S-Splines: A Simple Surface Solution for IGA and CAD

Xin Li[a], Thomas W. Sederberg[b]

[a] *Department of Mathematics, USTC, Hefei, Anhui, China.*
[b] *Department of Computer Science, Brigham Young University, Provo, Utah, USA.*

Abstract

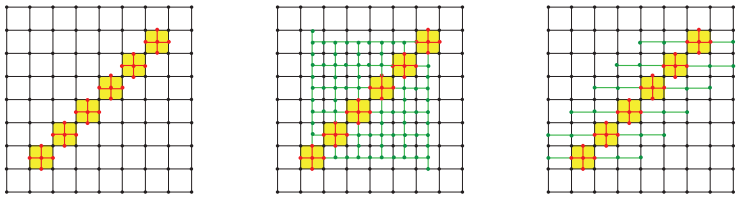
This paper introduces S-spline curves and surfaces. Local refinement of S-spline surfaces is much simpler to understand and to implement than T-spline refinement. Furthermore, no unwanted control points arise in S-spline refinement, unlike T-spline refinement. The refinement algorithm assures linear independence of blending functions. Thus, for isogeometric analysis, S-spline surfaces provide optimal degrees of freedom during adaptive local refinement. S-splines are compatible with NURBS and T-splines, and can easily be added to existing T-spline implementations.

Keywords: T-splines, isogeometric analysis, local refinement, linear independence

1. Introduction

T-splines and Analysis-Suitable T-Splines have been used widely in Isogeometric Analysis (IGA) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. One appeal for using T-splines in IGA is the ability to perform exact local refinement, where *exact* means that the surface is not changed when the new control points are added, and *local* means that the newly-added control points need not comprise an entire row, as in NURBS refinement.

While T-splines refinement is more local than NURBS refinement, it nonetheless generates additional, unrequested control points in all but the simplest cases. Figure 1.a shows an initial control grid of



a. b. c.

Figure 1: Control point propagation in T-splines local refinement algorithms. In Figure a. shows an initial grid of black control points along with 30 red control points that we wish to insert. Figures b. and c. show the refinement produced by the algorithms in [15] and [1] respectively. We did not request the green control points; they were forced upon us by the refinement algorithms.

black control points into which we desire to insert the red control points via a local refinement algorithm. Figure 1.b shows the result obtained from an implementation of the T-splines local refinement

Email addresses: lixustc@ustc.edu.cn (Xin Li), tom.sederberg@gmail.com (Thomas W. Sederberg)

algorithm in [15], and Figure 1.c shows the result using the analysis-suitable T-spline local refinement algorithm [1]. In both cases, some unrequested control points (shown in green) must be added to achieve exact refinement. This propagation of unwanted control points in exact refinement occurs in all previous variations of T-splines, including analysis-suitable T-splines [1, 16, 17, 18, 19], Modified T-splines [20], Weighted T-splines [21] and Truncated T-splines [22].

This paper introduces *S-spline surfaces* — a surface representation that supports an exact refinement algorithm that does not generate *any* additional control points. Thus, for example, Figure 1.a shows the final control point topology after exact local refinement with our new algorithm. We prove that the refinement algorithm preserves linear independence of blending functions. Thus, S-spline surfaces provide minimal degrees of freedoms during adaptive local refinement.

Section 2 introduces the basic concepts of S-splines in the context of curves. S-spline surfaces are presented in Section 3. Local refinement of S-spline surfaces is discussed in Section 4, and Section 5 proves conditions under which linear independence is preserved. An IGA example is presented in Section 6, and Section 7 offers concluding remarks.

Although the paper only discusses S-spline curves and surfaces, the concepts clearly also apply to trivariate volumes. And while we focus on bi-cubic surfaces in this paper, the results can be easily extended to any other degree.

2. S-Spline Curves

An S-spline curve is given by

$$\mathbf{P}(t) = \frac{\sum_{i=1}^{n} \mathbf{P}_i B_i(t)}{\sum_{i=1}^{n} B_i(t)} \tag{1}$$

where

$$B_i(t) = \beta_i B_{\vec{\mathbf{t}}_i}(t). \tag{2}$$

β_i is a scalar constant, and $B_{\vec{\mathbf{t}}_i}(t)$ is a cubic B-spline basis function defined by a local knot vector $\vec{\mathbf{t}}_i = [t_{\tau_i^0}, t_{\tau_i^1}, t_{\tau_i^2}, t_{\tau_i^3}, t_{\tau_i^4}]$. A conventional cubic B-spline curve is a special of an S-spline curve for which

$$\tau_i^1 = \tau_{i+1}^0,\ \tau_i^2 = \tau_{i+1}^1,\ \tau_i^3 = \tau_{i+1}^2,\ \tau_i^4 = \tau_{i+1}^3,\quad i = 1, \ldots, n-1. \tag{3}$$

The string of five increasing integers $\tau_i = [\tau_i^0, \tau_i^1, \tau_i^2, \tau_i^3, \tau_i^4]$ is called the *knot index vector* of $\vec{\mathbf{t}}_i$. $B_{\vec{\mathbf{t}}_i}(t)$ can be evaluated using the de Boor algorithm, or, using the notation $(j,k) = (t_{\tau_i^j} - t_{\tau_i^k})$, $(t,j) = (t - t_{\tau_i^j})$, and $(j,t) = (t_{\tau_i^j} - t)$, $B_{\vec{\mathbf{t}}_i}(t)$ can be written explicitly as

$$B_{\vec{\mathbf{t}}_i}(t) = \begin{cases} 0 & t \le t_{\tau_i^0} \text{ or } t \ge t_{\tau_i^4} \\ \frac{(t,0)^3}{(3,0)(2,0)(1,0)} & t \in [t_{\tau_i^0}, t_{\tau_i^1}] \\ \frac{(t,1)^2(4,t)}{(4,1)(3,1)(2,1)} + \frac{(t,1)(t,0)(3,t)}{(3,0)(3,1)(2,1)} + \frac{(2,t)(t,0)^2}{(3,0)(2,0)(2,1)} & t \in [t_{\tau_i^1}, t_{\tau_i^2}] \\ \frac{(3,t)^2(t,0)}{(3,0)(3,1)(3,2)} + \frac{(3,t)(4,t)(t,1)}{(4,1)(3,1)(3,2)} + \frac{(t,2)(4,t)^2}{(4,1)(4,2)(3,2)} & t \in [t_{\tau_i^2}, t_{\tau_i^3}] \\ \frac{(4,t)^3}{(4,1)(4,2)(4,3)} & t \in [t_{\tau_i^3}, t_{\tau_i^4}] \end{cases} \tag{4}$$

where care must be taken for multiple knots.

We now review refinement of B-spline basis functions and knot insertion for a cubic B-spline curve. Recall that for a B-spline basis function with local knot vector $\vec{\mathbf{t}}_i = [t_{\tau_i^0}, t_{\tau_i^1}, t_{\tau_i^2}, t_{\tau_i^3}, t_{\tau_i^4}]$, a knot $t_a \in (t_{\tau_i^0}, t_{\tau_i^4})$ can be inserted to yield

$$B_{[t_{\tau_i^0}, t_{\tau_i^1}, t_{\tau_i^2}, t_{\tau_i^3}, t_{\tau_i^4}]}(t) = c_1 B_{[t_{\tau_i^0}, t_{\tau_i^1}, t_{\tau_i^2}, t_{\tau_i^3} | t_a]}(t) + c_2 B_{[t_{\tau_i^1}, t_{\tau_i^2}, t_{\tau_i^3}, t_{\tau_i^4} | t_a]}(t), \tag{5}$$

where $[t_{\tau_i^0}, t_{\tau_i^1}, t_{\tau_i^2}, t_{\tau_i^3} | t_a]$ denotes inserting t_a into its proper position in the knot vector $[t_{\tau_i^0}, t_{\tau_i^1}, t_{\tau_i^2}, t_{\tau_i^3}]$, and

$$c_1 = \begin{cases} \frac{t_a - t_{\tau_i^0}}{t_{\tau_i^3} - t_{\tau_i^0}} & \text{if } t_{\tau_i^1} < t_a < t_{\tau_i^3} \\ 1 & \text{if } t_a \geq t_{\tau_i^3} \end{cases} \qquad c_2 = \begin{cases} \frac{t_{\tau_i^4} - t_a}{t_{\tau_i^4} - t_{\tau_i^1}} & \text{if } t_a > t_{\tau_i^1} \\ 1 & \text{if } t_a \leq t_{\tau_i^1} \end{cases} .$$

