

Approximation by interval Bézier curves

Thomas W. Sederberg, Brigham Young University
Rida T. Farouki, IBM Thomas J. Watson Research Center

Abstract

When a polynomial Bézier curve of degree n is used to approximate a given parametric curve $\mathbf{P}(t)$ by interpolating $n + 1$ data values (points and/or derivative vectors) sampled from $\mathbf{P}(t)$ at discrete parameter values $t_1, t_2, \dots \in [0, 1]$, the approximation error may be expressed as a polynomial in t (dependent on the nodes t_1, t_2, \dots) multiplied by the components $x^{(n+1)}(\xi_1)$ and $y^{(n+1)}(\xi_2)$ of the $(n + 1)^{\text{th}}$ derivative of $\mathbf{P}(t)$ at indeterminate points $\xi_1, \xi_2 \in (0, 1)$. If bounds on $x^{(n+1)}(t)$ and $y^{(n+1)}(t)$ for $t \in [0, 1]$ are known, we show how this error term may be incorporated in the approximating Bézier curve by elevating its degree and interpreting its control points to be *interval-valued*. We present the basic theory of such “interval Bézier curves” and illustrate the approximation method in practice by a variety of examples.

1 Introduction

In recent years there has been considerable interest in approximating the curves and surfaces that arise in computer-aided design applications by other curves and surfaces that are of lower degree, of simpler functional form, or require less data for their specification. The motivation for this activity arises from the practical need to communicate product data between diverse CAD/CAM systems that impose fundamentally incompatible constraints on their canonical representation schemes, e.g., restricting themselves to polynomial (rather than rational) forms, or limiting the polynomial degrees that they accommodate.

While most of the approximation schemes in the references cited above attempt at minimum to guarantee that an approximation will satisfy a prescribed tolerance, once this has been achieved none of them 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 the components of a mechanism. In this paper we describe a new form — the interval Bézier curve — that is capable of carrying such information, and we show how a complete description of approximation errors may be readily embodied in such forms in a straightforward and natural manner.

2 Interval arithmetic

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

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

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

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

Specifically, it is not difficult to 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 division is usually defined only for denominator intervals that do not contain 0. Occasionally, we shall wish to treat a single real value as a *degenerate* interval,

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

so that we can apply the rules (3) to mixed operands (e.g., $a + [b, c] = [a + b, a + c]$). We shall also make use of a convenient short-hand notation for intervals, denoting them by a single symbol enclosed in square parentheses, e.g., $[u] = [a, b]$ and $[v] = [c, d]$.

It can be verified from (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 $[u]$, $[v]$. An example of the converse case — the sum of scalar multiples of a given interval — is given in equation (23) below. For a more rigorous and detailed discussion of these matters, see [5].

Interval arithmetic offers an essentially infallible (although often pessimistic) means of monitoring error propagation in numerical algorithms that employ floating point arithmetic. Many familiar algorithms can be re-formulated to accept interval operands (e.g., [4]). The use of interval techniques in the context of geometric modeling calculations has been discussed in [6]; in this paper we shall be concerned primarily with the use of interval methods to take account of *errors of approximation*, although in section 3.3 we present a brief discussion of how to monitor *arithmetic errors* using interval Bézier form.

An *interval polynomial* is a polynomial whose coefficients are intervals. We shall 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

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

on $[0, 1]$. Using (3), the usual rules of polynomial arithmetic can be carried over with minor modifications to the case of interval polynomials (see [7]; polynomial arithmetic in the Bernstein basis is described in [3]). Since the basis functions (7) are all non-negative for $t \in [0, 1]$, we can also express (6) in the form

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

where

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

3 Interval Bézier curves

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

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

It is prudent to 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.

