INTERPOLATION, EXTRAPOLATION, DIFFERENTIATION AND ALL THAT

Neal Katz

References:

Numerical Methods, Dahlquist, Bjorck & Anderson, Chap. 4

Introduction to Numerical Analysis, Stoer & Bulirsch, Chap. 2

Numerical Methods, Hornbeck, Chaps. 3 & 4

Numerical Methods that Work, Acton, Chap. 3

Numerical Recipes, Press et al, Chap. 3

Interpolation in 1-D

- Traditionally very important because of wide use of tables. Still have tables to make codes more efficient and tables of input and output data. Also the techniques are used in other numerical algorithms, e.g. numerical integration.

- Will discuss some common schemes including: **polynomial** interpolation of which **linear** interpolation is a special case, **rational function** interpolation and **cubic spline** interpolation.

General Problem:

Take a family of functions: $f(x_i; a_0, a_1, \ldots a_n)$.

[Family in the sense that $\{a_0, \ldots, a_n\}$ parametrize the "family" that has the functional form of r.]

Interpolation problem:

Find the parameters $\{a_j\}$ so that for $(n+1)$ pairs of numbers (x_i, f_i):

$$f(x_i; a_j) = f_i.$$

Terminology:

(x_i, f_i) are called support points. Sometimes $f_i^{[n]}$ are known and may/should be used.

- Most widely used families are linear in the parameters:

$$f(x; \{a_j\}) = a_o f_o(x) + \ldots + a_n f_n(x).$$

For example: **Polynomial Interpolation:**

$$f(x; \{a_j\}) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n.$$

Polynomial Interpolation

- Often used, easy to understand and analyze.

Theorem: Given $(n+1)$ support points, $\exists$ a unique nth order polynomial such that $P(x_i) = f_i$.

This allows us to easily construct the polynomial for interpolation and know it is unique as follows:

- Suppose we had $L_i(x_k) = \delta_{ik}$ where the L_i are nth order polynomials.

- By uniqueness

$$f(x) = \sum_{i=0}^{n} f_i L_i(x).$$

- Need to find the polynomials L_i.

$$L_i(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}$$

or

$$f(x) = \sum_{i=0}^{n} f_i \sum_{k=0;k\neq i}^{n} \frac{x-x_k}{x_i-x_k}.$$

This is the **Lagrangian Interpolation** formula.

Linear Interpolation

- Simplest form of polynomial interpolation. Very easy, quick and useful.

- Just apply the Lagrange formula with $n = 2$.

$$y = Ay_i + By_{i+1}$$

where

$$A \equiv \frac{x_{i+1} - x}{x_{i+1} - x_i}$$

and

$$B \equiv 1 - A = \frac{x - x_i}{x_{i+1} - x_i}.$$

Polynomial Interpolation continued

- Could use the Lagrange formula directly but there are better algorithms.

  - **Neville's algorithm:** a recursive algorithm that uses small subsets of support points to build up higher order polynomials. Geared to only interpolating values not to getting a fitted polynomial.

  - **Newton's algorithm:** another recursive algorithm, derives a fitted polynomial.

Problems with polynomial interpolation:

- wiggles
- non–polynomial "looking" functions

- **Splines:** are "stiff" and make 1st and 2nd derivatives continuous.
- **Rational functions**: fit things with singularities.

Cubic Splines

- Idea comes from elasticity theory: bend an elastic rod around a set of points.

- $f(x)$ may be considered to be composed of 3rd order polynomials (cubics) such that the first two derivatives are continuous.

This, of course, can be formalized and generalized:

Let $(x_0, y_0) \ldots (x_n, y_n)$ be support points and $f(x)$ be a function such that

1) In each interval (x_{i-1}, x_i), $f(x)$ is a polynomial of degree s with $f(x_{i-1}) = y_{i-1}$ and $f(x_i) = y_i$.

2) The first $s-1$ derivatives are continuous across each interval.

- Let's stick to the cubic ones, i.e. $s = 3$, which have nice properties computationally.

