1992-09-01

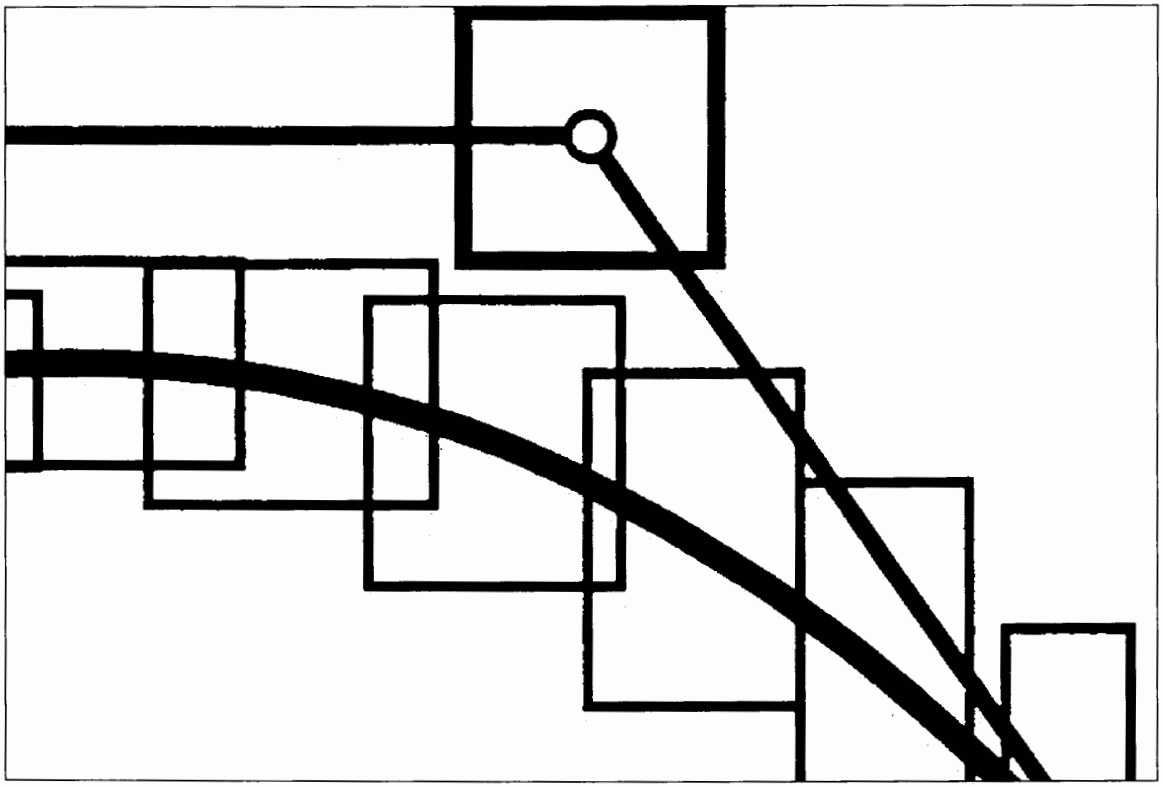
Approximation by interval Bezier curves

Thomas W. Sederberg
tom@cs.byu.edu

Rida T. Farouki

Original Publication Citation

Sederberg, T. W., and R. T. Farouki. "Approximation by Interval Bezier Curves." Computer Graphics and Applications, IEEE 12.5 (1992): 87-95.



Approximation by Interval Bezier Curves

Thomas W. Sederberg
Brigham Young University

Rida T. Farouki
IBM T.J. Watson Research Center

Unlike other curve and surface approximation schemes, interval Bezier curves can transfer a complete description of approximation errors to applications in other systems.

In recent years, there has been considerable interest in approximating the curves and surfaces that arise in CAD applications by using other curves and surfaces that are of lower degree, of simpler functional form, or require less data for their specification. The motivation for this research is the practical need to communicate product data between diverse CAD/CAM systems that impose fundamentally incompatible constraints on their canonical representation schemes. For example, some systems restrict themselves to polynomial (rather than rational) forms or limit the polynomial degrees that they accommodate.

Most approximation schemes attempt at minimum to guarantee that an approximation will satisfy a prescribed tolerance. However, none proposes to carry detailed information on the approximation error forward to subsequent applications in other systems. Such information can be of crucial importance, for example, in tolerance analyses of mechanical components. In this article, we describe a new form—the interval Bezier curve—that

0272-17-16/92/0900-0087$03.00 © 1992 IEEE

can carry such information, and we show how it readily and naturally embodies a complete description of approximation errors.

Interval arithmetic

By a scalar interval $[a, b]$ we mean a closed set of real values of the form

$$[a, b] = \{t \mid a \le t \le b\} \quad (1)$$

Given two such intervals $[a, b]$ and $[c, d]$, we define the result of a binary arithmetic operation $* \in \{+, -, \cdot, /\}$ on them as the set of all values obtained by applying $*$ to operands drawn from each interval:

$$[a, b] * [c, d] = \{x * y \mid x \in [a, b] \text{ and } y \in [c, d]\} \quad (2)$$

Specifically, we can verify that

$$\begin{aligned} [a, b] + [c, d] &= [a + c, b + d] \\ [a, b] - [c, d] &= [a - d, b - c] \\ [a, b] \cdot [c, d] &= [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)] \\ [a, b]/[c, d] &= [a, b] \cdot [1/d, 1/c] \end{aligned} \quad (3)$$

where we usually define division only for denominator intervals that do not contain 0. Occasionally, we wish to treat a single real value as a *degenerate* interval,

$$a = [a, a] \quad (4)$$

so we can apply the rules formulated in Equations 3 to mixed operands (for example, $a + [b, c] = [a + b, a + c]$). We also use a convenient shorthand notation for intervals, denoting them by a single symbol enclosed in square parentheses, for example, $[u] = [a, b]$ and $[v] = [c, d]$.

