Multilevel Refinable Triangular PSP-Splines (Tri-PSPS)

Qingde Li

Department of Computer Science, University of Hull, Hull, HU6 7RX, UK

Jie Tian

Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China

Abstract

A multi-level spline technique known as partial shape preserving splines (PSPS) [12] has recently been developed for the design of piecewise polynomial freeform geometric surfaces, where the basis functions of the PSPS can be directly built from an arbitrary set of polygons that partitions a giving parametric domain. This paper addresses a special type of PSPS, the triangular PSPS (Tri-PSPS), where all spline basis functions are constructed from a set of triangles. Compared with other triangular spline techniques, Tri-PSPS have several distinctive features. Firstly, for each given triangle, the corresponding spline basis function for any required degree of smoothness can be expressed in closed-form and directly written out in full explicitly as piecewise bivariate polynomials. Secondly, Tri-PSPS are an additive triangular spline technique, where the spline function built from a given triangle can be replaced with a set of refined spline functions built on a set of smaller triangles that partition the initial given triangle. In addition, Tri-PSPS are a multilevel spline technique, Tri-PSPS surfaces can be designed to have a continuously varying levels of detail, achieved simply by specifying a proper value for the smoothing parameter introduced in the spline functions. In terms of practical implementation, Tri-PSPS are a parallel computing friendly spline scheme, which can be easily implemented on modern programmable GPUs or on high performance computer clusters, since each of the basis functions of Tri-PSPS can be directly computed independent of each other in parallel.

Keywords: Triangular splines, Refinable spline, Spline basis functions,

Email addresses: `q.li@hull.ac.uk` (Qingde Li), `tian@ieee.org` (Jie Tian)

Multilevel splines, Partition of unity, Spline approximation

1. Introduction

Generalizing univariate splines to 2D are often achieved using the tensor-product of univariate splines. Building bivariate smooth non-tensor-product based multivariate spline basis functions is very difficult in general. This is true even for constructing C^1-smooth bivariate spline functions [1]. One difficulty in developing bivariate splines is in the infinite variety of the shape of the local support that a bivariate spline function may have. In one dimension, the only support of a univariate spline function can have is only an interval, for which the concepts of convex set and connected set mean the same thing. However, the two concepts differ completely when they are extended to 2D plane or higher dimensions. When the 2D counterpart of an interval is interpreted as a convex set, an interval can correspond to a convex polygon like triangle, rectangle and pentagon, or a convex region with a smooth boundary such as the area enclosed by an ellipse. When an interval is interpreted as a connected set, it can correspond to any simple polygon (a polygon with no edge crossing or no vertex shared by more than two edges), or any connected region with a smooth boundary.

One of the most popular bivariate spline techniques has been the triangular splines, where the spline basis functions are built from a set of triangles, due to the fact that triangle is the simplest type of polygon in 2D. In practical applications, triangular splines have been widely used as effective tools in approximation theory, computer-aided geometric design, image analysis, and numerical analysis[8]. So far, various efforts have been made to construct bivariate triangular splines. One of the well studied triangular spline techniques has been the Box splines. Box splines were introduced by de Boor and De Vore [4] and their detailed study can be found in book [5]. Bivariate box splines are piecewise polynomial functions built from a given set of directions. Despite their elegant theory and properties, the process of computing highly smooth box splines is in general very expensive [2]. The applications of box splines are also limited by its requirement that the underlying triangulation has to be regular due to the nature of its definition. A more general scheme to build a set of bivariate spline functions over any triangulation is the one developed by Dahmen et al. (sometimes referred to as DMS-splines)[3, 6]. DMS-splines are constructed based on a set of knot sequence inserted into the triangulated domain. However, as with box splines, the process of constructing DMS-splines with high degrees of smoothness, such as C^2-smoothness, is also very computationally intensive. In addition to their expensive computational cost, a more fundamental problem with DMS-splines is that there lacks a clear geometric meaning for the auxiliary knot sequences inserted into the triangulated domain during the process of constructing the DMS-splines. It is therefore not clear in what a way the inserted knots will affect the constructed spline functions, and subsequently the spline surfaces built upon these splines. Another way of constructing bivariate B-spline functions over a set of triangles is proposed in [13, 14] based on Delaunay configurations. This scheme allows to construct the spline functions in the form of polynomials in an elegant and simple way. It is similar to DMS-splines, but compared to DMS-splines, its knot selection procedure is more geometrically meaningful. A rational form of DMS-splines (RDMS) has also been introduced in a similar way as how NURBS are defined using B-splines[20]. As with DMS-splines, RDMS has similar issues with DMS-splines even though they are more effective in geometric modelling and shape deformation. Unlike the tensor-product based bivariate splines, triangular splines are even more expensive to refine locally in general. Developing locally refinable spline techniques has recently attracted increasing interest in computer aided geometric design, mainly due to their importance to isogeometric analysis. Some hierarchical spline techniques have been proposed to deal with the problem, but they are limited to either relatively regular space partitions, such as rectangular grid [22] and T-grid [16, 17, 15], or regular triangular grid[7].

A more flexible and convenient spline technique, called partial shape preserving splines (PSPS) [12], has recently been developed to built B-spline-like functions over arbitrary polygons. For a given set of polygons, whether they are convex or concave, or a mixture of polygons of different types, a bivariate function with any required degree of smoothness can be directly built based on each of the given polygons. If all these polygons form a partition of the domain, all the bivariate spline functions built with PSPS are non-negative piecewise polynomials and satisfy the law of partition of unity. PSPS are a kind of multilevel spline technique. With PSPS, a parameter can be associated to each control point to specify at what level one would want to approximate the fitted data or the control hull. In addition, PSPS can be locally refined. This feature of PSPS refers to an important property of PSPS spline functions, called additivity. With this property, any spline function built from a polygon can be expressed as the sum of a set of refined spline functions built from a set of smaller polygons that subdivide the initial polygon. This is an essential feature of multilevel splines, where the input control hulls are having a hierarchical structure, with different levels of control points corresponding to different levels of detail of required surface.

In this paper, we consider a special type of PSPS, the triangular partial shape preserving splines (Tri-PSPS), where the underlying polygons are only triangles. Unlike DMS-splines, Tri-PSPS have a clear geometric meaning. Tri-PSPS basis functions are obtained directly from a process of convolution over a triangulated domain, very similar to the construction of the univariate uniform B-spline basis functions. We have managed to provide a closed-form solution for the convolutions, so the basis functions of Tri-PSPS can be directly expressed as piecewise bivariate polynomials explicitly. Because of this, the implementation of Tri-PSPS is straightforward.

The meaning of the term "partial shape preserving" is that when the spline basis functions are used as blending functions, the geometric primitives involved in a blending operation can be preserved. Thus, the term"shape preserving" used in this paper is having a slightly different meaning from its conventional use.

Let the set of triangles $\mathbb{T} = \{\triangle_k\}_{k=0}^{K}$ be a triangulation of a 2D domain $\mathcal{D} \subseteq R^2$. That is, for any pair of triangles $\triangle_i, \triangle_j \in \mathbb{T}$, $i \neq j$, either $\triangle_i \bigcap \triangle_j = \emptyset$ or they only intersect at a vertex or at an edge. As having been discussed previously, there exits a variety of ways to construct triangular spline functions [21]. In this paper, we aim to construct from $\mathbb{T}$ a set of C^n-smooth piecewise polynomial bivariate functions $S^{(n)}(\mathbb{T})$:

$$S^{(n)}(\mathbb{T}) = \{B_\triangle(x,y) : B_\triangle(x,y) \in C^n(\mathcal{D}), \triangle \in \mathbb{T}\},$$

which satisfies the following properties:

(1) Locality. For each triangle $\triangle \in \mathbb{T}$, $B_\triangle(x,y)$ has a local support around triangle $\triangle$.
(2) Nonnegativity. For each triangle $\triangle$, $0 \leq B_\triangle(x,y) \leq 1$,
(3) $C^{(n)}$-continuity. Each $B_\triangle(x,y)$ can be built to have any required degree of smoothness.
(4) Closed-form expression. Each $B_\triangle(x,y)$ can be written out directly in closed form as piecewise bivariate polynomial functions.
(5) Partition of unity.

$$\sum_{\triangle \in \mathbb{T}} B_\triangle(x,y) = 1, \qquad (x,y) \in \mathcal{D}$$

(6) Refinability. if $\triangle$ is subdivided into a set of smaller triangles

$$\triangle = \bigcup_{k=0}^{K} \triangle_k,$$

then

$$B_{\triangle}(x,y) = \sum_{k=0}^{K} B_{\triangle_k}(x,y)$$

As with other types of triangular splines, Tri-PSPS can be applied directly to solve various real world problems, varying from developing robust localized numerical integration to approximation, shape design and finite element methods, isogeometric analysis. However, to make our attention more focused on the practical construction of Tri-PSPS functions, we will leave the full exploration to their relevant applications to interested readers.

2. Bivariate triangular spline functions as a process of convolution

Let $\mathbb{T}$ be a triangular partition of a 2D domain $\mathcal{D} \subseteq R^2$. We will detail in this section the process to directly construct a set of triangular spline functions which meet the requirements described above. For each $\triangle \in \mathbb{T}$, we first define $B^{(0)}_{\triangle}(x,y)$ as the characteristic function of the triangle $\triangle$:

$$B^{(0)}_{\triangle}(x,y) = \begin{cases} 0, & (x,y) \notin \triangle; \\ 1, & (x,y) \in \triangle \ . \end{cases}$$

Let $\square$ be the square $[-\delta,\delta] \times [-\delta,\delta]$ centred at the coordinate origin with $\delta > 0$. Starting from $B^{(0)}_{\triangle}(x,y)$, we define a sequence of bivariate functions in the following way

$$B^{(n)}_{\Delta,\delta}(x,y) \ = \ \frac{1}{4\delta^2} \int\int_{\mathcal{R}^2} B^{(n-1)}_{\Delta,\delta}(s,t)\chi_{\square}(s-x,t-y)dsdt, \qquad (n>0), \tag{1}$$

where $B^{(0)}_{\triangle,\delta}(x,y) = B^{(0)}_{\triangle}(x,y)$ and

$$\chi_{\square}(x,y) = \begin{cases} 1, & (x,y) \in [-\delta,\delta] \times [-\delta,\delta]; \\ 0, & elsewhere. \end{cases}$$

$B^{(n)}_{\Delta,\delta}(x,y)$ is called the order n triangular PSPS basis function.

