Natural Spline Interpolation

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

Created: January 5, 2008
Last Modified: June 8, 2022

Contents

1 Introduction

Let $\boldsymbol{F} : \mathbb{R} \to \mathbb{R}^m$ be a function whose domain $D \subseteq \mathbb{R}$ is a subset of real numbers and whose range $R \subseteq \mathbb{R}^m$ is a subset of m-tuples of real numbers with $m \geq 1$. The function is written as $\boldsymbol{F}(t)$ and is assumed to have continuous derivatives through order $k+1$. The j-th order derivative is written as $\boldsymbol{F}^{(j)}(t)$ with the understanding that the function itself is $\boldsymbol{F}^{(0)}(t)$.

Natural splines are interpolants of $\boldsymbol{F}(t)$ on the domain $[t_{\min}, t_{\max}]$ that consist of polynomials of degree $d = 2k+1$. The polynomials are constructed from n samples $(t_i, \boldsymbol{f}_i^{(0)}, \ldots, \boldsymbol{f}_i^{(k-1)})$ for $0 \leq i \leq n-1$. It is required that the t-samples are strictly increasing: $t_{\min} = t_0 < t_1 < \ldots, t_{n-2} < t_{n-1} = t_{\max}$. The choice of odd degree allows for choosing a symmetric set of constraints at the endpoints of the domain. The polynomials are

$$\boldsymbol{P}_i(t) = \sum_{j=0}^{d} \boldsymbol{p}_{i,j} \left(\frac{t - t_i}{\Delta_i} \right)^j, \quad 0 \leq i \leq n-2 \tag{1}$$

where $t \in [t_i, t_{i+1}]$ and $\Delta_i = t_{i+1} - t_i$. The piecewise polynomial is

$$\boldsymbol{P}(t) = \begin{cases} \boldsymbol{P}_0(t), & t \in [t_0, t_1] \\ \boldsymbol{P}_1(t), & t \in [t_1, t_2] \\ \vdots & \\ \boldsymbol{P}_{n-3}(t), & t \in [t_{n-3}, t_{n-2}] \\ \boldsymbol{P}_{n-2}(t), & t \in [t_{n-2}, t_{n-1}] \end{cases} \tag{2}$$

The coefficients $\boldsymbol{p}_{i,j}$ are chosen so that $\boldsymbol{P}(t)$ has continuous derivatives through order $k+1$.

The previous version of this document discussed only natural cubic splines. For free or clamped splines, the coefficients are solutions to a tridiagonal linear system which can be solved in $O(N)$ time for N coefficients. I used the pseudocode in an early edition of *Numerical Analysis* [1]. The book *Numerical Recipes in C* [2] also suggested this approach. The citations here are for the most recent versions of the books.

For closed splines, the previous version showed how to set up a block-matrix system which can be row-reduced to an upper-triangular block-matrix system. This type of system can be solved using back substitution in terms of the blocks. In this version of the document, I use the block-matrix approach but with symbolic algebra in order to avoid actually storing and modifying the matrix entries other than those in the lower-right block of the matrix. This approach is also $O(N)$ in time for N coefficients, but has fewer algebraic operations than if one were not to use symbolic algebra. As it turns out, this efficient approach works whether the spline is free, clamped or closed, which allows code sharing among these cases.

The number of polynomials is $n-1$, so we have $(d+1)(n-1)$ coefficients to construct. The samples themselves generate the constraints

$$\begin{aligned} \boldsymbol{P}_i^{(j)}(t_i) &= \boldsymbol{f}_i^{(j)} \text{ for } 0 \leq j \leq k-1 \text{ and } 0 \leq i \leq n-2 \\ \boldsymbol{P}_{n-2}^{(j)}(t_{n-1}) &= \boldsymbol{f}_{n-1}^{(j)} \text{ for } 0 \leq j \leq k-1 \end{aligned} \tag{3}$$

which consists of kn constraints. To obtain C^{k+1} continuity, the polynomials must match at the interior samples as well as their derivatives through order $k+1$,

$$\boldsymbol{P}_i^{(j)}(t_{i+1}) = \boldsymbol{P}_{i+1}^{(j)}(t_{i+1}) \text{ for } 0 \leq j \leq k+1 \text{ and } 0 \leq i \leq n-3 \tag{4}$$

which consists of $(k+2)(n-2)$ constraints. The number of constraints so far is $kn+(k+2)(n-2) = (d+1)(n-1)-2$ constraints. Two additional constraints are needed to reach $(d+1)(n-1)$ constraints in the $(d+1)(n-1)$ unknowns. Three choices are the following:

clamped spline:	$\boldsymbol{P}_0^{(k)}(t_0) = \boldsymbol{c}_0$ and $\boldsymbol{P}_{n-2}^{(k)}(t_{n-1}) = \boldsymbol{c}_1$ where $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ are specified
free spline:	$\boldsymbol{P}_0^{(k+1)}(t_0) = \boldsymbol{0}$ and $\boldsymbol{P}_{n-2}^{(k+1)}(t_{n-1}) = \boldsymbol{0}$
closed spline:	$\boldsymbol{f}_{n-1}^{(j)} = \boldsymbol{f}_0^{(j)}$ for $0 \le j \le k-1$, $\boldsymbol{P}_0^{(k)}(t_0) = \boldsymbol{P}_{n-2}^{(k)}(t_{n-1})$ and $\boldsymbol{P}_0^{(k+1)}(t_0) = \boldsymbol{P}_{n-2}^{(k+1)}(t_{n-1})$

The linear system formulation and solution are illustrated for natural cubic splines (degree 3) and for natural quintic splines (degree 5). The sketch of the extension to odd degrees 7 or larger is provided last.

2 Cubic Splines

The degree is $3 = d = 2k+1$, so $k = 1$. The n samples are $(t_i, \boldsymbol{f}_i)$ for $0 \le i \le n-1$. The number of polynomials is $n-1$, so we have $4(n-1)$ coefficients to construct. The samples themselves generate the constraints

$$\begin{aligned} \boldsymbol{P}_i(t_i) &= \boldsymbol{f}_i \ \text{ for } 0 \le i \le n-2 \\ \boldsymbol{P}_{n-2}(t_{n-1}) &= \boldsymbol{f}_{n-1} \end{aligned} \tag{5}$$

which consists of n constraints. To obtain C^2 continuity, the polynomials and their derivatives must match at the interior samples,

$$\begin{aligned} \boldsymbol{P}_i(t_{i+1}) &= \boldsymbol{P}_{i+1}(t_{i+1}) \quad \text{for } 0 \le i \le n-3 \\ \boldsymbol{P}_i^{(1)}(t_{i+1}) &= \boldsymbol{P}_{i+1}^{(1)}(t_{i+1}) \quad \text{for } 0 \le i \le n-3 \\ \boldsymbol{P}_i^{(2)}(t_{i+1}) &= \boldsymbol{P}_{i+1}^{(2)}(t_{i+1}) \quad \text{for } 0 \le i \le n-3 \end{aligned} \tag{6}$$

which consists of $3(n-2)$ constraints. The number of constraints so far is $n+3(n-2) = 4(n-1)-2$. Two additional constraints are needed to reach $4(n-1)$ constraints in the $4(n-1)$ unknowns. Three standard choices are

clamped spline:	$\boldsymbol{P}_0^{(1)}(t_0) = \boldsymbol{c}_0$ and $\boldsymbol{P}_{n-2}^{(1)}(t_{n-1}) = \boldsymbol{c}_1$ where $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ are specified
free spline:	$\boldsymbol{P}_0^{(2)}(t_0) = \boldsymbol{0}$ and $\boldsymbol{P}_{n-2}^{(2)}(t_{n-1}) = \boldsymbol{0}$
closed spline:	$\boldsymbol{f}_{n-1} = \boldsymbol{f}_0$, $\boldsymbol{P}_0^{(1)}(t_0) = \boldsymbol{P}_{n-2}^{(1)}(t_{n-1})$ and $\boldsymbol{P}_0^{(2)}(t_0) = \boldsymbol{P}_{n-2}^{(2)}(t_{n-1})$

2.1 Setting up the Linear System of Constraints

The polynomials and their relevant derivatives are

$$\begin{aligned} \boldsymbol{P}_i(t) &= \boldsymbol{p}_{i,0} + \boldsymbol{p}_{i,1}\left(\tfrac{t-t_i}{\Delta_i}\right) + \boldsymbol{p}_{i,2}\left(\tfrac{t-t_i}{\Delta_i}\right)^2 + \boldsymbol{p}_{i,3}\left(\tfrac{t-t_i}{\Delta_i}\right)^3 \\ \boldsymbol{P}_i^{(1)}(t) &= \left(\boldsymbol{p}_{i,1} + 2\boldsymbol{p}_{i,2}\left(\tfrac{t-t_i}{\Delta_i}\right) + 3\boldsymbol{p}_{i,3}\left(\tfrac{t-t_i}{\Delta_i}\right)^2\right)/\Delta_i \\ \boldsymbol{P}_i^{(2)}(t) &= \left(2\boldsymbol{p}_{i,2} + 6\boldsymbol{p}_{i,3}\left(\tfrac{t-t_i}{\Delta_i}\right)\right)/\Delta_i^2 \end{aligned} \tag{7}$$

for $0 \leq i \leq n-2$. The constraints of the first line of equation (5) imply

$$\boldsymbol{p}_{i,0} = \boldsymbol{f}_i \;\text{ for } 0 \leq i \leq n-2 \tag{8}$$

The constraints of equation (6) and the second line of equation (5) imply

$$\begin{aligned}
&\boldsymbol{p}_{i,1} + \boldsymbol{p}_{i,2} + \boldsymbol{p}_{i,3} = \boldsymbol{f}_{i+1} - \boldsymbol{f}_i && \text{for } 0 \leq i \leq n-2 \\
&\boldsymbol{p}_{i,1} + 2\boldsymbol{p}_{i,2} + 3\boldsymbol{p}_{i,3} = \sigma_i\, \boldsymbol{p}_{i+1,1} && \text{for } 0 \leq i \leq n-3 \\
&\boldsymbol{p}_{i,2} + 3\boldsymbol{p}_{i,3} = \sigma_i^2\, \boldsymbol{p}_{i+1,2} && \text{for } 0 \leq i \leq n-3
\end{aligned} \tag{9}$$

where $\sigma_i = \Delta_i/\Delta_{i+1}$. Choosing a free, clamped or closed spline, a linear system of $4(n-1)$ equations in $4(n-1)$ unknowns can be created from these constraints.