We can verify from Equations 3 that interval addition and multiplication are commutative and associative, but that multiplication does not, in general, distribute over addition. A notable exception is the multiplication of a sum of intervals by a scalar value, for which

$$\alpha([u] + [v]) = \alpha[u] + \alpha[v] \quad (5)$$

holds for arbitrary real values α and intervals $[u]$, $[v]$. (Equation 23 below gives an example of the converse case—the sum of scalar multiples of a given interval. For a more thorough discussion of these matters, see R.E. Moore's *Interval Analysis.*[1])

Interval arithmetic offers an essentially infallible (although often pessimistic) way to monitor error propagation in numerical algorithms that use floating-point arithmetic. Many familiar algorithms can be reformulated to accept interval operands.[2] Mudur and Koparkar discussed the use of interval techniques in geometric modeling calculations.[3] In this article, we are concerned primarily with the use of interval methods to account for *errors of approximation*, although in a later section we briefly discuss how to monitor *arithmetic errors* using interval Bezier curve forms.

An *interval polynomial* is a polynomial whose coefficients are intervals. We denote such polynomials in the form $[p](t)$ to distinguish them from ordinary (single-valued) polynomials. In general, we express an interval polynomial of degree n in the form

$$[p](t) = \sum_{k=0}^{n} [a_k, b_k]\, B_k^n(t) \quad (6)$$

in terms of the Bernstein polynomial basis

$$B_k^n(t) = \binom{n}{k} (1-t)^{n-k} t^k \quad \text{for } k = 0, \ldots, n \quad (7)$$

on [0, 1]. Using Equations 3, we can carry the usual rules of polynomial arithmetic over with minor modifications to the case of interval polynomials.[4] (Polynomial arithmetic in the Bernstein basis is described elsewhere.[5]) Since the basis functions (Equation 7) are all nonnegative for $t \in [0, 1]$, we can also express Equation 6 in the form

$$[p](t) = [p_{\min}(t), p_{\max}(t)] \qquad \text{for all } t \in [0, 1] \quad (8)$$

where

$$p_{\min}(t) = \sum_{k=0}^{n} a_k B_k^n(t) \quad \text{and} \quad p_{\max}(t) = \sum_{k=0}^{n} b_k B_k^n(t) \quad (9)$$

Interval Bezier curves

We define a vector-valued interval $[\mathbf{p}]$ in the most general terms as any compact set of points (x, y) in two dimensions. The *Minkowski sum* gives the addition of such sets:

$$[\mathbf{p}_1] + [\mathbf{p}_2] = \{(x_1 + x_2, y_1 + y_2) \mid (x_1, y_1) \in [\mathbf{p}_1]; (x_2, y_2) \in [\mathbf{p}_2]\} \quad (10)$$

We restrict our attention to vector-valued intervals that are just the *tensor products* of scalar intervals,

$$[\mathbf{p}] = [a, b] \times [c, d] = \{(x, y) \mid x \in [a, b] \text{ and } y \in [c, d]\} \quad (11)$$

We occasionally use the notation $([a, b], [c, d])$ instead of $[a, b] \times [c, d]$ for $[\mathbf{p}]$. Such vector-valued intervals are rectangular regions in the plane, and their addition is a trivial matter:

$$[\mathbf{p}_1] + [\mathbf{p}_2] = [a_1 + a_2, b_1 + b_2] \times [c_1 + c_2, d_1 + d_2] \quad (12)$$

where $[\mathbf{p}_1] = [a_1, b_1] \times [c_1, d_1]$ and $[\mathbf{p}_2] = [a_2, b_2] \times [c_2, d_2]$. The extension of these ideas to vector-valued intervals in spaces of higher dimension is straightforward.

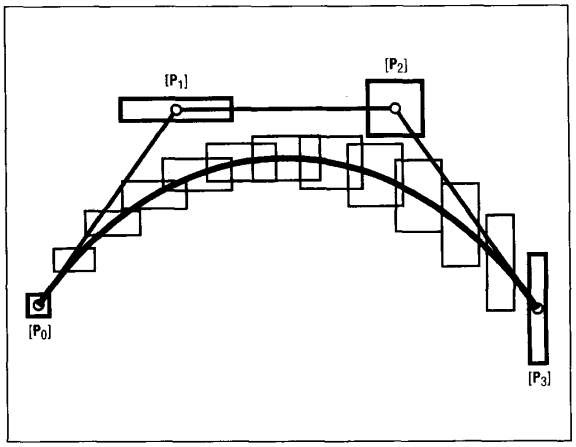


Figure 1. A cubic interval Bezier curve.

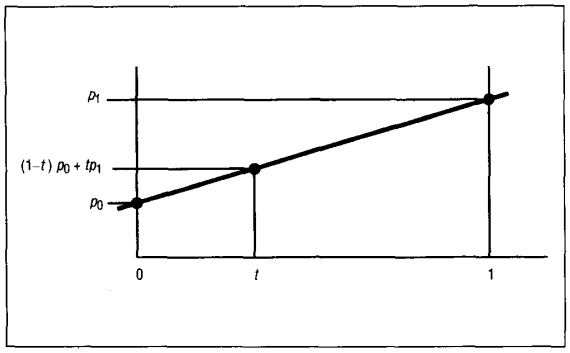


Figure 2. The affine map of two scalar points.

A Bezier curve on the parameter interval [0, 1] is defined by

$$\mathbf{P}(t) = \sum_{k=0}^{n} \mathbf{P}_k \, B_k^n(t) \tag{13}$$

where Equation 7 defines the Bernstein basis functions $B_k^n(t)$. The vector coefficients $\mathbf{P}_k = (x_k, y_k)$ in Equation 13 are called the *control points* of the curve.

An interval Bezier curve (that is, a Bezier curve with *vector-interval control points*) is written in the form

$$[\mathbf{P}](t) = \sum_{k=0}^{n} [\mathbf{P}_k] \, B_k^n(t) \tag{14}$$

where we assume that the $[\mathbf{P}_k]$ are rectangular (possibly degenerate) intervals of the form in Equation 11. Figure 1 shows a sample cubic interval Bezier curve.

For each $t \in [0, 1]$, the value $[\mathbf{P}](t)$ of the interval Bezier curve (Equation 14) is a vector interval with the following significance: For any Bezier curve $\mathbf{P}(t)$ whose control points satisfy $\mathbf{P}_k \in [\mathbf{P}_k]$ for $k = 0, \ldots, n$, we have $\mathbf{P}(t) \in [\mathbf{P}](t)$. Likewise, the entire interval Bezier curve (Equation 14) defines a region in the plane that contains all Bezier curves whose control points satisfy $\mathbf{P}_k \in [\mathbf{P}_k]$ for $k = 0, \ldots, n$.

