Globally C^s and Locally Controllable Interpolation on n-Dimensional Lattices

David Eberly, Geometric Tools, Redmond WA 98052
https://www.geometrictools.com/

Created: February 21, 2020

Contents

1 Introduction

This article describes an extension of *Hermite spline interpolation* to n-dimensional lattices, which provides us with a globally C^s polynomial function that has local control.

Consider an interval $[0,1]$ for which function values and derivatives through order s are known at the endpoints $x=0$ and $x=1$. There are $2s+2$ known values, allowing an interpolating polynomial of degree $d=2s+1$ that matches the function values and derivatives at the endpoints. Let the original function be $f(x)$ with i-th order derivative $f^{(i)}(x)$. The values $f^{(i)}(0)$ and $f^{(i)}(1)$ for $0 \le i \le s$ are known. If $g(x)$ is the interpolating polynomial, say,

$$g(x) = \sum_{j=0}^{d} c_j (1-x)^{d-j} x^j \tag{1}$$

a linear system for the unknown values c_j can be formulated. Specifically, the $2s+2$ linear equations are given by

$$g^{(i)}(0) = f^{(i)}(0), \;\; g^{(i)}(1) = f^{(i)}(1), \;\; 0 \le i \le s \tag{2}$$

For example, the simple case $s=0$, $g(x) = c_0(1-x) + c_1 x$ and the linear system has equations

$$f(0) = g(0) = c_0, \;\; f(1) = g(1) = c_1 \tag{3}$$

in which case

$$g(x) = f(0)(1-x) + f(1)x = \begin{bmatrix} f(0) & f(1) \end{bmatrix} \begin{bmatrix} 1-x \\ x \end{bmatrix} \tag{4}$$

Linear polynomials are continuous. The global aspect involves having multiple intervals, say, $[0,1]$ and $[1,2]$. Knowing $f(1)$ and $f(2)$, an interpolating polynomial on the second interval is $h(x) = f(1)[1-(x-1)] + f(2)(x-1)$. The function

$$P(x) = \begin{cases} g(x), & x \in [0,1] \\ h(x), & x \in [1,2] \end{cases} \tag{5}$$

is continuous at $x=1$ because $\lim_{x \to 1^-} P(x) = g(1) = f(1) = h(1) = \lim_{x \to 1^+} P(x)$.

Now consider the case $s=1$. The values $f(0)$, $f'(0)$, $f(1)$ and $f'(1)$ are known. Define $g(x) = c_0(1-x)^3 + c_1 x(1-x)^2 + c_2 x^2(1-x) + c_3 x^3$. The coefficients of g to force the matching at the endpoints are obtained by solving a linear system of 4 equations and leads to

$$g(x) = \begin{bmatrix} f(0) & 3f(0)+f'(0) & 3f(1)-f'(1) & f(1) \end{bmatrix} \begin{bmatrix} (1-x)^3 \\ x(1-x)^2 \\ x^2(1-x) \\ x^3 \end{bmatrix} \tag{6}$$

The interpolation is *locally controllable* in the sense that the first half of the 1×4 matrix of coefficients in the equation for $g(x)$ are determined solely by the known values at $x=0$ and the second half of the coefficients are determined solely by the known values at $x=1$.

The cubic polynomial is continuous. The global aspect involves having multiple intervals, say, $[0,1]$ and $[1,2]$. Knowing $f(2)$ and $f'(2)$, an interpolating polynomial on the second interval is

$$h(x) = \begin{bmatrix} f(1) & 3f(1)+f'(1) & 3f(2)-f'(2) & f(2) \end{bmatrix} \begin{bmatrix} (1-(x-1))^3 \\ (x-1)(1-(x-1))^2 \\ (x-1)^2(1-(x-1)) \\ (x-1)^3 \end{bmatrix} \tag{7}$$

Observe that $g(1) = h(1) = f(1)$ and $g'(1) = h'(1) = f'(1)$, so the piecewise function defined by $g(x)$ and $h(x)$ has continuous derivatives at $x = 1$; the piecewise function is therefore globally C^1.

Generally, the polynomial of equation (1) with constraints of equation (2) imply that c_0 through c_s are determined solely by $f^{(i)}(0)$ and c_{s+1} through c_{2s+1} are determined solely by $f^{(i)}(1)$. Thus, the interpolation is locally controllable. The matching of function and derivatives at endpoints ensures a globally C^s interpolation.

