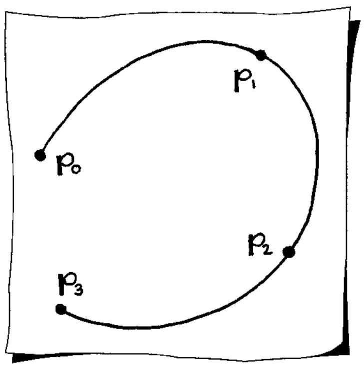
5

Putting Curves to Work

In this chapter, we will see the main uses of parametric curves: describing geometric shapes using methods such as interpolation and approximation.

5.1 Cubic Interpolation



Sketch 33.
A cubic Bézier curve through four given points.

Suppose you are given four points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ and you wish to pass a curve through them, just like the situation shown in Sketch 5.1. There, the points are 2D, but they might as well be 3D. This is called *interpolation.*

We may choose among many kinds of curves; for right now, we'll decide to use a cubic Bézier curve. Every point on a Bézier curve has a parameter value; in order to solve our problem, we have to assign a parameter value t_i to every $\mathbf{p}_i$. A natural choice is to associate each $\mathbf{p}_i$ with a parameter value $t_i = i/3$. Many other suitable choices exist;[1] more on this topic in Section 5.4.

Now our interpolation problem becomes:

[1] We must require that the t_i are increasing and no two t_i are the same.

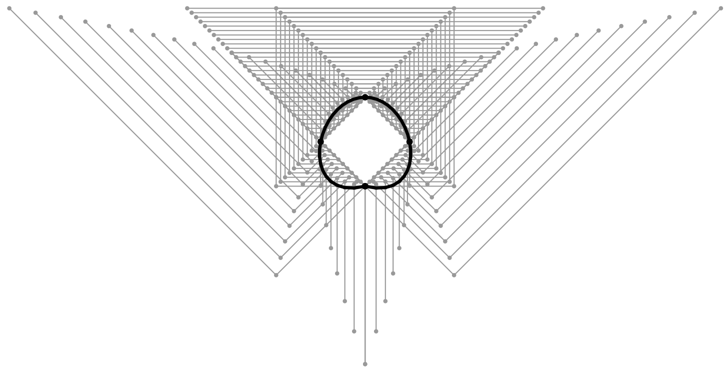


Figure 5.1.
An interpolating polynomial curve, evaluated at fourty points. Intermediate steps in the computations are also shown.

Given four point/parameter pairs $(\mathbf{p}_i, t_i)$, find a cubic Bézier curve $\mathbf{x}(t)$ such that

$$\mathbf{x}(t_i) = \mathbf{p}_i; \quad i = 0, 1, 2, 3. \tag{5.1}$$

This simply states that we want the Bézier curve to pass through the data points at the right parameter values.

The desired Bézier curve will be of the form

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

Writing out all interpolation conditions (5.1) yields

$$
\begin{aligned}
\mathbf{p}_0 &= B_0^3(t_0)\mathbf{b}_0 + B_1^3(t_0)\mathbf{b}_1 + B_2^3(t_0)\mathbf{b}_2 + B_3^3(t_0)\mathbf{b}_3,\\
\mathbf{p}_1 &= B_0^3(t_1)\mathbf{b}_0 + B_1^3(t_1)\mathbf{b}_1 + B_2^3(t_1)\mathbf{b}_2 + B_3^3(t_1)\mathbf{b}_3,\\
\mathbf{p}_2 &= B_0^3(t_2)\mathbf{b}_0 + B_1^3(t_2)\mathbf{b}_1 + B_2^3(t_2)\mathbf{b}_2 + B_3^3(t_2)\mathbf{b}_3,\\
\mathbf{p}_3 &= B_0^3(t_3)\mathbf{b}_0 + B_1^3(t_3)\mathbf{b}_1 + B_2^3(t_3)\mathbf{b}_2 + B_3^3(t_3)\mathbf{b}_3.
\end{aligned}
$$

