A Symmetric, Non-uniform, Refine and Smooth Subdivision Algorithm for General Degree B-splines

Thomas J. Cashman [a,*] Neil A. Dodgson [a] Malcolm A. Sabin [b]

[a] *Computer Laboratory, University of Cambridge, CB3 0FD, England*
[b] *Numerical Geometry Ltd, 19 John Amner Close, Ely, Cambridge CB6 1DT, England*

Notice: this is the author's version of a work that was accepted for publication in Computer Aided Geometric Design. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Computer Aided Geometric Design, [26, 1, 2009] doi:10.1016/j.cagd.2007.12.001

Abstract

Subdivision surfaces would find a greater number of applications if there was a scheme that included general degree NURBS as a special case. As a step towards such a scheme, we present a univariate refine and smooth subdivision algorithm that applies directly to regular regions of a surface and might, in future work, be generalised to incorporate extraordinary points. The algorithm is symmetric and non-uniform, is defined for general degree, and has similar properties to the uniform Lane-Riesenfeld refine and smooth construction.

Key words: Non-uniform, Subdivision, NURBS, Smoothing, Lane-Riesenfeld, Knot insertion

1. Introduction

We present an algorithm that doubles the number of knots in a B-spline curve, and can be directly applied in the two directions of tensor product B-spline surfaces. We designed it to be *general degree and non-uniform*, use a *refine and smooth* factorisation, and be *symmetric*. In this section, we motivate each of these properties in turn.

1.1. *General degree and non-uniform*

Since the late 1970s, NURBS have been the standard representation for sculptured surfaces used in manufacture. Subdivision surfaces were also introduced at the end of the 1970s and, unlike NURBS, do not require a rectangular control grid. Both are based on B-splines, and so quad-grid subdivision surfaces can be seen as a generalisation of a subset of NURBS. The generalisation could be as useful in CAD/CAM as it is in computer animation, where subdivision is now the de-facto standard. For CAD, however, the aspects in which subdivision is only a subset are regarded as important, and any new standard must be more compatible with NURBS than current subdivision surfaces.

* Corresponding author.
Email addresses: tc270@cam.ac.uk (Thomas J. Cashman), nad@cl.cam.ac.uk (Neil A. Dodgson), malcolm@geometry.demon.co.uk (Malcolm A. Sabin).

Our target is therefore a subdivision extension of NURBS in all their generality: non-uniform, rational, and of general degree. The second of these is straightforward, since control points can be weighted and the subdivision process executed in projective space without any further theory.

Sederberg et al. (1998) tackle non-uniform quadratic and cubic subdivision in a more ambitious setting, assigning a knot interval to every edge of the control polyhedron, not just to every strip of polyhedron faces. Müller et al. (2006) extend this work with limit-point rules for non-uniform cubic subdivision, but there has been no attempt to handle general degree at the same time. Working in the other direction, Prautzsch (1998) presents a general degree scheme for equal knot intervals, which is developed further by Warren and Weimer (2001), Zorin and Schröder (2001) and Stam (2001). None of these general degree schemes handle non-uniformity, however, and a subdivision scheme compatible with NURBS must encompass both at the same time.

1.2. *Refine and smooth*

Algorithms for non-uniform, general degree knot insertion have been available since Boehm (1980) and Cohen et al. (1980) independently arrived at different approaches. Both show that a point after knot insertion at degree d takes contributions from $O(d)$ control points. In the bivariate case, this worsens to $O(d^2)$, which prohibitively complicates the implementation and analysis of high degree subdivision schemes in irregular regions.

Existing general degree schemes for equal knot intervals work around this problem by adapting the Lane and Riesenfeld (1980) *refine and smooth* algorithm. This factorisation breaks uniform knot insertion into a refinement stage followed by local smoothing filters, each of which requires contributions from only $O(1)$ points. Any extension of these schemes to unequal knot intervals therefore requires a non-uniform analogue of the Lane-Riesenfeld algorithm. A univariate algorithm is an essential first step, as it can be directly applied to regular regions of a subdivision surface and provides a foundation for tackling extraordinary points.

1.3. *Symmetric*

Earlier work (Cashman et al., 2007) considered Schaefer's knot insertion (Schaefer and Goldman, 2009) as the required non-uniform factorisation. Schaefer's algorithm is asymmetric, however, since computed points change if the control polygon is indexed in the opposite direction. B-spline control points are invariant under this transformation, so knot insertion is inherently symmetric for both input and output. Schaefer's algorithm uses *asymmetric* smoothing steps to construct a *symmetric* output, and therefore needs a consistent ordering for control points. Choosing a curve's direction is trivial, but for the natural application to surfaces we must give an orientation to every edge and the orientation of parallel edges must be consistent. If the surface contains extraordinary points, Figure 1 shows that this is not always possible.

We therefore require a non-uniform, general degree refine and smooth algorithm that is also symmetric. Symmetry comes naturally in the uniform case, as Lane and Riesenfeld's intermediate points are the control points for a B-spline of lower degree. Unfortunately, our earlier work (Cashman et al., 2007) showed that this symmetry constraint does not generalise well to unequal intervals. Note that symmetry is neither necessary nor sufficient for an algorithm to reduce to the Lane-Riesenfeld construction when knot spacings are uniform. Schaefer and Goldman (2009) present two algorithms for cubic knot insertion, both of which reduce to Lane-Riesenfeld but only one of which is symmetric. On the other hand, the algorithm presented in this paper is symmetric, but does not reduce to the Lane-Riesenfeld algorithm for uniform knots.

1.4. *Contents*

Section 2 examines the algorithm in terms of the *blossom* function, which gives the greatest insight into its operation. An explicit implementation follows in Section 3, before we consider multiple knots in Section 4 and conclude in Section 5.

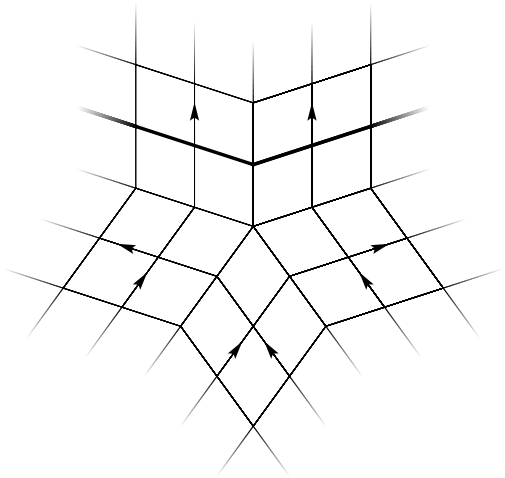


Fig. 1. An attempt to give an orientation to edges of a mesh containing an extraordinary point. There is no consistent orientation for the thickened edge.

