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
Last Modified: January 29, 2024

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.

2 Interpolation for 1D Lattices

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

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

2.1 Linear Interpolation

Consider the case $s=0$ where

$$g(x) = c_0(1-x) + c_1 x, \;\; x \in [0,1] \tag{3}$$

The samples at the interval endpoints are $f(0)$ and $f(1)$. The linear system of equations is $g(0)=f(0)$ and $g(1)=f(1)$. The solution is

$$c_0 = f(0), \;\; c_1 = f(1) \tag{4}$$

The interpolation is locally controllable because c_0 is determined solely by the sample at $x=0$ and c_1 is determined solely by the sample at $x=1$.

Two linear curves can be constructed from 3 samples $f(0)$, $f(1)$, and $f(2)$. The interpolating function $g_0(x)$ is constructed on $[0,1]$ using $f(0)$ and $f(1)$,

$$g_0(x) = f(0)(1-x) + f(1)x \tag{5}$$

The interpolating function $g_1(x)$ is constructed on $[1,2]$ using $f(1)$ and $f(2)$,

$$g_1(x) = f(1)(1-(x-1)) + f(2)(x-1) \tag{6}$$

The function

$$P(x) = \begin{cases} g_0(x), & x \in [0,1] \\ g_1(x), & x \in [1,2] \end{cases} \tag{7}$$

is continuous at $x=1$ because

$$\lim_{x \to 1^-} P(x) = g_0(1) = f(1) = g_1(1) = \lim_{x \to 1^+} P(x) \tag{8}$$

The idea extends to a sequence of $m+1$ samples $\{f(i)\}_{i=0}^{m}$. The piecewise curve $P(x)$ is defined for $x \in [0,m]$ and has pieces given by the interpolating polynomials

$$g_i(x) = f(i)(1-(x-i)) + f(i+1)(x-i) \tag{9}$$

for $0 \le i \le m-1$ and $x \in [i, i+1]$. $P(x)$ has C^0 continuity.

2.2 Cubic Interpolation

Consider the case $s = 1$ where

$$g(x) = c_0(1-x)^3 + c_1 x(1-x)^2 + c_2 x^2(1-x) + c_3 x^3, x \in [0,1] \tag{10}$$

The samples at the interval endpoints are $(f(0), f'(0))$ and $(f(1), f'(1))$. The linear system of equations is $g(0) = f(0)$, $g'(0) = f'(0)$, $g(1) = f(1)$, and $g'(1) = f'(1)$. The solution is

$$c_0 = f(0), \;\; c_1 = 3f(0) + f'(0), \;\; c_2 = 3f(1) - f'(1), \;\; c_3 = f(1) \tag{11}$$

The interpolation is locally controllable because c_0 and c_1 are determined solely by the sample at $x = 0$ and because c_2 and c_3 are determined solely by the sample at $x = 1$.

Two cubic curves can be constructed from 3 samples $(f(0), f'(0))$, $(f(1), f'(1))$, and $(f(2), f'(2))$. The interpolating function $g_0(x)$ is constructed on $[0,1]$ using $(f(0), f'(0))$ and $(f(1), f'(1))$,

$$g_0(x) = f(0)(1-x)^3 + (3f(0) + f'(0))(1-x)^2 x + (3f(1) + f'(1))(1-x)x^2 + f(1)x^3 \tag{12}$$

The interpolating function $g_1(x)$ is constructed on $[1,2]$ using $(f(1), f'(1))$ and $(f(2), f'(2))$,

$$\begin{aligned} g_1(x) \;\; &= \;\; f(1)(1-(x-1))^3 + (3f(1) + f'(1))(1-(x-1))^2(x-1) \\ &+ \;\; (3f(2) + f'(2))(1-(x-1))(x-1)^2 + f(2)(x-1)^3 \end{aligned} \tag{13}$$

The function and its derivative

$$P(x) = \left\{ \begin{array}{ll} g_0(x), & x \in [0,1] \\ g_1(x), & x \in [1,2] \end{array} \right\}, \;\; P'(x) = \left\{ \begin{array}{ll} g_0'(x), & x \in [0,1] \\ g_1'(x), & x \in [1,2] \end{array} \right\} \tag{14}$$

are continuous at $x = 1$ because

$$\begin{aligned} &\lim_{x \to 1^-} P(x) = g_0(1) = f(1) = g_1(1) = \lim_{x \to 1^+} P(x) \\ &\lim_{x \to 1^-} P'(x) = g_0'(1) = f'(1) = g_1'(1) = \lim_{x \to 1^+} P'(x) \end{aligned} \tag{15}$$

The idea extends to a sequence of $m+1$ samples $\{(f(i), f'(i)\}_{i=0}^{m}$. The piecewise curve $P(x)$ is defined for $x \in [0,m]$ and has pieces given by the interpolating polynomials

$$\begin{aligned} g_i(x) \;\; &= \;\; f(i)(1-(x-i))^3 + (3f(i) + f'(i))(1-(x-i))^2(x-i) \\ &+ \;\; (3f(i+1) + f'(i+1))(1-(x-i))(x-i)^2 + f(i+1)(x-i)^3 \end{aligned} \tag{16}$$

for $0 \le i \le m-1$ and $x \in [i, i+1]$. $P(x)$ has C^1 continuity.

2.3 Quintic Interpolation

Consider the case $s = 2$ where

$$g(x) = c_0(1-x)^5 + c_1(1-x)^4 x + c_2(1-x)^3 x^2 + c_3(1-x)^2 x^3 + c_4(1-x)x^4 + c_5 x^5, \quad x \in [0,1] \tag{17}$$

The samples at the intervals are $(f(0), f'(0), f''(0))$ and $(f(1), f'(1), f''(1))$. The linear system of equations is $g(0) = f(0)$, $g'(0) = f'(0)$, $g''(0) = f''(0)$, $g(1) = f(1)$, $g'(1) = f'(1)$, and $g''(1) = f''(1)$. The solution is

$$\begin{aligned} &c_0 = f(0), \quad c_1 = 5f(0) + f'(0), \quad c_2 = 10f(0) + 4f'(0) + f''(0)/2, \\ &c_3 = 10f(1) - 4f'(1) + f''(1)/2, \quad c_4 = 5f(1) - f'(1), \quad c_5 = f(1) \end{aligned} \tag{18}$$

The interpolation is locally controllable because c_0, c_1, and c_2 are determined solely by the sample at $x = 0$ and because c_3, c_4, and c_5 are determined solely by the sample at $x = 1$.

Two quintic curves can be constructed from 3 samples $(f(0), f'(0))$, $(f(1), f'(1))$, and $(f(2), f'(2))$. The interpolating function $g_0(x)$ is constructed on $[0,1]$ using $(f(0), f'(0))$ and $(f(1), f'(1))$,

$$\begin{aligned} g_0(x) \quad = \quad & f(0)(1-x)^5 + (5f(0) + f'(0))(1-x)^4 x \\ & + \ (10f(0) + 4f'(0) + f''(0)/2)(1-x)^3 x^2 \\ & + \ (10f(1) - 4f'(1) + f''(1)/2)(1-x)^2 x^3 \\ & + \ (5f(1) - f'(1))(1-x)x^4 + f(1)x^5 \end{aligned} \tag{19}$$

The interpolating function $g_1(x)$ is constructed on $[1,2]$ using $(f(1), f'(1))$ and $(f(2), f'(2))$,

$$\begin{aligned} g_1(x) \quad = \quad & f(1)(1-(x-1))^5 + (5f(1) + f'(1))(1-(x-1))^4 (x-1) \\ & + \ (10f(1) + 4f'(1) + f''(1)/2)(1-(x-1))^3 (x-1)^2 \\ & + \ (10f(2) - 4f'(2) + f''(2)/2)(1-(x-1))^2 (x-1)^3 \\ & + \ (5f(2) - f'(2))(1-(x-1))(x-1)^4 + f(2)(x-1)^5 \end{aligned} \tag{20}$$

The function, its first-order derivative, and its second-order derivative

$$P(x) = \left\{ \begin{array}{ll} g_0(x), & x \in [0,1] \\ g_1(x), & x \in [1,2] \end{array} \right\}, \quad P'(x) = \left\{ \begin{array}{ll} g_0'(x), & x \in [0,1] \\ g_1'(x), & x \in [1,2] \end{array} \right\}, \quad P''(x) = \left\{ \begin{array}{ll} g_0''(x), & x \in [0,1] \\ g_1''(x), & x \in [1,2] \end{array} \right\} \tag{21}$$

are continuous at $x = 1$ because

$$\begin{aligned} &\lim_{x \to 1^-} P(x) = g_0(1) = f(1) = g_1(1) = \lim_{x \to 1^+} P(x) \\ &\lim_{x \to 1^-} P'(x) = g_0'(1) = f'(1) = g_1'(1) = \lim_{x \to 1^+} P'(x) \\ &\lim_{x \to 1^-} P''(x) = g_0''(1) = f''(1) = g_1''(1) = \lim_{x \to 1^+} P''(x) \end{aligned} \tag{22}$$

The idea extends to a sequence of $m+1$ samples $\{(f(i), f'(i), f''(i)\}_{i=0}^m$. The piecewise curve $P(x)$ is defined

for $x \in [0,m]$ and has pieces given by the interpolating polynomials

$$\begin{aligned} g_i(x) &= f(i)(1-(x-i))^5 + (5f(i)+f'(i))(1-(x-i))^4(x-i) \\ &+ (10f(i)+4f'(i)+f''(i)/2)(1-(x-i))^3(x-i)^2 \\ &+ (10f(i+1)-4f'(i+1)+f''(i+1)/2)(1-(x-i))^2(x-i)^3 \\ &+ (5f(i+1)+f'(i+1))(1-(x-i))(x-i)^5 + f(i+1)(x-i)^5 \end{aligned} \tag{23}$$

for $0 \le i \le m-1$ and $x \in [i,i+1]$. $P(x)$ has C^2 continuity.