Equation (8) defines the constant terms of the polynomials. The linear system for the remaining coefficients is constructed from the constraints of equation (9). The system has $3(n-1)$ equations in $3(n-1)$ unknowns and can be written in block-matrix form,

$$\begin{bmatrix}
D & S_0 & O & O & \cdots & O & O & O \\
O & D & S_1 & O & \cdots & O & O & O \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
O & O & O & O & \cdots & O & D & S_{n-3} \\
L & O & O & O & \cdots & O & O & R
\end{bmatrix}
\begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{n-3} \\ P_{n-2} \end{bmatrix} =
\begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_{n-3} \\ G_{n-2} \end{bmatrix} \tag{10}$$

The blocks D, S_i, O, L and R are 3×3. The blocks P_i and G_i are $3 \times m$, each row representing a $1 \times m$ vector. The following blocks are independent of whether the spline is clamped, free or closed,

$$D = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 0 & 1 & 3 \end{bmatrix},\;
S_i = \begin{bmatrix} 0 & 0 & 0 \\ -\sigma_i & 0 & 0 \\ 0 & -\sigma_i^2 & 0 \end{bmatrix},\;
O = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{11}$$

and

$$P_i = \begin{bmatrix} \boldsymbol{p}_{i,1} \\ \boldsymbol{p}_{i,2} \\ \boldsymbol{p}_{i,3} \end{bmatrix} \text{ for } 0 \leq i \leq n-2, \quad
G_i = \begin{bmatrix} \boldsymbol{g}_{3i} \\ \boldsymbol{g}_{3i+1} \\ \boldsymbol{g}_{3i+2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_{i+1} - \boldsymbol{f}_i \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \text{ for } 0 \leq i \leq n-3 \tag{12}$$

The following blocks are dependent on the type of spline,

$$L = \begin{bmatrix} 0 & 0 & 0 \\ \ell_{10} & 0 & 0 \\ 0 & \ell_{21} & 0 \end{bmatrix},\;
R = \begin{bmatrix} 1 & 1 & 1 \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix},\;
G_{n-2} = \begin{bmatrix} \boldsymbol{f}_{n-1} - \boldsymbol{f}_{n-2} \\ \boldsymbol{g}_{3(n-2)+1} \\ \boldsymbol{g}_{3(n-2)+2} \end{bmatrix} \tag{13}$$

The specific values for the matrix entries are

type	L	R			G
clamped	$\ell_{10} = 1$	$r_{10} = 0$	$r_{11} = 0$	$r_{12} = 0$	$\boldsymbol{g}_{3(n-2)+1} = \Delta_0 \boldsymbol{c}_0$
	$\ell_{21} = 0$	$r_{20} = 1$	$r_{21} = 2$	$r_{22} = 3$	$\boldsymbol{g}_{3(n-2)+2} = \Delta_{n-2} \boldsymbol{c}_1$
free	$\ell_{10} = 0$	$r_{10} = 0$	$r_{11} = 1$	$r_{12} = 3$	$\boldsymbol{g}_{3(n-2)+1} = \boldsymbol{0}$
	$\ell_{21} = 1$	$r_{20} = 0$	$r_{21} = 0$	$r_{22} = 0$	$\boldsymbol{g}_{3(n-2)+2} = \boldsymbol{0}$
closed	$\ell_{10} = 1$	$r_{10} = -1$	$r_{11} = -2$	$r_{12} = -3$	$\boldsymbol{g}_{3(n-2)+1} = \boldsymbol{0}$
	$\ell_{21} = 1$	$r_{20} = 0$	$r_{21} = -1$	$r_{22} = -3$	$\boldsymbol{g}_{3(n-2)+2} = \boldsymbol{0}$

(14)

2.2 Solving the Linear System of Constraints

The D diagonal block has inverse D^{-1}, and the products $U_i = D^{-1}S_i$ are needed in the solving of the linear system (10),

$$D^{-1} = \begin{bmatrix} 3 & -2 & 1 \\ -3 & 3 & -2 \\ 1 & -1 & 1 \end{bmatrix}, \quad U_i = D^{-1}S_i = \begin{bmatrix} 2\sigma_i & -\sigma_i^2 & 0 \\ -3\sigma_i & 2\sigma_i^2 & 0 \\ \sigma_i & -\sigma_i^2 & 0 \end{bmatrix} \tag{15}$$

The linear system of equation (10) can be preprocessed by multiplying each block row by D^{-1}. The augmented matrix of the system is then

$$\left[\begin{array}{cccccccc|c} I & U_0 & O & O & \cdots & O & O & O & H_0 \\ O & I & U_1 & O & \cdots & O & O & O & H_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & O & I & U_{n-3} & H_{n-3} \\ L & O & O & O & \cdots & O & O & R & H_{n-2} \end{array}\right] \tag{16}$$

where I is the 3×3 identity matrix, $H_i = D^{-1}G_i$ for $0 \le i \le n-1$ and $H_{n-2} = G_{n-2}$. The products H_i are shown next but written as a column of vectors,

$$H_i = D^{-1}G_i = \begin{bmatrix} 3(\boldsymbol{f}_{i+1} - \boldsymbol{f}_i) \\ -3(\boldsymbol{f}_{i+1} - \boldsymbol{f}_i) \\ (\boldsymbol{f}_{i+1} - \boldsymbol{f}_i) \end{bmatrix} \tag{17}$$

for $0 \le i \le n-1$.

Row reductions can be applied to the augmented matrix to obtain an upper-triangular block matrix system.

The first operation multiplies the first block row by $-L$ and adds it to the last block row,

$$\left[\begin{array}{cccccccc|c} I & U_0 & O & O & \cdots & O & O & O & H_0 \\ O & I & U_1 & O & \cdots & O & O & O & H_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & O & I & U_{n-3} & H_{n-3} \\ O & -LU_0 & O & O & \cdots & O & O & R & H_{n-2} - LH_0 \end{array}\right] \tag{18}$$

The second operation multiplies the second block row by LU_0 and adds it to the last block row,

$$\left[\begin{array}{cccccccc|c} I & U_0 & O & O & \cdots & O & O & O & H_0 \\ O & I & U_1 & O & \cdots & O & O & O & H_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & O & I & U_{n-3} & H_{n-3} \\ O & O & LU_0U_1 & O & \cdots & O & O & R & H_{n-2} - LH_0 + LU_0H_1 \end{array}\right] \tag{19}$$

Additional block-row operations are applied, zeroing out the first nonzero block of the last row and causing the entry to the right of that block to be nonzero. The right-most column of the augmented matrix picks up another term. In equation (19), the first nonzero block of the last row is LU_0U_1. It is zeroed out and the block to the right becomes $-LU_0U_1U_2$. The right-most column becomes $H_{n-2} - LH_0 + LU_0H_1 - LU_0U_1H_2$. When the first nonzero block of the last row is in the entry to the left of Q, the block-row operation updates the Q value to a new value $\overline{Q}$. The right-most column of the augmented matrix is updated to $\overline{H}$,

$$\left[\begin{array}{cccccccc|c} I & U_0 & O & O & \cdots & O & O & O & H_0 \\ O & I & U_1 & O & \cdots & O & O & O & H_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & O & I & U_{n-3} & H_{n-3} \\ O & O & O & O & \cdots & O & O & \overline{R} & \overline{H} \end{array}\right] \tag{20}$$

where

$$\overline{R} = R + (-1)^n L \left(\prod_{i=0}^{n-3} U_i \right), \quad \overline{H} = H_{n-2} - L \sum_{i=0}^{n-3} \left((-1)^i \prod_{j=0}^{i-1} U_j \right) H_i \tag{21}$$

with the convention that $\prod_{j=0}^{-1} U_j = I$. The construction shows that during row reductions it is not necessary to actually read or write entries in the last block-row of the matrix except in the last two blocks.

