Chapter 2

BÉZIER CURVES

Bézier curves are named after their inventor, Dr. Pierre Bézier. Bézier was an engineer with the Renault car company and set out in the early 1960's to develop a curve formulation which would lend itself to shape design.

Engineers may find it most understandable to think of Bézier curves in terms of the center of mass of a set of point masses. For example, consider the four masses m_0, m_1, m_2, and m_3 located at points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$. The center of mass of these four point masses is given by the equation

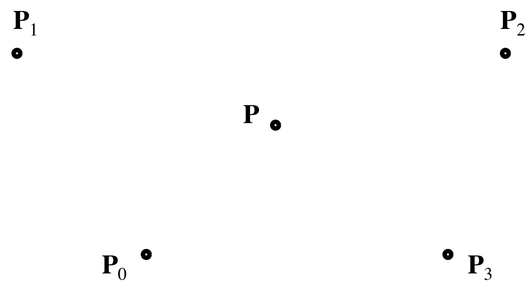

Figure 2.1: Center of mass of four points.

$$\mathbf{P} = \frac{m_0\mathbf{P}_0 + m_1\mathbf{P}_1 + m_2\mathbf{P}_2 + m_3\mathbf{P}_3}{m_0 + m_1 + m_2 + m_3}.$$

Next, imagine that instead of being fixed, constant values, each mass varies as a function of some parameter t. In specific, let $m_0 = (1-t)^3$, $m_1 = 3t(1-t)^2$, $m_2 = 3t^2(1-t)$ and $m_3 = t^3$. The values of these masses as a function of t is shown in this graph: Now, for each value of t, the masses assume different weights and their center of mass changes continuously. In fact, as t varies between 0 and 1, a curve is swept out by the center of masses. This curve is a cubic Bézier curve – *cubic* because the mass equations are cubic polynomials in t. Notice that, for any value of t, $m_0 + m_1 + m_2 + m_3 \equiv 1$, and so we can simply write the equation of this Bézier curve as $\mathbf{P} = m_0\mathbf{P}_0 + m_1\mathbf{P}_1 + m_2\mathbf{P}_2 + m_3\mathbf{P}_3$.

Note that when $t = 0$, $m_0 = 1$ and $m_1 = m_2 = m_3 = 0$. This forces the curve to pass through $\mathbf{P}_0$. Also, when $t = 1$, $m_3 = 1$ and $m_0 = m_1 = m_2 = 0$, thus the curve also passes through point $\mathbf{P}_3$. Furthermore, the curve is tangent to $\mathbf{P}_0 - \mathbf{P}_1$ and $\mathbf{P}_3 - \mathbf{P}_2$. These properties make Bézier curves

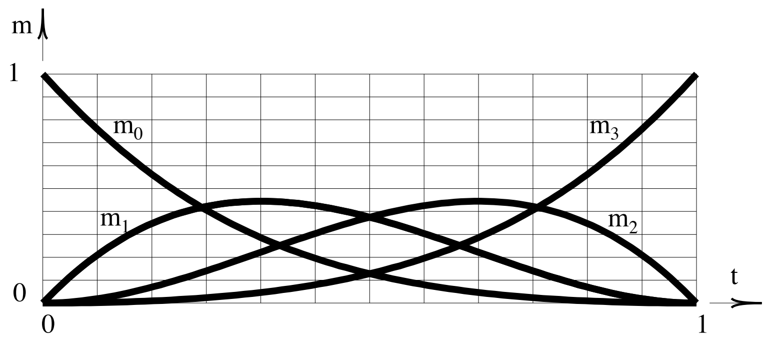

Figure 2.2: Cubic Bézier blending functions.

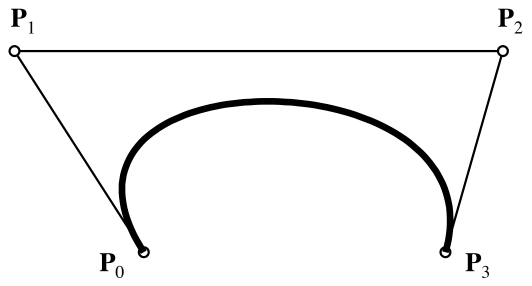

Figure 2.3: Cubic Bézier curve.

an intuitively meaningful means for describing free-form shapes. Here are some other examples of cubic Bézier curves which illustrate these properties. These variable masses m_i are normally called

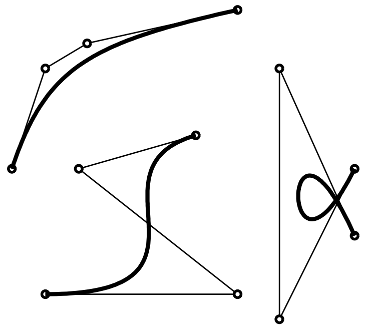

Figure 2.4: Examples of cubic Bézier curves.

blending functions and their locations $\mathbf{P}_i$ are known as *control points* or Bézier points. If we draw straight lines between adjacent control points, as in a dot to dot puzzle, the resulting figure is known as a *control polygon.* The blending functions, in the case of Bézier curves, are known as *Bernstein polynomials.* We will later look at other curves formed with different blending functions.

Bézier curves of any degree can be defined. Figure 2.5 shows sample curves of degree one through four. A degree n Bézier curve has $n+1$ control points whose blending functions are denoted $B_i^n(t)$,

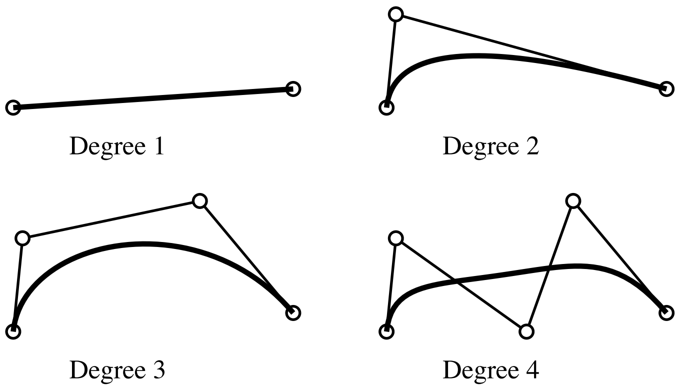

Figure 2.5: Bézier curves of various degree.

where

$$B_i^n(t) \;=\; \binom{n}{i}(1-t)^{n-i}t^i, \quad i = 0, 1, ..., n.$$

Recall that $\binom{n}{i}$ is called a *binomial coefficient*, sometimes spoken "n - *choose* - i", and is equal to $\frac{n!}{i!(n-i)!}$. In our introductory example, $n = 3$ and $m_0 = B_0^3 = (1-t)^3$, $m_1 = B_1^3 = 3t(1-t)^2$, $m_2 = B_2^3 = 3t^2(1-t)$ and $m_3 = B_3^3 = t^3$. $B_i^n(t)$ is also referred to as the *ith* Bernstein polynomial of degree n. The equation of a Bézier curve is thus:

$$\mathbf{P}(t) \;=\; \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i. \tag{2.1}$$

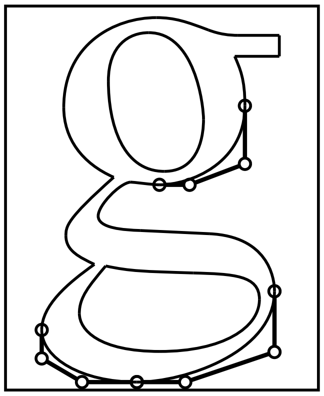

Figure 2.6: Font definition using Bézier curves.

