Polyhedral Control-Net Splines for Analysis

special issue for Leszek Demkowicz

Bhaskar Mishra and Jörg Peters
University of Florida

Abstract

Generalizing tensor-product splines to smooth functions whose control nets can outline more general topological polyhedra, bi-cubic polyhedral-net splines form a first-order differentiable piecewise polynomial space. Each polyhedral control net node is associated with one bi-cubic function. A polyhedral control net admits grid-, star-, n-gon-, polar- and three types of T-junction configurations. Analogous to tensor-product splines, polyhedral-net splines can both model curved geometry and represent higher-order functions on the geometry. This paper explores the use of polyhedral-net splines for solving elliptic partial differential equations on curved smooth free-form surfaces without additional meshing.

1. Introduction

Discontinuous Galerkin (DG) [8] and Petrov–Galerkin (dPG) methods [13, 14] do not require differentiability between elements. This benefits stability, simplifies implementation, enables parallel execution, and even allows locally adapting test function spaces on the fly for high local approximation order. By contrast, the widely-used tensor-product spline [10] represents differentiable polynomial function spaces. Notably, splines have been used to generalize the iso-parametric approach to higher-order finite elements on curved smooth geometry, see [4, 43, 1, 41, 7, 17] – but only if the geometry can be outlined by a control net in the form of a tensor-product grid. At irregularities – where the tensor-structure breaks down (see the 3-valent and the 5-valent points in Fig. 1) – the B-spline and its control net are not well-defined. Irregularities are no problem for DG spaces. But when the underlying geometry needs to be both curved and smooth, the DG approach comes up short, and requires extra constraints, or penalty functions, or post-processing to guarantee a differentiable shape and function space. Such penalty methods require careful calibration to converge to the proper solution. Moreover, depending on the differential equation, a lack of differentiability

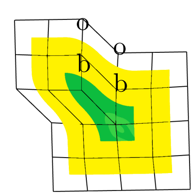
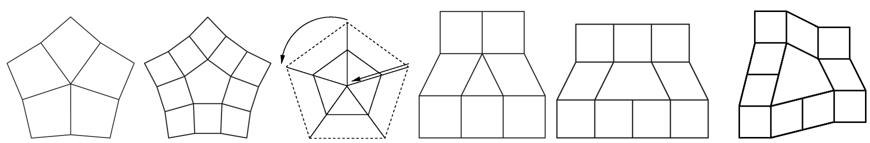


Figure 1: A polyhedral control net and polyhedral-net spline.

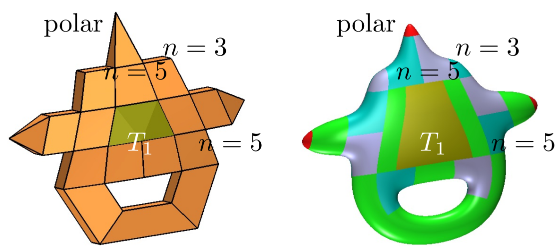
provides too large a computational space and may so yield outcomes that are not solutions to the original problem, e.g. non-physically discontinuous flow lines. Differentiability across irregularities is therefore both useful and a challenge when devising mathematical software.

Combining differentiability and flexibility, polyhedral-net splines [37] extend bi-quadratic (bi-2) tensor-product splines on regular, grid-like subnets to the non-tensor product sub-net configurations shown in Fig. 2.



(a) star-net (b) n-gon-net (c) polar-net (d) T_0-net (e) T_1-net (f) T_2-net

Figure 2: Six non-tensor-product polyhedral control net patterns. (a,b,c) The open source code library [37] covers star-, n-gon and plot configurations with $n, m \in \{3, 5, 6, 7, 8\}$ sectors.



(a) polyhedral control net (b) spline patch layout

Figure 3: The 'bottle opener' polyhedral-net spline from [37] that combines (a) non-tensor product patterns in close proximity (a 'tight layout') (b) The resulting polyhedral-net spline consisting of bi-2 splines, bi-3 polar caps, n-sided star-configurations (blue or gray), and surface pieces covering T_1-junctions. Fig. 9 shows the heat equation being solved on this challenge surface.

Analogous to a B-spline tensor-product control net, a polyhedral control net defines the overlap of the support of the polyhedral-net spline functions. The control net nodes provide handles for manipulating the shape, e.g. degrees of freedom for least squares fitting, or for solving partial differential equations, or for enforcing moments. The irregular polyhedral patterns listed in Fig. 2 can be placed in close proximity, enabling complex layouts such as Fig. 3. A polyhedral-net spline joins its bi-cubic (bi-3) pieces as a smooth piecewise polynomial function or surface expressed in Bernstein-Bézier form [12]. To introduce *creases or discontinuities,* one can manipulate individual bi-3 coefficients or insert fictitious facets with the geometry of a line segment. The splines can be refined using de Casteljau's algorithm [16] applied per piece, or, non-nestedly, by control net subdivision [6]. However, the sharp degree bounds proven in [28] rule out nested sss s based on¶ control-net refinement.

Overview. After brief overview of alternative smooth polynomial function spaces, Section 2 reviews tensor-product splines, polyhedral control nets and polyhedral-net splines. Section 3 extends the implementation of polyhedral-net splines [37] to solve basic partial differential equations on non-tensor-product meshes – without additional meshing.