These are four equations in the four unknowns $\mathbf{b}_0, \dots, \mathbf{b}_3$. In order to find a solution, it helps to write them in matrix form:

$$
\begin{bmatrix}\mathbf{p}_0\\ \mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\end{bmatrix} = \begin{bmatrix} B_0^3(t_0) & B_1^3(t_0) & B_2^3(t_0) & B_3^3(t_0)\\ B_0^3(t_1) & B_1^3(t_1) & B_2^3(t_1) & B_3^3(t_1)\\ B_0^3(t_2) & B_1^3(t_2) & B_2^3(t_2) & B_3^3(t_2)\\ B_0^3(t_3) & B_1^3(t_3) & B_2^3(t_3) & B_3^3(t_3)\end{bmatrix} \begin{bmatrix}\mathbf{b}_0\\ \mathbf{b}_1\\ \mathbf{b}_2\\ \mathbf{b}_3\end{bmatrix}. \tag{5.2}
$$

We further abbreviate this as

$$\mathbf{P} = M\mathbf{B}. \tag{5.3}$$

The solution is now simply

$$\mathbf{B} = M^{-1}\mathbf{P}. \tag{5.4}$$

While this looks like the solution to *one* linear system, it is really the solution to two or three systems, depending on the dimensionality of the $\mathbf{p}_i$. An example should clarify:

EXAMPLE 5.1 Let the $\mathbf{p}_i$ be given by

$$
\mathbf{p}_0 = \begin{bmatrix}-1\\0\end{bmatrix}, \quad \mathbf{p}_1 = \begin{bmatrix}0\\1\end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix}0\\-1\end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix},
$$

and set $t_i = i/3$. Then the matrix M for our linear system becomes

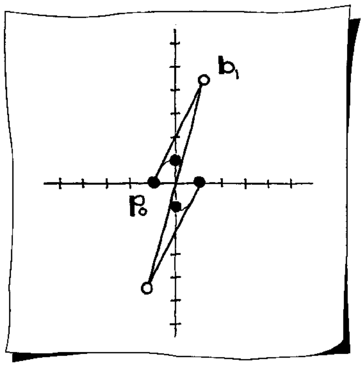
$$
M = \frac{1}{27}\begin{bmatrix}27 & 0 & 0 & 0\\ 8 & 12 & 6 & 1\\ 1 & 6 & 12 & 8\\ 0 & 0 & 0 & 27\end{bmatrix}.
$$

We send M and, successively, the two right-hand side vectors,

$$\begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}$$

to a linear system solver, which results in the two solution vectors

$$\begin{bmatrix} -1 \\ 7/6 \\ -7/6 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 \\ 9/2 \\ -9/2 \\ 0 \end{bmatrix}.$$



Sketch 34.
Cubic Bézier interpolation.

Thus, the Bézier points for the interpolating cubic are

$$\mathbf{b}_0 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \mathbf{b}_1 = \begin{bmatrix} 7/6 \\ 9/2 \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} -7/6 \\ -9/2 \end{bmatrix}, \quad \mathbf{b}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

This example is outlined in Sketch 5.1.

Note that if you explicitly computed M^{-1}, you may apply it to any number of cubic interpolation problems, just as long as you keep the choice of the t_i unchanged!

5.2 Interpolation Beyond Cubics

Polynomial interpolation also works when more than four data points are given. We would then have data points $\mathbf{p}_0, \ldots, \mathbf{p}_n$ and corresponding parameter values $t_0, \ldots, t_n$. The interpolation problem again leads to a linear system

$$\mathbf{P} = M\mathbf{B}; \tag{5.5}$$

now M is an $(n+1) \times (n+1)$ matrix with elements

$$m_{i,j} = B_j^n(t_i).$$

Again, it is solved using any linear system solver.

