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 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 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. Differentia-
bility 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.

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.

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 and its
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 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]. The sharp degree bounds proven in [28] rule out nested polyhedral control
net-based refinement (a subdivision algorithm) for bi-3 polyhedral-net splines.

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] :

\[
p : (u, v) \rightarrow p(u, v) := \sum_{i=0}^{k_1} \sum_{j=0}^{k_2} c_{ij} \psi_i^{d_1}(u) \psi_j^{d_2}(v).
\]

The coefficients \(c_{ij}\) of the B-splines \(\psi_i^{d_1}\) of degree \(d_1\) are degrees of freedom. Designers can edit the \(x, y, z\) coordinates of \(c_{ij}\) 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 \(c_{ij}\) to \(c_{i+1,j}\) and \(c_{i,j+1}\) wherever possible yields the 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:

\[
p : (u, v) \rightarrow p(u, v) := \sum_{i=0}^2 \sum_{j=0}^2 c_{ij} \psi^2(u - i) \psi^2(v - j).
\]

(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 succintly 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 \( n \).

<table>
<thead>
<tr>
<th>configuration</th>
<th>center</th>
<th>surrounded by</th>
<th>Fig.</th>
<th># patches</th>
<th>ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>tensor</td>
<td>4-valent point</td>
<td>4 quads</td>
<td>4b</td>
<td>1</td>
<td>[12]</td>
</tr>
<tr>
<td>star</td>
<td>( n )-valent</td>
<td>( n )</td>
<td>2a</td>
<td>( n ) for 3, 5</td>
<td>[24]</td>
</tr>
<tr>
<td>( n )-gon</td>
<td>( n )-gon</td>
<td>2n</td>
<td>2b</td>
<td>( n ) for 3, 5</td>
<td>[24]</td>
</tr>
<tr>
<td>( T_0 )</td>
<td>triangle(^{\dagger})</td>
<td>7</td>
<td>2d</td>
<td>( 2 \times 2 )</td>
<td>[26]</td>
</tr>
<tr>
<td>( T_1 )</td>
<td>pentagon(^{\dagger\dagger})</td>
<td>9</td>
<td>2e</td>
<td>( 4 \times 2 )</td>
<td>[26]</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>hexagon(^{\dagger\dagger\dagger})</td>
<td>9</td>
<td>2f</td>
<td>( 4 \times 4 )</td>
<td>[26]</td>
</tr>
<tr>
<td>polar</td>
<td>( n )-valent</td>
<td>( n ) triangles</td>
<td>2c</td>
<td>( n ) degenerate</td>
<td>[27]</td>
</tr>
</tbody>
</table>

\(^{\dagger}\) two vertices of valence 4 and one of valence 5. \(^{\dagger\dagger}\) four vertices of valence 4 and one of valence 3. \(^{\dagger\dagger\dagger}\) 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, \([12, 16]\))

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

(3)

where \( b_{ij} \) are the BB-coefficients and \( b_i^d(t) := \binom{d}{i}(1 - t)^{d - i}t^i \in \mathbb{R} \) are the Bernstein polynomials of degree \( d \). Connecting \( b_{ij} \) to \( b_{i+1,j} \) and \( b_{i,j+1} \) wherever possible yields the BB-net. When \( b_{ij} \in \mathbb{R}^3 \) and \( p \) 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 \( b_{i+1,j} - b_{ij} \), respectively \( b_{i,j+1} - b_{ij} \), 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 \( p \) and \( q \) 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 \( \beta : \mathbb{R}^2 \rightarrow \mathbb{R}, \beta(u) := (u + b(u)v, a(u)v) \) so that \( p(u, 0) = E = q(u, 0) \), see Fig. 5. Then \( q \) and \( p \) join \( G^1 \) if the partial derivatives satisfy the polynomial equation or syzygy

\[
\partial_v q(u, 0) + a(u)\partial_u p(u, 0) = b(u)\partial_u p(u, 0), \quad a(u) \neq 0, \quad u \in [0..1].
\]  

(4)
In other words, the partial derivatives $\partial_v q(u, 0)$ and $\partial_u q(u, 0) = \partial_u p(u, 0)$ along $E = q(u, 0)$ are a linear combination, with weights $a(u)$ and $b(u)$, of $\partial_v p(u, 0)$ and $\partial_u p(u, 0)$, and therefore lie in the tangent plane spanned by $\partial_v p(u, 0)$ and $\partial_u p(u, 0)$. 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 $a(u) := \text{const}$. Otherwise the spline pieces join geometrically smooth, short $C^1$. For example, [24] relies on a quadratic $\beta$ to join the surrounding surface to the $n$ bi-3 patches of the multi-patch cap and uses a linear $\beta$ 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 $\beta$ linearizes and localizes (4) as an equation in terms of the BB-coefficients of $p$ and $q$. We observe that, as is typical for not-a-knot splines, at global boundaries the outermost layer recedes as in Fig. 1.