2. The algorithm in terms of blossoming

Blossoming was introduced by Ramshaw (1987), although the technique of polar forms (Ramshaw, 1989) existed before Ramshaw's application to splines. The blossom of a degree-d polynomial $P(t)$, as summarised by Vouga and Goldman (2007), is the unique polynomial in d variables $p(u_1, u_2, \ldots, u_d)$ which satisfies the following three properties:

– symmetric: $p(u_{\sigma 1}, u_{\sigma 2}, \ldots, u_{\sigma d}) = p(u_1, u_2, \ldots, u_d)$ for any permutation σ.
– multiaffine: $p((1-\alpha)u_{11} + \alpha u_{12}, u_2, \ldots, u_d) = (1-\alpha)p(u_{11}, u_2, \ldots, u_d) + \alpha p(u_{12}, u_2, \ldots, u_d)$.
– diagonal: $p(t, t, \ldots, t) = P(t)$.

A consequence of this definition is the dual function property: if P_i is the ith control point of a B-spline $P(t)$ with knots u_i, then $P_i = p(u_{i+1}, \ldots, u_{i+d})$. We can abbreviate a blossom value by writing $p(u_1, u_2, \ldots, u_d)$ as just $u_1\, u_2\, \ldots\, u_d$.

2.1. *Overview*

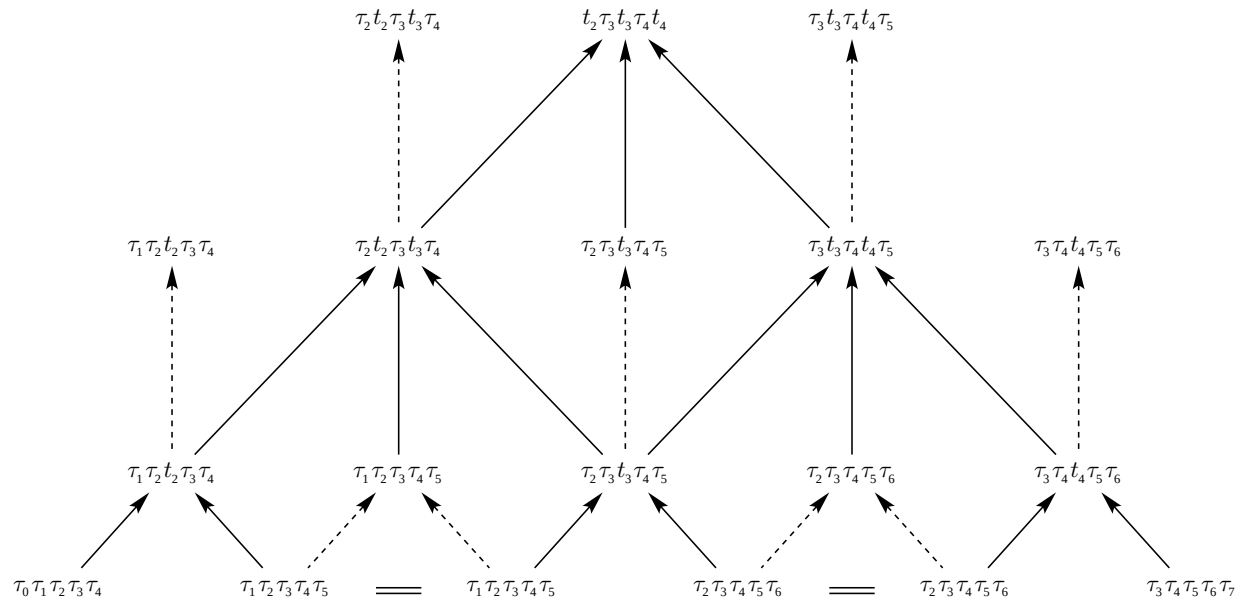
Our algorithm acts on a B-spline of degree d, and we require that exactly one new knot is inserted between each pair of adjacent original knots. We write τ_i for the ith knot of the original knot vector $\boldsymbol{\tau}$, and t_i for the knot inserted between knots τ_i and τ_{i+1}. As for all B-splines, we require $\tau_i \le t_i \le \tau_{i+1}$ for all i.

For odd d, the refine stage places a new point somewhere between each pair of original control points. Each smoothing stage copies half of the points unaltered, and the remaining half are derived from an affine combination of three points. Points are never copied twice, however, as the set of smoothed control points alternates at every step. This algorithm was developed independently by Schaefer and Goldman, and it may be significant that different researchers arrived at the same approach.

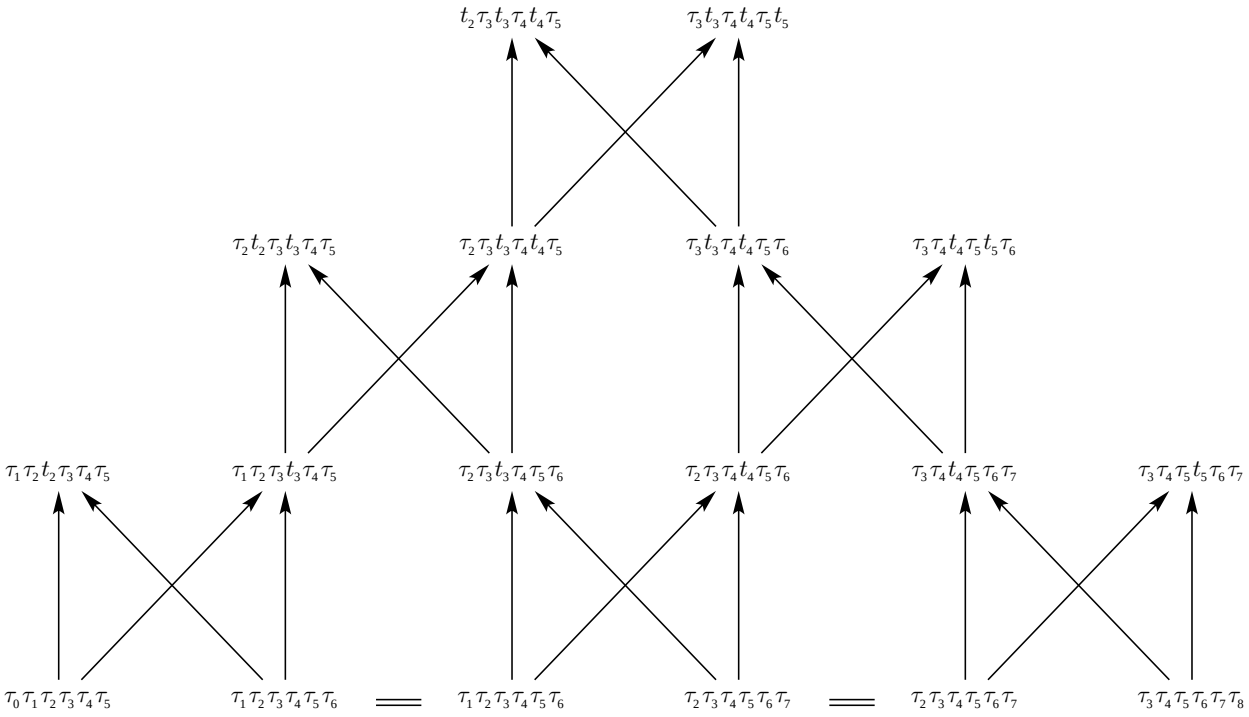
The algorithm for even d is novel. The refine stage simply doubles control points, and each smoothing stage then places a pair of new points somewhere on the edge between a pair of old points. Only half of the edges gain new points in this way, however, and with a parallel to odd degree, the smoothed half alternates at every step.

