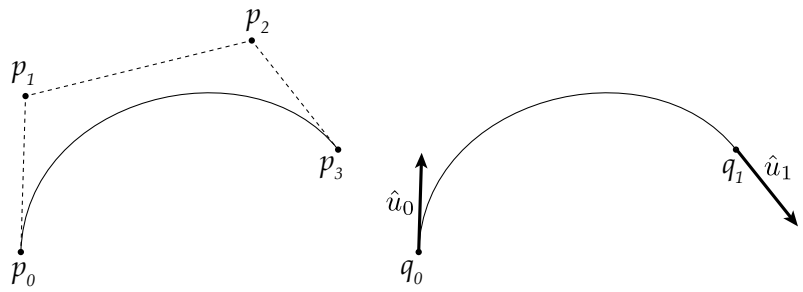
# Converting Between Bézier and Hermite Curves

CMU 15-462/662




Different file formats and software systems will use different internal representations for curves—in this exercise, you'll write some basic code for converting between such representations. Two common types of curves used in 2D graphics (*e.g.*, font rendering or illustration software) are *Bézier curves* and *Hermite splines*. A Bézier curve is specified by four control points; a Hermite curve is specified by two control points and two tangents. Actually, both of these curves are cubic polynomials—the only difference is that they're expressed with respect to different *bases*. In particular, a *cubic Bézier curve* is a linear combination of *cubic Bernstein bases* $B_i(t)$, where the control points $\mathbf{p}_i \in \mathbb{R}^n$ serve as constant coefficients:

$$\mathbf{p}(t) = \mathbf{p}_0 B_0(t) + \mathbf{p}_1 B_1(t) + \mathbf{p}_2 B_2(t) + \mathbf{p}_3 B_3(t), t \in [0,1].$$

The cubic Bernstein bases are given explicitly by

$$\begin{aligned} B_0(t) &= (1-t)^3, \\ B_1(t) &= 3(1-t)^2 t, \\ B_2(t) &= 3(1-t)t^2, \\ B_3(t) &= t^3. \end{aligned}$$

Similarly, a *cubic Hermite spline* is a linear combination

$$\mathbf{q}(t) = \mathbf{q}_0 H_{00}(t) + \mathbf{u}_0 H_{10}(t) + \mathbf{q}_1 H_{01}(t) + \mathbf{u}_1 H_{11}(t), t \in [0,1],$$

where $\mathbf{q}_0, \mathbf{q}_1 \in \mathbb{R}^n$ are the endpoints of the curve, $\mathbf{u}_0, \mathbf{u}_1 \in \mathbb{R}^n$ are the tangents at the two endpoints, and $H_i(t)$ are the *cubic Hermite bases*

$$\begin{aligned} H_{00}(t) &= (1+2t)(1-t)^2, \\ H_{10}(t) &= t(1-t)^2, \\ H_{01}(t) &= t^2(3-2t), \\ H_{11}(t) &= -t^2(1-t). \end{aligned}$$

Your task is to implement simple routines (prototyped below) that convert a cubic Bézier to a cubic Hermite curve and vice versa. Rather than hold your hand every step of the way, the purpose of this exercise is to give you some exposure to what it's "really" like to do graphics: you have a high-level task, and you have to figure out how to break it down, formulate the right equations, solve those equations, and turn the results into code. This kind of activity is really the bread and butter of doing computer graphics. We will however give you one small hint: *change of basis!* :-)

To get full credit you will need not only to produce correct code, but also to show your work and explain *how you got the expressions you've put in your code!*

```
// given coefficients for a cubic curve in Bezier form, computes the
// coefficients for an equivalent curve in Hermite form
void BezierToHermite( Vector  p0, Vector  p1, Vector  p2, Vector  p3,
                      Vector& q0, Vector& q1, Vector& u0, Vector& u1 )
{









}
```

```
// given coefficients for a cubic curve in Hermite form, computes the
// coefficients for an equivalent curve in Bezier form
void HermiteToBezier( Vector  q0, Vector  q1, Vector  u0, Vector  u1,
                      Vector& p0, Vector& p1, Vector& p2, Vector& p3 )
{









}
```