Acta Numerica (1994), *pp.* 411–466

Numerical geometry of surfaces

Malcolm Sabin
Department of Industrial Studies
University of Liverpool, England
E-mail: mal0r@liverpool.ac.uk

The mathematical techniques used within Computer Aided Design software for the representation and calculation of surfaces of objects are described. First the main techniques for dealing with surfaces as computational objects are described, and then the methods for enquiring of such surfaces the properties required for their assessment and manufacture.

CONTENTS

Notation Lower case letters are used to denote scalars. x, y and z are coordinates in $\mathbb{R}^3$, u and v are coordinates in parameter space. f, g and h are scalar functions. Scalar expressions are bracketted by round brackets ( ). Greek letters are usually scalar functions, except for α and θ, which are angles, and ρ which is a radius.

Upper case letters are used to denote vectors or, occasionally, matrices. P denotes a point, and N a surface normal vector. Point-valued or vector-valued expressions are bracketted by square brackets [ ]. Square brackets are also the convention for the vector triple product $[A, B, C] = [A \times B] \cdot C$

Unit vector expressions are denoted by the use of angle brackets $\langle\ \rangle$, or by the notation $\hat{N}$.

1. SURFACE DEFINITIONS

In data reduction and similar numerical activities, the word surface is sometimes used loosely to dramatize the behaviour of some function, and the surface equation with which most generalist mathematicians will be most comfortable is the simple form

$$z = f(x, y) \tag{1.1}$$

but numerical geometers working in the application of computing to design and manufacture soon found that this equation did not capture some of the most fundamental properties of manufactured artifacts.

The main problem was that equation (1.1) is firmly locked within one coordinate system. Rotating such a surface definition through a few degrees to a new position will in general not give a new definition of the same form as the old. We required surface descriptions which were closed (i.e. merely involved changes of coefficient values) under such operations as solid body rotations.

Those involved in computer graphics, who wanted to draw perspective pictures, preferred representations which were closed under perspective transformations.

In response to those needs, three generalizations have been used.

The first is the symmetric function of the coordinates.

$$f(x, y, z) = 0. \tag{1.2}$$

This form must have been discovered almost as soon as Descartes invented coordinates, and it has been the form in which the quadrics have been expounded ever since, in such text-books as Cohn (1961), McCrea (1960) and Eisenhart (1960).

The golden age of algebraic geometry in the mid- to late-1800s discovered all that was to be known about surfaces of this form when f was a polynomial function, and much of their knowledge still stands us in good stead today. See Book II of Coolidge (1963) for a good flavour of algebraic geometry.

This type of surface has been the main tool for those numerical geometers who have been representing machined artifacts, which are mainly bounded by surfaces which are plane or quadric, with the occasional torus. Such surfaces have fairly simple analytic equations.

The second is the parametric form

$$\begin{aligned} x &= f_1(u, v), \\ y &= f_2(u, v), \\ z &= f_3(u, v). \end{aligned} \tag{1.3}$$

This was devised by Gauss (1828) in his studies of mapping.

This form of surface has been the main tool used by those of us who have had to deal with smoothly flowing aesthetic shapes, such as those bounding

aircraft, cars, ships, shoes or consumer articles such as hair-driers or electric shavers. Their main attraction is that it is fairly simple to make a piecewise definition, whereby one set of coefficients can be used in one part of the u, v domain, another set in another. You can change the shape of the rear fender without altering the front.

The third is a form which has appeared in own own generation. It does not have a single simple equation in terms of coordinates, but instead has a procedural definition, which says that a piece of surface lies within some piece of space. If you want to nail it down more tightly, an algorithm is available which divides the piece you have into smaller pieces, each of which lies within a smaller piece of space.

2. Analytic surfaces

This approach treats a bivariate point set definition as the set of zeros of some function of position

$$f(P) = 0.$$

Because we are concerned with real geometry in the colloquial sense, we deal with functions whose coefficients are real, and which map from $\mathbb{R}^3$ to $\mathbb{R}$

A linear function gives a plane. Polynomial functions of higher order give, in general, curved surfaces. Quadratic functions give a family of surfaces called the *quadrics*, which includes the sphere, the cylinder, the cone, ellipsoids, and various paraboloids and hyperboloids. The first three of these are called the *natural quadrics*, and are important in that faces of machined objects are frequently of this form. In fact the whole technology of the representation of machined parts, whose complexity is primarily in the way large numbers of faces interact with each other, was built for at least its first ten years on natural quadrics only. In that context the faces are seen as boundaries between material and 'outside', and so the function is regarded as defining the half-space

$$f(P) < 0$$

rather than just the zero-set.

This section is concerned with faces in small numbers, and we do not distinguish particularly between the two sides of a surface.

Differential properties Consider a point P lying in a surface f. Because the surface is bivariate, there are directions in which we can move from P while remaining in the surface. Let such a direction be T, represented by a *tangent vector.*

Thus for small displacements such as $\mathrm{d}s$,

$$f(P + T\,\mathrm{d}s) = 0.$$

This can be expanded as a local Taylor series

$$f(P) + (\mathrm{d}f/\mathrm{d}x\, T_x + \mathrm{d}f/\mathrm{d}y\, T_y + \mathrm{d}f/\mathrm{d}z\, T_z)\, \mathrm{d}s = 0.$$

The term in brackets is just an inner product, which has to be zero

$$\mathrm{d}f/\mathrm{d}P \cdot T = 0.$$

Now T is any tangent vector, and $\mathrm{d}f/\mathrm{d}P$, which is a triple and can therefore be interpreted as a vector, must lie in the direction perpendicular to the surface, because its inner product with any tangent is zero. It is called the *surface normal*, and will usually be denoted by N.

Sometimes it is convenient to represent the direction rather than the magnitude of N. For this we take the vector in the same direction as N, but of unit magnitude. This is called the *unit surface normal*, and is denoted by $\hat{N}$.

If the function f is such that N is of unit magnitude near a point P of the surface, the value of f measures distance from the surface locally.

The second derivatives $\mathrm{d}^2 f/\mathrm{d}P^2$ form a matrix which encapsulates the local curvature behaviour. This is most obvious if we think of it as

$$\frac{\mathrm{d}}{\mathrm{d}P}\left(\frac{\mathrm{d}f}{\mathrm{d}P}\right),$$

which can be written as

$$\frac{\mathrm{d}N}{\mathrm{d}P}$$

so that the inner product with any displacement δP gives the corresponding change in surface normal.

2.1. Definitions

The most frequent form of analytic surface definition is by the specific geometric properties of the specific surface equation. For example, the centre and radius of a sphere, the vertex and semi-angle of a cone, a point on and the normal vector of a plane.

In a software system which is known to deal only with planes and quadrics, this initial data can be converted into the equivalent coefficient matrix of the homogeneous quadratic form.

$$P^i A_{ij} P^j = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} & A & & B \\ & & & \\ & B^T & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0. \qquad (2.1)$$

Any more general system needs to provide a procedural interface, which will be described in more detail under Interrogations, below. There is a trade-off between holding data in the form in which it was first supplied,

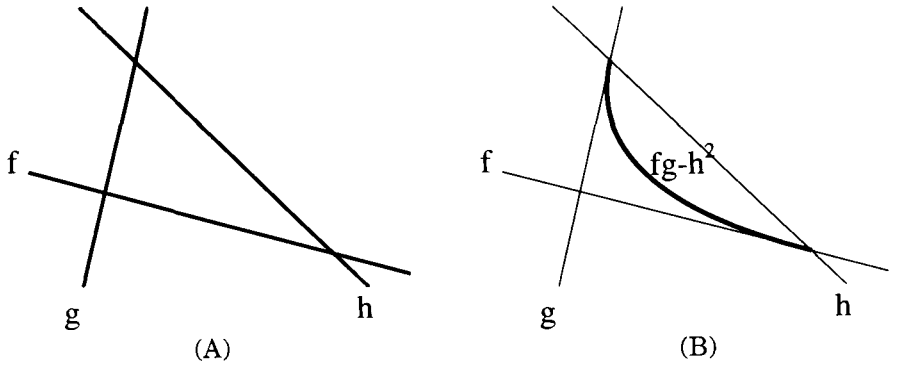

Fig. 1. Liming blends.

thus making editing more transparent, and doing as much pre-processing as possible to speed subsequent enquiries.

2.2. Blends and fillets

For the most part the analytic shapes which are actually defined are of low order, the highest being quartic for the torus. There is one exception to this: where blending surfaces are defined by an algebraic combination of the functions of the surfaces which they join.

The principle is a simple one, articulated by Liming in the late 1930s, but building on algebraic geometry ideas from the mid 19th century.

Let $f = 0$ and $g = 0$ be equations of two intersecting curves in two dimensions, and let $h = 0$ be the equation of a curve running close to their intersection. Then for any general value of a parameter λ, $fg - \lambda h^2 = 0$ is the equation of another curve, which lies tangent to each of f and g at its intersection with h.

If f, g and h are all of the same order, n, the order of the combination is $2n$. In Liming's work n was 1, and the blend curves conics.

If f, g and h are all normalized so that $\mathrm{d}f/\mathrm{d}P$ is of unit length locally, a value of λ of about 1 gives an intuitive fillet. Values nearer to 0 give a blend going closer to the intersection of f and g, and larger values give a blend which lies closer to h.

The same principle applies where f, g and h are surface equations in three dimensions. The blend, articulated by several groups of researchers almost simultaneously (Middleditch and Sears 1985; Rockwood and Owen, 1985; Hoffman and Hopcroft, 1986; 1987) is now a surface tangent to the base surfaces f and g all along their respective intersection curves with h.

Such surfaces are best described as blends, as the term fillet is usually taken to mean a surface generated by a ball rolling along the intersection

(except by numerical geometers, who prefer to use the term fillet more generally).

The rolling ball fillet is a much harder shape to represent. We have to determine the locus of the ball's centre, which is the intersection of two surfaces offset from the base surfaces by the radius of the ball, and then construct the envelope of the ball as it travels along this locus. All these operations are in fact closed in the universe dealt with by algebraic geometry, but the orders of the objects involved escalate inconveniently. The equations also tend to have spurious parts to their solutions which do not correspond to the intended shape.

For example, the offset of an ellipse in 2D is a curve of order 8, which includes both inwards and outwards offsets. These two parts are not separable, in general, by factorization of the eighth-order equation into two fourth-order factors.

3. Parametric surfaces

Although most sculptured surface software deals with analytic surfaces to some extent, if only for planes, it is the parametric representation which is the workhorse.

In this form, the surface is regarded as a mapping from a *parameter plane* into R^3. The computational representation of the surface is the set of coefficients of the mapping.

The most general case is thus

$$P = F(C : u, v), \tag{3.1}$$

where u and v are the parameters, coordinates in the parameter plane, and C is the set of coefficients.

Differential properties The defining equation of a parametric surface can be differentiated with respect to the parameters. The two partial derivatives $\mathrm{d}P/\mathrm{d}u$ and $\mathrm{d}P/\mathrm{d}v$ lie tangent to the surface. This can be seen from the definition of the derivative as the limit of a difference.

$$\frac{\mathrm{d}P}{\mathrm{d}u} = \lim_{\delta u \to 0} \left[\frac{P(u + \delta u, v) - P(u, v)}{\delta u} \right]. \tag{3.2}$$

Both of the points in this equation lie in the surface, and the secant vector between them becomes a tangent in the limit.

The cross product of the two derivative vectors lies perpendicular to the surface, and is called the surface normal vector, N.

$$N = \mathrm{d}P/\mathrm{d}u \times \mathrm{d}P/\mathrm{d}v. \tag{3.3}$$

The unit vector in the same direction as N is called the unit surface normal, $\hat{N}$.

The second derivatives with respect to parameter provide information about the curvature of the surface. This is evident from the fact that the derivative of N with respect to parameter involves the second derivatives of P.

Definitions Although the general form of parametric surface equation is as quoted in equation (3.1), the subset actually used is much smaller. There are only two forms at all widely used for primary definition, the piecewise polynomials, and the transfinite interpolants.

3.1. Functions linear in the coefficients

The first limitation is to functions of u and v, linear in the coefficients.

$$P = \sum C_i \phi_i(u, v), \tag{3.4}$$

where the ϕ_i are scalar valued functions and the C_i are vector-valued coefficients.

There have been various attempts to use *shape parameters*, which are coefficients taking a nonlinear role, notably the superellipses of the Lockheed Master Dimensions system, and the ν-splines of Nielson (1974), but such coefficients are not found to be particularly easy to use, except possibly for generating academic dissertations.

3.2. Points and vectors

It is useful at this point to distinguish between *points* and *vectors*. Both are represented by triples of coordinates, but they transform differently under solid body transformations. If we change coordinate system, the coordinates of a point P are transformed to

$$P' = M \cdot P + O, \tag{3.5}$$

where M is a 3×3 orthonormal matrix and O is a point.

A vector, typified by the displacement from one point to another, is transformed according to

$$P' = M \cdot P. \tag{3.6}$$

We assume that the set of basis functions is independent, so that there is no set of nonzero scalar coefficients α_i such that $\sum \alpha_i \phi_i(u, v) = 0$.

There can then be at most one subset of the ϕ_i such that the sum of the functions within this subset is identically equal to 1. All the bases of interest do have such a subset. Call these functions the ρ_i and the remainder the ψ_i. Let the coefficients of the ρ_i be called R_i and of the ψ_i S_i.

Then the equation for a general point of the surface can be written

$$P(u,v) = \sum R_i \rho_i(u,v) + \sum S_i \psi_i(u,v). \tag{3.7}$$

Now suppose that we apply a point transformation to the R_i and a vector transformation to the S_i.

$$\begin{aligned} P'(u,v) &= \sum (M \cdot R_i + O)\rho_i(u,v) + \sum M \cdot S_i \psi_i(u,v) \\ &= M \cdot \sum R_i \rho_i(u,v) + O \sum \rho_i(u,v) + M \cdot \sum S_i \psi_i(u,v) \\ &= M \cdot \Big(\sum R_i \rho_i(u,v) + S_i \psi_i(u,v) \Big) + O \sum \rho_i(u,v) \\ &= M \cdot P(u,v) + O. \end{aligned} \tag{3.8}$$

This is exactly the transformation of $P(u,v)$ as a point, which is what we require for the surface representation to be closed under solid body transformation.

3.3. Tensor product surfaces

Even within the limitation to scalar functions with point and vector coefficients, the choice of basis functions ϕ is dominated by the tensor products where the coefficients are viewed as forming some kind of rectangular grid:

$$P = \sum C_{ij} \phi_i(u) \phi_j(v). \tag{3.9}$$

The mainstream of development can be followed by considering what univariate functions are applied within a tensor product. This has the great convenience of allowing the surface equations to be explained in terms of curves,

$$P = \sum C_i \phi_i(u) \tag{3.10}$$

because the surface equation can be regarded as two nested summations. The inner one describes a curve swept out by a point moving as u changes, the outer the way that this curve sweeps out a surface as v changes.

3.4. Piecewise polynomials

The first serious work used Hermite cubics within foursided patches considered as independent entities to be stitched together with various degrees of continuity. Ferguson (1965; 1993) built a system which enabled continuity of the first derivative to be achieved between patches which shared corner data. In fact he initially omitted some of the components of the full tensor product, but these were later added to the APTLFT-FMILL numerical control programming system very widely used in the 1970s and early 1980s.

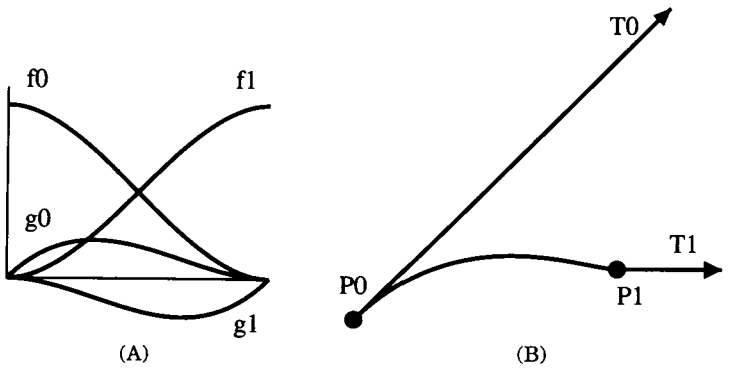

Fig. 2. Hermite functions and a Hermite curve.