As can be observed from the definition, the order n bivariate function $B^{(n)}_{\triangle}(x,y)$ built upon a triangle $\triangle \in \mathbb{T}$ will be non-negative piecewise polynomials with $C^{n-1}-$smoothness. From the definition of $B^{(0)}_{\triangle}(x,y)$, it is also easy to show that the set of functions $\{B^{(n)}_{\Delta,\delta}(x,y) : \triangle \in \mathbb{T}\}$ built from above process satisfies the law of partition of unity. In fact, for any $n > 0$, if all the order $n-1$ spline basis functions $\{B^{(n-1)}_{\Delta,\delta}(x,y) : \triangle \in \mathbb{T}\}$ have the property of partition of unity, then we have

$$\begin{aligned}
\sum_{\triangle\in\mathbb{T}} B^{(n)}_{\triangle,\delta}(x,y) &= \frac{1}{4\delta^2}\sum_{\triangle\in\mathbb{T}}\int\int_{\mathcal{R}^2} B^{(n-1)}_{\triangle,\delta}(s,t)\chi_{\Box}(s-x,t-y)dsdt \\
&= \frac{1}{4\delta^2}\int\int_{\mathcal{R}^2}\sum_{\triangle\in\mathbb{T}} B^{(n-1)}_{\triangle,\delta}(s,t)\chi_{\Box}(s-x,t-y)dsdt \\
&= \frac{1}{4\delta^2}\int\int_{\mathcal{R}^2}\chi_{\Box}(s-x,t-y)dsdt = 1. \qquad (2)
\end{aligned}$$

Since $\sum_{\triangle\in\mathbb{T}} B^{(0)}_{\triangle}(x,y) = 1$ when $(x,y) \notin \partial\triangle$, we see that the sequence of bivariate functions $B^{(k)}_{\triangle}(x,y)$ must have the property of partition of unity for $k = n$.

This idea of constructing the spline functions is not new and it will have little practical applications if the convolutions cannot be solved efficiently. Fortunately, a closed form solution can be obtained to the sequence of bivariate functions defined in (1)[11].

As has been shown in [11] that the sequence of convolutions defined in (1) can be be expressed as a linear combination of the convolutions defined on the three special types of half-open angles shown in Figure 1(note that all these open angles can be regarded as degenerated triangles with their third edges being at the infinity).

Note that the bivariate functions $B^{(n)}_{\Delta,\delta}(x,y)$ defined in (1) is only concerned with the relationship between triangle $\triangle$ and the square $[-\delta,\delta]^2$, they are invariant under the translation transformation. In fact, Let $T(\triangle)$ be the triangle obtained by translating the triangle $\triangle$ along vector $T = (t_x, t_y)$, then it can be seen that $B^{(n)}_{T(\Delta),\delta}(x,y) = B^{(n)}_{\Delta,\delta}(x-t_x, y-t_y)$. Because of this, we can assume that the apex vertices of the three half-open triangles shown in Figure 1 are all located at the coordinate origin. Let $\angle(m)$ be an open

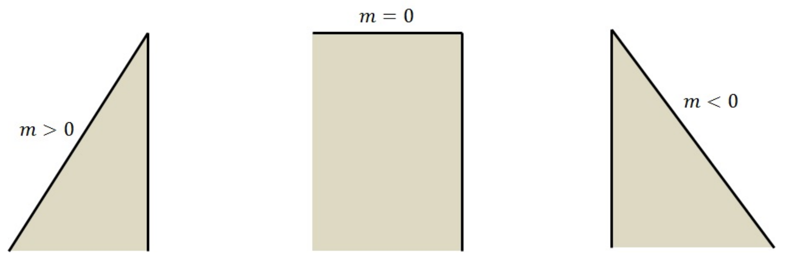


Figure 1: The convolutions defined in (1) can be expressed as a linear combination of the convolutions defined on above three types of open angles (triangles with one edge at the infinity), corresponding to the three cases regarding their top edge slopes m: $m > 0$, $m = 0$, $m < 0$.

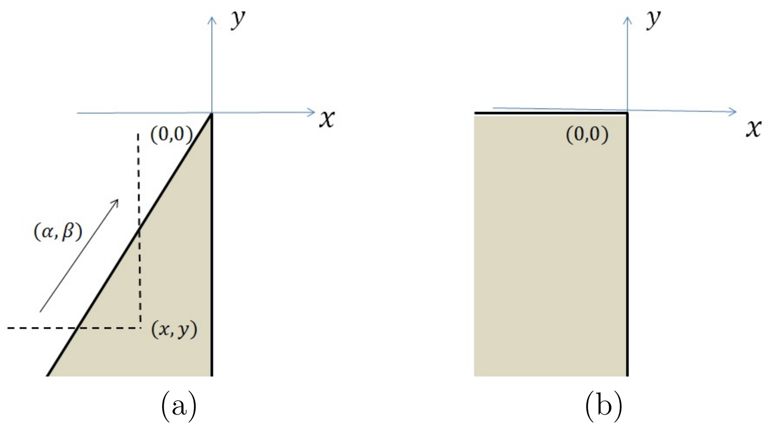


Figure 2: The calculation of $B^{(n)}_{\triangle,\delta}(x,y)$ can be reduced into the calculation of two special types of convolution, where the base triangles are half-open acute angle and the half-open right angle.

angle shown in Figure 1 with top edge slope m. Then, it can also be shown directly that

$$B^{(n)}_{\angle(m),\delta}(x,y) = B^{(n)}_{\angle(-m),\delta}(-x,y).$$

Therefore, to find the closed form solution for the convolutions defined in (1) with the base triangle being an open angle (a degenerated triangle with its third edge at the infinity) shown in Figure 1, we need only to consider the convolutions for the left two types of open angles shown in the figure.

In the following discussion, we will consider separately the two cases regarding whether an angle is acute angle (Figure 2(a)), where $\alpha > 0, \beta > 0$, or the right angle(2 (b)). The reason why the two cases are addressed separately is that the mathematical expressions of $B_{\triangle,\delta}^{(n)}(x,y)$ can be greatly simplified when the open angle $\triangle$ is a half-open right angle. In fact, for the half-open right angle, $B_{\triangle,\delta}^{(n)}(x,y)$ can be expressed as the tensor-product of univariate smooth step functions.

For simplicity, we denote by $B_{\angle(\alpha,\beta),\delta}^{(n)}(x,y)$ the convolution $B_{\triangle,\delta}^{(n)}(x,y)$ when $\triangle$ is an open acute angle shown in Figure 2 (a), and by $B_{\neg,\delta}^{(n)}(x,y)$ when $\triangle$ is the open right angle shown in Figure 2 (b).

2.1. Closed form Solution to $B_{\angle(\alpha,\beta),\delta}^{(n)}(x,y)$

Consider the following recursively defined sequence of convolutions with both $\alpha > 0$ and $\beta > 0$:

$$A_{\angle(\alpha,\beta)}^{(n)}(x,y) = \int\int_{\mathcal{R}^2} A_{\angle(\alpha,\beta)}^{(n-1)}(s,t)\chi_{\lrcorner}(s-x,t-y)dsdt, \qquad (n>0), \tag{3}$$

where $A_{\angle(\alpha,\beta)}^{(0)}(x,y)$ is defined as the characteristic function $\chi_{\angle(\alpha,\beta)}$ of the half-open acute angle $\angle(\alpha,\beta)$. That is,

$$A_{\angle(\alpha,\beta)}^{(0)}(x,y) = \chi_{\angle(\alpha,\beta)}(x,y) = \begin{cases} 1, & x \le 0, y < \frac{\beta}{\alpha}x; \\ 0, & elsewhere, \end{cases} \tag{4}$$

and $\chi_{\lrcorner}(x,y)$ is the characteristic function of half-open right angle $(-\infty,0] \times [0,\infty)$

$$\chi_{\lrcorner}(x,y) = \begin{cases} 1, & (x,y) \in (-\infty,0] \times [0,\infty); \\ 0, & elsewhere. \end{cases} \tag{5}$$

It has been shown in [11] that a closed form solution can be found for the sequence of convolutions defined in (3). More specifically, the bivariate function $A_{\angle(\alpha,\beta)}^{(n)}(x,y)$ can be explicitly expressed in closed-form in piecewise polynomials as follows:

$$A_{\angle(\alpha,\beta)}^{(n)}(x,y) = \begin{cases} \frac{1}{(2n)!\alpha^n\beta^n}(\beta x - \alpha y)^{2n}, & x \le 0, y < \frac{\beta}{\alpha}x; \\ \sum_{k=1}^{n} \frac{(-1)^{n+k}\alpha^k}{(n-k)!(n+k)!\beta^k} x^{n-k}y^{n+k}, & x > 0, y < 0; \\ 0, & \text{elsewhere.} \end{cases} \tag{6}$$

From its definition, it can be seen directly that $A^{(n)}_{\angle(\alpha,\beta)}(x,y)$ is a non-negative and C^{n-1} continuous piecewise polynomial function for any integer $n > 0$.

Now, let $\Box$ be the square $[-\delta,\delta]^2$ for any $\delta > 0$, and let

$$B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y) \;=\; \frac{1}{4\delta^2}\int\int_{\mathcal{R}^2} B^{(n-1)}_{\angle(\alpha,\beta),\delta}(s,t)\chi_{\Box}(s-x,t-y)dsdt, \quad (n>0), \tag{7}$$

where $B^{(0)}_{\angle(\alpha,\beta),\delta}(x,y) = A^{(0)}_{\angle(\alpha,\beta)}(x,y) = \chi_{\angle(\alpha,\beta)}(x,y)$.

Note that, for $(x,y) \notin \partial\Box$, $\chi_{\Box}(x,y)$ can be represented using $\chi_{\lrcorner}(x,y)$ defined in (5) in the following form:

$$\begin{aligned}\chi_{\Box}(x,y) \;&=\; \chi_{\lrcorner}(x-\delta,y+\delta) - \chi_{\lrcorner}(x+\delta,y+\delta)\\ &\quad -\chi_{\lrcorner}(x-\delta,y-\delta) + \chi_{\lrcorner}(x+\delta,y-\delta),\end{aligned}$$

The convolution $B^{(n)}_{\angle(\alpha,\beta)}(x,y)$ defined in (7) can thus be expressed as a linear combination of the convolutions of the form $A^{(n)}_{\angle(\alpha,\beta)}(x,y)$, for which a closed form solution has been found. In fact, for a given number $\delta > 0$, it has been shown that the convolution $B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y)$ can be directly expressed in terms of $A^{(n)}_{\angle(\alpha,\beta)}(x,y)$ in the following way:

$$B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y) = \frac{1}{(4\delta^2)^n}\sum_{i=0}^{n}\sum_{j=0}^{n}(-1)^{i+j}\binom{n}{i}\binom{n}{j}F_{i,j}(x,y), \tag{8}$$

where

$$F_{i,j}(x,y) = A^{(n)}_{\angle(\alpha,\beta)}(x+(n-2i)\delta,\ y-(n-2j)\delta)$$

Similar to $A^{(n)}_{\angle(\alpha,\beta)}(x,y)$, for $n > 0$, $B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y)$ is C^{n-1} continuous piecewise polynomial, and only takes value from interval $[0,1]$.