A common use for Bézier curves is in font definition. Figure 2.6 shows the outline a letter "g" created using Bézier curves. All PostScript font outlines are defined use cubic and linear Bézier curves.

2.1 Bézier Curves over Arbitrary Parameter Intervals

Equation 2.1 gives the equation of a Bézier curve which starts at $t = 0$ and ends at $t = 1$. It is useful, especially when fitting together a string of Bézier curves, to allow an arbitrary parameter interval:

$$t_0 \le t \le t_1$$

such that $\mathbf{P}(t_0) = \mathbf{P}_0$ and $\mathbf{P}(t_1) = \mathbf{P}_n$. This can be accomplished by modifying equation 2.1:

$$\mathbf{P}(t) \;=\; \frac{\sum_{i=0}^{n} \binom{n}{i} (t_1 - t)^{n-i} (t - t_0)^i \mathbf{P}_i}{(t_1 - t_0)^n} \;= \tag{2.2}$$

$$\sum_{i=0}^{n} \binom{n}{i} \left(\frac{t_1 - t}{t_1 - t_0}\right)^{n-i} \left(\frac{t - t_0}{t_1 - t_0}\right)^i \mathbf{P_i}.$$

2.2 Subdivision of Bézier Curves

The most fundamental algorithm for dealing with Bézier curves is the subdivision algorithm. This was devised in 1959 by Paul de Casteljau (who was working for the Citroen automobile company) and is referred to as the *de Casteljau algorithm.* It is sometimes known as the *geometric construction algorithm.*

Consider a Bézier curve defined over the parameter interval $[0, 1]$. It is possible to subdivide such a curve into two new Bézier curves, one of them over the domain $0 \le t \le \tau$ and the other over

$\tau \le t \le 1$. These two new Bézier curves, considered together, are equivalent to the single original curve from which they were derived.

As illustrated in Figure 2.7, begin by adding a superscript of 0 to the original control points, then compute

$$\mathbf{P}_i^j = (1-\tau)\mathbf{P}_i^{j-1} + \tau\mathbf{P}_{i+1}^{j-1}; \quad j = 1,\ldots,n; \quad i = 0,\ldots,n-j. \tag{2.3}$$

Then, the curve over the parameter domain $0 \le t \le \tau$ is defined using control points $\mathbf{P}_0^0, \mathbf{P}_0^1, \mathbf{P}_0^2, \ldots, \mathbf{P}_0^n$ and the curve over the parameter domain $\tau \le t \le 1$ is defined using control points $\mathbf{P}_0^n, \mathbf{P}_1^{n-1}, \mathbf{P}_2^{n-2}, \ldots, \mathbf{P}_n^0$.

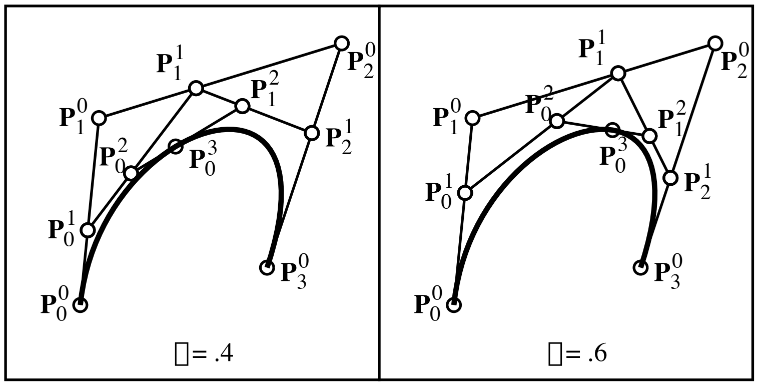

Figure 2.7: Subdividing a cubic Bézier curve.

Figure 2.8 shows that when a Bézier curve is repeatedly subdivided, the collection of control polygons converge to the curve.

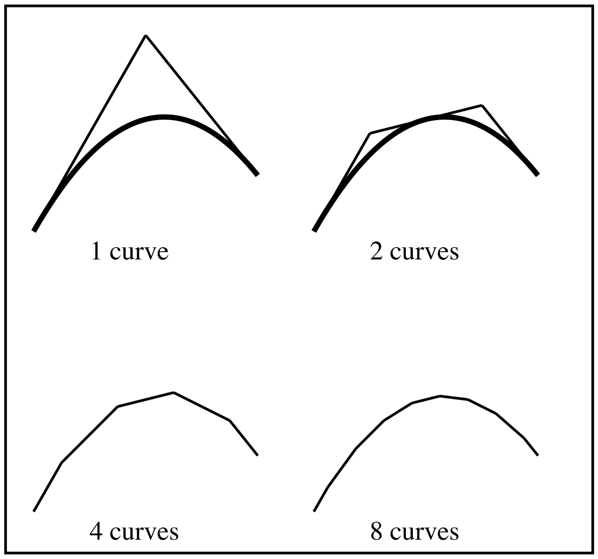

Figure 2.8: Recursively subdividing a quadratic Bézier curve.

One immediate value of the de Casteljau algorithm is that it provides a numerically stable means of computing the coordinates of any point along the curve. Its extension to curves of any degree should be obvious.

It can be shown that if a curve is repeatedly subdivided, the resulting collection of control points converges to the curve. Thus, one way of plotting a Bézier curve is to simply subdivide it an appropriate number of times and plot the control polygons.

The de Casteljau algorithm works even for parameter values outside of the original parameter interval. Fig. 2.9 shows a quadratic Bézier curve "subdivided" at $\tau = 2$. The de Castelau algorithm

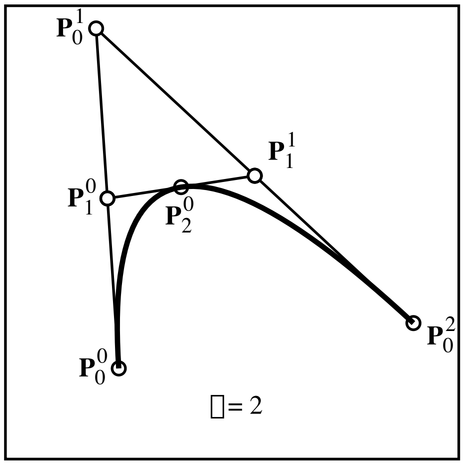

Figure 2.9: Subdividing a quadratic Bézier curve.

is numerically stable as long as the parameter subdivision parameter is within the parameter domain of the curve.

2.3 Degree Elevation

Another useful algorithm is the degree elevation algorithm. This has many uses. For example, some modeling systems use only cubic Bézier curves. However, it is important to be able to represent degree two curves exactly. A curve of degree two can be exactly represented as a Bézier curve of degree three or higher. In fact, any Bézier curve can be represented as a Bézier curve of higher degree.

