

Knot insertion and binary subdivision of splines

Michael S. Floater*

November 12, 2025

Abstract

In this paper we derive an efficient subdivision algorithm for the binary refinement of non-uniform spline functions or curves represented in the B-spline basis. We recursively insert the new knots into the divided differences that define the B-splines. This creates sequences of intermediate divided differences from which we construct sequences of normalized B-splines. From the refinement equations for these B-splines, the subdivision algorithm follows by induction. The coefficients computed at each step of the algorithm are themselves B-spline coefficients.

Math Subject Classification: 41A15, 65D07, 65D17

Keywords: Splines, B-splines, knot insertion, refinement, subdivision.

1 Introduction

There are two well known methods of refining a spline with knots in general position. One is Boehm's algorithm [1] which inserts one knot. It derives from the refinement of a divided difference, which leads to a simple formula for the new B-spline coefficients, expressing each as a weighted average of two of the old ones. The other is the Oslo algorithm [4] which inserts many different knots simultaneously. It computes each new B-spline coefficient using

*Department of Mathematics, University of Oslo, Moltke Moes vei 45, 0851 Oslo, Norway, email: michaelf@math.uio.no

a triangular recursive scheme that generalizes de Boor’s spline evaluation algorithm [6], and can be derived from the blossoming theory of Ramshaw [17], as described by Goldman [11] and Prautzsch et al. [16, Sec. 6.2].

A common application of knot insertion is to the *binary refinement* of a spline function or curve with simple but arbitrarily spaced knots, in which we insert one knot in each existing knot interval. For the special case of uniform knots and uniform insertions, there is an algorithm for this due to Lane and Riesenfeld [14, 13] that is more efficient than either Boehm’s algorithm or the Oslo algorithm. It starts by ‘doubling’ the B-spline coefficients, and then applies d steps of averaging to the doubled coefficients to compute the new coefficients, where d is the degree of the spline. It derives from the convolution property of uniform B-splines and was proposed in [14] as a method of rendering a uniform parametric spline curve: after just a few subdivisions one can simply plot the control polygon. It was also the starting point for the generalization of uniform tensor-product spline surfaces to surfaces with control meshes of arbitrary topology, for example those of Catmull and Clark [3] and Doo and Sabin [9]. More recently, binary refinement has been proposed as a tool for ‘ h -refinement’ of tensor-product splines in the IGA method of [12].

Thus it would be useful to have an efficient binary subdivision algorithm for non-uniform splines, similar to Lane-Riesenfeld. Two such algorithms were proposed by Schaefer and Goldman [18] and Cashman, Dodgson, and Sabin [2], based on blossoming. Both algorithms start with a doubling step and then apply averaging steps, requiring only d averages per knot, which is in fact less than the $2d$ averages per knot required by Lane-Riesenfeld. However, using blossoming to derive the algorithms of [18] and [2] would require carefully taking into account that a spline has several different blossoms, one for each polynomial piece.

In this paper we derive a subdivision algorithm in a different way. Both the initial B-splines and the refined B-splines are normalized divided differences of a truncated power. We successively insert knots into the initial divided differences in such a way that we obtain sequences of intermediate divided differences, reaching the refined divided differences in r steps of averaging, where $r := \lfloor d/2 \rfloor$. From these intermediate divided differences, we construct sequences of normalized B-splines. The subdivision algorithm then follows from the refinement equations for these B-splines. The resulting algorithm appears to agree with that of [2]. We also obtain a geometric interpretation of the algorithm: the coefficients computed at each step are themselves

B-spline coefficients. However, the intermediate B-spline sequences are not the standard ones because the knots of the B-splines are not consecutive in the refined knot vector. The algorithm is numerically stable because all averages are convex combinations.

The refinement equations for even and odd degree are different: for even degree it is sufficient to insert one knot at a time, while for odd degree we require a ‘double refinement’, Lemma 4, involving the insertion of two knots. Thus we treat the even and odd degrees separately, in Sections 5 and 6.