The bivariate function $B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y)$ is the building block of the triangular spline basis function $B^{(n)}_{\triangle}(x,y)$. As can be seen later, any triangular spline basis function $B^{(n)}_{\triangle}(x,y)$ can always be expressed as a linear combination of no more than six bivariate functions of the form $B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y)$.

Since C^1 and C^2 smooth triangular splines are most frequently used in practice, we describe here more specifically how to construct $B^{(1)}_{\angle(\alpha,\beta),\delta}(x,y)$, $B^{(2)}_{\angle(\alpha,\beta),\delta}(x,y)$, and $B^{(3)}_{\angle(\alpha,\beta),\delta}(x,y)$ from $A^{(1)}_{\angle(\alpha,\beta),\delta}(x,y)$, $A^{(2)}_{\angle(\alpha,\beta),\delta}(x,y)$, and $A^{(3)}_{\angle(\alpha,\beta),\delta}(x,y)$ respectively.

Case: $n = 1$

From (6), we have

$$A^{(1)}_{\angle(\alpha,\beta)}(x,y) = \begin{cases} \frac{1}{2\alpha\beta}(\beta x - \alpha y)^2, & x \le 0, y < \frac{\beta}{\alpha}x; \\ \frac{\alpha}{2\beta}y^2, & x > 0, y < 0; \\ 0, & \text{elsewhere.} \end{cases} \tag{9}$$

According to (8), we can see that the C^0-smooth $B^{(1)}_{\angle(\alpha,\beta),\delta}(x,y)$ is a linear combination of four functions obtained by translating $A^{(1)}_{\angle(\alpha,\beta)}(x,y)$ to the following four locations: $(-\delta,-\delta)$, $(\delta,-\delta)$, $(-\delta,\delta)$, (δ,δ). More specifically,

$$\begin{aligned} B^{(1)}_{\angle(\alpha,\beta),\delta}(x,y) &= \frac{1}{4\delta^2}[A^{(1)}_{\angle(\alpha,\beta)}(x-\delta,y+\delta) - A^{(1)}_{\angle(\alpha,\beta)}(x-\delta,y-\delta) \\ &\quad -A^{(1)}_{\angle(\alpha,\beta)}(x+\delta,y+\delta) + A^{(1)}_{\angle(\alpha,\beta)}(x+\delta,y-\delta)]. \end{aligned} \tag{10}$$

The relationship between $A^{(1)}_{\angle(\alpha,\beta)}(x,y)$ and $B^{(1)}_{\angle(\alpha,\beta),\delta}(x,y)$ can be illustrated with the mask shown in Figure 3, where the numbers in the circles are the coefficients used to combine the function $A^{(1)}_{\angle(\alpha,\beta)}(x,y)$ evaluated at four corner locations. Using the notations shown in the mask, equation (10) can be rewritten in a more compact form as followings:

$$\begin{aligned} B^{(1)}_{\angle(\alpha,\beta),\delta}(x,y) &= \frac{1}{4\delta^2}[A^{(1)}_{\angle(\alpha,\beta)}(\mathbf{P}_{00}) - A^{(1)}_{\angle(\alpha,\beta)}(\mathbf{P}_{10}) \\ &\quad -A^{(1)}_{\angle(\alpha,\beta)}(\mathbf{P}_{01}) + A^{(1)}_{\angle(\alpha,\beta)}(\mathbf{P}_{11})]. \end{aligned}$$

Case: $n = 2$

To directly write out C^1-continuous $B^{(2)}_{\angle(\alpha,\beta),\delta}(x,y)$, we use $A^{(2)}_{\angle(\alpha,\beta)}(x,y)$. From (6), we have

$$A^{(2)}_{\angle(\alpha,\beta)}(x,y) = \begin{cases} \frac{1}{4!(\alpha\beta)^2}(\beta x - \alpha y)^4, & x \le 0, y < \frac{\beta}{\alpha}x. \\ \frac{1}{4!}(\frac{\alpha}{\beta})^2 y^4 - \frac{1}{3!}(\frac{\alpha}{\beta})xy^3, & x > 0, y < 0. \\ 0, & \text{elsewhere.} \end{cases} \tag{11}$$

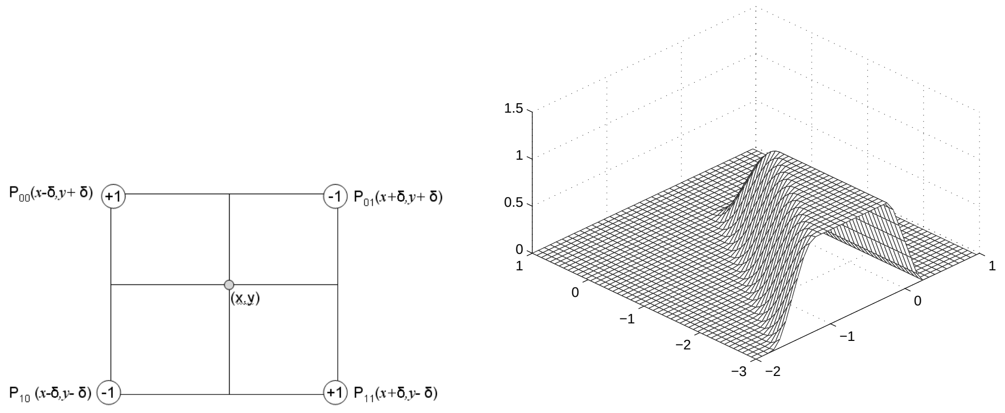


Figure 3: Left: The mask to illustrate how to directly write out $B^{(1)}_{\angle(\alpha,\beta),\delta}(x, y)$ in terms of functions $A^{(1)}_{\angle(\alpha,\beta)}(x, y)$. The numbers in the circles are the coefficients used to combine the translated functions of $A^{(1)}_{\angle(\alpha,\beta)}(x, y)$ corresponding to the four locations. Right: C^0-smooth $B^{(1)}_{\angle(\alpha,\beta),\delta}(x, y)$.

From (8), we can directly write out $B^{(2)}_{\angle(\alpha,\beta),\delta}(x, y)$ in terms of $A^{(2)}_{\angle(\alpha,\beta)}(x, y)$ in the following way based on the mask shown in Figure 4.:

$$\begin{array}{c} B^{(2)}_{\angle(\alpha,\beta),\delta}(x, y) = \frac{1}{(4\delta^2)^2}[\\ A^{(2)}_{\angle(\alpha,\beta)}(\mathbf{P}_{00}) - 2A^{(2)}_{\angle(\alpha,\beta)}(\mathbf{P}_{10}) + A^{(2)}_{\angle(\alpha,\beta)}(\mathbf{P}_{20}) \\ -2A^{(2)}_{\angle(\alpha,\beta)}(\mathbf{P}_{01}) + 4A^{(2)}_{\angle(\alpha,\beta)}(\mathbf{P}_{11}) - 2A^{(2)}_{\angle(\alpha,\beta)}(\mathbf{P}_{21}) \\ +A^{(2)}_{\angle(\alpha,\beta)}(\mathbf{P}_{02}) - 2A^{(2)}_{\angle(\alpha,\beta)}(\mathbf{P}_{12}) + A^{(2)}_{\angle(\alpha,\beta)}(\mathbf{P}_{22}) \ ]. \end{array}$$

Case 3: $n = 3$

Similarly, the C^2-continuous $B^{(3)}_{\angle(\alpha,\beta),\delta}(x, y)$ can be expressed in terms of $A^{(3)}_{\angle(\alpha,\beta)}$ according to the mask shown in Figure 5. That is,

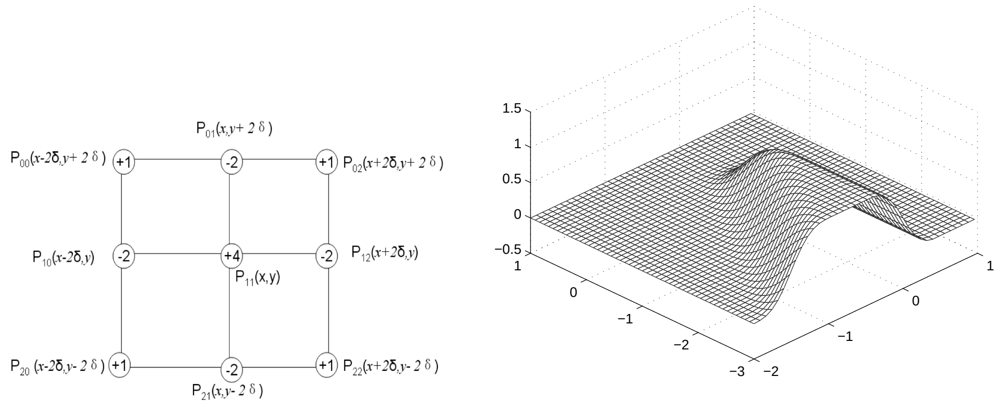


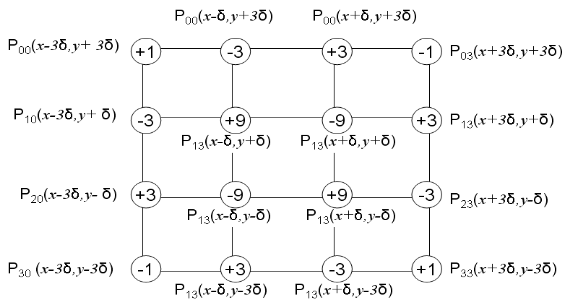
Figure 4: Left: The mask to illustrate how to directly write out $B^{(2)}_{\angle(\alpha,\beta),\delta}(x,y)$ in terms of $A^{(2)}_{\angle(\alpha,\beta)}(x,y)$. Right: C^1-smooth $B^{(2)}_{\angle(\alpha,\beta)}(x,y)$.