1.1. Alternative geometric functions spaces

Commonly used computational polynomial spaces for unstructured layout on planar domains include splines on triangulations [29] and radial basis functions [5]. For modeling complex geometric free-form shapes, tensor-product spline (NURBS) domains are carved up into complex regions by a restriction of the domain known as *trimming* (see e.g. [30] in the context of isogeometric design). Trimming causes heterogeneity in continuity, size, and parameter orientation, gaps in the geometry and special integration rules for the complex domains. The animation industry has instead adopted subdivision surfaces [15] that consist in theory of an infinite sequence of nested surface rings, in practice approximated by a fine faceted model. The survey of splines for irregularities on irregular meshes [36] characterizes subdivision surfaces as one of three singular surface constructions: singularities at corners [34, 40, 33, 47], singular edges [31, 45] and contracting faces, a.k.a. subdivision algorithms [38]. Splines leveraging geometric continuity, i.e. differentiable after a local change of variables include [35, 32, 9, 3] and for data fitting and simulation on planar domains [2, 19, 21, 20]. Analogous to the bi-2 generalizing polyhedral-net splines in this paper, extending bi-3 tensor-product splines to polyhedral control nets and surfaces of good shape is possible using piecewise polynomials of degree bi-4 or higher, see e.g. [22, 25].

2. Tensor-product splines, polyhedral control nets and polyhedral-net splines

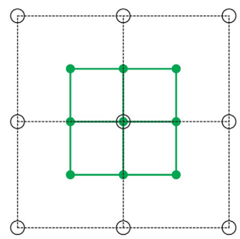
This section defines and summarizes the evolution of splines on regular tensor-product grids, Section 2.1, to the polyhedral control nets, Section 2.2, of polyhedral-net splines, Section 2.3.

2.1. Tensor-Product Splines

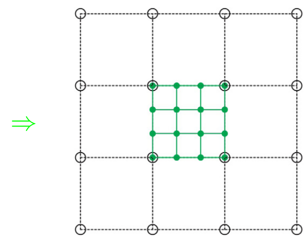
A tensor-product spline in two variables, (u, v) is a piecewise polynomial function of the form [10] :

$$\mathbf{p} : (u, v) \to \mathbf{p}(u, v) := \sum_{i=0}^{k_1} \sum_{j=0}^{k_2} \mathbf{c}_{ij} \psi_i^{d_1}(u) \psi_j^{d_2}(v). \tag{1}$$

The coefficients $\mathbf{c}_{ij}$ of the B-splines $\psi_i^{d_\ell}$ of degree d_ℓ are degrees of freedom. Designers can edit the x, y, z coordinates of $\mathbf{c}_{ij}$ to shape a surface while automatically maintaining differentiability. A fourth coordinate can serve to define the solution of a finite element computation on that geometry. Connecting the *control points* $\mathbf{c}_{ij}$ to $\mathbf{c}_{i+1,j}$ and $\mathbf{c}_{i,j+1}$ wherever possible yields the *control-net*



(a) bi-2 spline nets



(b) bi-3 spline nets

Figure 4: (a) The $(d+1)^2$ B-spline control points $\circ$ form a control net [10]. BB-coefficients $\bullet$ form a BB-net of a polynomial piece of degree bi-d [12]. (b) The $\Rightarrow$ indicates degree-raising of the BB-form and the corresponding B-spline control net.

that outlines the shape (see the convex hull property and the variation diminishing property, [10]).

The regular case of polyhedral-net splines are tensor-products of B-splines $\psi^d(t-k)$ of degree $d=2$ with a *uniform knot sequence* [12]; $k_1 \times k_2$ control net grids with quadrilateral faces define bi-quadratic (bi-2) polynomial pieces:

$$\mathbf{p} : (u,v) \to \mathbf{p}(u,v) := \sum_{i=0}^{2} \sum_{j=0}^{2} \mathbf{c}_{ij} \psi^2(u-i)\psi^2(v-j). \tag{2}$$

(For uniformity of degree with the irregular configurations, these can be re-represented in degree bi-3 form, see Fig. 4).

2.2. Patterns in polyhedral control nets

Fig. 2 lists irregular sub-nets of a polyhedral control net and Table 1 succinctly summarizes the structure of the *polyhedral layout configurations* and the resulting polyhedral-net splines. Note that for almost all meshes a single Catmull-Clark [6] step guarantees a quad mesh with star configurations only, i.e. a net that controls a polyhedral-net spline surface.

In polyhedral 3D modeling meshes with quad-strips are commonly used to outline a principal curves on the surface. Merging n such directions forms either an n-gon, see Fig. 2b for $n=5$, or a star-net with a central extraordinary point, see Fig. 2a for valence $n=5$, or a polar configuration, Fig. 2c. Star configuration may overlap, and their quadrilateral faces can have multiple non-4-valent vertices to form a 'tight layout' as in Fig. 3. A polar configuration consists of triangles that are quadrilaterals with one of the edges collapsed and that join with a removable singularity at the pole node. Polar configurations are used to cap off cylinders as when modeling airplane nose cones or finger tips.

T-joints model the transition from two finer quadrilaterals to one a coarser quadrilateral to adjust for density. A T_1-gon adjusts in one direction, T_2-gon has a T-junction in either parameter direction. The T_0-gon merges adjacent quad-strips in one direction without an explicit T-junction.