2.2. *Odd and even degree examples*

Figure 2 shows blossoming diagrams for the quintic and sextic algorithms, where each point is written as d parameters of the blossom function p. In each diagram, an affine combination is represented by ar-



(a) Quintic



(b) Sextic

Fig. 2. Blossoming diagrams for the algorithm for degrees 5 and 6

rows entering a point, and a dotted line is used where there is actually no calculation to be done. Arrows near doubled points are drawn to make the general structure as clear as possible, and doubled points are highlighted by = signs.

The diagrams are translation-invariant with respect to the point indices. So the algorithm's action at any point not drawn can be deduced by finding an isomorphic point which is in the diagram drawn. We are not considering end conditions at this stage, as an appropriate treatment requires multiple knots, which we discuss in Section 4.

3. The algorithm and its variations

Figure 3 gives an implementation of the algorithm in an imperative programming style, which we use in this section to describe possible variations for different applications. Appendices A and B give proofs for the correctness of this implementation.

We use the following abbreviated notation for writing affine combinations.

$$|\alpha\ \underline{\beta\ \gamma}\ \epsilon| = \begin{cases} \dfrac{\gamma-\beta}{\epsilon-\alpha} & , \quad \alpha \le \beta \le \gamma \le \epsilon \quad \text{and} \quad \alpha \neq \epsilon \\ 0 & , \quad \text{otherwise} \end{cases}$$

$$|\underline{\alpha\ \beta}\ \epsilon| = |\alpha\ \underline{\alpha\ \beta}\ \epsilon| = \begin{cases} \dfrac{\beta-\alpha}{\epsilon-\alpha} & , \quad \alpha \le \beta \le \epsilon \quad \text{and} \quad \alpha \neq \epsilon \\ 0 & , \quad \text{otherwise} \end{cases}$$

$$|\alpha\ \underline{\gamma\ \epsilon}| = 1 - |\underline{\alpha\ \gamma}\ \epsilon| = \begin{cases} \dfrac{\epsilon-\gamma}{\epsilon-\alpha} & , \quad \alpha \le \gamma \le \epsilon \quad \text{and} \quad \alpha \neq \epsilon \\ 1 & , \quad \text{otherwise} \end{cases}$$

Therefore, we have that $|\underline{\alpha\ \beta}\ \epsilon| + |\alpha\ \underline{\beta\ \gamma}\ \epsilon| + |\alpha\ \underline{\gamma\ \epsilon}| = 1$ for any $\alpha \le \beta \le \gamma \le \epsilon$, and $|\underline{\alpha\ \beta}\ \gamma| + |\alpha\ \underline{\beta\ \gamma}| = 1$ for any α, β and γ at all.

3.1. *Affine combinations of two points*

At line 13 of Figure 3, a weighted average combines three points to form a new one, as shown in Figure 4a. It may be preferable, for the reasons discussed in Section 1.2, to break this into the three averages of two points shown in Figure 4b instead. The following replacement for line 13 employs this two-stage construction.

$$\begin{aligned} \mathsf{P}^{2\lambda}_{i-\frac{1}{2}} &\leftarrow |u_i\ \underline{u_{i+\lambda}\ u_{i+\mathtt{d}-\lambda}}|\,\mathsf{P}^{2\lambda-1}_{i-1} + |\underline{u_i\ u_{i+\lambda}}\ u_{i+\mathtt{d}-\lambda}|\,\mathsf{P}^{2\lambda-1}_{i} \\ \mathsf{P}^{2\lambda}_{i+\frac{1}{2}} &\leftarrow |u_{i-\mathtt{d}+\lambda}\ \underline{u_{i-\lambda}\ u_i}|\,\mathsf{P}^{2\lambda-1}_{i} + |\underline{u_{i-\mathtt{d}+\lambda}\ u_{i-\lambda}}\ u_i|\,\mathsf{P}^{2\lambda-1}_{i+1} \\ \mathsf{P}^{2\lambda+1}_{i} &\leftarrow |u_{i-\mathtt{d}+\lambda}\ \underline{u_i\ u_{i+\mathtt{d}-\lambda}}|\,\mathsf{P}^{2\lambda}_{i-\frac{1}{2}} + |\underline{u_{i-\mathtt{d}+\lambda}\ u_i}\ u_{i+\mathtt{d}-\lambda}|\,\mathsf{P}^{2\lambda}_{i+\frac{1}{2}} \end{aligned}$$

3.2. *Using no additional memory*

It may be useful to have an implementation of the algorithm where points are calculated "in-place", rather than via intermediate points which require additional memory. The odd-degree case, as written in lines 3 to 13 of Figure 3, is directly converted to this form by identifying P^{λ}_{i} with the same memory location for every value of λ.

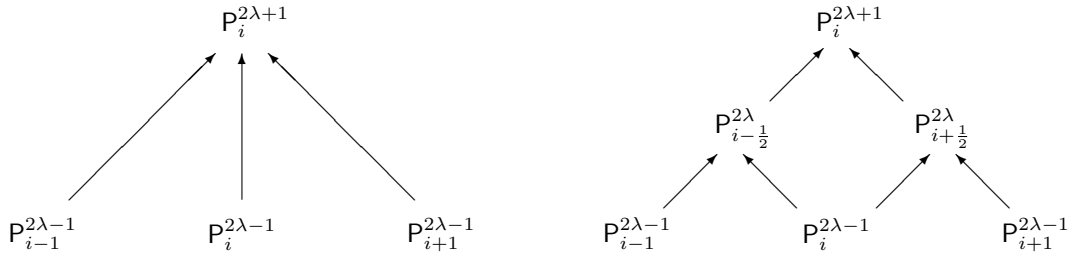
For even degree, however, writing $\mathsf{P}^{2\lambda}_{i}$ over $\mathsf{P}^{2\lambda-2}_{i}$ in line 21 makes $\mathsf{P}^{2\lambda-2}_{i-1}$ unavailable on the next loop iteration (in line 23). For this scenario, we therefore have a variation shown in Figure 5b, which finds $\mathsf{P}^{2\lambda}_{i}$ in terms of $\mathsf{P}^{2\lambda}_{i-1}$ instead. The required replacement for line 23 is

$$\mathsf{P}^{2\lambda}_{i} \leftarrow |u_{i-\lambda}\ \underline{u_{i+\lambda}\ u_{i+\mathtt{d}-\lambda+1}}|\,\mathsf{P}^{2\lambda}_{i-1} + |\underline{u_{i-\lambda}\ u_{i+\lambda}}\ u_{i+\mathtt{d}-\lambda+1}|\,\mathsf{P}^{2\lambda-2}_{i}$$

input : Degree d
input : Control points P_i for $0 \le i \le \mathsf{n}$
input : Knot vector $\boldsymbol{\tau}$, where $|\boldsymbol{\tau}| = \mathsf{n} + \mathsf{d}$
input : Knots to be inserted $\boldsymbol{t}$, where $\tau_i \le t_i \le \tau_{i+1}$
output: Control points $\mathsf{P}_i^{\mathsf{d}}$ for B-spline on the refined knot vector $\boldsymbol{\tau} \cup \boldsymbol{t}$