Affine maps

A key operation in the de Casteljau subdivision and degree-elevation algorithms for Bezier curves is computing the affine map

$$\mathbf{M}(p_0, p_1, t) = (1 - t)p_0 + tp_1 \tag{15}$$

of two points p_0 and p_1. (We consider for now the case where p_0 and p_1 are simply scalar values.) Figure 2 shows how this map can be visualized. We take the values of p_0 and p_1 as the vertical coordinates, and the values of t as horizontal coordinates. At $t = 0$, $y = p_0$, while at $t = 1$, $y = p_1$. We then represent the affine map graphically by drawing a straight line through p_0 and p_1. We can regard this line as the affine map function for the two points: At any value of t, the affine map is simply the vertical coordinate of the line.

Consider the affine map of two scalar intervals:

$$\begin{aligned}[\mathbf{M}]([a, b], [c, d], t) &= (1 - t)[a, b] + t[c, d] \\ &= \{(1 - t)u + tv \mid u \in [a, b] \text{ and } v \in [c, d]\}\end{aligned} \tag{16}$$

Figure 3 shows how we can visualize this affine map. For a given value of t, we generate $[\mathbf{M}]([a, b], [c, d], t)$ by drawing a vertical line at that t value and identifying all points on that line collinear with any point in $[a, b]$ at $t = 0$ and any point in $[c, d]$ at $t = 1$. Qualitatively, we see that the width of the affine map (Equation 16) lies between the widths of $[a, b]$ and $[c, d]$ when $t \in [0, 1]$, whereas the width of the affine map grows linearly—without bound—for t outside the unit interval.

The affine map of two vector-valued intervals $[\mathbf{P}_0]$ and $[\mathbf{P}_1]$ as defined in Equation 11 is simply

$$[\mathbf{M}]([\mathbf{P}_0], [\mathbf{P}_1], t) = \{(1 - t)\mathbf{u} + t\mathbf{v} \mid \mathbf{u} \in [\mathbf{P}_0] \text{ and } \mathbf{v} \in [\mathbf{P}_1]\} \tag{17}$$

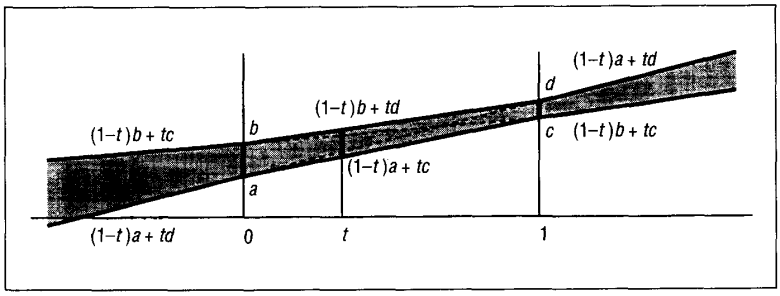


Figure 3. The affine map of two scalar intervals.

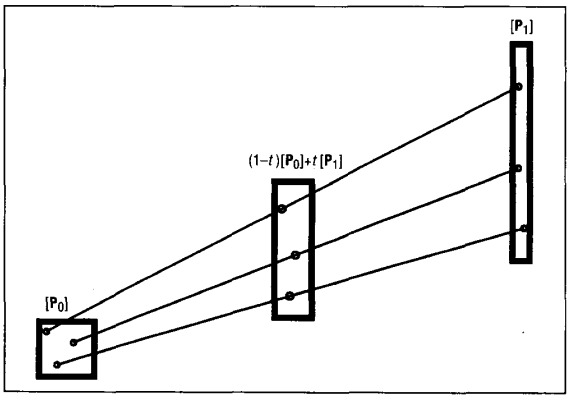


Figure 4. The affine map of two vector intervals.

as shown in Figure 4. The midpoint, width, and height of the affine map rectangle are simply affine maps of the midpoints, widths, and heights of $[\mathbf{P}_0]$ and $[\mathbf{P}_1]$.

For a Bezier curve of the form in Equation 13, we can regard the de Casteljau algorithm as a repeated application of Equation 15 to the control points $\{\mathbf{P}_k\}$. With $\mathbf{P}_k^{(0)} = \mathbf{P}_k$ for $k = 0, \ldots, n$, we set

$$\mathbf{P}_k^{(r+1)} = \mathbf{M}(\mathbf{P}_{k-1}^{(r)}, \mathbf{P}_k^{(r)}, t) \quad \text{for } k = r+1, \ldots, n \tag{18}$$

and $r = 0, \ldots, n-1$. This generates a triangular array of the quantities $\{\mathbf{P}_k^{(r)}\}$. The final entry $\mathbf{P}_n^{(n)}$ in this array corresponds to the point $\mathbf{P}(t)$ on the curve defined by Equation 13. The entries

$$\mathbf{P}_0^{(0)}, \mathbf{P}_1^{(1)}, \ldots, \mathbf{P}_n^{(n)} \quad \text{and} \quad \mathbf{P}_n^{(n)}, \mathbf{P}_n^{(n-1)}, \ldots, \mathbf{P}_n^{(0)} \tag{19}$$

on the left and right sides of the array are the control points for the subsegments of $\mathbf{P}(t)$ over the parameter intervals $[0, t]$ and $[t, 1]$, respectively. To apply Equation 18 to *interval* Bezier curves, we simply replace the quantities $\mathbf{P}_k^{(r)}$ by their interval counterparts, and invoke the appropriate rules of interval arithmetic, as illustrated in Figure 5.

Centered form

All the familiar algorithms and characteristics we associate with Bezier curves—the subdivision and degree-elevation algorithms, the variation-diminishing property, and convex-hull confinement—carry over to interval Bezier curves. We can formulate these operations more conveniently if we express the control points in *centered form*. We rewrite each control point $[\mathbf{P}_k]$ as follows:

$$\begin{aligned}[\mathbf{P}_k] &= \left([a_k, b_k], [c_k, d_k]\right) \\ &= \frac{1}{2}(a_k + b_k, c_k + d_k) + \frac{1}{2}(b_k - a_k, d_k - c_k)[i] \\ &= \overline{[\mathbf{P}_k]} + \mathbf{e}_k[i]\end{aligned} \tag{20}$$

where $[i]$ denotes the interval $[-1, +1]$. In Equation 20, the point $\overline{[\mathbf{P}_k]}$ is the *center* of the vector interval $[\mathbf{P}_k]$, while the vector $\mathbf{e}_k$ is the *error* of $[\mathbf{P}_k]$. The x and y components of $\mathbf{e}_k$ are necessarily nonnegative.

