GEOMETRIC MODELING: A First Course

4. Curves and Surfaces

This chapter discusses mathematical models and computational representations for curves and surfaces.

4.1 Mathematical Models for Curves and Surfaces

We all have an intuitive understanding of curves and surfaces. But can we answer mathematically these basic questions: What is a curve? What is a surface? It turns out that there are several acceptable answers, and that different branches of mathematics use different definitions. We first introduce some of the fundamental notions through the simplest possible examples: the straight line, which is a special case of a curve, and the plane, which is a special case of a surface.

4.1.1 Lines and Planes

A straight line, or simply a *line*, in Euclidean space is a set of points **p** that satisfy

$$\mathbf{p}-\mathbf{p}_0 = u(\mathbf{p}_1-\mathbf{p}_0), \quad u \quad (-\infty,+\infty), \quad \mathbf{p}_0 \neq \mathbf{p}_1 .$$

Here $\mathbf{p}_0$ and $\mathbf{p}_1$ are arbitrary but distinct points of the line. The equation above contains a parameter u and is called a *parametric equation*. As the parameter u takes all possible values from minus infinity to plus infinity, the point **p** traces the entire line.

The parametric equation of the line can be written in a different format. Algebraic manipulation of the original equation yields

$$\mathbf{p} = (1-u)\mathbf{p}_0 + u\mathbf{p}_1.$$

This is the fundamental equation of *linear interpolation*. It shows that

$$u=0 \quad \mathbf{p}=\mathbf{p}_0$$
$$u=1 \quad \mathbf{p}=\mathbf{p}_1$$

i.e., the line interpolates the two given points. Furthermore, for $0 \leq u \leq 1$, the point **p** is at an intermediate location between the two endpoints—see Figure 4.1.1.1. Therefore the equation of a *line segment* is simply

$$\mathbf{p}-\mathbf{p}_0 = u(\mathbf{p}_1-\mathbf{p}_0), \quad u \quad [0,1], \quad \mathbf{p}_0 \neq \mathbf{p}_1 .$$

Letting

$$a_0 = 1 - u$$
$$a_1 = u$$

the interpolation equation can also be written as

$$\mathbf{p} = a_0\mathbf{p}_0 + a_1\mathbf{p}_1$$
$$a_0 + a_1 = 1$$

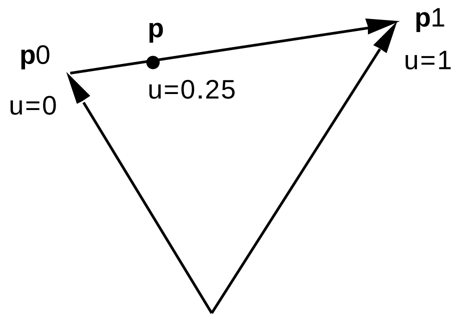

Figure 4.1.1.1 – Linear interpolation

Let us now consider a plane that passes through a point $\mathbf{p}_0$ and is normal to a unit vector $\boldsymbol{n}$. the equation of the plane is

$$(\mathbf{p} - \mathbf{p}_0).\boldsymbol{n} = 0$$

Figure 4.1.1.2 illustrates the geometry involved. If we denote the point coordinates and vector components in a given frame by

$$\mathbf{p} \leftrightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \boldsymbol{n} \leftrightarrow \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

and let

$$\mathbf{p}_0.\boldsymbol{n} = -d \ ,$$

the equation of the plane takes the familiar form

$$ax + by + cz + d = 0 \ .$$

This is called an *implicit equation*. Note that –*d* is the (perpendicular) distance between the plane and the origin.

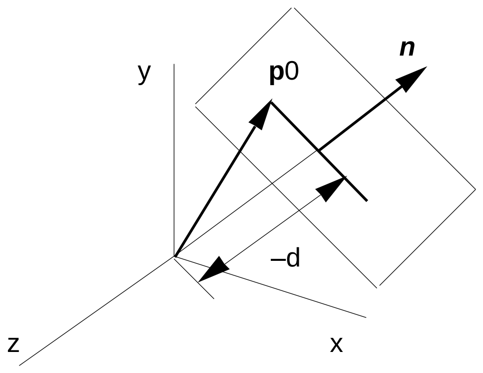

Figure 4.1.1.2 – A plane defined by a point and a normal vector

We defined a line by its parametric equation and a plane by its implicit equation. Lines also have implicit equations and planes parametric equations. For a line we consider two non-parallel planes and write

$$a_1 x + b_1 y + c_1 z + d_1 = 0$$
$$a_2 x + b_2 y + c_2 z + d_2 = 0$$

This system of equations defines a line implicitly. For a plane we take three non-collinear points and write the parametric equation as follows

$$\mathbf{p} - \mathbf{p}_0 = u(\mathbf{p}_1 - \mathbf{p}_0) + v(\mathbf{p}_2 - \mathbf{p}_0), \quad u, v \quad (-\infty, +\infty) .$$

This equation essentially states that **p** is an arbitrary point in a 2-D Euclidean space with a (generally non-orthonormal) frame defined by the three given points, with $\mathbf{p}_0$ as origin, and basis vectors $\mathbf{p}_1 - \mathbf{p}_0$ and $\mathbf{p}_2 - \mathbf{p}_0$. Because a plane is a two-dimensional entity, its parametric equation has two independent parameters u and v.

A rectangular subset of a plane, often called a rectangular *patch*, is easy to define, simply by restricting the values taken by u and v to a 2-D interval, e.g., $a \le u \le b, c \le v \le d$. Other subsets of a plane are not so easy to define. We will return to this issue later in this chapter.

4.1.2 Curves

Calculus textbooks and most of the literature on curve modeling define a curve as a set of points **p** that satisfy a set of parametric equations:

$$p_x = f_x(u)$$
$$p_y = f_y(u) .$$
$$p_z = f_z(u)$$

These can be abbreviated as

$$\mathbf{p} = \mathbf{f}(u) ,$$

where **f** is usually called a vector-valued function, because it has three components. The equations of a straight line are of this form, with **f** linear. If we let u range over the entire set of reals we obtain the complete curve. If we restrict u to an interval, we define a *curve segment.*

It is important to understand that the same curve (i.e., set of points) can be defined by different parametric equations. The function **f** defines not only a curve but also a *parameterization* for it, and there are many ways of parameterizing the same curve.

The function **f** and some of its derivatives are usually required to be continuous. If **f** is continuous, the curve is called C^0 continuous. If both the function and its first derivative are continuous, we say the curve is C^1 continuous. In general, a curve is C^i continuous if **f** plus its first i derivatives are continuous.

If a curve is parameterized by its *arc length* s, the derivatives of the generic point of the curve $\mathbf{p}(s)$ are related to the tangent and normal to the curve as follows

$$\frac{d\mathbf{p}}{ds} = \boldsymbol{t}$$
$$\frac{d\boldsymbol{t}}{ds} = \frac{1}{\rho}\boldsymbol{n}$$

Here $\boldsymbol{t}$ is the unit vector tangent to the curve, $\boldsymbol{n}$ is the unit normal, and ρ is the radius of curvature, i.e., the radius of the circle that best approximates the curve at point $\mathbf{p}$. These equations are derived in the standard texts on geometry, e.g. [do Carmo 1976]. Formulas for the tangent and normal also exist for other parameterizations, but are more complicated. We can construct a frame for each point $\mathbf{p}$ by defining a third vector $\boldsymbol{b}$, called the bi-normal, as

$$\boldsymbol{b} = \boldsymbol{t} \times \boldsymbol{n} \ .$$

The three vectors constitute an orthonormal frame, called the *Frenet frame*, that in general varies as $\mathbf{p}$ traces the curve.

C continuity, defined above, is also known as *parametric continuity,* because it is expressed in terms of the parameterization. Alternatively, one can define *geometric continuity,* which is intrinsic to a curve itself, and does not depend on the parameterization. A curve is G^1 continuous if its unit tangent varies continuously, and is G^2 if both the unit tangent and normal are continuous. It can be shown that parametric and geometric continuity are slightly different notions, and neither one implies the other. G continuity is the more important concept for applications, but it is harder to establish, and one often settles for C continuity.

The parametric curves most commonly used in geometric modeling are defined by functions **f** whose three components are polynomials in u, or are the quotients of two polynomials in u. They are known as *parametric polynomial* or *parametric rational* curves, respectively. Cubic curves are especially important because they can exhibit C^2 continuity. The equations of a parametric cubic can be written as

$$x = a_3u^3 + a_2u^2 + a_1u + a_0$$
$$y = b_3u^3 + b_2u^2 + b_1u + b_0$$
$$z = c_3u^3 + c_2u^2 + c_1u + c_0$$

If we interpret the coefficients as the coordinates of points

$$\mathbf{p}_3 \leftrightarrow \begin{bmatrix} a_3 \\ b_3 \\ c_3 \end{bmatrix}, \quad \mathbf{p}_2 \leftrightarrow \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} \quad \mathbf{p}_1 \leftrightarrow \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} \quad \mathbf{p}_0 \leftrightarrow \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix}$$

the curve equations become

$$\mathbf{p} = \mathbf{p}_3u^3 + \mathbf{p}_2u^2 + \mathbf{p}_1u + \mathbf{p}_0 \ ,$$

which shows that the generic point of a parametric polynomial curve can be expressed as a linear combination of other points, usually called *control points*.

