3
An Abridged History of Curve, Network, and Surface Design

To better understand the problems encountered in free-form shape design, we review the approaches that have been developed for curve, network, and surface design.

3.1 Curves and Curve Design

Much of the work on curve design has been to develop methods that behave naturally in response to user specifications. A great deal of work has been devoted to the mathematical modeling of the *draftman's spline*. A draftman's spline is a thin, smoothly bending strip of wood held in place by weights called ducks (Fig. 3.1). The first recorded work related to physical splines dates to the 18th century, to a study of *elastica* by James and Daniel Bernoulli, and Leonhard Euler. Elastica are idealized thin beams. The Bernoulli brothers postulated that the work required to bend a thin beam was proportional to the square of the curvature. In a letter to Euler, Daniel Bernoulli suggested that the shape of the elastica was such that its squared curvature was minimized. Acting on this suggestion Euler derived the differential equation of the curve and classified its forms [124].

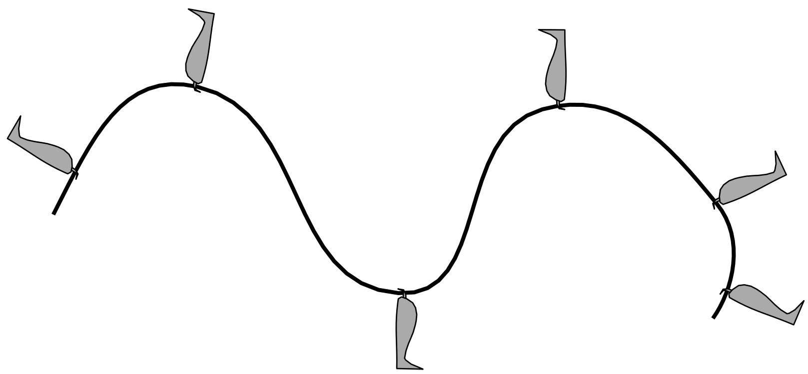

Figure 3.1. Draftman's Spline and Ducks.

In 1946, Schoenberg introduced the polynomial spline function, a piecewise polynomial function of degree n whose segments meet with C^{n-1} parametric continuity. In 1957, Holladay introduced the cubic spline for function interpolation and integration. The curve he developed minimized

$$\int \left[\frac{d^2}{dx^2} f(x) \right]^2 dx. \tag{3.1}$$

He noted that for curves with modest slopes, $f'(t) \ll 1$ the cubic spline provided a good approximation to the bending of a thin rod.

It is at this point that the development of curves and curve design techniques branches, with one vein of work building on the cubic spline and the other continuing to pursue improved approximations of *elastica* and other physical approaches. Since the latter branch is more closely related to MVC, we will first review that work, and return to the cubic spline and its descendants in section 3.1.2. The remainder of this section is broken down topically as follows:

These subsections represent a survey of work on curve design and we provide them to illuminate the prevalent problems in CAGD.

3.1.1 Nonlinear Splines: Elastica, MEC, Physically Based Curves, MVC

Minimum energy curves, minimum variation curves, and other members of this family require the solution of systems of nonlinear equations in order to compute their shape, thus their classification as *nonlinear splines.*

Even Mehlum [132] developed a technique for generating an approximating curve based on a spring model using the calculus of variations. In this model, the curve was suspended by springs attaching the curve to its control points (Fig. 3.2).

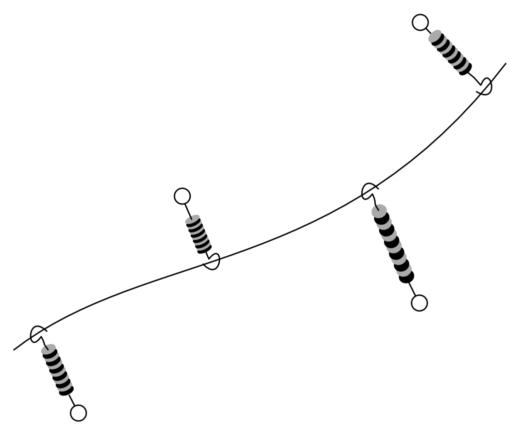

Figure 3.2. A Spring Curve.

An approximating curve is calculated by simulating the effect of springs attaching a rod to a series of anchors or control points.

Birkhoff et al. [14] presented a theoretical comparison of nonlinear mechanical splines, elastica, and Wilson-Fowler splines (Fig. 3.3), discussing possible methods for the computation of mechanical splines and elastica. The mechanical spline is a model of a thin beam passing through frictionless, swiveling supports; it is a special case of the elastica in which no tangential forces act on the beam or curve. Glass [78] presents the first algorithm to compute mechanical splines or minimum energy curves (MEC) as they have become widely known. His algorithm uses numerical techniques to compute discrete points on the curve. Mehlum [133] developed techniques for the computation of MEC, producing a piecewise circular arc approximation of arbitrary precision. Mehlum uses these methods in the Autokon system for curve and surface design [131]. Woodford [211] improved on Glass's algorithm for computing discrete MEC. Lee and Forsythe [118] discuss the variational and differential properties of open and closed MEC. Malcolm [125] further improves the computation of discrete MEC. Jerome [104] discusses the necessary and sufficient conditions for the existence of a minimum energy curve. Fischer and Jerome [65] discuss the stable and unstable equilibrium of MEC. Stable equilibrium of MEC refer to local minima of the energy functional, a configuration where the curve holds its shape with no constraint on its arc length. Golomb and Jerome [79] present theoretical results establishing conditions for equilibria, uniqueness, and regularity. Horn [101] thoroughly studies a specific MEC segment, defined by two points on a baseline with a vertical tangent constraint specified at each point. The resulting curve resembles a croquet hoop. Horn computes closed form expressions for the energy, arc length and maximum curvature of his subject curve.

Ohlin [149, 150] introduces the concept of consistency (2.16) and discusses consistent interpolation methods. He finds that, given requirements of second order continuity, invariance under similarity transformations, and consistency, there are very few acceptable interpolation methods. For function interpolation, Holladay's cubic spline is the only consistent solution. For curve interpolation, the available methods belong to a continuous family of curves that include MEC, a Cornu spiral[1], and the lemniscate. Independently he developed a fourth order continuous consistent spline that minimizes the arc length integral of the square of the arc length derivative of curvature, the MVC. Ohlin presents an algorithm for the computation of an MVC segment specified by position, tangent, and curvature at the end points. His algorithm computes discrete positions, tangents, and curvatures on the MVC and fits them with a piecewise quintic polynomial curve. Ohlin also outlines the advantages of MVC over MEC:

1. higher order continuity.
2. it caters to our human preference for circles.
3. curvatures at the extreme points may be estimated by the splining method, as well as the directions. (sic)
4. curvature and/or directions can be prescribed at all points, end points or interval, without losing continuity of curvature.
5. demanding to keep curvature as constant as possible rather than as small as possible will reduce the likelihood of undesired changes of sign in curvature.
6. as we have seen before, a stable solution to MEC does not always exist.

[1] A Cornu spiral is a curve with linearly varying curvature. Also known as *Euler spiral, linarc, lince,* and *clothoid*.

Kallay [114] describes a method based on finite sums and discrete optimization to compute the MEC in $\Re^3$ of fixed arc length. Jou and Han [111, 112] discuss the calculation of discrete MEC subject to arc length and tangent constraints.

In his thesis, Tom Rando [178] discusses the application of *optimization* to Bézier curve and surface design. With respect to curve design, he minimizes the arc length of the evolute of a curve segment specified by position, tangent, and curvature at the end points. This amounts to minimizing the arc length derivative of the radius of curvature,

$$\int_0^l \left(\frac{d\frac{1}{\kappa(s)}}{ds} \right)^2 ds. \qquad \textbf{(3.2)}$$

He also mentions the possibility of minimizing the arc length derivative of curvature, the MVC functional.

Finally, Celniker and Gossard [31] describe an interactive design system based on modeling curves and surfaces using physical simulation. The system uses finite element methods to compute the shape of curves subject to position and tangent constraints. The resulting curves converge on the MEC as the number of elements used is increased.

The line of work traced out in this section reflects a search for an optimal/ideal interpolant. This search began in a largely theoretical realm, ending in the application of these concepts in an interactive environment. As a further testament to the potential of these methods, Forrest [71] remarks that Autokon [131] is generally successful in producing curves acceptable to naval architects, aerodynamicists, and car stylists, from relatively crude data. Our work continues and parallels this vein; due to the increased speed of workstations, better solutions to interpolation and design problems are now possible.

3.1.2 Interpolating Splines: The Cubic Spline and its Descendants

As mentioned in the introduction to section 3.1 on page 33, the cubic spline was introduced by Holladay as an approximation of the MEC. The work following his introduction concerned methods for achieving an improved approximation to the MEC and reducing or eliminating the unwanted wiggles or oscillations sometimes observed in cubic splines. First, Asker [5] introduces several approaches that seem to have been lost or overlooked by later researchers. First Asker introduces a method allowing the stiffness of the spline to be varied along its length. This scheme provides a method to overcome wiggles. Two versions of variable stiffness are presented. First, stiffness varying in a piecewise constant fashion results in a C^1 cubic spline. And second, stiffness varying in a piecewise linear fashion results in a C^2 quartic spline. These methods are equivalent to the weighted spline of Salkauskas [186]. The second approach involves the establishment of a local coordinate system between adjacent interpolation points (Fig. 3.3). This curve has since become known as the Wilson-Fowler spline [77] and is used heavily by the DoE. The approach is designed to locally reduce the magnitude of $y'(x)$ thereby improving the approximation to the MEC. The Wilson-Fowler spline requires the iterative solution of a nonlinear system of equations. In addition to improving the approximation to the MEC, the Wilson-Fowler spline allows for the interpolation of multi-valued data sets, whereas cubic spline function interpolation is limited to single-valued curves.

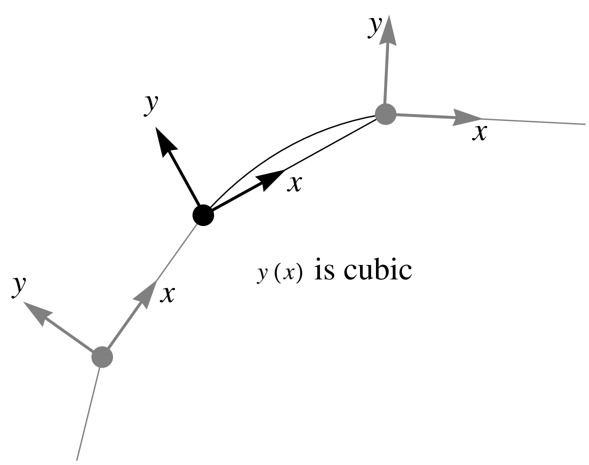

Figure 3.3. The Wilson-Fowler Spline.

Local coordinate systems are formed at each interpolation point of the Wilson-Fowler spline. The *x* axis of the coordinate system is aligned with the chord connecting adjacent points, the *y* axis is perpendicular to the chord.

Ferguson [64] introduces the parametric cubic spline curve by applying cubic spline function interpolation to vector valued data. A multi-valued curve defined by $x, y, [z], u$, where x, y, z specify the Euclidean space coordinates of the interpolation point and u is the independent curve parameter. The value of u specifies the location of the interpolation point in the parameter space of the spline. This is analogous to the role of the x-coordinate in function interpolation. Note that the sequence of u_i must be nondecreasing.