Let us recall now that a Bézier curve on the parameter interval $[0, 1]$ is defined by

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

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

An interval Bézier curve (i.e., a Bézier curve with *vector-interval control points*) is written in the form

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

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

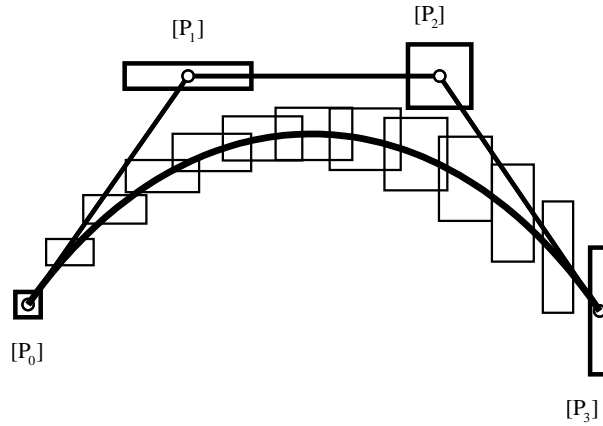


Figure 1: A cubic interval Bézier curve.

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

3.1 Affine maps

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

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

of two points p_0 and p_1 (we consider for now the case where p_0 and p_1 are simply scalar values). This map can be visualized as shown in Figure 2, where the values of p_0 and p_1 are taken as the vertical coordinates, and the values of t as horizontal coordinates. At $t = 0$, $y = p_0$ while at $t = 1$, $y = p_1$. The affine map is then represented graphically by drawing a straight line through p_0 and p_1 . This line can be regarded 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:

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

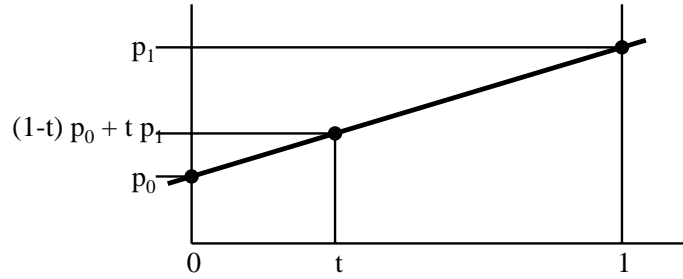


Figure 2: The affine map of two scalar points.

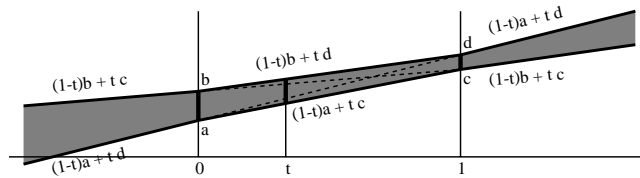


Figure 3: The affine map of two scalar intervals.

The affine map (16) can be visualized as shown in Figure 3 — for a given value of t , $[\mathbf{M}]([a, b], [c, d], t)$ is generated by drawing a vertical line at that t value and identifying all points on that line which are 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 (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

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

as shown in figure 4. Observe that the midpoint, width, and height of the affine map rectangle are simply affine

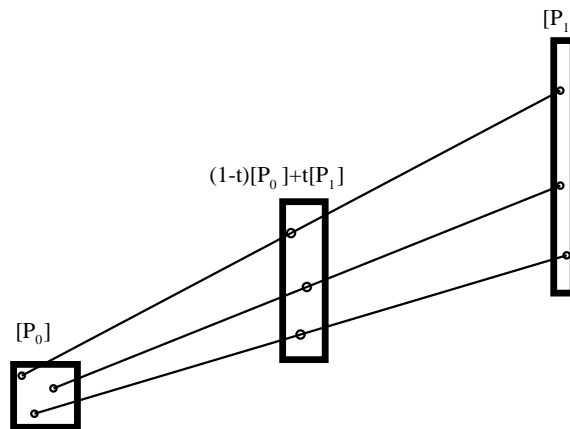


Figure 4: The affine map of two vector intervals.

maps of the midpoints, widths, and heights of $[\mathbf{P}_0]$ and $[\mathbf{P}_1]$.

