Approximation by Polynomial Splines on Curved Triangulations

Larry L. Schumaker and Annan Yu

Department of Mathematics, Vanderbilt University, Nashville, TN 37240, USA

January 13, 2021

Abstract

Spaces of polynomial splines defined on curved triangulations of curved domains are introduced and studied along with applications to interpolation and data fitting problems.

1 Introduction

In practice we are often faced with the problem of finding an approximation to a function that is defined on a curved planar domain Ω. Typical problems would involve interpolation or fitting of data, as well as the solution of boundary-value problems. A common approach to solving such problems is to use piecewise polynomial spline functions defined on some triangulation which only approximately covers the domain. Although it would seem more natural to work with piecewise polynomial splines defined directly on a curved triangulation of Ω, it seems that there has been very little study of such splines in the spline literature, see Remark 12.1. The purpose of this paper is to rectify this situation by developing basic properties of such spline spaces and then showing how to use them in practice.

The paper is organized as follows. In Sects. 2–4 we introduce curved domains and curved triangulations and show how to construct them. Polynomial splines defined on curved triangulations and how to store and evaluate them are discussed in Sect. 5. In Sect. 6 we describe a quasi-intepolation operator which is used in the following section to investigate the approximation power of our splines. Macro-element spaces defined on curved triangulations are discussed in Sect. 8 and used in Sects. 9 and 10 to solve certain scattered data interpolation problems. Finally, in Sect. 11 we present a useful algorithm for picking well-spaced points in a curved domain. We leave a treatment of boundary-value problems on curved domains to a separate paper, see [12].

2 Curved planar domains

Throughout the paper when we talk about a `curved domain` Ω we mean a closed connected set lying in $\mathbb{R}^2$ whose boundary $\partial\Omega$ is defined by $m+1$ disjoint non-self intersecting Lipschitz continuous parametric curves, where m is the number of holes in the domain. We do not require Ω to be convex, nor that the parametrizations of the boundary curves be arc-length parametrizations. Several examples of typical curved domains that might arise in practice are shown in Fig. 3.

In practice the curves definining the boundary of a curved domain Ω are usually defined as closed NURBs curves. But they could also be defined in terms of piecewise elementary parametric curves such as lines, circles, ovals, etc.. For our purposes we may assume without loss of generality that the boundary curves are oriented so we can tell on which side of the curve the domain lies on.

3 Curved triangulations

We begin by defining what we mean by a curved triangle.

Definition 3.1. *Suppose v_1, v_2, v_3 are three noncollinear points in the plane. Suppose e_1 is a curve segment with endpoints at v_1 and v_2. Similarly, let e_2 and e_3 be curve segments joining v_2, v_3 and v_3, v_1, respectively. Suppose that the e_i intersect each other only at at their endpoints. Then we define a* `curved triangle` *with* `vertices` *v_1, v_2, v_3 and edges e_1, e_2, e_3 to be the closed subset $\tilde{T}$ of $\mathbb{R}^2$ enclosed by the three edges.*

A curved triangle has three vertices and three edges. If all three edges are straight lines we call it an `ordinary triangle`. A curved triangle need not be convex. Each curved triangle $\tilde{T}$ is associated with a unique ordinary triangle T obtained by replacing each curved edge with a straight edge. We call this the `companion triangle`. We can now define what we mean by a curved triangulation.

Definition 3.2. *Suppose Ω is a curved domain in $\mathbb{R}^2$, and suppose that $\tilde{\triangle} = \{\tilde{T}_i\}_{i=1}^{n_t}$ is a set of curved triangles whose union exactly covers Ω, and is such that any two curved triangles in $\tilde{\triangle}$ can intersect each other only at a vertex or along a common edge. Then we call $\tilde{\triangle}$ a* `curved triangulation` *of the domain Ω. If we replace each of the $\tilde{T}_i$ by its companion triangle, then we get an ordinary triangulation $\triangle$ which we call the* `companion triangulation to` $\tilde{\triangle}$.

Note that according to this definition, all edges of a curved triangulation (including interior edges) can be curved. However, in practice we will work only with curved triangulations whose interior edges are all straight lines. Curved triangulations constructed by the methods of the following section will always have this property.

4 Constructing curved triangulations

Suppose Ω is a curved domain. In this section we outline an algorithm for constructing a curved triangulation of Ω. The first step is to construct a so-called inscribed triangulation.

Definition 4.1. *Suppose that $\triangle = \{T_i\}_{i=1}^{n_t}$ is an ordinary triangulation whose vertices lie in the domain Ω, and whose boundary vertices lie on the boundary of Ω. We call such a triangulation an* `inscribed triangulation of` Ω.

To construct an inscribed triangulation for a given curved domain Ω we first have to identify the points inside Ω and on the boundary of Ω that will be used for the vertices of the triangulation. The choice of the vertices depends on what kind of problem we are solving: a) For scattered data fitting methods such as those discussed in Sections 8-9 below, these points will be given to us as part of the problem. b) For constructing a quasi-interpolant as discussed in Sect. 6 (and also for solving boundary-value problems, see [12],) we are free to choose the vertices. In Section 11 we describe an algorithm that can be used for picking vertices.

Once we have selected the vertices, the next step is to construct the triangulation. For this step we use a `constrained Delaunay triangulation` $\triangle$ where the constraints make sure that the boundary vertices of $\triangle$ lie on the boundary of Ω. For the purposes of this paper we have carried out this step using the Matlab function `delaunayTriangulation`. This code produces triangulations whose smallest angles are maximized.

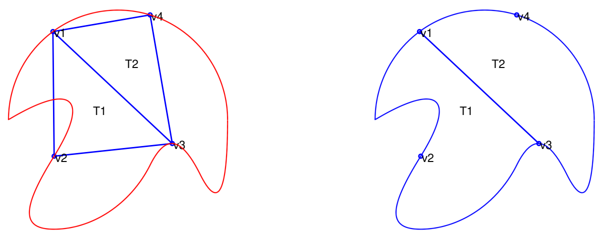

Figure 1: An inscribed triangulation of a curved domain and the associated curved triangulation

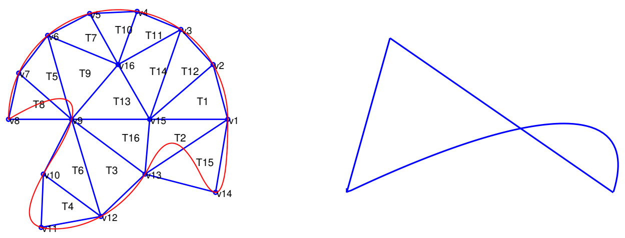

Figure 2: An inscribed triangulation of a curved domain that cannot be converted to a curved triangulation

We now present an algorithm for converting an initial inscribed triangulation into a curved triangulation of the curved domain. Let $\triangle = \{T_i\}_{i=1}^{n_t}$ be an inscribed triangulation associated with a curved domain Ω with m holes. Such a domain is defined by $m+1$ nonintersecting boundary curves $\partial\Omega_0, \ldots, \partial\Omega_m$.