Schweikert [193] introduces the spline under tension as a means to deal with the spurious oscillations sometimes observed in cubic splines. The solution to the system he set up results in a curve described by exponential functions. Cline [37], and Pilcher [163] further develop splines under tension. In part due to the expense of computing these exponential curves, Nielson [140] introduces the ν-spline as a C^1 piecewise cubic polynomial alternative to splines in tension. In addition to providing a similar tension parameter, ν-splines extend splines in tension by allowing tension to vary along the curve affording greater control over curve shape.

Hagen [94] introduced the τ-spline as a generalization of the ν-spline; a piecewise quintic exhibiting G^2 continuity. Later, Foley [69] introduced the cubic weighted ν-spline, a generalization of C^2 cubic splines, weighted splines, and ν-splines. Extending piecewise polynomial splines in another direction, Meier and Nowacki [135] describe splines that minimize:

$$\int f'''(t)^2 dt \text{ (jerk or shear) and } \int f^{(4)}(t)^2 dt \text{ (load).} \qquad \textbf{(3.3)}$$

The result is a smoother, higher degree interpolant; note that "jerk" approximates the derivative of curvature, and minimizing it approximates an MVC. Similarly, Pottmann [167] extends τ-splines adding a third derivative term to the functional for minimization. Pottmann's spline results in a G^3 continuous interpolant, with G^4 continuity achievable with uniform tension.

Thus far, the works discussed have introduced shape handles and increased continuity. We now discuss some methods for the automatic setting of shape parameters and for the automatic parameterization of spline curves; parameterization is an additional means of shape control. First, consider the problem of parameterization. There are several methods in wide use for establishing the parameterization of a cubic spline. The two simplest are *uniform parameterization* and *chord length parameterization*. Using uniform parameterization, interpolation points fall at evenly spaced parameter values. Chord length parameterization stipulates that the points be spaced parametrically relative to the Euclidean distances between successive points (Fig. 3.4). Marin [128] describes a parameterization technique that minimizes an objective function. The resulting parameterizations reduce overshoot, lower C^3 discontinuities, and center oscillations in the second derivative around zero. The solution is calculated in closed form for the case of cubic function interpolation; unfortunately, the solution is only calculated as a numerical approximation in curve interpolation problems. Farin [59:130-134] outlines other methods for determining improved parameterizations.

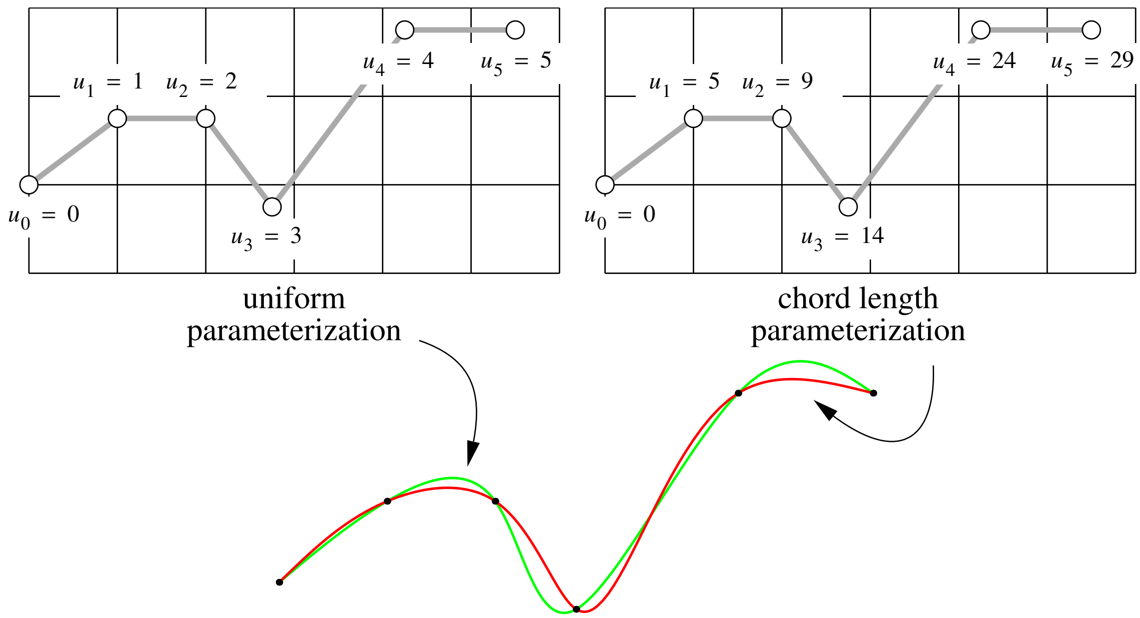

Figure 3.4. Uniform vs. Chord Length Parameterization.

A cubic spline is fit to data using uniform and chord length parameterizations. The chord length parameterized curve is smoother than the uniformly parameterized curve.

Methods for automatically setting shape handles are less well developed. The setting of shape handles is typically done interactively by the user/designer. Schweikert [193] discusses conditions necessary for the absence of unwanted inflections, and provides an explicit formula for setting the tension parameter of a uniformly parameterized cubic spline to satisfy these conditions. Fletcher and McAllister [67] describe an algorithm for automatically setting the tensions of a cubic spline function interpolant to remove unwanted oscillations. Other methods for automatic shape preservations are reviewed in section 3.1.5.

3.1.3 Local Interpolation Methods

Thus far we have described global methods for computing interpolating curves. In this section we review local methods for computing smooth interpolating curves. Using local methods, curves are typically less computationally expensive to calculate, and because local methods exhibit local control, the resulting curves can be easier to manipulate during the interactive design process. It is also possible to identify special configurations of control points which can then receive treatment tuned to the configuration. The primary disadvantage of local methods is that they possess curvature plots that are usually less smooth that those resulting from global and nonlinear splines. Furthermore local methods are provably inconsistent, with the exception of linear interpolation [198].

The earliest method of local interpolation, still in wide use, is due to Ackland [2]. Often referred to as *osculatory interpolation,* this method computes the slopes of a cubic function interpolant using a parabolic fit at each interpolation point, Fig. 3.5. A similar method due to Overhauser [151, 22] fits parametric quadratic curves to sets of three points. The overlapping curves are then linearly blended resulting in a C^1 cubic interpolant guaranteed free of extraneous wiggles. Like the cubic spline, Overhauser interpolation requires the user to specify the parameterization of the curve.

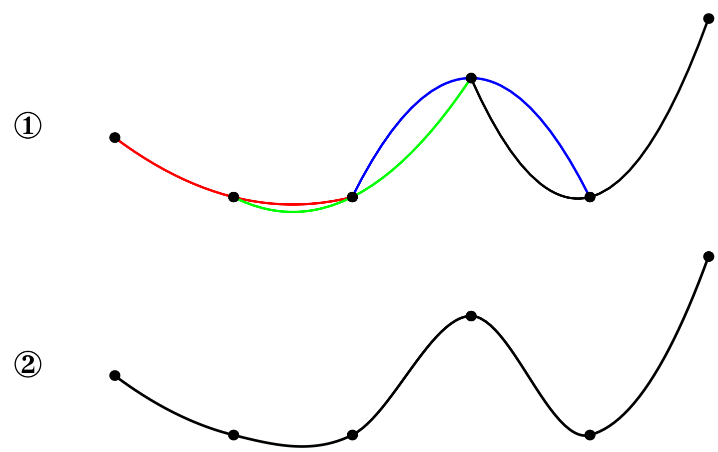

Figure 3.5. Osculatory Interpolation.

A local cubic function interpolant uses parabolas ① fitted to sets of three points to compute slopes for the cubic segments ②.

Akima [4] describes local procedures for the selection of tangent directions used in cubic function interpolation. This method computes the slope at an interpolation point solely from the slopes of the surrounding chords. The slope at point x_3, y_3 is

$$t_3 = \frac{|t_{4,5} - t_{3,4}| t_{2,3} + |t_{2,3} - t_{1,2}| t_{3,4}}{|t_{4,5} - t_{3,4}| + |t_{2,3} - t_{1,2}|}, \text{ (Fig. 3.6).}$$

This has the property that if adjacent chords are of equal slope then the slopes at the three incident points are set to the chord slope. This requires special handling of the case where $t_{1,2} = t_{2,3}, t_{3,4} = t_{4,5}, t_{1,2} \neq t_{3,4}$, $t_3 = 1/2\,(t_{2,3} + t_{3,4})$. The special case just

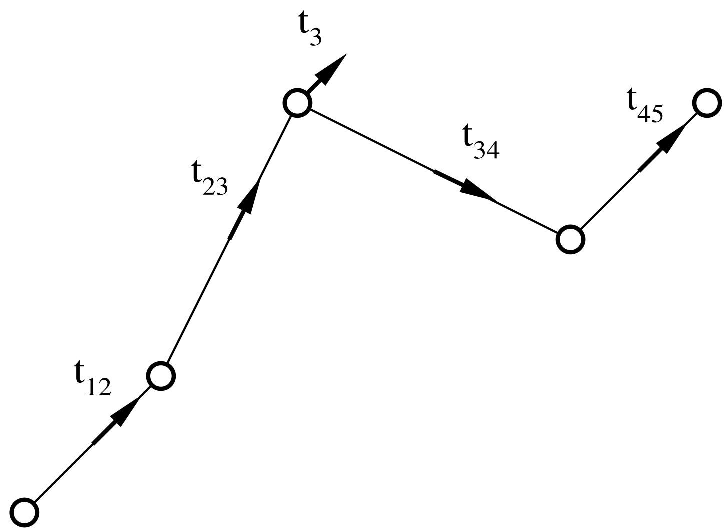

Figure 3.6. Akima's Local Method for Determining Function Slope.

mentioned renders the method sensitive to problems with numerical precision. The curve resulting from a fixed data set may vary in shape if it has a condition very close to this special case. For example, if the curve is calculated in single precision, the special case may be used, whereas if calculated in double precision, the general rule may be used. Another drawback to this particular special case handling is that it will cause the curve's shape to change discontinuously during interactive manipulation.

The Catmull-Rom spline represents a whole *class* of local interpolating splines. The most commonly used version is a C^1 cubic curve whose first derivatives at interpolation points are set equal to the scaled chord formed by joining the previous and next interpolation points. The scaling parameter is typically set to $1/2$, but may be varied and used as a shape handle, see Figure 3.7. This form of the Catmull-Rom spline is equivalent to Overhauser interpolation. The Catmull-Rom spline is generalized by DeRose and Barsky [48] who weaken the continuity from parametric to geometric, and then use the additional degrees of freedom to add more shape parameters.

Manning [127] describes an iterative algorithm for a cubic interpolatory curve with G^2 continuity. Manning was the first to work out the necessary and sufficient conditions for curvature continuity as a function of parametric derivatives. Because the curves are calculated independently of heuristic parameterization, they are free of oscillations due to unevenly spaced data. Oscillations often result from parameterization methods that do not adequately account for variations in the spacing of data.

