recursive subdivision there is no need for sorting, since the intersection curves are directly generated proceeding on the interpolated surfaces. Due to the above reasons and the fact that the method is independent on the parametric patch type used recursive subdivision becomes more and more popular as the references in the literature show [21], [40], [77], [28], [22].

The third approach is the "parametric grid" method, that is the constant u and v lines of one surface are intersected with the other surface. Considering the plane-parametric and quadric-parametric cases S_1 is given in f(x,y,z) = 0 form, while S_2 is defined as $\underline{r} = \underline{r}(u,v,)$. The points of the intersection curve must satisfy the f(x(u,v), y(u,v), z(u,v)) = 0 equation, that is, with given u_0 a high degree polynomial equation must be solved to obtain the appropriate (u_0,v_0) pair, an intersection point in the parametric plane. In most cases, this leads to numerical solutions. When S_1 is also defined in parametric form, constant parameter lines must be intersected with the other surface, thus a generally nonlinear system of equations must be solved to get one point [19].

This approach generates sequences of (u,v) pairs. Unless the density of the parameter lines is extremely large, when squared distance selection can be used, some sorting method has to be applied to join the consecutive points. The sorting can be performed either in the parametric plane or in 3D space, obtaining the curve points after substituting (u_0,v_0) into the S_2 equation.

The fourth approach is "contour following", that is, going along each contour point by point, following approximately the track of the intersection curve. Supposing an intersection point has been found, the next one is searched for, close to the tangential direction at the starting point. This can be calculated by the vector product of the normal vectors of S_1 and S_2 or by linear extrapolation of the previous points. Either some version of the Newton-Raphson method must be used to obtain points on both surfaces within a tolerance or some additional criterion must be applied to ensure that the distances from the surfaces do not grow beyond a reasonable limit. Alternatives can be found in [60], [48] and [26]. The most important feature of this

method is its simplicity, no sorting is needed, however its convergence and the problem of finding all starting points for the contours may cause difficulties.

As to the fifth basic approach, the literature contains several "least square minimisation algorithms", i.e. the square distance between the variable points on each surface is minimized. (One valuable example is Butterfield's thesis [16].) In this case inaccuracy and large computational cost when there are many datapoints are the main drawbacks.

After this brief review, some special considerations on surface-surface intersections in an up-to-date volumetric modeller will be presented. The most important is that the entire range of these geometric interrogations must be supplied for all types of surfaces.

The surface-surface intersection curves may become part of the model. The curves, on which edges lie, must generally satisfy tangential continuity. Consequently, piecewise linear interpolation of curves, which is acceptable in most graphic and NC machining applications, are not suitable for a solid modeller. Smooth approximation of the intersection curves must be generated. The approximating curve must be as good as possible in the sense that it must run as close to both surfaces as possible. That is, not only the points of the curve-fitting algorithm need to lie on both surfaces, but further local constraints (tangency,curvature), using the geometry of the surfaces, must be satisfied to guarantee the best approximation. The fitted curve must have a non-uniform characteristic, since the distances between the consecutive intersection points are very unlikely to be evenly spaced.

Generating a very fine grid or choosing a very small step length would result in too many data points,that is too many curve-segments along the intersection curve. This would mean that the intersection procedure is not only very time-consuming, but the storage needed for the data of an average object with free-form faces would grow extremely large.

One way of overcoming this is to use a basically coarse grid and a more sophisticated sorting technique with the above mentioned curve-fitting method. Another way of reducing the number of curve-segments is to introduce some adaptivity into the curve-fitting procedure and concatenate the consecutive curve-segments, if it is possible without exceeding the given tolerances. In general direct methods are preferred rather than numerical ones. In most cases,these eliminate the convergence problems and work with higher efficiency than iterative algorithms.

A special surface interrogation method, based on the above considerations, which is particularly efficient in case of double-quadratics will be described in Section 14.

4. TWO-OPERAND GEOMETRIC INTERROGATIONS

Two elements are always involved in INTJ-type geometric interrogations: their relationship or their common subset are to be determined. The two operands can be vectors, curves and surfaces, these constitute the geometrical data-structure of a boundary representation.

Here we are concerned with introducing free-form geometry into a conventional boundary modeller, thus interrogations where one of the operands is a dq-curve or a dq-surface will be described. The other operand can be a vector, a

straight line, a conic segment, a plane, a quadric surface and another dq-curve or dq-surface. Straight lines, conic segments, dq-curves and dq-surfaces are given in parametric form, while planes and quadrics are defined by implicit equations (see next section).

The result of the interrogations can be one of the following types:

1. empty set - there is no common part of the two geometric elements

2. coincidence - the two geometric elements coincide fully or partially, for example a dq-curve lies in the plane involved or some segments of the curve lie there

3. curvepoint - the result of the interrogation in respect to a curve-type geometric element is a parameter value u, which determines a unique point on the curve

4. surfacepoint - the result of the interrogation in respect to a surface-type geometric element is a (u,v) parameter pair, which determines a unique point on the surface

5. intersection curve - in case of surface-surface interrogations (see Section 14) the result can be one or more continuous curves. In simple cases the curve equation can be expressed explicitly, but with free-form geometry only a sequence of discrete intersection points can be determined.In this chapterthe method of finding the individual surface-points and the "sorting" of these points will be described, while the problem of fitting the possible best approximating curve to these points is discussed in the next chapter.

The actual output of the geometric interrogations is a list of the above elements. Quite frequently, not only a list of curvepoints or surfacepoints

are returned, but also a list of so-called pointpairs, where the values in respec to both operands are supplied.

The table below summarizes the possible result types of interrogations between double-quadratics and other geometric elements.

	VECTOR	STRAIGHT + CONIC	DQ CURVE	PLANE + QUADRIC	DQ SURFACE
DQ CURVE	1 or 3	1,2 or (3+3)	1,2 or (3+3)	1,2 or 3	1,2 or (3+4)
DQ SURFACE	1 or 4	1,2 or (3+4)	1,2 or (3+4)	1,2 or (a sequence of 4 → 5)	1,2 or (a sequence of 4+4 → 5)

TABLE 1.

5. SIMPLE GEOMETRIC ELEMENTS

5.1. VECTOR

(5.1.1) $$\underline{v} = (x,y,z)$$

A vector is defined by its three Cartesian coordinates or by its homogeneous coordinates

(5.1.2) $$\underline{v} = (x,y,z,w)$$

where return to the Cartesian space must be done by dividing with the term w, i.e.

(5.1.3) $$\underline{v} = (x/w , y/w , z/w)$$

5.2. STRAIGHT LINE

$$(5.2.1) \qquad \underline{r}(t) = [X(t)\ Y(t)\ Z(t)\ W(t)] = [\ t\ 1\ ] \begin{bmatrix} v_x & v_y & v_z & 0 \\ p_x & p_y & p_z & 1 \end{bmatrix}$$

A straight line is given in parametric form, the first row contains the direction vector, the second row is a point on the line. Both are given with their homogeneous coordinates for consistency with conics. A straight line can be bounded by restricting the parameter value to run from t_1 to t_2.

5.3. CONIC SEGMENT

$$(5.3.1) \qquad \underline{r}(t) = [\ X(t)\ Y(t)\ Z(t)\ W(t)\ ] = [t^2\ t\ 1] \begin{bmatrix} p_x & p_y & p_z & p_w \\ q_x & q_y & q_z & q_w \\ r_x & r_y & r_z & r_w \end{bmatrix}$$

A conic segment is also given in parametric form with homogeneous vectors. The type of the conic is determined by the p_w, q_w, r_w variables. It can be easily proved that depending on whether $4r_w p_w$ is smaller, equal or greater than $q_w q_w$ the curve is an ellipse, a parabola or a hyperbola, respectively [32].

In case of parabolas and ellipses the t parameter may run from minus to plus infinity, however in case of hyperbolas only a (t_{h1}, t_{h2}) interval is valid, where t_{h1} and t_{h2} are the roots of the $p_w t^2 + q_w t + r_w = 0$ equation. For most dq geometric interrogations we use a bounded parametric interval for parabolas and ellipses and we also tighten the interval of the hyperbolas. (One way of doing this is to form an enclosing sphere around the other geometric element and find the conic - sphere intersection points (see Section 11)).

5.4. PLANE

$$(5.4.1) \qquad [\ X\ Y\ Z\ 1\ ] \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = 0$$

A plane is described in the form of an implicit equation. (A,B,C) is the surface normal, D is the signed distance of the plane from the origin.

5.5. QUADRIC

$$(5.5.1) \qquad [\ X\ Y\ Z\ 1] \begin{bmatrix} a & e & f & g \\ e & b & h & i \\ f & h & c & j \\ g & i & j & d \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = 0$$

A general quadric surface is defined in the form of the above implicit equation. The type of the quadric is determined by the 10 quantities of the matrix (see [47] or [65]). The surface normal at a given (X,Y,Z) point can be calculated by

$$(5.5.2) \qquad \underline{n} = [\ X\ Y\ Z\ 1\ ] \begin{bmatrix} a & e & f & g \\ e & b & h & i \\ f & h & c & j \\ g & i & j & d \end{bmatrix}$$

6. DOUBLE-QUADRATIC CURVES

6.1. Equation

The double-quadratic curves are piecewise parametric curves, the consecutive curve-segments join together with slope-continuity. Dq-curves interpolate n+1 points - $\underline{P}_i$, i = 0,1,...n. An arbitrary curvepoint $\underline{P}$ is defined by its parameter value U, where ENTIER U gives the segment number, FRACTION U gives the local parameter value within the segment. A dq-curve can be closed, assuming that $\underline{P}_0 = \underline{P}_n$ and slope continuity is also satisfied there.

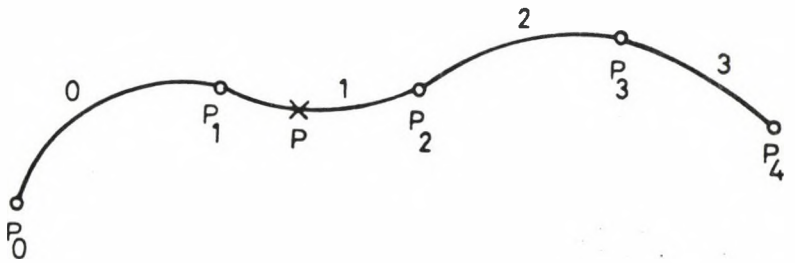


FIG.6.1.

Each individual curve-segment is represented by its endpoints and its tangent vectors. These vector quantities are blended by the double-quadratic blending functions as was written in the previous chapter. Here only the most important equations are repeated for further reference:

$$\underline{r}(u) = [1 \quad u \quad u^2] \quad [Q_u] \quad \begin{bmatrix} \underline{r}_A \\ \underline{r}_B \\ \dot{\underline{r}}_A \\ \dot{\underline{r}}_B \end{bmatrix} \qquad (6.1.1)$$

$$u = \{U\} - 0.5$$

where $-0.5 <= u <= 0.5$

and

$$[\ Q_u\ ] = \begin{bmatrix} 0.5 & 0.5 & 0.125 & -0.125 \\ -2 & 2 & -0.5 & -0.5 \\ -2 & 2 & -1.5 & -0.5 \\ 2 & -2 & 0.5 & 1.5 \end{bmatrix}$$

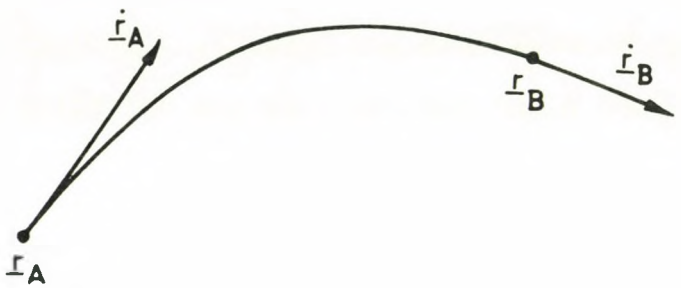


FIG.6.2.

(r_A, r_B represent the beginning and ending position vectors, $\dot{r}_A$, $\dot{r}_B$ the tangent vectors. The u parameter value runs from -0.5 to 0.5. The third or the fourth row of the sign-dependent Q_u matrix will be taken depending on the sign of the u-value.)

Each dq-curve segment is made of two quadratic pieces (parabolas), which can be twisted at the midpoint (u=0).

6.2. Characteristic polygon

A Bezier-type characteristic polygon can be constructed for the double-quadratic curves, preserving the most significant mathematical benefits of that form, such as convex-hull property, simple control for design, straightforward determination of the sign of curvature at the endpoints and the conditions for having loops and inflection etc. (See Chapter II.)

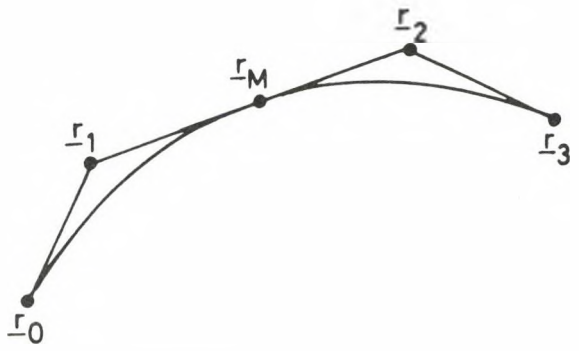


FIG.6.3.

Here the vectors of this polygon are

(6.2.1) $$\underline{r}_0 = \underline{r}_A,\ \underline{r}_1 = \underline{r}_A + 0.25\ \dot{\underline{r}}_A,$$
$$\underline{r}_2 = \underline{r}_B - 0.25\ \dot{\underline{r}}_B,\quad \underline{r}_3 = \underline{r}_B$$

It should be noted, that the midpoint of the dq-curve is identical to the midpoint of the characteristic polygon $\underline{r}_M$, that is

(6.2.2) $$\underline{r}_M = 0.5\ (\underline{r}_1+\underline{r}_2) = 0.5\ (\underline{r}_A+\underline{r}_B) + 0.125\ (\dot{\underline{r}}_A-\dot{\underline{r}}_B)$$

The curve at the midpoint is tangential to the $\underline{r}_2 - \underline{r}_1$ vector.

(6.2.3) $$\dot{\underline{r}}_M = 4\ (\underline{r}_2 - \underline{r}_1) = 2\ (\underline{r}_A - \underline{r}_B) - 0.5\ (\dot{\underline{r}}_A + \dot{\underline{r}}_B)$$

6.3 Vector coefficients

At the geometric interrogations of the dq-curves, the dq-curve segments are decomposed into quadratic pieces. In most cases it is advisible to calculate the vector coefficients of the quadratic piece and then only the parameter value of it must be substituted. The vector equation based on the $(\underline{r}_I, \underline{r}_{II}, \underline{r}_{III})$ vector triple is

$$\underline{r}(uu) = [\ 1\ uu\ uu^2\ ] \begin{bmatrix} \underline{r}_I \\ \underline{r}_{II} \\ 4\ (\underline{r}_{III} - \underline{r}_I) - 2\ \underline{r}_{II} \end{bmatrix} , \tag{6.3.1}$$

$$0 <= uu <= 0.5 .$$

For the first half

$$uu = \{U\} \text{ and } (\underline{r}_I, \underline{r}_{II}, \underline{r}_{III}) = (\underline{r}_A, \dot{\underline{r}}_A, \underline{r}_M) ,$$

for the second half

$$uu = \{U\} - 0.5 \text{ and } (\underline{r}_I, \underline{r}_{II}, \underline{r}_{III}) = (\underline{r}_M, \dot{\underline{r}}_M, \underline{r}_B) .$$

7. DOUBLE-QUADRATIC SURFACES

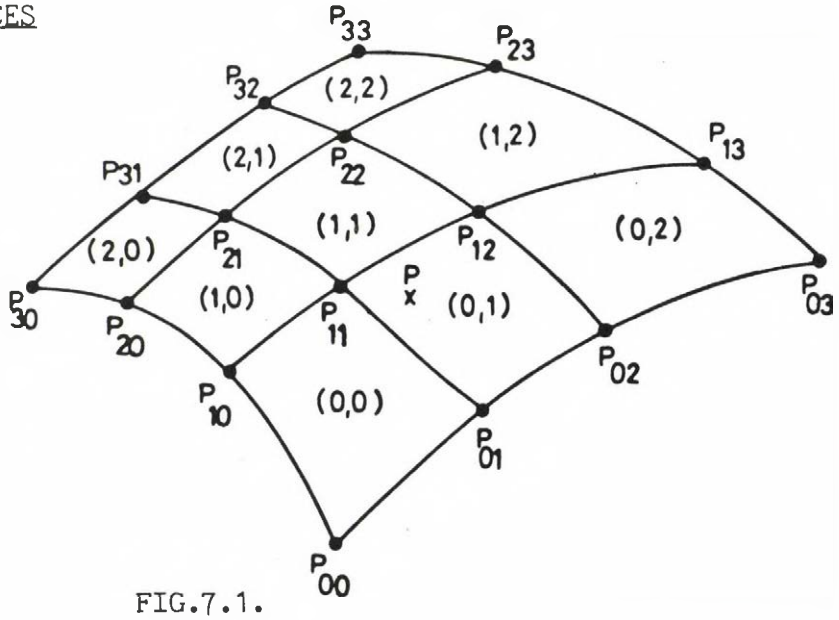


FIG.7.1.

7.1. Equation

The double-quadratic surfaces are made up of piecewise tensorproduct parametric patches, which join each other with tangential continuity. Dq-surfaces interpolate (n+1) x (m+1) points - $\underline{P}_{ij}$, i = 0,...m; j = 0,...n - arranged into a grid structure. An arbitrary surfacepoint is defined by its parameters (U,V), where (ENTIER U,ENTIER V) gives the row and column number of the patch involved, while (FRACTION U,FRACTION V) gives the local parameters within that patch. A dq-surface can be closed, if - by convention - $\underline{P}_{i0} = \underline{P}_{im}$ for all i = 0,...m and C1 continuity is satisfied along these boundaries.

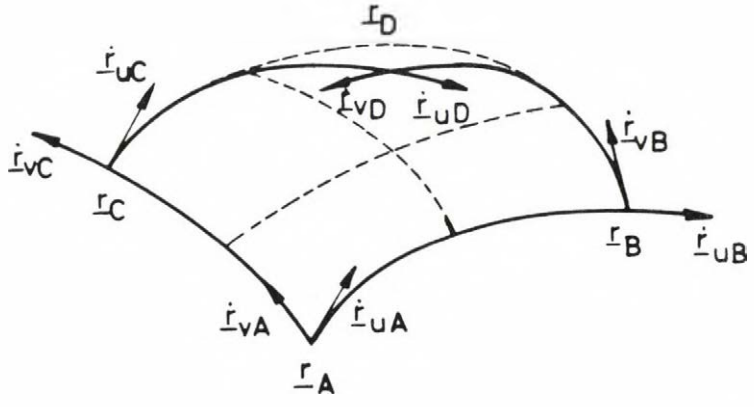


FIG.7.2.

Each individual patch is described by the tensor product form of a Coons' patch analogouosly to the bicubic representation, that is by 4 position vectors, 4 u-tangent vectors, 4 v-tangent vectors and 4 twist vectors; that makes 16 independent vector quantities altogether - 4 at each corner. (The u- and v-tangents relate to the u and v parametric directions, respectively.) As in case of dq-curves, the double-quadratic blending functions are used, given also by means of the sign-dependent dq-matrix.

The matrix equation of a dq-patch is

(7.1.1) $(u = \{U\}-0.5,\ v = \{V\}-0.5)$

$$\underline{r}(u,v) = [\ 1\ u\ u^2\ ]\ [\ Q_u\ ] \begin{bmatrix} \underline{r}_A & \underline{r}_C & \dot{\underline{r}}_{vA} & \dot{\underline{r}}_{vC} \\ \underline{r}_B & \underline{r}_D & \dot{\underline{r}}_{vB} & \dot{\underline{r}}_{vD} \\ \dot{\underline{r}}_{uA} & \dot{\underline{r}}_{uC} & \ddot{\underline{r}}_{uvA} & \ddot{\underline{r}}_{uvC} \\ \dot{\underline{r}}_{uB} & \dot{\underline{r}}_{uD} & \ddot{\underline{r}}_{uvC} & \ddot{\underline{r}}_{uvD} \end{bmatrix} [\ Q_v^T\ ] \begin{bmatrix} 1 \\ v \\ v^2 \end{bmatrix}$$

where $-0.5 <= u,v <= 0.5$ and Q_u is the sign-dependent dq-matrix in respect to u, Q_v^T is its transposed matrix in respect to v.

Internally, each dq-patch is made up of four biquadratic patches, inside the dq-patch C1 continuity is assured.

7.2. Characteristic polyhedron

The analogy between the characteristic polyhedron of dq-patches and the bicubic Bezier polyhedron can be drawn easily as in case of the dq-curves.

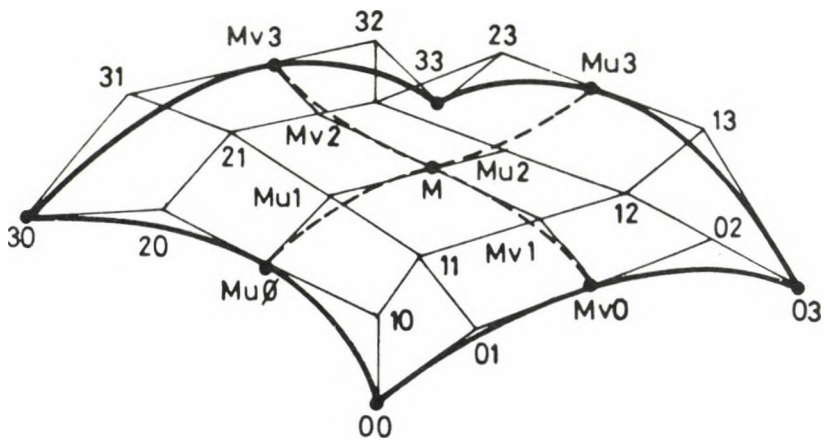


FIG.7.3.

$$\underline{r}_{00} = \underline{r}_A$$

$$\underline{r}_{01} = \underline{r}_A + \dot{\underline{r}}_{uA} / 4$$

$$\underline{r}_{10} = \underline{r}_A + \dot{\underline{r}}_{vA} / 4$$

$$\underline{r}_{11} = \underline{r}_A + \dot{\underline{r}}_{uA} / 4 + \dot{\underline{r}}_{vA} / 4 + \ddot{\underline{r}}_{uvA} / 16$$

(7.2.1)

$$\underline{r}_{03} = \underline{r}_B$$

$$\underline{r}_{02} = \underline{r}_B - \dot{\underline{r}}_{uB} / 4$$

$$\underline{r}_{13} = \underline{r}_B - \dot{\underline{r}}_{vB} / 4$$

$$\underline{r}_{12} = \underline{r}_B - \dot{\underline{r}}_{uB} / 4 + \dot{\underline{r}}_{vB} / 4 - \ddot{\underline{r}}_{uvB} / 16$$

etc.