Algorithm 4.1. *For each $j = 0, \ldots, m$,*

1) Suppose $u_1^j, \ldots, u_{n_j}^j$ are the boundary vertices of $\triangle$ that lie on $\partial\Omega_j$, ordered in counterclockwise order around $\partial\Omega$, and let $u_{n_j+1}^j = u_1^j$.

2) For each $1 \le i \le n_j$, replace the edge of the triangle in $\triangle$ with endpoints u_i^j and u_{i+1}^j by the curve segment e_i^j of $\partial\Omega_j$ that connects u_i^j and u_{i+1}^j.

Fig. 1 shows an example of a curved domain along with an inscribed triangulation that has been converted to a curved triangulation. It turns out that for a given curved domain Ω, not all inscribed triangulations of Ω can be converted to curved triangulations. Consider the inscribed triangulation shown in Fig. 2, and consider the triangle T_8 with vertices v_7, v_8 and v_9. If we replace the edge $\langle v_8, v_9\rangle$ of T_8 by the part of $\partial\Omega$ that connects these two vertices, we do not get a curved triangle, see Fig. 2 (right). The problem is that this curve segment intersects an interior edge of $\triangle$ at a point in the interior of that edge. For an inscribed triangulation to be convertible, we will have to exclude such anomalies.

Definition 4.2. *Suppose that $\triangle = \{T_i\}_{i=1}^{n_t}$ is an inscribed triangulation associated with a curved domain Ω, and let $e_1^j, \ldots, e_{n_j}^j$ be the curve segments introduced in step 2 of the algorithm above. Then we say that $\triangle$ is* `convertible` *provided that for all $0 \le j \le m$,*

1) The union of $e_1^j, \ldots, e_{n_j}^j$ cover all of $\partial\Omega_j$,

2) A boundary segment e_i^j of $\partial\Omega_j$ can intersect an interior edge e of $\triangle$ only at the endpoints of e.

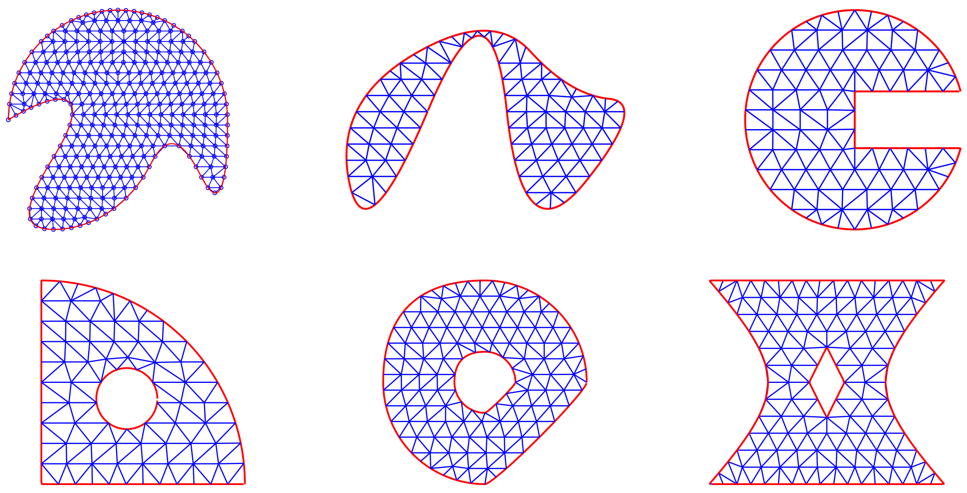

Figure 3: Curved triangulations of several curved domains

The inscribed triangulation in Fig. 1 (left) is convertible, but the one in Fig. 2 (left) is not. For curved domains Ω with well-behaved boundaries, we will generally get convertible inscribed triangulations if we choose enough well-spaced vertices on the boundary to follow the shape of the boundary.

In Fig. 3 we show curved triangulations of several different curved domains defined by NURBs, parametric curves, and implicit curves. Note that those in the last row have holes in them.

5 Polynomial splines on curved triangulations

We are ready to introduce the spline spaces of interest in this paper.

Definition 5.1. *Suppose $\tilde{\triangle}$ is a curved triangulation of a curved domain Ω, and let $0 \le r < d$. Then we define the space of* `polynomial splines of degree` d `and smoothness` r *on $\tilde{\triangle}$ as*

$$\mathcal{S}_d^r(\tilde{\triangle}) := \{s \in C^r(\Omega) : s|_{\tilde{T}} \in \mathcal{P}_d, \text{ all } \tilde{T} \in \tilde{\triangle}\}. \tag{5.1}$$

where $\mathcal{P}_d$ is the space of bivariate polynomials of degree d.

This space is clearly a finite dimensional linear space of functions. There is a close relationship between it and the classical polynomial spline spaces defined on ordinary triangulations, see [7] and [11]. Indeed, if $\tilde{s}$ is a polynomial spline on a curved triangulation, then it can be considered as the extension/restriction of a polynomial spline s on the associated ordinary triangulation. This holds since any polynomial is defined globally, and if T is the companion triangle associated with the curved triangle $\tilde{T}$ then both $\tilde{s}|_{\tilde{T}}$ and s_T are represented by the same polynomial. It follows that the dimension of $\mathcal{S}_d^r(\tilde{\triangle})$ and $\mathcal{S}_d^r(\triangle)$ are the same. For example,

$$\dim \mathcal{S}_d^0(\tilde{\triangle}) = n_v + (d-1)n_e + \binom{d-1}{2} n_t, \tag{5.2}$$

where n_v, n_e, n_t are the numbers of vertices, edges, and triangles in $\tilde{\triangle}$.

Because of the close connection between polynomial splines on a curved triangulation and polynomial splines on the associated ordinary triangulation, we can easily modify computational methods developed for polynomial splines on ordinary triangulations (see [11]) to work with splines on curved triangulations.