2 B-splines

Let $\mathbf{t} = (t_i)_{i \in \mathbb{Z}}$ be a bi-infinite sequence of reals, called knots, with $t_i < t_{i+1}$ and $\lim_{i \rightarrow \pm\infty} t_i = \pm\infty$. For any integer $d \geq 0$ we define the i -th B-spline of degree d as

$$B_{i,d,\mathbf{t}}(x) := (t_{i+d+1} - t_i)D_{i,d,\mathbf{t}}f, \quad x \in \mathbb{R}, \quad (1)$$

where $D_{i,d,\mathbf{t}}$ denotes the divided difference of order $d+1$,

$$D_{i,d,\mathbf{t}} := [t_i, t_{i+1}, \dots, t_{i+d+1}],$$

and f is the function

$$f(t) := \begin{cases} (t-x)^d, & \text{if } t \geq x; \\ 0, & \text{if } t < x. \end{cases} \quad (2)$$

Thus $D_{i,d,\mathbf{t}}$ is applied to the function $f(t)$ for each fixed x . Following [5], the B-spline $B_{i,d,\mathbf{t}}$ is a piecewise polynomial of degree d which is positive in (t_i, t_{i+d+1}) , and zero outside, and has smoothness C^{d-1} at $t_i, t_{i+1}, \dots, t_{i+d+1}$. Due to the normalizing factor $t_{i+d+1} - t_i$,

$$\sum_{i \in \mathbb{Z}} B_{i,d,\mathbf{t}}(x) = 1, \quad x \in \mathbb{R},$$

as shown for example in [7, p. 110].

3 Splines

We call a piecewise polynomial a *spline*. By a theorem of [5], any spline of degree d and continuity C^{d-1} at the knots of $\mathbf{t}$ can be expressed as

$$s(x) = \sum_{i \in \mathbb{Z}} c_i B_{i,d,\mathbf{t}}(x), \quad x \in \mathbb{R}, \quad (3)$$

with coefficients $c_i \in \mathbb{R}$. If we restrict s to the interval $[t_0, t_n]$ for some $n \geq 1$, then s consists of n polynomial pieces, and

$$s(x) = \sum_{i=-d}^{n-1} c_i B_{i,d,\mathbf{t}}(x), \quad x \in [t_0, t_n], \quad (4)$$

and the B-splines in this sum depend only on the knots $t_{-d}, \dots, t_{n+d}$.

4 Binary refinement

In this paper we consider the binary refinement of $\mathbf{t} = (t_i)_{i \in \mathbb{Z}}$ by adding one knot to every knot interval. Let u_i , $i \in \mathbb{Z}$, be the new knots with $t_i < u_i < t_{i+1}$. We will denote by $\boldsymbol{\tau} = (\tau_i)_{i \in \mathbb{Z}}$ the resulting refined knot vector defined by $\tau_{2i} := t_i$ and $\tau_{2i+1} := u_i$. Then the spline s in (3) can also be represented with respect to $\boldsymbol{\tau}$,

$$s(x) = \sum_{j \in \mathbb{Z}} b_j B_{j,d,\boldsymbol{\tau}}(x), \quad x \in \mathbb{R}, \quad (5)$$

and the task is to compute the b_j , $j \in \mathbb{Z}$, from the c_i , $i \in \mathbb{Z}$.

If we are only interested in the restriction $s|_{[t_0, t_n]}$, then, since

$$[t_0, t_n] = [\tau_0, \tau_{2n}],$$

we find

$$s(x) = \sum_{j=-d}^{2n-1} b_j B_{j,d,\boldsymbol{\tau}}(x), \quad x \in [t_0, t_n], \quad (6)$$

and the B-splines in this sum depend only on the knots $\tau_{-d}, \dots, \tau_{2n+d}$. In this restricted case the task is to compute $b_{-d}, \dots, b_{2n-1}$ from $c_{-d}, \dots, c_{n-1}$.

5 Refinement for even degree

Suppose $d = 2r$, $r \geq 1$. We would like to insert knots into the divided differences $D_{i,d,\mathbf{t}}$ to reach the refined divided differences $D_{j,d,\boldsymbol{\tau}}$ in r steps. We can do this by inserting each new u knot as close as possible to the centre of the t knots of the current divided difference. This will be made precise

in what follows. We obtain the following sequences of intermediate divided differences. For $k = 0, 1, \dots, r$, let

$$\begin{aligned} D_{2i}^k &:= [t_i, \dots, t_{i+d-k}, u_{i+r-k}, \dots, u_{i+r}], \\ D_{2i+1}^k &:= [t_{i+1}, \dots, t_{i+d-k+1}, u_{i+r-k}, \dots, u_{i+r}]. \end{aligned}$$

For example, after the first insertion we will obtain

$$\begin{aligned} D_{2i}^0 &= [t_i, \dots, t_{i+d}, u_{i+r}], \\ D_{2i+1}^0 &= [t_{i+1}, \dots, t_{i+d+1}, u_{i+r}], \end{aligned}$$

and after all the insertions we will end up with

$$\begin{aligned} D_{2i}^r &= [t_i, \dots, t_{i+r}, u_i, \dots, u_{i+r}] = D_{2i,d,\tau}, \\ D_{2i+1}^r &= [t_{i+1}, \dots, t_{i+r+1}, u_i, \dots, u_{i+r}] = D_{2i+1,d,\tau}. \end{aligned}$$

To write down the refinement equations for these divided differences, let us recall the basic refinement property that appears in [15, 10, 1]; see also [8] for some historical perspective.

