14 Sampling

Many applications in graphics require "fair" sampling of unusual spaces, such as the space of all possible lines. For example, we might need to generate random edges within a pixel, or random sample points on a pixel that vary in density according to some density function. This chapter provides the machinery for such probability operations. These techniques will also prove useful for numerically evaluating complicated integrals using *Monte Carlo integration*, also covered in this chapter.

14.1 Integration

Although the words "integral" and "measure" often seem intimidating, they relate to some of the most intuitive concepts found in mathematics, and they should not be feared. For our very non-rigorous purposes, a *measure* is just a function that maps subsets to $\mathbb{R}^+$ in a manner consistent with our intuitive notions of length, area, and volume. For example, on the 2D real plane $\mathbb{R}^2$, we have the area measure A which assigns a value to a set of points in the plane. Note that A is just a function that takes pieces of the plane and returns area. This means the domain of A is all possible subsets of $\mathbb{R}^2$, which we denote as the *power set* $\mathcal{P}(\mathbb{R}^2)$. Thus, we can characterize A in arrow notation:

$$A : \mathcal{P}(\mathbb{R}^2) \to \mathbb{R}^+.$$

An example of applying the area measure shows that the area of the square with side length one is one:

$$A([a, a+1] \times [b, b+1]) = 1,$$

where (a, b) is just the lower left-hand corner of the square. Note that a single point such as $(3, 7)$ is a valid subset of $\mathbb{R}^2$ and has zero area: $A((3, 7)) = 0$. The same is true of the set of points S on the x-axis, $S = (x, y)$ such that $(x, y) \in \mathbb{R}^2$ and $y = 0$, i.e., $A(S) = 0$. Such sets are called *zero measure sets*.

To be considered a measure, a function has to obey certain area-like properties. For example, we have a function $\mu : \mathcal{P}(\mathbb{S}) \to \mathbb{R}^+$. For μ to be a measure, the following conditions must be true:

1. The measure of the empty set is zero: $\mu(\emptyset) = 0$,

2. The measure of two distinct sets together is the sum of their measure alone. This rule with possible intersections is

$$\mu(A \cup B) = \mu(A) + \mu(B) - \mu(A \cap B),$$

where $\cup$ is the set union operator and $\cap$ is the set intersection operator.

When we actually compute measures, we usually use *integration*. We can think of integration as really just notation:

$$A(S) \equiv \int_{x \in S} dA(\mathbf{x}).$$

You can informally read the right-hand side as "take all points $\mathbf{x}$ in the region S, and sum their associated differential areas." The integral is often written other ways including

$$\int_S dA, \qquad \int_{\mathbf{x} \in S} d\mathbf{x}, \qquad \int_{\mathbf{x} \in S} dA_{\mathbf{x}}, \qquad \int_{\mathbf{x}} d\mathbf{x}.$$

All of the above formulas represent "the area of region S." We will stick with the first one we used, because it is so verbose it avoids ambiguity. To evaluate such integrals analytically, we usually need to lay down some coordinate system and use our bag of calculus tricks to solve the equations. But have no fear if those skills have faded, as we usually have to numerically approximate integrals, and that requires only a few simple techniques which are covered later in this chapter.

Given a measure on a set $\mathbb{S}$, we can always create a new measure by weighting with a nonnegative function $w : \mathbb{S} \to \mathbb{R}^+$. This is best expressed in integral notation. For example, we can start with the example of the simple area measure on $[0,1]^2$:

$$\int_{\mathbf{x}\in[0,1]^2} dA(\mathbf{x}),$$

and we can use a "radially weighted" measure by inserting a weighting function of radius squared:

$$\int_{\mathbf{x}\in[0,1]^2} \|\mathbf{x}\|^2 dA(\mathbf{x}).$$

To evaluate this analytically, we can expand using a Cartesian coordinate system with $dA \equiv dx\,dy$:

$$\int_{\mathbf{x}\in[0,1]^2} \|\mathbf{x}\|^2 dA(\mathbf{x}) = \int_{x=0}^{1}\int_{y=0}^{1} (x^2+y^2)\ dx\,dy.$$

The key thing here is that if you think of the $\|\mathbf{x}\|^2$ term as married to the dA term, and that these together form a new measure, we can call that measure ν. This would allow us to write $\nu(S)$ instead of the whole integral. If this strikes you as just a bunch of notation and bookkeeping, you are right. But it does allow us to write down equations that are either compact or expanded depending on our preference.

14.1.1 Measures and Averages

Measures really start paying off when taking averages of a function. You can only take an average with respect to a particular measure, and you would like to select a measure that is "natural" for the application or domain. Once a measure is chosen, the average of a function f over a region S with respect to measure μ is

$$\text{average}(f) \equiv \frac{\int_{x\in S} f(\mathbf{x})\ d\mu(\mathbf{x})}{\int_{x\in S} d\mu(\mathbf{x})}.$$

For example, the average of the function $f(x,y)=x^2$ over $[0,2]^2$ with respect to the area measure is

$$\text{average}(f) \equiv \frac{\int_{x=0}^{2}\int_{y=0}^{2} x^2\ dx\,dy}{\int_{x=0}^{2}\int_{y=0}^{2}\ dx\,dy} = \frac{4}{3}.$$

This machinery helps solve seemingly hard problems where choosing the measure is the tricky part. Such problems often arise in *integral geometry*, a field that studies measures on geometric entities, such as lines and planes. For example, one might want to know the average length of a line through $[0,1]^2$. That is, by definition,

$$\text{average(length)} = \frac{\int_{\text{lines } L \text{ through } [0,1]^2} \text{length}(L) d\mu(L)}{\int_{\text{lines } L \text{ through } [0,1]^2} d\mu(L)}.$$

All that is left, once we know that, is choosing the appropriate μ for the application. This is dealt with for lines in the next section.

14.1.2 Example: Measures on the Lines in the 2D Plane

What measure μ is "natural"?

If you parameterize the lines as $y = mx + b$, you might think of a given line as a point (m, b) in "slope-intercept" space. An easy measure to use would be $dm\,db$, but this would not be a "good" measure in that not all equal size "bundles" of lines would have the same measure. More precisely, the measure would not be invariant with respect to change of coordinate system. For example, if you took all lines through the square $[0,1]^2$, the measure of lines through it would not be the same as the measure through a unit square rotated 45 degrees. What we would really like is a "fair" measure that does not change with rotation or translation of a set of lines. This idea is illustrated in Figures 14.1 and 14.2.

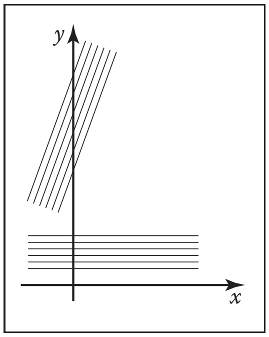


Figure 14.1. These two bundles of lines should have the same measure. They have different intersection lengths with the *y*-axis so using *db* would be a poor choice for a differential measure.

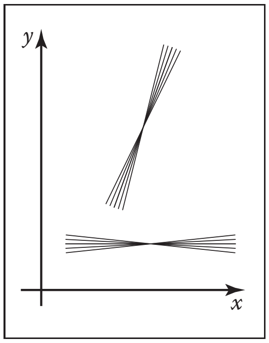


Figure 14.2. These two bundles of lines should have the same measure. Since they have different values for change in slope, using *dm* would be a poor choice for a differential measure.

To develop a natural measure on the lines, we should first start thinking of them as points in a dual space. This is a simple concept: the line $y = mx + b$ can be specified as the point (m, b) in a slope-intercept space. This concept is illustrated in Figure 14.3. It is more straightforward to develop a measure in (ϕ, b) space. In that space b is the y-intercept, while ϕ is the angle the line makes with the x-axis, as shown in Figure 14.4. Here, the differential measure $d\phi\,db$ almost works, but it would not be fair due to the effect shown in Figure 14.1. To account for the larger span b that a constant width bundle of lines makes, we must add a cosine factor:

$$d\mu = \cos\phi \; d\phi \; db.$$

It can be shown that this measure, up to a constant, is the only one that is invariant with respect to rotation and translation.

This measure can be converted into an appropriate measure for other parameterizations of the line. For example, the appropriate measure for (m, b) space is

$$d\mu = \frac{dm\; db}{(1+m^2)^{\frac{3}{2}}}.$$

For the space of lines parameterized in (u, v) space,

$$ux + vy + 1 = 0,$$

the appropriate measure is

$$d\mu = \frac{du\ dv}{(u^2 + v^2)^{\frac{3}{2}}}.$$

For lines parameterized in terms of (a, b), the x-intercept and y-intercept, the measure is

$$d\mu = \frac{ab\ da\ db}{(a^2 + b^2)^{\frac{3}{2}}}.$$

Note that any of those spaces are equally valid ways to specify lines, and which is best depends upon the circumstances. However, one might wonder whether there exists a coordinate system where the measure of a set of lines is just an area in the dual space. In fact, there is such a coordinate system, and it is delightfully simple; it is the *normal coordinates* which specify a line in terms of the normal distance from the origin to the line, and the angle the normal of the line makes with respect to the x-axis (Figure 14.5). The implicit equation for such lines is

$$x \cos\theta + y \sin\theta - p = 0.$$

And, indeed, the measure in that space is

$$d\mu = dp\ d\theta.$$

We shall use these measures to choose fair random lines in a later section.

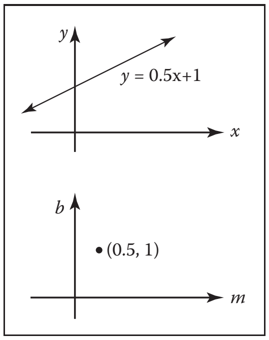


Figure 14.3. The set of points on the line $y = mx + b$ in (x, y) space can also be represented by a single point in (m, b) space so the top line and the bottom point represent the same geometric entity: a 2D line.

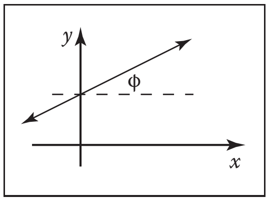


Figure 14.4. In angle-intercept space we parameterize the line by angle $\phi \in [-\pi/2, \pi/2)$ rather than slope.

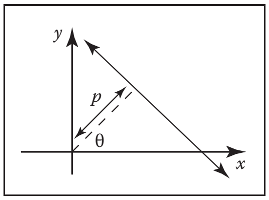


Figure 14.5. The normal coordinates of a line use the normal distance to the origin and an angle to specify a line.

14.1.3 Example: Measure of Lines in 3D

In 3D there are many ways to parameterize lines. Perhaps, the simplest way is to use their intersection with a particular plane along with some specification of their orientation. For example, we could chart the intersection with the xy plane along with the spherical coordinates of its orientation. Thus, each line would be specified as a (x, y, θ, ϕ) quadruple. This shows that lines in 3D are 4D entities, i.e., they can be described as points in a 4D space.

The differential measure of a line should not vary with (x, y), but bundles of lines with equal cross section should have equal measure. Thus, a fair differential measure is

$$d\mu = dx\, dy\ \sin\theta\, d\theta\, d\phi.$$

Another way to parameterize lines is to chart the intersection with two parallel planes. For example, if the line intersects the plane $z = 0$ at $(x = u, y = v)$ and the plane $z = 1$ at $(x = s, y = t)$, then the line can be described by the quadruple (u, v, s, t). Note, that like the previous parameterization, this one is degenerate for lines parallel to the xy plane. The differential measure is more complicated for this parameterization although it can be approximated as

$$d\mu \approx du\, dv\, a\, ds\, dt,$$

for bundles of lines nearly parallel to the z-axis. This is the measure often implicitly used in image-based rendering.

For sets of lines that intersect a sphere, we can use the parameterization of the two points where the line intersects the sphere. If these are in spherical coordinates, then the point can be described by the quadruple $(\theta_1, \phi_1, \theta_2, \phi_2)$ and the measure is just the differential area associated with each point:

$$d\mu = \sin\theta_1\, d\theta_1\, d\phi_1\ \sin\theta_2\, d\theta_2\, d\phi_2.$$

This implies that picking two uniform random endpoints on the sphere results in a line with uniform density. This observation was used to compute form-factors by Mateu Sbert in his dissertation (Sbert, 1997).

Note that sometimes we want to parameterize directed lines, and sometimes we want the order of the endpoints not to matter. This is a bookkeeping detail that is especially important for rendering applications where the amount of light flowing along a line is different in the two directions along the line.

14.2 Continuous Probability

Many graphics algorithms use probability to construct random samples to solve integration and averaging problems. This is the domain of applied continuous probability which has basic connections to measure theory.