2.4 Higher-Degree Interpolation

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.

For a degree d interpolation, the linear system has $2d$ equations and must be solved symbolically. The details are tedious for large d, so manual manipulation can be error prone. I used Mathematica [1] to generate the linear equations and to solve for the matrix of coefficients. It is sufficient to solve only for the linear equations for the first sample. The numbers in the matrix of coefficients have symmetry and sign values that can be exploited. I have abstracted the essence in the source code implementation.

For large d, floating-point rounding errors can cause problems solving the system and in evaluating the interpolating polynomials. The implementation allows for rational numbers and exact arithmetic, but this is typically slow.

3 Interpolation for 2D Lattices

Consider a square domain $[0,1]^2$ for which function values and some of the derivative values are known at the 4 corners of the domain. The interpolating polynomial of a specified odd degree d is

$$g(x,y) = \sum_{i=0}^{d} \sum_{j=0}^{d} c_{ij} P_{d,i}(x) P_{d,j}(y) \tag{24}$$

where $P_{d,m}(w) = (1-w)^{d-m} w^m$ for $0 \le m \le d$ and for $w \in [0,1]$. Define $s \ge 0$ so that $d = 2s+1$. Choices of the derivative values at the 4 corners will determine the coefficients c_{ij} so that $g(x,y)$ has C^s continuity.

A matrix C of size $(d+1)\times(d+1) = (2s+2)\times(2s+2)$ is used to store the c_{ij}, where i is the row index and j is the column index. The matrix C is partitioned into a 2×2 block matrix with blocks $B_{k\ell}$ for $0 \le k \le 1$ and $0 \le \ell \le 1$. Each block has size $(s+1)\times(s+1)$. We want the block $B_{k\ell}$ to be determined solely by the function and derivatives sample at the corner (k,ℓ). The implication is that we need $(s+1)^2$ function and derivative values specified at each corner.

For the interpolation algorithm presented here, c_{ij} depends on the partial derivative

$$\frac{\partial^{i+j} f}{\partial x^i \partial y^j} \tag{25}$$

evaluated at $(x, y) = (k, \ell)$ and all the partial derivatives of orders strictly smaller than $i + j$ evaluated at $(x, y) = (k, \ell)$. The c_{ij} in the block matrix are computed as the solution to a linear system of $(s + 1)^2$ equations in $(s + 1)^2$ unknowns.

The next sections illustrate the ideas for constructing linear ($d = 1$), cubic ($d = 3$), and quintic ($d = 5$) interpolators.

3.1 Linear Interpolation

Consider the case $s = 0$. Define the 2×1 vector

$$\boldsymbol{V}_1(t) = \left[\begin{array}{cc} (1-t) & t \end{array} \right]^{\mathsf{T}} \tag{26}$$

The linear interpolator is

$$g(x, y) = \boldsymbol{V}_1(x)^{\mathsf{T}} \left[\begin{array}{c|c} c_{00} & c_{01} \\ \hline c_{10} & c_{11} \end{array} \right] \boldsymbol{V}_1(y), \quad (x, y) \in [0, 1]^2 \tag{27}$$

The samples at the corners are $f(0, 0)$, $f(0, 1)$, $f(1, 0)$, and $f(1, 1)$. The linear system of equations are $g(0, 0) = f(0, 0)$, $g(0, 1) = f(0, 1)$, $g(1, 0) = f(1, 0)$, and $g(1, 1) = f(1, 1)$. The solution is

$$c_{00} = f(0, 0), \quad c_{01} = f(0, 1), \quad c_{10} = f(1, 0), \quad c_{11} = f(1, 1) \tag{28}$$

The interpolation is locally controllable because c_{00} is determined solely by the sample at $(x, y) = (0, 0)$, c_{01} is determined solely by the sample at $(x, y) = (0, 1)$, c_{10} is determined solely by the sample at $(x, y) = (1, 0)$, and c_{11} is determined solely by the sample at $(x, y) = (1, 1)$.

3.2 Cubic Interpolation

Consider the case $s = 1$. Define the 4×1 vector

$$\boldsymbol{V}_3(t) = \left[\begin{array}{cccc} (1-t)^3 & (1-t)^2 t & (1-t)t^2 & t^3 \end{array} \right]^{\mathsf{T}} \tag{29}$$

The cubic interpolator is

$$g(x, y) = \boldsymbol{V}_3(x)^{\mathsf{T}} \left[\begin{array}{cc|cc} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ \hline c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{array} \right] \boldsymbol{V}_3(y) \quad , \quad (x, y) \in [0, 1]^2 \tag{30}$$

The samples at the corners are $(f(i, j), f_x(i, j), f_y(i, j), f_{xy}(i, j)$ for $(i, j) \in \{0, 1\}^2$. The linear system of equations are $g(i, j) = f(i, j)$, $g_x(i, j) = f_x(i, j)$, $g_y(i, j) = f_y(i, j)$, and $g_{xy}(i, j) = f_{xy}(i, j)$ where the x- and y-subscripts indicate partial derivatives with respect to those variables. The system consists of 16 equations

in 16 unknowns. The solution is shown next and uses the symbols fij for $f(i,j)$, $fuij$ for $f_u(i,j)$, and $fuvij$ for $f_{uv}(i,j)$,

$$\begin{aligned}
c_{00} &= f00 & c_{03} &= f01\\
c_{10} &= 3f00 + fx00 & c_{13} &= 3f01 + fx01\\
c_{01} &= 3f00 + fy00 & c_{02} &= 3f01 - fy01\\
c_{11} &= 9f00 + 3fx00 + 3fy00 + fxy00 & c_{12} &= 9f01 + 3fx01 - 3fy01 - fxy01\\
c_{30} &= f10 & c_{33} &= f11\\
c_{20} &= 3f10 - fx10 & c_{23} &= 3f11 - fx11\\
c_{31} &= 3f10 + fy10 & c_{32} &= 3f11 - fy11\\
c_{21} &= 9f10 - 3fx10 + 3fy10 - fxy10 & c_{22} &= 9f11 - 3fx11 - 3fy11 + fxy11
\end{aligned} \tag{31}$$

Generally, c_{ij} has indices so that i corresponds to the x-variable of the interpolator and j corresponds to the y-variable of the interpolator.

The interpolation is locally controllable because the upper-left 2×2 block of the 4×4 coefficient matrix is determined solely by the sample at $(x,y) = (0,0)$, the upper-right 2×2 block is determined solely by the sample at $(x,y) = (0,1)$, the lower-left block is determined solely by the sample at $(x,y) = (1,0)$, and the lower-right block is determined solely by the sample at $(x,y) = (1,1)$.

The 4 blocks of solutions have symmetry in the coefficients and have sign values that can be exploited in an implementation. Also notice that the corner coefficients are computed first. At each corner, the two neighboring coefficients are computed second. The coefficient neighboring those two is computed third.

For example, in the block corresponding to $(f(0,0), f_x(0,0), f_y(0,0))$, the corner coefficient c_{00} is computed first. The two neighboring coefficients are c_{10} and c_{01} and are computed next. The coefficient neighboring these two is c_{11} and is computed last. Notice that c_{10} is in the $+x$ direction (so to speak) and is influenced by the derivative $f_x(0,0)$. Similarly, c_{01} is in the $+y$ direction (so to speak) and is influenced by the derivative $f_y(0,0)$. The coefficient c_{11} represents an increase in the x direction and an increase in the y direction, so it is influenced by $f_x(0,0)$, $f_y(0,0)$, and $f_{xy}(0,0)$.

Also for example, in the block corresponding to $(f(1,0), f_x(1,0), f_y(1,0))$, the corner coefficient c_{30} is computed first. The two neighboring coefficients are c_{20} and c_{31} and are computed next. The coefficient neighboring these two is c_{21} and is computed last. Notice that c_{20} is in the $-x$ direction (so to speak) and is influenced by the derivative $f_x(1,0)$ but with a negative sign. Similarly, c_{31} is in the $+y$ direction (so to speak) and is influenced by the derivative $f_y(1,0)$. The coefficient c_{21} represents a decrease in the x direction and an increase in the y direction, so it is influenced by $-f_x(1,0)$, $f_y(1,0)$, and $-f_{xy}(1,0)$.

Listing 1 contains pseudocode to implement the aforementioned patterns.