For a Bézier curve of the form (13), we may regard the de Casteljau algorithm as a repeated application of (15)

to the control points $\{\mathbf{P}_k\}$. With $\mathbf{P}_k^{(0)} = \mathbf{P}_k$ for $k = 0, \dots, n$, we set

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

and $r = 0, \dots, n-1$. 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 (13), while the entries

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

on the left- and right-hand 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 (18) to *interval* Bézier 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.

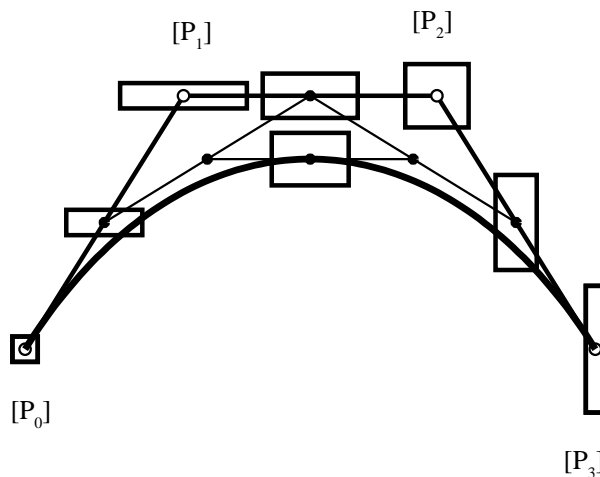


Figure 5: Interval de Casteljau algorithm.

3.2 Centered form

All the familiar algorithms and characteristics that we associate with Bézier curves — the subdivision and degree elevation algorithms, the variation-diminishing property and convex-hull confinement — carry over to interval Bézier curves. These operations can be more conveniently formulated if the control points are expressed in *centered form*. We re-write each control point $[\mathbf{P}_k]$ in the following manner:

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

where $[i]$ denotes the interval $[-1, +1]$. In (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 $[\mathbf{P}_k]$. Note that the x and y components of \mathbf{e}_k are necessarily non-negative.

Using centered form, the affine map of two interval points may be written as

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

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

For $0 \leq t \leq 1$, an interval Bézier curve may be expressed in centered form as follows:

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

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

Thus, the interval Bézier curve $[\mathbf{P}](t)$ over $0 \leq t \leq 1$ can be split into two independent “components” — a *center curve* $\overline{[\mathbf{P}]}(t)$, and an *error curve* $\mathbf{e}(t)$. Note that 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

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

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

$$\begin{aligned}
[\mathbf{M}](p_0 + e_0[i], p_1 + e_1[i], t) &= \\
\begin{cases} p_0(1-t) + p_1t + \{e_0(1-t) - e_1t\}[i] & \text{for } t < 0, \\ p_0(1-t) + p_1t + \{e_0(1-t) + e_1t\}[i] & \text{for } 0 \leq t \leq 1, \\ p_0(1-t) + p_1t + \{e_0(t-1) + e_1t\}[i] & \text{for } t > 1, \end{cases} & \tag{24} \\
= p_0(1-t) + p_1t + \{e_0|1-t| + e_1|t|\}[i] & \tag{25}
\end{aligned}$$

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

The de Casteljau algorithm (18) is essentially a repeated application of the affine map (24). Figure 5 illustrates for the case $t = \frac{1}{2}$. By studying the behavior of the de Casteljau algorithm applied to an interval Bézier curves outside the unit interval, equation (24) leads to the following expressions:

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

where $\mathbf{e}(t)$ is the error curve for $t \in [0, 1]$ defined in (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. That is, it often occurs when dealing with interval arithmetic that the widths of the intervals “balloon” so rapidly that the practical value of the interval technique is seriously impaired. For example, interval operations are generally not reversible:

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

Thus, it is noteworthy that ballooning does not occur with the de Casteljau algorithm when applied within the unit parameter interval. Even after repeated applications of the de Casteljau algorithm to an interval Bézier 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.

