

Chapter 22W

Splines and Subdivision Curves

22.1W Introduction

Everywhere in computer graphics are problems that need to be solved with smooth curves. Examples are

- You want to filter an image with a circularly symmetric filter that approximates a Gaussian, but has finite extent, so you want to create a smooth approximation to the truncated Gaussian.
- You want to let a user specify “key frames” in an animation and then interpolate between these by smoothly changing whatever parameters the user has set; the way the parameters are varied is defined by a smooth curve.
- You want a user to draw smooth curves to outline a cartoon character, but drawing with either a mouse or pen is difficult for many people; you’d like to take a rough drawing and clean it up into a smooth one, which the user can then adjust easily without re-introducing roughness.
- You’re creating a drawing program in which a user will be drawing arrows to label items; the tails of the arrows need to be curved, but neither arcs of circles or arcs of ellipses seem rich enough to you; you want a more general kind of smooth shape to work with.

In all cases *splines* are a good solution to your problems. Splines are parametric curves whose shape is governed by a sequence of “control points.” There are many kinds of splines. For some, called *interpolating splines* (see Figure 22W.1), the curve passes directly through the control points; for others (*approximating splines*) the curve is influenced by the control points, but may not pass through them. We’ve already encountered something very like splines: the *subdivision curves* from Chapter 4. The construction given there defines just one of many kinds of subdivision curve. Notice that the limit curve ended up passing near the

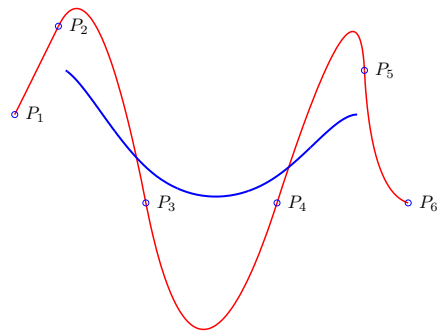


Figure 22W.1: Six control points P_0, \dots, P_5 and two splines; the first is an interpolating spline, which passes through the points; the other is an approximating spline, whose shape is guided by the control points, but which does not necessarily pass through them.

points of the initial polyline (which we could call “control points”), so this kind of subdivision curve would be called “approximating” rather than “interpolating.”

The relationship between subdivision curves (which are defined geometrically) and splines (which are defined parametrically) is very strong; we’ll discuss it in Section 22.16W.

22.1.1W Historical origins

The creation of smooth curves (and surfaces, but this chapter concentrates on curves, and we’ll only discuss those henceforth) was historically not only the domain of mathematicians, but of artisans as well; shipwrights drew designs for boats by drawing their cross-sections in several planes, for instance. These cross-sections were then measured and their shapes tabulated by recording several points on each curve. When it came time to build the ship, these measured points were laid out on a flat surface (the floor of a large room, for instance, with the points being marked by nails driven partway into the floor) at full scale, but then there was the challenge of joining them up smoothly. A simple connect-the-dots approach would have produced sharp corners in the ship, which was undesirable. Instead, strips of wood or metal were bent so as to pass through the chosen points, usually by bending the strip around the nails. The resulting curve was then traced

onto the floor, resulting in a model shape that determined how the ship was to be built. The thin pieces of wood or metal were known as *splines*, and the term has survived to this day to mean “a smooth curve whose shape is determined by several key points.” Splines on a smaller scale were widely used in drafting before the advent of computer aided design tools. Instead of pounding nails into a desktop, a draftsman would place lead weights (called *ducks*) with small hooks attached; the hooks would take the place of the nails in governing the shape of the spline.

22.1.2W Two examples: Hermite and Bézier curves

We begin by describing two types of curves that get a lot of use in spline-related applications. These provide a concrete instance of ideas developed in this chapter, and they’ll serve as examples from which we’ll generalize. They’re so common that having their description right away will help you start solving spline-based problems directly. Hermite curves are not, strictly speaking, a kind of spline: an Hermite curve is determined by two control points and a vector at each point; by contrast, Bézier curves are determined by four control points.

Calling a curve a “Bézier” or “Hermite” curve is slightly misleading: the Bézier and Hermite formulations provide two different ways to *specify* a curve. If you’re given a parametric cubic curve, there’s no way to tell whether it was specified by the Bézier approach, the Hermite, or some other approach entirely. We’ll see how to convert from a Bézier specification of a curve (a list of four points) to an Hermite specification (two points and two vectors) of exactly the same curve, and vice versa.

22.1.2W.1 Hermite curves

An *Hermite* (pronounced “air-meet”) curve goes from a point P to a point Q , with initial tangent vector \mathbf{v} and final tangent vector \mathbf{w} (Figure 22W.2). The points P and Q and the vectors \mathbf{v} and \mathbf{w} completely determine the curve.

The formula for the Hermite curve is:

$$\gamma(t) = (2t^3 - 3t^2 + 1)P + (-2t^3 + 3t^2)Q + (t^3 - 2t^2 + t)\mathbf{v} + (t^3 - t^2)\mathbf{w} \quad (22.1W)$$

$$= (2t^3 - 3t^2 + 1)P + (-2t^3 + 3t^2)Q + t(1-t)^2\mathbf{v} + t^2(1-t)\mathbf{w}. \quad (22.2W)$$

We’ll discuss shortly how these polynomials were found. For now, just convince yourself that in fact the curve satisfies $\gamma(0) = P$, $\gamma(1) = Q$, $\gamma'(0) = \mathbf{v}$ and $\gamma'(1) = \mathbf{w}$.

The four cubic polynomials in this expression tell us how the inputs are combined to make the curve γ . In particular, the factors of t and $(1-t)$ in the polynomials for \mathbf{v} and \mathbf{w} tell us that these inputs have no influence on the locations of the endpoints $\gamma(0)$ and $\gamma(1)$, while the factor of t^2 in the polynomial for Q shows that Q has no influence either on the location of $\gamma(0)$ or on the tangent vector $\gamma'(0)$ (see Exercise 22.1). The other polynomials can be read similarly. The graphs of these four polynomials, shown in Figure 22W.3, reveal this same information.

A curve written in terms of polynomials can be written in a matrix form. Letting $\mathbf{T}(t)$ denote a vector containing powers of t , i.e., $\mathbf{T}(t) = [1 \ t \ t^2 \ t^3]^T$, we

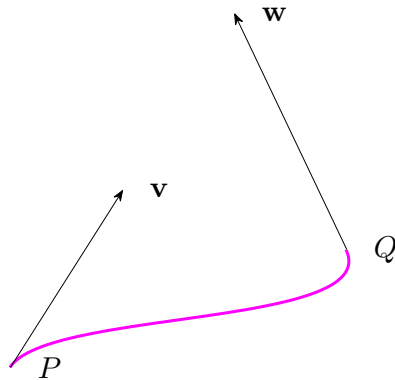


Figure 22W.2: An Hermite curve γ goes from $\gamma(0) = P$ to $\gamma(1) = Q$, with initial tangent vector $\gamma'(0) = \mathbf{v}$ and final tangent vector $\gamma'(1) = \mathbf{w}$.

can write

$$\gamma(t) = [P; Q; \mathbf{v}; \mathbf{w}] \cdot \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \cdot \mathbf{T}(t) \quad (22.3W)$$

where the first factor is a matrix whose columns are the coordinates of P , Q , \mathbf{v} , and \mathbf{w} , respectively (we'll use the semicolon notation for this in the future as well), and the middle matrix contains the coefficients of the polynomials for the Hermite spline.

Inline Exercise 22W.1: Multiply out, by hand, the second and third factors in the expression for $\gamma(t)$; you should get a column vector of four polynomials. Suppose that we had defined \mathbf{T} to be the vector $[t^3 \ t^2 \ t \ 1]^T$ instead; how would the second matrix in the expression for $\gamma(t)$ have to change?

Inline Exercise 22W.2: Suppose that $\zeta(t) = (1-t)P + tQ$. Write ζ in a matrix form like that of Equation 22.3W. Your vector $\mathbf{T}(t)$ will be just $[1 \ t]^T$.

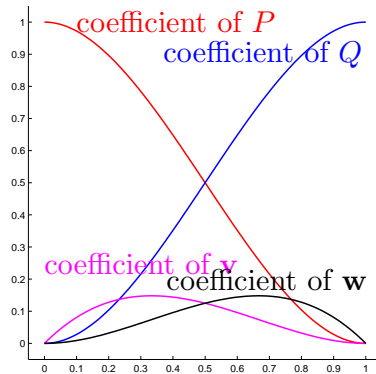


Figure 22W.3: The four Hermite polynomials, plotted on a single set of axes. At $t = 0$, the polynomial for P is 1, and all others are zero; the only one with nonzero derivative is the polynomial for \mathbf{v} , thus \mathbf{v} controls $\gamma'(0)$. Similar observations about the other curves tell us the relations of these polynomials to inputs to the Hermite curve.

22.1.2W.2 Programming

An Hermite curve is a function of five things: two points, two vectors, and the parameter t . In C#, we can implement an Hermite curve as a function of five variables, as shown in Listing 22W.1.

```

1 public Point hermite(double t, Point P, Point Q, Vector v,
2     Vector w)
3 {
4     double t2 = t*t;
5     double t3 = t*t2;
6
7     double p1 = 2. * t3 - 3. * t2 + 1.0;
8     double p2 = -2. * t3 +3. * t2;    // just 1 - p1.
9     double p3 = t3 - 2. * t2 + t;
10    double p4 = t3 - t2;
11
12    return AffineCombination(P, p1, Q, p2) + (p3 * v + p4 * w)
13    ;

```

Listing 22W.1: Nonoptimized C# code for an Hermite spline; the code assumes that the `Point` and `Vector` classes are already defined, as well as operators for combining them.

Such an implementation is not very efficient, however, if we intend to repeatedly call it with the same set of points and vectors, and vary only t ; used that way, it makes more sense to build up a matrix, as in Equation 22.3W, and re-use it. That idea, encapsulated in a class, is partly shown in Listing 22W.2.

```

1
2 public class HermiteCurve
3 {
4     private Mat24 SMat; // Mat24 is a 2x4 matrix class
5
6     public HermiteCurve(Point2 Start, Point2 Finish, Vector2
7         StartTangent, Vector2 FinishTangent) {
8
9         Mat24 Geom = new Mat44(Start.coords(), Finish.coords())
10            ,
11            StartTangent.coords(),
12            FinishTangent.coords());
13
14         Mat44 HMat =  $\begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$ ;
15
16         Smat = Geom * HMat;
17     }
18
19     public Point2 SplinePoint(double t){
20         Vec4 Tvec = new Vec4(1, t, t*t, t*t*t);
21         return Smat * Tvec;
22     }
23 }

```


Listing 22W.2: An Hermite curve class whose constructor builds a matrix from the control data; calculating points on the spline is done with a matrix-multiply. Shown is a 2D spline, but the generalization to 3D would just use a 3×4 matrix in place of the 2×4 one. The `coords` method of a `Point` returns its coordinates in the standard coordinate system; the same goes for `Vectors`. We're also assuming the existence of constructors for matrices in terms of their columns.

Inline Exercise 22W.3: How would you augment this class to allow one to compute several points on the spline at once, i.e., how would you write a method `public Point2[] SplinePoint (double[] t)` that took in an array of n doubles and produced an array of n Points on the spline curve? Hint: make a $4 \times n$ array containing powers of the t -values.

22.1.2W.3 What's an Hermite curve good for?

Hermite curves give a user easy control over the location of the curve endpoints and the tangents there. That's often useful when you have to edit a curve, but it's not always the best way to create one. One interface for creating Hermite curves has the user click on the first point, drag and release to indicate the first tangent, click on the last point, and finally drag and release to indicate the second tangent, at which point the curve is drawn.

Inline Exercise 22W.4: A simple way to draw a spline curve is to draw instead a very good polyline approximation of the curve. You also want to be sure that both endpoints of the curve are included in the polyline, rather than using nearby points, so that if you join two curves at a point, but not smoothly, you get a sharp corner rather than a “shortcut” across it. It’s tempting to just use lots of equispaced t -values, and hope that the results will look good. But how do you determine the spacing? In drawing a polyline approximation, it’s nice to avoid long straight edges, and to avoid too-rapid changes of direction at the vertices. In this exercise, we’ll do some crude estimates to ensure good curve drawings. (a) For the Hermite curve $\gamma(t) = (2t^3 - 3t^2 + 1)P + (-2t^3 + 3t^2)Q + t(1-t)^2\mathbf{v} + t^2(1-t)\mathbf{w}$ ($0 \leq t \leq 1$), write down $\gamma'(t)$ as a polynomial linear combination of \mathbf{v} , \mathbf{w} , and $\mathbf{u} = Q - P$, i.e., write $\gamma'(t) = q_1(t)\mathbf{v} + q_2(t)\mathbf{w} + q_3(t)(Q - P)$ for three polynomials q_1, q_2 , and q_3 . (b) Find the maximum value m_i of each $|q_i|$ ($i = 1, 2, 3$) on the interval $[0, 1]$. (c) Using the fact that $\|\mathbf{a} + \mathbf{b}\| \leq \|\mathbf{a}\| + \|\mathbf{b}\|$, we know that $\|\gamma'(t)\| \leq |q_1(t)|\|\mathbf{v}\| + |q_2(t)|\|\mathbf{w}\| + |q_3(t)|\|\mathbf{u}\|$; the largest value of $\|\gamma'(t)\|$, over all t in $[0, 1]$, is therefore bounded by $K = m_1\|\mathbf{v}\| + m_2\|\mathbf{w}\| + m_3\|\mathbf{u}\|$. (d) If we want each segment to have a length no greater than some constant L , we can simply choose a t -spacing that’s no greater than L/K . How would you determine such a spacing in a program? Be careful to consider special cases, such as $K = 0$, and the desire to have both $t = 0$ and $t = 1$ included in the list of t -values. (e) Perform a similar analysis that will ensure that the second derivative is no greater than some other constant M . (f) Write a small program that lets you experiment with the resulting polygonal approximations by varying K and M (and the endpoints and tangents of your Hermite curve,

of course).  The estimates on the derivative and second derivative given here is very crude; the t -value at which q_1 takes its maximum is nowhere near the one where q_3 is maximal, etc. Finer estimates are not hard to produce, and adaptive algorithms, which choose a t -value, estimate derivatives there, and use those to choose a subsequent t -value, etc., are also relatively simple.

Programming Exercise 22W.1: Implement the interface described above in the 2D testbed. When the user clicks-and-drags from P to S , you should use P as the starting point for the Hermite curve, and $S - P$ as the starting tangent \mathbf{v} ; as the user then moves the cursor in preparation for clicking-and-dragging the second point and tangent, you can constantly redisplay an Hermite curve, using the current cursor position for the final point, and the zero vector for the final tangent. When the user clicks the final point Q and begins to drag, use $M - Q$ (where M is the mouse position) as the final tangent \mathbf{w} and display the curve; when the user releases the mouse button, the curve should remain visible. When the user initiates another click-and-drag, you should clear the canvas and start again.

One can even draw a connected sequence of Hermite curves, using the last point (Q) of one segment as the initial point (P) of the next segment. To make the curves join up smoothly, one can also use the final tangent (\mathbf{w}) of one segment as the initial tangent (\mathbf{v}) of the next segment. As we’ll soon see, this approach to smoothness is somewhat limiting, but it lets you experiment with how Hermite curves can be controlled.

Now we return to the question of where the polynomials in the Hermite formulation came from. First, with a cubic polynomial there are four unknown coefficients; at the same time, we have, for each coordinate, four conditions (it should start here, end here, and have these derivatives at the start and the end). Since the number of conditions is the same as the number of unknowns, we expect to be able to find a unique solution to our problem. In such cases, it's best to *assume* that a solution exists, and see what the conditions tell us. Typically, they'll give formulas for the unknowns, and we can then verify that these have the desired properties.

We're looking for an expression of the form

$$\gamma(t) = p_1(t)P + p_2(t)Q + p_3(t)\mathbf{v} + p_4(t)\mathbf{w} \quad (22.4W)$$

where each p_i is a cubic polynomial, and where $\gamma(0) = P$, $\gamma(1) = Q$, $\gamma'(0) = \mathbf{v}$ and $\gamma'(1) = \mathbf{w}$. The first of these conditions tells us that

$$P = p_1(0)P + p_2(0)Q + p_3(0)\mathbf{v} + p_4(0)\mathbf{w}. \quad (22.5W)$$

Since we want this to be true for *any* values of P , Q , \mathbf{v} and \mathbf{w} , we must have

$$p_1(0) = 1 \quad (22.6W)$$

$$p_2(0) = 0 \quad (22.7W)$$

$$p_3(0) = 0 \quad (22.8W)$$

$$p_4(0) = 0. \quad (22.9W)$$

Similarly, the second, third and fourth conditions tell us other things about the p_i and their derivatives. (You should write these out). If we write $p_1(t) = at^3 + bt^2 + ct + d$, then the four resulting conditions on p_1 are

$$\begin{array}{rclcl} p_1(0) & = & & & d & = & 1 \\ p_1(1) & = & a & + & b & + & c & + & d & = & 0 \\ p_1'(0) & = & & & c & & & & & = & 0 \\ p_1'(1) & = & 3a & + & 2b & + & c & & & = & 0. \end{array}$$

The unique solution to this system of four equations in four unknowns is $a = 2$, $b = -3$, $c = 0$, $d = 1$, so $p_1(t) = 2t^3 - 3t^2 + 1$. Similar computations determine the other three polynomials.

Exercise 22W.7 shows another approach to finding these polynomials.

22.2W Bézier curves

A Bézier curve is, like an Hermite curve, specified by four parameters; in the case of the Bézier curve, however, all four are *points*, which we'll call P_1, P_2, P_3 and P_4 . The curve starts at P_1 , in the direction $P_2 - P_1$, and ends at P_4 , travelling in the direction $P_4 - P_3$. The formula for a Bézier curve, in both polynomial and matrix form, is

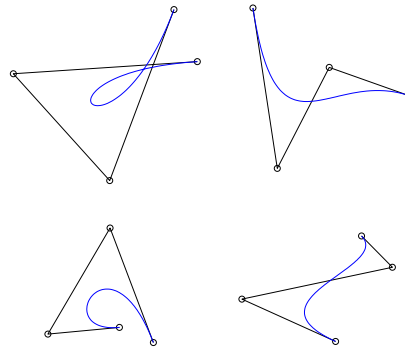



Figure 22W.4: Several Bézier curves, shown with their control hulls. Can you look at the Bernstein polynomials (plotted at the right) and tell why Bézier curves interpolate (i.e., pass through) their first and last control points?

$$\gamma(t) = [P_1; P_2; P_3; P_4] \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \quad (22.10W)$$

$$= (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4. \quad (22.11W)$$

The coefficients of P_1, \dots, P_4 are the degree three **Bernstein polynomials**. They follow a very regular pattern, and will appear again later. These polynomials, together with a selection of Bézier curves, are shown in Figure 22W.4.

Inline Exercise 22W.5: (a) Show that if the control points are at $x = 0, \frac{1}{3}, \frac{2}{3}, 1$ on the x -axis, then the Bézier curve is exactly $\gamma(t) = (t, 0)$. (b) More generally, if P_1, \dots, P_4 are equally spaced along a line, then the Bézier curve is exactly $\gamma(t) = (1-t)P_1 + tP_4$. (c)  Make a corresponding statement about Hermite curves: what property must the tangents have to make the Hermite curve from P to Q be $\gamma(t) = (1-t)P + tQ$?

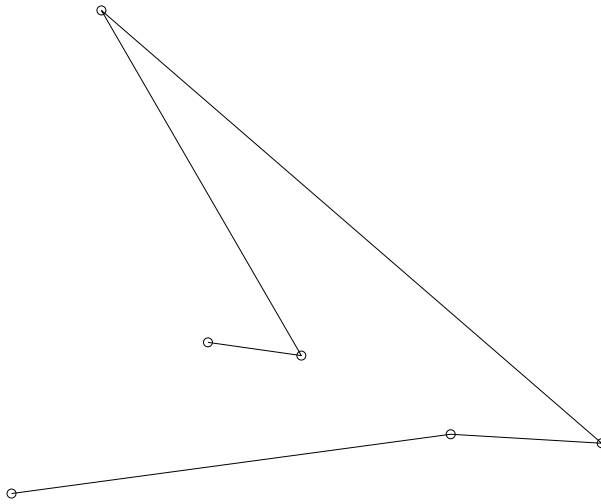


Figure 22W.5: A parametric plot of a piecewise degree-one polynomial curve; note that there are "breakpoints" at which the curve's direction changes abruptly. These correspond to two adjacent segments being defined by different degree-one polynomials.

Inline Exercise 22W.6: Consider a "quadratic Bézier curve" defined by three points instead of four, by the formula $\gamma(t) = (1-t)^2P_1 + 2t(1-t)P_2 + t^2P_3$. Where does this curve start and end, and what are its initial and final tangents?

22.3W Why piecewise cubics?

Before we discuss splines further, we digress briefly to give two rationales for why piecewise cubic curves are a natural choice. The first explanation is relatively simple; the second is somewhat more complex, and we omit some mathematical details. Neither is essential for understanding how to work with splines, but both are relevant in generalizing splines.

22.3.1W Planarity

Assuming we want to use piecewise polynomial curves, we can consider various degrees for the polynomials. With degree one polynomials, we get the kind of connect-the-dots curves that we saw in Chapter 9, which are simple but not smooth enough for most of our applications (see Figure 22W.5).

Moving to degree two polynomials, the resulting plots give us connected curves, and with some care, we can arrange that adjacent curve pieces meet, and that their tangent directions agree. Now the curves appear smooth as well as continuous (see Figure 22W.6).

It would seem that piecewise degree two curves are adequate: they bend nicely, they have no sharp corners (if we choose our polynomials carefully), and it's even not very difficult to make them pass through a sequence of chosen points. Indeed, for many simple tasks, they are completely adequate.

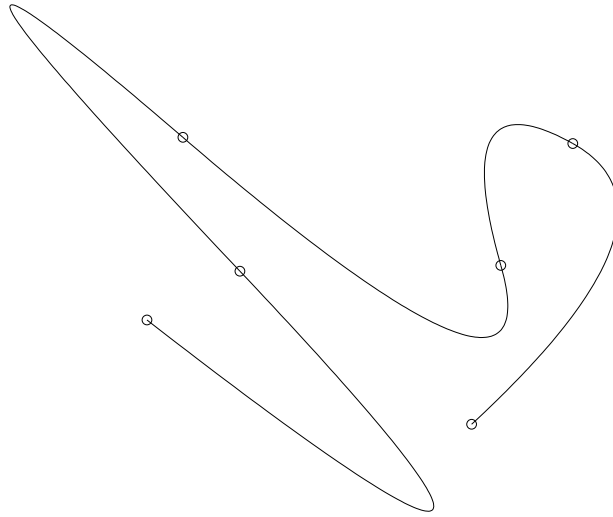


Figure 22W.6: A parametric plot of a piecewise degree-two polynomial curve, where adjacent polynomial segments have been chosen carefully so that they meet and their tangent directions agree at the meeting points. The result curve has no sharp corners.

But in three dimensions, something interesting happens: suppose that

$$q(t) = (x(t), y(t), z(t)) \quad (22.12W)$$

is a quadratic polynomial curve, i.e., that $x(t)$, $y(t)$, and $z(t)$ are all quadratics in the variable t . We'll see, in the next few paragraphs, that a parametric plot of the curve q lies in a single plane. That means that it cannot possibly take the shape of a helix, for instance: to model a spiral staircase, we'd have to glue together a collection of curves, each of which was planar, with an abrupt shift of the containing-plane at the join-points. This lack of *torsion* in the curves makes them inadequate for many important modeling tasks. Cubics, by contrast, *do* have torsion, and hence are widely used.