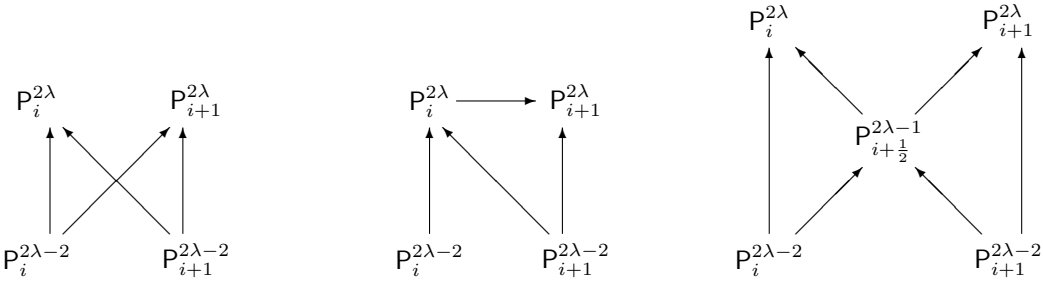
$\boldsymbol{u} \leftarrow \boldsymbol{\tau} \cup \boldsymbol{t}$ // such that each $\tau_i = u_{2i}$ and each $t_i = u_{2i+1}$
if $\mathsf{d} \bmod 2 = 1$ **then**
  // Odd degree: let τ_i correspond to P_i
  // Refine
  for $i \leftarrow 0$ **to** n **do**
    $\mathsf{P}_{2i}^1 \leftarrow \mathsf{P}_i$
  for $i \leftarrow 0$ **to** $\mathsf{n} - 1$ **do**
    $j \leftarrow 2i + 1$
    $\mathsf{P}_j^1 \leftarrow |u_{j-\mathsf{d}} \;\; \underline{u_j \;\; u_{j+\mathsf{d}}}|\, \mathsf{P}_i + |\underline{u_{j-\mathsf{d}} \;\; u_j} \;\; u_{j+\mathsf{d}}|\, \mathsf{P}_{i+1}$
  // Smooth
  for $\lambda \leftarrow 1$ **to** $(\mathsf{d} - 1)/2$ **do** // for each smoothing step
    for $i \leftarrow \lambda$ **to** $2\mathsf{n} - \lambda$ **do** // for each new point
      if $\lambda \bmod 2 = i \bmod 2$ **then**
        // Copy this point from previous step
        $\mathsf{P}_i^{2\lambda+1} \leftarrow \mathsf{P}_i^{2\lambda-1}$
      else
        // Calculate $\mathsf{P}_i^{2\lambda+1}$ by smoothing three points
        $\mathsf{P}_i^{2\lambda+1} \leftarrow |u_{i-\mathsf{d}+\lambda} \quad \underline{u_{i+\lambda} \;\; u_{i+\mathsf{d}-\lambda}}|\, \mathsf{P}_{i-1}^{2\lambda-1}$
        $+|u_{i-\mathsf{d}+\lambda} \;\; \underline{u_{i-\lambda} \;\; u_{i+\lambda}} \;\; u_{i+\mathsf{d}-\lambda}|\, \mathsf{P}_i^{2\lambda-1}$
        $+|\underline{u_{i-\mathsf{d}+\lambda} \;\; u_{i-\lambda}} \quad u_{i+\mathsf{d}-\lambda}|\, \mathsf{P}_{i+1}^{2\lambda-1}$
else
  // Even degree: let τ_i correspond to the edge $\mathsf{P}_{i-1}\mathsf{P}_i$
  // Refine
  for $i \leftarrow 0$ **to** n **do**
    $\mathsf{P}_{2i}^0 \leftarrow \mathsf{P}_i$
    $\mathsf{P}_{2i+1}^0 \leftarrow \mathsf{P}_i$
  // Smooth
  for $\lambda \leftarrow 1$ **to** $\mathsf{d}/2$ **do** // for each smoothing step
    for $i \leftarrow \lambda$ **to** $2\mathsf{n} + 1 - \lambda$ **do** // for each new point
      if $\lambda \bmod 2 = i \bmod 2$ **then**
        $\mathsf{P}_i^{2\lambda} \leftarrow |u_{i-\mathsf{d}+\lambda} \;\; \underline{u_{i-\lambda+1} \;\; u_{i+\mathsf{d}-\lambda+2}}|\, \mathsf{P}_i^{2\lambda-2} + |\underline{u_{i-\mathsf{d}+\lambda} \;\; u_{i-\lambda+1}} \;\; u_{i+\mathsf{d}-\lambda+2}|\, \mathsf{P}_{i+1}^{2\lambda-2}$
      else
        $\mathsf{P}_i^{2\lambda} \leftarrow |u_{i-\mathsf{d}+\lambda-1} \;\; \underline{u_{i+\lambda} \;\; u_{i+\mathsf{d}-\lambda+1}}|\, \mathsf{P}_{i-1}^{2\lambda-2} + |\underline{u_{i-\mathsf{d}+\lambda-1} \;\; u_{i+\lambda}} \;\; u_{i+\mathsf{d}-\lambda+1}|\, \mathsf{P}_i^{2\lambda-2}$

Fig. 3. Our algorithm for non-uniform refine and smooth subdivison



(a) Line 13 of Figure 3 (b) Two-stage variation

Fig. 4. Variations on the smoothing step for odd degree



(a) Lines 21 and 23 of Figure 3 (b) Constant memory variation (c) Two-stage variation

Fig. 5. Variations on the smoothing steps for even degree

3.3. *Two stages for even degree*

For the even degree case, shown in Figure 5a, lines 21 and 23 calculate a new point $\mathsf{P}_i^{2\lambda}$ at 'level' 2λ from two points at level $2\lambda-2$. While a two-stage construction does not reduce the size of the affine combination here, it may still be useful to consider calculating the points $\mathsf{P}_i^{2\lambda}$ via a point in $\mathsf{P}^{2\lambda-1}$, as shown in Figure 5c. In particular, this variation may prove important in a generalisation of the bivariate algorithm to surfaces containing extraordinary faces. For this modification, line 21 should be replaced with

$$\mathsf{P}_{i+\frac{1}{2}}^{2\lambda-1} \leftarrow |u_{i-\mathsf{d}+\lambda} \;\; \underline{u_{i+1} \;\; u_{i+\mathsf{d}-\lambda+2}}|\, \mathsf{P}_i^{2\lambda-2} + |\underline{u_{i-\mathsf{d}+\lambda} \;\; u_{i+1}} \;\; u_{i+\mathsf{d}-\lambda+2}|\, \mathsf{P}_{i+1}^{2\lambda-2}$$
$$\mathsf{P}_i^{2\lambda} \leftarrow |u_{i-\mathsf{d}+\lambda} \;\; \underline{u_{i-\lambda+1} \;\; u_{i+1}}|\, \mathsf{P}_i^{2\lambda-2} + |\underline{u_{i-\mathsf{d}+\lambda} \;\; u_{i-\lambda+1}} \;\; u_{i+1}|\, \mathsf{P}_{i+\frac{1}{2}}^{2\lambda-1}$$

and line 23 with

$$\texttt{// } \mathsf{P}_{i-\frac{1}{2}}^{2\lambda-1} \texttt{ was calculated in the previous loop iteration for } i$$
$$\mathsf{P}_i^{2\lambda} \leftarrow |u_i \;\; \underline{u_{i+\lambda} \;\; u_{i+\mathsf{d}-\lambda+1}}|\, \mathsf{P}_{i-\frac{1}{2}}^{2\lambda-1} + |\underline{u_i \;\; u_{i+\lambda}} \;\; u_{i+\mathsf{d}-\lambda+1}|\, \mathsf{P}_i^{2\lambda-2}$$