The problem now is to compute $\overline{R}$ and $\overline{H}$, which in turn require computing products of the U_i. Two important observations are useful in this regard. First, the U_i matrices have zero-valued last columns, which implies that products of the U_i have zero-valued last columns; for example,

$$U_0U_1 = \begin{bmatrix} a_0 & b_0 & 0 \\ c_0 & d_0 & 0 \\ e_0 & f_0 & 0 \end{bmatrix} \begin{bmatrix} a_1 & b_1 & 0 \\ c_1 & d_1 & 0 \\ e_1 & f_1 & 0 \end{bmatrix} = \begin{bmatrix} a_0a_1 + b_0c_1 & a_0b_1 + b_0d_1 & 0 \\ c_0a_1 + d_0c_1 & c_0b_1 + d_0d_1 & 0 \\ e_0a_1 + f_0c_1 & e_0b_1 + f_0d_1 & 0 \end{bmatrix} \tag{22}$$

Second, the premultiplication by L eliminates the need to keep track of multiplications and additions involving the last-row entries of the product; specifically,

$$LU_0U_1 = \begin{bmatrix} 0 & 0 & 0 \\ \ell_{10} & 0 & 0 \\ 0 & \ell_{21} & 0 \end{bmatrix} \begin{bmatrix} a_0a_1 + b_0c_1 & a_0b_1 + b_0d_1 & 0 \\ c_0a_1 + d_0c_1 & c_0b_1 + d_0d_1 & 0 \\ e_0a_1 + f_0c_1 & e_0b_1 + f_0d_1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ \ell_{10}(a_0a_1 + b_0c_1) & \ell_{10}(a_0b_1 + b_0d_1) & 0 \\ \ell_{21}(c_0a_1 + d_0c_1) & \ell_{21}(c_0b_1 + d_0d_1) & 0 \end{bmatrix} \tag{23}$$

Therefore, a product involves computing at most 4 entries which are the 2×2 upper-left submatrix of the U-product which is then swapped into the 2×2 lower-left submatrix of L times the U-product. In the implementation, each of the two nonzero rows is processed separately depending on whether ℓ_{ij} is 0 or 1. This leads to more computational savings for clamped splines ($\ell_{10} = 1$ and $\ell_{21} = 0$) and for free splines ($\ell_{10} = 0$ and $\ell_{21} = 1$). Closed splines require both rows to be processed ($\ell_{10} = \ell_{21} = 1$).

Listing 1 contains pseudocode for computing $\overline{R}$ and H.

Listing 1. Pseudocode for computing the $\overline{R}$ and $\overline{H}$ of equation (21). The first RowReduce function uses the actual block matrices. The second RowReduce function is optimized based on the observation made previous about the occurrence of a large number of zero-valued entries.

```
// inputs: L, {U_0,...,U_{n-3}}, R, {H_0,...,H_{n-2}}
// output: Rbar, Hbar

void RowReduce(int n, Matrix<Real,3,3> L, array<Matrix<Real,3,3> U, Matrix<Real,3,3> R,
    array<Matrix<Real,3,m> H, Matrix<Real,3,3>& Rbar, Matrix<Real,3,m>& Hbar)
{
    Matrix<Real,3,3> LUProduct = L;
    Real sign = -1;
    Hbar = H[n-2];
    for (int i = 0; i <= n-3; ++i)
    {
        Hbar = sign * LUProduct * H[i];
        LUProduct = LUProduct * U[i];
        sign = -sign;
    }

    Rbar = R + sign * LUProduct;
}

// This is the optimized version of the row reduction. Each H_i is a 3-tuple of m-tuples B_{3i}, B_{3i+1} and B_{3i+2}.
// On output, R is replaced by Rbar and Hbar consists of B_{3(n-2)-3}, B_{3(n-2)-2} and B_{3(n-2)-1}.
void RowReduce(int n, int m, Real ell10, Real ell21, array<Real> delta,
    Matrix<Real,3,3>& R, array<Vector<Real,3>>& B)
{
      if (ell10 == 1)
      {
          Vector<Real, m>& Btarget = B[3*(n-2)-2];
          Btarget -= B[0];
          Real sigma0 = delta[0] / delta[1], sigma0sqr = sigma0 * sigma0;
          Real LUProduct0 = 2 * sigma0, LUProduct1 = -sigma0sqr;
          Real sign = -1;
          for (int i = 1; i <= n-3; ++i)
          {
              Btarget -= sign * (LUProduct0 * B[3*i] + LUProduct1 * B[3*i+1]);
              Real sigmai = delta[i] / delta[i+1], sigmaisqr = sigmai * sigmai;
              T temp0 = LUProduct0 * (2 * sigmai) + LUProduct1 * (-3 * sigmai);
              T temp1 = LUProduct0 * (-sigmaisqr) + LUProduct1 * (2 * sigmaisqr);
              LUProduct[0] = temp0;
              LUProduct[1] = temp1;
              sign = -sign;
          }
          R(1, 0) += sign * LUProduct0;
```

```
            R(1, 1) += sign * LUProduct1;
        }

        if (ell21 == 1)
        {
            Vector<Real, m>& Btarget = B[3*(n-2)-1];
            Btarget -= B[1];
            Real sigma0 = delta[0] / delta[1], sigma0sqr = sigma0 * sigma0;
            Real LUProduct0 = -3 * sigma0, LUProduct1 = 2 * sigma0sqr;
            Real sign = -1;
            for (int i = 1; i <= n-3; ++i)
            {
                Btarget -= sign * (LUProduct0 * B[3*i] + LUProduct1 * B[3*i+1]);
                Real sigmai = delta[i] / delta[i+1], sigmaisqr = sigmai * sigmai;
                T temp0 = LUProduct0 * (2 * sigmai) + LUProduct1 * (-3 * sigmai);
                T temp1 = LUProduct0 * (-sigmaisqr) + LUProduct1 * (2 * sigmaisqr);
                LUProduct0 = temp0;
                LUProduct1 = temp1;
                sign = -sign;
            }
            R(2, 0) += sign * LUProduct0;
            R(2, 1) += sign * LUProduct1;
        }
    }
```

Finally, back substitution can be applied to the upper-triangular linear system of equation (20). The constant terms $\boldsymbol{p}_{i,0}$ are set to $\boldsymbol{f}_i$ for all $0 \le i \le n-2$. The remaining coefficients are

$$\begin{aligned} P_{n-2} &= \overline{R}^{-1}\,\overline{H} \\ P_{n-3} &= H_{n-3} - U_{n-3}P_{n-2} \\ &\vdots \\ P_0 &= H_1 - U_1 P_1 \end{aligned} \tag{24}$$

2.3 An Example for a Scalar Function $F(t)$

The example is for the scalar case of $m = 1$. It is simple enough to create examples with $m > 1$. The results are those of the Geometric Tools code https://www.geometrictools.com/GTE/Mathematics/NaturalCubicSpline.h.

2.3.1 Free Spline

The example uses $n = 5$ samples,

t	f
0.000	−0.72904599140643900
0.200	+0.67001717998915900
0.452	+0.93773554224846278
0.611	−0.55793191403459019
1.000	−0.38366589898599346

(25)

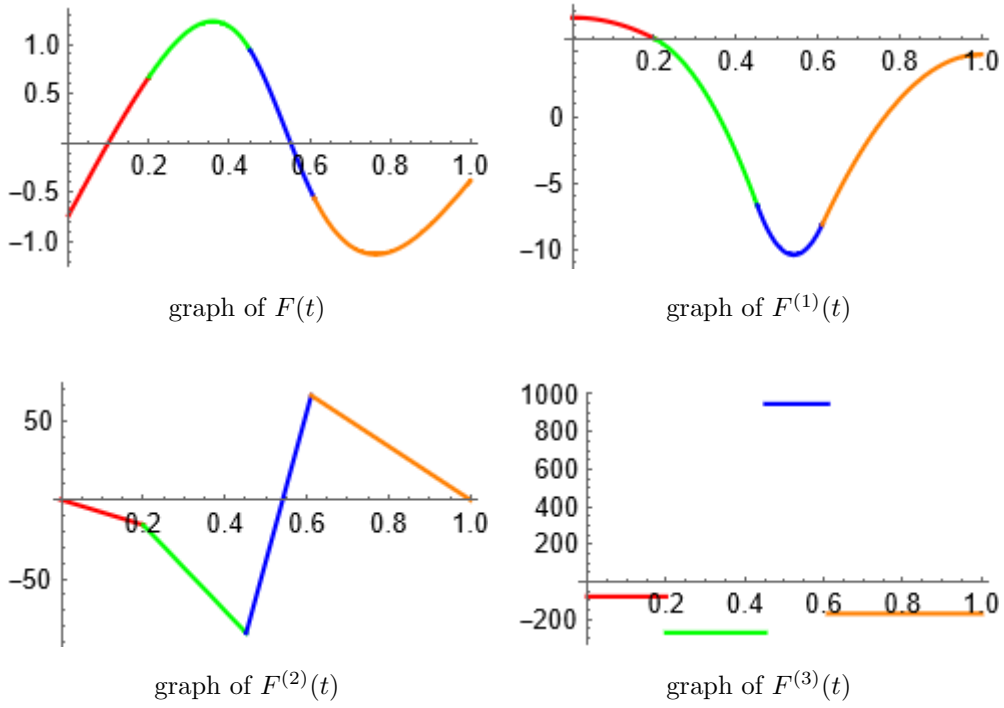
The free spline polynomials are $p_i(t) = \sum_{j=0}^{3} p_{i,j}(t-t_i)^j$ for $0 \le i \le 3$. The coefficients are shown truncated to 6 decimal places.

i	$p_{i,0}$	$p_{i,1}$	$p_{i,2}$	$p_{i,3}$
0	−0.729045	+1.504814	0.000000	−0.1057512
1	+0.670017	+1.496326	−0.503672	−0.7249359
2	+0.937735	−1.063675	−1.066305	+0.6343135
3	−0.557931	−3.164222	+5.007733	−1.6692444

(26)

Figure 1 shows the graphs of the cubic free spline function and its derivatives through order 3.