The parametric definition of a curve has its limitations. For example, space-filling “curves” can be defined by continuous **f** functions. These “curves” contradict our intuition of a curve as a 1-D entity, because they actually correspond to 2-D or 3-D regions of space. In addition, parametric curves can self-intersect. If self-intersections are undesirable, we can model a curve by a different mathematical entity called a manifold.

A *closed n-manifold* is a set that is locally just like an Euclidean space. For example, each point in a closed 1-manifold has a small interval around it that can be elastically deformed into an interval of the real line. An elastic deformation, without cutting or glueing, is called in topology a *homeomorphism*. Two sets are called *topologically equivalent* if they are related by a homeomorphism. Topology, like Euclidean geometry, is the study of properties of objects that remain invariant under certain transformations. For Euclidean geometry the relevant transformations were the isometries, whereas for topology they are the homeomorphisms. The top three 1-D objects of Figure 4.1.2.1 are homeomorphic, and so are the three bottom objects. However, top objects are not homeomorphic to bottom ones, because a top object cannot be elastically deformed into a bottom one without glueing its endpoints.

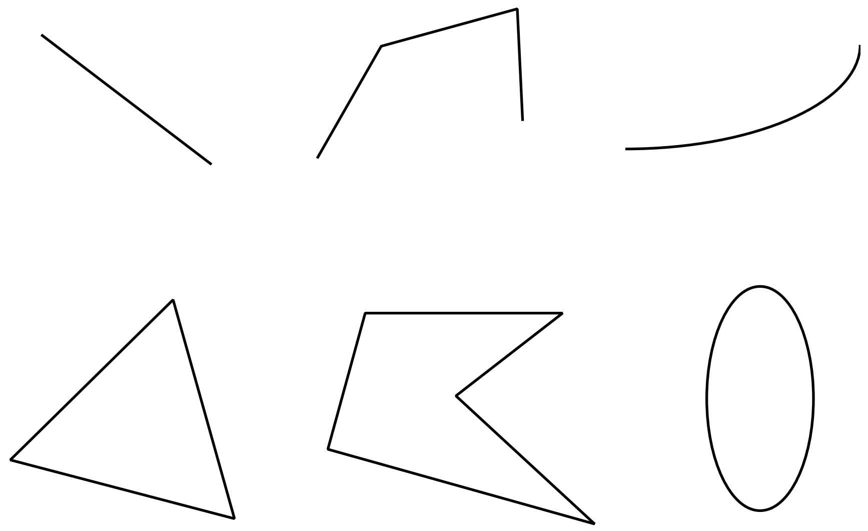

Figure 4.1.2.1 – Compact, connected 1-manifolds

The bottom objects in the figure are examples of closed 1-manifolds. The top objects are *bordered 1-manifolds*. Imagine a small disk centered about each of the points **p** of a 1-manifold X. Recall that the intersection of a disk centered at **p** with the set X is called the neighborhood of the point with respect to the set (Section 3.2.6). If X is a manifold, a neighborhood of **p** must be homeomorphic either to an open interval of the real line $(x - \varepsilon, x + \varepsilon)$ or to a "half interval" $[x, x+\varepsilon)$. If there are no points that correspond to half-intervals, the manifold is closed; otherwise it is bordered. It can be shown that there are only two kinds of connected compact (i.e. closed and bounded) sets in Euclidean space that are 1-manifolds. They are the line segment and the circle. All other connected compact 1-manifolds are topologically equivalent to either a line segment or a circle. Therefore Figure 4.1.2.1 illustrates all the topologically distinct connected compact 1-manifolds.

In contrast, Figure 4.1.2.2 shows a non-manifold. The point of contact of the two ellipses violates the definition of manifold. Its neighborhood is not homeomorphic to either an interval or a half interval; rather, it is topologically equivalent to a "cross" formed by two intersecting line segments, as shown in the figure.

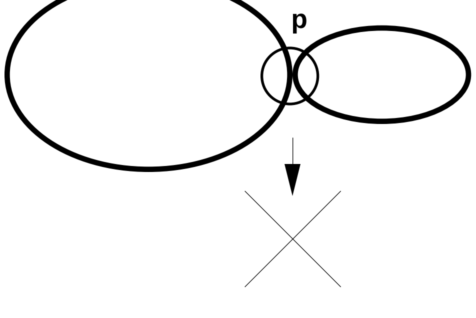

Figure 4.1.2.2 – Point **p** has a neighborhood homeomeorphic to two intersecting segments, and therefore the two ellipses do not constitute a 1-manifold

There is yet another definition of curve, through implicit equations:

$$f_1(x,y,z)=0$$
$$f_2(x,y,z)=0$$

This is a generalization of the implicit equation of a straight line, discussed in the previous section. The most interesting class of curves defined implicitly arises when the functions f are *algebraic*, i.e., they are polynomials in the spatial variables x, y, z.

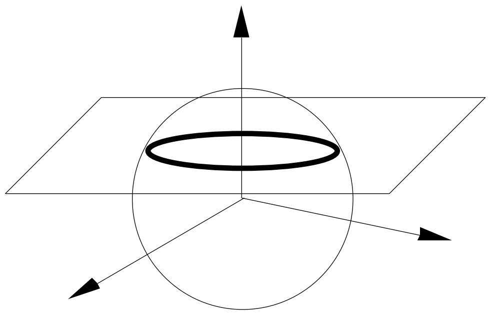

Figure 4.1.2.3 – A circle defined implicitly

A set defined by a system of algebraic equations is called an *algebraic variety*. There is a large body of mathematics, called *algebraic geometry*, that studies algebraic varieties. *Algebraic curves* in 3-space are special cases of algebraic varieties, defined by two simultaneous equations. Each of the equations per se corresponds to an *algebraic surface*, and the curve is defined as the intersection of the two surfaces. Figure 4.1.2.3 shows a circle defined by intersecting a sphere with a plane, with equations

$$x^2+y^2+z^2=1$$
$$z=0.5$$

Defining a curve as the intersection of two algebraic surfaces can also lead to non-intuitive "curves". For example, the intersection of two touching spheres is a point, not a 1-D entity. Therefore, it would be better to define a curve algebraically as the 1-D portion of the intersection variety. Unfortunately, computing the dimension of a general algebraic variety is a difficult problem.

All curves can be parameterized, but algebraic curves do not necessarily admit polynomial or rational parameterizations. On the other hand, parametric polynomial or rational curves can be *implicitized* and defined through algebraic varieties, but the degrees of the corresponding implicit equations generally are much higher than those of the original parametric equations.

4.1.3 Surfaces

We saw in the previous section that in mathematics there are several concepts of "curve". For surfaces, the situation is analogous. In calculus and differential geometry a surface is defined by parametric equations:

$$\begin{aligned} x &= f_x(u,v) \\ y &= f_y(u,v) \;. \\ z &= f_z(u,v) \end{aligned}$$

Now two parameters are needed, because surfaces are intrinsically two dimensional. These equations may be abbreviated as

$$\mathbf{p} = \mathbf{f}(u,v).$$

The vector-valued function **f** usually is required to satisfy certain continuity, or smoothness, conditions. If we restrict the parameters to a 2-D interval $a \le u \le b$, $c \le v \le d$, the resulting subset of the surface is a ("rectangular") patch. Other subsets of a parametric surface are not as easy to define. Typically they are defined through boundary representations, as discussed in Chapter 5.

Parametric surfaces used in geometric modeling typically are either polynomial or rational, i.e., the components of the function **f** are either polynomials in the u, v, variables or are quotients of two polynomials on these parameters.

At each point of a smooth surface there are infinitely many lines tangent to the surface. These lines define the tangent plane at the point. The tangent plane is perpendicular to the normal to the surface.

If we let

$$\begin{aligned} u &= g_u(t) \\ v &= g_v(t) \end{aligned}$$

and substitute in the parametric equation of a surface, we obtain

$$\mathbf{p} = \mathbf{f}[g_u(t), g_v(t)] = \mathbf{h}(t) \;.$$

This is an equation in a single parameter, and therefore it defines a curve. Since the points of the curve also satisfy the equation of the surface, the curve must lie on the surface. Different choices of functions g correspond to different curves on the surface. If we fix one of the parameters and let the other vary,

$$\begin{aligned} u &= u \\ v &= v_0 \end{aligned} \;,$$

or

$$\begin{aligned} u &= u_0 \\ v &= v \end{aligned} \;,$$

the corresponding curves

$$\begin{aligned} \mathbf{p} &= \mathbf{f}(u,v_0) = \mathbf{h}_{v_0}(u) \\ \mathbf{p} &= \mathbf{f}(u_0,v) = \mathbf{h}_{u_0}(v) \end{aligned}$$

are called *constant-parameter curves*. The tangents to the constant-parameter curves are the derivatives of the **h** functions. Therefore, the tangent vector to a constant-*v* curve is $\frac{\partial \mathbf{p}}{\partial u}$, and the tangent to a constant-*u* curve is $\frac{\partial \mathbf{p}}{\partial v}$. The normal to the surface must be perpendicular to both of these tangents, and therefore (assuming the two vectors are not parallel) the unit normal is given by

$$\boldsymbol{n} = \frac{\dfrac{\partial \mathbf{p}}{\partial u} \times \dfrac{\partial \mathbf{p}}{\partial v}}{\left|\dfrac{\partial \mathbf{p}}{\partial u} \times \dfrac{\partial \mathbf{p}}{\partial v}\right|} .$$

A planar surface has a fixed normal, but in general ***n*** varies from point to point on a curved surface. The set of normals to all the points of a surface constitute the surface's *Gaussian image*. One can think of the Gaussian image as a set of directions, or *direction cone*, or, equivalently, as the subset of the unit sphere where the normal directions intersect the sphere. The unit sphere is sometimes called the *Gaussian sphere*. For example, for a spherical surface the Gaussian image is the entire unit sphere; for a plane it is a single point; and for a cylinder it is a great circle on the Gaussian sphere. Gaussian images are used extensively in Computer Vision and in other applications, such as visibility studies.