Lemma 1 *For $x_0, x_1, \dots, x_{d+1}$ with $x_0 \neq x_{d+1}$ and any y ,*

$$[x_0, \dots, x_{d+1}] = \frac{y - x_0}{x_{d+1} - x_0} [x_0, \dots, x_d, y] + \frac{x_{d+1} - y}{x_{d+1} - x_0} [x_1, \dots, x_{d+1}, y].$$

This gives us a method of inserting a point y into the divided difference $[x_0, \dots, x_{d+1}]$. Thus, we can insert the knot u_{i+r} into $D_{i,d,t}$, and for $k = 0, 1, \dots, r-1$, insert $u_{i+r-k-1}$ into D_{2i}^k and u_{i+r+1} into D_{2i+1}^k , or equivalently, u_{i+r} into D_{2i-1}^k . In this way we deduce

Lemma 2

$$D_{i,d,t} = \frac{u_{i+r} - t_i}{t_{i+d+1} - t_i} D_{2i}^0 + \frac{t_{i+d+1} - u_{i+r}}{t_{i+d+1} - t_i} D_{2i+1}^0,$$

and for $k = 0, \dots, r-1$,

$$\begin{aligned} D_{2i-1}^k &= \frac{u_{i+r} - t_i}{t_{i+d-k} - t_i} D_{2i}^{k+1} + \frac{t_{i+d-k} - u_{i+r}}{t_{i+d-k} - t_i} D_{2i+1}^{k+1}, \\ D_{2i}^k &= \frac{u_{i+r-k-1} - t_i}{t_{i+d-k} - t_i} D_{2i}^{k+1} + \frac{t_{i+d-k} - u_{i+r-k-1}}{t_{i+d-k} - t_i} D_{2i+1}^{k+1}. \end{aligned}$$

5.1 B-spline sequences

For each $k = 0, 1, \dots, r$, we now define a sequence of B-splines by applying the divided differences D_j^k to f in (2) and normalizing so that the B-splines sum to 1: we let

$$\begin{aligned} B_{2i}^k(x) &:= (u_{i+r} - t_i) D_{2i}^k f, \\ B_{2i+1}^k(x) &:= (t_{i+d-k+1} - u_{i+r-k}) D_{2i+1}^k f. \end{aligned} \quad (7)$$

Then $B_j^r = B_{j,d,\tau}$ so we know that the B_j^r sum to 1. The fact that the B_j^k sum to 1 for $k < r$ will follow from Theorem 1. First we derive the refinement equations for the B_j^k . Applying the equations of Lemma 2 to f in (2) and using (7) to convert the $D_j^k f$ to $B_j^k(x)$ we obtain

Lemma 3

$$B_{i,d,t} = B_{2i}^0 + B_{2i+1}^0, \quad (8)$$

and for $k = 0, \dots, r-1$,

$$\begin{aligned} B_{2i-1}^k &= \frac{t_{i+d-k} - u_{i+r-k-1}}{t_{i+d-k} - t_i} B_{2i}^{k+1} + \frac{t_{i+d-k} - u_{i+r}}{t_{i+d-k} - t_i} B_{2i+1}^{k+1}, \\ B_{2i}^k &= \frac{u_{i+r-k-1} - t_i}{t_{i+d-k} - t_i} B_{2i}^{k+1} + \frac{u_{i+r} - t_i}{t_{i+d-k} - t_i} B_{2i+1}^{k+1}. \end{aligned} \quad (9)$$

5.2 Algorithm

Using Lemma 3, we now derive the subdivision algorithm. We recall the spline s in (3) and its refined representation in (5).

Theorem 1 Let $c_{2i}^0 := c_{2i+1}^0 := c_i$, and let

$$\begin{aligned} c_{2i}^k &:= \frac{t_{i+d-k+1} - u_{i+r-k}}{t_{i+d-k+1} - t_i} c_{2i-1}^{k-1} + \frac{u_{i+r-k} - t_i}{t_{i+d-k+1} - t_i} c_{2i}^{k-1}, \\ c_{2i+1}^k &:= \frac{t_{i+d-k+1} - u_{i+r}}{t_{i+d-k+1} - t_i} c_{2i-1}^{k-1} + \frac{u_{i+r} - t_i}{t_{i+d-k+1} - t_i} c_{2i}^{k-1}, \end{aligned} \quad (10)$$

for $k = 1, \dots, r$. Then

$$s(x) = \sum_{j \in \mathbb{Z}} c_j^k B_j^k(x), \quad x \in \mathbb{R}, \quad (11)$$

for all $k = 0, 1, \dots, r$, and therefore, $b_j = c_j^r$.