Using centered form, we can rewrite the affine map of two interval points as

$$\begin{aligned}&(1-t)[\mathbf{P}_0] + t[\mathbf{P}_1] \\ &\quad = (1-t)\{\overline{[\mathbf{P}_0]} + \mathbf{e}_0[i]\} + t\{\overline{[\mathbf{P}_1]} + \mathbf{e}_1[i]\} \\ &\quad = \{(1-t)\overline{[\mathbf{P}_0]} + t\overline{[\mathbf{P}_1]}\} + \{(1-t)\mathbf{e}_0 + t\mathbf{e}_1\}[i]\end{aligned} \tag{21}$$

where we assume that $0 \le t \le 1$. Thus, we can compute the affine map of two interval points by independently taking the affine maps of their centers and their errors.

For $0 \le t \le 1$, we can express an interval Bezier curve in centered form:

$$\begin{aligned}[\mathbf{P}](t) &= \sum_{k=0}^{n} [\mathbf{P}_k]\, B_k^n(t) \\ &= \sum_{k=0}^{n} \left\{\overline{[\mathbf{P}_k]} + \mathbf{e}_k[i]\right\} B_k^n(t) \\ &= \sum_{k=0}^{n} \overline{[\mathbf{P}_k]}\, B_k^n(t) + [i] \sum_{k=0}^{n} \mathbf{e}_k\, B_k^n(t) \\ &= \overline{[\mathbf{P}]}(t) + [i]\, \mathbf{e}(t)\end{aligned} \tag{22}$$

In bringing the interval $[i]$ outside the summation sign, we rely on the fact that, for each $k = 0, \ldots, n$, the x and y components of $\mathbf{e}_k$ are nonnegative and $B_k^n(t) \ge 0$ for $t \in [0, 1]$.

Thus, we can split the interval Bezier curve $[\mathbf{P}](t)$ over $0 \le t \le 1$ into two independent "components"—a *center curve* $\overline{[\mathbf{P}]}(t)$ and an *error curve* $\mathbf{e}(t)$. The control points $\mathbf{e}_k$ of $\mathbf{e}(t)$ all lie within the first quadrant, and consequently the error curve is itself contained within that quadrant for $t \in [0, 1]$.

Equation 21 suggests that for t outside the unit interval, the error curve should grow monotonically. Writing Equation 16 in centered form with $[a, b] = p_0 + e_0[i]$ and $[c, d] = p_1 + e_1[i]$ and noting that

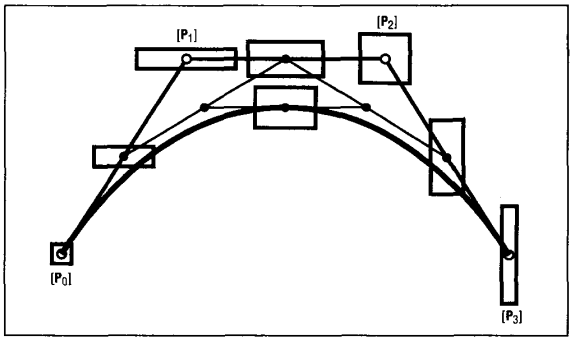


Figure 5. Interval de Casteljau algorithm.

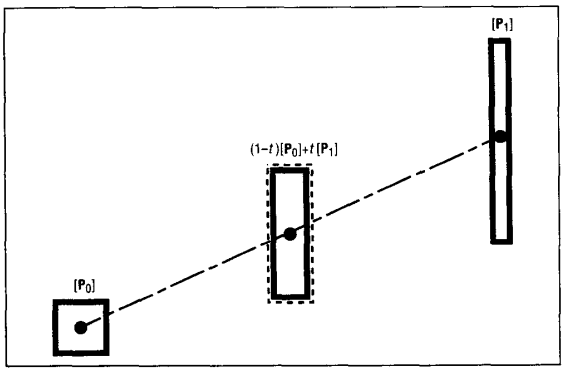


Figure 6. Affine map in floating point arithmetic.

$$\alpha[i] + \beta[i] = (|\alpha| + |\beta|)\,[i] \tag{23}$$

for arbitrary real numbers α and β, we see that the expressions

$$[\mathbf{M}](p_0 + e_0[i], p_1 + e_1[i], t) = \begin{cases} p_0(1-t) + p_1 t + \{e_0(1-t) - e_1 t\}[i] & \text{for } t < 0 \\ p_0(1-t) + p_1 t + \{e_0(1-t) + e_1 t\}[i] & \text{for } 0 \le t \le 1 \\ p_0(1-t) + p_1 t + \{e_0(t-1) + e_1 t\}[i] & \text{for } t > 1 \end{cases} \tag{24}$$

$$= p_0(1-t) + p_1 t + \{e_0\,|1-t| + e_1|t|\}[i] \tag{25}$$

describe the behavior of the affine map of two scalar intervals for all real t (see Figure 4).

The de Casteljau algorithm (Equation 18) is essentially a repeated application of the affine map defined by Equation 24. Figure 5 illustrates for the case $t = 1/2$. If we study the behavior of the de Casteljau algorithm applied to an interval Bezier curve outside the unit interval, Equation 24 leads to the following expressions:

$$[\mathbf{P}](t) = \begin{cases} \sum_{k=0}^{n} \overline{[\mathbf{P}_k]}\, B_k^n(t) + \mathbf{e}(t)[i] & \text{for } 0 \le t \le 1 \\ \sum_{k=0}^{n} \overline{[\mathbf{P}_k]}\, B_k^n(t) + \tilde{\mathbf{e}}(t)[i] & \text{for } t < 0 \text{ or } t > 1 \end{cases} \tag{26}$$

where $\mathbf{e}(t)$ is the error curve for $t \in [0, 1]$ defined in Equation 22, and

$$\tilde{\mathbf{e}}(t) = \sum_{k=0}^{n} (-1)^k\, \mathbf{e}_k\, B_k^n(t) \tag{27}$$

Interval arithmetic has a bad reputation for interval width inflation: Often the interval widths "balloon" rapidly and seriously impair the practical value of the interval technique. For example, interval operations are generally not reversible:

$$([1, 2] + [3, 4]) - [3, 4] = [0, 3] \tag{28}$$

But ballooning does not occur with the de Casteljau algorithm when we apply it within the unit parameter interval. Even after repeated applications of the de Casteljau algorithm to an interval Bezier curve, a point $[\mathbf{P}](t)$ evaluated on a subdivided region of the curve has the same size as the given point evaluated on the original curve, as long as the subdivision always occurs within the [0, 1] parameter domain. However, Equation 26 shows that serious interval inflation will occur when subdividing outside the unit interval.