Figure 1. The graphs of the cubic free spline function and its derivatives through order 3 are shown here, drawn using Mathematica [3].



The graphs visually verify the continuity of the spline function and its derivatives through order 2.

2.3.2 Clamped Spline

The example uses the same 5 samples as those for free splines; see equation (25). Additionally, the first-order derivatives are specified at the endpoints. These are $F^{(1)}(t_0) = -0.987$ and $F^{(1)}(t_4) = +0.654$. The clamped spline polynomials are $p_i(t) = \sum_{j=0}^{3} p_{i,j}(t-t_i)^j$ for $0 \le i \le 3$. The coefficients are shown truncated to 6

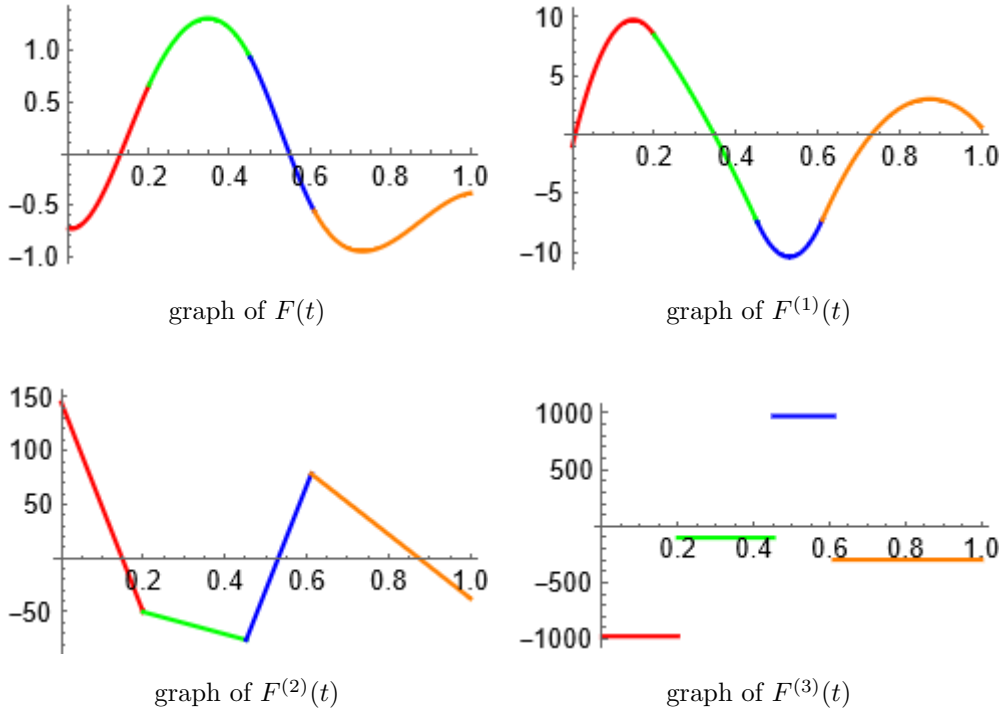
decimal places.

i	$p_{i,0}$	$p_{i,1}$	$p_{i,2}$	$p_{i,3}$
0	−0.729045	−0.987000	+4.246877	−1.860814
1	+0.670017	+2.424633	−2.120343	−0.036571
2	+0.937735	−1.215068	−0.887788	+0.607189
3	−0.557931	−2.860196	+5.589190	−2.554728

(27)

Figure 2 shows the graphs of the cubic clamped spline function and its derivatives through order 3.

Figure 2. The graphs of the cubic clamped spline function and its derivatives through order 3 are shown here, drawn using Mathematica [3].



The graphs visually verify the continuity of the spline function and its derivatives through order 2.

2.3.3 Closed Spline

The example uses $n = 5$ samples,

t	f
0.000	−0.72904599140643900
0.200	+0.67001717998915900
0.452	+0.93773554224846278
0.611	−0.55793191403459019
1.000	−0.72904599140643900

(28)

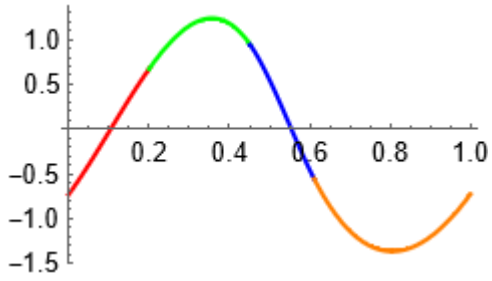
As required, the first and last function values are the same: $f_4 = f_0$. The closed spline polynomials are $p_i(t) = \sum_{j=0}^{3} p_{i,j}(t - t_i)^j$ for $0 \le i \le 3$. The coefficients are shown truncated to 6 decimal places.

i	$p_{i,0}$	$p_{i,1}$	$p_{i,2}$	$p_{i,3}$
0	−0.729045	+1.435994	+0.121235	−0.158167
1	+0.670017	+1.516994	−0.560845	−0.688430
2	+0.937735	−1.053683	−1.045467	+0.603483
3	−0.557931	−3.264096	+4.578855	−1.485873

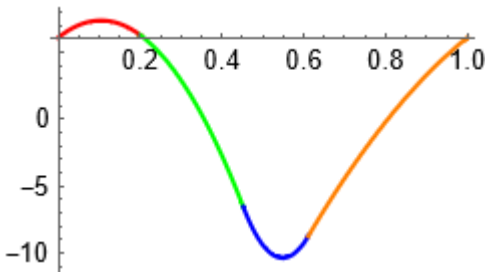
(29)

Figure 3 shows the graphs of the cubic closed spline function and its derivatives through order 3.

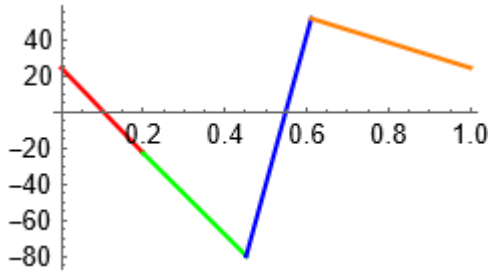
Figure 3. The graphs of the cubic closed spline function and its derivatives through order 3 are shown here, drawn using Mathematica [3].



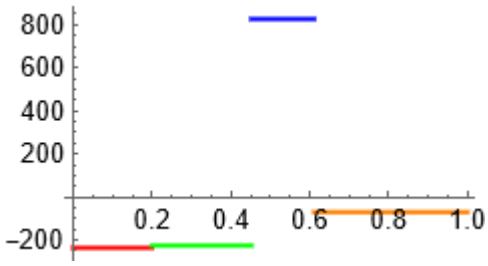
graph of $F(t)$



graph of $F^{(1)}(t)$



graph of $F^{(2)}(t)$



graph of $F^{(3)}(t)$

The graphs visually verify the continuity of the spline function and its derivatives through order 2.

3 Quintic Splines

The degree is $5 = d = 2k + 1$, so $k = 2$. The n samples are $(t_i, \boldsymbol{f}_i, \boldsymbol{f}_i^{(1)})$ for $0 \le i \le n-1$. The number of polynomials is $n-1$, so we have $6(n-1)$ coefficients to construct. The samples themselves generate the constraints

$$\begin{aligned} \boldsymbol{P}_i(t_i) &= \boldsymbol{f}_i \quad \text{for } 0 \le i \le n-2 \\ \boldsymbol{P}_{n-2}(t_{n-1}) &= \boldsymbol{f}_{n-1} \\ \boldsymbol{P}_i^{(1)}(t_i) &= \boldsymbol{f}_i^{(1)} \quad \text{for } 0 \le i \le n-2 \\ \boldsymbol{P}_{n-2}^{(1)}(t_{n-1}) &= \boldsymbol{f}_{n-1}^{(1)} \end{aligned} \tag{30}$$

which consists of $2n$ constraints. To obtain C^3 continuity, the polynomials and their derivatives must match a the interior samples,

$$\begin{aligned}
\boldsymbol{P}_i(t_{i+1}) &= \boldsymbol{P}_{i+1}(t_{i+1}) \quad \text{for } 0 \le i \le n-3 \\
\boldsymbol{P}_i^{(1)}(t_{i+1}) &= \boldsymbol{P}_{i+1}^{(1)}(t_{i+1}) \quad \text{for } 0 \le i \le n-3 \\
\boldsymbol{P}_i^{(2)}(t_{i+1}) &= \boldsymbol{P}_{i+1}^{(2)}(t_{i+1}) \quad \text{for } 0 \le i \le n-3 \\
\boldsymbol{P}_i^{(3)}(t_{i+1}) &= \boldsymbol{P}_{i+1}^{(3)}(t_{i+1}) \quad \text{for } 0 \le i \le n-3
\end{aligned} \tag{31}$$

which consists of $4(n-2)$ constraints. The number of constraints so far is $2n + 4(n-2) = 6(n-1) - 2$ constraints. Two additional constraints are needed to reach $6(n-1)$ constraints in the $6(n-1)$ unknowns. Three standard choices are

clamped spline:	$\boldsymbol{P}_0^{(2)}(t_0) = \boldsymbol{c}_0$ and $\boldsymbol{P}_{n-2}^{(2)}(t_{n-1}) = \boldsymbol{c}_1$ where $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ are specified
free spline:	$\boldsymbol{P}_0^{(3)}(t_0) = \boldsymbol{0}$ and $\boldsymbol{P}_{n-2}^{(3)}(t_{n-1}) = \boldsymbol{0}$
closed spline:	$\boldsymbol{f}_{n-1} = \boldsymbol{f}_0$, $\boldsymbol{f}_{n-1}^{(1)} = \boldsymbol{f}_0^{(1)}$, $\boldsymbol{P}_0^{(2)}(t_0) = \boldsymbol{P}_{n-2}^{(2)}(t_{n-1})$ and $\boldsymbol{P}_0^{(3)}(t_0) = \boldsymbol{P}_{n-2}^{(3)}(t_{n-1})$