Proof. The proof is by induction on k . Substituting (8) into (3), gives

$$s = \sum_i c_i (B_{2i}^0 + B_{2i+1}^0) = \sum_i c_{2i}^0 B_{2i}^0 + \sum_i c_{2i+1}^0 B_{2i+1}^0,$$

which proves (11) when $k = 0$.

Now suppose (11) holds for $k \in \{0, 1, \dots, r-1\}$. We apply (9), so that

$$\begin{aligned} s &= \sum_i c_{2i-1}^k B_{2i-1}^k + \sum_i c_{2i}^k B_{2i}^k \\ &= \sum_i c_{2i-1}^k \left(\frac{t_{i+d-k} - u_{i+r-k-1}}{t_{i+d-k} - t_i} B_{2i}^{k+1} + \frac{t_{i+d-k} - u_{i+r}}{t_{i+d-k} - t_i} B_{2i+1}^{k+1} \right) \\ &\quad + \sum_i c_{2i}^k \left(\frac{u_{i+r-k-1} - t_i}{t_{i+d-k} - t_i} B_{2i}^{k+1} + \frac{u_{i+r} - t_i}{t_{i+d-k} - t_i} B_{2i+1}^{k+1} \right) \\ &= \sum_i c_{2i}^{k+1} B_{2i}^{k+1} + \sum_i c_{2i+1}^{k+1} B_{2i+1}^{k+1}, \end{aligned}$$

the last step due to (10). $\square$

Letting $c_i = 1$ for all i implies both $s = 1$ and $c_j^k = 1$ for all j and so (11) implies that the B_j^k sum to 1 for all k .

If we restrict x to the interval $[t_0, t_n]$, then for $k = 0, 1, \dots, r-1$, equation (11) reduces to

$$s(x) = \sum_{j=2(-d+k)+1}^{2n-2} c_j^k B_j^k(x). \quad (12)$$

So, for $k < r$, we only compute the c_j^k appearing in (12). For these k , the number of coefficients c_j^k goes down by 2 each time we increase k . However, recalling (6), at the last step we need to compute $b_j = c_j^r$ for $j = -d, \dots, 2n-1$, and so the number of c_j^r is $2n+d$, the same as the number of c_j^{r-1} .

5.3 Examples

The algorithm for $d = 2$ is

$$\begin{aligned} b_{2i} &= \frac{t_{i+2} - u_i}{t_{i+2} - t_i} c_{2i-1}^0 + \frac{u_i - t_i}{t_{i+2} - t_i} c_{2i}^0, \\ b_{2i+1} &= \frac{t_{i+2} - u_{i+1}}{t_{i+2} - t_i} c_{2i-1}^0 + \frac{u_{i+1} - t_i}{t_{i+2} - t_i} c_{2i}^0, \end{aligned}$$

with $c_{2i-1}^0 = c_{i-1}$ and $c_{2i}^0 = c_i$. If $n = 1$ in (4) then s is a quadratic polynomial restricted to the knot interval $[t_0, t_1]$. Figure 1 shows the flow of the algorithm in this case.

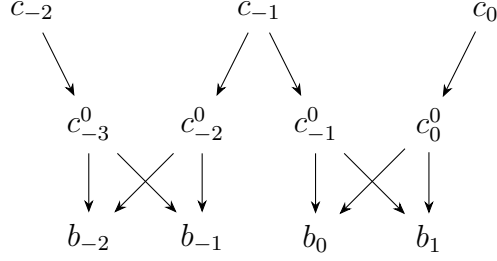


Figure 1: Algorithm when $d = 2$ and $n = 1$.

Figure 2 shows a numerical example of the algorithm for a quartic spline ($d = 4$) restricted to $[t_0, t_3]$ ($n = 3$) with

$$\begin{aligned} (t_{-4}, \dots, t_7) &:= (0, 0, 0, 0, 0, 2, 3, 6, 6, 6, 6, 6), \\ (c_{-4}, \dots, c_2) &:= (1.5, 2.3, 1.0, 1.5, 0.2, 1.0, 1.2) \end{aligned}$$

Here we define $u_i := t_i = t_{i+1}$ whenever $t_i = t_{i+1}$ for the multiple knots at the endpoints. The algorithm is used to refine the knot vector three times, each time making the uniform refinement $u_i := (t_i + t_{i+1})/2$. The vertices of the control polygons are formed as usual by placing the coefficients at knot averages.