Floating-point error propagation

In our discussion to this point, we have assumed that we can perform all arithmetic exactly, with no floating-point error. The error propagation in operations on Bernstein-form polynomials has been quite thoroughly discussed elsewhere.[6] We now demonstrate that the interval representation of Bezier curves provides a uniform framework within which to express both approximation and floating-point errors. Furthermore, the geometric flavor of interval Bezier curves provides some pedagogical insight into the nature of floating-point errors—a geometric setting in which to interpret the algebraic results.[6]

First we revisit the problem of computing the affine map of two vector intervals shown in Figure 4. When we use finite precision arithmetic, some uncertainty is introduced. To robustly represent that uncertainty, we widen the resulting interval an appropriate amount $\varepsilon(t)[i]$, as shown in Figure 6. In the figure, the affine map for $t = 1/2$ is shown with the exact arithmetic in solid lines and floating-point error bounds in dashed lines.

Given a floating-point mantissa of d binary digits, the *machine unit* for round off is

$$\eta = 2^{-d} \tag{29}$$

If $x * y$, where $* \in \{+, -, \times, \div\}$ denotes an exact computation and $\mathrm{fl}(x * y)$ denotes the floating-point, imprecise computation, then

$$\mathrm{fl}(x * y) \in x * y[1 - \eta, 1 + \eta] = x * y(1 + \eta[i]) \tag{30}$$

Applying this to the affine map (Equation 25) yields (see Equation 26 in Farouki and Rajan's article[6])

$$\mathrm{fl}[\mathbf{M}](p_0 + e_0[i], p_1 + e_1[i], t) \in [\mathbf{M}](p_0 + e_0[i], p_1 + e_1[i], t) + \varepsilon(t) \tag{31}$$

where

$$e(t) = \begin{cases} \left\{|p_0|(1-t) - |p_1|t + |p_0(1-t) + p_1 t|\right\} \eta[i] & \text{for } t < 0 \\ \left\{|p_0|(1-t) + |p_1|t + |p_0(1-t) + p_1 t|\right\} \eta[i] & \text{for } 0 \le t \le 1 \\ \left\{|p_0|(t-1) + |p_1|t + |p_0(1-t) + p_1 t|\right\} \eta[i] & \text{for } t > 1 \end{cases} \tag{32}$$

$$= \{|1-t||p_0| + |t||p_1| + |p_0(1-t) + p_1 t|\}\eta[i] \tag{33}$$

In words, to represent the effects of floating-point round-off in computing an affine map, we must fatten by an absolute amount $\varepsilon(t)$ the rectangle obtained by computing the affine map in exact arithmetic.

Approximation by interval polynomials

Let $f(t)$ be differentiable $n + 1$ times on the interval $[a, b]$ and let $a \le t_0 < \cdots < t_n \le b$ be an ordered sequence of $n + 1$ distinct points on that interval. The *Lagrange interpolant* to the sampled values $f_k = f(t_k)$, $k = 0, \ldots, n$ at these points is the unique polynomial of degree n given by

$$F_n(t) = \sum_{k=0}^{n} f_k L_k(t) \tag{34}$$

where the $n + 1$ linearly independent polynomials

$$L_k(t) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{t - t_j}{t_k - t_j} \quad \text{for } k = 0,\ldots,n \tag{35}$$

constitute the *Lagrange basis* for the nodes $t_0, \ldots, t_n$. This basis satisfies

$$L_k(t_j) = \delta_{jk} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise} \end{cases} \tag{36}$$

for $0 \le j, k \le n$, so $F_n(t)$ reproduces the values of $f(t)$ at each node: $F_n(t_k) = f_k$ for $k = 0, \ldots, n$.

Remainder formulas, interval approximants

At any other point on $[a, b]$, however, the values of $F_n(t)$ and $f(t)$ disagree in general. We define the error of the Lagrange interpolant by $E_n(t) = f(t) - F_n(t)$. If we have information on the behavior of the derivative $f^{(n+1)}(t)$ over $t \in [a, b]$, we can express $E_n(t)$ as the *Cauchy remainder*[7]

$$E_n(t) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{k=0}^{n} (t - t_k) \quad \text{for some } \xi \in (a,b) \tag{37}$$

For each t, it is not easy to determine the value ξ at which the right-hand side of Equation 37 gives the error $E_n(t)$ of the Lagrange interpolant. However, if we know *lower and upper bounds* $f_{\min}^{(n+1)}$ and $f_{\max}^{(n+1)}$ on $f^{(n+1)}(t)$ over $[a, b]$, then we can write

$$f(t) \in F_n(t) + \frac{\left[f_{\min}^{(n+1)}, f_{\max}^{(n+1)}\right]}{(n+1)!} \prod_{k=0}^{n} (t - t_k) \tag{38}$$

where the right-hand side is regarded as an *interval polynomial* $[F_{n+1}](t)$ *of degree* $n + 1$.

In the particular form of Equation 38, only the remainder term has a nondegenerate interval coefficient. However, if we represent this interval polynomial in another basis—for example, as

$$[F_{n+1}](t) = \sum_{k=0}^{n+1} [a_k, b_k] \binom{n+1}{k} \frac{(b-t)^{n+1-k}(t-a)^k}{(b-a)^{n+1}} \tag{39}$$

in the Bernstein basis of degree $n + 1$ on $[a, b]$, we find that in general *each* coefficient $[a_k, b_k]$ is a proper interval. The formulas giving these coefficients in terms of the nodes $t_0, \ldots, t_n$ and values $f_0, \ldots, f_n$, and the derivative bounds $f_{\min}^{(n+1)}$ and $f_{\max}^{(n+1)}$ are cumbersome to quote in full generality. We give explicit formulas below only for simpler cases of practical interest.