$$
\begin{gathered}
B^{(3)}_{\angle(\alpha,\beta),\delta}(x,y) = \tfrac{1}{(4\delta^2)^3}[\\
A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{00}) - 3A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{10}) + 3A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{20}) - A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{30}) \\
-3A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{01}) + 9A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{11}) - 9A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{21}) + 3A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{31}) \\
+3A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{02}) - 9A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{12}) + 9A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{22}) - 3A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{32}) \\
-A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{03}) + 3A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{13}) - 3A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{23}) + A^{(3)}_{\angle(\alpha,\beta)}(\mathbf{P}_{33})],
\end{gathered}
$$

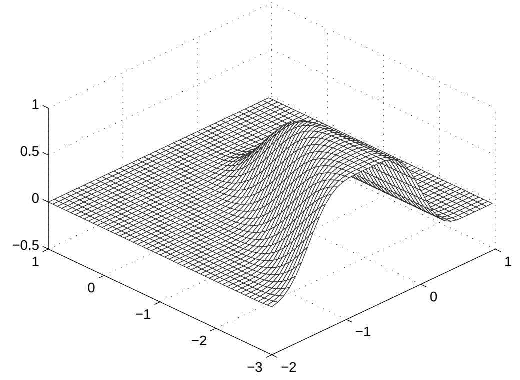
where

$$
A^{(3)}_{\angle(\alpha,\beta)}(x,y) = \begin{cases} \frac{1}{6!(\alpha\beta)^3}(\beta x - \alpha y)^6, & x \le 0, y < \frac{\beta}{\alpha}x; \\ \\ \frac{1}{6!}(\frac{\alpha}{\beta})^3 y^6 - \frac{1}{5!}(\frac{\alpha}{\beta})^2 x y^5 + \frac{1}{2!4!}(\frac{\alpha}{\beta}) x^2 y^4, & x > 0, y < 0; \\ \\ 0, & \text{elsewhere}; \end{cases} \tag{12}
$$

In practical applications, we often need to find the partial derivatives of triangular spline basis function $B^{(n)}_{\triangle}(x,y)$. Because the explicit form of the function $B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y)$ is known, the calculation of the partial differentiation to triangular spline function $B^{(n)}_{\triangle}(x,y)$ up to any required order is straightforward. From 8, we can see



(a)



(b)

Figure 5: (a). The mask to illustrate how to build $B^{(3)}_{\angle(\alpha,\beta),\delta}(x,y)$ in terms of $A^{(3)}_{\angle(\alpha,\beta)}(x,y)$. (b). C^2-smooth $B^{(3)}_{\angle(\alpha,\beta)}(x,y)$.

$$\frac{\partial B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y)}{\partial x} = \frac{1}{(4\delta^2)^n}\sum_{i=0}^{n}\sum_{j=0}^{n}(-1)^{i+j}\binom{n}{i}\binom{n}{j}\frac{\partial F_{i,j}(x,y)}{\partial x}. \quad (13)$$

$$\frac{\partial B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y)}{\partial y} = \frac{1}{(4\delta^2)^n}\sum_{i=0}^{n}\sum_{j=0}^{n}(-1)^{i+j}\binom{n}{i}\binom{n}{j}\frac{\partial F_{i,j}(x,y)}{\partial y}, \quad (14)$$

where

$$\frac{\partial F_{i,j}(x,y)}{\partial x} = \frac{\partial A^{(n)}_{\angle(\alpha,\beta)}(x+(n-2i)\delta,\ y-(n-2j)\delta)}{\partial x}$$

and

$$\frac{\partial F_{i,j}(x,y)}{\partial y} = \frac{\partial A^{(n)}_{\angle(\alpha,\beta)}(x+(n-2i)\delta,\ y-(n-2j)\delta)}{\partial y}$$

can be found directly from (6).

For instance, we can directly find the partial derivatives with respect to x for $A^{(3)}_{\angle(\alpha,\beta)}(x,y)$ up to the second order. More specifically, we have,

$$\frac{\partial A^{(n)}_{\angle(\alpha,\beta)}(x,y)}{\partial x} = \begin{cases} \frac{1}{5!\alpha^3\beta^2}(\beta x-\alpha y)^5, & x \le 0, y < \frac{\beta}{\alpha}x; \\ -\frac{1}{5!}(\frac{\alpha}{\beta})^2 y^5 + \frac{1}{4!}(\frac{\alpha}{\beta})xy^4, & x > 0, y < 0; \\ 0, & \text{elsewhere}; \end{cases} \tag{15}$$

(16)

$$\frac{\partial^2 A^{(n)}_{\angle(\alpha,\beta)}(x,y)}{\partial x^2} = \begin{cases} \frac{1}{4!\alpha^3\beta}(\beta x-\alpha y)^4, & x \le 0, y < \frac{\beta}{\alpha}x; \\ \frac{1}{4!}(\frac{\alpha}{\beta})y^4, & x > 0, y < 0; \\ 0, & \text{elsewhere}; \end{cases} \tag{17}$$

2.2. Closed form solution of $B^{(n)}_{\neg,\delta}(x,y)$ as a tensor product of smooth step functions

Similar to the process of developing a closed-form solution for $B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y)$, a closed-form solution can be obtained for $B^{(n)}_{\neg,\delta}(x,y)$, a special case of $B^{(n)}_{\triangle,\delta}(x,y)$, where $\triangle$ is the half-open rectangular area $\neg$ shown in Figure 2 (b). In fact, $B^{(n)}_{\neg,\delta}(x,y)$ can be expressed in a similar form to $B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y)$ by replacing $A^{(n)}_{\angle(\alpha,\beta)}$ in (8) with the following bivariate function

$$A^{(n)}_{\angle(1,0)} = \begin{cases} \frac{(-1)^n}{(n!)^2}x^n y^n, & x < 0, y < 0; \\ 0, & \text{elsewhere}. \end{cases} \tag{18}$$

However, $B^{(n)}_{\neg,\delta}(x,y)$ can be expressed in a more compact form as the tensor-product of univariate smooth step functions[12]. As the calculation of smooth step functions is much less computationally intensive, in terms of simplicity and computational cost, it is more preferable to express $B^{(n)}_{\neg,\delta}(x,y)$ as the tensor-product of smooth step functions. Below, we give an introduction to smooth step functions, especially about their constructions and properties.

In practice, different types of smooth step functions [10] can be constructed using several different methods. It is found that the piecewise polynomial smooth step functions introduced in [9] is closely related to $B^{(n)}_{\neg,\delta}(x,y)$. This type of smooth step functions can be defined either as an iterative process or directly in explicit form using the standard Heaviside unit step function. The iterative definition can be expressed in the following way:

$$\begin{aligned} H_0(x) &= \begin{cases} 0, & x < 0; \\ \frac{1}{2}, & x = 0; \\ 1, & x > 0. \end{cases} \\ H_n(x) &= \frac{1}{2}\Big(\ (1+\frac{x}{n})H_{n-1}(x+1) \\ &\qquad +(1-\frac{x}{n})H_{n-1}(x-1) \ \Big), \\ & n = 1, 2, 3, \cdots. \end{aligned} \tag{19}$$

The recursively defined smooth step function $H_n(x)$ shown in (19) can also be written out directly in closed-form in terms of the Heaviside unit step function in the following way:

$$H_n(x) = \frac{1}{n!2^n}\sum_{k=0}^{n}(-1)^k\binom{n}{k}(x+(n-2k))^n H_0(x+(n-2k)) \tag{20}$$

$H_n(x)$ can be considered as a generalization of the Heaviside unit step function and can be referred to as the order n smooth unit step function, or simply smooth step functions. Smooth step function $H_n(x)$ has the following properties:

Proposition 2.1. *For each function $H_n(x)$,*

(1) $H_n(x)$ is C^{n-1}-continuous for $n > 0$;
(2) $H_n(x)$ is a piecewise-polynomial function of degree n;
(3) $H_n(x)$ is monotonically increasing and takes value 1 when $x \geq n$, and 0 when $x \leq -n$;

(4) $H_n(x)+H_n(-x) = 1$, $H_n(0) = \frac{1}{2}$;

(5) $H_n(x) \geq H_{n-1}(x)$ *when* $x < 0$ *and* $H_n(x) \leq H_{n-1}(x)$ *when* $x \geq 0$, $n = 1, 2, \cdots$;

(6) $x(2H_n(x) - 1) \leq x(2H_{n-1}(x) - 1)$, $n = 1, 2, \cdots$.

Following the definition of $H_n(t)$ given in equation (19), the functions $H_1(x)$, $H_2(x)$, and $H_3(x)$ can be written out explicitly. Note that $H_n(x) = 1 - H_n(-x)$, we need only to write out these functions for $x \leq 0$.

$$H_1(x) = \begin{cases} 0, & x < -1; \\ \frac{1+x}{2}, & -1 \leq x \leq 0. \end{cases} \tag{21}$$

$$H_2(x) = \begin{cases} 0, & x < -2; \\ \frac{1}{2}(1 + \frac{x}{2})^2, & -2 \leq x < 0. \end{cases} \tag{22}$$

$$H_3(x) = \begin{cases} 0, & x < -3; \\ \frac{1}{48}(3 + x)^3, & -3 \leq x < -1; \\ \frac{1}{24}(12 + 9x - x^3), & -1 \leq x < 0. \end{cases} \tag{23}$$

From (20), the generalized smooth step functions of degree 1, 2 and 3 can also be expressed in the following form:

$$H_1(x) = \frac{1}{2}((x + 1)H_0(x + 1) - (x - 1)H_0(x - 1)) \tag{24}$$

$$H_2(x) = \frac{1}{8}((x + 2)^2 H_0(x + 2) - 2x^2 H_0(x) + (x - 2)^2 H_0(x - 2)) \tag{25}$$

$$\begin{aligned} H_3(x) = {} & \frac{1}{48}((x + 3)^3 H_0(x + 3) - 3(x + 1)^3 H_0(x + 1) \\ & +3(x - 1)^3 H_0(x - 1) - (x - 3)^3 H_0(x - 3)) \end{aligned} \tag{26}$$

Figure 6 presents a plot of the smooth step functions of degree 1 to degree 4.