Find the cubic functions by construction:

Let

$$h_i = x_i - x_{i-1},$$

$$d_i = \frac{y_i - y_{i-1}}{h_i}, \quad t = \frac{x - x_{i-1}}{h_i}$$

for one interval $[x_{i-1}, x_i]$, $i = 0, \ldots, n$.

Choose the cubic

$$q_i(x) = ty_i + (1-t)y_{i-1} + t(1-t)h_i[(k_{i-1} - d_i)(1-t) - (k_i - d_i)t]$$

where k_i are $m+1$ unknowns.

Now by construction

$$x = x_{i-1} \Leftrightarrow t = 0$$

and

$$x = x_i \Leftrightarrow t = 1$$

so

$$q_i(x_{i-1}) = y_{i-1} \text{ and } q_i(x_i) = y_i.$$

Since

$$\frac{dt}{dx} = \frac{1}{h_i}$$

then

$$q_i'(x) = \frac{y_i - y_{i-1}}{h_i} + (1-2t)[(k_{i-1}-d_i)(1-t)-(k_i-d_i)t] + t(1-t)[2d_i - k_i - k_{i-1}].$$

Therefore

$$q_i'(x_{i-1}) = d_i + k_{i-1} - d_i = k_{i-1}$$

and

$$q_i'(x_i) = d_i + (k_i - d_i) = k_i.$$

Hence the k's are first derivatives. If we replace i by $i+1$ everywhere it implies that

$$q_{i+1}'(x_i) = k_i$$

and

$$q_{i+1}'(x_i) = q_i'(x_i).$$

The first derivatives are continuous.

The continuity of second derivatives,

$$q_i''(x_i) = q_{i+1}''(x_i),$$

implies that

$$h_{i+1}k_{i-1} + 2(h_i + h_{i+1})k_i + h_i k_{i+1} = 3(h_i d_{i+1} + h_{i+1} d_i)$$

But there are only $m-1$ equations and $m+1$ unknowns because there are no conditions on the endpoints. There are three commonly used boundary conditions:

- Suppose one knows the first derivative at the endpoints then

$$\Rightarrow k_0, k_m$$

- "Natural boundary" from elasticity.

$$q_1''(x_0) = q_m''(x_m) = 0$$

  so the spline is a straight line beyond the interval

$$\begin{cases} 2k_o + k_1 = 3d_1 \\ k_{m-1} + 2k_m = 3d_m \end{cases}$$

- Set the third derivative to zero, i.e.

$$q_1''(x_0) = q_2''(x_1)$$

  and

$$q_{m-1}''(x_{m-1}) = q_m''(x_m).$$

  It is not often used but is useful for intrinsically smooth input.

Elastic boundary conditions give the following tridiagonal matrix:

$$
\begin{pmatrix}
2 & 1 & 0 & 0 & \dots & 0 \\
h_2 & 2(h_1+h_2) & h_1 & 0 & \dots & 0 \\
0 & h_3 & 2(h_2+h_3) & h_2 & & \vdots \\
\vdots & & \ddots & \ddots & \ddots & 0 \\
 & & & & & 0 \\
0 & \dots & & h_m & 2(h_{m-1}+h_m) & h_{m-1} \\
0 & \dots & & 0 & 1 & 2
\end{pmatrix}
\begin{pmatrix}
k_0 \\ k_1 \\ k_2 \\ k_3 \\ \vdots \\ k_{m-1} \\ k_m
\end{pmatrix}
$$

$$
= \begin{pmatrix}
3d_1 \\
3h_2 d_1 + 3h_1 d_2 \\
3h_3 d_2 + 3h_2 d_1 \\
\vdots \\
3h_m d_{m-1} + 3h_{m-1} d_m \\
3d_m
\end{pmatrix}
$$

Interpolation from a regular n-Dimensional grid

- **Nearest Grid Point Method (NGP):** take the value from the nearest grid point (involves only one grid value).

$$f(x) = \sum_i W(x - x_i) f(x_i)$$

where

$$W(x) = \begin{cases} 1 & |x| < h/2 \text{ or } x = h/2 \\ 0 & \text{otherwise} \end{cases}$$