Let us now consider the topological point of view. In topology a surface is a *2-manifold*, i.e., a set *X* such that the neighborhood with respect to *X* of each point **p** of *X* is topologically equivalent to either an open disk (i.e., a 2-D open ball) or to half a disk. If all neighborhhods are homeomorphic to disks, the manifold is *closed*, otherwise it is *bordered*. This definition is a direct generalization of the earlier notion of 1-manifold we encountered in the previous section, and can be generalized further, to spaces with higher dimensionality.

Figure 4.1.3.1 shows that the surface of a solid cube is an example of a piecewise-planar, or *polyhedral*, closed 2-manifold. It is a collection of polygonal faces such that the neighborhoods of vertices, or of points in the interior of an edge, or in the interior of a face, all can be deformed elastically so as to become disks.

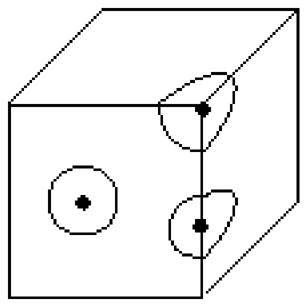
Figure 4.1.3.1 – The surface of a solid cube is a closed 2-manifold

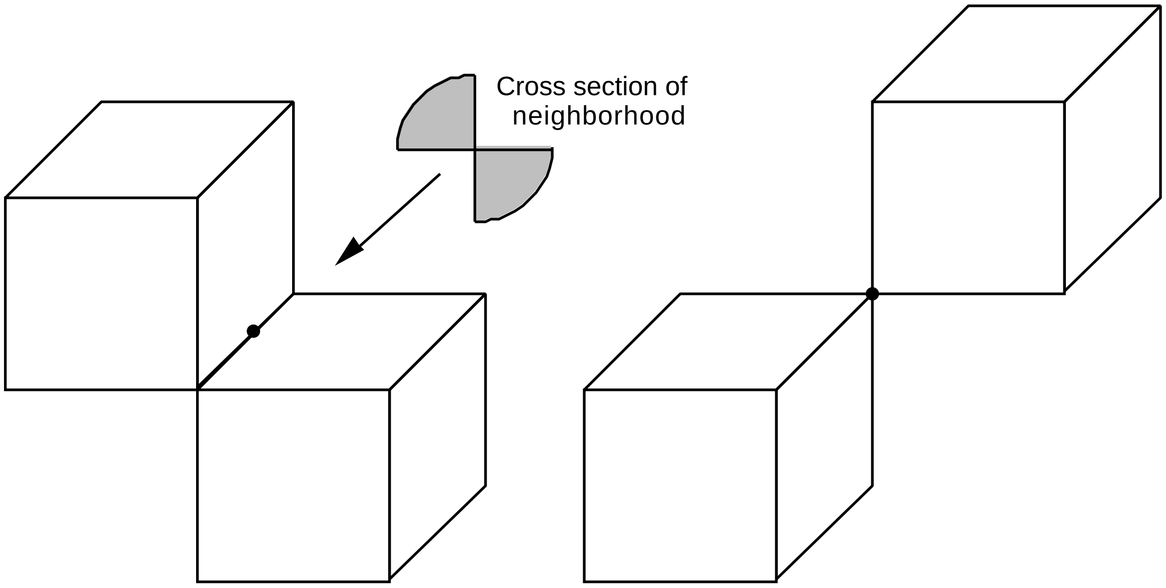

Figure 4.1.3.2 – The boundaries of two cubes glued at an edge or at a vertex are not 2-manifolds

In contrast, Figure 4.1.3.2 shows two polyhedral “surfaces” that are not manifolds. The object on the left fails to be a manifold because it has an edge that is shared by more han two faces. The neighborhhods of points on the common edge are made out of four half disks that cannot be flattened into a single disk without gluing. The object on the right has a vertex that is shared by two cones of faces. Its neighborhood can be deformed into two disks, but these have to be glued to obtain a single disk.

A surface that crosses itself has at least one edge of self-intersection, and therefore is analogous to the two cubes glued along an edge in Figure 4.1.3.2. It follows that self-intersecting surfaces are not 2-manifolds. It is also clear from this figure that the union of two 2-manifolds (the cube boundaries) may produce objects that are not manifolds. In other words, 2-manifolds are not algebraically closed under set operations.

We saw earlier that all connected, compact, closed 1-manifolds are topologically equivalent to a circle. For closed 2-manifolds the situation is much more complicated. First, there are so-called non-orientable 2-manifolds, which cannot be built in Euclidean 3-space, and therefore will not be discussed in this course. A general orientable 2-manifold that is closed, connected and compact must be homeomorphic to a sphere with n “handles” or “holes”, for $n = 0, 1, ...$ Figure 4.1.3.3 shows on the left a polyhedral 2-manifold that is topologically equivalent to a sphere with one handle, shown on the right. The sphere with one handle is itself homeomeorphic to a torus, *i.e.*, to a doughnut-shaped surface. It is easy to see if two manifolds are equivalent by imagining one being deformed elastically into the other.

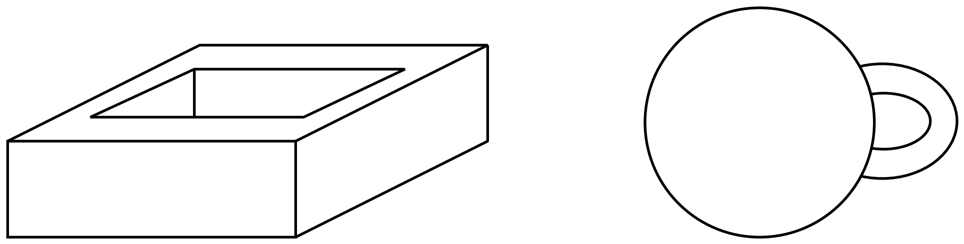

Figure 4.1.3.3 – A polyhedral surface and a topologically equivalent sphere with one handle

Finally, from the point of view of algebraic geometry a surface is an algebraic variety defined by a single implicit equation

$$f(x,y,z) = 0$$

where f is a polynomial in the spatial variables x, y, z. Algebraic surfaces may self-intersect, and sometimes are not truly two-dimensional.

A plane is the simplest example of an algebraic surface, and is defined by a linear implicit equation. Equations of degree higher than one correspond to curved surfaces. For example, a sphere of radius R, centered at the origin, is defined by

$$x^2 + y^2 + z^2 - R^2 = 0 \ ,$$

and a cylinder of radius R and axis coincident with the z coordinate axis is defined by

$$x^2 + y^2 - R^2 = 0 \ .$$

The normal to a surface $f(x,y,z) = 0$ is parallel to its *gradient* vector, defined as

$$f \leftrightarrow \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} .$$

Parametric polynomial or rational surfaces can be implicitized, but the resulting algebraic varieties typically have much higher degrees.

4.2 Representations for Curves

We saw in Section 4.1.2 that there are at least three useful models for curves: images of parametric functions, 1-manifolds, and algebraic (implicit) curves. This section focuses on parametric curves, because parameterizations are required by most of the algorithms that process curves. Many geometric modeling systems adopt 1-manifold models for curves, but still represent them parametrically. We begin with mathematical preliminaries. First we show that vector space concepts are useful to study polynomials, and introduce multivariate polynomials, called *blossoms*, which are associated with single-variable polynomials and are very useful computationally. Then we focus on approximation schemes for representing curves. We discuss Bézier and B-spline methods, which are the most widely used. Finally, we look at parametric curve representation by primitive instancing. Most of our examples are second degree curves, for simplicity, although the most commonly used curves in practice are cubics, because they can be C^2 continuous.

4.2.1 Vector Spaces of Polynomials

Consider the set of all the polynomials $f(u)$ in a single variable u that have degrees at most d. Addition of polynomials is defined as usual, by summing same-power coefficients. Multiplication by a real scalar corresponds to multiplying all the coefficients by the scalar. It is very easy to prove that this set with the two operations satisfies all the required axioms and is a vector space. The dimension of the space equals the maximum number of coefficients in a degree d polynomial, which is $d + 1$.

