Variational Quadratic Shape Functions for Polygons and Polyhedra

ASTRID BUNGE, TU Dortmund University, Germany
PHILIPP HERHOLZ, ETH Zurich, Switzerland
OLGA SORKINE-HORNUNG, ETH Zurich, Switzerland
MARIO BOTSCH, TU Dortmund University, Germany
MICHAEL KAZHDAN, Johns Hopkins University, USA

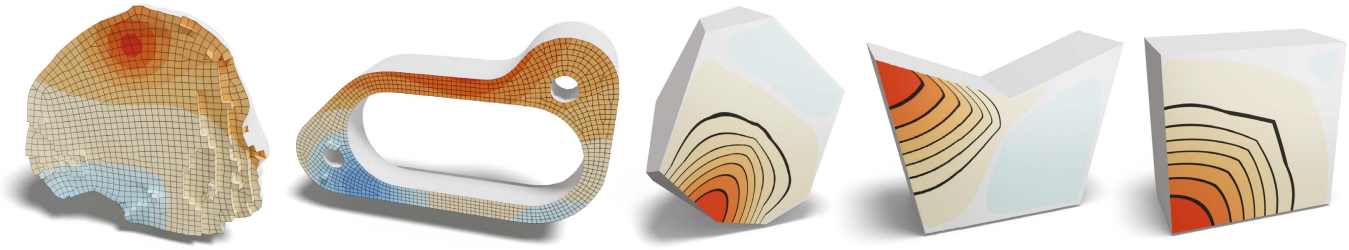


Fig. 1. Cross-sections through the solutions to a Poisson problem on polyhedral hex-dominant meshes (left). Shape functions for irregular convex and concave polyhedra as well as for a cube (right). The shape functions are variationally optimized with respect to a smoothness criterion, leading to piecewise quadratic local bases that are almost C^1 within elements. The C^1 discontinuities appear as non-smooth iso-lines in the plots. Red colors indicate positive values and blue colors indicate negative values.

Solving partial differential equations (PDEs) on geometric domains is an important component of computer graphics, geometry processing, and many other fields. Typically, the given discrete mesh *is* the geometric representation and should not be altered for simulation purposes. Hence, accurately solving PDEs on general meshes is a central goal and has been considered for various differential operators over the last years. While it is known that using higher-order basis functions on simplicial meshes can substantially improve accuracy and convergence, extending these benefits to general surface or volume tessellations in an efficient fashion remains an open problem. Our work proposes variationally optimized piecewise quadratic shape functions for polygons and polyhedra, which generalize quadratic $P2$ elements, exactly reproduce them on simplices, and inherit their beneficial numerical properties. To mitigate the associated cost of increased computation time, particularly for volumetric meshes, we introduce a custom two-level multigrid solver which significantly improves computational performance.

CCS Concepts: • **Mathematics of computing** → **Discretization**; • **Computing methodologies** → *Mesh geometry models.*

Additional Key Words and Phrases: Discrete Differential operators, polygonal and polyhedral meshes, Finite Elements

Authors' addresses: Astrid Bunge, TU Dortmund University, Dortmund, Germany, astrid.bunge@tu-dortmund.de; Philipp Herholz, ETH Zurich, Zurich, Switzerland, ph.herholz@gmail.com; Olga Sorkine-Hornung, ETH Zurich, Zurich, Switzerland, sorkine@inf.ethz.ch; Mario Botsch, TU Dortmund University, Dortmund, Germany, mario.botsch@tu-dortmund.de; Michael Kazhdan, Johns Hopkins University, Baltimore, MD, USA, misha@cs.jhu.edu.

0730-0301/2022/7-ART54
https://doi.org/10.1145/3528223.3530137

ACM Reference Format:
Astrid Bunge, Philipp Herholz, Olga Sorkine-Hornung, Mario Botsch, and Michael Kazhdan. 2022. Variational Quadratic Shape Functions for Polygons and Polyhedra. *ACM Trans. Graph.* 41, 4, Article 54 (July 2022), 14 pages. https://doi.org/10.1145/3528223.3530137

1 INTRODUCTION

Solving partial differential equations (PDEs) on geometric domains is an important component of computer graphics, geometry processing, and many other fields. Outside of computer graphics the geometric representation is usually a continuous CAD model, from which a discrete simulation mesh is generated specifically to support the numerical solution of the involved PDE – adapting the meshing resolution and element type to the requirements of the PDE solver. In computer graphics, however, the discrete mesh typically *is* the geometric representation, which should not be altered for simulation purposes. Hence, support for accurately solving PDEs on general meshes is a central goal.

The Laplace operator is ubiquitous in computer graphics and is typically discretized on triangle surface meshes and tetrahedral volume meshes using linear hat basis functions (a.k.a. $P1$ *elements*), leading to the well-known cotangent formulation [Desbrun et al. 1999; Dziuk 1988; Pinkall and Polthier 1993]. To handle arbitrary meshes, several recent approaches extend the discrete Laplacian to general polygon meshes [Alexa and Wardetzky 2011; Bunge et al. 2020; de Goes et al. 2020] and polyhedral meshes [Bunge et al. 2021].

In computer animation, elasticity simulations have been extended to polyhedral meshes by using generalized barycentric coordinates as finite element shape functions [Martin et al. 2008; Wicke et al. 2007]. Most such methods can be considered direct generalizations of standard $P1$ elements, as they reproduce them on triangles and tetrahedra. As a consequence, these methods provide a similar level

of numerical accuracy for a given mesh resolution and inherit the quadratic convergence rate under element refinement.

It is known that higher-order shape functions can offer improved accuracy and faster convergence at the price of higher computational cost. Quadratic shape functions have been proposed as a good compromise in several previous works (see, e.g., [Mezger et al. 2008; Schneider et al. 2019, 2022]), but they are currently restricted to simplicial or hexahedral meshes. We generalize quadratic shape functions to arbitrary polygonal/polyhedral meshes and demonstrate their superior numerical behavior on a range of experiments.

Inspired by the virtual refinement approach of Bunge et al. [2021; 2020], we split each cell (polygon or polyhedron) into simplices by inserting virtual vertices and then employing quadratic $P2$ elements on the refined mesh. This virtual refinement is not performed explicitly, but hidden from the user through special prolongation matrices that distribute the virtual degrees of freedom (DoFs) to the original DoFs of the input polyhedral mesh. While the construction of this prolongation is understood for *linear* shape functions (see [Bunge et al. 2020]), care has to be taken to attain the beneficial numerical properties of *quadratic* shape functions. We derive a localized per-element variational optimization that minimizes gradient discontinuities across virtual simplices within the cell. This results in **variational piecewise quadratic shape functions** for polygons and polyhedra, which generalize quadratic $P2$ elements, exactly reproduce them on simplices, and inherit their beneficial numerical properties.