To see that every quadratic parametric curve in 3D is planar, we'll write out the polynomials explicitly. So

$$q(t) = (a_x t^2 + b_x t + c_x, a_y t^2 + b_y t + c_y, a_z t^2 + b_z t + c_z). \quad (22.13W)$$

Letting $\mathbf{a} = (a_x, a_y, a_z)$, similarly for \mathbf{b} and \mathbf{c} , we can rewrite this in the form

$$q(t) = t^2 \mathbf{a} + t \mathbf{b} + \mathbf{c}. \quad (22.14W)$$

We'll now show that the parametric plot of the curve q lies in the plane perpendicular to the vector $\mathbf{n} = \mathbf{a} \times \mathbf{b}$, passing through $q(0) = \mathbf{c}$. That plane is defined as the set of points X satisfying

$$(X - \mathbf{c}) \cdot \mathbf{n} = 0. \quad (22.15W)$$

We're claiming that for every t , $q(t)$ is on that plane, i.e., that

$$(q(t) - \mathbf{c}) \cdot \mathbf{n} = 0. \quad (22.16W)$$

The proof is just a computation:

$$(q(t) - \mathbf{c}) \cdot \mathbf{n} = ((t^2\mathbf{a} + t\mathbf{b} + \mathbf{c}) - \mathbf{c}) \cdot \mathbf{n} \quad (22.17\text{W})$$

$$= (t^2\mathbf{a} + t\mathbf{b}) \cdot (\mathbf{a} \times \mathbf{b}) \quad (22.18\text{W})$$

$$= t^2\mathbf{a} \cdot (\mathbf{a} \times \mathbf{b}) + t\mathbf{b} \cdot (\mathbf{a} \times \mathbf{b}) \quad (22.19\text{W})$$

$$= t^2(0) + t(0) \quad (22.20\text{W})$$

$$= 0. \quad (22.21\text{W})$$

Inline Exercise 22W.7:



Generalize the argument above to show that any degree n polynomial curve in $(n + 1)$ -space lies entirely in an n -dimensional subspace. You may need to use the n -fold cross product to do this.

22.3.2W Bending energy

If $q(t) = (x(t), y(t), z(t))$ is a parametric curve representing the shape of a bent metal wire, then the squared magnitude of the second derivative, $\|q''(t)\|^2$ is an approximation of the energy required at the point $q(t)$ to bend an initially straight wire into that shape. (The approximation is most accurate for shapes that are almost straight, i.e., one that represent small deformations of the straight wire). The integral of this approximation over the entire wire gives an approximation of the total energy

$$E = \int_a^b \|q''(t)\|^2 dt, \quad (22.22\text{W})$$

where a and b are the starting and ending parameter values for the shape. Let's assume that $a = 0$ and $b = 1$ to simplify things.

Let's now consider the problem shown in Figure 22W.7: we ask "among all curves that go from $P = (1, 1)$ to $Q = (3, 1)$, with initial velocity $\mathbf{v} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ and final velocity $\mathbf{w} = \begin{pmatrix} 1 \\ -\frac{1}{2} \end{pmatrix}$, which one has the least total bending energy?" Such a curve should be the shape taken on by a piece of metal that's held in such a way as to satisfy the constraints.

Section 22.17.4W presents an argument that this energy-minimizing curve must satisfy the differential equation

$$q''''(t) = 0 \text{ for } 0 < t < 1, \quad (22.23\text{W})$$

which means that $q(t)$ must be a cubic in t . This differential equation is independent of P, Q, \mathbf{v} and \mathbf{w} , so the general solution to the problem is always a cubic. Our earlier analysis shows that the particular optimal solution can be found using the Hermite formulation.

22.3.3W Gluing together Bézier curves

Returning to the subject of Bézier curves, a single Bézier curve may be useful in some places, but longer curves, with more bends, interpolating many points, are

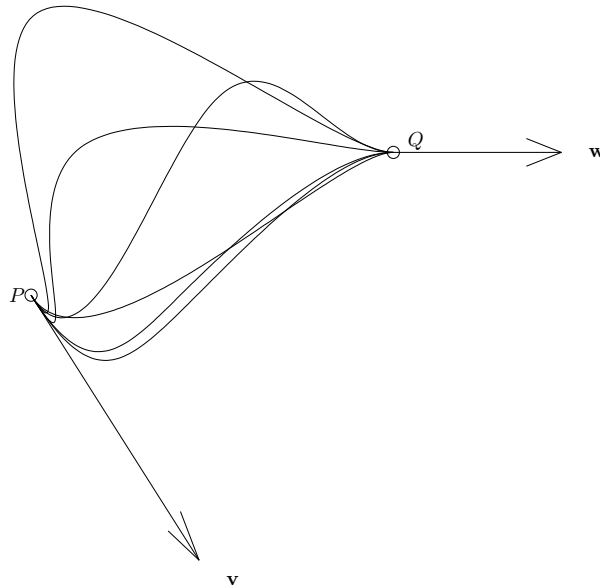


Figure 22W.7: We seek, among all parametric curves that start at $P = (1, 1)$ (at $t = 0$) and end at $Q = (3, 1)$ (at $t = 1$), and whose initial velocity is \mathbf{v} and whose final velocity is \mathbf{w} , one with the least total bending energy.

nice too. To build such a thing, we can join up multiple Bézier curves. Suppose that we have control point sequences P_1, \dots, P_4 and Q_1, \dots, Q_4 for two Bézier curves. How can we adjust the P s and Q s to make the end of the P -curve match the start of the Q -curve? Since the end of the P -curve is at P_4 , and the start of the Q -curve is at Q_1 , if we require that $P_4 = Q_1$, then the curves will connect together (Figure 22W.8).

The join between the curves is not generally smooth, however. If we call the curves γ_P and γ_Q , we've arranged that

$$\gamma_P(1) = \gamma_Q(0), \tag{22.24W}$$

but not that the *tangents* to the two curves match up. One way to do so is to insist that the tangent vectors to the two paths are also identical, i.e., that

$$\gamma'_P(1) = \gamma'_Q(0). \tag{22.25W}$$

Working through the algebra, this can be assured by making $Q_2 - Q_1 = P_4 - P_3$; since $Q_1 = P_4$, this simplifies to $Q_2 = P_4 + (P_4 - P_3)$.

Using γ_P and γ_Q we can create a new curve γ defined by

$$\gamma(t) = \begin{cases} \gamma_P(t) & 0 \leq t \leq 1 \\ \gamma_Q(t - 1) & 1 \leq t \leq 2 \end{cases} \tag{22.26W}$$

The new curve γ has $[0, 2]$ as its domain; on $[0, 1]$ it looks like γ_P , and on $[1, 2]$ it looks like γ_Q . Note, however, that this “looks like” means that for $1 \leq t \leq 2$, $\gamma(t) = \gamma_Q(t - 1)$, so that when we evaluate $\gamma(1.5)$, we compute $\gamma_Q(0.5)$, which is

$$p_1(0.5)Q_1 + \dots + p_4(0.5)Q_4. \tag{22.27W}$$

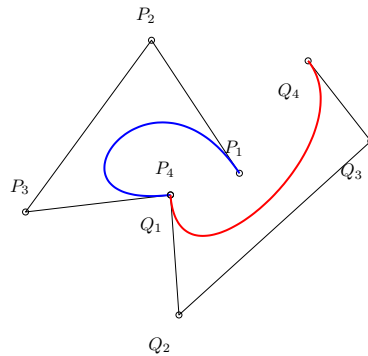


Figure 22W.8: Two Bézier curves that join together: the end of the first is the start of the second. Unfortunately, the join is not smooth.

so that each segment of the glued-together curve γ is evaluated by using the Bézier evaluation scheme.

22.4W Parametric and Geometric Continuity

The glued-together Bézier curve of the previous section has what is called C^1 -continuity, because the curve defined by

$$\gamma(t) = \begin{cases} \gamma_P(t) & 0 \leq t \leq 1 \\ \gamma_Q(t-1) & 1 \leq t \leq 2 \end{cases} \quad (22.28W)$$

is both continuous and has a continuous first derivative (we say “ γ is C^1 ”). In the earlier example, when we required only that $Q_1 = P_4$, the curve would be said to have C^0 -continuity, because although it was continuous, its first derivative might be discontinuous at $t = 1$. The idea can be generalized to require that the second derivative be continuous as well (C^2 -continuity), and so on. This notion of continuity – the matching of values, derivatives, second derivatives, and so on – is called **parametric continuity**, because it depends not only on the geometry of the curve, but on the parameterization.

Inline Exercise 22W.8: Consider the curve defined by $\gamma(t) = (1, 2t)$ for $0 \leq t \leq 1$ and by $\gamma(t) = (1, t + 2)$ for $1 \leq t \leq 2$. (a) Draw the parametric plot of this curve and observe that it looks “continuous and smooth” at the joint point (where $t = 1$). (b) Draw the *graph* of this curve, i.e., plot $\gamma(t)$ against t on three-dimensional axes. (c) Is γ C^1 ? (c) Now consider a different curve: let $\gamma(t) = (t^2, 0)$ for $-1 \leq t \leq 0$ and $\gamma(t) = (0, t^2)$ for $0 \leq t \leq 1$. Again draw a parametric plot and a graph of γ , and tell whether γ is C^1 .

A different condition for “continuity” requires only that the tangent vectors point in the same direction. Curves with that property at the join-points are said to be G^1 -*continuous* (and we sometimes refer to *geometric continuity* versus *parametric continuity*). What happens if one of the tangent vectors is the zero vector? Since it has no direction, the two directions cannot be equal. A cleaner way to express the condition for G^1 continuity is that the tangents must be positive multiples of one another. In summary, two Bézier curves γ_P and γ_Q join with C^1 continuity if $Q_1 = P_4$ and $Q_2 = P_4 + (P_4 - P_3)$, while they joint with G^1 continuity if $Q_1 = P_4$ and $Q_2 = P_4 + \alpha(P_4 - P_3)$ with $\alpha > 0$.

Inline Exercise 22W.9: Consider the two curves in the previous inline exercise. Which is G^1 ? Conclude that a C^1 curve is not necessarily G^1 , and vice versa.

22.5W Catmull-Rom splines

We’ll now work through an example spline with C^1 continuity, the *Catmull-Rom spline* or *CR spline*. One intended application of the CR-spline is in animation: if you are given the positions P_0, P_1, \dots, P_n of some object at time $t = 0, 1, 2, \dots, n$, you can use a CR-spline to fill in positions at the non-integer times in a way that looks reasonable. In particular, that means that if the positions P_0, P_1, \dots, P_n are equally spaced points on a straight line, the resulting spline $\gamma : [0, n] \rightarrow \mathbf{R}^2$ should be a uniform-speed straight line between the endpoints, i.e., should be

$$\gamma(t) = \frac{n-t}{n}P_0 + \frac{t}{n}P_n. \quad (22.29W)$$

Furthermore, to make the animation smooth, we want to have C^1 continuity.

Rather than having the user specify the tangents at each control point, the tangents are computed from nearby control points: the tangent at P_i (for $i \neq 0, n$) is parallel to $P_{i+1} - P_{i-1}$ (see Figure 22W.9). To make the remainder of the presentation simpler, we’ll add two “fictitious” control points, P_{-1} and P_{n+1} , defined by

$$\begin{aligned} P_{-1} &= P_0 - (P_1 - P_0) \\ P_{n+1} &= P_n + (P_n - P_{n-1}) \end{aligned}$$

Inline Exercise 22W.10: Suppose that P_0, \dots, P_n are given by $P_i = (i, i)$. What do the preceding formulas give for P_{-1} and P_{n+1} ?

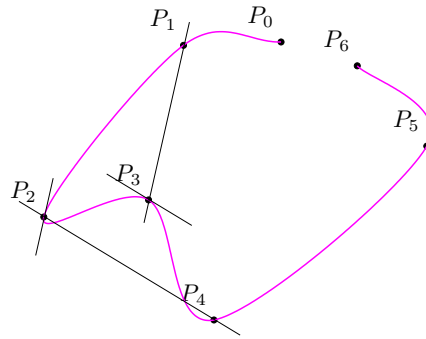


Figure 22W.9: A Catmull-Rom spline; the segment between P_2 and P_3 has a tangent (at P_2) that's parallel to the line from P_1 to P_3 ; the tangent at P_3 is parallel to the line from P_2 to P_4 .

Finally, we want our curve to be piecewise cubic on the interval from $t = i$ to $t = i + 1$; on this interval, the starting and ending points of the cubic are determined by P_i and P_{i+1} and the starting and ending tangents are affected by P_{i-1} and P_{i+2} , respectively, so we hope to write our curve, γ , on the interval $[i, i + 1]$ with an expression of the form

$$\gamma(t) = p_1(t - i)P_{i-1} + p_2(t - i)P_i + p_3(t - i)P_{i+1} + p_4(t - i)P_{i+2} \text{ for } i \leq t \leq i + 1, \quad (22.30W)$$

where the polynomials p_i (yet to be determined) are the same for every interval, and only the control points change.

Thus the problem to be solved is “What polynomials (if any) will create a spline that passes through the points with C^1 continuity, that has tangents in the right directions, and has the “fill in equispaced points on a line nicely” property?”

22.6W Catmull-Rom splines derived

In building Catmull-Rom splines, our goal is to take a sequence of control points P_0, P_1, \dots, P_n and build a piecewise cubic curve that passes through each point. One approach is to observe that the piece of curve between P_2 and P_3 is controlled

by P_1, P_2, P_3 and P_4 , so we should just compute it directly from those four points. To be explicit: we seek a curve γ with

$$\gamma(0) = P_2 \quad (22.31W)$$

$$\gamma(1) = P_3 \quad (22.32W)$$

$$\gamma'(0) = \frac{1}{2}(P_3 - P_1) \quad (22.33W)$$

$$\gamma'(1) = \frac{1}{2}(P_4 - P_2). \quad (22.34W)$$

The factors of $\frac{1}{2}$ on the tangents are there to ensure that if $P_i = (i, 0)$ on the x -axis, then the resulting curve will be a constant speed curve from P_2 to P_3 , i.e., will be $\gamma(t) = t + 2$.

Writing

$$\gamma(t) = p_1(t)P_1 + p_2(t)P_2 + p_3(t)P_3 + p_4(t)P_4, \quad (22.35W)$$

we can determine the polynomials p , just as we did for the Hermite curve. The results are

$$p_1(t) = \frac{1}{2}(-t^3 + 2t^2 - t)$$

$$p_2(t) = \frac{1}{2}(3t^3 - 5t^2 + 2)$$

$$p_3(t) = \frac{1}{2}(-3t^3 + 4t^2 + t)$$

$$p_4(t) = \frac{1}{2}(t^3 - t^2)$$

These are the four **Catmull-Rom basis functions** (see Figure 22W.10).

In matrix form, the Catmull-Rom spline segment for the four points is

$$\gamma_{CR}(t) = [P_1; P_2; P_3; P_4] \frac{1}{2} \begin{bmatrix} 0 & -1 & 2 & -1 \\ 2 & 0 & -5 & 3 \\ 0 & 1 & 4 & -3 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}. \quad (22.36W)$$

To handle the ends with such a matrix form, we would have to either (a) add new control points P_{-1} and P_{n+1} (see the exercises), or (b) derive separate matrices that conform to our endpoint interpolation conditions.

Each curve segment defined by Equation 22.36W is defined for $0 \leq t \leq 1$; if we want to have $0 \leq t \leq 1$ correspond to the segment from P_0 to P_1 , and $1 \leq t \leq 2$ correspond to the segment from P_1 to P_2 , and so on, we have to adjust things a little. In inefficient pseudocode, evaluation might look like this:

```

1
2 public CR(double t, Point[] Ps)
3 {
4     n = Ps.size(); // how many control points?
5     i = Math.floor(t); // if t = 3.3, we want to use P2, P3,
6                       // P4, and P5
7     ...test that 0 ≤ t ≤ n, and if t is in the first or
                       // lastsegment, handle special cases...

```

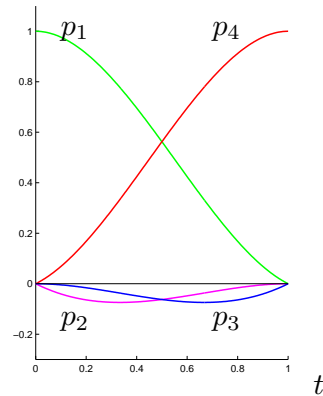


Figure 22W.10: The Catmull-Rom basis functions. The second and third have value 1 at $t = 0$ and $t = 1$, respectively, indicating that the curve goes from the second to the third control point.

```

8 | ...but the general case is this:
9 | return CRSegment(t-i, P[i-1], P[i], P[i+1], P[i+2])
10 | }
11 |
12 | private CRSegment(t, P1, P2, P3, P4)
13 | {
14 |     double p1 = 0.5 * (-t^3 + 2*t^2 - t);
15 |     ... similarly evaluate p2, p3, p4 ...
16 |     return p1 * P1 + p2 * P2 + p3 * P3 + p4 * P4;
17 | }

```

Listing 22W.3: Nonoptimized pseudocode for evaluating a Catmull-Rom spline.

In practice, the expression returned by `CRsegment` would need to be expressed as an affine combination, because neither scalar multiples of points nor sums of points are defined. (It *can* be expressed as an affine combination because the sum of the four polynomials is exactly 1, as you can verify.) Furthermore, the actual computation of the values for the polynomials would be done by matrix multiplications, as in Figure 22W.1. But the key idea is that to compute a point on the spline where $t = 3.3$, for instance, we use the fractional part of t (i.e., 0.3) as the parameter value for a Catmull-Rom segment.

Let's look at what that code says mathematically: it tells us that we're defining a function $\gamma : [0, n] \rightarrow \mathbf{R}^2$ with the rule

$$\gamma_t = \begin{cases} \dots & 0 \leq t \leq 1, \\ \dots & 1 \leq t \leq 2, \\ \vdots & \vdots \\ p_1(t-i)P_{i-1} + p_2(t-i)P_i + p_3(t-i)P_{i+1} + p_4(t-i)P_{i+2} & i \leq t \leq i+1 \\ \vdots & \vdots \\ \dots & n-1 \leq t \leq n. \end{cases} \quad (22.37W)$$

Let's look at just those cases of this expression in which P_4 appears:

$$\gamma_t = \begin{cases} \vdots \\ p_1(t-2)P_1 + p_2(t-2)P_2 + p_3(t-2)P_3 + \underbrace{p_4(t-2)P_4}_{\text{appears}} & 2 \leq t \leq 3 \\ p_1(t-3)P_2 + p_2(t-3)P_3 + \underbrace{p_3(t-3)P_4 + p_4(t-3)P_5}_{\text{appears}} & 3 \leq t \leq 4 \\ p_1(t-4)P_3 + \underbrace{p_2(t-4)P_4 + p_3(t-4)P_5 + p_4(t-4)P_6}_{\text{appears}} & 4 \leq t \leq 5 \\ \underbrace{p_1(t-5)P_4 + p_2(t-5)P_5 + p_3(t-5)P_6 + p_4(t-5)P_7}_{\text{appears}} & 5 \leq t \leq 6 \\ \vdots \end{cases} \quad (22.38W)$$

What is P_4 multiplied by? In the aggregate, it's multiplied by the function

$$c(t) = \begin{cases} \vdots \\ 0 & t < 2 \\ p_4(t-2) & 2 \leq t \leq 3 \\ p_3(t-3) & 3 \leq t \leq 4 \\ p_2(t-4) & 4 \leq t \leq 5 \\ p_1(t-5) & 5 \leq t \leq 6 \\ 0 & 6 < t \\ \vdots \end{cases} \quad (22.39W)$$

In general (i.e., excluding points near the start and finish), the point P_i is multiplied by a function very similar to this: it's zero except in the range $i-2 < t < i+2$, and on that range, it's defined by shifted copies of p_1, p_2, p_3 and p_4 . In fact, if we

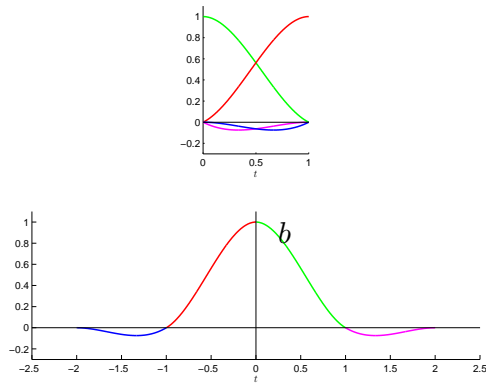


Figure 22W.11: The four Catmull-Rom basis functions, plotted on a single coordinate system, and then shifted and assembled to form the function b defined on the interval $[-2, 2]$. Because b is continuous and C^1 smooth, so is the Catmull-Rom spline. Because $b(0) = 1$, while $b(i) = 0$ for all other integers i , the Catmull-Rom spline is interpolating.

write

$$b(t) = \begin{cases} \vdots & \\ \vdots & \\ 0 & t < -2 \\ p_4(t+2) & -2 \leq t \leq -1 \\ p_3(t+1) & -1 \leq t \leq 0 \\ p_2(t) & 0 \leq t \leq 1 \\ p_1(t-1) & 1 \leq t \leq 2 \\ 0 & 2 < t \\ \vdots & \end{cases}, \quad (22.40W)$$

then we can say that the point P_i is multiplied by $b(t-i)$. The function b is assembled from the four Catmull-Rom basis functions, as shown in Figure 22W.11.


It's much easier to understand the properties of the Catmull-Rom spline by looking at b than at the individual basis polynomials p_1, \dots, p_4 . For instance, since b is clearly continuous, the Catmull-Rom spline is continuous. And because b has continuous derivative, the Catmull-Rom spline is C^1 . And because $b(i) = 0$ for

every integer i except 0, and $b(0) = 1$, we can see that at time $t = i$, the Catmull-Rom spline will pass exactly through P_i , i.e., it's interpolating.

In fact, if we continue to ignore the endpoints, we can write the Catmull-Rom spline in the form

$$\gamma_{CR}(t) = \sum_{i=0}^n P_i b(t - i). \quad (22.41W)$$

For any particular value of t , at most four terms in the sum will be nonzero, and the others needn't be evaluated. But the main value of this formulation of the Catmull-Rom spline is that we can see it as a weighted sum of the control points, where the weighting functions are all translated copies of one particular function, whose properties determine the shape of the spline overall.

 The summation form given for the Catmull-Rom spline is a discrete-continuous convolution, just like the ones we saw in Chapter 18. All the machinery of convolution can be brought to bear on the study of splines in general.