3.3 Floating point error propagation

Our discussion to this point has assumed that all arithmetic is performed exactly, with no floating point error. The error propagation in operations on Bernstein-form polynomials has been quite thoroughly discussed in [2]. We now demonstrate that the interval representation of Bézier curves provides a uniform framework within which to express both errors of approximation and floating point error. Furthermore, the geometric flavor of interval Bézier curves provides some pedagogical insight into the nature of floating point errors — a geometric setting in which to interpret the algebraic results in [2].

Let us first revisit the problem of computing the affine map of two vector intervals shown in Figure 4. When finite precision arithmetic is involved, some uncertainty is introduced. In order to robustly represent that uncertainty, the resulting interval should be widened an appropriate amount $\epsilon(t)[i]$ as shown in Figure 6. Here, the affine map for $t = \frac{1}{2}$ is shown with the exact arithmetic in solid and floating point error bounds in dashed line.

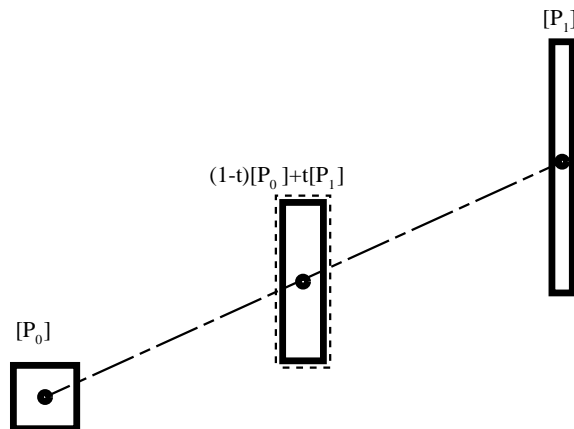


Figure 6: Affine map in floating point.

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

$$\eta = 2^{-d}. \quad (29)$$

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

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

Applying this to the affine map equation 25 yields (see equation 26 in [2])

$$\text{fl}[\mathbf{M}](p_0 + e_0[i], p_1 + e_1[i], t) \in [\mathbf{M}](p_0 + e_0[i], p_1 + e_1[i], t) + \epsilon(t) \quad (31)$$

where

$$\epsilon(t) = \begin{cases} \{|p_0|(1-t) - |p_1|t + |p_0(1-t) + p_1t|\} \eta [i] & \text{for } t < 0, \\ \{|p_0|(1-t) + |p_1|t + |p_0(1-t) + p_1t|\} \eta [i] & \text{for } 0 \leq t \leq 1, \\ \{|p_0|(t-1) + |p_1|t + |p_0(1-t) + p_1t|\} \eta [i] & \text{for } t > 1, \end{cases} \quad (32)$$

$$= \{|1-t||p_0| + |t||p_1| + |p_0(1-t) + p_1t|\} \eta [i]. \quad (33)$$

In words, to represent the effects of floating point roundoff in computing an affine map, the rectangle obtained by computing the affine map in exact arithmetic must be fattened an absolute amount $\epsilon(t)$.

4 Approximation by interval polynomials

Let $f(t)$ be differentiable $n + 1$ times on the interval $[a, b]$, and let $a \leq t_0 < \dots < t_n \leq b$ be an ordered sequence of $n + 1$ distinct points on that interval. The *Lagrange interpolant* to the sampled values $f_k = f(t_k)$, $k = 0, \dots, n$

at these points is the unique polynomial of degree n given by

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

where the $n + 1$ linearly independent polynomials

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

constitute the *Lagrange basis* for the nodes t_0, \dots, t_n . Since the basis (35) satisfies

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

for $0 \leq j, k \leq n$ it is clear that $F_n(t)$ reproduces the values of $f(t)$ at each node: $F_n(t_k) = f_k$ for $k = 0, \dots, n$.