While polynomial interpolation is guaranteed to work, it does not produce satisfying results for higher degrees. Figure 5.2 should be convincing enough. The top data set are simply 16 points read off a semicircle, spaced at equal angle increments. The bottom data set has one change: The gray data point now has an $x-$coordinate of 0.1025, whereas the correct value (supplied in the top part) is 0.1045. Thus a small change in data can lead to large changes in the interpolating curve. Processes with this property are called ill-conditioned.

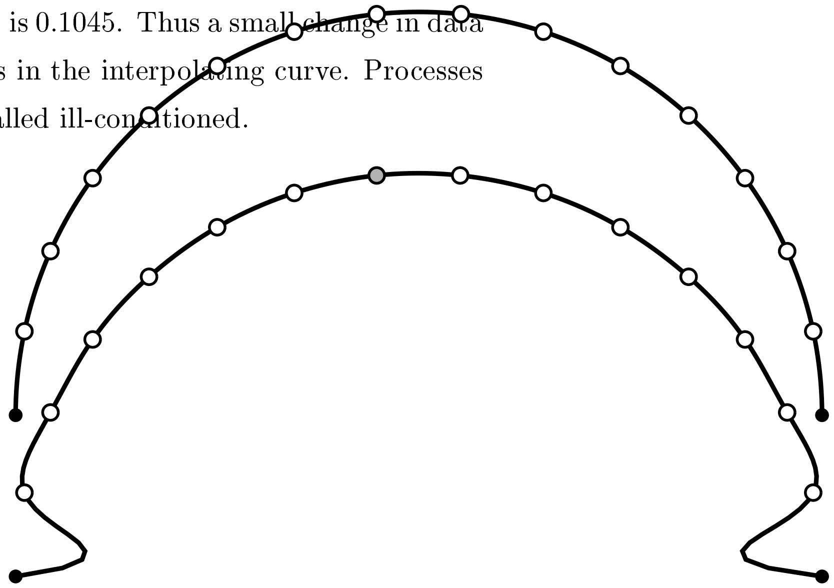


Figure 5.2.
Top: data from a circle; bottom: one point slightly modified.

We used the Bézier form of the interpolating curve here. Different polynomial forms will give the identical result. For example, we might write the interpolating curve in monomial form:

$$\mathbf{x}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \ldots + \mathbf{a}_n t^n.$$

Now the unknowns are the coefficients $\mathbf{a}_i$. This interpolation problem also leads to a linear system, written as

$$\mathbf{P} = M\mathbf{A};$$

now M is an $(n+1) \times (n+1)$ matrix with elements

$$m_{i,j} = t_i^j.$$

The unknowns $\mathbf{a}_i$ are collected in $\mathbf{A}$. The curve resulting from this system is the same as the one resulting from (5.5).

Another way of writing the interpolating polynomial curve is by usimg *Lagrange polynomials*. These are defined by

$$L_i^n(t) = \frac{(t-t_0)\ldots(t-t_{i-1})(t-t_{i+1})\ldots(t-t_n)}{(t_i-t_0)\ldots(t_i-t_{i-1})(t_i-t_{i+1})\ldots(t_i-t_n)}, \tag{5.6}$$

and allow a very direct form for the interpolant:

$$\mathbf{x}(t) = L_0^n(t)\mathbf{p}_0 + \ldots + L_n^n(t)\mathbf{p}_n. \tag{5.7}$$

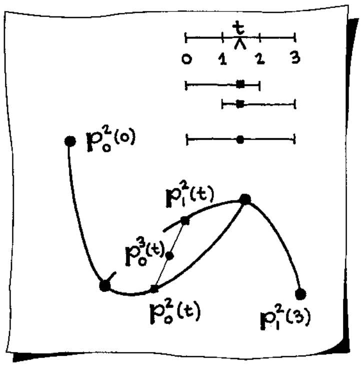
Because the data points appear explicitly, this is called the *cardinal form* of the interpolant to data points and parameters. Notice that in the numerator and denominator of the i^{th} Lagrange polynomial, the $(* - t_i)$ term is missing.