14.2.1 One-Dimensional Continuous Probability Density Functions

Loosely speaking, a *continuous random variable* x is a scalar or vector quantity that "randomly" takes on some value from the real line $\mathbb{R} = (-\infty, +\infty)$. The behavior of x is entirely described by the distribution of values it takes. This distribution of values can be quantitatively described by the *probability density function* (pdf), p, associated with x (the relationship is denoted $x \sim p$). The probability that x assumes a particular value in some interval $[a, b]$ is given by the following integral:

$$\text{Probability}(x \in [a, b]) = \int_a^b p(x)dx. \tag{14.1}$$

Loosely speaking, the probability density function p describes the relative likelihood of a random variable taking a certain value; if $p(x_1) = 6.0$ and $p(x_2) = 3.0$, then a random variable with density p is twice as likely to have a value "near" x_1 than it is to have a value near x_2. The density p has two characteristics:

$$p(x) \geq 0 \quad \text{(probability is nonnegative)}, \tag{14.2}$$

$$\int_{-\infty}^{+\infty} p(x)dx = 1 \quad (\text{Probability}(x \in \mathbb{R}) = 1). \tag{14.3}$$

As an example, the *canonical* random variable ξ takes on values between zero (inclusive) and one (non-inclusive) with uniform probability (here *uniform* simply means each value for ξ is equally likely). This implies that the probability density function q for ξ is

$$q(\xi) = \begin{cases} 1 & \text{if } 0 \leq \xi < 1, \\ 0 & \text{otherwise,} \end{cases}$$

The space over which ξ is defined is simply the interval $[0, 1)$. The probability that ξ takes on a value in a certain interval $[a, b] \in [0, 1)$ is

$$\text{Probability}(a \leq \xi \leq b) = \int_a^b 1\, dx \;=\; b - a.$$

14.2.2 One-Dimensional Expected Value

The average value that a real function f of a one-dimensional random variable with underlying pdf p will take on is called its *expected value*, $E(f(x))$ (sometimes written $Ef(x)$):

$$E(f(x)) = \int f(x)p(x)dx.$$

The expected value of a one-dimensional random variable can be calculated by setting $f(x) = x$. The expected value has a surprising and useful property: the expected value of the sum of two random variables is the sum of the expected values of those variables:

$$E(x + y) = E(x) + E(y),$$

for random variables x and y. Because functions of random variables are themselves random variables, this linearity of expectation applies to them as well:

$$E(f(x) + g(y)) = E(f(x)) + E(g(y)).$$

An obvious question to ask is whether this property holds if the random variables being summed are correlated (variables that are not correlated are called *independent*). This linearity property in fact does hold *whether or not* the variables are independent! This summation property is vital for most Monte Carlo applications.

14.2.3 Multidimensional Random Variables

The discussion of random variables and their expected values extends naturally to multidimensional spaces. Most graphics problems will be in such higher-dimensional spaces. For example, many lighting problems are phrased on the surface of the hemisphere. Fortunately, if we define a measure μ on the space the random variables occupy, everything is very similar to the one-dimensional case. Suppose the space S has associated measure μ; for example S is the surface of a sphere and μ measures area. We can define a pdf $p : S \mapsto \mathbb{R}$, and if x is a random variable with $x \sim p$, then the probability that x will take on a value in some region $S_i \subset S$ is given by the integral

$$\text{Probability}(x \in S_i) = \int_{S_i} p(x) d\mu.$$

Here *Probability* $($*event*$)$ is the probability that *event* is true, so the integral is the probability that x takes on a value in the region S_i.

In graphics, S is often an area ($d\mu = dA = dxdy$) or a set of directions (points on a unit sphere: $d\mu = d\omega = \sin\theta \, d\theta \, d\phi$). As an example, a two-dimensional random variable α is a uniformly distributed random variable on a disk of radius R. Here *uniformly* means uniform with respect to area, e.g., the way a bad dart player's hits would be distributed on a dart board. Since it is uniform, we know that $p(\alpha)$ is some constant. From the fact that the area of the disk is πr^2 and that the total probability is one, we can deduce that

$$p(\alpha) = \frac{1}{\pi R^2}.$$

This means that the probability that α is in a certain subset S_1 of the disk is just

$$\text{Probability}(\alpha \in S_1) = \int_{S_1} \frac{1}{\pi R^2} dA.$$

This is all very abstract. To actually use this information, we need the integral in a form we can evaluate. Suppose S_i is the portion of the disk closer to the center than the perimeter. If we convert to polar coordinates, then α is represented as a (r, ϕ) pair, and S_1 is the region where $r < R/2$. Note, that just because α is uniform, it does not imply that ϕ or r are necessarily uniform (in fact, ϕ is uniform, and r is not uniform). The differential area dA is just $r\,dr\,d\phi$. Thus,

$$\text{Probability}\left(r < \frac{R}{2}\right) = \int_0^{2\pi} \int_0^{\frac{R}{2}} \frac{1}{\pi R^2} r\,dr\,d\phi = 0.25.$$

The formula for expected value of a real function applies to the multidimensional case:

$$E(f(x)) = \int_S f(x)p(x)d\mu,$$

where $x \in S$ and $f : S \mapsto \mathbb{R}$, and $p : S \mapsto \mathbb{R}$. For example, on the unit square $S = [0,1] \times [0,1]$ and $p(x,y) = 4xy$, the expected value of the x coordinate for $(x,y) \sim p$ is

$$\begin{aligned} E(x) &= \int_S f(x,y)p(x,y)dA \\ &= \int_0^1 \int_0^1 4x^2 y\,dx\,dy \\ &= \frac{2}{3}. \end{aligned}$$

Note that here $f(x,y) = x$.

14.2.4 Variance

The *variance*, $V(x)$, of a one-dimensional random variable is, by definition, the expected value of the square of the difference between x and $E(x)$:

$$V(x) \equiv E([x - E(x)]^2).$$

Some algebraic manipulation gives the non-obvious expression:

$$V(x) = E(x^2) - [E(x)]^2 .$$

The expression $E([x - E(x)]^2)$ is more useful for thinking intuitively about variance, while the algebraically equivalent expression $E(x^2) - [E(x)]^2$ is usually convenient for calculations. The variance of a sum of random variables is the sum of the variances *if the variables are independent*. This summation property of variance is one of the reasons it is frequently used in analysis of probabilistic models. The square root of the variance is called the *standard deviation*, σ, which gives some indication of expected absolute deviation from the expected value.

14.2.5 Estimated Means

Many problems involve sums of independent random variables x_i, where the variables share a common density p. Such variables are said to be *independent identically distributed* (iid) random variables. When the sum is divided by the number of variables, we get an estimate of $E(x)$:

$$E(x) \approx \frac{1}{N}\sum_{i=1}^{N} x_i.$$

As N increases, the variance of this estimate decreases. We want N to be large enough so that we have confidence that the estimate is "close enough." However, there are no sure things in Monte Carlo; we just gain statistical confidence that our estimate is good. To be sure, we would have to have $N = \infty$. This confidence is expressed by the *Law of Large Numbers*:

$$\text{Probability}\left[E(x) = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} x_i\right] = 1.$$

14.3 Monte Carlo Integration

In this section, the basic Monte Carlo solution methods for definite integrals are outlined. These techniques are then straightforwardly applied to certain integral problems. All of the basic material of this section is also covered in several of the classic Monte Carlo texts. (See the Notes section at the end of this chapter.)

As discussed earlier, given a function $f : S \mapsto \mathbb{R}$ and a random variable $x \sim p$, we can approximate the expected value of $f(x)$ by a sum:

$$E(f(x)) = \int_{x\in S} f(x)p(x)d\mu \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i). \tag{14.4}$$

Because the expected value can be expressed as an integral, the integral is also approximated by the sum. The form of Equation (14.4) is a bit awkward; we would usually like to approximate an integral of a single function g rather than a product fp. We can accomplish this by substituting $g = fp$ as the integrand:

$$\int_{x \in S} g(x) d\mu \approx \frac{1}{N} \sum_{i=1}^{N} \frac{g(x_i)}{p(x_i)}. \tag{14.5}$$

For this formula to be valid, p must be positive when g is nonzero.

So to get a good estimate, we want as many samples as possible, and we want the g/p to have a low variance (g and p should have a similar shape). Choosing p intelligently is called *importance sampling*, because if p is large where g is large, there will be more samples in important regions. Equation (14.4) also shows the fundamental problem with Monte Carlo integration: *diminishing return*. Because the variance of the estimate is proportional to $1/N$, the standard deviation is proportional to $1/\sqrt{N}$. Since the error in the estimate behaves similarly to the standard deviation, we will need to quadruple N to halve the error.

Another way to reduce variance is to partition S, the domain of the integral, into several smaller domains S_i, and evaluate the integral as a sum of integrals over the S_i. This is called *stratified sampling*, the technique that jittering employs in pixel sampling (Chapter 4). Normally only one sample is taken in each S_i (with density p_i), and in this case the variance of the estimate is:

$$var\left(\sum_{i=1}^{N} \frac{g(x_i)}{p_i(x_i)}\right) = \sum_{i=1}^{N} var\left(\frac{g(x_i)}{p_i(x_i)}\right). \tag{14.6}$$

It can be shown that the variance of stratified sampling is never higher than unstratified if all strata have equal measure:

$$\int_{S_i} p(x) d\mu = \frac{1}{N} \int_S p(x) d\mu.$$

The most common example of stratified sampling in graphics is jittering for pixel sampling as discussed in Section 13.4.

As an example of the Monte Carlo solution of an integral I, set $g(x)$ equal to x over the interval (0, 4):

$$I = \int_0^4 x\, dx = 8. \tag{14.7}$$

The impact of the shape of the function p on the variance of the N sample estimates is shown in Table 14.1. Note that the variance is reduced when the shape of p is similar to the shape of g. The variance drops to zero if $p = g/I$, but

Method	**Sampling function**	**Variance**	**Samples needed for standard error of 0.008**
importance	$(6-x)/(16)$	$56.8N^{-1}$	887,500
importance	$1/4$	$21.3N^{-1}$	332,812
importance	$(x+2)/16$	$6.3N^{-1}$	98,437
importance	$x/8$	0	1
stratified	$1/4$	$21.3N^{-3}$	70

Table 14.1. Variance for Monte Carlo estimate of $\int_0^4 x\,dx$.

I is not usually known or we would not have to resort to Monte Carlo. One important principle illustrated in Table 14.1 is that stratified sampling is often *far* superior to importance sampling (Mitchell, 1996). Although the variance for this stratification on I is inversely proportional to the cube of the number of samples, there is no general result for the behavior of variance under stratification. There are some functions for which stratification does no good. One example is a white noise function, where the variance is constant for all regions. On the other hand, most functions will benefit from stratified sampling, because the variance in each subcell will usually be smaller than the variance of the entire domain.

14.3.1 Quasi–Monte Carlo Integration

A popular method for quadrature is to replace the random points in Monte Carlo integration with *quasi-random* points. Such points are deterministic, but are in some sense uniform. For example, on the unit square $[0,1]^2$, a set of N quasi-random points should have the following property on a region of area A within the square:

$$\text{number of points in the region} \approx AN.$$

For example, a set of regular samples in a lattice has this property.

Quasi-random points can improve performance in many integration applications. Sometimes care must be taken to make sure that they do not introduce aliasing. It is especially nice that, in any application where calls are made to random or stratified points in $[0,1]^d$, one can substitute d-dimensional quasi-random points with no other changes.

The key intuition motivating quasi–Monte Carlo integration is that when estimating the average value of an integrand, any set of sample points will do, provided they are "fair."

14.4 Choosing Random Points

We often want to generate sets of random or pseudorandom points on the unit square for applications such as distribution ray tracing. There are several methods for doing this, e.g., jittering (see Section 13.4). These methods give us a set of N reasonably equidistributed points on the unit square $[0, 1]^2$: (u_1, v_1) through (u_N, v_N).

Sometimes, our sampling space may not be square (e.g., a circular lens), or may not be uniform (e.g., a filter function centered on a pixel). It would be nice if we could write a mathematical transformation that would take our equidistributed points (u_i, v_i) as input and output a set of points in our desired sampling space with our desired density. For example, to sample a camera lens, the transformation would take (u_i, v_i) and output (r_i, ϕ_i) such that the new points are approximately equidistributed on the disk of the lens. While we might be tempted to use the transform

$$\phi_i = 2\pi u_i,$$
$$r_i = v_i R,$$

it has a serious problem. While the points do cover the lens, they do so nonuniformly (Figure 14.6). What we need in this case is a transformation that takes equal-area regions to equal-area regions—one that takes uniform sampling distributions on the square to uniform distributions on the new domain.