The basis functions were

$$\begin{aligned} f_0(u) &= 1 - 3u^2 + 2u^3, \\ f_1(u) &= 3u^2 - 2u^3, \\ g_0(u) &= u(1-u)^2, \\ g_1(u) &= -(1-u)u^2. \end{aligned} \tag{3.11}$$

In this scheme, the coefficient of f_0 in a curve is the initial point, $u = 0$, that of f_1 the final point, $u = 1$. The coefficient of g_0 is the first derivative of position with respect to parameter at $u = 0$, of g_1 the first derivative at $u = 1$. The finite piece of curve defined in this way is the image of the interval from 0 to 1. In this context the distinction between open and closed intervals is not significant. Similarly, the piece of tensor product surface is the image of the unit square of parameter plane.

Clearly, with coefficients of this form, pieces of curve which will join with either continuity of position only or continuity of position and first derivative can easily be defined. This is achieved by having adjacent pieces share defining data.

In the surface context, the ff products have as coefficients the four corners of the patch, the fg products the corner values of the derivatives with respect to u, and the gf products the derivatives with respect to v. The gg products which Ferguson initially omitted have as coefficients the mixed partials $\mathrm{d}^2P/\mathrm{d}u\,\mathrm{d}v$.

The success of this scheme must not be understated. Hundreds of millions of dollars worth of aerospace parts have been machined successfully with this mathematics. However, it had two problems: using different magnitudes for the first derivative vector would give different curves, and it was not obvious how to choose the right magnitude first time; and that the mixed partials were even harder to choose.

One solution to both these problems was published by de Boor in 1962. He

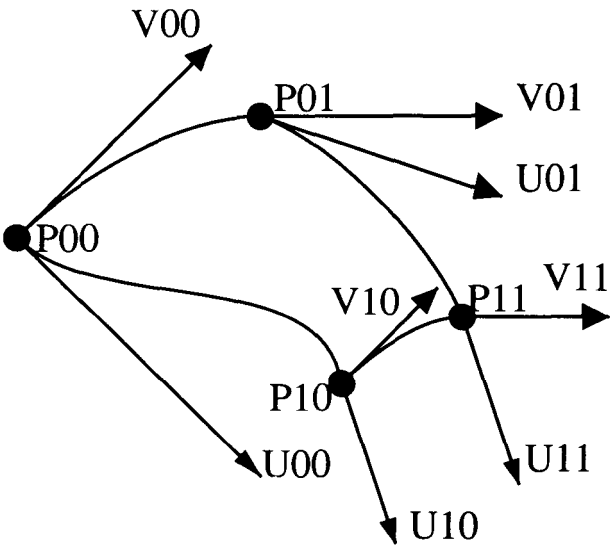

Fig. 3. Hermite surface definition.

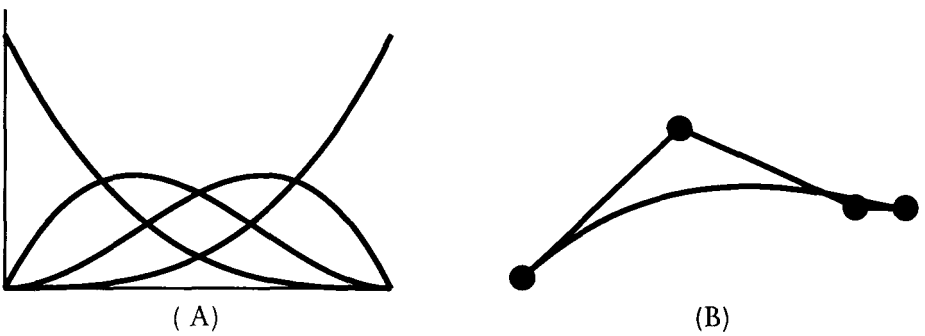

Fig. 4. Bernstein functions and a Bernstein curve.

used interpolating cubic splines as differentiation operators to determine all the required derivatives, starting from an array of points to be interpolated.

The other was devised independently by de Casteljau and by Bezier, both working in the French automobile industry (Bezier 1971).

They used, instead of the Hermite basis, the Bernstein basis

$$\phi_i(u) = \left(\frac{6}{i!(3-i)!}\right)(1-u)^i u^{3-i}, \quad i = 0, \ldots, 3. \tag{3.12}$$

The coefficients of these functions in the curve context are a sequence of points, usually thought of as forming an open polygon, usually called the control polygon. This has the same end-points as the curve itself, and is tangent to the curve at the end-points. Further, the curve has an inflexion

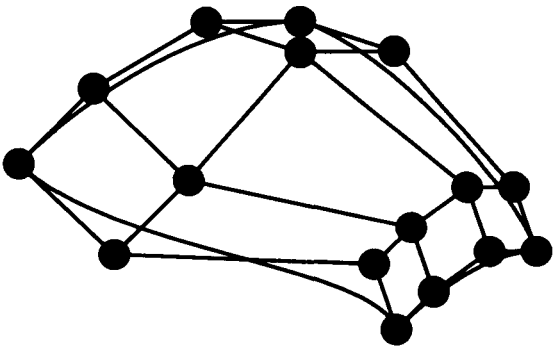

Fig. 5. Bernstein surface definition.

only if the polygon does. (It can also have two inflexions if the curve has a loop.)

In the surface context the coefficients form a rectangular network of points whose corners are the corners of the patch, and whose edges are the control polygons of the edges of the patch.

This approach solved both the vector magnitudes and the second derivatives problems. The obviously right vector magnitude was normally that which made all three of the polygon legs about the same length, but vectors could be shortened predictably to avoid inflexions; and a smoothly laid out control network would give good second derivatives.

It also made it possible for higher- or lower-order polynomials to be used with facility. The originators found useful transformations which permitted a part of a patch, trimmed by a cubic in uv-space to be expressed as a single patch, but of higher order.

However, the assembly of an array of patches to model, for example, a car door with a feature line running along it, was still a detailed task, which had to be performed with great accuracy if discontinuities of tangent plane were not to creep in.

3.5. B-splines

The next step in the development was a combination of the Bernstein and spline ideas. Bernstein brought legitimacy to the idea that control points need not be interpolated, splines the idea that a complete surface could be regarded as a single entity with piecewise basis functions, rather than a collection of independent pieces. The Ph.D. dissertation of Riesenfeld, supervised by Gordon (see Gordon and Riesenfeld, 1974) explored the use of the B-spline functions, first described much earlier by Schoenberg (1946), as analogues for the Bernstein polynomials.

B-spline functions are well explained in de Boor (1987). A univariate

B-spline function is a piecewise polynomial defined over a particular partitioning of the real line into segments over each of which the function is polynomial. The values at which these segments meet are called *knots.*

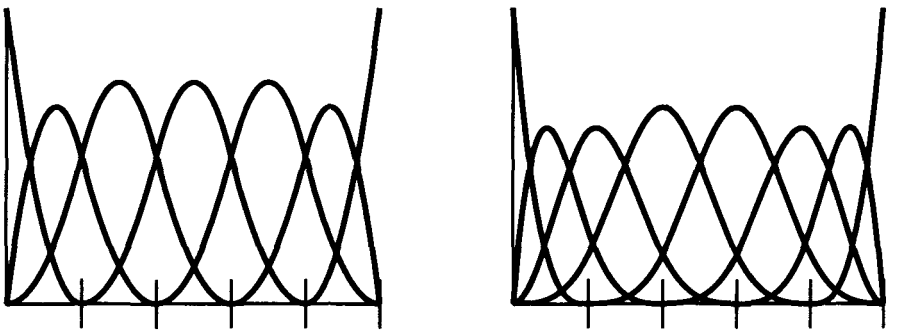

Fig. 6. Quadratic and cubic B-spline basis functions.

A B-spline function has finite support, i.e. it is nonzero only within a finite abscissa interval. It is strictly positive in the nonzero region. A nondegenerate B-spline has the maximum continuity possible with polynomial pieces of a given order, and minimum support. These properties are sufficient to define the function within a scaling factor, and the scaling factor for curve definition purposes is chosen so that the sum of all the nonzero B- splines at any given abscissa is exactly unity. As shown above, this partition of unity property is required so that when all the coefficients are transformed as points under a solid body transformation, the shape of the curve remains invariant.

By taking the limit as one of the pieces of abscissa becomes shorter and shorter, the concept of *coincident knots* (or even *multiple knots*) is obtained.

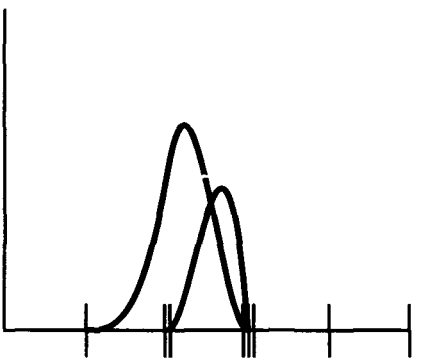

Fig. 7. Some degenerate B-splines.

B-splines with coincident knots can have discontinuities of lower than expected derivative, and narrower support regions.

A B-spline curve may be compared with a high-order Bernstein curve. The individual polynomial pieces for the B-spline are of lower order, but there are more of them. A change in the position of any one control point will, in general, affect only part of the B-spline curve, but all of the Bernstein curve.

A B-spline curve may also be compared with a collection of Bernstein curve segments. There are, in general, fewer control points for the same curve regarded as a B-spline. Moving one of the B-spline control points alters more of the shape, but maintains the continuity properties. Moving one of the Bernstein control points alters the shape, and in general reduces the degree of continuity.

In fact, a single Bernstein segment can be regarded as a special case of a B-spline, in which the domain is a single segment, with a sufficiently multiple knot at each end. Because B-spline curves can be concatenated to form another B-spline curve, a sequence of Berstein curves can also be regarded as a degenerate B-spline.

3.6. NURBS

Within the B-spline context, there are variations. If all the segments are of equal length in parameter space, the B-spline is termed *uniform*; if the software permits segments to differ in length, the B-spline is termed *non-uniform.*

The schemes described above all map from $\mathbb{R}^2$ to $\mathbb{R}^3$. It is also possible to map not into Euclidean space, but into projective space. It was mentioned above that the graphics community found it useful to have descriptions which were closed under perspective transformations.

This is achieved by mapping from $\mathbb{R}^2$ to $\mathbb{R}^4$, into the space of homogeneous coordinates.

Such an approach also makes it possible for the parametric form to represent a circle or ellipse exactly, using polynomial basis functions.

The conversion to Euclidean coordinates, which is necessary at some stage, is achieved by dividing the first three homogeneous coordinates by the fourth. This division gives the name *Rational* to the variant.

The full description Non-Uniform Rational B-Splines is universally truncated to the acronym *NURBS.*

Each control point now has four components. The extra degree of freedom is the *weight* by which the coordinates of a point are multiplied to give the first three components. These weights have the nature of nonlinear coefficients, and so are not easy to use when trying to achieve specific effects. Most designers use them only to match circular arcs; otherwise they

just use unit weights. However, the ability to match conics, the closure under perspective transformations, and the better match to algebraic geometry assumptions make NURBS the scheme normally chosen for surface representation systems of the late 20th century.

3.7. Multivariate B-splines

There is a further extension of the B-spline ideas, described in de Boor (1993b), which should be usable as a set of basis functions for nontensor product surface description. The arrangement of control points need not be strictly rectangular. However, the definition of the partitioning of the domain is a nonlinear control, and they do not give the topological freedom which recursive division definitions offer. To the best of my knowledge no commercial surface definition software has yet applied them.

3.8. Transfinite interpolation

The previous sections have dealt with surfaces defined by a finite number of coefficients, usually the positions of control points. A second important stream defines a surface patch in terms of its bounding curves. Because each curve contains an infinite set of points, these surface definitions are referred to as *transfinite*. If the curves are themselves represented by a finite description, this can be mapped through to give a finite description surface as a special case of a transfinite one.

The first transfinite surface was devised by Coons (1967), and dealt with the problem of building a parametric surface defined over the unit square, meeting a pre-defined curve on each of its four sides.

There is a compatibility requirement that the curves should meet at the corners, and we assume that this is met.

Let the four edges be $U_0(v)$, $U_1(v)$, $V_0(u)$ and $V_1(u)$. The compatible corners are

$$\begin{aligned} U_0(0) = V_0(0) &= C_{00}, \\ U_0(1) = V_1(0) &= C_{10}, \\ U_1(0) = V_0(1) &= C_{01}, \\ U_1(1) = V_1(1) &= C_{11}. \end{aligned} \tag{3.13}$$

We require a surface $P(u, v)$, such that

$$\begin{aligned} P(u, 0) &= V_0(u), \\ P(u, 1) &= V_1(u), \\ P(0, v) &= U_0(v), \\ P(1, v) &= U_1(v). \end{aligned} \tag{3.14}$$

Coons approached this in two stages: first to match two of the edges and then to worry about the other two.

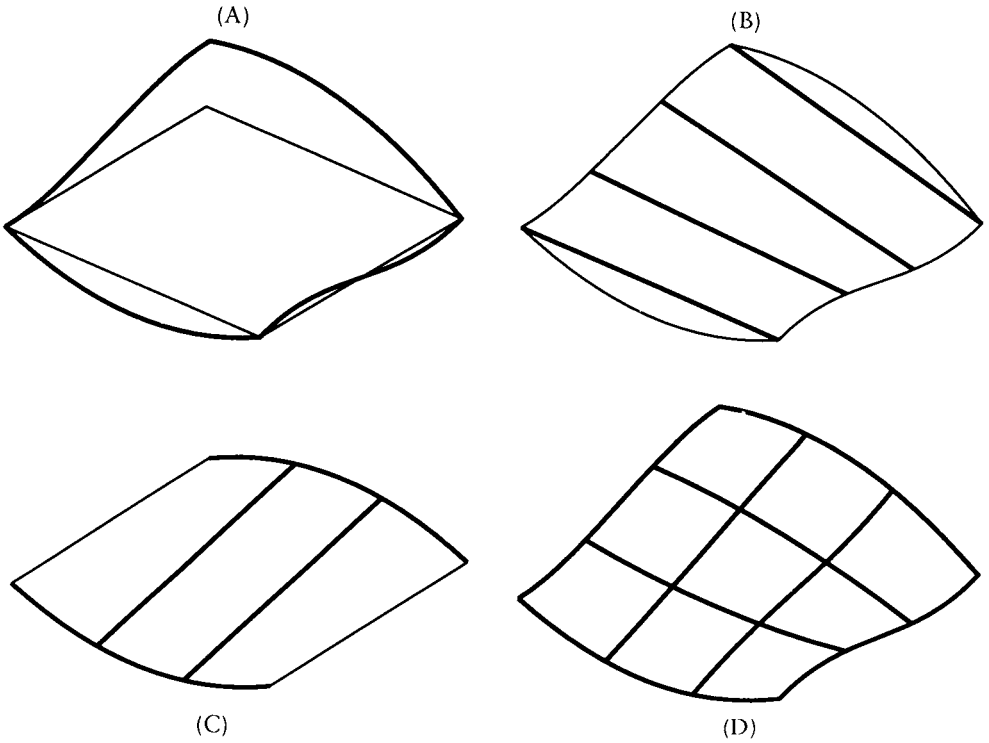

Fig. 8. Coons two-way blending.

The first two conditions above may be met by simple linear interpolation across the patch

$$P(u,v) = (1-v)V_0(u) + vV_1(u). \tag{3.15}$$

This does not match the other two conditions. There is a discrepancy of

$$U_0(v) - (1-v)V_0(0) - vV_1(0)$$

at the $u = 0$ edge and of

$$U_1(v) - (1-v)V_0(1) - vV_1(1)$$

at the $u = 1$ edge. So a second linear interpolation gives a correction function which can be added to the first try. Substituting the corner points by name, the correction becomes

$$(1-u)[U_0(v) - (1-v)C_{00} - vC_{01}] + u[U_1(v) - (1-v)C_{10} - vC_{11}] \tag{3.16}$$

giving the overall surface

$$\begin{aligned} P(u,v) \;=\; & (1-v)V_0(u) + vV_1(u) + (1-u)U_0(v) + uU_1(v) - (1-u)(1-v)C_{00} \\ & -(1-u)vC_{01} - u(1-v)C_{10} - uvC_{11}. \end{aligned} \tag{3.17}$$

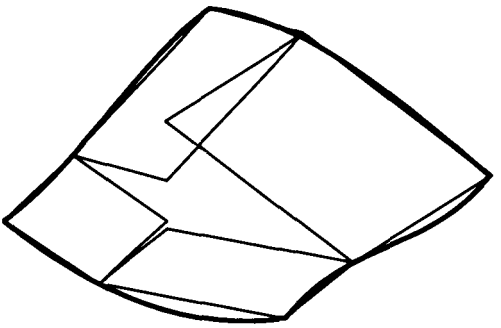

Fig. 9. Parallelogram construction.