3.1 Setting up the Linear System of Constraints

The polynomials and their relevant derivatives are

$$\begin{aligned}
\boldsymbol{P}_i(t) &= \boldsymbol{p}_{i,0} + \boldsymbol{p}_{i,1}\left(\tfrac{t-t_i}{\Delta_i}\right) + \boldsymbol{p}_{i,2}\left(\tfrac{t-t_i}{\Delta_i}\right)^2 + \boldsymbol{p}_{i,3}\left(\tfrac{t-t_i}{\Delta_i}\right)^3 + \boldsymbol{p}_{i,4}\left(\tfrac{t-t_i}{\Delta_i}\right)^4 + \boldsymbol{p}_{i,5}\left(\tfrac{t-t_i}{\Delta_i}\right)^5 \\
\boldsymbol{P}_i^{(1)}(t) &= \left(\boldsymbol{p}_{i,1} + 2\boldsymbol{p}_{i,2}\left(\tfrac{t-t_i}{\Delta_i}\right) + 3\boldsymbol{p}_{i,3}\left(\tfrac{t-t_i}{\Delta_i}\right)^2 + 4\boldsymbol{p}_{i,4}\left(\tfrac{t-t_i}{\Delta_i}\right)^3 + 5\boldsymbol{p}_{i,5}\left(\tfrac{t-t_i}{\Delta_i}\right)^4\right)/\Delta_i \\
\boldsymbol{P}_i^{(2)}(t) &= \left(2\boldsymbol{p}_{i,2} + 6\boldsymbol{p}_{i,3}\left(\tfrac{t-t_i}{\Delta_i}\right) + 12\boldsymbol{p}_{i,4}\left(\tfrac{t-t_i}{\Delta_i}\right)^2 + 20\boldsymbol{p}_{i,5}\left(\tfrac{t-t_i}{\Delta_i}\right)^3\right)/\Delta_i^2 \\
\boldsymbol{P}_i^{(3)}(t) &= \left(6\boldsymbol{p}_{i,3} + 24\boldsymbol{p}_{i,4}\left(\tfrac{t-t_i}{\Delta_i}\right) + 60\boldsymbol{p}_{i,5}\left(\tfrac{t-t_i}{\Delta_i}\right)^2\right)/\Delta_i^3
\end{aligned} \tag{32}$$

for $0 \le i \le n-2$. The constraints of the first and third lines of equation (30) imply

$$\begin{aligned}
\boldsymbol{p}_{i,0} &= \boldsymbol{f}_i \quad \text{for } 0 \le i \le n-2 \\
\boldsymbol{p}_{i,1} &= \Delta_i \boldsymbol{f}_i^{(1)} \quad \text{for } 0 \le i \le n-2
\end{aligned} \tag{33}$$

The constraints of equation (31) and the second and fourth lines of equation (30) imply

$$\begin{aligned}
\boldsymbol{p}_{i,2} + \boldsymbol{p}_{i,3} + \boldsymbol{p}_{i,4} + \boldsymbol{p}_{i,5} &= \boldsymbol{f}_{i+1} - \boldsymbol{f}_i - \Delta_i \boldsymbol{f}_i^{(1)} && \text{for } 0 \le i \le n-2 \\
2\boldsymbol{p}_{i,2} + 3\boldsymbol{p}_{i,3} + 4\boldsymbol{p}_{i,4} + 5\boldsymbol{p}_{i,5} &= \Delta_i(\boldsymbol{f}_{i+1}^{(1)} - \boldsymbol{f}_i^{(1)}) && \text{for } 0 \le i \le n-3 \\
\boldsymbol{p}_{i,2} + 3\boldsymbol{p}_{i,3} + 6\boldsymbol{p}_{i,4} + 10\boldsymbol{p}_{i,5} &= \sigma_i^2\, \boldsymbol{p}_{i+1,2} && \text{for } 0 \le i \le n-3 \\
\boldsymbol{p}_{i,3} + 4\boldsymbol{p}_{i,4} + 10\boldsymbol{p}_{i,5} &= \sigma_i^3\, \boldsymbol{p}_{i+1,3} && \text{for } 0 \le i \le n-3
\end{aligned} \tag{34}$$

where $\sigma_i = \Delta_i/\Delta_{i+1}$. Choosing a free, clamped or closed spline, a linear system of $6(n-1)$ equations in $6(n-1)$ unknowns can be created from these constraints.

Equation (33) defines the constant and linear terms of the polynomials. The linear system for the remaining coefficients is constructed from the constraints of equation (34). The system has $4(n-1)$ equations in $4(n-1)$ unknowns and can be written in block-matrix form,

$$\begin{bmatrix} D & S_0 & O & O & \cdots & O & O & O \\ O & D & S_1 & O & \cdots & O & O & O \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & O & D & S_{n-3} \\ L & O & O & O & \cdots & O & O & R \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_{n-3} \\ P_{n-2} \end{bmatrix} = \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_{n-3} \\ G_{n-2} \end{bmatrix} \tag{35}$$

The blocks D, S_i, O, L and R are 4×4. The blocks P_i and G_i are $5 \times m$, each row representing a $1 \times m$ vector. The following blocks are independent of whether the spline is clamped, free or closed,

$$D = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 4 & 5 \\ 1 & 3 & 6 & 10 \\ 0 & 1 & 4 & 10 \end{bmatrix}, \quad S_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\sigma_i^2 & 0 & 0 & 0 \\ 0 & -\sigma_i^3 & 0 & 0 \end{bmatrix}, \quad O = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{36}$$

and

$$P_i = \begin{bmatrix} \boldsymbol{p}_{i,2} \\ \boldsymbol{p}_{i,3} \\ \boldsymbol{p}_{i,4} \\ \boldsymbol{p}_{i,5} \end{bmatrix} \text{ for } 0 \le i \le n-2, \quad G_i = \begin{bmatrix} \boldsymbol{g}_{4i} \\ \boldsymbol{g}_{4i+1} \\ \boldsymbol{g}_{4i+2} \\ \boldsymbol{g}_{4i+3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_{i+1} - \boldsymbol{f}_i - \Delta_i \boldsymbol{f}_i^{(1)} \\ \Delta_i (\boldsymbol{f}_{i+1}^{(1)} - \boldsymbol{f}_i^{(1)}) \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \text{ for } 0 \le i \le n-3 \tag{37}$$

The following blocks are dependent on the type of spline,

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \ell_{20} & 0 & 0 & 0 \\ 0 & \ell_{31} & 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 4 & 5 \\ r_{20} & r_{21} & r_{22} & r_{23} \\ r_{30} & r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad G_{n-2} = \begin{bmatrix} \boldsymbol{g}_{4(n-2)} \\ \boldsymbol{g}_{4(n-2)+1} \\ \boldsymbol{g}_{4(n-2)+2} \\ \boldsymbol{g}_{4(n-2)+3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_{n-1} - \boldsymbol{f}_{n-2} - \Delta_{n-2} \boldsymbol{f}_{n-2}^{(1)} \\ \Delta_{n-2} (\boldsymbol{f}_{n-1}^{(1)} - \boldsymbol{f}_{n-2}^{(1)}) \\ \boldsymbol{g}_{4(n-2)+2} \\ \boldsymbol{g}_{4(n-2)+3} \end{bmatrix} \tag{38}$$

The specific values for the matrix entries are

type	L	R				G
clamped	$\ell_{20} = 1$	$r_{20} = 0$	$r_{21} = 0$	$r_{22} = 0$	$r_{23} = 0$	$\boldsymbol{g}_{4(n-2)+2} = \Delta_0^2 \boldsymbol{c}_0/2$
	$\ell_{31} = 0$	$r_{30} = 1$	$r_{31} = 3$	$r_{32} = 6$	$r_{33} = 10$	$\boldsymbol{g}_{4(n-2)+3} = \Delta_{n-2}^2 \boldsymbol{c}_1/2$
free	$\ell_{20} = 0$	$r_{20} = 0$	$r_{21} = 1$	$r_{22} = 4$	$r_{23} = 10$	$\boldsymbol{g}_{4(n-2)+2} = \boldsymbol{0}$
	$\ell_{31} = 1$	$r_{30} = 0$	$r_{31} = 0$	$r_{32} = 0$	$r_{33} = 0$	$\boldsymbol{g}_{4(n-2)+3} = \boldsymbol{0}$
closed	$\ell_{20} = 1$	$r_{20} = -1$	$r_{21} = -3$	$r_{22} = -6$	$r_{23} = -10$	$\boldsymbol{g}_{4(n-2)+2} = \boldsymbol{0}$
	$\ell_{31} = 1$	$r_{30} = 0$	$r_{31} = -1$	$r_{32} = -4$	$r_{33} = -10$	$\boldsymbol{g}_{4(n-2)+3} = \boldsymbol{0}$