Compared to computations on surface meshes, solving PDEs on volumetric meshes is considerably more expensive – an effect that is accentuated for higher-order shape functions. Our second contribution is a simple two-level **multigrid solver**, which is again hidden from the user and offers superior computational performance compared to a sparse direct supernodal solver.

An implementation of our method is available at https://github.com/mkazhdan/VariationalPolyShapeFunctions to facilitate further research.

In the following, we review related work (Section 2) and describe the virtual refinement method (Section 3), before proposing our variational quadratic shape functions and multigrid scheme for polygonal/polyhedral meshes (Section 4). In Section 5 we evaluate the numerical performance of our method for a variety of experiments on surface and volume meshes, demonstrating that our method compares favorably to existing approaches.

2 RELATED WORK

2.1 Polygon/Polyhedral Meshes

Most geometry processing applications require one essential step – a reasonable discretization of the spatial domain into a suitable tessellation. Only then is it possible to solve partial differential equations with the help of numerical approaches like the Finite Element Method (FEM) or Discrete Exterior Calculus (DEC).

To this end, triangle and quadrilateral elements have been well-studied in numerical analysis for many years. However, several applications have sparked interest in more general polygonal and polyhedral shapes. Complex geometries are not always easy to represent with only triangles/tetrahedra or quads/hexahedra and, especially in fields like solid mechanics or bio mechanics, general polyhedral elements can give a more natural representation of the domain [Tabarraei and Sukumar 2006]. Additionally, artists and architects often use general polygons, e.g. hexagons, to achieve desired properties within their models [Wang and Liu 2010].

For volume meshes, despite the increased interest in this topic over the last decade, automatically generating pure hexahedral meshes with prescribed properties has proven to be difficult. Relaxing the requirement of exclusively using hexahedral elements for tessellation motivates *hex-dominant* mesh generation algorithms, which are able to robustly and automatically create polyhedral meshes consisting mostly of well-shaped hexahedra at the cost of a small number of general polyhedral elements [Gao et al. 2017; Sokolov et al. 2016]. However, despite their prevalence, the previously described meshes are not directly supported in most standard geometry processing code bases. We address this problem by providing a new perspective on quadratic polygonal and polyhedral basis functions through which we are able to extend established techniques in computer graphics to various types of tessellations.

2.2 Polygon/Polyhedral Laplacians

In computer graphics and geometry processing, discretizations of differential operators are often needed and typically based on the Finite Elements Method or on Discrete Exterior Calculus. In recent years, several approaches for discretizing the Laplace operator over general polygonal and polyhedral meshes have been proposed. A principal challenge arising in this field is that non-planar polygons, either in the case of surface meshes or as separating faces between volumetric cells, do not bound a canonical surface in three-dimensional space. For two-manifolds, Alexa and Wardetzky [2011] handle the problem by projecting every polygon onto the plane that yields its largest projected area, combined with an inner product stabilization influenced by works on mimetic finite differences [Brezzi et al. 2005]. Following the goal of efficiently computing parallel transport of tangent vectors on curved surfaces, Sharp et al. [2019] define a discrete connection Laplacian, which is given by the trace of the second covariant derivative.

The approach of de Goes et al. [2020] allows for a variety of discrete differential operators to be extended to polygon meshes. They base their idea on the work of [Alexa and Wardetzky 2011] and generalize the mimetic differences [Brezzi et al. 2005] and virtual element method [Beirão da Veiga et al. 2013a] to define a new gradient operator, circumventing the usage of non-planar polygons.

A variety of discretizations are based on the use of special prolongation and restriction operators, creating hierarchies of virtual tessellations depending on the chosen weights within the matrices. De Goes et al. [2016] propose Subdivison Exterior Calculus, using Alexa and Wardetzky's [2011] polygon Laplacian in combination with prolongation matrices whose weights align with Catmull-Clark or Loop subdivison rules. Bunge et al. [2020] define a polygon Laplacian motivated by the virtual node method [Tang et al. 2009]. They split each polygon into a triangle fan by inserting a virtual vertex and compute the cotangent Laplacian on the refined mesh, where

this virtual refinement is hidden from the user by special prolongation/restriction matrices. By also inserting virtual vertices in polyhedral cells, this approach can be extended to polyhedral meshes, as discussed in [Bunge et al. 2021]. The latter paper introduces the Diamond Laplacian for polygon/polyhedral meshes, which uses the Discrete Duality Finite Volume method [Coudière and Hubert 2011; Hermeline 2009] to define an alternative Laplacian operator based on the Finite Volume Method on the virtually refined mesh. The diamond construction assigns a larger neighborhood to each virtual vertex, thererby increasing its accuracy but also yielding system matrices that are less sparse.

Although the discretization of a Laplacian is a valuable tool in geometry processing, our explicit construction of finite element shape functions further extends the possible set of applications we are a able to address.

2.3 Shape Functions for Polygons and Polyhedra

A displacement-based finite element method for polygons and polyhedra was presented by Rashid et al. [2006]. They omit the typical transformation to a reference element and instead define basis functions on the physical coordinates of the mesh. However, since strict continuity between the elements cannot be satisfied, this method is a non-conforming approach, generally leading to a more complex formulation. From another perspective, generalized barycentric coordinates can also be used to define shape functions for both polygons and polyhedra [Hormann and Sukumar 2017]. The central idea is to express each point inside a cell as a weighted average of cell vertices analogously to barycentric coordinates for simplices. Popular examples are Wachspress coordinates [Wachspress 1975], mean value coordinates [Floater 2003; Ju et al. 2005], harmonic coordinates [Joshi et al. 2007], or maximum entropy coordinates [Hormann and Sukumar 2008; Sukumar 2004].