Listing 1. Pseudocode for computing the c_{ij} of the cubic interpolator $g(x,y)$. The type T denotes a numeric type such as float or double. It can also represent an exact rational type.

```
void BlockSolve(in T f, in T fx, in T fy, in T fxy, out T c00, out T c10, out T c01, out T c11)
{
    c00 = f;
    c10 = 3 * f + fx;
    c01 = 3 * f + fy;
    c11 = 9 * f + 3 * fx + 3 * fy + fxy;
}

void FullSolve(in T f[2][2], in T fx[2][2], in T fy[2][2], out T c[4][4])
{
    for (int b0 = 0; b0 <= 1; ++b0)
    {
        int z0 = 3 * b0 + 0, p0 = 1 * b0 + 1, s0 = 1 - 2 * b0;
        for (int b1 = 0; b1 <= 1; ++b1)
```

```
        {
            int z1 = 3 * b1 + 0, p1 = 1 * b1 + 1, s1 = 1 - 2 * b1;
            BlockSolve(f[b0][b1], s0*fx[b0][b1], s1*fy[b0][b1], s0*s1*fxy[b0][b1], c[z0][z1], c[p0][z1], c[z0][p1], c[p0][p1]);
        }
    }
}

// b0 = 0, z0 = 0, p0 = 1, s0 = +1
// b1 = 0, z1 = 0, p1 = 1, s1 = +1
// BlockSolve(f00, +fx00, +fy00, +fxy00, c00, c10, c01, c11);

// b0 = 0, z0 = 0, p0 = 1, s0 = +1
// b1 = 1, z1 = 3, p1 = 2, s1 = -1
// BlockSolve(f01, +fx01, -fy01, -fxy01, c03, c13, c02, c12);

// b0 = 1, z0 = 3, p0 = 2, s0 = -1
// b1 = 0, z1 = 0, p1 = 1, s1 = +1
// BlockSolve(f10, -fx10, +fy10, -fxy10, c30, c20, c31, c21);

// b0 = 1, z0 = 3, p0 = 2, s0 = -1
// b1 = 1, z1 = 3, p1 = 2, s1 = -1
// BlockSolve(f11, -fx11, -fy11, +fxy11, c33, c23, c32, c22);
```

3.3 Quintic Interpolation

Consider the case $s = 2$. Define the 6×1 vector

$$\boldsymbol{V}_5(t) = \begin{bmatrix} (1-t)^5 & (1-t)^4 t & (1-t)^3 t^2 & (1-t)^2 t^3 & (1-t) t^4 & t^3 \end{bmatrix}^{\mathsf{T}} \tag{32}$$

The quintic interpolator is

$$g(x,y) = \boldsymbol{V}_5(x)^{\mathsf{T}} \left[\begin{array}{ccc|ccc} c_{00} & c_{01} & c_{02} & c_{03} & c_{04} & c_{05} \\ c_{10} & c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{20} & c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ \hline c_{30} & c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{40} & c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ c_{50} & c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \end{array}\right] \boldsymbol{V}_5(y), \quad (x,y) \in [0,1]^2 \tag{33}$$

The samples at the corners are

$$(f(i,j), f_x(i,j), f_y(i,j), f_{xx}(i,j), f_{xy}(i,j), f_{yy}(i,j), f_{xxy}(i,j), f_{xyy}(i,j), f_{xxyy}(i,j)) \tag{34}$$

for $i \in \{0,1\}$ and $j \in \{0,1\}$. The linear system of equations are $g(i,j) = f(i,j)$, $g_x(i,j) = f_x(i,j)$, $g_y(i,j) = f_y(i,j)$, $g_{xx}(i,j) = f_{xx}(i,j)$, $g_{xy}(i,j) = f_{xy}(i,j)$, $g_{yy}(i,j) = f_{yy}(i,j)$, $g_{xxy}(i,j) = f_{xxy}(i,j)$, $g_{xyy}(i,j) = f_{xyy}(i,j)$, and $g_{xxyy}(i,j) = f_{xxyy}(i,j)$ where the x- and y-subscripts indicate partial derivatives with respect to those variables. The system consists of 36 equations in 36 unknowns. The solution is shown next and uses the symbols fij for $f(i,j)$, $fuij$ for $f_u(i,j)$, $fuvij$ for $f_{uv}(i,j)$, $fuvwij$ for $f_{uvw}(i,j)$, and f_{uvwtij} for $f_{uvwt}(i,j)$,

$$
\begin{aligned}
c00 &= f00\\
c10 &= 5f00 + fx00\\
c01 &= 5f00 + fy00\\
c20 &= 10f00 + 4fx00 + fxx00/2\\
c11 &= 25f00 + 5fx00 + 5fy00 + fxy00\\
c02 &= 10f00 + 4fy00 + fyy00/2\\
c21 &= 50f00 + 20fx00 + 10fy00 + (5fxx00)/2 + 4fxy00 + fxxy00/2\\
c12 &= 50f00 + 10fx00 + 20fy00 + 4fxy00 + (5fyy00)/2 + fxyy00/2\\
c22 &= 100f00 + 40fx00 + 40fy00 + 5fxx00 + 16fxy00 + 5fyy00 + 2fxxy00 + 2fxyy00 + fxxyy00/4\\
\\
c05 &= f01\\
c15 &= 5f01 + fx01\\
c04 &= 5f01 - fy01\\
c25 &= 10f01 + 4fx01 + fxx01/2\\
c14 &= 25f01 + 5fx01 - 5fy01 - fxy01\\
c03 &= 10f01 - 4fy01 + fyy01/2\\
c24 &= 50f01 + 20fx01 - 10fy01 + (5fxx01)/2 - 4fxy01 - fxxy01/2\\
c13 &= 50f01 + 10fx01 - 20fy01 - 4fxy01 + (5fyy01)/2 + fxyy01/2\\
c23 &= 100f01 + 40fx01 - 40fy01 + 5fxx01 - 16fxy01 + 5fyy01 - 2fxxy01 + 2fxyy01 + fxxyy01/4\\
\\
c50 &= f10\\
c40 &= 5f10 - fx10\\
c51 &= 5f10 + fy10\\
c30 &= 10f10 - 4fx10 + fxx10/2\\
c41 &= 25f10 - 5fx10 + 5fy10 - fxy10\\
c52 &= 10f10 + 4fy10 + fyy10/2\\
c31 &= 50f10 - 20fx10 + 10fy10 + (5fxx10)/2 - 4fxy10 + fxxy10/2\\
c42 &= 50f10 - 10fx10 + 20fy10 - 4fxy10 + (5fyy10)/2 - fxyy10/2\\
c32 &= 100f10 - 40fx10 + 40fy10 + 5fxx10 - 16fxy10 + 5fyy10 + 2fxxy10 - 2fxyy10 + fxxyy10/4\\
\\
c55 &= f11\\
c45 &= 5f11 - fx11\\
c54 &= 5f11 - fy11\\
c35 &= 10f11 - 4fx11 + fxx11/2\\
c44 &= 25f11 - 5fx11 - 5fy11 + fxy11\\
c53 &= 10f11 - 4fy11 + fyy11/2\\
c34 &= 50f11 - 20fx11 - 10fy11 + (5fxx11)/2 + 4fxy11 - fxxy11/2\\
c43 &= 50f11 - 10fx11 - 20fy11 + 4fxy11 + (5fyy11)/2 - fxyy11/2\\
c33 &= 100f11 - 40fx11 - 40fy11 + 5fxx11 + 16fxy11 + 5fyy11 - 2fxxy11 - 2fxyy11 + fxxyy11/4
\end{aligned}
\tag{35}
$$

The symmetry and sign ideas of Section 3.2 are used in the implementation shown in Listing 2.

Listing 2. Pseudocode for computing the c_{ij} of the quintic interpolator $g(x, y)$. The type T denotes a numeric type such as float or double. It can also represent an exact rational type.

```
void BlockSolve(
    in T f, in T fx, in T fy, in T fxx, in T fxy, in T fyy, in T fxxy, in T fxyy, in T fxxyy,
    out T c00, out T c10, out T c01, out T c20, out T c11, out T c02, out T c21, out T c12, out T c22)
{
    c00 = f;
    c10 = 5 * f + fx;
    c01 = 5 * f + fy;
    c20 = 10 * f + 4 * fx + fxx/2;
    c11 = 25 * f + 5 * fx + 5 * fy + fxy;
    c02 = 10 * f + 4 * fy + fyy/2;
    c21 = 50 * f + 20 * fx + 10 * fy + (5 * fxx)/2 + 4 * fxy + fxxy/2;
    c12 = 50 * f + 10 * fx + 20 * fy + 4 * fxy + (5 * fyy)/2 + fxyy/2;
    c22 = 100 * f + 40 * fx + 40 * fy + 5 * fxx + 16 * fxy + 5 * fyy + 2 * fxxy + 2 * fxyy + fxxyy/4;
}
```

```
void FullSolve(
    in T f[2][2],
    in T fx[2][2], in T fy[2][2],
    in T fxx[2][2], in T fxy[2][2], in T fyy[2][2],
    in T fxxy[2][2], in T fxyy[2][2],
    in T fxxyy[2][2],
    out T c[6][6])
{
    for (int b0 = 0; b0 <= 1; ++b0)
    {
        int z0 = 5 * b0 + 0, p0 = 3 * b0 + 1, q0 = 1 * b0 + 2, s0 = 1 - 2 * b0;
        for (int b1 = 0; b1 <= 1; ++b1)
        {
            int z1 = 5 * b1 + 0, p1 = 3 * b1 + 1, q1 = 1 * b1 + 2, s1 = 1 - 2 * b1;
            BlockSolve(
                f[b0][b1],
                s0*fx[b0][b1], s1*fy[b0][b1],
                s0*s0*fxx[b0][b1], s0*s1*fxy[b0][b1], s1*s1*fyy[b0][b1],
                s0*s0*s1*fxxy[b0][b1], s0*s1*s1*fxyy[b0][b1],
                s0*s0*s1*s1*fxxyy[b0][b1],
                c[z0][z1],
                c[p0][z1], c[z0][p1],
                c[q0][z1], c[p0][p1], c[z0][q1],
                c[q0][p1], c[p0][q1],
                c[q0][q1]);
        }
    }



    // b0 = 0, z0 = 0, p0 = 1, q0 = 2, s0 = +1
    // b1 = 0, z1 = 0, p1 = 1, q1 = 2, s1 = +1
    // BlockSolve(f00, +fx00, +fy00, +fxx00, +fxy00, +fyy00, +fxxy00, +fxyy00, +fxxyy00, c00, c10, c01, c20, c11, c02, c21, c12, c22);

    // b0 = 0, z0 = 0, p0 = 1, q0 = 2, s0 = +1
    // b1 = 1, z1 = 5, p1 = 4, q1 = 3, s1 = -1
    // BlockSolve(f01, +fx01, -fy01, +fxx01, -fxy01, +fyy01, -fxxy01, +fxyy01, +fxxyy01, c05, c15, c04, c25, c14, c03, c24, c13, c23);

    // b0 = 1, z0 = 5, p0 = 4, q0 = 3, s0 = -1
    // b1 = 0, z1 = 0, p1 = 1, q1 = 2, s1 = +1
    // BlockSolve(f10, -fx10, +fy10, +fxx10, -fxy10, +fyy10, +fxxy10, -fxyy10, +fxxyy10, c50, c40, c51, c30, c41, c52, c31, c42, c32);

    // b0 = 1, z0 = 5, p0 = 4, q0 = 3, s0 = -1
    // b1 = 1, z1 = 5, p1 = 4, q1 = 3, s1 = -1
    // BlockSolve(f11, -fx11, -fy11, +fxx11, +fxy11, +fyy11, -fxxy11, -fxyy11, +fxxyy11, c55, c45, c54, c35, c44, c53, c34, c43, c33);
}
```