6 Refinement for odd degree

Suppose $d = 2r + 1$, $r \geq 0$. Similar to the even degree case, we want to insert knots into the divided differences $D_{i,d,\mathbf{t}}$ to reach the refined divided differences $D_{j,d,\boldsymbol{\tau}}$ in r steps, but we now need to insert pairs of knots at each step to keep the u knots centred with respect to the t knots of each current divided difference. For $k = -1, 0, 1, \dots, r + 1$, let

$$D_i^k := [t_i, \dots, t_{i+d-k}, u_{i+r-k}, \dots, u_{i+r}].$$

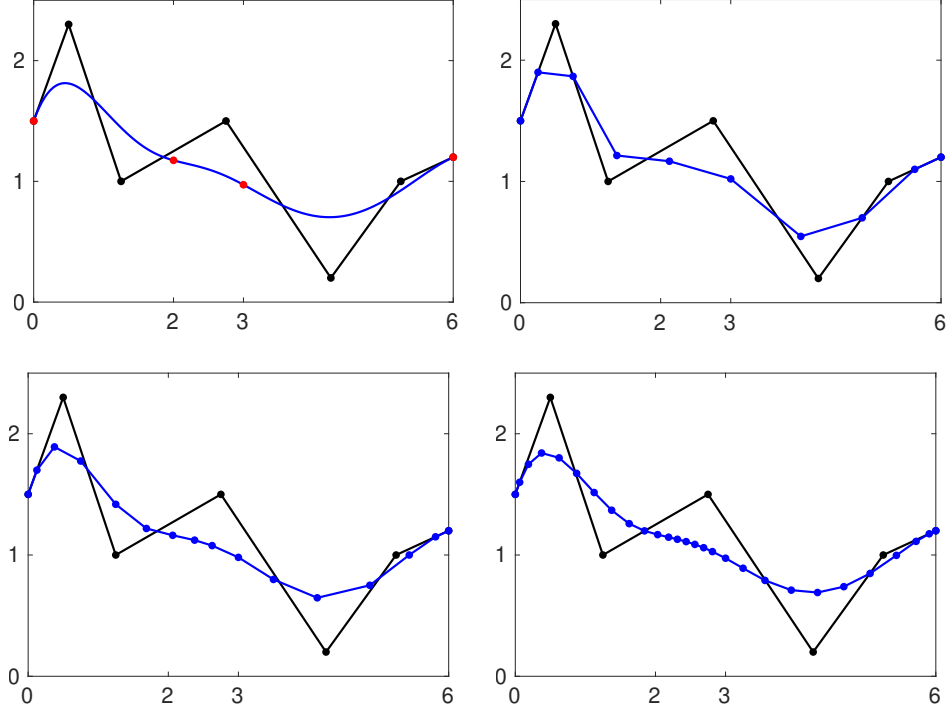


Figure 2: Refinement of a quartic spline.

Then, for example,

$$\begin{aligned}
 D_i^{-1} &= D_{i,d,t}, \\
 D_i^0 &= [t_i, \dots, t_{i+d}, u_{i+r}], \\
 D_i^1 &= [t_i, \dots, t_{i+d-1}, u_{i+r-1}, u_{i+r}],
 \end{aligned}$$

and

$$\begin{aligned}
 D_i^r &= [t_i, \dots, t_{i+r+1}, u_i, \dots, u_{i+r}] = D_{2i,d,\tau}, \\
 D_i^{r+1} &= [t_i, \dots, t_{i+r}, u_{i-1}, \dots, u_{i+r}] = D_{2i-1,d,\tau}.
 \end{aligned}$$

To derive the refinement equations we use the following ‘double refinement’.