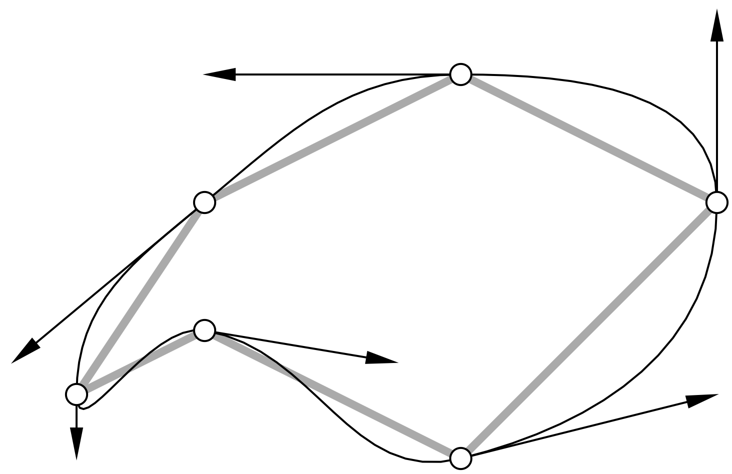

Figure 3.7. Catmull-Rom Spline Interpolation.
The cubic segments of a cardinal spline are defined by the positions of their end points and first derivatives equal to the direction and $1/2$ the magnitude of the chord formed by joining the neighboring interpolation points.

Ellis and McLain [55] offer an alternative to osculatory function interpolation. A cubic curve is fit to the data surrounding an interpolation point, and the slope of this cubic is used to specify the slope of the cubic interpolant at the interpolation point. The fit cubic is computed to pass through the immediately adjacent interpolation points and give a least squares fit to the next neighboring points on either side.

Ball [7] describes an alternative cubic curve segment specification in some respects similar to cubic Bézier curves (3.1.4). The Ball curve is specified by its endpoints, the tangent directions at those points, and a single interpolated point on the interior of the curve. Higher order generalizations of the Ball curve are described by Said [185].

Forrest [74] describes a generalization of the Bézier segment (3.1.4) from polynomial to rational curves. In particular he discusses the cubic segment and its flexible use as a space curve and for the specification of conic sections. Pavlidis [154] describes a curve fitting technique based on conic curve segments. Pratt [174] also developed an interpolant based on conic splines, using available degrees of freedom to approximately minimize the variation of curvature along the curve. Most recently, Pottmann [168] describes a locally controlled G^2 conic spline which interpolates position, tangent, and curvature. The interpolation points are joined by two conic segments per span.

Jordan and Schindler [110] present a locally controlled G^2 rational polynomial curve with a tension shape parameter. Nielson [141] presents a rational (cubic over quadratic) spline where segments are planar with zero curvature at the end points. This scheme also has a tension parameter and is trivially G^2.

De Boor, et al. [44] describe a planar cubic G^2 interpolant exhibiting convexity preservation and high accuracy. The curve is defined by position, tangent, and curvature at the end points of each segment. There are restrictions on the specification of curvature that allow a solution to the interpolation problem. They suggest that position and tangent be specified and that curvature be used as a shape parameter. Peters [157] develops the construction of a local quartic G^2 space curve that interpolates position, tangent, and curvature. If either tangent or curvature or both are not specified, methods based on degree reduction or derivative minimization are used to initialize them.

3.1.4 Bézier Curves and Composition Constructions

In an important development, de Casteljau [45] presents what became known as the Bézier curve [12]. The importance of this curve lies in the fact that it is widely used in the construction of local interpolants. His work describes techniques for the calculation of points on the curve which also provide control polygons for the curve segments on either side of the calculated point (Fig. 3.8). In recent work, Ramshaw [175, 176], using multi-affine maps, provides an elegant and illuminating description of Bézier and B-spline curves (3.1.7).

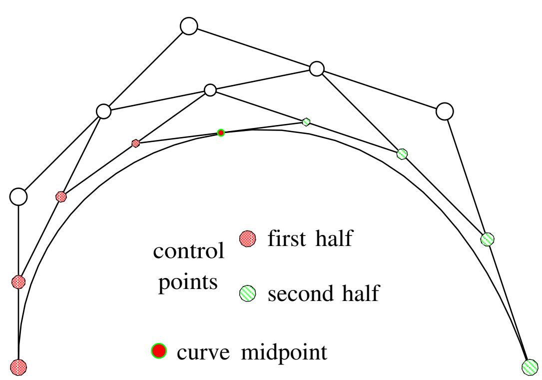

Figure 3.8. Quartic Bézier Subdivision.

Computation of points on a Bézier curve and subdivision of a Bézier curve is accomplished by repeated linear interpolation.

In order to compose local interpolants from Bézier segments, conditions for continuity are required. Farin [61] derives the necessary and sufficient conditions on the control points of cubic Bézier segments in order for adjacent segments to meet with G^2 continuity. Hagen [93] provides conditions for torsion continuity, noting that this does not imply G^3 continuity. Böhm [19] extends Hagen's conditions to rational Bézier curve segments. Most recently, Pottmann [166] provides constructions for composing G^3 continuous curves from Bézier segments.

Higashi et al. [98] describe constructions for fair Bézier curves specified by position, tangent, and curvature. The construction is designed to produce curves with smoothly varying curvature, MVC-like curves. Schaback [189] describes a construction for a unique G^2 interpolant composed of Bézier segments which preserves convexity. The input is restricted such that the sum of the angles at the ends of a chord is less than $90°$; they offer no solution in cases where this restriction is violated. Curves are made up of quadratic segments fitted to convex regions and cubic segments fitted to areas requiring a point of inflection. Straight lines are used in areas where data points are colinear.

In a new approach to interpolation, Séquin introduced the procedural spline [195]. Cubic Bézier segments are joined with G^1 continuity using rule based, geometrical methods to find first derivative directions and magnitudes. Like Akima's work this approach recognizes special cases such as three colinear points. This work is carried on by Shirman [197] who describes G^2 continuous curves, in some cases using multiple segments between interpolation points.

Rando [178] discusses the calculation of individual Bézier curve segments based on minimizing the variation of the radius of curvature,

$$\int \left(\frac{d\left(\frac{1}{\kappa}\right)}{ds} \right)^2 ds. \qquad \textbf{(3.4)}$$

This approach is similar to an MVC restricted to a pair of constraint points with a single intervening curve. An extension to the more general setting of multiple Bézier segments and multiple constraint sets is straightforward. However, as a functional, (3.4) has the drawback that it does not allow points of inflection, i.e., where $\kappa = 0$. This constraint renders the functional unusable in most CAD environments.

3.1.5 Shape Preserving Splines

Work on shape preservation begins with Schweikert's splines in tension [193]. Tension is set to eliminate extraneous wiggles in the interpolating curve. More recently, shape preservation has been formally defined as monotonicity and convexity preservation.

Fritsch and Carlson [76] present a method for calculating a monotone cubic function interpolant. They derive the necessary and sufficient conditions for monotonicity and construct an algorithm that produces smooth monotone curves from monotone input data. McLaughlin [130] describes a monotonicity preserving interpolation technique that employs straight lines and parabolas to produce a monotonic interpolant. Schumaker [192] presents an interactive algorithm that generates a C^1 quadratic spline that is convexity and monotonicity preserving. The algorithm preserves the extra degrees of freedom as shape handles available to the user. In a series of papers Fletcher and McAllister [68, 66, 67] discuss techniques by which bias and tension are used to automatically achieve convexity preservation. Gregory [89] presents an algorithm for automatic shape preservation. The algorithm uses a C^2 rational cubic representation and solves nonlinear shape consistency equations to determine the curve. Braibant et al. [21] applies finite element analysis to B-spline and Bézier curve design satisfying design constraints such as convexity. Ferguson [63] and later Jones [109] describes a system based on constrained optimization for achieving convexity of an interpolant. Goodman and Unsworth [80] use a piecewise cubic and straight line curve to form a curvature continuous interpolant that is locally convexity preserving; the solution of two nonlinear equations is required per segment.

Roulier [181] presents conditions for cubic Bézier segments of positive curvature. A segment exhibiting solely positive curvature is free of inflections and is thus convex. Later work by Roulier et al. [182] on curve fairness and monotonically varying curvature shows that MVC are naturally convex. Among curve segments specified by geometric constraints, position, tangent and curvature; if a curve exists with monotone curvature then the curve minimizing curvature variation will have monotone curvature.

3.1.6 Intrinsic Splines

Intrinsic, or alternatively *geometric splines* produce curves from geometrically-based, parametrically invariant curve specifications. Nutbourne, et al. [148] begin work in this area with curve design techniques based on curvature profile integration. Using this method, a designer specifies the curvature over the length of the curve, and the curve is calculated by integrating the curvature plot (Fig. 3.9). Adams [3] presents a similar method addressing the integration of torsion in addition to curvature. Bolton [23] describes a system based on biarc interpolation (Fig. 3.10). In this method, curve segments

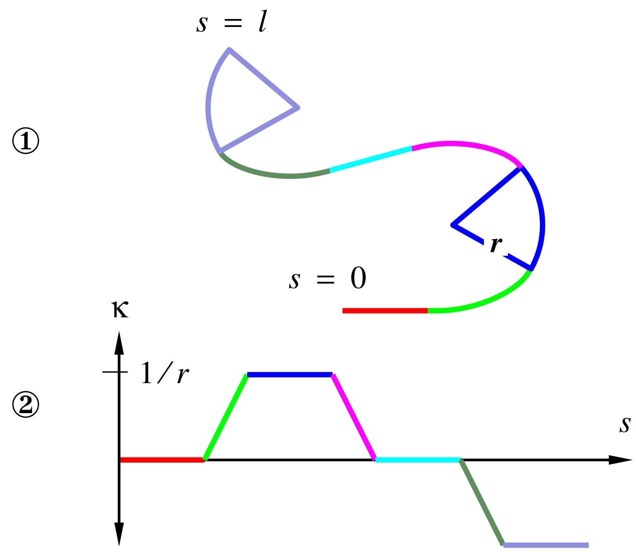

Figure 3.9. Curves from Curvature Integration.
The curve ① reflects the curvature profile ②; it starts as a straight line, curvature increases linearly to a constant curvature $1/r$, etc.

are specified by positions and tangents at their end points. The position and tangent values are interpolated by pairs of circular arcs meeting with tangent continuity. Pal and Nutbourne [152] discuss a method for computing a curve segment specified by end point position, and optionally, tangent and curvature. Their approach uses from one to four *linarc* segments to satisfy a given constraint set. Pal [153] extends the approach from two to three dimensions. Schechter [190] describes a more general system where multiple linarcs are used to compute curve segments specified by point, tangent, and curvature. A large family of alternate solutions is available from which a designer may select. Remember that the linarc was one member of the family of consistent curves studied by Ohlin (3.1.1).

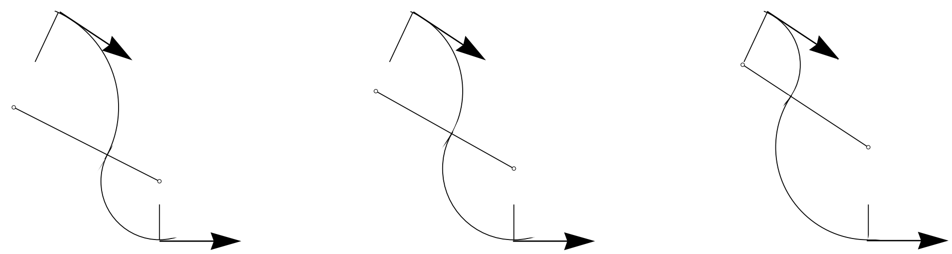

Figure 3.10. Biarc Interpolation.
Biarc interpolation yields a one parameter family of solutions; three solutions to a single problem are shown.