A second degree polynomial in u can be written as

$$f(u) = \begin{bmatrix} 1 & u & u^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix},$$

where the a's are the usual coefficients of the powers of u. Comparing this expression with the expansion of a vector as a linear combination of a basis

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{e}_0 & \boldsymbol{e}_1 & \boldsymbol{e}_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

we see that the monomial functions

$$\begin{bmatrix} 1 & u & u^2 \end{bmatrix}$$

constitute a basis for the space of polynomials of degree ≤2. The components of the vector (*i.e.*, polynomial) f are the coefficients

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}.$$

The monomial basis is also called the *power basis.*

If we select a basis, any polynomial in the vector space can be represented by the column matrix of its components. Any choice of basis leads to representations that are unambiguous, unique, and of the same size. However, these representations differ considerably in other characteristics. For example, the power basis has severe computational drawbacks, because it often introduces large numerical errors in computations. It is also ill-suited for curve *design*, because the coefficients do not have a simple, intuitive effect on the shape of the curve.

A quadratic polynomial, being a vector in a 3-D space, has 3 degrees of freedom. These can be constrained in a geometrically meaningful fashion so as to control the polynomial shape. For example, let us require that a polynomial be represented by the following three quantities:

$$b_0 = f(0)$$
$$b_1 = f(0) + \frac{1}{2} f'(0)$$
$$b_2 = f(1)$$

where the prime denotes the derivative. The geometric meaning of these constraints is shown in Figure 4.2.1.1.

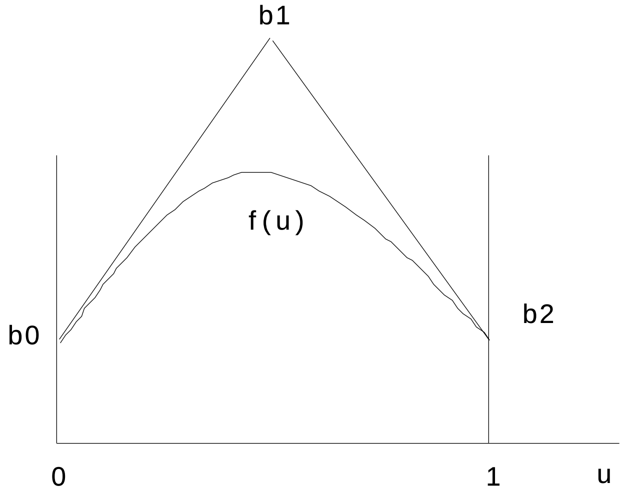

Figure 4.2.1.1 – A quadratic polynomial and its three defining points.

The derivative of *f* is

$$f'(u) = \begin{bmatrix} 0 & 1 & 2u \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

and therefore the constraints above can be expressed in the power basis representation as

$$f(0) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = b_0$$

$$f(0) + \frac{1}{2} f'(0) = \begin{bmatrix} 1 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = b_1$$

$$f(1) = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = b_2$$

These three equations can be written together in matrix form as

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & ½ & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}.$$

The column matrices contain the components of the vector *f* in two basis:

$$F^p = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \quad F^b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

Here F^p corresponds to the power basis components. The other basis is called the *Bernstein basis*, and the corresponding polynomials are the *Bernstein polynomials*.

The matrix equation that expresses the constraints can be re-written as

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & ½ & 0 \\ 1 & 1 & 1 \end{bmatrix} F^p = F^b$$

But we know from Section 3.4 that the components of a vector in the two basis are related by

$$F^p = M^b_{pb} F^b$$

where M^b_{pb} is the matrix corresponding to the transformation that maps the vectors of the power basis into those of the Bernstein basis. It follows that

$$(M^p_{pb})^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & ½ & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

Inverting this matrix gives us the desired tranformation matrix

$$M_{pb}^{p} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix}.$$

We also know that the columns of this matrix contain the components of the new basis expressed in the old basis. Therefore, the Bernstein polynomials are the following

$$\begin{aligned} B_0^2(u) &= 1-2u+u^2 = (1-u)^2 \\ B_1^2(u) &= 2u-2u^2 = 2u(1-u)\,. \\ B_2^2(u) &= u^2 \end{aligned}$$

Here the superscript in the basis functions denotes the maximum degree of the polynomials in the space.

A similar analysis for an arbitrary degree *d* yields the general Bernstein polynomials:

$$B_i^d = \binom{d}{i} u^i (1-u)^{d-i} \quad .$$

This is precisely the form of the binomial distribution, well known in probability and statistics. For example, if the probability of heads in a coin toss is u, the probability of exactly i heads in d tosses is the Bernstein polynomial $B_i^d(u)$. The probabilistic interpretation of the Bernstein basis can be exploited in the geometric modeling context—see [Goldman 1983].

Thus far we have considered only one polynomial function. Now let us define a point $\mathbf{p}(u)$ lying in a parametric curve in the Euclidean plane, with coordinates

$$\mathbf{p}(u) \leftrightarrow \begin{bmatrix} x(u) \\ y(u) \end{bmatrix},$$

and let these coordinates be second degree polynomials in u. Proceeding as in Section 4.1.2 we can write

$$\mathbf{p}(u) = \begin{bmatrix} B_0^2(u) & B_1^2(u) & B_2^2(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}.$$

A curve in this representation is called a *Bézier curve*, and the **b**’s are called its *Bézier control points*, which together form the *control polygon*. It follows from our derivation for the single-function case that

$$\mathbf{p}(0) = \mathbf{b}_0$$
$$\mathbf{p}(0) + \frac{1}{2}\boldsymbol{t}(0) = \mathbf{b}_1 ,$$
$$\mathbf{p}(1) = \mathbf{b}_2$$

where $\boldsymbol{t}$ is the tangent to the curve.

4.2.2 Blossoms

We now discuss the *blossoming* approach, which provides very useful tools for dealing with Bézier and spline curves. The *blossom* $g(u_1, u_2, \cdots, u_d)$ that corresponds to a polynomial $f(u)$ of degree d is a multivariate polynomial with the following properties:

1) $f(u) = g(u, u, \cdots, u)$
2) $g(u_1, u_2, \cdots, u_d) = g[Permutation(u_1, u_2, \cdots, u_d)]$
3) $g(\cdots, u_i = a_0 q_0 + a_1 q_1, \cdots) = a_0 g(\cdots, u_i = q_0, \cdots) + a_1 g(\cdots, u_i = q_1, \cdots), \quad a_0 + a_1 = 1$

The first, or *diagonal*, property shows that the univariate polynomial can be recovered from its blossom by setting all the variables in the blossom to the same u value. Note that there are d variables in the blossom g for an f polynomial of degree d. The second property indicates that the order of the variables in the blossom is unimportant. The third provides an interpolation formula for computing blossom values, and will be used extensively in the algorithms described below. It can be shown [Ramshaw 1987] that the blossom of a polynomial $f(u)$ is well-defined and always exists.

Let us begin our exploration of blossom algorithms. We will work with second degree polynomials for simplicity, but the generalization to higher degrees is obvious. We start with the blossom values $g(0,0)$, $g(0,1)$ and $g(1,1)$ and compute $f(u) = g(u,u)$ for any u between 0 and 1 as follows. We interpolate between $g(0,0)$ and $g(0,1)$ by using Property 3 of the blossom, setting

$$u_i = u_2$$
$$q_0 = 0, \quad q_1 = 1$$
$$a_0 = 1 - u, \quad a_1 = u$$

so as to obtain

$$g(0,u) = (1-u)g(0,0) + ug(0,1) \ .$$

We interpolate similarly between $g(0,1)$ and $g(1,1)$ to compute $g(u,1)$. Then we interpolate once more between the intermediate results $g(0,u)$ and $g(u,1)$ to produce the final value $g(u,u)$. This computation can be summarized as shown by the graph of Figure 4.2.2.1, where the arcs denote multiplication by the factors indicated, and the nodes denote sum.

$$\begin{tikzcd}
g(0,0) \arrow[dr, "1-u"'] & & g(0,1) \arrow[dl, "u"] \arrow[dr, "1-u"'] & & g(1,1) \arrow[dl, "u"] \\
& g(0,u) \arrow[dr, "1-u"'] & & g(u,1) \arrow[dl, "u"] & \\
& & g(u,u) & &
\end{tikzcd}$$

Figure 4.2.2.1 – Blossom evaluation by successive interpolation

Doing the calculations in full yields

$$f(u) = g(u,u) = g(0,0)[(1-u)(1-u)] + g(0,1)[u(1-u) = (1-u)u] + g(1,1)u^2 \ .$$

The functions of u that multiply the g values are the Bernstein basis functions of the previous section, and therefore

$$f(u) = g(u,u) = g(0,0)B_0^2(u) + g(0,1)B_1^2(u) + g(1,1)B_2^2(u) \ .$$

It follows that the g's must be the Bézier points of f, *i.e.*, the components of the (polynomial) vector f in the Bernstein basis:

$$\begin{aligned} g(0,0) &= b_0 \\ g(0,1) &= b_1 \\ g(1,1) &= b_2 \end{aligned}$$

A similar computation can be used for each coordinate of a point $\mathbf{p}(u)$ on a Bézier curve. Therefore we discovered a computational scheme for evaluating points on a Bézier curve by successive interpolation of its Bézier control points. This procedure has a simple geometric interpretation shown in Figure 4.2.2.2. In the figure, and elsewhere in the sequel, we denote points by the sequence of arguments in the corresponding blossom values. For example, the point $g(0,1)$ is denoted simply by 01.