4. Multiple knots

We have not yet discussed end conditions or the behaviour around multiple knots. It is possible, but undesirable, to use our algorithm directly in these cases. Setting $\tau_i = t_i = \cdots = t_{i+m-2} = \tau_{i+m-1}$, for example, gives a knot of multiplicity m. The subdivision algorithm then outputs a knot of multiplicity $2m-1$, and so the limit curve will have a knot of unbounded multiplicity and therefore an infinitely-repeated point. This is unlikely to be an acceptable result, and so we would like a way of dealing with multiple knots that allows us to subdivide without increasing multiplicity.

A proper treatment is outside the scope of this paper, but if a knot is fully multiple ($m = \mathsf{d}$), then subdivision without increasing multiplicity is simply a case of filtering P^{d} to remove repeated points. Naturally, it is possible to find which points to discard by simple consideration of the knot vector; no geometrical tests

are required. If it is permissible for a knot of multiplicity m, where $1 < m < \mathsf{d}$, to become fully multiple after a few subdivision steps, then the filtering solution again applies. As for all knot insertion, the limit curve is unaffected by the increase in multiplicity. This process creates two pieces of limit curve at every multiple knot, each of which meets the other at a Bézier end condition.

If neither of the above solutions are acceptable, then we need a modification of our algorithm which does not increase the multiplicity of multiple knots. This is a topic for future research, but a useful observation is that degree d subdivision with a doubled knot is locally identical to knot insertion for degree $\mathsf{d}-1$. This generalises to higher multiplicities (with correspondingly lower degrees), but it is not immediately clear how the different algorithms should mesh together.

5. Conclusions

We have described an algorithm that doubles the number of knots in a B-spline curve in the spirit of Lane and Riesenfeld's 'refine and smooth' algorithm. Our algorithm has the advantage of working with unequal knot intervals, which the Lane-Riesenfeld algorithm (Lane and Riesenfeld, 1980) does not, and of being symmetric, which the Schaefer non-uniform algorithm (Schaefer and Goldman, 2009) is not. We believe that these factors make the algorithm a promising foundation on which to build a subdivision extension of NURBS.

In future work we will tackle, more comprehensively, the problem of multiple knots discussed in Section 4. We will also take the tensor product of this algorithm and explore modifications that generalise NURBS to include extraordinary points. The question is not whether such generalisations exist, but whether any of them lead to limit surfaces with useful properties.

Appendix A. Proof for odd degree

To help with the proof for odd degree, we introduce the function $s^d(n,i)$, which evaluates the symmetric blossom for degree d by taking n arguments from $\boldsymbol{t}$ and $d-n$ from $\boldsymbol{\tau}$. We have the restriction that $n \leq \frac{d+1}{2}$. When the resulting knots are written in order, the index of the central knot is given by i, and those from $\boldsymbol{t}$ are as close as possible to the centre without omitting a knot from $\boldsymbol{\tau}$ or violating symmetry. For example,

$$\begin{aligned}
s^5(0,3) &= \quad p(\tau_1,\tau_2,\tau_3,\tau_4,\tau_5)\\
s^5(1,3) &= \quad p(\tau_2,\tau_3,t_3,\tau_4,\tau_5)\\
s^5(2,3) &= \quad p(\tau_2,t_2,\tau_3,t_3,\tau_4)\\
s^5(3,3) &= \quad p(t_2,\tau_3,t_3,\tau_4,t_4)\\
s^7(2,4) &= p(\tau_2,\tau_3,t_3,\tau_4,t_4,\tau_5,\tau_6)
\end{aligned}$$

Theorem 1. For odd degree d, $\mathsf{P}^{\mathsf{d}}_{(\mathsf{d}-1)/2} = s^{\mathsf{d}}(\frac{\mathsf{d}-1}{2}, \lfloor\frac{\mathsf{d}-1}{4}\rfloor)$ and $\mathsf{P}^{\mathsf{d}}_{(\mathsf{d}+1)/2} = s^{\mathsf{d}}(\frac{\mathsf{d}+1}{2}, \lfloor\frac{\mathsf{d}+1}{4}\rfloor)$.

These two expressions produce consecutive control points for the B-spline on the refined knot vector. By observation, half of the new control points must have a blossom with $\frac{\mathsf{d}-1}{2}$ knots from $\boldsymbol{t}$, and the other half must have a blossom with $\frac{\mathsf{d}+1}{2}$. For example, the output points in the top row of Figure 2a, where $\mathsf{d}=5$, have either 2 or 3 knots from $\boldsymbol{t}$. It is harder to understand, however, why the second arguments of the s-functions (i.e. the indices of the blossoms' central knots) result in consecutive points.

If a symmetric set of knots has $\frac{\mathsf{d}-1}{2}$ from $\boldsymbol{t}$ where $\frac{\mathsf{d}-1}{2}$ is even, then the central knot comes from $\boldsymbol{\tau}$. Let the index of this knot be i. The next control point will have $\frac{\mathsf{d}+1}{2}$ knots from $\boldsymbol{t}$, which is odd, so the central knot comes from $\boldsymbol{t}$ and the index of this knot must also be i (as $\tau_i \leq t_i \leq \tau_{i+1}$).

On the other hand, if $\frac{\mathsf{d}-1}{2}$ is odd, then the central knot in the first set comes from $\boldsymbol{t}$. Let the index of this knot again be i. The next control point, which will have an even number of knots from $\boldsymbol{t}$, has a central knot from $\boldsymbol{\tau}$ and so the index of this knot must be $i+1$; an index of i would give the previous control point instead of the next.