5.3 Aitken's Algorithm

This is a recursive algorithm to compute points on the interpolating polynomial curve; it has some of the characteristics of the de Casteljau algorithm. It is best explained for the cubic case. Assume that we have (somehow) found a quadratic curve $\mathbf{p}_0^2(t)$ through $\mathbf{p}_0, \mathbf{p}_1$, and $\mathbf{p}_2$ as well as another one, $\mathbf{p}_1^2(t)$ through $\mathbf{p}_1, \mathbf{p}_2$, and $\mathbf{p}_3$. From these two curves, it is possible to find the interpolating cubic by simple linear interpolation:

$$\mathbf{p}_0^3(t) = \frac{t_3 - t}{t_3 - t_0}\mathbf{p}_0^2(t) + \frac{t - t_0}{t_3 - t_0}\mathbf{p}_1^2(t). \tag{5.8}$$

Sketch 5.3 illustrates.



Sketch 35.
Aitken's algorithm.

It is easy to verify that (5.8) does indeed interpolate to all four data points. Let's first check $\mathbf{p}_0$:

$$\mathbf{p}_0^3(t_0) = \frac{t_3 - t_0}{t_3 - t_0}\mathbf{p}_0^2(t_0) + \frac{t_0 - t_0}{t_3 - t_0}\mathbf{p}_1^2(t_0) = \mathbf{p}_0.$$

Interpolation at $\mathbf{p}_3$ is established in the same way. Now for $\mathbf{p}_1$: We simply observe that the factors in (5.8) sum to one. Since both $\mathbf{p}_0^2(t_1) = \mathbf{p}_1$ and $\mathbf{p}_1^2(t_1) = \mathbf{p}_1$, it then follows that also $\mathbf{p}_0^3(t_1) = \mathbf{p}_1$. For $\mathbf{p}_2$, an analogous argument holds.

This begs a question, namely how did we find the quadratic interpolants to begin with? The same process works again:

$$\mathbf{p}_0^2(t) = \frac{t_2 - t}{t_2 - t_0}\mathbf{p}_0^1(t) + \frac{t - t_0}{t_2 - t_0}\mathbf{p}_1^1(t), \tag{5.9}$$

$$\mathbf{p}_1^2(t) = \frac{t_3 - t}{t_3 - t_1}\mathbf{p}_1^1(t) + \frac{t - t_1}{t_3 - t_1}\mathbf{p}_2^1(t). \tag{5.10}$$

In this equation, new terms $\mathbf{p}_i^1$ are introduced, but they are simply linear interpolants of the data. For example,

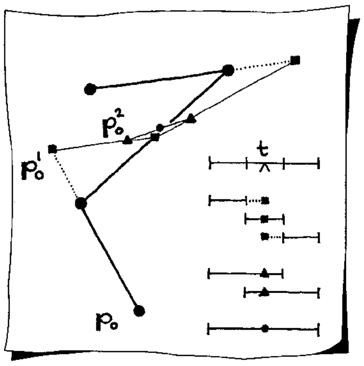
$$\mathbf{p}_1^1(t) = \frac{t_2 - t}{t_2 - t_1}\mathbf{p}_1 + \frac{t - t_1}{t_2 - t_1}\mathbf{p}_2.$$

Just as in the de Casteljau algorithm, it is convenient to arrange the intermediate points in a triangular array:

$$\begin{array}{llll}
\mathbf{p}_0 & & & \\
\mathbf{p}_1 & \mathbf{p}_0^1 & & \\
\mathbf{p}_2 & \mathbf{p}_1^1 & \mathbf{p}_0^2 & \\
\mathbf{p}_3 & \mathbf{p}_2^1 & \mathbf{p}_1^2 & \mathbf{p}_0^3.
\end{array}$$

The points in the left-most column are given (as well as a parameter value for each point). Aitken's algorithm computes the points in each successive column, and the point on the curve is $\mathbf{p}_0^3$.