The figure illustrates the computation of the point that corresponds to u=1/3, given the Bézier points 00, 01 and 11. First we intepolate between 00 and 01. To do this, we construct a point 0u along the segment 00 to 01, at a distance from 00 that corresponds to one third (u=1/3) of the distance between 00 and 01. Similarly, we construct a point u1 along the segment 01 to 11. Finally, we interpolate between 0u and u1 to find uu. Note that this is a purely geometric construction, involving only ratios of line segments. This computation is fast and numerically stable, and is known as *de Casteljau's algorithm*, after its inventor, an engineer with the French car maker Citroën. (Interestingly, Bézier and de Casteljau, working independently and for competing companies, came up with essentially the same approach for defining free-form curves and surfaces. Bézier published his work, whereas de Calsteljau did not; his manuscripts surfaced only recently, many years after the original work was done.) De Casteljau's algorithm provides a very convenient method for tracing a curve, *i.e.*, for computing successive points along the curve. Curve tracing is a fundamental capability, used for many applications such as curve display and computing intersections.

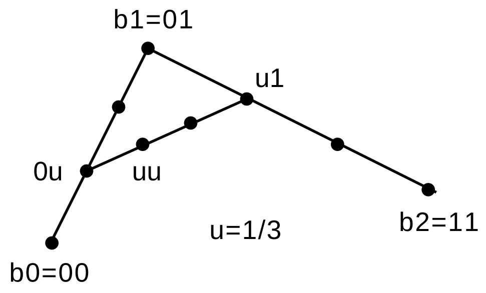

Figure 4.2.2.2 – De Casteljau's algorithm

The computation diagrammed in Figure 4.2.2.1 can be generalized to higher degrees, to parameter intervals other than [0,1], and to unequal *u* arguments. A key requirement is that the successive control points in the top row, from left to right, be such that each pair in the sequence differs in only one of the *u* arguments. Otherwise, one cannot interpolate between successive points by using Property 3 of the blossoms. For example, for degree 3, the Bézier points are

$$g(0,0,0),\ g(0,0,1),\ g(0,1,1), g(1,1,1)\ .$$

For an interval $[k_0,\ k_1]$, it can be shown that the Bézier points are obtained from the blossom by substituting 0 by k_0 and 1 by k_1 in the expressions above. For example, the Bézier points for a degree 2 polynomial are

$$g(k_0,k_0),\ g(k_0,k_1),\ g(k_1,k_1)\,.$$

The blossom computation is still valid, provided that we use the correct interpolation formulae that correspond to the situation depicted in Figure 4.2.2.3.

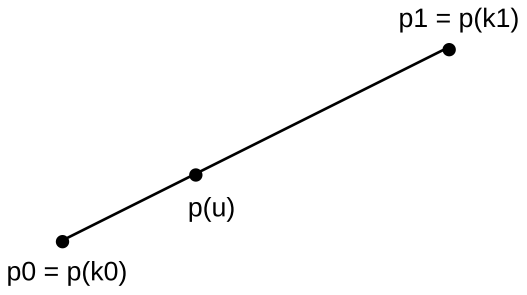

Figure 4.2.2.3 – Linear interpolation between parameters other than 0 and 1.

Instead of multiplication by (1-*u*) and *u* we now need slightly more complicated expressions:

$$\mathbf{p}(u) = \frac{k_1 - u}{k_1 - k_0}\mathbf{p}_0 + \frac{u - k_0}{k_1 - k_0}\mathbf{p}_1\,.$$

Note that the parameter expressions have changed, but the geometric construction embodied in de Casteljau's algorithm has not. This may be more obvious if we re-write the interpolation formula as

$$\mathbf{p}(u) = \mathbf{p}_0 + \frac{u - k_0}{k_1 - k_0}(\mathbf{p}_1 - \mathbf{p}_0),$$

which shows that we are adding to the initial point $\mathbf{p}_0$ a vector in the direction of the line segment, and with a magnitude that corresponds to the ratio of the distances between $\mathbf{pp}_0$ and $\mathbf{p}_1\mathbf{p}_0$. The result of the successive interpolation, since it implements de Casteljau's algorithm, is still a Bézier curve, but now parameterized in the interval $[k_0, k_1]$ instead of [0,1]. The parameter values k_0, and k_1 are usually called *knots*.

We can generalize further, and use a successive interpolation scheme to evaluate blossom points that are not in the curve, i.e., points that do not satisfy

$$u_1 = u_2 = \cdots = u_d = u.$$

All that is needed is to use the appropriate linear interpolation formulae, but with different u's. Figure 4.2.2.4 shows an important example for a quadratic curve. The control points in this example are known as the de Boor points, which will be discussed in detail in the next sections.

$$\begin{tikzcd}
g(k_0,k_1) \arrow[dr, "\frac{k_2-u_2}{k_2-k_0}"'] & & g(k_1,k_2) \arrow[dl, "\frac{u_2-k_0}{k_2-k_0}"'] \arrow[dr, "\frac{k_3-u_2}{k_3-k_1}"] & & g(k_2,k_3) \arrow[dl, "\frac{u_2-k_1}{k_3-k_1}"] \\
& g(u_2,k_1) \arrow[dr, "\frac{k_2-u_1}{k_2-k_1}"'] & & g(u_2,k_2) \arrow[dl, "\frac{u_1-k_1}{k_2-k_1}"] & \\
& & g(u_1,u_2) & &
\end{tikzcd}$$

Figure 4.2.2.4 – Evaluating a general quadratic blossom from its de Boor points

The first two points in the top row have the same k_1. We interpolate between k_0 and k_2, using a ratio that corresponds to u_2. The result is g(u_2, k_1). The second and third point in the first row have the same k_2. We interpolate between k_1 and k_3, with ratios that correspond to u_2. Finally, we interpolate the two intermediate results between k_1 and k_2, with ratios that correspond to u_1. Observe in Figure 4.2.2.4 that we still have three control points, although they are not the Bézier points, but now we have four knots instead of two as in the previous examples.

One might think that evaluating a blossom for points that do not lie on the curve serves no practical purpose, but we will see soon that this construction can be quite useful. The computation diagrammed in Figure 4.2.2.4 is a generalized de Casteljau construction, using different ratios and *u* values at each level.

4.2.3 Bézier Curves

We saw earlier that a Bézier curve of degree *d* is unambiguously defined by a control polygon with *d* + 1 vertices. A point on the curve is given by a linear combination of the control points $\mathbf{b}_i$. For example, for degree 2:

$$\mathbf{p}(u) = \begin{bmatrix} B_0^2(u) & B_1^2(u) & B_2^2(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}.$$

The Bernstein basis polynomials in the interval of interest are all positive, as shown in Figure 4.2.3.1.

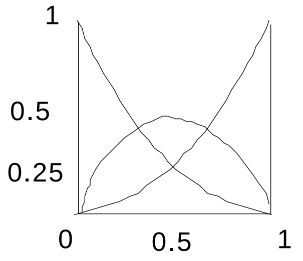

Figure 4.2.3.1 – Bernstein basis in the interval [0,1].

Furthermore, the sum of the basis functions for any value of *u* equals 1:

$$(1-u)^2 + 2u(1-u) + u^2 = 1 - 2u + u^2 + 2u - 2u^2 + u^2 = 1.$$

This has two important consequences. First, we can move the curve simply by moving its control points. To see why, consider a general rigid motion **T**, which can always be decomposed into a rotation **R** about the origin, followed by a translation by a vector δ. Applying **T** to a point $\mathbf{p}(u)$ on the curve, and using the fact that **R** is linear and that the basis functions add to 1, yields

$$\begin{aligned}
\mathbf{T}[\mathbf{p}(u)] &= \mathbf{T}[\sum_{i=0}^{2} B_i^2(u)\mathbf{b}_i] \\
&= \mathbf{R}[\sum_{i=0}^{2} B_i^2(u)\mathbf{b}_i] + \delta \\
&= \sum_{i=0}^{2} B_i^2(u)\mathbf{R}(\mathbf{b}_i) + [\sum_{i=0}^{2} B_i^2(u)]\delta \\
&= \sum_{i=0}^{2} B_i^2(u)[\mathbf{R}(\mathbf{b}_i) + \delta] \\
&= \sum_{i=0}^{2} B_i^2(u)\mathbf{T}(\mathbf{b}_i)
\end{aligned}$$

Therefore the transformed points $\mathbf{T}(\mathbf{b}_i)$ are the Bézier points for the transformed curve $\mathbf{T}[(\mathbf{p}(u)]$. The property we just derived is called *affine invariance*. Observe that an expansion in the power basis does not have this property, which is yet another reason not to use the power basis.

A second consequence is that $\mathbf{p}(u)$ is a *convex combination* of the control points, which implies that it lies in the *convex hull* of the control polygon. This is an important property of Bézier curves. It can be used to compute an *enclosure* for the curve. Enclosures are widely used in geometric modeling, as we will see later in this course. Figure 4.2.3.2 shows a cubic Bézier curve inside the convex hull of its control polygon.