Now note that $\lfloor \frac{\mathsf{d}+1}{4} \rfloor = 1 + \lfloor \frac{\mathsf{d}-1}{4} \rfloor$ if and only if $\frac{\mathsf{d}-1}{2}$ is odd. Therefore a proof of Theorem 1 will show that the algorithm in Figure 3 produces the required control points for the knot vector $\boldsymbol{\tau} \cup \boldsymbol{t}$ when d is odd. It is sufficient to prove that the algorithm works at the two points $\mathsf{P}^{\mathsf{d}}_{(\mathsf{d}-1)/2}$ and $\mathsf{P}^{\mathsf{d}}_{(\mathsf{d}+1)/2}$, as a proof at any other point is identical after a shift in subscript. In general, the algorithm is invariant under a change in index, so if

$$\mathsf{P}^d_i = s^{\mathsf{d}}(x, q) \quad \text{for any } i,\ x \text{ and } q, \text{ then} \quad \mathsf{P}^d_{i+2} = s^{\mathsf{d}}(x, q+1). \tag{A.1}$$

Proof. We will proceed by induction, with the hypothesis that for all odd δ such that $1 \le \delta \le \mathsf{d}$,

$$\mathsf{P}^{\delta}_{(\delta-1)/2} = s^{\mathsf{d}}\left(\frac{\delta-1}{2}, \left\lfloor \frac{\delta-1}{4} \right\rfloor\right) \quad \text{and} \tag{A.2}$$

$$\mathsf{P}^{\delta}_{(\delta+1)/2} = s^{\mathsf{d}}\left(\frac{\delta+1}{2}, \left\lfloor \frac{\delta+1}{4} \right\rfloor\right). \tag{A.3}$$

From the dual function property, we have that $\mathsf{P}_i = s^{\mathsf{d}}(0, i)$. Therefore $\mathsf{P}^1_{2i} = \mathsf{P}_i = s^{\mathsf{d}}(0, i)$, so $\mathsf{P}^1_0 = s^{\mathsf{d}}(0, 0)$. Furthermore,

$$\begin{aligned}
\mathsf{P}^1_1 &= |u_{1-\mathsf{d}} \ \underline{u_1 \ u_{1+\mathsf{d}}}|\, \mathsf{P}^1_0 + |\underline{u_{1-\mathsf{d}} \ u_1} \ u_{1+\mathsf{d}}|\, \mathsf{P}^1_1 \qquad \text{from line 7 of Figure 3} \\
&= |\tau_{(1-\mathsf{d})/2} \ \underline{t_0 \ \tau_{(1+\mathsf{d})/2}}|\, s^{\mathsf{d}}(0,0) + |\underline{\tau_{(1-\mathsf{d})/2} \ t_0} \ \tau_{(1+\mathsf{d})/2}|\, s^{\mathsf{d}}(0,1) \\
&= s^{\mathsf{d}}(1,0)
\end{aligned}$$

since $\tau_{(1-\mathsf{d})/2}$ is the smallest knot in $s^{\mathsf{d}}(0,0)$ and $\tau_{(1+\mathsf{d})/2}$ is the largest in $s^{\mathsf{d}}(0,1)$. This affine combination of blossoms therefore inserts the knot t_0 into the set of remaining knots, all of which appear in both $s^{\mathsf{d}}(0,0)$ and $s^{\mathsf{d}}(0,1)$. We therefore have the results we need for the base case: $\mathsf{P}^1_0 = s^{\mathsf{d}}(0,0)$ and $\mathsf{P}^1_1 = s^{\mathsf{d}}(1,0)$.

Now consider $\mathsf{P}^{\delta}_{(\delta-1)/2}$ for odd δ. At the relevant iteration of line 10, $\lambda = \frac{\delta-1}{2}$ and $i = \frac{\delta-1}{2}$, so certainly $\lambda \bmod 2 = i \bmod 2$. Therefore, line 11 gives that

$$\begin{aligned}
\mathsf{P}^{\delta}_{(\delta-1)/2} &= \mathsf{P}^{\delta-2}_{(\delta-1)/2} = \mathsf{P}^{\delta-2}_{(\delta-2+1)/2} \\
&= s^{\mathsf{d}}\left(\frac{\delta-2+1}{2}, \left\lfloor \frac{\delta-2+1}{4} \right\rfloor\right) \quad \text{from (A.3)} \\
&= s^{\mathsf{d}}\left(\frac{\delta-1}{2}, \left\lfloor \frac{\delta-1}{4} \right\rfloor\right)
\end{aligned}$$

as required to prove (A.2).

To prove (A.3), consider the point $\mathsf{P}^{\delta}_{(\delta+1)/2}$. We have $\lambda = \frac{\delta-1}{2}$ and $i = \frac{\delta+1}{2}$, therefore $\lambda \bmod 2 \neq i \bmod 2$ and so line 13 of Figure 3 gives us

$$\begin{aligned}
\mathsf{P}^{\delta}_{(\delta+1)/2} &= |u_{\delta-\mathsf{d}} \ \underline{u_\delta \ u_{\mathsf{d}+1}}|\, \mathsf{P}^{\delta-2}_{(\delta-1)/2} + |u_{\delta-\mathsf{d}} \ \underline{u_1 \ u_\delta} \ u_{\mathsf{d}+1}|\, \mathsf{P}^{\delta-2}_{(\delta+1)/2} + |\underline{u_{\delta-\mathsf{d}} \ u_1} \ u_{\mathsf{d}+1}|\, \mathsf{P}^{\delta-2}_{(\delta+3)/2} \\
&= |u_{\delta-\mathsf{d}} \ \underline{u_\delta \ u_{\mathsf{d}+1}}|\, \mathsf{P}^{\delta-2}_{(\delta-1)/2} + |u_{\delta-\mathsf{d}} \ \underline{u_1 \ u_\delta} \ u_{\mathsf{d}+1}|\, \mathsf{P}^{\delta-2}_{((\delta-3)/2)+2} + |\underline{u_{\delta-\mathsf{d}} \ u_1} \ u_{\mathsf{d}+1}|\, \mathsf{P}^{\delta-2}_{((\delta-1)/2)+2} \\
&= |u_{\delta-\mathsf{d}} \ \underline{u_\delta \ u_{\mathsf{d}+1}}|\, s^{\mathsf{d}}(\tfrac{\delta-1}{2}, \lfloor \tfrac{\delta-1}{4} \rfloor) \quad \text{from (A.3)} \\
&\quad + |u_{\delta-\mathsf{d}} \ \underline{u_1 \ u_\delta} \ u_{\mathsf{d}+1}|\, s^{\mathsf{d}}(\tfrac{\delta-3}{2}, \lfloor \tfrac{\delta-3}{4} \rfloor + 1) \quad \text{from (A.2) and (A.1)} \\
&\quad + |\underline{u_{\delta-\mathsf{d}} \ u_1} \ u_{\mathsf{d}+1}|\, s^{\mathsf{d}}(\tfrac{\delta-1}{2}, \lfloor \tfrac{\delta-1}{4} \rfloor + 1) \quad \text{from (A.3) and (A.1)} \\
&= |\tau_{(\delta-\mathsf{d})/2} \ \underline{t_{(\delta-1)/2} \ \tau_{(\mathsf{d}+1)/2}}|\, s^{\mathsf{d}}(\tfrac{\delta-1}{2}, \lfloor \tfrac{\delta-1}{4} \rfloor) \\
&\quad + |\tau_{(\delta-\mathsf{d})/2} \ \underline{t_0 \ t_{(\delta-1)/2}} \ \tau_{(\mathsf{d}+1)/2}|\, s^{\mathsf{d}}(\tfrac{\delta-3}{2}, \lfloor \tfrac{\delta+1}{4} \rfloor) \\
&\quad + |\underline{\tau_{(\delta-\mathsf{d})/2} \ t_0} \ \tau_{(\mathsf{d}+1)/2}|\, s^{\mathsf{d}}(\tfrac{\delta-1}{2}, \lfloor \tfrac{\delta+3}{4} \rfloor) \\
&= |\alpha \ \underline{\gamma \ \eta}|\, A + |\alpha \ \underline{\beta \ \gamma} \ \eta|\, B + |\underline{\alpha \ \beta} \ \eta|\, C
\end{aligned}$$

where

– $\alpha = \tau_{(\delta-\mathsf{d})/2}$, $\beta = t_0$, $\gamma = t_{(\delta-1)/2}$ and $\eta = \tau_{(\mathsf{d}+1)/2}$.
– $A = s^{\mathsf{d}}(\frac{\delta-1}{2}, \lfloor \frac{\delta-1}{4} \rfloor)$, $B = s^{\mathsf{d}}(\frac{\delta-3}{2}, \lfloor \frac{\delta+1}{4} \rfloor)$ and $C = s^{\mathsf{d}}(\frac{\delta-1}{2}, \lfloor \frac{\delta+3}{4} \rfloor)$.