2 Interpolation for 2D Lattices

The extension of Hermite spline interpolation is presented here for the case $n = 2$. Global continuity of s-th order derivatives is a direct consequence of the fitting, similar to the case $n = 1$. Cnsider a function $f(x, y)$ defined on a square $[0,1]^2$ that is to be fitted with a polynomial $g(x, y)$ of degree $d = 2k+1$ in each variable.

2.1 Bilinear Interpolation

The case $s = 0$ is that of bilinear interpolation. The values $f(0,0)$, $f(1,0)$, $f(0,1)$ and $f(1,1)$ are know. The function is to be fitted with a bilinear polynomial

$$g(x,y) = \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix} \tag{8}$$

The trivial solution for the coefficients is $c_{00} = f(0,0)$, $c_{01} = f(0,1)$, $c_{10} = f(1,0)$ and $c_{11} = f(1,1)$.

2.2 Bicubic Interpolation

The case $s = 1$ is that of bicubic interpolation. The interpolating polynomial is of the form

$$g(x,y) = \begin{bmatrix} (1-x)^3 & x(1-x)^2 & x^2(1-x) & x^3 \end{bmatrix} \left[\begin{array}{cc|cc} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ \hline c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{array}\right] \begin{bmatrix} (1-y)^3 \\ y(1-y)^2 \\ y^2(1-y) \\ y^3 \end{bmatrix} \tag{9}$$

The matrix is partitioned to make it clear that local controllability is obtained by allowing each corner of the domain to define the 2×2 submatrix corresponding to it.

For example, the upper-left block has entries c_{00}, c_{01}, c_{10} and c_{11} that should be determined solely by $f(0,0)$, $f_x(0,0)$, $f_y(0,0)$ and $f_{xy}(0,0)$. If g and its derivatives are computed at $(0,0)$, it is clear that c_{00} is influenced by $g(0,0)$; c_{10} is influenced by $g_x(0,0)$, where g_x is the partial derivative of g with respect to x; c_{01} is influenced by $g_y(0,0)$, where g_y is the partial derivative of g with respect to y and c_{11} is influenced by $g_{xy}(0,0)$, where g_{xy} is the mixed second-order partial derivative of g. This suggests that $f(0,0)$, $f_x(0,0)$, $f_y(0,0)$ and $f_{xy}(0,0)$ must be specified.

Similar arguments apply to the three other corners of the domain. The values of f, f_x, f_y and f_{xy} are specified at the four corners; that is, there are 16 known values that participate in a linear system of 16 equations in the 16 unknown values c_{ij}. The solution is listed next, using back substitution.

$$
\begin{aligned}
c_{00} &= f(0,0) & c_{03} &= f(0,1)\\
c_{10} &= 3c_{00} + f_x(0,0) & c_{13} &= 3c_{03} + f_x(0,1)\\
c_{01} &= 3c_{00} + f_y(0,0) & c_{02} &= 3c_{03} - f_y(0,1)\\
c_{11} &= -9c_{00} + 3c_{10} + 3c_{01} + f_{xy}(0,0) & c_{12} &= -9c_{03} + 3c_{13} + 3c_{02} - f_{xy}(0,1)\\
\\
c_{30} &= f(1,0) & c_{33} &= f(1,1)\\
c_{20} &= 3c_{30} - f_x(1,0) & c_{23} &= 3c_{33} - f_x(1,1)\\
c_{31} &= 3c_{30} + f_y(1,0) & c_{32} &= 3c_{33} - f_y(1,1)\\
c_{21} &= -9c_{30} + 3c_{20} + 3c_{31} - f_{xy}(1,0) & c_{22} &= -9c_{33} + 3c_{23} + 3c_{32} + f_{xy}(1,1)
\end{aligned}
\tag{10}
$$

Notice that the corner coefficients are assigned first. At each corner, the two neighboring coefficients are computed second. The coefficient neighboring those two is compute third. In a computer program, a macro may be used to simplify the code, as shown in listing 1.