22.7W Representing curves by matrices

As we saw in the case of the Bézier and Hermite curves, we can express a polynomial curve in the form

$$\gamma(t) = \mathbf{GMT}(t), \quad (22.42W)$$

where \mathbf{G} , the *geometry matrix*, is a matrix whose columns store the coordinates of the geometric controls for the curve; four points, in the case of Bézier curves, or two points and two vectors in the case of Hermite curves, \mathbf{M} , the *basis matrix*, stores the coefficients for the blending polynomials, one polynomial per row, and $\mathbf{T}(t)$ is the vector with entries $1, t, t^2$, and t^3 . Using this form, it's easy to check the conditions for a Bézier curve, defined by points P_1, \dots, P_4 and an Hermite curve, defined by its starting and ending points A and B , and its starting and ending vectors \mathbf{v} , and \mathbf{w} , to be identical. Using the subscripts H and B for Hermite and Bézier, respectively, we want to have

$$\mathbf{G}_B \mathbf{M}_B \mathbf{T}(t) = \mathbf{G}_H \mathbf{M}_H \mathbf{T}(t) \quad (22.43W)$$

for every value of t . This will happen (see the exercises) if and only if

$$\mathbf{G}_B \mathbf{M}_B = \mathbf{G}_H \mathbf{M}_H. \quad (22.44W)$$

i.e., if


$$\mathbf{G}_B = \mathbf{G}_H \mathbf{M}_H \mathbf{M}_B^{-1}. \quad (22.45W)$$

Writing that in terms of the individual controls, and recalling the basis matrices, this becomes

$$[P_1; P_2; P_3; P_4] = [A; B; \mathbf{v}; \mathbf{w}] \cdot \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \quad (22.46W)$$

$$= [A; B; \mathbf{v}; \mathbf{w}] \cdot \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & -\frac{1}{3} & 0 \end{bmatrix}. \quad (22.47W)$$

Reading that last matrix column by column, it says that the first control point P_1 for the Bézier curve is the first control point A for the Hermite curve; the second control point P_2 of the Bézier curve is $A + \frac{1}{3}\mathbf{v}$; the third Bézier control point P_3 is $B - \frac{1}{3}\mathbf{w}$, and the last Bézier control point P_4 is the second Hermite control point, B .

 The expression $\gamma(t) = \mathbf{GMT}(t)$, can be read in several ways. If we multiply \mathbf{GM} first, the resulting matrix linearly transforms a cubic curve in 4-space $(1, t, t^2, t^3)$ into our cubic in the plane. On the other hand, if we first multiply $\mathbf{MT}(t)$, we get the four polynomials that are used in combining the control data. Thus the spline curve can be seen as a linear combination (where the coefficients are *points* or *vectors*) of certain polynomials. Because by varying the control data we can get all possible cubic curve segments, these four polynomials constitute a *basis* for the set of degree three polynomials. (This can be seen, equivalently, by noting that \mathbf{M} , in all the cases we'll study, is invertible.) In the case of the Bézier spline, the four rows of \mathbf{M} sum to one. That tells us (see the exercises) that if we move each control point by the same amount \mathbf{u} , the Bézier curve for the new control points will be exactly the Bézier curve for the original points, translated by \mathbf{u} .

22.8W Summary so far

At this point you know how to create a curve segment (Hermite) starting from one point and direction and going to another point and direction (or from one point to another, passing nearby two intermediate points – a Bézier curve), and how to assemble these into a C^1 spline (the Catmull-Rom spline); you can even build a basic drawing program with a click-the-control-points mechanism for making Catmull-Rom splines (which can then have their control points or tangent vectors adjusted to change the shape, after which they are no longer Catmull-Rom splines, of course). You've also seen that the curve segments in a spline can be expressed in terms of matrices, and that piecewise cubics are a natural choice for the curve segments because they can be glued together nicely and can have torsion.

22.9W B-splines

We will now generalize the previous approach — the expression of a spline in the form $\sum P_i b(t - i)$ — by considering other possible functions b . By choosing

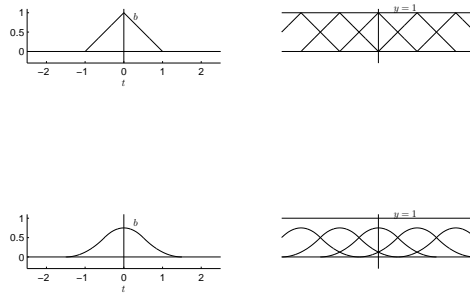


Figure 22W.12: Each function b shown at left is suitable for building a spline because copies of b , translated by integer amounts, shown on the right, sum to the constant function 1.

some particularly nice functions to play the role of b , we can ensure some very nice properties of the resulting spline. For instance,

- if the function b is nonzero only on a small interval, then the sum, for each t , will in practice consist of only a few nonzero terms;
- if the function is smooth, then the spline will be smooth, and
- if the spline is to interpolate its control points, then the function must satisfy $b(0) = 1$ and $b(i) = 0$ for integers $i \neq 0$.

There are also constraints on the function b : it's important that the sum $\sum P_i b(t-i)$ be an affine combination of the points P_i ; otherwise the spline will not be translation-equivariant. To say that the sum is an affine combination (regardless of t) is to say that $\sum b(t-i) = 1$ for every t ; pictorially, if we take many copies of the graph of b and translate each by an integer amount on the x -axis, the sum of the resulting graphs must be the constant function 1, as shown in Figure 22W.12.

If the function b , in addition to satisfying this constraint, also happens to take on values between 0 and 1 (i.e., $b(t)$ is never negative, and never greater than one), then the expression $\sum P_i b(t-i)$, for a particular value of t , is not only an *affine* combination of just a few of the control points, it's a *convex* combination of them, and the resulting point must lie in the convex hull of those control points.

22.9.1W Coordinate-wise constructions

When we wrote down the formula for a Bézier curve:

$$\gamma(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4, \quad (22.48W)$$

we combined the points $P_1 = (x_1, y_1), \dots, P_4 = (x_4, y_4)$ with weights determined by four polynomials. In doing so, we performed essentially the same computation for each coordinate. The x -coordinate of $\gamma(t)$, for instance, is

$$\gamma(t)_x = (1-t)^3 x_1 + 3t(1-t)^2 x_2 + 3t^2(1-t) x_3 + t^3 x_4, \quad (22.49W)$$

while the y -coordinate is

$$\gamma(t)_y = (1-t)^3 y_1 + 3t(1-t)^2 y_2 + 3t^2(1-t) y_3 + t^3 y_4. \quad (22.50W)$$

Instead of formulating the Bézier curve problem as “Given four points, build a cubic curve going from P_1 to P_4 , with initial and final tangents determined by $3(P_2 - P_1)$ and $3(P_4 - P_3)$,” we could have said “Given four values v_1, \dots, v_4 , build a cubic-polynomial combination $h(t) = p_1(t)v_1 + \dots + p_4(t)v_4$ with the property that $h(0) = v_1$, $h(1) = v_4$, $h'(0) = 3(v_2 - v_1)$, and $h'(1) = 3(v_4 - v_3)$.” The polynomials p_1, \dots, p_4 that solve this problem would then also solve the original problem, when used coordinate-wise to blend control points rather than individual values.

This idea — that we can go from a several-coordinate interpolation problem to a single-coordinate problem and back — is a powerful one. It also can simplify notation in some cases, which is valuable: in building splines, we have sets of basis polynomials, their individual coefficients, control-point lists, and coordinates of individual control points, each of which can require an index; it’s easy to get lost in a sea of indices. Removing even a single index is a useful improvement.

22.9.2W Piecewise-Constant splines

The simplest B-spline is the piecewise-constant B-spline. It’s defined by a sequence P_0, P_1, \dots, P_n of control points (we’ll often denote such a sequence by $\{P_i\}_0^n$ or simply $\{P_i\}$, when the limits are clear), and has the form

$$\gamma(t) = \sum_{i=0}^n b_0(t-i) P_i, \quad (22.51W)$$

where b_0 is the bump function defined by

$$b_0(t) = \begin{cases} 1 & -0.5 \leq t < 0.5 \\ 0 & \text{otherwise} \end{cases}, \quad (22.52W)$$

which we saw frequently in Chapter 18. A parametric plot of γ consists of just the points $\{P_i\}$, because those are the only values taken on by γ . As mentioned in Section 22.9.1W, we can think of such a spline one coordinate at a time. If the y -coordinate of the point P_i is y_i , then the y -coordinate of the curve γ is just

$$\gamma_y(t) = \sum_{i=0}^n b_0(t-i) y_i. \quad (22.53W)$$

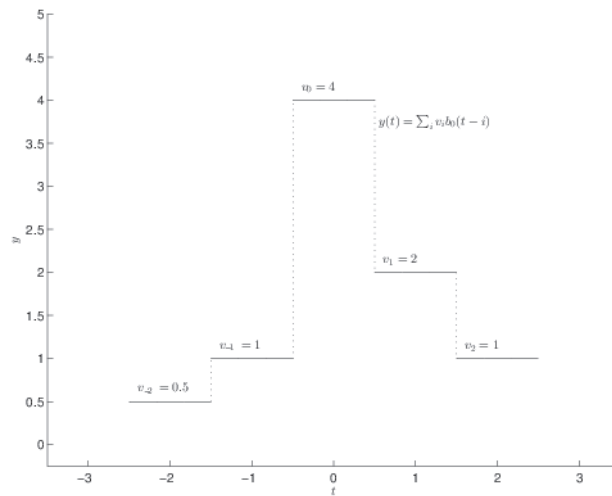


Figure 22W.13: The control values are at $y = \dots, 1, 4, 2, \dots$. The degree zero B-spline on these control values can be graphed (rather than parametrically plotted); the graph is constant on unit length intervals.

The graph of γ_y is step-like (see Figure 22W.13).

Higher-order splines will arise from repeated convolutions of Equation 22.51W with b_0 . This is the approach taken by I.J. Schoenberg [19] in his seminal work on the subject. But modern approaches use a slightly different function: instead of using a unit-width bump centered on the origin, they use a unit-width bump that's nonzero on $[0, 1]$, i.e., they define

$$b_0(t) = \begin{cases} 1 & 0 \leq t < 1 \\ 0 & \text{otherwise} \end{cases}. \quad (22.54W)$$

To be consistent with these approaches, we'll use this latter definition of b_0 for the remainder of this chapter, and will redefine b_1, b_2 , etc., accordingly. Doing so means that the graph in Figure 22W.13 gets shifted, as shown in Figure 22W.14.

22.9.3W Linear splines

The formula for the degree-zero B-spline was

$$\sum_{i=0}^n b_0(t-i)P_i, \quad (22.55W)$$

which makes sense for $0 \leq t \leq n$, in the sense that outside of this domain, the sum in Equation 22.55W will no longer be a convex combination of the control points.

Equation 22.55W can be seen as a discrete-continuous convolution between a discrete “signal” — the sequence of P s — and a continuum signal, b_0 . Convoluting again with b_0 should “smooth out” the degree-zero B-spline, and indeed, it does

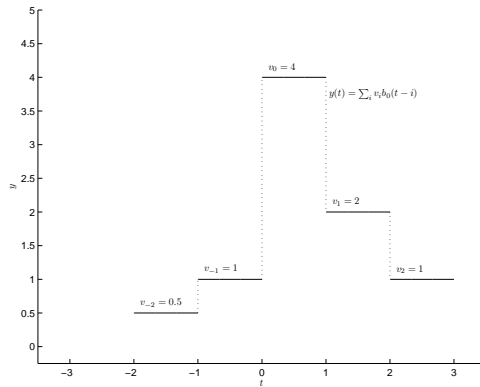


Figure 22W.14: The graph of Figure 22W.13, recomputed using the bump-function that's nonzero on $[0, 1]$ instead of $[-\frac{1}{2}, \frac{1}{2}]$.

so. If we consider the curve defined by

$$\gamma(t) = b_0 \star \sum_{i=0}^n b_0(t-i)P_i, \tag{22.56W}$$

$$\gamma(t) = \sum_{i=0}^n (b_0 \star b_0)(t-i)P_i, \tag{22.57W}$$

we discover that it's just the piecewise-linear path that we called a “connect-the-dots” curve in Chapter 9. We can see this by explicitly computing the convolution of b_0 with itself; the result is the function

$$b_1(t) = \begin{cases} t & 0 \leq t \leq 1 \\ 2-t & 1 \leq t \leq 2 \end{cases} = \begin{cases} t & 0 \leq t \leq 1 \\ 1-(t-1) & 1 \leq t \leq 2 \end{cases}, \tag{22.58W}$$

as we saw in Chapter 18. We've written the expressions so that the i th case is presented as a function of $t-i$, which we'll do for all the B-splines.

Having defined b_1 , we can define the degree-one B-spline by

$$\gamma(t) = \sum_{i=0}^n b_1(t-i)P_i. \tag{22.59W}$$

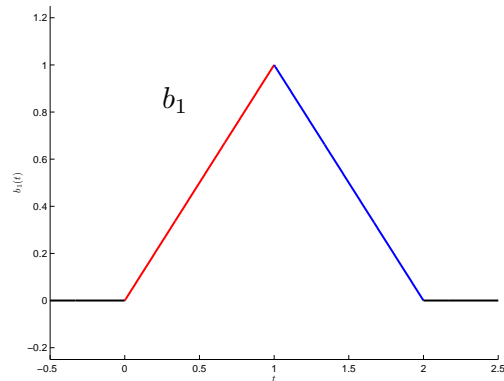


Figure 22W.15: The degree one B-spline b_1 is just a tent function with width two and height one. Properties of this function help us understand properties of degree-one B-splines in general.

Inline Exercise 22W.11: Explain why the expression in Equation 22.59W only represents a convex combination of the control points for $1 \leq t \leq n$. In general, for degree- k B-splines as we'll formulate them, the domain of definition is $k \leq t \leq n$.

The properties of γ can be inferred from those of b_1 . To reinforce the point: by looking at the plot of the degree-one B-spline function b_1 (Figure 22W.15), we can determine properties that degree-one B-splines must have in general (although we can also determine these more directly, just by looking at the connect-the-dots plot).

For instance, because $b_1(t)$ is zero outside of the interval $0 < t < 2$, we can conclude that in the sum

$$\gamma(t) = \sum_i v_i b_1(t - i), \quad (22.60W)$$

for each particular value of t , there are at most two nonzero terms in the sum (the ones at $i = \lfloor t \rfloor$ and $i = \lceil t \rceil$). So each point of the curve defined by γ is influenced by the positions of at most two control points. We'll see shortly that it's actually a convex combination of the two.

Inline Exercise 22W.12: In Chapter 18, we convolved the box function centered at 0 – let’s call that b – with itself, and got a tent function centered at 0: $t + 1$ for $-1 \leq t \leq 0$ and $1 - t$ for $0 \leq t \leq 1$. Here we’ve shifted b to the right by one half unit, so that $b_0(t) = b(t - 1/2)$. The convolution $b_0 \star b_0$ ends up being $b \star b$ shifted right by a whole unit. This is explained by a general rule: shifting either factor in a convolution by an amount k shifts the convolution by k as well. So shifting both halves by $\frac{1}{2}$ shifts the convolution by 1. (a) Suppose that $f : \mathbf{R} \rightarrow \mathbf{R}$ and $h : \mathbf{R} \rightarrow \mathbf{R}$ are functions with $f(x) = h(x - k)$ and $g : \mathbf{R} \rightarrow \mathbf{R}$ is another function. Let $F = f \star g$ and $H = h \star g$. Use the definition of convolution to write out $F(x)$ and $H(x - k)$. (b) Use substitution in the integral to show that the two are equal.

The function b_1 is defined by two polynomial pieces, each defined on a unit-length interval. When we move to quadratic and cubic B-splines, we’ll get functions b_2 and b_3 made of three and four pieces, respectively, and we’ll want to refer to those individual pieces. We’ll define

$$b_{1,0}(t) = t \quad \text{for } 0 \leq t \leq 1, \text{ and} \quad (22.61W)$$

$$b_{1,1}(t) = 1 - t \quad \text{for } 0 \leq t \leq 1, \quad (22.62W)$$

so that

$$b_1(t) = \begin{cases} b_{1,0}(t) & 0 \leq t \leq 1 \\ b_{1,1}(t - 1) & 1 \leq t \leq 2 \\ 0 & \text{otherwise,} \end{cases} \quad (22.63W)$$

as shown in Figure 22W.16. To simplify later expressions, it’s convenient to have the $b_{i,k}$ defined outside the unit interval as well; we define them to be zero there. With that definition, we can say

$$b_1(t) = b_{1,0}(t) + b_{1,1}(t - 1). \quad (22.64W)$$

Notice that

- $b_{k,i}$ describes the part of b_k that’s defined on the interval from i to $i + 1$.
- $b_{k,i}(t)$ makes sense for $0 \leq t \leq 1$, so in b_i , it appears in the form $b_{k,i}(t - i)$.

To be consistent, we define $b_{0,0}(t) = 1$ for $0 \leq t \leq 1$.

The linear B-spline with control points $\{P_i\}$ is defined by

$$\gamma(t) = \sum_{i=0}^n b_1(t - i)P_i, \quad (22.65W)$$

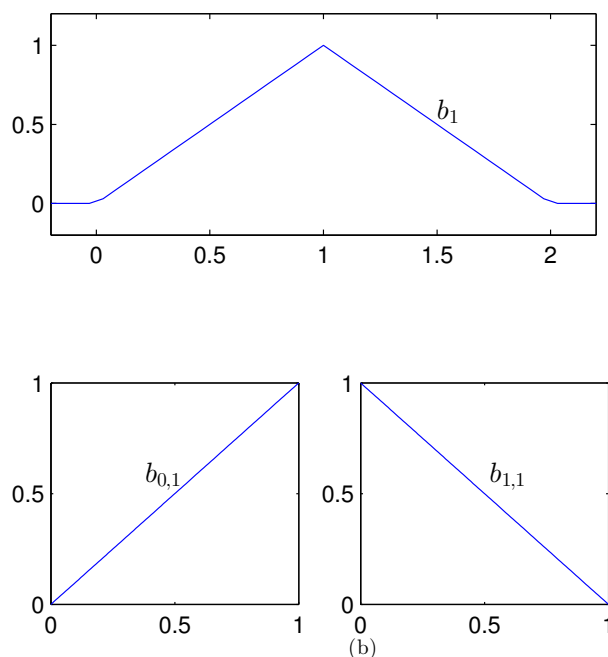


Figure 22W.16: (a) The tent function b_1 , used in defining the linear B-spline. (b) The functions $b_{1,0}$ and $b_{1,1}$, each defined on the unit interval, which together can be used to build b_1 .

but it can be rewritten. Suppose there are three control points. Then we have

$$\begin{aligned}
 \gamma(t) &= \sum_{i=0}^3 b_1(t-i)P_i \\
 &= b_1(t)P_0 + b_1(t-1)P_1 + b_1(t-2)P_2 \\
 &= (b_{1,0}(t) + b_{1,1}(t-1))P_0 + \\
 &\quad (b_{1,0}(t-1) + b_{1,1}(t-2))P_1 + \\
 &\quad (b_{1,0}(t-2) + b_{1,1}(t-3))P_2 \\
 &= b_{1,0}(t)P_0 + \\
 &\quad [b_{1,1}(t-1)P_0 + b_{1,0}(t-1)P_1] + \\
 &\quad [b_{1,1}(t-2)P_1 + b_{1,0}(t-2)P_2] + \\
 &\quad b_{1,1}(t-3)P_2.
 \end{aligned}$$

More generally, this has the form

$$\gamma(t) = b_{1,0}(t)P_0 + \left(\sum_{i=1}^{n-1} b_{1,0}(t-i)P_i + b_{1,1}(t-i)P_{i-1} \right) + b_{1,1}(t-n)P_n. \quad (22.66W)$$

Each term in the summation is either a convex combination of two adjacent control points (when $i < t < i+1$) or is zero. (The terms at the ends are special cases; if our spline is defined by an infinite sequence of control points, then the end cases

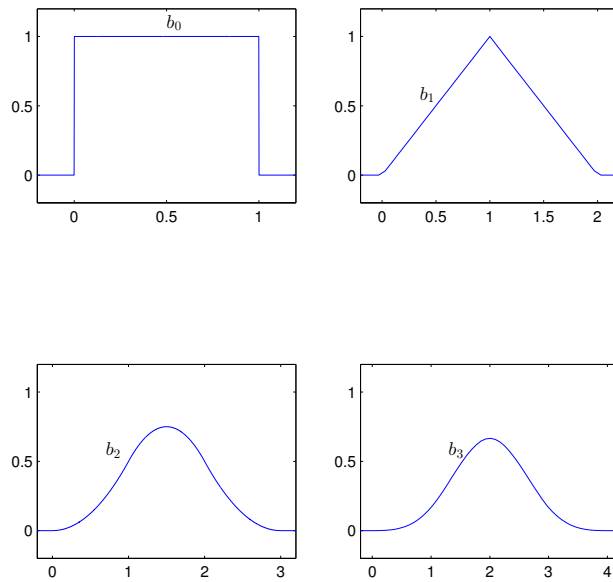


Figure 22W.17: The basis functions for various degrees and their supports; the function of degree d has support $[0, d + 1]$.

disappear.) (This form also makes clear the answer to the second-to-last inline exercise.)

Higher degree splines

The generalization to higher degrees is now straightforward: we can convolve with b_0 repeatedly to get a piecewise-degree- k polynomial curve f_k whose shape is controlled by the values $\{v_i\}$. A typical segment of this curve will be influenced by $k + 1$ control values, and a typical control value will influence the shape of the curve on an interval of length $k + 1$. This is explicitly carried out for quadratic and cubic splines in Section 22.9.4W.

Just as before, the characteristics of the function b_k become characteristics of f . For instance,

- Because b_k is $k - 1$ times differentiable, with continuous derivatives, f_k is $k - 1$ times differentiable, with continuous derivatives.

At the same time, the width of the support of b_k is k ; this means that

- for each value of t , at most k of the P_i s have nonzero coefficients in $f_k(x; \{P_i\})$.

Furthermore, because the values taken on by b_0 are between zero and one, and its integral is 1.0, the values taken on by b_1 lie between 0 and 1 (inclusive), and the same is true for b_2 , b_3 , and so on. Finally, because the translates of b_0 sum to the constant function one, i.e.

$$\sum_i b_0(x - i) = 1 \text{ for all } x, \quad (22.67W)$$

and because convolving b_0 with the constant function 1 results in the constant function 1, we get that

$$\sum_i b_1(x - i) = 1 \text{ for all } x, \quad (22.68W)$$

and similarly for b_2, b_3 , and so on. As a consequence, we know

- The function f_k , evaluated at t , results in a value that is a combination of at most $k + 1$ of the control values (i.e., at most $k + 1$ control values have nonzero coefficients); the sum of the coefficients of these values is one.
- In consequence, the value $f_k(x)$ lies in the convex hull of these $k + 1$ control values.

The curves f_k are called the **uniform B-splines of order k** . In many ways they are the easiest splines to understand, in part because of the repeated-convolution form of their definition. Each of the statements above, about “values,” can be converted to a geometric statement by replacing “value” with “point”: each point on a cubic B-spline, for instance, is computed as a convex combination of four of the control points for the spline.

From a practical point of view, the convolution definition is not terribly useful; one would not want to have to compute multiple integrals to find one point on a spline curve. Fortunately, one need only compute the functions b_0, b_1, b_2, \dots once, and one can do this explicitly (the functions that one must integrate can be integrated in elementary terms, and the results are in fact polynomials). Furthermore, the integration process is so regular that one can deduce an algebraic pattern.

But if one does not know in advance what degree spline one might use, there’s still a relatively simple way to compute points on the curve: there happens to be a recursive formula, due to DeBoor [6] for evaluating B-splines.

22.9.4W Quadratic and Cubic Splines

We’ve seen constant (degree-zero) and linear (degree-one) B-splines in detail. The next two forms — quadratic and cubic — are also very useful. Quadratic B-splines are C^1 -smooth, but no longer interpolate their control points; cubics are C^2 -smooth, but come even less close to their control points. For each degree from zero to three, we’ll show (1) the basis functions, using the interval $[0, 1]$ as the support for b_0 , (2) the matrix form for computing a spline segment, and (3) the basis functions assembled into a single function, translated copies of which serve as the coefficients of each control point.