We illustrate by raising the degree of a cubic Bézier curve to degree four. We denote the cubic Bézier curve in the usual way:

$$\mathbf{x}(t) = \mathbf{P}_0 B_0^3(t) + \mathbf{P}_1 B_1^3(t) + \mathbf{P}_2 B_2^3(t) + \mathbf{P}_3 B_3^3(t)$$

Note that this curve is not changed at all if we multiply it by $[t + (1 - t)] \equiv 1$. However, multiplication in this way does serve to raise the degree of the Bézier curve by one. Its effect in the cubic case is to create five control points from the original four. Those five control points thus define a degree four Bézier curve which is precisely the same as the original degree three curve. The new control points $\mathbf{P}_i^*$ are easily found as follows:

$$\mathbf{P}_0^* = \mathbf{P}_0$$

$$\mathbf{P}_1^* = \frac{1}{4}\mathbf{P}_0 + \frac{3}{4}\mathbf{P}_1$$

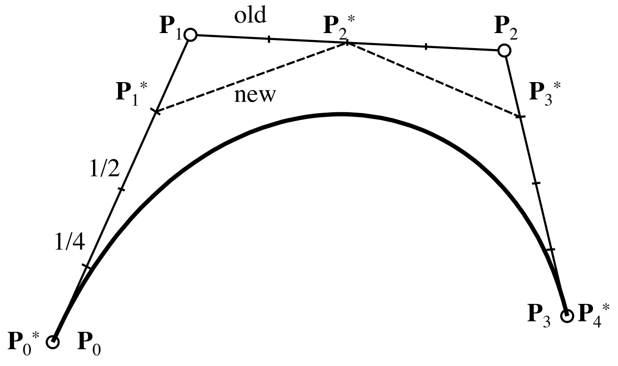

Figure 2.10: Degree elevation.

$$\mathbf{P}_2^* = \frac{2}{4}\mathbf{P}_1 + \frac{2}{4}\mathbf{P}_2$$

$$\mathbf{P}_3^* = \frac{3}{4}\mathbf{P}_2 + \frac{1}{4}\mathbf{P}_3$$

$$\mathbf{P}_4^* = \mathbf{P}_3$$

This works for any degree n as follows:

$$\mathbf{P}_i^* = \alpha_i \mathbf{P}_{i-1} + (1-\alpha_i)\mathbf{P}_i, \qquad \alpha_i = \frac{i}{n+1}.$$

Furthermore, this degree elevation can be applied repeatedly to raise the degree to any level, as shown in Figure 2.11. Note that the control polygon converges to the curve itself.

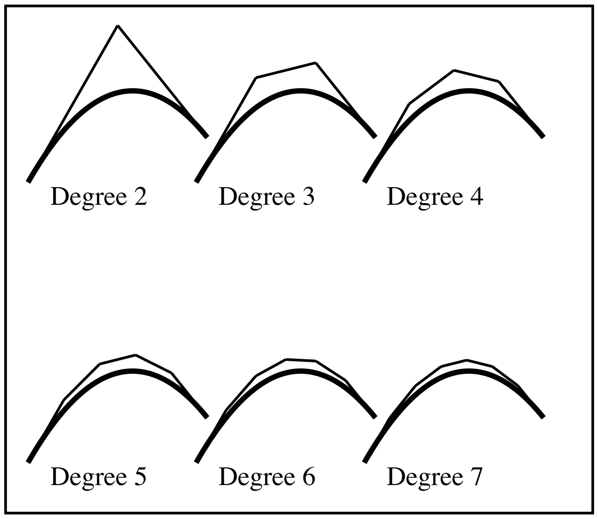

Figure 2.11: Repeated degree elevation.

2.4 Distance between Two Bézier Curves

The problem often arises of determining how closely a given Bézier curve is approximated by a second Bézier curve. For example, if a given cubic curve can be adequately represented by a degree elevated quadratic curve, it would be computationally advantageous to replace the cubic curve with the quadratic curve.

Given two Bézier curves

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{P}_i B_i^n(t); \qquad \mathbf{Q}(t) = \sum_{i=0}^{n} \mathbf{Q}_i B_i^n(t)$$

the vector $\mathbf{P}(t) - \mathbf{Q}(t)$ between points of equal parameter value on the two curves can itself be expressed as a Bézier curve

$$\mathbf{D}(t) = \mathbf{P}(t) - \mathbf{Q}(t) = \sum_{i=0}^{n} (\mathbf{P}_i - \mathbf{Q}_i) B_i^n(t)$$

whose control points are $\mathbf{D}_i = \mathbf{P}_i - \mathbf{Q}_i$. The vector from the origin to the point $\mathbf{D}(t)$ is $\mathbf{P}(t) - \mathbf{Q}(t)$. Thus, the distance between the two curves is bounded by the largest distance from the origin to any of the control points $\mathbf{D}_i$.

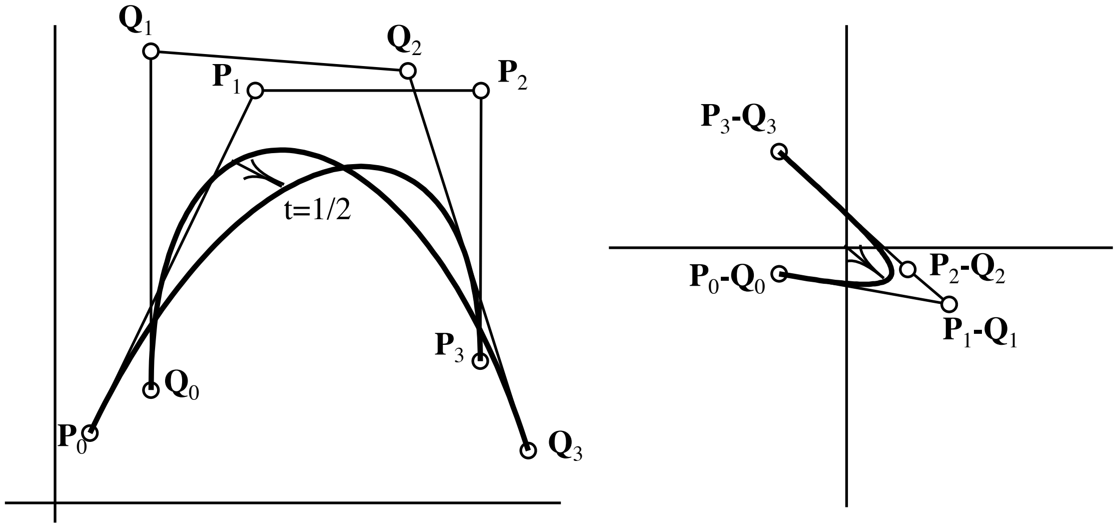

Figure 2.12: Difference curve.

2.5 Derivatives

The parametric derivatives of a Bézier curve can be determined geometrically from its control points. For a curve of degree n with control points $\mathbf{P}_i$, the first parametric derivative can be expressed as a curve of degree $n-1$ with control points $\mathbf{D}_i$ where

$$\mathbf{D}_i \; = \; n(\mathbf{P}_{i+1} \; - \; \mathbf{P}_i)$$

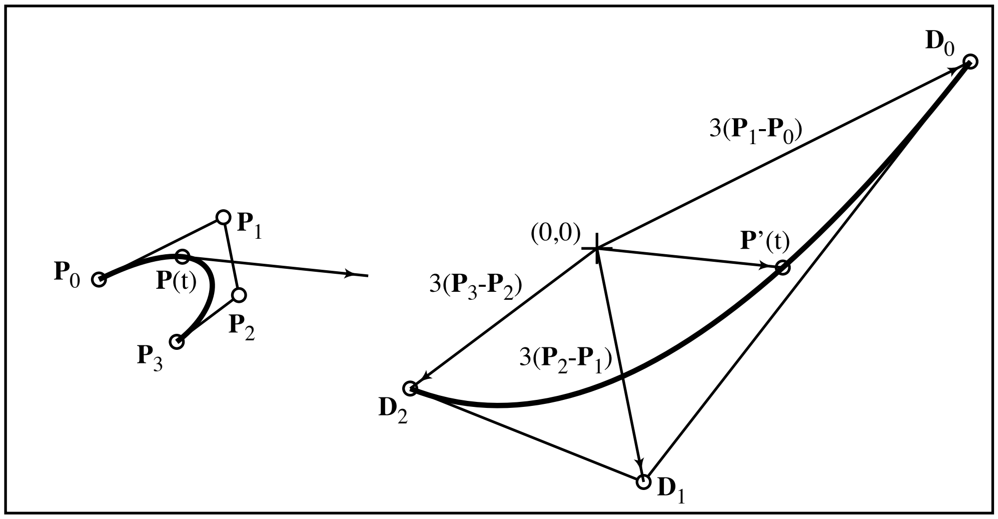

Figure 2.13: Hodograph.