---

4 Interpolation for 3D Lattices

Consider a cube domain $[0,1]^3$ for which function values and some of the derivative values are known at the 8 corners of the domain. The interpolating polynomial of a specified odd degree d is

$$g(x,y,z) = \sum_{i=0}^{d}\sum_{j=0}^{d}\sum_{k=0}^{d} c_{ijk} P_{d,i}(x) P_{d,j}(y) P_{d,k}(z) \tag{36}$$

where $P_{d,m}(w) = (1-w)^{d-m} w^m$ for $0 \le m \le d$ and for $w \in [0,1]$. Define $s \ge 0$ so that $d = 2s+1$. Choices of the derivative values at the 8 corners will determine the coefficients c_{ijk} so that $g(x,y,z)$ has C^s continuity.

A tensor C of size $(d+1)\times(d+1)\times(d+1) = (2s+2)\times(2s+2)\times(2s+2)$ is used to store the c_{ijk}, where i is the row index, j is the column index, and k is the slice index. Tensor C is partitioned into a $2\times2\times2$ block tensor with blocks $B_{k\ell m}$ for $0 \le k \le 1$, $0 \le \ell \le 1$, and $0 \le m \le 1$. Each block has size $(s+1)\times(s+1)\times(s+1)$. We want the block $B_{k\ell m}$ to be determined solely by the function and derivatives sample at the corner (k,ℓ,m). The implication is that we need $(s+1)^3$ function and derivative values specified at each corner.

For the interpolation algorithm presented here, c_{ijk} depends on the partial derivative

$$\frac{\partial^{i+j+k} f}{\partial x^i \partial y^j \partial z^k} \tag{37}$$

evaluated at $(x,y,z) = (k,\ell,m)$ and all the partial derivatives of orders strictly smaller than $i+j+k$ evaluated at $(x,y,z) = (k,\ell.m)$. The c_{ijk} in the block matrix are computed as the solution to a linear system of $(s+1)^3$ equations in $(s+1)^3$ unknowns.

The next sections illustrate the ideas for constructing linear ($d=1$), cubic ($d=3$), and quintic ($d=5$) interpolators.

4.1 Linear Interpolation

Consider the case $s=0$. Define the 2×1 vector

$$\boldsymbol{V}_1(t) = \begin{bmatrix} (1-t) & t \end{bmatrix}^{\mathsf{T}} \tag{38}$$

The linear interpolator is

$$\begin{aligned} g(x,y,z) &= (1-z)\left(\boldsymbol{V}_1(x)^{\mathsf{T}} \left[\begin{array}{c|c} c_{000} & c_{010} \\ \hline c_{100} & c_{110} \end{array}\right] \boldsymbol{V}_1(y)\right) \\ &+ z\left(\boldsymbol{V}_1(x)^{\mathsf{T}} \left[\begin{array}{c|c} c_{001} & c_{011} \\ \hline c_{101} & c_{111} \end{array}\right] \boldsymbol{V}_1(y)\right) \end{aligned} \tag{39}$$

for $(x,y,z) \in [0,1]^3$. The 8 blocks are each of size $1\times1\times1$, each shown in a unique color. For example, the block B_{000} has a single red-colored element $\{c_{000}\}$. The samples at the corners are $f(r,s,t)$, where $(r,s,t) \in \{0,1\}^3$. The linear system consists of equations $g(r,s,t) = f(r,s,t)$. The solution is

$$c_{ijk} = f(i,j,k), \quad (i,j,k) \in \{0,1\}^3 \tag{40}$$

4.2 Cubic Interpolation

Consider the case $s = 1$. Define the 4×1 vector

$$\boldsymbol{V}_3(t) = \begin{bmatrix} (1-t)^3 & (1-t)^2 t & (1-t)t^2 & t^3 \end{bmatrix}^{\mathsf{T}} \tag{41}$$

The cubic interpolator is

$$\begin{aligned} g(x,y,z) \;=\; & (1-z)^3 \left(\boldsymbol{V}_3(x)^{\mathsf{T}} \left[\begin{array}{cc|cc} c_{000} & c_{010} & c_{020} & c_{030} \\ c_{100} & c_{110} & c_{120} & c_{130} \\ \hline c_{200} & c_{210} & c_{220} & c_{230} \\ c_{300} & c_{310} & c_{320} & c_{330} \end{array}\right] \boldsymbol{V}_3(y) \right) \\ & + (1-z)^2 z \left(\boldsymbol{V}_3(x)^{\mathsf{T}} \left[\begin{array}{cc|cc} c_{001} & c_{011} & c_{021} & c_{031} \\ c_{101} & c_{111} & c_{121} & c_{131} \\ \hline c_{201} & c_{211} & c_{221} & c_{231} \\ c_{301} & c_{311} & c_{321} & c_{331} \end{array}\right] \boldsymbol{V}_3(y) \right) \\ & + (1-z) z^2 \left(\boldsymbol{V}_3(x)^{\mathsf{T}} \left[\begin{array}{cc|cc} c_{002} & c_{012} & c_{022} & c_{032} \\ c_{102} & c_{112} & c_{122} & c_{132} \\ \hline c_{202} & c_{212} & c_{222} & c_{232} \\ c_{302} & c_{312} & c_{322} & c_{332} \end{array}\right] \boldsymbol{V}_3(y) \right) \\ & + z^3 \left(\boldsymbol{V}_3(x)^{\mathsf{T}} \left[\begin{array}{cc|cc} c_{003} & c_{013} & c_{023} & c_{033} \\ c_{103} & c_{113} & c_{123} & c_{133} \\ \hline c_{203} & c_{213} & c_{223} & c_{233} \\ c_{303} & c_{313} & c_{323} & c_{333} \end{array}\right] \boldsymbol{V}_3(y) \right) \end{aligned} \tag{42}$$

for $(x, y, z) \in [0, 1]^3$. The same-colored elements form a block. For example, the block B_{000} has red-colored elements $\{c_{000}, c_{010}, c_{100}, c_{110}, c_{001}, c_{011}, c_{101}, c_{111}\}$. There are 8 blocks of coefficients, each block of size $2 \times 2 \times 2$.

The samples at the corners are

$$(f(r,s,t), f_x(r,s,t), f_y(r,s,t), f_z(r,s,t), f_{xy}(r,s,t), f_{xz}(r,s,t), f_{yz}(r,s,t), f_{xyz}(r,s,t)) \tag{43}$$

for $(r, s, t) \in \{0, 1\}^3$. The linear system consists of equations $g(i,j,k) = f(i,j,k)$, $g_x(i,j,k) = f_x(i,j,k)$, $g_y(i,j,k) = f_y(i,j,k)$, $g_z(i,j,k) = f_z(i,j,k)$, $g_{xy}(i,j,k) = f_{xy}(i,j,k)$, $g_{xz}(i,j,k) = f_{xz}(i,j,k)$, $g_{yz}(i,j,k) = f_{yz}(i,j,k)$, and $g_{xyz}(i,j,k) = f_{xyz}(i,j,k)$ where the x-, y-, and z-subscripts indicate partial derivatives with respect to those variables. The system consists of 64 equations in 64 unknowns. The solution is shown next and use the symbols $fijk$ for $f(i,j,k)$, $fuijk$ for $f_u(i,j,k)$, $fuvijk$ for $f_{uv}(i,j,k)$, and $fuvwijk$ for $f_{uvw}(i,j,k)$,

$$
\begin{aligned}
c000 &= f000 \\
c100 &= 3f000 + fx000 \\
c010 &= 3f000 + fy000 \\
c001 &= 3f000 + fz000 \\
c110 &= 9f000 + 3fx000 + 3fy000 + fxy000 \\
c101 &= 9f000 + 3fx000 + 3fz000 + fxz000 \\
c011 &= 9f000 + 3fy000 + 3fz000 + fyz000 \\
c111 &= 27f000 + 9fx000 + 9fy000 + 9fz000 \\
&\quad + 3fxy000 + 3fxz000 + 3fyz000 + fxyz000
\end{aligned}
\qquad
\begin{aligned}
c003 &= f001 \\
c103 &= 3f001 + fx001 \\
c013 &= 3f001 + fy001 \\
c002 &= 3f001 - fz001 \\
c113 &= 9f001 + 3fx001 + 3fy001 + fxy001 \\
c102 &= 9f001 + 3fx001 - 3fz001 - fxz001 \\
c012 &= 9f001 + 3fy001 - 3fz001 - fyz001 \\
c112 &= 27f001 + 9fx001 + 9fy001 - 9fz001 \\
&\quad + 3fxy001 - 3fxz001 - 3fyz001 - fxyz001
\end{aligned}
$$