Hermite interpolation

Hermite interpolation involves the interpolation of the *values* and *derivatives* of $f(t)$ at specified points. We can regard it as a limiting form of Lagrange interpolation in which a sequence $m + 1$ consecutive nodes $t_k, \ldots, t_{k+m}$ merges. Then $F_n(t)$ is required to match the value $f(t_k)$ and first m derivatives $f'(t_k), f''(t_k), \ldots, f^{(m)}(t_k)$ of $f(t)$ at t_k. (In other words, $F_n(t)$ has an m-fold osculation to $f(t)$ at t_k.)

In particular, suppose $t_0, \ldots, t_r$ is a monotone sequence of $r + 1$ distinct nodes, with which we associate nonnegative integers $m_0, \ldots, m_r$ so $m_0 + \cdots + m_r + r = n$. Then the unique polynomial $F_n(t)$ satisfying the interpolation problem

$$F_n^{(i)}(t_k) = f^{(i)}(t_k) \quad \text{for } i = 0, \ldots, m_r \quad \text{and} \quad k = 0, \ldots, r \tag{40}$$

has the remainder term

$$E_n(t) = f(t) - F_n(t) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{k=0}^{r} (t - t_k)^{m_k+1} \tag{41}$$

From the above, we can write an expression analogous to Equation 38, namely

$$f(t) \in F_n(t) + \frac{\left[f_{\min}^{(n+1)}, f_{\max}^{(n+1)}\right]}{(n+1)!} \prod_{k=0}^{r} (t - t_k)^{m_k+1} \tag{42}$$

The interval-valued error term above has the same coefficient as in Equation 38, but its dependence on t is different.

An important case of the general Hermite problem is the symmetrical interpolation of values and derivatives to order $(n - 1)/2$ of a function $f(t)$ at the endpoints of the unit interval $[0, 1]$ by a polynomial $F_n(t)$ of odd degree n,

$$F_n^{(i)}(0) = f^{(i)}(0) \quad \text{and} \quad F_n^{(i)}(1) = f^{(i)}(1) \quad \text{for } i = 0, \ldots, (n-1)/2 \tag{43}$$

If we define the Hermite basis $\{H_k^n(t)\}$ of odd degree n on $[0, 1]$ by demanding that the boundary conditions

$$\left.\frac{d^j H_k^n}{dt^j}\right|_{t=0} = \left.\frac{d^j H_{n-k}^n}{dt^j}\right|_{t=1} = \delta_{jk} \tag{44}$$

be satisfied for $j, k = 0, \ldots, (n-1)/2$, we can write the solution to Equation 43 as

$$F_n(t) = \sum_{k=0}^{(n-1)/2} f^{(k)}(0) H_k^n(t) + f^{(k)}(1) H_{n-k}^n(t) \tag{45}$$

The remainder term (Equation 37) in this case becomes

$$E_n(t) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\, t^{(n+1)/2} (t-1)^{(n+1)/2} \tag{46}$$

For example, the cubic Hermite basis on [0, 1] is $H_0^3(t) = 2t^3 - 3t^2 + 1$, $H_1^3(t) = t^3 - 2t^2 + t$, $H_2^3(t) = t^3 - t^2$, $H_3^3(t) = -2t^3 + 3t^2$.

Some simple examples

We now consider some simple cases. Let $f(t)$ be twice differentiable on [0, 1], with $f_0 = f(0)$ and $f_1 = f(1)$. The linear interpolant to these endpoint values is just $F_1(t) = f_0(1-t) + f_1 t$, and we can write $f(t) = F_1(t) + 1/2\ f''(\xi)t(t-1)$ for some $\xi \in (0, 1)$. Thus, if the second derivative $f''(t)$ is confined to the interval $[f''] = [f''_{\min}, f''_{\max}]$ for all $t \in [0, 1]$, we have

$$f(t) \in F_1(t) - \frac{[f'']}{2} t(1-t) \quad \text{for all } t \in [0,1] \tag{47}$$

To formulate Equation 47 as a quadratic interval polynomial $[F_2(t)]$, we "degree elevate" $F_1(t)$ by multiplying it by $1 = (1-t) + t$ to obtain

$$f(t) \in [F_2](t) = \sum_{k=0}^{2} [F_{2,k}]\ B_k^2(t) \tag{48}$$

where

$$[F_{2,0}] = f_0,\ [F_{2,1}] = \frac{2f_0 + 2f_1 - [f'']}{4},\ [F_{2,2}] = f_1 \tag{49}$$

Thus, for a linear interpolant, the resulting quadratic interval polynomial bounds $f(t)$ on [0, 1] from below and above by the parabolas

$$\begin{aligned} F_{2,\min}(t) &= f_0 B_0^2(t) + \frac{2f_0 + 2f_1 - f''_{\max}}{4} B_1^2(t) + f_1 B_2^2(t) \\ F_{2,\max}(t) &= f_0 B_0^2(t) + \frac{2f_0 + 2f_1 - f''_{\min}}{4} B_1^2(t) + f_1 B_2^2(t) \end{aligned} \tag{50}$$

For cubic Hermite interpolation to function values f_0, f_1 and derivatives f_0', f_1' at the endpoints of [0, 1], the Bernstein-Bezier form of the interpolant is

$$F_3(t) = f_0 B_0^3(t) + \frac{3f_0 + f_0'}{3} B_1^3(t) + \frac{3f_1 - f_1'}{3} B_2^3(t) + f_1 B_3^3(t) \tag{51}$$

Thus, if the fourth derivative lies within the interval $[f^{(4)}] = [f^{(4)}_{\min}, f^{(4)}_{\max}]$ for all $t \in [0, 1]$, we can write

$$f(t) \in F_3(t) + \frac{[f^{(4)}]}{24} t^2(t-1)^2 = \sum_{k=0}^{4} [F_{4,k}]\ B_k^4(t) = [F_4](t) \tag{52}$$

where the coefficients of the quartic interval polynomial on the right are given by

$$\begin{aligned} &[F_{4,0}] = f_0,\ [F_{4,1}] = f_0 + \frac{1}{4} f_0', \\ &[F_{4,2}] = \frac{1}{2}(f_0 + f_1) + \frac{1}{6}(f_0' - f_1') + \frac{1}{144}[f^{(4)}], \\ &[F_{4,3}] = f_1 - \frac{1}{4} f_1',\ [F_{4,4}] = f_1 \end{aligned} \tag{53}$$

In symmetrical Hermite interpolation of function values and derivatives to order $(n-1)/2$ at interval endpoints, only the middle coefficient $[F_{n+1,(n+1)/2}]$ of the interpolating interval polynomial $[F_{n+1}](t)$ has nonzero width, since the residual (Equation 46) contributes only to the term involving the basis function $B^{(n+1)}_{(n+1)/2}(t)$. The width of this middle coefficient is $[f^{(n+1)}_{\max} - f^{(n+1)}_{\min}]/[(n+1)n \ldots 1/2(n+3)]^2$. The width of $[F_{n+1}](t)$ degenerates to zero at $t = 0$ and 1.