4.1 Remainder formulae and 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 $E_n(t) = f(t) - F_n(t)$, and if we have information on the behavior of the derivative $f^{(n+1)}(t)$ over $t \in [a, b]$, we may express $E_n(t)$ as *Cauchy remainder* [1]:

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

Although, for each t , the value ξ at which the right hand side of (37) gives the error $E_n(t)$ of the Lagrange interpolant is not easily determined, 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 may write

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

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

In the particular form (38), only the remainder term has a non-degenerate interval coefficient, but if we choose to represent this interval polynomial in another basis — for example, as

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

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 formulae giving these coefficients in terms of the nodes t_0, \dots, t_n and values f_0, \dots, 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 formulae below only for simpler cases of practical interest.

4.2 Hermite interpolation

Hermite interpolation involves the interpolation of the *values* and *derivatives* of $f(t)$ at specified points. It may be regarded as a limiting form of Lagrange interpolation in which a sequence $m + 1$ consecutive nodes t_k, \dots, t_{k+m} merges; $F_n(t)$ is then required to match the value $f(t_k)$ and first m derivatives $f'(t_k), f''(t_k), \dots, f^{(m)}(t_k)$ of $f(t)$ at t_k (i.e., $F_n(t)$ has an m -fold osculation to $f(t)$ at t_k).

In particular, suppose t_0, \dots, t_r is a monotone sequence of $r + 1$ distinct nodes, with which we associate non-negative integers m_0, \dots, m_r such that $m_0 + \dots + m_r + r = n$. Then the unique polynomial $F_n(t)$ satisfying the interpolation problem

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

has the remainder term

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

From the above, we can write down an expression analogous to (38), namely

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

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

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

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

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

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

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

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

The remainder term (37) in this case becomes

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

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

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 end-point values is just $F_1(t) = f_0(1-t) + f_1 t$, and we may write $f(t) = F_1(t) + \frac{1}{2} f''(\xi) t(t-1)$ 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

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

In order to formulate the above as a quadratic interval polynomial $[F_2(t)]$, we degree-elevate $F_1(t)$ by multiplying it by $1 = (1-t) + t$ to obtain

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

where

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

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

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

For cubic Hermite interpolation to function values f_0, f_1 and derivatives f'_0, f'_1 at the end-points of $[0, 1]$, the Bernstein-Bézier form of the interpolant is

$$F_3(t) = f_0 B_0^3(t) + \frac{3f_0 + f'_0}{3} B_1^3(t) + \frac{3f_1 - f'_1}{3} B_2^3(t) + f_1 B_3^3(t). \quad (51)$$

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

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

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

$$\begin{aligned} [F_{4,0}] &= f_0, \quad [F_{4,1}] = f_0 + \frac{1}{4}f'_0, \\ [F_{4,2}] &= \frac{1}{2}(f_0 + f_1) + \frac{1}{6}(f'_0 - f'_1) + \frac{1}{144}[f^{(4)}], \\ [F_{4,3}] &= f_1 - \frac{1}{4}f'_1, \quad [F_{4,4}] = f_1. \end{aligned} \quad (53)$$

Note that in symmetric Hermite interpolation of function values and derivatives to order $(n-1)/2$ at interval end points, only the middle coefficient $[F_{n+1, (n+1)/2}]$ of the interpolating interval polynomial $[F_{n+1}](t)$ has non-zero width, since the residual (46) contributes only to the term involving the basis function $B_{(n+1)/2}^{n+1}(t)$. The width of this middle coefficient is $[f_{\max}^{(n+1)} - f_{\min}^{(n+1)}] / [(n+1)n \cdots \frac{1}{2}(n+3)]^2$. Note also that the width of $[F_{n+1}](t)$ degenerates to zero at $t = 0$ and 1 .