Table 1: Configurations of polyhedral control nets and polyhedral-net splines. All patches are in BB-form of degree bi-3. A restriction to $n \in \{3, 5, \ldots, 8\}$ is only in the distributed code [37]; the underlying theory allows for higher n.

name	configuration center	surrounded by	Fig.	# patches	ref
tensor	4-valent point	4 quads	4b	1	[12]
star	n-valent	n	2a	n for $3, 5$ $4n$ for $n > 5$	[24]
n-gon	n-gon	$2n$	2b	n for $3, 5$ $4n$ for $n > 5$	[24]
T_0	triangle†	7	2d	2×2	[26]
T_1	pentagon††	9	2e	4×2	[26]
T_2	hexagon†††	9	2f	4×4	[26]
polar	n-valent	n triangles	2c	n degenerate	[27]

† two vertices of valence 4 and one of valence 5. †† four vertices of valence 4 and one of valence 3
††† three consecutive vertices of valence 4 and two of valence 3 separated by one vertex of valence 4

2.3. Piecewise polynomial polyhedral-net splines

A polyhedral-net spline consists of smoothly-joined polynomial pieces in Bernstein-Bézier form (BB-form, [12, 16])

$$\mathbf{p}(u,v) := \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} \mathbf{b}_{ij} b_i^{d_1}(u) b_j^{d_2}(v), \quad (u,v) \in [0..1]^2 \tag{3}$$

where $\mathbf{b}_{ij}$ are the BB-coefficients and $b_k^d(t) := \binom{d}{k}(1-t)^{d-k}t^k \in \mathbb{R}$ are the Bernstein polynomials of degree d. Connecting $\mathbf{b}_{ij}$ to $\mathbf{b}_{i+1,j}$ and $\mathbf{b}_{i,j+1}$ wherever possible yields the *BB-net*. When $\mathbf{b}_{ij} \in \mathbb{R}^3$ and $\mathbf{p}$ is a piece of the surface, it is called a *patch*. A bi-d patch has $d+1 \times d+1$ BB-coefficients, see Fig. 4b. The BB-net defines a *single* polynomial piece whereas the B-spline control net represents a *piecewise* polynomial function. (The two bases can be converted into another by knot insertion, respectively knot removal [11, 12]). The BB-form can be evaluated using de Casteljau's algorithm [46], differentiated by forming differences $\mathbf{b}_{i+1,j} - \mathbf{b}_{ij}$, respectively $\mathbf{b}_{i,j+1} - \mathbf{b}_{ij}$, and integrated by summing the BB-coefficients [16]. A patch of degree less than bi-3 can be expressed in degree bi-3 form by 'degree-raising' [12, 16]. The polyhedral-net splines used in this paper are therefore assumed to be of uniform degree bi-3.

By default, two polyhedral-net spline pieces $\mathbf{p}$ and $\mathbf{q}$ abutting along a shared boundary E join with *matching first derivatives after a change of variables* – as is appropriate for surfaces. Denote the change of variables along E as $\beta : \mathbb{R}^2 \to \mathbb{R}$, $\beta(u) := (u + b(u)v, a(u)v)$ so that $\mathbf{p}(u,0) = E = \mathbf{q}(u,0)$, see Fig. 5. Then $\mathbf{q}$ and $\mathbf{p}$ join G^1 if the partial derivatives satisfy the polynomial equation or syzygy (∂ precedes evaluation)

$$\partial_v \mathbf{q}(u,0) + a(u)\partial_v \mathbf{p}(u,0) = b(u)\partial_u \mathbf{p}(u,0), \quad a(u) \neq 0, \; u \in [0..1]. \tag{4}$$

In other words, the partial derivatives $\partial_v \mathbf{q}(u,0)$ and $\partial_u \mathbf{q}(u,0) = \partial_u \mathbf{p}(u,0)$ along $E = \mathbf{q}(u,0)$ are a linear combination, with weights $a(u)$ and $b(u)$, of $\partial_v \mathbf{p}(u,0)$ and $\partial_u \mathbf{p}(u,0)$, and therefore lie in the tangent plane spanned by $\partial_v \mathbf{p}(u,0)$ and $\partial_u \mathbf{p}(u,0)$. Tensor-product and the polar variant of polyhedral-net splines are internally C^1 and join parameterically C^1 due to the choice $b(u) := 0$ and $a(u) := const$. Otherwise, the spline pieces join geometrically smooth, short G^1. For example, [24] relies on a quadratic β to join the surrounding surface to the n bi-3 patches of the multi-patch cap and uses a linear β internal to the cap. Remarkably, [24], [26], [27] derive the explicit formulas of the off-hand non-linear polynomial relation (4) and express the BB-coefficients of the polyhedral-net spline in terms of local neighborhoods of the input polyhedral control net. This is possible only with the help of extensive symbolic computation and due to the fact that a specific, careful choice of β linearizes and localizes (4) as an equation in terms of the BB-coefficients of $\mathbf{p}$ and $\mathbf{q}$. We observe that, as is typical for not-a-knot splines, at global boundaries the outermost layer recedes as in Fig. 1.

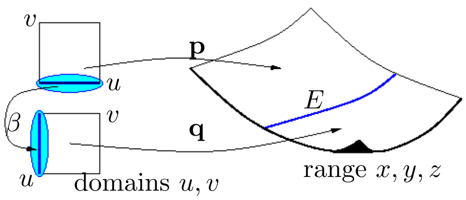


Figure 5: Patches $\mathbf{p}$ and $\mathbf{q}$ joining with change of variables (reparameterization by β) along an edge E.

3. Computing with polyhedral-net splines

Evidently polyhedral-net splines support many applications including industrial design, visualization, animation, moment computation, re-approximation, reconstruction, etc.. Here we show their usefulness for computing solutions of partial differential equations on manifolds. The appeal of polyhedral-net spline is that the applications are supported by the same representation as the geometry, without additional meshing. The irregular patterns can be viewed as both structurally necessary but also as a form of local adaptation.

First, in Section 3.1, 3.2 and 3.3, we test the new elements by artificially introducing irregular polyhedral patterns into an otherwise regular grid to be able to compare with a known exact solution. The experiments indicate that the irregular patterns in polyhedral-net splines do not noticeably increase the local error. Second, in Section 3.4, we compute second-order equations on a free-form surface modeled by polyhedral-net splines.