(7.2.2) $$\underline{r}_{Mui} = (\underline{r}_{i1} + \underline{r}_{i2}) / 2 ,$$

(7.2.3) $$\underline{r}_{Mvi} = (\underline{r}_{1i} + \underline{r}_{2i}) / 2 ; \quad i = 0,1,2,3$$

(7.2.4) $$\underline{r}_M = (\underline{r}_{Mu1} + \underline{r}_{Mu2}) / 2 = (\underline{r}_{Mv1} + \underline{r}_{Mv2}) / 2 =$$
$$= (\underline{r}_{11} + \underline{r}_{12} + \underline{r}_{21} + \underline{r}_{22}) / 4 .$$

The $\underline{r}_M$, $\underline{r}_{Mui}$, $\underline{r}_{Mvi}$ midpoints do not belong strictly to the characteristic polyhedron. They are always internal points of the convex hull. Their role is, that the easiest way to calculate the biquadratic vector coefficients is to use them.

It should also be noted here, that the $\underline{r}_M$ midpoint is identical to the surface midpoint at (u=0,v=0) and the surface is tangential to the plane determined by the $\underline{r}_{Mu2} - \underline{r}_{Mu1}$ and $\underline{r}_{Mv2} - \underline{r}_{Mv1}$ vectors.

7.3. Vector coefficients

For the geometric interrogations of the dq-surfaces, the dq-patches are decomposed into biquadratic pieces. In most cases it is advisable to calculate the vector coefficients of the biquadratic piece and then only the u,v values must be substituted.

The equation is given in the following form

$$(7.3.1) \qquad \underline{r}(uu,vv) = [\ 1\ uu\ uu^2\ ]\ [\ A\ ] \begin{bmatrix} 1 \\ vv \\ vv^2 \end{bmatrix}$$

where $0 <= uu,vv <= 0.5$ and

$$(7.3.2) \qquad [\ A\ ] = \begin{bmatrix} \underline{a}_{00} & \underline{a}_{01} & \underline{a}_{02} \\ \underline{a}_{10} & \underline{a}_{11} & \underline{a}_{12} \\ \underline{a}_{20} & \underline{a}_{21} & \underline{a}_{22} \end{bmatrix}$$

For the first quarter $uu = \{U\}$ and $vv = \{V\}$ and the $\underline{a}_{ij}$ vectorcoefficients will be calculated using the

$$(7.3.3) \qquad [\ R_I\ ] = \begin{bmatrix} r_{00} & r_{01} & r_{Mv0} \\ r_{10} & r_{11} & r_{Mv1} \\ r_{Mu0} & r_{Mu1} & r_M \end{bmatrix}$$

matrix. In a similar way, in the second quarter - $uu = \{U\}-0.5$, $vv = \{V\}$ in the third - $uu = \{U\}$, $vv = \{V\}-0.5$ and in the fourth - $uu = \{U\}- 0.5$, $vv = \{V\}- 0.5$. The R_{II}, R_{III} and R_{IV} matrices can be obtained by selecting the appropriate nine vectors of the generalized characteristic polyhedron.

Having the R_i matrix A can be determined by

$$[\ A\ ] = [\ B\ ]\ [\ R_i\ ]\ [\ B^T\ ]$$

where

$$(7.3.4) \qquad [\ B\ ] = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 4 & 0 \\ 4 & -8 & 4 \end{bmatrix}$$

In practice, due to the simplicity of matrix B (zero elements, constant factors) - the $\underline{a}_{ij}$ vector coefficients are calculated by taking account of similar considerations as in the case of dq-curves (see (6.3.1)) instead of performing matrix multiplications.

8. GLOBAL TESTS

A very important issue for geometric interrogations is to achieve lowest computation times possible. One way of doing this is to perform global tests to decide whether the two geometric elements interfere at all. In most cases this means that the geometrically complex elements are covered by surrounding boxes (or spheres) and first the simple geometric elements with boxes, or boxes with boxes are checked for interference. If there is none, we know that the result is "empty", if there is, we perform the actual interrogating calculations.

In case of dq-curves and dq-surfaces boxes are mostly preferred instead of spheres due to their computational simplicity. Spheres would result in second degree implicit equations and also much bigger coverage, which would reduce the advantage gained by the global tests. It must be kept in mind that these preliminary tests have to be computationally very cheap, otherwise they lead to just the opposite effect one may expect. In case of dq-s xyz-boxes are used, that is the bounding planes are orthogonal to the x,y,z axis. These boxes are characterized by two points lying on a box diagonal, which runs from (xmin,ymin,zmin) to (xmax,ymax,zmax). As it is well-known, the convex hull property for parametric quadratics consequently for double-quadratics as well is satisfied. Creating the characteristic polygons and polyhedrons for the dq-s (see Section 6.2 and 7.2) we get a set of points $\{ \underline{r}_i \}$, which must be

covered by an xyz-box, that is

$$\{ \underline{r}_i \} = \{ x_i, y_i, z_i \}, \quad i = 1,2,....n$$

$$\begin{aligned} &xmin = MIN\ x_i, \quad xmax = MAX\ x_i, \\ &ymin = MIN\ y_i, \quad ymax = MAX\ y_i, \\ &zmin = MIN\ z_i, \quad zmax = MAX\ z_i \qquad i = 1,2,....n \end{aligned} \tag{8.1}$$

Since dq-curves and dq-surfaces are given in piecewise form global tests are generally performed at two levels. The first-level box will cover the whole dq-surface (dq-curve). If the interference-check is positive, we create small boxes around the individual biquadratics (quadratics) and perform tests at this second level. These select the patches (segments) where intersection may possibly occur. In simple cases this second-level check will not be performed on the supposition that it would be more expensive than the actual interrogation.

Table 2. shows the necessary global tests for dq-s with other geometricons.
ϕ - means, that it is not worth to perform a global test for one piece,
$*$ - means, the same test will be made for the box surrounding the single patches. The "quadratic cuts box" case is equivalent to the "straight or conic cuts box" case, with parametric interval [0,0.5].

	VECTOR	STRAIGHT	CONIC	DQ-CURVE	PLANE	QUADRIC	DQ-SURF.
DQ URVE	Point in dqcbox ϕ	Straight cuts dqcbox ϕ	Conic cuts dqcbox ϕ	Dqcbox cuts dqcbox Quadratic cuts dqcbox	Plane cuts dqcbox ϕ	Quadric cuts dqcbox ϕ	Dqcbox cuts dqcbox Quadratic cuts dqcbox
DQ JRFACE	Point in dqsbox $*$	Straight cuts dqsbox $*$	Conic cuts dqsbox $*$	Dqcbox cuts dqsbox Quadratic cuts dqsbox	Plane cuts dqsbox $*$	Quadric cuts dqsbox $*$	Dqsbox cuts dqsbox $*$

Table 2.

Therefore six basic cases remain:

1. point in box
2. straight cuts box
3. conic cuts box
4. plane cuts box
5. quadric cuts box
6. box cuts box

Before going into details, some notations will be introduced. A box B is given by

$$B = (xmin, xmax, ymin, ymax, zmin, zmax) .$$

It has 8 boxpoints, denoted by BP(i), 12 boxedges (bounded straight line segments), denoted by BE(i) and 6 boxplanes, denoted by BPL(i), which all will be used later for different tests.

Due to the computational inaccuracies frequently experienced in geometric modelling, boxes fattened by a little real number or in other words toleranced inequalities will be introduced. Instead of a < b , a > b the %< , %> operators (BUILD) will be used, that is

a %< b means a - littlereal < b

a %> b means a + littlereal > b

8.1 Point in box

The point P = (x,y,z) lies inside the box B if the condition below is satisfied.

xmin %< x AND xmax %> x
AND ymin %< y AND ymax %> y
AND zmin %< z AND zmax %> z

8.2 Straight cuts box

The line segment $\underline{L}(t)$ given in the (5.2.1) parametric form with parametric interval $[t_l, t_h]$ cuts the box B or lies entirely within it if the condition below is satisfied.

$\underline{L}(0.5(t_l+t_h))$ is inside the box OR for at least one boxpoint BP(i) it holds,that the nearest point of $\underline{L}(t)$ to BP(i) denoted $\underline{L}(t_i)$ is inside B and $t_l \%< t_i \%< t_h$.

8.3 Conic cuts box

The conic segment $\underline{C}(t)$ given in the (5.3.1) parametric form, with parametric interval $[\ t_l,\ t_h\ ]$ cuts the box B or lies entirely within it if the condition below is satisfied.

$\underline{C}(0.5(t_l+t_h))$ is inside the box OR for at least one boxplane BPL(i) the intersection of the conic and BPL(i), denoted $\underline{C}(i)$ is inside B and $t_l \%< t_i \%< t_h$

8.4 Plane cuts box

The plane PL(x,y,z) given in the (5.4.1) implicit form cuts the box B, if the condition below is satisfied.

For at least two different boxpoints BP(i) and BP(j) the signs of their distances from the plane are different or with other words, there is at least one boxpoint which is "above" and another which is "below" the plane, viewing it from the origin.

8.5 Quadric cuts box

The quadric Q(x,y,z) = 0 given in the (5.5.1) implicit form cuts the box B or lies entirely within it if the condition below is satisfied.

> An arbitrary point of the quadric is inside B OR for at least one boxedge BE(i) the intersection point of Q and BE(i) is inside the box. (Quadric - straight line intersection means only the solution of a second-degree equation in t, substituting (5.2.1) into (5.5.1).)

8.6. Box cuts box

The box B_1 cuts the box B_2 or either of them lies in the interior of the other if the condition below is satisfied.

> The $(xmin_1, xmax_1)$ - $(xmin_2, xmax_2)$
> AND the $(ymin_1, ymax_1)$ - $(ymin_2, ymax_2)$
> AND the $(zmin_1, zmax_1)$ - $(zmin_2, zmax_2)$ intervals
> are overlapping. (Two intervals are considered to be overlapping if
>
> $min_2 <= min_1 <= max_2$ OR $min_1 <= min_2 <= max_1$
>
> holds.)

9. VECTOR - DQ-CURVE INTERROGATION

Given a $\underline{P} = (x_0,y_0,z_0)$ vector and a dq-curve $\underline{dqc}(u)$. Assuming that the "point in dqcbox" test is positive, we want to find the u_0 parameter value for which

$$\underline{dqc}(u_0) = \underline{P} \quad . \tag{9.1}$$

Quite frequently the $\underline{P}$ vector is a result of previous computations, so strictly speaking $\underline{P}$ will not lie on the curve, even if it should, due to inaccuracy problems. That is why the task to be solved is set up in a different way - find the nearest dq-curve point to $\underline{P}$. If

(9.2) $$| \underline{dqc}(u_0) - \underline{P} | < \text{tolerance} ,$$

then consider u_0 a good curvepoint, otherwise the result is "empty".

Decomposing the dq-curve into quadratic pieces, the i-th $\underline{r}_i(t)$ will be given in the following form (see 6.3)

(9.3) $$\underline{r}_i(u) = \underline{a}_2 u^2 + \underline{a}_1 u + \underline{a}_0 \qquad 0 <= u <= 0.5$$

(This may describe a straight line or a parabola.)

For the nearest point to $\underline{P}$, the $\underline{P} - \underline{r}_i(u_0)$ vector must be orthogonal to the tangential direction at u_0 or $\underline{P} - \underline{r}_i(u_0)$ must be nullvector. Therefore

(9.4) $$(\underline{P} - \underline{r}_i(u)) \dot{\underline{r}}_i(u) = 0$$

This gives an at most third degree equation in u. The roots which lie within the given parametric interval will be selected for further examination until we find the curvepoint, which satisfies the tolerance criterion.

10. CURVE - DQ-CURVE INTERROGATION

Given a straight line or a conic segment or a dq-curve to be intersected with another dq-curve. After executing the "straight cuts dqcbox", "conic cuts dqcbox" or "dqcbox cuts dqcbox" global tests, respectively and assuming they are positive, the dq-curve(s) are decomposed into quadratic pieces (see 6.3).

Depending on whether the quadratic equation represents a straight line or a parabola, elementary interrogations between "straight and straight", "straight and conic" and "conic and conic" will be made.

In case of "straight and straight" the two lines must always have a common plane, assuming their intersect each other. In the remaining two cases coplanarity test must be executed at first. If the two curves are not coplanar the possible intersection points can be obtained by intersecting the conic-plane with the other - straight line or conic - segment. (This means simple substitution of 5.2.1 or 5.3.1 into the 5.4.1 equation, resulting a linear or second degree equation for the parameter of the intersecting curve.) If the two curves are coplanar the case is more difficult. A quadric surface will be constructed, which contains the conic and that will be intersected with the other - straight or conic - segment. (Here substitution of 5.2.1 or 5.3.1 into the 5.5.1 equation leads to a second degree or a fourth degree equation in respect to the parameter of the intersecting curve.)

It must be emphasized that the coplanarity tests must also be performed by using tolerances. The latter mentioned coplanar case is also solved by curve-surface intersection instead of intersecting in plane to avoid the tolerance problem, when two curves run very closely to each other, but mathematically they do not intersect each other. In the chosen way, the tolerance oriented vector - curve interrogation will give the correct answer, whether the possible intersection point really represents an intersection point of the two curves. Of course, both curvepoints must be checked,whether they fall into the corresponding parametric intervals.

11. PLANE - DQ-CURVE AND QUADRIC - DQ-CURVE INTERROGATIONS

Given a plane or a quadric surface to be intersected with a dq-curve. After executing the "plane cuts dqcbox" or the "quadric cuts dqcbox" global tests, if they are positive, the dq-curve will be decomposed into quadratic pieces as previously. Substituting the 5.3.1 equation into 5.4.1 or 5.5.1 we obtain a second degree or fourth degree equation for the unknown parameter values. The roots must be checked, whether they fall into the given parametric interval.

12. VECTOR - DQ-SURFACE INTERROGATION

Given a $\underline{P} = (x,y,z)$ vector and a dq-surface $\underline{dqs}(u,v)$. It is assumed to be non-selfintersecting and it is also assumed that the first derivatives exist everywhere. Assuming that the "point in dqsbox" test is positive, we want to find the (u_0, v_0) parameter values for which

$$\underline{dqs}(u_0,v_0) = \underline{P} \tag{12.1}$$

Instead of this - due to the inaccuracy problems previously described - we have to use the condition below

$$| \underline{dqs}(u_0,v_0) - \underline{P} | = \text{minimum} \quad . \tag{12.2}$$

If the nearest distance is smaller than the tolerance we have got (u_0,v_0), otherwise the result is empty.

The dq-surface is decomposed into biquadratic subpatches (see 7.3) and for the corresponding boxes we again perform the "point in dqsbox" tests. When it is positive, we look for the nearest point of that biquadratic patch. This procedure is terminated when the first (u_0,v_0) is found.

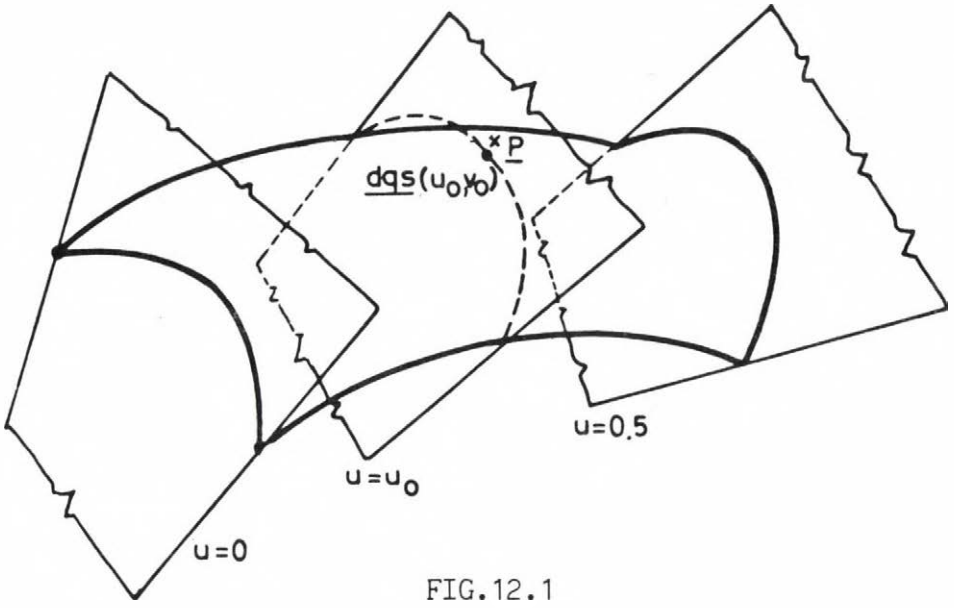


FIG.12.1

The algorithm to be presented is based on the fact that for each constant parameter line of a biquadratic patch a plane may be constructed which contains it. Let us suppose that the biquadratic patch is given in the form of

(12.3) $$\underline{r}(u,v) = \underline{A}(u) + \underline{B}(u)\cdot v + \underline{C}(u)\cdot v^2$$

$$0 <= u,v <= 0.5 \quad .$$

The normal of a u-constant plane can be constructed as follows.

CASE 1. If the u-constant parameter lines of the patch are "real" quadratics, i.e. $\underline{B}(u) \times \underline{C}(u) \neq 0$ (they are not parallel and $\underline{C}(u) \neq 0$) the normals of these "parabola planes" are

(12.4) $$\underline{n}(u) = \underline{B}(u) \times \underline{C}(u)$$

(Note that if there are discrete u_0 values for which the above expression is zero, instead of it, the limit value below - using L'Hospital's rule - must be taken

(12.5) $$\underline{n}_0(u_0) = \lim_{u \to u_0} (\underline{B}(u) \times \underline{C}(u)) / \mid \underline{B}(u) \times \underline{C}(u) \mid).$$

CASE 2. If the u-constant parameter lines are all linear, i.e. $\underline{B}(u) \times \underline{C}(u) = 0$ for all u in the given parametric interval, the normal can be calculated as

the vectorproduct of the surface normal at a fixed v_0 and the parameter line direction vector, i.e.

$$\underline{N}(u,v_0) = \underline{r}_u(u,v_0) \times \underline{r}_v(u,v_0) \tag{12.6}$$

$$\underline{n}(u) = \underline{N}(u,v_0) \times \underline{B}(u)$$

Having the normal, the plane equation as a function of u is

$$\underline{n}(u)\,(\underline{A}(u) - (x,y,z)) = 0 \quad , \tag{12.7}$$

where $\underline{A}(u)$ is obviously a point of the plane.

IF $\underline{P}$ is on the surface or very close to it, there will be a parametric plane, which contains it (see Fig. 12.1). This is so, since the above plane is varying continuously as u runs from 0 to 0.5. These planes occupy a continuous part of the space between the u = 0 and u = 0.5 planes. This certain u value can be obtained by substituting $\underline{P}$ into the (12.7) equation and solving the polynomial in u. After the plane and by means of it the nearest u constant parameter line is determined, the corrresponding v value can be calculated finding the "nearest point" on the

$$\underline{r}(v) = \underline{A}(u_0) + \underline{B}(u_0)\, v + \underline{C}(u_0)\, v^2 \tag{12.8}$$

quadratic in the same way as we did in Section 9.

The selection whether the first or the second case holds must be done before we calculate the actual coefficients of the(12.7)equation. If u or v represent the "real" quadratic case, we apply case 1, which will give a 6th degree equation (12.7). (The process goes similarly for v, if we swap the role of u and v in the previous equations.) If both u and v are linear, that leads to a simple 3rd degree equation.

Special care must be taken to handle the slightly degenerate patches where the surface-normal and the parametric plane normal are almost parallel. (This must be checked at each (u_0,v_0).)

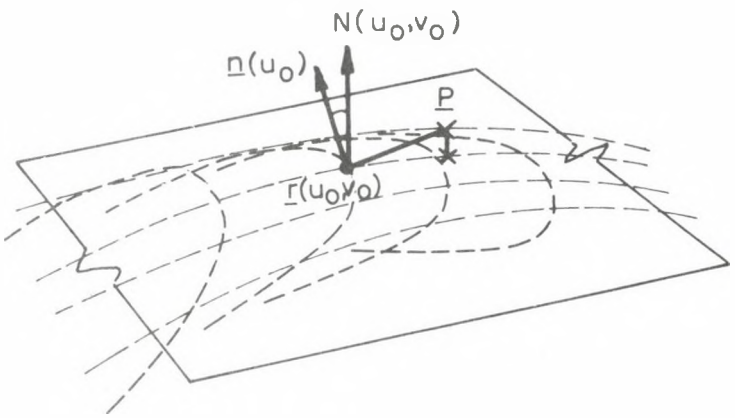


FIG. 12.2.

As the above figure shows, here the angle between $\underline{n}(u_0)$ and $\underline{N}(u_0,v_0)$ is relatively small and $\underline{r}(u_0,v_0)$ will not give the nearest point on the surface. Inspite of this in most cases the distance to the surface will be within tolerance and (u_0,v_0) will be accepted. Otherwise, that is in the case of points which are not too close and/or relatively small angles between the normals, an iterative method is suggested for finding the nearest u constant parameter line. By interval-halfing in u at each step the distance from the given parameter line and the point must be calculated, until the nearest is found. This method works in the most degenerate case, where the patch is planar with parabolic parameter lines in both parametric directions.

13. CURVE - DQ-SURFACE INTERROGATIONS

Given a $\underline{q}(t)$ parametric curve with parametric boundaries t_l,t_h and a dq-surface $\underline{dqs}(u,v)$ with the same assumptions as in Section 12.