(39)

3.2 Solving the Linear System of Constraints

The D diagonal block has inverse D^{-1}, and the products $U_i = D^{-1}S_i$ are needed in the solving of the linear system (35),

$$D^{-1} = \begin{bmatrix} 10 & -6 & 3 & -1 \\ -20 & 14 & -8 & 3 \\ 15 & -11 & 7 & -3 \\ -4 & 3 & -2 & 1 \end{bmatrix}, \quad U_i = D_i^{-1}S = \begin{bmatrix} -3\sigma_i^2 & \sigma_i^3 & 0 & 0 \\ 8\sigma_i^2 & -3\sigma_i^3 & 0 & 0 \\ -7\sigma_i^2 & 3\sigma_i^3 & 0 & 0 \\ 2\sigma_i^2 & -\sigma_i^3 & 0 & 0 \end{bmatrix} \tag{40}$$

where U_i is defined to be the product indicated in the equation.

The linear system of equation (35) can be preprocessed by multiplying each block row by the appropriate inverse of the D-matrix. The augmented matrix of the system is then

$$\left[\begin{array}{cccccccc|c} I & U_0 & O & O & \cdots & O & O & O & H_0 \\ O & I & U_1 & O & \cdots & O & O & O & H_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & O & I & U_{n-3} & H_{n-3} \\ L & O & O & O & \cdots & O & O & R & H_{n-2} \end{array}\right] \tag{41}$$

where I is the 4×4 identity matrix, $H_i = D^{-1}G_i$ for $0 \le i \le n-1$ and $H_{n-2} = G_{n-2}$. The products H_i are shown next but written as a column of vectors,

$$H_i = D_i^{-1}G_i = \begin{bmatrix} 10(\boldsymbol{f}_{i+1} - \boldsymbol{f}_i - \Delta_i \boldsymbol{f}_i^{(1)}) & - & 6(\Delta_i(\boldsymbol{f}_{i+1}^{(1)} - \boldsymbol{f}_i^{(1)})) \\ -20(\boldsymbol{f}_{i+1} - \boldsymbol{f}_i - \Delta_i \boldsymbol{f}_i^{(1)}) & + & 14(\Delta_i(\boldsymbol{f}_{i+1}^{(1)} - \boldsymbol{f}_i^{(1)})) \\ 15(\boldsymbol{f}_{i+1} - \boldsymbol{f}_i - \Delta_i \boldsymbol{f}_i^{(1)}) & - & 11(\Delta_i(\boldsymbol{f}_{i+1}^{(1)} - \boldsymbol{f}_i^{(1)})) \\ -4(\boldsymbol{f}_{i+1} - \boldsymbol{f}_i - \Delta_i \boldsymbol{f}_i^{(1)}) & + & 3(\Delta_i(\boldsymbol{f}_{i+1}^{(1)} - \boldsymbol{f}_i^{(1)})) \end{bmatrix} \tag{42}$$

for $0 \le i \le n-1$.

Using the same method of row reduction as in the cubic case of equation (20), we obtain

$$\left[\begin{array}{cccccccc|c} I & U_0 & O & O & \cdots & O & O & O & H_0 \\ O & I & U_1 & O & \cdots & O & O & O & H_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ O & O & O & O & \cdots & O & I & U_{n-3} & H_{n-3} \\ O & O & O & O & \cdots & O & O & \overline{R} & \overline{H} \end{array}\right] \tag{43}$$

where

$$\overline{R} = R + (-1)^n L \left(\prod_{i=0}^{n-3} U_i \right), \quad \overline{H} = H_{n-2} - L \sum_{i=0}^{n-3} \left((-1)^i \prod_{j=0}^{i-1} U_j \right) H_i \tag{44}$$

with the convention that $\prod_{j=0}^{-1} U_j = I$. The construction shows that during row reductions it is not necessary to actually read or write entries in the last block-row of the matrix except in the last two blocks.

The problem now is to compute $\overline{R}$ and $\overline{H}$, which in turn require computing products of the U_i. Two important observations are useful in this regard. First, the last two columns of the U_i matrices are zero, which implies that products of the U_i have zero-valued last two columns; for example,

$$U_0U_1 = \begin{bmatrix} a_0 & b_0 & 0 & 0 \\ c_0 & d_0 & 0 & 0 \\ e_0 & f_0 & 0 & 0 \\ g_0 & h_0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 & b_1 & 0 & 0 \\ c_1 & d_1 & 0 & 0 \\ e_1 & f_1 & 0 & 0 \\ g_1 & h_1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} a_0a_1 + b_0c_1 & a_0b_1 + b_0d_1 & 0 & 0 \\ c_0a_1 + d_0c_1 & c_0b_1 + d_0d_1 & 0 & 0 \\ e_0a_1 + f_0c_1 & e_0b_1 + f_0d_1 & 0 & 0 \\ g_0a_1 + h_0c_1 & g_0b_1 + h_0d_1 & 0 & 0 \end{bmatrix} \tag{45}$$

Second, the premultiplication by L eliminates the need to keep track of multiplications and additions involving the last-row entries of the product; specifically,

$$LU_0U_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \ell_{20} & 0 & 0 & 0 \\ 0 & \ell_{31} & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0a_1 + b_0c_1 & a_0b_1 + b_0d_1 & 0 & 0 \\ c_0a_1 + d_0c_1 & c_0b_1 + d_0d_1 & 0 & 0 \\ e_0a_1 + f_0c_1 & e_0b_1 + f_0d_1 & 0 & 0 \\ g_0a_1 + h_0c_1 & g_0b_1 + h_0d_1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \ell_{20}(a_0a_1 + b_0c_1) & \ell_{20}(a_0b_1 + b_0d_1) & 0 & 0 \\ \ell_{31}(c_0a_1 + d_0c_1) & \ell_{31}(c_0b_1 + d_0d_1) & 0 & 0 \end{bmatrix} \tag{46}$$

Therefore, a product involves computing at most 4 entries which are the 2×2 upper-left submatrix of the U-product which is then swapped into the 2×2 lower-left submatrix of L times the U-product. In the implementation, each of the two nonzero rows is processed separately depending on whether ℓ_{ij} is 0 or 1. This leads to more computational savings for clamped splines ($\ell_{20} = 1$ and $\ell_{31} = 0$) and for free splines ($\ell_{20} = 0$ and $\ell_{31} = 1$). Closed splines require both rows to be processed ($\ell_{20} = \ell_{31} = 1$).

Listing 2 contains pseudocode for computing $\overline{R}$ and H.

Listing 2. Pseudocode for computing the $\overline{R}$ and $\overline{H}$ of equation (44). The first RowReduce function uses the actual block matrices. The second RowReduce function is optimized based on the observation made previous about the occurrence of a large number of zero-valued entries.

```
// inputs: L, {U_0,...,U_{n-3}}, R, {H_0,...,H_{n-2}}
// output: Rbar, Hbar

void RowReduce(int n, Matrix<Real,4,4> L, array<Matrix<Real,4,4> U, Matrix<Real,4,4> R,
    array<Matrix<Real,4,m> H, Matrix<Real,4,4>& Rbar, Matrix<Real,4,m>& Hbar)
{
    Matrix<Real,4,4> LUProduct = L;
    Real sign = -1;
    Hbar = H[n-2];
    for (int i = 0; i <= n-3; ++i)
    {
        Hbar = sign * LUProduct * H[i];
        LUProduct = LUProduct * U[i];
        sign = -sign;
    }

    Rbar = R + sign * LUProduct;
}

// This is the optimized version of the row reduction. Each H_i is a 4-tuple of m-tuples B_{4i}, B_{4i+1}, B_{4i+2} and B_{4i+3}.
// On output, R is replaced by Rbar and Hbar consists of B_{4(n-2)-4}, B_{4(n-2)-3}, B_{4(n-2)-2} and B_{4(n-2)-1}.
void RowReduce(int n, int m, Real ell20, Real ell31, array<Real> delta,
```

```
    Matrix<Real,4,4>& Q, array<Vector<Real,4>>& B)
{
    if (ell20 == 1)
    {
        Vector<Real, m>& Btarget = B[4*(n-2)-2];
        Btarget -= B[0];
        Real sigma0 = delta[0] / delta[1], sigma0sqr = sigma0 * sigma0;
        Real LUProduct0 = -3 & sigma0, LUProduct1 = sigma0sqr;
        Real sign = -1;
        for (int i = 1; i <= n-3; ++i)
        {
            Btarget -= sign * (LUProduct0 * B[4*i] + LUProduct1 * B[4*i+1]);
            Real sigmai = delta[i] / delta[i+1], sigmaisqr = sigmai * sigmai;
            T temp0 = LUProduct0 * (-3 * sigmai) + LUProduct1 * (8 * sigmai);
            T temp1 = LUProduct0 * (sigmaisqr) + LUProduct1 * (-3 * sigmaisqr);
            LUProduct[0] = temp0;
            LUProduct[1] = temp1;
            sign = -sign;
        }
        R(2, 0) += sign * LUProduct0;
        R(2, 1) += sign * LUProduct1;
    }

    if (ell31 == 1)
    {
        Vector<Real, m>& Btarget = B[4*(n-2)-1];
        Btarget -= B[1];
        Real sigma0 = delta[0] / delta[1], sigma0sqr = sigma0 * sigma0;
        Real LUProduct0 = 8 * sigma0, LUProduct1 = -3 * sigma0sqr;
        Real sign = -1;
        for (int i = 1; i <= n-3; ++i)
        {
            Btarget -= sign * (LUProduct0 * B[4*i] + LUProduct1 * B[4*i+1]);
            Real sigmai = delta[i] / delta[i+1], sigmaisqr = sigmai * sigmai;
            T temp0 = LUProduct0 * (-3 * sigmai) + LUProduct1 * (8 * sigmai);
            T temp1 = LUProduct0 * (sigmaisqr) + LUProduct1 * (-3 * sigmaisqr);
            LUProduct0 = temp0;
            LUProduct1 = temp1;
            sign = -sign;
        }
        R(3, 0) += sign * LUProduct0;
        R(3, 1) += sign * LUProduct1;
    }
}
```