3.1. Poisson's Equation on a polyhedral-net spline parameterized physical domain

We assess the impact on the error of inserting irregular mesh patterns into a regular grid and solving Poisson's equation

$$\text{find } \mathtt{u} : \Omega \to \mathbb{R} : \begin{cases} \Delta \mathtt{u} & = -f \quad \text{in } \Omega \in \mathbb{R}^2, \\ \mathtt{u} & = 0 \qquad \text{on } \partial\Omega, \end{cases} \tag{5}$$

over a physical domain Ω parameterized by $\mathbf{x} :== \sum_i \mathbf{x}_i \phi_i$, i.e. polyhedral-net splines ϕ_i with control points $\mathbf{x}_i$. The weak Galerkin form of Poisson's equation on the physical domain Ω (pulled back to the domain of ϕ_j by $\mathbf{x}^{-1}$) projected into the space of polyhedral-net spline functions ϕ_j is

$$\int_\Omega \nabla \mathtt{u}_h \cdot \nabla \phi_i(\mathbf{x}^{-1}) d_\Omega = \int_\Omega f \phi_i(\mathbf{x}^{-1}) d_\Omega, \quad \mathtt{u}_h := \sum_j \mathbf{c}_j \phi_j(\mathbf{x}^{-1}).$$

The equation can therefore be rewritten as a matrix equation $K\mathbf{c} = \mathbf{f}$ to be solved for the coefficient vector $\mathbf{c}$ of $\mathtt{u}_h$ where

$$K_{ij} := \int_\Omega \nabla \phi_i(\mathbf{x}^{-1}) \cdot \nabla \phi_j(\mathbf{x}^{-1}) d_\Omega, \quad \mathbf{f}_i := \int_\Omega f \phi_i(\mathbf{x}^{-1}) d_\Omega. \tag{6}$$

Since each polyhedral-net spline ϕ_i is in piecewise BB-form, the domain of each ϕ_i consists of a collection of unit squares $\square := [-1..1]^2$. That is, the pre-image of Ω, parameterized by $\sum_i \mathbf{x}_i \phi_i$, partitions into a sum of unit squares that we enumerate with the label α. By change of variables to $\square$, with the gradient $J_\alpha := \nabla_{\mathbf{s}} \mathbf{x}_\alpha = \begin{bmatrix} \partial_s x & \partial_t x \\ \partial_s y & \partial_t y \end{bmatrix} \in \mathbb{R}^{2\times 2}$ and $\mathtt{J} := \sqrt{\det(J_\alpha^{\mathtt{t}} J_\alpha)} \in \mathbb{R}$, (6) becomes, cf. [32], p. 153:

$$K_{ij} := \sum_\alpha \int_\square (\nabla \phi_i)^{\mathtt{t}} (J_\alpha^{\mathtt{t}} J_\alpha)^{-1} (\nabla \phi_j) \ \mathtt{J} \, \mathrm{d}\square, \quad \mathbf{f}_i := \sum_\alpha \int_\square f \phi^i \ \mathtt{J} \, \mathrm{d}\square. \tag{7}$$

Here the sum is over all pieces α where ϕ_i has support.

3.2. Implementation Details

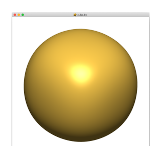
The code [37] collects the seven sub-net configurations from the input faceted free-form topological polyhedron (in .obj format [44], see Fig. 6a) and generates the BB-coefficients of the pieces of a smooth spline that can be visualized by the online WebGL viewer at [39], see Fig. 6b. Each x, y, z coordinate is a polyhedral-net spline. We intercept the output to compute the geometry terms $(J_\alpha^{\mathtt{t}} J_\alpha)^{-1}$ and $\mathtt{J}$ and access the connectivity information to determine where basis functions ϕ_i overlap in support to minimize the computational work in assembling K and $\mathbf{f}$. Exact derivative polynomials are obtained by differencing BB-coefficients, e.g. $\mathbf{b}_{i+1,j} - \mathbf{b}_{i,j}$ for differentiation with respect to the first variable. For Gauss quadrature on the unit square domain, for each α, all functions in (7) are evaluated at Gauss points mapped linearly to the interval [0..1] that is natural for the BB-form. Assembly is therefore no more costly than for tensor-product splines.

```
v 1.0 1.0 -1.0
v 1.0 -1.0 -1.0
v 1.0 1.0 1.0
v 1.0 -1.0 1.0
v -1.0 1.0 -1.0
v -1.0 -1.0 -1.0
v -1.0 1.0 1.0
v -1.0 -1.0 1.0
```

```
f 1 5 7 3
f 4 3 7 8
f 8 7 5 6
f 6 2 4 8
f 2 1 3 4
f 6 5 1 2
```