We want to find the curve-surface intersection points for which

$$\underline{q}(t_0) = \underline{dqs}(u_0, v_0) \tag{13.1}$$

The algorithm presented here will work efficiently for any type of curve, always supposing its intersection with a plane can be simply determined. (This is the case for straight lines, conics and the decomposed quadratic pieces of dq-curves we are interested in and also for third or fourth degree parametric curves.) The algorithm is specific, in the sense that it uses the planar constant parameter line property of biquadratic patches, as we did in the previous Section.

Assuming that the global "curve cuts dqsbox" test was positive, we decompose the dq-surface into biquadratic subpatches and deal with the ones where the "small box test" showed that intersection may occur. We can assign planes to each constant parameter lines as we did in Section 12.(Here again the u constant parameter lines are choosen - see equation 12.3.)

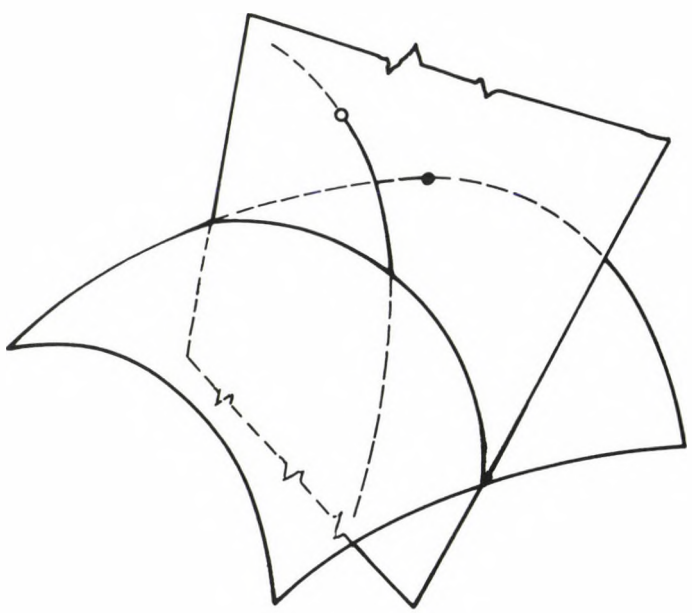


FIG. 13.1.

The principle of the algorithm is that once a parametric curve intersects a patch,

either it must lie entirely in a certain parameter line plane (Fig.13.1)

or in a very small vicinity of the plane which contains the intersection point there must be two planes for which the curve-plane intersection points are "above" and "below" the surface respectively, as is shown on Fig. 13.2. (We do not go into deatils concerning touching points.)

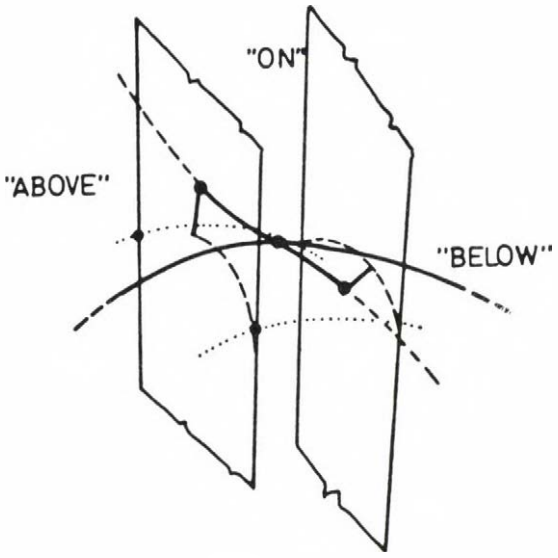


FIG.13.2.

At the very beginning it must be determined whether or not the first case holds, since this leads at most to conic-conic intersection in a plane, (supposing dq-curves are the most complicated curves we have). Moreover, this test must be made because otherwise the iterative method suggested for the second case will fail.

If there is a plane which contains entirely the parametric curve, then the plane, constructed below must contain the whole parameter line and the curve. Starting from the equation (12.3) and taking the endpoints of a u parameter line and the given curve segment we obtain

$$\underline{v}_1(u) = \underline{r}(u,0.5) - \underline{r}(u,0)$$
$$\underline{v}_2 = \underline{q}(t_2) - \underline{q}(t_1) , \quad t_1 \neq t_2 . \tag{13.2}$$

The normal is

$$\underline{n}(u) = \underline{v}_1(u) \times \underline{v}_2 , \tag{13.3}$$

so the plane equation can be

$$\underline{n}(u) \, (\underline{q}(t_1) - (x,y,z)) = 0. \tag{13.4}$$

Substituting $\underline{A}(u)$ into (x,y,z), it gives an at most 4th degree equation for the unknown u. Once the possible u-s are obtained, a third point of the constant parameter line and a third point of the other curve will be chosen to test whether they are fully contained in the plane. (In case of conics a sufficient condition for this is if three points lie in that plane.) If this first case is satisfied, after making planar curve-curve intersections (see Section 10) the interrogation for the current biquadratic patch can be concluded.

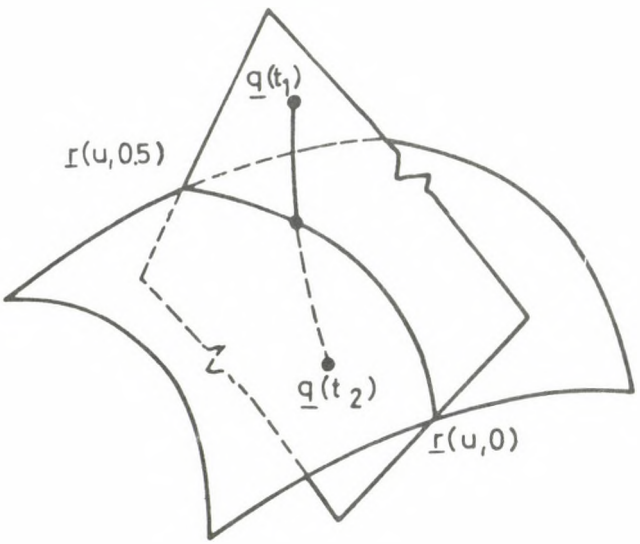


FIG.13.3

If the first case is not satisfied we shall make interval-halving iterations in u, until the desired intersection points are obtained within the given tolerance. For each u, we calculate the parameter line plane and also decide, whether the point $\underline{P}_i$ where the curve intersects this plane is above or below the surface. This can be done easily, once we have a point $\underline{P}_B$, which is in the plane and "below" the current parameter line. To obtain this point, the surface normal must be calculated and the "below" point will be in the half-plane, where the projection of the negated normal points. The next step is to intersect the $\underline{P}_i\underline{P}_B$ straight line segment with the parameter line and according to the number and position of these, to decide whether $\underline{P}_i$ is on, above or below the surface. (see Fig. 13.4 - $\underline{P}_1,\underline{P}_2,\underline{P}_3$)

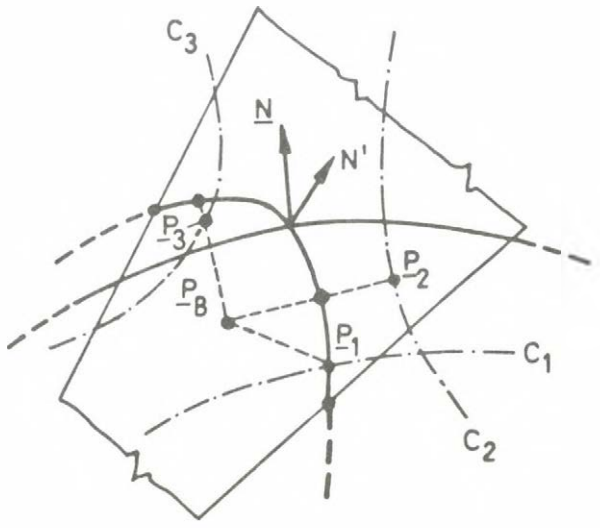


FIG.13.4.

This interval-halving procedure converges quite quickly. The plane intervals, which do not contain any piece of the curve or only pieces which entirely go above or below the surface can be thrown away, while the halving procedure culminates around the planes, where at one side "above", at the other side "below" points are found. A tolerance criterion stops the iteration.

The above algorithm is relatively fast since only vector and scalar products must be calculated along with solving second degree equations. An alternative method could be, that at each step, the distance from the current parameter line is also calculated and Newton-Raphson like iteration in u is performed at the critical intervals.

14. SURFACE - DQ-SURFACE INTERROGATIONS

The considerations outlined in Section 3 very much influenced the implementation of the double-quadratic surface intersection algorithm in BUILD. A particular combination of the parametric grid and the contour following methods with some heuristics was chosen, using the basic features of quadratics.

Given two surfaces S_1 and S_2. S_1 can be a plane , a quadric or a dq-surface given by the 5.4.1, 5.5.1 and 7.1.1 equations, S_2 is always a dq-surface. After making a box around S_2 global interference is tested, performing "plane cuts dqsbox", "quadric cuts dqsbox" and "dqsbox cuts dqsbox" tests, respectively. If these are positive, the test are repeated for each biquadratic patches of S_2, to select the ones, which contain possible intersection curves. Pieces of the intersection curves in respect to the individual biquadratic patches will be determined and later these pieces will be concatenated.

The algorithm generates sequences of (u,v) pairs with their tangential direction in the parametric plane of S_2 and sorting is also performed here. It holds for all (u,v)-s, that substituting them into the S_2 equation, the points obtained will lie on both S_1 and S_2. Basically the parametric image of the

intersection curves (termed u-v contours) will be obtained and curve fitting in 3D will be applied only in the final phase.

Note an important feature here: the double-quadratic intersection algorithm works independently from the type of the other surface. The only data we need to interrogate S_1 is an intersection point of S_1 and a constant parameter line of S_2. This means parametric quadratic - plane, quadratic - quadric and quadratic - dq-surface intersections respectively. (These cases were discussed formerly in Section 10,11 and 13.) The time needed to get this data essentially determines the speed of the whole u-v contour generation. Consequently, the double-quadratic algorithm is very efficient for planar and quadric surfaces, which constitute the majority of surfaces in a solid modeller, since only second and forth degree equations most be solved using known direct formulas. The free-form surface-surface intersection uses iterative methods, so its efficiency is lower, however, it is still much better than for higher-degree parametric patches.

After obtaining the (u,v) intersection points in the parametric plane of S_2, we also need a tangential direction for sorting. This can be calculated by deriving the implicit equations for the plane or quadric cases. If S_1 is also a dq-surface, we have to substitute it by its tangential plane at the intersection point and using that implicit equation to determine the tangential direction we need for S_2.

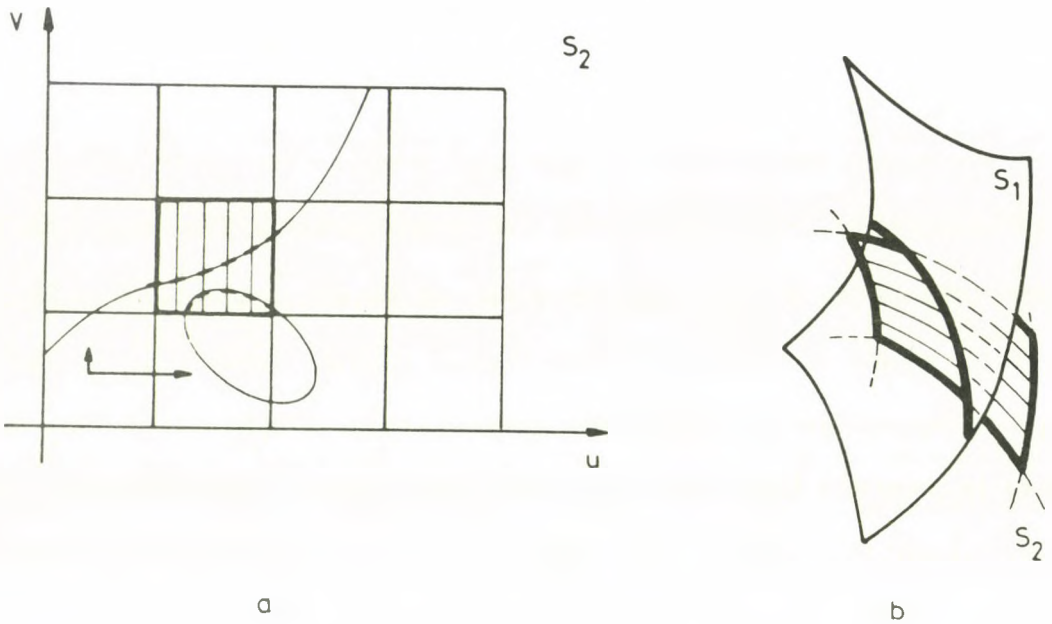


FIG. 14.1.

Instead of using a very fine grid of S_2's u-v lines, only a relatively coarse initial grid of only u-lines will form the basis for a sophisticated sorting algorithm (Fig.14.1). If necessary, it will search for missing (u,v) points between the existing ones. This practically means only local refinement of certain parts of the grid, ensuring that the contour-sections, going almost parallel to the u-constant parameter lines and the strongly curved sections will also be found.

The intersection process is performed patch by patch from left to right and from bottom up in the S_2 parametric plane (see Fig. 14.1). The separate parts of the u-v contours, which cross patch-boundaries are simultaneously linked together during the above scanning procedure. The intersection curves always run "from left to right" and "upwards" if possible, until they reach some surface boundaries or join up in the case of closed curves.

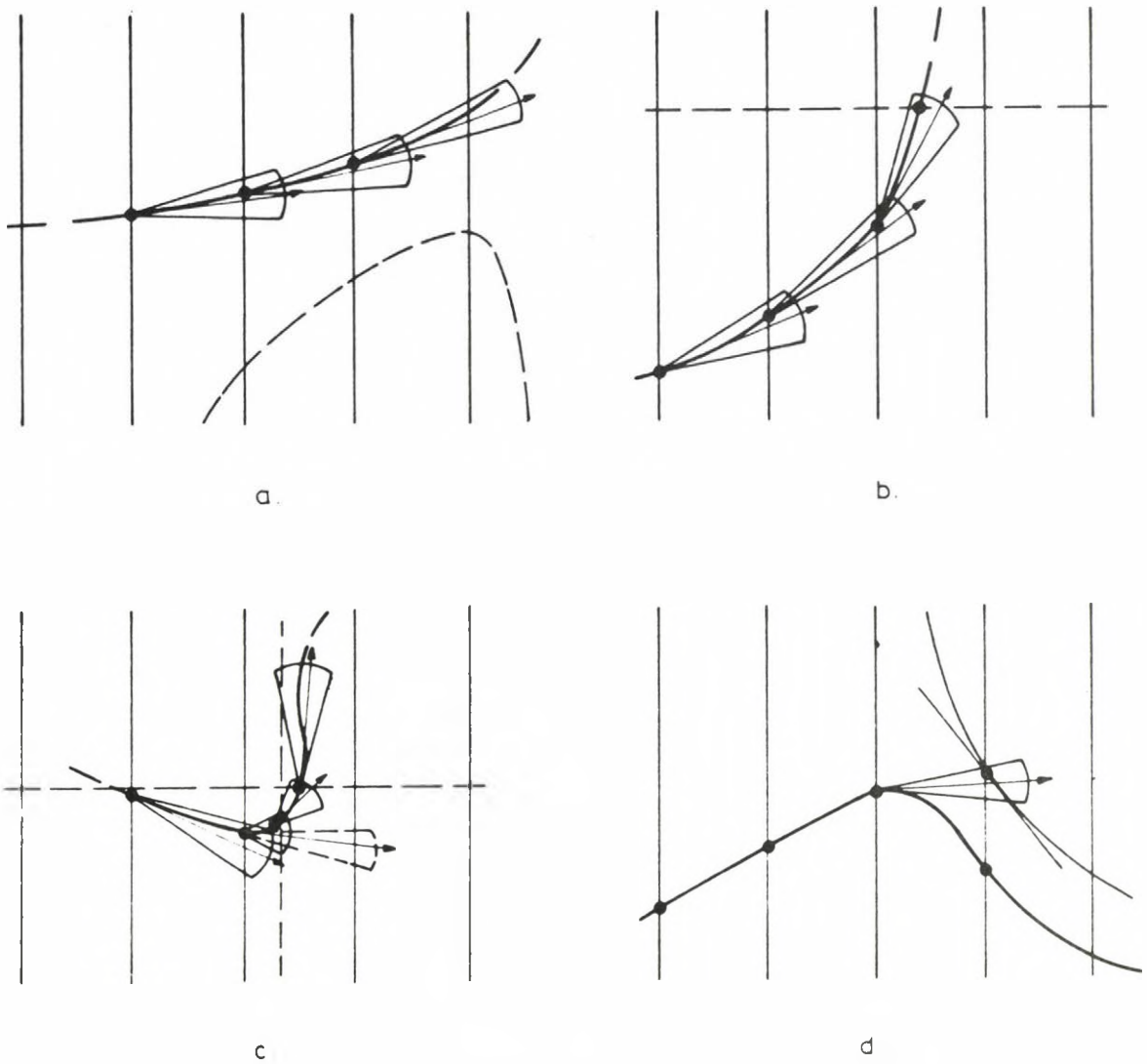


FIG.14.2

Once the data-points and their tangential line are given, a so-called isosceles search-triangle helps to differentiate the possible next points. Here a simplified version of the sorting procedure will be presented. The angle and the height of the triangle together with the width of the u parameter lines determines the basic step length, which is further reduced at points of higher curvature.

The current point and the tangent give the orientation of the search triangle. Two search triangles are generated at the starting point of a loop. Sorting proceeds smoothly if there is one and only one point in the search-triangle

(Fig. 14.2.a). If there are none and the tangential line is steep enough, the next point will be found on a v constant parameter line - Fig. 14.2.b. If none of the above conditions holds, this means either there is high curvature at the current point or more than one intersection curve run in the vicinity of the next point. In this case, the search triangle will be shortened and/or tightened - Fig.14.2.c, and new intermediate points searched for. The triangle is also shortened if a test with rotation away from the tangential direction also fails - Fig.14.2.d.

The above algorithm was reliable when designing "real engineering objects". However, considered from the theoretical point of view, the deficiences of the parametric grid and the contour following methods still remain. Extremely small loops between the neighbouring u-parameter lines will not be discovered and very close intersection lines with almost parallel tangents cannot be distinguished in certain cases. If the algorithm fails, incomplete u-v contours will be found and in this case the procedure must be repeated with refined step length parameters.

The final step in generating the intersection curve is to fit a smooth curve through the set of discrete points calculated above. Piecewise linear interpolation is not sufficient for storing edge equations in a solid modeller.

Curve fitting can be solved by some global procedure, such as the least-squares minimalization or spline interpolation. (Formerly B-splines were used for this purpose in BUILD.) Based on the arguments summarized in Section 3 a new, local, adaptive curve-fitting method, using double-quadratics was introduced. Given a double-quadratic curve segment with its endpoints and the 3D tangential lines, which can be calculated from the surface normals, the appropriate magnitude of the tangent vectors can be adjusted to get the best

possible approximation of the intersection curve in this region. This only involves the solution of a linear equation system with two variables, obtained from local least square minimisation of data points. The fitting procedure also tests, whether a double-quadratic segment covering more than two neighbouring intersection points can be fitted without loss in accuracy. When compared with the previous methods the resulting dq-curves are more accurate, also the store needed and the number of segments are considerably reduced. More details can be found in the next chapter.

15. CONCLUSION

The complete range of dq-curve and dq-surface interrogations was presented. These procedures made it possible to handle double-quadratic type free-form surfaces in the same way as the conventional curves and surfaces in the BUILD volumetric modeller. Because of the advantageous properties of dq-s, these interrogations overcome the particular problems that arise when trying to create solid models in an efficient way. The (plane - dq-surface) and (quadric - dq-surface) interrogations are comparable in CPU time with the general (quadric - quadric) ones. Unlike other types of parametric patches, acceptable interactive response times were experienced even at (dq - dq) intersections.

Chapter IV

A SIMPLE ADAPTIVE CURVE-FITTING ALGORITHM FOR GENERATING INTERSECTION CURVES IN VOLUMETRIC MODELLERS

1. INTRODUCTION

At present most volumetric modelling systems can only represent solids bounded by simple analytic surfaces. This is due to the difficulties arising from incorporating free-form geometry into the model. These difficulties come mostly from the free-form intersection problems along with the problem of how to generate accurate intersection curves in a reliable way within reasonable time and store limitations.

After the introduction of the double-quadratic mathematics and the geometric interrogations for these elements, in this chapter we concentrate on a specific problem, that is how to generate free-form intersection curves based on the local features of the dq-curves.

As was described previously, most surface-surface intersection algorithms consist of three-basic phases, apart from the very simple cases, when the equation of the intersection curve can be expressed explicitly [32], [62]. The first phase is calculation of separate intersection points, the second is generation of their sequences, i.e. joining up of the consecutive points (often called "sorting") and the third is curve-fitting, i.e. generation of a smooth curve, going through the points obtained previously. These phases are often overlapping depending on the intersection methods applied. Quite frequently the third phase is neglected since linear interpolation is

appropriate for the application involved. This is not the case for advanced geometric modelling systems, particularly those with boundary representation where every curve and surface equation must be supplied. These curves must be at least tangentially continuous and also very accurate in order to avoid geometric inconsistencies.

There are many algorithms for fitting curves to a set of datapoints (see for example [32]) , but most of them do not cope with the special requirements of surface-surface intersections in a geometric modeller. In addition to accuracy and tangential continuity, schemes based on the local features of the surfaces are preferred, possibly using direct methods instead of iterative ones. Not only must the calculation time be relatively short, but also the use of data storage to represent these curves must be efficient. The number of the curve segments can be much less than the number of the initial datapoints, when the individual curve segments can be fitted to more than two consecutive datapoints without loss in accuracy, supposing relatively low curvature/chordlength ratio makes this possible.

The former B-spline curve fitting in BUILD [65] had the benefits of curvature continuity and a very compact data representation, but its global characteristic and inaccuracy sometimes led to certain problems. Two of them are illustrated in FIG.1 and FIG.2. Keeping the internal tolerance of the model to 10^{-5}, the largest deviation from the intersecting surfaces may have reached even 10^{-3} between the datapoints, which may have made the model inconsistent. This is illustrated for example, when three surfaces are to be intersected. The V vertex is calculated as the intersection of the S1-S2 curve with the S3, but it is out of tolerance with respect to the S1-S3 curve, on which it should also lie (FIG.1). The other example is, that due to the global scheme, the intersection curve may even run beyond the edge of the surface (FIG.2).