Let's look at our first example. With $f(t) = e^t$ we have $f(0) = f'(0) = 1, f(1) = f'(1) = e$, and $f^{(4)}(t) \in [1, e]$ for $t \in [0, 1]$. The interval control points (Equation 53) of the Hermite interpolant are then

$$\begin{aligned} &[F_{4,0}] = 1,\ [F_{4,1}] = \frac{5}{4},\ [F_{4,2}] = \frac{1}{144}[97 + 48e, 96 + 49e], \\ &[F_{4,3}] = \frac{3}{4}e,\ [F_{4,4}] = e \end{aligned} \tag{54}$$

We gain a better idea of the width of the middle control point from $[F_{4,2}] \approx [1.5797, 1.5916]$. Thus, the upper and lower bounds deviate from $f(t) = e^t$ by no more than about 1 percent over the entire interval [0, 1].

In our second example, with $f(t) = \sin(\pi t/2)$ we have $f(0) = 0, f'(0) = \pi/2, f(1) = 1, f'(1) = 0$, and $f^{(4)}(t) \in [0, \pi^4/16]$ for $t \in [0, 1]$. In this case, we have

$$\begin{aligned} &[F_{4,0}] = 0, [F_{4,1}] = \frac{\pi}{8}, [F_{4,2}] = \frac{1}{2{,}304}[192(6+\pi), 192(6+\pi) + \pi^4] \\ &[F_{4,3}] = 1, [F_{4,4}] = 1 \end{aligned} \tag{55}$$

and the interval-valued coefficient is somewhat wider: $[F_{4,2}] \approx [0.7618, 0.8041]$.

If the function $f(t)$ to be approximated on [0, 1] is actually a *polynomial* of degree n with Bernstein coefficients $p_0, \ldots, p_n$, we can write its rth derivative as

$$f^{(r)}(t) = \frac{n!}{(n-r)!} \sum_{k=0}^{n-r} \Delta^r p_k \, B_k^{n-r}(t) \tag{56}$$

in terms of the rth forward differences $\Delta^r p_k$. By the convex-hull property we then have the derivative bounds

$$\frac{n!}{(n-r)!} \min_k \Delta^r p_k \le f^{(r)}(t) \le \frac{n!}{(n-r)!} \max_k \Delta^r p_k \tag{57}$$

for $t \in [0, 1]$.

Approximation by interval Bezier curves

For brevity, we restrict our discussion to the approximation of plane curves; the extension to three dimensions is straightforward. Much of the preceding discussion concerning the interpolation of scalar functions applies also to the interpolation of vector-valued functions—that is, parametric curves. However, we must interpret the remainder term differently.

If $\mathbf{P}_0, \ldots, \mathbf{P}_n$ is a sequence of points corresponding to $n + 1$ distinct parameter values $t_0, \ldots, t_n$ on a plane parametric curve $\mathbf{P}(t) = \{x(t), y(t)\}$, the Lagrange interpolant $\mathbf{P}_n(t)$ to these points is simply

$$\mathbf{P}_n(t) = \sum_{k=0}^{n} \mathbf{P}_k L_k(t) \tag{58}$$

but the remainder formula is *not* obtained by merely substituting $\mathbf{P}^{(n+1)}(\xi)$ in place of $f^{(n+1)}(\xi)$ in Equation 37. Rather, we must write errors for the x and y components of $P_n(t) = \{X_n(t), Y_n(t)\}$ separately, and we then have

$$x(t) = X_n(t) + \frac{x^{(n+1)}(\xi_1)}{(n+1)!} \prod_{k=0}^{n} (t - t_k) \quad \text{and}$$
$$y(t) = Y_n(t) + \frac{y^{(n+1)}(\xi_2)}{(n+1)!} \prod_{k=0}^{n} (t - t_k) \tag{59}$$

for some $\xi_1, \xi_2 \in (a, b)$, where $\xi_1 \neq \xi_2$ in general. However, defining a vector-valued interval for $\mathbf{P}^{(n+1)}(t)$ over $t \in [a, b]$ in the form

$$[\mathbf{P}^{(n+1)}] = [x_{\min}^{(n+1)}, x_{\max}^{(n+1)}] \times [y_{\min}^{(n+1)}, y_{\max}^{(n+1)}] \tag{60}$$

lets us express the remainder term as

$$\mathbf{P}(t) \in \mathbf{P}_n(t) + \frac{[\mathbf{P}^{(n+1)}]}{(n+1)!} \prod_{k=0}^{n} (t - t_k) \tag{61}$$

In formulating interval-polynomial approximants to curves, it is natural to consider also *piecewise* interval polynomials, that is, *interval splines*. This extension is not difficult, and we give here only a brief sketch of the main ideas. We can construct a C^2 piecewise-cubic curve to interpolate any sequence of points $\mathbf{P}_0, \ldots, \mathbf{P}_n$ in the plane. Typically, this requires imposing "end conditions" and solving a tridiagonal linear system for derivatives $\mathbf{P}'_0, \ldots, \mathbf{P}'_n$ at the data points. For each span k of the spline, we then construct the cubic Hermite interpolant on $t \in [0, 1]$ to the endpoints $\mathbf{P}(0) = \mathbf{P}_{k-1}$, $\mathbf{P}(1) = \mathbf{P}_k$ and derivatives $\mathbf{P}'(0) = \mathbf{P}'_{k-1}$, $\mathbf{P}'(1) = \mathbf{P}'_k$, namely $\mathbf{P}(t) = \mathbf{P}(0)H_0^3(t) + \mathbf{P}'(0)H_1^3(t) + \mathbf{P}'(1)H_2^3(t) + \mathbf{P}(1)H_3^3(t)$. At each of the data points $\mathbf{P}_0, \ldots, \mathbf{P}_n$, the width of the interval spline shrinks to zero.