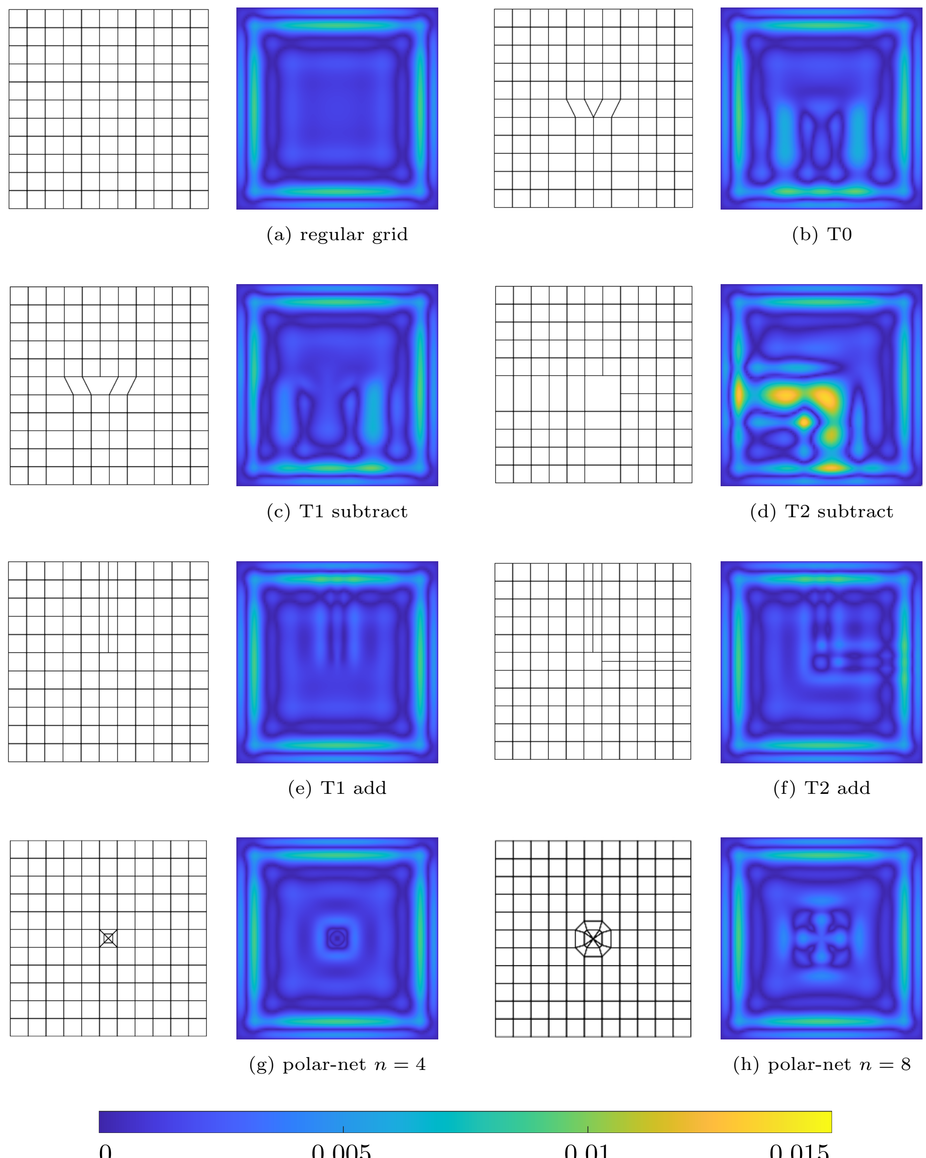
$\Longrightarrow$

(a) Input: cube.obj



(b) surface

Figure 6: [37] The (a) complete input .obj formatted data and (b) the output polyhedral-net spline surface visualized in Bézierview [39].



(a) regular grid (b) T0 (c) T1 subtract (d) T2 subtract (e) T1 add (f) T2 add (g) polar-net $n = 4$ (h) polar-net $n = 8$

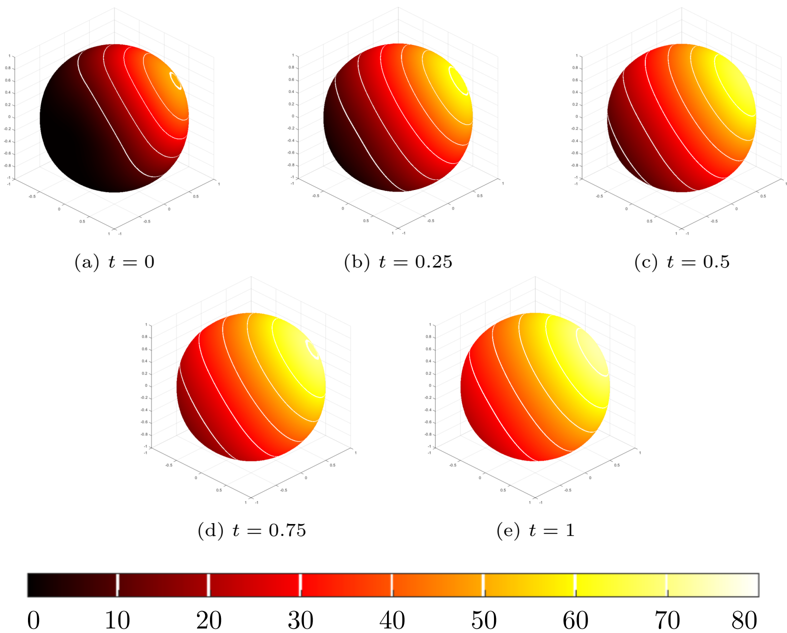
Figure 7: Comparison to regular splines by artificially introducing polyhedral control-net configurations. The color measures the absolute error between the approximate and the exact solution $\mathtt{u} = (u^2 - 1)(v^2 - 1)$ on the domain $[-1..1]^2$.

We enforce the boundary conditions $\mathtt{u}_h(\partial\Omega) = 0$, see Fig. 7, by modifying the polyhedral-net splines ϕ_j that overlap the boundary. Recall that the ϕ_j are bi-quadratic (bi-2) splines in the regular regions that form the boundary. The only bi-2 splines with support straddling the boundary are therefore those whose central Greville abscissa (the midpoint of the support) just outside the boundary (two such spline coefficients are labeled o in Fig. 1) or just inside, labeled b in Fig. 1. The BB-coefficients of type o splines are zero in the interior, those of type b are zero outside. Setting the BB-coefficients on the boundary to zero to satisfy $\mathtt{u}_h(\partial\Omega) = 0$ implies that type o splines make no contribution to the equation and can be ignored. For type b, we only set the BB-coefficients of their bi-3 representation to zero on the boundary to retain maximal approximation order. In the regular case displayed in Fig. 7a, the symmetric matrix K has $n = 10 \times 10$ columns.