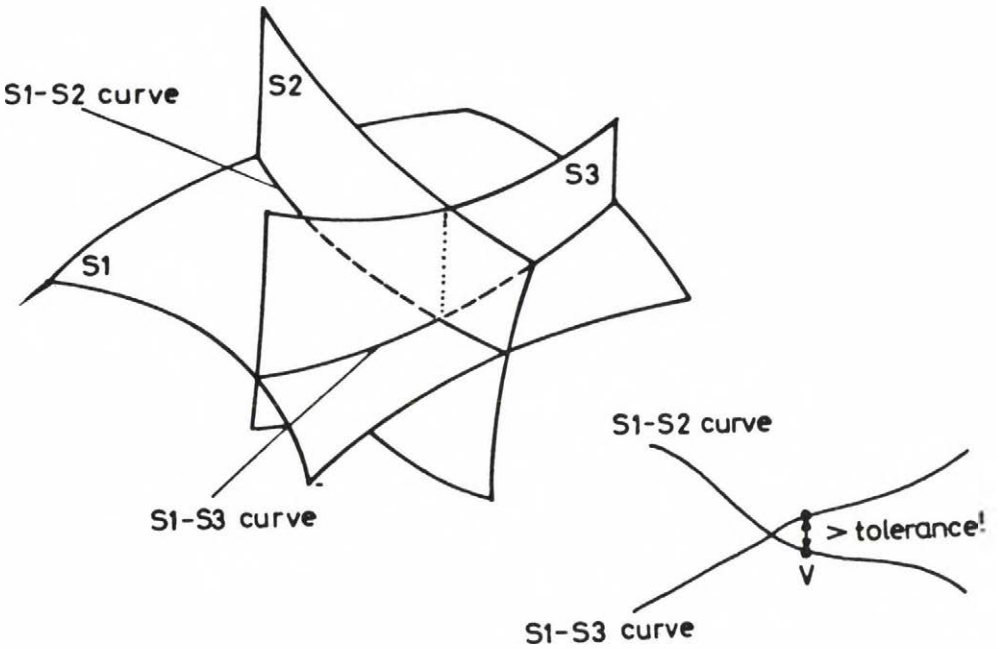


FIGURE 1.

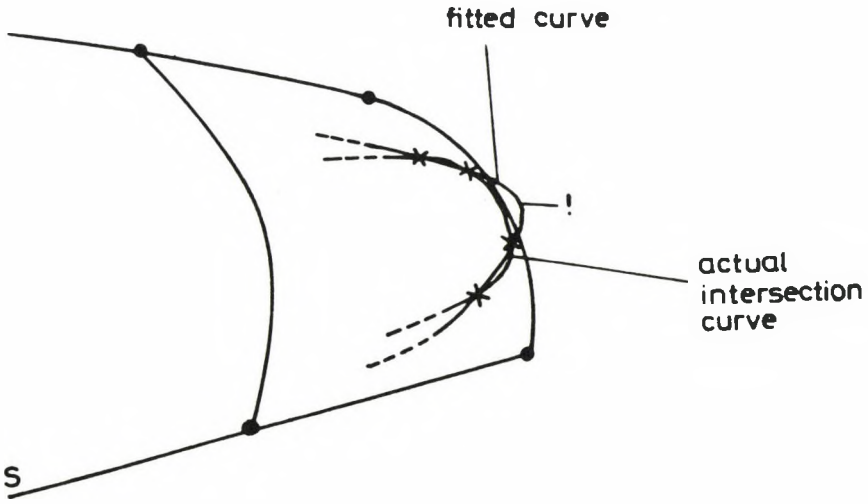


FIGURE 2.

The simple adaptive dq-curve fitting was created specifically keeping in mind the above considerations about intersection curves in a geometric modeller. Here only the local dq-curve fitting will be discussed, the details of obtaining the initial intersection points of the surfaces can be found in the previous chapter.

The details of Bspline interpolation are also only referenced here [65] [32].

The description of the dq-algorithm will be followed by examples evaluating some test runs of surface-surface intersections for both Bspline and dq-curves.

2. THE DQ-CURVEFITTING ALGORITHM

Given a sequence of points, $\underline{P}_i$ (i=1,...,n) resulting from some surface-surface intersection algorithm. Each $\underline{P}_i$ lies on both surfaces (S_1 and S_2) involved. It is also assumed that the normal vectors at $\underline{P}_i$ to both S_1 and S_2 can be calculated and that they are not parallel: that is the two surfaces are not tangential along the intersection curve. The result of the algorithm to be presented is a sequence of C1 continuous dq-curve segments optimized according to the local features of the surfaces, which gives a very good approximation to the actual intersection curve.

The tangential direction of the intersection curve at each $\underline{P}_i$ can be calculated by the vector product of the normal vectors to S_1 and S_2, using the above assumption. Any curve in S_1 going through $\underline{P}_i$ is perpendicular to the normal there and this holds to the S_2 normal as well. Therefore, the tangent vector of the intersection curve is orthogonal to both normals, thus

$$\underline{t}_{i0} = (\underline{n}_{iS1} \times \underline{n}_{iS2}) \; / \; |\underline{n}_{iS1} \times \underline{n}_{iS2}| \; .$$

(This sense of direction is taken by convention.)

A dq-curve segment is determined by its endpoint position and tangent vectors (FIG.3). The so-called fullness of the curve-segment or in other words the extent of how far the curve-segment goes along its tangential lines at the endpoints can be altered by the magnitudes of the tangent vectors.

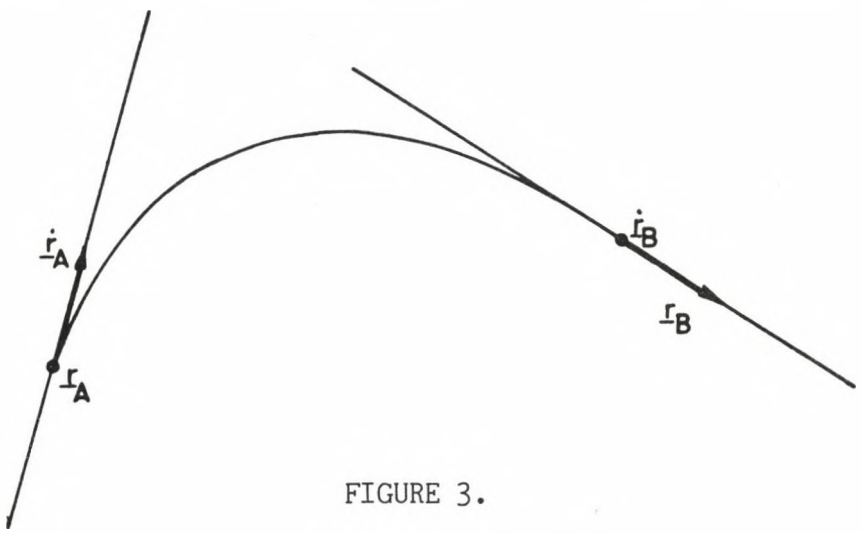


FIGURE 3.

Each dq-curve segment will start at a particular P_i and end at another P_{i+k+1}, where k>=0 and i+k+1 <= n.

The 0-version of the dq-curve fitting creates one dq-segment between two neighbouring datapoints (k=0). The endpoints and the tangential directions are also given, the remaining tangent magnitudes can be set according to the $|P_{i+1} - P_i|$ chordlength, which gives a reasonable value, when nothing is known about the interior of the curve and the curvature/chordlength ratio is not high within this segment. This latter condition is hopefully satisfied by the procedure which generates the intersection points. Unfortunately we cannot optimize the midpoint or the interior points of this curve-segment, since for that we would need to express the distances from the surfaces as a function of a curve-point parameter, which is not possible.

This 0-version fitting gives good results, however the store capacity needed is relatively big: the number of segments is always one less than the number of datapoints, either we have a simple or a complicated intersection curve.

The adaptive dq-curve fitting algorithm attempts to make a dq-curve segment between the $\underline{P}_i$ and $\underline{P}_{i+k+1}$ points by forcing the current dq-curve segment to lie as close as possible to the $\underline{P}_{i+1},\ldots,\ \underline{P}_{i+k}$ points in a least squares sense. The initial k value is set according to the "turning" of the tangent vectors along the $\underline{P}$ points. A linear system of equations is obtained with the two unknown magnitudes of the endpoint tangent vectors. (See Appendix 1.)

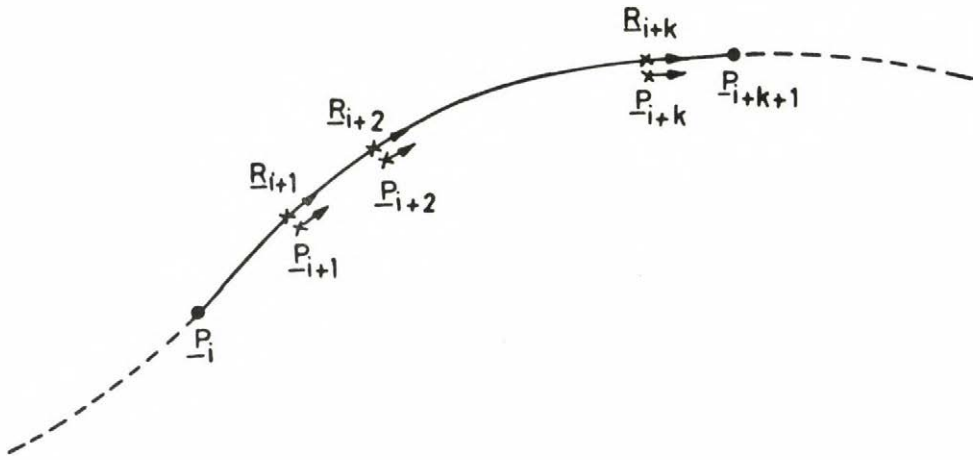


FIGURE 4.

Once these magnitudes have been calculated we need to decide whether the "best" segment fitted on the j=i,i+1,...,i+k+1 datapoints is accurate enough or not. This is performed by checking the so-called distance and angle tolerances. First we calculate the nearest curvepoints to each $\underline{P}_j$ point,

denoted by $\underline{R}_j$. (See Appendix 2.) For acceptance the distance tolerance must be greater than $|\underline{P}_j - \underline{R}_j|$ and the angle-tolerance must be greater than the angle between the dq-curve segment at $\underline{R}_j$ and the actual tangent vector of the intersection curve $\underline{T}_{j0}$ for each $j=i+1,\ldots,i+k$, (FIG.4).

If the dq-segment covering k datapoints goes out of the tolerances anywhere between the two ends, then k must be reduced, while if it is within the tolerances another curvefitting attempt will be made with a bigger k to find the optimal number of datapoints to be concatenated within one dq-segment. Obviously the final result is the dq-segment, which fits the highest number of consecutive datapoints and then the procedure can start again for the next datapoints until the last one is reached.

This "learning" procedure assures that if subsets of the initial data lie approximately on a straight line or a double-quadratic-like segment, those points will be covered by only one segment and the total number of the segments will be drastically reduced.

3. CONCLUSION

The 0-version and the adaptive dq-curve fitting have been implemented in BUILD according to the principles presented. For evaluating B-spline and dq-curves a small analysis program was written, which calculated the mean and the largest distances of a curve from a given surface, in our case the distances of the actual approximating intersection curve and the surfaces to be intersected. By means of this the following conclusions have been drawn based on several test-runs.

The calculation time of the 0-version curve fitting is 1-2 times more than that of Bspline fitting. The adaptive dq-fitting takes 2-5 times longer than the simple one. However, higher accuracy means more dq-curve segments (that is less attempts at optimizing) consequently shorter time.

Considering the number of initial datapoints and the number of segments generated by dq-fitting, it results in significantly less segments than those of Bspline fitting. It must be kept in mind that Bsplines need

(number of curvesegments+3)*3

scalar quantities for data storage, while dq-curves, if they are tangentially continuous need

(number of curvesegments+1)*7

scalars, i.e. the store needed is about 7/3 that for Bspline, if the number of segments are the same. Fortunately the dq-curves use many less segments. Dq-curve fitting with around 1/10 as many segments as for Bsplines (7/30 of Bspline store) gives 5-10 times worse accuracy. Around 1/4 or less (7/12 of Bspline store) always gives better accuracy, while around 1/2 as many (7/6 of Bspline store) gives 10 times or more better accuracy. Around 2/3 as many (14/9 of Bspline store) and also with the 0-version dq-fitting gives significantly better accuracy: 20-100 times better in mean-distances, 5-50 times better in largest distances.

These results are illustrated by two out of the many test runs, which can be found in Appendix 3. (A dq-surface with a plane, and another dq-surface with a cylinder are intersected.) The tables there show the calculation time, the number of segments, the mean and largest distances from both surfaces for Bspline curves and dq-curves with both 0-version and adaptive fittings. In the last cases the distance and angle tolerances are also given.

Some pictures of the generated dq-curves can be found in Appendix 4 (FIG.7 and FIG.8).

To sum up, using the presented dq-curve fitting the higher accuracy and the smaller number of curve segments lead to a significant improvement for representing complicated solid bodies. This improvement is even more significant when the number of the datapoints is relatively small.

4. APPENDIX - 1

Least-square curve-fitting of a double quadratic curve segment

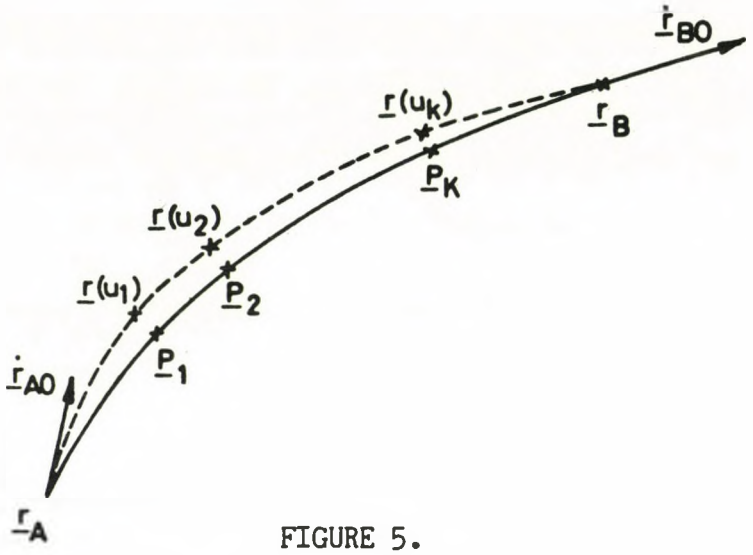


FIGURE 5.

Given the two endpoints (r_A and r_B) of a dq-curve segment and also the tangential directions there (r_{AO} and r_{BO} unitvectors). We want to fit the best dq-curve to the k datapoints between r_A and r_B, denoted by P_i, i=1,2,...,k. (FIG.5.) It is also assumed that there is a reasonable preliminary parametric distribution of $(u_1,u_2,...,u_k)$, for which $u_i < u_{i+1}$; $-0.5 <= u_i <= 0.5$ and $r(u_i)$ is near to P_i for i=1,2,...,k. (This can be obtained by setting the magnitudes of our "trial" dq-segment according to the chordlength and finding the nearest curvepoints to each P_i - see Appendix 2.) The [-0.5,0.5] parametric interval was chosen, which enables us to make simple calculations later using the sign-dependent dq-matrix.

Actually we have two scalar degrees of freedom, the magnitudes of the tangent vectors at the endpoints and want to find the best dq-curve for which

$$\sum_{i=1}^{k} | P_i - r(u_i) |^2 = \text{minimum.}$$

DQ-CURVE FITTING

With the two unknown magnitudes - α and β - the dq-curve equation can be written as follows:

$$\underline{r}(\alpha,\beta,u) = [1\ u\ u^2] \begin{bmatrix} 0.5 & 0.5 & 0.125 & 0.125 \\ -2 & 2 & -0.5 & -0.5 \\ \begin{matrix}-2 & 2 & -1.5 & -0.5\end{matrix} \\ \begin{matrix}2 & -2 & 0.5 & 1.5\end{matrix} \end{bmatrix} \begin{bmatrix} \underline{r}_A \\ \underline{r}_B \\ \alpha\cdot\dot{\underline{r}}_{A0} \\ \beta\cdot\dot{\underline{r}}_{B0} \end{bmatrix} =$$

$$= [\ x_1(u)\ x_2(u)\ x_3(u)\ x_4(u)\ ] \begin{bmatrix} \underline{r}_A \\ \underline{r}_B \\ \alpha\cdot\dot{\underline{r}}_{A0} \\ \beta\cdot\dot{\underline{r}}_{B0} \end{bmatrix}$$

where $-0.5 <= u <= 0.5.$

The equation to be minimized becomes

$$S(\alpha,\beta) = \sum_{i=1}^{k} | \underline{P}_i - \underline{r}(\alpha,\beta,u_i) |^2 = \text{minimum},$$

$S(\alpha,\beta)$ has its extremum where

$$\partial S / \partial\alpha = 0 \quad \text{and} \quad \partial S / \partial\beta = 0 .$$

This leads to a linear system of equations with two unknowns, which may have no solution or an infinite number of solutions or one solution. In the last case that solution obviously gives the desired minimum value of the best fit.

Using the $\partial f^2(\alpha) / \partial\alpha = 2\cdot f(\alpha)\ f'(\alpha)$ formula the previous equations can be written as

$$\sum_{i=1}^{k} [\ \underline{P}_i - (x_1(u_i)\underline{r}_A + x_2(u_i)\underline{r}_B + x_3(u_i)\alpha\,\dot{\underline{r}}_{A0} + x_4(u_i)\beta\,\dot{\underline{r}}_{B0})\ ]\ x_3(u_i)\dot{\underline{r}}_{A0} = 0$$

$$\sum_{i=1}^{k} [\ \underline{P}_i - (x_1(u_i)\underline{r}_A + x_2(u_i)\underline{r}_B + x_3(u_i)\alpha\,\dot{\underline{r}}_{A0} + x_4(u_i)\beta\,\dot{\underline{r}}_{B0})\ ]\ x_4(u_i)\dot{\underline{r}}_{B0} = 0$$

which make the equations below:

$$a\alpha + b\beta + c = 0$$

$$d\alpha + e\beta + f = 0$$

where

$$a = -\sum_{i=A}^{k} x_3^2(u_i)$$

$$b = -\sum_{i=A}^{k} x_3(u_i)x_4(u_i)\dot{r}_{AO}\dot{r}_{BO}$$

$$c = \sum_{i=A}^{k} (P_i - x_1(u_i)r_A - x_2(u_i)r_B) x_3(u_i) \dot{r}_{AO}$$

$$d = -\sum_{i=A}^{k} x_3(u_i)x_4(u_i)\dot{r}_{AO}\dot{r}_{BO}$$

$$e = -\sum_{i=A}^{k} x_4^2(u_i)$$

$$f = \sum_{i=A}^{k} (P_i - x_1(u_i)r_A - x_2(u_i)r_B) x_4(u_i) \dot{r}_{BO}$$

If $|ae - bd| \neq 0$ the known unique solution is

$$\alpha = (bf - ce) / (ae - bd)$$

$$\beta = (af - cd) / (ae - bd)$$

5. APPENDIX - 2

The nearest point on a double-quadratic segment

Given a point P and a parametric curve-segment $r(u)$, $u_1 < u <= u_2$ in space. We want to find the nearest curve point to P denoted by Q. If there is a nearest point inside the given parametric interval, then $Q = r(u_0)$ for a certain u_0, $(u_1 < u_0 < u_2)$ and $|Q - P| <= |r(u) - P|$ for all $u \neq u_0$ in the given interval, otherwise the nearest point will be equal to the nearer end-point of the curvesegment, i.e. $r(u_1)$ or $r(u_2)$.

In the former case the QP vector must be orthogonal to the curve at u_0, that is the scalar product of $P - r(u_0)$ and $\dot{r}(u_0)$ vectors must be zero. This

orthogonality condition can be superficially interpreted in an indirect way, that is, if $\underline{QP}$ was not orthogonal to the curve, then moving a bit on the tangential line towards the direction, where the angle between $\underline{Q}'$ and the tangential line is less than 90 would result in a point $\underline{Q}''$ being nearer to $\underline{P}$ than $\underline{Q}'$.

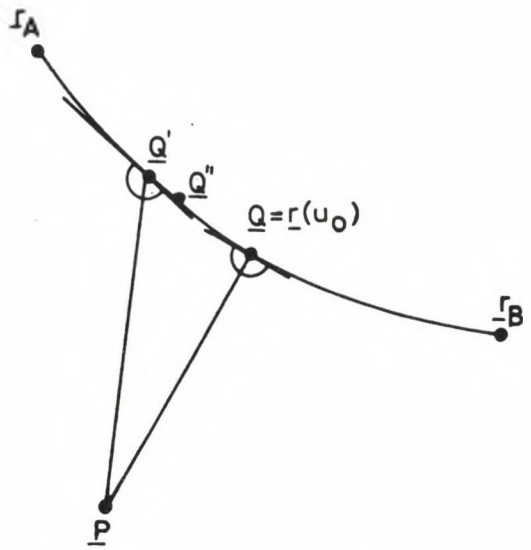


FIGURE 6.

This leads to an equation to be solved for u, i.e.

$$(\underline{P} - \underline{r}(u)) \quad \dot{\underline{r}}(u) = 0$$

If the above equation has more than one solution, the root within $[u_1,u_2]$ which supplies the smallest distance between $\underline{P}$ and $\underline{r}(u)$ must be selected.

The above equation even in case of cubics leads to a fifth-degree polynomial in u, which can be solved only numerically. In case of double-quadratics, due to the quadratic pieces, two directly solvable third-degree polynomial are obtained. (Note that considering dq-curve fitting we may assume that the point $\underline{P}$ is very near to the curve-segment and the curvature/chordlength ratio is low, therefore apart from the cases where $\underline{P}$ lies close to the midpoint of the dq-segment, only one equation must be solved.)