Using the shorthand notation $\langle 12345 \rangle \equiv B_{[t_1,t_2,t_3,t_4,t_5]}(t)$, a conventional cubic B-spline curve with knot vector $[t_1, t_2, t_3, t_4, t_6, t_7, t_8, t_9]$ can be written

$$\mathbf{P}(t) = \langle 12346 \rangle \mathbf{P}_1 + \langle 23467 \rangle \mathbf{P}_2 + \langle 34678 \rangle \mathbf{P}_3 + \langle 46789 \rangle \mathbf{P}_4. \tag{6}$$

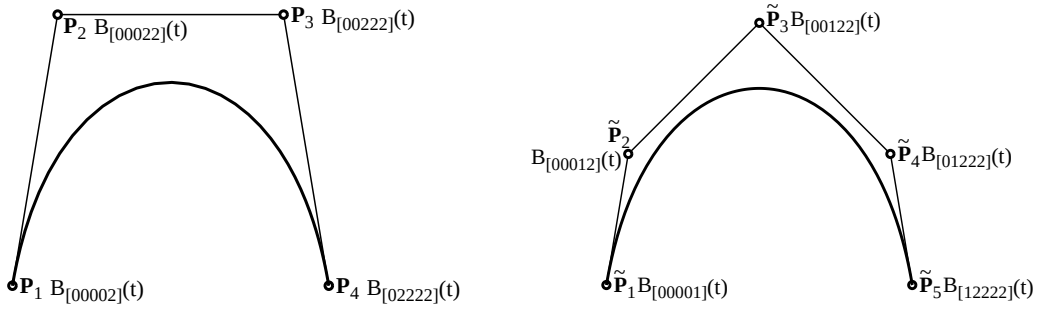
If we insert a knot t_5, the equation of the refined curve is

$$\tilde{\mathbf{P}}(t) = \langle 12345 \rangle \tilde{\mathbf{P}}_1 + \langle 23456 \rangle \tilde{\mathbf{P}}_2 + \langle 34567 \rangle \tilde{\mathbf{P}}_3 + \langle 45678 \rangle \tilde{\mathbf{P}}_4 + \langle 56789 \rangle \tilde{\mathbf{P}}_5$$

To ensure $\mathbf{P}(t) = \tilde{\mathbf{P}}(t)$, the values of the $\tilde{\mathbf{P}}_i$ are determined by applying (5) to each of the blending functions of $\mathbf{P}(t)$ and collecting terms:

$$\begin{array}{l|l} \mathbf{P}(t) & \\ \hline = \langle 12346 \rangle \mathbf{P}_1 & = \langle 12345 \rangle \mathbf{P}_1 + \frac{t_6 - t_5}{t_6 - t_2} \langle 23456 \rangle \mathbf{P}_1 \\ + \langle 23467 \rangle \mathbf{P}_2 & \quad + \frac{t_5 - t_2}{t_6 - t_2} \langle 23456 \rangle \mathbf{P}_2 + \frac{t_7 - t_5}{t_7 - t_3} \langle 34567 \rangle \mathbf{P}_2 \\ + \langle 34678 \rangle \mathbf{P}_3 & \quad + \frac{t_5 - t_3}{t_7 - t_3} \langle 34567 \rangle \mathbf{P}_3 + \frac{t_8 - t_5}{t_8 - t_4} \langle 45678 \rangle \mathbf{P}_3 \\ + \langle 46789 \rangle \mathbf{P}_4 & \quad + \frac{t_5 - t_4}{t_8 - t_4} \langle 45678 \rangle \mathbf{P}_4 + \langle 56789 \rangle \mathbf{P}_4 \\ \hline = \tilde{\mathbf{P}}(t) & = \langle 12345 \rangle \tilde{\mathbf{P}}_1 + \langle 23456 \rangle \tilde{\mathbf{P}}_2 + \langle 34567 \rangle \tilde{\mathbf{P}}_3 + \langle 45678 \rangle \tilde{\mathbf{P}}_4 + \langle 56789 \rangle \tilde{\mathbf{P}}_5 \end{array} \tag{7}$$

From (7) we see that $\mathbf{P}(t) = \tilde{\mathbf{P}}(t)$ if $\tilde{\mathbf{P}}_1 = \mathbf{P}_1$, $\tilde{\mathbf{P}}_2 = \frac{t_6 - t_5}{t_6 - t_2} \mathbf{P}_1 + \frac{t_5 - t_2}{t_6 - t_2} \mathbf{P}_2$, $\tilde{\mathbf{P}}_3 = \frac{t_7 - t_5}{t_7 - t_3} \mathbf{P}_2 + \frac{t_5 - t_3}{t_7 - t_3} \mathbf{P}_3$, $\tilde{\mathbf{P}}_4 = \frac{t_8 - t_5}{t_8 - t_4} \mathbf{P}_3 + \frac{t_5 - t_4}{t_8 - t_4} \mathbf{P}_4$, $\tilde{\mathbf{P}}_5 = \mathbf{P}_4$. Figure 2.a shows a B-spline curve with knot vector $[0, 0, 0, 0, 2, 2, 2, 2]$, and Figure 2.b shows that curve after conventional B-Spline refinement has been performed by inserting a knot at $t = 1$, as described in (7).



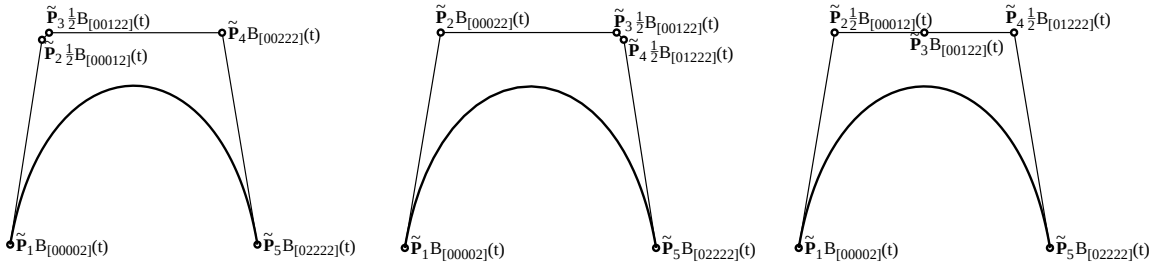
(a) Initial B-Spline curve.

(b) After knot insertion.

Figure 2: B-spline curve before and after knot insertion.

The primary use of B-spline refinement is to create more degrees of freedom. Artists use those degrees of freedom to create a more detailed design, and IGA uses them to decrease approximation error. Equation (7) reminds us that inserting a knot into a cubic B-spline curve involves refining all four blending functions; similarly, refinement of T-spline surfaces involves refinement of numerous blending functions. The development of S-splines was motivated by the simple but powerful observation that extra degrees of freedom can be obtained by refining as few as one or two blending functions, and that, while the resulting curve or surface is not a conventional NURBS, it can be useful for CAD or IGA.

Figure 3.a shows the result of refining only $B_2(t)$ for the B-spline curve in Figure 2.a, and expressing the resulting curve in S-spline form. In Figure 3.b, only $B_3(t)$ is refined, and in Figure 3.c, both $B_2(t)$ and $B_3(t)$ are refined.



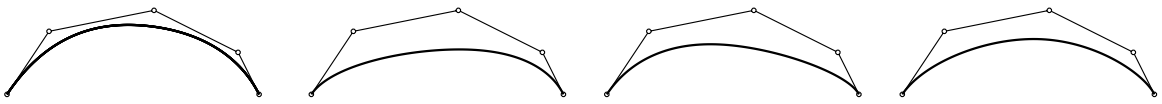
(a) **Class** 1_2. Only $B_2(t)$ is refined. (b) **Class** 1_3. Only $B_3(t)$ is refined. (c) **Class** $2_{2,3}$. $B_1(t)$ and $B_2(t)$ refined.

Figure 3: S-spline curves obtained by Class 1 and Class 2 exact refinement.