Coons (1967) then took the 'add a correction' idea further, to achieve a match to given positions and cross edge derivatives, so that two adjacent patches, both being forced to match the same cross-derivatives, would land up tangent plane continuous. This used the f_0 and f_1 functions as described under Hermite surfaces above. However, it was not so successful, as, in the special case where the bounding edges were cubics, the twist terms were found to be zero.

Further, this approach gave only C^1 composite surfaces using cubic pieces. The splines gave C^1 using quadratics or C^2 using cubics.

Gordon (1969) made a much better generalization, by using interpolating spline functions as his generalization of the linear blending in the first-order Coons surface. This allowed a smooth surface to be passed through any compatible net of curves of constant u and curves of constant v.

The interpolation theorists formalized these ideas, and discovered that the Boolean sum of two interpolation or approximation operators was a powerful way of building new operators. Barnhill (1974) applied this technique to generate a transfinite surface in a triangle with given edges, and Gregory (1986) made one for n-sided regions with $n > 4$.

There is a second intuitive construction for the four-sided Coons patch.

Given a set of boundaries as before, and a value for the u, v pair at which a point is to be constructed, evaluate the edges at the appropriate points to give edge points $U_0(v)$, $V_1(u)$, $U_1(v)$, $V_0(u)$.

Now take each corner point, with the edge points on the adjacent edges, and construct a parallelogram. The fourth corner is an estimator for the required point. Take a weighted mean of these estimators, using weights inversely proportional to the logical area of the parallelogram.

As the required point nears one of the edges, the estimators from the corners at the ends of that edge have as their limit the edge point, and the weight of those two parallelograms dominate. Thus the surface interpolates the edges as required.

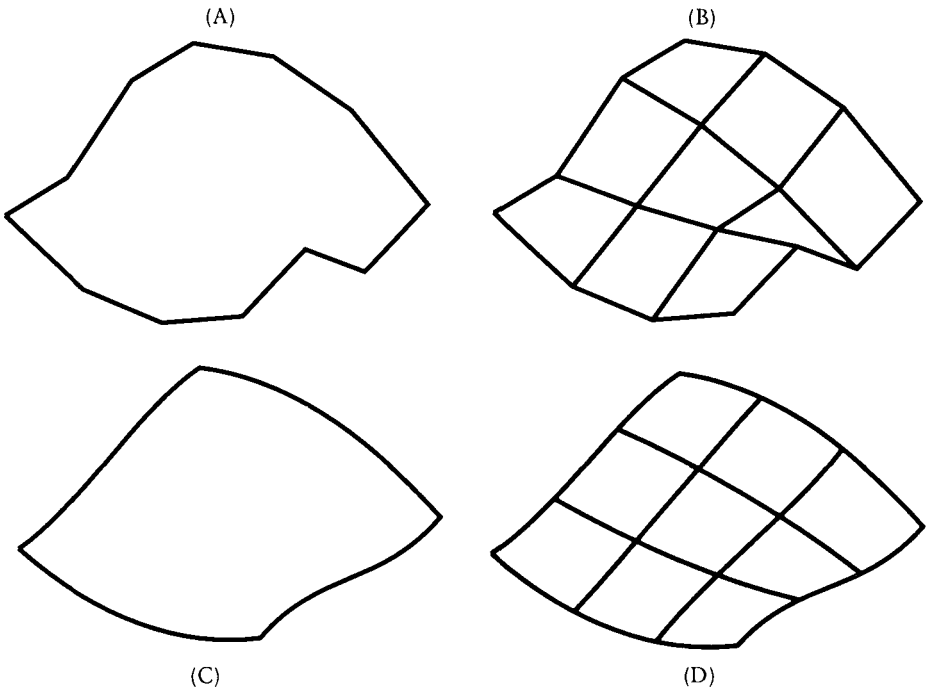

Fig. 10. Coons/Bernstein morphism.

This can be generalized to n-sided regions, and can also be applied to provide a way of estimating a surface when only three or even two of the sides are known.

Another intriguing property of the Coons first-order form is that if a cut is made across a patch, along a line of constant parameter, dividing the original patch into two, fitting each patch into its new boundaries gives exactly the same surface as the original.

Yet another is that if a first-order Coons patch is fitted to boundaries which are Bernstein curves, fitting the Coons surface to the curves gives exactly the same result as fitting a discrete interpolant to the control points of the boundary and then using Bernstein to interpolate the surface.

The transfinite techniques are extremely powerful, but more detailed in implementation than the B-spline ideas. They tend to be used in part-programming systems for numerically controlled machine tools, and in mesh generation for finite element analysis rather than in surface design systems.

A further method has been devised for fitting surfaces to desired boundaries. The Coons construction is actually the solution of a hyperbolic differential equation. Bloor and Wilson (1989; 1990) have tried using elliptic equations to determine the interior of a region with known boundary. There is much more computation in this case, because a point in the interior of an

elliptic equation solution depends on all the boundary points, and so they now regard their technique as a way of generating the interior control points in a conventional NURBS representation.

3.9. Offset parametric surfaces

Offsets were mentioned above, as a step in the construction of rolling ball fillets. They are also extremely important in surface systems for two other purposes: the construction of the faces of objects which have a small but nonzero thickness, where a nominal surface may well lie in the centre of the object; and the calculation of tool centre paths for numerically controlled machining. (Bell *et al.*, 1974).

The latter is more demanding, and requires more generality.

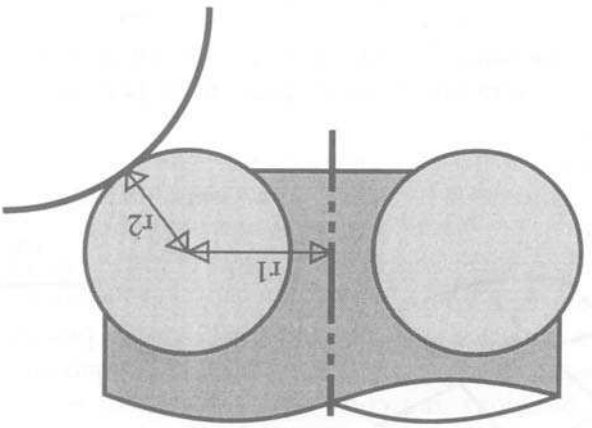

Fig. 11. Toroidal offset.

In the general case a machining cutter has teeth which sweep out a surface which is a surface of revolution made up of cylindrical, conical and toroidal pieces. The optimal cutting region is usually on just one of these pieces, and so the form of that piece is regarded as being the cuttter form for the purpose of computing a specific cut. It is usually a toroid.

Given a point at which the cutter envelope is to be tangential to the surface being machined, the question is where to put the tool datum point. Let this be the centre of the torus. Let the major and minor radii of the torus be r_1 and r_2 respectively.

Then moving up the surface normal from the cutting point by r_2 gives a point at the centre of a tooth. Moving perpendicular to the tool axis through r_1 gives the required tool centre.

$$P'(u,v) = P(u,v) + r_2\hat{N} + r_1\langle A \times N \times A\rangle. \tag{3.18}$$

This is algebraically complex, because of the square roots implicit in the unit vectors, but computationally straightforward.

Tool paths can be computed in either of two ways, depending on circumstance. The first is to compute the path of the cutting point first, and then for each point along this path, correct it to give the tool centre. This is relevant when it is required to leave a particular trace on the surface.

The second is to treat the offset equation as the definition of an offset surface, and use the offset surface within some interrogation which defines the actual tool centre path. For example, the path where a cutter cleans out the intersection of two surfaces, leaving a fillet as side-effect, must be computed by intersecting the two offset surfaces. Efficient milling cutters, used for roughing, demand that the bottom of the cutter is never used for cutting; the tool centre path must lie in a plane perpendicular to the cutter axis, and so a plane section through the offset surface is used.

The offset surface can either be held for interrogation as an approximation using some other form, or the offset equation can be applied dynamically every time the evaluation of a point on it is required.

It should be noted that the offsetting context described above can introduce discontinuities of position where the surface normal lies in the direction of the cutter axis. A tiny disk around such a point maps into a thin annulus of radius r_1. This indicates that accurate approximation by one of the standard forms may be difficult.

4. Recursive division surfaces

One of the properties shared by the Bernstein surfaces and the B-splines is that their bases are variation diminishing. Any plane cuts a B-spline curve no more often than it cuts the control polygon. In the surface case, the equivalent property is that a piece of surface lies entirely within the convex hull of its control points.

This means that there is a simple test which may identify quickly that there is no intersection. This leads to a 'divide and conquer' style of interrogation based on the idea that if a piece of surface is simple enough it can be interrogated directly; if not, it can be divided into two, and a test applied to see whether each half contains any result of the interrogation. Examples of such interrogation algorithms will be presented later in this section.

The key to such an approach is the ability to produce cheaply the control networks for the halves of a subdivided surface. In the case of the Bernstein basis, the de Casteljau construction provides this, where the new control points are generated through a tableau of simple linear combinations. In the case of B-splines the knot insertion algorithms play a similar role, which pretend that a single span of the B-spline is actually two, although there is no discontinuity at the join.

It was observed that all of these constructions of new polygons and control nets consisted of taking linear combinations of existing control points, the coefficients in these linear combinations depending on the detail of the surface mathematics.

The structure of the interrogations did not depend on that detail at all. Provided that a new set of control points could be calculated, the interrogations would work.

This meant that it was possible to put forward a surface definition in terms of the division construction. Catmull and Clark (1978) generalized the bicubic B-spline surface, and Doo and Sabin (1978) the biquadratic B-spline surface. What both achieved was the ability to have a control network which matched the desired natural topology of the surface being defined, whereas the basic tensor product B-spline enforced that the control network had to be a regular rectangular grid of control points.

This was not just an aesthetic issue. A unit square of the real parameter plane can map nicely onto pieces of surface which are bounded and square, or onto a finite length of cylinder, or onto a complete torus. It can map onto all of a sphere except one point, but in the neighbourhood of that point the mapping is highly singular. It cannot map onto closed surfaces of higher topological type. The recursive division control networks have no such limitation, and oriented surfaces, open or closed, of any topology (see Griffiths, 1976) can be represented as unitary entities. There is only a small question of how to handle nonoriented surfaces.

The rules of this game were rapidly sorted out. If the control network were actually regular in some region, the generalized construction should be equivalent to one of the standard tensor product formulations. The irregularities should not multiply as division takes place, and then at each stage of division, more and more of the surface area of the surface can be identified as equivalent to a standard form. The unknown regions round isolated irregularities can then be analysed in terms of the eigenstructures of the operator which replaces the configuration of control points round the singularity by a new, smaller configuration.

Despite the topological freedom which this approach gives, there are some difficulties which have led to it not being taken up at all widely.

The first is that the quadratic, which behaves well in terms of mathematical analysis, and has continuity of tangent plane but not, of course, of curvature, at the singular points, tends to give rather bulgy surfaces noticeably different from what one expects. A control network in the form of a cube, for example, gives a closed surface significantly different from a sphere. In other words, the method introduces features of shorter spatial wavelength than the density of control points implies. These features are artifacts. Adding extra density of control points merely increases the spatial frequency of the artifacts.

The second is that the cubic, which suffers from the same problem but less severely, does not have a formulation giving continuity of curvature at singularities. The mathematical uncertainty associated with this may have discouraged system builders from basing their software on this approach.

A third is that it is the uniform B-spline which can be generalized in this way. The non-uniform capability has been found so useful that to lose it would be a backwards step.

5. Conversions between different representations

It is often the case that the same point-set can be represented in more than one way. As will be seen in the next section, on Interrogations, it may be advantageous to have more than one representation available. Indeed, recursive division interrogation is important because it can be used for parametric surfaces.

(i) The natural quadrics have a familiar parametrization in terms of trigonometric functions as well as the analytic form. So do tori.

Quadrics also have a parametric representation in terms of rational parametric quadratics, and the tori in terms of rational parametric biquadratics. The quadratic representation of the quadrics maps the entire parameter plane onto the surface of the quadric; if it is required to cover a complete quadric by the unit square a quartic representation is available, but there is always at least one singular point.

It is also true that nonsingular cubic analytic surfaces can be parametrized as rational polynomials, but it is not clear how much higher in order one can go and still have rational polynomial equivalents. Even for quadrics the degenerate case is difficult to parametrize (consider the case where the quadric consists of two planes), and the cubic cylinder whose generator is a plane cubic with no double point cannot be parametrized with rational polynomials.

Parametrization with elliptic functions or with functions including square roots will doubtless extend parametrizability, but eventually the fact that higher order surfaces can easily be of high topological type will get in the way, forcing the use of a complex parameter plane, which rather spoils any advantage of having both analytic and parametric forms available.

(ii) On the other hand, all rational parametric polynomials have an implicit (analytic) form, equal in order to the highest total degree in the polynomial.

This result applies only to a single piece of parametric surface. Where a surface has a piecewise definition, each piece has its own analytic equation, and deciding which piece is relevant at any given sample point is nontrivial.

(iii) The parametric bipolynomials can all be expresssed as B-splines, and thence as recursive division surfaces.

Transfinite surfaces can also be expressed as recursive division surfaces provided that:
— the curve cutting the surface in two can be evaluated;
— the bounding curves can be subdivided; and
— bounding boxes can be evaluated for all these curves.
The bounding box for a transfinite surface can then be expressed in terms of the boxes of its bounding curves and that of its corner points. The resultant box is rather slack, and so this kind of surface will not be very efficient for recursive division interrogation.

(iv) Recursive division surfaces which are of use all consist of parametric pieces. These pieces are of reasonable size where the control network is regular, but form an infinite regress around the points of singularity. The regress is terminated by the unresolved piece becoming small enough and flat enough to be approximated by a plane polygon. This is not really a conversion, since all recursive division interrogation falls back on some other method at the leaves of the division tree.

6. SURFACE INTERROGATIONS

A surface defined by any of the methods above is a barren thing indeed if we cannot make use of that definition in the production of real artifacts. To do this we need to ask questions of the surface such as

'What shape must I make the supporting members to fit inside the skin?'

'What path must a milling cutter take to machine the shape of this surface?'

'What shape does this piece of surface have to be in the flat, so that I can cut it out before I bend it into its final form?'

We also need to ask questions of a candidate surface to find out if its shape will be satisfactory for its purpose. Examples of these concerns are

'Make me a shaded image as seen from here.'

'Show me a set of closely spaced plane sections.'

'Calculate the lift and drag of this wing.'

All these questions are higher or lower level examples of interrogations, and the ability to provide the answers robustly and at acceptable computing cost are the important issues in the provision of surface software.

The list of possible enquiries is endless. Those described here include those most obviously required, and also others chosen to illustrate additional paradigms.

Within the scope of this section come the purely geometric enquiries; these need to be combined with application-specific knowledge of illumination models for rendering of shaded images, of feeds and speeds for machining, and of flow simulation calculations for the aerodynamics calculations, to provide such function.

This discussion of interrogation techniques starts with important properties of surfaces on which we can build. It then gives examples of those interrogations which return results consisting of isolated points, continues with those which return curves, and then proceeds to those which process complete surfaces.

Within each group one example is taken in reasonable depth; others are then dealt with by identifying their equations and mentioning any special features which need to be taken into account.

A summary follows each group of interrogations.

In most cases we consider all three of

- analytic surfaces, where all we can ask is the value and derivatives of the defining function at specific points in space;
- parametric ones where we assume that we know nothing about the surface except its boundary in parameter space and what we can determine by enquiring the position and derivatives of the surface point at specific parameter values;
- recursive division surfaces where all we can do is ask for a piece of surface to be split into a number of pieces, and enquire the extent of some hull round each piece.