5.1 Storing a spline on a curved triangulation

The computational methods discussed in [11] for dealing with splines on an ordinary triangulation are based on the fact that for any $r \geq 0$, $\mathcal{S}_d^r(\triangle)$ is a subspace of $\mathcal{S}_d^0(\triangle)$. This means that we can store a spline s in $\mathcal{S}_d^r(\triangle)$ as a spline in $\mathcal{S}_d^0(\triangle)$. As explained in Sect.4.9 of [11], this can accomplished most efficiently by storing a vector c containing the Bernstein–Bézier coefficients of s, see the books [7] and [11]. Each such coefficient is associated with a `domain point` which may be at a vertex, in the interior of an edge, or inside a triangle. It follows that to store a spline $\tilde{s} \in \mathcal{S}_d^r(\tilde{\triangle})$, we simply consider the associated spline $s \in \mathcal{S}_d^r(\triangle)$ and store its coefficient vector. The Matlab package accompanying the book [11] contains a function `getindex` that can be used to recover the indices of the B-coefficients associated with the $\binom{d+2}{2}$ domain points in a given triangle T. The resulting coefficients are precisely the coefficients defining s on T and $\tilde{T}$.

5.2 Evaluating a spline on a curved triangulation

Let $\tilde{s}$ be a polynomial spline defined on a curved triangulation $\tilde{\triangle}$ of a curved domain Ω, and suppose we want to evaluate $\tilde{s}$ at a point u in Ω. The first step in finding $\tilde{s}(u)$ is to locate which curved triangle $\tilde{T}$ of $\tilde{\triangle}$ that u lies in. Once we know this we can look up the coefficients corresponding to the domain points $\mathcal{D}_{d,T}$ lying in the associated companion triangle T. We can then use them to find the value of the corresponding polynomial at any point in $\mathbb{R}^2$, and in particular at u. That will give us the value of $\tilde{s}(u)$.

To help solve the problem of which curved triangle $\tilde{T}$ contains a given point $u \in \Omega$, it is convenient to construct another ordinary triangulation $\bar{\triangle}$ associated with $\tilde{\triangle}$ which we call the `extended triangulation`. Here is an outline of an algorithm that can be used to create an extended triangulation associated with a given curved triangulation $\tilde{\triangle}$. Let $\triangle$ be the companion triangulation.

Algorithm 5.1. *Suppose (x, y) are the vectors of Cartesian coordinates of the vertices of $\triangle$ and let n_b be the number of boundary vertices.*

1) pick an offset value $\varepsilon > 0$

2) For each boundary vertex of (x_i, y_i) of $\tilde{\triangle}$, let $(\bar{x}_i, \bar{y}_i)$ be the point that lies at an offset distance of ε in the direction of the outward normal vector at (x_i, y_i),

3) To define $\bar{\triangle}$, supplement $\triangle$ with a ring of $2n_b$ triangles whose vertices are chosen from the boundary vertices of $\triangle$ along with thet new points $(\bar{x}_i, \bar{y}_i)$ introduced in step 2.

In Fig. 4 (right) we show an extended triangulation $\bar{\triangle}$ associated with the curved triangulation in Fig. 4 (left).

Note that the union of the triangles in an inscribed triangulation $\triangle$ may not cover all of Ω. However, if we choose ε large enough the union of the triangles in the extended triangulation $\bar{\triangle}$ will cover Ω. Now suppose we want to evaluate a spline s at a point u in Ω. If u lies in one of the triangles T of the inscribed triangulation, then we look up the B-coefficients associated with T and evaluate the corresponding polynomial. If u lies in one of the triangles in the outer ring, we have to use the coefficients associated with a neighboring triangle in $\triangle$. The indices of these neighbors can be stored in a vector of length $2n_b$.

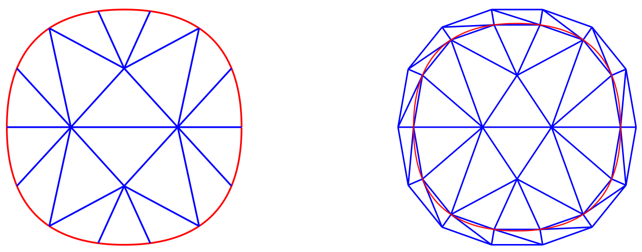

Figure 4: A curved triangulation and an associated extended triangulation

6 A quasi-interpolation operator

Let $\tilde{\triangle}$ be a curved triangulation of a curved domain Ω. In this section we show how to construct a quasi-interpolation operator Q that maps continuous functions on Ω to splines in $\mathcal{S}_d^0(\tilde{\triangle})$. The mapping Q will be a locally bounded linear projection which we will use in the following section to show that the space $\mathcal{S}_d^0(\tilde{\triangle})$ has full approximation power.

Algorithm 6.1. *For $i = 1, \ldots, n_t$*

1) Given a curved triangle $\tilde{T}_i$, let T_i be the companion triangle associated with $\tilde{T}_i$, and let $\hat{T}_i$ be an ordinary triangle that is contained in $\tilde{T}_i$,

2) Let $\mathcal{D}_{d,\hat{T}_i}$ be the set of domain points of order d associated with $\hat{T}_i$,

3) Find the $n_d := \binom{d+2}{2}$ B-coefficients of a polynomial p_i of degree d that interpolates the values of f at the points in $\mathcal{D}_{d,\hat{T}_i}$,

4) Use these coefficients to calculate the values $F = \{F_\xi := p_i(\xi)\}_{\xi \in \mathcal{D}_{d,T_i}}$,

5) For each edge e of T_i that is an interior edge of $\tilde{\triangle}$ and for each domain point $\xi \in \mathcal{D}_{d,T_i}$ lying on e, replace the value F_ξ by $f(\xi)$,

6) Compute the coefficients of a polynomial of degree d that interpolates the modified values F at the domain points $\mathcal{D}_{d,T_i}$,

7) Store these coefficients in a vector c according to the storage convention described in Sect. 4.9.1 of [11].

This algorithm defines a mapping from $C(\Omega)$ to a piecewise polynomial Qf of degree d defined on the curved triangulation $\tilde{\triangle}$. Moreover, for every interior edge e of $\tilde{\triangle}$, the polynomials defined on the two triangles sharing e interpolate the same data, and thus are the same. This guarantees that Qf is continuous and thus belongs to $\mathcal{S}_d^0(\tilde{\triangle})$.

For curved triangles T in $\triangle$ that contain their companion ordinary triangle, we can take $\hat{T}_i$ to be that companion triangle. Otherwise we will have to select a smaller triangle for $\hat{T}_i$. It is not hard to write code to do this automatically. To get the best possible error bounds, we should choose $\hat{T}_i$ to be the largest triangle that fits in $\tilde{T}_i$. Its orientation is not important.

To illustrate how Algorithm 6.1 works, consider the curved triangulation $\tilde{\triangle}$ shown in Fig. 5 (left). For this example there are six triangles $\tilde{T}$ where we cannot choose $\hat{T}_i$ to be the companion triangle to $\tilde{T}_i$. These are marked in Fig. 5 (left), where rather than showing the triangles $\hat{T}_i$ themselves, we have plotted the degree three domain points that each of them contains.