$$
\begin{aligned}
c030 &= f010 \\
c130 &= 3f010 + fx010 \\
c020 &= 3f010 - fy010 \\
c031 &= 3f010 + fz010 \\
c120 &= 9f010 + 3fx010 - 3fy010 - fxy010 \\
c131 &= 9f010 + 3fx010 + 3fz010 + fxz010 \\
c021 &= 9f010 - 3fy010 + 3fz010 - fyz010 \\
c121 &= 27f010 + 9fx010 - 9fy010 + 9fz010 \\
&\quad - 3fxy010 + 3fxz010 - 3fyz010 - fxyz010
\end{aligned}
\qquad
\begin{aligned}
c033 &= f011 \\
c133 &= 3f011 + fx011 \\
c023 &= 3f011 - fy011 \\
c123 &= 9f011 + 3fx011 - 3fy011 - fxy011 \\
c032 &= 3f011 - fz011 \\
c132 &= 9f011 + 3fx011 - 3fz011 - fxz011 \\
c022 &= 9f011 - 3fy011 - 3fz011 + fyz011 \\
c122 &= 27f011 + 9fx011 - 9fy011 - 9fz011 \\
&\quad - 3fxy011 - 3fxz011 + 3fyz011 + fxyz011
\end{aligned}
\tag{44}
$$

$$
\begin{aligned}
c300 &= f100 \\
c200 &= 3f100 - fx100 \\
c310 &= 3f100 + fy100 \\
c301 &= 3f100 + fz100 \\
c210 &= 9f100 - 3fx100 + 3fy100 - fxy100 \\
c201 &= 9f100 - 3fx100 + 3fz100 - fxz100 \\
c311 &= 9f100 + 3fy100 + 3fz100 + fyz100 \\
c211 &= 27f100 - 9fx100 + 9fy100 + 9fz100 \\
&\quad - 3fxy100 - 3fxz100 + 3fyz100 - fxyz100
\end{aligned}
\qquad
\begin{aligned}
c303 &= f101 \\
c203 &= 3f101 - fx101 \\
c313 &= 3f101 + fy101 \\
c302 &= 3f101 - fz101 \\
c213 &= 9f101 - 3fx101 + 3fy101 - fxy101 \\
c202 &= 9f101 - 3fx101 - 3fz101 + fxz101 \\
c312 &= 9f101 + 3fy101 - 3fz101 - fyz101 \\
c212 &= 27f101 - 9fx101 + 9fy101 - 9fz101 \\
&\quad - 3fxy101 + 3fxz101 - 3fyz101 + fxyz101
\end{aligned}
$$

$$
\begin{aligned}
c330 &= f110 \\
c230 &= 3f110 - fx110 \\
c320 &= 3f110 - fy110 \\
c331 &= 3f110 + fz110 \\
c220 &= 9f110 - 3fx110 - 3fy110 + fxy110 \\
c231 &= 9f110 - 3fx110 + 3fz110 - fxz110 \\
c321 &= 9f110 - 3fy110 + 3fz110 - fyz110 \\
c221 &= 27f110 - 9fx110 - 9fy110 + 9fz110 \\
&\quad + 3fxy110 - 3fxz110 - 3fyz110 + fxyz110
\end{aligned}
\qquad
\begin{aligned}
c333 &= f111 \\
c233 &= 3f111 - fx111 \\
c323 &= 3f111 - fy111 \\
c223 &= 9f111 - 3fx111 - 3fy111 + fxy111 \\
c332 &= 3f111 - fz111 \\
c232 &= 9f111 - 3fx111 - 3fz111 + fxz111 \\
c322 &= 9f111 - 3fy111 - 3fz111 + fyz111 \\
c222 &= 27f111 - 9fx111 - 9fy111 - 9fz111 \\
&\quad + 3fxy111 + 3fxz111 + 3fyz111 - fxyz111
\end{aligned}
$$

Generally, c_{ijk} has indices so that i corresponds to the x-variable of the interpolator, j corresponds to the y-variable of the interpolator, and k corresponds to the z-variable of the interpolator.

The symmetry and sign ideas of Section 3.2 are used in the implementation shown in Listing 3.

Listing 3. Pseudocode for computing the c_{ijk} of the cubic interpolator $g(x, y, z)$. The type T denotes a numeric type such as float or double. It can also represent an exact rational type.

```
void BlockSolve(
    in T f,in T fx, in T fy, in T fz, in T fxy, in T fxz, in T fyz, in T fxyz,
    out T c000, out T c100, out T c010, out T c001, out T c110, out T c101, out T c011, out T c111)
{
    c000 = f;
    c100 = 3 * f + fx;
    c010 = 3 * f + fy;
    c001 = 3 * f + fz;
    c110 = 9 * f + 3 * fx + 3 * fy + fxy;
    c101 = 9 * f + 3 * fx + 3 * fz + fxz;
    c011 = 9 * f + 3 * fy + 3 * fz + fyz;
    c111 = 27 * f + 9 * fx + 9 * fy + 9 * fz + 3 * fxy + 3 * fxz + 3 * fyz + fxyz;
}
```

```
void FullSolve(
    in T f[2][2][2],
    in T fx[2][2][2], in T fy[2][2][2], in T fz[2][2][2],
    in T fxy[2][2][2], in T fxz[2][2][2], in T fyz[2][2][2],
    in T fxyz[2][2],
    out T c[4][4][4])
{
    for (int b0 = 0; b0 <= 1; ++b0)
    {
        int z0 = 3 * b0 + 0, p0 = 1 * b0 + 1, s0 = 1 - 2 * b0;
        for (int b1 = 0; b1 <= 1; ++b1)
        {
            int z1 = 3 * b1, p1 = 1 * b1 + 1, s1 = 1 - 2 * b1;
            for (int b2 = 0; b2 <= 1; ++b2)
            {
                int z2 = 3 * b2 + 0, p2 = 1 * b2 + 1, s2 = 1 - 2 * b2;
                BlockSolve(
                    f[b0][b1][b2],
                    s0*fx[b0][b1][b2], s1*fy[b0][b1][b2], s2*fz[b0][b1][b2],
                    s0*s1*fxy[b0][b1][b2], s0*s2*fxz[b0][b1][b2]], s1*s2*fyz[b0][b1][b2],
                    s0*s1*s2*fxyz[b0][b1][b2],
                    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]);
            }
        }
    }

    // b0 = 0, z0 = 0, p0 = 1, s0 = +1
    // b1 = 0, z1 = 0, p1 = 1, s1 = +1
    // b2 = 0, z2 = 0, p2 = 1, s2 = +1
    // BlockSolve(f000, +fx000, +fy000, +fz000, +fxy000, +fxz000, +fyz000, +fxyz000, c000, c100, c010, c001, c110, c101, c011, c111)

    // b0 = 0, z0 = 0, p0 = 1, s0 = +1
    // b1 = 0, z1 = 0, p1 = 1, s1 = +1
    // b2 = 1, z2 = 3, p2 = 2, s2 = -1
    // BlockSolve(f001, +fx001, +fy001, -fz001, +fxy001, -fxz001, -fyz001, -fxyz001, c003, c103, c013, c002, c113, c102, c012, c112)

    // b0 = 0, z0 = 0, p0 = 1, s0 = +1
    // b1 = 1, z1 = 3, p1 = 2, s1 = -1
    // b2 = 0, z2 = 0, p2 = 1, s2 = +1
    // BlockSolve(f010, +fx010, -fy010, +fz010, -fxy010, +fxz010, -fyz010, -fxyz010, c030, c130, c020, c031, c120, c131, c021, c121)

    // b0 = 0, z0 = 0, p0 = 1, s0 = +1
    // b1 = 1, z1 = 3, p1 = 2, s1 = -1
    // b2 = 1, z2 = 3, p2 = 2, s2 = -1
    // BlockSolve(f011, +fx011, -fy011, -fz011, -fxy011, -fxz011, +fyz011, +fxyz011, c033, c133, c023, c032, c123, c132, c022, c122)

    // b0 = 1, z0 = 3, p0 = 2, s0 = -1
    // b1 = 0, z1 = 0, p1 = 1, s1 = +1
    // b2 = 0, z2 = 0, p2 = 1, s2 = +1
    // BlockSolve(f100, -fx100, +fy100, +fz100, -fxy100, -fxz100, +fyz100, -fxyz100, c300, c200, c310, c301, c210, c201, c311, c211)

    // b0 = 1, z0 = 3, p0 = 2, s0 = -1
    // b1 = 0, z1 = 0, p1 = 1, s1 = +1
    // b2 = 1, z2 = 3, p2 = 2, s2 = -1
    // BlockSolve(f101, -fx101, +fy101, -fz101, -fxy101, +fxz101, -fyz101, +fxyz101, c303, c203, c313, c302, c213, c202, c312, c212)

    // b0 = 1, z0 = 3, p0 = 2, s0 = -1
    // b1 = 1, z1 = 3, p1 = 2, s1 = -1
    // b2 = 0, z2 = 0, p2 = 1, s2 = +1
    // BlockSolve(f110, -fx110, -fy110, +fz110, +fxy110, -fxz110, -fyz110, +fxyz110, c330, c230, c320, c331, c220, c231, c321, c221)

    // b0 = 1, z0 = 3, p0 = 2, s0 = -1
    // b1 = 1, z1 = 3, p1 = 2, s1 = -1
    // b2 = 1, z2 = 3, p2 = 2, s2 = -1
    // BlockSolve(f111, -fx111, -fy111, -fz111, +fxy111, +fxz111, +fyz111, -fxyz111, c333, c233, c323, c332, c223, c232, c322, c222)
```

4.3 Quintic Interpolation

Consider the case $s = 2$. Define the 6×1 vector

$$\boldsymbol{V}_5(t) = \begin{bmatrix} (1-t)^5 & (1-t)^4 t & (1-t)^3 t^2 & (1-t)^2 t^3 & (1-t) t^4 & t^3 \end{bmatrix}^{\mathsf{T}} \tag{45}$$