Limiting the available knowledge about the surface form to this extent makes the techniques extremely general, and certainly not restricted to the specific surface equations described in Sections 2, 3 and 4.

6.1. Basic interrogation properties

This section defines the basic interrogations which must be supported for each of the three surface forms, if the methods described below are to be applied.

Basic properties of analytic surfaces An analytic surface is one whose point set is the set of zeros of some computable scalar function.

It is necessary to be able to compute the value of the function, f, at any point, P.

It is necessary to compute its first derivative, $\mathrm{d}f/\mathrm{d}P$, with respect to the coordinates, and, for some interrogations, particularly those concerned with curvature properties, the second derivative, $\mathrm{d}^2f/\mathrm{d}P^2$.

The first derivative is a vector which for points actually lying on the

surface, is perpendicular to the local tangent plane. This we shall denote by N, and call the *normal vector.*

It is unusual for this to be of unit length. The vector of the same direction, but of unit length, is called the *unit normal vector* and is denoted by $\hat{N}$.

The second derivative is a symmetric tensor which may also be regarded as $\mathrm{d}N/\mathrm{d}P$.

Numerical differentiation is always available as a way of providing these derivatives, but for most surfaces found in CAD (computer aided design) systems, it is faster to differentiate the code which evaluates the function, and thereby produce accurate pointwise derivatives, using code which can often share comomon subexpressions with the basic evaluation.

Basic properties of parametric surfaces A parametric surface is a map from some part of $\mathbb{R}^2$, the *parameter plane* to $\mathbb{R}^3$, together with a description of the domain.

It is necessary to be able to evaluate this map at any point, (u, v), of the domain. Because the map is point-valued, we shall use the symbol P for both the map and the resulting point.

It is necessary to be able to evaluate the first derivatives, $\mathrm{d}P/\mathrm{d}u$ and $\mathrm{d}P/\mathrm{d}v$ of the map with respect to the parameters, and for some interrogations the second derivatives, $\mathrm{d}^2P/\mathrm{d}u^2$, $\mathrm{d}^2P/\mathrm{d}u\,\mathrm{d}v$ and $\mathrm{d}^2P/\mathrm{d}v^2$.

Again, specific code for these derivatives is preferable to numerical differentiation.

The two first derivatives are vectors which lie within the local tangent plane of the surface. The vector cross product $\mathrm{d}P/\mathrm{d}u \times \mathrm{d}P/\mathrm{d}v$ is a vector perpendicular to the local tangent plane.It is called the surface normal, and denoted by N. Again, it is not generally of unit length, and a unit surface normal is defined in the same way as for analytic surfaces.

If $||N|| = 0$ the surface is improperly parametrized, and the methods described below will probably fail. Note that $||N|| > 0$ implies $||\mathrm{d}P/\mathrm{d}u|| > 0$ and $||\mathrm{d}P/\mathrm{d}v|| > 0$.

It is necessary to be able to determine when a point (u, v) lies outside the domain.

It is also necessary to be able to scan round the boundary. The domain is a closed point set, so that parameter pairs lying on the boundary can be evauated as within the domain. The boundary will be treated here as a collection of curves in the parameter plane themselves parametrized by a scalar variable t.

Basic properties of recursive division surfaces A recursive division surface is the limit of the set of points in some collection of hulls, each hull in a collection corresponding to a piece of the total surface.

A hull is any convex point set guaranteed to contain all the points of its piece of surface. The convex hull is the minimal such, but slacker hulls, such

as the convex hull of a set of B-spline control points, or the bounding box thereof, generally lead to faster algorithms because the hull tests become much simpler. The implicit assumption here is that a hull is the intersection of a number of support planar half-spaces of predetermined orientations. Exactly which orientations are used is decided by the interrogation software writer. The bounding box is not misleading as a mental image, though using more support directions can improve performance.

It is necessary to be able to evaluate the hull (i.e. the pedal distances of the support planes) for a piece of surface.

It is necessary to be able to divide the piece of surface into two, and then produce a hull for each part.

Recursive division interrogations will only be efficient if as this division proceeds, the hull shrinks in linear dimension proportionately with the piece of surface it represents. The total volume of the hulls of all the pieces of a surface must shrink as the division is made to deeper levels.

For interrogations relating to orientation and curvature properties it is also necessary to be able to evaluate a hull containing the unit surface normal vectors of all points in the piece.

It is necessary to be able to decide when a piece of surface is small or simple enough for direct methods to be applied. As a default, the condition can be used that the entire hull is within the geometric precision required, but this demands very deep division, which is expensive in computing time.

For the purposes of this section, it is assumed that the boundary of the recursive division surface is that implicit in its piece structure. This will not be the case when recursive division methods are used for robustness on trimmed parametric surfaces. In such cases resolving the boundary issues will have to use the methods described under the parametric surfaces paragraphs of each interrogation.

7. Nilvariate interrogations

Within this section we are looking for results which take the form of single points. In general there may be multiple solutions. The type of each interrogation is therefore

$$\text{surface} \times \text{auxiliary data} \mapsto \text{set of points}$$

where the auxiliary data are whatever is needed for the specific enquiry. There are no particular data-structure complications.

7.1. Raycasting

This is the problem of finding the intersection of a straight line with a surface. It can be used in graphics in order to render a shaded image of a surface, and it is also a useful constituent of other algorithms.

Straight lines have a number of alternative representations. We can convert between them, and we use this ability to use the most convenient form for each manifestation of this interrogation.

The most fundamental form for any algebraic curve is probably the Cayley form

$$l(P,Q) = 0 \text{ if the line through } P \text{ and } Q \text{ meets the line } L. \tag{7.1}$$

In the case of a straight line the function l is bilinear.

$$l(P,Q) = P^i \cdot L_{ij} \cdot Q^j. \tag{7.2}$$

By taking two arbitrary values for Q we reach the form where the line is defined as the intersection between two planes

$$\begin{aligned} P^i.F_i &= 0, \\ P^i.G_i &= 0. \end{aligned}$$

In Euclidean coordinates a plane, F, can represented by a point thereon, Q_F, together with a plane normal F.

$$\begin{aligned} [P - Q_F] \cdot F &= 0, \\ [P - Q_G] \cdot G &= 0. \end{aligned} \tag{7.3}$$

For good conditioning, the normals F and G should be orthogonal to each other.

Taking any two distinct points of the line (and in homogeneous coordinates the intersections of the line with the four coordinate planes will always contain two distinct points), P_1 and P_2, any linear combination $\alpha P_1^i + \beta P_2^i$ will also satisfy equations (7.3).

In Euclidean coordinates we need to apply the normalization $\alpha + \beta = 1$, and it is most convenient to express the general point of the line in terms of

$$P = P_1 + \alpha T, \text{ where } T = [P_2 - P_1]. \tag{7.4}$$

From a given point P, known to lie on the line, the value of α can be recovered by

$$\alpha := \frac{[P - P_1] \cdot T}{T \cdot T}. \tag{7.5}$$

For some purposes the normalization can be carried further. If T is a unit vector, α measures distance along the line. This is useful only when dealing with metric properties, or when (7.5) is being invoked really often for a single line.

Raycasting analytic surfaces Here we use the parametric form of the line $P = P_1 + \alpha T$. Substituting this into the analytic equation $f(P) = 0$ gives an equation in α, thus reducing the problem to root-finding.

Root-finding of general functions is a nontrivial exercise, but for polynomials there are several well-understood methods. The problems are essentially those of conditioning: ill conditioned geometry can exist in the form of almost-tangencies, and will lead to ill conditioned roots. The purely numerical ill conditioning can be minimized by use of the Bernstein basis for all polynomials (Farouki, 1987a,b; Farouki and Rajan, 1988a,b).

When this interrogation is being used for graphics, the interesting root is the smallest positive one, corresponding to the nearest leaf of the surface to the eye point in the direction of view.

Raycasting parametric surfaces If L is represented as the intersection of two planes, the problem becomes one of solving two simultaneous equations.

$$\begin{aligned} [P(u,v) - Q_F] \cdot F &= 0, \\ [P(u,v) - Q_G] \cdot G &= 0. \end{aligned} \tag{7.6}$$

From any given starting point (u_0, v_0) sufficiently close to the solution, Newton iteration is fast and effective.

Expanding the surface as a local Taylor series about P_0 $(= P(u_0, v_0))$ gives

$$P(u_0 + \delta u, v_0 + \delta v) = P_0 + (\mathrm{d}P/\mathrm{d}u)_0 \delta u + (\mathrm{d}P/\mathrm{d}v)_0 \delta v. \tag{7.7}$$

Substituting this into equation (7.3) gives

$$\begin{aligned} [P_0 - Q_F] \cdot F + (\mathrm{d}P/\mathrm{d}u)_0 \cdot F\delta u + (\mathrm{d}P/\mathrm{d}v)_0 \cdot F\delta v &= 0, \\ [P_0 - Q_G] \cdot G + (\mathrm{d}P/\mathrm{d}u)_0 \cdot G\delta u + (\mathrm{d}P/\mathrm{d}v)_0 \cdot G\delta v &= 0, \end{aligned} \tag{7.8}$$

which may be cast as a matrix equation for δu and δv

$$\begin{bmatrix} \mathrm{d}P/\mathrm{d}u \cdot F & \mathrm{d}P/\mathrm{d}v \cdot F \\ \mathrm{d}P/\mathrm{d}u \cdot G & \mathrm{d}P/\mathrm{d}v \cdot G \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \begin{bmatrix} -[P_0 - Q_F] \cdot F, \\ -[P_0 - Q_G] \cdot G. \end{bmatrix} \tag{7.9}$$

solution of which gives increments of u and v to the next estimate.

$$\begin{aligned} u_{i+1} &:= u_i + \delta u_i, \\ v_{i+1} &:= v_i + \delta v_i. \end{aligned}$$

Iteration proceeds until the point evaluated lies within the required tolerance of each of the planes. For this test to be most appropriate, the planes themselves should be close to orthogonal.

The straightfoward Newton method described above almost always works fast and efficiently. Experience suggests that in typical surface interrogation situations the average number of steps taken is two (three evaluations altogether).

However, in the worst case, it is possible for divergence to happen. It is also possible for the computed increment to lead to a point outside the domain of the surface, so that strictly it cannot be evaluated.

Enforcement of convergence Even though the actual step may not result in reduction of the residuals of both halves of (7.3), the increment direction is a descent direction, and so a small enough step in that direction will lead to a better estimate. This is exactly the line search issue in unconstrained optimization (Nocedal, 1992).

Because this situation occurs rarely, we need not hunt for an optimum algorithm. A simple fix-up considers each of the residuals separately.

Let the residuals of one equation in (7.3) at two successive estimates be r_i and r_{i+1}.

If these are of opposite sign, the appropriate fraction of the step to take is $r_i/(r_i - r_{i+1})$, derived by linear interpolation: if they are of the same sign, use $r_i/(2r_{i+1})$, derived by linear interpolation on estimates of first derivative of residual.

Each of the equations in (7.3) gives an estimated fraction of the step to take. (If $|r_{i+1}| < |r_i|$ then the appropriate fraction is 1.) Use whichever fraction of the estimated step is smaller.

This gives a new P_{i+1} for which the residuals must again be estimated, and if necessary the fix-up must be repeated. In order to be able to prove termination of this loop there has to be a counter, because geometric situations can occur where there is no solution. In the case of raycasting this happens when the surface we are dealing with is not C^0, (for example, the offset of a surface which is not C^1) but when applying the same approach to situations where the equations being solved involve derivatives of the surface it can happen when some higher continuity is lacking.

The refinement part of the interrogation then has to report back to its calling code that no solution exists in this locality.

Enforcement of boundary When the next estimate lies outside the domain, we need to find a local point within the domain, so that the iteration can continue, because although one estimate has gone outside, the final solution may lie inside the domain. Since the boundary is typically well behaved, at most two segments of boundary are involved. If two are involved we use the corner itself as the next estimate; if only one, the intersection of the increment with the boundary.

If the solution really does lie outside the boundary, the refinement step must again report absence of a solution to its calling code.

Multiple solutions Multiple solutions can occur. It is possible for a ray to cut even a single bicubic patch in as many as 18 distinct points.

The second case of figure 12 is rather more worrying, as there is no separation of the solutions by a locus where the surface normal is orthogonal to the ray axis.

Each starting point leads to at most one solution. Where there are multiple solutions we need multiple starting points. Different starting points may

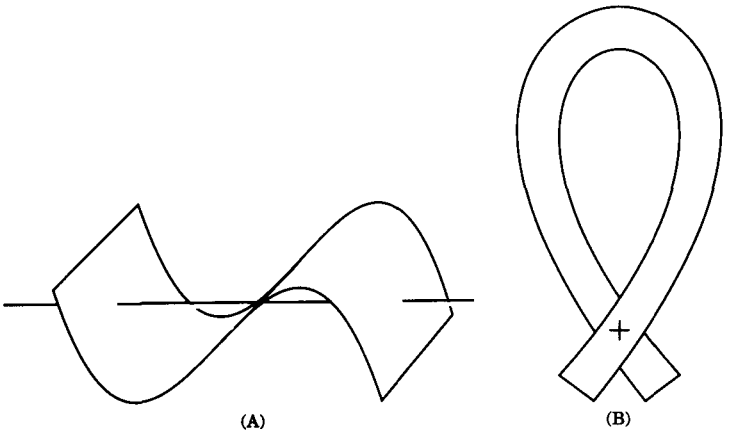

Fig. 12. Multiple solutions for raycasting.

converge to close estimates of a common solution: two different starting points may also converge to close but different solutions.

Handling this issue is thoroughly messy and *ad hoc*. Do not expect profound mathematics here.

The simplest approach is to use a dense spatter of starting points, and refine each of them. Finally, sort the solutions along the line, and combine those which are close in the parameter plane as well as in space.

This is unsatisfactory on two counts. First, there is no obvious way to decide how dense is dense enough: second, even a 10×10 grid means 100 evaluations; each refinement only three.

Reducing the number of start points considered is well worth while, reducing the number actually refined even more so.

System builders should consider associating information with each parametric surface, indicating how crinkly it is. This could be determined at the time the surface is defined, or over-ridden manually by the system operator. This information would then be used to decide the density of the spatter for all subsequent interrogations.

When the spatter is dense enough, the start points can be ranked according to the likelihood of their refining to a new solution. The first measure, in the case of raycasting, is the distance of the candidate from the line. However, there may be several starts close to one solution, while another solution may be some distance from the nearest start which leads to it. Starts which have a nearby start closer to the line should therefore be ranked low.

Coherence Rays are seldom cast in ones. A standard technique for improving performance is to use information from previous nearby instances of the same enquiry. For example, the places where the ray for an adjacent

pixel cut the surface are likely to be close to where this one cuts it, and can act as excellent start points.

However, situations can change from one pixel to the next, and overuse of coherence can lead to being caught on the wrong leaf of a surface.

Two rays can have different sets of cut points if either a surface boundary or a silhouette come between them.

Raycasting recursive division surfaces In a recursive division of a surface, all those pieces whose hulls do not intersect the line can be eliminated. This is efficient, particularly when the hulls are aligned with the direction of the line. It is probably worth setting up a special set of hulls when a large array of parallel rays are to be cast.

If the hulls are not aligned, it may be simpler just to cull all pieces except those which cross both of the planes.

In the graphics situation, it is also possible to eliminate all pieces which lie further from the eye point than the nearest root found so far, and so some sorting of the pieces on the stack may be worth while, so that the nearest root is found early.

7.2. Maximum extent

This enquiry determines the places within a surface which might well be points of maximum extent in a direction given by a vector T. Its primary use is as an auxiliary interrogation for other enquiries, particularly plane sections. It is chosen for inclusion here to illustrate enquiries which depend on surface derivatives.

Maximum extent of parametric surfaces The equation which solution points satisfy is $T \times N = 0$, where N is the surface normal, given by $\mathrm{d}P/\mathrm{d}u \times \mathrm{d}P/\mathrm{d}v$.

When this is true, we have $\mathrm{d}P/\mathrm{d}u \cdot T = 0$ and $\mathrm{d}P/\mathrm{d}v \cdot T = 0$.

Each of these has a residual at any given surface point, and so we can solve for increments to u and v which reduce these residuals to zero in a way precisely analogous to the raycasting algorithm.