See Sketch 5.3 for the geometry of this computation. The parameter spans in the steps of Aitken's algorithm, such as in (5.8), can seem pretty complicated at first. However, there is an easy way to keep track of this. As depicted in Sketch 5.3 the first step of the algorithm where the $\mathbf{p}_i^1$ are computed involves spans in the parameter space of length one. Consider all such spans, and with respect to each, determine where the current t-value resides. The span itself corresponds to the input $\mathbf{p}_i$, $\mathbf{p}_{i+1}$ points. In the next step, where the $\mathbf{p}_i^2$ are computed, we now use spans of length two. The points of Step 2 are calculated using the points from Step 1.



Sketch 36.
The intermediate steps in Aitken's algorithm.

EXAMPLE 5.2 We will evaluate the interpolating cubic through the four data points

$$\mathbf{p}_0 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \mathbf{p}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

with parameter values $(t_0, t_1, t_2, t_3) = (0, 1, 2, 3)$, at $t = 1.5$.

The corresponding scheme is:

$$
\begin{array}{cccc}
\begin{bmatrix}-1\\0\end{bmatrix} & & & \\
\begin{bmatrix}0\\1\end{bmatrix} & \begin{bmatrix}0.5\\1.5\end{bmatrix} & & \\
\begin{bmatrix}0\\-1\end{bmatrix} & \begin{bmatrix}0\\0\end{bmatrix} & \begin{bmatrix}0.125\\0.375\end{bmatrix} & \\
\begin{bmatrix}1\\0\end{bmatrix} & \begin{bmatrix}-0.5\\-1.5\end{bmatrix} & \begin{bmatrix}-0.125\\-0.375\end{bmatrix} & \begin{bmatrix}0\\0\end{bmatrix}
\end{array},
$$

hence

$$\mathbf{p}_0^3(1.5) = \begin{bmatrix}0\\0\end{bmatrix}.$$

A sampling of the computation of the intermediate points:

$$
\begin{aligned}
\mathbf{p}_0^1 &= -0.5\mathbf{p}_0 + 1.5\mathbf{p}_1 \\
\mathbf{p}_0^2 &= 0.25\mathbf{p}_0^1 + 0.75\mathbf{p}_1^1 \\
\mathbf{p}_0^3 &= 0.5\mathbf{p}_0^2 + 0.5\mathbf{p}_1^2.
\end{aligned}
$$

The symmetry in this example makes the other combinations easy to guess.

Aitken's algorithm works for degrees other than cubic. For an n^{th} degree interpolating curve, we will generate intermediate points $\mathbf{p}_i^r$, computed by

$$\mathbf{p}_i^r(t) = \frac{t_{i+r} - t}{t_{i+r} - t_i}\mathbf{p}_i^{r-1}(t) + \frac{t - t_i}{t_{i+r} - t_i}\mathbf{p}_{i+1}^{r-1}(t) \tag{5.11}$$

for $r = 1, \ldots, n$ and $i = 0, \ldots, n-r$. The general form (5.11) of Aitken's algorithm uses *linear interpolation* between two points

$\mathbf{p}_i^{r-1}$ and $\mathbf{p}_{i+1}^{r-1}$ over the parameter interval $[t_{i+r}, t_i]$. This may be viewed as an affine map of this interval onto the line through $\mathbf{p}_i^{r-1}$ and $\mathbf{p}_{i+1}^{r-1}$. (This point was illustrated in Sketch 5.3).

Figure 5.1 illustrates Aitken's algorithm when executed for fourty evaluations. All intermediate points $\mathbf{p}_i^r$ are shown in that figure. Compare with the corresponding Figure 4.4 for the de Casteljau algorithm!