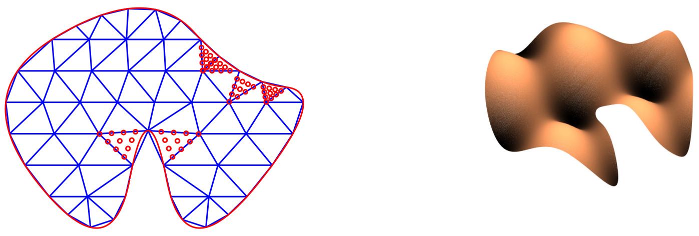

Figure 5: (left) Domain points in the triangles $\hat{T}_i$ chosen in step 1 of Algorithm 6.1. (right) The quintic quasi-interpolating spline of Example 6.1

In the remainder of this section we give several examples exploring how well Q approximates smooth functions.

Example 6.1. *Approximate the function $f = \sin(10x) + \sin(10y)$ on the curved domain shown in Fig. 5 (left) on a sequence of finer and finer curved triangulations. Use the spline space $\mathcal{S}^0_d(\tilde{\triangle})$ with $d = 5$.*

Discussion: To explore the rate of convergence, we need a sequence of triangulations whose mesh sizes go down by approximately a factor of 1/2 in each step. To construct such a sequence, we use Algorithm 11.1 in Sect. 11 below for $ny = 5, 9, 17, 33, 65$. The following table shows the value of ny, the number of triangles, the number of coefficients of the spline, and both the max and RMS errors on a set of 57092 points covering Ω.

ny	nt	nc	emax	rms	rates	
5	49	676	4.37e-03	2.47e-04		
9	160	2106	5.40e-05	3.44e-06	6.34	6.17
17	542	6961	9.64e-07	6.76e-08	5.81	5.67
33	1989	25216	6.77e-09	1.22e-09	7.15	5.80
65	7638	96166	1.12e-10	2.16e-11	5.92	5.82

Fig. 5 (right) shows the quintic quasi-interpolating spline Qf corresponding to the curved triangulation on the left. The table show an approximation rate of six for both the max and RMS errors. This is optimal for splines of degree five. □

Here is an example with a more unusual curved domain.

Example 6.2. *Repeat Example 6.1 for the second curved domain shown in Fig. 3 with $d = 6$.*

Discussion: Using Algorithm 11.1 with $ny = 5, 9, 17, 33, 65$ leads to a sequence of curved triangulatons whose mesh sizes go down by a factor of approximately 1/2 at each step. The following table shows the values of ny, the number of triangles, the number of coefficients of the spline, and both the max and RMS errors on a set of 44755 points covering Ω.

ny	nt	nc	emax	rms	rates	
5	40	823	2.23e-03	1.64e-04		
9	118	2287	5.59e-06	2.81e-07	8.64	9.19
17	409	7648	4.85e-08	1.96e-09	6.85	7.16
33	1543	28318	1.45e-10	1.16e-11	8.39	7.40
65	5921	107626	1.43e-12	1.02e-13	6.66	6.83

Fig. 6 (left) shows the the quasi-interpolating spline of degree six corresponding to the curved triangulation of Ω with `ny = 9`. The table shows an approximation rate of seven. This is optimal for splines of degree six. $\square$

We now present an example of a curved domain with a hole.

Example 6.3. *Approximate the function* $f = \sin(10x) + \sin(10y)$ *on the fifth curved domain shown in Fig. 3. Use the spline space* $\mathcal{S}_d^0(\tilde{\triangle})$ *with* $d = 5$.

Discussion: We again use Algorithm 11.1 with $ny = 5, 9, 17, 33, 65$ to construct a sequence of triangulations with mesh sizes going down by a factor of approximately 1/2 at each step. The following table shows the value of ny, the number of triangles, the number of coefficients of the spline, and both the max and RMS errors on a set of 43149 points covering Ω.

ny	nt	nc	emax	rms	rates	
5	38	535	7.46e-02	5.21e-03		
9	122	1625	8.34e-04	8.01e-05	6.48	6.02
17	414	5355	1.20e-05	1.42e-06	6.11	5.82
33	1498	19060	1.46e-07	2.74e-08	6.37	5.69
65	5772	72790	7.44e-09	4.68e-10	4.30	5.87

Fig. 6 (right) shows the resulting quasi-interpolating spline corresponding to the triangulation with `ny = 9`. The table shows an approximation rate of six. This is optimal for splines of degree five. $\square$

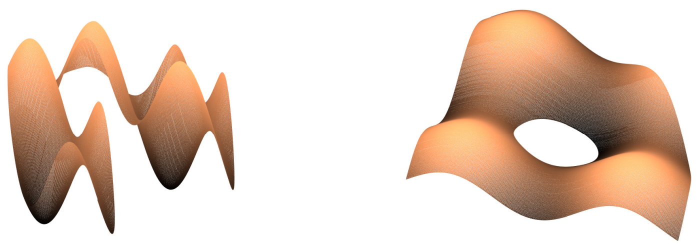

Figure 6: The quasi-interpolating splines for Examples 6.2 and 6.3

7 Approximation power of $\mathcal{S}_d^0(\tilde{\triangle})$

Error bounds for how well smooth functions can be approximated by polynomial splines on ordinary triangulations play a central role in classical spline theory. We can derive results for polynomial splines on curved triangulations using the quasi-interpolation operator introduced in Sect. 6. For any bounded set of points P in $\mathbb{R}$ we define its diameter $|P|$ to be the diameter of the smallest disk that contains P. Given a curved triangulation $\tilde{\triangle} = \{\tilde{T}_i\}_{i=1}^{n_t}$, let $\hat{T}_1, \ldots, \hat{T}_{n_t}$ be the subtriangles selected in Algorithm 6.1, and let

$$\beta := \max_i \frac{|\tilde{T}_i|}{|\hat{T}_i|}. \tag{7.1}$$

Theorem 7.1. *The quasi-interpolation operator* Q *defined above has the following properties:*

1) Q *is a linear projection mapping* $C^0(\Omega)$ *onto* $\mathcal{S}_d^0(\tilde{\triangle})$, *2) Q is local in the sense that for every curved triangle $\tilde{T}$, $Qf|_{\tilde{T}}$ depends only on values of f in $\tilde{T}$,*

3) Q is locally bounded in the max norm, i.e., there exists a constant K_Q depending only on β such that for every curved triangle $\tilde{T}$, $\|Qf\|_{\tilde{T}} \leq K_Q \|f\|_{\tilde{T}}$.