The Taylor expansions of $\mathrm{d}P/\mathrm{d}u$ and $\mathrm{d}P/\mathrm{d}v$ are

$$\frac{\mathrm{d}P}{\mathrm{d}u}(u_0+\delta u, v_0+\delta v) = \frac{\mathrm{d}P}{\mathrm{d}u}(u_0,v_0) + \left(\frac{\mathrm{d}^2P}{\mathrm{d}u^2}\right)_0 \delta u + \left(\frac{\mathrm{d}^2P}{\mathrm{d}u\,\mathrm{d}v}\right)_0 \delta v$$

$$\frac{\mathrm{d}P}{\mathrm{d}v}(u_0+\delta u, v_0+\delta v) = \frac{\mathrm{d}P}{\mathrm{d}v}(u_0,v_0) + \left(\frac{\mathrm{d}^2P}{\mathrm{d}u\,\mathrm{d}v}\right)_0 \delta u + \left(\frac{\mathrm{d}^2P}{\mathrm{d}v^2}\right)_0 \delta v$$

so that the matrix equation for δu and δv becomes

$$\begin{bmatrix} \mathrm{d}^2P/\mathrm{d}u^2 \cdot T & \mathrm{d}^2P/\mathrm{d}u\,\mathrm{d}v \cdot T \\ \mathrm{d}^2P/\mathrm{d}u\,\mathrm{d}v \cdot T & \mathrm{d}^2P/\mathrm{d}v^2 \cdot T \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \begin{bmatrix} -\mathrm{d}P/\mathrm{d}u \cdot T \\ -\mathrm{d}P/\mathrm{d}v \cdot T \end{bmatrix}. \qquad (7.10)$$

Solution, including forced convergence and handling of the boundary proceeds exactly as in the raycasting case.

Maximum extent of recursive division surfaces Because we are basically solving the equations

$$\mathrm{d}P/\mathrm{d}u \cdot T = 0 \quad \text{and} \quad \mathrm{d}P/\mathrm{d}v \cdot T = 0,$$

recursive subdivision has to use culling based on the hulls of the first derivatives.

This is straightforward if we are using the recursive division form of a parametric surface, but if we are using a fully general recursive division surface the independent derivatives may not exist over pieces encountered early in the subdivision.

In this circumstance we need to recast the overall equation to use the surface normal hull.

Set up two auxiliary vectors, R and S, perpendicular to T and to each other.

We are now solving for

$$R \cdot N = 0, \quad S \cdot N = 0,$$

so if the surface normal hull of a piece does not cross the plane perpendicular to R passing through the origin, or if it does not cross the plane perpendicular to S, we can cull the piece.

7.3. Nearest point

We seek the nearest point, P, of a given surface to a given point Q.

This is a useful 'building-block' interrogation, second only to raycasting, since it is often useful for particular places on a surface to be indicated by the operator by specifying a nearby point, leaving the program to supply the precision.

It is also used in machining, where a path is traced until the tool centre meets a 'check surface'. This is checked efficiently by measuring the distance from a sample point to the nearest point on the check surface. The tool can then move that far along its trajectory before another test need be made.

Nearest point on an analytic surface Because we are not considering boundaries or discontinuities on analytic surfaces, the nearest point lies at the foot of a perpendicular.

A first estimate is derived very simply as

$$P_1 = Q - \left(\frac{f(Q)}{N_Q \cdot N_Q} \right) N_Q.$$

Typically this first estimate, when evaluated, will neither lie on the surface f nor have its normal passing through Q.

However, the Taylor expansion of f about P_1 gives a better estimate of the local tangent plane and the local perpendicular direction.

Unfortunately, if the distance of Q from the surface is larger than the local radius of curvature, the obvious iteration, intersecting the line from Q along N_1 with the Taylor plane, can oscillate and even diverge. It is necessary to take second derivatives into account.

The equation set to be solved is

$$\begin{aligned} f(P) &= 0, \\ [P-Q] \times N &= 0, \end{aligned}$$

where the second equation has rank 2, giving three equations in the three coordinates of P.

Let the residuals at P_1 of the equations be r_1 and R_1 respectively. Then for a step δP, the residuals will alter by amounts estimated by differentiating the above, and the Newton step is given by

$$\begin{aligned} \frac{\mathrm{d}f}{\mathrm{d}P}(P_1) \cdot \delta P &= -r_1 \\ -N(P_1) \times \delta P + [P-Q] \times \left[\frac{dN}{\mathrm{d}P} \cdot \delta P\right] &= -R_1. \end{aligned} \tag{7.11}$$

With this extra term, the iteration can converge on a point P whose distance from Q is a local maximum, rather than minimum. This is unlikely with surfaces of low order, but it is a wise precaution, having converged on an initial solution to set up a spatter of points at, for example, the vertices of a regular icosahedron centred on Q, with radius QP. Provided all of these points have a value of f of the same sign as Q there is confidence that the right solution has been reached. If not, a good starting point for a new convergence will be the point on the line between Q and the spatter point of most opposite sign of f, divided in the ratio of the two f-values.

Nearest point on a parametric surface The parametric surface case brings two complications: the first is finding a good place in the parameter plane from which to start; the second is the boundary.

Clearly this is a classical example of constrained optimization. Unfortunately, the mathematical interest in optimization has been in solving problems of high dimension in machines of limited memory. The dimension here is only two, and the ambition level significantly higher in terms of robustness and performance in the typical well-behaved cases.

Finding a good starting point can be addressed by spattering, just as in raycasting. In this case there is an argument for finding the nearest point on the boundary to Q, and using that as the first starting point. This is because scanning round the boundary is only a univariate haystack to find the needle in, and because the nearest point on the boundary may indeed

be the solution. Unlike raycasting, there is always at least one nearest point of a surface, and it may lie on the boundary. Starting from such a point is easier than having to work out what to do when the normal process tries to cross the boundary.

The same Newton principle gives as an iterative solution method based on the equations

$$[P-Q]\cdot \mathrm{d}P/\mathrm{d}u = 0, \quad [P-Q]\cdot \mathrm{d}P/\mathrm{d}v = 0,$$

$$\begin{bmatrix} \phi_{uu} & \phi_{uv} \\ \phi_{vu} & \phi_{vv} \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \begin{bmatrix} -[P_1-Q]\cdot \mathrm{d}P/\mathrm{d}u \\ -[P_1-Q]\cdot \mathrm{d}P/\mathrm{d}u \end{bmatrix}, \tag{7.12}$$

where

$$\begin{aligned} \phi_{uu} &= \mathrm{d}P/\mathrm{d}u\cdot \mathrm{d}P/\mathrm{d}u + [P_1-Q]\cdot \mathrm{d}^2P/\mathrm{d}u^2 \\ \phi_{uv} &= \mathrm{d}P/\mathrm{d}u\cdot \mathrm{d}P/\mathrm{d}v + [P_1-Q]\cdot \mathrm{d}^2P/\mathrm{d}u\,\mathrm{d}v \\ \phi_{vu} &= \mathrm{d}P/\mathrm{d}u\cdot \mathrm{d}P/\mathrm{d}v + [P_1-Q]\cdot \mathrm{d}^2P/\mathrm{d}u\,\mathrm{d}v \quad \text{and} \\ \phi_{vv} &= \mathrm{d}P/\mathrm{d}v\cdot \mathrm{d}P/\mathrm{d}v + [P_1-Q]\cdot \mathrm{d}^2P/\mathrm{d}v^2. \end{aligned}$$

The problem with the straightforward Newton method converging on a furthest point occurs in the parametric case too. With a sparse initial spatter it is probably more likely. It can be detected by the Newton step not being a descent direction. When this happens an appropriate response is to use the steepest descent direction instead. The actual step to use is that given by equation (7.12) with the second derivative terms omitted. This fails only when an initial estimate falls exactly on a local maximum of distance.

Nearest point on a recursive division surface In a recursive division of a surface, all those pieces can be eliminated whose nearest point is further from Q than the nearest furthest-point of any piece met so far. More sophisticated tests using the hull of surface normals would risk discarding a nearest point which was not a foot of a perpendicular.

7.4. Umbilic points

An umbilic point is one at which the principal curvatures are equal, so that the principal directions are indeterminate. Calculating the positions of any umbilics is the first stage in dividing a surface into principal patches.

This enquiry is included here as an example of the use of second derivatives of a parametric surface.

The condition for an umbilic is that for some value of the local curvature k,

$$\begin{aligned} \mathrm{d}^2P/\mathrm{d}u^2\cdot N &= k\,\mathrm{d}P/\mathrm{d}u\cdot \mathrm{d}P/\mathrm{d}u, \\ \mathrm{d}^2P/\mathrm{d}u\,\mathrm{d}v\cdot N &= k\,\mathrm{d}P/\mathrm{d}u\cdot \mathrm{d}P/\mathrm{d}v, \\ \mathrm{d}^2P/\mathrm{d}v^2\cdot N &= k\,\mathrm{d}P/\mathrm{d}v\cdot \mathrm{d}P/\mathrm{d}v. \end{aligned}$$

Eliminating k from these three equations, and noting that a well behaved surface may easily have $\mathrm{d}P/\mathrm{d}u \cdot \mathrm{d}P/\mathrm{d}v = 0$, we can derive two equations of reasonable symmetry.

$$\begin{aligned} \frac{\mathrm{d}P}{\mathrm{d}u} \cdot \frac{\mathrm{d}P}{\mathrm{d}u} \frac{\mathrm{d}^2P}{\mathrm{d}v^2} \cdot N &= \frac{\mathrm{d}P}{\mathrm{d}v} \cdot \frac{\mathrm{d}P}{\mathrm{d}v} \frac{\mathrm{d}^2P}{\mathrm{d}u^2} \cdot N, \\ \left(\frac{\mathrm{d}P}{\mathrm{d}u} \cdot \frac{\mathrm{d}P}{\mathrm{d}u} + \frac{\mathrm{d}P}{\mathrm{d}v} \cdot \frac{\mathrm{d}P}{\mathrm{d}v} \right) \frac{\mathrm{d}^2P}{\mathrm{d}u\,\mathrm{d}v} \cdot N &= \frac{\mathrm{d}P}{\mathrm{d}u} \cdot \frac{\mathrm{d}P}{\mathrm{d}v} \left[\frac{\mathrm{d}^2P}{\mathrm{d}u^2} + \frac{\mathrm{d}^2P}{\mathrm{d}v^2} \right] \cdot N. \end{aligned}$$

Provided that the surface is C^2, these two equations can be solved using exactly the same techniques as raycasting and maximum extent above. Note that N is not a constant, but itself depends on the first derivatives of P, and so the four terms of the Newton matrix become fairly complicated expressions involving first, second and third derivatives.

7.5. Reflections of points

Suppose that we want to know at what point of a surface $P(u, v)$ the reflection appears of a point Q, as seen from a point E. This requirement appears in the checking of candidate surfaces for fairness in car design.

This problem is closely related to that of nearest points; the nearest point, P, to a given point, Q, if it lies within a surface, rather than on its boundary, is also a point at which a light ray from Q to Q bounces off the surface. The difference is that there may be no solutions, or many.

The equations to be solved are that

$$[\langle P - Q \rangle + \langle P - E \rangle] \times N = 0, \tag{7.13}$$

which may be resolved into

$$\begin{aligned} [\langle P - Q \rangle + \langle P - E \rangle] \cdot \mathrm{d}P/\mathrm{d}u &= 0, \\ [\langle P - Q \rangle + \langle P - E \rangle] \cdot \mathrm{d}P/\mathrm{d}v &= 0. \end{aligned}$$

We do require second derivatives for stability, but all zeros of the left-hand side function are valid solutions, and so, unlike nearest point, situations where the local Hessian is negative or mixed are required solutions.

7.6. Summary of nilvariate interrogations

We have seen examples above of enquiries which depend only on surface position, and those which depend on derivatives of the surface. Each has its own pair of equations which can normally be solved by an initial scan, followed by Newton iteration.

Generally speaking the biggest problem is finding the global solution when local ones also exist, or all the solutions if we want all of them. This demands fine scanning, except where recursive division methods can be applied.

The other problem is the interaction of the boundary, which may cut off solutions which the iteration would lead to, or which may require constrained solutions, satisfying different equations.

Recursive division interrogation avoids all of these problems, but the prospect of setting up hulls for many higher derivatives is a somewhat daunting one.

8. Univariate interrogations

Within this section we are looking for results which take the form of curves. In general there may be multiple pieces of curve within the required solution, and so the type of each interrogation is

$$\text{surface} \times \text{auxiliary data} \mapsto \text{set of curves.}$$

In the case of surface/surface intersection, the second surface is part of the auxiliary data, though the treatment is relatively symmetric between the two surfaces.

The data structure complication which does arise is the question *'What is a curve?'*.

A curve is a set of points satisfying some definitional equation, but this set, unlike the sets of points returned from univariate interrogations, contains a continuum of points. We therefore need a finite representation.

The most concise finite representation is the definition of the curve which forms the input to the interrogation, but that is not explicit enough. The whole point of the interrogation algorithms is to calculate an explicit form which can be drawn or printed or used in subsequent interrogations without repeating much calculation.

The ideal almost-explicit form is a set of coefficients of a parametric equation, so that positions along the curve can be mass-produced by just plugging values of parameters into some point-valued expression. This is seldom possible to do exactly.

What we can do is to provide the coefficients of a parametric *approximation* to the curve, and a highly convenient form is the set of coefficients of a first-order B-spline. These take the form of a sequence of points lying on the curve, sufficiently dense that points generated by linear interpolation between them lie within some operator-specified tolerance of the true curve.

The sound procedure is to hold, as well as this approximation, the original definition, so that if more accuracy is required, the approximation may be used as an initial starting point for further iteration. If this is done, the points on the curve need only be calculated densely enough for low-resolution applications, such as screen graphics. Higher precision can be obtained as and when it is required. To make this as fast as possible, every point needs to have, not just its coordinates, but also its parameter values in whatever surfaces are involved in its definition.

The convenient unit with which to deal is the connected piece. There will be a finite number of connected pieces in the curves we deal with, and so just collecting these into a set is quite acceptable.

8.1. Plane sections

Making a plane section through a surface is the most important single interrogation. It is required for effective visual evaluation of a surface's smoothness and also for the shaping of templates to assist in manufacture.

The process of calculating such curves is closely related to that of tracing contours in geographic systems.

The result of a particular cut may be one piece of curve, none, or several. A cut through a trimmed surface may meet the boundary or it may form a closed loop. The surface may be almost tangent to the sectioning plane, resulting in a tiny contour, or, in the limit, a single point.

A plane is the set of points whose coordinates satisfy some linear equation, $P^i F_i = 0$.

In Euclidean coordinates this takes the form $[P - P_F] \cdot F = 0$, where P_F is a point lying on the plane, and F is a vector perpendicular to the plane (the plane normal).

For some purposes it is also useful to have a parametric representation of the plane. This can be created by choosing two unit vectors, R and S, perpendicular to F and to each other. Then any point generated by $P := P_F + \alpha R + \beta S$ will satisfy the plane equation.

One way to find a pair such as R and S is to find which of the coordinate directions X, Y and Z has the smallest magnitude of dot product with F. Call it D. If F is nonzero, then $D \times F$ cannot be zero. Normalize it and call it R. Then take $T \times R$, normalize it and call it S.

Plane sections through analytic surfaces The best approach to this is to treat the plane as a parametric surface and use the methods described in surface/surface intersections below.

Plane sections through parametric surfaces There are two phases. The first is the identification of the topology of the required curve and isolation of one point lying on each of the pieces. Then from each of the start points, the curve can be traced around.

Finding start points Clearly, in the case of pieces of curve which meet the boundary it is most convenient to start from a point at the boundary, and so a scan round the boundary identifying points where the boundary cuts the plane is a good start.

Finding each start point itself has two stages: the identification that a start point exists, by noting the change of sign between two sample points; and the homing in. Along the boundary this is essentially a root-finding

operation. Finding roots of general functions is a topic deserving a section in its own right. Polynomials are easier, since there is a known upper bound on the maximum number of roots, and derivatives are easily accessed.

If the boundary can be traversed in a consistent direction (for example, surface on the right if your head is pointing in the direction of the surface normal), it is possible to label each crossing of the plane depending on whether the curve is going from the positive side of the plane to the negative or *vice versa*. We only need to trace from the members of one set, not both, and the choice can be made consistent with the detail of the tracing code.