Listing 1. Pseudocode for solving the linear system that determines the coefficients of the fitted bicubic polynomial.

```
#define Solve(v00,v10,v01,v11,f,fx,fy,fxy) \
    v00 = f; \
    v10 = 3*v00 + (fx); \
    v01 = 3*v00 + (fy); \
    v11 = -9*v00 + 3*v10 + 3*v01 + (fxy)

Solve(c00,c10,c01,c11,  f00,   fx00,   fy00,   fxy00);
Solve(c03,c13,c02,c12,  f01,   fx01,  -fy01,  -fxy01);
Solve(c30,c20,c31,c21,  f10,  -fx10,   fy10,  -fxy10);
Solve(c33,c23,c32,c22,  f11,  -fx11,  -fy11,   fxy11);
```

2.3 Biquintic Interpolation

The case $s = 2$ is that of biquintic interpolation. The construction is analogous to that for bicubic interpolation. The interpolating polynomial is of the form

$$
g(x,y) = \begin{bmatrix} (1-x)^5 \\ x(1-x)^4 \\ x^2(1-x)^3 \\ x^3(1-x)^2 \\ x^4(1-x) \\ x^5 \end{bmatrix}^{\mathsf{T}} \left[\begin{array}{cccccc} c_{00} & c_{01} & c_{02} & c_{03} & c_{04} & c_{05} \\ c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{20} & c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ \hline c_{30} & c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{40} & c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ c_{50} & c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \end{array}\right] \begin{bmatrix} (1-y)^5 \\ y(1-y)^4 \\ y^2(1-y)^3 \\ y^3(1-y)^2 \\ y^4(1-y) \\ y^5 \end{bmatrix}
\tag{11}
$$

The matrix is partitioned to make it clear that local controllability is obtained by allowing each corner of the domain to define the 3×3 submatrix corresponding to it. The 36 polynomial coefficients are listed next, using the propagation-from-corner back substitution shows in the bicubic case.

$$
\begin{aligned}
c_{00} &= f(0,0) \\
c_{10} &= 5c_{00} + f_x(0,0) \\
c_{01} &= 5c_{00} + f_y(0,0) \\
c_{20} &= -10c_{00} + 4c_{10} + f_{xx}(0,0)/2 \\
c_{11} &= -25c_{00} + 5c_{10} + 5c_{01} + f_{xy}(0,0) \\
c_{02} &= -10c_{00} + 4c_{01} + f_{yy}(0,0)/2 \\
c_{21} &= 50c_{00} - 20c_{10} - 10c_{01} + 5c_{20} + 4c_{11} + f_{xxy}(0,0)/2 \\
c_{12} &= 50c_{00} - 10c_{10} - 20c_{01} + 4c_{11} + 5c_{02} + f_{xyy}(0,0)/2 \\
c_{22} &= -100c_{00} + 40c_{10} + 40c_{01} - 10c_{20} - 16c_{11} - 10c_{02} + 4c_{21} + 4c_{12} + f_{xxyy}(0,0)/4
\end{aligned}
\tag{12}
$$

$$
\begin{aligned}
c_{05} &= f(0,1) \\
c_{15} &= 5c_{05} + f_x(0,1) \\
c_{04} &= 5c_{05} - f_y(0,1) \\
c_{25} &= -10c_{05} + 4c_{15} + f_{xx}(0,1)/2 \\
c_{14} &= -25c_{05} + 5c_{15} + 5c_{04} - f_{xy}(0,1) \\
c_{03} &= -10c_{05} + 4c_{04} + f_{yy}(0,1)/2 \\
c_{24} &= 50c_{05} - 20c_{15} - 10c_{04} + 5c_{25} + 4c_{14} - f_{xxy}(0,1)/2 \\
c_{13} &= 50c_{05} - 10c_{15} - 20c_{04} + 4c_{14} + 5c_{03} + f_{xyy}(0,1)/2 \\
c_{23} &= -100c_{05} + 40c_{15} + 40c_{04} - 10c_{25} - 16c_{14} - 10c_{03} + 4c_{24} + 4c_{13} + f_{xxyy}(0,1)/4
\end{aligned}
\tag{13}
$$

$$
\begin{aligned}
c_{50} &= f(1,0) \\
c_{40} &= 5c_{50} - f_x(1,0) \\
c_{51} &= 5c_{50} + f_y(1,0) \\
c_{30} &= -10c_{50} + 4c_{40} + f_{xx}(1,0)/2 \\
c_{41} &= -25c_{50} + 5c_{40} + 5c_{51} - f_{xy}(1,0) \\
c_{52} &= -10c_{50} + 4c_{51} + f_{yy}(1,0)/2 \\
c_{31} &= 50c_{50} - 20c_{40} - 10c_{51} + 5c_{30} + 4c_{41} + f_{xxy}(1,0)/2 \\
c_{42} &= 50c_{50} - 10c_{40} - 20c_{51} + 4c_{41} + 5c_{52} - f_{xyy}(1,0)/2 \\
c_{32} &= -100c_{50} + 40c_{40} + 40c_{51} - 10c_{30} - 16c_{41} - 10c_{52} + 4c_{31} + 4c_{42} + f_{xxyy}(1,0)/4
\end{aligned}
\tag{14}
$$