3.1.7 Local Approximating Splines: The B-spline and its Descendants

The B-spline was one of the first splines used in CAGD and several new concepts were introduced via the B-spline. It is because of its importance to the field in general that a discussion of work on B-splines is included here. The B-spline is an approximating spline which exhibits local control. The curve is defined by control points and follows the control polygon in a predictable and intuitive fashion (Fig. 3.11).

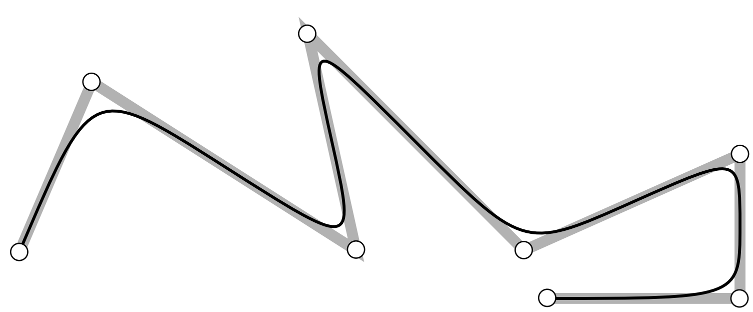

Figure 3.11. A Quadratic B-spline.

Early work on B-splines begins with Schoenberg [191], Cox [42] and de Boor [43] and was applied to approximation problems. Gordon and Reisenfeld [82] describe the first application of B-splines to problems of CAGD. In his thesis, Versprille [206] introduced the nonuniform[1] rational B-spline (NURB) to computer aided design applications. The main advantage of the rational form over the integral form of the B-spline is the ability to exactly represent conic curves. The rational form of the B-spline is described using the control points of the integral B-spline augmented with a weight at each control point. The weight acts as a shape handle with which the curve's shape can be further controlled. Increasing the weight associated with a vertex draws the curve closer to the associated control point. Similarly, decreasing the weight of a control point causes the curve to move away from that point. Tiller [203] has also discussed the use of NURBS for CAGD.

Barsky [9] introduced the β-spline as a *uniformly parameterized* geometric spline. A geometric spline satisfies geometric continuity rather than the parametric continuity of a traditional spline. Uniform parameterization stipulates that the knot values, u_i, be evenly spaced, $u_{i-1} - u_i = 1$. The β-spline incorporates two global shape handles, β_1, β_2, that describe the mapping between geometric and parametric continuity. First order parametric continuity is achieved by matching first derivatives at segment-segment knots u_i,

$$\vec{C}'_{i-1}(u_i) = \vec{C}'_i(u_i). \tag{3.5}$$

β_1 scales the first derivative of the composite curve on the minus side of the knot,

$$\beta_1 \vec{C}'_{i-1}(u_i) = \vec{C}'_i(u_i). \tag{3.6}$$

Second order parametric continuity is achieved by matching second derivatives,

$$\vec{C}''_{i-1}(u_i) = \vec{C}''_i(u_i). \tag{3.7}$$

β_2 acts as a weighting parameter, as it is increased the curve moves toward the associated control point. Algebraically, the second derivative on the plus side of the knot is calculated as

$$\beta_1^2 \vec{C}''_{i-1}(u_i) + \beta_2 \vec{C}'_{i-1}(u_i) = \vec{C}''_i(u_i). \tag{3.8}$$

As described here β_1, β_2 are global shape handles, Barsky and Beatty [8] describe the local control of the beta shape handles.

[1] nonuniform refers to curve parameterization, curve segment joints may be unevenly spaced parametrically.

Cohen [38] introduced LT-splines which were later shown by Joe [106] to be equivalent to discrete β-splines. In a series of papers [105, 106, 107], Joe presents nonuniform β-splines, β-spline subdivision or knot insertion, the equivalence of non-uniformity and β_1, and rational β-splines. Pham [162] describes the mapping between a conic β-spline with cubicly varying β_2 to a conic B-spline.

3.1.8 Variable Locality

We have described many curve generation methods that exhibit local control. Many of the methods discussed inherently allow for curve refinement through subdivision, or knot insertion. In subdividing a curve, control points are added, further restricting the extent of influence of control points. This allows the portion of the curve influenced by a control point to be reduced, providing a means for fine tuning curve shape. None of the methods described thus far provide a mechanism for increasing the portion of the curve influenced by a given control point. Given this, once detail is added to a curve it is quite difficult to make more global modifications to a curve's shape. Forsey [75] describes a curve design technique based on B-splines that has embedded mechanisms for controlling the locality of curve modifications. His curves, *hierarchical B-splines*, define a hierarchy of B-splines. At each level in the hierarchy, B-splines are defined relative to their parents in the hierarchy. Also, at each level the B-spline is subdivided increasing the number of control points and the locality of their effect. Because splines in the hierarchy are defined in a relative fashion, modifications made at one level of the hierarchy are automatically propagated to those splines lower in the hierarchy, whose definitions are relative to the modified segment. Forsey's approach is a significant improvement in the area of control of locality, however the hierarchy of curves is restrictive and it is not possible to achieve arbitrary locality.

3.1.9 Surveys

Brodlie and Böhm have independently compiled excellent surveys of techniques for curve design and representation.

By Brodlie:

A Review of Methods for Curve and Function Drawing [23].

Methods for Drawing Curves [24].

By Böhm:

On Cubics: A Survey [20].

A Survey of Curve and Surface Methods in CAGD [18].

3.2 Network Computation

Many surface modeling and data interpolation schemes use a mesh or network of curves as a key component in the construction of a smooth surface. In many cases the network computation is not tied to surface computation so that a variety of network computation methods may be paired up with a single surface modeling technique.

Chiyokura and Kimura [33] describe a system for the blending and filleting of polyhedral models. Polyhedron edges are replaced by portions of cylinders and cones, vertices are replaced by more complex surfaces. Rounding operations are specified by radii at the end points of polyhedron edges. The blending surfaces are computed by first creating a mesh of quadratic Bézier curves and then spanning them with Gregory patches. The curve mesh is automatically generated from the blending specification (Fig. 3.12). The end points of the curves making up the edges of the network are constrained to have tangents in the planes of the original polyhedron faces. These methods are successful in the limited application of filleting and blending the edges of polyhedral solids. The networks produced are only tangent plane continuous and result in surfaces lacking fairness. Farin describes a G^1 interpolant that operates on triangulated data consisting of position and normal information. A network of cubic curves is constructed under the restriction that the surface patches meet with tangent continuity [58].

Nielson [142] introduces the minimum norm network (MNN) for C^1 bivariate function interpolation. The MNN is a network of cubic curves that meet at interpolated vertices with tangent plane continuity while minimizing a linear energy term that approximates strain energy. The computation of the network requires the solution of a global system of linear equations. This method applies only to bivariate functions and is not suitable for modeling.

Piper [164] describes a local constructive technique for the computation of a G^1 network of cubic curves. This construction assumes that both surface position and surface tangent plane information are provided. The curves are specified by their endpoints (supplied explicitly) and first derivatives. The direction of the first derivative is established by projecting the chord, connecting the end points of the cubic, onto the plane specified at the vertex (Fig. 3.13). The magnitude of the first derivative is set equal to chord length plus one third of the distance that the end of the chord was projected. Similarly, Séquin [195] computes a G^1 cubic curve mesh from a polyhedral model. The surface normals at the vertices of the polyhedron are computed as a weighted average of the normals of the incident faces. The weights are proportional to the angle formed by the face's edges incident to the vertex. Once the normal is fixed, chord projection is used to establish curve tangent direction, and the chord length is used for the derivative magnitude.

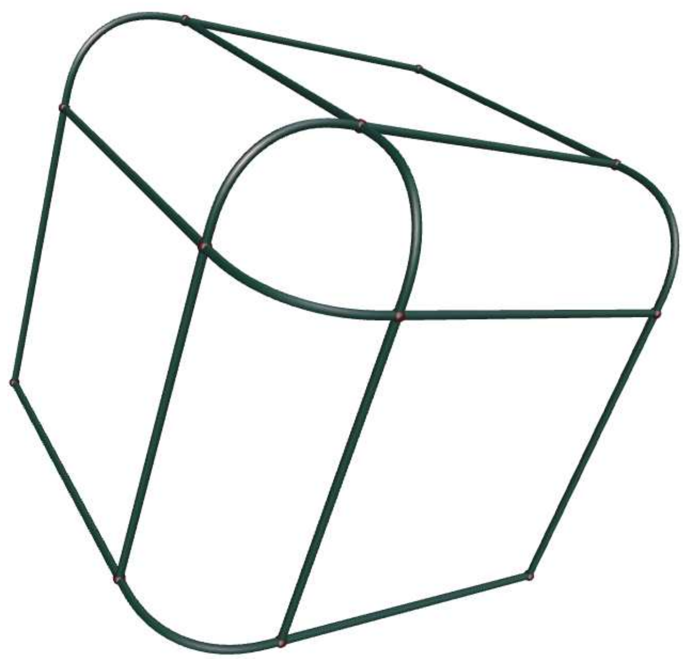

Figure 3.12. A Curve Network

One corner of a box is blended, the curve tangents are constrained to the planes defined by the faces at their endpoints.

Shirman and Séquin [200] present a construction for a local G^1 cubic curve network used for polyhedral smoothing. Surface normals at vertices are computed as a weighted average of face normals. The weighting used is either proportional to the area of the incident face or to the angle formed by the face's edges that are incident to the vertex. The first derivatives at the end points of the curves of the network are set using chord projection and chord length. They also introduce the "opposite edge method" where opposing curves are constructed to meet with tangent vector continuity. A pleasing spline [198] is fit through the subject vertex and two of its opposing neighbors. At even order vertices this is straightforward, and the normal is defined as an average of the normals implied by the curves intersecting at the vertex (Fig. 3.14①). Odd order vertices require the introduction of a "phantom" vertex at the midpoint of the segment connecting the two "opposite" vertices (Fig. 3.14②). In his thesis, Shirman [198] assesses the relative merits of several different methods for the construction of G^1 cubic networks. He concludes that using the "opposite edge method" for normal and tangent direction calculation and that using chord length to set first derivative magnitude are superior among the methods tested.