Example 1. 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 (53) of the Hermite interpolant are then:

$$[F_{4,0}] = 1, \quad [F_{4,1}] = \frac{5}{4}, \quad [F_{4,2}] = \frac{1}{144}[97 + 48e, 96 + 49e], \quad [F_{4,3}] = \frac{3}{4}e, \quad [F_{4,4}] = e. \quad (54)$$

We can 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 $\sim 1\%$ over the entire interval $[0, 1]$.

Example 2. 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

$$[F_{4,0}] = 0, \quad [F_{4,1}] = \frac{\pi}{8}, \quad [F_{4,2}] = \frac{1}{2304}[192(6 + \pi), 192(6 + \pi) + \pi^4], \quad [F_{4,3}] = 1, \quad [F_{4,4}] = 1. \quad (55)$$

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

Note that if the function $f(t)$ to be approximated on $[0, 1]$ is actually a *polynomial* of degree n with Bernstein coefficients p_0, \dots, p_n , its r -th derivative can be written as

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

in terms of the r -th forward differences $\Delta^r p_k$, and by the convex hull property we then have the derivative bounds

$$\frac{n!}{(n-r)!} \min_k \Delta^r p_k \leq f^{(r)}(t) \leq \frac{n!}{(n-r)!} \max_k \Delta^r p_k \quad \text{for } t \in [0, 1]. \quad (57)$$

5 Approximation by interval Bézier 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, i.e., parametric curves. There is, however, an important difference regarding the interpretation of the remainder term.

If $\mathbf{P}_0, \dots, \mathbf{P}_n$ is a sequence of points corresponding to $n+1$ distinct parameter values t_0, \dots, 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

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

but the remainder formula in this case 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 $\mathbf{P}_n(t) = \{X_n(t), Y_n(t)\}$ separately, and we then have

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

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

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

allows us to express the remainder term as

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

In formulating interval-polynomial approximants to curves, it is natural to consider also *piecewise* interval polynomials, i.e., *interval splines*. This extension is not difficult, and we give here only a brief sketch of the main ideas. It is well known that a C^2 piecewise-cubic curve can be constructed to interpolate any sequence of points $\mathbf{P}_0, \dots, \mathbf{P}_n$ in the plane. Typically, this requires imposing “end conditions” and solving a tridiagonal linear system for derivatives $\mathbf{P}'_0, \dots, \mathbf{P}'_n$ 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 end points $\mathbf{P}(0) = \mathbf{P}_{k-1}$, $\mathbf{P}(1) = \mathbf{P}_k$ 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 $\mathbf{P}_0, \dots, \mathbf{P}_n$ the width of the interval spline shrinks to zero.

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

$$\begin{aligned} [\mathbf{P}_{4,0}] &= \mathbf{P}(0), \quad [\mathbf{P}_{4,1}] = \mathbf{P}(0) + \frac{1}{4}\mathbf{P}'(0), \\ [\mathbf{P}_{4,2}] &= \frac{1}{2}(\mathbf{P}(0) + \mathbf{P}(1)) + \frac{1}{6}(\mathbf{P}'(0) - \mathbf{P}'(1)) + \frac{1}{144}[\mathbf{P}^{(4)}], \\ [\mathbf{P}_{4,3}] &= \mathbf{P}(1) - \frac{1}{4}\mathbf{P}'(1), \quad [\mathbf{P}_{4,4}] = \mathbf{P}(1). \end{aligned} \quad (62)$$

Example 3. A quarter circle, $\mathbf{P}(t) = (\cos \frac{\pi t}{2}, \sin \frac{\pi t}{2})$, can be approximated as a quartic interval Bézier curve using the coefficients in (55):

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

The above approximates an *arc length* parameterization of the circle to within two significant digits.

6 Conclusion

The ideas presented in this paper are of general nature. In some applications, better methods exist for bounding the error. For example, [10] presents a method for approximating rational Bézier curves with polynomial Bézier curves, and the resulting error bound is tighter than can be obtained in this more general method. Likewise, [8] presents a method for bounding the error of Hermite approximation of offset curves which is superior to this general method. This paper was condensed from a more extensive report [9].