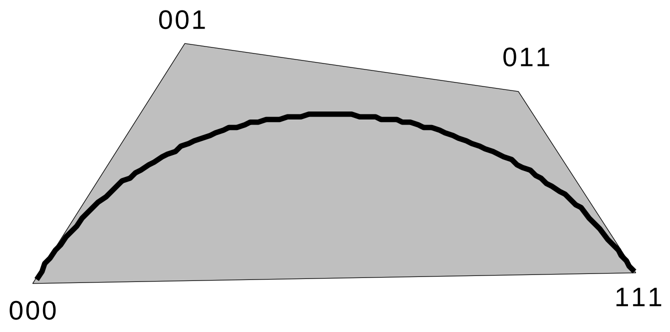

Figure 4.2.3.2 – Cubic Bézier curve and the convex hull of its control points.

By construction (see Section 4.2.1) the curve passes through the endpoints of the control polygon and is tangent to the adjoining edges. But the inner vertices of the polygon do not lie on the curve. This is evident from their blossom arguments, which are unequal. Therefore, Bézier methods represent curves through approximation, not interpolation.

The control points of a Bézier curve provide much better design handles than the coefficients in the power basis. It is relatively easy to shape a curve by moving the control points. Because curve tracing can be done at high speed, a curve design system can provide immediate graphic feedback to the user as he or she drags the control points. We shall see in the next section that B-spline curves are even better for shape design than their Bézier counterparts. The major advantages of the Bézier formulation are its numerical stability and the simplicity of the algorithms associated with it.

The blossom computation can also be used to evaluate the Bézier points for a different parameter interval in the curve. Suppose that we are given the cubic shown in Figure 4.2.3.3, with the control polygon defined by the points 000, 001, 011, and 111, as usual. We know that the Bézier points for the parameter interval [0,k] are 000, 00k, 0kk, and kkk. These points can be calculated by the blossom algorithm, and are also shown in the figure for 0 < k < 1. Now, we can similarly calculate the Bézier points for the interval [k,1], *i.e.*, the points kkk, kk1, k11 and 111. We have not changed the geometry of the curve, because we are still working with the same blossom, and a blossom is uniquely associated with a curve. But we have replaced the original control polygon by two new polygons, one for the parameter interval [0,k] and the other for the interval [k,1]. Therefore we have *subdivided* the curve into two Bézier curve segments, each of which can be manipulated separately. Note, however, that if we move the newly-found control points independently, the two segments in general will no longer belong to a single cubic curve.

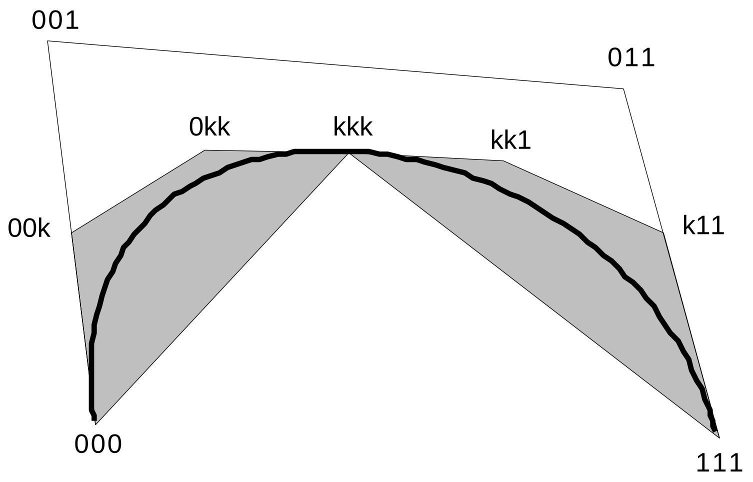

Figure 4.2.3.3 – Subdivision of a Bézier cubic curve

The blossom algorithm can also be used to find the *de Boor points* of a curve, which are of the form

$$\mathbf{d}_0 = k_0k_1, \quad \mathbf{d}_1 = k_1k_2, \quad \mathbf{d}_2 = k_2k_3$$

for a quadratic curve,

$$\mathbf{d}_0 = k_0k_1k_2, \quad \mathbf{d}_1 = k_1k_2k_3, \quad \mathbf{d}_2 = k_2k_3k_4, \quad \mathbf{d}_3 = k_3k_4k_5$$

for a cubic, and so on for higher degrees.

Given the de Boor points, the curve can also be traced by using the blossom algorithm, as we saw in the previous section. A point on the curve is a linear combination of the de Boor control points. Therefore, we have changed the basis of the polynomial vector space, from the Bernstein basis to a new one, which is called the *B-spline basis*. (The B stands for basis.) In the B-spline representation, the curve generally does not go through any of its de Boor control points. We will see in the next section, that the B-spline basis has significant advantages from the point of view of shape design.

The blossom algorithm is all we need to convert between the two basis, and to trace a curve in either representation. To convert from the Bernstein basis to the B-spline basis, we use the blossom computation to find the de Boor points starting with the Bézier points. To convert from the B-spline representation to its Bézier counterpart we evaluate the Bézier points from the de Boor points. To trace the curve we evaluate successive points on the curve by the blossom algorithm, starting either with the Bézier or the de Boor points.

4.2.4 B-Spline Curves

A single polynomial curve of degree *d* has only *d* + 1 degrees of freedom. Suppose that we want to interpolate or approximate a large number of points, to gain a finer control of curve shape. There are two options. We can use a polynomial of high degree, or a piece-wise polynomial function of low degree, which amounts to several polynomial segments joined together smoothly. High degree polynomials often have undesirable oscillations that are difficult to control, and also have poor computational behavior because of numerical errors. Piece-wise polynomial functions, called *splines*, are usually preferable. Spline curves got their name from physical splines, which consist of flexible metal or wooden strips adjusted by means of weights, and have been used for many years for designing curves.

Figure 4.2.4.1 introduces some of the nomenclature. The *u* axis is broken into intervals, called *spans*, delimited by *knots*. Within each span the spline $f(u)$ coincides with a polynomial of low degree, typically a cubic. Different spans correspond to different polynomials, and therefore also to different blossoms. At the knots certain continuity conditions are required. For degree *d*, we can obtain up to C^{d-1} continuity. Note that C^d continuity at a knot would force the two polynomials in adjacent spans to coincide, and we would no longer have a true spline. We will see below that additional knots are often introduced before and after the desired spans, and therefore the number of knots usually is not $s + 1$.

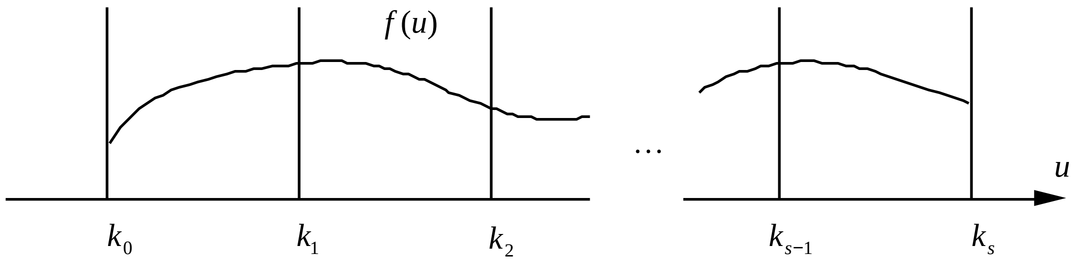

Figure 4.2.4.1 – A spline over *s* spans, and its knot sequence.

Within each span the spline is an ordinary polynomial, and these satisfy all the vector space axioms. Addition of two C^m functions, or multiplication of a C^m function by a scalar, produce another function with the same continuity properties. Therefore all the splines $f(u)$ of degree at most *d*, defined over a knot sequence $\{k_i\}$, and with given continuity conditions at each knot, constitute a vector space. What is the dimension of this space? For concreteness, consider cubic splines, assume maximum C^2 continuity at all interior knots of the sequence, and no continuity requirements at the end knots. A cubic has 4 degrees of freedom, and therefore the first span contributes 4 dimensions. The second span polynomial, however, is constrained by 3 continuity conditions (for the function and its first two derivatives) at k_1, the first knot of the second span. Therefore the second span only contributes $4 – 3 = 1$ additional dimension. The same reasoning applies to all the other remaining spans, including the last one. Hence, the total number of degrees of freedom, or the dimension of the vector space, is 4 + (s – 1). Similar arguments apply to arbitrary degrees. In general, for degree d, s spans, and maximal continuity at interior knots, the dimension of the space of splines is $d + s$.

To construct a spline one could use a Bézier polynomial for each span, and introduce constraints to ensure continuity at the knots. But this quickly becomes very complicated. A more elegant approach constructs the spline over all the spans simultaneously, and automatically guarantees continuity between spans. The key to this construction is to work with a specific basis for the spline space, called the *B-spline basis.* The basis vectors themselves have the desired continuity properties, and this ensures that all the other vectors of the space, which are linear combinations of the basis, have the same properties as well.