Did you notice that every data point is involved in the calculation of a point on the interpolating curve? Polynomial interpolation is a *global* operation, as we see in the formulation (5.5).

5.4 Approximation

In many applications, one is given more data points than should be interpolated by a polynomial curve: recall from Section 5.2 that higher degree interpolation becomes ill-conditioned. In such cases, an *approximating* curve will be needed. Such a curve does not pass through the data points exactly; rather it passes near them, still capturing the shape suggested by the given points. The technique best known for finding such curves is known as *least squares approximation*. Figure 5.3 illustrates.

We are now given $l+1$ data points $\mathbf{p}_0, \ldots, \mathbf{p}_l$, each $\mathbf{p}_i$ being associated with a parameter value t_i. We wish to find a polynomial curve $\mathbf{x}(t)$ of a given degree n such that the distances $\|\mathbf{p}_i - \mathbf{x}(t_i)\|$ are small. Ideally, we would have $\mathbf{p}_i = \mathbf{x}(t_i); i = 0, \ldots, l$. If our polynomial curve $\mathbf{x}(t)$ is of the form

$$\mathbf{x}(t) = \mathbf{b}_0 B_0^n(t) + \ldots + \mathbf{b}_n B_n^n(t)$$

we would like the following to hold:

$$\begin{aligned} \mathbf{b}_0 B_0^n(t_0) + \ldots + \mathbf{b}_n B_n^n(t_0) &= \mathbf{p}_0 \\ &\vdots \\ \mathbf{b}_0 B_0^n(t_l) + \ldots + \mathbf{b}_n B_n^n(t_l) &= \mathbf{p}_l. \end{aligned}$$

This may be condensed into matrix form:

$$\begin{bmatrix} B_0^n(t_0) & \ldots & B_n^n(t_0) \\ & \vdots & \\ & \vdots & \\ B_0^n(t_l) & \ldots & B_n^n(t_l) \end{bmatrix} \begin{bmatrix} \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 \\ \vdots \\ \vdots \\ \mathbf{p}_l \end{bmatrix}. \tag{5.12}$$

Or, even shorter:

$$M\mathbf{B} = \mathbf{P}. \tag{5.13}$$

Since we assume the number l of data points is larger than the degree n of the curve, this linear system is clearly *overdetermined.* We attack it by simply multiplying both sides by M^{T}:

$$M^{\mathrm{T}} M\mathbf{B} = M^{\mathrm{T}}\mathbf{P}. \tag{5.14}$$

This is a linear system with $n+1$ equations in $n+1$ unknowns, with a square and symmetric coefficient matrix $M^{\mathrm{T}}M$. Its solution is straightforward since $M^{\mathrm{T}}M$ is always invertible. The linear system (5.14) is called the system of *normal equations.* The curve $\mathbf{B}$ is the one polynomial of degree n which minimizes the sum of the $\|\mathbf{p}_i - \mathbf{x}(t_i)\|$. Note that any modification of the t_i would result in an entirely different solution.

An example is shown in Figure 5.3. The data set, 79 points, are from a cross section of a wing (it's a noisy data set). It is

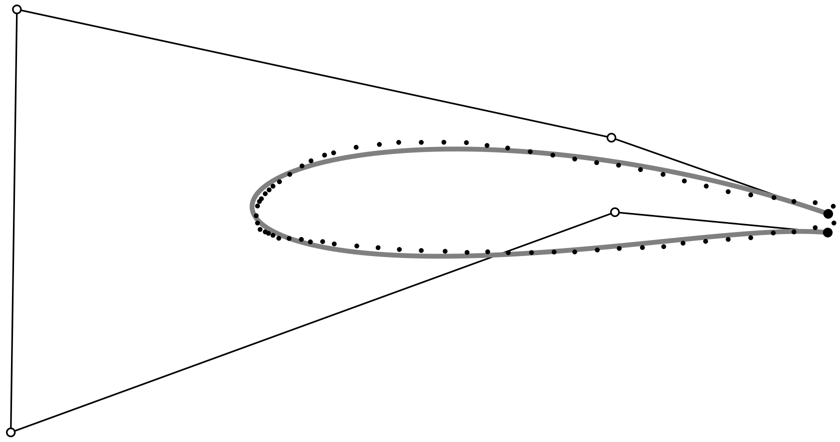


Figure 5.3.
Least squares approximation to a wing. A quintic Bézier curve with chord length parameters assigned to the data. Figure courtesy of Jeong-Jun Song.