These coordinates have been mainly used for cage-based deformation, which is also the motivation for higher-order constructions [Langer and Seidel 2008]. Recently Longva et al. [2020] introduced a higher-order cage based simulation algorithm, which through the virtual element method (VEM) can handle polygons and polyhedra. However, unlike our method, they do not construct explicit basis functions, since the VEM [Beirão da Veiga et al. 2013b] is based on the construction of virtual bases that follow clear definitions but are not computed in practice. The connection between this approach and polygonal/polyhedral finite elements was analyzed by Manzini et al. [2014]. The work of Gilette et al. [2016] uses generalized barycentric coordinates to define conforming scalar-valued and vector-valued basis functions of differential form order $k = 0, \ldots, 2$ for polygons and $k = 0, \ldots, 3$ for polyhedra, inspired by Whitney differential forms. Wicke et al. [2007] employed mean value coordinates as shape functions for convex polyhedra in a finite element elasticity simulation. Their approach was generalized to non-convex polyhedra by Martin et al. [2008] through the use of harmonic coordinates. Schneider et al. [2019] adapt the latter harmonic elements to define shape functions for general polyhedra in otherwise hex-dominant meshes. This allows them to extend spline-based approaches from pure hex meshes to hex-dominant mixed meshes. However, their basis construction expects special mesh configurations (general polyhedra are always separated by hexahedra), typically requiring an initial mesh refinement step. Additionally they need to explicitly enforce PDE-dependent conditions to guarantee higher-order convergence. Our method generally uses more degrees of freedom to attain the same convergence order, but it works for arbitrary polyhedral meshes and does not need PDE-dependent modifications. Bishop et al. [2014] also worked with harmonic coordinates to define shape functions on star-shaped polygons and polyhedra. Similar to Rashid and Selimotic [2006], they solve the harmonic system directly on the polyhedron instead of on the reference element. However, their integration scheme requires correcting the shape functions' derivatives in order to satisfy the divergence theorem and obtain necessary consistency properties.

Compared to the basis functions in this section, our shape functions for polygons/polyhedra are considerably simpler, since they are piecewise quadratic, and therefore can more easily and efficiently be constructed, differentiated, and integrated.

2.4 Higher-Order Basis Functions

In the context of more restricted tessellations (triangles/tetrahedra or quads/hexahedra), linear basis functions are commonly used for geometry processing. However, recent studies [Schneider et al. 2022] have pointed out that using higher-order basis functions, especially the quadratic Lagrange basis, yields more accurate results than linear elements in several settings. For example, Mezger et al. [2008] pointed out that using quadratic finite element shape functions in the context of shape editing offers not only increased numerical accuracy, but also allows for superior geometric approximation in the sense of smooth mesh interpolation.

Schneider et al. [2018] leverage the accuracy of higher-order elements by selectively increasing the degree of the shape functions for individual elements, based on their quality, allowing the authors to efficiently obtain weak solutions to a PDE that do not depend on mesh quality. In [Beirão da Veiga et al. 2017], higher-order polynomial degrees were investigated for the virtual element method and analyzed for three-dimensional problems. While also achieving higher accuracy and faster convergence, the VEM, as stated before, does not construct explicit basis functions. With our method, we are not bound to simplicial or quadrilateral elements and can extend the numerical benefits of quadratic basis functions to any kind of tessellation.

The 2D polygonal finite element basis defined by Aurojyoti et al. [2019] achieves global C^1 smoothness by elevating the degree of generalized barycentric coordinates through Bernstein-Bezier functions. While global C^1 smoothness is crucial for solving their fourth-order thin plate problem, it comes at the costs of more complex shape function that require careful numerical integration, especially for non-convex elements. In contrast, our resulting shape functions can be integrated analytically and does not require a canonical base domain, which is hard to define, especially for arbitrary polyhedra.

3 BACKGROUND

In this section we introduce the notion of Lagrange basis functions and revisit the approach of Bunge et al. [2020] for the linear case.

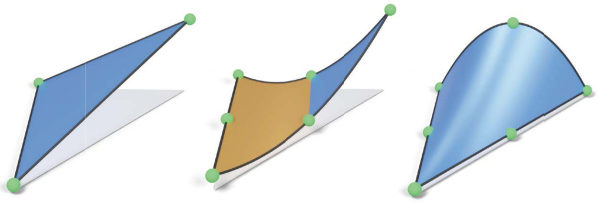


Fig. 2. Lagrange basis functions restricted to a single triangle. Linear basis (left) and two quadratic basis functions, one associated with a vertex (center) and another one with an edge midpoint (right). The functions are plotted as height fields over their domain (gray triangle). Gold and blue regions indicate negative and positive values respectively.

3.1 Lagrange Basis Functions

Consider a simplicial mesh $(\mathcal{V}, \mathcal{S})$ with vertices $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ and simplex set $\mathcal{S}$. The linear Lagrange basis function at vertex $\mathbf{v}_i$ is defined as the unique C^0 continuous and piecewise linear function ψ_i with the Lagrange interpolation property

$$\psi_i(\mathbf{v}_j) = \delta_{ij} := \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

A function is piecewise linear on a mesh if it is linear on each simplex $\sigma \in \mathcal{S}$. Considering a single triangle, for example, we obtain functions as depicted in Figure 2 (left). Degrees of freedom u_i are associated to vertices $\mathbf{v}_i$, spanning the space of continuous and piecewise linear functions

$$f(\mathbf{x}) = \sum_i u_i \psi_i(\mathbf{x}).$$

Quadratic Lagrange basis functions (Figure 2 center and right) additionally provide degrees of freedom at edge midpoints and are defined analogously to the linear case: The basis function is the unique piecewise quadratic function per simplex that takes on the value 1 at a specific node (vertex or edge midpoint) and the value 0 at all others. Note that the above definition of linear/quadratic shape functions (respective $P1/P2$ elements) holds equivalently for 3D tetrahedral meshes.

By construction, linear and quadratic Lagrange basis functions have the property that their sum is pointwise equal to one inside the element (partition of unity). Following the interpolation property of Lagrange functions, the sum has to be equal to one at all nodes, and the only linear/quadratic function satisfying this property is the constant function $f(\mathbf{x}) \equiv 1$.

3.2 The Linear Virtual Refinement Method

The concept of Lagrange basis functions does not readily carry over to polyhedral meshes. First, polygons in 3D are not necessarily planar, making it hard to define a parameter domain for the shape functions. But even if a polygon is planar, there are insufficient degrees of freedom to meet the interpolation constraints in Equation (1). For example, the space of linear functions is three-dimensional, but each shape function would be constrained to interpolate more than three values, one value at each vertex.

The idea proposed by Bunge et al. [2020] is to introduce a new *virtual* vertex inside of each polygon, to use the virtual vertices to define a triangulation of the domain (see Figure 3), and then to construct the Lagrange basis functions on the triangulation as discussed above. For a polygon mesh with vertices $\mathcal{V}$ and polygons $\mathcal{P}$, this gives a basis for piecewise linear functions on the refined mesh with $|\mathcal{V}| + |\mathcal{P}|$ vertices. However, the goal is to construct a basis function for each of the $|\mathcal{V}|$ vertices of the original mesh. Therefore, the shape functions associated to the virtual vertices are distributed to the vertices of the original n-gon using weights $w_1, \ldots, w_n$ with $\sum_i w_i = 1$. In other words, if ψ_i are the $(n+1)$ linear Lagrange basis functions on the refined polygon (Figure 3, top row), the n functions at the original vertices are $\varphi_i = \psi_i + w_i \psi_0$ for $i = 1, \ldots, n$ (Figure 3, bottom row), with ψ_0 the basis function associated to the virtual vertex. This construction guarantees that the piecewise linear polygon shape functions φ_i obey the Lagrange interpolation and partition of unity property.