Proof: For any curved triangle $\tilde{T}_i$, the polynomial defining Qf on $\tilde{T}_i$ is the polynomial interpolating f at the domain points $\mathcal{D}_{d,\hat{T}_i}$. This gives properties 1) and 2). The B-coefficients of Qf relative to the triangle $\hat{T}_i$ are computed from a linear system whose matrix M is the same for all such triangles. Then $\|M^{-1}\| \|f\|_{\hat{T}_i}$ provides an upper bound for the size of these coefficients. By properties of the Bernstein polynomials, we conclude that $\|Qf\|_{\hat{T}_i} \leq K_1 \|f\|_{\hat{T}_i}$, where $K_1 = \|M^{-1}\|$. Since Qf is a polynomial of degree d on $\tilde{T}_i$, its max norm on $\tilde{T}_i$ is bounded by β^d times its norm on $\hat{T}_i$, and property 3) follows. $\square$

We are ready to present our approximation theorem. Let $\tilde{T}$ be a curved triangle in a curved triangulation $\tilde{\triangle}$. We define the `shape parameter of` $\tilde{T}$ to be the ratio

$$\kappa_{\tilde{T}} := \frac{|\tilde{T}|}{\rho_{\tilde{T}}},$$

where $\rho_{\tilde{T}}$ is the radius of the largest disk that can be embedded in $\tilde{T}$.

Theorem 7.2. *Let Q be the quasi-interpolant defined above. Then there exists a constant K such that for every $f \in C^{m+1}(\Omega)$ with $0 \leq m \leq d$.*

$$\|D_x^\alpha D_y^\beta (f - Qf)\|_{\tilde{T}} \leq K\, |\tilde{T}|^{d+1-\alpha-\beta}\, |f|_{d+1,\tilde{T}}, \tag{7.2}$$

for all $0 \leq \alpha + \beta \leq d$. The constant K depends only on d, the shape parameter $\kappa_{\tilde{T}}$, the constant β defined above, and the Lipschitz constant of the boundary of $\tilde{T}$.

Proof: By Theorem 1.9 of [7], there exists an averaged Taylor polynomial p of degree d depending on f such that (7.2) holds with Qf replaced by p. By the triangle inequality we have

$$\|D_x^\alpha D_y^\beta (f - Qf)\| \leq \|D_x^\alpha D_y^\beta (f - p)\| + \|D_x^\alpha D_y^\beta (p - Qf))\|. \tag{7.3}$$

Now using the fact that $Qp = p$, and applying the Markov inequality for derivatives of polynomials, we can write

$$\begin{aligned} \|D_x^\alpha D_y^\beta (p - Qf))\| &= \|D_x^\alpha D_y^\beta (Qp - Qf))\| \\ \leq C\rho_{\tilde{T}}^{-(\alpha+\beta)} \|Q(f-p)\| &\leq CK_Q \rho_{\tilde{T}}^{-(\alpha+\beta)} \|(f-p)\| \\ &\leq CK_Q \kappa_{\tilde{T}}^{\alpha+\beta} |\tilde{T}|^{d+1-\alpha-\beta} |f|_{d+1,\tilde{T}}, \end{aligned} \tag{7.4}$$

where all norms are taken to be the max norm over $\tilde{T}$, and C is the constant in the Markov inequality. Inserting this in (7.3) we get (7.2). $\square$

8 Macro-element spaces

So-called macro-element spaces play an important role in classical spline theory, see [7] and [11]. They are typically spaces of so-called super splines. Here is the definition of a general super-spline space on a curved triangulation.

Definition 8.1. *Suppose $\tilde{\triangle}$ is a curved triangulation of a curved domain Ω, and let $0 \leq r < \rho < d$. Then we define the space of* `super splines of degree` d`, smoothness` r`, and super-smoothness` *ρ on $\tilde{\triangle}$ as*

$$\mathcal{S}_d^{r,\rho}(\tilde{\triangle}) := \{s \in \mathcal{S}_d^r(\tilde{\triangle}) \cap C^\rho(v) \text{ for every vertex } v \text{ of } \tilde{\triangle}\}. \tag{8.1}$$

Macro-element spline spaces are often defined on some kind of regular refinement of a given triangulation. For ordinary triangulations, the most commonly used refinements are the so-called `Clough-Tocher` and `Powell-Sabin` refinement, see [7] and [11]. The Clough-Tocher refinement involves splitting each triangle into three subtriangles using the barycenter, while the Powell-Sabin refinement involves splits into six subtriangles using incenters. The fact that a given inscribed triangulation is convertible does not imply that a refinement of it will also be convertible. However, for cuved domains with well-behaved boundaries, if we use inscribed triangulations with enough boundary points we will generally be able to construct associated curved triangulations. Fig. 7 shows the Clough-Tocher and Powell-Sabin refinements of the curved triangulation shown in Fig. 4 (left).

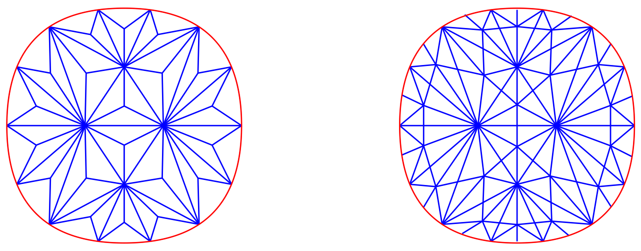

Figure 7: The Clough-Tocher and Powell-Sabin refinements of the curved triangulation shown in Fig. 4 (left)

9 Two-stage methods for scattered data interpolation

In this section we discuss several algorithms for solving the following `scattered data interpolation problem`. Let $\tilde{\triangle}$ be a curved triangulation of a curved domain Ω.

Problem 9.1. *Suppose* $\{(x_i, y_i)\}_{i=1}^n$ *is a set of scattered points in* Ω*, and let* f *be a function defined on* Ω*, Find a spline* s *defined on a curved triangulation* $\tilde{\triangle}$ *of* Ω *such that*

$$s(x_i, y_i) = f(x_i, y_i), \qquad i = 1, \ldots, n. \tag{9.1}$$

We are going to solve this problem by adapting the two-stage methods described in Sect. 6.7 of [11]. The basic idea is to use Hermite interpolation schemes where the needed derivative information is approximated from the scattered data. In [11] five such methods are developed using polynomial splines on ordinary triangulations. In the following subsections we show how to adapt three of these methods to the case of curved domains. The other two method discussed there can also be extended in a similar way to the curved case.

9.1 A local scattered data method based on $\mathcal{S}_5^{1,2}(\tilde{\triangle})$

As discussed in Sect. 5.5 of [11] for scattered data interpolation problems using ordinary triangulations, the first stage in working with this kind of super-spline space is to estimate derivatives up to order two at each scattered data point along with a cross-derivative at the midpoint of each boundary edge. By working on the companion ordinary triangulation we can carry out this step using local least-square polynomial approximation with the function `derest15` in the spline package distributed with the book [11]. The coefficients of the interpolating spline can then be found with the function `arg15`.