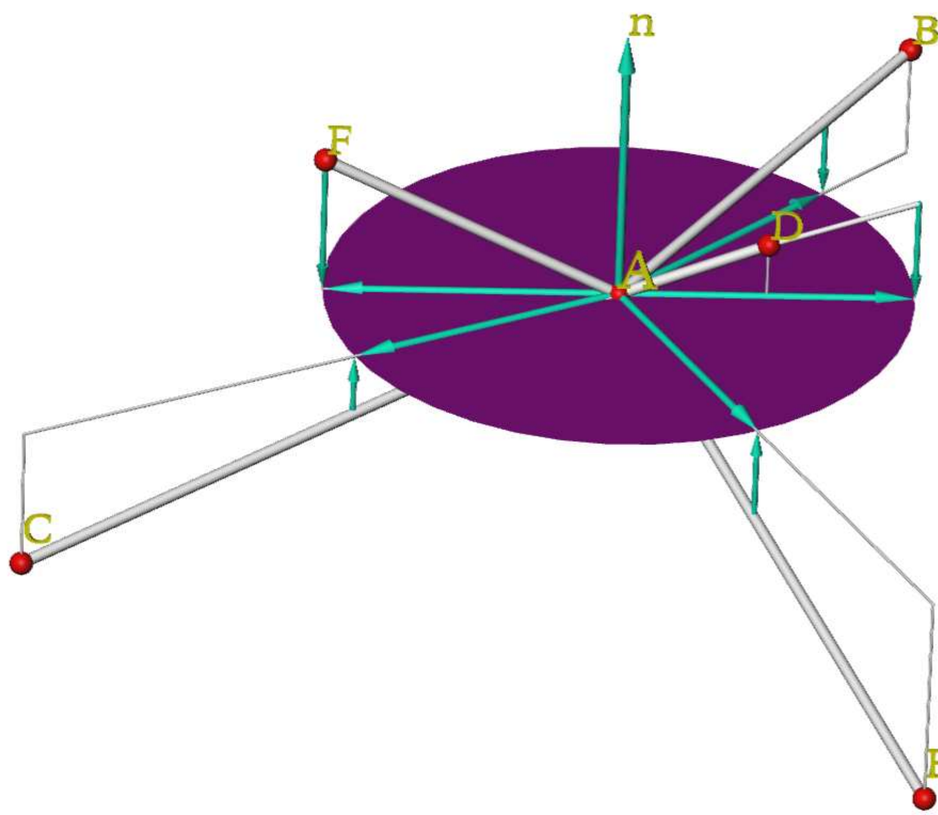

Figure 3.13. Piper's First Derivative Computation.

First derivative directions are indicated by arrows in the surface tangent plane. First derivative magnitudes are set equal to the chord length plus one third the distance from the neighboring point to the tangent plane.

Mann, et al. [126] present a survey of triangular interpolants. They found most interpolants to be unsatisfactory, and in particular found that the surfaces could be improved significantly by improving the quality of the curve networks. They adapt a method by de Boor, et al. [44] to create an improved mesh. It requires that surface position, normal, and curvature be provided by the designer or a sampled function. Additionally, the curves of the network must remain planar. Because of this restriction, the tangent direction is chosen by computing a plane in which the curve must lie. This is accomplished by taking the cross product of the chord and the average of the normals specified at the chord's end points. The tangent direction is along the line of intersection of the surface tangent plane and the plane containing the curve. While they find this approach to result in improved curve networks with G^2 continuity, they discuss a number of

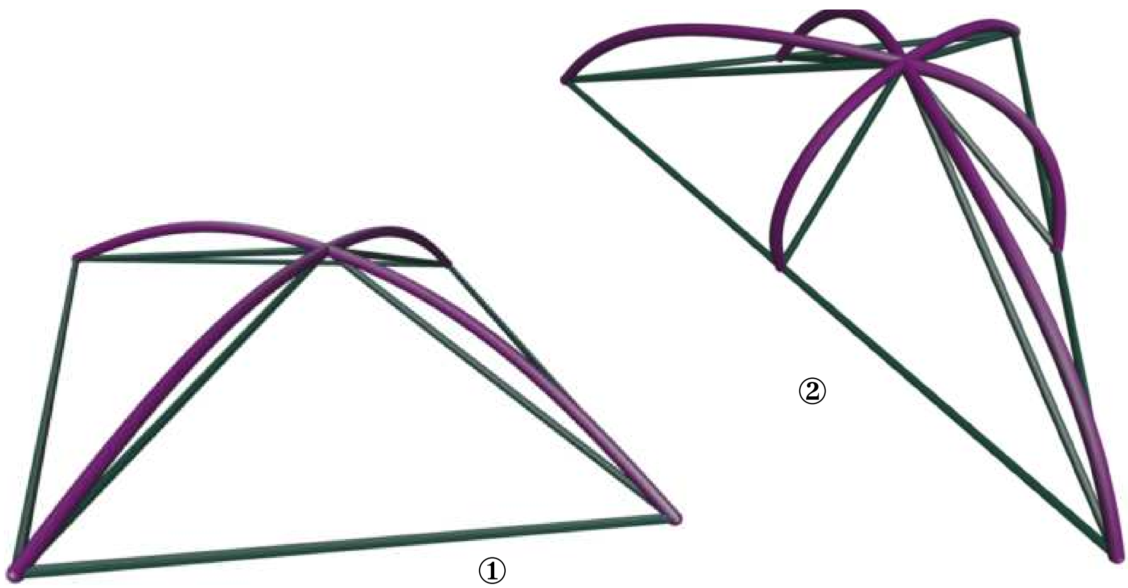

Figure 3.14. Opposite Edge Method.

At even order vertices ① the "opposite edge method" fits a pleasing spline through pairs of opposite vertices and the subject vertex, at odd order vertices ②, the mid points of edges serve as curve end points. Note that curves of network ② run from the vertices (corners) to the central point, and that the curves from the central point to the sides are only used in the construction.

associated problems. First, for each curve of the network, zero to three solutions exist. If no solutions exist they must fall back on some other method for curve definition. In the case where multiple solutions exist, while their shapes are nearly indistinguishable, their parameterizations vary greatly. The results of using different parameterizations dramatically affects the quality of the resulting surface. Mann et al. have outlined the difficulties with the methods described by Piper and Shirman. Both of these lack G^2 continuity and fairness. Though possessing G^2 continuity, the method resulting from adapting de Boor et al. lacks fairness.

Nielson [143] proposes an interactive design system using a minimum norm network (MNN) generalized to $\Re 3$. In the proposed system, users specify position, surface normal, and a tension shape handle. This modeling system holds some promise. However, because it lacks G^2 continuity and uses a linear approximation to strain energy, it will probably not exhibit a high degree of fairness. Pottmann [169] presents a generalization of MNN that results in a C^2 function interpolant. This method applies only to bivariate functions and is not suitable for modeling.

Moreton and Séquin [136] describe an early version of the MVN. G^2 minimum energy networks are described that minimize the MEC functional (4.1), the MVC functional (4.2), or a weighted combination of the two. The surface continuity construction used in this early work lacks the component of curvature in the binormal direction (section 5.1) and is generally less compact and less efficient. In section 5.1 we describe the representation and continuity constructions used in the computation of MVN.

3.3 Surfaces and Surface Design

MVS are designed to solve many of the problems encountered in the computer aided design of curved surfaces. In this section we review work on curved surface design and representation. To start, Ferguson [64] describes one of the earliest techniques for the description of general free-form surfaces. A mesh of cubic splines is computed and blended using tensor product patches with zero length twist vectors. The resulting surface interpolates a rectangular mesh of data points in $\Re^3$. The remainder of this section is topically broken down as follows:

3.3.1 Patches

3.3.1.1 Coons Patches

Coons [40] describes a surface representation based on the blending of boundary curves; Forrest [72] provides a good overview of Coons' work. A linear Coons patch is defined by adding the ruled surfaces formed by opposing curve boundaries and subtracting the bilinear surface formed by the four corners of the patch (Fig. 3.15). Using linear Coons patches to blend a network of curves results in a G^0 surface. To overcome the lack of tangent continuity Coons describes a patch capable of interpolating cross boundary tangent information in addition to mesh curves. The Coons biquintic patch is an extension of a tensor product patch using Hermite blending functions. Gordon [83] generalizes Coons patches to interpolate an arbitrary continuous functions along mesh lines. Gordon's work leads to a body of work on transfinite interpolants (3.3.1.4). Gregory [90] describes a different generalization of Coons patches that allow for the independent specification of patch twist vectors (3.3.2.2). This specification removes the constraints on the twist vector at the expense of a parametric singularity, an increase in degree, and the introduction of rational blending functions. This work is important since it introduces a mechanism by which networks of patches can be constructed locally (3.3.1.5). Gregory and Charrot [87] describe a triangular version of the twist compatible patch.

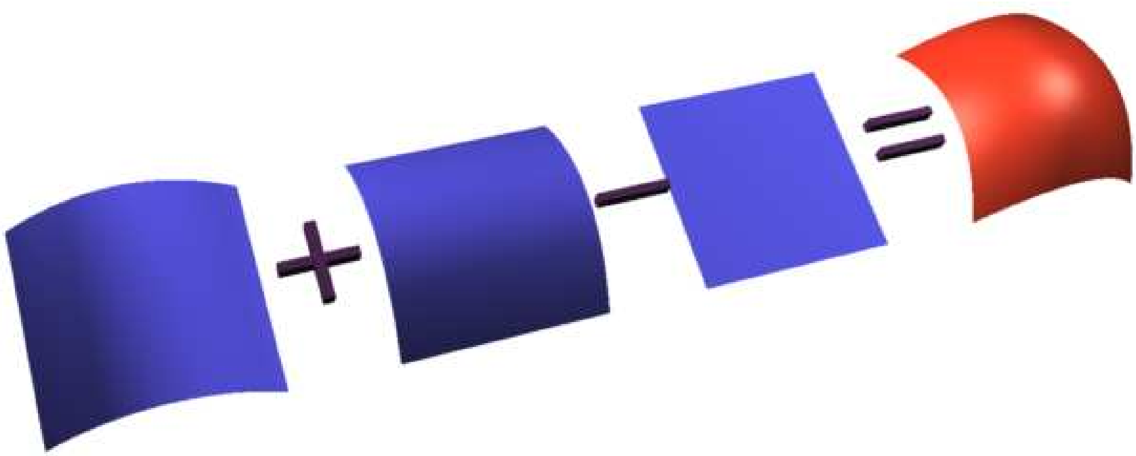

Figure 3.15. Linear Coons Patch Construction

3.3.1.2 Bézier Patches

Bézier [12] describes the mathematical underpinnings of UNISURF, the modeler used at Renault. In addition to describing the construction and calculation of Bézier patches (Fig. 3.16) he touches on several issues which became major research topics in later years: the vertex enclosure problem (3.3.2.2), patch-patch continuity (3.3.2.1), and free-form deformations. Earlier, in unpublished work, de Casteljau [45] develops what has become known as the triangular Bézier patch (3.3.1.6)(Fig. 3.17). Included in his work are stable evaluation algorithms for Bézier curves and Bézier triangles.