For example, the cubic Bézier curve in Fig. 2.13 has a first derivative curve as shown. Of course, the first derivative of a parametric curve provides us with a tangent vector. This is illustrated in Fig. 2.13 for the point $t = .3$. This differentiation can be repeated to obtain higher derivatives as well.

The first derivative curve is known as a *hodograph*. It is interesting to note that if the hodograph passes through the origin, there is a cusp corresponding to that point on the original curve!

Note that the hodograph we have just described relates only to Bézier curves, *NOT* to rational Bézier curves or any other curve that we will study. The derivative of any other curve must be computed by differentiation.

2.6 Continuity

In describing a shape using free-form curves, it is common to use several curve segments which are joined together with some degree of continuity. Normally, a single Bézier curve will not suffice to define a complex shape. Recall the last time you used a French curve. You most likely were not able to find a single stretch along the French curve which met your needs, and were forced to segment your curve into three or four pieces which could be drawn by the French curve. These adjoining pieces probably had the same tangent lines at their common endpoints. This same piecewise construction is used in Bézier curves.

There are two types of continuity that can be imposed on two adjoining Bézier curves: *parametric* continuity and *geometric* continuity. In general, two curves which are parametric continuous to a certain degree are also geometric continuous to that same degree, but the reverse is not so.

Parametric continuity is given the notation C^i, which means *ith* degree parametric continuity. This means that the two adjacent curves have identical *ith* degree parametric derivatives, as well as all lower derivatives. Thus, C^0 means simply that the two adjacent curves share a common endpoint. C^1 means that the two curves not only share the same endpoint, but also that they have the same tangent vector at their shared endpoint, in magnitude as well as in direction. C^2 means that two curves are C^1 and in addition that they have the same second order parametric derivatives at their shared endpoint, both in magnitude and in direction.

In Fig. 2.14, the two curves are obviously at least C^0 because $\mathbf{p}_3 \equiv \mathbf{q}_0$. Furthermore, they are

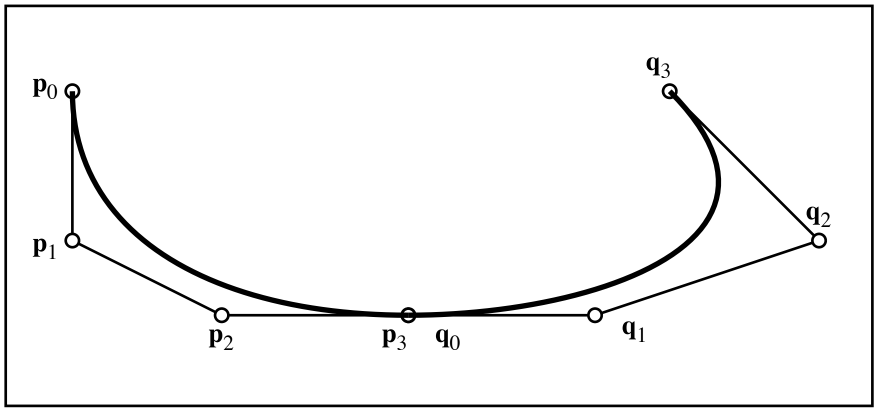

Figure 2.14: C^2 Bézier curves.

C^1 if line segments $\mathbf{p}_2 - \mathbf{p}_3$ and $\mathbf{q}_0 - \mathbf{q}_1$ are collinear and of equal length. It also appears that they are C^2, which you can verify by sketching the second derivative curves.

The conditions for geometric continuity (also known as *visual* continuity) are less strict than for parametric continuity. For G^1, we merely require that line segments $\mathbf{p}_2 - \mathbf{p}_3$ and $\mathbf{q}_0 - \mathbf{q}_1$ are collinear, but they need not be of equal length. This amounts to saying that they have a common tangent line, though the magnitude of the tangent vector may be different. G^2 (second order visual or geometric continuity) means that the two neighboring curves have the same tangent line and also the same curvature at their common boundary. The curvature of a Bézier curve at its endpoint is given by

$$\kappa = \frac{n-1}{n}\frac{h}{a^2}$$

where n is the degree of the curve and a and h are as shown in Fig. 2.15. Note that a is the length of the first leg of the control polygon, and h is the perpendicular distance from $\mathbf{P}_2$ to the first leg of the control polygon.

Two curves that are G^n can always be reparameterized so that they are C^n. This provides a practical definition of G^n continuity.

2.7 Three Dimensional Bézier Curves

It should be obvious that if the control points happen to be defined in three dimensional space, the resulting Bézier curve is also three dimensional. Such a curve is sometimes called a *space* curve, and a two dimensional curve is called a *planar* curve. Note that since a degree two Bézier curve is defined using three control points, every degree two curve is planar, even if the control points are in a three dimensional coordinate system.

2.8 Rational Bézier Curves

It is possible to change the shape of a Bézier curve by scaling the blending functions B_i^n of the control points by values which we denote w_i. These w_i are known as control point *weights*. Since it

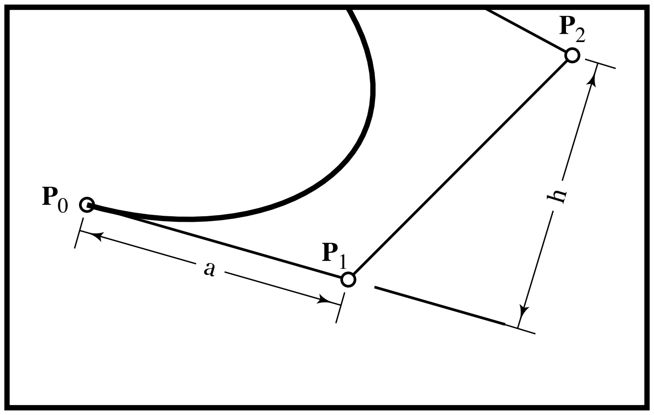

Figure 2.15: Endpoint curvature.

is essential for the blending functions to sum to one (or else the curve will change with the coordinate system), we must normalize the blending functions by dividing through by their total value. Thus, the equation becomes

$$\frac{w_0 B_0^n(t)\mathbf{P}_0 \ + \ \ldots \ + \ w_n B_n^n(t)\mathbf{P}_n}{w_0 B_0^n(t) \ + \ \ldots \ + \ w_n B_n^n(t)}.$$

The effect of changing a control point weight is illustrated in Fig. 2.16. This type of curve is known as a *rational Bézier curve*, because the blending functions are rational polynomials, or the *ratio* of two polynomials. The Bézier curves that we have dealt with up to now are sometimes known as *non-rational* or *integral* Bézier curves. There are more important reasons for using rational Bézier curves than simply the increased control it provides over the shape of the curve. For example, a perspective drawing of a 3D Bézier curve (integral or rational) is a rational Bézier curve. Also, rational Bézier curves are needed to exactly express all conic sections. A degree two integral Bézier curve can only represent a parabola. Exact representation of circles requires rational degree two Bézier curves.

A rational Bézier curve can be viewed as the projection of a 3-D curve. Fig. 2.17 shows two curves: a 3-D curve and a 2-D curve. The 2-D curve lies on the plane $z = 1$ and it is defined as the projection of the 3-D curve onto the plane $z = 1$. One way to consider this is to imagine a funny looking cone whose vertex is at the origin and which contains the 3-D curve. In other words, this cone is the collection of all lines which contain the origin and a point on the curve. Then, the 2-D rational Bézier curve is the intersection of the cone with the plane $z = 1$.