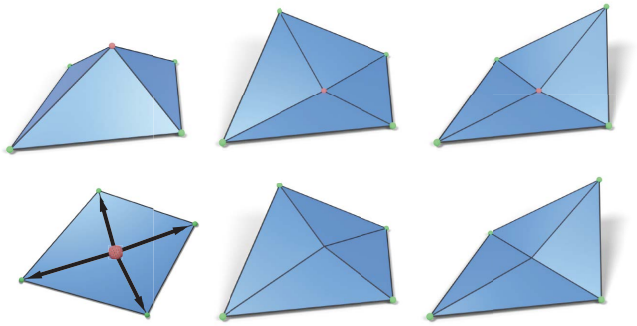


Fig. 3. Linear shape functions ψ_j on a refined quad element (top row) are combined to form a basis φ_i on the original polygon (bottom row).

In theory, any point can be chosen as the virtual vertex for triangulating the polygon. However, for the resulting systems to be well-behaved, it is reasonable to require the virtual vertex to lie close to the surface implied by the polygon's edges. Bunge et al. [2020] argue that for a polygon $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$, the position of the virtual vertex $\mathbf{v}_0 \in \mathbb{R}^3$ should be defined as the minimizer of the sum of squared triangle areas. The virtual vertex is then expressed as an affine combination of polygon vertices $\mathbf{v}_0 = \sum_i w_i \mathbf{v}_i$, since choosing a set of weights with this property guarantees linear precision for the shape functions. Usually, there is more than one set of weights that is able to represent the virtual vertex. The authors therefore add an L_2 regularization and solve the linearly constrained quadratic optimization problem

$$\{w_1, \ldots, w_n\} = \underset{\{w_1, \ldots, w_n\}}{\arg\min} \sum_{i=1}^{n} w_i^2 \tag{2}$$

$$s.t. \sum_{i=1}^{n} \text{area}(\mathbf{v}_i, \mathbf{v}_{i+1}, \mathbf{v}_0)^2 \text{ is minimized} \tag{3}$$

to determine the weights for each polygon. Using these weights they express shape functions φ_i as linear combination of the shape functions ψ_i. This mapping can be conveniently performed using

prolongation matrices [Bunge et al. 2020]. For a single polygonal face f the prolongation matrix is

$$\mathbf{P}_{ij}^f = \begin{cases} w_j & i = 0 \\ \delta_{ij} & \text{otherwise.} \end{cases}$$

Assembling the per-face matrices $\mathbf{P}^f$ into a global prolongation matrix $\mathbf{P} \in \mathbb{R}^{(|\mathcal{V}|+|\mathcal{P}|)\times|\mathcal{V}|}$ makes it straight-forward to construct the system matrices. For example, to construct the stiffness matrix $\mathbf{S}'$ associated with the basis $\{\varphi_i\}$, first construct the (standard cotangent) stiffness matrix $\mathbf{S}$ on the simplicial refinement and then multiply with $\mathbf{P}$ from both sides [Bunge et al. 2020]

$$\mathbf{S}' = \mathbf{P}^\top \mathbf{S} \mathbf{P}.$$

Following this methodology, [Bunge et al. 2021] introduce a two-step prolongation process to extend the approach to polyhedral domains. They first assign a virtual vertex to each polygonal face of the polyhedra and solve for the prolongation matrix distributing the shape functions at these virtual vertices to the original vertices. Then, given the triangulation of the polygonal faces, they further introduce a virtual vertex in the interior of each polyhedron to construct a tetrahedralization of the cell. They then solve for the prolongation matrix distributing the new virtual shape functions to the shape functions at the original vertices and the polygonal virtual vertices. Similar to the polygonal case, the weights are obtained by replacing the squared area with the squared volume in Equation (3), while still maintaining the L_2 regularization. Composing the two prolongation matrices describes how shape functions at all virtual vertices (polygonal and polyhedral) are distributed to the original vertices.

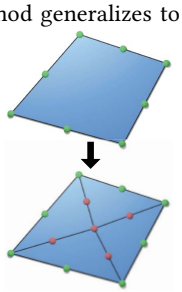
Unfortunately, these ideas do not directly extend to higher-order basis functions, as we detail in the next section. Our main contribution is a generalization of the method to quadratic Lagrange elements on polygonal and polyhedral meshes.

4 METHOD

We first extend the method of Bunge et al. [2020] to quadratic basis functions on polygon meshes, followed by a generalization to polyhedral domains.

4.1 Quadratic Basis Functions for Polygons

Suppose we are given a polygon $(\mathbf{v}_1, \ldots, \mathbf{v}_n) \subset \mathbb{R}^3$. As in [Bunge et al. 2020] we first introduce the minimizer of the sum of squared triangle areas as virtual vertex $\mathbf{v}_0 \in \mathbb{R}^3$, splitting the polygon into n triangles. For visualization purposes we commonly resort to planar polygons embedded in $\mathbb{R}^2$, however, our method generalizes to arbitrary polygons in $\mathbb{R}^3$. Similar to the linear case we are interested in basis functions associated with nodes on the polygon boundary that obey the Lagrange property. However, a direct generalization of the method of Bunge et al. [2020] to quadratic basis functions is not obvious due to the additional degrees of freedom at the edge midpoints. The inset illustrates the situation: We have $2n$ degrees of freedom on the polygon boundary (green), which we call *coarse* nodes C,



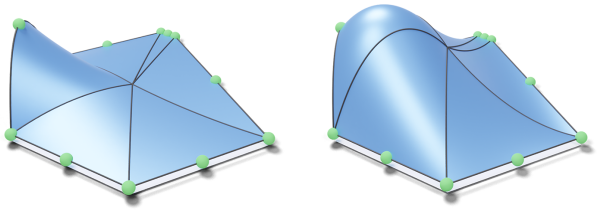


Fig. 4. Prolongation weights computed using a regularizer on their norm lead to very local basis functions at the cost of smoothness (left). Our proposed energy explicitly leads to basis functions that prioritize smoothness across internal edges (right).

and $n+1$ virtual degrees of freedom $\mathcal{K}$ (red). We call the union of both sets the set of *fine* nodes $\mathcal{F} = C \bigcup \mathcal{K}$. The location of the fine nodes are denoted by $\mathbf{p}_i$. On the virtual triangulation we can easily construct the unique quadratic Lagrange basis ψ_i for each fine node. We need to find weights w_{ij} that redistribute virtual degrees of freedom j to coarse nodes i, forming shape functions φ_i of the form

$$\varphi_i = \psi_i + \sum_{j \in \mathcal{K}} w_{ij} \psi_j \quad \text{for } i \in C. \tag{4}$$