In Figures 2 and 3, each control point label is followed by the blending function for that control point. By comparing equations before and after refinement as in (7), we see that the curve in Figure 3.a is identical to the curve in Figure 2.a if $\tilde{\mathbf{P}}_1 = \mathbf{P}_1$, $\tilde{\mathbf{P}}_2 = \mathbf{P}_2$, $\tilde{\mathbf{P}}_3 = \mathbf{P}_2$, $\tilde{\mathbf{P}}_4 = \mathbf{P}_3$, $\tilde{\mathbf{P}}_5 = \mathbf{P}_4$. Likewise, the curve in Figure 3.b is identical to the curve in Figure 2.a if $\tilde{\mathbf{P}}_1 = \mathbf{P}_1$, $\tilde{\mathbf{P}}_2 = \mathbf{P}_2$, $\tilde{\mathbf{P}}_3 = \mathbf{P}_3$, $\tilde{\mathbf{P}}_4 = \mathbf{P}_3$, $\tilde{\mathbf{P}}_5 = \mathbf{P}_4$. The curve in Figure 3.c is identical to the curve in Figure 2.a if $\tilde{\mathbf{P}}_1 = \mathbf{P}_1$, $\tilde{\mathbf{P}}_2 = \mathbf{P}_2$, $\tilde{\mathbf{P}}_3 = (\mathbf{P}_2 + \mathbf{P}_3)/2$, $\tilde{\mathbf{P}}_4 = \mathbf{P}_3$, $\tilde{\mathbf{P}}_5 = \mathbf{P}_4$.

Note that the curves in Figure 3 are not expressible in conventional B-spline form because the blending functions do not satisfy (3), but they are expressible in S-spline form.

Refinement involving the splitting of $i = 1, \ldots, 4$ blending functions will be referred to as Class i refinement, and subscripts such as in Class 1_2 and Class $2_{2,3}$ indicate which blending functions are split to create the S-spline curve. Conventional B-spline curve refinement is Class 4.

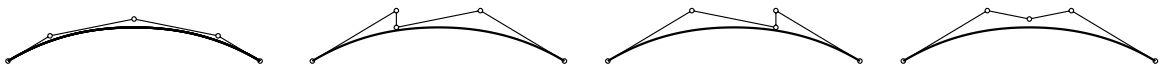


(a) B-spline. (b) Class 1_2. (c) Class 1_3. (d) Class $2_{2,3}$.

Figure 4: S-spline curves with global knot vector $[0, 0, 0, 0, 1, 2, 2, 2, 2]$. Same control points, but blending functions from different classes.

Figure 4.a shows S-spline curves based on the control polygon but with blending functions from different classes. The curves from Class 1 and 2 lie further from the control polygon than the B-spline curve, because some of the blending functions have $\beta = .5$. The blending functions for Class 1 are not symmetric, but they have a useful property that B-spline curves do not: vector $\tilde{\mathbf{P}}_3 - \tilde{\mathbf{P}}_2$ for Class 1_2, and vector $\tilde{\mathbf{P}}_3 - \tilde{\mathbf{P}}_4$ for Class 1_4, are proportional to the jump in third derivative across the knot that appears in their respective blending functions only. Thus, if those vectors are length zero, the knot can be removed.

The blending functions for the curves in Figures 2.b and 3 define the same spline space, so a curve created in any of these classes has an equivalent representation in each of the other classes. Figure 5.a shows a B-spline curve, and Figures 5.b-d show the same curve represented in different classes. All of these classes obey the convex hull property.



(a) B-spline. (b) Class 1_2. (c) Class 1_3. (d) Class $2_{2,3}$.

Figure 5: S-spline curves with global knot vector $[0, 0, 0, 0, 1, 2, 2, 2, 2]$. Same curve, but control points from different classes.

In the examples we have shown for Class $i > 1$ refinement, i blending functions are split into two new blending functions, but only $i + 1$ of those $2i$ blending functions are unique. For example, (7) shows a Class 4 refinement in which the four blending functions are each split to create eight blending functions, but only five are unique. This assures that if an S-spline curve with n control points is refined by inserting a single knot, the refined curve will have at most $n+1$ control points. If the refined curve were to have more than $n + 1$ control points the new set of blending functions would not be linearly independent. A set of i blending functions which generate $i+1$ unique blending functions upon refinement is called *refinement compatible*. Any i adjacent blending functions for a B-spline curve are refinement compatible. In Figure 3.c, the two adjacent blending functions $B_{[00002]}(t)$ and $B_{[00012]}(t)$ are not refinement compatible, because insertion of a knot at, say, 1.5 in each of those blending functions creates four unique blending functions, not three.

Lemma 1. *Given an S-spline curve $\mathbb{S}$ that has linearly independent blending functions, an S-spline curve $\mathbb{S}'$ is obtained by inserting a knot t_a into $\mathbb{S}$ using Class 1, 2, 3, or 4 refinement. Denote by r_1 the largest multiplicity with which t_a occurs in any blending function of $\mathbb{S}$, and by r_2 the multiplicity of t_a in the blending function(s) that are split during the refinement ($r_1 = 0$ means that t_a does not occur in any blending function). If $r_1 = r_2 < 4$, $\mathbb{S}'$ has linearly independent blending functions.*

Proof. We prove that the lemma holds for Class 2 refinement. The proofs for the other classes follow the same structure.

The blending functions before refinement are $B_i(t) = B_{[t_{\tau_i^0}, t_{\tau_i^1}, t_{\tau_i^2}, t_{\tau_i^3}, t_{\tau_i^4}]}(t)$, $i = 1, \ldots, m$ and the central knots before refinement are thus $\{t_{\tau_1^2}, t_{\tau_2^2}, \ldots, t_{\tau_m^2}\}$, not necessarily in numerical order. Arrange the $B_i(t)$ such that the two upon which we perform Class 2 refinement are $B_{m-1}(t)$ and B_m, and insert the new knot t_a into $B_{m-1}(t)$ and B_m. Since $B_{m-1}(t)$ and B_m are refinement compatible, $B_m(t) = B_{[t_{\tau_{m-1}^1}, t_{\tau_{m-1}^2}, t_{\tau_{m-1}^3}, t_{\tau_{m-1}^4}, t_{\tau_m^4}]}(t)$. Denote the blending functions after refinement as $\hat{B}_i(t)$, where $\hat{B}_i(t) = B_i(t)$, $i = 1, \ldots, m-2$, $\hat{B}_{m-1}(t) = B_{[t_{\tau_{m-1}^0}, t_{\tau_{m-1}^1}, t_{\tau_{m-1}^2}, t_{\tau_{m-1}^3} | t_a]}(t)$, $\hat{B}_m(t) = B_{[t_{\tau_{m-1}^1}, t_{\tau_{m-1}^2}, t_{\tau_{m-1}^3}, t_{\tau_{m-1}^4} | t_a]}(t)$, and $\hat{B}_{m+1}(t) = B_{[t_{\tau_{m-1}^2}, t_{\tau_{m-1}^3}, t_{\tau_{m-1}^4}, t_{\tau_m^4} | t_a]}(t)$.

$$B_{m-1}(t) = c_1^{m-1}\hat{B}_{m-1}(t) + c_2^{m-1}\hat{B}_m(t), \quad B_m(t) = c_1^m\hat{B}_m(t) + c_2^m\hat{B}_{m+1}(t) \tag{8}$$

where the c_i^j are given in (5). From (8),

$$\hat{B}_{m-1}(t) = D_1 B_{m-1}(t) + D_2 B_m(t) + D_3\hat{B}_{m+1}(t), \qquad \hat{B}_m(t) = D_4 B_m(t) + D_5\hat{B}_{m+1}(t)$$

where $D_1 = \frac{1}{c_1^{m-1}} \neq 0$, $D_4 = \frac{1}{c_1^m} \neq 0$, and the other D_i are similar functions of the c_i^j.