- **Cloud In Cell (CIC):** use multilinear interpolation (involves 8 nearest grid points). In 3-d

$$\begin{aligned} f(x,y,z) =& (1-A_x)(1-A_y)(1-A_z)x_{i,j,k} + (1-A_x)(1-A_y)(A_z)x_{i,j,k+1} + \\ & (1-A_x)(A_y)(1-A_z)x_{i,j+1,k} + (1-A_x)(A_y)(A_z)x_{i,j+1,k+1} + \\ & (A_x)(1-A_y)(1-A_z)x_{i+1,j,k1} + (A_x)(1-A_y)(A_z)x_{i+1,j,k+1} + \\ & (A_x)(A_y)(1-A_z)x_{i+1,j+1,k} + (A_x)(A_y)(A_z)x_{i+1,j+1,k+1} \end{aligned}$$

where

$$A_x = \frac{x - x_i}{h} \quad , \quad A_y = \frac{y - y_i}{h} \quad \text{and} \quad A_z = \frac{z - z_i}{h}$$

and h is the grid spacing.

- **Triangular Shaped Cloud (TSC):** use multiquadratic interpolation (involves 27 nearest grid points).

Interpolation on a Disordered Point Set:

Spline Kernel Interpolation

- The integral interpolant of any function $f(r)$ is

$$< f(\mathbf{r}) >= \int W(\mathbf{r}-\mathbf{r}';h)f(\mathbf{r}')d\mathbf{r}'$$

  where the integration is over all space.

- h is the smoothing length and specifies the extent of the averaging volume.

- W is the smoothing kernel which must have two properties

$$\int W(\mathbf{r}-\mathbf{r}';h)d\mathbf{r} = 1$$

  and

$$\lim_{h\to 0} W(\mathbf{r};h) = \delta(\mathbf{r}-\mathbf{r}').$$

- If the values of $f(r)$ are only known at a finite number of points then

$$< f(\mathbf{r}) >= \sum_{j=1}^{N} \frac{f(\mathbf{r}_j)}{< n(\mathbf{r}_j) >} W(\mathbf{r}-\mathbf{r}_j;h).$$

- In particular if a mass m_j is associated with each particle then

$$< \rho(\mathbf{r}) >= \sum_{j=1}^{N} m_j W(\mathbf{r} - \mathbf{r}_j; h)$$

and

$$< f(\mathbf{r}) >= \sum_{j=1}^{N} m_j \frac{f_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j; h).$$

The Kernel

- $W(\mathbf{r})$ must be differentiable to at least the same order as that of the terms present in the dynamical equations.

- A common choice for the kernel is a Gaussian,

$$W(r,h) = \frac{1}{\pi^{3/2}h^3}e^{-(r^2/h^2)}.$$

- Another choice that has compact support is a spherically symmetric spline kernel

$$W(r,h) = \frac{1}{\pi h^3}\begin{cases} 1 - \frac{3}{2}\left(\frac{r}{h}\right)^2 + \frac{3}{4}\left(\frac{r}{h}\right)^3 & 0 \le \frac{r}{h} \le 1 \\ \frac{1}{4}\left(2 - \frac{r}{h}\right)^3 & 1 \le \frac{r}{h} \le 2 \\ 0 & \frac{r}{h} \ge 2 \end{cases} .$$

Accuracy and Errors

- If $W(\mathbf{r})$ is an even function and spherically symmetric, the interpolant errors are

$$< f(\mathbf{r}) >= f(\mathbf{r}) + c\frac{h^2}{6}\nabla^2 f + O(h^3)$$

where

$$c = \int u^2 h^3 W(u) d\mathbf{u}.$$

- Could choose $W(\mathbf{r})$ so that the second order term also vanishes. However, $W(\mathbf{r})$ would have to be negative over part of its range and this could introduce negative densities.

- $f(\mathbf{r})$ can always be replaced by its smoothed equivalent to within the order of accuracy of the smoothing process itself, *e.g.*

$$\left\langle \frac{A(\mathbf{r})}{B(\mathbf{r})} \right\rangle = \frac{< A(\mathbf{r}) >}{< B(\mathbf{r}) >} + O(h^2).$$

- In three dimensions the errors scale either as $n^{-1}(\log n)^2$ or as $n^{-1}\log n$ depending on assumptions made in the error analysis.