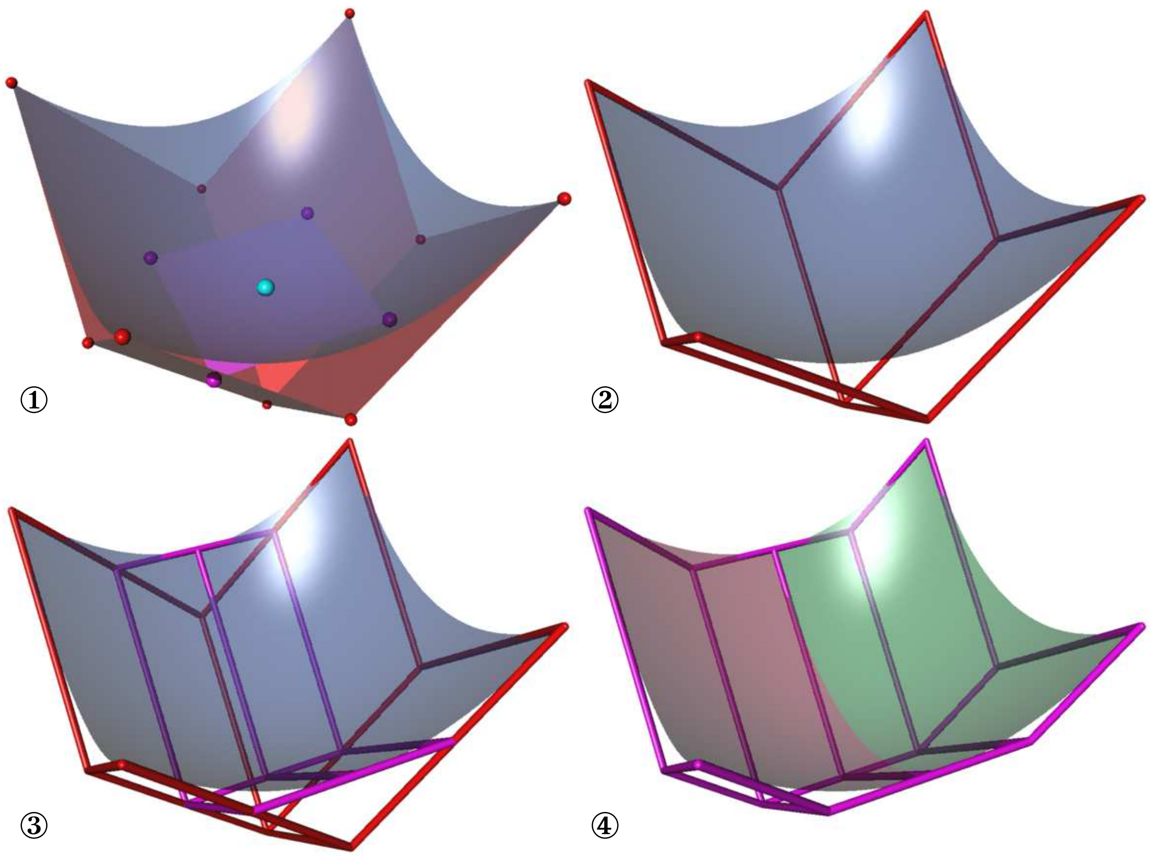

Figure 3.16. Quadratic Tensor Product Bézier Patch Construction
① Points on a tensor product patch are computed using repeated bilinear interpolation. ②, ③, and ④ A patch is subdivided by repeated linear interpolation.

3.3.1.3 B-spline Surfaces

The use of B-spline surfaces in CAGD was introduced by Gordon and Riesenfeld [82]. Versprille [206], and later Tiller [203], introduce the use of rational B-splines to CAGD. The rational B-spline is capable of exactly representing quadric surfaces such as spheres and cylinders. More recent work related to B-spline surfaces has been in the areas of *n*-sided patches (3.3.1.6) and arbitrary control meshes (3.3.1.7).

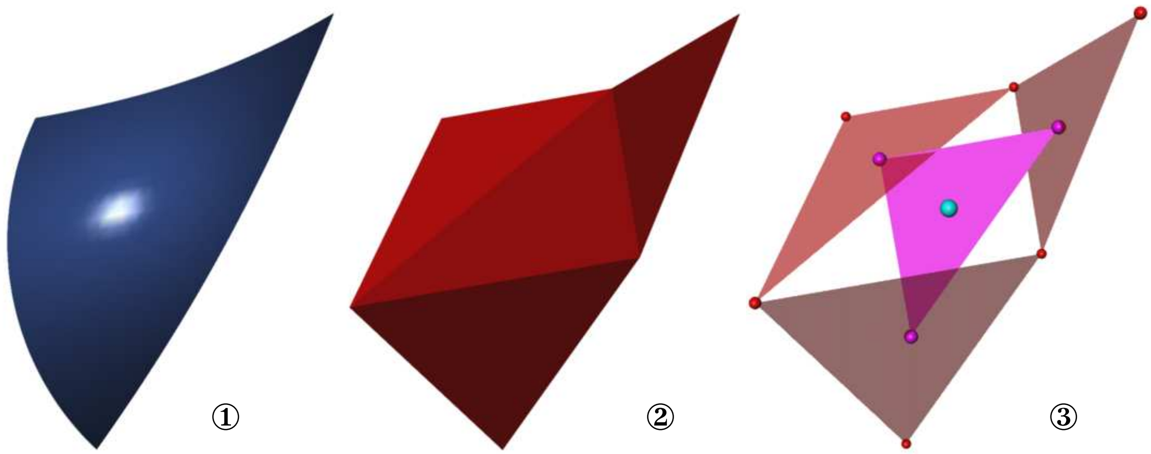

Figure 3.17. Quadratic Triangular Bézier Patch Construction.
① a Bézier triangle; ② triangular control hull; ③ finding the center point of the patch through repeated linear interpolation.

3.3.1.4 Transfinite Interpolants

Introduced by Gordon [83], transfinite surfaces are important because they represent one solution to the smooth assembly of collections of patches. Since transfinite interpolants interpolate arbitrary boundary data, it is trivial to smoothly piece together surface patches. Coons [40] describes the application of transfinite interpolation to the blending of a rectangular mesh B-spline curves. Gregory and Charrot [87] describe the first published triangular interpolant. This interpolant matches position and tangent data at triangle boundaries. Nielson [144] describes a transfinite triangular patch that interpolates both patch boundary curves and cross boundary tangent functions. A discrete version of the patch is presented where vertex normals are linearly interpolated along the boundaries. Using the discrete version of the patch, it is trivial to construct a G^1 interpolant over an arbitrary triangular mesh. Hahn [95] uses transfinite techniques to fill an *n*-sided hole with *n* rectangular patches. Hagen and Pottmann [92] describe a G^2 transfinite triangular patch which, like Nielson's triangle, can be discretized and used to create a G^2 interpolant over an arbitrary triangular mesh.

3.3.1.5 Gregory Patches

Developed by Chiyokura and Kimura [33], Gregory patches were inspired by Gregory's [90] modified Coons patches. They use similar rational blending functions to remove the twist constraints from tensor product Bézier patches (3.3.2.2). The Gregory patch has two twist vertices at each patch corner. A point $\vec{S}(u, v)$ on the surface is computed by linearly blending the twist vertex pairs to form a Bézier patch which varies as a function of u and v (Fig. 3.18). These vertex pairs are shown connected by a transparent rod with the twist vertex position for $u = v = 0.5$ shown in the middle. Chiyokura [34] further discusses the application of Gregory patches to irregular mesh interpolation. Gregory patches form an integral part of the commercial CAD system DesignBase [36]. Shirman and Séquin [196] describe an extension of Gregory patches providing shape parameters, *shear, bulge,* and *tilt* while maintaining G^1 continuity. Chiyokura, et al. [35] extend Gregory patches to patches with rational boundary curves.

3.3.1.6 Triangular and *n*-Sided Patches

Non-degenerate triangular patches suitable for computer aided design were introduced by Gregory and Charrot [87]. Their interpolant is formed from the blend of three separate triangles satisfying the constraints associated with a single side of the triangle. Farin [58] describes a triangular interpolant for data in $\Re^3$. First, boundary curves are computed, then a patch is fitted into each opening, and finally each patch is subdivided in Clough-Tocher fashion (Fig. 3.22), and the boundaries are adjusted to G^1 continuity. Sabin [184] describes three and five sided patches that are easily included into surfaces composed of biquadratic tensor product patches. Similar to their triangular patch, Charrot and Gregory [32] describe a pentagonal patch resulting from the blending of five rectangular patches. The patches are placed at each corner of the opening and blended to form G^1 continuous boundaries. Hosaka and Kimura [102] describe three, five, and six sided patches compatible with Bézier patches. The methods match quadratic and cubic adjacent patches. Piper [164] presents a construction for a pair of Bézier triangles to meet with G^1 and interpolate position and tangent specifications at the endpoints of the shared edge. Using this construction, he describes a scattered data interpolant using a Clough-Tocher splitting scheme. Hahn [95] describes a method for filling an n-sided hole with rectangular patches. This approach is a generalization of Charrot and Gregory [32] and uses a transfinite interpolant to solve the problem for arbitrary *n.* Jones [108] describes a method for constructing a collection of rectangular patches meeting with tangent or curvature continuity in order to fill an *n*-sided hole, for odd *n.* For tangent and curvature continuity, the resulting patches are respectively biquintic, and biseptic. Gregory and Hahn [86] describe a polygonal patch for filling an *n*-sided hole in a rectangular patch complex. The resulting patch is rational polynomial, G^2 continuous with surrounding patches, and

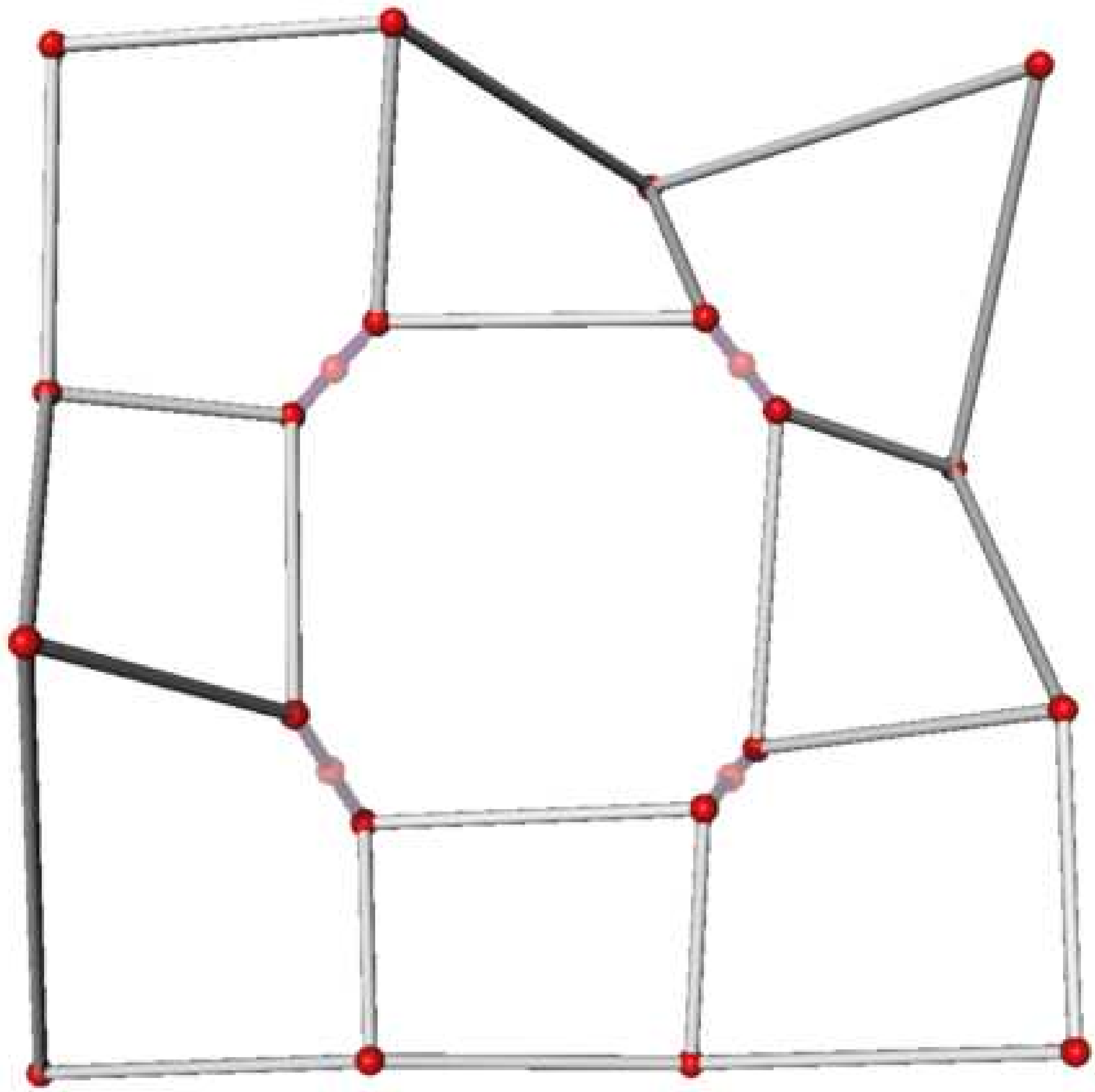

Figure 3.18. A Gregory Patch

A bicubic Gregory patch control hull; the dual twist vertices are shown connected by a transparent rod. The vertex in the middle of this rod is positioned for surface parameter values $u = v = 0.5$.

affords several shape parameters. Loop and DeRose [121] present the S-patch as an n-sided generalization of triangular and quadrilateral Bézier patches. A C^k algorithm for joining S-patches with Bézier triangles is also presented.