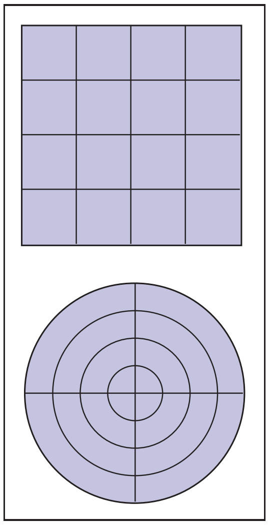


Figure 14.6. The transform that takes the horizontal and vertical dimensions uniformly to (r, ϕ) does not preserve relative area; not all of the resulting areas are the same.

There are several ways to generate such nonuniform points or uniform points on non-rectangular domains, and the following sections review the three most often used: function inversion, rejection, and Metropolis.

14.4.1 Function Inversion

If the density $f(x)$ is one-dimensional and defined over the interval $x \in [x_{\min}, x_{\max}]$, then we can generate random numbers α_i that have density f from a set of uniform random numbers ξ_i, where $\xi_i \in [0, 1]$. To do this, we need the cumulative probability distribution function $P(x)$:

$$\text{Probability}(\alpha < x) = P(x) = \int_{x_{\min}}^{x} f(x')d\mu.$$

To get α_i, we simply transform ξ_i:

$$\alpha_i = P^{-1}(\xi_i),$$

where P^{-1} is the inverse of P. If P is not analytically invertible, then numerical methods will suffice, because an inverse exists for all valid probability distribution functions.

Note that analytically inverting a function is more confusing than it should be due to notation. For example, if we have the function

$$y = x^2,$$

for $x > 0$, then the inverse function is expressed in terms of y as a function of x:

$$x = \sqrt{y}.$$

When the function is analytically invertible, it is almost always that simple. However, things are a little more opaque with the standard notation:

$$f(x) = x^2,$$
$$f^{-1}(x) = \sqrt{x}.$$

Here x is just a dummy variable. You may find it easier to use the less standard notation:

$$y = x^2,$$
$$x = \sqrt{y},$$

while keeping in mind that these are inverse functions of each other.

For example, to choose random points x_i that have density

$$p(x) = \frac{3x^2}{2}$$

on $[-1, 1]$, we see that

$$P(x) = \frac{x^3 + 1}{2},$$

and

$$P^{-1}(x) = \sqrt[3]{2x - 1},$$

so we can "warp" a set of canonical random numbers $(\xi_1, \cdots, \xi_N)$ to the properly distributed numbers

$$(x_1, \cdots, x_N) = (\sqrt[3]{2\xi_1 - 1}, \cdots, \sqrt[3]{2\xi_N - 1}).$$

Of course, this same warping function can be used to transform "uniform" jittered samples into nicely distributed samples with the desired density.

If we have a random variable $\alpha = (\alpha_x, \alpha_y)$ with two-dimensional density (x, y) defined on $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, then we need the two-dimensional distribution function:

$$\text{Probability}(\alpha_x < x \text{ and } \alpha_y < y) = F(x, y) = \int_{y_{\min}}^{y} \int_{x_{\min}}^{x} f(x', y') d\mu(x', y').$$

We first choose an x_i using the marginal distribution $F(x, y_{\max})$ and then choose y_i according to $F(x_i, y)/F(x_i, y_{\max})$. If $f(x, y)$ is separable (expressible as $g(x)h(y)$), then the one-dimensional techniques can be used on each dimension.

Returning to our earlier example, suppose we are sampling uniformly from the disk of radius R, so $p(r, \phi) = 1/(\pi R^2)$. The two-dimensional distribution function is

$$\text{Probability}(r < r_0 \ \text{ and } \ \phi < \phi_0) = F(r_0, \phi_0) = \int_0^{\phi_0} \int_0^{r_0} \frac{r dr d\phi}{\pi R^2} = \frac{\phi r^2}{2\pi R^2}.$$

This means that a canonical pair (ξ_1, ξ_2) can be transformed to a uniform random point on the disk:

$$\phi = 2\pi\xi_1,$$
$$r = R\sqrt{\xi_2}.$$

This mapping is shown in Figure 14.7.

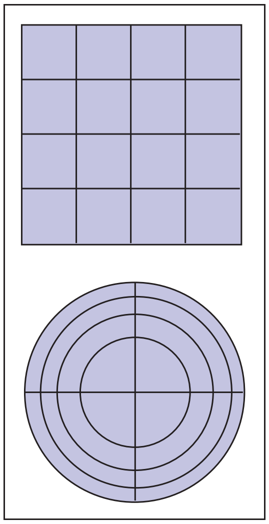


Figure 14.7. A mapping that takes equal area regions in the unit square to equal area regions in the disk.

To choose reflected ray directions for some realistic rendering applications, we choose points on the unit hemisphere according to the density:

$$p(\theta, \phi) = \frac{n+1}{2\pi} \cos^n \theta,$$

where n is a Phong-like exponent, θ is the angle from the surface normal and $\theta \in [0, \pi/2]$ (is on the upper hemisphere) and ϕ is the azimuthal angle ($\phi \in [0, 2\pi]$). The cumulative distribution function is

$$P(\theta, \phi) = \int_0^{\phi} \int_0^{\theta} p(\theta', \phi') \sin\theta' d\theta' d\phi'. \tag{14.8}$$

The $\sin\theta'$ term arises because, on the sphere, $d\omega = \cos\theta d\theta d\phi$. When the marginal densities are found, p (as expected) is separable, and we find that a (ξ_1, ξ_2) pair of canonical random numbers can be transformed to a direction by

$$\theta = \arccos\left((1 - \xi_1)^{\frac{1}{n+1}}\right),$$
$$\phi = 2\pi\xi_2.$$

Again, a nice thing about this is that a set of jittered points on the unit square can be easily transformed to a set of jittered points on the hemisphere with the desired distribution. Note that if n is set to 1, we have a diffuse distribution, as is often needed.

Often we must map the point on the sphere into an appropriate direction with respect to a *uvw* basis. To do this, we can first convert the angles to a unit vector $\vec{a}$:

$$\mathbf{a} = (\cos\phi \sin\theta,\ \sin\phi\sin\theta,\ \cos\theta)$$

As an efficiency improvement, we can avoid taking trigonometric functions of inverse trigonometric functions (e.g., $\cos(\arccos\theta)$). For example, when $n = 1$ (a diffuse distribution), the vector $\mathbf{a}$ simplifies to

$$\mathbf{a} = \left(\cos(2\pi\xi_1)\sqrt{\xi_2}, \sin(2\pi\xi_1)\sqrt{\xi_2}, \sqrt{1-\xi_2}\right)$$

14.4.2 Rejection

A *rejection* method chooses points according to some simple distribution and rejects some of them that are in a more complex distribution. There are several scenarios where rejection is used, and we show some of these by example.

Suppose we want uniform random points within the unit circle. We can first choose uniform random points $(x, y) \in [-1, 1]^2$ and reject those outside the circle. If the function $r()$ returns a canonical random number, then the procedure is:

```
done = false
while (not done) do
    $x = -1 + 2r()$
    $y = -1 + 2r()$
    if ($x^2 + y^2 < 1$) then
        done = true
```

If we want a random number $x \sim p$ and we know that $p : [a, b] \mapsto \mathbb{R}$, and that for all x, $p(x) < m$, then we can generate random points in the rectangle $[a, b] \times [0, m]$ and take those where $y < p(x)$:

```
done = false
while (not done) do
    $x = a + r()(b - a)$
    $y = r()m$
    if ($y < p(x)$) then
        done = true
```

This same idea can be applied to take random points on the surface of a sphere. To pick a random unit vector with uniform directional distribution, we first pick a random point in the unit sphere and then treat that point as a direction vector by taking the unit vector in the same direction:

```
done = false
**while** (not done) **do**
  $x = -1 + 2r()$
  $y = -1 + 2r()$
  $z = -1 + 2r()$
  **if** $((l = \sqrt{x^2 + y^2 + z^2}) < 1)$ **then**
    done = true
$x = x/l$
$y = y/l$
$z = z/l$
```

Although the rejection method is usually simple to code, it is rarely compatible with stratification. For this reason, it tends to converge more slowly and should thus be used mainly for debugging, or in particularly difficult circumstances.

14.4.3 Metropolis

The *Metropolis* method uses random *mutations* to produce a set of samples with a desired density. This concept is used extensively in the *Metropolis Light Transport* algorithm referenced in the chapter notes. Suppose we have a random point x_0 in a domain S. Further, suppose for any point x, we have a way to generate random $y \sim p_x$. We use the marginal notation $p_x(y) \equiv p(x \to y)$ to denote this density function. Now, suppose we let x_1 be a random point in S selected with underlying density $p(x_0 \to x_1)$. We generate x_2 with density $p(x_1 \to x_0)$ and so on. In the limit, where we generate an infinite number of samples, it can be proved that the samples will have some underlying density determined by p regardless of the initial point x_0.

Now, suppose we want to choose p such that the underlying density of samples to which we converge is proportional to a function $f(x)$ where f is a nonnegative function with domain S. Further, suppose we can evaluate f, but we have little or no additional knowledge about its properties (such functions are common in graphics). Also, suppose we have the ability to make "transitions" from x_i to x_{i+1} with underlying density function $t(x_i \to x_{i+1})$. To add flexibility, further suppose we add the potentially nonzero probability that x_i transitions to itself, i.e., $x_{i+1} = x_i$. We phrase this as generating a potential candidate $y \sim t(x_i \to y)$

and "accepting" this candidate (i.e., $x_{i+1} = y$) with probability $a(x_i \to y)$ and rejecting it (i.e., $x_{i+1} = x_i$) with probability $1 - a(x_i \to y)$. Note that the sequence $x_0, x_1, x_2, \ldots$ will be a random set, but there will be some correlation among samples. They will still be suitable for Monte Carlo integration or density estimation, but analyzing the variance of those estimates is much more challenging.

Now, suppose we are given a transition function $t(x \to y)$ and a function $f(x)$ of which we want to mimic the distribution, can we use $a(y \to x)$ such that the points are distributed in the shape of f? Or more precisely,

$$\{x_0, x_1, x_2, \ldots\} \sim \frac{f}{\int_s f}.$$

It turns out this can be forced by making sure the x_i are *stationary* in some strong sense. If you visualize a huge collection of sample points x, you want the "flow" between two points to be the same in each direction. If we assume the density of points near x and y are proportional to $f(x)$ and $f(y)$, respectively, then the flow in the two directions should be the same:

$$\begin{aligned}\text{flow}(x \to y) &= kf(x)t(x \to y)a(x \to y),\\ \text{flow}(y \to x) &= kf(y)t(y \to x)a(y \to x),\end{aligned}$$

where k is some positive constant. Setting these two flows constant gives a constraint on a:

$$\frac{a(y \to x)}{a(x \to y)} = \frac{f(x)t(x \to y)}{f(y)t(y \to x)}.$$

Thus, if either $a(y \to x)$ or $a(x \to y)$ is known, so is the other. Making them larger improves the chance of acceptance, so the usual technique is to set the larger of the two to 1.

A difficulty in using the Metropolis sample generation technique is that it is hard to estimate how many points are needed before the set of points is "good." Things are accelerated if the first n points are discarded, although choosing n wisely is nontrivial.

14.4.4 Example: Choosing Random Lines in the Square

As an example of the full process of designing a sampling strategy, consider the problem of finding random lines that intersect the unit square $[0, 1]^2$. We want this process to be fair; that is, we would like the lines to be uniformly distributed within the square. Intuitively, we can see that there is some subtlety to this problem; there are "more" lines at an oblique angle than in horizontal or vertical directions. This is because the cross section of the square is not uniform.

Our first goal is to find a function-inversion method, if one exists, and then to fall back on rejection or Metropolis if that fails. This is because we would like to have stratified samples in line space. We try using normal coordinates first, because the problem of choosing random lines in the square is just the problem of finding uniform random points in whatever part of (r, θ) space corresponds to lines in the square.