Interpolation. The shape functions constructed according to (4) obey the Lagrange interpolation property $\varphi_i(\mathbf{p}_j) = \delta_{ij}$ at all coarse nodes $j \in C$ by construction, because the fine basis obeys $\psi_i(\mathbf{p}_j) = 0$ for all $i \in \mathcal{K}$ and $j \in C$. As a consequence we have C^0 continuity across polygon edges: The function values for each shape function are either zero at the polygon edge or the unique quadratic function satisfying the Lagrange interpolating conditions at the three nodes along that edge. Consequently, we can choose prolongation weights w_{ij} independently for each polygon, which is crucial for a linear-time, parallelizable implementation. Within each polygon the shape functions φ_i are trivially C^0 because they are linear combinations of C^0 functions ψ_i. However, the functions are not C^1 along the virtual edges (connecting the polygon vertices to the virtual vertex). This does not come as a surprise since quadratic shape functions are generally not C^1 across element edges.

Prolongation matrix. The construction of the prolongation matrix is analogous to the linear case described in Section 3.2. The only difference is that we have more than one virtual vertex. For a face f the local prolongation matrix is

$$\mathbf{P}_{ij}^f = \begin{cases} \delta_{ij} & i \in C, \\ w_{ji} & i \in \mathcal{K}, \end{cases}$$

and the per-face prolongation matrices are assembled into a global prolongation matrix $\mathbf{P}$.

Naive extension. The challenge is choosing weights so that the resulting coarse basis is 'nice' – resulting in favorable numerical behaviour. A direct generalization of the method by Bunge et al. [2020], introduced in Section 3.2, could look as follows: For a polygon with n vertices we have $2n$ coarse and $n+1$ virtual degrees of freedom. Find the $|C| \cdot |\mathcal{K}| = 2n \cdot (n+1)$ weights w_{ij} such that:

(1) The partition of unity property holds:

$$\sum_{i \in C} w_{ij} = 1 \quad \text{for } j \in \mathcal{K}. \tag{5}$$

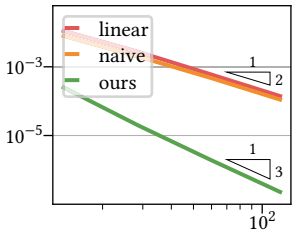
(2) The virtual node positions can be expressed as affine combination of coarse nodes using the weights:

$$\mathbf{p}_j = \sum_{i \in C} w_{ij} \mathbf{p}_i \quad \text{for } j \in \mathcal{K}. \tag{6}$$

(3) The sum of squared weights is minimized while satisfying the first two constraints

$$\{w_{ij}\} = \arg\min_{\{w_{ij}\}} \sum_{i,j} w_{ij}^2. \tag{7}$$

The solution to this linearly constrained quadratic optimization problem defines a prolongation matrix $\mathbf{P}^f$ whose entries define the basis functions φ_i. Figure 4 (left) shows one such function at an edge node. However, this construction has several shortcomings. Most importantly, it does not reproduce the desired cubic convergence rate of $P2$ elements due to its lack of smoothness. While slightly more accurate than the linear construction by virtue of the additional degrees of freedom, the naive approach converges at the same rate as linear basis functions. The inset demonstrates this convergence behavior on a set of 2D Voronoi meshes, comparing the naive approach to our quadratic basis construction proposed below. As expected from quadratic shape functions, our approach converges cubically. Another issue with the naive approach is that it fails to reproduce the standard quadratic Lagrange basis for triangles and yields shape functions with distinctly visible C^1 discontinuities, exposing the underlying virtual triangulation. These observations motivate our design of a C^1-favoring quadratic objective.



Variational energy minimization. Since the fine shape functions ψ_i are generally not C^1 across virtual edges, their linear combination is not guaranteed to be either. To define 'nice' prolongation weights we replace objective (7) of the naive approach with the squared gradient difference integrated along all virtual edges, summed over all coarse basis functions. Figure 4 (right) demonstrates that optimizing this objective subject to constraints (5) and (6) produces shape functions that are significantly smoother compared to the naive approach. Specifically, we solve the following quadratic optimization problem

$$W = \arg\min_{W} \sum_{i \in C} \sum_{\sigma \in \mathcal{E}^*} \int_\sigma \|\nabla_\sigma^+ \varphi_i - \nabla_\sigma^- \varphi_i\|^2 \, d\sigma \tag{8}$$
$$\text{s.t. constraints (5) and (6) are satisified}$$

with respect to the set of prolongation weights $W = \{w_{ij}\}$. Here $\mathcal{E}^*$ is the subset of edges in the virtual triangulation that are incident to the virtual vertex ($\mathcal{E}^* = \{(\mathbf{v}_0, \mathbf{v}_i)\}_{1 \le i \le n}$). The operator ∇_σ^+ represents the gradient with respect to the right triangle of the edge and the operator ∇_σ^- with respect to the left one.

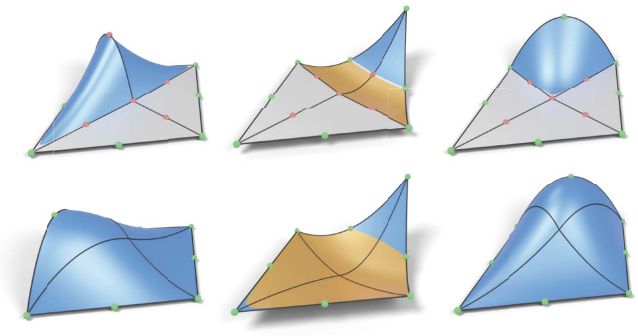


Fig. 5. Quadratic shape functions on a refined triangle mesh (top row) are combined to form shape functions on the original polygon. Regions colored in gold have negative function values. The functions are C^∞ everywhere except at the edges where they only have C^0 continuity (like the standard quadratic Lagrange basis).

The inset visualizes the two differing gradients of the basis function φ_i at $\mathbf{x}$. The C^1 discontinuities can also be seen as derivative discontinuities in the function's isolines at virtual edges. If the energy vanishes for a set of weights, all basis functions, restricted to the polygon, are C^1.

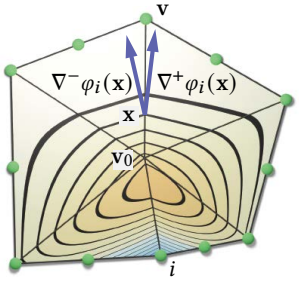


Figure 5 shows a set of quadratic Lagrange shape functions on the refined triangle mesh (top row). Minimizing the cross-edge gradient difference (8) leads to a set of piecewise quadratic polygonal shape functions for the coarse nodes (bottom row). While we try to construct shape functions that are, with respect to our energy, as C^1 as possible inside the polygons, they are generally not C^1 across the polygon edges, just like quadratic Lagrange basis functions (see Figure 4, right).