– the blossom for A contains α and β (but not γ or η).
– the blossom for B contains α and η (but not β or γ).
– the blossom for C contains γ and η (but not α or β).
Each of these assertions can be checked by considering the possible forms of s^{d} in A, B and C. We then have

$$\mathsf{P}^{\delta}_{(\delta+1)/2} = |\alpha\ \ \underline{\gamma\ \ \eta}|\, A + |\underline{\alpha\ \ \gamma}\ \ \eta|\,(|\alpha\ \ \underline{\beta\ \ \gamma}|\, B + |\underline{\alpha\ \ \beta}\ \ \gamma|\, C)$$

By the multiaffine property, the blossom for $(|\alpha\ \ \underline{\beta\ \ \gamma}|\, B + |\underline{\alpha\ \ \beta}\ \ \gamma|\, C)$ therefore contains β and η (but not α or γ), and the blossom for $\mathsf{P}^{\delta}_{(\delta+1)/2}$ contains β and γ (but not α or η). This result is identical to $B = s^{\mathsf{d}}(\frac{\delta-3}{2}, \lfloor\frac{\delta+1}{4}\rfloor)$, except with two new knots from $\boldsymbol{t}$ symmetrically added as blossom arguments. The index of the central knot has not changed, and so we have

$$\mathsf{P}^{\delta}_{(\delta+1)/2} = s^{\mathsf{d}}\left(\frac{\delta-3}{2}+2, \left\lfloor\frac{\delta+1}{4}\right\rfloor\right) = s^{\mathsf{d}}\left(\frac{\delta+1}{2}, \left\lfloor\frac{\delta+1}{4}\right\rfloor\right)$$

as required for the second half of the inductive hypothesis (A.3). We have therefore shown the correctness of the algorithm at all odd degrees d. □

Appendix B. Proof for even degree

The proof for even degree follows a similar pattern. Again, we introduce functions that allow us to easily write down general degree blossoms of the required form. $l^d(n,i)$ is the degree d blossom with $n \le d/2$ arguments from $\boldsymbol{t}$ and $d-n$ from $\boldsymbol{\tau}$. When written in order, the knots from $\boldsymbol{t}$ are as close to the centre as possible without missing a knot from $\boldsymbol{\tau}$. Since d is even, there is no symmetrical position for these knots, and $l^d(n,i)$ places them with a bias to the 'left' (i.e. the lower-valued end). A corresponding function $r^d(n,i)$ has a bias to the higher-valued end. In both cases, the index of the lowest knot from $\boldsymbol{t}$ is given by i. This definition requires a special case for when n is 0 — here continuity requires that i is the index of the $(d/2)$th lowest knot for l, or the '$(d/2)+1$'th lowest knot for r. For example,

$$\begin{aligned}
l^6(2,3) &= p(\tau_3, t_3, \tau_4, t_4, \tau_5, \tau_6)\\
l^4(0,3) &= p(\tau_2, \tau_3, \tau_4, \tau_5)\\
r^8(3,2) &= p(\tau_1, \tau_2, t_2, \tau_3, t_3, \tau_4, t_4, \tau_5)\\
r^6(3,1) &= p(\tau_1, t_1, \tau_2, t_2, \tau_3, t_3)
\end{aligned}$$

Theorem 2. For even degree d, $\mathsf{P}^{\mathsf{d}}_{\mathsf{d}/2} = l^{\mathsf{d}}(\frac{\mathsf{d}}{2}, 0)$ and $\mathsf{P}^{\mathsf{d}}_{(\mathsf{d}/2)+1} = r^{\mathsf{d}}(\frac{\mathsf{d}}{2}, 1)$.

$l^{\mathsf{d}}(\frac{\mathsf{d}}{2}, 0)$ and $r^{\mathsf{d}}(\frac{\mathsf{d}}{2}, 1)$ are consecutive control points on the refined knot vector. As for odd degree, a proof at these two points is sufficient to show the correctness of the algorithm at all points.

Proof. The proof for even degree also uses induction. We will show that for even δ such that $0 \le \delta \le \mathsf{d}$,

$$\mathsf{P}^{\delta}_{\delta/2} = l^{\mathsf{d}}(\delta/2, 0) \quad \text{and} \quad \mathsf{P}^{\delta}_{(\delta/2)+1} = r^{\mathsf{d}}(\delta/2, 1) \tag{B.1}$$

which will prove Theorem 2 by using $\delta = \mathsf{d}$.

When $\delta = 0$, we need to show $\mathsf{P}^0_0 = l^{\mathsf{d}}(0,0)$ and $\mathsf{P}^0_1 = r^{\mathsf{d}}(0,1)$. Lines 16 and 17 give us that $\mathsf{P}^0_0 = \mathsf{P}^0_1 = \mathsf{P}_0$. But note that $l^{\mathsf{d}}(0,0) = r^{\mathsf{d}}(0,1)$, and $l^{\mathsf{d}}(0,0) = \mathsf{P}_0$ follows from the correspondence of τ_0 with the edge $\mathsf{P}_{-1}\mathsf{P}_0$. The hypothesis (B.1) therefore holds for $\delta = 0$.

Now consider $\mathsf{P}^{\delta}_{\delta/2}$ for even δ. This point is assigned in line 21, since $\lambda = i = \frac{\delta}{2}$. We have that

$$\begin{aligned}
\mathsf{P}^{\delta}_{\delta/2} &= |u_{\delta-\mathsf{d}}\ \ \underline{u_1\ \ u_{\mathsf{d}+2}}|\, \mathsf{P}^{\delta-2}_{\delta/2} + |\underline{u_{\delta-\mathsf{d}}\ \ u_1}\ \ u_{\mathsf{d}+2}|\, \mathsf{P}^{\delta-2}_{(\delta/2)+1}\\
&= |u_{\delta-\mathsf{d}}\ \ \underline{u_1\ \ u_{\mathsf{d}+2}}|\, \mathsf{P}^{\delta-2}_{((\delta-2)/2)+1} + |\underline{u_{\delta-\mathsf{d}}\ \ u_1}\ \ u_{\mathsf{d}+2}|\, \mathsf{P}^{\delta-2}_{((\delta-2)/2)+2}\\
&= |u_{\delta-\mathsf{d}}\ \ \underline{u_1\ \ u_{\mathsf{d}+2}}|\, r^{\mathsf{d}}(\tfrac{\delta-2}{2}, 1) + |\underline{u_{\delta-\mathsf{d}}\ \ u_1}\ \ u_{\mathsf{d}+2}|\, l^{\mathsf{d}}(\tfrac{\delta-2}{2}, 1) \quad \text{from (B.1)}
\end{aligned}$$