If we actually sample $\mathbf{P}_0, \ldots, \mathbf{P}_n$ from another parametric curve $r(t)$ and we know vector-valued bounds on the derivative $\mathbf{r}^{(4)}(t)$ between consecutive points, we can replace the cubic Hermite interpolants by quartic interval Bezier arcs with control points

$$[\mathbf{P}_{4,0}] = \mathbf{P}(0), \; [\mathbf{P}_{4,1}] = \mathbf{P}(0) + \frac{1}{4}\mathbf{P}'(0),$$
$$[\mathbf{P}_{4,2}] = \frac{1}{2}\big(\mathbf{P}(0) + \mathbf{P}(1)\big) + \frac{1}{6}\big(\mathbf{P}'(0) - \mathbf{P}'(1)\big) + \frac{1}{144}[\mathbf{P}]^{(4)},$$
$$[\mathbf{P}_{4,3}] = \mathbf{P}(1) - \frac{1}{4}\mathbf{P}'(1), \; [\mathbf{P}_{4,4}] = \mathbf{P}(1) \tag{62}$$

As a final example, we approximate a quarter circle, $\mathbf{P}(t) = (\cos(\pi t)/2, \sin(\pi t)/2)$, as a quartic interval Bezier curve using the coefficients in Equation 55:

$$[\mathbf{P}_{4,0}] = (1,0), \; [\mathbf{P}_{4,1}] = \left(1, \frac{\pi}{8}\right),$$
$$[\mathbf{P}_{4,2}] = \big([0.7618, 0.8041], [0.7618, 0.8041]\big),$$
$$[\mathbf{P}_{4,3}] = \left(\frac{\pi}{8}, 1\right), \; [\mathbf{P}_{4,4}] = (0,1) \tag{63}$$

This equation approximates an *arc-length* parameterization of the circle to within two significant digits (see Figure 7).

Conclusion

The ideas we present in this article are of a general nature. In some applications, better methods exist for bounding the error. For example, a method[8] for approximating rational Bezier curves with polynomial Bezier curves results in an error bound tighter than can be obtained with this more general method. Likewise, a method[9] for bounding the error of Hermite approximation of offset curves is also superior to the method we described here. ❑

Acknowledgments

This article was condensed from a more extensive IBM research report (no. RC17503). Sederberg was supported in part by NSF grant numbers DMC-8657057 and DMC-8813688 and through a grant from IBM.

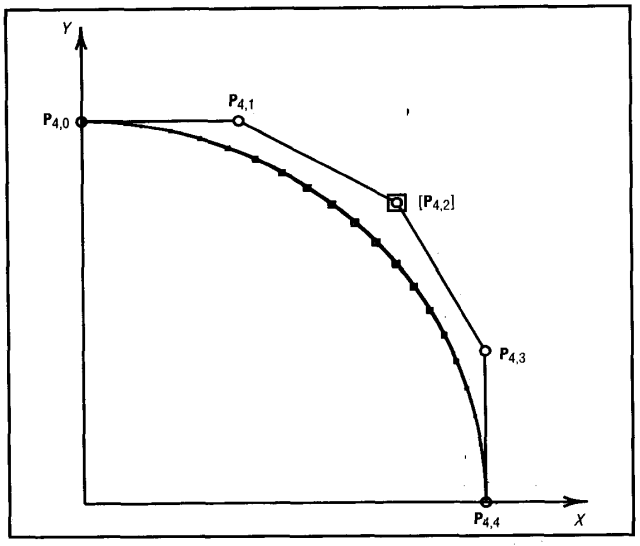


Figure 7. Approximate arc-length parameterization of a circle.

References

1. R.E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1966.
2. E. Hansen, "Interval Forms of Newton's Method," *Computing*, Vol. 20, 1978, pp. 153-163.
3. S.P. Mudur and P.A. Koparkar, "Interval Methods for Processing Geometric Objects," *IEEE CG&A*, Vol. 4, No. 2, Feb.1984, pp. 7-17.
4. J. Rokne, "Reducing the Degree of an Interval Polynomial," *Computing*, Vol. 14, 1975, pp. 5-14.
5. R.T. Farouki and V.T. Rajan, "Algorithms for Polynomials in Bernstein Form," *Computer-Aided Geometric Design*, Vol. 5, No. 1, June 1988, pp. 1-26.
6. R.T. Farouki and V.T. Rajan, "On the Numerical Condition of Polynomials in Bernstein Form," *Computer-Aided Geometric Design*, Vol. 4, No. 3, Nov. 1987, pp. 191-216.
7. P.J. Davis, *Interpolation and Approximation*, Dover, New York, 1963.
8. T.W. Sederberg and M. Kakimoto, "Approximating Rational Curves Using Polynomial Curves," in *NURBS for Curve and Surface Design*, G. Farin, ed., SIAM, Philadelphia, 1991, pp. 149-158.
9. T.W. Sederberg and D.B. Buehler, "Offsets of Bezier Curves: Hermite Approximation with Error Bound," to appear in *Mathematical Methods in CAGD II*, T. Lyche and L. Schumaker, eds., Academic Press, New York, 1992.

Thomas W. Sederberg is an associate professor of civil engineering at Brigham Young University. His research is in computer-aided geometric design and computer graphics.

Sederberg received his BS from Brigham Young University in civil engineering and his PhD from Purdue University in mechanical engineering. He is a member of ACM Siggraph and IEEE.

Rida T. Farouki is a research staff member at the IBM T.J. Watson Research Center, Yorktown Heights, N.Y. Apart from his work in computer-aided geometric design, he has published papers in various areas of computational physics, including geometrical optics, stellar dynamics, and plasma physics.

Farouki received a BA in engineering science from Oxford University and a PhD in theoretical astrophysics from Cornell University.

Sederberg can be reached at the Department of Civil Engineering, Brigham Young University, Provo, UT 84602; Farouki at IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598.