As can be observed directly, the degree n smooth step function $H_n(x)$ is strictly monotone increasing over the interval $[-n, n]$. We call this interval the rising range of a smooth step function. The smooth step function with any specified rising range can be defined easily by introducing a nonnegative number $\delta > 0$ as follows:

$$H_{n,\delta}(x) = H_n(nx/\delta). \tag{27}$$

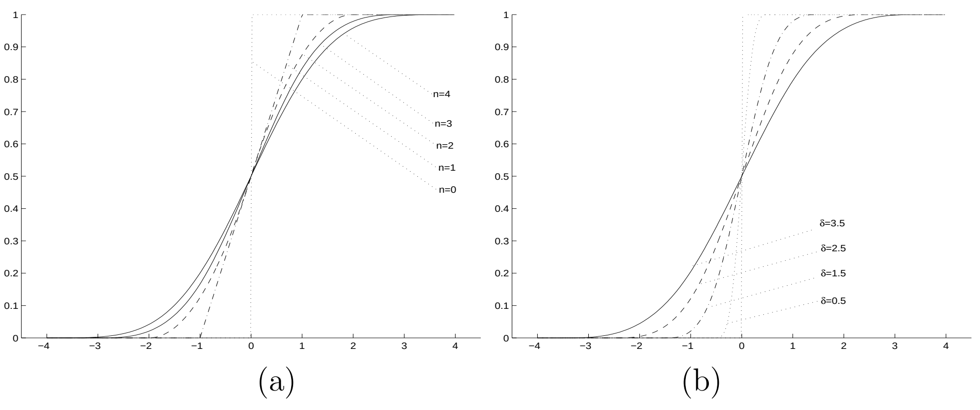


Figure 6: (a). Piecewise polynomial smooth unit step functions of degree 1 to degree 4. (b). Piecewise polynomial smooth step function $H_{3,\delta}(x)$ with different rising ranges specified using δ.

Obviously, $H_{n,\delta}(x) = 1$ when $x \geq \delta$, and $H_{n,\delta}(x) = 0$ when $x < -\delta$.

Figure 6(b) shows some C^2-continuous cubic smooth step functions $H_{3,\delta}(x)$ with different values of rising range parameter δ.

The derivatives of smooth step functions can be found easily for $n > 1$. From (20), it can be seen that the derivatives of $H_n(x)$ can be directly expressed explicitly using the Heaviside unit step function. When $n > 1$,

$$H_n'(x) = \frac{1}{(n-1)!2^n}\sum_{k=0}^{n}(-1)^k\binom{n}{k}(x+(n-2k))^{n-1}H_0(x+(n-2k)) \tag{28}$$

In general, for $i < n$, the i^{th} order derivative of degree n smooth step function $H_n(x)$ can be expressed explicitly as

$$H_n^{(i)}(x) = \frac{1}{(n-i)!2^n}\sum_{k=0}^{n}(-1)^k\binom{n}{k}(x+(n-2k))^{n-i}H_0(x+(n-2k)) \tag{29}$$

With (29), the relevant derivatives for $H_2(x)$ and $H_3(x)$ can immediately be obtained as

$$H_2'(x) = \frac{1}{4}((x+2)H_0(x+2) - 2xH_0(x) + (x-2)H_0(x-2)) \tag{30}$$

$$H_3'(x) = \frac{1}{16}((x+3)^2H_0(x+3) - 3(x+1)^2H_0(x+1)$$

$$+3(x-1)^2H_0(x-1)-(x-3)^2H_0(x-3)) \quad (31)$$

$$H_3''(x) = \frac{1}{8}((x+3)H_0(x+3)-3(x+1)H_0(x+1)$$
$$+3(x-1)H_0(x-1)-(x-3)H_0(x-3)) \quad (32)$$

Let $\neg$ be the half-open right angle shown in Figure 2 (b). Consider the bivariate functions defined in the following way:

$$B^{(0)}_{\neg,\delta}(x,y) = \begin{cases} 1, & x \le 0, y \le 0; \\ 0, & elsewhere. \end{cases}$$
$$B^{(n)}_{\neg,\delta}(x,y) = \int\int_{\mathcal{R}^2} B^{(n-1)}_{\neg,\delta}(s,t)\chi_\square(s-x,t-y)dsdt,$$
$$(n>0), \quad (33)$$

where $\chi_\square(x,y)$ is the characteristic function of the square $[-\delta,\delta]\times[-\delta,\delta]$.

It can be shown that $B^{(n)}_{\neg,\delta}(x,y)$ can be expressed as the tensor-product of smooth step functions in the following way:

$$B^{(0)}_{\neg,\delta}(x,y) = H_n(-x/\delta)\times H_n(-y/\delta).$$

3. Triangular spline functions

Let $\angle^*(\alpha,\beta)$ be an open angles shown in Figure 2 with their non-vertical edge parallel to a vector $\mathbf{e}(\alpha,\beta)$, $\alpha^2+\beta^2>0$, $\alpha\ge 0$. Note that the calculation of the convolution corresponding to an open angle with a negative slope can be turned into a problem of calculating the convolution defined on an open angle with a positive slope. More specifically, suppose the slope edge of the open angle shown in Figure 7 parallel to a vector $\mathbf{e}(\alpha,-\beta)$, $\alpha>0$, $\beta>0$. Then we have

$$B^{(n)}_{\angle(\alpha,-\beta),\delta}(x,y) = B^{(n)}_{\angle(\alpha,\beta),\delta}((-x,y). \quad (34)$$

From the results shown in previous section, the solution to the sequence of convolutions defined in (1) built upon the open angle $\angle^*(\alpha,\beta)$ can be expressed in closed-form explicitly as

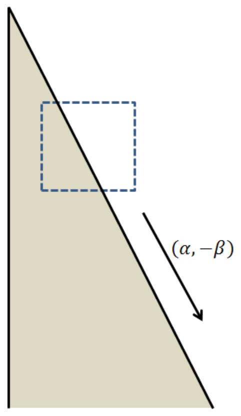


Figure 7: Triangle edge with negative slope

$$B^{(n)}_{\angle^*(\alpha,\beta),\delta}(x,y) = \begin{cases} 0, & \alpha = 0, \beta > 0; \\ H_n(\frac{1}{\delta}(-x))H_n(\frac{1}{\delta}(-y)), & \\ & \alpha > 0, \beta = 0; \\ B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y), & \\ & \alpha > 0, \beta > 0 \\ B^{(n)}_{\angle(\alpha,-\beta),\delta}(-x,y), & \\ & \alpha > 0, \beta < 0. \end{cases} \tag{35}$$

For any given triangle $\triangle$, we now show that the sequence of bivariate functions $B^{(n)}_{\triangle}(x,y)$ built upon triangle $\triangle$ according to (1) can be expressed as a linear combination of a few explicitly represented piecewise bivariate polynomial functions presented in (35).

Let E be an edge of a triangle $\triangle$ defined by two vertices $\mathbf{v}_0$, $\mathbf{v}_1$. Consider the infinite open trapezoid $\sqcap(\mathbf{v}_0,\ \mathbf{v}_1)$ shown in Figure 8, enclosed by line segment E and two downward vertical rays starting from vertices $\mathbf{v}_0$, $\mathbf{v}_1$ respectively. Just as an open angle can be considered as a big triangle with its third edge at the infinity, $\sqcap(\mathbf{v}_0, \mathbf{v}_1)$ can also be viewed as a 'big' triangle with its third vertex at the infinity. In (1), if $\sqcap(\mathbf{v}_0,\ \mathbf{v}_1)$ is used as the base triangle in the convolution, it can be shown directly that $B^{(n)}_{\sqcap(\mathbf{v}_0,\mathbf{v}_1),\delta}(x,y)$

can be expressed as the difference between the two functions obtained by translating $B^{(n)}_{\angle(\alpha,\beta),\delta}(x,y)$ to vertices $\mathbf{v}_0$ and $\mathbf{v}_1$ respectively. Let $\mathbf{v}_1 - \mathbf{v}_0 = (\alpha, \beta)$ be the vector representing the orientation of the top edge of the open area $\sqcap(\mathbf{v}_0, \mathbf{v}_1)$.

$$B^{(n)}_{\sqcap(\mathbf{v}_0,\mathbf{v}_1),\delta}(x,y) = \begin{cases} 0; & \alpha = 0; \\ B^{(n)}_{\angle(\alpha,\beta),\delta}((x,y)-\mathbf{v}_1) - B^{(n)}_{\angle(\alpha,\beta),\delta}((x,y)-\mathbf{v}_0), & \\ \qquad \alpha > 0, \beta \geq 0; & \\ B^{(n)}_{\angle(\alpha,-\beta),\delta}((-x,y)-\mathbf{v}_0) - B^{(n)}_{\angle(\alpha,-\beta),\delta}((-x,y)-\mathbf{v}_1), & \\ \qquad \alpha > 0, \beta < 0; & \\ B^{(n)}_{\angle(-\alpha,\beta),\delta}((-x,y)-\mathbf{v}_1) - B^{(n)}_{\angle(-\alpha,\beta),\delta}((-x,y)-\mathbf{v}_0), & \\ \qquad \alpha < 0, \beta \geq 0; & \\ B^{(n)}_{\angle(-\alpha,-\beta),\delta}((x,y)-\mathbf{v}_1) - B^{(n)}_{\angle(-\alpha,-\beta),\delta}((x,y)-\mathbf{v}_0), & \\ \qquad \alpha < 0, \beta < 0, & \end{cases} \tag{36}$$

For any given triangle, three bivariate functions can be obtained in this way based on the three edges of the given triangle. According to the way we construct the spline functions, it can be seen directly that $B^{(n)}_{\triangle,\delta}(x,y)$ is simply a signed sum of the three bivariate functions built upon the three edges of a triangle, where a positive sign is associated with a function built upon an upper edge, and a negative sign is associated with a lower edge. More specifically, let the vertices $\mathbf{v}_0(x_0,y_0)$, $\mathbf{v}_1(x_1,y_1)$, $\mathbf{v}_2(x_2,y_2)$ of a triangle $\triangle$ be specified in counterclockwise order, and let $\alpha_0 = x_0 - x_1$, $\alpha_1 = x_1 - x_2$, and $\alpha_2 = x_2 - x_0$. Then these αs are the y-components of the triangle's edge normal pointing outwards. Thus,

$$\begin{aligned} B^{(n)}_{\triangle,\delta}(x,y) &= sign(\alpha_0) B^{(n)}_{\sqcap(\mathbf{v}_0,\mathbf{v}_1)}(x,y) + sign(\alpha_1) B^{(n)}_{\sqcap(\mathbf{v}_1,\mathbf{v}_2)}(x,y) \\ &\quad + sign(\alpha_2) B^{(n)}_{\sqcap(\mathbf{v}_2,\mathbf{v}_0)}(x,y). \end{aligned} \tag{37}$$