The dq-curve equation can be written in the form below:

$$\underline{r}(u) = \begin{cases} \underline{a}_2\, uu^2 + \underline{a}_1\, uu + \underline{a}_0 & uu = u\ ,\ 0 <= u <= 0.5 \\ \underline{b}_2\, uu^2 + \underline{b}_1\, uu + \underline{b}_0 & uu = u - 0.5\ ,\ 0.5 <= u <= 1\ . \end{cases}$$

With given endpoint position and tangent vectors $(\underline{r}_A, \underline{r}_B, \dot{\underline{r}}_A, \dot{\underline{r}}_B)$, the midpoint vectors can be obtained as

$$\underline{r}_M = 0.5\ (\underline{r}_A + \underline{r}_B) + 0.125\ (\dot{\underline{r}}_A - \dot{\underline{r}}_B)$$
$$\dot{\underline{r}}_M = 2.0\ (\underline{r}_B - \underline{r}_A) - 0.5\ (\dot{\underline{r}}_A + \dot{\underline{r}}_B)$$

By means of these the vector coefficients can be expressed as follows:

$$\underline{a}_0 = \underline{r}_A$$
$$\underline{a}_1 = \dot{\underline{r}}_A$$
$$\underline{a}_2 = 4.0\ (\underline{r}_M - \underline{r}_A) - 2.0\ \dot{\underline{r}}_A$$

$$\underline{b}_0 = \underline{r}_M$$
$$\underline{b}_1 = \dot{\underline{r}}_M$$
$$\underline{b}_2 = 4.0\ (\underline{r}_B - \underline{r}_M) - 2.0\ \dot{\underline{r}}_M$$

Having $(\underline{a}_0, \underline{a}_1, \underline{a}_2)$ or $(\underline{b}_0, \underline{b}_1, \underline{b}_2)$, the coefficients of the polynomial to be solved are (here only the $\underline{a}$-s are quoted)

$$c_3 uu^3 + c_2 uu^2 + c_1 uu + c_0 = 0$$

$$c_3 = 2.0\ \underline{a}_2\ \underline{a}_2$$
$$c_2 = 3.0\ \underline{a}_2\ \underline{a}_1$$
$$c_1 = 2.0\ (\underline{a}_0 - \underline{P})\ \underline{a}_2 + \underline{a}_1 \cdot \underline{a}_1$$
$$c_0 = (\underline{a}_0 - \underline{P})\ \underline{a}_1$$

Finally we need to select the root for which $| \underline{P} - \underline{r}(uu) |$ is the smallest and $0 <= uu <= 0.5$.

6. APPENDIX - 3

Numerical evaluation of Bspline and dq-curve fitting with different fit-parameters

INTERSECTION CURVE OF A DQ-SURFACE AND A PLANE

- open curve with 91 initial datapoints
- characteristics of different fitted curves

Fitting type	BSP	DQ-Ø	DQ-A_1	DQ-A_2	DQ-A_3	DQ-A_4	DQ-A_5
Number of segments	90	90	9	12	23	33	57
Calculation time	0.050	0.133	0.656	0.520	0.417	0.326	0.212
Dist. tolerance	-	-	0.001	0.0005	0.0001	0.00005	0.00001
Angle tolerance	-	-	1	0.5	0.1	0.05	0.01
S1-mean distance 10^{-5}	6.98	0.05	33.71	15.82	1.8	0.66	0.15
S1-largest distance 10^{-5}	87.65	1.95	151.99	58.52	10.33	4.38	2.24
S2-mean distance 10^{-5}	0	0	0	0	0	0	0
S2-largest distance 10^{-5}	0	0	0	0	0	0	0

INTERSECTION CURVE OF A DQ-SURFACE AND A CYLINDER

- closed curve with 156 initial datapoints
- characteristics of different fitted curves

Fitting type	BSP	DQ-Ø	DQ-A_1	DQ-A_2	DQ-A_3	DQ-A_4	DQ-A_5
Number of segments	156	156	16	21	44	64	119
Calculation time	0.203	0.270	1.336	1.226	0.847	0.686	0.513
Dist. tolerance	-	-	0.001	0.0005	0.0001	0.00005	0.00001
Angle tolerance	-	-	1	0.5	0.1	0.05	0.01
S1-mean distance 10^{-5}	1.37	0.07	18.87	6.26	0.58	0.2	0.08
S1-largest distance 10^{-5}	9.52	4.87	92.27	44.79	4.46	2.53	3.85
S2-mean distance 10^{-5}	2.74	0.03	24.12	10.34	1.1	0.4	0.06
S2-largest distance 10^{-5}	36.93	0.33	81.81	46.20	7.09	3.45	0.55

Notations: DQ-0: 0-version, DQ-A: adaptiv dq-fitting
S1 - first surface (DQ), S2 - second surface (plane or cylinder)

7. APPENDIX - 4

Double-quadratic intersection curves

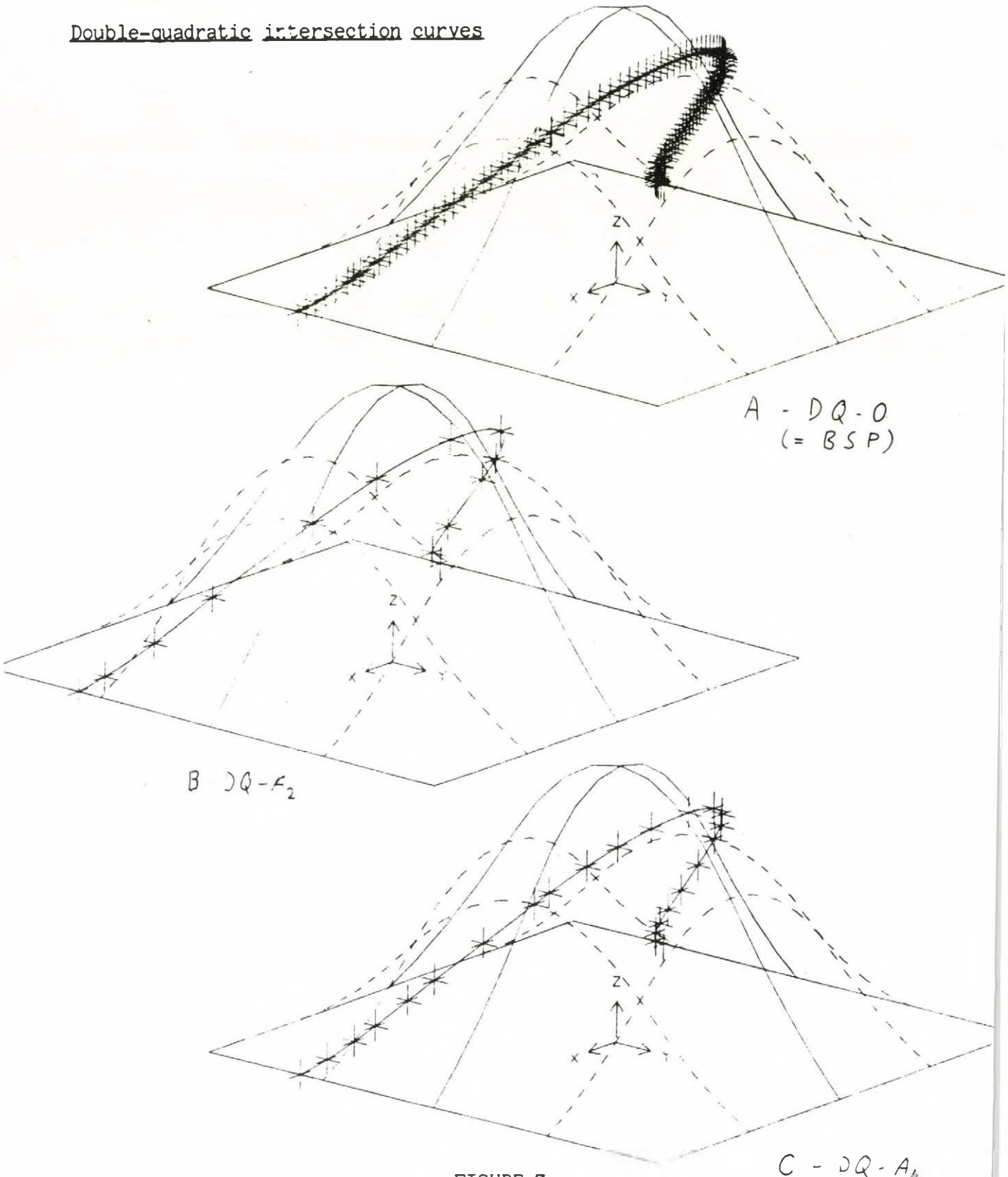


FIGURE 7.

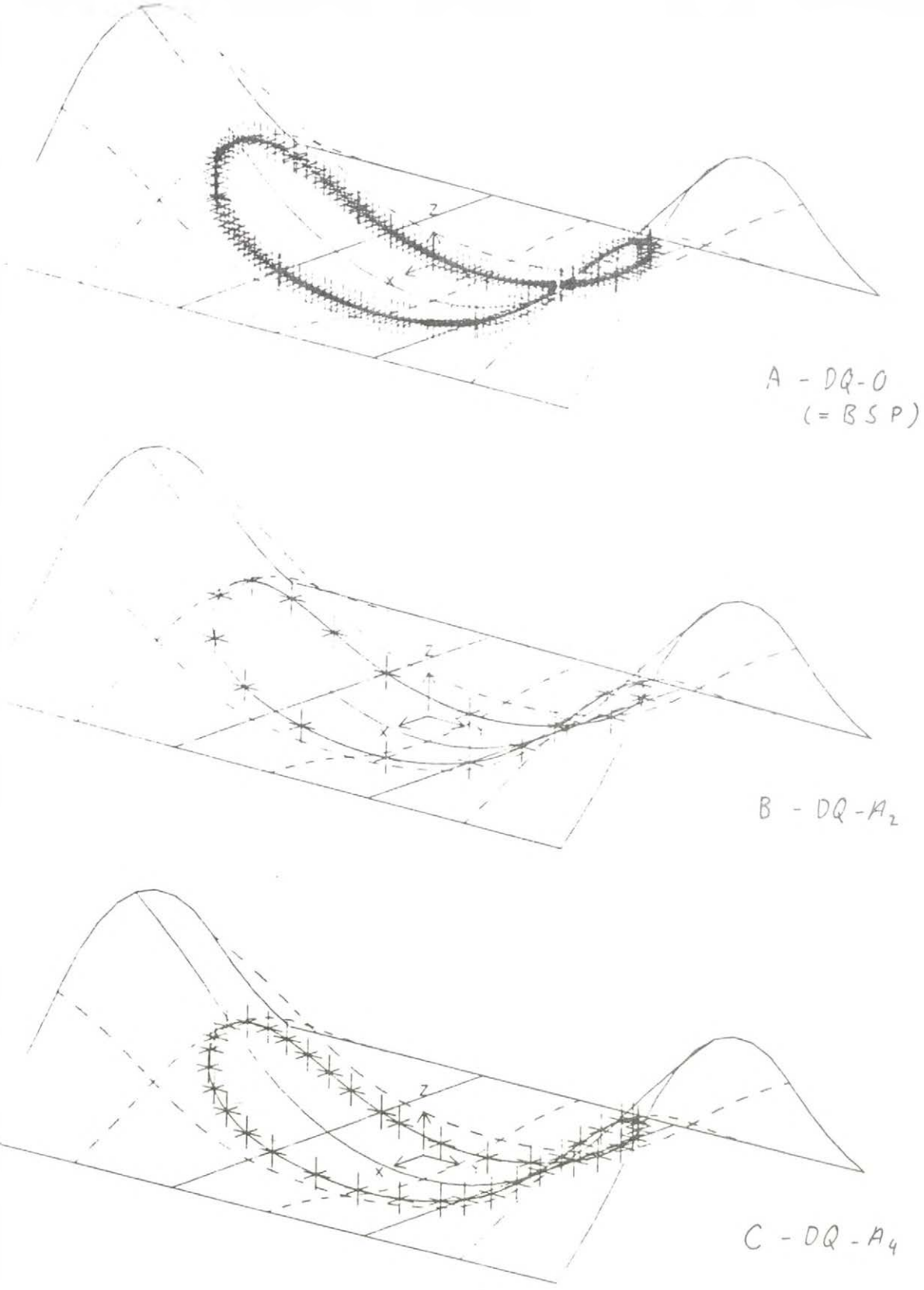


FIGURE 8.

Chapter V

ANALOGY BETWEEN GENERATING PLANAR INTERSECTION CURVES AND SILHOUETTE CURVES OF (DOUBLE-)QUADRATIC SURFACES

1. INTRODUCTION

Fast picture generation is a fundamental task in computer aided geometric modelling. Continuous visual control is needed when creating models of objects with complex geometry. With raster-scan terminals nice, shaded, hidden-surface pictures can be generated once the design is completed. However in the present state of the art of computer graphics these algorithms would demand enormuous computing power if they were to be used in interactive sessions. This is the main reason why line-drawings of objects still remain the primary output during interactive geometric design.

When only planar faces are involved the wire frame drawing of the object, possibly with hidden-line removal gives a satisfactory image. However, this is not the case when the object is bounded by curved faces, in addition to edges we need at least the so-called "silhouette curves" to fully understand the geometry of the object. Actually these curves locally separate the visible and non-visible parts of the object when it is viewed from a given eye-position.

To illustrate the above consideration a wireframe picture and the same with silhouette curves are presented in Fig.1 and 2. Without silhouettes we would not have any idea about the interior of the curved faces. It is noted, that the visual effect can be improved by adding other features to the drawing such as constant parameter lines or hatching lines, etc. (Fig.3 and 4) The final most satisfactory understanding of the object can be achieved by means of its orthogonal views (Fig.5).

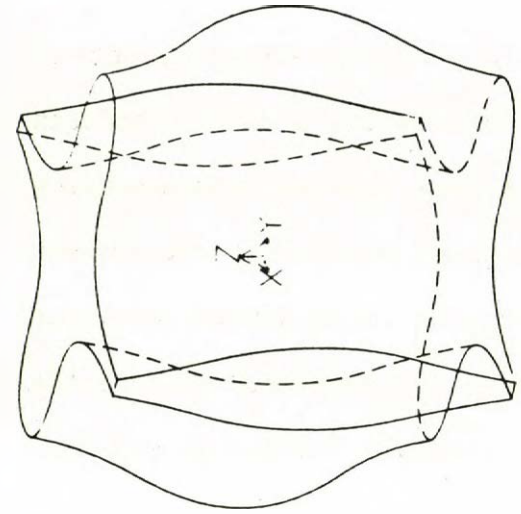


Figure 2

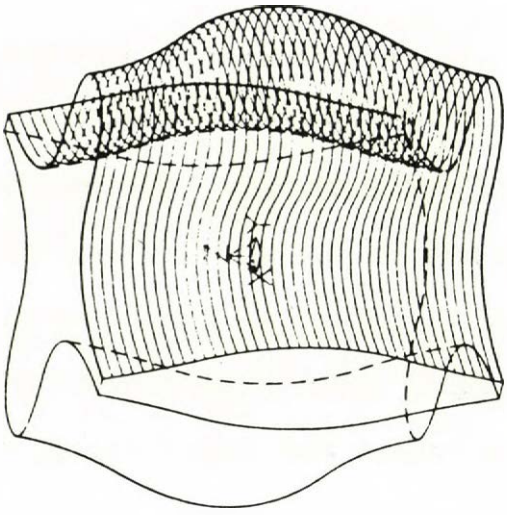


Figure 4

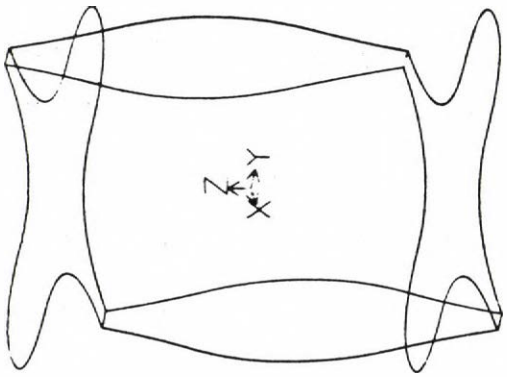


Figure 1

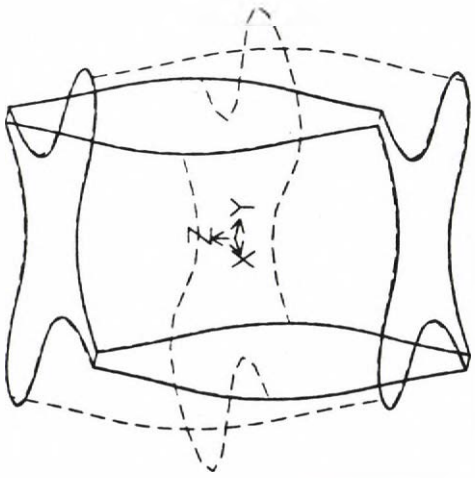


Figure 3

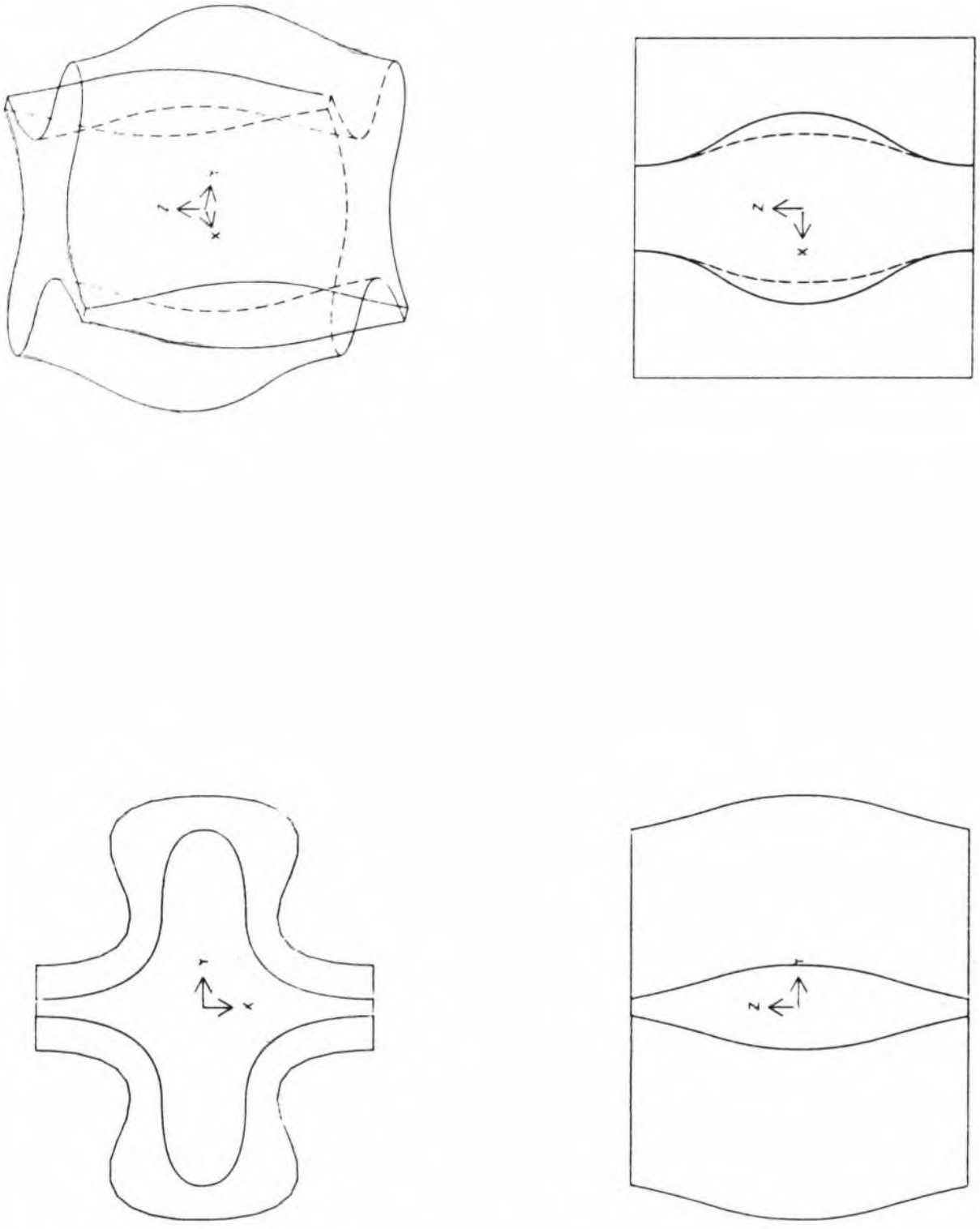


Figure 5

The double-quadratic curves and surfaces were chosen in BUILD primarily due to their simplicity in computing geometric interrogations and intersections - see previous chapters. This simplicity can also be found in the process of generating silhouette curves. In this chapter only the property of double--quadratic patches that they are made up of biquadratic patches will be used.

The basic steps of the silhouette algorithm will be introduced through an analogy between generating planar intersection curves and silhouette curves of biquadratic patches. The pieces generated for the individual patches are concatenated at a higher level. In the last section some further problems of silhouette curve generation in a solid modeller will be described.

2. INTERSECTION OF A PLANE AND A BIQUADRATIC PATCH

The procedure of generating the intersection curves of a plane and a parametric biquadratic patch can be split into simple tasks as follows.

1. Make a global test with the current biquadratic patch to see whether it intersects the given plane at all. (The easiest way of doing this is to construct the Bezier type characteristic polyhedron of the patch and a box surrounding it and examine whether or not the plane cuts this box. This test is based on the convex-hull property of quadratic patches - see [32].)

2. If the test is positive, generate a sequence of intersection points lying on both surfaces, otherwise ignore that patch.

3. By means of some appropriate "sorting" algorithm, determine the sequence of these discrete points.

4. Finally, making the mapping from parametric plane to 3D space, use the local features of the surfaces to fit a good approximating curve through the intersection points.

The equation of a biquadratic patch can be given in the following form

$$\underline{r}(u,v) = [u]\,[B_2]\,[Q_2]\,[B_2]^T[v]^T \tag{2.1}$$

where $[u]=[\,1\ u\ u^2]$, $[v]=[1\ v\ v^2]$, $0 \le u,v \le 0.5$,

$$[B_2] = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 4 & 0 \\ 4 & -8 & 4 \end{bmatrix}, \qquad [Q_2] = \begin{bmatrix} \underline{q}_{00} & \underline{q}_{01} & \underline{q}_{02} \\ \underline{q}_{10} & \underline{q}_{11} & \underline{q}_{12} \\ \underline{q}_{20} & \underline{q}_{21} & \underline{q}_{22} \end{bmatrix}.$$

The B_2 matrix contains the quadratic Bezier coefficients, while Q_2 gives the points of the characteristic polyhedron, which can be easily calculated from the tangential parameters of the quadratic or double-quadratic patch.