Partition of unity. As in the linear case, satisfying Equation (5) ensures a partition of unity for the quadratic basis. Specifically, since the Lagrange basis ψ_i sums to one, we have

$$\sum_{i \in C} \varphi_i = \sum_{i \in C} \left(\psi_i + \sum_{j \in \mathcal{K}} w_{ij} \psi_j \right) = \sum_{i \in C} \psi_i + \sum_{\substack{i \in C \\ j \in \mathcal{K}}} w_{ij} \psi_j = \sum_{i \in \mathcal{F}} \psi_i \equiv 1.$$

Linear precision. Our shape functions are linearly precise, which is a direct consequence of the reproduction property enforced as constraint (6). To be linearly precise the restriction of any linear function to the virtual triangulation must be in the span of the basis. To see that this is the case, consider a linear function $u \colon \mathbb{R}^3 \to \mathbb{R}$,

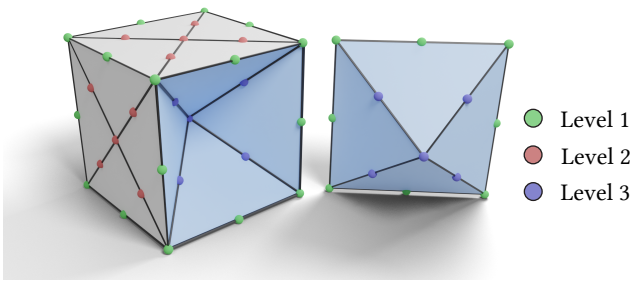


Fig. 6. Node positions for a refined cube. Parts of the mesh are detached for visualization purposes. All 2-faces are split into triangles using a virtual vertices. The volume is decomposed into tetrahedra by introducing a central virtual vertex which is connected to all face triangles. The nodes are colored according to their type.

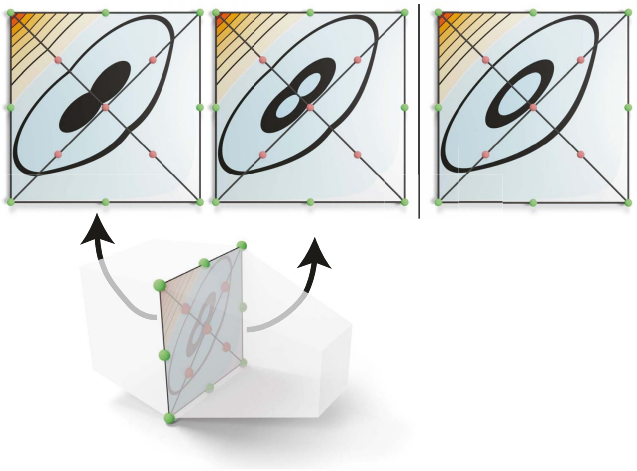


Fig. 7. Direct prolongation of all virtual degrees of freedom will result in shape functions that are not C^0 across shared faces (left). The values of the shape functions computed for each cell individually differ on the interface between both cells. To remedy this problem we compute the prolongation in two steps, which guarantees consistent values across the interface while still allowing the shape functions to be computed per-element without knowledge of neighbouring cells. See Section 4.2 for details.

let $u_i = u(\mathbf{p}_i)$ be the evaluation of u at node $\mathbf{p}_i$, and consider the sum $\sum_{i\in C} u_i\varphi_i$. By the reproduction property we have

$$\sum_{i\in C} u_i\varphi_i = \sum_{i\in C} u_i\left(\psi_i + \sum_{j\in\mathcal{K}} w_{ij}\psi_j\right)$$
$$= \sum_{i\in C} u_i\psi_i + \sum_{j\in\mathcal{K}} \underbrace{\left(\sum_{i\in C} u_i w_{ij}\right)}_{u_j}\psi_j = \sum_{j\in\mathcal{F}} u_j\psi_j = u,$$

because the reproduction property (6) extends to arbitrary linear functions. On the one hand, the sum is in the span of the φ_i. On the other, it can be expressed as the sum of the Lagrange basis functions ψ_j, weighted by the values of u at nodes $\mathbf{p}_j$. Since the Lagrange basis has linear precision, it follows that the latter sum equals u within the triangulation and hence that u is in the span of the φ_i.

Reproduction of quadratic shape functions. Although quadratic Lagrange shape functions are readily available for triangles, we can still apply our method to a triangle, virtually refining it into three triangles. In this case we obtain the coarse triangle's Lagrange shape functions as the unique solution. Consequently our method can be considered a generalization of quadratic $P2$ elements from triangles to polygons. We provide a proof in the supplementary material.

4.2 Quadratic Basis Functions for Polyhedra

The extension of our method to volumes does not change the demands we make on the prolonged basis, but involves a virtual refinement similar to that of [Bunge et al. 2021]. We introduce new virtual vertices within the polyhedron's boundary faces as well as a virtual polyhedral vertex. For the virtual face vertices we choose the point that minimizes the sum of squared triangle areas; for the virtual polyhedral vertex we use the minimizer of squared tetrahedra volumes of the resulting tessellation. Figure 6 demonstrates the procedure on a cube: Each face is split into four triangles which are connected to the central virtual vertex, forming 24 virtual tetrahedra. Each edge of this tesselation is once again equipped with midpoint nodes corresponding to the 3D quadratic Lagrange basis. We distinguish three types of nodes, as indicated by the colors in Figure 6. Level 1 nodes (green) are degrees of freedom that are defined with respect to the polyhedron itself, while level 2 (red) and level 3 (blue) nodes depend on virtual vertices. The prolongation weights will, as in the polygonal case, distribute basis functions ψ_j at virtual nodes (red and blue) to coarse nodes (green). To obtain the weights we minimize the volumetric equivalent to the quadratic energy (8). Instead of integrating along virtual edges shared by two triangles we now integrate over virtual triangles shared by two tetrahedra. Four such triangles for the cube are depicted in Figure 6 in light blue. We again integrate the squared gradient difference of the shape functions, this time over triangles of the virtual tetrahedra that do not tessellate the boundary of the polyhedron:

$$W = \arg\min_W \sum_{i\in C}\sum_{\sigma\in\mathcal{T}^*}\int_\sigma \|\nabla^+_\sigma\varphi_i - \nabla^-_\sigma\varphi_i\|^2 d\sigma \quad (9)$$
$$\text{s.t. constraints (5) and (6) are satisified,}$$

where $\mathcal{T}^*$ is the subset of triangles in the virtual tetrahedralization of the polyhedron that are incident to the virtual polyhedral vertex. For the cube, there are 36 such triangles: 12 joining the cube's edges to the interior virtual vertex and 24 connecting the four virtual edges on each of the six faces to the virtual cell vertex. In this context, the symbols ∇^+_σ and ∇^-_σ denote the gradients of φ_i on the tetrahedron to the right and to the left of the shared face σ, respectively. Figure 8 shows several polyhedra, with associated shape functions φ_i at vertices and edge midpoints. Though the functions are piecewise quadratic and can exhibit gradient discontinuities across virtual triangles in general, they reproduce the Lagrange shape functions for tetrahedra and are strictly quadratic in that case.