Consider the region where $-\pi/2 < \theta < 0$. What values of r correspond to lines that hit the square? For those angles, $r < \cos\theta$ are all the lines that hit the square as shown in Figure 14.8. Similar reasoning in the other four quadrants finds the region in (r, θ) space that must be sampled, as shown in Figure 14.9. The equation of the boundary of that region $r_{\max}(\theta)$is

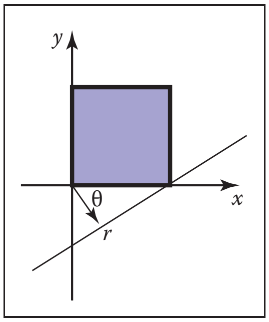


Figure 14.8. The largest distance r corresponds to a line hitting the square for $\theta \in [-\pi/2, 0]$. Because the square has sidelength one, r $= \cos\theta$.

$$r_{\max}(\theta) = \begin{cases} 0 & \text{if } \theta \in [-\pi, -\frac{\pi}{2}], \\ \cos\theta & \text{if } \theta \in [-\frac{\pi}{2}, 0], \\ \sqrt{2}\cos(\theta - \frac{\pi}{4}) & \text{if } \theta \in [0, \frac{\pi}{2}], \\ \sin\theta & \text{if } \theta \in [\frac{\pi}{2}, \pi]. \end{cases}$$

Because the region under $r_{\max}(\theta)$ is a simple function bounded below by $r = 0$, we can sample it by first choosing θ according to the density function:

$$p(\theta) = \frac{r_{\max}(\theta)}{\int_{-\pi}^{\pi} r_{\max}(\theta) d\theta}.$$

The denominator here is 4. Now, we can compute the cumulative probability distribution function:

$$P(\theta) = \begin{cases} 0 & \text{if } \theta \in [-\pi, -\frac{\pi}{2}], \\ (1 + \sin\theta)/4 & \text{if } \theta \in [-\frac{\pi}{2}, 0], \\ (1 + \frac{\sqrt{2}}{2}\sin(\theta - \frac{\pi}{4}))/2 & \text{if } \theta \in [0, \frac{\pi}{2}], \\ (3 - \cos\theta)/4 & \text{if } \theta \in [\frac{\pi}{2}, \pi]. \end{cases}$$

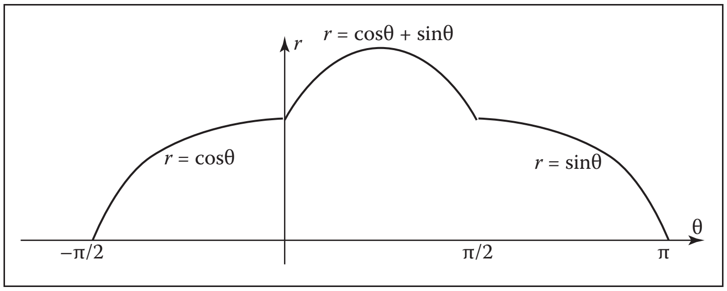


Figure 14.9. The maximum radius for lines hitting the unit square $[0,1]^2$ as a function of θ.

We can invert this by manipulating $\xi_1 = P(\theta)$ into the form $\theta = g(\xi_1)$. This yields

$$\theta = \begin{cases} \arcsin(4\xi_1 - 1) & \text{if } \xi_1 < \frac{1}{4}, \\ \arcsin(\frac{\sqrt{2}}{2}(2\xi_1 - 1)) + \frac{\pi}{4} & \text{if } \xi_1 \in [\frac{1}{4}, \frac{3}{4}], \\ \arccos(3 - 4\xi_1) & \text{if } \xi_1 > \frac{3}{4}. \end{cases}$$

Once we have θ, then r is simply:

$$r = \xi_2 r_{\text{max}}(\theta).$$

As discussed earlier, there are many parameterizations of the line, and each has an associated "fair" measure. We can generate random lines in any of these spaces as well. For example, in slope-intercept space, the region that hits the square is shown in Figure 14.10. By similar reasoning to the normal space, the density function for the slope is

$$p(m) = \frac{1 + |m|}{4}$$

with respect to the differential measure

$$d\mu = \frac{dm}{(1 + m^2)^{\frac{3}{2}}}.$$

This gives rise to the cumulative distribution function:

$$P(m) = \begin{cases} \frac{1}{4} + \frac{m+1}{4\sqrt{1+m^2}} & \text{if } m < 0, \\ \frac{3}{4} + \frac{m-1}{4\sqrt{1+m^2}} & \text{if } m \geq 0. \end{cases}$$

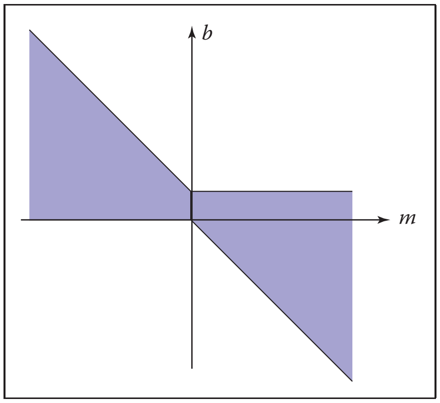


Figure 14.10. The region of (m,b) space that contains lines that intersect the unit square $[0,1]^2$.

These can be inverted by solving two quadratic equations. Given an m generated using ξ_1, we then have

$$b = \begin{cases} (1-m)\xi_2 & \text{if } \xi < \frac{1}{2}. \\ -m + (1+m)\xi_2 & \text{otherwise.} \end{cases}$$

This is not a better way than using normal coordinates; it is just an alternative way.

Frequently Asked Questions

- **This chapter discussed probability but not statistics. What is the distinction?**

Probability is the study of how likely an event is. Statistics infers characteristics of large, but finite, populations of random variables. In that sense, statistics could be viewed as a specific type of applied probability.

- **Is Metropolis sampling the same as the Metropolis Light Transport Algorithm?**

No. The *Metropolis Light Transport* (Veach & Guibas, 1997) algorithm uses Metropolis sampling as part of its procedure, but it is specifically for rendering, and it has other steps as well.

Notes

The classic reference for geometric probability is *Geometric Probability* (Solomon, 1978). Another method for picking random edges in a square is given in *Random–Edge Discrepancy of Supersampling Patterns* (Dobkin & Mitchell, 1993). More information on quasi-Monte Carlo methods for graphics can be found in *Efficient Multidimensional Sampling* (Kollig & Keller, 2002). Three classic and very readable books on Monte Carlo methods are *Monte Carlo Methods* (Hammersley & Handscomb, 1964), *Monte Carlo Methods, Basics* (Kalos & Whitlock, 1986), and *The Monte Carlo Method* (Sobel, Stone, & Messer, 1975).

Exercises

1. What is the average value of the function xyz in the unit cube $(x, y, z) \in [0, 1]^3$?

2. What is the average value of r on the unit-radius disk: $(r, \phi) \in [0, 1] \times [0, 2\pi)$?

3. Show that the uniform mapping of canonical random points (ξ_1, ξ_2) to the barycentric coordinates of any triangle is: $\beta = 1 - \sqrt{1 - \xi_1}$, and $\gamma = (1 - u)\xi_2$.

4. What is the average length of a line inside the unit square? Verify your answer by generating ten million random lines in the unit square and averaging their lengths.

5. What is the average length of a line inside the unit cube? Verify your answer by generating ten million random lines in the unit cube and averaging their lengths.

6. Show from the definition of variance that $V(x) = E(x^2) - [E(x)]^2$.

Michael Gleicher

15 Curves

15.1 Curves

Intuitively, think of a *curve* as something you can draw with a pen. The curve is the set of points that the pen traces over an interval of time. While we usually think of a pen writing on paper (e.g., a curve that is in a 2D space), the pen could move in 3D to generate a *space curve*, or you could imagine the pen moving in some other kind of space.

Mathematically, definitions of curve can be seen in at least two ways:

1. the continuous image of some interval in an n-dimensional space;

2. a continuous map from a one-dimensional space to an n-dimensional space.

Both of these definitions start with the idea of an interval range (the time over which the pen traces the curve). However, there is a significant difference: in the first definition, the curve is the set of points the pen traces (the image), while in the second definition, the curve is the mapping between time and that set of points. For this chapter, we use the first definition.

A curve is an infinitely large set of points. The points in a curve have the property that any point has two neighbors, except for a small number of points that have one neighbor (these are the endpoints). Some curves have no endpoints, either because they are infinite (like a line) or they are *closed* (loop around and connect to themselves).

Because the "pen" of the curve is thin (infinitesimally), it is difficult to create filled regions. While space-filling curves are possible (by having them fold over themselves infinitely many times), we do not consider such mathematical oddities here. Generally, we think of curves as the outlines of things, not the "insides."

The problem that we need to address is how to specify a curve—to give a name or representation to a curve so that we can represent it on a computer. For some curves, the problem of naming them is easy since they have known shapes: line segments, circles, elliptical arcs, etc. A general curve that does not have a "named" shape is sometimes called a *free-form* curve. Because a free-form curve can take on just about any shape, they are much harder to specify.

There are three main ways to specify curves mathematically:

1. Implicit curve representations define the set of points on a curve by giving a procedure that can test to see if a point in on the curve. Usually, an implicit curve representation is defined by an *implicit function* of the form

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

   so that the curve is the set of points for which this equation is true. Note that the implicit function f is a scalar function (it returns a single real number).

2. Parametric curve representations provide a mapping from a *free parameter* to the set of points on the curve. That is, this free parameter provides an index to the points on the curve. The parametric form of a curve is a function that assigns positions to values of the free parameter. Intuitively, if you think of a curve as something you can draw with a pen on a piece of paper, the free parameter is time, ranging over the interval from the time that we began drawing the curve to the time that we finish. The *parametric function* of this curve tells us where the pen is at any instant in time:

$$(x, y) = \mathbf{f}(t).$$

   Note that the parametric function is a vector-valued function. This example is a 2D curve, so the output of the function is a 2-vector; in 3D it would be a 3-vector.

3. Generative or procedural curve representations provide procedures that can generate the points on the curve that do not fall into the first two categories. Examples of generative curve descriptions include subdivision schemes and fractals.

Remember that a curve is a set of points. These representations give us ways to specify those sets. Any curve has many possible representations. For this reason, mathematicians typically are careful to distinguish between a curve and its representations. In computer graphics we are often sloppy, since we usually only refer to the representation, not the actual curve itself. So when someone says “an implicit curve,” they are either referring to the curve that is represented by some implicit function or to the implicit function that is one of the representations of some curve. Such distinctions are not usually important, unless we need to consider different representations of the same curve. We will consider different curve representations in this chapter, so we will be more careful. When we use a term like “polynomial curve,” we will mean the curve that can be represented by the polynomial.

By the definition given at the beginning of the chapter, for something to be a curve it must have a parametric representation. However, many curves have other representations. For example, a circle in 2D with its center at the origin and radius equal to 1 can be written in implicit form as

$$f(x, y) = x^2 + y^2 - 1 = 0,$$

or in parametric form as

$$(x, y) = \mathbf{f}(t) = (\cos t, \sin t), \quad t \in [0, 2\pi).$$

The parametric form need not be the most convenient representation for a given curve. In fact, it is possible to have curves with simple implicit or generative representations for which it is difficult to find a parametric representation.

Different representations of curves have advantages and disadvantages. For example, parametric curves are much easier to draw, because we can sample the free parameter. Generally, parametric forms are the most commonly used in computer graphics since they are easier to work with. Our focus will be on parametric representations of curves.

15.1.1 Parameterizations and Reparameterizations

A *parametric curve* refers to the curve that is given by a specific parametric function over some particular interval. To be more precise, a parametric curve has a given function that is a mapping from an interval of the parameters. It is often convenient to have the parameter run over the unit interval from 0 to 1. When the free parameter varies over the unit interval, we often denote the parameter as u.

If we view the parametric curve to be a line drawn with a pen, we can consider $u = 0$ as the time when the pen is first set down on the paper and the unit of time to be the amount of time it takes to draw the curve ($u = 1$ is the end of the curve).

The curve can be specified by a function that maps time (in these unit coordinates) to positions. Basically, the specification of the curve is a function that can answer the question, "Where is the pen at time u?"

If we are given a function $\mathbf{f}(t)$ that specifies a curve over interval $[a, b]$, we can easily define a new function $\mathbf{f}_2(u)$ that specifies the same curve over the unit interval. We can first define

$$g(u) = a + (b - a)u,$$

and then

$$\mathbf{f}_2(u) = \mathbf{f}(g(u)).$$

The two functions, $\mathbf{f}$ and $\mathbf{f}_2$ both represent the same curve; however, they provide different *parameterizations* of the curve. The process of creating a new parameterization for an existing curve is called *reparameterization*, and the mapping from old parameters to the new ones (g, in this example) is called the *reparameterization function.*

If we have defined a curve by some parameterization, infinitely many others exist (because we can always reparameterize). Being able to have multiple parameterizations of a curve is useful, because it allows us to create parameterizations that are convenient. However, it can also be problematic, because it makes it difficult to compare two functions to see if they represent the same curve.

The essence of this problem is more general: the existence of the free parameter (or the element of time) adds an invisible, potentially unknown element to our representation of the curves. When we look at the curve after it is drawn, we don't necessarily know the timing. The pen might have moved at a constant speed over the entire time interval, or it might have started slowly and sped up. For example, while $u = 0.5$ is halfway through the parameter space, it may not be halfway along the curve if the motion of the pen starts slowly and speeds up at the end. Consider the following representations of a very simple curve:

$$\begin{aligned} (x, y) &= \mathbf{f}(u) = & (u, u), \\ (x, y) &= \mathbf{f}(u) = & (u^2, u^2), \\ (x, y) &= \mathbf{f}(u) = & (u^5, u^5). \end{aligned}$$

All three functions represent the same curve on the unit interval; however when u is not 0 or 1, $\mathbf{f}(u)$ refers to a different point depending on the representation of the curve.