1. The box test must be performed with the smallest box, which contains all the vector elements of the Q_2 matrix.

2. To get one point of the intersection curve we write $\underline{r}(u,v)$ in the following form:

$$\underline{r}(u,v) = (x(u,v),\ y(u,v),\ z(u,v)). \tag{2.2}$$

The equation of the plane is

$$A \cdot x + B \cdot y + C \cdot z + D = 0 \tag{2.3}$$

So, after substitution the equation

$$f(u,v) = A \cdot x(u,v) + B \cdot y(u,v) + C \cdot z(u,v) + D = 0 \tag{2.4}$$

is produced. f(u,v) is a second-degree equation in both u and v. By fixing one of the parameters (say u) the second degree equation can be easily solved for the other parameter (v). The obtained (u_0,v_0) pair gives one intersection point in the u-v parameter plane.

3. The "sorting" step is not strictly relevant considering the analogy to be drawn here. If the sorting is done in the parameter plane the necessary tangential direction can be calculated by determining the partial derivatives of f(u,v) - (2.4).

4. The 3D intersection points can be obtained by substituting the (u_0,v_0) pairs into the original equation (2.1). The actual intersection curve must be formed by using the local features of the surfaces (tangency, curvature). In most cases to calculate the tangent of the intersection curve is sufficient for the curve-fitting algorithm used (see Chapter IV.). In case of surface-surface intersection this can be obtained as the vector product of the surface normals at the given intersection points, assuming the surfaces are not tangential there. (This can be simply explained that all curves on the surface going through a surfacepoint including the intersection curve are orthogonal to the surface normal at that point.)

3. SILHOUETTE CURVE GENERATION TO A BIQUADRATIC PATCH

The procedure of generating the silhouette curve to a biquadratic patch can be split into simple tasks in the same way as in Section 2.

1. Test the current quadratic patch to see whether there is any silhouette curve within it.

2. If the test is positive, then generate a sequence of silhouette points, otherwise ignore that patch.

3. By means of some appropriate sorting algorithm, determine the sequence of these discrete points.

4. Finally, after making the mapping from parametric plane to 3D space, use the local properties of the surface and the silhouette curve to fit a good approximating curve through the silhouette points.

Given the viewing direction $\underline{w} = (wx, wy, wz)$. The points on the surface and also their inverse mapping to the u-v parametric plane can be divided into "towards" and "away" regions. In the first region the normal vectors point towards the viewing position, that is their projections to the viewing vector are negative, while in the second the normals point away and their projections are positive. The boundary curves between the two regions give the silhouette curves. For these points the projection of the normal vector to the viewing vector is a nullvector.

Formulating this mathematically - at a silhouette point the scalar product of the normal vector ($\underline{n}(u,v)$) and the viewing vector ($\underline{w}$) must be zero and this expression must also change its sign. This results again in an implicit scalar function with two variables as in the case of the previous intersection problem.

$$g(u,v) = \underline{n}(u,v) \cdot \underline{w} = 0 \tag{3.1}$$

where

$$\underline{n}(u,v) = \dot{\underline{r}}_u(u,v) \times \dot{\underline{r}}_v(u,v) = \begin{bmatrix} i & j & k \\ x_u & y_u & z_u \\ x_v & y_v & z_v \end{bmatrix} \tag{3.2}$$

Fortunately $\underline{n}(u,v)$ in case of quadratics is only a third degree equation in both u and v. Consequently $\underline{n}(u,v)$ can be considered as a vector function describing a bicubic patch, while $\underline{w}$ can be interpreted as the normal vector of a plane going through the origin (D=0). Thus, the silhouette line problem is equivalent to determining the intersection curves of a bicubic patch and a plane.

1. The above analogy makes it easy to find out whether there is any silhouette curve within the current patch. $\underline{n}(u,v)$ can be written in the form of

$$\underline{n}(u,v) = [u]\,[N]\,[v]^T \tag{3.3}$$

where $[u] = [\,1\ u\ u^2\ u^3]$, $[v] = [1\ v\ v^2\ v^3]$, $0 <= u,v <= 0.5$,

as we did in Section 2. N contains the vector coefficients by powers in u and v as obtained from the (3.2) vector product. Now if N is considered as the matrix of a bicubic patch, then its Bezier-type characteristic polyhedron can also be calculated in an inverse way. That is

$$[N] = [B_3][Q_3][B_3]^T \tag{3.4}$$

where

$$[B_3] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -6 & 6 & 0 & 0 \\ 12 & -24 & 12 & 0 \\ -8 & 24 & -24 & 8 \end{bmatrix}$$

(Note the parametric interval [0,0.5] is used.)

With the known matrix N, Q_3 which contains the points of the corresponding characteristic polyhedron can be calculated as follows:

$$[Q_3] = [B_3^{-1}][N][B_3^{-1}]^T. \tag{3.5}$$

The inverse of the Bezier matrix is

$$[B_3^{-1}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/6 & 0 & 0 \\ 1 & 1/3 & 1/12 & 0 \\ 1 & 1/2 & 1/4 & 1/8 \end{bmatrix} \tag{3.6}$$

1. Having determined the vectors of the characteristic polyhedron the smallest box containing all of them can be made and the box cuts plane test can be executed.

2. The calculation of a silhouette point is done in a similar way as in Section 2. The only difference is that g(u,v) (see (3.2)) is a third degree equation in both u and v. By fixing one of the parameters (say u) a third degree equation must be solved for the other parameter (v). The resulting (u_0,v_0) pair will represent one silhouette point in the u-v plane.

3. The same argument holds as was described in point 3. of Section 2.

4. To get the tangent of the silhouette curve at a given point is more complicated. Let us suppose that we are at the (u_0,v_0) silhouette point in the parametric plane. Substitution into the patch equation (2.1) will give the point, which the silhouette curve goes through. The equation of the tangential line there can be simply calculated:

(3.7) $$\alpha u + \beta v + \gamma = 0$$

where

$$\alpha = \dot{g}_u(u_0,v_0)$$

$$\beta = \dot{g}_v(u_0,v_0)$$

$$\gamma = -\dot{g}_u(u_0,v_0)u_0 - \dot{g}_v(u_0,v_0)v_0 \quad .$$

Note that if both derivatives vanish, the second derivatives define the tangential direction, if it exists at all at (u_0,v_0).

In the vicinity of (u_0,v_0) the g(u,v) curve can be approximated by its tangential line. Let us introduce an s silhouette parameter for which

(3.8) a) if $\beta \neq 0$ at (u_0,v_0) than $s = u$

b) else $\alpha \neq 0$ at (u_0,v_0) than $s = v$.

Expressing u and v as a function of s we obtain

a) $u = s, \qquad v = (-\alpha/\beta)\, s - (\gamma/\beta)$

b) $u = (-\beta/\alpha)\, s - (\gamma/\alpha), \qquad v = s$

The 3D silhouette curve denoted $\underline{p}(s)$ can be well approximated by $\bar{\underline{p}}(s)$ in th vicinity of $\underline{r}(u_0,v_0)$.

(3.9) $$\underline{p}(s) \cong \bar{\underline{p}}(s) = \underline{r}(u(s),v(s))$$

At (u_0,v_0) it holds that $\underline{p}(s) = \bar{\underline{p}}(s)$ and $d\underline{p}(s)/ds = d\bar{\underline{p}}(s)/ds$.

Using the chain rule, the silhouette tangent is

(3.10) $$\frac{d\bar{\underline{p}}(s)}{ds} = \frac{\partial \underline{r}(u,v)}{\partial u} \cdot \frac{du}{ds} + \frac{\partial \underline{r}(u,v)}{\partial v} \cdot \frac{dv}{ds}$$

That is

$$\text{a)} \quad d\bar{p}/ds = \dot{r}_u - (\alpha/\beta)\,\dot{r}_v = (\dot{r}_u\beta - \dot{r}_v\alpha)/\beta$$

$$\text{b)} \quad d\bar{p}/ds = -(\beta/\alpha)\,\dot{r}_u + \dot{r}_v = (\dot{r}_u\beta - \dot{r}_v\alpha)\,/\,(-\alpha).$$

Therefore either case is taken the direction of the tangential line to the silhouette curve can be computed as

$$p_0 = (\dot{r}_u\beta - \dot{r}_v\alpha)\,/\,|\dot{r}_u\beta - \dot{r}_v\alpha| .$$

(The actual unit tangent can be either p_0 or $-p_0$ depending on the sequence of the silhouette points and the choice of the s silhouette parameter.)

The silhouette curve generation for double-quadratics is relatively fast, since the computation of a silhouette point means only the solution of a third degree equation. We do not have to use numerical methods as, for example, in case of bicubic patches, where the normal vector is given by a fifth degree equation in u and v. In BUILD a general sorting algorithm [70] and a general adaptive curve fitting algorithm [73] have been implemented, which perform the concatenation of the pieces of curve of the individual biquadratic patches. This is why, once the (u_0,v_0) values and the tangents in the u-v plane and in 3D are supplied, the above algorithms work without knowing whether a surface-surface intersection or a silhouette generation is to be performed.

4. FURTHER PROBLEMS IN CONNECTION WITH SILHOUETTE CURVE GENERATION

In the previous section we presented the principle of generating "mathematical" silhouette curves. Unfortunately, there are many special cases which we cannot discuss here, only mention them:

- a biquadratic patch can become a degenerate planar patch, where all the points may be silhouette points,

- a biquadratic patch can contain a straight line parallel to the viewing direction, where there may be real silhouette points, although the normal does not change its sign,
- the silhouette curves can be tangentially discontinuous across the patch boundaries (this can be a problem at sorting or curve-fitting)
- etc.

All these special cases must be treated carefully.

To use the "mathematical" silhouette curves to draw pictures in a volume modeller several further considerations apply. The two most important will be briefly described.

Open silhouette curves may occur as several pieces, closed ones definitely do, according to the change of normal vectors crossing them. Viewing the silhouette curves from a given eye-position, if the normals change from "towards" to "away" the term "visible silhouettes" is used, while in the opposite case they are termed "non-visible". The latter case means, that this piece is covered by a non-visible region on the nearest side of the silhouette curve with respect to the viewing position. Of course the visible silhouettes are not necessarily visible from the viewpoint, they may be hidden by other faces of the object in between them and the viewing position.

There are points where a visible silhouette turns into non-visible or vice-versa. These are the so-called "silhouette breakpoints" and the mathematical silhouette curve must be cut at these (Fig.6). At the breakpoints the tangent to the silhouette curve is parallel to the viewing vector as shown in Fig. 7, they can be found directly if the silhouette curves are approximated by (double-) quadratic curves.

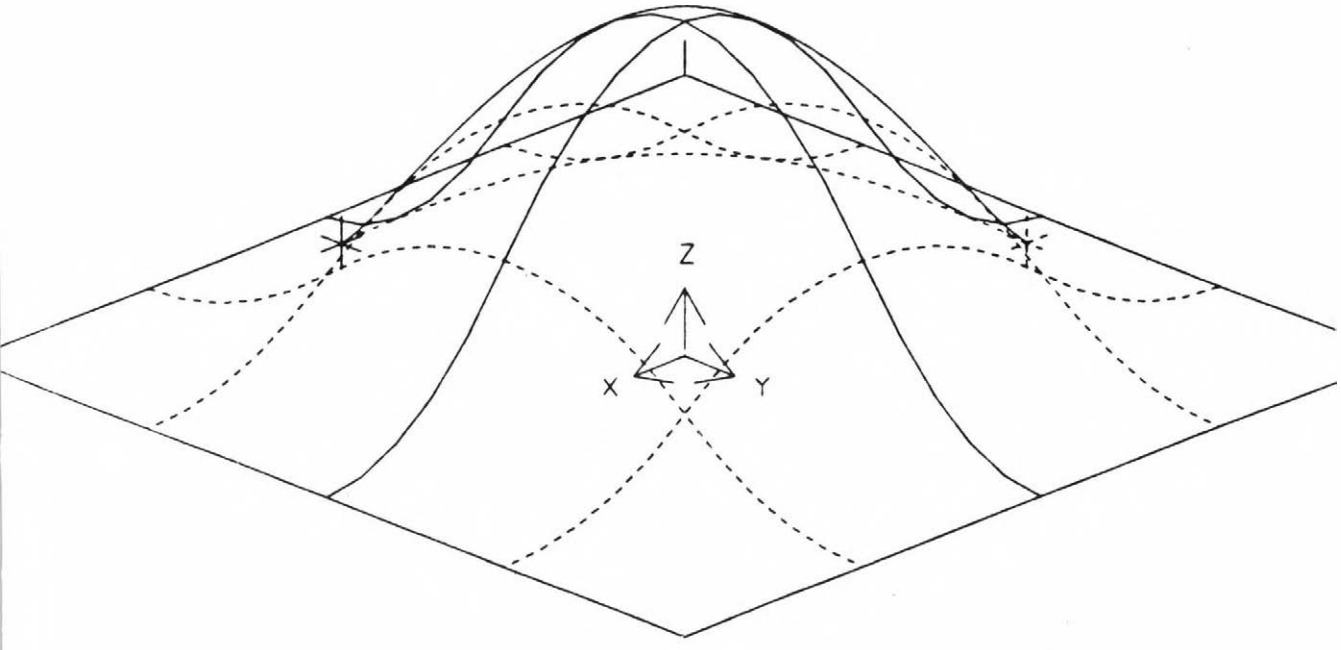


Figure 6

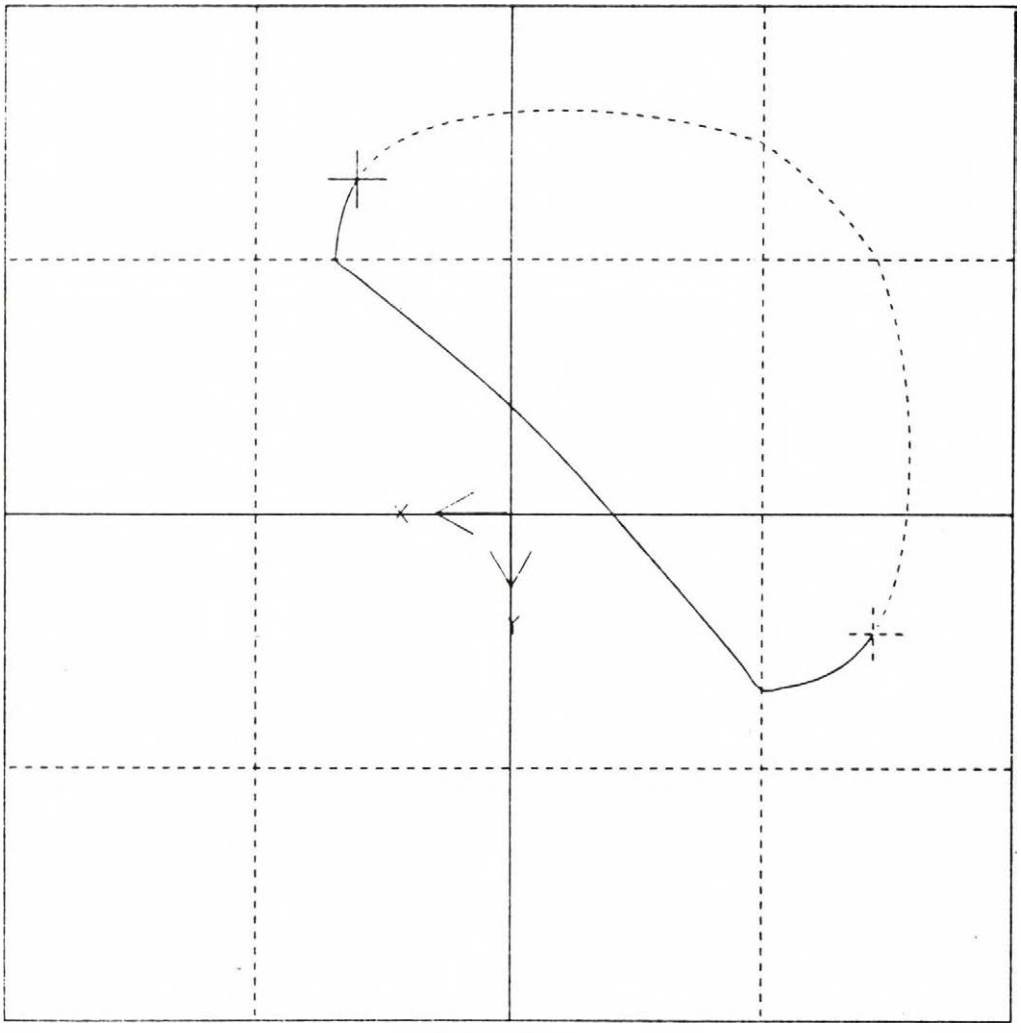


Figure 7

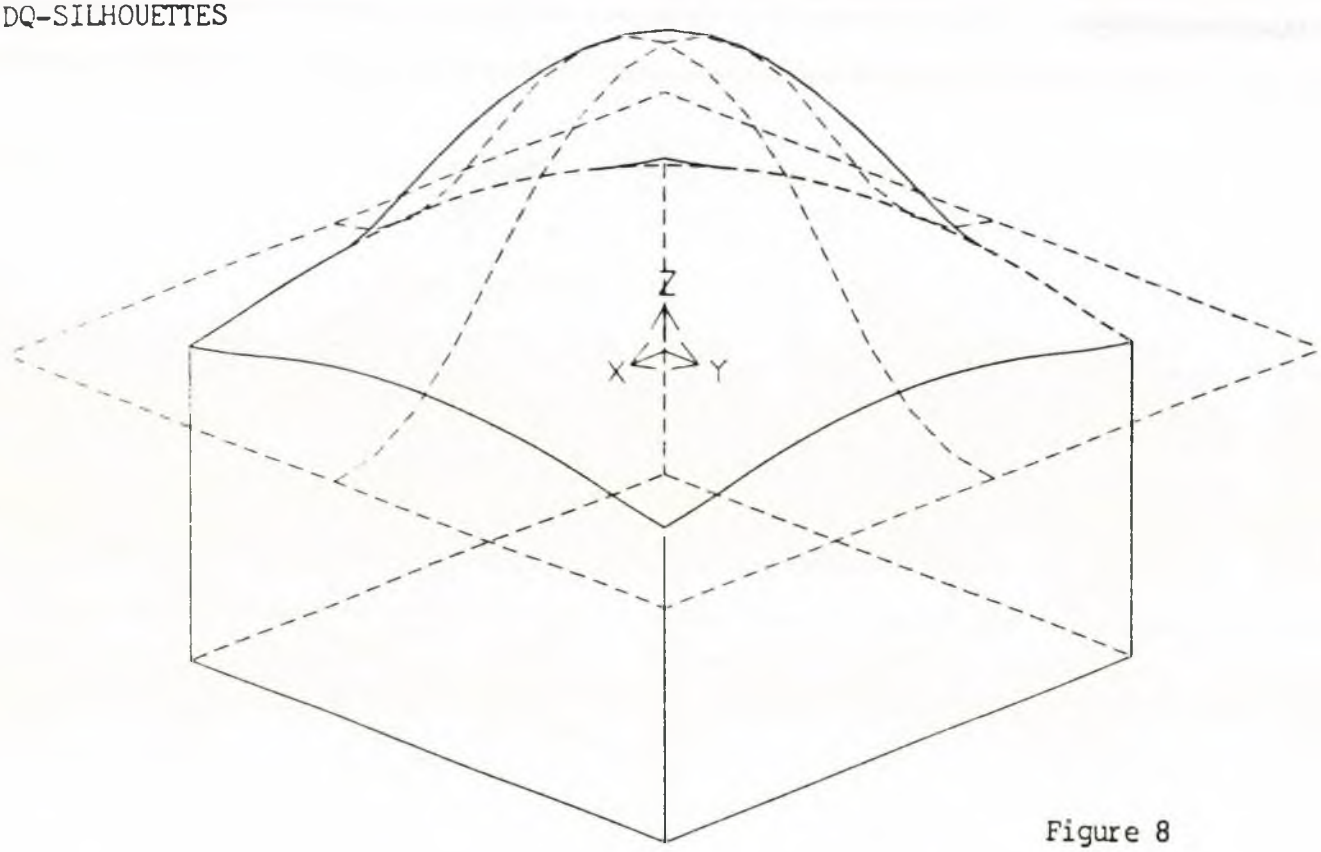


Figure 8

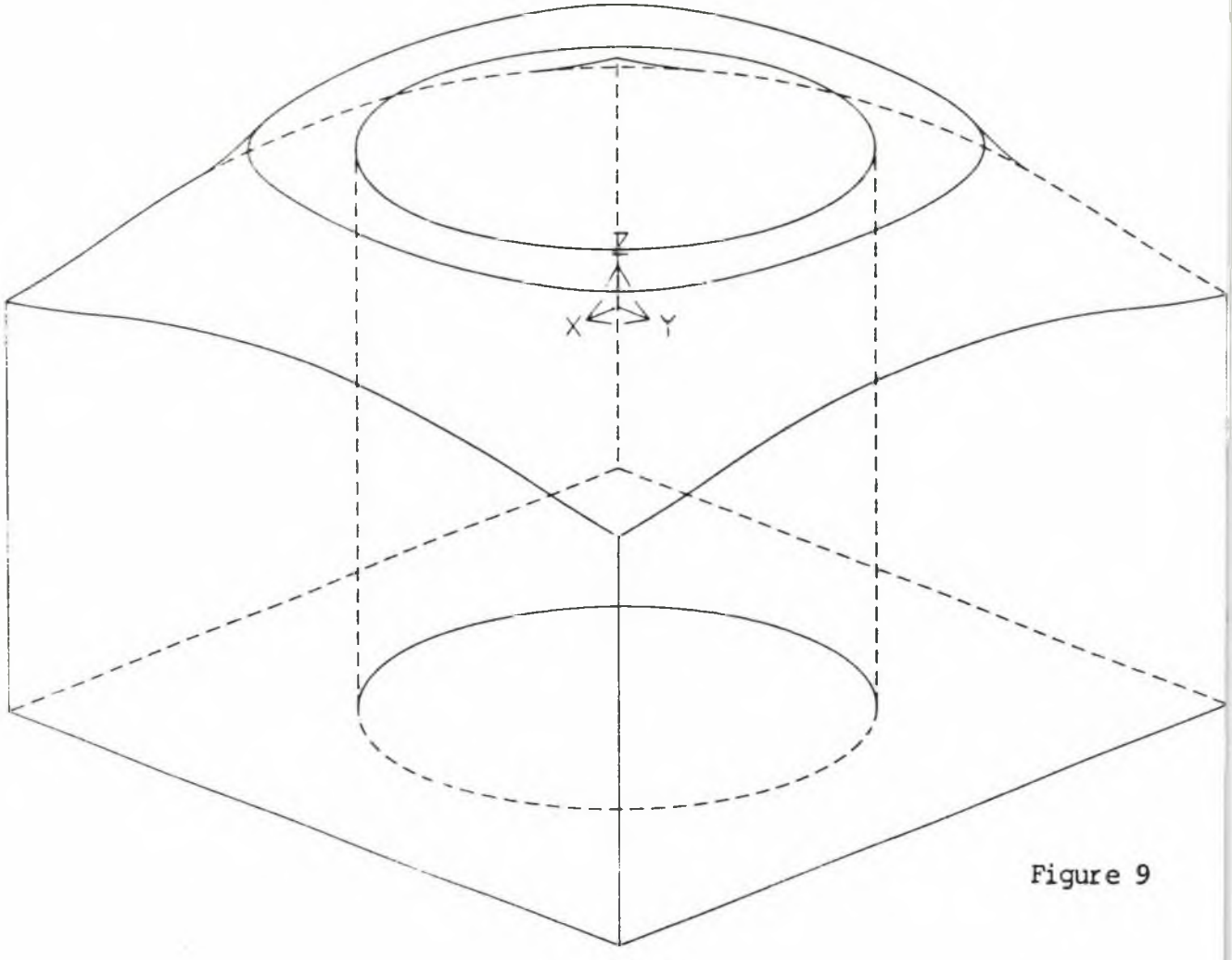


Figure 9

The tangent of a quadratic piece (4.1) must be parallel to the viewing vector ($\underline{w}$).