There is, however, a caveat that we have to address in the volumetric case. If we directly construct a prolongation matrix $\mathbf{P}^1_{2,3}$ to

redistribute level 2 and 3 nodes to level 1 nodes, we fail to preserve C^0 continuity between neighboring polyhedral cells. This is because direct prolongation solves for the contribution of virtual nodes to coarse nodes by integrating gradient mismatch over triangles *interior to a cell*. As a result, the contribution of a level 2 node to a coarse node will depend on the cell over which the prolongation weights are computed. This is not a problem for polygons since in that case *every* fine basis functions associated with a virtual node is supported within a single polygon and not shared through a common edge.

Figure 7 (left) demonstrates the problem for the coarse shape function associated to the top left corner of a quad shared by two polyhedra. If we compute the prolongation weights by distributing all virtual nodes to level 1 nodes in a single step, the shape functions defined by the two polyhedra sharing the quad do not agree on the quad (Figure 7 top left). The problem can be solved by splitting the prolongation into two steps:

(1) For each boundary face we first compute prolongation weights satisfying Equation (8), distributing level 2 nodes to level 1 nodes.
(2) Then, we solve for the prolongation matrix $\mathbf{P}$, distributing level 2 and level 3 nodes to level 1 nodes, solving Equation (9). For this second solve we fix the prolongation weights already computed in the first step as hard constraints.

This amounts to first solving for a variational basis on the boundary of the polyhedron, and then adjusting the values of the basis functions *in the interior of the polyhedron* so as to minimize the cross-edge gradient difference there as well. By construction, the per-face prolongation weights, computed in the first step, are defined by optimizing for C^1 continuity within boundary faces. Thus, two face-adjacent polyhedra necessarily distribute a level 2 node in the same way and the derived basis is guaranteed to be continuous. Figure 7 (right) shows the values of a shape function restricted to a face. The same values can be computed using the 2-step prolongation in either of the two cells.

As in the 2D case, we reproduce the quadratic Lagrange basis functions when computed on arbitrary tetrahedra, independent of the choice of virtual vertex (see supplementary material).

4.3 Implementation

Conceptually the implementation of our method is quite simple and can be easily parallelized over the mesh cells. The central task is to minimize quadratic energies of the form (8) and (9). For each cell, we begin by computing the $|\mathcal{F}| \times |\mathcal{F}|$ matrix giving the 'gradient discontinuity mass' of the finer Lagrange basis functions:

$$\mathbf{K}_{ij} = \sum_{\sigma \in \mathcal{S}^*} \int_\sigma \left\langle \nabla_\sigma^+ \psi_i - \nabla_\sigma^- \psi_i, \nabla_\sigma^+ \psi_j - \nabla_\sigma^- \psi_j \right\rangle d\sigma,$$

where $\mathcal{S}^*$ is the set of edges $\mathcal{E}^*$ or triangles $\mathcal{T}^*$ connected to the interior virtual vertex. This energy is computed in an intrinsic fashion using the metric tensor defined by the embedding of the (original and virtual) vertices, and reduces to the integration of a quadratic polynomial over a simplex. Then, the (Dirichlet-regularized) energy associated with a prolongation matrix $\mathbf{P}$ is

$$E(\mathbf{P}) = \mathrm{Tr}\left(\mathbf{P}^\top (\mathbf{K} + \varepsilon \mathbf{S}) \mathbf{P}\right)$$

Fig. 8. Cross-sections showing our quadratic shape functions, for different polyhedra: While the shape functions are optimized for smoothness, they may exhibit small gradient discontinuities across virtual faces, visible as sharp corners along iso-curves. For tetrahedra (top), where we reproduce quadratic Lagrange shape functions, the functions are C^1 inside the simplex.

where $\mathbf{S}$ is the stiffness matrix defined on the cell ($\mathbf{S}_{ij} = \int \langle \nabla \psi_i, \nabla \psi_j \rangle$, integrated over the triangles/tets in the cell), added to act as a regularizer since $\mathbf{K}$ can be singular for non-simplicial cells. In principle, the regularizer should only be added when a cell is non-simplicial, in order to guarantee quadratic reproduction. However, as ε is taken to be very small (10^{-8} in our implementation) the effect of always including the regularizer is negligible in practice. In the supplementary material we show that the choice of a Dirichlet regularizer in particular guarantees nice properties for our system. (Specifically, it ensures that the φ_i are uniquely defined, that the partition of unity

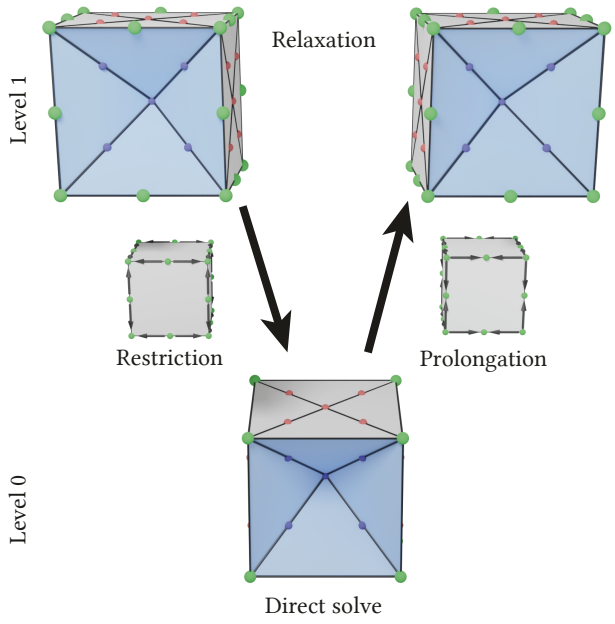


Fig. 9. A V-cycle of our multigrid implementation. The fine level relaxes the linear system using degrees of freedom at the vertices and edge-midpoints (top). The restriction/prolongation operators distribute information from/to edge-midpoints to/from vertices (middle). At the coarse level, we use a direct solve to solve the system discretized over the vertices (bottom).

property is automatically satisfied, without requiring explicit constraints, and that the linear precision property is automatically satisfied whenever the cell is *flat* – that is, whenever the d-dimensional cell has the property that the vertices of the cell as well as the virtual vertices all lie within a d-dimensional plane.)