For the degree-zero spline, the basis function is the box

$$b_0(t) = \begin{cases} 1 & 0 \leq t < 1 \\ 0 & \text{otherwise} \end{cases}. \quad (22.69W)$$

The domain of definition is $0 \leq t \leq n$, where n is the number of control points.

The matrix form for the i th segment (t between i and $i + 1$) is

$$\gamma(t) = \mathbf{G} \cdot \mathbf{M} \cdot \mathbf{T}_0(t - i), \quad i \leq t < i + 1 \quad (22.70W)$$

where the geometry matrix \mathbf{G} has just a single column,

$$\mathbf{G} = [P_i], \quad (22.71W)$$

the basis matrix \mathbf{M} is 1×1 :

$$\mathbf{M} = [1], \quad (22.72W)$$

and the \mathbf{T} vector has only t to the zero power:

$$\mathbf{T}(t) = [1]. \quad (22.73W)$$

The linear (degree-one) B-spline is somewhat more interesting. The function b_1 function is the tent:

$$b_1(t) = \begin{cases} t & 0 \leq t \leq 1 \\ 2-t & 1 \leq t \leq 2 \\ 0 & \text{otherwise} \end{cases} = \begin{cases} t & 0 \leq t \leq 1 \\ 1-(t-1) & 1 \leq t \leq 2 \\ 0 & \text{otherwise} \end{cases}. \quad (22.74W)$$

The domain of definition is $1 \leq t \leq n$, where n is the number of control points. The matrix form for the i th segment (t between i and $i+1$) is

$$\gamma(t) = \mathbf{G} \cdot \mathbf{M} \cdot \mathbf{T}_1(t-i), \quad i \leq t < i+1 \quad (22.75W)$$

where the geometry matrix \mathbf{G} has two columns,

$$\mathbf{G} = [P_i; P_{i-1}], \quad (22.76W)$$

the basis matrix \mathbf{M} is 2×2 :

$$\mathbf{M} = \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix}, \quad (22.77W)$$

and the $\mathbf{T}(t)$ vector has t to two powers:

$$\mathbf{T}(t) = \begin{bmatrix} 1 \\ t \end{bmatrix}. \quad (22.78W)$$

Inline Exercise 22W.13: It may surprise you that the control points appear in decreasing order. Verify for yourself that $\gamma(1.75)$ really does turn out to be $\frac{1}{4}P_0 + \frac{3}{4}P_1$ by explicitly evaluating Equation 22.75W at $t = 1.75$.

The quadratic (degree-two) B-spline is still more interesting. The function b_2 is the convolution of the box and the tent:

$$b_2(t) = \begin{cases} \frac{1}{2}t^2 & 0 \leq t \leq 1 \\ \frac{1}{2}(-2t^2 + 6t - 3) & 1 \leq t \leq 2 \\ \frac{1}{2}(3-t)^2 & 2 \leq t \leq 3 \\ 0 & \text{otherwise} \end{cases} = \begin{cases} \frac{1}{2}t^2 & 0 \leq t \leq 1 \\ \frac{1}{2}(-2(t-1)^2 + 2(t-1) + 1) & 1 \leq t \leq 2 \\ \frac{1}{2}(1-(t-2))^2 & 2 \leq t \leq 3 \\ 0 & \text{otherwise} \end{cases}. \quad (22.79W)$$

The domain of definition is $2 \leq t \leq n$, where n is the number of control points. The matrix form for the i th segment (t between i and $i+1$) is

$$\gamma(t) = \mathbf{G} \cdot \mathbf{M} \cdot \mathbf{T}_1(t-i), \quad i \leq t < i+1 \quad (22.80W)$$

where the geometry matrix \mathbf{G} has three columns,

$$\mathbf{G} = [P_i; P_{i-1}; P_{i-2}], \quad (22.81W)$$

the basis matrix \mathbf{M} is 3×3 :

$$\mathbf{M} = \frac{1}{2} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 2 & -2 \\ 1 & -2 & 1 \end{bmatrix}, \quad (22.82W)$$

and the $\mathbf{T}(T)$ vector has t to three powers:

$$\mathbf{T}(t) = \begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix}. \quad (22.83W)$$

Finally, for the uniform cubic B-spline, the basis function has support of width 4:

$$b_3(t) = \frac{1}{6} \begin{cases} t^3 & 0 \leq t \leq 1 \\ -3(t-1)^3 + 3(t-1)^2 + 3(t-1) + 1 & 1 \leq t \leq 2 \\ 3(t-2)^3 - 6(t-2)^2 + 4 & 2 \leq t \leq 3 \\ (1-(t-3))^3 & 3 \leq t \leq 4 \\ 0 & \text{otherwise} \end{cases}. \quad (22.84W)$$

The domain of definition is $3 \leq t \leq n$, where n is the number of control points. The matrix form for the i th segment (t between i and $i+1$) is

$$\gamma(t) = \mathbf{G} \cdot \mathbf{M} \cdot \mathbf{T}_1(t-i), \quad i \leq t < i+1 \quad (22.85W)$$

where the geometry matrix \mathbf{G} has four columns,

$$\mathbf{G} = [P_i; P_{i-1}; P_{i-2}; P_{i-3}], \quad (22.86W)$$

the basis matrix \mathbf{M} is 4×4 :

$$\mathbf{M} = \frac{1}{6} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 3 & 3 & -3 \\ 4 & 0 & -6 & 3 \\ 1 & -3 & 3 & -1 \end{bmatrix}, \quad (22.87W)$$

and the $\mathbf{T}(T)$ vector has t to four powers:

$$\mathbf{T}(t) = \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}. \quad (22.88W)$$

22.9.5W Refinement

Figure 22W.18 shows an interesting property of B-splines: if we know the control points for a spline γ , we can construct control points for a curve that traverses the first half of γ ! (We can, of course, also find one that traverses the second half.) The curve segment defined by the four points has domain $2 \leq t \leq 3$.

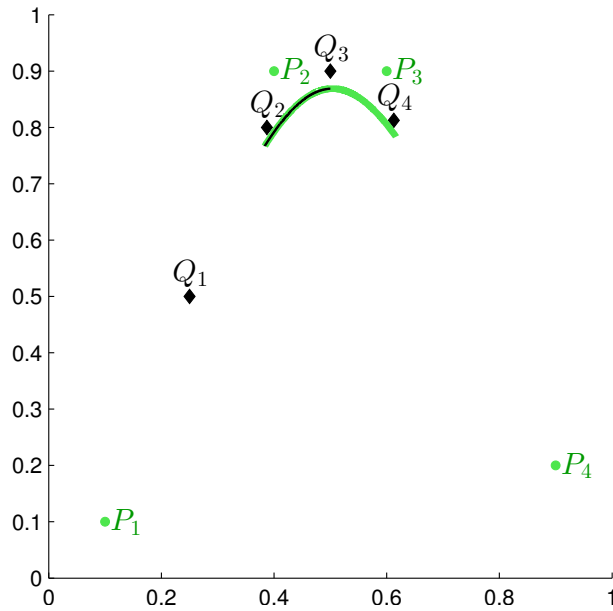


Figure 22W.18: The cubic B-spline γ , drawn in a thick green line, is defined by the control points P_0, \dots, P_3 . The cubic B-spline β , drawn in a thin black line, is defined by the control points Q_0, \dots, Q_3 . These have been chosen so that the entire β curve matches the first half of the γ curve, i.e., so that $\beta(2+t) = \gamma(2+t/2)$ for $t \in [0, 1]$.

Give the control points P_i ($i = 1, \dots, 4$), we find the control points Q_i by simply writing down what it means for $\beta(2+t) = \gamma(2+t/2)$. The matrix form for this uses the fractional part, f , of t , which lies between 0 and 1; for the two expressions to be equal, we need

$$[Q_3; Q_2; Q_1; Q_0] \cdot \mathbf{M} \cdot \mathbf{T}(2f) = [P_3; P_2; P_1; P_0] \cdot \mathbf{M} \cdot \mathbf{T}(f). \quad (22.89W)$$

for every $f \in [0, \frac{1}{2}]$.

Here \mathbf{M} is the B-spline basis matrix; to simplify matters, we'll use \mathbf{P} and \mathbf{Q} to denote the two geometry matrices, so the condition is now

$$\mathbf{Q} \cdot \mathbf{M} \cdot \mathbf{T}(2f) = \mathbf{P} \cdot \mathbf{M} \cdot \mathbf{T}(f). \quad (22.90W)$$

Since $\mathbf{T}(2f) = \begin{bmatrix} 1 \\ 2f \\ 4f^2 \\ 8f^3 \end{bmatrix}$, we can write

$$\mathbf{T}(2f) = \mathbf{H}\mathbf{T}(f) \quad (22.91W)$$

$$= \begin{bmatrix} 1 & & & \\ & 2 & & \\ & & 4 & \\ & & & 8 \end{bmatrix} \mathbf{T}(f) \quad (22.92W)$$

Substituting, we get

$$\mathbf{Q} \cdot \mathbf{M} \cdot \mathbf{H} \cdot \mathbf{T}(f) = \mathbf{P} \cdot \mathbf{M} \cdot \mathbf{T}(f). \quad (22.93W)$$

Since this is true for every $f \in [0, \frac{1}{2}]$, the matrices must be equal (see Exercise 22W.8). Hence we can compute the coordinates for the Q points:

$$\mathbf{Q} \cdot \mathbf{M} \cdot \mathbf{H} = \mathbf{P} \cdot \mathbf{M} \quad (22.94W)$$

$$\mathbf{Q} = \mathbf{P} \cdot \mathbf{M} \cdot \mathbf{H}^{-1} \cdot \mathbf{M}^{-1}. \quad (22.95W)$$

Everything we've said so far applies to *all* the classes of splines we've discussed, since nothing special about the B -spline basis matrix has been used: you can *always* regenerate the first or second half of a spline segment by choosing new control points. Now let's compute explicitly:

$$\mathbf{Q} = \mathbf{P} \cdot \mathbf{M} \cdot \mathbf{H}^{-1} \cdot \mathbf{M}^{-1} \quad (22.96W)$$

$$= \mathbf{P} \cdot \frac{1}{6} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 3 & 3 & -3 \\ 4 & 0 & -6 & 3 \\ 1 & -3 & 3 & -1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & \frac{1}{2} & & \\ & & \frac{1}{4} & \\ & & & \frac{1}{8} \end{bmatrix} \left(\frac{1}{6} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 3 & 3 & -3 \\ 4 & 0 & -6 & 3 \\ 1 & -3 & 3 & -1 \end{bmatrix} \right)^{-1} \quad (22.97W)$$

$$= \mathbf{P} \cdot \frac{1}{8} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 6 & 4 & 1 & 0 \\ 1 & 4 & 6 & 4 \\ 0 & 0 & 1 & 4 \end{bmatrix}. \quad (22.98W)$$

This tells us that $Q_3 = \frac{1}{8}(P_1 + 6P_2 + P_3)$, $Q_2 = (P_1 + P_2)/2$, $Q_1 = \frac{1}{8}(P_0 + 6P_1 + P_2)$, and $Q_0 = (P_0 + P_1)/2$. (Remember that the Q s and P s appear in decreasing order!)

If we did an analogous computation for the second half of the curve, we'd find that the four control points were exactly Q_1, Q_2, Q_3 , and $(P_2 + P_3)/2$.

Thus if we had a B -spline defined by a sequence of points including P_0, \dots, P_3 , we could replace those four with $Q_0 = (P_0 + P_1)/2$, $Q_1 = \frac{1}{8}(P_0 + 6P_1 + P_2)$, $Q_2 = (P_1 + P_2)/2$, $Q_3 = \frac{1}{8}(P_1 + 6P_2 + P_3)$, and $Q_4 = (P_2 + P_3)/2$, and similarly for each other set of four sequential control points. Figure 22W.19 demonstrates this.

In the figure, the vertices of the new "refined" control polygon are closer to the B -spline than those of the original polygon, and it certainly appears that if we refined again and again, the control polygon's points would approach the B -spline itself. We'll show, presently, that this is indeed the case.

There's an alternative path to the same result (i.e., to finding a refined control polygon that defines the same curve). It's based on the observation that if we compress the cubic B -spline function b_3 by a factor of two, i.e., we define

$$c_3(t) = b_3(2t), \quad (22.99W)$$

then b_3 can be written as a weighted sum of translates of c_3 , as shown in Figure 22W.21. The details are given in the exercises, but the main idea is that a corresponding statement is certainly true for b_0 : the box function on $[0, 1]$ can certainly be written as a sum of two half-as-wide box functions, as Figure 22W.20 shows. Convoluting this equality with a half-width box gives a corresponding statement for b_1 ; convoluting twice more leads to the statement for b_3 .

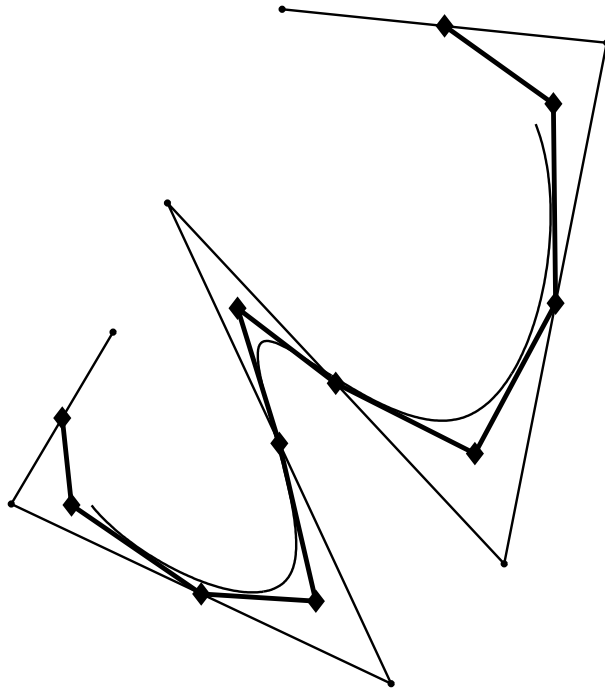


Figure 22W.19: A spline defined by a sequence of control points P_i , shown as a polygon with thin edges, and the refined set of control points, shown as a polygon with thick edges, that defines the same B-spline. The refined set includes midpoints of all the original control-polygon edges, and a $1/8 - 3/4 - 1/8$ blend of each original control point with its two neighbors. Thus the refined control polygon has about twice as many vertices as the original.

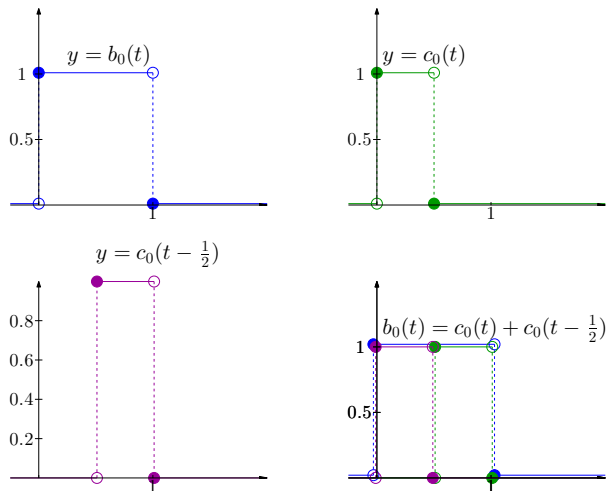


Figure 22W.20: The box function b_0 is a sum of two translated half-width box-functions. Letting $c_0(t) = b_0(2t)$, we have $b_0(t) = c_0(t) + c_0(t - \frac{1}{2})$.

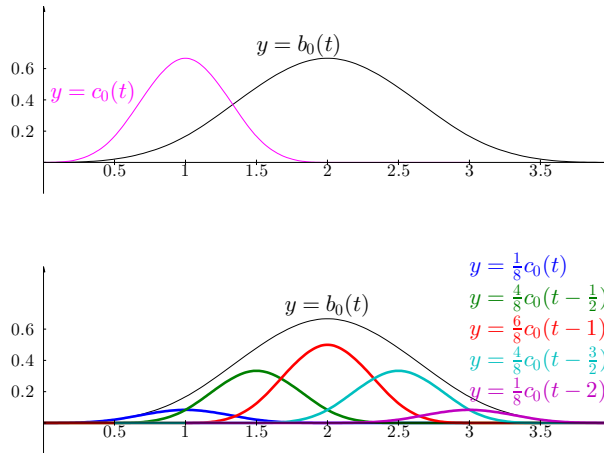


Figure 22W.21: The cubic B-spline basis function b_3 can be written as a weighted sum of five translated copies of the function $c_3(t) = b_3(2t)$; the five colored curves sum up to the thin black curve, b_3 .

That b_3 is a weighted sum of translates of c_3 (the weights are $1/8$ of 1, 4, 6, 4, and 1) implies the refinement formula for cubic splines; to see this, consider a spline

$$\gamma(t) = \sum P_i b_3(t - i). \tag{22.100W}$$

We'll write out a small portion of this sum, and then substitute the c_3 combination for the b_3 s:

$$\gamma(t) = \dots + P_0 b_3(t) + P_1 b_3(t - 1) + P_2 b_3(t - 2) + \dots \tag{22.101W}$$

$$\frac{1}{8} P_0 (c_3(t) + 4c_3(t - 1/2) + 6c_3(t - 1) + 4c_3(t - 3/2) + \underline{c_3(t - 2)}) + \tag{22.102W}$$

$$\frac{1}{8} P_1 (c_3(t - 1) + 4c_3(t - 3/2) + \underline{6c_3(t - 2)} + 4c_3(t - 5/2) + c_3(t - 3)) + \tag{22.103W}$$

$$\frac{1}{8} P_2 (\underline{c_3(t - 2)} + 4c_3(t - 5/2) + 6c_3(t - 3) + 4c_3(t - 7/2) + c_3(t - 4)) + \dots \tag{22.104W}$$

If we gather terms multiplying each individual translate of c_3 , we see, for instance, that $c_3(t - 2)$ (underlined in the equation above) is multiplied by $\frac{1}{8}P_0 + \frac{6}{8}P_1 + \frac{1}{8}P_2$, while $c_3(t - 3/2)$ is multiplied by $\frac{1}{2}P_0 + \frac{1}{2}P_1$. In short, we can see that $\gamma(t)$ can be written as a sum of translates of $c_3(t)$, with the coefficients alternating between (a) averages of adjacent P_i s, and (b) $(1/8, 3/4, 1/8)$ weighted averages of triples of P_i s. If we replace c_3 with b_3 in this sum, the resulting curve does not change geometrically (although its parameter domain doubles in length), and we have recovered the refinement scheme we discovered earlier.

The preceding paragraphs provide only the most informal sketch of a proof that the two methods lead to the same result, but for the math-savvy, it's not hard to make this rigorous. The second approach has the advantage that it generalizes

nicely: any time a function f of some “size” (the support of b_3 is the interval $[-2, 2]$, so we could say its “size” is four) can be written as a weighted sum of functions of some smaller size (particularly ones that are scaled-down versions of f), a similar approach works. This is an important idea in the study of *wavelets*, which can be widely used to efficiently represent functions in computer graphics.

We’ll return to this idea of refining the control polygon for a B-spline when we discuss subdivision in Section 22.14W.

22.10W Non-uniform B-splines

In uniform B-splines, each segment was defined on an interval of unit length. We’ll see, in this section, why segments of other lengths, including *zero*, are in fact natural to include, and allow the user important kinds of shape control on a curve. The significant point is the control of continuity without the loss of geometric control. In the case of uniform B-splines, we know that the curves are all parametrically smooth up to second degree; for certain configurations of control points, however, the parametric plots can have reduced degrees of continuity (curvature or even tangent discontinuities), but these *only* occur in these specially-aligned-control-point configurations, which invariably result in changes to the character of the curve. For instance, to get a sharp corner in a uniform cubic B-spline requires that each of the segments meeting at the corner be linear. Nonuniform B-splines (NUBS) address this limitation.

Just as a uniform B-spline consists of segments that are joined at a sequence of equispaced t -values, a nonuniform B-spline consists of segments joined at a sequence of arbitrarily-spaced t -values. For a B-spline of degree d , we’ll have a nondecreasing **knot sequence** $t_{-d+1} \leq t_{-d} \leq \dots \leq t_n$ (the reason for the peculiar numbering will be apparent shortly). The t -values are called “knots” and must satisfy $t_i \leq t_{i+1}$ for all i , i.e., the knot sequence must be nondecreasing. If we choose a sequence of consecutive integers as knots, the nonuniform B-spline will turn out to be a uniform B-spline¹ (see the exercises).

Nonuniform B-splines have an advantage over uniform B-splines: you can insert an extra knot and control point without changing the curve at all. With uniform B-splines, this is impossible in general, for two reasons. First, if you have a uniform B-spline of degree d with control points P_0, \dots, P_k , its domain is $d \leq t \leq k$; if you insert an extra control point between, say, P_2 and P_3 , then the domain will be $d \leq t \leq k + 1$, and two functions with different domains cannot be the same. But this objection seems quibbling; perhaps we could require only that the B-splines *appear* the same, i.e., have the same parametric plots. With this reduced criterion, you *can* do control-point insertion in degree zero and degree one uniform B-splines. For degree zero, you just duplicate one of the control points; for degree one, you insert an extra control point anywhere on an edge of the control polygon. But for degree two, no such insertion technique is possible in general.

Associated to a knot sequence is a collection of **nonuniform B-spline basis functions** for each degree d , having several important properties:

- Wherever $d + 1$ of the basis functions of degree d are nonzero, they sum to one.

1. This suggests that “nonuniform” is not a great choice of names; nonetheless, it’s firmly embedded in the literature.

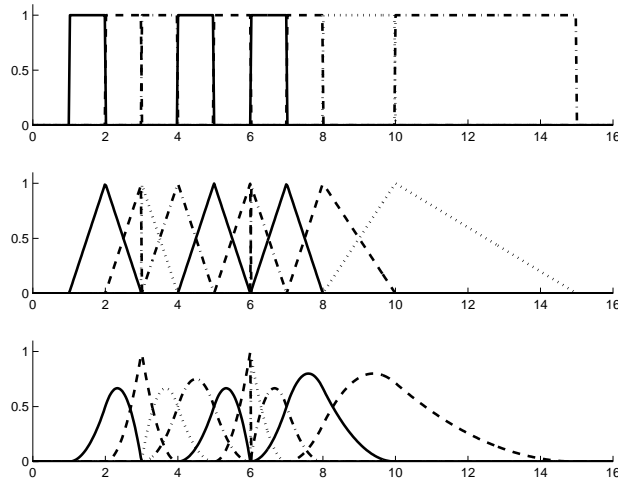


Figure 22W.22: The basis-functions of degree zero, one, and two, for the knot sequence 1, 2, 3, 3, 4, 5, 6, 6, 6, 7, 8, 10, 15; these knot-values were chosen to show the effects of non-uniform spacing and replicated knots. Basis functions of each degree cycle through four different line-types (solid, dashed, dotted, and dot-dash) to help distinguish them. Note that as the degree increases, the number of splines of that degree decreases. Note, too, that for the degree-zero basis functions, the sum of the functions is one for $1 \leq t \leq 15$, while for degree-one basis functions, the sum is one for $2 \leq t \leq 10$; over what range is the sum one for the degree-two basis functions?

- The degree- d basis functions are simply expressed in terms of those of degree $d - 1$ (for $d > 0$).
- The degree-zero basis functions are simply the characteristic functions of the half-open intervals $[t_i, t_{i+1})$.