If we are given a parameterization of a curve, we can use it directly as our specification of the curve, or we can develop a more convenient parameterization. Usually, the *natural parameterization* is created in a way that is convenient (or natural) for specifying the curve, so we don't have to know about how the speed changes along the curve.

If we know that the pen moves at a constant velocity, then the values of the free parameters have more meaning. Halfway through parameter space is halfway along the curve. Rather than measuring time, the parameter can be thought to measure length along the curve. Such parameterizations are called *arc-length* parameterizations because they define curves by functions that map from the distance along the curve (known as the arc length) to positions. We often use the variable s to denote an arc-length parameter.

Technically, a parameterization is an arc-length parameterization if the magnitude of its *tangent* (that is, the derivative of the parameterization with respect to the parameter) has constant magnitude. Expressed as an equation,

$$\left|\frac{d\mathbf{f}(s)}{ds}\right|^2 = c.$$

Computing the length along a curve can be tricky. In general, it is defined by the integral of the magnitude of the derivative (intuitively, the magnitude of the derivative is the velocity of the pen as it moves along the curve). So, given a value for the parameter v, you can compute s (the arc-length distance along the curve from the point $\mathbf{f}(0)$ to the point $\mathbf{f}(v)$) as

$$s = \int_0^v \left|\frac{d\mathbf{f}(t)}{dt}\right|^2 dt, \tag{15.1}$$

where $\mathbf{f}(t)$ is a function that defines the curve with a natural parameterization.

Using the arc-length parameterization requires being able to solve Equation (15.1) for t, given s. For many of the kinds of curves we examine, it cannot be done in a closed-form (simple) manner and must be done numerically.

Generally, we use the variable u to denote free parameters that range over the unit interval, s to denote arc-length free parameters, and t to represent parameters that aren't one of the other two.

15.1.2 Piecewise Parametric Representations

For some curves, defining a parametric function that represents their shape is easy. For example, lines, circles, and ellipses all have simple functions that define the points they contain in terms of a parameter. For many curves, finding a function that specifies their shape can be hard. The main strategy that we use to create complex curves is divide-and-conquer: we break the curve into a number of simpler smaller pieces, each of which has a simple description.

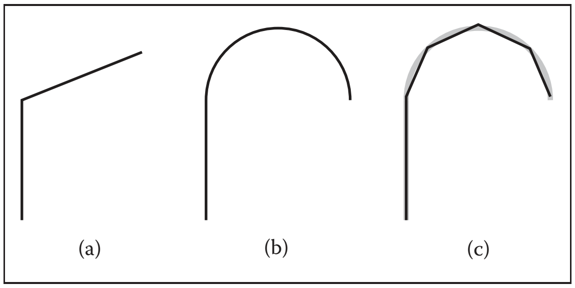


Figure 15.1. (a) A curve that can be easily represented as two lines; (b) a curve that can be easily represented as a line and a circular arc; (c) a curve approximating curve (b) with five line segments.

For example, consider the curves in Figure 15.1. The first two curves are easily specified in terms of two pieces. In the case of the curve in Figure 15.1(b), we need two different kinds of pieces: a line segment and a circle.

To create a parametric representation of a compound curve (like the curve in Figure 15.1(b)), we need to have our parametric function switch between the functions that represent the pieces. If we define our parametric functions over the range $0 \leq u \leq 1$, then the curve in Figures 15.1(a) or (b) might be defined as

$$\mathbf{f}(u) = \begin{cases} \mathbf{f}_1(2u) & \text{if } u \leq 0.5, \\ \mathbf{f}_2(2u-1) & \text{if } u > 0.5, \end{cases} \tag{15.2}$$

where $\mathbf{f}_1$ is a parameterization of the first piece, $\mathbf{f}_2$ is a parameterization of the second piece, and both of these functions are defined over the unit interval.

We need to be careful in defining the functions $\mathbf{f}_1$ and $\mathbf{f}_2$ to make sure that the pieces of the curve fit together. If $\mathbf{f}_1(1) \neq \mathbf{f}_2(0)$, then our curve pieces will not connect and will not form a single continuous curve.

To represent the curve in Figure 15.1(b), we needed to use two different types of pieces: a line segment and a circular arc. For simplicity's sake, we may prefer to use a single type of piece. If we try to represent the curve in Figure 15.1(b) with only one type of piece (line segments), we cannot exactly re-create the curve (unless we use an infinite number of pieces). While the new curve made of line segments (as in Figure 15.1(c)) may not be exactly the same shape as in Figure 15.1(b), it might be close enough for our use. In such a case, we might prefer the simplicity of using the simpler line segment pieces to having a curve that more accurately represents the shape.

Also, notice that as we use an increasing number of pieces, we can get a better approximation. In the limit (using an infinite number of pieces), we can exactly represent the original shape.

One advantage to using a piecewise representation is that it allows us to make a tradeoff between

1. how well our represented curve approximates the real shape we are trying to represent;
2. how complicated the pieces that we use are;
3. how many pieces we use.

So, if we are trying to represent a complicated shape, we might decide that a crude approximation is acceptable and use a small number of simple pieces. To improve the approximation, we can choose between using more pieces and using more complicated pieces.

In computer graphics practice, we tend to prefer using relatively simple curve pieces (either line segments, arcs, or polynomial segments).

15.1.3 Splines

Before computers, when draftsmen wanted to draw a smooth curve, one tool they employed was a stiff piece of metal that they would bend into the desired shape for tracing. Because the metal would bend, not fold, it would have a smooth shape. The stiffness meant that the metal would bend as little as possible to make the desired shape. This stiff piece of metal was called a *spline*.

Mathematicians found that they could represent the curves created by a draftman's spline with piecewise polynomial functions. Initially, they used the term spline to mean a smooth, piecewise polynomial function. More recently, the term spline has been used to describe any piecewise polynomial function. We prefer this latter definition.

For us, a *spline* is a piecewise polynomial function. Such functions are very useful for representing curves.

15.2 Curve Properties

To describe a curve, we need to give some facts about its properties. For "named" curves, the properties are usually specific according to the type of curve. For example, to describe a circle, we might provide its radius and the position of its center. For an ellipse, we might also provide the orientation of its major axis and the ratio of the lengths of the axes. For free-form curves however, we need to have a more general set of properties to describe individual curves.

Some properties of curves are attributed to only a single location on the curve, while other properties require knowledge of the whole curve. For an intuition of the difference, imagine that the curve is a train track. If you are standing on the track on a foggy day, you can tell that the track is straight or curved and whether or not you are at an endpoint. These are *local* properties. You cannot tell whether or not the track is a closed curve, or crosses itself, or how long it is. We call this type of property, a *global* property.

The study of local properties of geometric objects (curves and surfaces) is known as *differential geometry.* Technically, to be a differential property, there are some mathematical restrictions about the properties (roughly speaking, in the train-track analogy, you would not be able to have a GPS or a compass). Rather than worry about this distinction, we will use the term *local* property rather than differential property.

Local properties are important tools for describing curves because they do not require knowledge about the whole curve. Local properties include

- continuity,
- position at a specific place on the curve,
- direction at a specific place on the curve,
- curvature (and other derivatives).

Often, we want to specify that a curve includes a particular point. A curve is said to *interpolate* a point if that point is part of the curve. A function f interpolates a value v if there is some value of the parameter u for which $f(t) = v$. We call the place of interpolation, that is the value of t, the $site$.

15.2.1 Continuity

It will be very important to understand the local properties of a curve where two parametric pieces come together. If a curve is defined using an equation like Equation (15.2), then we need to be careful about how the pieces are defined. If $\mathbf{f}_1(1) \neq \mathbf{f}_2(0)$, then the curve will be “broken”—we would not be able to draw the curve in a continuous stroke of a pen. We call the condition that the curve pieces fit together *continuity* conditions because if they hold, the curve can be drawn as a continuous piece. Because our definition of ”curve” at the beginning of the chapter requires a curve to be continuous, technically a ”broken curve” is not a curve.

In addition to the positions, we can also check that the derivatives of the pieces match correctly. If $\mathbf{f}_1'(1) \neq \mathbf{f}_2'(0)$, then the combined curve will have an abrupt change in its first derivative at the switching point; the first derivative will not be continuous. In general, we say that a curve is C^n continuous if all of its derivatives up to n match across pieces. We denote the position itself as the zeroth derivative, so that the C^0 continuity condition means that the positions of the curve are continuous, and C^1 continuity means that positions and first derivatives are continuous. The definition of curve requires the curve to be C^0.

An illustration of some continuity conditions is shown in Figure 15.2. A discontinuity in the first derivative (the curve is C^0 but not C^1) is usually noticeable because it displays a sharp corner. A discontinuity in the second derivative is sometimes visually noticeable. Discontinuities in higher derivatives might matter, depending on the application. For example, if the curve represents a motion, an abrupt change in the second derivative is noticeable, so third derivative continuity is often useful. If the curve is going to have a fluid flowing over it (for example, if it is the shape for an airplane wing or boat hull), a discontinuity in the fourth or fifth derivative might cause turbulence.

The type of continuity we have just introduced (C^n) is commonly referred to as *parametric continuity* as it depends on the parameterization of the two curve pieces. If the "speed" of each piece is different, then they will not be continuous. For cases where we care about the shape of the curve, and not its parameterization, we define *geometric continuity* that requires that the derivatives of the curve pieces match when the curves are parameterized equivalently (for example, using an arc-length parameterization). Intuitively, this means that the corresponding derivatives must have the same direction, even if they have different magnitudes.

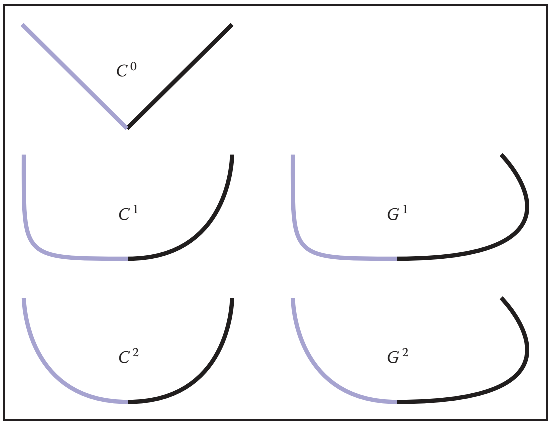


Figure 15.2. An illustration of various types of continuity between two curve segments.

So, if the C^1 continuity condition is

$$\mathbf{f}_1'(1) = \mathbf{f}_2'(0),$$

the G^1 continuity condition would be

$$\mathbf{f}_1'(1) = k\,\mathbf{f}_2'(0),$$

for some value of scalar k. Generally, geometric continuity is less restrictive than parametric continuity. A C^n curve is also G^n except when the parametric derivatives vanish.

15.3 Polynomial Pieces

The most widely used representations of curves in computer graphics is done by piecing together basic elements that are defined by polynomials and called polynomial pieces. For example, a line element is given by a linear polynomial. In Section 15.3.1, we give a formal definition and explain how to put pieces of polynomial together.

15.3.1 Polynomial Notation

Polynomials are functions of the form

$$f(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n. \tag{15.3}$$

The a_i are called the *coefficients*, and n is called the degree of the polynomial if $a_n \neq 0$. We also write Equation (15.3) in the form

$$\mathbf{f}(t) = \sum_{i=0}^{n} \mathbf{a}_i t^i. \tag{15.4}$$

We call this the *canonical* form of the polynomial.

We can generalize the canonical form to

$$\mathbf{f}(t) = \sum_{i=0}^{n} \mathbf{c}_i b_i(t), \tag{15.5}$$

where $b_i(t)$ is a polynomial. We can choose these polynomials in a convenient form for different applications, and we call them *basis functions* or *blending functions* (see Section 15.3.5). In Equation (15.4), the t^i are the $b_i(t)$ of Equation (15.5). If the set of basis functions is chosen correctly, any polynomial of degree $n+1$ can be represented by an appropriate choice of $\mathbf{c}$.

The canonical form does not always have convenient coefficients. For practical purposes, throughout this chapter, we will find sets of basis functions such that the coefficients are convenient ways to control the curves represented by the polynomial functions.

To specify a curve embedded in two dimensions, one can either specify two polynomials in t: one for how x varies with t and one for how y varies with t; or specify a single polynomial where each of the $\mathbf{a}_i$ is a 2D point. An analogous situation exists for any curve in an n-dimensional space.