(4.1) $$\underline{r}(t) = \underline{a}_2 t^2 + \underline{a}_1 t + \underline{a}_0$$

(4.2) $$\dot{\underline{r}}(t) = 2 \underline{a}_2 t + \underline{a}_1 = k\underline{w}$$

After multiplying by $k\underline{w}$ we obtain

(4.3) $$t = -(\underline{a}_1 \underline{w}) / (2 \underline{a}_2 \underline{w})$$

The t value represent a real breakpoint if it falls into the given [0,0.5] parametric interval.

The other problem to be mentioned here is as follows. Up to now we have only talked about generating silhouette curves for a (double-)quadratic surface. However the faces in a solid model of an object are bounded regions lying on the dq-surfaces. Consequently the silhouette curve may need to be further divided into those parts which lie inside the region of the surface covered by the face. This is done by face vs. silhouette curve intersection to avoid tolerance problems. Having computed the points where the silhouette curve crosses the edges of the face, the segments of the curve between the crossing points are examined to see whether that interval is outside the body (Fig.8 and Fig.9). In these figures pieces of the silhouette curves of the top surface have been removed where they cross the top face and outside the vertical sides of the object.

Chapter VI

CONCLUSIONS

CONCLUSIONS

1.Free-form BUILD objects - some engineering examples

In this section some typical examples are given to illustrate the use of the free-form design facilities of the BUILD solid modeller. The operations used were outlined in Chapter I, the mathematical and computational background in the following ones.

The active surface of the mould in Fig. 1.1 was defined by two profiles. After making a blended surface and a DQPRISM from it, the object was subtracted from a block together with two cylinders resulting in the final part.

The object in Fig.1.2 is based on a 'two blocks - cylinder' part. (a) shows the original solid and the dq-surface. In (b) the result object can be seen after sectioning with this surface. (c) was generated by reflecting the whole object in the plane of the back face. To obtain the counterpart of this simple stamping-die (d), the 'positive' half was subtracted from a block.

The four-leg rib in Fig.1.4 was created using a DQOFFSET primitive as can be seen in Fig. 1.3. One leg was designed by sectioning the offset primitive by two perpendicular planes. After the two reflections the cylinder in the middle was added (Fig. 1.4).

The duct-like object in Fig. 1.5 was formed by two identical profiles, one end rotated by 90 degrees. The closed surface was transformed to be a solid using the DQCYLINDER command.

a

b

c

d

Figure 1.1

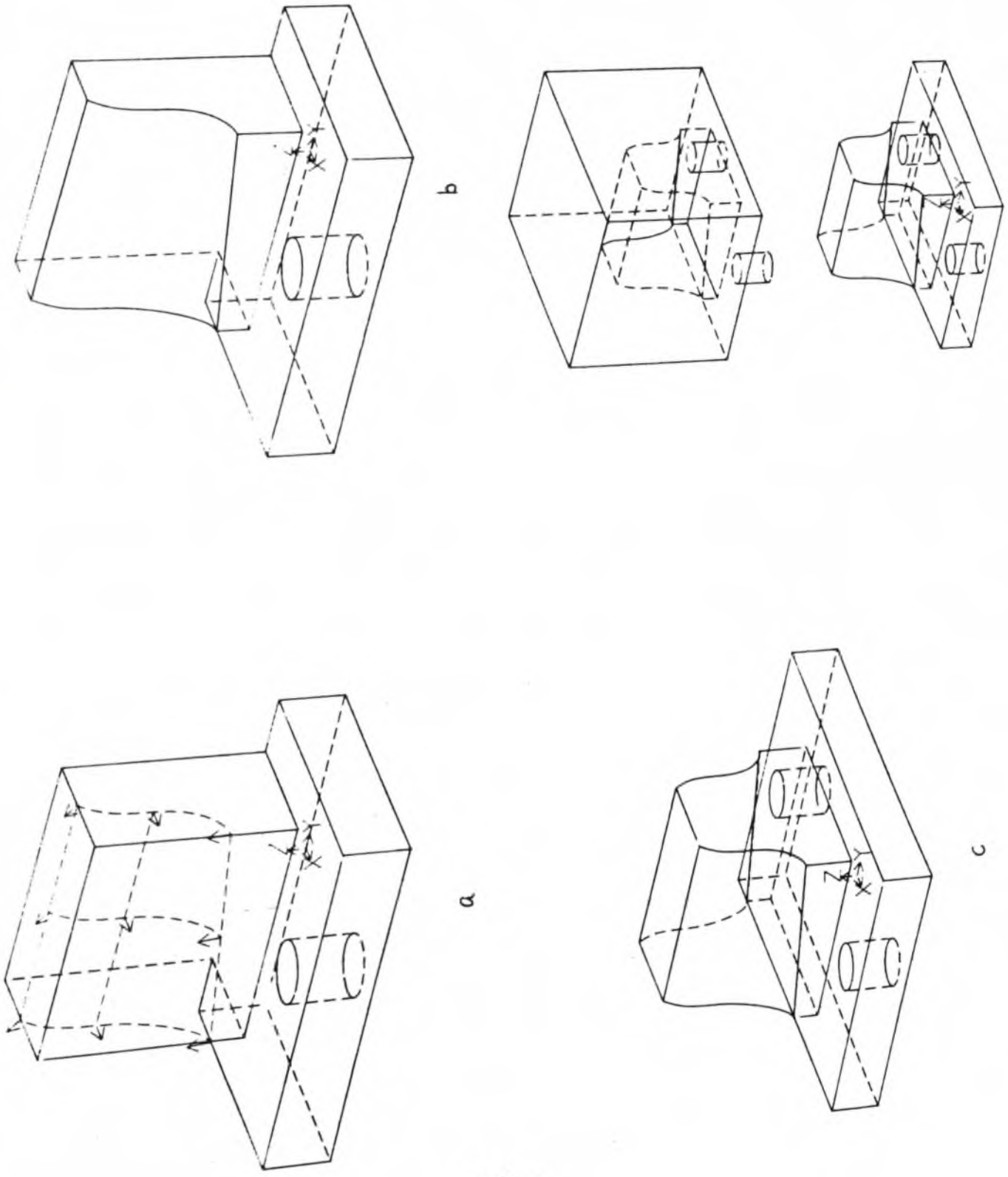


Figure 1.2

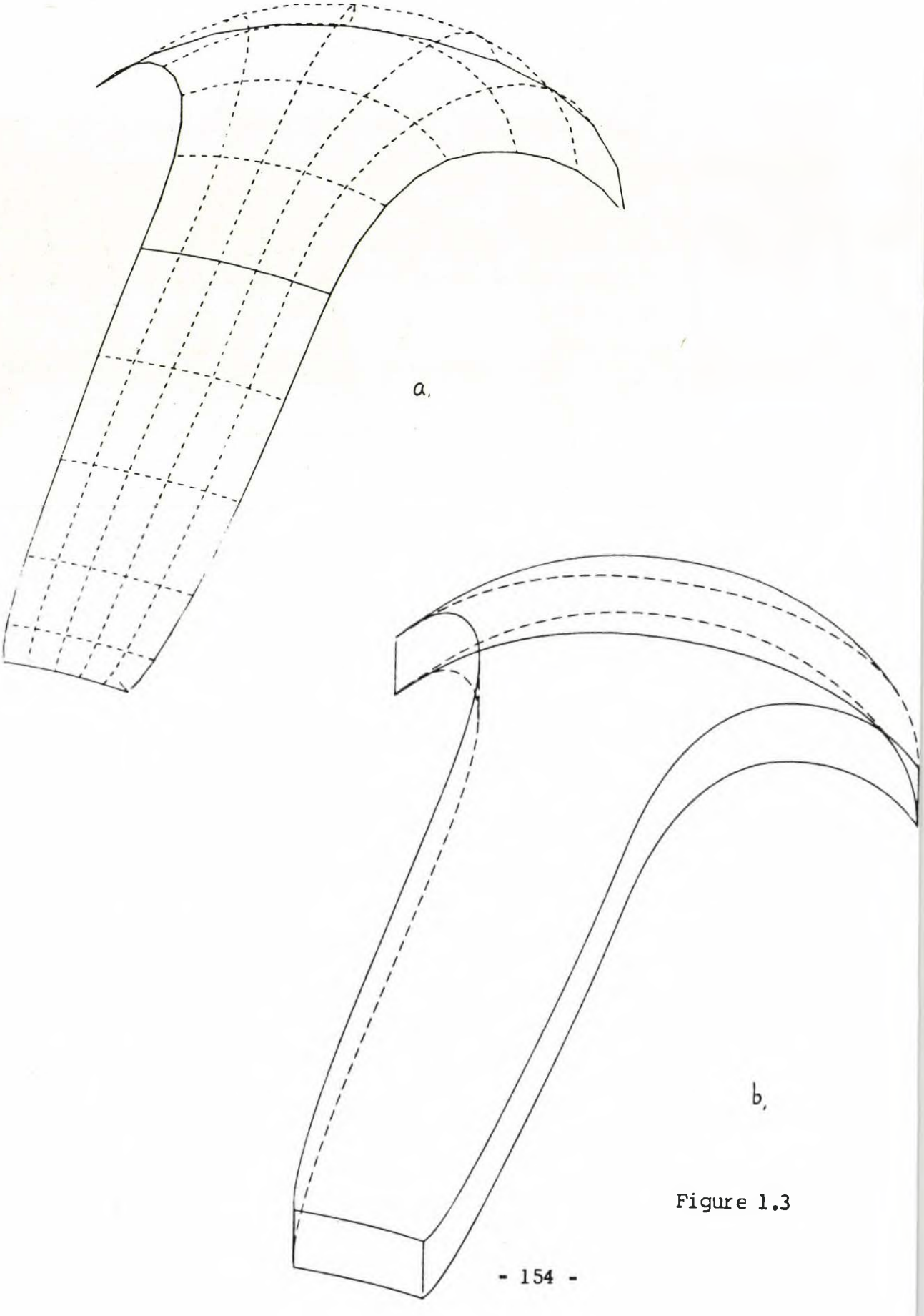


Figure 1.3

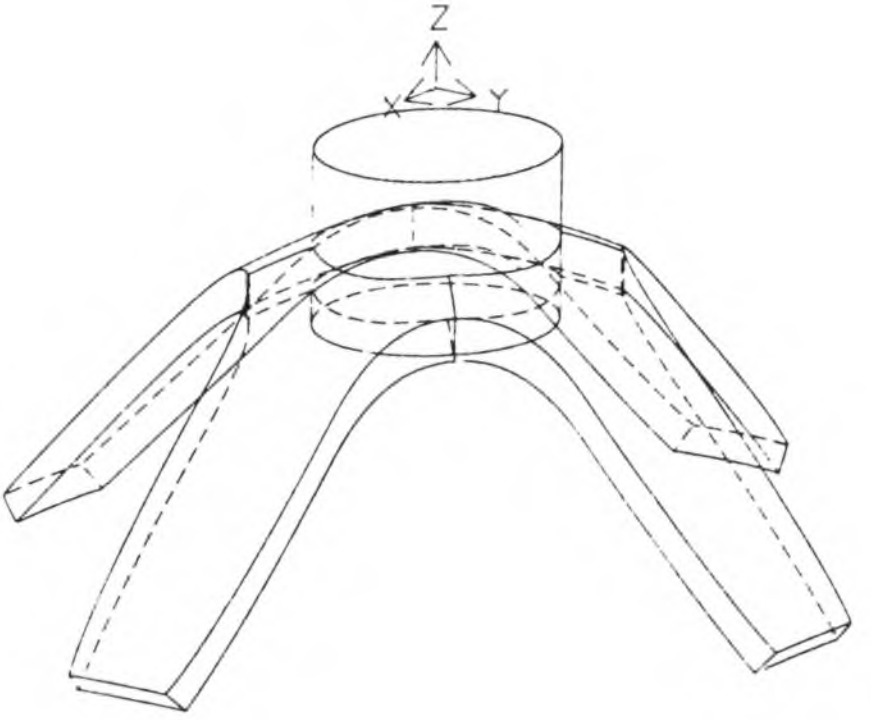


Figure 1.4

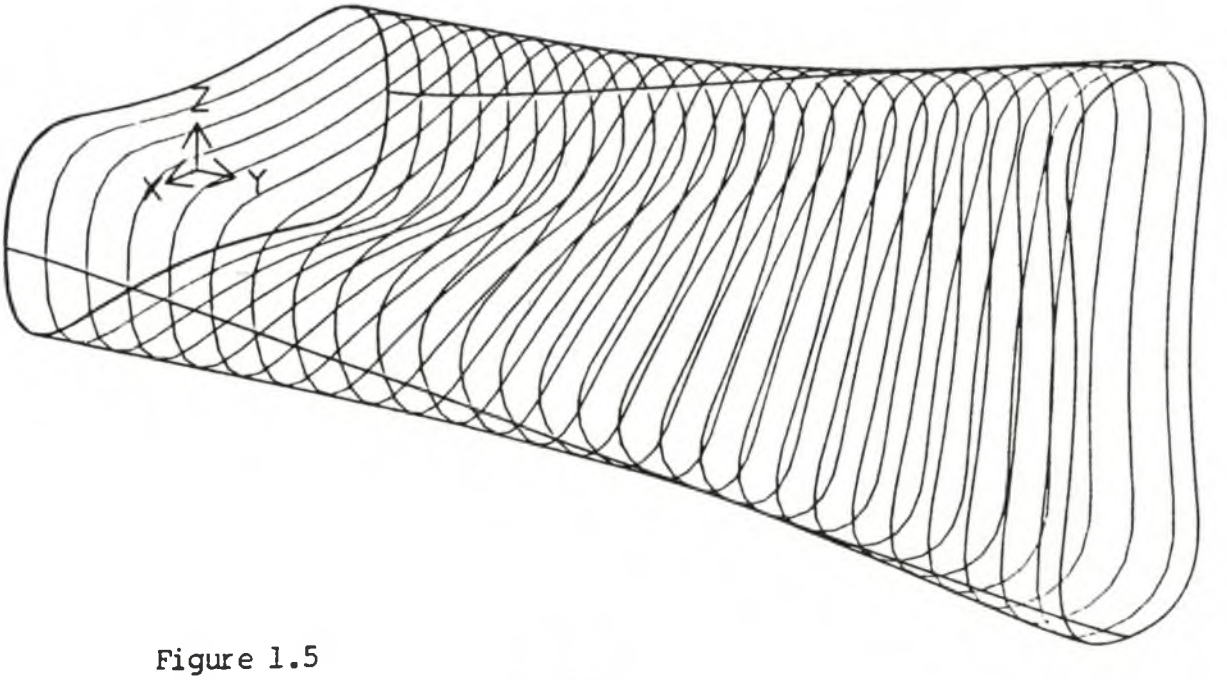


Figure 1.5

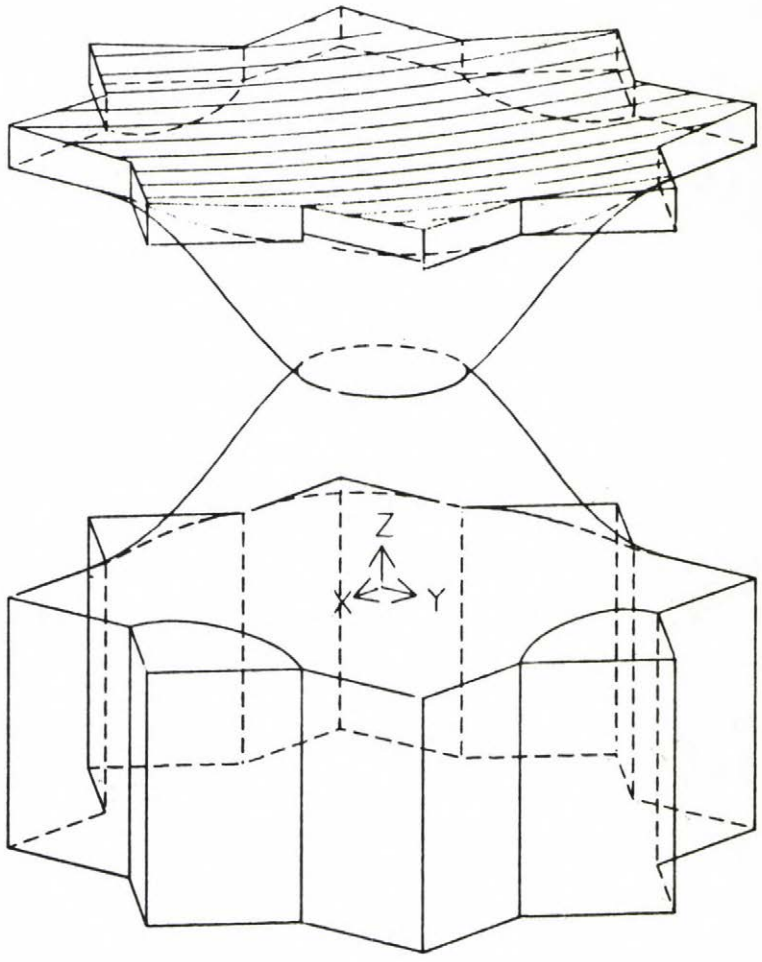
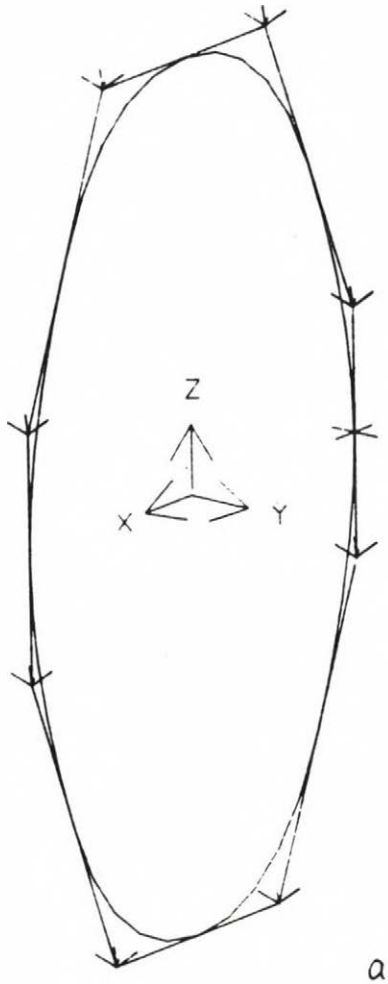
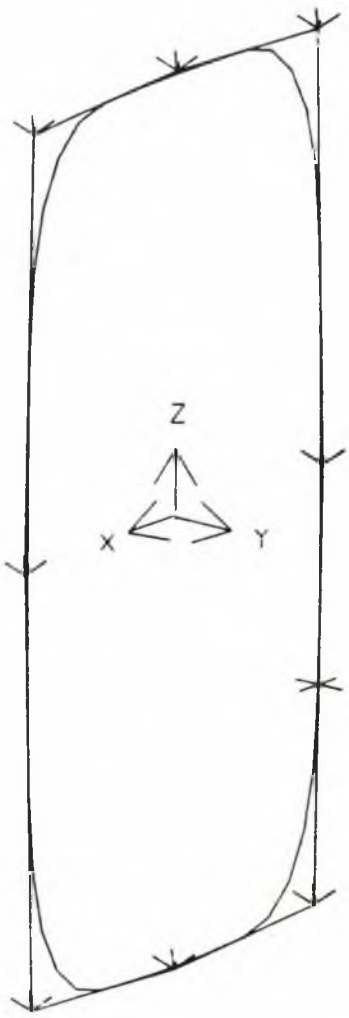