Figure 22W.22 shows the basis functions for several degrees and a particular knot-sequence. The degree-zero basis functions are easy to understand but hard to see, since their graphs join into a single continuous line at $y = 1$, the difference in line-styles used to draw them helps distinguish them. Notice that some of the intervals have length zero, and the graphs of the corresponding basis functions have only a single point.

The inductive formula for higher-order B-splines in terms of lower-order ones is relatively simple; we'll build up case by case. Letting $B_{i,d}$ denote the i th spline of degree d , we have

$$B_{i,0} = \begin{cases} 1 & t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (22.105W)$$

and

$$B_{i,1} = \frac{t - t_i}{t_{i+1} - t_i} B_{i,0}(t) + \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}} B_{i+1,0}(t). \quad (22.106W)$$

Notice that this has the form something $\cdot B_{i,0}$ + something else $\cdot B_{i+1,0}$, i.e., it's a weighted sum of the two splines of lower order. The first coefficient (the first fraction) is a function that rises from 0 to 1 as t varies from t_i to t_{i+1} ; the second falls from 1 to 0 as t varies from t_{i+1} to t_{i+2} . Note that this is just a generalization of the inductive formula we saw earlier for uniform B-splines.

Inline Exercise 22W.14: Write out the expressions for $B_{4,1}(t)$ and $B_{5,1}(t)$. Verify that the coefficients of $B_{5,0}(t)$ are a pair of functions that sum to 1 on the interval $(t_5, t_6]$ (i.e., on the support of $B_{5,1}$). Generalize, and conclude something about the sum $\sum_i B_{i,1}(t)$ as a function of t .

The expression for $B_{i,1}$ above does not make sense when $t_i = t_{i+1}$ or $t_{i+1} = t_{i+2}$; in that case, the meaning is modified: any term that's undefined because of a divide-by-zero is to be ignored.

Inline Exercise 22W.15: On what interval is the function $B_{4,1}$ nonzero? Generalize to describe the support of the function $B_{i,1}$ in terms of the knot sequence. For which values of i does your answer make sense? Does it make sense for $i = 0$, for instance? For $i = -1$?

The recurrence for second-degree nonuniform B-splines is exactly analogous to that for first degree: two adjacent first-degree B-splines are blended by functions that rise from zero to one (or fall from one to zero) over their respective supports:

$$B_{i,2} = \frac{t - t_i}{t_{i+2} - t_i} B_{i,1}(t) + \frac{t_{i+2} - t}{t_{i+3} - t_{i+1}} B_{i+1,1}(t). \quad (22.107W)$$

Inline Exercise 22W.16: Write out an analogous recurrence for the degree-three nonuniform B-spline.

Inline Exercise 22W.17: What's the support of $B_{i,d}$ in general? For which values of i does this make sense?

Inline Exercise 22W.18: Explain why the nonuniform basis functions of degree d sum to the constant function 1.

Because the nonuniform B-spline basis functions are all non-negative (convince yourself of this!) and sum to one, a B-spline, defined as a sum of these functions multiplied by control points, will always lie in the convex hull of the control points; indeed, because only certain basis functions are nonzero on each interval, the corresponding points on the B-spline curve will lie in the convex hull of the corresponding control points. For example, the function $B_{1,1}$ is nonzero on the interval $t_1 \leq t < t_3$; $B_{2,1}$ is nonzero only on $t_2 \leq t < t_4$, and $B_{3,1}$ is nonzero only on $t_3 \leq t < t_5$. Thus on the interval $t_3 \leq t \leq t_4$, the curve defined by $\gamma(t) = \sum_i B_{i,1}(t)P_i$ lies in the convex hull of P_1, P_2 , and P_3 .

22.10.1W Effects of repeated knots and repeated control points

Repeated control points in a nonuniform B-spline have the same effect as those in a uniform B-spline: they constrain one or more curve-segments to lie in the convex hull of fewer than $d + 1$ points. As the number of repetitions increases towards the degree, d , two adjacent segments will reduce to simple line segments; when a control point is replicated $d + 1$ times, one segment becomes just a point, with a line-segment to either side. Furthermore, as the multiplicity of a control-point

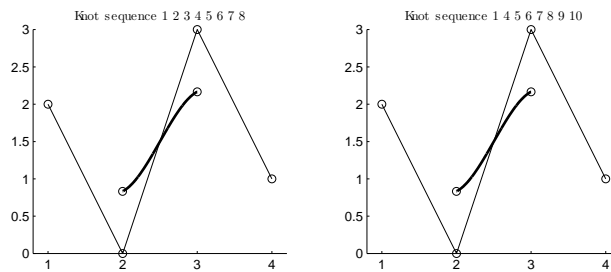


Figure 22W.23: (a) A nonuniform B-spline with knot sequence [12345678], (b) A nonuniform B-spline with the same control points, but knot sequence [145678910], showing that the two resulting splines are the same.

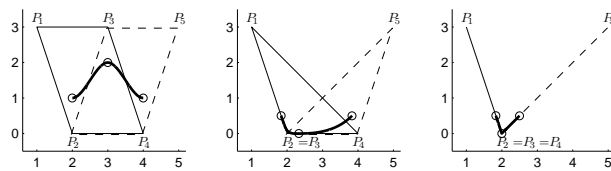


Figure 22W.24: (a) A two-segment uniform B-spline with control points P_1 through P_5 ; knots are shown as small circles. (b) A similar spline, but P_3 has been moved to coincide with P_2 . (c) Now P_4 has been moved to coincide with P_3 and P_2 . The solid and dashed polygons show the convex hulls in which the first and second segments must lie; because of the control-point duplications, these degenerate to become just line segments, and the curve ends up with a sharp corner at the repeated control point.

increases, the curve is pulled closer and closer to that point, until at multiplicity d the curve actually interpolates (passes through) the point.

By contrast, repeated knots in a nonuniform B-spline reduce the degree of continuity at the knot. When all knots are distinct in a cubic nonuniform B-spline, for instance, the curve is C^2 , but when a knot is duplicated, the two adjacent segments join at the knot with only C^1 continuity; a triple knot yields C^0 continuity (the curve is continuous, but the tangent changes suddenly at the knot); a quadruple knot yields C^{-1} continuity, i.e., the curve is not even continuous at the knot. This is illustrated in Figures 22W.23 – 22W.26. Indeed, the actual spacing of the knots is largely irrelevant; the only thing that influences the curve shape is when two adjacent knots are identical. It therefore is possible to work with NURBS using only integer knots, with some knots being replicated. One advantage of this is that once can consider all possible sequences of four adjacent-or-identical integers starting at 1 (i.e., (1,2,3,4), (1,1,2,3), (1,2,2,3), etc.), and compute, once and for all, the associated NURBS basis functions, and represent them in the same matrix form that we used for other curve types.

22.10.1W.1 Knot insertion

Nonuniform B-splines are rich enough to allow *knot insertion*: we can take a nonuniform B-spline with a control-point sequence $\{P_i\}$ and knot-sequence $\{t_i\}$ and insert an extra control point and knot, say $\dots, P_1, P_0, P_1, Q, P_2, P_3, \dots$ and $t_{-1}, t_0, t_1, s, t_2, t_3, \dots$ and have the resulting spline be identical. We can do this for any new knot value (as long as, in our example, $t_1 \leq s \leq t_2$), but having chosen the new knot value, we must choose the new point Q carefully so as not to disturb

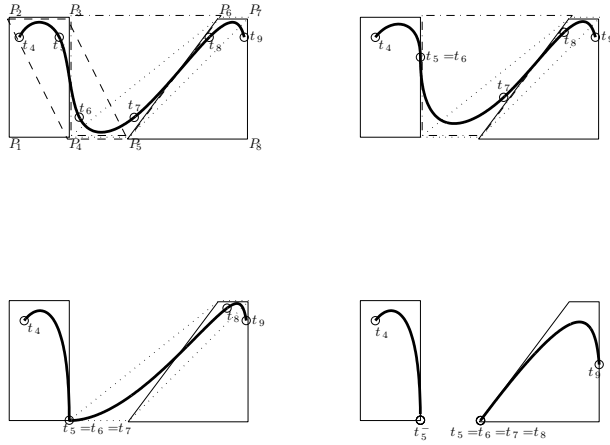


Figure 22W.25: (a) A five-segment nonuniform B-spline with control points P_1 through P_6 and knot-sequence $(t_1, \dots, t_{11}) = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)$; (b) A similar spline, but t_6 has been set equal to t_5 , removing the second segment of the curve, and causing the control hulls for the first and third to overlap in a line segment instead of a triangle; the curve point associated to $t_5 = t_6$ is forced to lie on that line. (c) Now t_7 has also been made identical to t_5 and t_6 , eliminating another curve segment and making the overlap of adjacent hulls degenerate to a single point, through which the curve must pass. (d) When we do this once more (setting $t_8 = t_7 = t_6 = t_5$), we actually generate a discontinuity in the curve. The point labelled t_5^- corresponds to a parameter value just a tiny bit smaller than t_5 , because the curve segment from t_4 to t_5 is defined for $t_4 \leq t < t_5$, i.e., it terminates just before the parameter reaches t_5 . Note that as we raised the knot's multiplicity, we reduced the degree of continuity of the spline at that knot by one each time.

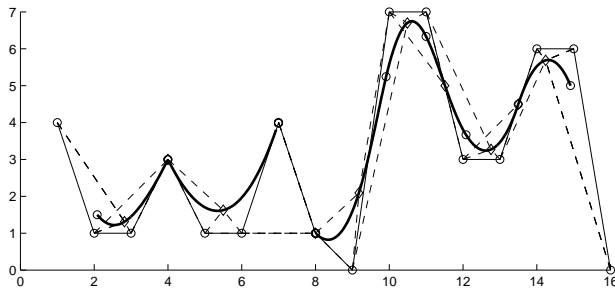


Figure 22W.26: A nonuniform B-spline in which control points are chosen to lie sequentially along the x -axis, but with varying y -values; certain knots are replicated to show the continuity-reducing effect of this replication. The knot-sequence is $[0, 1, 2, 3, 4, 4, 4, 5, 5, 5, 5, 6, 7, 8, 9, 9, 10, 11, 12, 13]$. Each control point has a dashed line to the midpoint of each segment that it influences.

the shape of the spline. The details are beyond the scope of this book, but are discussed in any book on splines.

22.10.2W Rational Curves

While the curve types we've discussed are quite rich, they cannot be used to exactly represent certain important curves, namely the *conics*, the solutions to equations of the form $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$, which include ellipses, hyperbolas, and circles. To make this clear, consider the case of a cubic curve

$$\gamma(t) = (x(t), y(t)) = (a_x + b_x t + c_x t^2 + d_x t^3, a_y + b_y t + c_y t^2 + d_y t^3). \quad (22.108W)$$

Suppose that $\gamma(t)$ lies on the circle $x^2 + y^2 = 1$ for every t . Then the functions x and y must satisfy $x(t)^2 + y(t)^2 = 1$. It's not hard to show (see the exercises) that the coefficient of t^6 in the left side of this equation is $d_x^2 + d_y^2$; the coefficient of t^6 on the right is zero; hence $d_x = d_y = 0$. Similar arguments show that $c_x = b_x = c_y = b_y = 0$ as well, from which we conclude that both x and y are constant functions! The only cubic curves whose images lie on a unit circle are degenerate ones that remain at a single point.

If we are hoping to use our curves to describe the shapes of manufacture, circles are essential. Fortunately, there's a useful observation that lets us produce circles (and other conics) with cubic curves: even though there's no cubic curve in the plane that traces out a circle, there *is* a cubic curve in 3-space which, seen in perspective, traces out an arc of a circle in the image plane. Using a perspective projection based on looking along the z -axis, from the origin, and projecting onto the $z = 1$ plane, this means we can find a curve $\gamma(t) = (x(t), y(t), z(t))$, with x, y , and z all cubics in t , with the property that

$$(x(t)/z(t))^2 + (y(t)/z(t))^2 = 1 \quad (22.109W)$$

for all t . In particular, if we let

$$x(t) = 1 - t^2 \quad y(t) = 2t \quad z(t) = 1 + t^2 \quad (22.110W)$$

then the equation above holds. Figure 22W.27 shows the curve γ in 3-space and its projection to the $z = 1$ plane; in 3-space, γ describes a parabola; in the projection plane, it traces out all but one point of a circle.

Given a curve $\gamma(t) = (x(t), y(t), z(t))$ in 3-space, the associated *rational curve* $\Gamma(t) = (X(t), Y(t))$ is defined by $X(t) = x(t)/z(t)$, $Y(t) = y(t)/z(t)$. At values of t where $z(t) = 0$, $X(t)$ and $Y(t)$ are undefined (and near those values they tend to vary rapidly with respect to t , creating challenges for various numerical techniques). It's conventional to replace the letter z with w , however, and refer to this extra coordinate as a *weight*. Thus a rational spline in the plane is typically described by a sequence of control points (x_i, y_i) in the plane, each with an associated weight w_i ; to *compute* points on the spline, we consider triples (x_i, y_i, w_i) in 3-space, find the spline $t \mapsto (x(t), y(t), w(t))$ that they define, and then divide by w to get $X(t) = x(t)/w(t)$ and $Y(t) = y(t)/w(t)$. So while the user of a rational spline considers the weights to be distinct from the coordinates, the implementor treats all three as coordinates in a higher-dimensional space.

Both uniform and nonuniform B-spline curves have rational forms, called *rational B-splines* and *nonuniform rational B-splines* (or *NURBS*) (and knot-insertion, although slightly more complex, works for NURBS as well as NUBS).

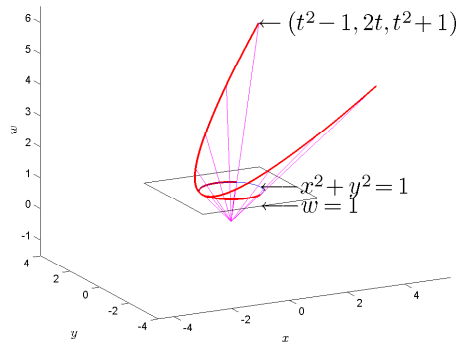


Figure 22W.27: The curve $(x(t), y(t), z(t)) = (1 - t^2, 2t, 1 + t^2)$ in 3-space, seen from the origin, looking along the z -axis, looks like a circle; here we see the curve projected into the $z = 1$ plane; the only point of the unit circle that's missing is $(x, y) = (-1, 0)$, which corresponds to $t = \pm\infty$.

22.10.3W Transformations and rational splines

Supposed we have a spline η based on control points P_i , and we transform each of the points P_i by an affine transformation T to get the points $Q_i = T(P_i)$. If we now form a spline ζ based on the points Q_i , how will it be related to η ? It turns out that $\zeta(t) = T(\eta(t))$ for all t , i.e., we can either compute the spline and then transform all the points along it, or we can transform the control points and compute a new spline – the two results will be the same.

What about rational splines? Is the same thing true? Suppose we have a rational spline, γ , based on points P_i in 3-space, and we transform those points by some transformation T to get points $Q_i = T(P_i)$. We form a new spline, λ , based on the Q_i . We then compute Γ and Λ , the associated rational curves. How are $\Gamma(t)$ and $\Lambda(t)$ related, if at all? One might hope that $\Gamma(t) = T(\Lambda(t))$, but this doesn't even make sense: Γ and Λ are curves in 2-space, while T is a transformation on 3-space. In fact, the most we can say is that $\lambda = T \circ \gamma$, i.e., we can compute the spline in 3-space before or after transformation, but we must do so before the “homogeneous division by w .”

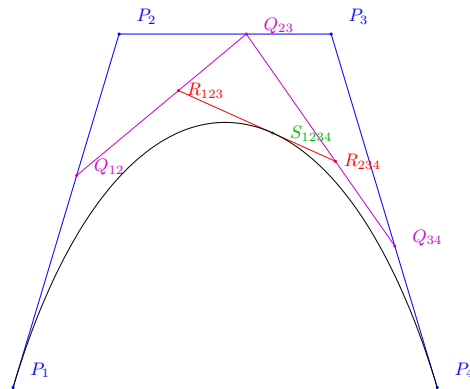


Figure 22W.28: The de Casteljau construction of a curve for four control points: we divide the edges between adjacent control points into ratios t and $1-t$ to get point $Q_{i,i+1}(t)$; we repeat the operation on the Q s to get $R_{i,i+1,i+2}(t)$, and finally join two R s to get a point $S(t)$. As we vary t from 0 to 1, the points $S(t)$ traces out a curve that starts at P_1 and ends at P_4 , and lies within the convex hull of the P s.

22.11W The de Casteljau algorithm

As a kind of transition from parametric cubic splines to subdivision curves, we'll now investigate a particular way to construct a curve from a set of four control points, due to de Casteljau [8]. The idea is quite simple: we take the edges P_1P_2 , P_2P_3 , and P_3P_4 and divide them into ratios t and $1-t$ (where t is some number between 0 and 1), producing points $Q_{12}(t)$, $Q_{23}(t)$, and $Q_{24}(t)$ (see figure 22W.28). We repeat the operation on the Q s to get points $R_{123}(t)$ and $R_{234}(t)$. Finally, we repeat the operation on the R s to get a single point $S(t)$. As we vary t , the point $S(t)$ traces out a curve.

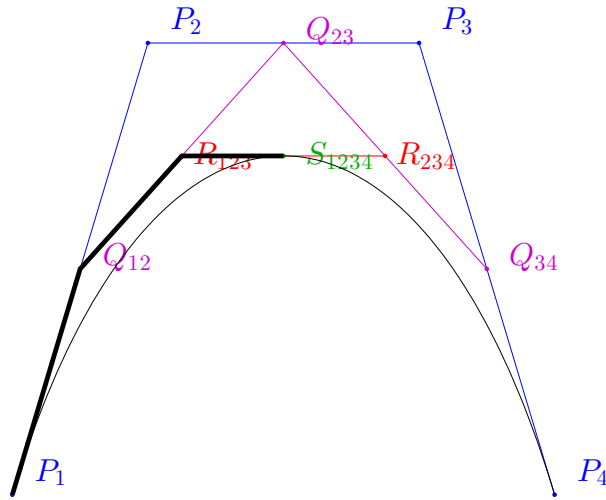


Figure 22W.29: The de Casteljau construction for $t = \frac{1}{2}$. The boldface polygon at the left (consisting of $P_1, Q_{12}(\frac{1}{2}), R_{123}(\frac{1}{2}),$ and $S(\frac{1}{2})$), and the corresponding four points on the right hand side ($S(\frac{1}{2}), R_{234}(\frac{1}{2}), Q_{34}(\frac{1}{2}),$ and P_4) each define their own curves through the de Casteljau construction. Calling these $S_L(t)$ and $S_R(t)$ (for ‘left’ and ‘right’), we can ask how $S_L, S_R,$ and S are related.

The curve traced by $S(t)$ is a cubic, as we can see by writing out the individual terms:

$$Q_{i,i+1}(t) = (1-t)P_i + tP_{i+1} \tag{22.111W}$$

$$R_{i,i+1,i+2}(t) = (1-t)Q_{i,i+1}(t) + tQ_{i+1,i+2}(t) \tag{22.112W}$$

$$= (1-t)((1-t)P_i + tP_{i+1}) + t((1-t)P_{i+1} + tP_{i+2}) \tag{22.113W}$$

$$= (1-t)^2P_i + 2(1-t)tP_{i+1} + t^2P_{i+2} \tag{22.114W}$$

$$S(t) = (1-t)R_{123} + tR_{234} \tag{22.115W}$$

$$= (1-t)((1-t)^2P_1 + 2(1-t)tP_2 + t^2P_3) \tag{22.116W}$$

$$+ t((1-t)^2P_2 + 2(1-t)tP_3 + t^2P_4) \tag{22.117W}$$

$$= (1-t)^3P_1 + 3(1-t)^2tP_2 + 3(1-t)t^2P_3 + t^3P_4 \tag{22.118W}$$

As you can see, the curve traced out by $S(t)$ is just the Bézier curve! The coefficients of the P_i s are just the Bernstein polynomials. So this geometric construction corresponds to a polynomial curve description. We’ll study this kind of correspondence in much more detail in the next section. Before moving on, however, let’s look at a particular instance of the de Casteljau construction, namely $t = \frac{1}{2}$ (see Figure 22W.29).

The left half of the construction yields a polygon consisting of $P_1, Q_{12}(\frac{1}{2}), R_{123}(\frac{1}{2}),$ and $S(\frac{1}{2})$; if we applied the de Casteljau construction to this polygon, we’d get a curve $t \mapsto S_L(t)$ that starts at P_1 , and ends at S , just as does the left half of the original de Casteljau construction, i.e., $S([0, \frac{1}{2}])$. It turns out that S and S_L are very closely related: for $0 \leq t \leq 1$, we have $S_L(t) = S(t/2)$; similarly, for the right half of the curve, we can define S_R , and find that $S_R(t) = S(\frac{1}{2} + t/2)$. In other words, rather than applying the de Casteljau construction for every value of t as a way to trace out the curve, we could do it just for $t = \frac{1}{2}$, to find the mid-point of the curve, and then apply it separately to each half, to find the $t = 1/4$ and

$t = 3/4$ points. Working recursively, we can find $S(t)$ for every point t of the form $n/2^k$, where n is an integer from 0 to 2^k , and k is any nonnegative integer (such points are called *dyadics*). Since every real number between 0 and 1 is arbitrarily close to a dyadic point, we've effectively computed the shape of the curve².

Programming Exercise 22W.2: Use the 2D testbed to write a program that lets the user click on four control points and then move a slider from 0 to 1; as the slider moves to value t , show the de Casteljau construction of $S(t)$, and highlight the point $S(t)$ on the Bézier curve determined by the four points.

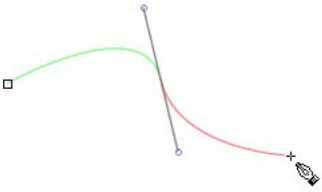


Figure 22W.30: Drawing a smooth path in Inkscape.

22.12W A curve-drawing program

With the techniques described so far, you can build a basic curve-editing program. We'll model our program after the "curve" portion of Inkscape, a simple and easy to use drawing program. Because we'll want to use the word "curve" for a small piece of such a drawing, we'll use the word "path" for a sequence of smaller pieces. Figure 22W.30 shows a path being drawn. The user clicked on the starting point at the left, then clicked in mid-figure, at the joint between the red and green arcs and dragged downward and a little to the right, releasing at the lower end of the blue tangent line, and has moved to the right preparing to either add more points or double-click to end the path. The control points are called *nodes*; those at the end of a sequence of segments are called *endnodes*; the others are *internal nodes*. The small diamonds or squares that indicate the nodes during editing, and the small circles at the ends of the tangents are called *handles*.

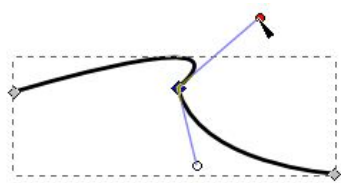


Figure 22W.31: The user adjusts one of two tangents at a bend in the path.

During the dragging operation, the tangent line was visible and moved and adjusted its length as the user moved the mouse. As the user moves on to click the next point, the red portion of the path is continuously updated.

Once a path has been initially sketched, the user can either draw more paths or switch to path-editing mode, shown in Figure 22W.31. Here the user has clicked on the central node, causing its tangent to be shown, and then has indicated that it should not be a "smooth" point any more, but instead should be a "corner" point, in which the two tangents can be manipulated independently. By clicking on the handle at the end of the upper tangent, the user is adjusting the shape of the upper curve near the "corner."