3.3. No intrinsically increased error over bi-2 splines

By artificially introducing¶ polyhedral control-net configurations¶ into the regular grid on a square, Fig. 7 illustrates the *absence of intrinsically increased error* for the various polyhedral-net spline configurations, compared to bi-quadratic tensor-product splines. In that sense, the polyhedral-net spline framework exacts no cost for the ability to handle irregular configurations. To accurately measure the error and gauge the impact of the irregular polyhedral control net configurations, we choose a trivial geometric (physical surface) domain parameterization $x(u, v) = u$, $y(u, v) = v$, $z(u, v) = 0$ and cover $[-1..1]^2$. That is, we compute the graph of a function. We will see that computations on more complex free-form surfaces are no more difficult in the implemented framework (see Fig. 9). Solutions on more complex domains are of course harder to verify due to the lack of explicit solutions for free-form surfaces. We choose $f := 2(u^2+v^2-2)$ so that we know the exact solution of (5) to be $\mathtt{u} = (u^2 - 1)(v^2 - 1)$.

Starting with the tensor grid on $[-1..1]^2$ as a basic test, we solve and display the absolute difference between the polyhedral-net splines and the exact solution of (5) in Fig. 7. Although the exact solution is bi-quadratic, enforcing the boundary constraints with fixed resolution results in an error along and near the boundary, of less than 1%, see Fig. 7a. Note the error bar below the plots in Fig. 7 that is common to all plots. Fig. 7c and Fig. 7d are obtained by removing grid-lines from Fig. 7a, whereas Fig. 7e and Fig. 7f are obtained by adding grid-lines. The location and disappearance of the subtractive case maximal errors shows that the error is dominated by the reduction in the degrees of freedom when eliminating half a column and half a row of control points, and not due to distortion near the T-configuration. Fig. 7c, 7d, 7e and 7f remind of T-splines. However, T-splines [42, 18] start with quadrilaterals and require a global parameterization and then insert partitions and T-junctions. In general, the input free-form control nets cannot already include T-junctions [23, Fig 18]. By contrast, polyhedral-net splines accepted input T-configurations. Fig. 7g and Fig. 7h show polar configurations . There is a decrease in error near the pole and no noteworthy increase in error in the neighborhood.



(a) $t = 0$ (b) $t = 0.25$ (c) $t = 0.5$

(d) $t = 0.75$ (e) $t = 1$

Figure 8: Progression of geodesic fronts over time t when solving the heat equation over the polyhedral spline with input specified in Fig. 6. The temperature at the upper right vertex, the heat source, is kept constant to $80°$. Only eight polyhedral-net spline coefficients are computed per time step.

We do not show adaptive convergence rates under refinement since polyhedral-net splines have local support and the examples establish the relation to the error of bi-2 tensor-product splines on the regular mesh. Due to the local support, any error due to the parameterization patterns does not spread out to the regular part. Therefore, in the 2-norm, the error by splitting quadrilaterals into 2×2 or by applying Catmull-Clark subdivision to the polyhedral control net, is dominated by the regular mesh. The condition numbers are of the same magnitude as for bi-2 splines on the regular partition. We note in passing that, for adaptive refinement, we can use a variant of splines with 'hanging nodes', see e.g. [18].

3.4. Solving elliptic differential equations on free-form surfaces

Having verified correctness of the implementation on the a square domain for all configurations, we solve the heat equation on polyhedral-net spline free-form surfaces, i.e. a second-order elliptic partial differential equation with a solution evolving over time. Since our focus is on complex surface geometry, we do not consider more complicated, say weighted fourth order heat equations as are addressed in the single variable function case in [48]. Compared to the classical bi-variate heat equation with heat source f, $\partial_t u = \Delta_{\mathbf{x}} u + f$ on the plane, the challenge is that $\Delta_{\mathbf{x}}$ is now the surface-dependent Laplace-Beltrami

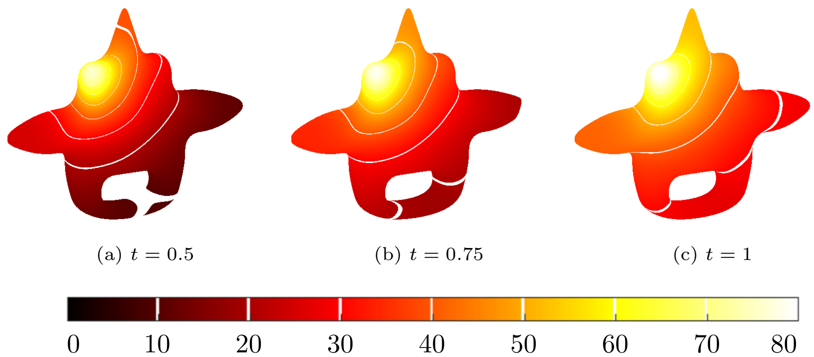


Figure 9: Progression of geodesic fronts for the polyhedral-net spline with ‘bottle opener’ input specified in Fig. 3 exposed to Florida sun at the upper left corner. (a) A low gradient causes a wide white level indicator curve at the lower right. The surface has no gaps and is defined by 59 control points.The heat computation determines 58 temperature coefficients since one is set.

operator. That is, the operator is treated as in (7), with the geometry entering via the first fundamental form $(J_\alpha^{\mathrm{t}} J_\alpha) : \mathbb{R}^{2\times 3}, \mathbb{R}^{3\times 2} \to \mathbb{R}^{2\times 2}$ of the surface and its determinant $\mathtt{J}$. The time step is implemented explicitly as a forward difference and the equation at each time step is solved with the *eigen* package. Since the examples are closed free-form surfaces, no boundary conditions are required.