We prove that the $\hat{B}_i(t)$ are linearly independent by showing that $\sum \hat{B}_i(t)d_i = 0$ only if all $d_i = 0$.

$$\begin{aligned}
0 &= \sum_{i=1}^{m+1} d_i \hat{B}_i(t) \\
&= \sum_{i=1}^{m-2} d_i B_i(t) + d_{m-1}\hat{B}_{m-1}(t) + d_m \hat{B}_m(t) + d_{m+1}\hat{B}_{m+1}(t) \\
&= \sum_{i=1}^{m-2} d_i B_i(t) + d_{m-1}D_1 B_{m-1}(t) + (d_{m-1}D_2 + d_m D_4)B_m(t) + (d_{m-1}D_3 + d_m D_5 + d_{m+1})\hat{B}_{m+1}(t)
\end{aligned}$$

Since the multiplicity of knot t_a in blending function $\hat{B}_{m+1}(t)$ is $r_2 + 1$ and its multiplicity is $< r_2 + 1$ in all other $\hat{B}_i(t)$, the coefficient of $\hat{B}_{m+1}(t)$ must be zero, else the sum could not be zero because it would not be C^3 at t_a. The remaining m terms in the sum form a linear combination of the original m blending functions $B_i(t)$, but they are assumed to be linearly independent, so all of the remaining coefficients of the sum must be zero. Hence,

$$d_{m-1}D_1 = 0, \quad d_{m-1}D_2 + d_m D_4 = 0, \quad d_{m-1}D_3 + d_m D_5 + d_{m+1} = 0.$$

But, $D_1 \neq 0$, so $d_{m-1} = 0$. Likewise, $D_4 \neq 0$, so $d_m = 0$, and this forces $d_{m+1} = 0$. Thus, the only solution to the sum is for all $d_i = 0$, so the $\hat{B}_i(t)$ are linearly independent. □

3. S-Spline Surfaces

An S-spline surface is given by

$$\mathbf{P}(s,t) = \sum_{i=1}^{n} \mathbf{P}_i B_i(s,t) \tag{9}$$

where $\mathbf{P}_i = \omega_i(x_i, y_i, z_i, 1) \in \mathbb{P}^3$ is a control point with Cartesian coordinates (x_i, y_i, z_i) and weight $\omega_i \in \mathbb{R}$. $B_i(s,t)$ is a bivariate blending function defined

$$B_i(s,t) = \beta_i B_{\vec{\mathbf{s}}_i}(s) B_{\vec{\mathbf{t}}_i}(t) \tag{10}$$

where $B_{\vec{\mathbf{s}}_i}(s)$ and $B_{\vec{\mathbf{t}}_i}(t)$ are univariate B-spline blending functions defined as in (4), with knot vectors $\vec{\mathbf{s}}_i = [s_{\sigma_i^0}, s_{\sigma_i^1}, s_{\sigma_i^2}, s_{\sigma_i^3}, s_{\sigma_i^4}]$ and $\vec{\mathbf{t}}_i = [t_{\tau_i^0}, t_{\tau_i^1}, t_{\tau_i^2}, t_{\tau_i^3}, t_{\tau_i^4}]$. The blending function weight β_i is computed algorithmically, while the control point weight ω_i can be user specified. As in the curve case, knot vectors $\vec{\mathbf{s}}_i$ and $\vec{\mathbf{t}}_i$ are associated with knot index vectors $\sigma_i = [\sigma_i^0, \sigma_i^1, \sigma_i^2, \sigma_i^3, \sigma_i^4]$ and $\tau_i = [\tau_i^0, \tau_i^1, \tau_i^2, \tau_i^3, \tau_i^4]$, which are sequences of increasing integers. $\vec{\mathbf{s}}_i$ and $\vec{\mathbf{t}}_i$ are subsequences of global knot vectors $\vec{\mathbf{s}} = [s_0, s_1, \ldots, s_c]$ and $\vec{\mathbf{t}} = [t_0, t_1, \ldots, t_r]$ respectively.

As with T-splines, the control points $\mathbf{P}_i$ are arranged topologically in a control mesh called a T-mesh. The pre-image of an S-spline surface can be diagrammed in the (s,t) parameter plane using what we will call a domain T-mesh. Figure 6 shows an illustration of a domain T-mesh with $c = r = 10$. The region $\mathsf{A} = [2, c-2] \times [2, r-2]$ is highlighted in yellow. The domain T-mesh is a rectangular decomposition of A. Vertices P_i, shown as black filled circles in Figure 6, lie at rectangle corners. Each P_i in the domain T-mesh corresponds to a control point $\mathbf{P}_i$ in the control mesh.

In Figure 6, the strings of five horizontal and vertical integers next to several control points are the index vectors σ_i and τ_i. For example, vertex P_{10} has Cartesian coordinates $(7, 3)$ and blending function

$$B_i(s,t) = B_{[s_3, s_5, s_7, s_8, s_9]}(s) B_{[t_1, t_2, t_3, t_5, t_7]}(t).$$

In the local knot vectors $\vec{\mathbf{s}}_i$ and $\vec{\mathbf{t}}_i$, $s_{\sigma_i^2}$ and $t_{\tau_i^2}$ are called the central knots, and σ_i^2 and τ_i^2 are the central knot indices. In a T-spline, $\vec{\mathbf{s}}_i$ and $\vec{\mathbf{t}}_i$ are inferred from the domain T-mesh [23]. In an S-spline surface, the only relationship that $\vec{\mathbf{s}}_i$ and $\vec{\mathbf{t}}_i$ have with the domain T-mesh is that $P_i = (\sigma_i^2, \tau_i^2)$. In a typical setting, prior to refinement for IGA, the values of the $\vec{\mathbf{s}}_i$ and $\vec{\mathbf{t}}_i$ are those for the initial NURBS or T-spline CAD model (such as in Figure 9.a). The $\vec{\mathbf{s}}_i$ and $\vec{\mathbf{t}}_i$ values are then altered by the local refinement algorithm.

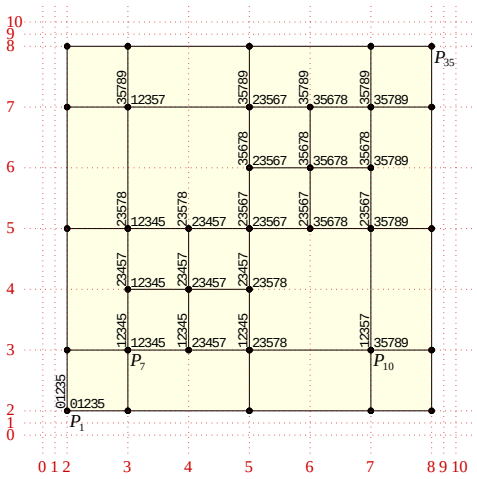


Figure 6: Example of a Domain T-mesh

4. Exact Refinement

We chose the name **S**-splines because exact refinement for S-spline surfaces is significantly more **S**imple to understand and implement than for T-splines. Section 4.1 discusses the insertion of a single control point, and Section 4.2 discusses how to split one or more faces into four faces.

4.1. Inserting a single control point

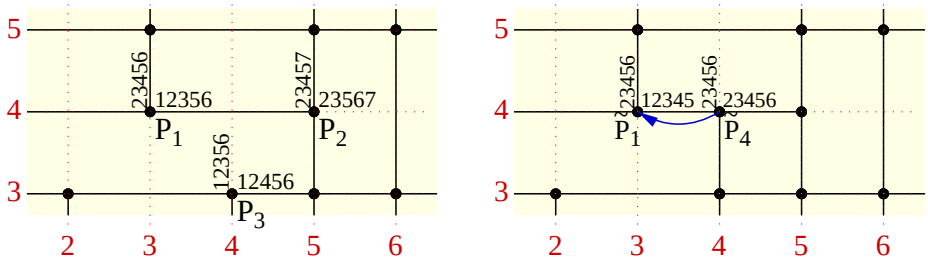
Figure 7.a shows a region of a T-mesh into which we wish to insert a new control point $\tilde{P}_4$ with knot index coordinates $(4, 4)$. There are three ways we can do this using Class 1 refinement. Figure 7.b shows the result of splitting the blending function $B_1(s,t) = B_{[s_1,s_2,s_3,s_5,s_6]}(s)B_{[t_2,t_3,t_4,t_5,t_6]}(t)$ by inserting a knot s_4 into $B_{[s_1,s_2,s_3,s_5,s_6]}(s)$. To assure exact refinement (i.e., that the refined surface is identical to the initial surface), we require