Here are several examples illustrating the performance of the method.

Example 9.1. *Consider the function* $f(x, y) = \sin(20x) + \sin(20y)$ *on the domain shown in Fig. 5 (left). Approximate this function with polynomial splines in the spaces* $\mathcal{S}_5^{1,2}(\tilde{\triangle})$ *for a sequence of curved triangulations with decreasing mesh size.*

Discussion: We create the sequence of triangulations using the algorithm discussed in Sect. 11. It requires choosing a number `ny` of horizontal lines to control the mesh size. Here we work with `ny = 9,17,33,65`. For each choice of `ny` the table gives the total time to compute the spline, the number of triangles and number of coefficients and the max and RMS errors measured on a well distributed set of 57092 points spread out across Ω.

ny	time	nc	cond	emax	rms	rates	
9	0.51	160	2106	1.35e+00	4.09e-01		
17	0.44	542	6961	2.69e-01	4.49e-02	2.32	3.19
33	1.45	1989	25216	4.66e-02	3.00e-03	2.53	3.90
65	1.47	7638	96166	2.24e-03	1.73e-04	4.38	4.12

The table shows that the method is producing a rate of convegence of at least four. As discussed in Sect. 5.5 of [11], if we had exact derivatives we should be getting order six convergence since we are working with quintic splines. However, as explained in Sect. 6.7.3 of that book, since we are using cubic polynomials to estimate derivatives, we can only expect a rate of convergence of four, which is what we are getting here. □

Here is an example involving a curved domain with a hole.

Example 9.2. *Consider the function* $f(x, y) = \sin(20x) + \sin(20y)$ *on the fifth domain shown in Fig. 3 (left). Approximate this function with polynomial splines in the spaces* $\mathcal{S}_5^{1,2}(\tilde{\triangle})$ *for a sequence of curved triangulations with decreasing mesh size.*

Discussion: We construct the sequence of triangulations using Algorithm 11.1 with `ny = 9,17,33,65`. For each choice of `ny` the following table gives the total time to compute the spline, the number of triangles and number of coefficients and the max and RMS errors measured on a well distributed set of 43149 points spread out across Ω.

ny	time	nc	cond	emax	rms	rates	
9	0.03	122	1625	8.25e-01	2.54e-01		
17	0.08	414	5355	1.48e-01	2.51e-02	2.48	3.34
33	0.42	1498	19060	1.06e-02	1.46e-03	3.80	4.11
65	1.11	5772	72790	8.97e-04	8.81e-05	3.57	4.05

The results are quite similar to those in the previous example. We are again getting fourth order convergence. □

9.2 A local scattered data method based on $\mathcal{S}_3^1(\tilde{\triangle}_{CT})$

Given a curved triangulation $\tilde{\triangle}$, let $\tilde{\triangle}_{CT}$ be the associated Clough-Tocher refinement of $\tilde{\triangle}$. In this section we are going to work with the macro-element space $\mathcal{S}_3^1(\tilde{\triangle}_{CT})$ of C^1 cubic splines defined on $\tilde{\triangle}_{CT}$. Following Sect. 6.7.2 of [11] the first step is to estimate the gradients of f at each of the points $\{(x_i, y_i)\}_{i=1}^n$ using a local quadratic least-squares polynomial. We also must estimate a cross-derivative at the center of each edge of the initial triangulation. For this we can use the function `derestct` described in Sect. 6.7.2 of [11].

Here is an example.

Example 9.3. *Consider the function $f(x, y) = \sin(20x) + \sin(20y)$ on the domain shown in Fig. 5 (left). Approximate this function with polynomial splines in the spaces $\mathcal{S}_3^1(\tilde{\triangle}_{CT})$ for a sequence of curved Clough-Tocher triangulations with decreasing mesh size.*

Discussion: We construct the curved triangulations using Algorithm 11.1 with `ny = 9,17,33,65`. For each choice of `ny` the following table gives the total time to compute the spline, the number of triangles and number of coefficients, and the max and RMS errors measured on a well distributed set of 57092 points spread out across Ω.

ny	time	nc	cond	emax	rms	rates	
9	0.54	480	2224	1.53e+00	4.18e-01		
17	0.68	1626	7429	2.63e-01	4.34e-02	2.54	3.27
33	2.04	5967	27064	4.34e-02	2.56e-03	2.60	4.08
65	2.35	22914	103528	2.01e-03	1.19e-04	4.43	4.43

The table shows that we are getting a convergence rate of four which is optimal for cubic splines. □

9.3 A local scattered data method based on $\mathcal{S}_2^1(\tilde{\triangle}_{PS})$

Given a curved triangulation $\tilde{\triangle}$, let $\tilde{\triangle}_{PS}$ be the associated Powell-Sabin refinement of $\tilde{\triangle}$ discussed above. In this section we are going to work with the macro-element space $\mathcal{S}_2^1(\tilde{\triangle}_{PS})$ of C^1 quadratic splines defined on $\tilde{\triangle}_{PS}$. Following Sect. 6.7.1 of [11] the first step is to estimate the gradients of f at each of the points $\{(x_i, y_i)\}_{i=1}^n$ using a local quadratic least-squares polynomial.

Here is an example.

Example 9.4. *Consider the function $f(x, y) = \sin(20x) + \sin(20y)$ on the domain shown in Fig. 5 (left). Approximate this function with polynomial splines in the spaces $\mathcal{S}_2^1(\tilde{\triangle}_{PS})$ for a sequence of curved Powell-Sabin triangulations with decreasing mesh size.*

Discussion: We work with values of `ny = 9,17,33,65`. For each choice of `ny` the following table gives the total time to compute the spline, the number of triangles and number of coefficients, and the max and RMS errors measured on a well distributed set of 57092 points spread out across Ω.

ny	time	nc	cond	emax	rms	rates	
9	0.02	960	2005	1.56e+00	4.28e-01		
17	0.06	3252	6653	3.05e-01	4.82e-02	2.36	3.15
33	0.22	11934	24151	5.12e-02	3.43e-03	2.57	3.81
65	0.77	45828	92209	2.46e-03	2.64e-04	4.38	3.70

Since we are using quadratic splines we expect convergence of order three – here we seem to be getting an even higher rate. □