$$
\begin{aligned}
c_{55} &= f(1,1) \\
c_{45} &= 5c_{55} - f_x(1,1) \\
c_{54} &= 5c_{55} - f_y(1,1) \\
c_{35} &= -10c_{55} + 4c_{45} + f_{xx}(1,1)/2 \\
c_{44} &= -25c_{55} + 5c_{45} + 5c_{54} + f_{xy}(1,1) \\
c_{53} &= -10c_{55} + 4c_{54} + f_{yy}(1,1)/2 \\
c_{34} &= 50c_{55} - 20c_{45} - 10c_{54} + 5c_{35} + 4c_{44} - f_{xxy}(1,1)/2 \\
c_{43} &= 50c_{55} - 10c_{45} - 20c_{54} + 4c_{44} + 5c_{53} - f_{xyy}(1,1)/2 \\
c_{33} &= -100c_{55} + 40c_{45} + 40c_{54} - 10c_{35} - 16c_{44} - 10c_{53} + 4c_{34} + 4c_{43} + f_{xxyy}(1,1)/4
\end{aligned}
\tag{15}
$$

Notice that the corner coefficients are assigned first. At each corner, the three neighboring coefficients are computed second. The coefficients neighboring those two are compute next, and so on. In a computer program, a macro may be used to simplify the code, as shown in listing 2.

Listing 2. Pseudocode for solving the linear system that determines the coefficients of the fitted biquintic polynomial.

```
#define Solve(v00,v10,v01,v20,v11,v02,v21,v12,v22,f,fx,fy,fxx,fxy,fxz,fxxy,fxyy,fxxyy) \
    v00 = f; \
    v10 = 5*v00 + (fx); \
    v01 = 5*v00 + (fy); \
    v20 = -10*v00 + 4*v10 + (fxx)/2; \
    v11 = -25*v00 + 5*v10 + 5*v01 + (fxy); \
    v02 = -10*v00 + 4*v01 + (fyy)/2; \
    v21 = 50*v00 - 20*v10 - 10*v01 + 5*v20 + 4*v11 + (fxxy)/2; \
    v12 = 50*v00 - 10*v10 - 20*v01 + 4*v11 + 5*v02 + (fxyy)/2; \
    v22 = -100*v00 + 40*v10 + 40*v01 - 10*v20 - 16*v11 - 10*v02 + 4*v21 + 4*v12 + (fxxyy)/4

Solve(c00,c10,c01,c20,c11,c02,c21,c12,c22, f00,  fx00,  fy00, fxx00,  fxy00, fyy00,  fxxy00,  fxyy00, fxxyy00);
Solve(c05,c15,c04,c25,c14,c03,c24,c13,c23, f01,  fx01, -fy01, fxx01, -fxy01, fyy01, -fxxy01,  fxyy01, fxxyy01);
Solve(c50,c40,c51,c30,c41,c52,c31,c42,c32, f10, -fx10,  fy10, fxx10, -fxy10, fyy10,  fxxy10, -fxyy10, fxxyy10);
Solve(c55,c45,c54,c35,c44,c53,c34,c43,c33, f11, -fx11, -fy11, fxx11,  fxy11, fyy11, -fxxy11, -fxyy11, fxxyy11);
```

---

3 Interpolation for 3D Lattices

The concepts for 2D lattices extend naturally to 3D lattices.

3.1 Trilinear Interpolation

The case $s = 0$ is that of trilinear interpolation. The values $f(u, v, w)$ for $(u, v, w) \in \{0, 1\}^3$ are known and the function is to be fitted with a trilinear polynomial

$$g(x, y, z) = \sum_{i=0}^{1} \sum_{j=0}^{1} \sum_{k=0}^{1} c_{ijk} P_{1,i}(x) P_{1,j}(y) P_{1,k}(z) \tag{16}$$

where $P_{d,m}(t) = (1 - t)^{d-m} t^m$. The trivial solution for the coefficients is $c_{ijk} = f(i, j, k)$.