To certify correctness of the implementation, we start with the simple cube (8 nodes, 6 faces) whose data are given in Fig. 6. We seed the temperature at one of the 8 vertices of the cube and observe the progress of the heat level curves over time. Only eight polyhedral-net spline temperature coefficients need to be computed per time step. This yields a progression of geodesic fronts, as illustrated by the white lines in Fig. 8. Next, we test the solver on the complex and tight ensemble of polyhedral configurations, nicknamed ‘bottle opener’, that was shown in Fig. 3. We compute the heat progression and show a heat distribution time series in Fig. 9 ‘until the opener becomes too hot to handle’.

4. Conclusion

Analogous to tensor-product splines, but for more general control net configurations, polyhedral-net splines have been shown to model curved geometry (without trimming) and to represent higher-order functions on that geometry. This setup has potential to be used for analysis on curved smooth objects – *without additional meshing*. As a proof-of-concept, we derived time-evolving solutions of an elliptic partial differential equation. Since polyhedral-net splines are differentiable, also fourth-order equation Galerkin solvers make sense and can be implemented.

Acknowledgement We thank William Gregory for support with C++ programming and Kyle Lo for generating input models and explaining data structures of the polyhedral-net spline code.

References

[1] F.T.K. Au and Y.K. Cheung. 1993. Isoparametric Spline Finite Strip for Plane Structures. *Computers & Structures* 48 (1993), 22–32.

[2] Michel Bercovier and Tanya Matskewich. 2017. *Smooth Bézier surfaces over unstructured quadrilateral meshes.* Vol. 22. Springer International Publishing, Berlin, New York.

[3] Ahmed Blidia, Bernard Mourrain, and Gang Xu. 2020. Geometrically smooth spline bases for data fitting and simulation. *Computer Aided Geometric Design* 78 (March 2020), 101814.

[4] V. Braibant and C. Fleury. 1984. Shape Optimal Design using B-splines. *Computer Methods in Applied Mechanics and Engineering* 44 (1984), 247–267.

[5] Martin D. Buhmann. 2009. *Radial Basis Functions - Theory and Implementations.* Cambridge monographs on applied and computational mathematics, Vol. 12. Cambridge University Press. I–X, 1–259 pages. `http://www.cambridge.org/de/academic/subjects/mathematics/numerical-analysis/radial-basis-functions-theory-and-implementations`

[6] E. Catmull and J. Clark. 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* 10 (Sept. 1978), 350–355.

[7] F. Cirak, M. Ortiz, and P. Schröder. 2000. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *Internat. J. Numer. Methods Engrg.* 47 (April 2000), 2039–2072.

[8] B. (Bernardo) Cockburn, George Karniadakis, and Chi-Wang Shu. 2000. *Discontinuous Galerkin methods: theory, computation, and applications.* Vol. 11. Springer-Verlag Inc., pub-SV:adr. xi + 470 pages.

[9] Annabelle Collin, Giancarlo Sangalli, and Thomas Takacs. 2016. Analysis-suitable G1 multi-patch parametrizations for C1 isogeometric spaces. *Computer Aided Geometric Design* 47 (2016), 93–113.

[10] C. de Boor. 1978. *A Practical Guide to Splines.* Springer.

[11] Carl de Boor. 1986. *B (asic)-Spline Basics.* Technical Report. U of Wisconsin, Mathematics Research Center.

[12] C. de Boor. 1987. B-form basics. In *Geometric Modeling: Algorithms and New Trends*, G. Farin (Ed.). SIAM, 131–148.

[13] Leszek Demkowicz and Jayadeep Gopalakrishnan. 2010. A class of discontinuous Petrov–Galerkin methods. Part I: The transport equation. *Computer Methods in Applied Mechanics and Engineering* 199, 23-24 (2010), 1558–1572.

[14] Leszek Demkowicz and Jayadeep Gopalakrishnan. 2011. A class of discontinuous Petrov–Galerkin methods. II. Optimal test functions. *Numerical Methods for Partial Differential Equations* 27, 1 (2011), 70–105.

[15] Tony DeRose, Michael Kass, and Tien Truong. 1998. Subdivision Surfaces in Character Animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM Press, New York, 85–94.

[16] Gerald Farin. 1988. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press.

[17] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. 2005. Isogeometric Analysis: CAD, Finite Elements, NURBS, Exact Geometry and Mesh Refinement. *Computer Methods in Applied Mechanics and Engineering* 194 (2005), 4135–4195.

[18] Hongmei Kang, Jinlan Xu, Falai Chen, and Jiansong Deng. 2015. A new basis for PHT-splines. *Graphical Models* 82 (2015), 149–159.

[19] Mario Kapl, Giancarlo Sangalli, and Thomas Takacs. 2018. Construction of analysis-suitable G1 planar multi-patch parameterizations. *Computer-Aided Design* 97 (2018), 41–55.

[20] Mario Kapl, Giancarlo Sangalli, and Thomas Takacs. 2019. Isogeometric analysis with C^1 functions on planar, unstructured quadrilateral meshes. *The SMAI journal of computational mathematics* (2019), 67–86.

[21] Mario Kapl, Giancarlo Sangalli, and Thomas Takacs. 2019. An isogeometric C ^{1} subspace on unstructured multi-patch planar domains. *Computer Aided Geometric Design* 69 (2019), 55–75.