When the user is in path-editing mode, a menu bar (Figure 22W.32) appears that allows various operations. In editing mode, the path is displayed with its nodes indicated as small diamonds (for endnodes or "corner" points — see below) or squares (for smooth or symmetric nodes). The user can select a single node, by clicking on it, or a pair of adjacent nodes, by clicking on the curve or line segment between them. In each case, the selected nodes are highlighted, and their tangents, if any, are shown.



Figure 22W.32: Path-editing tools in Inkscape.

Reading from left to right, the available operations are

- **Add** a node to the middle of the selected segment (does nothing if a single node is selected).
- **Remove** the selected node or nodes.

2. Indeed, from the point of view of computing, dyadic points are exactly those that are representable in binary with finitely many bits, so they're the numbers that we mostly see in computers.

- **Join** two selected endnodes.
- **Insert a segment** between two selected endnodes.
- **Delete a segment** between two adjacent internal nodes.
- **Split a path** at an internal node, creating two identical but independent endnodes, one for the end of the first part, one for the start of the second part.
- **Corner:** change the selected nodes to “corner” nodes, whose two tangent control handles are independent.
- **Smooth:** change the selected node(s) to “smooth” nodes, in which the two tangents are parallel.
- **Symmetric:** change the selected node(s) to be “symmetric”, with the two tangents being equal and opposite.
- **Line:** Make the selected segment(s) into line(s).
- **Curve:** Make the selected segment(s) into curve(s).

Each of these operations is only vaguely defined by the description above. Questions remain, like “When we make a segment into a line, and both ends were previously ‘symmetric,’ what should they be after the conversion?” Making wise choices in situations like this has a large effect on usability of the system. You should be guided by the *principle of least surprise*: do the thing that most users expect, or are least surprised by in practice. Some of the choices made in Inkscape seem unnatural to us, so we won’t try to emulate them exactly. Indeed, all we want to discuss here is the underlying data structures for representing paths in a program.

To capture the general design of paths in Inkscape, we create a model. First, a path is a sequence of segments, and each segment is either a line or a curve. The segments need not be connected; a path may have several connected subsequences which are not (visually) connected to one another. All editing operations will apply to a single path. (Inkscape does, however, allow the user to join two paths into one larger path, so that their endnodes can have a segment inserted to join them.) Each segment is either a line-segment or a curve, and has a start node and an end node. Two adjacent segments in a path often share a node, although not always. A node shared by two segments is “internal”; a node that’s in only one segment is an “endnode.” If all the segments containing a node are deleted, then the node is deleted as well.

Because typically curve-segments in Inkscape are controlled by two points and two tangents, it seems natural to represent them as Hermite curves.

Nodes themselves are complex:

- A node joining two straight segments has only one adjustable characteristic: its position.
- A node joining a straight segment to a curve segment has a position and a tangent (on the curve side). The tangent may be arbitrarily adjustable, making the node a corner, or may be constrained to be parallel to the line-segment, making the corner “smooth.” Because for a line segment there’s no tangent *magnitude*, only a direction, there’s no notion of a “symmetric” node joining a line- to a curve-segment.

- A node joining two curve segments has a position and two tangents. The tangent may be independently adjustable (a “corner”), required to be parallel and point in opposite directions (“smooth”) or required to be parallel, in opposite directions, with the same magnitude (“symmetric”)
- An endnode for a line segment has only a position.
- An endnode for a curve segment has both a position and a tangent vector.

Inkscape also seems to allow curve segments where only one endnode has a tangent; it appears that these may be represented by the quadratic analog of an Hermite curve, in which we specify two endpoints and one tangent. Since such segments are difficult to control (we find ourselves selecting the no-tangent end and applying the “smooth” operation to give it a tangent), we’ll omit this case.

So our design now has several classes of objects:

A *path* is a sequence of *subpaths*, and a subpath is a sequence of *segments*.

There are two kinds of segments: *line* and *curve*.

Each *segment* has two *nodes*. A node may be a *corner*, *smooth node*, or *symmetric node*. A line-segment may not have a symmetric node.

For simplicity, we’ll insist that a node has a reference to any segment that contains it; we’ll call the list of containing segments the *parents* of the node. A node must have at least one parent, and no more than two. A node with one parent is an “end node,” while one with two parents is an “internal node.”

With this design, we have a rich enough model to support all the inkscape operations.

During interaction with a path, a *selection* is a subset of the path’s nodes. We’ll say that a segment is ‘selected’ if both its nodes are selected, so if we select three nodes in sequence, joined by two segments, and perform the “add” operation, two new nodes will be created, one on each segment. What if we select the endnode of one subpath and the endnode of another subpath and perform the “add” operation? We cannot add a control point mid-segment, because there’s no segment. A convenient (and low-surprise) choice is to say “we add a new point in the interior of every selected segment,” so that in this cases, no nodes are added at all. Think through the remaining operations (except for “Join,” which seems under-specified to us) and try to consider all possible cases and what the behavior should be in each case.

So far we’ve talked about the behavior in broad terms, but what’s entailed in adding a new node in the middle of a segment? There are two cases.

For a line segment from node n_1 to n_2 , we should add the midpoint m as a new node, create line segments from n_1 to m and m to n_2 , and replace the segment from n_1 to n_2 in the path with these two new segments, as shown in Figure 22W.33. We should also update the “selection” when we are finished with all segment additions so that it includes the newly created node.

For a curve segment, we have an Hermite curve γ defined by its starting point P , starting tangent \mathbf{v} , ending point Q , and ending tangent \mathbf{w} . We can create a new point $X = \gamma(\frac{1}{2})$, and a new vector $\mathbf{u} = \gamma'(\frac{1}{2})$. We then replace our curve segment by two new ones, the first defined by P, X, \mathbf{v} and \mathbf{u} , the second defined by X, Q, \mathbf{u} , and \mathbf{w} . We once again must add X to the selected-nodes collection after we’ve completed the addition operation for all selected segments.

The *deletion* operation raises a new complexity, even when there’s only a single internal node selected. Suppose that node joins a curve-segment AB to a line-segment BC (Figure 22W.34). Deleting node B will replace those two segments

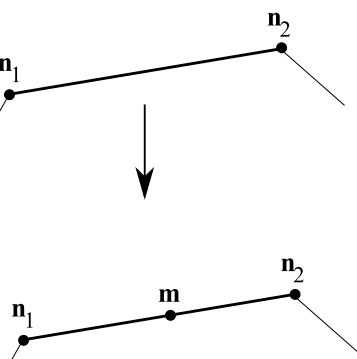


Figure 22W.33: Inserting the midpoint in a line segment.

with a single one. Should the replacement segment be a curve or a line? And should its nodes be corner nodes, smooth nodes, or symmetric nodes?

There's no "right answer" to questions like this. But one possible answer is to say "it should be a curve segment, because that retains any tangent information we had before the operation; if it's unwanted, it's easy to change the segment to a line. But if we make the new segment a line, we've lost the tangent information forever"³. To build a curve segment, we need two points — the node locations *A* and *C* provide these — and two tangent vectors. The tangent to the Hermite curve at *A* provides one, but there's no tangent vector for the line-segment at *C*, only a *direction*. An *ad hoc* compromise is to make a tangent vector at *C* whose direction is determined by the line segment *BC*, but whose magnitude is the same as the tangent magnitude at *A*. If node *C* is a corner node, this tangent-setting can be done easily, since the tangents at a corner node are independent. If it is a smooth node, then the subsequent tangent at *C* was already in the direction *BC*, so again we've got what we need. And node *C* cannot be a symmetric node, because line segments may not contain symmetric nodes.

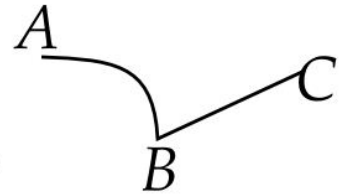


Figure 22W.34: How can we delete node *B*?

Inline Exercise 22W.19: What, if anything, must be done if node *C* is an endnode? Are there any special cases to handle?

Inline Exercise 22W.20: Suppose that *C* is an endnode, and we want to delete it. The analysis of the preceding paragraph does not apply, since it assumes that the node to be deleted is an internal node. Make some choices about how endnode deletion should behave. Be sure to consider the special case where the endnode is one end of a single-segment subpath!

To finish the design, you must consider the effects of all possible operations on all possible sets of selected nodes. Once you've made a few decisions, others will start to follow naturally, as part of the goal of consistency from operation to operation. For instance, if you insert a node in a segment and then delete it immediately, there should be no net effect. And while we've said that there are no right answers for how to behave when an operation is under-defined, there are certainly *wrong* ones. For instance, we could have chosen the missing tangent, in our point-deletion example, to always be a unit vector in the $+y$ -direction, but this choice would have been universally despised by users.

Programming Exercise 22W.3: Implement a curve-drawing program like the one described above. You'll need to make decisions about many things, like "when should a node's tangents and their handles be visible?" and "How can the user adjust a tangent whose length is zero, so that its handle and the position handle for the node occupy the same space?" Ask a friend to use your program and critique it.

Programming Exercise 22W.4: In inkscape path-drawing mode, when the user clicks a sequence of points, the result is a polyline. Add a feature to your path-editor so that when the user shift-clicks a sequence of points, your program generates a Catmull-Rom spline through those points, which it then immediately converts into a sequence of Hermite curve-segments.

3. In practice, Inkscape makes the opposite choice: if either adjacent segment is a line, the replacement segment is made a line as well.

22.13W Direct Manipulation of Splines

We've treated splines so far as curves defined by a sequence of control points, whose very name suggests that they should be used to control the shape of the curve. But for many users, a more *direct* interaction makes better sense. In **direct manipulation** a user can click-and-drag on an arbitrary point of a spline curve, moving that point to a new location. In this section, we show how to implement two versions of direct manipulation for a Bézier curve.

The starting point for our direct manipulation is a Bézier curve γ with control points P_1, \dots, P_4 , a user-click at some location $A = \gamma(t_0)$ on the curve, followed by a drag-release at a new location B . The output is a new set of control points Q_1, \dots, Q_4 for a new curve ζ with $\zeta(t_0) = B$. Figure 22W.35 shows the idea. The user clicks and drags (see red arrow) near the right end of the blue Bézier curve, governed by the blue control points; the points are moved somewhat to generate the red curve set and curve. The magenta arrow shows the result of a different manipulation that starts near the center of the blue curve.

Figure 22W.35: Direct manipulation of the blue curve leads to the red and magenta curves.

As described, the task is grossly under-specified. There are many possible solutions. For instance, we can add the vector $B - A$ to each P_i to generate the corresponding point Q_i . This would translate the entire curve by $B - A$, solving the problem. We could, in a more extreme approach, simply set $Q_1 = \dots = Q_4 = B$, so that the entire second Bézier curve consisted of the point B . Neither of these seems like a good solution, however — the control points are moved a lot to achieve a relatively small change.

To further constrain the problem, we'll seek the control set Q_1, \dots, Q_4 with the property that the vectors $\mathbf{v}_i = Q_i - P_i$ are as small as possible, in the least-squares sense, i.e., that the *sum of the squared control-point displacements is as small as possible*. That is the approach that was used to produce Figure 22W.35.

Before we work on the details of the manipulation process itself, we assumed that the user clicked on a point $A = \gamma(t_0)$; in practice, a user will seldom do so. Instead, the user clicks *near* the curve, and we have to find a close point on the curve. A practical solution is to pick a hundred equally-spaced (in t) points on the curve, compute their distances from the click-point, select the nearest one, and call its location A (and its parameter value t_0). You can do somewhat better by finding the closest *two* points, and then using binary subdivision on the t -interval between them to get a more precise approximation of the closest point to the user-click.

Recall that the Bézier curve γ is defined by

$$\gamma(t) = [P_1; P_2; P_3; P_3] \mathbf{M}_B \mathbf{t}(t) \quad (22.119W)$$

$$= \mathbf{G} \mathbf{M}_B \mathbf{t}(t), \quad (22.120W)$$

where \mathbf{G} is the geometry matrix and \mathbf{M}_B is the Bézier basis matrix. We have the parameter value t_0 , and the point $A = \gamma(t_0)$. Our problem is to replace \mathbf{G} with $\mathbf{G} + \Delta \mathbf{G}$ so that

$$\zeta(t) = (\mathbf{G} + \Delta \mathbf{G}) \mathbf{M}_B \mathbf{t}(t), \quad (22.121W)$$

satisfies $\zeta(t_0) = B$, and $\Delta \mathbf{G}$ is as small as possible. To simplify notation, we'll use \mathbf{H} instead of $\Delta \mathbf{G}$. Subtracting $\gamma(t_0) = A$ from $\zeta(t_0) = B$ gives

$$\mathbf{H} \mathbf{M}_B \mathbf{t}(t_0) = B - A. \quad (22.122W)$$

Replacing $B - A$ with \mathbf{v} , we want to solve

$$\mathbf{H}\mathbf{M}_B\mathbf{t}(t_0) = \mathbf{v} \quad (22.123W)$$

for the unknown \mathbf{H} . Notice that the unknown here is a 4×2 matrix, and *not* the vector $\mathbf{t}(t_0)$, which is known! The goal is find the unknown 2×4 matrix that satisfies these equations *and* has the property that the sum of the Euclidean lengths of its four column vectors is as small as possible. This is a constrained optimization problem: the sum we're minimizing turns out to be $\sum_{ij} h_{ij}^2$ (where each h is an entry of \mathbf{H}); the constraint is that \mathbf{H} satisfies Equation 22.123W. One can approach the optimization problem with the technique of Lagrange multipliers, but we'll pursue a somewhat more geometric analysis.

First, suppose that \mathbf{v} happens to be aligned with the x -axis. Then since the y -coordinates of the control points affect only the y -component of γ , there's no reason to change any of them: doing so cannot possibly move $\gamma(t_0)$ closer to B . So in this case, we can see the the change in each control-point's position will be in the x -direction, i.e., *will be a multiple of the desired displacement \mathbf{v}* . A similar analysis applies when \mathbf{v} is aligned with the y -axis, and indeed you can see (by changing coordinates) that it's true in general. In the optimal solution of Equation 22.123W, the columns of \mathbf{H} , which represent the displacements of the control points, must all be multiples of \mathbf{v} . We have thus reduced our problem from eight unknowns (the entries of \mathbf{H}) to just four (the multipliers for \mathbf{v}). We can write

$$\mathbf{H} = \mathbf{v}\mathbf{c}^T \quad (22.124W)$$

so that the first column of \mathbf{G} is $\begin{bmatrix} c_1 v_1 \\ c_1 v_2 \end{bmatrix}$, and similarly for the other three columns. The vector \mathbf{c} is now our unknown. Furthermore, the thing we're optimizing — the sum of the squares of all the h_{ij} s — is proportional to $\|\mathbf{c}\|^2$.

Inline Exercise 22W.21: Verify the last sentence by computing the constant of proportionality.

Since \mathbf{M}_B and $\mathbf{t}(t_0)$ are constants, we'll combine them into $\mathbf{r} = \mathbf{M}_B\mathbf{t}(t_0)$. (What are the dimensions of \mathbf{r} ?) The constraint equation $\mathbf{H}\mathbf{M}_B\mathbf{t}(t_0) = \mathbf{v}$ now becomes

$$\mathbf{v}\mathbf{c}^T\mathbf{r} = \mathbf{v}. \quad (22.125W)$$

so that $\mathbf{c} \cdot \mathbf{r} = 1$. The shortest vector \mathbf{c} solving Equation 22.125W is $\mathbf{c} = \frac{\mathbf{r}}{\|\mathbf{r}\|}$, so

$$\mathbf{H} = \frac{1}{\|\mathbf{r}\|} \mathbf{v}\mathbf{r}^T, \quad (22.126W)$$

from which we can find the geometry matrix $\mathbf{G} + \mathbf{H}$ for the new curve ζ .

An alternative approach is based on a theorem for linear algebra: for any $k \times n$ matrix \mathbf{E} , every vector $\mathbf{x} \in \mathbf{R}^n$ can be written uniquely as a sum

22.14W Subdivision Curves

We now shift from curves defined by polynomials to those defined by a class of simple geometric constructions called *subdivision*, in which a polygon is modified

to have about twice as many points (and the original points are usually moved somewhat); this process is repeated in the hopes of approximating a smooth curve. The advantage of subdivision is that it's very easy to implement and understand; indeed, we saw a simple example in chapter 4. A small drawback to subdivision curves is that it's not obvious, at the start, how one might find an arbitrary point on a subdivision curve without carrying out an infinity of subdivision steps. We'll resolve that through the careful study of one class of subdivision curves, which will turn out to be just the same as cubic B-spline curves.

You might well ask why, if they're really the same as B-spline curves, we trouble to go through another derivation of them. The answer is that the methods used to see that the two curve-types are identical are exactly the same methods used to study the relationship between B-spline *surfaces* and subdivision *surfaces*, but the mathematics is considerably simpler in the curve case, and yet contains all the essential ideas of the surface case. Furthermore, that same proof also shows that one can use subdivision curves as a way to draw approximations of B-splines, a method that can be very practical.

22.14.1W Midpoint subdivision

We'll consider a very simple subdivision rule (see Figure 22W.36) in which we subdivide a polygon by inserting new vertices at the midpoint of each edge, and then moving the original vertices to a new location that's a convex combination of the vertex and its two neighbors (we saw an example of this earlier in Chapter 4). We'll work with the case where the combination is *symmetric*, in the sense that each of the neighbors contributes equally to the new location,

$$P_i^{\text{new}} = (1 - 2s)P_i + sP_{i-1} + sP_{i+1}, \quad (22.127W)$$

which can be rewritten in the form

$$P_i^{\text{new}} = (1 - 2s)P_i + 2s \frac{P_{i-1} + P_{i+1}}{2}, \quad (22.128W)$$

i.e., the new location of P_i is a weighted sum of the old location and the midpoint of its two neighbors. This symmetry restriction is not actually necessary (see the exercises), but non-symmetric subdivision rules can lead to peculiar results. We'll be considering a particular case, namely $s = \frac{1}{8}$, but other values for s lead to interesting results as well (see the exercises).

Programming Exercise 22W.5: Write a program, based on the 2D testbed, in which subdivision is made more generic: a polygon is subdivided by inserting the midpoint of each edge, but the new vertex location for P_i is $aP_{i-1} + bP_i + cP_{i+1}$, where $a + b + c = 1$. Make a and b be user-adjustable. What happens if one or both is negative? Greater than one? Experiment, and describe the results. One possible interface for choosing a , b , and c is to show a triangle ABC , and let the user click a point in or near the triangle; the barycentric coordinates of the clicked point then provide values for a , b , and c . If the user clicks A , then $a = 1$ and $b = c = 0$, and similarly for the other vertices.

Programming Exercise 22W.6: (a) Write a program, based on the 2D testbed, to test subdivision with $s = 1/8$. Let the user click the points of a polygon, and when it's complete, show 4 levels of subdivision. Also let the user move the original vertices after subdivision has been done, and show how the subdivided curves move. (b) How large a region of influence does one original vertex have?

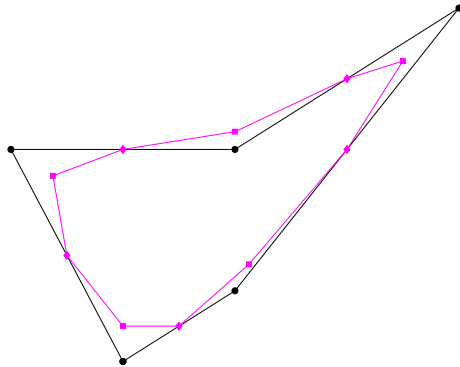


Figure 22W.36: The midpoint subdivision rule: the original polygon, marked with circles, has new points inserted at the middle of each edge (diamonds) and each original point P is moved to a convex combination of P and its two neighbors, i.e., $P_i^{new} = (1 - 2s)P_i + sP_{i-1} + sP_{i+1}$.

If you start with ten vertices, and move one of them, does it affect the entire limit curve (or, in this approximation, all points of the 4th-level subdivision curve)? Or is its influence limited to some smaller part of the curve? How much?

Figure 22W.37 shows an initial polygon that's had midpoint subdivision with $s = 1/8$ applied to it repeatedly. We'll refer to the $s = 1/8$ version of midpoint subdivision as "subdivision" from now on, with it being understood that we're just considering this special case; note that in this case, the new location of P_i is just $\frac{3}{4}P_i + \frac{1}{8}P_{i-1} + \frac{1}{8}P_{i+1}$. As you can see, more and more finely subdivided polygons appear to approach a smooth curve in the limit.

We can also subdivide a polyline P_1, P_2, \dots, P_n in the same way, except that the endpoints P_1 and P_n have no left or right neighbors, respectively, so we cannot define edge points there. Our solution will be to ignore these points, so the subdivided curve has $(n - 1)$ edge points (between every pair of initial vertices) and $(n - 2)$ newly-located vertices, for a total of $2n - 3$ points. (One can also include the original endpoints, without moving them; see the exercises.) So for large n , subdivision approximately doubles the number of vertices in the polyline.

For many readers, this is all you'll ever need to know about subdivision: there's a rule for inserting new points and moving old ones a little, and if you do it often enough, you get something that looks smooth, and indeed, looks an awful lot like

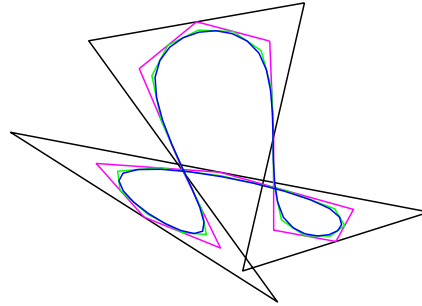


Figure 22W.37: An initial polygon, subdivided several times by the midpoint rule with $s = 1/8$. This repeated subdivision leads to a smooth curve.

a spline curve. Of course, if you've experimented with different ways of combining neighboring points to move the old points, you've discovered that for some combination rules, the limiting curve is not at all nice; indeed, the limit may not even exist. It's therefore prudent to understand under what conditions subdivision actually converges, especially because we'll be generalizing the idea of subdivision to surfaces in Chapter 23W. We'll therefore continue on here with an analysis of subdivision curves, their limits, and their relationship to splines.

22.15W A central example

Suppose we consider a sequence of control points $P_i = (i, 0)$ for $i \neq 0$, and $P_0 = (0, 1)$, as shown in Figure 22W.38, connected by a polyline. Repeated subdivision of this polygon appears to converge to a smooth curve whose y -values are nonzero for $-2 < x < 2$; this curve appears to be identical to the graph of the cubic B-spline basis function $b_3(t)$. (We're using here the symmetric B-spline basis functions that appeared in Chapter 18, i.e., the one whose support is $[-2, 2]$ rather than $[0, 2]$, because it makes the index-handling a great deal simpler). If this apparent similarity can be proved, then it's fairly easy (see Exercise 22W.9) to show that subdivision curves are in fact exactly B-splines.

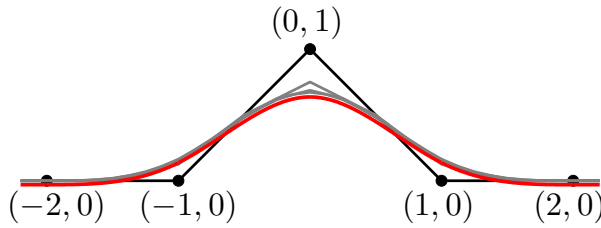


Figure 22W.38: A sequence of control points on the x -axis, except for a central one on the y -axis. Under repeated subdivision (the first two levels shown in faint grey lines), this converges to a smooth curve that's nonzero exactly on $-2 < x < 2$, and which appears identical to the degree three B-spline b_3 , which is plotted, slightly offset in y in red.