Varying Resolution

- To further increase the dynamic range it is also possible to allow h to vary spatially.

- If one adopts the scatter interpretation of SPH then

$$< f(\mathbf{r}) >= \int W(\mathbf{r}-\mathbf{r}', h(\mathbf{r}'))f(\mathbf{r}')d\mathbf{r}'$$

implying that the discrete estimate of the density is computed according to

$$< \rho(\mathbf{r}_i) >= \sum_{j=1}^{N} m_j W(r_{ij}, h_j).$$

- The estimates are still second order accurate. For the scatter interpretation

$$\begin{aligned} < f(\mathbf{r}) >= f(\mathbf{r}) + c_1 \frac{h^2(\mathbf{r})}{6}\nabla^2 f + c_2 \frac{\nabla h \cdot \nabla h}{6} f(\mathbf{r}) + \\ c_3 \frac{h(\mathbf{r})}{3} \nabla f \cdot \nabla h + c_3 \frac{h(\mathbf{r})\nabla^2 h}{6} f(\mathbf{r}) + O(h^3) \end{aligned}$$

where the constants

$$c_1 = h^{-2}\int_0^\infty 4\pi u^4 h^5 W(u) du,$$
$$c_2 = \int_0^\infty 4\pi u^4 h^5 \frac{\partial^2 W}{\partial h^2} du,$$
$$c_3 = h^{-1}\int_0^\infty 4\pi u^4 h^5 \frac{\partial W}{\partial h} du,$$

and assuming $W(u) \equiv W(h\mathbf{u}, h)$.

Differentiation on a grid: Finite Differences

- Often used for finding derivatives on regularly spaced data.

Consider the Taylor series expansion of a function $f(x)$:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \ldots$$

Solving for $f'(x)$ yields

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(x) - \frac{h^2}{6}f'''(x) + \ldots$$

or

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h).$$

Define

$$f(x+h) = f_{i+1} \quad \& \quad f(x) = f_j$$

and the first forward difference of f at i as

$$\Delta f_j \equiv f_{j+1} - f_j$$

then

$$f'(x) = \frac{\Delta f_j}{h} + O(h).$$

This is called the **first forward difference** approximation of error $O(h)$.

Defining the first backward difference of f at j as

$$\nabla f_j \equiv f_j - f_{j-1}$$

and using the Taylor series expansion to determine $f(x-h)$ one derives

$$f'(x) = \frac{\nabla f_j}{h} + O(h),$$

the **first backward difference** approximation of error $O(h)$.

Subtracting the Taylor series expansions for $f(x+h)$ and $f(x-h)$ gives

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{h^3}{3} f'''(x) + \ldots$$

Solving for $f'(x)$ gives

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

or

$$f'(x) = \frac{f_{j+1} - f_{j-1}}{2h} + O(h^2).$$

This is called the **central difference** representation and is accurate to $O(h^2)$.

Similar techniques can be used to derive expression for higher order derivatives. For example

$$f''(x) = \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2} + O(h^2).$$

Differentiation on a Disordered Point Set:

Spline Kernel Interpolation

- Gradients are calculated in a similar manner to normal interpolation.

  By definition

$$< \nabla f(\mathbf{r}) >= \int W(\mathbf{r} - \mathbf{r}'; h) \nabla f(\mathbf{r}') d\mathbf{r}'.$$

  Integrating by parts yields

$$< \nabla f(\mathbf{r}) >= \int f(\mathbf{r}') \nabla W(\mathbf{r} - \mathbf{r}'; h) d\mathbf{r}'.$$

- For a finite number of particles with mass m_j

$$< (\nabla f)_i >= \sum_{j=1}^{N} m_j \frac{f_j}{\rho_j} \nabla W(\mathbf{r} - \mathbf{r}_j; h).$$

- Higher accuracy is obtained by using the relation $\rho \nabla f = \nabla(\rho f) - f \nabla \rho$

  giving

$$< (\nabla f)_i >= \frac{1}{\rho_i} \sum_{j=1}^{N} m_j (f_j - f_i) \nabla W(\mathbf{r} - \mathbf{r}_j; h).$$