The quintic interpolator is

$$\begin{aligned} g(x,y,z) \;=\; & (1-z)^5 \left(\boldsymbol{V}(x)^{\mathsf{T}} \left[\begin{array}{cccc|cc} c_{000} & c_{010} & c_{020} & c_{030} & c_{040} & c_{050} \\ c_{100} & c_{110} & c_{120} & c_{130} & c_{140} & c_{150} \\ c_{200} & c_{210} & c_{220} & c_{230} & c_{240} & c_{250} \\ \hline c_{300} & c_{310} & c_{320} & c_{330} & c_{340} & c_{350} \\ c_{400} & c_{410} & c_{420} & c_{430} & c_{440} & c_{450} \\ c_{500} & c_{510} & c_{520} & c_{530} & c_{540} & c_{550} \end{array}\right] \boldsymbol{V}(y) \right) \\ & + (1-z)^4 z \left(\boldsymbol{V}(x)^{\mathsf{T}} \left[\begin{array}{cccc|cc} c_{001} & c_{011} & c_{021} & c_{031} & c_{041} & c_{051} \\ c_{101} & c_{111} & c_{121} & c_{131} & c_{141} & c_{151} \\ c_{201} & c_{211} & c_{221} & c_{231} & c_{241} & c_{251} \\ \hline c_{301} & c_{311} & c_{321} & c_{331} & c_{341} & c_{351} \\ c_{401} & c_{411} & c_{421} & c_{431} & c_{441} & c_{451} \\ c_{501} & c_{511} & c_{521} & c_{531} & c_{541} & c_{551} \end{array}\right] \boldsymbol{V}(y) \right) \\ & + (1-z)^3 z^2 \left(\boldsymbol{V}(x)^{\mathsf{T}} \left[\begin{array}{cccc|cc} c_{002} & c_{012} & c_{022} & c_{032} & c_{042} & c_{052} \\ c_{102} & c_{112} & c_{122} & c_{132} & c_{142} & c_{152} \\ c_{202} & c_{212} & c_{222} & c_{232} & c_{242} & c_{252} \\ \hline c_{302} & c_{312} & c_{322} & c_{332} & c_{342} & c_{352} \\ c_{402} & c_{412} & c_{422} & c_{432} & c_{442} & c_{452} \\ c_{502} & c_{512} & c_{522} & c_{532} & c_{542} & c_{552} \end{array}\right] \boldsymbol{V}(y) \right) \\ & + (1-z)^2 z^3 \left(\boldsymbol{V}(x)^{\mathsf{T}} \left[\begin{array}{cccc|cc} c_{003} & c_{013} & c_{023} & c_{033} & c_{043} & c_{053} \\ c_{103} & c_{113} & c_{123} & c_{133} & c_{143} & c_{153} \\ c_{203} & c_{213} & c_{223} & c_{233} & c_{243} & c_{253} \\ \hline c_{303} & c_{313} & c_{323} & c_{333} & c_{343} & c_{353} \\ c_{403} & c_{413} & c_{423} & c_{433} & c_{443} & c_{453} \\ c_{503} & c_{513} & c_{523} & c_{533} & c_{543} & c_{553} \end{array}\right] \boldsymbol{V}(y) \right) \\ & + (1-z) z^4 \left(\boldsymbol{V}(x)^{\mathsf{T}} \left[\begin{array}{cccc|cc} c_{004} & c_{014} & c_{024} & c_{034} & c_{044} & c_{054} \\ c_{104} & c_{114} & c_{124} & c_{134} & c_{144} & c_{154} \\ c_{204} & c_{214} & c_{224} & c_{234} & c_{244} & c_{254} \\ \hline c_{304} & c_{314} & c_{324} & c_{334} & c_{344} & c_{354} \\ c_{404} & c_{414} & c_{424} & c_{434} & c_{444} & c_{454} \\ c_{504} & c_{514} & c_{524} & c_{534} & c_{544} & c_{554} \end{array}\right] \boldsymbol{V}(y) \right) \\ & + z^5 \left(\boldsymbol{V}(x)^{\mathsf{T}} \left[\begin{array}{cccc|cc} c_{005} & c_{015} & c_{025} & c_{035} & c_{045} & c_{055} \\ c_{105} & c_{115} & c_{125} & c_{135} & c_{145} & c_{155} \\ c_{205} & c_{215} & c_{225} & c_{235} & c_{245} & c_{255} \\ \hline c_{305} & c_{315} & c_{325} & c_{335} & c_{345} & c_{355} \\ c_{405} & c_{415} & c_{425} & c_{435} & c_{445} & c_{455} \\ c_{505} & c_{515} & c_{525} & c_{535} & c_{545} & c_{555} \end{array}\right] \boldsymbol{V}(y) \right) \end{aligned} \tag{46}$$

for $(x, y, z) \in [0, 1]^3$. The same-colored elements form a block. For example, the block B_{000} has 27 red-colored elements in the upper-left 3×3 portions of the six 6×6 matrices. There are 8 blocks of coefficients, each block of size $3 \times 3 \times 3$.

The samples at the corners require the function and derivatives shown next, each sample evaluted at $(r, s, t) \in \{0, 1\}^3$,

$$\left(\begin{array}{l} f, f_x, f_y, f_z, \\ f_{xx}, f_{xy}, f_{xz}, f_{yy}, f_{yz}, f_{zz}, \\ f_{xxy}, f_{xxz}, f_{xyy}, f_{xyz}, f_{xzz}, f_{yyz}, f_{yzz}, \\ f_{xxyy}, f_{xxyz}, f_{xxzz}, f_{xyyz}, f_{xyzz}, f_{yyzz}, \\ f_{xxyyz}, f_{xxyzz}, f_{xyyzz}, \\ f_{xxyyzz} \end{array}\right) \tag{47}$$

The linear system consists of 216 equations of the form

$$\left.\frac{\partial^{i+j+k} g(x,y,z)}{\partial x^i \partial y^j \partial^k}\right|_{(x,y,z)=(r,s,t)} = \frac{\partial^{i+j+k} f(r,s,t)}{\partial x^i \partial y^j \partial^k} \tag{48}$$

where the left-hand side has derivatives computed first followed by evaluation at (r, s, t) and the right-hand side are the sample values. The system consists of 216 equations in 216 unknowns. However, as seen in previous cases, we can solve a system of 27 equations in 27 unknowns for each of the 8 blocks.

The solution for the block B_{000} of the coefficients is shown next. It is implicit that the function and derivative values are specified at the corner $(0, 0, 0)$,

$$\begin{aligned}
c000 &= f \\
c100 &= 5f + fx \\
c010 &= 5f + fy \\
c001 &= 5f + fz \\
c200 &= 10f + 4fx + fxx/2 \\
c110 &= 25f + 5fx + 5fy + fxy \\
c101 &= 25f + 5fx + 5fz + fxz \\
c020 &= 10f + 4fy + fyy/2 \\
c011 &= 25f + 5fy + 5fz + fyz \\
c002 &= 10f + 4fz + fzz/2 \\
c210 &= 50f + 20fx + 10fy + (5fxx)/2 + 4fxy + fxxy/2 \\
c201 &= 50f + 20fx + 10fz + (5fxx)/2 + 4fxz + fxxz/2 \\
c120 &= 50f + 10fx + 20fy + 4fxy + (5fyy)/2 + fxyy/2 \\
c111 &= 125f + 25fx + 25fy + 25fz + 5fxy + 5fxz + 5fyz + fxyz \\
c102 &= 50f + 10fx + 20fz + 4fxz + (5fzz)/2 + fxzz/2 \\
c021 &= 50f + 20fy + 10fz + (5fyy)/2 + 4fyz + fyyz/2 \\
c012 &= 50f + 10fy + 20fz + 4fyz + (5fzz)/2 + fyzz/2 \\
c220 &= 100f + 40fx + 40fy + 5fxx + 16fxy + 5fyy + 2fxxy + 2fxyy + fxxyy/4 \\
c211 &= 250f + 100fx + 50fy + 50fz + (25fxx)/2 + 20fxy + 20fxz + 10fyz + (5fxxy)/2 + (5fxxz)/2 + 4fxyz + fxxyz/2 \\
c202 &= 100f + 40fx + 40fz + 5fxx + 16fxz + 5fzz + 2fxxz + 2fxzz + fxxzz/4 \\
c121 &= 250f + 50fx + 100fy + 50fz + 20fxy + 10fxz + (25fyy)/2 + 20fyz + (5fxyy)/2 + 4fxyz + (5fyyz)/2 + fxyyz/2 \\
c112 &= 250f + 50fx + 50fy + 100fz + 10fxy + 20fxz + 20fyz + (25fzz)/2 + 4fxyz + (5fxzz)/2 + (5fyzz)/2 + fxyzz/2 \\
c022 &= 100f + 40fy + 40fz + 5fyy + 16fyz + 5fzz + 2fyyz + 2fyzz + fyyzz/4 \\
c221 &= 500f + 200fx + 200fy + 100fz + 25fxx + 80fxy + 40fxz + 25fyy + 40fyz + 10fxxy + 5fxxz + 10fxyy + 16fxyz \\
&\quad + 5fyyz + (5fxxyy)/4 + 2fxxyz + 2fxyyz + fxxyyz/4 \\
c212 &= 500f + 200fx + 100fy + 200fz + 25fxx + 40fxy + 80fxz + 40fyz + 25fzz + 5fxxy + 10fxxz + 16fxyz + 10fxzz \\
&\quad + 5fyzz + 2fxxyz + (5fxxzz)/4 + 2fxyzz + fxxyzz/4 \\
c122 &= 500f + 100fx + 200fy + 200fz + 40fxy + 40fxz + 25fyy + 80fyz + 25fzz + 5fxyy + 16fxyz + 5fxzz + 10fyyz \\
&\quad + 10fyzz + 2fxyyz + 2fxyzz + (5fyyzz)/4 + fxyyzz/4 \\
c222 &= 1000f + 400fx + 400fy + 400fz + 50fxx + 160fxy + 160fxz + 50fyy + 160fyz + 50fzz + 20fxxy + 20fxxz \\
&\quad + 20fxyy + 64fxyz + 20fxzz + 20fyyz + 20fyzz + (5fxxyy)/2 + 8fxxyz + (5fxxzz)/2 + 8fxyyz + 8fxyzz \\
&\quad + (5fyyzz)/2 + fxxyyz + fxxyzz + fxyyzz + fxxyyzz/8
\end{aligned} \tag{49}$$

It is possible to solve symbolically for the coefficients corresponding to the other 7 blocks and to include them in this document. Instead, the symmetry and sign ideas of Section Section 4.2 are used in the implementation shown in Listing 4.