approximated by a least squares quintic using parameter values selected to reflect the spacing of the data.

The choice of the "right" degree for this type of problem is not easy. Typically, it would be a trial-and-error process.

We used the Bézier representation above, however (5.12) could be rewritten using any other set of basis functions, such as the monomial form.

5.5 Finding the Right Parameters

In both curve interpolation and approximation, we encountered statements like "given parameter values t_i." In practice, these are not normally given and have to be made up. If there are l

points $\mathbf{p}_i$, an obvious choice is to set $t_i = i/l$; this is known as the *uniform set of parameters* .

Another, not too involved choice is to arrange the parameters such that they reflect the spacing of the data points: If the distance between two points is relatively large, then their parameter values should also be fairly different. This method is known as *chord length parameters* and is given by

$$t_0 = 0 \tag{5.15}$$

$$t_1 = t_0 + \|\mathbf{p}_1 - \mathbf{p}_0\| \tag{5.16}$$

$$\vdots \tag{5.17}$$

$$t_P = t_{l-1} + \|\mathbf{p}_l - \mathbf{p}_{l-1}\|. \tag{5.18}$$

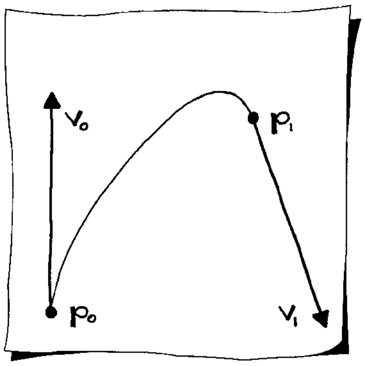
If desired (it makes no difference to the interpolation or approximation result), the parameters may be *normalized* by scaling the parameters to live between zero and one:

$$t_i = \frac{t_i - t_0}{t_l - t_0}.$$

In general, the chord length parameter selection method is superior to the uniform method since it takes into account the geometry of the data. For example, in the case of the interpolation problem from Figure 5.2, the distorted data set with chord length parameters generates a curve that is indistinguishable from the true circle.

5.6 Hermite Interpolation

So far, we have discussed curve fitting to points only. Sometimes, one also knows tangent vectors, and a different kind of interpolation problem arises, known as *cubic Hermite interpolation.* We are now given two points $\mathbf{p}_0, \mathbf{p}_1$ and two tangent vectors $\mathbf{v}_0, \mathbf{v}_1$. The objective is to find a cubic polynomial curve $\mathbf{x}(t)$ that interpolates to these data:



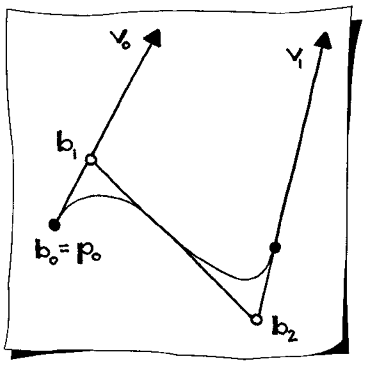
Sketch 37.
Cubic Hermite interpolation.

$$\mathbf{x}(0) = \mathbf{p}_0,$$
$$\dot{\mathbf{x}}(0) = \mathbf{v}_0,$$
$$\dot{\mathbf{x}}(1) = \mathbf{v}_1,$$
$$\mathbf{x}(1) = \mathbf{p}_1.$$

This situation is shown in Sketch 5.6.