[22] Kęstutis Karčiauskas, Thien Nguyen, and Jörg Peters. 2016. Generalizing bicubic splines for modeling and IGA with irregular layout. *Computer-Aided Design* 70 (2016), 23–35.

[23] Kęstutis Karčiauskas, Daniele Panozzo, and Jörg Peters. 2017. T-junctions in spline surfaces. *ACM Transactions on Graphics (TOG)* 36, 5 (2017), 1–9.

[24] Kęstutis Karčiauskas and Jörg Peters. 2015. Smooth multi-sided blending of biquadratic splines. *Computers & Graphics* 46 (2015), 172–185.

[25] Kęstutis Karčiauskas and Jörg Peters. 2019. High quality refinable G-splines for locally quad-dominant meshes with T-gons. *Computer Graphics Forum* 38, 5 (2019), 151–161.

[26] Kęstutis Karčiauskas and Jörg Peters. 2020. Low degree splines for locally quad-dominant meshes. *Computer Aided Geometric Design* 83 (2020), 1–12.

[27] Kęstutis Karčiauskas and Jörg Peters. 2020. Smooth polar caps for locally quad-dominant meshes. *Computer Aided Geometric Design* 81 (06 2020), 1–12. `https://doi.org/10.1016/j.cagd.2020.101908` PMC7343232.

[28] Kęstutis Karčiauskas and Jörg Peters. 2021. Least Degree G^1-refinable Multi-sided Surfaces Suitable for Inclusion into C^1 bi-2 Splines. *Computer-Aided Design* 130 (2021), 1–12.

[29] Ming-Jun Lai and Larry L. Schumaker. 2007. *Spline Functions on Triangulations.* Cambridge University Press, Cambridge, UK. `https://doi.org/10.1017/CBO9780511721588`

[30] B Marussig and TJR Hughes. 2018. A Review of Trimming in Isogeometric Analysis: Challenges, Data Exchange and Simulation Aspects. *Arch Comput Methods Eng.* 25, 4 (2018), 1059–1127. Erratum in: Arch Comput Methods Eng. 2018;25(4):1131..

[31] Ashish Myles and Jörg Peters. 2011. C^2 splines covering polar configurations. *Computer-Aided Design* 43, 11 (2011), 1322–1329.

[32] Thien Nguyen, Kęstutis Karčiauskas, and Jörg Peters. 2016. C^1 finite elements on non-tensor-product 2d and 3d manifolds. *Appl. Math. Comput.* 272, 1 (2016), 148–158.

[33] Thien Nguyen and Jörg Peters. 2016. Refinable C^1 spline elements for irregular quad layout. *Computer Aided Geometric Design* 43 (March 29 2016), 123–130.

[34] J. Peters. 1991. Smooth interpolation of a mesh of curves. *Constructive Approximation* 7 (1991), 221–247.

[35] J. Peters. 1995. C^1-surface splines. *SIAM J. Numer. Anal.* 32, 2 (1995), 645–666.

[36] J. Peters. 2019. Splines for Meshes with Irregularities. *The SMAI journal of computational mathematics* S5 (2019), 161–183.

[37] Jörg Peters, Kyle Lo, and Kęstutis Karčiauskas. 2023. Algorithm 1032: Bi-cubic splines for polyhedral control nets. *ACM Tr on Math Software* 49 (March 2023). Issue 1.

[38] J. Peters and U. Reif. 2008. *Subdivision Surfaces.* Geometry and Computing, Vol. 3. Springer-Verlag, New York. i–204 pages.

[39] J. Peters and X. Wu. 2021. BezierView (short bview): a light weight viewer that renders Bezier patches. https://www.cise.ufl.edu/research/SurfLab/bview

[40] Ulrich Reif. 1998. TURBS—Topologically Unrestricted Rational B-Splines. *Constructive Approximation* 14 (1998), 57–77.

[41] Uwe Schramm and Walter D. Pilkey. 1993. The coupling of geometric descriptions and finite elements using NURBS - A study in shape optimization. *Finite elements in Analysis and Design* 340 (1993), 11–34.

[42] Thomas W Sederberg, Jianmin Zheng, Almaz Bakenov, and Ahmad Nasri. 2003. T-splines and T-NURCCs. *ACM transactions on graphics (TOG)* 22, 3 (2003), 477–484.

[43] Y.K. Shyy, C. Fleury, and K. Izadpanah. 1988. Shape Optimal Design using higher-order elements. *Computer Methods in Applied Mechanics and Engineering* 71 (1988), 99–116.

[44] Wavefront Technologies. 2021. Wavefront .obj file. https://en.wikipedia.org/wiki/Wavefront_.obj_file

[45] D Toshniwal, H Speleers, R R Hiemstra, and TJR Hughes. 2017. Multi-degree smooth polar splines: A framework for geometric modeling and isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* 316 (2017), 1005–1061.

[46] Wikipedia contributors. 2021. De Casteljau's algorithm — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=De_Casteljau%27s_algorithm&oldid=1043959654 [Online; accessed 1-December-2021].

[47] Meng Wu, Bernard Mourrain, André Galligo, and Boniface Nkonga. 2017. Hermite Type Spline Spaces over Rectangular Meshes with Complex Topological Structures. *Communications in Computational Physics* 21, 3 (2017), 835–866.

[48] Xuehua Yang, Lijiao Wu, and Haixiang Zhang. 2023. A space-time spectral order sinc-collocation method for the fourth-order nonlocal heat model arising in viscoelasticity. *Appl. Math. Comput.* 457 (2023), 128192.