Lemma 4 For $x_0, x_1, \dots, x_{d+1}$ with $x_0 \neq x_{d+1}$ and any y_0, y_1 ,

$$\begin{aligned} [x_0, \dots, x_{d+1}] &= \frac{y_0 - x_0}{x_{d+1} - x_0} [x_0, \dots, x_d, y_0] \\ &\quad + \frac{y_1 - y_0}{x_{d+1} - x_0} [x_1, \dots, x_d, y_0, y_1] \\ &\quad + \frac{x_{d+1} - y_1}{x_{d+1} - x_0} [x_1, \dots, x_{d+1}, y_1]. \end{aligned}$$

Proof. This follows from the three-term expansion

$$\begin{aligned} [x_1, \dots, x_{d+1}] - [x_0, \dots, x_d] &= [x_1, \dots, x_d, y_0] - [x_0, \dots, x_d] \\ &\quad + [x_1, \dots, x_d, y_1] - [x_1, \dots, x_d, y_0] \\ &\quad + [x_1, \dots, x_{d+1}] - [x_1, \dots, x_d, y_1]. \end{aligned}$$

□

This lemma gives us a method of inserting two points y_0 and y_1 into a divided difference $[x_0, \dots, x_{d+1}]$. Using it to insert $u_{i+r-k-1}$ and u_{i+r+1} into D_i^k for each $k = -1, 0, 1, \dots, r-1$ gives

Lemma 5 For $k = -1, 0, \dots, r-1$,

$$D_i^k = \frac{u_{i+r-k-1} - t_i}{t_{i+d-k} - t_i} D_i^{k+1} + \frac{u_{i+r+1} - u_{i+r-k-1}}{t_{i+d-k} - t_i} D_{i+1}^{k+2} + \frac{t_{i+d-k} - u_{i+r+1}}{t_{i+d-k} - t_i} D_{i+1}^{k+1}.$$

6.1 B-spline sequences

Now define

$$\begin{aligned} B_{2i}^k(x) &:= (t_{i+d-k} - t_i) D_i^k f, \\ B_{2i+1}^k(x) &:= (u_{i+r+1} - u_{i+r-k}) D_{i+1}^{k+1} f, \end{aligned} \tag{13}$$

for $k = 0, 1, \dots, r$, with f in (2). Then $B_j^r = B_{j,d,\tau}$ and so these sum to 1 and it will follow from Theorem 2 that the B_j^k sum to 1 for $k < r$. Applying the equations in Lemma 5 to f and using (13) to convert $D_j^k f$ to $B_j^k(x)$ we obtain

Lemma 6

$$B_{i,d,t} = \frac{u_{i+r} - t_i}{t_{i+d} - t_i} B_{2i}^0 + B_{2i+1}^0 + \frac{t_{i+d+1} - u_{i+r+1}}{t_{i+d+1} - t_{i+1}} B_{2i+2}^0, \tag{14}$$

and for $k = 0, \dots, r-1$,

$$\begin{aligned} B_{2i-1}^k &= \frac{u_{i+r} - u_{i+r-k-1}}{t_{i+d-k-1} - t_i} B_{2i}^{k+1}, \\ B_{2i}^k &= \frac{u_{i+r-k-1} - t_i}{t_{i+d-k-1} - t_i} B_{2i}^{k+1} + B_{2i+1}^{k+1} + \frac{t_{i+d-k} - u_{i+r+1}}{t_{i+d-k} - t_{i+1}} B_{2i+2}^{k+1}. \end{aligned} \quad (15)$$

6.2 Algorithm

The subdivision algorithm now follows from Lemma 6.

Theorem 2 *Let*

$$c_{2i}^0 := \frac{t_{i+d} - u_{i+r}}{t_{i+d} - t_i} c_{i-1} + \frac{u_{i+r} - t_i}{t_{i+d} - t_i} c_i, \quad (16)$$

and $c_{2i+1}^0 := c_i$, and for $k = 1, \dots, r$, let

$$c_{2i}^k := \frac{t_{i+d-k} - u_{i+r}}{t_{i+d-k} - t_i} c_{2i-2}^{k-1} + \frac{u_{i+r} - u_{i+r-k}}{t_{i+d-k} - t_i} c_{2i-1}^{k-1} + \frac{u_{i+r-k} - t_i}{t_{i+d-k} - t_i} c_{2i}^{k-1}, \quad (17)$$

and $c_{2i+1}^k := c_{2i}^{k-1}$. Then

$$s(x) = \sum_{j \in \mathbb{Z}} c_j^k B_j^k(x), \quad x \in \mathbb{R}, \quad (18)$$

for all $k = 0, 1, \dots, r$, and therefore, $b_j = c_j^r$.

Proof. The proof is by induction on k . Substituting (14) into (3), gives

$$\begin{aligned} s &= \sum_i c_i \left(\frac{u_{i+r} - t_i}{t_{i+d} - t_i} B_{2i}^0 + B_{2i+1}^0 + \frac{t_{i+d+1} - u_{i+r+1}}{t_{i+d+1} - t_{i+1}} B_{2i+2}^0 \right) \\ &= \sum_i \left(\frac{t_{i+d} - u_{i+r}}{t_{i+d} - t_i} c_{i-1} + \frac{u_{i+r} - t_i}{t_{i+d} - t_i} c_i \right) B_{2i}^0 + \sum_i c_i B_{2i+1}^0 \\ &= \sum_i c_{2i}^0 B_{2i}^0 + \sum_i c_{2i+1}^0 B_{2i+1}^0, \end{aligned}$$

the last step due to (16).