Finding start points within the interior of the surface has to be achieved by setting up some grid of sample points, and looking for sign changes between them. One approach is to determine all points where the surface normal is parallel to the plane normal, or where the boundary tangent is perpendicular to the plane normal, and set up a spanning tree between them. This uses the *maximum extent* enquiry described above.

Marching along the intersection The process of taking one step along the curve, from one known point to the next, currently unknown, is a univariate interrogation in its own right. The point we are seeking lies in the surface – that will be ensured by working in the parameter plane – and also in the cutting plane. The third condition that we need, to tie it down, is that the step along the curve is appropriate.

Suppose that the requirement is to take steps of some specific length, l_R. Then a good approximation is that the new point lies on a plane perpendicular to the tangent to the section at the start point and a distance l_R from it. We can now use the refinement step of the raycasting interrogation described above to home in on the new point, using the parameter values of the initial point as our initial estimate.

The same general strategy applies when more sophisticated step-length strategies are used, because each step-length strategy can be expressed as a locally linear condition equivalent to a second plane. Because the position of the step-length control plane typically moves when we know more about the surface near the solution, it is not worth taking the raycasting refinement more than one step at a time. Just as in constrained optimization by penalty functions (Wright, 1992), a single step gives enough precision to define the problem closer.

Step-length criteria Even in the case where equally spaced points are required, the normal to the step control plane can be updated at each step, to point along the chord from the last point to the current estimate of this one.

It is more typical, however, to require that generated points should be denser in regions of high curvature of the curve that they represent. There are a number of possible rules for this increase in density.

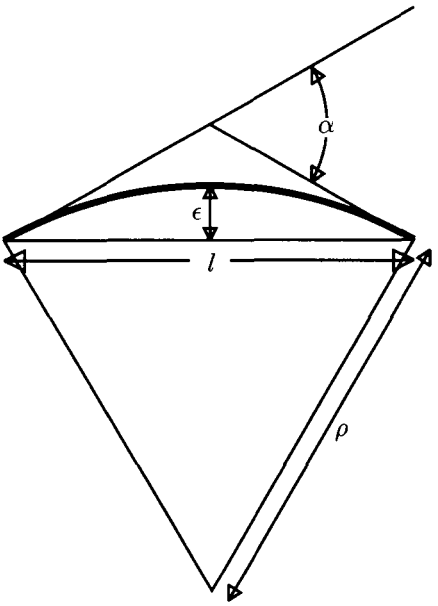

Fig. 13. Relationship between α, l, ϵ and ρ.

One of the simplest is that angles between the tangents at successive data points should make equal steps. If α_R is the required angle and ρ is the local radius of curvature, then the step length required is $2\rho \sin \alpha_R$, which for the angles likely to be used in this criterion is essentially $2\rho\alpha_R$. Now ρ will vary from point to point along the intersection, and so it needs to be estimated.

As soon as one step has been taken we can use the same equation in reverse. Let the tangent (given by $\langle F \times [\mathrm{d}P/\mathrm{d}u \times \mathrm{d}P/\mathrm{d}v]\rangle$) at the previous point be T_{i-1}, and the tangent at the current estimate be T_i. Then α_i, the actual angle between successive tangents, is close to $|T_{i-1} \times T_i|$. Similarly, l_i, the actual step taken, is $|P_i - P_{i-1}|$. We can therefore estimate $\rho = l_i/2\alpha_i$

Thus the required step length is this region is $l = l_i\alpha_R/\alpha_i$.

The new step control plane has $\langle P_i - P_{i-1}\rangle$ as its normal vector and $P_i + l\langle P_i - P_{i-1}\rangle$ as its sample point.

In situations where the sectioning plane is almost tangent to a surface (of negative Gaussian curvature), it is possible for two branches of the curve to come very close indeed to each other. It is essential to check that $T_i \cdot T_{i-1}$ is positive. A negative value is an indication that such a singularity is nearby, and that at very minimum the value of l_R should temporarily be reduced. A better response to this situation is to locate the singularity explicitly, model the situation round it and proceed using that information (Bajaj *et al.*, 1988).

The ideal step-length rule from the control of approximation point of view is that the error at the mid-chord should be constant from step to step, and

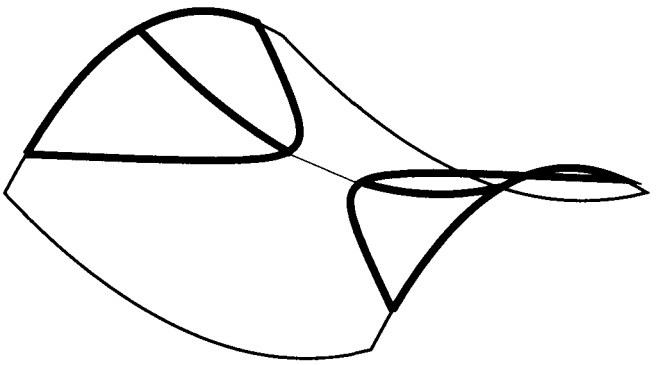

Fig. 14. Close-to-singular plane section.

equal to the desired tolerance. This corresponds to the length rule $l^2 = 8\epsilon\rho$, where ϵ is the required error. However, it is safer to temper this rule with a maximum step length for use in almost flat areas, and with a maximum angle for use in very tight corners.

A serendipitous combination is to require that rather than $l/l_R = 1$ or $\alpha/\alpha_R = 1$ conditions, we apply

$$l/l_R + \alpha/\alpha_R = 1.$$

At very small curvatures the actual α values will be small, and so the behaviour is dominated by the l term, and similarly when the curvatures are high, the angle term dominates.

For a range of curvatures about an order of magnitude wide, centred on $2\alpha_R/l_R$, the chord height error is close to $\alpha_R l_R/4$, and so it is possible to determine appropriate α_R and l_R from the mean principal curvature (determined when the surface is defined) and the required tolerance.

Handling the boundary What happens when the raycasting refinement step reports that it has hit a boundary?

Because surfaces are sometimes defined by plane cross-sections, and it is unreasonable to prevent those sections from being calculated, the first response is that if the next estimate returned is an acceptable next point for the curve (within tolerance of the sectioning plane, and with a step length within, say, 10% of the current ideal), it should be accepted, and the section continued as normal.

If the section piece started from a boundary, it is likely that the end of the piece has been reached. The accurate final point of the piece should be determined by homing in along the boundary in the neighbourhood of the point just found.

Handling failure What happens when the raycasting refinement step reports that it has failed to find a point which reduces the initial error?

This is almost certainly due to a C^0 discontinuity in the surface, and so the appropriate response is to locate the discontinuity and treat it as a local piece of boundary. If knowledge of the structure of the surface is accessible (for example, a knot-vector in the case of B-spline surfaces or the offset parameters in the case of a toroidal offset surface with a surface normal near the axis of the toroid) this can be used to define the local boundary. If not, special code may have to be written to explore the vicinity and deduce what the boundary should be.

Handling a closed loop When a curve piece starts at the boundary it also finishes at the boundary, and the mechanism described above covers terminating the piece. If it starts at a point in the interior, it may reach a boundary, (in which case work has been duplicated, because the same piece should have been found from a boundary start point), or else it returns to its own start point. However, it is most unlikely to return exactly to the start point, or even to within precision tolerance of it. What will typically happen is that at some stage it will overshoot. Let S be the start point, and P_i and P_{i+1} be the previous and current points. The simplest test to make is to see whether S lies in the sphere whose diameter is P_iP_{i+1} which is just

$$[P_i - S] \cdot [P_{i+1} - S] < 0$$

If this test indicates that S is inside the sphere, it should next be checked that the step from P_i to S would have a step-length criterion less than the step from P_i to P_{i+1}. If it does, then the loop has closed, and S can be accepted as the final point. If not, then we were unlucky, and started the curve piece at a place where two leaves approached each other closely.

Plane sections through recursive division surfaces In a recursive subdivision of a surface, any piece whose hull does not intersect the plane may be eliminated. This is reasonably efficient, particularly if the hulls are aligned with a face parallel to the plane.

A fuller elaboration of the detail of a recursive division bivariate interrogation follows in the next section (8.2), on surface/surface intersections. The plane section case is somewhat simpler.

8.2. Surface/surface intersection

This is normally regarded as the 'difficult' interrogation, but in fact it is very much of a kind with the others.

The main complication from the point of view of this section is that there are six possible combinations of the types of surface we consider. To reduce this complication we shall ignore the two most awkward combinations, of recursive division surfaces with other kinds.

Intersection of a parametric surface with an analytic This is the

easiest case to deal with, because it is an almost trivial generalization of the plane section algorithm. Wherever we evaluate a parametric surface point we can immediately carry out a Taylor series expansion of the function which defines the analytic surface. The first two terms of this Taylor series define a local plane, which is used during that step as the local view of the analytic surface.

Intersection of two analytic surfaces This case can follow the same general principle, but has to be carried through in three variables, the space coordinates, instead of two.

Once a start point is found, the stepping process works by setting up a stepping direction vector as the cross product of the two surface normals. A step is made along this of the required amount, and the surfaces re-evaluated. The stepping plane and the two local linearizations are now three planes, the intersection point of which is the next candidate curve point. This iteration continues until the point found lies within the computational tolerance of both surfaces.

The starting point can be found by a slight variation on this. An arbitrary point is taken, and the Taylor planes evaluated there. Also evaluated is a third plane, whose normal is the cross product of those of the Taylor planes, and which passes through the arbitrary starting point. The intersection of these three planes gives a better approximation to a starting point. In typical situations where analytic shapes are encountered, the start point for a particular intersection will be known to lie on a third surface: in such cases, the intersection of three Taylor planes will converge to one of the intersections of the three surfaces.

The main problem is detecting the closure of a closed loop intersection without running the risk of aborting too early when two leaves of the intersection come close together. Related to this is taking the correct trace through singular situations where two surfaces are tangent to each other.

While this approach may be necessary if the analytic surfaces are fully general, there is another approach which may be applied if it is known that the only surfaces which will be met are planes, quadrics and tori. This uses the algebraic geometry result that the intersection curve of any two such surfaces may be parametrized by a curve of genus at most three and order at most eight. The genus three property means that for each value of curve parameter we can find four points on the curve by solving a quartic equation. The problem reduces to stepping along the curve by solving a quartic at each step. This could use the Tartaglia closed form, or else could just use the previous roots as first approximations for an iterative root finder.

This means tracing complex roots, but, although I have not implemented this myself, I do not regard it as particularly difficult. An interesting point is that from the algebraic geometry point of view, the more singular the

configuration becomes, with points of common tangency between the surfaces, the lower the genus gets, and therefore the simpler the algebraic solution. See Farouki (1987a,b) for further elaboration on this point.

Intersection of two parametric surfaces This case takes us into even higher dimensionalities, as we have to trace a path in both parametric spaces. This is not a problem. Hoffman (1989) argues strongly that numerical tracing in spaces of many dimensions gives simpler, more reliable code than trying to eliminate the bulk of them algebraically. A useful tactic here, however, breaks down the system of four equations into two pairs, by resolving the equations

$$P_1(u_1, v_1) = P_2(u_2, v_2)$$

along the two surface normals and their mutual perpendicular.

This can be viewed geometrically as finding the intersection of one parametric surface with the local tangent plane of the other and the stepping plane, and then repeating the process for the other.

This means that the code for the refinement step of ray casting can be used yet again.

The main problem here is finding a start point which lies in (or on the boundary of) both surfaces. If we merely trail round one boundary we are likely to find a point which is outside the boundary of the other.

Intersection of two recursive division surfaces This case shows the recursive division method at its most elegant. In the outer product of two recursive division surfaces we may eliminate any pair of pieces whose hulls do not intersect.

This algorithm needs care in implementation to achieve efficiency. Specific points are:

- not to throw away any information;
- to use simple hulls, such as bounding boxes;
- to rebuild the hulls bottom up rather than just top down;
- to split only one surface, in only one direction, at each stage;
- to choose which surface to split;
- to choose which direction to split it in;
- to rejoin the constructed pieces of curve as soon as possible on the way out of the recursion.

The algorithm applying these tenets (based on Nasri (1987)) is no more complicated than some of the baroque methods which have been developed for making sense of the heap of small pieces of curve which results from not obeying the last one.

It requires as a data structure for each surface a binary tree in which each node holds the local data from which subdivision takes place, a hull, a flag

which labels the node as primitive, subdivided or not subdivided, and if subdivided, the direction of subdivision and the parameter value at which the subdivision takes place.

It requires as a data structure for the pieces of curve a linked list structure within each curve. Each piece also has data identifying the logical place in the subdivision hierarchy of each of its ends, and these ends are linked together into chains accessible from the appropriate nodes of the surface trees.

Algorithm

```
Procedure Intersect(s1:surface, s2:surface):curveset
begin
        if      disjoint(s1,s2)
        then nil
        elseif  primitive(s1) and primitive(s2)
        then    primitiveintersection(s1,s2)
        else    choosewhich(s1,s2,which)
                choose direction(which, dirn)
                if    which = s1
                then Split( s1, dirn, s1a, s1b)
                        Combine( Intersect(s1a, s2),
                                 Intersect(s1b, s2) )
                else  Split( s2, dirn, s2a, s2b)
                        Combine( Intersect(s1, s2a),
                                 Intersect(s1, s2b) )
                endif
        endif
end
```

The procedure *disjoint* simply tests whether hulls overlap.

The procedure *primitive* returns TRUE if is argument is labelled as primitive.

The procedure *choosewhich* decides which of the two surfaces to split. This is based on whether either is primitive (they will not both be) and if neither is, on the relative sizes of the two hulls. The nonprimitive with the largest extent in any direction will be chosen for splitting.

The procedure *choosedirection* decides which way to split. Again this will be based on which direction of split does most to reduce the largest hull dimension. If the surface is already split, this procedure merely reports the direction previously chosen.

The procedure *Split* implements the splitting if it has not already taken place. As soon as the two new hulls are available, the hull of the surface node being split is updated, to bring the bounds in, if possible. If this results

in tighter bounds they are propagated upwards until either the top of the tree is reached or no narrowing takes place. The two halves are examined for simplicity, interpreted simply as planarity, and if either half is found to be close to planar, it is marked as primitive, otherwise as undivided.

The procedure *Combine* takes those pieces of curve which are linked to the common edge of the two halves of the split surface, and sews them together into larger pieces. Where one side has been split to a deeper level than the other, its endpoint is chosen to give the coordinates of the common point.

To gain most advantage from this approach, the subdivisions will not be thrown away when a single intersection has been calculated. A great deal of work can be saved by keeping all the surface nodes for the intersection of $s1$ with $s3$ which is likely to be required next.

Clearly this arrangement is highly demanding of memory, but sufficient memory for reasonable problems is now available on current workstations. Further, if memory does become exhausted, surface nodes can be thrown away and their memory reused, provided that the immediate parent is flagged as no longer subdivided. If the data are required again, it will be recomputed, and in the meantime the tighter bounds on the parent are still having their good effect.

The only improvement available in the literature (Sederberg and Meyers, 1988) and (de Montaudouin, 1989) on the above scheme uses the concept of *co-simplicity* to fine-tune and accelerate the point at which a nonrecursive method can be used. If the surface normals of $s1$ can be bounded into a cone, $c1$, and those of $s2$ into a cone $c2$, and the cones $c1$ and $c2$ are disjoint (except at the origin), then there can at most be one piece of intersection curve between $s1$ and $s2$, and it will not form a closed loop. It is therefore safe to take recourse to marching methods to evaluate the detail of the piece of intersection.

This refinement requires more data per node, to hold the bounds on the surface normal, but it should permit the recursion to be truncated much earlier, and also give better resolution when two surfaces are almost tangent. It affects the criteria used for deciding which surface to split and which way, because it becomes just as important to reduce the surface normal spread quickly as to reduce the physical extent.

The two awkward cases which we have omitted are best handled by multiple representations. The analytic surfaces which are found in geometric modelling systems also have parametric forms, and can then be converted on, to a recursive division representation.

8.3. Reflection lines in a parametric surface

This interrogation determines the curves on a surface at which light rays are reflected from some curve C in space to an eye at point E.

It is used in validation of surfaces in car styling. It is also an example of another paradigm for univariate interrogation, in which nilvariate enquiries are made recursively until the necessary density is reached.