15.3.2 A Line Segment

To introduce the concepts of piecewise polynomial curve representations, we will discuss line segments. In practice, line segments are so simple that the mathematical derivations will seem excessive. However, by understanding this simple case, things will be easier when we move on to more complicated polynomials.

Consider a line segment that connects point $\mathbf{p}_0$ to $\mathbf{p}_1$. We could write the parametric function over the unit domain for this line segment as

$$\mathbf{f}(u) = (1-u)\mathbf{p}_0 + u\mathbf{p}_1. \tag{15.6}$$

By writing this in vector form, we have hidden the dimensionality of the points and the fact that we are dealing with each dimension separately. For example, were we working in 2D, we could have created separate equations:

$$\begin{aligned} f_x(u) &= (1-u)x_0 + ux_1, \\ f_y(u) &= (1-u)y_0 + uy_1. \end{aligned}$$

The line that we specify is determined by the two endpoints, but from now on we will stick to vector notation since it is cleaner. We will call the vector of control parameters, $\mathbf{p}$, the *control points*, and each element of $\mathbf{p}$, a *control point*.

While describing a line segment by the positions of its endpoints is obvious and usually convenient, there are other ways to describe a line segment. For example,

1. the position of the center of the line segment, the orientation, and the length;
2. the position of one endpoint and the position of the second point relative to the first;
3. the position of the middle of the line segment and one endpoint.

It is obvious that given one kind of a description of a line segment, we can switch to another one.

A different way to describe a line segment is using the canonical form of the polynomial (as discussed in Section 15.3.1),

$$\mathbf{f}(u) = \mathbf{a}_0 + u\mathbf{a}_1. \tag{15.7}$$

Any line segment can be represented either by specifying $\mathbf{a}_0$ and $\mathbf{a}_1$ or the endpoints ($\mathbf{p}_0$ and $\mathbf{p}_1$). It is usually more convenient to specify the endpoints, because we can compute the other parameters from the endpoints.

To write the canonical form as a vector expression, we define a vector $\mathbf{u}$ that is a vector of the powers of u:

$$\mathbf{u} = \left[1 \; u \; u^2 \; u^3 \; \ldots \; u^n\right],$$

so that Equation (15.4) can be written as

$$\mathbf{f}(u) = \mathbf{u} \cdot \mathbf{a}. \tag{15.8}$$

This vector notation will make transforming between different forms of the curve easier.

Equation (15.8) describes a curve segment by the set of polynomial coefficients for the simple form of the polynomial. We call such a representation the *canonical* form. We will denote the parameters of the canonical form by $\mathbf{a}$.

While it is mathematically simple, the canonical form is not always the most convenient way to specify curves. For example, we might prefer to specify a line segment by the positions of its endpoints. If we want to define $\mathbf{p}_0$ to be the beginning of the segment (where the segment is when $u = 0$) and $\mathbf{p}_1$ to be the end of the line segment (where the line segment is at $u = 1$), we can write

$$\begin{aligned} \mathbf{p}_0 &= \mathbf{f}(0) &= [1\;0] \cdot [\mathbf{a}_0\;\mathbf{a}_1], \\ \mathbf{p}_1 &= \mathbf{f}(1) &= [1\;1] \cdot [\mathbf{a}_0\;\mathbf{a}_1]. \end{aligned} \tag{15.9}$$

We can solve these equations for $\mathbf{a}_0$ and $\mathbf{a}_1$:

$$\begin{aligned} \mathbf{a}_0 &= \mathbf{p}_0, \\ \mathbf{a}_1 &= \mathbf{p}_1 - \mathbf{p}_0. \end{aligned}$$

Matrix Form for Polynomials

While this first example was easy enough to solve, for more complicated examples it will be easier to write Equation (15.9) in the form

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \end{bmatrix}.$$

Alternatively, we can write

$$\mathbf{p} = \mathbf{C}\,\mathbf{a}, \tag{15.10}$$

where we call $\mathbf{C}$, the *constraint matrix*.[1] If having vectors of points bothers you, you can consider each dimension independently (so that $\mathbf{p}$ is $[x_0\ x_1]$ or $[y_0\ y_1]$ and $\mathbf{a}$ is handled correspondingly).

We can solve Equation (15.10) for $\mathbf{a}$ by finding the inverse of $\mathbf{C}$. This inverse matrix which we will denote by $\mathbf{B}$ is called the *basis* matrix. The basis matrix is very handy since it tells us how to convert between the convenient parameters $\mathbf{p}$ and the canonical form $\mathbf{a}$, and, therefore, gives us an easy way to evaluate the curve

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

We can find a basis matrix for whatever form of the curve that we want, providing that there are no nonlinearities in the definition of the parameters. Examples of nonlinearly defined parameters include the length and angle of the line segment.

Now, suppose we want to parameterize the line segment so that $\mathbf{p}_0$ is the half-way point ($u = 0.5$), and $\mathbf{p}_1$ is the ending point ($u = 1$). To derive the basis matrix for this parameterization, we set

$$\begin{aligned} \mathbf{p}_0 = &\quad \mathbf{f}(0.5) = &\quad 1\,\mathbf{a}_0 + 0.5\,\mathbf{a}_1, \\ \mathbf{p}_1 = &\quad \mathbf{f}(1) = &\quad 1\,\mathbf{a}_0 + 1\,\mathbf{a}_1. \end{aligned}$$

So

$$\mathbf{C} = \begin{bmatrix} 1 & .5 \\ 1 & 1 \end{bmatrix},$$

and therefore

$$\mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 2 & -1 \\ -2 & 2 \end{bmatrix}.$$

15.3.3 Beyond Line Segments

Line segments are so simple that finding a basis matrix is trivial. However, it was good practice for curves of higher degree. First, let's consider quadratics (curves of degree two). The advantage of the canonical form (Equation (15.4)) is that it works for these more complicated curves, just by letting n be a larger number.

[1] We assume the form of a vector (row or column) is obvious from the context, and we will skip all of the transpose symbols for vectors.

A quadratic (a degree-two polynomial) has three coefficients, $\mathbf{a}_0$, $\mathbf{a}_1$, and $\mathbf{a}_2$. These coefficients are not convenient for describing the shape of the curve. However, we can use the same basis matrix method to devise more convenient parameters. If we know the value of u, Equation (15.4) becomes a linear equation in the parameters, and the linear algebra from the last section still works.

Suppose that we wanted to describe our curves by the position of the beginning ($u = 0$), middle[2] ($u = 0.5$), and end ($u = 1$). Entering the appropriate values into Equation (15.4):

$$\begin{array}{llllll} \mathbf{p}_0 & = \mathbf{f}(0) & = \mathbf{a}_0 + 0^1 & \mathbf{a}_1 & + 0^2 & \mathbf{a}_2, \\ \mathbf{p}_1 & = \mathbf{f}(0.5) & = \mathbf{a}_0 + 0.5^1 & \mathbf{a}_1 & + 0.5^2 & \mathbf{a}_2, \\ \mathbf{p}_2 & = \mathbf{f}(1) & = \mathbf{a}_0 + 1^1 & \mathbf{a}_1 & + 1^2 & \mathbf{a}_2. \end{array}$$

So the constraint matrix is

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & .5 & .25 \\ 1 & 1 & 1 \end{bmatrix},$$

and the basis matrix is

$$\mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 4 & -1 \\ 2 & -4 & 2 \end{bmatrix}.$$

There is an additional type of constraint (or parameter) that is sometimes convenient to specify: the derivative of the curve (with respect to its free parameter) at a particular value. Intuitively, the derivatives tell us how the curve is changing, so that the first derivative tells us what direction the curve is going, the second derivative tells us how quickly the curve is changing direction, etc. We will see examples of why it is useful to specify derivatives later.

For the quadratic,

$$\mathbf{f}(u) = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2,$$

the derivatives are simple:

$$\mathbf{f}'(u) = \frac{d\mathbf{f}}{du} = \mathbf{a}_1 + 2\mathbf{a}_2 u,$$

and

$$\mathbf{f}''(u) = \frac{d^2\mathbf{f}}{du^2} = \frac{d\mathbf{f}'}{du} = 2\mathbf{a}_2.$$

[2]Notice that this is the middle of the parameter space, which might not be the middle of the curve itself.

Or, more generally,

$$\begin{aligned} \mathbf{f}'(u) &= \textstyle\sum_{i=1}^{n} iu^{i-1}\mathbf{a}_i, \\ \mathbf{f}''(u) &= \textstyle\sum_{i=2}^{n} i(i-1)u^{i-2}\mathbf{a}_i. \end{aligned}$$

For example, consider a case where we want to specify a quadratic curve segment by the position, first, and second derivative at its middle ($u = 0.5$).

$$\begin{array}{llllllll} \mathbf{p}_0 & = \mathbf{f}(0.5) & = & \mathbf{a}_0 + & 0.5^1 & \mathbf{a}_1 + & & 0.5^2 & \mathbf{a}_2, \\ \mathbf{p}_1 & = \mathbf{f}'(0.5) & = & & & \mathbf{a}_1 + & 2 & 0.5 & \mathbf{a}_2, \\ \mathbf{p}_2 & = \mathbf{f}''(0.5) & = & & & & 2 & & \mathbf{a}_2. \end{array}$$

The constraint matrix is

$$\mathbf{C} = \begin{bmatrix} 1 & .5 & .25 \\ 0 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix},$$

and the basis matrix is

$$\mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 1 & -.5 & .125 \\ 0 & 1 & -.5 \\ 0 & 0 & .5 \end{bmatrix}.$$

15.3.4 Basis Matrices for Cubics

Cubic polynomials are popular in graphics (See Section 15.5). The derivations for the various forms of cubics are just like the derivations we've seen already in this section. We will work through one more example for practice.

A very useful form of a cubic polynomial is the *Hermite* form, where we specify the position and first derivative at the beginning and end, that is,

$$\begin{array}{llllllllll} \mathbf{p}_0 & = & \mathbf{f}(0) & = \mathbf{a}_0 + & 0^1\,\mathbf{a}_1 & + & 0^2\,\mathbf{a}_2 + & & 0^3\,\mathbf{a}_3, \\ \mathbf{p}_1 & = & \mathbf{f}'(0) & = & \mathbf{a}_1 & +2 & 0^1\,\mathbf{a}_2 + & 3 & 0^2\,\mathbf{a}_3, \\ \mathbf{p}_2 & = & \mathbf{f}(1) & = \mathbf{a}_0 + & 1^1\,\mathbf{a}_1 & + & 1^2\,\mathbf{a}_2 + & & 1^3\,\mathbf{a}_3, \\ \mathbf{p}_3 & = & \mathbf{f}'(1) & = & \mathbf{a}_1 & +2 & 1^1\,\mathbf{a}_2 + & 3 & 1^2\,\mathbf{a}_3. \end{array}$$

Thus, the constraint matrix is

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix},$$

and the basis matrix is

$$\mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix}.$$

We will discuss Hermite cubic splines in Section 15.5.2.

15.3.5 Blending Functions

If we know the basis matrix, $\mathbf{B}$, we can multiply it by the parameter vector, $\mathbf{u}$, to get a vector of functions

$$\mathbf{b}(u) = \mathbf{u}\,\mathbf{B}.$$

Notice that we denote this vector by $\mathbf{b}(u)$ to emphasize the fact that its value depends on the free parameter u. We call the elements of $\mathbf{b}(u)$ the *blending functions*, because they specify how to blend the values of the control point vector together:

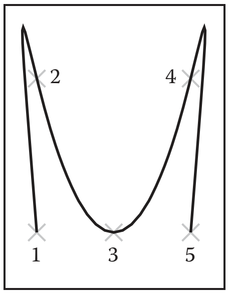
$$\mathbf{f}(u) = \sum_{i=0}^{n} \mathbf{b}_i(u)\mathbf{p}_i. \qquad (15.11)$$

It is important to note that for a chosen value of u, Equation (15.11) is a *linear* equation specifying a *linear blend* (or weighted average) of the control points. This is true no matter what degree polynomials are "hidden" inside of the $\mathbf{b}_i$ functions.

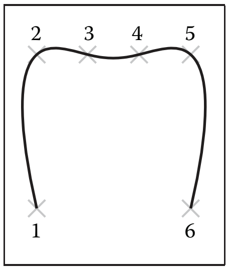
Blending functions provide a nice abstraction for describing curves. Any type of curve can be represented as a linear combination of its control points, where those weights are computed as some arbitrary functions of the free parameter.