Since $\alpha = 0$, $B^{(n)}_{\sqcap(\alpha,\beta),\delta}((x,y) = 0$ for a vertical edge. Thus, we can ignore the vertical edge of a triangle when constructing $B^{(n)}_{\triangle,\delta}(x,y)$. This means,

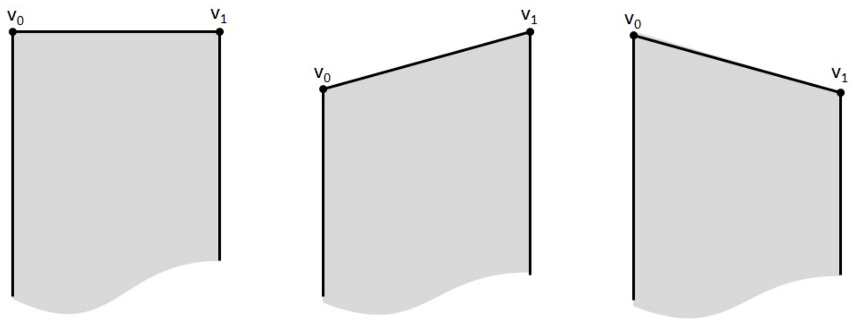


Figure 8: Infinite vertical open trapezoids corresponding to a triangle edge.

when a triangle has a vertical edge, $B^{(n)}_{\triangle,\delta}(x,y)$ is simply the difference of two functions $B^{(n)}_{\sqcap(\mathbf{v}_1,\mathbf{v}_2)}(x,y)$ corresponding to the top edge and the bottom edge of the triangle respectively.

With the properties of integration it can be shown that $B^{(n)}_{\Delta,\delta}(x,y)$ has the following properties:

1. $0 \leq B^{(n)}_{\Delta,\delta}(x,y) \leq 1$.
2. $B^{(n)}_{\triangle,\delta}(x,y)$ has a C^{n-1} continuity.
3. $B^{(n)}_{\triangle,\delta}(x,y)$ is piecewise polynomial.
4. $B^{(n)}_{\triangle,\delta}(x,y)$ has a finite support for any finite triangle $\triangle$.
5. $B^{(n)}_{\triangle,\delta}(x,y)$ is additive. That is, if two triangles $\triangle_1$ and $\triangle_2$ do not intersect or they only intersect at their edges, then

$$B^{(n)}_{\triangle_1\cup\triangle_2,\delta}(x,y) = B^{(n)}_{\triangle_1,\delta}(x,y) + B^{(n)}_{\triangle_2,\delta}(x,y).$$

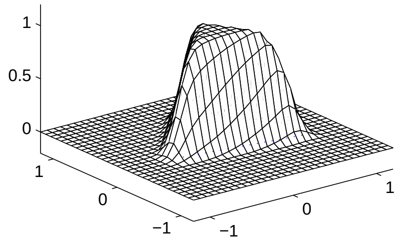
6. Partition of unity. For a 2D domain $\mathcal{D}$, if

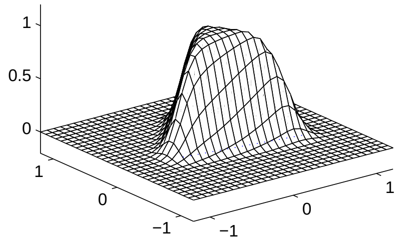
$$\bigcup_k \triangle_k = \mathcal{D}, \quad area(\Delta_i \bigcap_{i\neq j} \Delta_j) = 0,$$

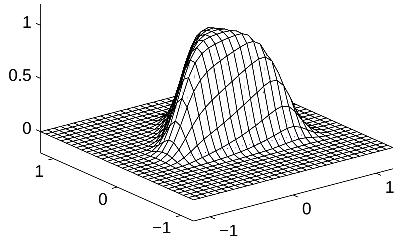
then

$$\sum_k B^{(n)}_{\triangle_k,\delta}(x,y) = 1, \quad (x,y) \in \mathcal{D}.$$

As have been discussed in [11], the construction of a spline function based on a given polygon is simply a process of constructing a set of bivariate functions associated to the vertices of the polygon. Since the explicit representation is known for each of these vertex based functions, the calculation of the


(a)


(b)


(c)

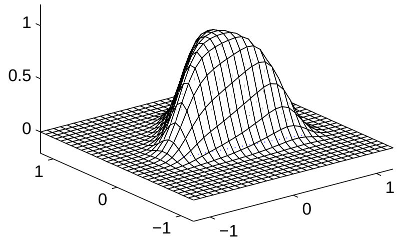

(d)

Figure 9: Triangular spline functions built with different degrees of smoothness, with $\delta = 0.2$: (a). C^1-continuous $B^{(2)}_{\triangle,0.2}(x,y)$; (b). C^2-continuous $B^{(3)}_{\triangle,0.2}(x,y)$; (c). C^3-continuous $B^{(4)}_{\triangle,0.2}(x,y)$; (d). C^4-continuous $B^{(5)}_{\triangle,0.2}(x,y)$.

spline basis function is straightforward. Note also that the construction of different order spline basis functions only differs in the computational cost required for computing the function $A^{(n)}_{\angle(\alpha,\beta)}(x,y)$ defined in (6), there is no extra effort required in terms of the implementation of constructing a higher order smooth spline basis function.

Figures 9 to 10 show the triangular spline functions constructed with different degrees of smoothness and different values of the smooth range parameter δ.

As Tri-PSPS functions can be directly written out as piecewise polynomials in closed form, it is straightforward to construct a set of spline basis functions from any given triangular partition of a 2D domain. Figure 11 shows three sets of Tri-PSPS constructed with different values of smoothing parameter δ.

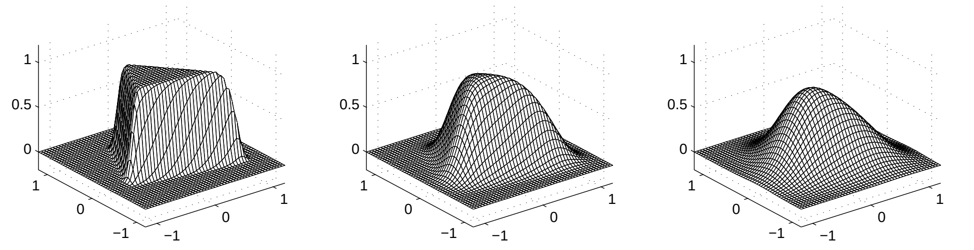


Figure 10: C^2-smoothness triangular spline functions built with different values of smoothing range parameter δ: (a). $\delta = 0.05$; (b). $\delta = 0.15$; (c). $\delta = 0.25$.

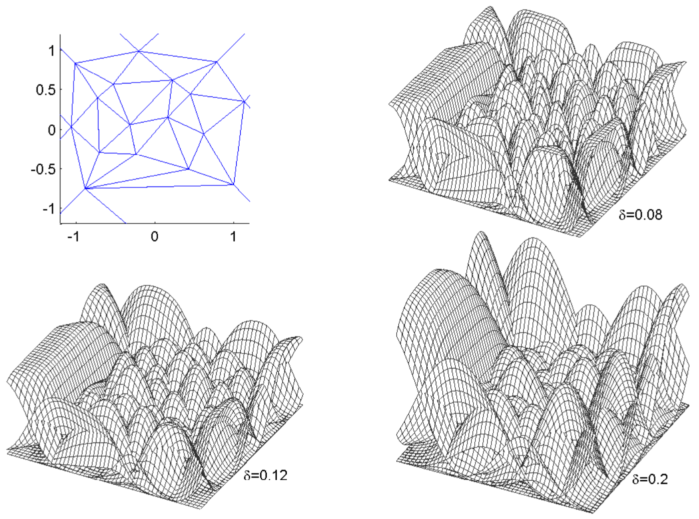


Figure 11: A set of triangular spline functions built from a set of triangles. Top left: Triangulated domains; Top tight: C^2-smooth triangular spline functions constructed with $\delta = 0.08$; Low left: C^2-smooth triangular spline functions constructed with $\delta = 0.12$; Low right: C^2-smooth triangular spline functions constructed with $\delta = 0.2$;

Note that if the orientation of an edge of a triangle $\triangle$ is described by a unit vector $\mathbf{v}(\alpha, \beta)$, $B_{\triangle}^{(n)}(x, y) = 0$ when $\alpha y - \beta x > n(\alpha + \beta)\delta$. Because of this, the calculation of a triangular spline function $B_{\triangle(\mathbf{v})}^{(n)}(x, y)$ needs only to be performed locally around its base triangle.

4. Surface design using Tri-PSPS

Let $\triangle_0, \triangle_1, \triangle_2, \cdots, \triangle_K$ be a triangulation of a 2D domain $\mathcal{D}$. Let $B_{\triangle_k}^{(n)}(x, y)$, $k = 0, 1, 2, \cdots, K$, be the sequence of Tri-PSPS functions built upon these triangles. Tri-PSPS surfaces take the following form

$$S(x, y) = \sum_{i=0}^{K} P_i(x, y) B_{\triangle_i}^{(n)}(x, y),$$

where $P_i(x, y)$ is the i^{th} control geometric primitive associated to triangle $\triangle_i$, $i = 0, 1, 2, \cdots, K$. Traditionally, the control geometric primitives used are mainly a set of control points. With the shape-preserving feature of Tri-PSPS, we can easily turn the design of a complex surface as a task of designing a set of simple surfaces. In Figure 12, the left figure shows a control point-based Tri-PSPS surface and the right one is a Tri-PSPS surface obtained by blending a set of planar control primitives.

As have been pointed out, Tri-PSPS are a kind of multilevel spline technique. The levels of detail can be easily incorporated into the parameter δ. In Figure 13, a noisy point set is used to show how levels of detail in surface design can be easily implemented using different δ values.

Surface design with levels of detail can also be achieved by using the additivity property of Tri-PSPS. The idea is to first organise the control data points or primitives in a hierarchical structure, with different control points at different levels corresponding to different levels of detail of a required surface. The additivity of Tri-PSPS makes Tri-PSPS an ideal spline technique for editing surface's level of detail. This is because, with the feature of additivity, the spline function built from one big triangle can be replaced with a set of refined Tri-PSPS functions built on a set of smaller subdivision triangles of the original triangle (see Figure 14). Obviously, high levels of detail of a surface can be achieved by partitioning the domain using much smaller triangles, as it is shown in Figure 15.