Figure 1.6



a,



b,

Figure 1.7

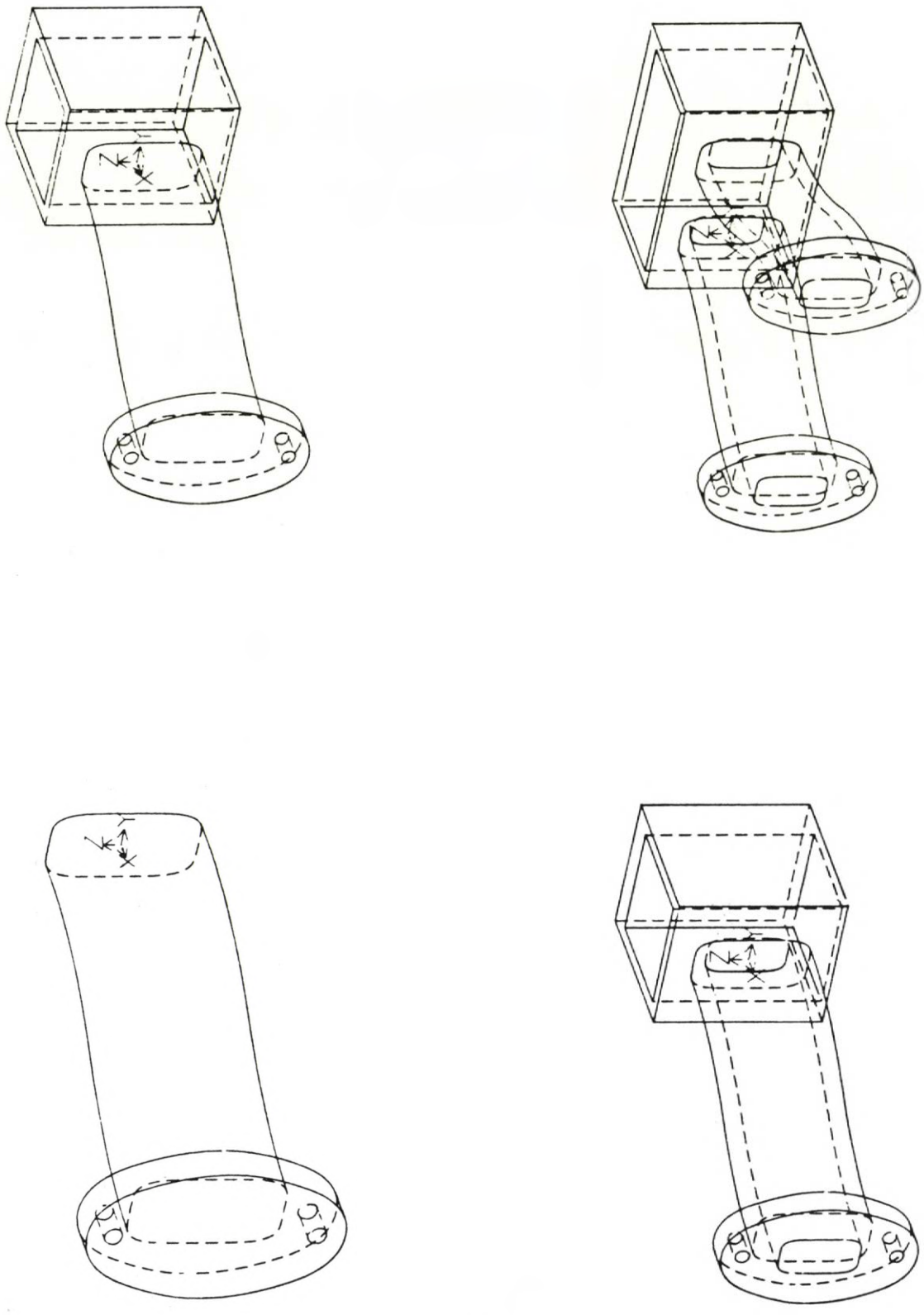


Figure 1.8

Figure 1.9

The bottom of the object in Fig.1.6 was created by adding two DQPRISM-s, having a "humpy" surface on the top face. After replicating and reflecting this, another "add" to the original solid was performed. Finally, a free-form sectioning surface cut off the missing part of the top.

A swept plate of a B-spline profile with two holes was added to the duct (DQCYLINDER), defined by another closed B-spline curve. (The profiles can be seen in Fig. 1.7, while the plate+duct object in Fig. 1.8 (a).) After adding this to a cube, the "material removal" was carried out by substracting a reduced cube and the offsetted internal ducts. Removing the right side of the box and reflecting the solid led to the final shape of this exhaust manifold like object (Fig. 1.9).

2.Suggestions for further research

Concerning the integration of double-quadratic curves and surfaces into solid modells, basically there are three directions to move towards to extend the present results. The most obvious would be the use of rational double--quadratic blending functions. In this way, the original advantageous features, such as second-degree polynomials, planar curve-segments, which are important for the geometric interrogations could be preserved, moreover, extra degree of design freedom could be gained. It must be emphasized that the algorithms described in this thesis can be generalized to rational dq-s, making formal or slight modifications.

CONCLUSIONS

As is well-known, rational quadratic curves can represent all conic curves, rational quadratic surfaces include all quadric surfaces. Thus lines, planes and rational double-quadratics could give a full coverage to represent precisely the conventional and free-form parts of mechanical engineering design products.

The second important extension would be the introduction of 3-sided (or generally n-sided) dq-patches. The 3-sided dq-patch can be constructed according to the known techniques described by Sabin [61], Barnhill [2], and Charrot and Gregory [19]. The use of n-sided patches leads us to the problem of handling and joining together free-form surfaces with general topology, which is a recent topic in surface modelling.

Last, but not least, the author's belief is that geometric modelling is in the state of transition. The next generation modellers will provide new types of user-handles for design, gradually approaching the present engineering expectations. One aspect of this is the automatic generation of composite free-form shapes, this explains the importance of developing more complex and more flexible operations for integrating free-form geometry into solid models. At the moment, we know only a subset of what is beleived can be done, however, it is hoped that using the general geometric interface concept, the introduction of new operations will result in new problems, not in the internal model-building field, but rather in the area of appropriately specifying the engineering requirements for the modellers.

3. Summary

Today, the integration of free-form surfaces into solid modelling is one of the most challenging problems of computer aided geometric design. The difference in the mathematical equations used for representing parametric and simple analytic surfaces resulted in totally different techniques in surface and solid modelling. The BUILD double-quadratic project examined the full range of the free-form vs. solid synthesis problem. New operations for integrating free-form surfaces into solid models and new techniques for interrogating solid models with free-form geometry have been developed. The solution choosen is based on the so-called double-quadratic curves and surfaces. Dq-s represent a good compromise between geometrical complexity and computational simplicity, as was shown in the first part of the thesis. The above mentioned operations and geometric interrogations can be performed, efficiently using them. All operations in BUILD are performed using a central geometric interface for obtaining information of individual or pairs of geometric elements. This thesis covered the geometrical, the relating algorithmical and computational aspects of this central nucleus, where dq-curves and dq-surfaces were involved.

The BUILD double-quadratic project, which was based on the results presented here, have proved that the free-form elements can be handled exactly in the same way as the conventional ones in a solid modeller with boundary representation. Using dq-s the computation time needed for combining free-form solids remained in the same range as that for bodies with quadric surfaces. Reasonable response times were experienced, even in interactive sessions, where complex free-form parts were created.

CONCLUSIONS

The results of the double-quadratic project have been presented and welcomed at several international forums. According to the best knowledge of the author, in early 1984 the solution chosen was unique compared to other modelling systems, which attack this synthesis problem of computational geometry.

REFERENCES

[1] Anderson, C.: The new BUILD User's Guide, CAD Group Document 116, Cambridge University Engineering Department, (1983)

[2] Barnhill, R.E.: Computer Aided Surface Representation and Design, in: Surfaces in Computer Aided Geometric Design, (Barnhill and Boehm eds.), North Holland, (1983)

[3] Bernard,F.: CATIA: du dessin au volume, de la cinematique aux calculs scientifiques, de la commande numerique a la robotique, un outil complet de CFAO pour la mechanique, in: Proc. MICAD 84, Paris, (1984)

[4] Bezier, P.: Numerical Control: Mathematics and Applications, Wiley, (1972)

[5] Bezier, P.: Mathematical and Practical Possibilities of UNISURF, in Computer Aided Geometric Design, (Barnhill and Riesenfeld eds.), Academic Press, (1974)

[6] Bishop, A.W.: ROMULUS 2 and its Role in CAE, in: Proc. MICAD 84, Paris, (1984)

[7] Bolton, K.M.: Biarc curves, Computer Aided Design, Vol. 7, No. 2, 89-92, (1975)

[8] Boyse, J.W. & Gilchrist, J.E.: GMSOLID: Interactive Modelling for Design and Analysis of Solids, IEEE Computer Graphics and Applications, March (1982)

REFERENCES

[9] Braid, I.C.: Designing with Volumes, PhD Thesis, Computer Laboratory, University of Cambridge, (1973)

[10] Braid, I.C.: On Storing and Changing Shape Information, Computer Graphics, Vol. 12, No. 3, (1978)

[11] Braid, I.C.: New directions in geometric modelling, CAD Group Document No. 98, Computer Laboratory, University of Cambridge, (1978)

[12] Braid, I.C., Hillyard, R.C. and Stroud, I.A.: Stepwise Construction of Polyhedra in Geometric Modelling, CAD Group Document No. 100, Computer Laboratory, University of Cambridge, (1978)

[13] Braid, I.C. and Hillyard, R.C.: Analysis of Dimensions and Tolerances in Computer Aided Geometric Design, Computer Aided Design, Vol. 10, No. 3, (1978)

[14] Braid, I.C.: Superficial Blends in Geometric Modelling, CAD Group Document No. 105, Computer Laboratory, University of Cambridge, (1980)

[15] Braid, I.C.: Geometric Modelling - Ten Years On, CAD Group Document No. 103, Computer Laboratory, University of Cambridge, (1979)

[16] Butterfield, K.R.: The development and applications of algorithms associated with surface representation, PhD Thesis, Brunel University, Uxbridge, Middlesex, (1978)

[17] Catmull, E. and Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes, Computer Aided Design, Vol. 10, No. 6, (1978)

REFERENCES

[18] Chaikin, G.: An algorithm for high-speed curve generation, Computer Graphics and Image Processing, Vol. 3, No. 3, (1974)

[19] Charrot, P. and Gregory, J.A.: A pentagonal surface patch for computer aided geometric design, in: Computer Aided Geometric Design, Vol. 1, (1984)

[20] Chiyokura, H. and Kimura, F: Design of Solids with Free-Form Surfaces, Computer Graphics Vol. 17, No. 3, (1983); Proc. ACM SIGGRAPH 83, Detroit, (1983)

[21] Cohen, E., Lyche, T. and Riesenfeld, R.F.: Discrete B-splines and Subdivision Techniques in Computer Aided Geometric Design and Computer Graphics, Computer Graphics and Image Processing, Vol. 14, No. 2, (1980)

[22] Cohen. E.: Some Mathematical Tools for a Modeler's Workbench, IEEE Computer Graphics and Applications, Vol. 3, No. 7, (1983)

[23] Coons, S.A.: Surfaces for Computer Aided Geometric Design of Space Forms, Report MAC-TR-41, Project MAC, M.I.T., (1967)

[24] Cox, M.G.: An algorithm for spline interpolation, Journal of the Institute of Mathematics and Applications, Vol. 10, (1975)

[25] dePont, J.J.: Essays on the Cyclide Patch, PhD Thesis, Cambridge University Engineering Department, (1984)

REFERENCES

[26] Design and manufacture of sculptured surfaces, (edited by Renner, G.), Working Study, Computer and Automation Institute, Hungarian Academy of Sciences, (1978) (in Hungarian)

[27] Diedenhoven, H. and Schunk, W.: Geometric Modelling with PROREN, Proc. 2nd CAM-I Geometric Modelling Seminar, Cambridge, Report No. p-83-GM--01, CAM-I, (1983)

[28] Donken, T.: A design system for products with sculptured surfaces, Proc. of CAD 84 Conf., Brighton, (1984)

[29] Doo, D.W.H.: A subdivision algorithm for smoothing down ireggular shaped polyhedrons, Brunel University, Uxbridge, Middlesex, (1978)

[30] Doo, D. and Sabin, M.A.: Behaviour of recursive division surfaces near extraordinary points, Computer Aided Design, Vol. 10, No. 6, (1978)

[31] Engeli, M.: Geometric Modelling of Complex Forms with EUKLID, in :Proc. 12th CIRP International Seminar on Manufacturing Systems, Belgrade, (1980)

[32] Faux, I.D. and Pratt, M.J. : Computational Geometry for Design and Manufacture, Ellis Horwood, Chichester, U.K., (1980)

[33] Ferguson, J.C.: Multivariate Curve Interpolation, Journal ACM, Vol. 11, No. 2, (1964)

[34] Forrest, A.R.: Curves and Surfaces for Computer Aided Geometric Design, PhD Thesis, University of Cambridge, (1968)

[35] Forrest, A.R. : Notes on Chaikin's Algorithm, Computational Geometry Memorandum, CMG-74-1, University of East Anglia, (1974)

[36] Gaal, B. and Varady. T.: Experiences and further development of the FFS (Free-Form Shapes) system, Proc. of the 9th IFAC World Congress, Budapest, (1984)

[37] Geisow, A.D.: Results of Benchmark Tests Using MEDUSA, in: Proc. 2nd CAM-I Geometric Modelling Seminar, Cambridge, Report No. P-83-GM-01, CAM-I, (1983)

[38] Goldstein, R. and Malin, L.: 3D Modelling with the SYNTHAVISION System, in: Proc. 1st Ann. Conf. on Computer Graphics in CAD/CAM Systems, Cambridge, Mass., (1979)

[39] Gordon, W.J. and Riesenfeld, R.F.: B-spline curves and surfaces, in : Computer Aided Geometric Design, (Barnhill and Riesenfeld eds.), Academic Press, (1974)

[40] Hanna, S.L., Abel, J.F. and Greenberg, D.P.: Intersection of Parametric Surfaces by Means of Look-Up Tables, IEEE, Computer Graphics and Applications, (1983)

[41] Hosaka,M. and Kimura, F.: Geometric modelling with interactive graphics, Proc. CAD in Mechanical Engineering, Politecnico di Milano, (1976)

[42] Jared, G.E.M. and Stroud, I.A.: Local operations in the BUILD system, in: Advances in CAD/CAM, Proc. PROLAMAT'82, Leningrad, North Holland, (1982)

[43] Jared, G.E.M. and Varady, T.: Synthesis of volume modelling and sculptured surfaces in BUILD, in: Proc. CAD 84 Conf., Brighton, (1984)

[44] Klosterman, A.L., Ard, R.H. and Klahs, J.W.: A geometric modelling program for the system designer, in: Proc. Conf. on CAD/CAM Technology in Mechanical Engineering, M.I.T Press, (1982)

[45] Kimura, F., Kawabe, S. and Sata, T.: A study on product modelling for integration of CAD/CAM, (to be published in Computers in Industry), (1984)

[46] Kyprianou,L.K.: Shape Classification in Computer Aided Design, PhD Dissertation, University of Cambridge, (1980)

[47] Levin, J.: Mathematical models for determining the intersections of quadric surfaces, Computer Graphics and Image Processing, Vol. 11, (1979)

[48] Lodwick, G.D. and Whittle, J.: A technique for automatic contouring field data, Australian Computer Journal, 2, (1970)

[49] Nutbourne, A.W.: A Cubic Spline Package, Part 2 - The Mathematics, Computer Aided Design, Vol. 5, No. 1, (1973)

REFERENCES

[50] Nykanen, M., Nystrom, M. and Katainen, A.: UNIBLOCK Modelling System, in: Proc. 2nd CAM-I Geometric Modelling Seminar, Cambridge, Report No. P-83-GM-01, CAM-I, (1983)

[51] Okino,N., et al: TIPS-1 Technical Information Processing System for Computer Aided Design, Drawing and Manufacturing,in: Proc. Prolamat'73, Budapest, (1973)

[52] Okino,N.: TIPS-1, in: Proc. 2nd CAM-I Geometric Modelling Seminar, Cambridge, Report No. P-83-GM-01, CAM-I, (1983)

[53] Parkinson, A.: An automatic NC data generation facility for the BUILD solid modelling system, CAD Group, Cambridge University Engineering Department, (1984)

[54] Peters, G.J.: Interactive computer graphics application of the parametric bicubic surface to engineering design problems, in: Computer Aided Geometric Design, (Barnhill and Riesenfeld eds.), Academic Press, (1974)

[55] Pratt, M.J. and Varady, T.: Design techniques for the definition of solid objects with free-form geometry, (to be published in Computer Aided Geometric Design Journal, 1985)

[56] Renner, G.: Conventional elements of engineering drawing - free-form curves, Proc. Eurographics'84, Copenhagen, North Holland, (1984)

REFERENCES

[57] Riesenfeld, R.F.: On Chaikin's Algorithm, Technical Note, University of Utah, (1974)

[58] Rockwood, A.P.: Introducing Sculptured Surfaces into a Geometric Modeller, in: Proc. General Motors Geometric Modelling Seminar, Detroit, (1983)

[59] Sabin, M.A.: A 16-point bicubic formulation suitable for multipatch surfaces, Report No. VTO/MS/155, British Aircraft Corporation, Weybridge, (1969)

[60] Sabin, M.A.: Interrogation techniques for parametric surfaces, Report No. VTO/MS/150, British Aircraft Corporation, Weybridge, (1968)

[61] Sabin, M.A.: The use of piecewise forms for the numerical representation of shape, Dissertation, Tanulmanyok No. 60, Computer and Automation Institute, Hungarian Academy of Sciences, (1976)

[62] Sabin, M.A.: Contouring - A review of methods for scattered data, Mathematical Methods in Computer Graphics and Design, (Edited by Brodlie, K.W.), Academic Press, (1980)

[63] Sabin, M.A.: Non-rectangular Surface Patches Suitable for Inclusion in a B-spline Surface, in: Proc. Eurographics'83, Zagreb, North Holland, (1983)

[64] Sederberg, T.W. et al: Implicit representation of parametric curves and surfaces, Computer Graphics and Image Processing, (1983)

REFERENCES

[65] Solomon, B.J.: Geomoetric Interrogations in Volume Modelling (?), PhD Thesis, University of Cambridge, (1984)

[66] Spur,G. et al: COMPAC - Aspects of Geometric Modelling, in: Proc. MICAD'84, Paris, Hermes Publishing, (1984)

[67] Theron, M.: L'algebre des solides et la CFAO en mechanique - Un exemple: le systeme EUCLID, in: Proc. MICAD 84, Paris, Hermes Publishing, (1984)

[68] Varady, T.: Some practical aspects of an experimental system for design and manufacture of sculptured surfaces, Computers in Industry, Vol. 3, (Coons' Memorial Issue), (1982)

[69] Varady, T.: Synthesis of volume modelling and sculptured surfaces in BUILD-2, in: Proc. of Joint Anglo-Hungarian Seminar on CAGD, Budapest, (1982)

[70] Varady, T.: Surface-surface intersections for double-quadratic parametric patches in a solid modeller, in: Proc. of the Third Joint Anglo-Hungarian Seminar on CAGD, Cambridge, (1983)

[71] Varady,T.: Basic equations and simple geometric properties of double--quadratic curves and surfaces, CAD Group Document 117, Cambridge University Engineering Department, (1984)

[72] Varady, T.: Geometric interrogations for double-quadratic curves and double-quadratic surfaces, CAD Group Document 118, Cambridge University Engineering Department, (1984)

REFERENCES

[73] Varady, T.: A simple adaptive curve-fitting algorithm for generating intersection curves in volumetric modellers, CAD Group Document 119, Cambridge University Engineering Department, (1984)

[74] Varady. T.: Analogy between generating planar intersection curves and silhouette curves of (double-)quadratic surfaces, CAD Group Document 120, Cambridge University Engineering Department, (1984)

[75] Varady, T.: Double-quadratic picture book, CAD Group Document 121, Cambridge University Engineering Department, (1984)

[76] Varady, T.: Operations for integrating free-form surfaces into volumetric modellers, (submitted to PROLAMAT'85 Conf., Paris), (1984)

[77] Veenman, P.: The design of sculptured surfaces using recursive subdivision techniques, in: Conf. on CAD/CAM Technology in Mechanical Engineering, M.I.T., Cambridge, Mass.,(1982)

[78] Veenman, P.: Private communication, Shape Data Ltd., Cambridge, (1984)

[79] Voelcker, H.B. et al.: An introduction to PADL, TM-22, Production Automation Project Report, University of Rochester, (1974)

[80] Wordenweber, B.: Automatic Mesh Generation of 2 and 3 Dimensional Curvilinear Manifolds, PhD Thesis, Computer Laboratory, University of Cambridge, (1981)

LIST OF FIGURES

FIGURES

FIGURES

FIGURES

Chapter V

Chapter VI

1984-BEN JELENTEK MEG:

155/1984 Deák, Hoffer, Mayer, Németh, Potecz, Prékopa, Straziczky: Termikus erőműveken alapuló villamosenergiarendszerek rövidtávu, optimális, erőművi menetrendjének meghatározása hálózati feltételek figyelembevételével.

156/1984 Radó Péter: Relációs adatbáziskezelő rendszerek összehasonlitó vizsgálata

157/1984 Ho Ngoc Luat: A geometriai programozás fejlődései és megoldási módszerei

158/1984 PROCEEDINGS of the 3rd International Meeting of Young Computer Scientists.
Edited by: J. Demetrovics and J. Kelemen

159/1984 Bertók Péter: A system for monitoring the machining operation in automatic manufacturing systems

160/1984 Ratkó István: Válogatott számitástechnikai és matematikai módszerek orvosi alkalmazása

161/1984 Hannák László: Többértékü logikák szerkezetéről.

162/1984 Kocsis J. - Fetyiszov V.: Rugalas autamatizált rendszerek: megbizhatóság és irányitási problémák

163/1984 Kalavszky Dezső: Meleghengermüvi villamos hurokemelő hajtás vizsgálata

164/1984 Knuth Előd: Specifikációs adatbázis modellek

165/1984 Petróczy Judit: Publikációk 1983

1985-BEN EDDIG MEGJELENTEK:

166/1985 Radó Péter: Információs rendszerek számitógépes tervezése

167/1985 Studies in Applied Stochastic Programming I. Szerkesztette: Prékopa András

168/1985 Böszörményi László - Kovács László - Martos Balázs Szabó Miklós: LILIPUTH

169/1985 Horváth Mátyás: Alkatrészgyártási folyamatok automatizált tervezése

170/1985 Márkus Gábor: Algoritmus mátrix alapu logaritmus kiszámitására kriptográfiai alkalmazásokkal

5326