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

$$\begin{cases}
\Delta u = -f & \text{in } \Omega \in \mathbb{R}^2, \\
u = 0 & \text{on } \partial \Omega,
\end{cases}$$

(5)
over a physical domain $\Omega$ parameterized by $x := \sum_i x_i \phi_i$, i.e. polyhedral-net splines $\phi_i$ with control points $x_i$. The weak Galerkin form of Poisson’s equation on the physical domain $\Omega$ (pulled back to the domain of $\phi_j$ by $x^{-1}$) projected into the space of polyhedral-net spline functions $\phi_j$ is

$$\int_{\Omega} \nabla u_h \cdot \nabla \phi_i(x^{-1}) d\Omega = \int_{\Omega} f \phi_i(x^{-1}) d\Omega, \quad u_h := \sum_j c_j \phi_j(x^{-1}).$$

The equation can therefore be rewritten as a matrix equation $Kc = f$ to be solved for the coefficient vector $c$ of $u_h$ where

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

Since each polyhedral-net spline $\phi_i$ is in piecewise BB-form, the domain of each $\phi_i$ consists of a collection of unit squares $\Box := [-1..1]^2$. That is, the pre-image of $\Omega$, parameterized by $\sum_i x_i \phi_i$, partitions into a sum of unit squares that we enumerate with the label $\alpha$. By change of variables to $\Box$, with the gradient $J_\alpha := \nabla x_\alpha = \left[ \begin{array}{c} \partial_x x_i \\ \partial_y x_i \end{array} \right] \in \mathbb{R}^{2 \times 2}$ and $J := \sqrt{\det(J_\alpha^T J_\alpha)} \in \mathbb{R}$, (6) becomes, cf. [32], p. 153:

$$K_{ij} := \sum_\alpha \int_{\Box} (\nabla \phi_i)^T (J_\alpha^T J_\alpha)^{-1} (\nabla \phi_j) J d\Box, \quad f_i := \sum_\alpha \int_{\Box} f \phi_i J d\Box. \quad (7)$$

Here the sum is over all pieces $\alpha$ where $\phi_i$ 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 $(J_\alpha^T J_\alpha)^{-1}$ and $J$ and access the connectivity information to determine where basis functions $\phi_i$ overlap in support to minimize the computational work in assembling $K$ and $f$. Exact derivative polynomials are obtained by differencing BB-coefficients, e.g. $b_{i+1,j} - b_{i,j}$ for differentiation with respect to the first variable. For Gauss quadrature on the unit square domain, for each $\alpha$, 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.

![Figure 6: (a) complete input .obj formatted data and (b) the output polyhedral-net spline surface visualized in Bézierview [39].](image)
Figure 7: Absolute error between approximate solution and exact solution for the various polyhedral control net configurations. The exact solution $u = (u^2 - 1)(v^2 - 1)$ varies between 0 and 1 on the domain $[-1..1]^2$. 
We enforce the boundary conditions $u_h(\partial \Omega) = 0$ by modifying the polyhedral-net splines $\phi_j$ that overlap the boundary. Recall that the $\phi_j$ 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 $u_h(\partial \Omega) = 0$ 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.

3.3. No intrinsically increased error over bi-2 splines

This section demonstrates the absence of intrinsically increased error compared to bi-quadratic tensor-product splines when the domain is a square, i.e. admits tensor-product splines as well as various polyhedral-net spline configurations. So there is no intrinsic trade-off cost for the ability to handle irregularities in the polyhedral-net spline framework. 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 $u = (u^2 - 1)(v^2 - 1)$.

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.

We do not show 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 $2 \times 2$ 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 $f$, $\partial_t u = \Delta_x u + f$ on the plane, the challenge is that $\Delta_x$ is now the surface-dependent Laplace-Beltrami operator. That is, the operator is treated as in (7), with the geometry entering via the first fundamental form $(J_\alpha^T J_\alpha) : \mathbb{R}^{2 \times 3} \mathbb{R}^{3 \times 2} \to \mathbb{R}^{2 \times 2}$ of the surface and its determinant $J$. The time step is implemented explicitly as a forward
difference and the equation at each time step is solved with the \textit{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. This yields a progression of geodesic fronts, as illustrated by the white lines in Fig. 8. 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 – \textit{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