3.2 Tricubic Interpolation

The case $s = 1$ is that of tricubic interpolation. The values for $f(u, v, w)$, $f_x(u, v, w)$, $f_y(u, v, w)$, $f_z(u, v, w)$, $f_{xy}(u, v, w)$, $f_{xz}(u, v, w)$, $f_{yz}(u, v, w)$ and $f_{xyz}(u, v, w)$ for $(u, v, w) \in \{0, 1\}^3$ are known. Fit the function with a tricubic polynomial that matches the specifed f-values,

$$g(x, y, z) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} c_{ijk} P_{3,i}(x) P_{3,j}(y) P_{3,k}(z) \tag{17}$$

where $P_{d,m}(t) = (1 - t)^{d-m} t^m$. Consider the vector of polynomials

$$\mathbf{P}_3(t) = ((1 - t)^3, (1 - t)^2 t, (1 - t)t^2, t^3) \tag{18}$$

with derivative

$$\mathbf{P}_3'(t) = (-3(1 - t)^2, (1 - t)^2 - 2(1 - t)t, (1 - t)2t - t^2, 3t^2) \tag{19}$$

Evaluating at the endpoints,

$$\mathbf{P}_3(0) = (1,0,0,0),\ \ \mathbf{P}_3(1) = (0,0,0,1),\ \ \mathbf{P}'_3(0) = (-3,1,0,0),\ \ \mathbf{P}'_3(1) = (0,0,-1,3) \tag{20}$$

The derivatives needed to match f-values at $(0,0,0)$ are

$$\begin{aligned}
g_x(x,y,z) &= \textstyle\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} c_{ijk} P'_{3,i}(x) P_{3,j}(y) P_{3,k}(z) \\
g_y(x,y,z) &= \textstyle\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} c_{ijk} P_{3,i}(x) P'_{3,j}(y) P_{3,k}(z) \\
g_z(x,y,z) &= \textstyle\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} c_{ijk} P_{3,i}(x) P_{3,j}(y) P'_{3,k}(z) \\
g_{xy}(x,y,z) &= \textstyle\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} c_{ijk} P'_{3,i}(x) P'_{3,j}(y) P_{3,k}(z) \\
g_{xz}(x,y,z) &= \textstyle\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} c_{ijk} P'_{3,i}(x) P_{3,j}(y) P'_{3,k}(z) \\
g_{yz}(x,y,z) &= \textstyle\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} c_{ijk} P_{3,i}(x) P'_{3,j}(y) P'_{3,k}(z) \\
g_{xyz}(x,y,z) &= \textstyle\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} c_{ijk} P'_{3,i}(x) P'_{3,j}(y) P'_{3,k}(z)
\end{aligned} \tag{21}$$

Evaluating at $(0,0,0)$, we have 8 equations in 8 unknowns

$$\begin{aligned}
f(0,0,0) &= c_{000} \\
f_x(0,0,0) &= -3c_{000} + c_{100} \\
f_y(0,0,0) &= -3c_{000} + c_{010} \\
f_z(0,0,0) &= -3c_{000} + c_{001} \\
f_{xy}(0,0,0) &= 9c_{000} - 3c_{100} - 3c_{010} + c_{110} \\
f_{xz}(0,0,0) &= 9c_{000} - 3c_{100} - 3c_{001} + c_{101} \\
f_{yz}(0,0,0) &= 9c_{000} - 3c_{010} - 3c_{001} + c_{011} \\
f_{xyz}(0,0,0) &= -27c_{000} + 9c_{100} + 9c_{010} + 9c_{001} - 3c_{110} - 3c_{101} - 3c_{011} + c_{111}
\end{aligned} \tag{22}$$

The solution is

$$\begin{aligned}
c_{000} &= f(0,0,0) \\
c_{100} &= 3c_{000} + f_x(0,0,0) \\
c_{010} &= 3c_{000} + f_y(0,0,0) \\
c_{001} &= 3c_{000} + f_z(0,0,0) \\
c_{110} &= -9c_{000} + 3c_{100} + 3c_{010} + f_{xy}(0,0,0) \\
c_{101} &= -9c_{000} + 3c_{100} + 3c_{001} + f_{xz}(0,0,0) \\
c_{011} &= -9c_{000} + 3c_{010} + 3c_{001} + f_{yz}(0,0,0) \\
c_{111} &= 27c_{000} - 9c_{100} - 9c_{010} - 9c_{001} + 3c_{110} + 3c_{101} + 3c_{011} + f_{xyz}(0,0,0)
\end{aligned} \tag{23}$$

As in the bicubic construction, the code uses a macro that also supports computation at the other 7 corners of the domain, as shown in listing 3.

