

## Dyadic Interpolation Schemes

Mike Day, January 2008

Given a set of equally-spaced samples of some function, we'd like to be able to make educated guesses for the values which the function would take on at points located half-way between the samples. If we had a scheme for doing this, we could then apply it recursively to generate the values at the  $1/4$  and  $3/4$  points; then the  $1/8$ ,  $3/8$ ,  $5/8$  and  $7/8$  points, and so on. Points of the form  $n/2^m$  are called *dyadic points*, so an interpolation process of this type is called a *dyadic interpolation scheme*.

In many applications we'd prefer to keep the scheme efficient by only examining points in some small neighbourhood in order to reconstruct any given point. One such method is to simply take the average of the two neighbouring points, but of course the quality of reconstruction won't be all that good – we'll just get a piecewise-linear interpolant. By using a little more information from the samples, can we improve on this?

### Deslauriers-Dubuc Interpolation

One logical way to generalize this averaging process is to consider a larger neighbourhood of points, symmetrically balanced around the point of interest, and fit an interpolating polynomial through these points. By sampling this polynomial at the midpoint, we obtain the reconstructed value. The more points we span, the higher the degree of the polynomial, and in some sense the better the quality of the reconstruction – 2 points define a straight line, i.e. a first-degree polynomial, 4 points a cubic polynomial, 6 points a polynomial of degree 5, and so on. In general, given  $2n$  points we construct an interpolating polynomial of degree  $2n-1$  and sample its midpoint.

We can derive an expression in terms of the sample values for the interpolant of any degree (e.g. the Lagrange form), but since we're always going to be sampling it at the same point for any given degree – the centre of the neighbourhood of samples – we can precompute a set of fixed weights to be applied to the sample values. That is, to find the interpolated midpoint, take a linear combination of the neighbouring values using precomputed weights. This process is called *Deslauriers-Dubuc interpolation* (sometimes abbreviated to 'DD-interpolation').

Here are the coefficients for the first few DD-interpolants. Rows correspond to interpolants of degree 1, 3, 5, 7, 9 respectively. The first row is self-evident – for the linear interpolant just average the 2 neighbours. Each successive row takes in one more neighbour on either side.

				$\frac{1}{2}$	$\frac{1}{2}$				
			$-\frac{1}{16}$	$\frac{9}{16}$	$\frac{9}{16}$	$-\frac{1}{16}$			
		$\frac{3}{256}$	$-\frac{25}{256}$	$\frac{150}{256}$	$\frac{150}{256}$	$-\frac{25}{256}$	$\frac{3}{256}$		
	$-\frac{5}{2048}$	$\frac{49}{2048}$	$-\frac{245}{2048}$	$\frac{1225}{2048}$	$\frac{1225}{2048}$	$-\frac{245}{2048}$	$\frac{49}{2048}$	$-\frac{5}{2048}$	
$\frac{35}{65536}$	$-\frac{405}{65536}$	$\frac{2268}{65536}$	$-\frac{8820}{65536}$	$\frac{39690}{65536}$	$\frac{39690}{65536}$	$-\frac{8820}{65536}$	$\frac{2268}{65536}$	$-\frac{405}{65536}$	$\frac{35}{65536}$

As with many kinds of subdivision, the limiting curve generated by repeating the subdivision ad infinitum will not generally be a polynomial (except in cases where the samples themselves lie on a polynomial of degree  $2n-1$ , where  $2n$  is the width of the filter), but it will satisfy some nice smoothness properties. Especially useful is the property that as the filter width increases, the resulting curve tends towards the sum of sinusoids constituting the DFT of the samples.

The form of a general entry in this table, denoting the row by  $n$  and the column by  $k$  (counting from 0 in each case, with the column number increasing from the centre line of the table) is:

$$f(n, k) = \frac{(-1)^k ((2n+1)!)^2}{(2k+1)2^{4n+1} (n!)^2 (n-k)!(n+k+1)!}$$

Using Stirling's approximation, we find that as  $n \rightarrow \infty$ ,  $f(n, k) \rightarrow \frac{2(-1)^k}{(2k+1)\pi}$ .

So in the limiting case, the row of coefficients tends towards this pattern:

$$\frac{2}{\pi} \left( \dots + \frac{1}{9} - \frac{1}{7} + \frac{1}{5} - \frac{1}{3} + 1 + 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \dots \right)$$

which corresponds to a sinc filter, the perfect reconstruction filter.

### Derivatives

Once we've carried out the DD-interpolation scheme to the required level of refinement, or indeed given a set of equally spaced samples produced by any method, we might wish to generate the derivative values for the sampled function.

It's easy to develop a similar scheme to approximate the gradient at a given point, based on the values at neighbouring points. Note that this is a slightly different kind of process – we're assuming we already have all the points of interest and don't need to generate any intermediates. We just want to guess at the derivative for each existing point. (It would be perfectly possible to mimic the DD-interpolation scheme exactly, in order to generate derivatives at the first level of dyadic points, but that approach is a little less useful.)

The lowest-quality derivative is just the 'central difference', half the difference of the neighbour on the right minus the neighbour on the left (assuming the step size to be 1, otherwise also divide by the step size). This corresponds to the gradient at the midpoint of the quadratic interpolant through the central point and its 2 neighbours.

But of course we can evaluate the gradient for any interpolant of even degree in the same manner, just by taking in more neighbours. Here are the coefficients for the first few interpolants. Rows correspond to the derivatives of interpolants of degree 2, 4, 6, 8, 10 respectively. Note that the central point itself plays no part in the computed value for any degree.