Suppose that C is a curve stored in exactly the same form as all the results of our univariate enquiries described so far. It has a start point, which can be used as auxiliary data for the nilvariate calculation of a reflection point. Each point in the resulting set of points is a start point for a reflection curve, and these should be taken one at a time.

Imagine a point moving along C from that start point. In fact C is parametrized, and so we can formalize this movement by differentiating C with respect to parameter. For high precision we should differentiate the true curve underlying C, but for graphics purposes the piecewise linear approximation will be quite adequate. The derivative is $\mathrm{d}C(t)/\mathrm{d}t$

As the point on C moves, the result of carrying out the nilvariate reflection also moves, and differentiating the reflection equation (equation (7.13) above) gives $\mathrm{d}P/\mathrm{d}C(t)$, and the chain rule then gives $\mathrm{d}P/\mathrm{d}t$. This is the tangent to the reflection curve.

We can judge an appropriate increment in t from the magnitude of $\mathrm{d}P/\mathrm{d}t$, and can then step along C using a bracketting in t to find points on the reflection curve which satisfy the step-length rules. At each step we can use the previous point on the curve as an initial estimate of the next, thus using coherence to simplify the process.

This method is probably adequate for judging car body panels, where the reflection curves do not hit the boundary, and where there is only one reflection of each C.

However, a full algorithm needs to handle correctly the cases where reflections fall off the edges of the surface, and also those cases where the reflection falls back on again. It also needs to cater for the bifurcation which can happen when the surface has an inflexion.

The first problem is fairly straightforward, since tracing can just stop when a boundary is reached. The second needs to be addressed by scanning the boundary and identifying the points and t-values at which reflections of C-points lie on the boundary.

The third requires the explicit evaluation of the parabolic lines of the surface, which can then be treated in the same way as the boundary as a source of start points for new bits of reflection curve.

Other interrogations using the same paradigm will require some other specific internal 'boundaries' depending on the detail of their equations.

8.4. Geodesics across a surface

A geodesic is the path between two points of a surface which lies entirely within the surface and whose length is least. Calculus of variations shows

that it is the solution of a differential equation which states that the component of curvature within the tangent plane (the geodesic curvature) is zero.

Geodesics map into straight lines during the flattening of a developable, and so this calculation is applied in the determination of development. It is also a good example of a differential equation interrogation.

Geodesics across an analytic surface Let the two ends be P_1 and P_2. We can determine the surface normals N_1 and N_2 at these points from the gradient of the function defining the surface.

A surprisingly accurate cubic approximation to a geodesic is constructed by postulating that the Bezier control points of this cubic are given by

$$B_1 = (2P_1 + P_2)/3 + \alpha N_1 + \beta N_2, \tag{8.1}$$

$$B_2 = (P_1 + 2P_2)/3 + \gamma N_1 + \delta N_2, \tag{8.2}$$

and choosing the unknowns, α, β, γ and δ from the conditions that the tangent at each end must be perpendicular to the normal there, and that the local osculating plane at each end contains the surface normal. From these we can deduce that $\delta = 2\beta$ and $\alpha = 2\gamma$, and thence that

$$\begin{bmatrix} 2N_1 \cdot N_1 & N_1 \cdot N_2 \\ N_1 \cdot N_2 & 2N_2 \cdot N_2 \end{bmatrix} \begin{bmatrix} \gamma \\ \beta \end{bmatrix} = \begin{bmatrix} -D \cdot N_1/3 \\ D \cdot N_2/3 \end{bmatrix}. \tag{8.3}$$

The matrix on the left is always nonsingular, and so this equation can be solved to give the required control points.

The cubic may well depart from the surface in its interior. Evaluation of the distance of points evaluated on it from the surface gives the first measure of error.

The second measure of error is to evaluate a point at the parametric centre of the curve, and project it back on to the surface using the nearest-point algorithm. Then apply the same process as above, to determine an approach direction and a departure direction at this central point. The angle between these directions gives another measure of error. Let this angle be θ, and the distance between the two ends be l. A lateral movement of the midpoint (that is, a movement in the tangent plane there and in a plane perpendicular to the line joining P_1 and P_2) of $l\theta/4$ will approximately correct this mismatch.

However, inserting additional points at the quarter points would probably discover the need for a further correction: the best estimate of the correction needed if we inserted really densely is $l\theta/2$. The number of points necessary for insertion to give a solution accurate to a given tolerance ϵ is the square root of (estimated error/required error). These can be inserted either all at once or in binary stages, with a relaxation at every step correcting the positions.

Geodesics across a parametric surface An exactly similar approach can be taken over a parametric surface; the only additional stage is resolving back the components of the vector $B_1 - P_1$ into components in the parameter plane.

Because $B_1 - P_1$ lies in the tangent plane at P_1, it must be of the form

$$\mathrm{d}P/\mathrm{d}u\,\delta u + \mathrm{d}P/\mathrm{d}v\,\delta v.$$

To evaluate δu and δv take the triple product of $B_1 - P_1$ first with $\mathrm{d}P/\mathrm{d}v$ and N_1, then with $\mathrm{d}P/\mathrm{d}u$ and N_1. These give scalar equations

$$\delta u = [B_1 - P_1, \mathrm{d}P/\mathrm{d}v, N]/[\mathrm{d}P/\mathrm{d}u, \mathrm{d}P/\mathrm{d}v, N],$$
$$\delta v = [\mathrm{d}P/\mathrm{d}u, B_1 - P_1, N]/[\mathrm{d}P/\mathrm{d}u, \mathrm{d}P/\mathrm{d}v, N].$$

The cubic is now set up in parameter space, and a similar procedure of interpolating a central point and checking the accuracy is applied.

8.5. Lines of curvature

At any point of a surface (except an umbilic) there are two directions of *principal curvature*. These are the directions in which the curvature of a normal section is largest or least, and they are also directions in which there is no twist.

A line of curvature is a curve whose tangent is always along a local direction of curvature. Tracing such a curve is essentially the numerical solution of a first-order differential equation, and the obvious procedure is marching along it.

The most convenient equation to solve is that corresponding to the no-twist condition. Let T be a tangent to the curve at P, locally parametrized by t, and N the surface normal. As a point moves along T, the surface normal becomes

$$N(t + \delta t) = N + \mathrm{d}N/\mathrm{d}t\delta t,$$

and the condition of no twist is that $\mathrm{d}N/\mathrm{d}t$ lies in the plane of N and T. (N and T are always perpendicular, and so they always span a plane.)

$$[N, T, \mathrm{d}N/\mathrm{d}t] = 0. \tag{8.4}$$

Lines of curvature across an analytic surface At the start point, P, of the curve we evaluate the local surface normal, N, and the tensor $\mathrm{d}^2 f/\mathrm{d}P^2$ which may be viewed as $\mathrm{d}N/\,\mathrm{d}P$.

The direction, T, we need to march in satisfies the equations

$$T \cdot N = 0, \tag{8.5}$$
$$[N, T, dN/\mathrm{d}P \cdot T] = 0. \tag{8.6}$$

This is homogeneous in T, and so any multiple of a solution is also a

solution. In particular, the exact reverse is also a solution. It is also quadratic in T, and so we expect there to be two distinct solutions.

Unless the context gives a reason for choosing one of the four rather than another, all four need to be treated equally, as start directions for lines of curvature.

However, once a particular path has been chosen, further steps can always be made in the direction most coherent with that of arrival at the current point.

The actual solution of the equations is easily carried out by constructing two vectors spanning the plane perpendicular to N and using coordinates in this plane as two freedoms, thus automatically satisfying the first equation. The second becomes a homogeneous quadratic equation in the two coordinates, whose solution ratios give the possible directions for T.

Once a direction is determined, the same methods of step length control apply as in previous curve interrogations.

The difference here is that there is no way of correcting back onto the true curve in any absolute sense, as there was when intersecting with other surfaces. Now the effect of making steps along the tangent to the curve at each point is to drift away from the true curve fairly rapidly. This is a familiar effect in solution of ordinary differential equations, and the appropriate response is the predictor–corrector paradigm (see Allgower and Georg (1993)). Because we have to iterate at each point to get back on to the surface itself, the corrector step gives no extra computing.

Lines of curvature across a parametric surface In this case we have our vectors spanning the tangent plane ready-made in $\mathrm{d}P/\mathrm{d}u$ and $\mathrm{d}P/\mathrm{d}v$. The equivalent equation to (8.5) is

$$\begin{bmatrix} \delta u & \delta v \end{bmatrix} \begin{bmatrix} \phi_{uu} & \phi_{uv} \\ \phi_{vu} & \phi_{vv} \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = 0 \tag{8.7}$$

where

$$\begin{aligned}
\phi_{uu} &= N \cdot \frac{\mathrm{d}^2P}{\mathrm{d}u\,\mathrm{d}v} \frac{\mathrm{d}P}{\mathrm{d}u} \cdot \frac{\mathrm{d}P}{\mathrm{d}u} - N \cdot \frac{\mathrm{d}^2P}{\mathrm{d}u^2} \frac{\mathrm{d}P}{\mathrm{d}u} \cdot \frac{\mathrm{d}P}{\mathrm{d}v}, \\
\phi_{uv} &= N \cdot \frac{\mathrm{d}^2P}{\mathrm{d}u\,\mathrm{d}v} \frac{\mathrm{d}P}{\mathrm{d}u} \cdot \frac{\mathrm{d}P}{\mathrm{d}v} - N \cdot \frac{\mathrm{d}^2P}{\mathrm{d}u^2} \frac{\mathrm{d}P}{\mathrm{d}v} \cdot \frac{\mathrm{d}P}{\mathrm{d}v}, \\
\phi_{vu} &= N \cdot \frac{\mathrm{d}^2P}{\mathrm{d}v^2} \frac{\mathrm{d}P}{\mathrm{d}u} \cdot \frac{\mathrm{d}P}{\mathrm{d}u} - N \cdot \frac{\mathrm{d}^2P}{\mathrm{d}u\,\mathrm{d}v} \frac{\mathrm{d}P}{\mathrm{d}u} \cdot \frac{\mathrm{d}P}{\mathrm{d}v}, \\
\phi_{vv} &= N \cdot \frac{\mathrm{d}^2P}{\mathrm{d}u^2} \frac{\mathrm{d}P}{\mathrm{d}u} \cdot \frac{\mathrm{d}P}{\mathrm{d}v} - N \cdot \frac{\mathrm{d}^2P}{\mathrm{d}u\,\mathrm{d}v} \frac{\mathrm{d}P}{\mathrm{d}v} \cdot \frac{\mathrm{d}P}{\mathrm{d}v}.
\end{aligned}$$

Correction of each step to minimize drift is still necessary even though evaluation from the parameters ensures that generated points lie on the surface.

8.6. Summary of univariate interrogations

We have seen above examples of both algebraic equations, where accuracy can be maintained at each step, independent of what has come before, and of differential equations, whose tracing is possible from any start point, and in which numerical errors will inexorably, if slowly, build up.

Marching methods work well for both in situations where the geometry is actually well conditioned, and normal applications in manufacturing tend to give these situations.

In marching methods we first need to identify the topology of the solution; how many pieces the solution has, and how they interact with the boundary of the surface(s). Then we step along each piece, choosing the step length to match the local curvature of the result. Within each step we first find a first approximation for the next point and then refine it. Refinement can often use the refinement step from raycasting.

If, however, robustness is important the recursive subdivision techniques are more appropriate to algebraic equations and the differential equations should be cast ideally as boundary value, rather than initial value questions.

9. Bivariate interrogations

Within this section we are looking for results which map an entire surface into a new form. The examples are concerned with graphics.

9.1. Rendering a surface

With the advent of workstations capable of displaying a wide range of colours, it has become desirable to be able to compute exactly what colour each pixel should be in an image of the objects our software deals with.

One technique for this is raytracing, which uses geometric optics to determine what can be seen at each pixel.

The first level of this technique merely uses the nilvariate raycasting algorithm to find out what point on a given surface lies at the intersection of the surface with a ray through the eye and a given pixel.

In a typical model, with many surfaces, there is also the process of choosing the nearest surface, as well as that of choosing the nearest intersection on the given surface.

Once the nearest intersection has been found, its illumination is determined by applying an illumination model, which uses the positions and colours of the lights, the positions of the surface point and the eye, the surface normal, and the various reflection coefficients of the material of which the surface purports to be made. Such models typically use separate models of diffuse and specular reflection, and, if the coefficients are well-chosen, can give surprisingly recognizable impressions of different materials.

The process may be taken further, by starting either a reflection ray or a refraction ray (or both) from the visible surface point, and repeating the process to find out what can be seen reflected (or refracted) in the surface.

This capability is usually shown off in images of glass or shiny metal objects. It just uses the same raycasting code again.

Finally, for duller materials, shadows can also be determined using the same code. A point on a surface is illuminated if it is visible from the light. A ray can be fired from the light towards each point visible from the eye. If the first intersection is the visible point it is illuminated, if not, not.

Multiple lights just means lots more calls to the raycaster.

This technology runs out of steam when diffuse reflections from one surface to another dominate the illumination, as in images of domestic interior scenes. A simultaneous solution for the illumination levels on all surfaces then needs to be invoked. This technique is called *'luminosity'* (Hall, 1990).

Nor is raycasting the fastest technique with modern workstations which have significant amounts of special purpose hardware designed for rendering. These displays are driven, not pixel by pixel, but facet by facet, where facets are small pieces of surface, small enough to be treated as plane.

In order to keep the number of facets small, while still giving the impression of a smooth surface, there are two ways in which graphics workstations typically cheat. The first is called Gouraud shading, in which a true surface normal is evaluated at each vertex of each facet; the illumination model is applied there, and the illumination value is interpolated linearly across the facet. The second is called Phong shading. Again a true surface normal is evaluated at every vertex, and then an effective surface normal vector is interpolated across the facet, with the illumination model being applied at every pixel. The application of such methods can be detected in images which have smooth surfaces with polygonal edges.

In order to drive such powerful displays, it is necessary to split each surface up into facets.

9.2. Facetting a surface

One technique, applicable to both parametric and recursive division surfaces, is to base the facetting on the parametric structure of the surface.

In the simplest parametric case, a regular subdivision is made on a regular parametric grid, giving four-sided facets. If any facet is too far out of plane, it is just subdivided further.

In the recursive subdivision case, each piece of surface is examined for planarity. If it is flat enough it is issued as a facet, if not it is subdivided further. The planarity test can use surface normal hulls or else it can set up an approximate surface normal for an entire piece and just measure the upper bound on thickness along this direction.

Another approach is to divide the surface into triangular pieces, triangles being flat by definition. The problem here is that some parts of the surface will require small triangles, others can accept large ones, if the criterion on triangle size is economically matching the true surface within a stated tolerance.

Such a triangulation can be used for communicating models to certain 'rapid prototyping' machines, and also as a discretization for certain types of aerodynamic or electromagnetic analyses, as well as for graphics.

A method which has been used in mesh generation for finite-element analysis (Cavendish, 1974) is first to create vertices all round the boundary at an appropriate density, then to create a Delaunay triangulation of those vertices, and finally to insert additional vertices, always updating the Delaunay triangulation, until every triangle is small enough.

The interesting part of this process is that, while many different optimality criteria give the same triangulation of a set of points in two dimensions, there is no clean equivalent in three dimensions. It is better therefore to do the triangulation in the parameter plane.

However, the distortion introduced by the local affineness of the mapping from parameter space to real space means that a triangulation which is Delaunay in parameter space is likely to be a very poor one when mapped into object space. It is necessary to compensate for this distortion.

There are just three places within the overall process where compensation is necessary. Within the Delaunay process itself there is a 'swap test' which decides whether a pair of triangles forming a quadrilateral should be swapped to split the quadrilateral by its other diagonal.

Then there is the decision as to whether a triangle is acceptable in the final tesselation, or whether it should have an extra vertex inserted in it, and, finally, if an extra vertex is to be created, there is the computing of where it should go.

In each case, the local configuration can be mapped from parameter space into an orthonormal coordinate system which can be thought of as being of the tangent plane at the centroid of the three or four points concerned.

This gives well proportioned triangles, even on surfaces where $\mathrm{d}P/\mathrm{d}u$ and $\mathrm{d}P/\mathrm{d}v$ vary widely over the surface and are far from orthonormal.