10 Minimal energy interpolation of scattered data

In this section we show how the minimal energy method discussed in Sect. 6.3 of [11] for polynomial splines on ordinary triangulations can be adapted to work with curved triangulations. Suppose $\tilde{\triangle} := \{\tilde{T}_i\}_{i=1}^{n_t}$ is a curved triangulation of a curved domain Ω and let $\triangle := \{T_i\}_{i=1}^{n_t}$ be the asssociated inscribed triangulation of Ω. Let $\{(x_i, y_i)\}_{i=1}^n$ be the Cartesian coordinates of the vertices of $\tilde{\triangle}$, and let $\mathcal{S}(\tilde{\triangle})$ be a space of splines defined on $\tilde{\triangle}$. Our aim is to find a spline $s \in \mathcal{S}(\tilde{\triangle})$ that interpolates the data in the sense that

$$s(x_i, y_i) = f(x_i, y_i), \qquad i = 1, \ldots, n, \tag{10.1}$$

and which has minimal energy, where we measure the `energy of the spline` by the expression

$$E(s) = \sum_{\nu=1}^{n_t} \int_{\tilde{T}_\nu} [(s_{xx})^2 + 2(s_{xy})^2 + (s_{yy})^2]\, dx\, dy, \tag{10.2}$$

Definition 10.1. *Suppose*

$$U := \{s \in \mathcal{S}(\tilde{\triangle}) : s(x_i, y_i) = z_i,\ i = 1, \ldots, n\}. \tag{10.3}$$

and that s is a spline in $\mathcal{S}(\tilde{\triangle})$ that minimizes $E(s)$ over the set U. Then we call s a `minimal energy interpolating spline`.

Note that to compute the energy as defined in (10.2) we need to compute integrals over curved triangles. In practice, this would require quadrature formulae for integrating functions over curved triangles. Such formulae can be constructed (see Remark **??**). But numerical experiments show that the method works just as well if we compute energy using the ordinary triangulations in the inscribed triangulation $\triangle$ associated with $\tilde{\triangle}$.

As discussed in Sect.6.3 of [11], finding a minimal energy spline interpolant is just a matter of setting up and solving an appropriate linear system of equations. The Matlab package that goes with that book includes scripts to carry this out for ordinary triangulations. To make them work on curved triangulations we simply have to add commands to deal with evaluation on the the entire curved domain. This is easily done by building an extended triangulation. We now give several examples based on the space $\mathcal{S}_2^1(\tilde{\triangle}_{PS})$, where $\tilde{\triangle}_{PS}$ is a curved triangulation created by applying a Powell-Sabin split to a given initial curved triangulation.

Example 10.1. *Consider the function $f(x, y) = \sin(20x) + \sin(20y)$ on the domain shown in Fig. 5 (left). Approximate this function with minimal energy splines in the spaces $\mathcal{S}_2^1(\tilde{\triangle}_{PS})$ for a sequence of curved Powell-Sabin triangulations with decreasing mesh size.*

Discussion: We use Algorithm 11.1 to create curved triangulations with `ny = 5,9,17,33`. For each choice of `ny` the following table gives the total time to compute the spline, the number of triangles and number of coefficients and the max and RMS errors measure on a well distributed set of 57092 points spread out across Ω.

ny	time	nc	cond	emax	rms	rates	
5	0.02	294	639	3.49e+00	1.31e+00		
9	0.17	960	2005	8.45e-01	2.27e-01	2.04	2.53
17	0.46	3252	6653	1.16e-01	2.08e-02	2.87	3.45
33	5.98	11934	24151	2.54e-02	3.02e-03	2.19	2.78

As discussed in Sect.6.3.3 of [11], with minimal energy spline interpolation we can only expect second order convergence, which is what we are getting in this example. □

Here is an example involving a curved domain with a hole.

Example 10.2. *Consider the function $f(x, y) = \sin(20x) + \sin(20y)$ on the fifth domain shown in Fig. 3 (left). Approximate this function with minimal energy polynomial splines in the spaces $\mathcal{S}_2^1(\tilde{\triangle}_{PS})$ for a sequence of curved Powell-Sabin triangulations with decreasing mesh size.*

Discussion: The following table gives the same information as the previous one.

ny	time	nc	cond	emax	rms	rates	
5	0.02	228	504	3.98e+00	1.25e+00		
9	0.05	732	1544	3.97e-01	9.95e-02	3.33	3.65
17	0.45	2484	5112	5.99e-02	1.08e-02	2.73	3.21
33	3.70	8988	18244	1.12e-02	1.59e-03	2.42	2.76

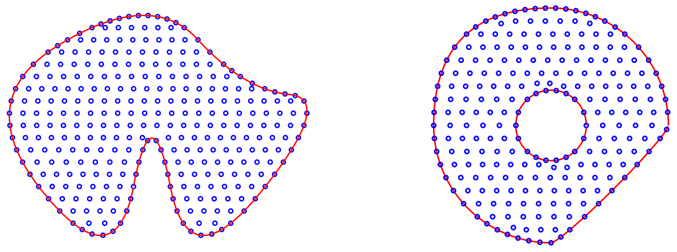

Figure 8: Points selected by Algorithm 11.1 with $n_y = 17$ for two curved domains

The results are quite similar to those in the previous example. We are again getting quadratic convergence. □

11 Picking well-spaced points in a curved domain

Suppose Ω is a curved planar domain with m holes whose boundary $\partial\Omega$ is defined by $m+1$ separate parametric curves $\partial\Omega_0, \ldots, \partial\Omega_m$. Let $a = \min\{y : (x, y) \in \Omega\}$ and $b = \max\{y : (x, y) \in \Omega\}$. In this section we give an outline of a simple algorithm that can be used to pick well-spaced points in Ω and on its boundaries.

Algorithm 11.1. *1. Pick a positive integer n_y and let $h = (b - a)/ny$. Start with empty sets $\mathcal{I}$ and $\mathcal{B}$.*

2. For $i = 1, \ldots, ny - 1$ let ℓ_i be the horizontal line $\{(x, y) : y = a + ih\}$.

3. Find the intersections of ℓ_i with $\partial\Omega$. This creates one or more line segments with endpoints on $\partial\Omega$. Add the endpoints to $\mathcal{B}$, and add points at spacing approximately h in the interior of each line segment to the set $\mathcal{I}$.

4. Add enough points to $\mathcal{B}$ (and adjust if necessary) to get nearly equally spaced points with spacing h on each of the boundary curves $\partial\Omega_0, \ldots, \partial\Omega_m$.

We have two applications in mind for the points produced by this algorithm: a) to construct an inscribed triangulation of Ω, b) to produce large sets of well-spaced points that cover Ω. We will use those for computing errors and plotting splines on curved triangulations. The parameter n_y controls the number of points selected and their spacing. Doubling the size of n_y generally reduces the mesh size of the corresponding constrained Delaunay triangulation by a factor of approximately 1/2. Our implementation can produce tens of thousands of points in a couple of seconds. Fig. 8 shows the points selected by Algorithm 11.1 for two curved domains whose boundaries are defined by NURBs curves. These points were generated with $n_y = 17$. But we get much denser sets of points if use larger values of n_y. In the examples above we took $n_y = 251$, which for the domains examined here gave between forty and fifty thousand points.