Finally, back substitution can be applied to the upper-triangular linear system of equation (43). The constant terms $\boldsymbol{p}_{i,0}$ are set to $\boldsymbol{f}_i$ and the linear terms $\boldsymbol{p}_{i,1}$ are set to $\boldsymbol{f}_i^{(1)}$ for all $0 \le i \le n-2$. The remaining coefficients are

$$\begin{aligned} P_{n-2} &= \overline{R}^{-1}\,\overline{H} \\ P_{n-3} &= H_{n-3} - U_{n-3}P_{n-2} \\ &\vdots \\ P_0 &= H_1 - U_1 P_1 \end{aligned} \tag{47}$$

3.3 An Example for a Scalar Function $F(t)$

The example is for the scalar case of $m = 1$. It is simple enough to create examples with $m > 1$. The results are those of the Geometric Tools code https://www.geometrictools.com/GTE/Mathematics/NaturalQuinticSpline.h.

3.3.1 Free Spline

The example uses $n = 5$ samples,

t	f	f'
0.000	−0.72904599140643900	−0.77507096788763941
0.200	+0.67001717998915900	0.27952671419630559
0.452	+0.93773554224846278	0.75686129079768771
0.611	−0.55793191403459019	−0.0073253554103394070
1.000	−0.38366589898599346	−0.59585723032045212

(48)

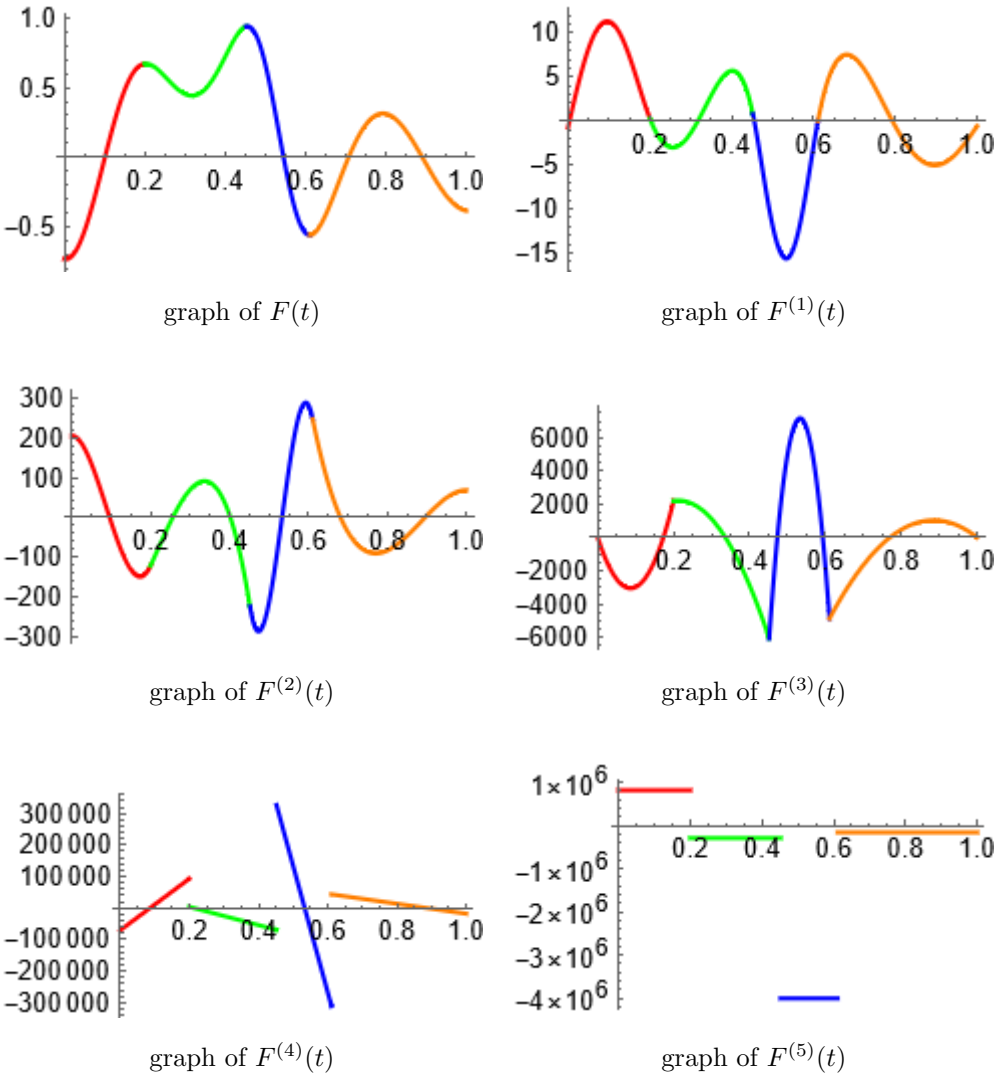
The free spline polynomials are $p_i(t) = \sum_{j=0}^{5} p_{i,j}(t - t_i)^j$ for $0 \le i \le 3$. The coefficients are shown truncated to 6 decimal places.

i	$p_{i,0}$	$p_{i,1}$	$p_{i,2}$	$p_{i,3}$	$p_{i,4}$	$p_{i,5}$
0	−0.729045	−0.155014	+4.094487	0.000000	−4.723995	+2.183585
1	+0.670017	+0.070440	−3.831883	+5.880846	+0.600056	−2.451742
2	+0.937735	+0.120340	−2.829074	−4.078290	+8.685269	−3.393912
3	−0.557931	−0.002849	+18.606397	−47.978412	+41.252148	−11.703018

(49)

Figure 4 shows the graphs of the quintic free spline function and its derivatives through order 5.

Figure 4. The graphs of the quintic free spline function and its derivatives through order 5 are shown here, drawn using Mathematica [3].



The graphs visually verify the continuity of the spline function and its derivatives through order 3.

3.3.2 Clamped Spline

The example uses the same 5 samples as those for free splines; see equation (48). Additionally, the second-order derivatives are specified at the endpoints. These are $F^{(2)}(t_0) = -0.987$ and $F^{(2)}(t_4) = +0.654$. The clamped spline polynomials are $p_i(t) = \sum_{j=0}^{5} p_{i,j}(t-t_i)^j$ for $0 \le i \le 3$. The coefficients are shown truncated

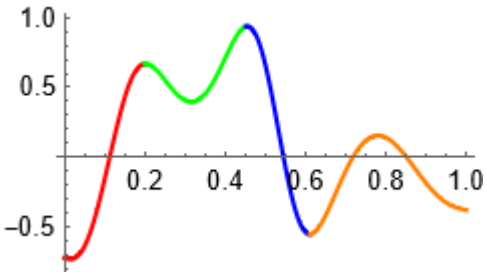
to 6 decimal places.

i	$p_{i,0}$	$p_{i,1}$	$p_{i,2}$	$p_{i,3}$	$p_{i,4}$	$p_{i,5}$
0	−0.729045	−0.155014	−0.019740	+11.558280	−15.497874	+5.513410
1	+0.670017	+0.070440	−5.077199	+9.403552	−2.709405	−1.419669
2	+0.937735	+0.120340	−2.913967	−3.926177	+8.635720	−3.411584
3	−0.557931	−0.002849	+17.992492	−51.241085	+49.619208	−16.193500

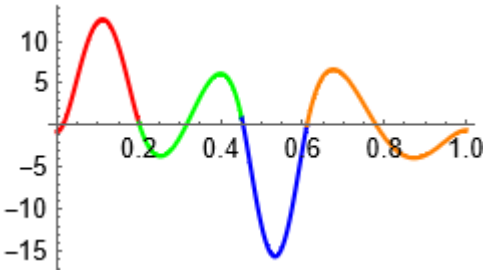
(50)

Figure 5 shows the graphs of the quintic clamped spline function and its derivatives through order 5.