$$
\begin{aligned}
&= |\tau_{(\delta-\mathsf{d})/2} \;\; \underline{t_0 \;\; \tau_{(\mathsf{d}/2)+1}}|\, r^{\mathsf{d}}(\tfrac{\delta-2}{2}, 1) + |\underline{\tau_{(\delta-\mathsf{d})/2} \;\; t_0} \;\; \tau_{(\mathsf{d}/2)+1}|\, l^{\mathsf{d}}(\tfrac{\delta-2}{2}, 1) \\
&= l^{\mathsf{d}}(\tfrac{\delta}{2}, 0) \quad \text{as required.}
\end{aligned}
$$

The lowest knot from $\boldsymbol{t}$ is t_1 in both $r^{\mathsf{d}}(\frac{\delta-2}{2}, 1)$ and $l^{\mathsf{d}}(\frac{\delta-2}{2}, 1)$. The affine combination above inserts the knot t_0, so the number of knots from $\boldsymbol{t}$ has increased by 1 and t_0 is now the lowest knot from $\boldsymbol{t}$. Furthermore, since $\tau_{(\mathsf{d}/2)+1}$ is the highest knot in $l^{\mathsf{d}}(\frac{\delta-2}{2}, 1)$, we can see that the affine combination replaces a high-valued knot from $\boldsymbol{\tau}$ for a low-valued knot from $\boldsymbol{t}$ in $l^{\mathsf{d}}(\frac{\delta-2}{2}, 1)$. The result is therefore another set of knots in the l^{d} pattern, and so the resulting blossom must be $l^{\mathsf{d}}(\frac{\delta}{2}, 0)$.

Now we need to show the second part of the inductive hypothesis (B.1) by considering $\mathsf{P}^{\delta}_{(\delta/2)+1}$. In calculating this point, $\lambda = \frac{\delta}{2}$ but $i = \frac{\delta}{2} + 1$, so from line 23 we have

$$
\begin{aligned}
\mathsf{P}^{\delta}_{(\delta/2)+1} &= |u_{\delta-\mathsf{d}} \;\; \underline{u_{\delta+1} \;\; u_{\mathsf{d}+2}}|\, \mathsf{P}^{\delta-2}_{\delta/2} + |\underline{u_{\delta-\mathsf{d}} \;\; u_{\delta+1}} \;\; u_{\mathsf{d}+2}|\, \mathsf{P}^{\delta-2}_{(\delta/2)+1} \\
&= |u_{\delta-\mathsf{d}} \;\; \underline{u_{\delta+1} \;\; u_{\mathsf{d}+2}}|\, \mathsf{P}^{\delta-2}_{((\delta-2)/2)+1} + |\underline{u_{\delta-\mathsf{d}} \;\; u_{\delta+1}} \;\; u_{\mathsf{d}+2}|\, \mathsf{P}^{\delta-2}_{((\delta-2)/2)+2} \\
&= |u_{\delta-\mathsf{d}} \;\; \underline{u_{\delta+1} \;\; u_{\mathsf{d}+2}}|\, r^{\mathsf{d}}(\tfrac{\delta-2}{2}, 1) + |\underline{u_{\delta-\mathsf{d}} \;\; u_{\delta+1}} \;\; u_{\mathsf{d}+2}|\, l^{\mathsf{d}}(\tfrac{\delta-2}{2}, 1) \quad \text{from (B.1)} \\
&= |\tau_{(\delta-\mathsf{d})/2} \;\; \underline{t_{\delta/2} \;\; \tau_{(\mathsf{d}/2)+1}}|\, r^{\mathsf{d}}(\tfrac{\delta-2}{2}, 1) + |\underline{\tau_{(\delta-\mathsf{d})/2} \;\; t_{\delta/2}} \;\; \tau_{(\mathsf{d}/2)+1}|\, l^{\mathsf{d}}(\tfrac{\delta-2}{2}, 1) \\
&= r^{\mathsf{d}}(\tfrac{\delta}{2}, 1)
\end{aligned}
$$

Verifying that $r^{\mathsf{d}}(\frac{\delta+2}{2}, 1)$ is the result of this affine combination is similar to checking $l^{\mathsf{d}}(\frac{\delta+2}{2}, 0)$ above. This result concludes the proof of (B.1) and hence Theorem 2. □

References

Boehm, W., 1980. Inserting new knots into B-spline curves. Computer-Aided Design 12 (4), 199–201.

Cashman, T. J., Dodgson, N. A., Sabin, M. A., 2007. Non-uniform B-Spline Subdivision Using Refine and Smooth. In: Martin, R. R., Sabin, M. A., Winkler, J. R. (Eds.), 12th IMA Conference on the Mathematics of Surfaces. Vol. 4647 of Lecture Notes in Computer Science. Springer, pp. 121–137.

Cohen, E., Lyche, T., Riesenfeld, R., 1980. Discrete B-splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics. Computer Graphics and Image Processing 14 (2), 87–111.

Lane, J. M., Riesenfeld, R. F., 1980. A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces. IEEE Transactions on Pattern Analysis and Machine Intelligence 2 (1), 35–46.

Müller, K., Reusche, L., Fellner, D., 2006. Extended subdivision surfaces: Building a bridge between NURBS and Catmull-Clark surfaces. ACM Transactions on Graphics 25 (2), 268–292.

Prautzsch, H., 1998. Smoothness of subdivision surfaces at extraordinary points. Advances in Computational Mathematics 9 (3), 377–389.

Ramshaw, L., 1987. Blossoming: A Connect-the-Dots Approach to Splines. Tech. Rep. 19, Digital Systems Research Center.

Ramshaw, L., 1989. Blossoms are polar forms. Computer Aided Geometric Design 6 (4), 323–358.

Schaefer, S., Goldman, R., 2009. Non-uniform Subdivision for B-splines of Arbitrary Degree. Computer Aided Geometric Design 26 (1), 75–81.

Sederberg, T. W., Zheng, J., Sewell, D., Sabin, M., 1998. Non-Uniform Recursive Subdivision Surfaces. Proceedings of SIGGRAPH 98, 387–394.

Stam, J., 2001. On subdivision schemes generalizing uniform B-spline surfaces of arbitrary degree. Computer Aided Geometric Design 18 (5), 383–396.

Vouga, E., Goldman, R., 2007. Two blossoming proofs of the Lane-Riesenfeld algorithm. Computing 79 (2), 153–162.

Warren, J., Weimer, H., 2001. Subdivision Methods for Geometric Design. Morgan Kaufmann.

Zorin, D., Schröder, P., 2001. A unified framework for primal/dual quadrilateral subdivision schemes. Computer Aided Geometric Design 18 (5), 429–454.