We will write $\mathbf{x}$ in cubic Bézier form, and therefore must determine four Bézier points $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$. Two of them are quickly determined:

$$\mathbf{b}_0 = \mathbf{p}_0, \quad \mathbf{b}_3 = \mathbf{p}_1.$$



Sketch 38.
Cubic Hermite interpolation and Bézier solution.

For the remaining two, we recall (from Section 3.3) the endpoint derivative for Bézier curves:

$$\dot{\mathbf{x}}(0) = 3\Delta\mathbf{b}_0, \quad \dot{\mathbf{x}}(1) = 3\Delta\mathbf{b}_2.$$

We can easily solve for $\mathbf{b}_1$ and $\mathbf{b}_2$:

$$\mathbf{b}_1 = \mathbf{p}_0 + \frac{1}{3}\mathbf{v}_0, \qquad \mathbf{b}_2 = \mathbf{p}_1 - \frac{1}{3}\mathbf{v}_1.$$

This solution is shown in Sketch 5.6.

It is possible to rewrite this result such that the given data appear *explicitly* in the equation for the interpolant. So far, our interpolant is in Bézier form:

$$\mathbf{x}(t) = \mathbf{p}_0 B_0^3(t) + (\mathbf{p}_0 + \frac{1}{3}\mathbf{v}_0) B_1^3(t) + (\mathbf{p}_1 - \frac{1}{3}\mathbf{v}_1) B_2^3(t) + \mathbf{p}_1 B_3^3(t).$$

We simply rearrange:

$$\mathbf{x}(t) = \mathbf{p}_0 H_0^3(t) + \mathbf{v}_0 H_1^3(t) + \mathbf{v}_1 H_2^3(t) + \mathbf{p}_1 H_3^3(t), \qquad (5.19)$$

where we have set

$$\begin{aligned} H_0^3(t) &= B_0^3(t) + B_1^3(t), \\ H_1^3(t) &= \frac{1}{3} B_1^3(t), \\ H_2^3(t) &= -\frac{1}{3} B_2^3(t), \\ H_3^3(t) &= B_2^3(t) + B_3^3(t). \end{aligned}$$

The H_i^3 are called "cubic Hermite polynomials." Analogous to the Lagrange polynomials, the Hermite polynomials are the *cardinal form* for the interpolant to data points and tangent vectors since the input data appear explicitly.

The length of the tangent vectors $\mathbf{v}_0$ and $\mathbf{v}_1$ are an important factor for the final curve's shape. This length is not very intuitive to a user; we thus recommend against the use of the Hermite form unless exact derivatives are known.

5.7 Exercises

1. Let four points be given by

$$\mathbf{p}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 3 \\ 9 \end{bmatrix}.$$

   Setting $t_i = i/3$, find the Bézier polygon of the interpolating cubic.

2. Using the same data as in Exercise 1, evaluate the interpolating cubic at $t = 0.5$ using Aitken's algorithm. (Be sure to

sketch the spans as well as the intermediate points!) Verify that you get the same answer by applying the de Casteljau algorithm to the Bézier curve from the previous problem.

3. Sketch the interpolant to the three points

$$\begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

   with parameters $t_i = (0, 1, 2)$. Use Aitken's algorithm to evaluate at $t = 0.5$ and $t = 1.5$ to guess the shape of the curve. Now repeat this exercise, but with parameters $t_i = (0, 0.25, 2)$. Explain the result.

4. Sketch (manually, if you like) the three quadratic Lagrange polynomials for $(t_0, t_1, t_2) = (0, 4, 5)$.

5. What are the Hermite forms of the three Bézier curves from Section 3.6?

6. The data points of Figure 5.3 are in the file `wing.dat` from the web site. Find and plot the least squares fit using degree three and seven curves with chord length parameters.