We already encountered the B-spline form of a single-span curve in Section 4.2.2, and we saw that its corresponding control points are the de Boor points. For a quadratic curve, with uniformly-spaced integer knots, the de Boor points are given by the blossom values 01, 12, 23. These points define the curve in the parametric interval [1,2]. What are the corresponding basis functions? They can be evaluated as usual, by the blossom algorithm, as shown in Figure 4.2.4.2.

$$\begin{tikzcd}
01 \arrow[dr, "\frac{2-u}{2-0}"'] & & 12 \arrow[dl, "\frac{u-0}{2-0}"] \arrow[dr, "\frac{3-u}{3-1}"'] & & 23 \arrow[dl, "\frac{u-1}{3-1}"] \\
& u1 \arrow[dr, "\frac{2-u}{2-1}"'] & & u2 \arrow[dl, "\frac{u-1}{2-1}"] & \\
& & uu & &
\end{tikzcd}$$

Figure 4.2.4.2 – B-Spline evaluation

The basis function that corresponds to the 01 component (control point) can be found easily from the figure by assuming that points 12 and 23 are zero and 01 is unity. The other basis vectors can be computed similarly. The results are:

$$\frac{2-u}{2-0}\frac{2-u}{2-1} = \tfrac{1}{2}(2-u)^2 = N_0^2(u)$$

$$\frac{u-0}{2-0}\frac{2-u}{2-1} + \frac{3-u}{3-1}\frac{u-1}{2-1} = \tfrac{1}{2}[u(2-u)+(3-u)(u-1)] = N_1^2(u)$$

$$\frac{u-1}{3-1}\frac{u-1}{2-1} = \tfrac{1}{2}(u-1)^2 = N_2^2(u)$$

where the N's are the B-spline basis functions, shown in Figure 4.2.4.3.

The basis function $N_1^2(u)$ is shown in its entirety in the figure, whereas the other two are shown only in the interval [1,2]. If we drew these two functions completely, we would see that all the basis functions are shifted versions of one another. (This would not be true if the knots were non-uniform.) We still have 3 control points and 3 basis functions, although they are now the de Boor points and the B-spline basis, instead of the Bézier points and the Bernstein basis. However, we now have four knots instead of the two needed for a Bézier curve. To define the curve in the interval [1,2] we had to add an extra knot before [1,2] and another after it. In general, we need to add $d - 1$ knots before and after the desired interval. This gives a total of $2d$ knots that must be considered for each given interval.

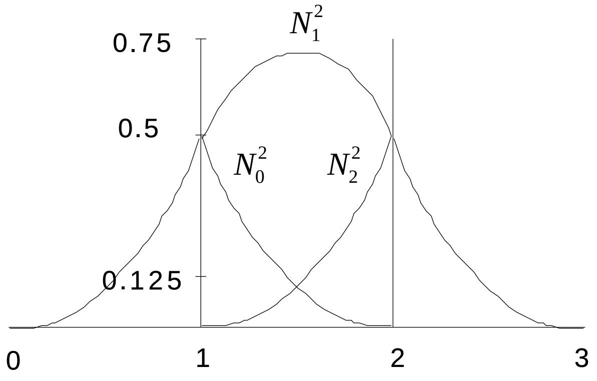

Figure 4.2.4.3 – B-Spline basis functions

The basis functions are non-negative and add to unity for any u. Therefore a B-spline curve, like its Bézier counterpart, has the *convex hull property*: it lies entirely within the convex hull of its control polygon. It also has the *affine invariance* property: the curve can be moved in space simply by moving its control points. The *support* of a basis function, *i.e.*, the region over which the function is non-zero, is precisely 3 spans. In general, a B-spline basis function of degree d has a support of $d + 1$ spans. This implies that changing the position of the control point that corresponds to a specific basis function only affects the curve over the support of the basis function, which is $d + 1$ spans. Thus, control point manipulation has a *local effec*t. This is very convenient for curve design, since it allows local tuning of the curve’s shape without causing side effects in other regions.

B-Splines are especially interesting when there are several spans. We use the de Boor points for successive spans, and share some of them across adjacent spans. For example, a quadratic spline with 6 control points covers the 4 spans shown in Figure 4.2.4.4. Each triple of de Boor points, at the top of the figure, contributes to the 1-span interval shown.

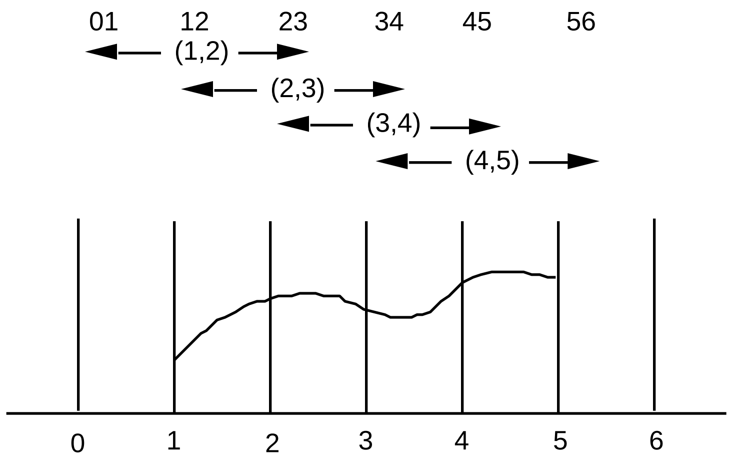

Figure 4.2.4.4 – A B-spline over the four- span interval [1,5].

The vector space of quadratic splines that corresponds to this 4-span example has dimension $s + d = 6$. Therefore we should have 6 basis functions and 6 control points. The basis functions are translated versions of the bell-shaped curve shown in Figure 4.2.4.3. Each has a support of $d + 1 = 3$ spans. Note that we had to add $d - 1 = 1$ knots at the beginning and at the end of the desired 4-span interval [1,5]. This gives a total of

$$(d-1)+(s+1)+(d-1) = s+2d-1$$

knots for the general case, or 7 for our example. If the knots are a sequence of integers starting with 0, for s spans and degree d, the sequence is

$$\underbrace{0 \quad \cdots \quad d-2}_{\text{Initial } d-1 \text{Knots}} \quad \underbrace{d-1 \quad \cdots \quad s+d-1}_{\text{Desired } s \text{ Spans}} \quad \underbrace{s+d \quad \cdots \quad s+2d-2}_{\text{Final } d-1 \text{ Knots}} .$$

To evaluate the spline we use the blossom algorithm with the de Boor points that correspond to each desired span. For example the points 12, 23, 34 correspond to the interval [2,3]. In general, for a spline of degree d and s spans, with knot sequence (0, 1, 2, …, $s + 2d - 2$), the $d + 1$ de Boor points

$$i(i+1)\ldots(i+d-1), \quad (i+1)(i+2)\ldots(i+d), \quad \ldots \quad (i+d)(i+d+1)\ldots(i+2d-1)$$

correspond to the span $[i + d - 1, i + d]$, where $0 \le i \le s - 1$. (Recall that each blossom for degree d has d arguments, which here are knot values.) There are $2d$ knots that affect this calculation:

$$i, i+1, \ldots, i+2d-1 .$$

It is convenient to index the de Boor points $\mathbf{d}$ by the value of the leading knot in the corresponding blossom. The sequence of de Boor points above is then simply

$$\mathbf{d}_i, \mathbf{d}_{i+1}, \ldots, \mathbf{d}_{i+d},$$

with $0 \le i \le s - 1$.

Each span of a spline has a corresponding polynomial and a blossom associated with it. Therefore, we can also represent the polynomial within a span by its Bézier points, and these can be computed by the blossom algorithm from their de Boor counterparts. Doing this for all the spans of a B-spline, we convert it into a sequence of Bézier curves, each represented by its Bézier points. The endpoints of the Bézier control polygons are shared between adjacent spans. This conversion is very useful, because many of the algorithms for curve evaluation and manipulation are significantly faster for Bézier curves than for B-splines. Some curve and surface modelers provide B-splines for curve and surface design, because they automatically ensure the desired continuity across spans, but convert them internally into the Bézier form for computational purposes.

Our discussion of B-splines can be generalized in several directions. First, we have tacitly assumed that all the knots are distinct. If there are coincidences, the previous results apply, with minor modifications. One can show that two coincident knots correspond to the loss of one order of continuity. For an intuitive feeling of why this happens, condider the span in Figure 4.2.4.5, and imagine that the width tends to zero, which implies that the two knots coincide in the limit. The spline originally is continuous over the span, but as the width approaches zero, the function has to vary more and more rapidly, and eventually jumps between the two values in the adjacent spans. We will return to the topic of multiple knots at the end of this section.

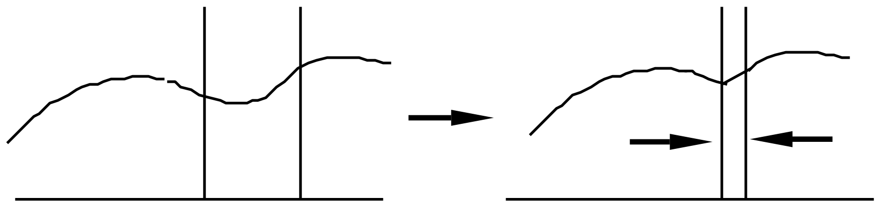

Figure 4.2.4.5 – When knots coincide, a spline looses continuity.