We won't actually prove this claim directly, but we will instead analyze the limit for a single point of the original polyline, namely P_0 . We'll show that under repeated subdivision, P_0 moves towards $(0, 3/4)$, and more generally, for any sequence of control points, P_0 moves towards $1/8P_{-1} + \frac{3}{4}P_0 + \frac{1}{8}P_1$, which happens to be $b_3(-2)P_{-2} + b_3(-1)P_{-1} + b_3(0)P_0 + b_3(1)P_1 + b_3(2)P_2$, i.e., exactly the point at $t = 0$ on the cubic B-spline defined by the vertices of the original polyline.

Surprisingly, this single computation, together with a small amount of continuity analysis, will suffice to prove that subdivision curves are B-splines.

22.16W Analysis of subdivision curves

We begin with some notation. We start with a control sequence $\{P_i\}$; we'll apply subdivision to get a new control sequence, and repeat this process. So keep track, we'll use a superscript; the initial sequence will be given a superscript of 0, so

$$P_i^0 = P_i. \quad (22.129W)$$

We'll associate to this the parametric polyline through these points, i.e.,

$$\gamma^0(t) = \sum_i b_1(t-i)P_i^0, \quad (22.130W)$$

as shown in Figure 22W.39.

The sequence arising from subdividing once will be called $\{P_i^1\}$, but the subscripts in this case are allowed to be half-integers; for the next level, quarter-integer subscripts are allowed, and so on. This has the advantage that the point arising from combining P_3^0 with its neighbors is called P_3^1 ; the point arising from combining P_3^1 with its neighbors is called P_3^2 , and it therefore makes sense to speak of

$$\lim_{n \rightarrow \infty} P_3^n, \quad (22.131W)$$

for example. For the next few paragraphs, we'll be studying the limit of P_0^n .

We'll also want to speak about the piecewise-linear parametric curve $\gamma^1(t)$ that passes through the points $\{P_i^1\}$, and has the property that for each integer or half-integer i , $\gamma^1(i) = P_i^1$. It's an easy exercise to see that γ^1 can be defined by

$$\gamma^1(t) = \sum_i b_1(2(t-i))P_i^1, \quad (22.132W)$$

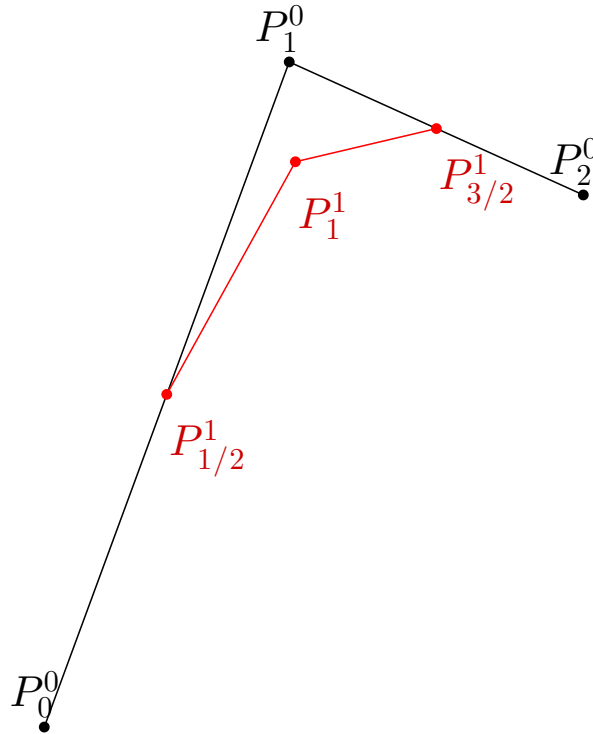


Figure 22W.39: Vertices in the initial control point sequence are labelled P_i^0 ; the degree-one B-spline (i.e., polyline) passing through them is γ^1 . For each i , $\gamma^1(i) = P_i^0$. The first level of subdivision (shown in red), has points labelled P_i^1 , but the subscripts are allowed to be integers or half-integers. In the second level subdivision, indices can be quarter-integers, and so on.

and that for further subdivision, we can define an analogous curve by

$$\gamma^n(t) = \sum_i b_1(2^n(t - i))P_i^n. \tag{22.133W}$$

With these parametric curves defined, we can state our goal more formally: we seek to show that for every t ,

$$\lim_{n \rightarrow \infty} \gamma^n(t) = \gamma(t), \tag{22.134W}$$

where

$$\gamma(t) = \sum_i P_i b_3(t - i). \tag{22.135W}$$

As we said above, we'll begin by showing this for $t = 0$. To do so, we'll study P_0^0 and its left and right neighbors, P_{-1}^0 and P_1^0 . We'll write the coordinates of these points as column vectors, and assemble them into a matrix,

$$\mathbf{N}^0 = [P_{-1}^0; P_0^0; P_1^0], \tag{22.136W}$$

where we've chosen the letter N to stand for "neighborhood". After one level of subdivision, P_0^0 changes to P_0^1 , and it gets two *new* neighbors: $P_{-1/2}^1$ and $P_{1/2}^1$. We

can similarly assemble *these* into a matrix

$$\mathbf{N}^1 = \left[P_{-\frac{1}{2}}^1; P_0^1; P_{\frac{1}{2}}^1 \right]. \quad (22.137W)$$

We can do the same for subsequent levels of subdivision. Now let's examine how these matrices are related. The rules for subdivision say that we insert new points at edge-midpoints, and move existing points to a $(1/8, 3/4, 1/8)$ weighted combination of their neighborhood. That is to say

$$\mathbf{N}^1 = \mathbf{N}^0 \begin{bmatrix} \frac{1}{2} & \frac{1}{8} & 0 \\ \frac{1}{2} & \frac{3}{4} & \frac{1}{2} \\ 0 & \frac{1}{8} & \frac{1}{2} \end{bmatrix}. \quad (22.138W)$$

Inline Exercise 22W.22: Explain why the first column of \mathbf{N}^1 is, as determined by this formula, is actually the midpoint of the edge from P_{-1}^0 and P_0^0 . Verify that the other two columns make sense as well.

Let \mathbf{S} denote this matrix, so that $\mathbf{N}^1 = \mathbf{N}^0 \mathbf{S}$. Then because the subdivision rule at every stage is exactly the same, we can also write

$$\mathbf{N}^2 = \mathbf{N}^1 \mathbf{S} \quad (22.139W)$$

$$= \mathbf{N}^0 \mathbf{S}^2, \quad (22.140W)$$

and more generally, that $\mathbf{N}^k = \mathbf{N}^0 \mathbf{S}^k$. To determine what happens when k gets large (i.e., where P_0 goes), we'll need to compute large powers of \mathbf{S} . Expressing large powers of matrices is almost always done via eigenvalues and eigenvectors; that's the approach we'll use here. We can write \mathbf{S} as a product

$$\mathbf{S} = \mathbf{RDL} \quad (22.141W)$$

$$= \left(\frac{1}{6} \begin{bmatrix} 1 & -1 & 1 \\ 4 & 0 & -2 \\ 1 & 1 & 1 \end{bmatrix} \right) \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \\ -3 & 0 & 3 \\ 2 & -1 & 2 \end{bmatrix}, \quad (22.142W)$$

where the rows of the matrix \mathbf{L} are left eigenvectors of \mathbf{S} , the columns of \mathbf{R} are right eigenvectors, and the diagonal matrix \mathbf{D} contains the eigenvalues.

Inline Exercise 22W.23: (a) Verify that this decomposition is correct by multiplying matrices. (b) Also verify that $\mathbf{LR} = \mathbf{RL} = \mathbf{I}$, and that \mathbf{L} really does consist of left-eigenvectors, by showing that $\mathbf{LS} = \mathbf{DL}$.

This decomposition lets us compute powers of \mathbf{S} easily. For instance,

$$\mathbf{S}^2 = (\mathbf{RDL})(\mathbf{RDL}) \quad (22.143W)$$

$$= (\mathbf{RD})(\mathbf{LR})(\mathbf{DL}) \quad (22.144W)$$

$$= (\mathbf{RD})(\mathbf{I})(\mathbf{DL}) \quad (22.145W)$$

$$= (\mathbf{RD}^2\mathbf{L}). \quad (22.146W)$$

Higher powers of \mathbf{S} are similar: $\mathbf{S}^k = \mathbf{R}\mathbf{D}^k\mathbf{L}$ for any integer k . We can use this to compute $\lim_{k \rightarrow \infty} \mathbf{S}^k$:

$$\lim_{k \rightarrow \infty} \mathbf{S}^k = \lim_{k \rightarrow \infty} \mathbf{R}\mathbf{D}^k\mathbf{L} = \mathbf{R} \left(\lim_{k \rightarrow \infty} \mathbf{D}^k \right) \mathbf{L} \quad (22.147W)$$

$$= \mathbf{R} \begin{bmatrix} 1^k & & \\ & \frac{1}{2}^k & \\ & & \frac{1}{4}^k \end{bmatrix} \mathbf{L} \quad (22.148W)$$

$$= \mathbf{R} \begin{bmatrix} 1 & & \\ & 0 & \\ & & 0 \end{bmatrix} \mathbf{L} = 1/6 \begin{bmatrix} 1 & 1 & 1 \\ 4 & 4 & 4 \\ 1 & 1 & 1 \end{bmatrix} \quad (22.149W)$$

For convenience, we'll call this \mathbf{S}^∞ .

With this in mind, we can see what happens to the neighborhood of P_0 under repeated subdivision. We know that

$$\mathbf{N}^k = \mathbf{S}^k \mathbf{N}^0; \quad (22.150W)$$


as $k \rightarrow \infty$, the right hand side approaches $\mathbf{S}^\infty \mathbf{N}^0$, which is

$$\begin{bmatrix} Q \\ Q \\ Q \end{bmatrix}, \quad (22.151W)$$

where $Q = \frac{1}{6}P_{-1} + \frac{4}{6}P_0 + \frac{1}{6}P_1$. In words, the center point P_0 approaches Q , as do its two closest neighbors at each level of subdivision. This limit point is exactly $\gamma(0)$, where γ is the degree-three B-spline for the control points $\{P_i\}$.

Inline Exercise 22W.24: Verify this claim.

So we've shown that P_0 , under repeated subdivision, approaches $\gamma(0)$. An essentially identical proof shows that P_i ($i = \dots, -2, -1, 0, 1, 2, \dots$) approaches

$\gamma(i)$.  A nice alternative proof is to let $Q_i = P_{i+1}$. Then the result we've just proved shows that Q_0 converges to the right point on *its* B-spline, but Q_0 is just P_1 , so a little index-shifting shows that P_1 converges to the proper point, and so on.

We'll now show that $P_{\frac{1}{2}}$ converges to $\gamma(\frac{1}{2})$; essentially the same proof will prove the same thing for any half-integer index. Furthermore, it will show that the same result is true for quarter-integer or eighth-integer indexes, etc. In short, we'll have shown that every 2-rational point t in the subdivision polygon converges to $\gamma(t)$.

Consider one level of subdivision applied to the control polygon $\{P_i^0\}$, creating the polygon $\{P_i^1\}$, as shown in Figure 22W.40.

Among the five red points in the figure, the central one is $P_{\frac{1}{2}}$. The argument of the preceding paragraphs shows that this point, under repeated subdivision, approaches a limit that is $\zeta(0)$, where ζ is the degree-3 B-spline based on the five red control points. But the analysis of Section 22.9.5W says that $\zeta(0)$ is exactly $\gamma(\frac{1}{2})$.

An analogous argument holds for every point of the first subdivision polygon, or of the second, or the third, and we can conclude that every dyadic point in the

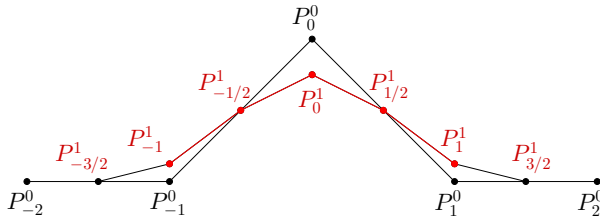


Figure 22W.40: One level of subdivision, near P_0^0 . The five points marked in red are exactly those discovered when we refined the control mesh in Section 22.9.5W.

original control polygon actually converges to the corresponding limit point on the B-spline defined by the control polygon.

What about all the points that aren't dyadic? They, too, converge to a point on the B-spline. \int One way to prove this is to show that the convergence of the γ^n to γ is *uniform* [21] on any closed interval (the proof is beyond the scope of this book). It then follows that the subdivision limit curve ζ is continuous. And if two continuous curves (the limit curve and the B-spline γ) agree on the dyadics, then they must agree everywhere. Hence the limit curve must be exactly the B-spline γ .

The eigenstructure of the subdivision matrix can be further analyzed to determine properties of the derivatives of the limit curve, although we know those already, since it's a B-spline. But for other subdivision schemes, where the exact nature of the limit curve may not be known, this kind of analysis is necessary.

22.17W Mathematical Topics

\int We now discuss a few more details of some mathematical properties of splines and subdivision.

22.17.1W Variation diminishing curves

Both Bézier and B-spline curves have the property that they are *variation diminishing*, i.e., every line has no more intersections with curve than with the control polygon for the curve. This apparently abstract property has interesting practical consequences, however: if you want to test whether a ray intersects a Bézier segment, you can test instead whether it intersects the control polygon; if not, you're done. But if it *does* intersect the control polygon, you can apply the de Casteljau algorithm to separate the polygon into two pieces, and test for an intersection with each part. Applying this approach recursively not only establishes whether an intersection exists, but narrows down the possible location of the intersection. This idea, generalized to surfaces, can be useful in raytracing spline-based surfaces.

In the course of simulations, one often needs to know whether two objects are interpenetrating (i.e., have collided during the previous step in the simulation). If the objects are in 2D and defined by spline curves, this means that one needs to test for intersections between spline curves. Fortunately, in the case of B-splines, that's relatively easy: one can compare the defining polygons, and if these do not intersect, then the splines do not intersect. On the other hand, if the polygons *do*

intersect, one can split the B-spline with a method that's closely related to the de Casteljau algorithm, the *Oslo algorithm* [4], inserting a new knot and altering the control hull to more closely approximate the spline. Doing so repeatedly again helps determine the absence of intersections or localize intersections that exist.

22.17.2W Data fitting

Three kinds of data fitting arise often in graphics:

- User-input data, like a sketched curve, sometimes needs to be smoothed and represented in a way that's easy to edit later. Most drawing programs have some form of free-form input, for instance.
- Scanned data, such as the geometry of some physical item, or the BRDF of some material, may be so large that it's impractical to work with it directly. It's often wise to choose some model, and fit the model to the data (“We'll represent the BRDF as a sum of a few specular peaks and a sum of low-degree spherical harmonics⁴.”)
- Computed data, such as the irradiance at each point of a surface, may be so expensive to compute that we can only afford a few samples. To estimate the values at nearby points, we may want to fit some relatively smoothly-varying function to the known data.

The first of these problems was addressed, in part, by Banks and Cohen [2], who performed real-time online fitting of data with B-splines. “Online” here means “the B-splines were updated as each new datum arrived, rather than being computed once the entire stroke was available.” The fitting included detecting discontinuities such as sharp corners in the sketched curve. Their fitting criterion was one of “how close is the B-spline to the input data?” together with a notion of a “fair curve.” A more modern approach might actually model the user's stroke as being a noisy sample of an underlying B-spline curve, with the noise at successive samples being correlated at some scale; when you try to draw a circle and fail slightly, the result is usually lumpy rather than jagged. With this model, one could then ask “among all underlying curves, for which one is this set of data most likely?” To formulate such a problem properly requires a model of the a priori probability of any particular model. Reasonable choices for this might include preferring few knots over many, small curvature over large, etc. With this model in hand, one could apply techniques like expectation maximization to fit a B-spline to the data.

For the second problem, in which the noise in the data is uncorrelated and more uniformly distributed, so that the data is generally referred to as a “point cloud,” Wang and Pottman [25] describe a computationally efficient approach to fitting by minimizing an approximation of the sum of the squared distances from each point to the curve.

For the third problem, it's essential to have some reasonable hypothesis about the interpolated version of the function. The simplest “agnostic” fitting is “nearest neighbor interpolation,” in which the value at a point P is determined by finding the nearest point Q at which the value is known, and then using the value there

4. *Spherical harmonics* are the analog, for the sphere, of the Fourier basis functions $\cos(nx)$ and $\sin(nx)$ are to the circle.

as the value for P . As the number of known points increases, this approach does about as well as possible, but when the number of known points is small, it is not terribly satisfactory at an intuitive level. Other versions – k -nearest-neighbor interpolation, where values from multiple nearby neighbors are combined – do somewhat better, and spline-interpolation of the data may be a reasonable choice if one believes that the underlying surface is smooth. But there is no single, simple answer; the entire subject of *density estimation* in statistics attempts to address this problem; not surprisingly, this is also closely related to problems in machine learning.

22.17.3W Adjusting subdivision curves

Imagine that in the course of subdividing a polygon, you accidentally made an error with one vertex at some level of subdivision, but continued onwards. The resulting limit curve would still be smooth, but it would be “wrong” in a small area — the region over which that vertex had an influence. Now invert that idea: suppose you started with a polygon and subdivided it to get a limit curve, but wanted to alter the shape of the limit curve by dragging some point (or part) of the limit curve to a new location. How would you alter the original polygon (or some intermediate-level vertex) to effect this change? This is the *direct manipulation* problem, and as stated, it’s ill-posed. One solution for moving a point is to move the entire original polygon by the required amount; the target point will move by that amount as well, of course. Another approach would be to find *some* vertex of the original polygon that influenced that point’s position, and simply move that one vertex enough to effect the desired change. There are an infinity of answers between these two. To address such problems, it’s traditional to “regularize” them — to add in some preference for certain answers over others, so that we don’t end up with lots of equally-good answers. A simple regularization for this problem would be that “the sum of the squared lengths of the displacements of the original vertices should be as small as possible.” If intermediate vertices can be moved as well as original ones, then a further regularization might prefer early-stage vertex motions over late-stage vertex motions, for instance. (This is completely analogous to the direct manipulation of spline curves in our drawing program example.)

For subdivision curves, a sophisticated approach to the direct manipulation problem has been developed by Zhou et al. [26].

22.17.4W Calculus of variations: an introduction

We asked, early in this chapter, “Among all parametric curves with domain $[0, 1]$ starting at P with tangent \mathbf{v} and ending at Q with tangent \mathbf{w} , which one minimizes

$$\int_0^1 \|\gamma''(t)\|^2 dt?'' \quad (22.152W)$$

In this section, we’ll briefly show how to convert such a problem (minimizing an integral over a family of functions) into a different problem (solving a differential equation with certain boundary conditions) which is often simpler. In doing so, we’ll ignore a great many technical details; those interested in learning more should consult a book on the calculus of variations [10].

Rather than solving the problem above, we’ll start with a much easier problem, because it illustrates almost all the important features while involving much less

work. The disadvantage is that it's completely contrived; it's not a problem anyone has ever wanted to solve.

Consider the family of curves

$$\gamma_s(t) = (t, s(t^2 - t) + 1), \quad (22.153W)$$

where s can be any real number, and t ranges from 0 to 1. The curve γ_0 is a straight line from $(0, 1)$ to $(1, 1)$. The curve γ_s also joins those same two points, no matter what s is, but when s is nonzero, the curve is no longer a straight line.

We can compute the "energy" of γ_s , defined by

$$E(s) = \int_0^1 \|\gamma'_s(t)\|^2 dt \quad (22.154W)$$

for each value of s . (The name "energy" is conventional, but it has nothing to do with ordinary energy in a physical sense; it just gives us a name to use for the next few paragraphs.) In fact, the energy for this particular problem is so simple that we can compute it explicitly. First, $\gamma'_s(t) = (1, s(2t-1))$, so $\|\gamma'_s(t)\|^2 = 1 + s^2(2t-1)^2$. Integrating explicitly gives

$$E(s) = 1 + \frac{s^2}{3}. \quad (22.155W)$$

It's easy to see that as we vary s , the function $E(s)$ has a minimum at $s = 0$. (If we were being fancy, we'd use calculus and compute $\frac{dE}{ds}$ and set it to zero and solve for s .) Thus, among all functions in our very limited class under consideration (the γ_s 's), there's one that minimizes energy, at it happens to be γ_0 .

Suppose, though, that we could not directly integrate. We could still determine which value of s minimized $E(s)$. Here's how. We write γ_s as a sum:

$$\gamma_s(t) = \gamma(t) + s(0, t^2 - t) \quad (22.156W)$$

where $\gamma(t) = (t, 1)$. Letting $h(t) = (0, t^2 - t)$, we say that γ_s is a **variation** of γ (or sometimes call h itself a variation). Writing

$$\gamma_s(t) = \gamma(t) + sh(t), \quad (22.157W)$$

we can rewrite the integral for E :

$$\begin{aligned} E(s) &= \int_0^1 \|\gamma'_s(t)\|^2 dt \\ &= \int_0^1 (\gamma'(t) + sh'(t)) \cdot (\gamma'(t) + sh'(t)) dt \\ &= \int_0^1 \gamma'(t) \cdot \gamma'(t) + 2s\gamma'(t) \cdot h'(t) + s^2h'(t) \cdot h'(t) dt \\ &= \int_0^1 \gamma'(t) \cdot \gamma'(t) dt + s \int_0^1 2\gamma'(t) \cdot h'(t) dt + s^2 \int_0^1 h'(t) \cdot h'(t) dt. \end{aligned}$$

Without filling in the explicit formulas for γ and h , let's ask what would happen if we tried to minimize $E(s)$ as a function of s . We'd compute

$$\begin{aligned}\frac{dE}{ds}(s) &= \frac{d}{ds} \left\{ \int_0^1 \gamma'(t) \cdot \gamma'(t) dt + s \int_0^1 2\gamma'(t) \cdot h'(t) dt + s^2 \int_0^1 h'(t) \cdot h'(t) dt \right\} \\ &= \int_0^1 2\gamma'(t) \cdot h'(t) dt + 2s \int_0^1 h'(t) \cdot h'(t) dt \\ &= \int_0^1 (2\gamma'(t) + 2sh'(t)) \cdot h'(t) dt.\end{aligned}$$

Without the explicit formulas for γ and h , it's not at all clear what value of s makes this expression zero. But let's integrate by parts, using $u = 2\gamma'(t) + 2sh'(t)$ and $dv = h'(t)dt$; we then get

$$\frac{dE}{ds}(s) = (2\gamma'(t) + 2sh'(t)) \cdot h(t) \Big|_0^1 - \int_0^1 (2\gamma''(t) + 2sh''(t)) \cdot h(t) dt.$$

Because $h(1) = h(0) = (0, 0)$, the first term is zero. And because $\gamma''(t) = 0$ everywhere, the second term is zero when $s = 0$.

Now look at that last computation again: the only property of h that we used was the fact that $h(1) = h(0) = (0, 0)$. That means that if we took *any* variation of γ that left the endpoints fixed (i.e., for which $h(1) = h(0) = (0, 0)$), we'd find that $s = 0$ was the least-energy curve among all the γ_s curves. So we can conclude that γ is the least energy curve between the two endpoints. To be explicit: suppose that μ was a curve between the points with lower energy. Then let $h = \mu - \gamma$. Considering all curves $\gamma_s(t) = \gamma(t) + sh(t)$, we'd find the minimum energy occurred at $s = 1$, but that contradicts the conclusion that the minimum, for any variation, occurs at $s = 0$.