If the 2-D rational Bézier curve has control points (x_i, y_i) with corresponding weights w_i, then the (X, Y, Z) coordinates of the 3-D curve are $(x_i w_i, y_i w_i, w_i)$. Denote points on the 3-D curve using upper case variables $(X(t), Y(t), Z(t))$ and on the 2-D curve using lower case variables $(x(t), y(t))$. Then, any point on the 2-D rational Bézier curve can be computed by computing the corresponding point on the 3-D curve, $(X(t), Y(t), Z(t))$, and projecting it to the plane $z = 1$ by setting

$$x(t) = \frac{X(t)}{Z(t)}, \quad y(t) = \frac{Y(t)}{Z(t)}.$$

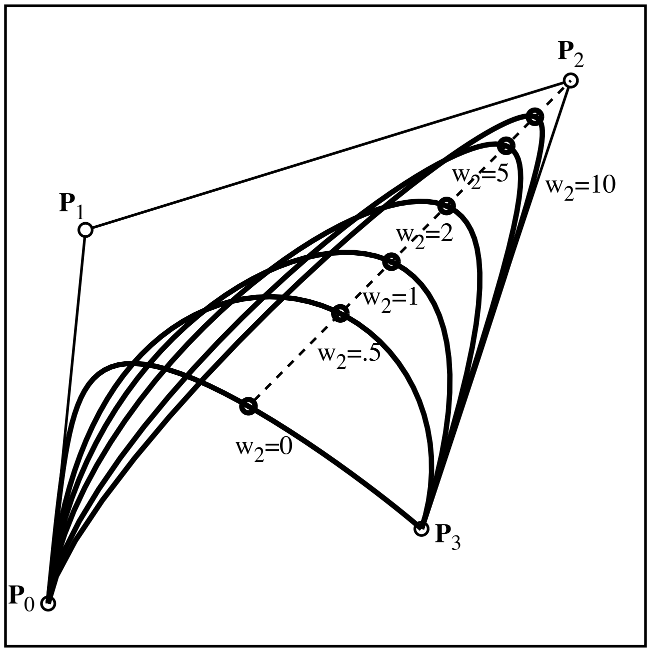

Figure 2.16: Rational Bézier curve.

2.9 Tangency and Curvature

The equations for the endpoint tangency and curvature of a rational Bézier curve must be computed using the quotient rule for derivatives — it does not work to simply compute the tangent vector and curvature for the three dimensional non-rational Bézier curve and then project that value to the (x, y) plane. For a degree n rational Bézier curve,

$$x(t) = \frac{x_n(t)}{d(t)} =$$

$$\frac{w_0 x_0 \binom{n}{0}(1-t)^n + w_1 x_1 \binom{n}{1}(1-t)^{n-1}t + w_2 x_2 \binom{n}{2}(1-t)^{n-2}t^2 + \ldots}{w_0 \binom{n}{0}(1-t)^n + w_1 \binom{n}{1}(1-t)^{n-1}t + w_2 \binom{n}{2}(1-t)^{n-2}t^2 + \ldots};$$

$$y(t) = \frac{y_n(t)}{d(t)} =$$

$$\frac{w_0 y_0 \binom{n}{0}(1-t)^n + w_1 y_1 \binom{n}{1}(1-t)^{n-1}t + w_2 y_2 \binom{n}{2}(1-t)^{n-2}t^2 + \ldots}{w_0 \binom{n}{0}(1-t)^n + w_1 \binom{n}{1}(1-t)^{n-1}t + w_2 \binom{n}{2}(1-t)^{n-2}t^2 + \ldots}$$

the equation for the tangent vector $t = 0$ must be found by evaluating the following equations:

$$\dot{x}(0) = \frac{d(0)\dot{x}_n(0) - \dot{d}(0)x_n(0)}{d^2(0)}; \qquad \dot{y}(0) = \frac{d(0)\dot{y}_n(0) - \dot{d}(0)y_n(0)}{d^2(0)}$$

from which

$$\mathbf{P}'(0) = \frac{w_1}{w_0} n(\mathbf{P}_1 - \mathbf{P}_0).$$

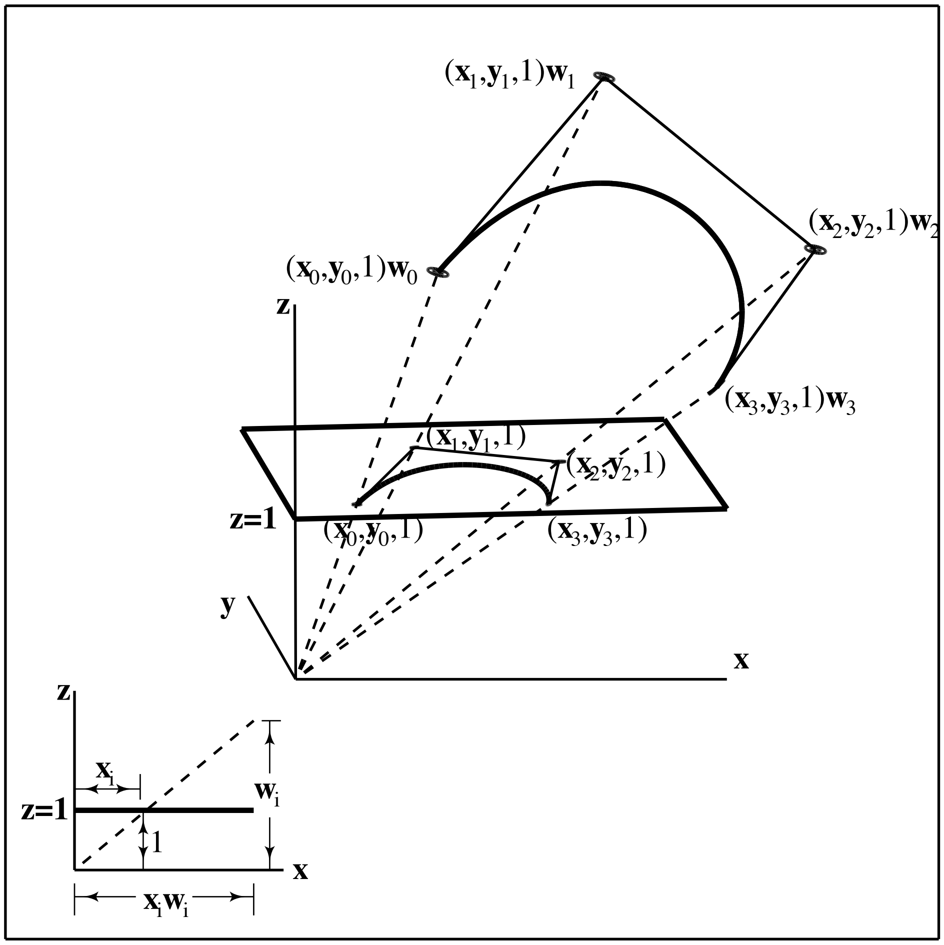

Figure 2.17: Rational curve as the projection of a 3-D curve.

and the curvature κ at $t = 0$ is found by evaluating the curvature equation

$$\kappa = \frac{|\dot{x}\ddot{y} - \dot{y}\ddot{x}|}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}}$$

from which it can be shown

$$\kappa(0) = \frac{w_0 w_2}{w_1^2} \frac{n-1}{n} \frac{h}{a^2}$$

where a and h are as shown in Figure 2.15.

