Chapter 3

Polynomial Evaluation and Basis Conversion

3.1 Horner's Algorithm in Power Basis

Horner's algorithm can evaluate a power basis polynomial

$$y(t) = \sum_{i=0}^{n} p_i t^i$$

using n multiplies and adds. The trick is to rewrite the polynomial in nested form:

$$y(t) = (\cdots((p_n t + p_{n-1})t + p_{n-2})t + \cdots + p_1)t + p_0$$

This can be written

$$\begin{aligned} h_n &= p_n \\ h_{i-1} &= t \cdot h_i + p_{i-1}, \quad i = n, \ldots, 1 \\ y(t) &= h_0 \end{aligned}$$

or, as a C funtion,

```
float p_horner(float *p,int n, float t)
{
float h;
int i;

   h = p[n];
   for(i=n-1; i>=0; i--)
      h = t*h + p[i];
   return  h;
}
```

This is the fastest algorithm for evaluating a power basis polynomial at a single point.

For evaluating a polynomial at several evenly spaced intervals ($f(0)$, $f(a)$, $f(2a)$, $f(3a)$, etc.) forward differencing works faster (see Chapter 4) but suffers from numerical instability.

3.2 Horner's Algorithm in Bernstein Basis

We have seen that one way to evaluate a Bézier curve (that is, to compute the Cartesian coordinates of a point on a Bézier curve) is to use the de Casteljau algorithm. Another option is to convert it to power basis (see Section 3.3) and then use Horner's algorithm in Section **??**. Alternatively, Horner's algorithm can be modified to work directly on Bernstein polynomials. We can use the recurrence relation for binomial coefficients

$$\binom{n}{i} = \frac{n-i+1}{i}\binom{n}{i-1}$$

to derive a nested multiplication which does not involve the binomial coefficients directly:

```
float b_horner(float *b, int n, float t)
{
float u, bc, tn, tmp;
int i;

   u = 1.0 - t;
   bc = 1;
   tn = 1;
   tmp = b[0]*u;
   for(i=1; i<n; i++){
      tn = tn*t;
      bc = bc*(n-i+1)/i;
      tmp = (tmp + tn*bc*b[i])*u;
   }
    return (tmp + tn*t*b[n]);
}
```

3.3 Basis Conversion

A fundamental problem in CAGD is that of polynomial basis conversion. We here consider the common case of conversion between power and Bernstein bases. For example, given a Bézier curve of any degree, we want to find the equivalent power basis equation and vice versa.

Denote a polynomial in Bernstein form

$$B(t) = \sum_{i=0}^{n} b_i \binom{n}{i} (1-t)^{n-i} t^i$$

and a polynomial in power form

$$P(t) = \sum_{i=0}^{n} p_i t^i.$$

The problem we consider is, given the b_i of a polynomial in Bernstein basis, find the p_i for the equivalent power basis polynomial (Bernstein to power basis conversion) or given the p_i find the b_i (power to Bernstein conversion).

An elegant solution to this problem can be obtained by performing a Taylor's series expansion. The Taylor's expansion of $B(t)$ is:

$$B(t) \equiv B(0) + B'(0)t + \frac{B''(0)t^2}{2} + \ldots + \frac{B^{(i)}(0)t^i}{i} + \ldots.$$

A power basis polynomial is equivalent to a Bernstein basis polynomial ($P(t) \equiv B(t)$) if and only if

$$\frac{P^{(i)}(0)t^i}{i!} \equiv \frac{B^{(i)}(0)t^i}{i!}, \quad i = 0, \ldots, n.$$

But, for power basis,

$$\frac{P^{(i)}(0)}{i!} = p_i$$

so

$$p_i = \frac{B^{(i)}(0)}{i!}, \ i = 0, \ldots, n. \tag{3.1}$$

The terms $B^{(i)}(0)$ can be found by setting up a difference table. Remember from our study of hodographs that the coefficients of the derivative of a polynomial in Bernstein form (for example, the x or y component of a Bézier curve) are:

$$n(b_1 - b_0), \quad n(b_2 - b_1), \ldots, n(b_n - b_{n-1}).$$

The coefficients of the second derivative are:

$$n(n-1)(b_2 - 2b_1 + b_0), \quad n(n-1)(b_3 - 2b_2 + b_1), \ldots, n(n-1)(b_n - 2b_{n-1} + b_{n-2}).$$

Since $B(0) = \sum_{i=0}^{n} b_i \binom{n}{i}(1-0)^{n-i}0^i = b_0$, we have

$$B'(0) = n(b_1 - b_0), \quad B''(0) = n(n-1)(b_2 - 2b_1 + b_0)$$

$$B^{(i)}(0) = n(n-1)\cdots(n-i+1)\sum_{j=0}^{i}(-1)^{(i-j+1)}\binom{i}{j}b_j.$$

This can be written more neatly if we define the recurrence

$$b_i^j = b_{i+1}^{j-1} - b_i^{j-1}$$

with $b_i^0 \equiv b_i$. Then

$$B^{(i)}(0) = n(n-1)\cdots(n-i+1)b_0^i = \frac{n!}{(n-i)!}b_0^i.$$

From equation 3.1,

$$p_i = \frac{n!}{(n-i)!i!}b_0^i = \binom{n}{i}b_0^i.$$

Thus, the problem reduces to one of finding the values b_0^i. This is easily done using a difference table:

$b_0^0 = b_0 = p_0$	$b_1^0 = b_1$	$\ldots$	$b_n^0 = b_n$
$b_0^1 = b_1^0 - b_0^0 = p_1/\binom{n}{1}$	$b_1^1 = b_2^0 - b_1^0$	$\ldots$	$b_n^0 = b_n^0 - b_{n-1}^0$
$b_0^2 = b_1^1 - b_0^1 = p_2/\binom{n}{2}$	$b_1^2 = b_2^1 - b_1^1$	$\ldots$	$b_n^2 = b_n^1 - b_{n-1}^1$
$\ldots$	$\ldots$	$\ldots$	$\ldots$
$b_0^{n-1} = b_1^{n-2} - b_0^{n-2} = p_{n-1}/\binom{n}{n-1}$	$b_1^{n-1} = b_2^{n-2} - b_1^{n-2}$		
$b_0^n = b_1^{n-1} - b_0^{n-1} = p_n$			

Thus, to perform Bernstein to power basis conversion, load the Bernstein coefficients into the top row and compute the difference table. Scale the left column by $\binom{n}{i}$, and you have the power coefficients.

To perform power to Bernstein conversion, divide the power coefficients by $\binom{n}{i}$, load them into the left column, compute the difference table backwards, and read the Bernstein coefficients off the top row.

3.3.1 Example

Convert to power basis the degree 4 Bernstein form polynomial with coefficients $(1, 3, 4, 6, 8)$. This is done by setting up the difference table

```
 1    3    4    6    8
 2    1    2    2
-1    1    0
 2   -1
-3
```

so the power coefficient are taken from the left column, times the binomial coefficients:

$$p_0 = 1\binom{4}{0} = 1$$

$$p_1 = 2\binom{4}{1} = 8$$

$$p_2 = -1\binom{4}{2} = -6$$

$$p_3 = 2\binom{4}{3} = 8$$

$$p_4 = -3\binom{4}{4} = -3$$

3.3.2 Closed Form Expression

The conversion from Bernstein to power basis can be written concisely as follows:

$$p_i = \sum_{k=0}^{i} b_k \binom{n}{i}\binom{i}{k}(-1)^{i-k}.$$