Listing 4. Pseudocode for computing the c_{ijk} of the quintic interpolator $g(x, y, z)$. The type T denotes a numeric type such as float or double. It can also represent an exact rational type.

```
void BlockSolve(
    in T f,
    in T fx, in T fy,fz,
    in T fxx, in T fxy, in T fxz, in T fyy, in T fyz, in T fzz,
    in T fxxy, in T fxxz, in T fxyy, in T fxyz, in T fxzz, in T fyyz, in T fyzz,
    in T fxxyy, in T fxxyz, in T fxxzz, in T fxyyz, in T fxyzz, in T fyyzz,
    in T fxxyyz, in T fxxyzz, in T fxyyzz,
    in T fxxyyzz,
    out T c000,
    out T c100, out T c010, out T c001,
    out T c200, out T c110, out T c101, out T c020, out T c011, out T c002,
    out T c210, out T c201, out T c120, out T c111, out T c012, out T c021, out T c102,
    out T c220, out T c211, out T c202, out T c121, out T c112, out T c022,
    out T c221, out T c212, out T c122,
    out T c222)
{
    c000 = f;
    c100 = 5 * f + fx;
    c010 = 5 * f + fy;
    c001 = 5 * f + fz;
    c200 = 10 * f + 4 * fx + fxx/2;
    c110 = 25 * f + 5 * fx + 5 * fy + fxy;
    c101 = 25 * f + 5 * fx + 5 * fz + fxz;
    c020 = 10 * f + 4 * fy + fyy/2;
    c011 = 25 * f + 5 * fy + 5 * fz + fyz;
    c002 = 10 * f + 4 * fz + fzz/2;
    c210 = 50 * f + 20 * fx + 10 * fy + (5 * fxx)/2 + 4 * fxy + fxxy/2;
    c201 = 50 * f + 20 * fx + 10 * fz + (5 * fxx)/2 + 4 * fxz + fxxz/2;
    c120 = 50 * f + 10 * fx + 20 * fy + 4 * fxy + (5 * fyy)/2 + fxyy/2;
    c111 = 125 * f + 25 * fx + 25 * fy + 25 * fz + 5 * fxy + 5 * fxz + 5 * fyz + fxyz;
    c102 = 50 * f + 10 * fx + 20 * fz + 4 * fxz + (5 * fzz)/2 + fxzz/2;
    c021 = 50 * f + 20 * fy + 10 * fz + (5 * fyy)/2 + 4 * fyz + fyyz/2;
    c012 = 50 * f + 10 * fy + 20 * fz + 4 * fyz + (5 * fzz)/2 + fyzz/2;
    c220 = 100 * f + 40 * fx + 40 * fy + 5 * fxx + 16 * fxy + 5 * fyy + 2 * fxxy + 2 * fxyy + fxxyy/4;
    c211 = 250 * f + 100 vfx + 50 * fy + 50 * fz + (25 * fxx)/2 + 20 * fxy + 20 * fxz + 10 * fyz
           + (5 * fxxy)/2 + (5 * fxxz)/2 + 4 * fxyz + fxxyz/2;
    c202 = 100 * f + 40 * fx + 40 * fz + 5 * fxx + 16 * fxz + 5 * fzz + 2 * fxxz + 2 * fxzz + fxxzz/4;
    c121 = 250 * f + 50 * fx + 100 * fy + 50 * fz + 20 * fxy + 10 * fxz + (25 * fyy)/2 + 20 * fyz
           + (5 * fxyy)/2 + 4 * fxyz + (5 * fyyz)/2 + fxyyz/2;
    c112 = 250 * f + 50 * fx + 50 * fy + 100 * fz + 10 * fxy + 20 * fxz + 20 * fyz + (25 * fzz)/2
           + 4 * fxyz + (5 * fxzz)/2 + (5 * fyzz)/2 + fxyzz/2;
    c022 = 100 * f + 40 * fy + 40 * fz + 5 * fyy + 16 * fyz + 5 * fzz + 2 * fyyz + 2 * fyzz + fyyzz/4;
    c221 = 500 * f + 200 * fx + 200 * fy + 100 * fz + 25 * fxx + 80 * fxy + 40 * fxz + 25 vfyy + 40 * fyz
           + 10 * fxxy + 5 * fxxz + 10 * fxyy + 16 * fxyz + 5 * fyyz
           + (5 * fxxyy)/4 + 2 * fxxyz + 2 * fxyyz + fxxyyz/4;
    c212 = 500 * f + 200 * fx + 100 * fy + 200 * fz + 25 * fxx + 40 * fxy + 80 * fxz + 40 * fyz + 25 * fzz
           + 5 * fxxy + 10 * fxxz + 16 * fxyz + 10 * fxzz + 5 * fyzz
           + 2 * fxxyz + (5 * fxxzz)/4 + 2 * fxyzz + fxxyzz/4;
    c122 = 500 * f + 100 * fx + 200 * fy + 200 * fz + 40 * fxy + 40 * fxz + 25 * fyy + 80 * fyz + 25 * fzz
           + 5 * fxyy + 16 * fxyz + 5 * fxzz + 10 * fyyz + 10 * fyzz
           + 2 * fxyyz + 2 * fxyzz + (5 * fyyzz)/4 + fxyyzz/4;
    c222 = 1000 * f + 400 * fx + 400 * fy + 400 * fz
           + 50 * fxx + 160 * fxy + 160 * fxz + 50 * fyy + 160 * fyz + 50 * fzz
           + 20 * fxxy + 20 * fxxz + 20 * fxyy + 64 * fxyz + 20 * fxzz + 20 * fyyz + 20 * fyzz
           + (5 * fxxyy)/2 + 8 * fxxyz + (5 * fxxzz)/2 + 8 * fxyyz + 8 * fxyzz + (5 * fyyzz)/2
           + fxxyyz + fxxyzz + fxyyzz
           + fxxyyzz/8;
}

void FullSolve(
    in T f[2][2][2],
    in T fx[2][2][2], in T fy[2][2][2], in T fz[2][2][2],
    in T fxx[2][2][2], in T fxy[2][2][2], in T fxz[2][2][2], in T fyy[2][2][2], in T fyz[2][2][2], in T fzz[2][2][2],
    in T fxxy[2][2][2], in T fxxz[2][2][2], in T fxyy[2][2][2], in T fxyz[2][2][2], in T fxzz[2][2][2], in T fyyz[2][2][2], in T fyzz[2][2][2],
    in T fxxyy[2][2][2], in T fxxyz[2][2][2], in T fxxzz[2][2][2], in T fxyyz[2][2][2], in T fxyzz[2][2][2], in T fyyzz[2][2][2],
    in T fxxyyz[2][2][2], in T fxxyzz[2][2][2], in T fxyyzz[2][2][2],
```

```
    in T fxxyyzz[2][2][2],
    out T c[6][6][6])
{
    for (int b0 = 0; b0 <= 1; ++b0)
    {
        int z0 = 5 * b0 + 0, p0 = 3 * b0 + 1, q0 = 1 * b0 + 2, s0 = 1 - 2 * b0;
        for (int b1 = 0; b1 <= 1; ++b1)
        {
            int z1 = 5 * b1 + 0, p1 = 3 * b1 + 1, q1 = 1 * b1 + 2, s1 = 1 - 2 * b1;
            for (int b2 = 0; b2 <= 1; ++b2)
            {
                int z2 = 5 * b2 + 0, p2 = 3 * b2 + 1, q2 = 1 * b2 + 2, s2 = 1 - 2 * b2;
                BlockSolve(
                    f[b0][b1][b2],
                    s0*fx[b0][b1][b2],
                    s1*fy[b0][b1][b2],
                    s2*fz[b0][b1][b2],
                    s0*s0*fxx[b0][b1][b2],
                    s0*s1*fxy[b0][b1][b2],
                    s0*s2*fxz[b0][b1][b2],
                    s1*s1*fyy[b0][b1][b2],
                    s1*s2*fyz[b0][b1][b2],
                    s2*s2*fzz[b0][b1][b2],
                    s0*s0*s1*fxxy[b0][b1][b2],
                    s0*s0*s2*fxxz[b0][b1][b2],
                    s0*s1*s1*fxyy[b0][b1][b2],
                    s0*s1*s2*fxyz[b0][b1][b2],
                    s0*s2*s2*fxzz[b0][b1][b2],
                    s1*s1*s2*fyyz[b0][b1][b2],
                    s1*s2*s2*fyzz[b0][b1][b2],
                    s0*s0*s1*s1*fxxyy[b0][b1][b2],
                    s0*s0*s1*s2*fxxyz[b0][b1][b2],
                    s0*s0*s2*s2*fxxzz[b0][b1][b2],
                    s0*s1*s1*s2*fxyyz[b0][b1][b2],
                    s0*s1*s2*s2*fxyzz[b0][b1][b2],
                    s1*s1*s2*s2*fyyzz[b0][b1][b2],
                    s0*s0*s1*s1*s2*fxxyyz[b0][b1][b2],
                    s0*s0*s1*s2*s2*fxxyzz[b0][b1][b2],
                    s0*s1*s1*s2*s2*fxyyzz[b0][b1][b2],
                    s0*s0*s1*s1*s2*s2*fxxyyzz[b0][b1][b2],
                    c[z0][z1][z2], c[p0][z1][z2], c[z0][p1][z2], c[z0][z1][p2],