2.10 Reparametrization

Any integral parametric curve $\mathbf{X}(t) = (x(t), y(t))$ can be reparametrized by the substitution $t = f(u)$. If $f(u) = a_0 + a_1 u$, then the reparametrization has the effect of changing the range over which the curve segment is defined. Thus, two Bézier subdivisions can always accomplish exactly what a linear parameter substitution does.

It is also legal for $f(u)$ to be nonlinear. This, of course, does not change the shape of the curve but it does cause the curve to be improperly parametrized, which means that to each point on the curve there corresponds more than one parameter value u. There are occasions when it is desirable to do this. If so, however, it is advisable that the curve does not end up being *multiply traced*, which means that portions of the curve get redrawn as the parameter sweeps from zero to one.

A rational parametric curve can be reparametrized with the substitution $t = f(u)/g(u)$. In this case, it is actually possible to perform a rational-linear reparametrization which does not change the endpoints of our curve segment. If we let

$$t = \frac{a(1-u) + bu}{c(1-u) + du}$$

and want $u = 0$ when $t = 0$ and $u = 1$ when $t = 1$, then $a = 0$ and $b = d$. Since we can scale then numerator and denominator without affecting the reparametrization, set $c = 1$ and we are left with

$$t = \frac{bu}{(1-u) + bu}$$

A rational Bézier curve

$$\mathbf{X}(t) = \frac{\binom{n}{0} w_0 \mathbf{P}_0 (1-t)^n + \binom{n}{1} w_1 \mathbf{P}_1 (1-t)^{n-1} t + \ldots + \binom{n}{n} w_n \mathbf{P}_n t^n}{\binom{n}{0} w_0 (1-t)^n + \binom{n}{1} w_1 (1-t)^{n-1} t + \ldots + \binom{n}{n} w_n t^n}$$

can be raparametrized without changing its endpoints by making the substitutions

$$t = \frac{bu}{(1-u) + bu}, \quad (1-t) = \frac{(1-u)}{(1-u) + bu}.$$

After multiplying numerator and denominator by $((1-u) + bu)^n$, we obtain

$$\mathbf{X}(t) = \frac{\binom{n}{0}(b^0 w_0) \mathbf{P}_0 (1-t)^n + \binom{n}{1}(b^1 w_1) \mathbf{P}_1 (1-t)^{n-1} t + \ldots + \binom{n}{n}(b^n w_n) \mathbf{P}_n t^n}{\binom{n}{0} b^0 w_0 (1-t)^n + \binom{n}{1} b^1 w_1 (1-t)^{n-1} t + \ldots + \binom{n}{n} b^n w_n t^n}$$

In other words, if we scale the weights w_i by b^i, the curve will not be changed!

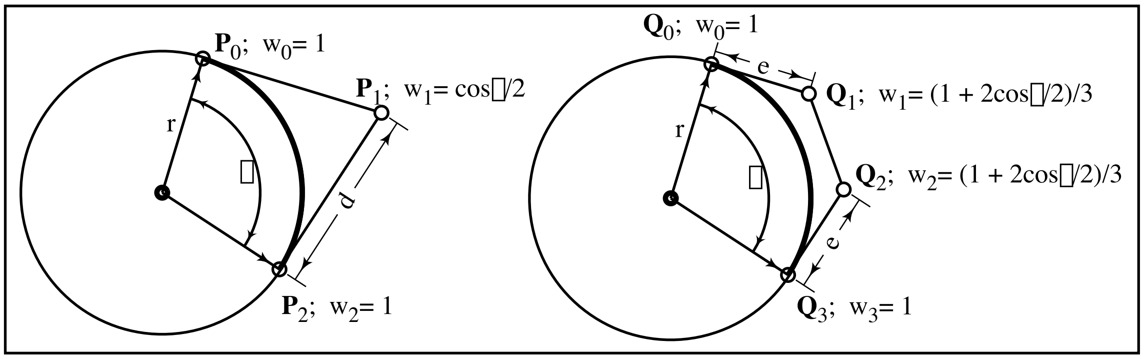

Figure 2.18: Circular arcs.

2.11 Circular Arcs

Circular arcs can be exactly represented using rational Bézier curves. Figure 2.18 shows a circular arc as both a degree two and a degree three rational Bézier curve. Of course, the control polygons are tangent to the circle. The degree three case is a simple degree elevation of the degree two case. The length e is given by

$$e = \frac{2\sin\frac{\theta}{2}}{1 + 2\cos\frac{\theta}{2}} r.$$

The degree two case has trouble when θ approaches 180° since $\mathbf{P}_1$ moves to infinity, although this can be remedied by just using homogeneous coordinates. The degree three case has the advantage that it can represent a larger arc, but even here the length e goes to infinity as θ approaches 240°. For large arcs, a simple solution is to just cut the arc in half and use two cubic Bézier curves. A complete circle can be represented as a degree five Bézier curve as shown in Figure 2.19. Here, the

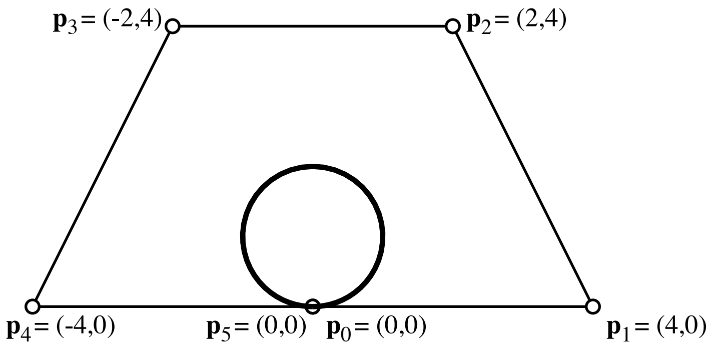

Figure 2.19: Circle as Degree 5 Rational Bézier Curve.

weights are $w_0 = w_5 = 1$ and $w_1 = w_2 = w_3 = w_4 = \frac{1}{5}$.

2.12 Explicit Bézier Curves

An explicit Bézier curve is one for which the x-coordinates of the control points are evenly spaced between 0 and 1. That is, $\mathbf{P}_i = (\frac{i}{n}, y_i),\ i = 0, \ldots, n$. Since $\sum_{i=0}^{n} \frac{i}{n} B_i^n(t) \equiv t[(1-t)+t]^n \equiv t$, such a Bézier curve takes on the important special form

$$\begin{aligned} x &= t \\ y &= f(t) \end{aligned}$$

or simply

$$y = f(x).$$

An explicit Bézier curve is sometimes called a non-parametric Bézier curve. It is just a polynomial function expressed in the Bernstein polynomial basis. Figure 2.20 shows a degree five explicit Bézier curve.

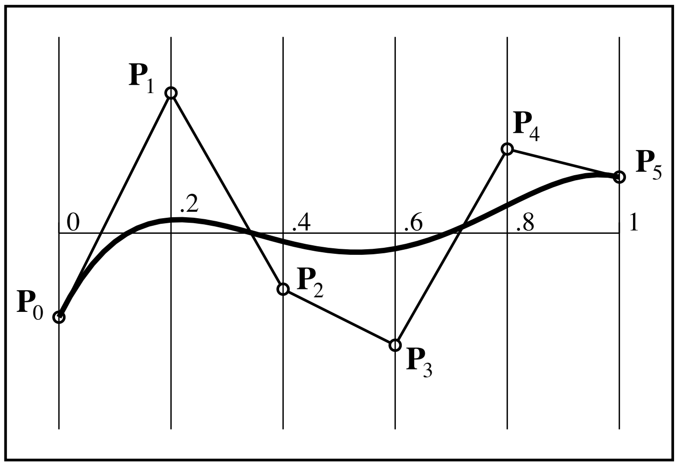

Figure 2.20: Explicit Bézier curve.