Listing 3. Pseudocode for solving the linear system that determines the coefficients of the fitted tricubic polynomial.

```
#define Solve(v000,v100,v010,v001,v110,v101,v011,v111,f,fx,fy,fz,fxy,fxz,fyz,fxyz) \
    v000 = f; \
    v100 = 3*v000 + (fx); \
    v010 = 3*v000 + (fy); \
    v001 = 3*v000 + (fz); \
    v110 = -9*v000 + 3*v100 + 3*v010 + (fxy); \
    v101 = -9*v000 + 3*v100 + 3*v001 + (fxz); \
    v011 = -9*v000 + 3*v010 + 3*v001 + (fyz); \
    v111 = 27*v000 - 9*v100 - 9*v010 - 9*v001 + 3*v110 + 3*v101 + 3*v011 + (fxyz)

Solve(c000,c100,c010,c001,c110,c101,c011,c111,
    f000, fx000, fy000, fz000, fxy000, fxz000, fyz000, fxyz000);  // +x +y +z

Solve(c300,c200,c310,c301,c210,c201,c311,c211,
    f100,-fx100, fy100, fz100,-fxy100,-fxz100, fyz100,-fxyz100);  // -x +y +z

Solve(c030,c130,c020,c031,c120,c131,c021,c121,
    f010, fx010,-fy010, fz010,-fxy010, fxz010,-fyz010,-fxyz010);  // +x -y +z

Solve(c330,c230,c320,c331,c220,c231,c321,c221,
    f110,-fx110,-fy110, fz110, fxy110,-fxz110,-fyz110, fxyz100);  // -x -y +z

Solve(c003,c103,c013,c002,c113,c102,c012,c112,
    f001, fx001, fy001,-fz001, fxy001,-fxz001,-fyz001,-fxyz001);  // +x +y -z

Solve(c303,c203,c313,c302,c213,c202,c312,c212,
    f101,-fx101, fy101,-fz101,-fxy101, fxz101,-fyz101, fxyz101);  // -x +y -z

Solve(c033,c133,c023,c032,c123,c132,c022,c122,
    f011, fx011,-fy011,-fz011,-fxy011,-fxz011, fyz011, fxyz011);  // +x -y -z

Solve(c333,c233,c323,c332,c223,c232,c322,c222,
    f111,-fx111,-fy111,-fz111, fxy111, fxz111, fyz111,-fxyz111);  // -x -y -z
```

Function calls can be used rather than macros.

```
void Solve (float& v000, float& v100, float& v001, float& v110, float& v101, float& v011, float& v111,
    float f, float fx, float fy, float fz, float fxy, float fxz, float fyz, float fxyz)
{
  v000 = f;
  v100 = 3*v000 + fx;
  v010 = 3*v000 + fy;
  v001 = 3*v000 + fz;
  v110 = -9*v000 + 3*v100 + 3*v010 + fxy;
  v101 = -9*v000 + 3*v100 + 3*v001 + fxz;
  v011 = -9*v000 + 3*v010 + 3*v001 + fyz;
  v111 = 27*v000 - 9*v100 - 9*v010 - 9*v001 + 3*v110 + 3*v101 + 3*v011 + fxyz;
}

for (int i2 = 0; i2 <= 1; ++i2)
{
    int z2 = 0 + 3*i2;
    int p2 = 1 + 2*i2;
    float s2 = 1 - 2*i2;
    for (int i1 = 0; i1 <= 1; ++i1)
    {
        int z1 = 0 + 3*i1;
        int p1 = 1 + 2*i1;
        float s1 = 1 - 2*i1;
        for (int i0 = 0; i0 <= 1; ++i0)
        {
            int z0 = 0 + 3*i0;
            int p0 = 1 + 2*i0;
            float s0 = 1 - 2*i0;
            Solve(
                c(z0,z1,z2),
                c(p0,z1,z2),
                c(z0,p1,z2),
                c(z0,z1,p2),
```

```
                    c(p0,p1,z2),
                    c(p0,z1,p2),
                    c(z0,p1,p2),
                    c(p0,p1,p2),
                    f(i0,i1,i2),
                    s0*fx(i0,i1,i2),
                    s1*fy(i0,i1,i2),
                    s2*fz(i0,i1,i2),
                    s0*s1*fxy(i0,i1,i2),
                    s0*s2*fxz(i0,i1,i2),
                    s1*s2*fyz(i0,i1,i2),
                    s0*s1*s2*fxyz(i0,i1,i2));
            }
        }
    }
```

4 Finite Difference Estimates

Algorithms for these are presented in Derivative Approximations by Finite Differences. The estimates can be used for the derivatives when only the function values are specified for the fitting by a global interpolation.