$$\mathbf{P}_1 B_1(s,t) = \tilde{\mathbf{P}}_1 \tilde{B}_1(s,t) + \tilde{\mathbf{P}}_4 \tilde{B}_4(s,t) \tag{11}$$

where $\tilde{B}_1(s,t) = \beta_1 B_{[s_1,s_2,s_3,s_4,s_5]}(s)B_{[t_2,t_3,t_4,t_5,t_6]}(t)$ and $\tilde{B}_4(s,t) = \beta_4 B_{[s_2,s_3,s_4,s_5,s_6]}(s)B_{[t_2,t_3,t_4,t_5,t_6]}(t)$. From (5),

$$B_{[s_1,s_2,s_3,s_5,s_6]}(s) = \frac{s_4 - s_1}{s_5 - s_1} B_{[s_1,s_2,s_3,s_4,s_5]}(s) + \frac{s_6 - s_4}{s_6 - s_2} B_{[s_2,s_3,s_4,s_5,s_6]}(s),$$

so (11) is satisfied if $\tilde{\mathbf{P}}_1 = \tilde{\mathbf{P}}_4 = \mathbf{P}_1$, $\beta_1 = \frac{s_4 - s_1}{s_5 - s_1}$, and $\beta_4 = \frac{s_6 - s_4}{s_6 - s_2}$. Figures 7.c and d show two alternatives for adding a control point at $(4, 4)$ via Class 1 refinement. The blue arc in each of Figures 7.b–d originate at the "child" control point (the new control point) and points at their "parent" control point, or the



(a) Original Domain T-mesh. (b) Refine only $B_{\vec{\mathbf{s}}_1}(s)$.

(c) Refine only $B_{\vec{\mathbf{s}}_2}(s)$. (d) Refine only $B_{\vec{\mathbf{t}}_3}(t)$.

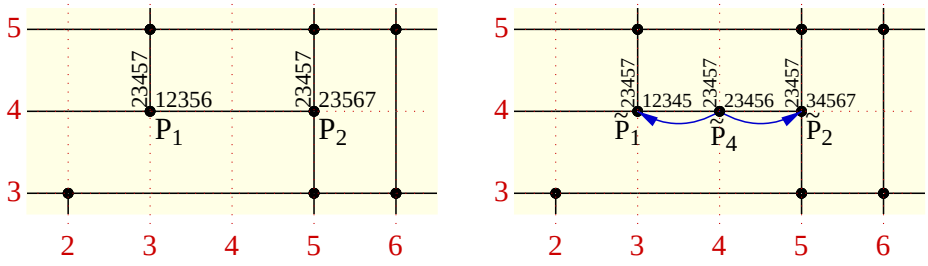
Figure 7: Inserting a control point at knot index coordinates (4,4) using Class 1 refinement.

one whose blending function was split. For exact Class 1 refinement, both child and parent should be positioned at the original coordinates of the parent.

Class 2 refinement can be performed on two blending functions $B_i(s,t) = \beta_i B_{\vec{\mathbf{s}}_i}(s) B_{\vec{\mathbf{t}}_i}(t)$ and $B_j(s,t) = \beta_j B_{\vec{\mathbf{s}}_j}(s) B_{\vec{\mathbf{t}}_j}(t)$ that are refinement compatible. This means either that $\vec{\mathbf{t}}_i$ and $\vec{\mathbf{t}}_j$ have four knots in common, and $\vec{\mathbf{s}}_i = \vec{\mathbf{s}}_j$; or that $\vec{\mathbf{s}}_i$ and $\vec{\mathbf{s}}_j$ have four knots in common, and $\vec{\mathbf{t}}_i = \vec{\mathbf{t}}_j$. In Figure 7.a, $B_1(s,t)$ and $B_2(s,t)$ are not refinement compatible because $\vec{\mathbf{t}}_i \neq \vec{\mathbf{t}}_j$, but they are refinement compatible in Figure 8.a.

Inserting the knot s_4 into $B_{\vec{\mathbf{s}}_1}(s)$ and $B_{\vec{\mathbf{s}}_2}(s)$ and collecting terms, we achieve exact refinement by setting

$$\tilde{\mathbf{P}}_1 = \mathbf{P}_1, \quad \tilde{\mathbf{P}}_2 = \mathbf{P}_2, \quad \tilde{\mathbf{P}}_4 = \frac{(s_6 - s_4)\mathbf{P}_1 + (s_4 - s_2)\mathbf{P}_2}{s_6 - s_2}, \quad \beta_1 = \frac{s_4 - s_1}{s_5 - s_1}, \quad \beta_2 = 1, \quad \beta_4 = \frac{s_7 - s_4}{s_7 - s_3}.$$



(a) Original T-mesh. (b) After Class 2 refinement.

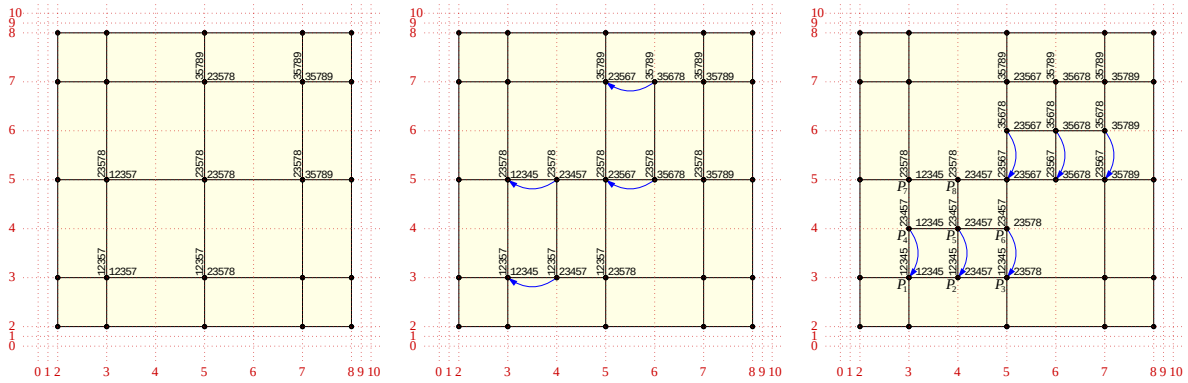
Figure 8: Inserting a control point at knot index coordinates (4,4) using Class 2 refinement.

As in Figure 7, the blue arcs in Figure 8.b indicate which blending functions were split to create the new blending function. If there are three or four refinement-compatible blending functions, it is possible to perform Class 3 or 4 refinement.

4.2. Splitting a face into fourths

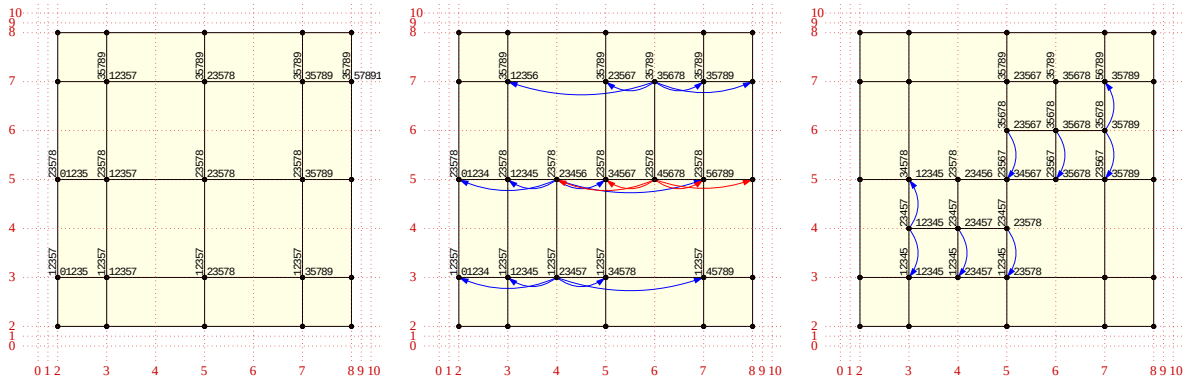
As illustrated in Figures 13.d–f, refinement for IGA applications typically consists of recursively splitting faces into fourths. This is easily accomplished using Class 1 refinements. Figure 9.a shows a B-spline control grid and knot index vectors for seven blending functions. In Figure 9.b, four new control points were created using Class 1 refinements, and in Figure 9.c, six additional control points were created.