2.13 Integrating Bernstein polynomials

Recall that the hodograph (first derivative) of a Bézier curve is easily found by simply differencing adjacent control points (Section 2.5). It is equally simple to compute the integral of a Bernstein polynomial. Since the integral of a polynomial in Bernstein form

$$p(t) = \sum_{i=0}^{n} p_i B_i^n(t) \tag{2.4}$$

is that polynomial whose derivative is $p(t)$. If the desired integral is a degree $n+1$ polynomial in Bernstein form

$$q(t) = \sum_{i=0}^{n+1} q_i B_i^{n+1}(t), \tag{2.5}$$

we have

$$p_i = (n+1)(q_{i+1} - q_i). \tag{2.6}$$

Hence, $q_0 = 0$ and

$$q_i = \frac{\sum_{j=0}^{i-1} p_j}{n+1}, \quad i = 1, n+1. \tag{2.7}$$

Note that if $p(t)$ is expressed as an explicit Bézier curve, $q(t)$ can be interpreted as the area under $p(t)$ between the lines $x = 0$ and $x = t$. Thus, the entire area under an explicit Bézier curve can be computed as simply the average of the control points! This is so because

$$q(1) = q_{n+1} = \frac{\sum_{j=0}^{n} p_j}{n+1}. \tag{2.8}$$

2.14 Forward Differencing

Horner's algorithm is the fastest method for evaluating a polynomial at a single point. For a degree n polynomial, it requires n multiplies and n adds.

If a polynomial is to be evaluated at several evenly spaced values $t, t+\delta, t+2\delta, \ldots, t+k\delta$, the fastest method is to use *forward differences.*

Consider a degree 1 polynomial

$$f(t) = a_0 + a_1 t.$$

The difference between two adjacent function values is

$$\Delta_1(t) = f(t+\delta) - f(t) = [a_0 + a_1(t+\delta)] - [a_0 + a_1 t] = a_1\delta.$$

Thus, $f(t)$ can be evaluated at several evenly spaced points using the algorithm:

```
$\Delta_1 = a_1\delta$
$t_0 = 0$
$f(0) = a_0$
**for** $i = 1$ **to** $k$ **do**
$t_i = t_{i-1} + \delta$
$f(t_i) = f(t_{i-1}) + \Delta_1$
**endfor**
```

Thus, each successive evaluation requires only one add, as opposed to one add and one multiply for Horner's algorithm.

This idea extends to polynomials of any degree. For the quadratic case,

$$f(t) = a_0 + a_1 t + a_2 t^2.$$

The difference between two adjacent function values is

$$\Delta_1(t) = f(t+\delta) - f(t) = [a_0 + a_1(t+\delta) + a_2(t+\delta)^2] - [a_0 + a_1 t + a_2 t^2]$$

$$\Delta_1(t) = a_1\delta + a_2\delta^2 + 2a_2 t\delta.$$

We can now write

```
$t_0 = 0$
$f(0) = a_0$
**for** $i = 1$ **to** $k$ **do**
$t_i = t_{i-1} + \delta$
$\Delta_1(t_i) = a_1\delta + a_2\delta^2 + 2a_2t_{i-1}\delta$
$f(t_i) = f(t_{i-1}) + \Delta_1(t_{i-1})$
**endfor**
```

In this case, $\Delta(t)$ is a linear polynomial, so we can evaluate it as above, by defining

$$\Delta_2(t) = \Delta_1(t+\delta) - \Delta_1(t) = 2a_2\delta^2$$

and our algorithm now becomes

```
$t_0 = 0$
$f(0) = a_0$
$\Delta_1 = a_1\delta + a_2\delta^2$
$\Delta_2 = 2a_2\delta^2$
**for** $i = 1$ **to** $k$ **do**
$t_i = t_{i-1} + \delta$
$f(t_i) = f(t_{i-1}) + \Delta_1$
$\Delta_1 = \Delta_1 + \Delta_2$
**endfor**
```

It should be clear that for a degree n polynomial, each successive evaluation requires n adds and no multiplies! For a cubic polynomial

$$f(t) = a_0 + a_1t + a_2t^2 + a_3t^3,$$

the forward difference algorithm becomes

```
$t_0 = 0$
$f(0) = a_0$
$\Delta_1 = a_1\delta + a_2\delta^2 + a_3\delta^3$
$\Delta_2 = 2a_2\delta^2 + 6a_3\delta^3$
$\Delta_3 = 6a_3\delta^3$
**for** $i = 1$ **to** $k$ **do**
$t_i = t_{i-1} + \delta$
$f(t_i) = f(t_{i-1}) + \Delta_1$
$\Delta_1 = \Delta_1 + \Delta_2$
$\Delta_2 = \Delta_2 + \Delta_3$
**endfor**
```

Several questions remain. First, what are the initial values for the Δ_i if we want to start at some value other than $t = 0$. Second, what is a general equation for the Δ_i for a general degree n polynomial $f(t)$. Also, what if our polynomial is not in power basis.

These questions can be answered almost trivially by observing the following. Since $t_{i+1} = t_i + \delta$, we have

$$\Delta_1(t_i) = f(t_{i+1}) - f(t);$$

$$\Delta_j(t_i) = \Delta_{j-1}(t_{i+1}) - \Delta_{j-1}(t_i), \quad j = 2, \ldots, n;$$

$$\Delta_n(t_i) = \Delta_n(t_{i+1}) = \Delta_n(t_{i+k}) = \text{a constant}$$

$$\Delta_{n+1} = 0$$

Thus, our initial values for $\Delta_j(t_i)$ can be found by simply computing $f(t_i),\ f(t_{i+1}), \ldots, f(t_{i+n})$ and from them computing the initial differences. This lends itself nicely to a table. Here is the table for a degree four case:

$f(t_i)$	$f(t_{i+1})$	$f(t_{i+2})$	$f(t_{i+3})$	$f(t_{i+4})$
$\Delta_1(t_i)$	$\Delta_1(t_{i+1})$	$\Delta_1(t_{i+2})$	$\Delta_1(t_{i+3})$	
$\Delta_2(t_i)$	$\Delta_2(t_{i+1})$	$\Delta_2(t_{i+2})$		
$\Delta_3(t_i)$	$\Delta_3(t_{i+1})$			
$\Delta_4(t_i)$				
0	0	0	0	0

To compute $f(t_{i+5})$, we simply note that every number R in this table, along with its right hand neighbor R_{right} and the number directly beneath it R_{down} obey the rule

$$R_{right} = R + R_{down}.$$

Thus, we can simply fill in the values

$$\Delta_4(t_{i+1}) = \Delta_4(t_i) + 0$$

$$\Delta_3(t_{i+2}) = \Delta_3(t_{i+1}) + \Delta_4(t_{i+1})$$

$$\Delta_2(t_{i+3}) = \Delta_2(t_{i+2}) + \Delta_3(t_{i+2})$$

$$\Delta_1(t_{i+4}) = \Delta_1(t_{i+3}) + \Delta_2(t_{i+3})$$

$$f(t_{i+5}) = f(t_{i+4}) + \Delta_1(t_{i+4})$$

Note that this technique is independent of the basis in which $f(t)$ is defined. Thus, even if it is defined in Bernstein basis, all we need to do is to evaluate it $n + 1$ times to initiate the forward differencing.

