Chapter 4

Forward Differencing

Horner's algorithm is the fastest method for evaluating a polynomial at a single point. For a degree n polynomial, it requires n multiplies and n adds.

If a polynomial is to be evaluated at several evenly spaced values $t, t+\delta, t+2\delta, \ldots, t+k\delta$, the fastest method is to use *forward differences.*

Consider a degree 1 polynomial

$$f(t) = a_0 + a_1 t.$$

The difference between two adjacent function values is

$$\Delta_1(t) = f(t+\delta) - f(t) = [a_0 + a_1(t+\delta)] - [a_0 + a_1 t] = a_1\delta.$$

Thus, $f(t)$ can be evaluated at several evenly spaced points using the algorithm:

$\Delta_1 = a_1\delta$

$t_0 = 0$

$f(0) = a_0$

for $i = 1$ **to** k **do**

$t_i = t_{i-1} + \delta$

$f(t_i) = f(t_{i-1}) + \Delta_1$

endfor

Thus, each successive evaluation requires only one add, as opposed to one add and one multiply for Horner's algorithm.

This idea extends to polynomials of any degree. For the quadratic case,

$$f(t) = a_0 + a_1 t + a_2 t^2.$$

The difference between two adjacent function values is

$$\Delta_1(t) = f(t+\delta) - f(t) = [a_0 + a_1(t+\delta) + a_2(t+\delta)^2] - [a_0 + a_1 t + a_2 t^2]$$

$$\Delta_1(t) = a_1\delta + a_2\delta^2 + 2a_2 t\delta.$$

We can now write

$t_0 = 0$

$f(0) = a_0$

for $i = 1$ **to** k **do**

$t_i = t_{i-1} + \delta$

$\Delta_1(t_i) = a_1\delta + a_2\delta^2 + 2a_2 t_{i-1}\delta$

$f(t_i) = f(t_{i-1}) + \Delta_1(t_{i-1})$

endfor

In this case, $\Delta(t)$ is a linear polynomial, so we can evaluate it as above, by defining

$$\Delta_2(t) = \Delta_1(t + \delta) - \Delta_1(t) = 2a_2\delta^2$$

and our algorithm now becomes

$t_0 = 0$

$f(0) = a_0$

$\Delta_1 = a_1\delta + a_2\delta^2$

$\Delta_2 = 2a_2\delta^2$

for $i = 1$ **to** k **do**

$t_i = t_{i-1} + \delta$

$f(t_i) = f(t_{i-1}) + \Delta_1$

$\Delta_1 = \Delta_1 + \Delta_2$

endfor

It should be clear that for a degree n polynomial, each successive evaluation requires n adds and no multiplies! For a cubic polynomial

$$f(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3,$$

the forward difference algorithm becomes

$t_0 = 0$

$f(0) = a_0$

$\Delta_1 = a_1\delta + a_2\delta^2 + a_3\delta^3$

$\Delta_2 = 2a_2\delta^2 + 6a_3\delta^3$

$\Delta_3 = 6a_3\delta^3$

for $i = 1$ **to** k **do**

$t_i = t_{i-1} + \delta$

$f(t_i) = f(t_{i-1}) + \Delta_1$

$\Delta_1 = \Delta_1 + \Delta_2$

$\Delta_2 = \Delta_2 + \Delta_3$

endfor

Several questions remain. First, what are the initial values for the Δ_i if we want to start at some value other than $t = 0$. Second, what is a general equation for the Δ_i for a general degree n polynomial $f(t)$. Also, what if our polynomial is not in power basis.

These questions can be answered almost trivially by observing the following. Since $t_{i+1} = t_i + \delta$, we have

$$\Delta_1(t_i) = f(t_{i+1}) - f(t);$$
$$\Delta_j(t_i) = \Delta_{j-1}(t_{i+1}) - \Delta_{j-1}(t_i), \quad j = 2, \ldots, n;$$
$$\Delta_n(t_i) = \Delta_n(t_{i+1}) = \Delta_n(t_{i+k}) = \text{a constant}$$
$$\Delta_{n+1} = 0$$

Thus, our initial values for $\Delta_j(t_i)$ can be found by simply computing $f(t_i),\ f(t_{i+1}), \ldots, f(t_{i+n})$ and from them computing the initial differences. This lends itself nicely to a table. Here is the table for a degree four case:

$f(t_i)$	$f(t_{i+1})$	$f(t_{i+2})$	$f(t_{i+3})$	$f(t_{i+4})$
$\Delta_1(t_i)$	$\Delta_1(t_{i+1})$	$\Delta_1(t_{i+2})$	$\Delta_1(t_{i+3})$	
$\Delta_2(t_i)$	$\Delta_2(t_{i+1})$	$\Delta_2(t_{i+2})$		
$\Delta_3(t_i)$	$\Delta_3(t_{i+1})$			
$\Delta_4(t_i)$				
0	0	0	0	0

To compute $f(t_{i+5})$, we simply note that every number R in this table, along with its right hand neighbor R_{right} and the number directly beneath it R_{down} obey the rule

$$R_{right} = R + R_{down}.$$

Thus, we can simply fill in the values

$$\Delta_4(t_{i+1}) = \Delta_4(t_i) + 0$$
$$\Delta_3(t_{i+2}) = \Delta_3(t_{i+1}) + \Delta_4(t_{i+1})$$
$$\Delta_2(t_{i+3}) = \Delta_2(t_{i+2}) + \Delta_3(t_{i+2})$$
$$\Delta_1(t_{i+4}) = \Delta_1(t_{i+3}) + \Delta_2(t_{i+3})$$
$$f(t_{i+5}) = f(t_{i+4}) + \Delta_1(t_{i+4})$$

Note that this technique is independent of the basis in which $f(t)$ is defined. Thus, even if it is defined in Bernstein basis, all we need to do is to evaluate it $n+1$ times to initiate the forward differencing.

For example, consider the degree 4 polynomial for which $f(t_i) = 1$, $f(t_{i+1}) = 3$, $f(t_{i+2}) = 2$, $f(t_{i+3}) = 5$, $f(t_{i+4}) = 4$. We can compute $f(t_{i+5}) = -24$, $f(t_{i+6}) = -117$, and $f(t_{i+7}) = -328$ from the following difference table:

$t:$	t_i	t_{i+1}	t_{i+2}	t_{i+3}	t_{i+4}	t_{i+5}	t_{i+6}	t_{i+7}
$f(t):$	1	3	2	5	4	-24	-117	-328
$\Delta_1(t):$	2	-1	3	-1	-28	-93	-211	
$\Delta_2(t):$	-3	4	-4	-27	-65	-118		
$\Delta_3(t):$	7	-8	-23	-38	-53			
$\Delta_4(t):$	-15	-15	-15	-15	-15			
$\Delta_5(t):$	0	0	0	0	0	0	0	0

Example
For a certain cubic polynomial $f(t)$, we have:

$$f(1) = 1; \quad f(2) = 2; \quad f(3) = 4; \quad f(5) = 15.$$

Solve for $f(4)$ using forward differencing.

Solution

t :	1	2	3	4	5
$f(t)$:	1	2	4	x	15
$\Delta_1(t)$:	1	2	$x - 4$	$15 - x$	
$\Delta_2(t)$:	1	$x - 6$	$19 - 2x$		
$\Delta_3(t)$:	$x - 7$	$25 - 3x$			
$\Delta_4(t)$:	$32 - 4x$				

For a cubic polynomial, $\Delta_4(t) = 0$, so $32 - 4x = 0$ and $x = 8$.

4.1 Choosing δ

This raises the important question of how to select an appropriate value for δ when using forward differencing to plot a curve. One way to determine δ is so that distance from the the curve to its polygonal approximation lies within a tolerance. A discussion of how to chose δ that will satisfy such a requirement is found in Section 10.6.

A second criterion that might be used to choose δ arises in the problem of rasterizing a curve. This means to "turn on" a contiguous series of pixels that the curve passes through, without any gaps. If the control points of the degree n Bézier curve are $\mathbf{P}_i = (x_i, y_i)$ in *pixel coordinates*, then let

$$d = \max\{x_i - x_{i-1}, y_i - y_{i-1}\} \quad i = 1, \ldots, n$$

If you now evaluate the curve at $n * d + 1$ evenly spaced values of t and paint each resulting pixel, there will be no gaps in the drawing of the curve. Another way to say this, compute

$$\delta = \frac{1}{n * d}.$$

Then, compute the points $\mathbf{P}(i * \delta)$, $i = 0 \ldots, n * d$. Note that $n * d$ is an upper bound on the magnitude of the x or y component of the first derivative of the curve.