Now suppose (18) holds for some $k \in \{0, 1, \dots, r-1\}$. Then using (15),

$$\begin{aligned}
s &= \sum_i c_{2i-1}^k B_{2i-1}^k + \sum_i c_{2i}^k B_{2i}^k \\
&= \sum_i c_{2i-1}^k \frac{u_{i+r} - u_{i+r-k-1}}{t_{i+d-k-1} - t_i} B_{2i}^{k+1} \\
&\quad + \sum_i c_{2i}^k \left(\frac{u_{i+r-k-1} - t_i}{t_{i+d-k-1} - t_i} B_{2i}^{k+1} + B_{2i+1}^{k+1} + \frac{t_{i+d-k} - u_{i+r+1}}{t_{i+d-k} - t_{i+1}} B_{2i+2}^{k+1} \right) \\
&= \sum_i c_{2i}^{k+1} B_{2i}^{k+1} + \sum_i c_{2i+1}^{k+1} B_{2i+1}^{k+1},
\end{aligned}$$

the last step due to (17). $\square$

If we restrict x to the interval $[t_0, t_n]$, then for $k = 0, 1, \dots, r-1$, equation (18) reduces to

$$s(x) = \sum_{j=2(-d+k+1)}^{2n-2} c_j^k B_j^k(x). \quad (19)$$

So, for $k < r$, we only compute the c_j^k appearing in (19). The number of c_j^k goes down by 2 each time we increase k until $k = r-1$ but the number of c_j^{r-1} and c_j^r is the same, $2n+d$.

6.3 Examples

The algorithm for $d = 3$ is

$$\begin{aligned}
b_{2i} &= \frac{t_{i+2} - u_{i+1}}{t_{i+2} - t_i} c_{2i-2}^0 + \frac{u_{i+1} - u_i}{t_{i+2} - t_i} c_{2i-1}^0 + \frac{u_i - t_i}{t_{i+2} - t_i} c_{2i}^0, \\
b_{2i+1} &= c_{2i}^0,
\end{aligned}$$

with

$$\begin{aligned}
c_{2i}^0 &= \frac{t_{i+3} - u_{i+1}}{t_{i+3} - t_i} c_{i-1} + \frac{u_{i+1} - t_i}{t_{i+3} - t_i} c_i, \\
c_{2i+1}^0 &= c_i.
\end{aligned}$$

If $n = 1$ in (4) then s is a cubic polynomial restricted to the knot interval $[t_0, t_1]$. Figure 3 shows the flow of the algorithm in this case.

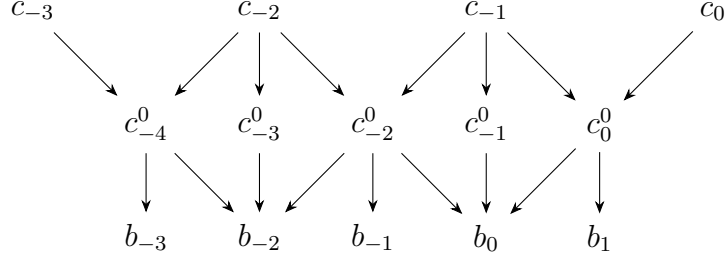


Figure 3: Algorithm when $d = 3$ and $n = 1$.

Figure 4 shows a numerical example of the algorithm for a quintic spline ($d = 5$) restricted to $[t_0, t_3]$ ($n = 3$) with

$$(t_{-5}, \dots, t_8) := (0, 0, 0, 0, 0, 0, 2, 3, 6, 6, 6, 6, 6, 6),$$

$$(c_{-5}, \dots, c_2) := (1.0, 0.6, 2.3, 1.0, 1.5, 0.2, 1.0, 1.2).$$

As for the quartic example, the algorithm is used to make three uniform refinements of the knot vector.

References

- [1] W. Boehm, *Inserting new knots into B-spline curves*, Comp.-Aided Design **12** (1980), 199–201.
- [2] T. J. Cashman, N. A. Dodgson, and M. A. Sabin, *A symmetric, non-uniform refine and smooth subdivision algorithm for general degree B-splines*, Comp. Aided Geom. Design **26** (2009), 94–104.
- [3] E. Catmull and J. Clark, *Recursively generated B-spline surfaces on arbitrary topological meshes*, Comp.-Aided Design **10** (1978), 350–355.
- [4] E. Cohen, T. Lyche, and R. Riesenfeld, *Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics*, Comp. Graph. Image Proc. **14** (1980), 87–111.
- [5] H. B. Curry and I. J. Schoenberg, *On Pólya frequency functions IV: the fundamental spline functions and their limits*, J. Analyse Math. **17** (1966), 71–107.