15.3.6 Interpolating Polynomials

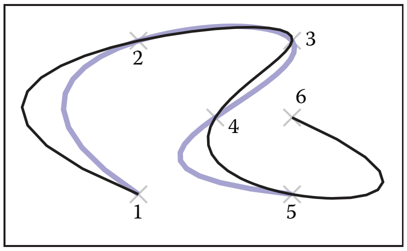
In general, a polynomial of degree n can interpolate a set of $n + 1$ values. If we are given a vector $\mathbf{p} = (p_0, \ldots, p_n)$ of points to interpolate and a vector $\mathbf{t} = (t_0, \ldots, t_n)$ of increasing parameter values, $t_i \neq t_j$, we can use the methods



(a) Interpolating polynomial through five points



(b) Interpolating polynomial through six points



(c) Interpolating polynomial through five and six points

Figure 15.3. Interpolating polynomials through multiple points. Notice the extra wiggles and over-shooting between points. In (c), when the sixth point is added, it completely changes the shape of the curve due to the non-local nature of interpolating polynomials.

described in the previous sections to determine an $n+1 \times n+1$ basis matrix that gives us a function $f(t)$ such that $f(t_i) = p_i$. For any given vector $\mathbf{t}$, we need to set up and solve an $n = 1 \times n+1$ linear system. This provides us with a set of $n+1$ basis functions that perform interpolation:

$$\mathbf{f}(t) = \sum_{i=0}^{n} \mathbf{p}_i b_i(t).$$

These interpolating basis functions can be derived in other ways. One particularly elegant way to define them is the *Lagrange form:*

$$b_i = \prod_{j=0, j\neq i}^{n} \frac{x - t_j}{t_i - t_j}. \tag{15.12}$$

There are more computationally efficient ways to express the interpolating basis functions than the Lagrange form (see De Boor (1978) for details).

Interpolating polynomials provide a mechanism for defining curves that interpolate a set of points. Figure 15.3 shows some examples. While it is possible to create a single polynomial to interpolate any number of points, we rarely use high-order polynomials to represent curves in computer graphics. Instead, interpolating splines (piecewise polynomial functions) are preferred. Some reasons for this are considered in Section 15.5.3.

15.4 Putting Pieces Together

Now that we've seen how to make individual pieces of polynomial curves, we can consider how to put these pieces together.

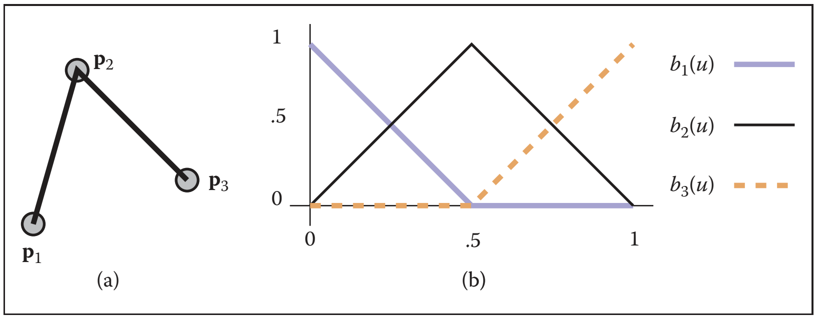


Figure 15.4. (a) Two line segments connect three points; (b) the blending functions for each of the points are graphed at right.

15.4.1 Knots

The basic idea of a piecewise parametric function is that each piece is only used over some parameter range. For example, if we want to define a function that has two piecewise linear segments that connect three points (as shown in Figure 15.4(a)), we might define

$$\mathbf{f}(u) = \begin{cases} \mathbf{f}_1(2u) & \text{if } 0 \le u < \frac{1}{2}, \\ \mathbf{f}_2(2u-1) & \text{if } \frac{1}{2} \le u < 1, \end{cases} \tag{15.13}$$

where $\mathbf{f}_1$ and $\mathbf{f}_2$ are functions for each of the two line segments. Notice that we have rescaled the parameter for each of the pieces to facilitate writing their equations as

$$\mathbf{f}_1(u) = (1-u)\mathbf{p}_1 + u\mathbf{p}_2.$$

For each polynomial in our piecewise function, there is a site (or parameter value) where it starts and ends. Sites where a piece function begins or ends are called *knots*. For the example in Equation (15.13), the values of the knots are 0, 0.5, and 1.

We may also write piecewise polynomial functions as the sum of basis functions, each scaled by a coefficient. For example, we can rewrite the two line segments of Equation (15.13) as

$$\mathbf{f}(u) = \mathbf{p}_1 b_1(u) + \mathbf{p}_2 b_2(u) + \mathbf{p}_3 b_3(u), \tag{15.14}$$

where the function $b_1(u)$ is defined as

$$b_1(u) = \begin{cases} 1-2u & \text{if } 0 \le u < \frac{1}{2}, \\ 0 & \text{otherwise}, \end{cases}$$

and b_2 and b_3 are defined similarly. These functions are plotted in Figure 15.4(b).

The knots of a polynomial function are the combination of the knots of all of the pieces that are used to create it. The *knot vector* is a vector that stores all of the knot values in ascending order.

Notice that in this section we have used two different mechanisms for combining polynomial pieces: using independent polynomial pieces for different ranges of the parameter and blending together piecewise polynomial functions.

15.4.2 Using Independent Pieces

In Section 15.3, we defined pieces of polynomials over the unit parameter range. If we want to assemble these pieces, we need to convert from the parameter of the overall function to the value of the parameter for the piece. The simplest way to do this is to define the overall curve over the parameter range $[0, n]$ where n is the number of segments. Depending on the value of the parameter, we can shift it to the required range.

15.4.3 Putting Segments Together

If we want to make a single curve from two line segments, we need to make sure that the end of the first line segment is at the same location as the beginning of the next. There are three ways to connect the two segments (in order of simplicity):

1. Represent the line segment as its two endpoints, and then use the same point for both. We call this a *shared-point* scheme.

2. Copy the value of the end of the first segment to the beginning of the second segment every time that the parameters of the first segment change. We call this a *dependency* scheme.

3. Write an explicit equation for the connection, and enforce it through numerical methods as the other parameters are changed.

While the simpler schemes are preferable since they require less work, they also place more restrictions on the way the line segments are parameterized. For example, if we want to use the center of the line segment as a parameter (so that the user can specify it directly), we will use the beginning of each line segment and the center of the line segment as their parameters. This will force us to use the dependency scheme.

Notice that if we use a shared-point or dependency scheme, the total number of control points is less than $n * m$, where n is the number of segments and m

is the number of control points for each segment; many of the control points of the independent pieces will be computed as functions of other pieces. Notice that if we use either the shared-point scheme for lines (each segment uses its two endpoints as parameters and shares interior points with its neighbors), or if we use the dependency scheme (such as the example one with the first endpoint and midpoint), we end up with $n + 1$ controls for an n-segment curve.

Dependency schemes have a more serious problem. A change in one place in the curve can propagate through the entire curve. This is called a lack of *locality.* Locality means that if you move a point on a curve it will only affect a local region. The local region might be big, but it will be finite. If a curve's controls do not have locality, changing a control point may affect points infinitely far away.

To see locality, and the lack thereof, in action, consider two chains of line segments, as shown in Figure 15.5. One chain has its pieces parameterized by their endpoints and uses point-sharing to maintain continuity. The other has its pieces parameterized by an endpoint and midpoint and uses dependency propagation to keep the segments together. The two segment chains can represent the same curves: they are both a set of n connected line segments. However, because of locality issues, the endpoint-shared form is likely to be more convenient for the user. Consider changing the position of the first control point in each chain. For the endpoint-shared version, only the first segment will change, while all of the segments will be affected in the midpoint version, as in Figure 15.5. In fact, for any point moved in the endpoint-shared version, at most two line segments will change. In the midpoint version, all segments after the control point that is moved will change, even if the chain is infinitely long.

In this example, the dependency propagation scheme was the one that did not have local control. This is not always true. There are direct sharing schemes that are not local and propagation schemes that are local.

We emphasize that locality is a convenience of control issue. While it is inconvenient to have the entire curve change every time, the same changes can be made to the curve. It simply requires moving several points in unison.

15.5 Cubics

In graphics, when we represent curves using piecewise polynomials, we usually use either line segments or cubic polynomials for the pieces. There are a number of reasons why cubics are popular in computer graphics:

- Piecewise cubic polynomials allow for C^2 continuity, which is generally sufficient for most visual tasks. The C^1 smoothness that quadratics offer is

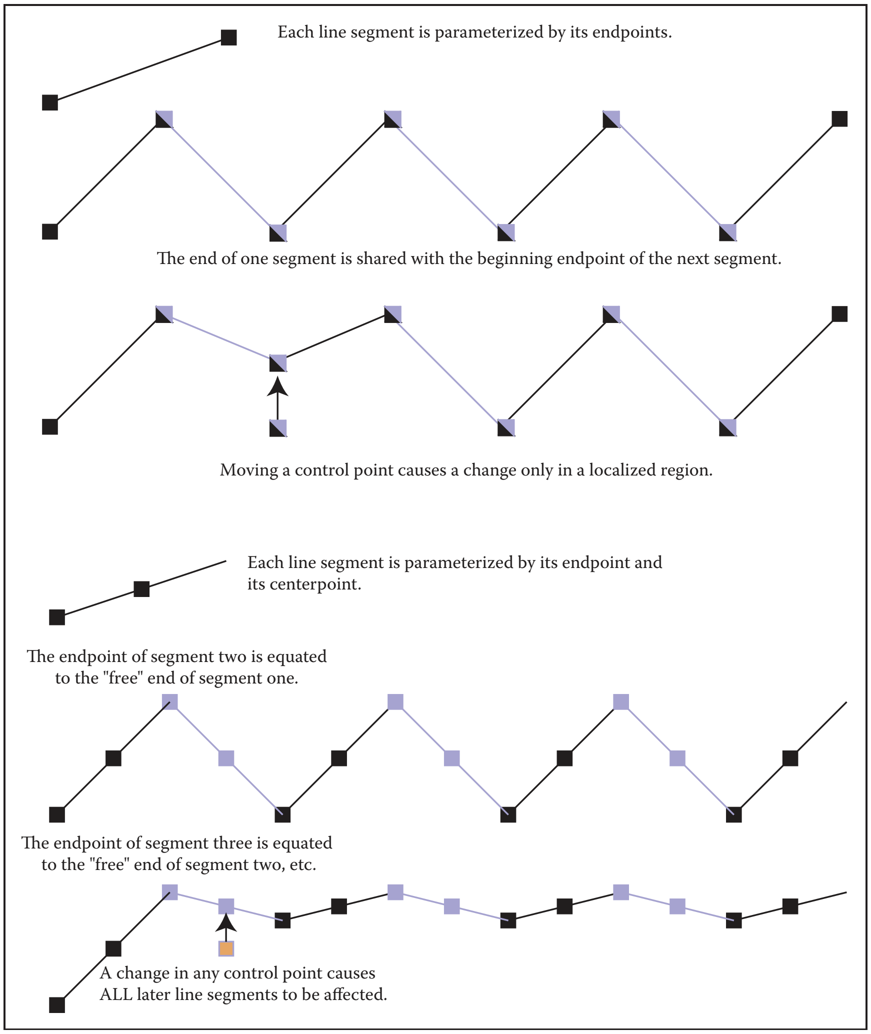

Each line segment is parameterized by its endpoints.
The end of one segment is shared with the beginning endpoint of the next segment.
Moving a control point causes a change only in a localized region.
Each line segment is parameterized by its endpoint and its centerpoint.
The endpoint of segment two is equated to the "free" end of segment one.
The endpoint of segment three is equated to the "free" end of segment two, etc.
A change in any control point causes ALL later line segments to be affected.

Figure 15.5. A chain of line segments with local control and one with non-local control.

often insufficient. The greater smoothness offered by higher-order polynomials is rarely important.

- Cubic curves provide the minimum-curvature interpolants to a set of points. That is, if you have a set of $n + 3$ points and define the "smoothest" curve that passes through them (that is the curve that has the minimum curvature over its length), this curve can be represented as a piecewise cubic with n segments.

- Cubic polynomials have a nice symmetry where position and derivative can be specified at the beginning and end.

- Cubic polynomials have a nice tradeoff between the numerical issues in computation and the smoothness.

Notice that we do not have to use cubics; they just tend to be a good tradeoff between the amount of smoothness and complexity. Different applications may have different tradeoffs. We focus on cubics since they are the most commonly used.

The canonical form of a cubic polynomial is

$$\mathbf{f}(u) = \mathbf{a}_0 + \mathbf{a}_1\, u + \mathbf{a}_2\, u^2 + \mathbf{a}_3\, u^3.$$

As we discussed in Section 15.3, these canonical form coefficients are not a convenient way to describe a cubic segment.