For exact refinement, i.e., the geometry is not altered, none of the original control points in Figure 9.c are moved, and $\mathbf{P}_2 = \mathbf{P}_4 = \mathbf{P}_5 = \mathbf{P}_1$, $\mathbf{P}_6 = \mathbf{P}_3$, $\mathbf{P}_8 = \mathbf{P}_7$, etc.



(a) Original B-spline. (b) S-spline after first insertions. (c) S-spline after second insertions.

Figure 9: Splitting control mesh faces into fourths using Class 1 refinements.



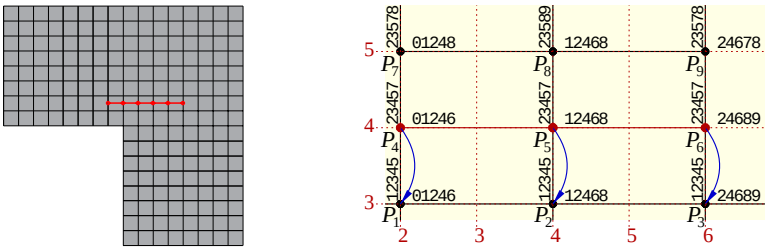
(a) Original B-spline. (b) Four Class 4 refinements. (c) Two Class 2 and four Class 1 refinements.

Figure 10: Splitting control mesh faces into fourths using maximum possible classes.

An alternative refinement strategy is to insert each new control point using the maximum class possible. In Figure 10.b, four new control points are inserted using Class 4 refinements. In Figure 10.c, two control points are inserted using Class 2 refinement, and four are inserted using Class 1.

5. Linear Independence

As illustrated in Figures 9 and 13, the pattern of refinement typically used in IGA is to split faces into fourths. Our proof of linear independence is facilitated if all new control points along a new knot line are inserted simultaneously, such as the six new control points in Figure 11.a. Our discussion focuses on new rows, but applies to new columns as well.



(a) Original domain T-mesh. (b) Inserting $\mathbf{P}_4$, $\mathbf{P}_5$, and $\mathbf{P}_6$ using Class 1 refinements.

Figure 11: Inserting a row of control points.

Definition 1. Row Linear Independence. *Expanding the equation of a blending function (10),*

$$B_i(s,t) = \beta_i B_{\vec{\mathbf{s}}_i}(s) B_{\vec{\mathbf{t}}_i}(t) \quad = \quad \beta_i B_{[s_{\sigma_i^0}, s_{\sigma_i^1}, s_{\sigma_i^2}, s_{\sigma_i^3}, s_{\sigma_i^4}]}(s) B_{[t_{\tau_i^0}, t_{\tau_i^1}, t_{\tau_i^2}, t_{\tau_i^3}, t_{\tau_i^4}]}(t),$$

each $B_i(s,t)$ associated with a row with knot index k have central knot index $\tau_i^2 = k$. Define $\mathbb{B}_k^R = \{B_{\vec{\mathbf{s}}_i}(s) | \tau_i^2 = \gamma\}$ for all rows. Likewise, for all columns, define $\mathbb{B}_k^C = \{B_{\vec{\mathbf{t}}_i}(t) | \sigma_i^2 = k\}$. An S-spline is said to be row linearly independent if the blending functions in each set $\mathbb{B}_k^R$ and $\mathbb{B}_k^C$ are linearly independent.

Any T-spline (or NURBS) surface is row linearly independent, because the rules for inferring blending functions from a T-mesh [23] yield blending functions $B_{\vec{\mathbf{s}}_i}(s)$ along any row of control points (or, $B_{\vec{\mathbf{t}}_i}(t)$ along any column of control points) that are equivalent to blending functions for a NURBS curve.

Interestingly, an S-spline surface that has linearly independent blending functions is not necessarily row linearly independent. But, if we start with a NURBS or T-Spline surface and repeatedly perform Row Insertion operations, the resulting S-spline surface will be both row linearly independent and have linearly independent blending functions, as Lemma 2 and Theorem 1 show.

Definition 2. Row Insertion *is the operation of inserting a row of new control points, as in Figure 11.a, with the following conditions:*

1. *$r_2 \geq r_1$, where r_1 the largest multiplicity with which the new knot (t_4 in Figure 11.b) occurs in any local knot vectors $\vec{\mathbf{t}}_i$, and r_2 is the multiplicity of that knot in local knot vectors $\vec{\mathbf{t}}_i$ of the blending functions that are split during the refinement ($r_1 = 0$ means that t_a does not occur in any blending function).*
2. *Referring to Figure 11.b, the local knot vectors $\vec{\mathbf{s}}_i$ for P_4, P_5, and P_6 must either be the same as those for P_1, P_2, and P_3, respectively, or for those of P_7, P_8, and P_9, respectively. In the example in Figure 11.b, this condition is met by doing Class 1 refinements on Row 3. It could also have been met by doing Class 1 refinements on Row 5. If blending functions on Row 3 are refinement compatible with blending functions on Row 5, the condition could also be satisfied by performing refinements of Class ≥ 2.*

Lemma 2. *If Row Insertion is performed on an S-spline* $\mathbf{P}(s,t)$ *that is row linearly independent, the resulting S-spline* $\hat{\mathbf{P}}(s,t)$ *is row linearly independent.*

Proof. Refer to Figure 11.b and consider the step of inserting points P_4, P_5, and P_6 along knot line t_4. By the definition of Row Insertion, t_4 does not appear in any previous blending functions, and the $B_{\vec{\mathbf{s}}_i}(s)$ functions for P_4, P_5, and P_6 are simple scales of the $B_{\vec{\mathbf{s}}_i}(s)$ functions for P_1, P_2, and P_3. Therefore, the blending functions for P_4, P_5, and P_6 are linearly independent. The new $B_{\vec{\mathbf{s}}_i}(s)$ functions for P_1, P_2, and P_3 are simple scales of their initial values, so the blending functions along t_3 remain linearly independent. Since the insertion operation does not change the $B_{\vec{\mathbf{s}}_i}(s)$ values for any other row, the blending functions along all other horizontal knot lines are also linearly independent.

The only columns whose $B_{\vec{\mathbf{t}}_i}(t)$ blending functions change are those with knot indices 2, 4, and 6. But the refinements of those blending functions amount to refinement of S-spline curves, so by Lemma 1, the blending functions along those columns are linear independent. All other columns remain unchanged, and thus remain linear independent. □

Theorem 1. *If Row Insertion is performed on an S-spline* $\mathbf{P}(s,t)$ *with linearly independent blending functions and that is also row linearly independent, the blending functions of the resulting S-spline surface* $\tilde{\mathbf{P}}(s,t)$ *are linearly independent.*

Proof. Let t_λ be the t knot value along which the new row of control points is inserted. Denote the initial n blending functions $B_i(s,t) = \beta_i B_{\vec{\mathbf{s}}_i}(s) B_{\vec{\mathbf{t}}_i}(t)$. After Row Insertion, these blending functions are changed to $\hat{B}_i(s,t)$. The blending functions for the m new control points are $\tilde{B}_i(s,t) = \tilde{\beta}_i \tilde{B}_{\vec{\mathbf{s}}_i}(s) \tilde{B}_{\vec{\mathbf{t}}_i}(t)$.

Denote $S_1 = \{B_1(s,t), \ldots, B_n(s,t), \tilde{B}_1(s,t), \ldots, \tilde{B}_m(s,t)\}$ and $S_2 = \{\hat{B}_1(s,t), \ldots, \hat{B}_n(s,t), \tilde{B}_1(s,t), \ldots, \tilde{B}_m(s,t)\}$ and let $\mathbb{S}_1$ and $\mathbb{S}_2$ be the linear space spanned by the sets S_1 and S_2, respectively.