Drawing this final conclusion depended critically on $\gamma''(t)$ being everywhere zero.

Now let's go a step further. Suppose we hadn't been given γ , either. We just want to find the least energy curve between $(0, 1)$ and $(1, 1)$. To be explicit, the problem is to find, among all curves with $[0, 1]$ as their domain, and satisfying $\gamma(0) = (0, 1)$ and $\gamma(1) = (1, 1)$, the one for which $E(\gamma)$ is a minimum.

Here's how we approach this problem: we *assume* that there's some particular minimum energy curve, and we call it γ . If we vary γ by forming


$$\gamma_s(t) = \gamma(t) + sh(t) \quad (22.158W)$$


for some function h with $h(0) = h(1) = (0, 0)$, we know that $E(s)$ is minimized when $s = 0$. That means that $\frac{dE}{ds}(0) = 0$. Working through the same integration by parts, we can conclude that

$$- \int_0^1 (2\gamma''(t) + 2sh''(t)) \cdot h(t) dt = 0 \quad (22.159W)$$

when $s = 0$, i.e.,

$$-2 \int_0^1 \gamma''(t) \cdot h(t) dt = 0 \quad (22.160W)$$

no matter what the variation h is! The only function f with the property that $\int f \cdot h = 0$ for every h is the everywhere zero function.  This is somewhat glib. The only *continuous* function with this property is the zero function. Furthermore, we don't know that $\int f \cdot h = 0$ for every function h ; we only know this for every function h that's zero at both ends of the interval. But it turns out that this is enough to imply the result.

 Here are some details. For the first part — is γ continuous? — the answer is implicit in the problem: the definition of energy relies on the derivative of γ , so we're assuming γ is differentiable, hence that it's continuous. On the other hand, our solution required integration by parts and the assumption that γ'' exists. So we really should have phrased the original problem as “Among all twice differentiable paths from $(0, 0)$ to $(1, 1)$, which has the least energy?” We'll now proceed with the analysis assuming that this is the question under discussion.

Let's look at the second question: if f is twice differentiable, and the integral of $f \cdot h$ is zero for every variation h that's zero at each endpoint, must f be everywhere zero?

First, consider any interval $[a, b] \subset [0, 1]$ with $0 < a < b < 1$. We'll divide the interval into quarters, so $a < q < c < r < b$, where c is the center point $((a + b)/2)$, and q and r are the quarter-points, i.e., $a + \frac{b-a}{4}$ and $b - \frac{b-a}{4}$ respectively.

Having done so, we can find a function h that's zero on $[0, a]$ and $[b, 1]$, and 1 on $[q, r]$, and continuous and differentiable on the whole interval. To do so, we just piece together five functions: the zero function on $[0, a]$, a function that rises from 0 to 1 on the interval $[a, q]$, with derivative 0 at each end, the constant function 1 on $[q, r]$, a function that falls from 1 to 0 on $[r, b]$ (again with zero derivatives at the ends), and the constant function 0 on $[b, 1]$. If we write $g(x) = x^2(3 - 2x)$, then the rising part can be written as $g(\frac{x-a}{q-a})$, and the falling part as $1 - g(\frac{x-r}{b-r})$. We'll call this five-part function a “bump on the interval $[a, b]$.”

Second, observe that although the function γ'' that we're working with maps $[0, 1]$ to \mathbf{R}^2 , we can work with each coordinate independently. So what we'll show is that if f is a twice differentiable real-valued function and

$$\int_0^1 f(t)h(t) dt = 0 \quad (22.161W)$$

for every differentiable function h on $[0, 1]$ with $h(0) = h(1) = 0$, then f must be zero.

Third, note that if we show that $f(t) = 0$ for every t that's strictly between 0 and 1, then $f(0) = f(1) = 0$ as well. (The proof is an application of the intermediate value theorem.)

Fourth, observe that if f satisfies Equation 22.161W, then so does kf for any real number k .

Finally, suppose that f is nonzero at some point t_0 strictly between 0 and 1. Replace f with $t \mapsto \frac{f(t)}{f(t_0)}$ by the fourth observation, and then we have $f(t_0) = 1$. We'll derive a contradiction from this assumption that f is nonzero.

Since f is continuous, we know that for any $\epsilon > 0$, there's a $\delta > 0$ such that for $t_0 - \delta < t < t_0 + \delta$, $|f(t) - 1| < \epsilon$. Picking $\epsilon = \frac{1}{2}$, we find that on the interval $(t_0 - \delta, t_0 + \delta)$, $f(t)$ is at least $\frac{1}{2}$. We can choose δ to be so small that $0 < t_0 - \delta < t_0 + \delta < 1$. (Just keep dividing δ by two until these inequalities hold.) As we do so, we get a possibly smaller interval on which f is at least $\frac{1}{2}$.

Now apply the first observation to build a function h that's zero outside the interval $(t_0 - \delta, t_0 + \delta)$, but has value 1 on the interval $(t_0 - \frac{\delta}{2}, t_0 + \frac{\delta}{2})$. What's the integral of fh on the unit interval?

$$\int_0^1 f(t)h(t) dt = \int_{t_0-\delta}^{t_0+\delta} f(t)h(t) dt \quad \text{because } h \text{ is zero outside this interval} \quad (22.162W)$$

$$\geq \int_{t_0-\delta}^{t_0+\delta} \frac{1}{2}h(t) dt \quad \text{because } f > \frac{1}{2} \text{ on this interval} \quad (22.163W)$$

$$\geq \frac{1}{2} \int_{t_0-\delta}^{t_0+\delta} h(t) dt \quad (22.164W)$$

$$\geq \frac{1}{2} \int_{t_0-\frac{\delta}{2}}^{t_0+\frac{\delta}{2}} h(t) dt \quad \text{because } h \geq 0 \quad (22.165W)$$

$$\geq \frac{1}{2} \int_{t_0-\frac{\delta}{2}}^{t_0+\frac{\delta}{2}} 1 dt \quad \text{because } h(t) = 1 \text{ on this interval} \quad (22.166W)$$

$$= \frac{\delta}{2} > 0. \quad (22.167W)$$

Since the integral of fh is supposed to be zero for every variation h , but it's nonzero for this one, the assumption that $f(t_0) \neq 0$ must have been invalid.

Thus we conclude that $-2\gamma''(t) = 0$ for all $t \in [0, 1]$. That makes γ linear, and the only linear function that goes from $(0, 1)$ to $(1, 1)$ is $\gamma(t) = (0, t)$.

The essential features of the argument above are these: first, we need to assume that the energy minimization problem has a solution, and that the solution is twice differentiable, and the second derivative is continuous; second, we said that for γ to be a minimum, every variation of γ had to increase E , so dE/ds had to be zero at $s = 0$, no matter what variation h we used; third, we used integration by parts to convert this to an integral of the form

$$\int_0^1 \text{expression involving } \gamma \cdot h = 0 \quad (22.168W)$$

which had to be zero for every variation h , which meant that the expression involving γ had to be zero everywhere.

Suppose we now try to minimize

$$\int_0^1 \|\gamma''(t)\|^2 dt \quad (22.169W)$$

which was our original problem, subject to the conditions that $\gamma(0) = P$, $\gamma(1) = Q$, $\gamma'(0) = \mathbf{v}$ and $\gamma'(1) = \mathbf{w}$. In this case, the only allowable variations are ones for which $h(0) = h(1) = (0, 0)$ and $h'(0) = h'(1) = (0, 0)$, because if the derivatives at the ends were not zero, then $\gamma_s(t) = \gamma(t) + sh(t)$ would not have $\gamma'_s(0) = \mathbf{v}$ and $\gamma'_s(1) = \mathbf{w}$ as required.

Once again we assume that a minimizing curve γ exists, and that it's differentiable, with continuous derivatives (in this case, it'll turn out to need a continuous

fourth derivative). Writing out the minimization problem, taking a derivative, and setting to zero yields

$$\begin{aligned}
 0 &= \frac{d}{ds} \left\{ \int_0^1 \gamma''(t) \cdot \gamma''(t) dt + s \int_0^1 2\gamma''(t) \cdot h''(t) dt + s^2 \int_0^1 h''(t) \cdot h''(t) dt \right\} \text{ at } s = 0 \\
 &= \int_0^1 2\gamma''(t) \cdot h''(t) dt + 2s \int_0^1 h''(t) \cdot h''(t) dt \text{ at } s = 0 \\
 &= \int_0^1 (2\gamma''(t) + 2sh''(t)) \cdot h''(t) dt \text{ at } s = 0 \\
 &= \int_0^1 2\gamma''(t) \cdot h''(t) dt.
 \end{aligned}$$

Integrating by parts twice (we won't show the details) gives

$$0 = \int_0^1 \gamma'''(t) \cdot h(t) dt. \quad (22.170W)$$

The “ uv ” terms for each integration by parts turn out to be zero because of the assumptions about h . The final equation tells us that $\gamma'''(t) = 0$ for all t , i.e., that γ must be a cubic.

For any integral expression like the ones we've just examined, there's an associated differential equation; for the first, the equation was $-2\gamma''(t) = 0$; for the second, it was $\gamma'''(t) = 0$. In general, this is called the **Euler-Lagrange** equation for the variational problem, and after a little practice, it becomes easy to see a general pattern and write down the Euler-Lagrange equation simply by looking at the integrand.

An essentially identical approach is used to solve similar problems in two dimensions. A classic is the **thin plate spline** problem, in which one seeks a function of two variables $f(x, y)$ that takes on particular values z_i at certain points (x_i, y_i) ($i = 1, \dots, n$), and minimizes the integral of $f_{xx}^2(x, y) + 2f_{xy}^2(x, y) + f_{yy}^2(x, y)$. This corresponds to the shape taken by a thin sheet of metal when it's deformed by applying point forces at certain points of the surface. The solution to this problem has the form

$$f(x, y) = A + Bx + Cy + \sum_{i=1}^{n-3} c_i \phi(\|(x - x_i, y - y_i)\|) \quad (22.171W)$$

where ϕ is the function $\phi(r) = r^2 \log r$. The actual values of A, B, C and the c_i are determined by setting $f(x_i, y_i) = z_i$, which gives n equations in the $n - 3$ unknown c_i together with A, B , and C .

Such “thin plate splines” have been applied in many situations in graphics, from morphing [16] to the interpolation of things like scattered samples of radiance as mentioned above. Their generalization to three dimensions also makes an appearance in the field of implicit surfaces (see Chapter 24).

22.18W Notes and further reading

The theory of splines was developed in applied mathematics, in trying to find ways to approximate a function $f : \mathbf{R} \rightarrow \mathbf{R}$. It was known that one could use polynomials to approximate any continuous function, but to get a good approximation

could require a very high degree for the approximating polynomial. The notion of “goodness of fit” was the so-called “sup-norm”, in which g fit f with accuracy c if $|f(x) - g(x)| < c$ for every x , which meant that very wiggly curves could be used to closely approximate smooth ones, as long as the wiggles were small. Splines provided a way to generate low-degree but accurate approximations, at the cost of having to specify a different polynomial on each of many intervals. B-splines were introduced by Schoenberg [19] in 1946, but they were not particularly central until the early 1970s, when Carl de Boor had been developing a recursive, stable method for evaluating B-splines. At that point Riesenfeld’s dissertation [?] showed that B-splines could also be used for *ab initio* modeling in many contexts, which (along with prior industrial adoption of splines for computer-aided modeling of existing artifacts) helped make B-splines a standard tool for shape modeling for decades.

The descriptions in this chapter of splines in terms of polynomial-bits-that-glue-up-nicely, energy-minimizing curves, repeated convolutions, and as polynomials related to probabilities barely scratch the surface of the subject; they’re also related to geometric constructions and to signal processing, for example.

The idea that the cubic B-spline basis curve can be written as a weighted sum of scaled-down copies of itself is related to wavelets and to fractals [23].

Subdivision curves were introduced in 1973 by Chaikin, who described a method for “corner cutting” of polygons to gradually smooth them out. At each stage, he introduced new points $\frac{1}{4}$ and $\frac{3}{4}$ of the way along each edge, and deleted the original vertices. The resulting polygons converged to a limit curve which can be analyzed much as we analyzed subdivision curves. The limit curve turns out to be a quadratic B-spline curve.

The subdivision curves we’ve described are the one-dimensional analog of the subdivision *surfaces* described by Catmull and Clark, which we’ll encounter in the next chapter. One view of such a subdivision scheme is that it provides a way to convert a discrete signal (a sequence of values, like $\dots, 0, 0, 1, 0, 0, \dots$) into a continuum signal (the degree-three B-spline basis curve, in the case of the subdivision scheme we concentrated on); because the map from sequences to continuum signals is linear, it can be written as convolution with some kernel (in this case, the cubic B-spline basis curve). The relation between the subdivision rule and the convolution kernel, particularly in the case of surfaces, is a rich one [20].

The mathematics of subdivision has been extremely carefully studied; modern papers on the subject require substantial mathematical sophistication. An article by Dyn and Levin [7] carries out the analysis hinted at (including the uniform convergence argument) in this chapter, which can be understood by a student who has completed a good undergraduate course in real analysis.

Exercises

Exercise 22W.7: We can derive the Hermite polynomials a second way. Writing the coefficients of the four Hermite polynomials p_1, \dots, p_4 as the rows of a matrix M , we have

$$\begin{bmatrix} p_1(t) \\ p_2(t) \\ p_3(t) \\ p_4(t) \end{bmatrix} = \mathbf{MT}(t) \quad (22.172W)$$

The condition that $p_1(0) = 1$, while $p_2(0) = p_3(0) = p_4(0) = 0$ tells us that

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} p_1(0) \\ p_2(0) \\ p_3(0) \\ p_4(0) \end{bmatrix} = \mathbf{MT}(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (22.173\text{W})$$

By writing down the constraints on the values of the p_i s at $t = 1$, and on their derivatives at $t = 0, 1$, we get three more such equations. The left hand side in each on is \mathbf{e}_i for $i = 1, 2, 3, 4$. These can be assembled, columnwise, into a matrix equation; the assembly of the four \mathbf{e}_i vectors produces the identity, so we get

$$\mathbf{I} = \mathbf{M}[\mathbf{T}(0); \mathbf{T}(1); \mathbf{T}'(0); \mathbf{T}'(1)]. \quad (22.174\text{W})$$

Fill in the four columns of this last matrix, and then invert the matrix to find \mathbf{M} , and from this write down the four Hermite polynomials.

Exercise 22W.8: (a) Show that the vectors $\mathbf{T}(t_1), \mathbf{T}(t_2), \mathbf{T}(t_3), \mathbf{T}(t_4)$ are linearly independent for any four distinct values of the t_i . (It may be easier to do this with two \mathbf{T} -vectors that only contain the constant and linear terms, and then with three \mathbf{T} -vectors that contain the constant, linear, and quadratic terms, and then induce a pattern. If not, you may find you want to read about Vandermonde matrices.)

(b) Conclude that if two Bézier curves agree at any four distinct values of t , then they are identical (as parametric curves).

Exercise 22W.9: We claimed that knowing that the piecewise-linear interpolants for the repeated subdivisions of the sequence of values $\dots, 0, 0, 1, 0, 0, \dots$ converged to a curve that was the cubic B-spline basis function b_3 was sufficient to show that subdivision of any polyline always led to cubic B-splines using as control points the initial polyline's vertices. We'll show that here in five steps. To simplify matters, let's pick an origin Z (for "zero") and write each original control point $P_i = Z + p_i$, so that the p_i are vectors in \mathbf{R}^2 . (The proof we'll write down will clearly generalize to \mathbf{R}^n .) The broad idea of the proof is first, that the rule for subdivision applies to the x -coordinates and the y -coordinates in identical ways, so we might as well study them one at a time, i.e., look at subdivision on sequences of numbers, and second, that sequences of numbers can be written as sums of multiples of translates of the sequence $\dots, 0, 0, 1, 0, 0, \dots$. Because subdivision is linear, we can bring the subdivision operation inside the sum and scalar multiples. Because our subdivision rule is shift-equivariant (which we define below), we can also pull it inside the translation.

(a) Explain why if $\{p_i\}_{i \in \mathbf{Z}}$ and $\{q_i\}_{i \in \mathbf{Z}}$ are polylines that lead, through subdivision, to limit curves γ_p and γ_q respectively, then $\{p_i + q_i\}_{i \in \mathbf{Z}}$ leads to a limit curve γ_{p+q} with the property that $\gamma_{p+q} = \gamma_p + \gamma_q$. Also explain why, if each p_i is $(0, 0)$, the limit curve γ_p satisfies $\gamma_p(t) = (0, 0)$ for all t . Finally, argue that if $\{x_i\}$ and $\{y_i\}$ are sequences of numbers, then applying the subdivision rule to each sequence to get limit-functions f_x and f_y means that applying subdivision to the point-sequence $\{(x_i, y_i)\}$ will give a limit curve (f_x, f_y) .

(b) Writing a polyline $\{s_i\}_{i \in \mathbf{Z}}$ as $\{(x_i, y_i)\}_{i \in \mathbf{Z}}$, and $\{p_i = (x_i, 0)\}_{i \in \mathbf{Z}}$ and $\{q_i = (0, y_i)\}_{i \in \mathbf{Z}}$, apply the results of part a to show that if the limit curve of subdivision on the three curves are γ_s, γ_p , and γ_q , then $\gamma_s = \gamma_p + \gamma_q$, and that (for instance), $\gamma_p = (f, 0)$, where f is the limit-function for subdivision of the value-sequence $\{x_i\}$.

These first two parts show that the action of subdivision on a sequence of points depends only on its action on sequences of *numbers*.

(c) Show that subdivision on number-sequences is *shift-equivariant*, meaning that if the sequence $\{i \mapsto x_i\}$ leads to a limit function $t \mapsto f(t)$, explain why the sequence the sequence $\{i \mapsto x_{i+1}\}$ leads to the limit function $t \mapsto f(t - 1)$. If $\mathbf{x} = \{i \mapsto x_i\}$ is a sequence, denote by $T(\mathbf{x})$ (the “translate” of the sequence) the sequence $\{i \mapsto x_{i+1}\}$.

(d) Let \mathbf{u} be the sequence with all values 0 except for $u_0 = 1$. Show that the sequence $\mathbf{x}\{i \mapsto x_i\}$ can be written as a sum of multiples of shifts of u , namely

$$\mathbf{x} = \sum_{i=-\infty}^{\infty} x_i T^i(u)$$

(e) Conclude that since subdivision applied to u results in b_3 , subdivision applied to \mathbf{x} results in the limit function

$$\mathbf{x} = \sum_{i=-\infty}^{\infty} x_i b_3(t - i)$$

and then that subdivision of the sequence $\{p_i\}_{i \in \mathbf{Z}}$ leads to the limit curve $\gamma_p(t) = \sum_{i=-\infty}^{\infty} p_i b_3(t - i)$.

Bibliography

- 1 Marc Alexa. Linear combination of transformations. *ACM Trans. Graph.*, 21(3):380–387, July 2002. 43
- 2 Michael Banks and Elaine Cohen. Real time spline curves from interactively sketched data. In *Proceedings of the 1990 symposium on Interactive 3D graphics*, I3D '90, pages 99–107, New York, NY, USA, 1990. ACM. 140
- 3 E. Catmull and J. Clark. Seminal graphics. chapter Recursively generated B-spline surfaces on arbitrary topological meshes, pages 183–188. ACM, New York, NY, USA, 1998. 162
- 4 E. Cohen, T. Lyche, and R. Riesenfeld. Discrete n -splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14(2):87–111, October 1980. 140
- 5 S. A. Coons. Surfaces for computer-aided design of space forms. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1967. 155
- 6 Carl de Boor. On calculating with b-splines. *J. Approx. Theory*, page 50–62, 1972. 110
- 7 Nira Dyn and David Levin. Analysis of hermite-interpolatory subdivision schemes, 1998. 148
- 8 Gerald Farin and Dianne Hansford. *The Essentials of CAGD*. A K Peters, Ltd., Natick, MA, 2000. 124
- 9 David R. Forsey and Richard H. Bartels. Hierarchical b-spline refinement. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 205–212, New York, NY, USA, 1988. ACM. 162
- 10 I.M. Gelfand, S.V. Fomin, and R.A. Silverman. *Calculus of Variations*. Dover Books on Mathematics. Dover Publications, 2000. 141
- 11 Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using catmull-clark surfaces. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 35–44, New York, NY, USA, 1993. ACM. 162, 165, 167, 175
- 12 W.R. Hamilton. *Lectures on Quaternions: Containing a Systematic Statement of a New Mathematical Method; of which the Principles Were Communicated*

- in 1843 to the Royal Irish Academy; and which Has Since Formed the Subject of Successive Courses of Lectures, Delivered in 1848 and Subsequent Years, in the Halls of Trinity College, Dublin: with Numerous Illustrative Diagrams, and with Some Geometrical and Physical Applications.* Number v. 2-4 in Lectures on Quaternions: Containing a Systematic Statement of a New Mathematical Method; of which the Principles Were Communicated in 1843 to the Royal Irish Academy; and which Has Since Formed the Subject of Successive Courses of Lectures, Delivered in 1848 and Subsequent Years, in the Halls of Trinity College, Dublin: with Numerous Illustrative Diagrams, and with Some Geometrical and Physical Applications. Hodges and Smith, 1853. 37
- 13 Leif Kobbelt. \mathbb{S}^3 -subdivision. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 103–112, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 169, 171
 - 14 Denis Kovacs, Jason Mitchell, Shanon Drone, and Denis Zorin. Real-time creased approximate subdivision surfaces. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 155–160, New York, NY, USA, 2009. ACM. 177
 - 15 Adi Levin. Modified subdivision surfaces with continuous curvature. *ACM Trans. Graph.*, 25(3):1035–1040, July 2006. 168
 - 16 Peter Litwinowicz and Lance Williams. Animating images with drawings. In *SIGGRAPH 1994 Proceedings*, pages 409–412. ACM, 1994. 147
 - 17 C.T. Loop. *Smooth Subdivision Surfaces Based on Triangles*. Department of Mathematics, University of Utah, 1987. 169
 - 18 Lee Markosian, Jonathan M. Cohen, Thomas Crulli, and John Hughes. Skin: a constructive approach to modeling free-form shapes. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 393–400, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. 177
 - 19 I.J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quart. Appl. Math.*, 4:45–99, 1120141, 1946. 104, 148
 - 20 Peter Schröder and Wim Sweldens. Spherical wavelets: efficiently representing functions on the sphere. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 161–172, New York, NY, USA, 1995. ACM. 148, 168
 - 21 M. Spivak. *Calculus*. Addison-Wesley world student series. Publish or Perish, 1980. 139
 - 22 Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 395–404, New York, NY, USA, 1998. ACM. 167, 174
 - 23 R. Szeliski and D. Terzopoulos. From splines to fractals. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '89, pages 51–60, New York, NY, USA, 1989. ACM. 148
 - 24 Alex Vlachos, Jörg Peters, Chas Boyd, and Jason L. Mitchell. Curved pn triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics*,

- I3D '01, pages 159–166, New York, NY, USA, 2001. ACM. 172
- 25 Wenping Wang, Helmut Pottmann, and Yang Liu. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Trans. Graph.*, 25(2):214–238, April 2006. 140
- 26 Kun Zhou, Xin Huang, Weiwei Xu, Baining Guo, and Heung-Yeung Shum. Direct manipulation of subdivision surfaces on gpus. *ACM Trans. Graph.*, 26(3), July 2007. 141
- 27 Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques, SIGGRAPH '97*, pages 259–268, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 177