3.3.1.7 Subdivision Surfaces and Splines on Arbitrary Networks

Catmull and Clark [29] introduced the subdivision surface. Subdivision surfaces are formed by taking a network of points and recursively subdividing the network to produce a B-spline-like surface as the number of subdivisions increases (Fig. 3.19). In the case where the network is restricted to a rectangular mesh, the resulting surface is exactly the uniform B-spline surface resulting from treating the network as the control points of the surface. In fact, as the number of subdivisions increases, major portions of surfaces on arbitrary nets form tensor product B-spline surfaces, it is only in areas around original vertices of order other than four that irregular behavior occurs. Doo and Sabin [53] analyze the behavior of subdivision surfaces around *extraordinary* points (those vertices of the network that have order other than four.) Their analysis shows that the limiting surface is G^1 and indicates methods that might further improve the behavior of surfaces near singular points. Beeker [11] describes a system for smoothing polyhedral networks with four-sided openings and vertices with 3,4, and 5 incident edges. The resulting surface is composed of bicubic patches joining with G^1 continuity. Van Wijk [208] studies the fitting of tensor product β-spline patches to networks with four-sided faces, and vertices of order 4 or odd order vertices. Nasri [138] extends these works to surfaces that interpolate the network control points. This is accomplished by deriving a new network that places the original control points in the centers of the new network openings. Computing the new network is very similar to computing the dual of a planar graph. Tan and Chen [201] use the ideas of Catmull and Clark to construct a network of biquadratic B-spline patches.

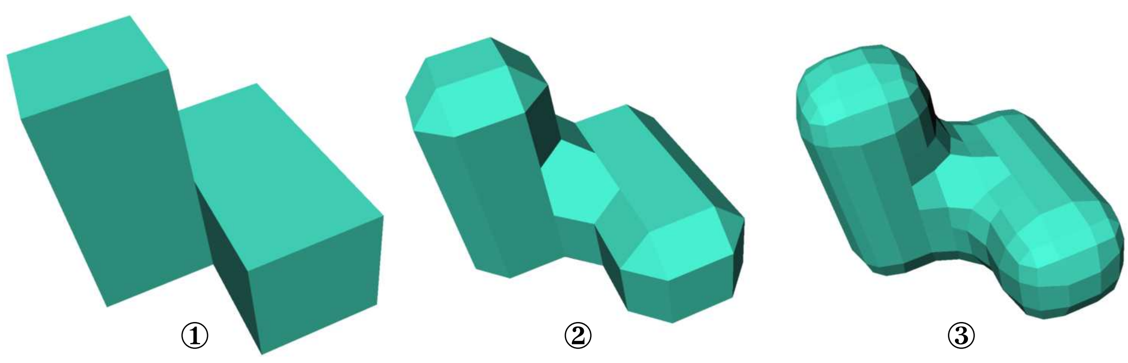

Figure 3.19. Two Generations of a Subdivision Surface.
① The original polyhedral model. ② The model after the edges and vertices have been removed. ③ The model after the edges and vertices of ② have been removed.

Starting with the same network specification, they only subdivide those regions containing or adjacent to extraordinary regions. This approach is taken at each level of the recursive subdivision, isolating the singular regions with geometric convergence (Fig. 3.20). Ball and Storry [6] further study the behavior of subdivision surfaces near extraordinary points. Their results, based on an eigen analysis of the subdivision matrices, show what

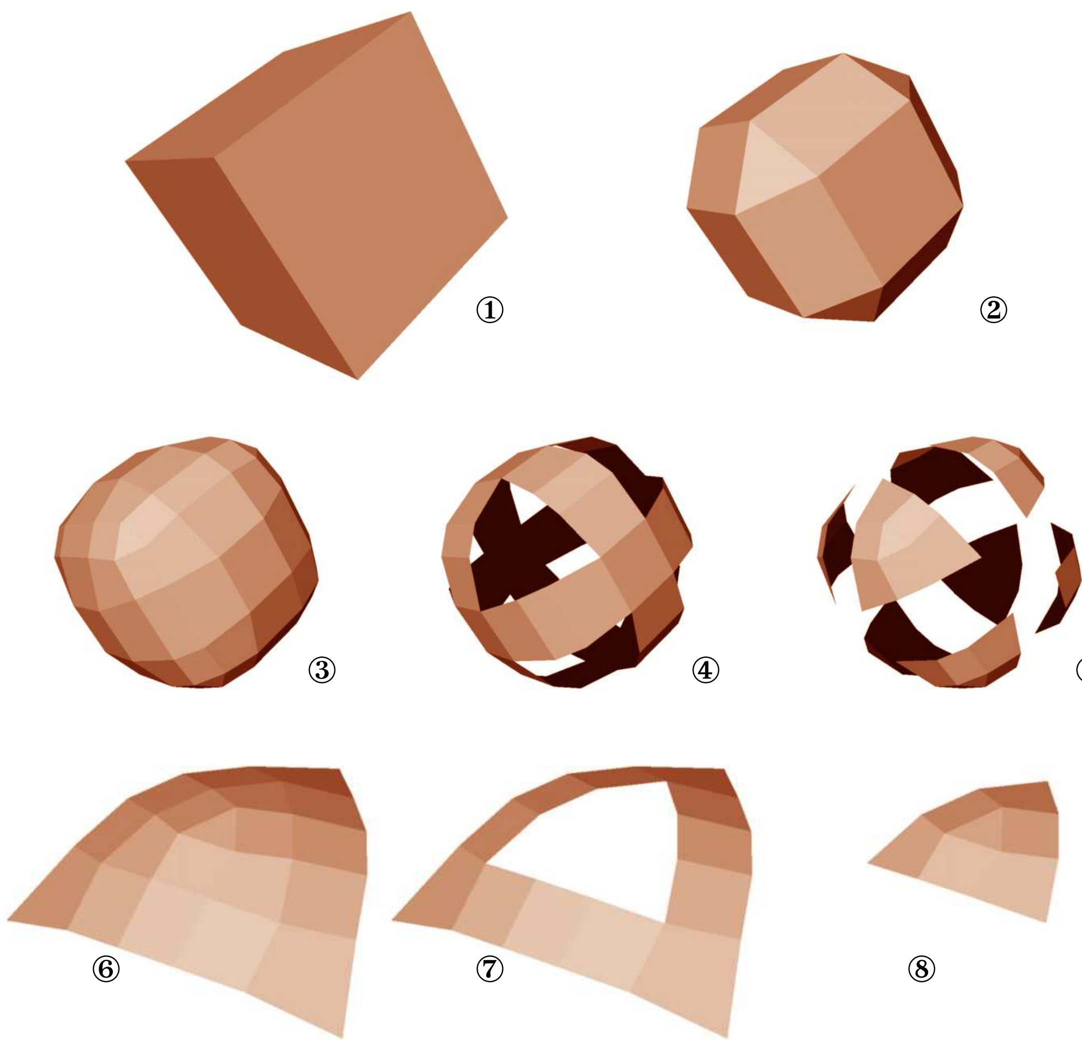

Figure 3.20. Four Steps in the Isolation of Singular Points

① The original polyhedron. ②, ③ After one and two levels of subdivision. ④ The regions representable by tensor product B-splines after two subdivisions. ⑤ The singular regions after two levels of subdivision. ⑥ Another level of subdivision applied to one of the singular regions. ⑦ The B-spline regions after the third level of subdivision. ⑧ The remaining singular region after three levels of subdivision.

conditions are necessary for G^1 continuity. Brunet [26] describes an extension of Doo and Sabin's method combined with Nasri's approach that includes shape handles and automatic methods designed to reduce jumps in curvature between patch boundaries. Cavaretta and Micchelli [30] provide a thorough analysis of subdivision curve and surface algorithms. Dyn, et al. [54] present an approach quite similar to Brunet's, no comparison is made with earlier work. Höllig and Mögerle [99] introduce a new spline space, G-splines. This new spline is derived from the work of Goodman [81] and defines splines on networks composed of four-sided faces. Goodman's work is similar to that of Van Wijk's β-spline surfaces on restricted networks. Loop and DeRose [122] describe a generalization of B-spline surfaces that can be applied to models of arbitrary topology. S-patches [121] are used as the building blocks of the surface. The network of patches is restricted to one of two types: a network of four sided patches, or a network with exactly four faces incident to each vertex. Lee and Majid [119] present methods for the generation of Bézier and B-spline surfaces from polyhedral control networks. This is similar to Goodman's approach where shape handles are incorporated into the construction. Finally, Nasri [139] has generalized his recursive subdivision algorithm to support the interpolation of both vertex positions and vertex normals.

3.3.1.8 Principal Patches and Cyclides

Martin [129] describes the use of principal patches for geometric modeling. A principal patch is a patch whose edges fall along lines of principal curvature. Since lines of principal curvature have zero geodesic torsion, it is simple to form G^1 continuous joins between patches. Martin uses portions of surfaces of revolution and portions of Dupin's cyclides as principal patches. In similar papers Böhm [17] and Pratt [173] explore the use of cyclides in geometric modeling. They derive the Bézier form of a cyclide and provide several blending examples. Cyclides are particularly interesting because they are optimal MVS surfaces, i.e the MVS functional is zero for any cyclide.

3.3.2 Continuity

3.3.2.1 Continuity Conditions and Constructions

Bézier [12] discusses the necessary and sufficient conditions for tangent continuity between adjoining patches. He states that the partial derivative vectors of patches should be coplanar (Fig. 3.21)

$$\begin{vmatrix} \dfrac{\partial \vec{F}(u,v)}{\partial u} \\ \dfrac{\partial \vec{F}(u,v)}{\partial v} \equiv \dfrac{\partial \vec{G}(w,v)}{\partial v} \\ \dfrac{\partial \vec{G}(w,v)}{\partial w} \end{vmatrix} = 0.$$

Bézier enforced this condition by using the relation

$$\frac{\partial \vec{F}(u,v)}{\partial u} = \lambda(v)\frac{\partial \vec{F}(u,v)}{\partial v} + \mu(v)\frac{\partial \vec{G}(w,v)}{\partial w},$$

where λ and μ vary linearly.