The spacing of the points produced by this algorthm can usually be improved by making minor adjustments to the location of some of the vertices, in particular those near the boundaries. This can be beneficial when the points are used to construct inscribed triangulations since we generally want to make sure that the smallest angle in the triangulation is not too small. Indeed, the adjustments can be accomplished by creating the constrained Delaunay triangulation, computing smallest angles for each triangle, and then using them to guide how to move the points.

12 Remarks

Remark 12.1. For recent work on the use of piecewise polynomials defined directly on curved triangulations in solving boundary-value problems on curved domains, see [3, 4, 5]. These papers deal with the restricted class of domains with piecewise conic boundaries, and use polynomials of different degrees on selected curved triangles. For references to older work, see [1, 2, 8, 10, 13], and references therein.

Remark 12.2. It should be emphasized that in this paper we are dealing only with domains Ω that lie in the plane. We are not allowing triangles that are curved in space such as those lying on the surface of a sphere.

Remark 12.3. Allowing the interior edges of a curved triangulation to be curved makes it very difficult to be able to locate which curved triangle in the triangulation contains a given point. This is why in practice we restrict ouselves to curved triangulations whose interior edges are straight lines.

Remark 12.4. Throughout this paper we are assuming the reader is familiar with the Bernstein–Bézier approach to dealing with piecewise polynomials on triangulations, see [7] and [11], so we don't bother to review this theory here.

Remark 12.5. The spaces of polynomial splines on curved triangulations considered in this paper are finite dimensional linear spaces. Following what was done in the literature for splines on ordinary triangulations, see [7], we can construct explicit bases for these spaces. However, it should be emphasized that none of the computational methods described here make use of any basis.

Remark 12.6. There is a particularly nice set of basis functions for the space $\mathcal{S}_d^0(\tilde{\triangle})$ introduced in Sect. 5. Suppose $\mathcal{D}_{d,\triangle}$ is the set of domain points of degree d for the companion triangulation $\triangle$ associated with $\tilde{\triangle}$. For each $\xi \in \mathcal{D}_{d,\triangle}$, let s_ξ be the spline in $\mathcal{S}_d^0(\tilde{\triangle})$ whose coefficients are all zero except for $c_\xi = 1$. Then just as for for the case of ordinary triangulations (see Sect. 5.4 of [7]), the splines $\{s_\xi\}_{\xi \in \mathcal{D}_{d,\tilde{\triangle}}}$ form a basis for $\mathcal{S}_d^0(\tilde{\triangle})$. These basis functions are nonnegative, form a partition of unity, and have local supports, namely a single curved triangle, two adjoining curved triangles, or the union of all curved triangles attached to a common vertex.

Remark 12.7. In Sect. 7 we discussed the approximation power of our spline spaces as measured in the max-norm. Following developments in the classical spline literature for ordinary splines, it is possible to extend the results to deal with the L_p norms.

Remark 12.8. The local two-stage scattered data interpolation methods discussed here make use of C^1 splines, but it is also possible to construct similar methods for curved triangulations using splines of higher smoothness. For example, we could easily extend the method based on C^2 splines on the so-called Wang refinement, see Sect. 6.7.4 of [11].

Remark 12.9. In Sect. 10 we illustrated the use of the minimal energy method for solving scattered data fitting problems using the space $\mathcal{S}_2^1(\tilde{\triangle}_{PS})$. But as shown in [11], the method can also be used with other macro-element spaces such as $\mathcal{S}_3^1(\tilde{\triangle}_{CT})$ or $\mathcal{S}_5^{1,2}(\tilde{\triangle})$.

Remark 12.10. Quadrature rules for computing integrals over ordinary triangles are discussed in Sect. 4.6 of [11]. For a curved triangle $\tilde{T}$, we can first approximate $\tilde{T}$ by a union of ordinary triangles or rectangles, and then use the standard quadature rules on each of the pieces. For example, rectangles are used in [9].

Remark 12.11. In Chapter 8 of [11] a variety of methods based on polynomial splines on ordinary triangulations were presented for fitting functions with various kinds of shape constraints. All of these methods can also be extended to deal with problems on curved domains by working with curved triangulations instead.

Remark 12.12. Polynomial splines on triangulations are also useful tools in solving data fitting problems, especially in the setting of noisy data, see Chapter 7 of [11]. All of the methods discussed there, including penalized least-squares methods, can also be extended to work with curved triangulations of curved domains.

References

[1] Bernardi, C.: Optimal finite-element interpolation of curved domains. SIAM J. Numer. Anal. **26** (1989), 1212 – 1240.

[2] Brenner, S., Scott, R.: *The Mathematical Theory of Finite Element Methods*. Springer (New York), 1994.

[3] Davydov, O., Kostin, G., Saeed, A.: Polynomial finite element method for domains enclosed by piecewise conics. Comput. Aided Geom. Design **45** (2016), 48 - 72.

[4] Davydov, O., Saeed, A.; C1 quintic splines on domains enclosed by piecewise conics and numerical solution of fully nonlinear elliptic equations. Appl. Numer. Math. **116** (2017), 172 – 183.

[5] Davydov, O., Yeo, W. P.: Approximation by C1 splines on piecewise conic domains. in Approximation theory XV: San Antonio 2016, Springer PROMS 201, 21 - 37.

[6] Feng, L., Alliez, P., Busé, L., Delingette, H.,Desbrun, M.: Curved Optimal Delaunay Triangulation, ACM Trans. Graph.**37**, 4, Article 61 (2018), 16pp.

[7] Lai, M. J., Schumaker, L. L.: *Spline Functions on Triangulations*, Cambridge University Press (Cambridge), 2007.

[8] Lenoir, M.: Optimal isoparametric finite elements and error estimates for domains involving curved boundaries. SIAM J. Numer. Anal. **23** (1986), 562 – 580.

[9] Rathoda1, H.T., Hariprasadb, H. S.,Vijayakumarb, K. V., Ratho, B. , Nagabhushanac, C. S.: Numerical integration over curved domains using convex quadrangulations and Gauss Legendre quadrature rules, Int. J. Engr. and Comp. Sci. **2**, (2013), 3290-3332

[10] Ruas, Vitoriano: Optimal Lagrange and Hermite finite elements for Dirichlet problems in curved domains with straight-edged triangles. Z. Angew. Math. Mech. **100** (2020), 28pp.

[11] Schumaker, L. L.: *Spline Functions: Computational Methods*, SIAM (Philadelphia), 2015.

[12] Schumaker, L. L.: Use of polynomial splines on curved triangulations to solve boundary-value problems on curved domains. In preparation.

[13] Scott, R.: Finite element techniques for curved boundaries. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1973.