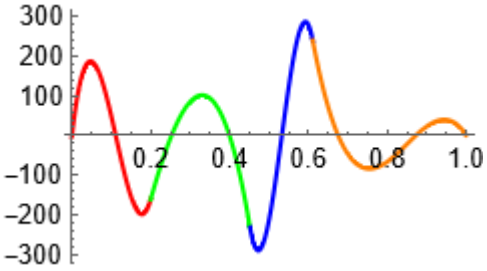
Figure 5. The graphs of the quintic clamped spline function and its derivatives through order 5 are shown here, drawn using Mathematica [3].



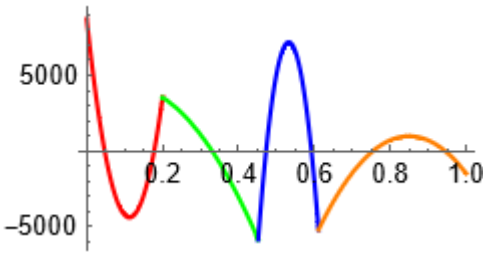
graph of $F(t)$



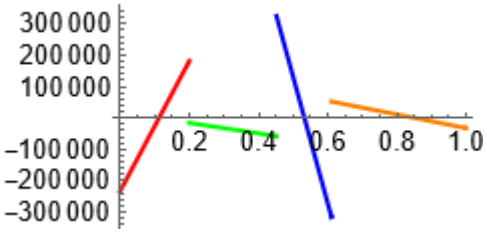
graph of $F^{(1)}(t)$



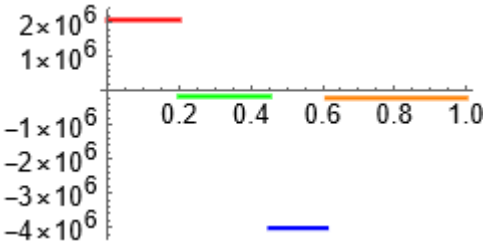
graph of $F^{(2)}(t)$



graph of $F^{(3)}(t)$



graph of $F^{(4)}(t)$



graph of $F^{(5)}(t)$

The graphs visually verify the continuity of the spline function and its derivatives through order 3.

3.3.3 Closed Spline

The example uses $n = 5$ samples,

t	f	f'
0.000	−0.72904599140643900	−0.77507096788763941
0.200	+0.67001717998915900	0.27952671419630559
0.452	+0.93773554224846278	0.75686129079768771
0.611	−0.55793191403459019	−0.0073253554103394070
1.000	−0.72904599140643900	−0.77507096788763941

(51)

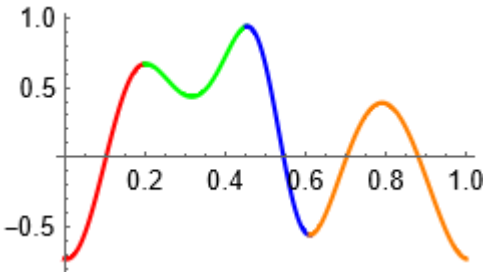
As required, the first and last function values and first derivative values are the same: $f_4 = f_0$, $f_4^{(1)} = f_0^{(1)}$. The closed spline polynomials are $p_i(t) = \sum_{j=0}^{5} p_{i,j}(t - t_i)^j$ for $0 \le i \le 3$. The coefficients are shown truncated to 6 decimal places.

i	$p_{i,0}$	$p_{i,1}$	$p_{i,2}$	$p_{i,3}$	$p_{i,4}$	$p_{i,5}$
0	−0.729045	−0.155014	+3.233658	+2.422474	−6.986456	+2.884401
1	+0.670017	+0.070440	−4.085920	+6.642567	−0.161272	−2.198096
2	+0.937735	+0.120340	−2.829230	−4.014775	+8.558706	−3.330708
3	−0.557931	−0.002849	+18.983772	−45.206341	+32.918695	−6.864390

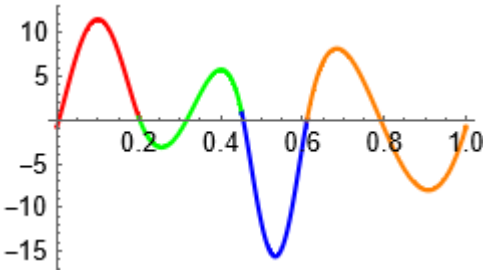
(52)

Figure 6 shows the graphs of the quintic closed spline function and its derivatives through order 5.

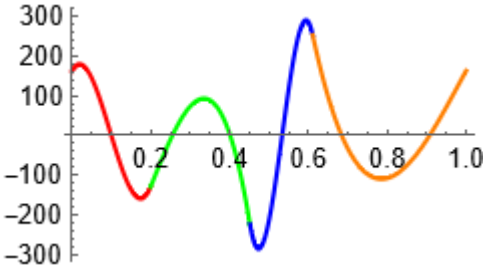
Figure 6. The graphs of the quintic closed spline function and its derivatives through order 3 are shown here, drawn using Mathematica [3].



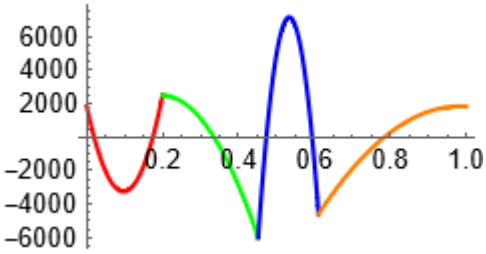
graph of $F(t)$



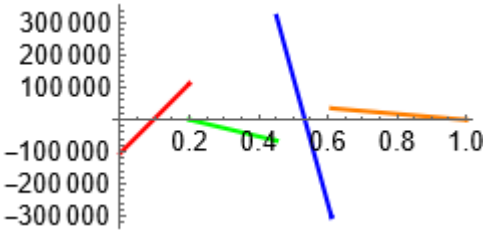
graph of $F^{(1)}(t)$



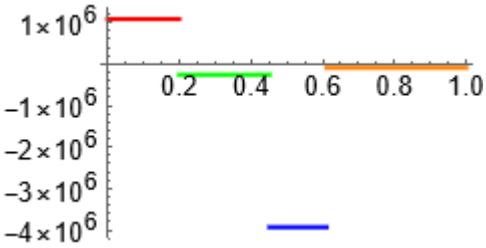
graph of $F^{(2)}(t)$



graph of $F^{(3)}(t)$



graph of $F^{(4)}(t)$



graph of $F^{(5)}(t)$

The graphs visually verify the continuity of the spline function and its derivatives through order 3.

4 Splines of Larger Odd Degree

The analysis for natural cubic and quintic splines shows there are patterns that occur, allowing for extension to larger degrees. Let the degree be $d = 2k + 1$ for $k \geq 1$.

The matrix system is constructed similar to that of equation (10). The D, S_i,), L and R are $(k+2)\times(k+2)$ block matrices. The P_i and G_i are $(k+2)\times m$ matrices.

The pattern for the D matrices is that the elements in each column are the first part of a row of Pascal's triangle. For example, the quintic D has elements in column 0 of $(1, 2, 1)$. The coefficients in column 1 are $(1, 3, 3, 1)$. The elements of column 2 are $(1, 4, 6, 4)$ where the final 1 of the row of Pascal's triangle is ignored. The elements of column 3 are $(1, 5, 10, 10)$ where the final $(5, 1)$ of the row of Pascal's triangle is ignored.

The matrix) has all zero elements. The matrix S_i has all zero rows except for the last 2 rows. The next to last row has leading element $-\sigma_i^k$ and all other elements 0. The last row has leading element 0 followed by $-\sigma_i^{k+1}$ and then followed by all zeros.

Eacj G_i matrix has k nonzero rows. Each row is a difference between $\boldsymbol{f}_{i+1}^{(j)}$ and an expression which is the first part of a polynomial with coefficients involving the $\boldsymbol{f}$ derivatives for index i multiplied by appropriate powers of Δ_i. The last 2 rows of each G_i are the zero vector except for G_{n-2} that has nonzero vectors that occur when the spline is clamped.

The matrix L has all zero entries except for 1 or 2 elements. A clamped spline has $L_{k,0} = 1$ and $L_{k+1,1} = 0$. A free spline has $L_{k,0} = 0$ and $L_{k+1,1} = 1$. A closed spline has $L_{k,0} = L_{k+1,1} = 1$.

The matrix R has k rows of integer values. These come from the coefficients of the last polynomial and its derivatives. The final 2 rows depend on the spline type. The elements of these rows can be determined by writing the constraints for the particular spline type and associating the relevant coefficients with the Q-entries.

For small degrees d, the inverse D can be computed symbolically. As the degree increases, the symbolic expression becomes unwieldy. In this case, numerical inversion is required. The matrices H_i are also complicated to code symbolically for large d, but these can also be computed numerically.

The expressions for $\overline{R}$ and $\overline{H}$ are the same as those for cubic and quintic splines.

The code for row reductions and for back substitution does not increase in complexity.

References

[1] Richard L. Burden, J. Douglas Faires, and Annette M. Burden. *Numerical Analysis*. Cengage Learning, Boston, MA, 10th edition, 2015.

[2] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, United Kingdom, 2nd edition, 1992.

[3] Wolfram Research, Inc. *Mathematica 13.0.0*. Wolfram Research, Inc., Champaign, Illinois, 2021.