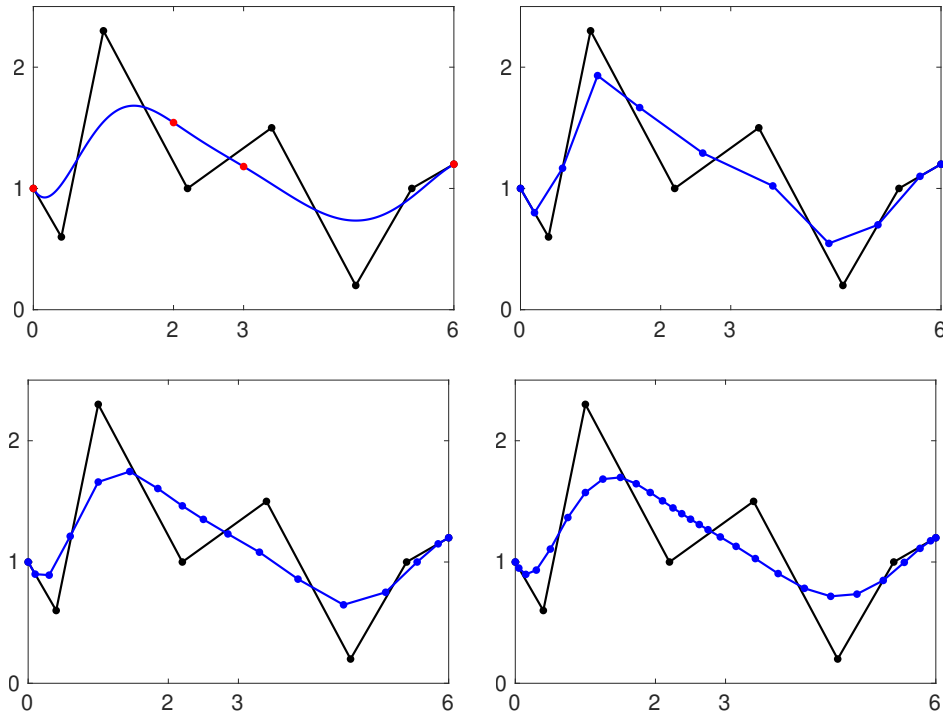


Figure 4: Refinement of a quintic spline.

- [6] C. de Boor, *On calculating with B-splines*, J. Approx. Theory **6** (1972), 50–62.
- [7] C. de Boor, *A practical guide to splines*, Springer-Verlag, 1978.
- [8] C. de Boor and A. Pinkus, *The B-spline recurrence relations of Chakalov and of Popoviciu*, J. Approx. Theory **124** (2003), 115–123.
- [9] D. Doo and M. Sabin, *Behaviour of recursive subdivision surfaces near extraordinary points*, Comp.-Aided Design **10** (1978), 356–360.
- [10] J. Favard, *Sur l'interpolation*, J. Math. Pures Appl. **19** (1940), 281–306.
- [11] R. N. Goldman, *Blossoming and knot insertion algorithms for B-spline curves*, Comp. Aided Geom. Design **7** (1990), 69–81.
- [12] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Comput. Methods Appl. Mech. Eng. **194** (2005), 4135–4195.

- [13] J. M. Lane and R. F. Riesenfeld, *The application of total positivity to computer aided curve and surface design*, Tech. report, Dept. of Computer Science, University of Utah, Salt Lake City, 1977.
- [14] ———, *A theoretical development for the computer generation and display of piecewise polynomials surfaces*, IEEE Trans. Pattern Analysis and Machine Intelligence **2** (1980), 35–46.
- [15] T. Popoviciu, *Sur quelques propriétés des fonctions d'une ou de deux variables réelles*, Phd thesis, Faculté des Sciences de Paris, 1933.
- [16] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-spline techniques*, Springer, 2002.
- [17] L. Ramshaw, *Blossoms are polar forms*, Comp. Aided Geom. Design **6** (1989), 323–358.
- [18] S. Schaefer and R. Goldman, *Non-uniform subdivision for B-splines of arbitrary degree*, Comp. Aided Geom. Design **26** (2009), 75–81.