Acknowledgments: The first author was supported in part by NSF grant numbers DMC-8657057 and DMC-8813688, and through a grant from IBM.

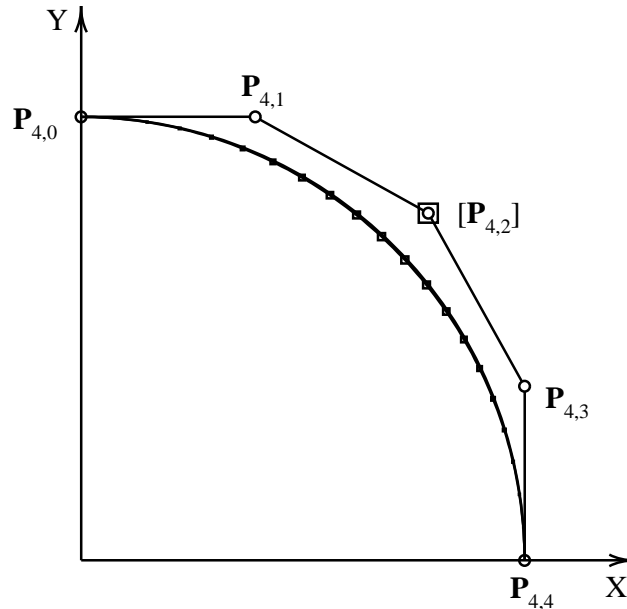


Figure 7: Approximate arc length parameterization of circle.

References

- [1] Philip J. Davis. *Interpolation and Approximation*. Dover, New York, 1963.
- [2] Rida T. Farouki and V. T. Rajan. On the numerical condition of polynomials in Bernstein form. *Computer Aided Geometric Design*, 4:191–216, 1987.
- [3] Rida T. Farouki and V. T. Rajan. Algorithms for polynomials in Bernstein form. *Computer Aided Geometric Design*, 5:1–26, 1988.
- [4] E. Hansen. Interval forms of Newton’s method. *Computing*, 20:153–163, 1978.
- [5] R. E. Moore. *Interval Analysis*. Prentice–Hall, Englewood Cliffs, NJ, 1966.
- [6] S. P. Mudur and P. A. Koparkar. Interval methods for processing geometric objects. *IEEE Computer Graphics and Applications*, 2:7–17, 1984.
- [7] J. Rokne. Reducing the degree of an interval polynomial. *Computing*, 14:5–14, 1975.
- [8] Thomas W. Sederberg and David B. Buehler. Offsets of Bézier curves: Hermite approximation with error bound. In Tom Lyche and Larry Schumaker, editors, *Mathematical Methods in CAGD and Image Processing*, New York, 1992. Academic Press.
- [9] Thomas W. Sederberg and Rida T. Farouki. Approximation by interval bézier curves. IBM Research Report RC17503, December 1991.
- [10] Thomas W. Sederberg and Masanori Kakimoto. Approximating rational curves using polynomial curves. In Gerald Farin, editor, *NURBS for Curve and Surface Design*, pages 149–158, Philadelphia, 1991. SIAM.

Biographical Sketches

Rida T. Farouki received the BA in engineering science from Oxford University in 1978 and the PhD in theoretical astrophysics from Cornell University in 1983. He worked at GE Research and Development Center in Schenectady, NY from 1983 to 1986, and is currently a research staff member at IBM Thomas J. Watson Research Center in Yorktown Heights, NY. 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.

Thomas W. Sederberg received the BS degree from Brigham Young University in Civil Engineering, and the PhD from Purdue University in Mechanical Engineering in 1983. He is currently an associate professor of Civil Engineering at Brigham Young University. His research is in the area of computer aided geometric design and computer graphics. He is a member of ACM SIGGRAPH and IEEE. He served as papers chair for SIGGRAPH '91.