We prove that the blending functions in S_1 are linearly independent by seeking c_i and d_j such that

$$\sum_{i=1}^{n} c_i B_i(s,t) + \sum_{j=1}^{m} d_j \tilde{B}_j(s,t) \equiv 0. \tag{12}$$

Using r_2 as defined in Definition 2, take the partial derivative,

$$\left. \frac{\partial^{3-r_2} \sum_{j=1}^{m} d_j \tilde{B}_j(s,t)}{\partial t} \right|_{t_\lambda^-}^{t_\lambda^+} = -\left. \frac{\partial^{3-r_2} \sum_{i=1}^{n} c_i B_i(s,t)}{\partial t} \right|_{t_\lambda^-}^{t_\lambda^+}. \tag{13}$$

where $|_{t_\lambda^-}^{t_\lambda^+}$ denotes the difference in the limits of the partial derivatives at t_λ as approached from the left and from the right. Since all $B_i(s,t)$ have continuous derivatives of order $3 - r_d$ at $t = t_\lambda$, the right side of (13) equals zero so

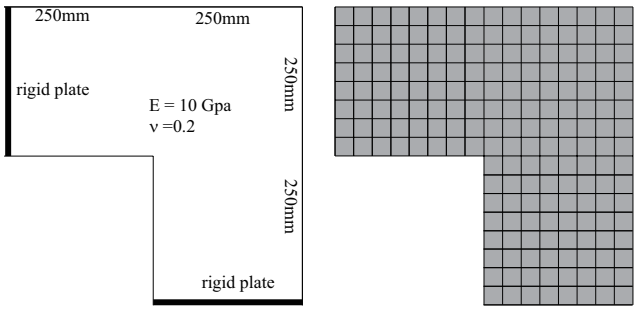
$$\left. \frac{\partial^3 \sum_{j=1}^{m} d_j \tilde{B}_j(s,t)}{\partial t} \right|_{t_\lambda^-}^{t_\lambda^+} = \sum_{j=1}^{m} d_j \tilde{\beta}_j \tilde{B}_{\vec{\mathbf{s}}_j}(s) \left(\left. \frac{\partial^3 \tilde{B}_{\vec{\mathbf{t}}_j}(t)}{\partial t} \right|_{t_\lambda^-}^{t_\lambda^+} \right) = \sum_{j=1}^{m} d_j \tilde{\beta}_j \tilde{e}_j \tilde{B}_{\vec{\mathbf{s}}_j}(s) = 0, \tag{14}$$

where $\tilde{e}_j = \left(\left. \frac{\partial^3 \tilde{B}_{\vec{\mathbf{t}}_j}(t)}{\partial t} \right|_{t_\lambda^-}^{t_\lambda^+} \right)$. Since $\tilde{B}_{\vec{\mathbf{t}}_j}(t)$ always has a discontinuous derivative of order $3 - r_d$ at $t = t_\lambda$ for any $\vec{\mathbf{t}}_j$, all $\tilde{e}_j \neq 0$. Since $\tilde{\beta}_j$ is only assigned by the refinement algorithm, all $\tilde{\beta}_j \neq 0$. By Lemma 2, $\tilde{\mathbf{P}}(s,t)$ is row linearly independent, and since the $\tilde{B}_{\vec{\mathbf{s}}_j}(s)$, $i = 1, \ldots, m$ are the blending functions on the row $t = t_\lambda$, they are linearly independent. Therefore, the only way (14) can be satisfied is for $(d_j \tilde{\beta}_j \tilde{e}_j) = 0$, $j = 1, \ldots, m$. But since $\tilde{\beta}_j$ and $\tilde{e}_j$ are never zero, we must have $d_j = 0$, $j = 1, \ldots, m$ and

(12) leaves us $\sum_{i=1}^{n} c_i B_i(s,t) = 0$. According to the hypothesis, the $B_i(s,t)$ are linearly independent, so we must have $c_i = 0$, $i = 1, \ldots, n$. Therefore, the blending functions in S_1 are linearly independent and $\dim(\mathbb{S}_1) = n + m$. Since any blending function in S_1 can be represented as a linear combination of the blending functions in S_2, $\mathbb{S}_1 \subseteq \mathbb{S}_2$, which implies $\dim(\mathbb{S}_2) \geq n + m$. However, since $\mathbb{S}_2$ is a linear space spanned by $n+m$ blending functions, $\dim(\mathbb{S}_2) \leq n+m$. Thus, $\dim(\mathbb{S}_2) = n+m$ and the blending functions in S_2 are also linearly independent. □

6. Example of Using S-Splines for IGA

This section reports the results of applying S-spline local refinement to an IGA analysis of an implicit gradient damage model. Continuum damage models are widely used for the simulation of diffuse fracture processes [24]. Among the gradient damage formulations, the implicit gradient enhancement is considered the most successful [25]. This issue is improved using isogeometric finite elements to overcome the problems associated with the use of mixed formulations and meshless methods for gradient damage formulations [4]. In this section, we will consider the L-shaped specimen shown in Figure 12 using the gradient damage formulations in [4]. In this example Young's modulus is 10 GPa and Poisson's ratio is 0.2. The left and bottom edges are modeled as rigid plates. Pinned roller supports are placed at the left and bottom ends of the bottom and left rigid plates respectively. This ensures that only a rigid rotation is allowed at the support. The supports are then displaced symmetrically in the x and y directions as shown. The problem configuration and loading ensures that the damage field starts developing at the concave corner and migrates at a forty-five degree angle towards the upper right corner of the domain. Increasing resolution only in the vicinity of the damage field is accomplished using S-spline local refinement.



a. Problem description. b. Initial T-mesh.

Figure 12: Implicit gradient damage problem over an L-shaped domain.

We compare the efficiency of the old and new refinement algorithms in Figure 13 in terms of the number of control points. At each refinement level, we split the neighboring five faces along the diagonal edges into four faces. Additional T-mesh topology is then added by the refinement algorithms to ensure nestedness in the resulting refined space. The resulting T-meshes generated by the algorithm in [15] are shown in Figure 13 a, b, and c; those generated by the algorithm in [1] are shown in Figure 13 d,e, and f; and those generated by the refinement algorithms in this paper are shown in Figure 13 g, h, and i respectively. We can see that the new algorithm converges to an optimal refinement. requested refinement pattern.

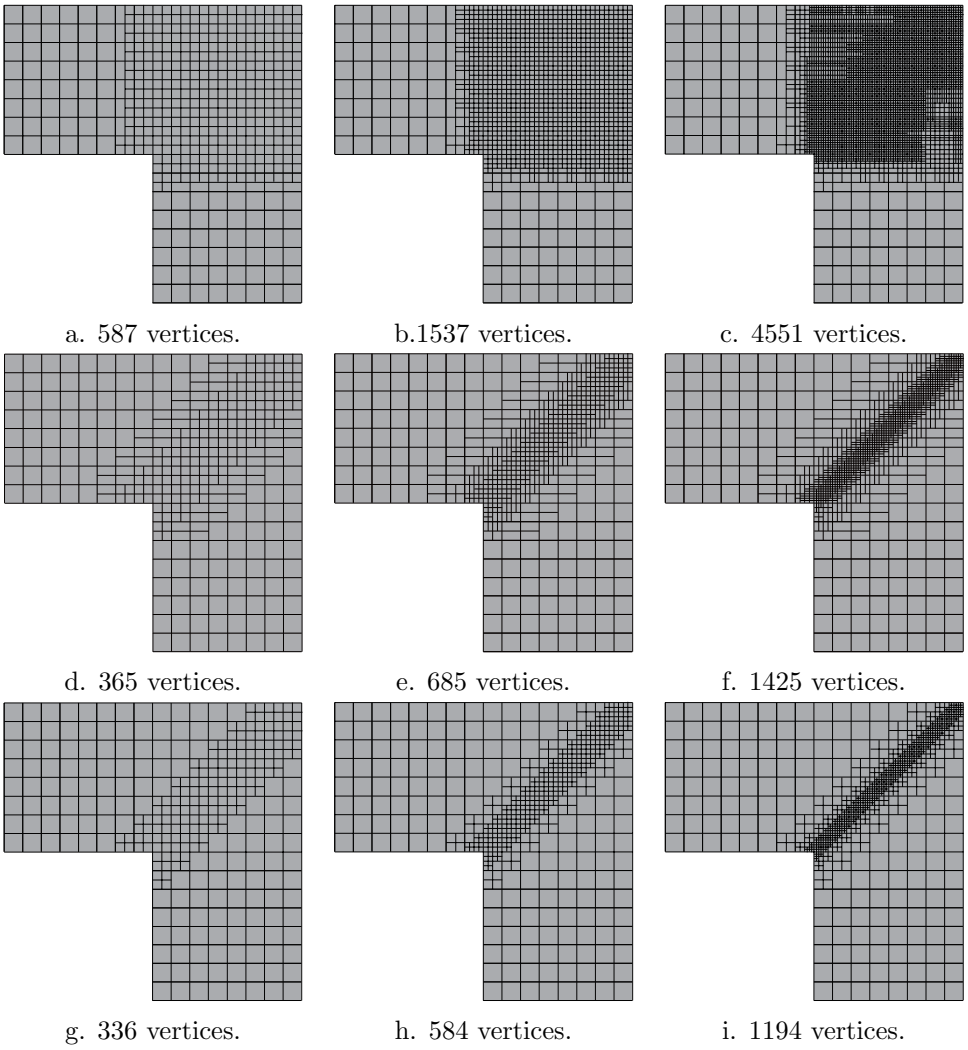


Figure 13: Sequences of refined T-meshes. Images a, b, and c are meshes generated by the algorithm in [15]; d, e, and f were generated by the algorithm in [1]; and g, h, and i were generated by the algorithm in this paper.