The final step, of making the triangles' proportions and densities fit the local surface curvatures, is achieved by using, not an orthonormal system in the tangent plane, but a system which is orthonormal with respect to the sign-corrected second derivative matrix.

Take the principal curvatures and principal directions of curvature at a point on the surface, and scale the vectors by the square roots of the magnitudes of the radii of curvature. These two vectors are now conjugate with respect to an ellipse which is either the Dupin indicatrix, or else has the same maximum deflection from the tangent plane on both sides. Under the

necessary final transformation, this ellipse is the image of a circle. If the maximum deflection from the tangent plane is the required tolerance, it is the image of a unit circle. Any triangle inscribed in it has an error no greater than the tolerance, and if the triangle is well enough shaped to include its (mapped) circumcentre, the error is equal to the specified tolerance.

REFERENCES

E.L. Allgower and K. Georg (1993), 'Continuation and path following', *Acta Numerica 1993*, Cambridge University Press (Cambridge), 1–64.

S. Aomura and T. Uehara (1990), 'Self-intersection of an offset surface', *Comput. Aided Des.* **22**, 417–422.

A.P. Armit (1971), 'Curve and surface design: using multipatch and multiobject design systems', *Comput. Aided Des.* **3**, 3–12.

C.L. Bajaj (1989), 'Geometric modelling with algebraic surfaces', in *The Mathematics of Surfaces III* (D.C. Handscomb, ed.), Clarendon (Oxford), 3–48.

C.L. Bajaj, C.M. Hoffman, J.E. Hopcroft and R.E. Lynch (1988), 'Tracing surface intersections', *Comput. Aided Geom. Des.* **5**, 285–308.

R.E. Barnhill (1974), 'Smooth interpolation over triangles', in *Computer Aided Geometric Design* (R.E. Barnhill and R.F. Riesenfeld, eds), Academic Press (New York), 45–70.

R.E. Barnhill, G. Farin, M. Jordan and B.R. Piper (1987), 'Surface/surface intersection', *Comput. Aided Geom. Des.* **4**, 3–16.

R.E. Barnhill and R.F. Riesenfeld, eds (1974), *Computer Aided Geometric Design*, Academic Press (New York).

C. Bell, B. Landi and M. Sabin (1974), 'The programming and use of numerical control to machine sculptured surfaces', in *Proceedings 14th MTDR Conference*, Macmillan (London), 233–238.

P. Bezier (1971) 'An existing system in the automobile industry', *Proc. R. Soc. London* A **321**, 207–218.

L. Biard and P. Chenin (1991), 'Ray tracing rational parametric surfaces', in *Curves and Surfaces*, (J. Laurent, A. le Mehaute and L.L. Schumaker, eds), Academic Press (New York), 37–42.

M.I.G. Bloor and M.J. Wilson (1989), 'Generating blend surfaces using partial differential equations', *Comput. Aided Des.* **21**, 165–171.

M.I.G. Bloor and M.J. Wilson (1990), 'Representing PDE surfaces in terms of B-splines', *Comput. Aided Des.* **22**, 324–331.

C. de Boor (1962), 'Bicubic Spline Interpolation', *J. Math. Phys.* **41**, 212–273.

C. de Boor (1987), 'B-form basics', in *Geometric Modelling: Algorithms and New Trends* (G. Farin, ed.), SIAM (Philadelphia), 131–148.

C. de Boor (1993a), *B-Spline Basics* in *Fundamental Developments of Computer Aided Geometric Modelling* (L. Piegl, ed.), Academic Press (New York), 327–350.

C. de Boor (1993b), 'Multivariate piecewise polynomials', *Acta Numerica 1993*, Cambridge University Press (Cambridge), 65–110.

H. Burger and R. Schaback (1993), 'A parallel multistage method for surface/surface intersection', *Comput. Aided Geom. Des.* **10**, 277–292.

E. Catmull and J. Clark (1978), 'Recursively generated B-spline surfaces on arbitrary topological meshes', *Comput. Aided Des.* **10**, 350–355.

J.C. Cavendish (1974), 'Automatic triangulation of arbitrary planar domains for the FE method', *Int. J. Numer. Meth. Engrg* **8**, 679–696.

P.M. Cohn (1961), *Solid Geometry*, Routledge and Kegan Paul (London).

J.L. Coolidge (1963), *A History of Geometrical Methods*, rep. 1963 of 1938 book, Dover (New York).

S.A. Coons (1967), 'Surfaces for computer aided design of space forms', MAC–TR–41, Massachusetts Institute of Technology.

W. Dahmen (1989) 'Smooth piecewise quadric surfaces', in *Mathematical Methods in Computer Aided Geometric Design* (T. Lyche and L.L. Schumaker, eds), Academic Press (New York), 181–194.

D. Doo and M.A. Sabin (1978), 'Behaviour of recursive division surfaces near extraordinary points', *Comput. Aided Des.* **10**, 356–362.

J.P. Duncan and S.G. Mair (1980), *Sculptured Surfaces in Engineering and Medicine*, Cambridge University Press (Cambridge, UK).

R.A. Earnshaw, ed. (1985), *Fundamental Algorithms for Computer Graphics*, NATO ASI F17, Springer (Berlin).

H. Einar and E. Skappel (1973), 'FORMELA: A general design and production system for sculptured products', *Comput. Aided Des.* **5**, 68–76

L.P. Eisenhart (1960), *Coordinate Geometry*, 1938 repub 1960 Dover (New York).

G. Farin (1982), 'Designing C1 surfaces consisting of triangular cubic patches', *Comput. Aided Des.* **14**, 253–256.

G. Farin, ed. (1987), *Geometric Modelling: Algorithms and New Trends*, SIAM (Philadelphia).

G. Farin (1988), *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press (New York).

R.T. Farouki (1987a), 'Direct surface section evaluation', in *Geometric Modelling: Algorithms and New Trends* (G. Farin, ed.), SIAM (Philadelphia), 319–334.

R.T. Farouki (1987b), 'Numerical stability on geometric algorithms and representations', in *Mathematics of Surfaces III* (D. C. Handscomb, ed.), 83–114.

R.T. Farouki and V.T. Rajan (1987), 'On the numerical conditioning of polynomials in Bernstein form', *Comput. Aided Geom. Des.* **4**, 191–216.

R.T. Farouki and V.T. Rajan (1988a), 'Algorithms for polynomials "in Bernstein form" ', *Comput. Aided Geom. Des.* **5**, 1–26.

R.T. Farouki and V.T. Rajan (1988b), 'On the numerical condition of algebraic curves and surfaces (part 1)', *Comput. Aided Geom. Des.* **5**, 215–252.

I.D. Faux and M.J. Pratt (1979), *Computational Geometry for Design and Manufacture*, Ellis Horwood (New York).

J. Ferguson (1964) 'Multivariable curve interpolation', *JACM* **11**, 221–228.

J. Ferguson (1993), 'F-methods for freeform curve and hypersurface definition', in *Fundamental Developments of Computer Aided Geometric Modelling* (L. Piegl, ed.), Academic Press (New York), 99–116.

D. Filip, R. Magedson and R. Markot (1986), 'Surface algorithms using bounds on derivatives', *Comput. Aided Geom. Des.* **3**, 295–312.

T. Garrity and J. Warren (1989), 'On computing the intersection of a pair of algebraic surfaces', *Comput. Aided Geom. Des.* **6**, 137–154.

K.F. Gauss (1828), *General Investigation of Curved Surfaces* (J.C. Morehead and A.M. Hiltebeitel, trans.), reprinted Raven Press, 1965.

W.J. Gordon (1969), 'Spline blended surface interpolation through curve networks', *J. Math. Mech.* **18**, 10, 931–952.

W.J. Gordon and R.F. Riesenfeld (1974), 'B-spline curves and surfaces', in *Computer Aided Geometric Design* (R.E. Barnhill and R.F. Riesenfeld, eds), Academic Press (New York), 95–126.

J.A. Gregory, ed. (1986), *The Mathematics of Surfaces*, Clarendon (Oxford), 217–232.

H.B. Griffiths (1976), *Surfaces*, Cambridge University Press (Cambridge).

R. Hall (1990), 'Algorithms for realistic image synthesis', *Computer Graphics Techniques*, (D.F. Rogers and R.A. Earnshaw, eds), Springer (Berlin), 189–231.

D.C. Handscomb, ed. (1989), *The Mathematics of Surfaces III*, Clarendon Press (Oxford).

C. Hoffman (1989), *Geometric and Solid Modelling*, Morgan Kaufmann (San Mateo, CA).

C. Hoffman and J. Hopcroft (1986), 'Quadratic blending surfaces', *Comput. Aided Des.* **18**, 301–306.

C. Hoffman and J. Hopcroft (1987), 'The potential method for blending surfaces and corners', in *Geometric Modelling: Algorithms and New Trends* (G. Farin, ed.), SIAM (Philadelphia), 347–366.

J. Hoschek (1988), 'Spline approximation of offset curve', *Comput. Aided Geom. Des.* **5**, 33–40.

J.K. Johnstone (1993), 'A new intersection algorithm for cyclides and swept surfaces using circle decomposition', *Comput. Aided Geom. Des.* **10**, 1–24

S. Katz and T.W. Sederberg (1988), 'Genus of the intersection curve of two rational surface patches', *Comput. Aided Geom. Des.* **5**, 253–258.

A. Kaufmann (1991), 'A distributed algorithm for surface/plane intersection', in *Curves and Surfaces* (J. Laurent, A. le Mehaute and L.L. Schumaker, eds), Academic Press (New York) 251–254.

R. Klass (1980), 'Correction of local surface irregularities using reflection lines', *Comput. Aided Des.* **12**, 73–78.

P.A. Koparkar and S.P. Mudur (1985), 'Subdivision techniques for processing geometric objects', *Fundamental Algorithms for Computer Graphics* (R.A. Earnshaw, ed.), NATO ASI F17, Springer (Berlin) 751–801.

P.A. Koparkar and S.P. Mudur (1986), 'Generation of continuous smooth curves resulting from operations on parametric surface patches', *Comput. Aided Des.* **18**, 193–206.

G.A. Kriezis, P.V. Prakash and N.M. Patrikalakis (1990), 'A method for intersecting algebraic surfaces with rational polynomial patches', *Comput. Aided Des.* **22**, 645–654.

G.A. Kriezis, N.M. Patrikalakis and F-E. Wolter (1992), 'Topological and differential-equation methods for surface intersections', *Comput. Aided Des.* **24**, 41–55.

D. Lasser (1986), 'Intersection of parametric surfaces in the Bernstein–Bezier representation', *Comput. Aided Des.* **18**, 186–192.

P-J. Laurent, A. le Mehaute and L.L. Schumaker (1991), *Curves and Surfaces*, Academic Press (New York).

R.A. Liming (1979) *Mathematics for Computer Graphics*, Aero.

T. Lyche and L. L. Schumaker (1989), *Mathematical Methods in Computer Aided Geometric Design* Academic Press (New York).

R.P. Markot and R.L. Magedson (1989), 'Solutions of tangential surface and curve intersections', *Comput. Aided Des.* **21**, 421–429.

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

R.R. Martin, ed. (1987), *The Mathematics of Surfaces II*, Clarendon (Oxford).

W.H. McCrea (1960), *Analytic Geometry of Three Dimensions*, University Mathematical Texts, Oliver and Boyd (Edinburgh).

A. Middleditch and K. Sears (1985), 'Blend surfaces for set-theoretic volume modelling systems', *SIGRAPH Comput. Graphics* **19**, 161–170.

Y. de Montaudouin (1989), 'Cross product of cones of revolution', *Comput. Aided Des.* **21**, 404.

Y. de Montaudouin (1991), 'Resolution of $P(x, y) = 0$', *Comput. Aided Des.* **23**, 653–654.

G. Mullenheim (1990), 'Convergence of a surface/surface intersection algorithm', *Comput. Aided Geom. Des.* **7**, 415–424.

G. Mullenheim (1991), 'On determining start points for a surface/surface intersection', *Comput. Aided Geom. Des.* **8**, 401–408.

A.H. Nasri (1987), 'A polyhedral subdivision method for free-form surfaces', *ACM ToG* **6**, 29–73.

A.H. Nasri (1991), 'Boundary-corner control in recursive subdivision surfaces', *Comput. Aided Des.* **23**, 405–410.

G.M. Nielson (1974), 'Some piecewise polynomial alternatives to splines under tension', in *Computer Aided Geometric Design* (R.E. Barnhill and R.F. Riesenfeld, eds), Academic Press (New York) 209–235.

J. Nocedal (1992), 'Theory of algorithms for unconstrained optimization', *Acta Numerica 1992*, Cambridge University Press (Cambridge), 199–242

K. Ohkura and Y. Kakazu (1992), 'Generalization of the potential method for blending three surfaces', *Comput. Aided Des.* **24**, 599–610.

J.C. Owen and A.P. Rockwood (1987), 'Intersection of general implicit surfaces', in *Geometric Modelling: Algorithms and New Trends* (G. Farin, ed.), SIAM (Philadelphia), 335–346.

Q.S. Peng (1984), 'An algorithm for finding the intersection lines between two B-spline surfaces', *Comput. Aided Des.* **16**, 191–196.

J. Peters (1990), 'Smooth mesh interpolation with cubic patches', *Comput. Aided Des.* **22**, 109–120.

C.S. Petersen (1984), 'Adaptive contouring of three-dimensional surfaces', *Comput. Aided Geom. Des.* **1**, 61–74.

B. Pham (1992), 'Offset curves and surfaces: a brief survey', *Comput. Aided Des.* **24**, 223–229.

L. Piegl (1993), *Fundamental Developments of Computer Aided Geometric Modelling*, Academic Press (New York).

E. Polak (1971), *Computational Methods in Optimization*, Academic Press (New York).

M.J. Pratt and A.D. Geisow (1986), 'Surface/surface intersection problems', in *The Mathematics of Surfaces* (J.A. Gregory, ed.) Clarendon (Oxford), 117–142.

R.F. Riesenfeld (1975), 'On Chaikin's algorithm', *Comput. Graph. Image Proc.* **4**, 304–310.

R.G. Robertson (1966), *Descriptive Geometry*, Pitman (London).

A. Rockwood and J. Owen (1985), 'Blending surfaces in solid geometric modelling', in *Geometric Modelling: Algorithms and New Trends* (G. Farin, ed.), SIAM (Philadelphia), 367–384.

D.F. Rogers and R.A. Earnshaw, eds, (1990), *Computer Graphics Techniques*, Springer (Berlin).

P.K. Scherrer and B.M. Hilberry (1978), 'Determining distance to a surface represented in piecewise fashion with surface patches', *Comput. Aided Des.* **10**, 320–324.

I.J. Schoenberg (1946), 'Contributions to the problem of approximation of equidistant data by analytic functions', *Quart. Appl. Math.* **4**, 45–99.

T.W. Sederberg (1987), 'Algebraic geometry for surface and solid modelling', in *Geometric Modelling: Algorithms and New Trends* (G. Farin, ed.), SIAM (Philadelphia), 29–42.

T.W. Sederberg and R.J. Meyers (1988), 'Loop detection in surface patch intersections', *Comput. Aided Geom. Des.* **5**, 161–172.

T.W. Sederberg, H. Christiansen and S. Katz (1989), 'Improved test forclosed loops in surface intersections', *Comput. Aided Des.* **21**, 505–508.

T.W. Sederberg and X. Wang (1987), 'Rational hodographs', *Comput. Aided Geom. Des.* **4**, 333–336.

D.J.T. Storry and A.A. Ball (1989), 'Design of an n-sided patch from Hermite boundary data', *Comput. Aided Geom. Des.* **6**, 111–120.

I.E. Sutherland, R.F. Sproull and R.A. Schumaker (1974), 'A characterisation of ten hidden surface algorithms', *Comput. Surv.* **6**, 1–55.

G.W. Vickers (1977), 'Computer-aided manufacture of marine propellors', *Comput. Aided Des.* **9**, 267–274

J. Woodwark, ed., (1989), *Geometric Reasoning*, Clarendon (Oxford).

M.H. Wright (1992), 'Interior methods for constrained optimization', *Acta Numerica 1992*, Cambridge University Press (Cambridge), 341–407.

C-G. Yang (1987), 'On speeding up ray tracing of B-spline surfaces', *Comput. Aided Des.* **19**, 122–130.