Tri-PSPS can be used in various ways for solving real world problems. To demonstrate the potential applications of Tri-PSPS, a few more examples

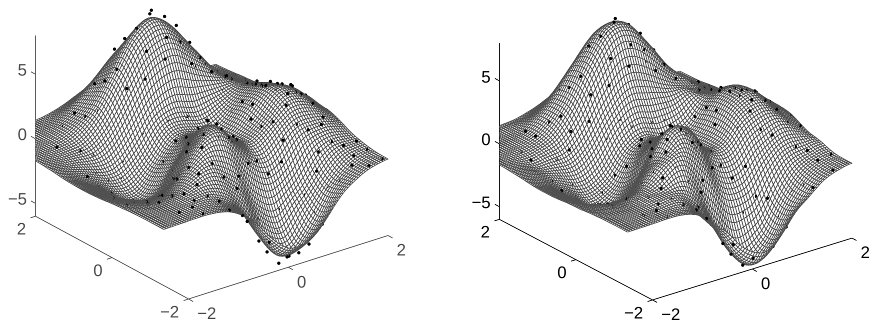


Figure 12: Surface constructed from the same set of points using Tri-PSPS. Left: C^2-smooth Tri-PSPS surface built directly from original point data using C^2-smooth Tri-PSPS with $\delta = 0.15$. Right: C^2-smooth Tri-PSPS surface built from planar primitive for each control hull face using C^2-smooth Tri-PSPS with $\delta = 0.2$. In general, any required approximation accuracy to the control points can be achieved as long as δ is small enough or each local geometric approximation patch is sufficiently accurate.

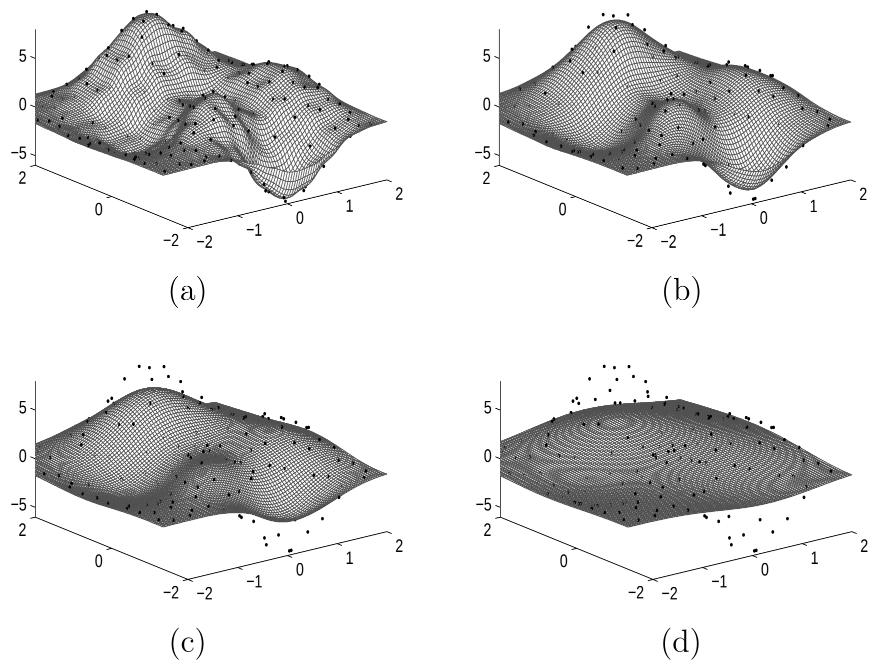


Figure 13: C^2-smooth Tri-PSPS surfaces built with decreased levels of detail: (a). $\delta = 0.1$; (b). $\delta = 0.2$; (c). $\delta = 0.4$; (d). $\delta = 0.8$;

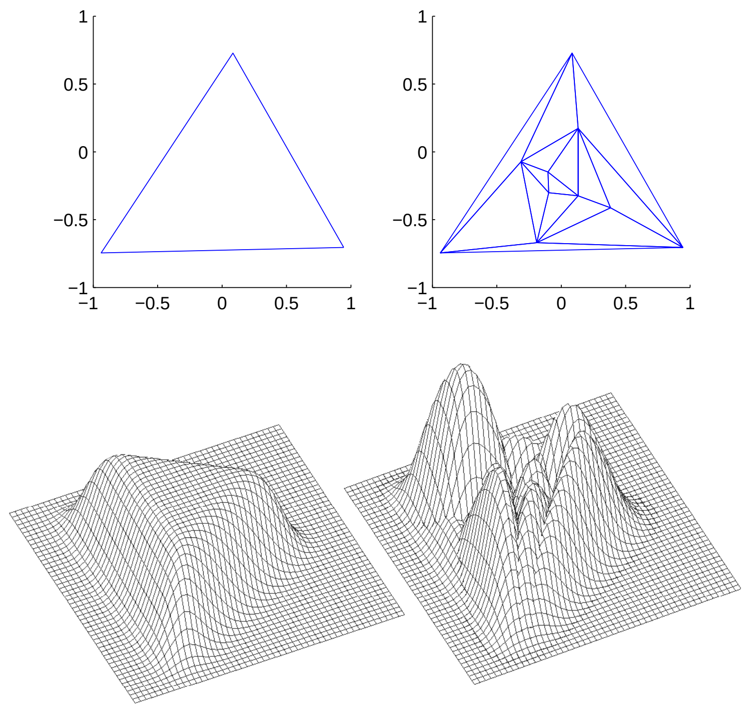


Figure 14: This figure demonstrate the property of additivity of triangular splines. Top row: The initial triangle (left) and a triangle subdivision (right) to the initial triangle; Bottom row: The C^2-smooth triangular spline function built from the initial big triangle (left) and the set of C^2-smooth triangular spline functions built from the set of smaller subdivision triangles (right). The sum of these refined triangular spline functions equal to the one shown on the left.

are provided in Figure 16.

5. Further Discussions

5.1. From Tri-PSPS to Spline Functions on Arbitrary Polygons

In [11], it is shown that for any polygon, whether it be a simple polygon or a polygon with holes, a spline basis function can be directly created from the polygon. As any polygon can be triangulated as a set of triangles, we can also directly construct a spline function with local support around an arbitrary polygon, thanks to the property of additivity of Tri-PSPS. The idea is to subdivide a given polygon $\triangle$ as a set of triangles $\triangle_0$, $\triangle_1$, $\cdots$, $\triangle_K$, and to build a spline function as the sum:

$$B_{\triangle}^{(n)} = \sum_{i=0}^{K} B_{\triangle_i}^{(n)}(x, y).$$

The function built in this way for an arbitrary polygon is independent of the ways that the given polygon is triangulated. In general, a polygon can be divided into different sets of triangles, but the additivity property of Tri-PSPS will guarantee that the total sum of the refined Tri-PSPS functions will be the same, which is independent of the ways a triangle is subdivided.

5.2. Tri-PSPS constructed from transformed triangles

Tri-PSPS is translation and scaling invariant, in the sense that a Tri-PSPS basis function constructed from a transformed triangle can be obtained by transforming the Tri-PSPS basis function built on the initial triangle. This means, when we construct the Tri-PSPS basis functions based on a set of translated or scaled triangles, we can first build the spline basis functions based on their original positions, orientations and sizes of a set of triangles and then transform these functions to obtain the Tri-PSPS functions corresponding to the transformed triangles.

However, Tri-PSPS basis functions are not rotational transformation independent as can be directly observed from the definition of Tri-PSPS basis functions. Once the orientation of a triangle is changed, the corresponding Tri-PSPS basis function needs to be recalculated.

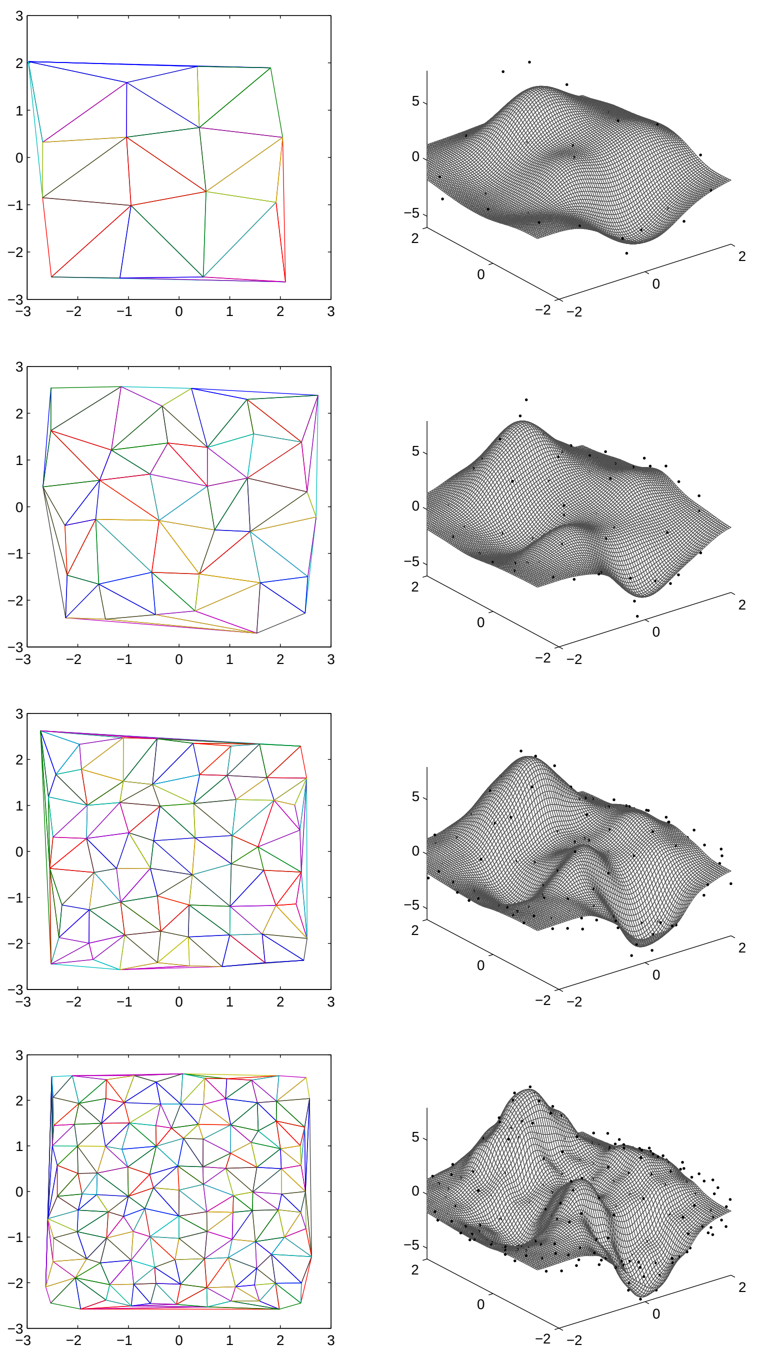
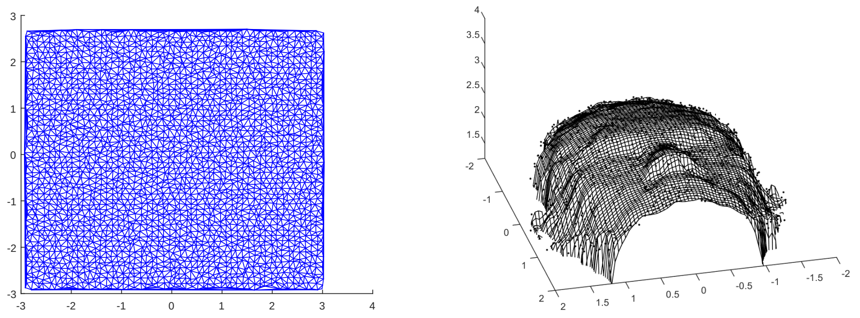
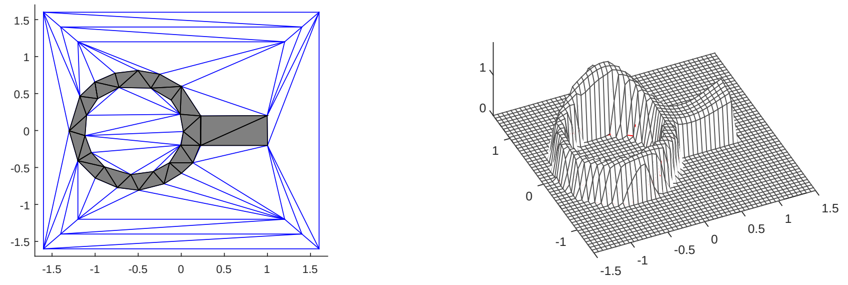