				$-\frac{1}{2}$	0	$\frac{1}{2}$				
			$\frac{1}{12}$	$-\frac{8}{12}$	0	$\frac{8}{12}$	$-\frac{1}{12}$			
		$-\frac{1}{60}$	$\frac{9}{60}$	$-\frac{45}{60}$	0	$\frac{45}{60}$	$-\frac{9}{60}$	$\frac{1}{60}$		
	$\frac{3}{840}$	$-\frac{32}{840}$	$\frac{168}{840}$	$-\frac{672}{840}$	0	$\frac{672}{840}$	$-\frac{168}{840}$	$\frac{32}{840}$	$-\frac{3}{840}$	
$-\frac{2}{2520}$	$\frac{25}{2520}$	$-\frac{150}{2520}$	$\frac{600}{2520}$	$-\frac{2100}{2520}$	0	$\frac{2100}{2520}$	$-\frac{600}{2520}$	$\frac{150}{2520}$	$-\frac{25}{2520}$	$\frac{2}{2520}$

The table is antisymmetric, and the form of a general right-side entry in the table, denoting the row by n (starting from n=1 for the top row) and the column by k (counting right from the centre line, with k=1 corresponding to the first non-zero entry) is:

$$g(n, k) = \frac{(-1)^{k+1} (n!)^2}{k(n-k)!(n+k)!}$$

Using Stirling's approximation, we find that as  $n \rightarrow \infty$ ,  $g(n, k) \rightarrow \frac{(-1)^{k+1}}{k}$ .

So in the limiting case, the row of coefficients tends towards this pattern:

$$\dots -\frac{1}{5} + \frac{1}{4} - \frac{1}{3} + \frac{1}{2} - 1 \quad 0 \quad + 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} \dots$$

### The Hermite Form

It may frequently arise that we need to reconstruct both the function and its derivative, based on a set of equally-spaced samples of both. In this case it may make more sense to work with the Hermite form of the interpolating polynomial. In this form, given the  $2n$  points surrounding the point under construction, we have values for the function at each point, but also values for its derivative, a total of  $4n$  constraints. This allows us to construct an interpolating polynomial of degree  $4n-1$ .

The simplest case corresponds to examining one point each side. We can fit a cubic polynomial through these 2 points in such a way that both the 2 function values and the 2 derivative values are simultaneously satisfied. If we expand our neighbourhood to 4 points, the interpolant jumps to degree 7. If instead we used traditional DD-interpolation, we would require a neighbourhood containing 8 points to yield a degree 7 interpolant. Although the same number of values (8) is used in both cases, the Hermite form offers the advantage that we need only consider a neighbourhood of 4 points rather than 8. (On the other hand, there are cases where this property actually becomes a disadvantage too.)

For want of a better name, I will dub this process ‘HD-interpolation’.

Here are the coefficients for the first two HD-interpolants. The first row considers 2 neighbours and generates a cubically interpolated midpoint. The second row considers 4 neighbours and generates the midpoint on a 7<sup>th</sup>-degree interpolating polynomial. The table entries are now  $2 \times 2$  matrices, because for each sampled point we have a vector of 2 values (function and derivative), and for each constructed point we output a similar vector of 2 values.

$$\frac{1}{8} \begin{pmatrix} 4 & -12 \\ 1 & -2 \end{pmatrix} \quad \frac{1}{8} \begin{pmatrix} 4 & 12 \\ -1 & -2 \end{pmatrix}$$

$$\frac{1}{512} \begin{pmatrix} 13 & -10 \\ 3 & -2 \end{pmatrix} \quad \frac{1}{512} \begin{pmatrix} 243 & -810 \\ 81 & -162 \end{pmatrix} \quad \frac{1}{512} \begin{pmatrix} 243 & 810 \\ -81 & -162 \end{pmatrix} \quad \frac{1}{512} \begin{pmatrix} 13 & 10 \\ -3 & -2 \end{pmatrix}$$

So, for example, the first row would be applied as follows. Given points A and B with function/derivative pairs  $(f_A, g_A)$  and  $(f_B, g_B)$  respectively, such that each  $g$  value equals the sampled derivative multiplied by the distance between samples, then the corresponding pair  $(f_C, g_C)$  at the midpoint C is given by  $(f_C, g_C) = (f_A, g_A) * M_A + (f_B, g_B) * M_B$ , where  $M_A$  and

$M_B$  are the two matrices from the first row of the table. (Notice that the resulting  $g_C$  value represents the reconstructed derivative multiplied by the *new* distance between samples, which is halved compared to the previous level of iteration.)

In every case though, the interpolated function value is just a linear combination of the neighbouring function values and derivative values, and the interpolated derivative value is just another such linear combination.

Note again the symmetry and antisymmetry present in the table. If we wished to implement this method using the fewest arithmetic operations, we would probably begin by taking sums of opposing function samples and differences of opposing derivative samples, and construct the weighted sums from these intermediates. Further rows could be computed using Mathematica if necessary. I have yet to derive the limiting form for these coefficients.

The water systems in *Resistance: Fall of Man*, and *Ratchet & Clank Future: Tools of Destruction* employed a method which used the first row of the HD-interpolation table. Much better reconstruction quality could probably be achieved using either the second row of this table, or one of the rows of the DD-interpolation table corresponding to a higher degree than the 3<sup>rd</sup>.