We seek forms of cubic polynomials for which the coefficients are a convenient way to control the resulting curve represented by the cubic. One of the main conveniences will be to provide ways to ensure the connectedness of the pieces and the continuity between the segments.

Each cubic polynomial piece requires four coefficients or control points. That means for a piecewise polynomial with n pieces, we may require up to $4n$ control points if no sharing between segments is done or dependencies used. More often, some part of each segment is either shared or depends on an adjacent segment, so the total number of control points is much lower. Also, note that a control point might be a position or a derivative of the curve.

Unfortunately, there is no single “best” representation for a piecewise cubic. It is not possible to have a piecewise polynomial curve representation that has all of the following desirable properties:

1. each piece of the curve is a cubic;

2. the curve interpolates the control points;

3. the curve has local control;

4. the curve has C^2 continuity.

We can have any three of these properties, but not all four; there are representations that have any combination of three. In this book, we will discuss cubic B-splines that do not interpolate their control points (but have local control and are C^2); Cardinal splines and Catmull-Rom splines that interpolate their control points and offer local control, but are not C^2; and natural cubics that interpolate and are C^2, but do not have local control.

The continuity properties of cubics refer to the continuity between the segments (at the knot points). The cubic pieces themselves have infinite continuity in their derivatives (the way we have been talking about continuity so far). Note that if you have a lot of control points (or knots), the curve can be wiggly, which might not seem "smooth."

15.5.1 Natural Cubics

With a piecewise cubic curve, it is possible to create a C^2 curve. To do this, we need to specify the position and first and second derivative at the beginning of each segment (so that we can make sure that it is the same as at the end of the previous segment). Notice that each curve segment receives three out of its four parameters from the previous curve in the chain. These C^2 continuous chains of cubics are sometimes referred to as *natural* cubic splines.

For one segment of the natural cubic, we need to parameterize the cubic by the positions of its endpoints and the first and second derivative at the beginning point. The control points are therefore

$$\begin{array}{rlllllll}
\mathbf{p}_0 = & \mathbf{f}(0) & = \mathbf{a}_0 & + & 0^1\mathbf{a}_1 & + & 0^2\,\mathbf{a}_2 & + \;\; 0^3\,\mathbf{a}_3, \\
\mathbf{p}_1 = & \mathbf{f}'(0) & = & & 1^1\mathbf{a}_1 & +2 & 0^1\,\mathbf{a}_2 & +3 \;\; 0^2\,\mathbf{a}_3, \\
\mathbf{p}_2 = & \mathbf{f}''(0) & = & & & 2 & 1^1\mathbf{a}_2 & +6 \;\; 0^1\,\mathbf{a}_3, \\
\mathbf{p}_3 = & \mathbf{f}(1) & = \mathbf{a}_0 & + & 1^1\,\mathbf{a}_1 & + & 1^2\,\mathbf{a}_2 & + \;\; 1^3\,\mathbf{a}_3.
\end{array}$$

Therefore, the constraint matrix is

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

and the basis matrix is

$$\mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5 & 0 \\ -1 & -1 & -.5 & 1 \end{bmatrix}.$$

Given a set of n control points, a natural cubic spline has $n-1$ cubic segments. The first segment uses the control points to define its beginning position, ending position, and first and second derivative at the beginning. A dependency scheme copies the position, and first and second derivative of the end of the first segment for use in the second segment.

A disadvantage of natural cubic splines is that they are not local. Any change in any segment may require the entire curve to change (at least the part after the change was made). To make matters worse, natural cubic splines tend to be ill-conditioned: a small change at the beginning of the curve can lead to large changes later. Another issue is that we only have control over the derivatives of the curve at its beginning. Segments after the beginning of the curve determine their derivatives from their beginning point.

15.5.2 Hermite Cubics

Hermite cubic polynomials were introduced in Section 15.3.4. A segment of a cubic Hermite spline allows the positions and first derivatives of both of its endpoints to be specified. A chain of segments can be linked into a C^1 spline by using the same values for the position and derivative of the end of one segment and for the beginning of the next.

Given a set of n control points, where every other control point is a derivative value, a cubic Hermite spline contains $(n-2)/2$ cubic segments. The spline interpolates the points, as shown in Figure 15.6, but can guarantee only C^1 continuity.

Hermite cubics are convenient because they provide local control over the shape, and provide C^1 continuity. However, since the user must specify both positions and derivatives, a special interface for the derivatives must be provided. One possibility is to provide the user with points that represent where the derivative vectors would end if they were "placed" at the position point.

15.5.3 Cardinal Cubics

A *cardinal cubic spline* is a type of C^1 interpolating spline made up of cubic polynomial segments. Given a set of n control points, a cardinal cubic spline uses

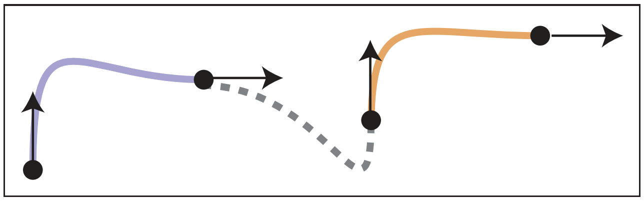


Figure 15.6. A Hermite cubic spline made up of three segments.

$n-2$ cubic polynomial segments to interpolate all of its points except for the first and last.

Cardinal splines have a parameter called *tension* that controls how “tight” the curve is between the points it interpolates. The tension is a number in the range $[0, 1)$ that controls how the curve bends toward the next control point. For the important special case of $t = 0$, the splines are called *Catmull-Rom* splines.

Each segment of the cardinal spline uses four control points. For segment i, the points used are i, $i+1$, $i+2$, and $i+3$ as the segments share three points with their neighbors. Each segment begins at its second control point and ends at its third control point. The derivative at the beginning of the curve is determined by the vector between the first and third control points, while the derivative at the end of the curve is given by the vector between the second and fourth points, as shown in Figure 15.7.

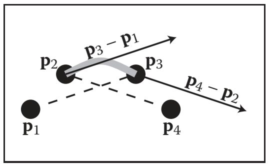


Figure 15.7. A segment of a cardinal cubic spline interpolates its second and third control points ($\mathbf{p}_2$ and $\mathbf{p}_3$), and uses its other points to determine the derivatives at the beginning and end.

The tension parameter adjusts how much the derivatives are scaled. Specifically, the derivatives are scaled by $(1-t)/2$. The constraints on the cubic are therefore

$$\begin{aligned}
\mathbf{f}(0) &= \mathbf{p}_2,\\
\mathbf{f}(1) &= \mathbf{p}_3,\\
\mathbf{f}'(0) &= \tfrac{1}{2}(1-t)(\mathbf{p}_3-\mathbf{p}_1),\\
\mathbf{f}'(1) &= \tfrac{1}{2}(1-t)(\mathbf{p}_4-\mathbf{p}_2).
\end{aligned}$$

Solving these equations for the control points (defining $s = (1-t)/2)$ gives

$$\begin{array}{rlllllll}
\mathbf{p}_0 = & \mathbf{f}(1) - \frac{2}{1-t}\mathbf{f}'(0) & = \mathbf{a}_0 & +(1-\frac{1}{s}) & \mathbf{a}_1 & + & \mathbf{a}_2 & + \quad \mathbf{a}_3,\\
\mathbf{p}_1 = & \mathbf{f}(0) & = \mathbf{a}_0, & & & & & \\
\mathbf{p}_2 = & \mathbf{f}(1) & = \mathbf{a}_0 & + & \mathbf{a}_1 & + & \mathbf{a}_2 & + \quad \mathbf{a}_3,\\
\mathbf{p}_3 = & \mathbf{f}(0) + \frac{1}{s}\mathbf{f}'(1) & = \mathbf{a}_0 & +\frac{1}{s} & \mathbf{a}_1 & +2\frac{1}{s} & \mathbf{a}_2 & +3\frac{1}{s} \quad \mathbf{a}_3.
\end{array}$$

This yields the cardinal matrix

$$\mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0\\ -s & 0 & s & 0\\ 2s & s-3 & 3-2s & -s\\ -s & 2-s & s-2 & s \end{bmatrix}.$$

Since the third point of segment i is the second point of segment $i+1$, adjacent segments of the cardinal spline connect. Similarly, the same points are used to specify the first derivative of each segment, providing C^1 continuity.

Cardinal splines are useful, because they provide an easy way to interpolate a set of points with C^1 continuity and local control. They are only C^1, so they sometimes get “kinks” in them. The tension parameter gives some control over what happens between the interpolated points, as shown in Figure 15.8, where a

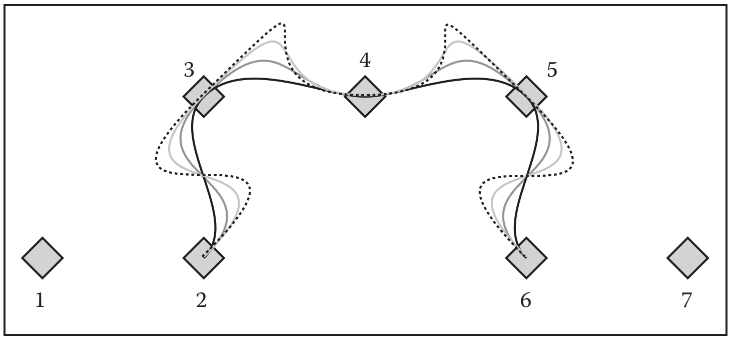


Figure 15.8. Cardinal splines through seven control points with varying values of tension parameter *t*.

set of cardinal splines through a set of points is shown. The curves use the same control points, but they use different values for the tension parameters. Note that the first and last control points are not interpolated.

Given a set of n points to interpolate, you might wonder why we might prefer to use a cardinal cubic spline (that is a set of $n-2$ cubic pieces) rather than a single, order n polynomial as described in Section 15.3.6. Some of the disadvantages of the interpolating polynomial are:

- The interpolating polynomial tends to overshoot the points, as seen in Figure 15.9. This overshooting gets worse as the number of points grows larger. The cardinal splines tend to be well behaved in between the points.

- Control of the interpolating polynomial is not local. Changing a point at the beginning of the spline affects the entire spline. Cardinal splines are local: any place on the spline is affected by its four neighboring points at most.

- Evaluation of the interpolating polynomial is not local. Evaluating a point on the polynomial requires access to all of its points. Evaluating a point on the piecewise cubic requires a fixed small number of computations, no matter how large the total number of points is.

There are a variety of other numerical and technical issues in using interpolating splines as the number of points grows larger. See De Boor (2001) for more information.

A cardinal spline has the disadvantage that it does not interpolate the first or last point, which can be easily fixed by adding an extra point at either end of the sequence. The cardinal spline also is not as continuous—providing only C^1 continuity at the knots.

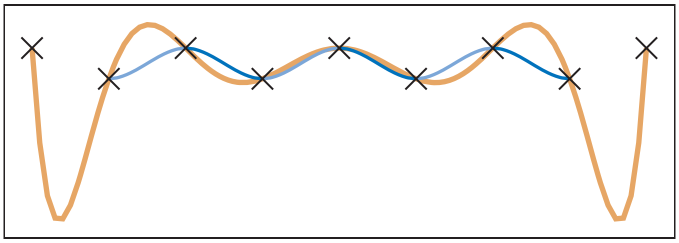


Figure 15.9. Splines interpolating nine control points (marked with small crosses). The thick orange line shows an interpolating polynomial. The thin line shows a Catmull-Rom spline. The latter is made of seven cubic segments, which are each shown in alternating blue tones.

15.6 Approximating Curves

It might seem like the easiest way to control a curve is to specify a set of points for it to interpolate. In practice, however, interpolation schemes often have undesirable properties because they have less continuity and offer no control of what happens between the points. Curve schemes that only approximate the points are often preferred. With an approximating scheme, the control points influence the shape of the curve, but do not specify it exactly. Although we give up the ability to directly specify points for the curve to pass through, we gain better behavior of the curve and local control. Should we need to interpolate a set of points, the positions of the control points can be computed such that the curve passes through these interpolation points.

The two most important types of approximating curves in computer graphics are Bézier curves and B-spline curves.

15.6.1 Bézier Curves

Bézier curves are one of the most common representations for free-form curves in computer graphics. The curves are named for Pierre Bézier, one of the people who was instrumental in their development. Bézier curves have an interesting history where they were concurrently developed by several independent groups.

A Bézier curve is a polynomial curve that approximates its control points. The curves can be a polynomial of any degree. A curve of degree d is controlled by $d + 1$ control points. The curve interpolates its first and last control points, and the shape is directly influenced by the other points.

Often, complex shapes are made by connecting a number of Bézier curves of low degree, and in computer graphics, cubic ($d = 3$) Bézier curves are commonly used for this purpose. Many popular illustration programs, such as Adobe Illus-