As a second generalization, assume that the knots are not the integer values 0, 1, ..., but rather take arbitrary values in a non-decreasing sequence

$$k_0, k_1, \ldots, k_{s+2d-1}.$$

All of our theory and algorithms are still valid, provided that we replace

$$\begin{aligned}
&0 \to k_0\\
&1 \to k_1\\
&\vdots\\
&s+2d-1 \to k_{s+2d-1}
\end{aligned}$$

In general the knots are not evenly spaced and the resulting spline is called *non-uniform*.

Finally, one can work in homogeneous coordinates, with each coordinate being a non-uniform B-spline. Since normalization involves division by the *w* coordinate, the resulting curves are no longer polynomial in Euclidean space. Each coordinate is a ratio of polynomials, and therefore it is a *rational* function. The resulting splines are called *Non-Uniform Rational B-Splines* (*NURBS*), and are widely used for modeling free-form or sculptured objects. NURBS curves can represent conics such as circles and hyperbolas exactly, and can be manipulated in projective space, which is sometimes advantageous. (Note that parametric quadratic polynomials are parabolas.)

Let us return now to the multiple knot case. The de Boor points for a span of a quadratic B-spline that corresponds to a parametric interval [*b*, *c*] must be of the form

ab *bc* *cd*

Four knots, *a*, *b*, *c*, *d*, are involved, but they need not be all distinct. For example, if *a* = *b* we have a double knot at the beginning of the sequence. The blossom values of the de Boor points become

aa *ac* *cd*

and the parametric interval becomes [*a*, *c*].

If we let the last two knots coincide as well, we obtain the control points

aa *ad* *dd*

and the corresponding interval [*a*, *d*]. But these control points are actually the Bézier points for the same blossom (and hence curve). Therefore, a quadratic Bézier curve is simply a special case of a single-span B-spline with double knots at the beginning and end of the parametric interval. In general, a Bézier curve of degree *d* is a single-span B-spline of the same degree, with *d* multiple knots at the beginning and end of the parametric interval.

Now we will see that the blossom algorithm for curve evaluation and the subdivision algorithm amount essentially to a knot insertion procedure. Consider the standard de Boor points for a quadratic span

01 12 23

and evaluate the spline at *u* = 1.5, as shown in Figure 4.2.4.6.

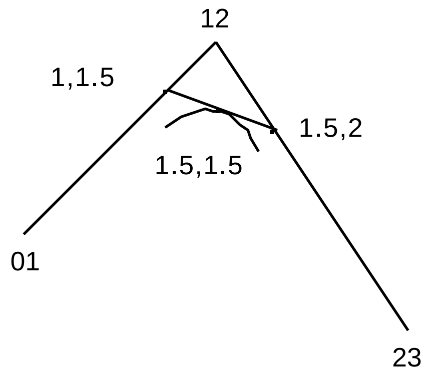

Figure 4.2.4.6 – Spline evaluation for *u* = 1.5.

Observe that the triple

(0,1) (1,1.5) (1.5,1.5)

is the set of de Boor points with knots 0, 1, 1.5, 1.5, and corresponding parametric interval [1, 1.5]. Similarly, the triple

$$(1.5,1.5) \quad (1.5,2) \quad (2,3)$$

is the set of de Boor points with knots 1.5, 1.5, 2, 3 and interval [1.5, 2]. Therefore our evaluation procedure has subdivided the original spline for the parameter interval [1,2] into two spline segments, one for the interval [0, 1.5] and the other for [1.5, 2], and has computed the de Boor points for each of the new spans. Each of the new curve segments has a double knot at the joint between the segments. Therefore we have inserted into the original knot sequence a double knot at 1.5. In general, for a spline of degree *d*, evaluation and subdivision are accomplished by the insertion of a knot of multipicity *d*.

Continuing this line of reasoning, we can think of the conversion of a quadratic B-spline to the Bézier form as knot insertion, so as to obtain double knots at the beginning and end of the desired span.

4.3 Representations for Surfaces

Both parametric and implicit surfaces are useful in geometric modeling. The most important examples of parametric surfaces are Bézier and B-spline bounded surfaces, or *patches*. In practice, implicit algebraic surfaces are typically of low degree, and the most commonly used are the quadrics, or second degree surfaces.

4.3.1 Bézier and B-Spline Patches

Both Bézier and B-spline curves of degree *d* can be written as

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{v}_i \phi_i^d(u)$$

where the $\mathbf{v}$'s are the control points (sometimes also called control *vertices*), the $\varphi(u)$ are the basis functions, and n is either d, for Bézier curves, or $s - 1$, for B-splines with s spans. This expression generalizes readily to surface patches as

$$\mathbf{p}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{v}_{ij} \psi_{ij}^d(u) .$$

Now we have two parameters u and v, and a grid of control vertices, and new basis functions that depend on both parameters. The most common patches are called *tensor-product* surfaces, and have basis functions that are the products of two of the familiar 1-dimensional Bernstein or B-spline basis functions:

$$\mathbf{p}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{v}_{ij} \phi_i^d(u) \phi_j^d(u) .$$

This surface equation can be re-written as

$$\mathbf{p}(u,v) = \sum_{i=0}^{n} \left(\sum_{j=0}^{m} \mathbf{v}_{ij}\,\phi_j^d(v) \right) \phi_i^d(u)$$
$$= \sum_{i=0}^{n} \mathbf{c}_i(v)\phi_i^d(u)$$

where

$$\mathbf{c}_i(v) = \sum_{j=0}^{m} \mathbf{v}_{ij}\phi_j^d(v).$$

These expressions provide us with a simple recipe for computing points on the surface. First fix i and v. For each i, compute the value $\mathbf{c}_i(v)$ as a point on a Bézier or B-spline curve defined by the control points $\mathbf{v}_{ij}$. Next, consider the $\mathbf{c}$'s as new control points, and compute $\mathbf{p}(u,v)$ as a point on a curve with such control points. The procedure is illustrated in Figure 4.3.1 for a biquadratic Bézier patch.

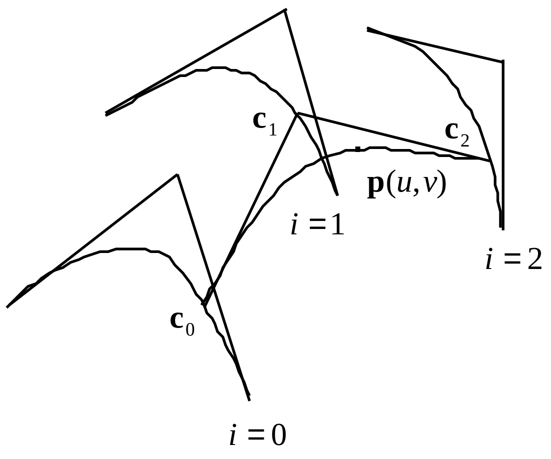

Figure 4.3.1 – Evaluating a point on a biquadratic Bézier patch.

For each fixed i = 0, 1, 2 we apply the blossom algorithm with the desired value of v to the appropriate control polygon to find the $\mathbf{c}$'s. These form a new control polygon. We apply the blossom procedure to the new control polygon with the desired u. The result is the point $\mathbf{p}(u,v)$ on the surface. Observe that we have a grid of 9 control points for the patch. The surface only interpolates the 4 corner points, and the $\mathbf{c}$ points obtained in the first phase of the computation generally do not lie on the surface.

Other computations we studied for curves, for example, subdivision, also generalize easily to surfaces.

4.3.2 Quadrics

The quadrics are the algebraic surfaces of second degree. The implicit equation for a generic quadric may be written as

$$ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k = 0.$$

Therefore the surface can be represented by an array containing the coefficients of this implicit equation. More elegantly, the coefficients can be collected in the following symmetric matrix

$$Q = \begin{bmatrix} a & d & f & g \\ d & b & e & h \\ f & e & c & j \\ g & h & j & k \end{bmatrix}.$$

If we write the generic point of 3-space in homogenous coordinates

$$X = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

the equation of a quadric in homogeneous coordinates becomes

$$X^t Q X = 0.$$

This equation is convenient for applying geometric transformations to a quadric surface, and for other practical and theoretical purposes.

The most common quadric surfaces are the sphere, the cylinder and the cone, collectively known as the *natural quadrics*. These are typically represented by primitive instancing. For example, a sphere may be represented by a point (the center) and the radius, a cylinder by an applied vector (along the axis) and the radius, and the cone by an applied vector (along the axis and anchored at the apex) and the aperture angle. Alternatively, we may define the surfaces in an agreed standard pose, and represent them by size parameters and a rigid motion that takes the surface from its standard pose to its actual pose. For a cylinder, for example, the standard pose could be such that the cylinder's axis coincides wih the *z*-axis. Then, the radius of the cylinder plus a rigid motion would suffice to represent any of the cylinder's instances. Representations that include a rigid motion are wasteful of storage, but are computationally convenient. (They were used extensively in the PADL systems developed at the University of Rochester in the 1970s and early 1980s.) Suppose, for example, that we want to intersect a line with a cylinder. This is straightforward in a frame in which the cylinder is in standard position, and we can easily transform the line to such a frame by using the rigid motion in the cylinder's representation.