Veron, et al. [205] characterize first and second order continuity between polynomial tensor product patches in terms of the geometric measures. They then proceed to characterize *n*th order continuity in terms of reparameterization. Farin [60] presents sufficient constructions for all three combinations of triangular and rectangular joins of polynomial surface patches. Kahmann [113] derives the conditions for second order continuity between tensor product Bézier patches. Briefly, if the patches meet with G^1 continuity, then G^2 continuity may be achieved by setting up the patches such that their asymptotic directions are colinear across the patch-patch boundary. This construction holds unless the patch-patch boundary follows an asymptotic direction of the surface. Kahmann also notes that construction of G^2 continuous patch networks is extremely difficult due to the second order analogue of the twist compatibility problem (3.3.2.2). DeRose [47] discusses two different, but equivalent, characterizations of geometric continuity. The first is based on reparameterization, and the second is based on the theory of differentiable manifolds. Höllig [100] discusses continuity conditions for triangular and quadrilateral patches. G^1 and G^2 continuity of Bézier patches is discussed in terms of β constraints as developed by DeRose [47]. Hahn [96] characterizes G^k continuity in terms of a connecting diffeomorphism (reparameterization). He considers continuity between adjacent patches and where several patches meet around a vertex. Pegna and Wolter [155] provide alternative conditions for G^2 continuity across patch-patch boundaries. Considering a pair of patches that meet with G^1 continuity, they show that these patches need only have equal normal curvatures in a single direction that crosses the

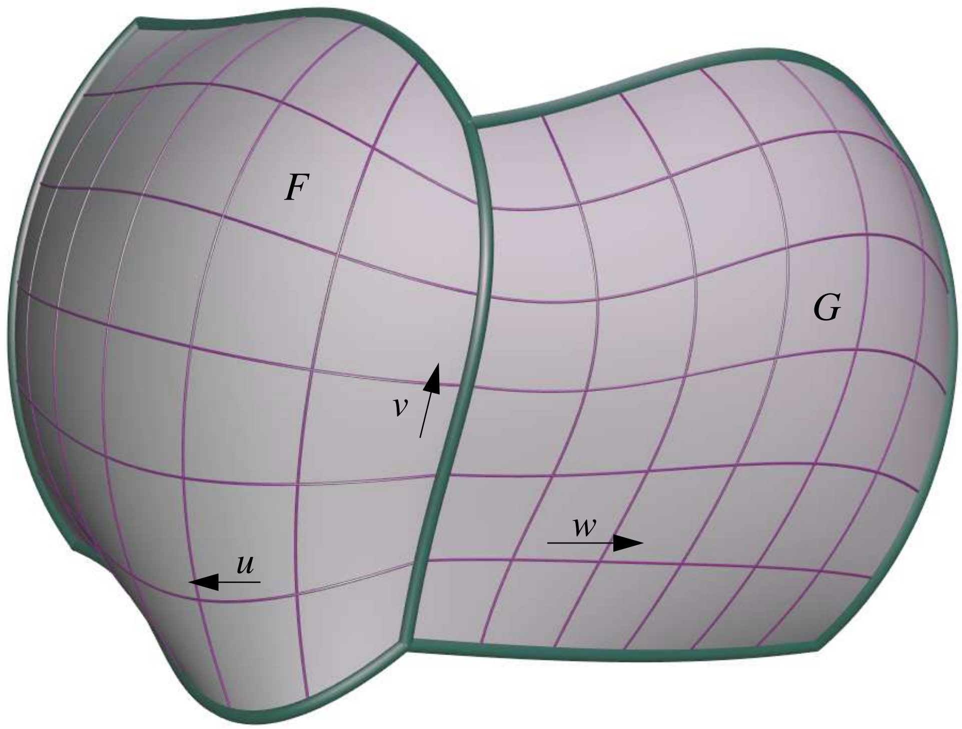

Figure 3.21. Patch-patch Continuity
Patches $\vec{F}(u, v)$ and $\vec{G}(v, w)$ meeting with tangent continuity.

patch-patch boundary in order to assure G^2 continuity. They also list other sufficient conditions for G^2 continuity across patch-patch boundaries: the osculating paraboloids agree along the boundary, the principal curvatures and directions agree, the Dupin's indicatrices and curvature signs agree everywhere, the asymptotic directions agree, and the second fundamental tensors are identical at all points. Degen [46] discusses constructions for joining adjacent Bézier patches with G^1 and G^2 continuity. The G^1 construction assumes both a boundary curve and a cross boundary tangent function. This symmetric approach is also used to form G^2 boundaries.

Vinacua and Brunet [207] present a construction for G^1 continuity between rational Bézier patches. DeRose [49] describes the necessary and sufficient conditions for tangent continuity between tensor product and triangular Bézier patches in both polynomial and rational forms. Liu [120] discusses conditions for G^1 continuity between tensor product Bézier patches and between triangular Bézier patches; both the polynomial and rational cases are considered.

3.3.2.2 Vertex Enclosure

Van Wijk [208] studies the vertex enclosure problem and provides a solution for the computation of a G^1 surface with vertices of order 4 or order *n*, where *n* is odd. Sarraga [188] characterizes the vertex enclosure problem and provides a solution to the interpolation of an unrestricted mesh of cubic Bézier curves meeting with G^1 continuity. The mesh is restricted to vertices of order 3, 4, or 5 with four sided openings. In the first of a series of papers on smooth interpolation [160], Peters describes rectangulation as a method of surface construction. In the first step of this method a triangulation is converted to a rectangulation (Fig. 3.22). This transformation allows the construction of a G^1 continuous surface composed of bicubic and biquartic patches. Peters [158] presents a nice classification of methods for solving the G^1 vertex enclosure problem. Peters presents a solution to the odd order vertex problem where the surface normal varies linearly along the patch-patch boundary. Next, Peters [156] presents a scheme based on patch splitting for the creation of a G^1 surface. The resulting surface is composed of cubic Bézier patches. Then, he describes an alternative method that exploits a parametric singularity to solve the vertex enclosure problem [159]. Finally, Peters [161] describes a solution to the vertex enclosure problem. A G^2 compatible, otherwise arbitrary, cubic curve mesh is interpolated using quartic and quintic tensor product and triangular patches.

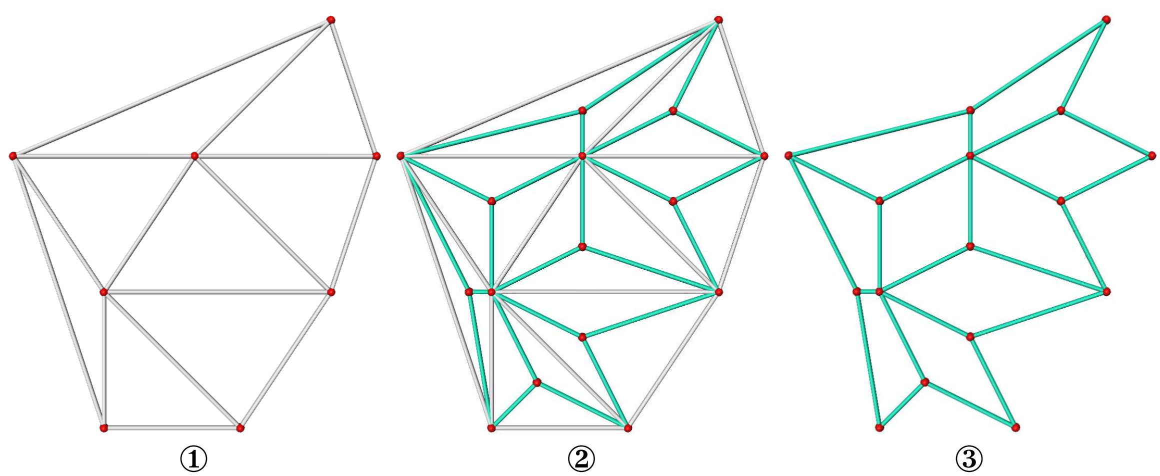

Figure 3.22. Rectangulation.

A triangular mesh ① is converted to a quadrilateral mesh ③ by Clough-Tocher subdivision ② and dropping the original edges.

3.3.3 Finite Element Analysis, Minimization, Optimization, and Fairing

Pramila [171] uses finite element techniques to design ship hull surfaces. A linearized fairness functional is designed to minimize strain energy. Klass [117] describes a fairing system that corrects surface irregularities by smoothing lines of reflection. Modifications to lines of reflection are mapped back to modifications to the surface from which they were derived. Kjellander [116] proposes fairing tensor product surfaces by fairing meshes of curves. This is accomplished by reducing the strain energy associated with a mesh point interactively selected by a designer. Nowacki and Reese [145] describe a ship surface definition and fairing system under development. They propose to minimize the strain energy of G^2 Coons patches by varying their interior degrees of freedom. Hagen and Schulze [91] describe a system for computing a surface from points in $\Re^3$. Their approach uses functional minimization to shape generalized Coons patches. Linearized strain energy is used as the surface objective function. Ferguson, et al. [62] use constrained optimization to generate B-spline surfaces with convex isocurves. Farin and Sapidis [57] present a local method for curve and surface fairing. The approach works based on knot removal and reinsertion. The surface is faired by fairing the curves corresponding to the rows and columns of the control network. Curves are faired by identifying areas of undesirable curvature and the associated knot. The offending knot is then removed and reinserted, resulting in a curve of improved fairness. Nowacki, et al. [147] describe a surface approximation scheme based on minimization. They minimize the sum of the strain energy of mesh lines and the potential energy of springs attaching the mesh lines to data points. Rando and Roulier [177] discuss fairness metrics for curve and surface design. They observe that fairness metrics should parameterization independent; fairness metrics should be a function of intrinsic surface geometry. Kallay and Ravani [115] describe a method based on the minimization of a functional approximating strain energy for computing the twist vectors of a mesh of bicubic patches with fixed boundaries. Their approach is applicable to any quadratic functional. Rando and Roulier [179] discuss the fairing of Bézier patches by iteratively adjusting control points to minimize various fairness functionals that are designed for specific surface types. Their functionals include: *flattening*—tend toward developable surfaces, *rounding*—tend toward spherical surfaces, *rolling*—tend toward cylindrical or conical surfaces. Celniker and Gossard [31] develop curve and surface elements for performing energy based surface modeling. The surface element is a rational polynomial and is joined with neighboring elements with G^1 continuity. Their modeling scheme computes surfaces that minimize linearized strain energy while interpolating data points and simulating forces applied to the surface.

3.3.4 Surveys

Several authors have compiled excellent surveys concerning various aspects of free-form surface design.

By Böhm, et al.:

> *A Survey of Curve and Surface Methods in CAGD* [18].

By Herron:

> *Techniques for Visual Continuity* [97].

By Gregory et al.:

> *Smooth Parametric Surfaces and n-Sided Patches* [88].

By Peters:

> *Local Smooth Surface Interpolation: A Classification* [158].

By Mann et al.:

> *A Survey of Parametric Scattered Data Fitting Using Triangular Interpolants* [126].