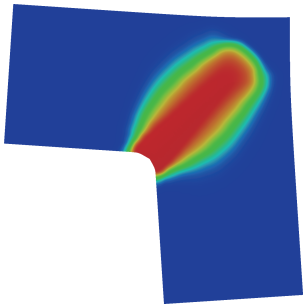


Figure 14: Contour plot of the sixth-order damage field obtained using the new S-spline refinement algorithm.

7. Conclusions and future work

S-spline surfaces show promise for use in IGA. A major advantage that S-splines have over T-splines is that exact local refinement can be performed without propagating unwanted control points. Furthermore, the refinement algorithm for S-splines is significantly more simple to understand and to implement than for T-splines.

Future IGA-related research is needed to study the conditioning of the matrices, extend S-splines to handle extraordinary points, and run performance tests against competing local refinement strategies such as AS-T-splines and hierarchical B-splines. Trivariate S-splines should have the same minimal-degree-of-freedom advantages for volumetric IGA as S-spline surfaces have for shell elements.

S-splines also show promise for use in CAD. An ongoing complaint that users of T-splines have expressed is the extra control points that usually arise when doing T-splines local refinement. Designers often find these extra control points confusing. Since S-splines avoid this problem, they might be attractive to designers in the CAD industry. A drawback of Class 1 refinement is that it results in two control points having the same Cartesian coordinates and with smaller weights. It remains to be seen if a friendly user interface can mitigate this problem.

The use of S-splines for shape optimization problems is an obvious application that should also be explored. Since control points are manipulated algorithmically in shape optimization, a friendly user interface is not needed.

Acknowledgements

The first author was supported by the NSF of China (No.60873109, No.60903148), Chinese Universities Scientific Fund, and Chinese Academy of Science (Startup Scientific Research Foundation).

[1] M. A. Scott, X. Li, T. W. Sederberg, T. J. R. Hughes, Local refinement of analysis-suitable T-splines, Computer Methods in Applied Mechanics and Engineering 213-216 (2012) 206–222.

[2] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, T. W. Sederberg, Isogeometric analysis using T-splines, Computer Methods in Applied Mechanics and Engineering 199 (5-8) (2010) 229 – 263.

[3] M. A. Scott, M. J. Borden, C. V. Verhoosel, T. J. R. Hughes, Isogeometric Finite Element Data Structures based on Bézier Extraction of T-splines, International Journal for Numerical Methods in Engineering, 86 (2011) 126–156.

[4] C. V. Verhoosel, M. A. Scott, T. J. R. Hughes, de Borst, R., An isogeometric analysis approach to gradient damage models, International Journal for Numerical Methods in Engineering, 86(1) (2011) 115–134.

[5] C. V. Verhoosel, M. A. Scott, R. de Borst, T. J. R. Hughes, An isogeometric approach to cohesive zone modeling, International Journal for Numerical Methods in Engineering, 87 (2011) 336–360.

[6] M. J. Borden, C. V. Verhoosel, M. A. Scott, T. J. R. Hughes, C. M. Landis, A phase-field description of dynamic brittle fracture, Computer Methods in Applied Mechanics and Engineering 217-220 (2012) 77–95.

[7] D. J. Benson, Y. Bazilevs, E. De Luycker, M. C. Hsu, M. A. Scott, T. J. R. Hughes, T. Belytschko, A generalized finite element formulation for arbitrary basis functions: from isogeometric analysis to XFEM, International Journal for Numerical Methods in Engineering 83 (2010) 765–785.

[8] M. Dörfel, B. Jüttler, B. Simeon, Adaptive isogeometric analysis by local h-refinement with T-splines, Computer Methods in Applied Mechanics and Engineering 199 (5-8) (2009) 264–275.

[9] L. B. Veiga, A. Buffa, D. C. G. Sangalli, Isogeometric analysis using T-splines on two-patch geometries, Comput. Methods Appl. Mech. Engrg. 200 (2011) 1787–1803.

[10] A. Buffa, D. Cho, M. Kumar, Characterization of T-splines with reduced continuity order on T-meshes, Comput. Methods Appl. Mech. Engrg. 201-204 (2012) 112–126.

[11] M. A. Scott, R. N. Simpson, J. A. Evans, S. Lipton, S. P. A. Bordas, T. J. R. Hughes, T. W. Sederberg, Isogeometric boundary element analysis using unstructured T-splines, Comput. Methods Appl. Mech. Engrg. 254 (2013) 197–221.

[12] R. Dimitri, L. D. Lorenzis, M. A. Scott, P. Wriggers, R. L. Taylor, G. Zavarise, Isogeometric large deformation frictionless contact using T-splines, Computer Methods in Applied Mechanics and Engineering 269 (2014) 394–414.

[13] L. Liu, Y. Zhang, T. J. R. Hughes, M. A. Scott, T. W. Sederberg, Volumetric T-spline construction using boolean operations, Engineering with Computers 30 (2014) 425–439.

[14] D. Schillinger, L. Dede, M. A. Scott, J. A. Evans, M. J. Borden, E. Rank, T. J. R. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of nurbs, immersed boundary methods, and T-spline cad surfaces, Computer Methods in Applied Mechanics and Engineering 249-252 (2014) 116–150.

[15] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng, T. Lyche, T-spline simplification and local refinement, ACM Transactions on Graphics 23 (3) (2004) 276–283.

[16] X. Li, J. Zheng, T. W. Sederberg, T. J. R. Hughes, M. A. Scott, On the linear independence of T-splines blending functions, Computer Aided Geometric Design, 29 (2012) 63–76.

[17] X. Li, M. A. Scott, Analysis-suitable T-splines: characterization, refinablility and approximation, Mathematical Models and Methods in Applied Sciences 24(06) (2014) 1141–1164.

[18] L. B. Veiga, A. Buffa, D. C. G. Sangalli, Analysis-suitable T-splines are dual-compatible, Comput. Methods Appl. Mech. Engrg 249-252 (2012) 42–51.

[19] L. B. Veiga, A. Buffa, G. Sangalli, R. Vazquez, Analysis-suitable T-splines of arbitrary degree: definition and properties, Mathematical Models and Methods in Applied Sciences 23 (2013) 1979–2003.

[20] H. Kang, F. Chen, J. Deng, Modified T-splines, Computer Aided Geometric Design 30 (2013) 827–843.

[21] L. Liu, Y. J. Zhang, X. Wei, Weighted T-splines with application in reparameterizing trimmed NURBS surfaces, Computer Methods in Applied Mechanics and Engineering 295 (2015) 108 – 126.

[22] X. Wei, Y. Zhanga, L. Liu, T. J. Hughes, Truncated T-splines: Fundamentals and methods, Computer Methods in Applied Mechanics and Engineering.

[23] T. W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCSs, ACM Transactions on Graphics 22 (3) (2003) 477–484.

[24] J. Lemaitre, J. Chaboche, Mechanics of solid materials, Cambridge University Press: Cambridge, 1990.

[25] R. Peerlings, R. de Borst, W. Brekelmans, J. de Vree, Gradient enhanced damage for quasi-brittle materials, International Journal for Numerical Methods in Engineering 39 (1996) 3391–3403.