                    c[q0][z1][z2], c[p0][p1][z2], c[p0][z1][p2], c[z0][q1][z2],
                    c[z0][p1][p2], c[z0][z1][q2], c[q0][p1][z2], c[q0][z1][p2],
                    c[p0][q1][z2], c[p0][p1][p2], c[p0][z1][q2], c[z0][q1][p2],
                    c[z0][p1][q2], c[q0][q1][z2], c[q0][p1][p2], c[q0][z1][q2],
                    c[p0][q1][p2], c[p0][p1][q2], c[z0][q1][q2], c[q0][q1][p2],
                    c[q0][p1][q2], c[p0][q1][q2], c[q0][q1][q2]);
            }
        }
    }

    // b0 = 0, z0 = 0, p0 = 1, q0 = 2, s0 = +1
    // b1 = 0, z1 = 0, p1 = 1, q1 = 2, s1 = +1
    // b2 = 0, z2 = 0, p2 = 1, q2 = 2, s2 = +1
    // BlockSolve(
    // f000,
    // +fx000, +fy000, +fz000,
    // +fxx000, +fxy000, +fxz000, +fyy000, +fzz000, +fyz000,
    // +fxxy000, +fxxz000, +fxyy000, +fxyz000, +fxzz000, +fyyz000, +fyzz000,
    // +fxxyy000, +fxxyz000, +fxxzz000, +fxyyz000, +fxyzz000, +fyyzz000,
    // +fxxyyz000, +fxxyzz000, +fxyyzz000,
    // +fxxyyzz000,
    // c000,
    // c100, c010, c001,
    // c200, c110, c101, c020, c011, c002,
    // c210, c201, c120, c111, c102, c021, c012,
    // c220, c211, c202, c121, c112, c022,
    // c221, c212, c122,
    // c222);

    // b0 = 0, z0 = 0, p0 = 1, q0 = 2, s0 = +1
    // b1 = 0, z1 = 0, p1 = 1, q1 = 2, s1 = +1
```

```
// b2 = 1, z2 = 5, p2 = 4, q2 = 3, s2 = −1
// BlockSolve(
// f001,
// +fx001, +fy001, −fz001,
// +fxx001, +fxy001, −fxz001, +fyy001, −fyz001, +fzz001,
// +fxxy001, −fxxz001, +fxyy001, −fxyz001, +fxzz001, −fyyz001, +fyzz001,
// +fxxyy001, −fxxyz001, +fxxzz001, −fxyyz001, +fxyzz001, +fyyzz001,
// −fxxyyz001, +fxxyzz001, +fxyyzz001,
// +fxxyyzz001,
// c005,
// c105, c015, c004,
// c205, c115, c104, c025, c014, c003,
// c215, c204, c125, c114, c013, c024, c103,
// c225, c214, c203, c124, c113, c023,
// c224, c213, c123,
// c223);

// b0 = 0, z0 = 0, p0 = 1, q0 = 2, s0 = +1
// b1 = 1, z1 = 5, p1 = 4, q1 = 3, s1 = −1
// b2 = 0, z2 = 0, p2 = 1, q2 = 2, s2 = +1
// BlockSolve(
// f010,
// +fx010, −fy010, +fz010,
// +fxx010, −fxy010, +fxz010, +fyy010, −fyz010, +fzz010,
// −fxxy010, +fxxz010, +fxyy010, −fxyz010, +fxzz010, +fyyz010, −fyzz010,
// +fxxyy010, −fxxyz010, +fxxzz010, +fxyyz010, −fxyzz010, +fyyzz010,
// +fxxyyz010, −fxxyzz010, +fxyyzz010,
// +fxxyyzz010,
// c050,
// c150, c040, c051,
// c250, c140, c151, c030, c041, c052,
// c240, c251, c130, c141, c152, c031, c042,
// c230, c241, c252, c131, c142, c032,
// c231, c242, c132,
// c232);

// b0 = 0, z0 = 0, p0 = 1, q0 = 2, s0 = +1
// b1 = 1, z1 = 5, p1 = 4, q1 = 3, s1 = −1
// b2 = 1, z2 = 5, p2 = 4, q2 = 3, s2 = −1
// BlockSolve(
// f011,
// +fx011, −fy011, −fz011,
// +fxx011, −fxy011, −fxz011, +fyy011, +fyz011, +fzz011,
// −fxxy011, −fxxz011, +fxyy011, +fxyz011, +fxzz011, −fyyz011, −fyzz011,
// +fxxyy011, +fxxyz011, +fxxzz011, −fxyyz011, −fxyzz011, +fyyzz011,
// −fxxyyz011, −fxxyzz011, +fxyyzz011,
// +fxxyyzz011,
// c055,
// c155, c045, c054,
// c255, c145, c154, c035, c044, c053,
// c245, c254, c135, c144, c153, c034, c043,
// c235, c244, c253, c134, c143, c033,
// c234, c243, c133,
// c233);

// b0 = 1, z0 = 5, p0 = 4, q0 = 3, s0 = −1
// b1 = 0, z1 = 0, p1 = 1, q1 = 2, s1 = +1
// b2 = 0, z2 = 0, p2 = 1, q2 = 2, s2 = +1
// BlockSolve(
// f100,
// −fx100, +fy100, +fz100,
// +fxx100, −fxy100, −fxz100, +fyy100, +fyz100, +fzz100,
// +fxxy100, +fxxz100, −fxyy100, −fxyz100, −fxzz100, +fyyz100, +fyzz100,
// +fxxyy100, +fxxyz100, +fxxzz100, −fxyyz100, −fxyzz100, +fyyzz100,
// +fxxyyz100, +fxxyzz100, −fxyyzz100,
// +fxxyyzz100,
// c500,
// c400, c510, c501,
// c300, c410, c401, c520, c511, c502,
// c310, c301, c420, c411, c402, c521, c512,
// c320, c311, c302, c421, c412, c522,
// c321, c312, c422,
```

```
    // c322);

    // b0 = 1, z0 = 5, p0 = 4, q0 = 3, s0 = −1
    // b1 = 0, z1 = 0, p1 = 1, q1 = 2, s1 = +1
    // b2 = 1, z2 = 5, p2 = 4, q2 = 3, s2 = −1
    // BlockSolve(
    // f101,
    // −fx101, +fy101, −fz101,
    // +fxx101, −fxy101, +fxz101, +fyy101, −fyz101, +fzz101,
    // +fxxy101, −fxxz101, −fxyy101, +fxyz101, −fxzz101, −fyyz101, +fyzz101,
    // +fxxyy101, −fxxyz101, +fxxzz101, +fxyyz101, −fxyzz101, +fyyzz101,
    // −fxxyyz101, +fxxyzz101, −fxyyzz101,
    // +fxxyyzz101,
    // c505,
    // c405, c515, c504,
    // c305, c415, c404, c525, c514, c503,
    // c315, c304, c425, c414, c403, c524, c513,
    // c325, c314, c303, c424, c413, c523,
    // c324, c313, c423,
    // c323);

    // b0 = 1, z0 = 5, p0 = 4, q0 = 3, s0 = −1
    // b1 = 1, z1 = 5, p1 = 4, q1 = 3, s1 = −1
    // b2 = 0, z2 = 0, p2 = 1, q2 = 2, s2 = +1
    // BlockSolve(
    // f110,
    // −fx110, −fy110, +fz110,
    // +fxx110, +fxy110, −fxz110, +fyy110, −fyz110, +fzz110,
    // −fxxy110, +fxxz110, −fxyy110, +fxyz110, −fxzz110, +fyyz110, −fyzz110,
    // +fxxyy110, −fxxyz110, +fxxzz110, −fxyyz110, +fxyzz110, +fyyzz110,
    // +fxxyyz110, −fxxyzz110, −fxyyzz110,
    // +fxxyyzz110,
    // c550,
    // c450, c540, c551,
    // c350, c440, c451, c530, c541, c552,
    // c340, c351, c430, c441, c452, c531, c542,
    // c330, c341, c352, c431, c442, c532,
    // c331, c342, c432,
    // c332);

    // b0 = 1, z0 = 5, p0 = 4, q0 = 3, s0 = −1
    // b1 = 1, z1 = 5, p1 = 4, q1 = 3, s1 = −1
    // b2 = 1, z2 = 5, p2 = 4, q2 = 3, s2 = −1
    // BlockSolve(
    // f111,
    // −fx111, −fy111, −fz111,
    // +fxx111, +fxy111, +fxz111, +fyy111, +fyz111, +fzz111,
    // −fxxy111, −fxxzv, −fxyy111, −fxyz111, −fxzz111, −fyyz111, −fyzz111,
    // +fxxyy111, +fxxyz111, +fxxzz111, +fxyyz111, +fxyzz111, +fyyzz111,
    // −fxxyyz111, −fxxyzz111, −fxyyzz111,
    // +fxxyyzz111,
    // c555,
    // c455, c545, c554,
    // c355, c445, c454, c535, c544, c553,
    // c345, c354, c435, c444, c453, c534, c543,
    // c335, c344, c353, c434, c443, c533,
    // c334, c343, c433,
    // c333);
}
```

---

5 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.

References

[1] Wolfram Research, Inc. *Mathematica 14.0.0*. Wolfram Research, Inc., Champaign, Illinois, 2024.