For example, consider the degree 4 polynomial for which $f(t_i) = 1$, $f(t_{i+1}) = 3$, $f(t_{i+2}) = 2$, $f(t_{i+3}) = 5$, $f(t_{i+4}) = 4$. We can compute $f(t_{i+5}) = -24$, $f(t_{i+6}) = -117$, and $f(t_{i+7}) = -328$ from the following difference table:

t :	t_i	t_{i+1}	t_{i+2}	t_{i+3}	t_{i+4}	t_{i+5}	t_{i+6}	t_{i+7}
$f(t)$:	1	3	2	5	4	−24	−117	−328
$\Delta_1(t)$:	2	−1	3	−1	−28	−93	−211	
$\Delta_2(t)$:	−3	4	−4	−27	−65	−118		
$\Delta_3(t)$:	7	−8	−23	−38	−53			
$\Delta_4(t)$:	−15	−15	−15	−15	−15			
$\Delta_5(t)$:	0	0	0	0	0	0	0	0

2.15 Forward Differencing

Horner's algorithm is the fastest method for evaluating a polynomial at a single point. For a degree n polynomial, it requires n multiplies and n adds.

If a polynomial is to be evaluated at several evenly spaced values $t, t+\delta, t+2\delta, \ldots, t+k\delta$, the fastest method is to use *forward differences.*

Consider a degree 1 polynomial

$$f(t) = a_0 + a_1 t.$$

The difference between two adjacent function values is

$$\Delta_1(t) = f(t+\delta) - f(t) = [a_0 + a_1(t+\delta)] - [a_0 + a_1 t] = a_1\delta.$$

Thus, $f(t)$ can be evaluated at several evenly spaced points using the algorithm:

```
$\Delta_1 = a_1\delta$
$t_0 = 0$
$f(0) = a_0$
for $i = 1$ to $k$ do
$t_i = t_{i-1} + \delta$
$f(t_i) = f(t_{i-1}) + \Delta_1$
endfor
```

Thus, each successive evaluation requires only one add, as opposed to one add and one multiply for Horner's algorithm.

This idea extends to polynomials of any degree. For the quadratic case,

$$f(t) = a_0 + a_1 t + a_2 t^2.$$

The difference between two adjacent function values is

$$\Delta_1(t) = f(t+\delta) - f(t) = [a_0 + a_1(t+\delta) + a_2(t+\delta)^2] - [a_0 + a_1 t + a_2 t^2]$$

$$\Delta_1(t) = a_1\delta + a_2\delta^2 + 2a_2 t\delta.$$

We can now write

```
$t_0 = 0$
$f(0) = a_0$
for $i = 1$ to $k$ do
$t_i = t_{i-1} + \delta$
$\Delta_1(t_i) = a_1\delta + a_2\delta^2 + 2a_2 t_{i-1}\delta$
$f(t_i) = f(t_{i-1}) + \Delta_1(t_{i-1})$
endfor
```

In this case, $\Delta(t)$ is a linear polynomial, so we can evaluate it as above, by defining

$$\Delta_2(t) = \Delta_1(t+\delta) - \Delta_1(t) = 2a_2\delta^2$$

and our algorithm now becomes

```
$t_0 = 0$
$f(0) = a_0$
$\Delta_1 = a_1\delta + a_2\delta^2$
$\Delta_2 = 2a_2\delta^2$
for $i = 1$ to $k$ do
$t_i = t_{i-1} + \delta$
$f(t_i) = f(t_{i-1}) + \Delta_1$
$\Delta_1 = \Delta_1 + \Delta_2$
endfor
```

It should be clear that for a degree n polynomial, each successive evaluation requires n adds and no multiplies! For a cubic polynomial

$$f(t) = a_0 + a_1t + a_2t^2 + a_3t^3,$$

the forward difference algorithm becomes

```
$t_0 = 0$
$f(0) = a_0$
$\Delta_1 = a_1\delta + a_2\delta^2 + a_3\delta^3$
$\Delta_2 = 2a_2\delta^2 + 6a_3\delta^3$
$\Delta_3 = 6a_3\delta^3$
for $i = 1$ to $k$ do
$t_i = t_{i-1} + \delta$
$f(t_i) = f(t_{i-1}) + \Delta_1$
$\Delta_1 = \Delta_1 + \Delta_2$
$\Delta_2 = \Delta_2 + \Delta_3$
endfor
```

Several questions remain. First, what are the initial values for the Δ_i if we want to start at some value other than $t = 0$. Second, what is a general equation for the Δ_i for a general degree n polynomial $f(t)$. Also, what if our polynomial is not in power basis.

These questions can be answered almost trivially by observing the following. Since $t_{i+1} = t_i + \delta$, we have

$$\Delta_1(t_i) = f(t_{i+1}) - f(t);$$

$$\Delta_j(t_i) = \Delta_{j-1}(t_{i+1}) - \Delta_{j-1}(t_i), \quad j = 2, \ldots, n;$$

$$\Delta_n(t_i) = \Delta_n(t_{i+1}) = \Delta_n(t_{i+k}) = \text{a constant}$$

$$\Delta_{n+1} = 0$$

Thus, our initial values for $\Delta_j(t_i)$ can be found by simply computing $f(t_i),\ f(t_{i+1}),\ldots,f(t_{i+n})$ and from them computing the initial differences. This lends itself nicely to a table. Here is the table for a degree four case:

$f(t_i)$	$f(t_{i+1})$	$f(t_{i+2})$	$f(t_{i+3})$	$f(t_{i+4})$
$\Delta_1(t_i)$	$\Delta_1(t_{i+1})$	$\Delta_1(t_{i+2})$	$\Delta_1(t_{i+3})$	
$\Delta_2(t_i)$	$\Delta_2(t_{i+1})$	$\Delta_2(t_{i+2})$		
$\Delta_3(t_i)$	$\Delta_3(t_{i+1})$			
$\Delta_4(t_i)$				
0	0	0	0	0

To compute $f(t_{i+5})$, we simply note that every number R in this table, along with its right hand neighbor R_{right} and the number directly beneath it R_{down} obey the rule

$$R_{right} = R + R_{down}.$$

Thus, we can simply fill in the values

$$\Delta_4(t_{i+1}) = \Delta_4(t_i) + 0$$

$$\Delta_3(t_{i+2}) = \Delta_3(t_{i+1}) + \Delta_4(t_{i+1})$$

$$\Delta_2(t_{i+3}) = \Delta_2(t_{i+2}) + \Delta_3(t_{i+2})$$

$$\Delta_1(t_{i+4}) = \Delta_1(t_{i+3}) + \Delta_2(t_{i+3})$$

$$f(t_{i+5}) = f(t_{i+4}) + \Delta_1(t_{i+4})$$

Note that this technique is independent of the basis in which $f(t)$ is defined. Thus, even if it is defined in Bernstein basis, all we need to do is to evaluate it $n+1$ times to initiate the forward differencing.

For example, consider the degree 4 polynomial for which $f(t_i) = 1$, $f(t_{i+1}) = 3$, $f(t_{i+2}) = 2$, $f(t_{i+3}) = 5$, $f(t_{i+4}) = 4$. We can compute $f(t_{i+5}) = -24$, $f(t_{i+6}) = -117$, and $f(t_{i+7}) = -328$ from the following difference table:

$t:$	t_i	t_{i+1}	t_{i+2}	t_{i+3}	t_{i+4}	t_{i+5}	t_{i+6}	t_{i+7}
$f(t):$	1	3	2	5	4	-24	-117	-328
$\Delta_1(t):$	2	-1	3	-1	-28	-93	-211	
$\Delta_2(t):$	-3	4	-4	-27	-65	-118		
$\Delta_3(t):$	7	-8	-23	-38	-53			
$\Delta_4(t):$	-15	-15	-15	-15	-15			
$\Delta_5(t):$	0	0	0	0	0	0	0	0