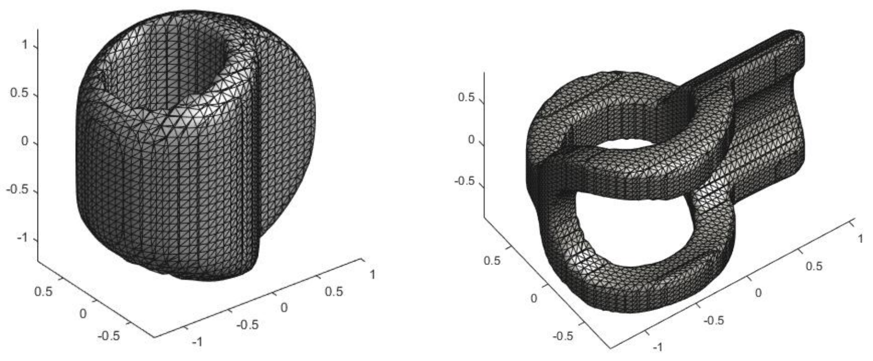
Figure 15: High level of surface details can be achieved by subdividing the domain using much smaller triangles.



(a)



(b)



(c)

Figure 16: Examples of geometric design using Tri-PSPS: (a). Dense point cloud fitting. (b). An explicit Tri-PSPS surface patch (right) designed using the triangulation configuration shown left. (c). An implicit Tri-PSPS surface designed based on the triangulated pattern shown in the left figure in (b).

5.3. Tri-PSPS are parallel computing friendly

Unlike traditional spline technique, Tri-PSPS is particularly suitable to be implemented in parallel processing systems. This is because each spline function is defined locally independent of each other. This is a big difference between Tri-PSPS and traditional spline functions, whose construction often involve the use of a sequence of neighbour triangles.

5.4. Degree-elevation free

In traditional spline techniques like NURBS, interpolating a control point or a portion of line segment is achieved by using lower degrees of spline basis functions. Tri-PSPS use the smoothing parameter introduced in the spline scheme to achieve control point interpolation or control polygon edge interpolation. This indicates that no degree-elevation process is required when perform, for instance, isogeometric analysis. In practice, Tri-PSPS technique is also much easier to implement, as local editing of a Tri-PSPS surface can be done directly through triangle subdivision, without having to handle knots insertion which in general is not a trivial process. Tri-PSPS can be used to construct analysis-suitable parameterization in a similar way as NURBS for isogeometric analysis[16, 17, 15, 18, 19], which will be investigated in a separate paper.

6. Summary

In this paper, we present a new type of triangular spline technique, the Tri-PSPS. Compared with other similar triangular spline techniques, Tri-PSPS have several distinctive features. The most important feature of Tri-PSPS is that the spline basis functions of different degrees can be directly written out in closed form, which is a great advantage over other triangular spline techniques in terms of computational efficiency. The second important property of Tri-PSPS is that it is additive. This means Tri-PSPS are a refinable spline scheme, as each of its spline basis functions can be replaced with a set of refined basis functions built on subdivided triangles. Thirdly, Tri-PSPS are a multilevel spline technique, the required surface details can be controlled either through the specification of a proper value for the smoothing parameter or by using hierarchical control points or control polygons. In terms of practical implementation, Tri-PSPS are a parallel computing friendly spline scheme, which can be easily implemented on modern programmable GPUs or on high performance computer clusters. This is based on the fact that the basis functions of Tri-PSPS can be directly computed independent of its neighboring triangles. Lastly but not least, the implementation of Tri-PSPS is straightforward as the exact form of each triangle spline function can be written out in closed-form explicitly.

References

[1] Gerard Awanou, Ming-Jun Lai, and Paul Wenston. The multivariate spline method for scattered data fitting and numerical solutions of partial differential equations. *Wavelets and splines: Athens*, 2006:24–74, 2005.

[2] Charles K. Chui and Ming-Jun Lai. Algorithms for generating B-nets and graphically displaying spline surfaces on three- and four-directional meshes. *Computer Aided Geometric Design*, 8:479493, 2001.

[3] Wolfgang Dahmen, Charles A. Micchelli, and Hans-Peter Seidel. Blossoming Begets B-Spline Bases Built Better by B-Patches. *Mathematics of Computation*, 59(199):97–115, 1992.

[4] Carl de Boor and Ron DeVore. Approximation by smooth multivariate splines. *Trans. Amer. Math. Soc.*, 276(2):775–788, 1983.

[5] Carl de Boor, Klaus Höllig, and Sherman Riemenschneider. *Box Splines*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[6] Philip Fong and Hans-Peter Seidel. An implementation of multivariate B-spline surfaces over arbitrary triangulations. In *Proceedings of the conference on Graphics interface '92*, pages 1–10, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[7] Hongmei Kang, Falai Chen, and Jiansong Deng. Hierarchical B-splines on regular triangular partitions. *Graphical Models*, 76(5):289–300, 2014.

[8] Ming-Jun Lai and Larry L Schumaker. *Spline functions on triangulations*. Cambridge University Press, 2007.

[9] Qingde Li. Smooth piecewise polynomial blending operations for implicit shapes. *Computer Graphics Forum*, 26(2):157–171, 2007.

[10] Qingde Li, John. G. Griffiths, and James Ward. Constructive implicit fitting. *Comput. Aided Geom. Des.*, 23(1):17–44, 2006.

[11] Qingde Li and Jie Tian. 2D piecewise algebraic splines for implicit modeling. *ACM Trans. on Graph.*, 28(2):1–19, 2009.

[12] Qingde Li and Jie Tian. Partial shape-preserving splines. *Computer-Aided Design*, 43(4):394–409, 2011.

[13] Marian Neamtu. Bivariate simplex B-splines: A new paradigm. In *Proceedings of the 17th Spring Conference on Computer Graphics*, SCCG '01, pages 71–, Washington, DC, USA, 2001. IEEE Computer Society.

[14] Marian Neamtu. What is the natural generalization of univariate splines to higher dimensions? *in Mathematical Methods for Curves and Surfaces, T. Lyche and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville*, pages 355–392, 2001.

[15] N Nguyen-Thanh, J Muthu, X Zhuang, and T Rabczuk. An adaptive three-dimensional RHT-splines formulation in linear elasto-statics and elasto-dynamics. *Computational Mechanics*, 53(2):369–385, 2014.

[16] N Nguyen-Thanh, H Nguyen-Xuan, Stephane Pierre Alain Bordas, and T Rabczuk. Isogeometric analysis using polynomial splines over hierarchical T-meshes for two-dimensional elastic solids. *Computer Methods in Applied Mechanics and Engineering*, 200(21-22):1892–1908, 2011.

[17] Nhon Nguyen-Thanh, J Kiendl, H Nguyen-Xuan, R Wüchner, KU Bletzinger, Y Bazilevs, and T Rabczuk. Rotation free isogeometric thin shell analysis using PHT-splines. *Computer Methods in Applied Mechanics and Engineering*, 200(47):3410–3424, 2011.

[18] Gang Xu, Bernard Mourrain, Régis Duvigneau, and André Galligo. Analysis-suitable volume parameterization of multi-block computational domain in isogeometric applications. *Computer-Aided Design*, 45(2):395–404, 2013.

[19] Gang Xu, Bernard Mourrain, Régis Duvigneau, and André Galligo. Constructing analysis-suitable parameterization of computational domain from cad boundary by variational harmonic method. *Journal of Computational Physics*, 252:275–289, 2013.

[20] Gang Xu, Guo-Zhao Wang, and Xiao-Diao Chen. Free-form deformation with rational DMS-spline volumes. *Journal of computer science and technology*, 23(5):862–873, 2008.

[21] Jia Yue, Zhang Yongjie, Xu Gang, Zhuang Xiaoying, and Rabczuk Timon. Reproducing kernel triangular B-spline-based FEM for solving pdes. *Computer Methods in Applied Mechanics and Engineering*, 267(Complete):342–358, 2013.

[22] Urška Zore and Bert Jüttler. Adaptively refined multilevel spline spaces from generating systems. *Computer Aided Geometric Design*, 31(7):545–566, 2014.