As the entries in $\mathbf{P}$ are linear in the weights $\{w_{ij}\}$, this gives a quadratic energy in the prolongation weights. We add the partition of unity (5) and linear reproduction (6) constraints using Lagrange multipliers and solve the resulting KKT system.

4.4 Multigrid

Unfortunately, the transition from polygons to polyhedra reveals the 'curse of dimensionality'. For both polygonal and polyhedral meshes, the total number of nodes is equal to the number of vertices plus the number of edges in the mesh. Additionally, a function associated with a node is supported on all cells sharing that node. However, since the system matrix will typically have a non-zero entry for every pair of nodes whose associated functions have overlapping support, we get significantly denser system matrices for polyhedral meshes. For example, the local stiffness matrix of a single hexahedron contains 400 non-zero entries while the stiffness matrix of a single quad contains only 64.

An approach to improve the sparsity of the system matrix would be to use the refined tetrahedral mesh explicitly and forego the polyhedral basis approach. Though this would reduce the number of non-zero entries per row, it would come at the cost of a significantly higher-dimensional system matrix, resulting in a similar problem of computational complexity. We propose a solution to this issue which consists of a custom multigrid approach tailored to our virtual vertex setting, which provides a significant performance boost while still still generating an accurate solution.

Our multigrid hierarchy consists of two levels: The coarse level contains nodes at vertices (Figure 9, bottom row), the fine level also contains nodes at the edges of the input polyhedron (top row). Note that these levels are not directly related to the notion of level used in Section 4. We perform multiple V-cycles as depicted in Figure 9 to solve linear systems $\mathbf{Ax} = \mathbf{b}$ represented in our basis. A single V-cycle consists of four steps.

Relaxation. This step performs n_{it} Gauss-Seidel iterations. We implemented a parallel version using greedy graph coloring to identify maximally independent sets of nodes.

Restriction. The restriction operation can be expressed using the matrix $\mathbf{R} \in \mathbb{R}^{|\mathcal{V}|\times(|\mathcal{V}|+|\mathcal{E}|)}$ for a polyhedral mesh with $|\mathcal{V}|$ vertices and $|\mathcal{E}|$ edges:

$$\mathbf{R}_{ij} = \begin{cases} 1 & j \text{ is vertex node and } i = j, \\ \frac{1}{2} & j \text{ is edge node and } i \text{ is indicent to } j, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Given an approximate (fine) solution $\mathbf{x}$ the restriction computes the coarse level residual as $\mathbf{b}^C = \mathbf{R}(\mathbf{b} - \mathbf{Ax})$.

Direct Solve. At the coarse level we employ the sparse supernodal Cholesky solver implemented in Cholmod [Chen et al. 2008] to solve

$$\mathbf{RAR}^\top \mathbf{x}^C = \mathbf{b}^C. \tag{11}$$

The modified system is not only smaller, but also significantly sparser compared to the original one and is solved more efficiently.

Prolongation. The prolongation operation uses the coarse solution to correct the estimated fine solution. Using the Galerkin formulation, this can be expressed in terms of the transpose of the restriction operation: $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{R}^\top \mathbf{x}^C$.

The multigrid implementation has two parameters: the number of relaxation iterations n_{it} and the number of V-cycles n_{cy}. We found $n_{\text{it}} = 3$ to be optimal in terms of total convergence time, which we evaluate empirically in the next section.

5 EVALUATION

We evaluate the proposed variational quadratic shape functions in a variety of geometry processing applications for both polyhedral and polygonal meshes. Example tessellations can be found in Figure 10 (3D) and Figure 17 (2D). We use a variety of quantitative tests discussed by Bunge et al. [2021], focusing on problems solved with existing polygon Laplacians as well as other basis functions designed for general volume and surface meshes. Additionally, qualitative examples highlight visible benefits of using quadratic basis functions as compared to the more common linear elements.

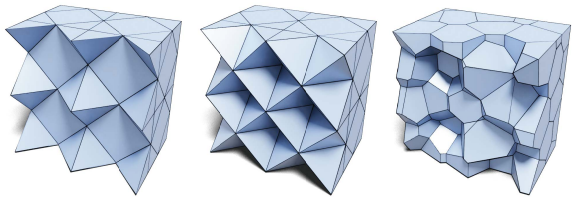


Fig. 10. Polyhedral meshes used in our experiments. From left to right: Pyramids, Truncated, Voronoi.

5.1 Numerical Accuracy

On surface meshes, we compare the Laplacian operator obtained through our basis functions to the existing discretizations of Alexa and Wardetzky [2011], Bunge et al. [2020], de Goes et al. [2020] and Bunge et al. [2021]. According to the recommendation of the original authors, we set the hyper-parameters to be $\lambda = 2$ for [Alexa and Wardetzky 2011] and $\lambda = 1$ for [de Goes et al. 2020] and the virtual vertex for the works of [Bunge et al. 2020] and [Bunge et al. 2021] to be the squared triangle area minimizer. Additionally, Laplacians discretized through polygon basis functions obtained with harmonic basis functions [Martin et al. 2008] and the Poly-Spline finite element method [Schneider et al. 2019] are discussed. We forego a quantitative comparison to the mean value coordinate basis functions [Wicke et al. 2007] since they are only defined on meshes with convex tessellations. In the case of polyhedral domains, we compare to the volume extension of Bunge et al. [2020] introduced by [Bunge et al. 2021] and their volumetric Diamond Laplacian with their suggested choice for the virtual cell vertex. The harmonic bases from Martin et al. [2008] and Schneider et al.'s [2019] Poly-Spline basis functions are equally extendable to volume meshes and discussed in the evaluation. To verify the quality of the results we used the code published by the authors for the works of [Bunge et al. 2021, 2020; Schneider et al. 2019].

Poisson Equation. We analyze the convergence behavior of the Laplacian under refinement by solving the Poisson equation $\Delta u = f$ on different tessellations of the unit square and unit cube for the Laplacian of the Franke test functions [Franke 1979] with Dirichlet boundary conditions. We solve the discrete system $\mathbf{Su} = \mathbf{Mb}$, where the boundary values for the right-hand side $\mathbf{b}$ are fixed to the analytic values of the 2D or 3D Franke test function at the respective vertex position. The formulas for the tests functions can be found in the Appendix section of [Schneider et al. 2019]. Given a mesh $\mathcal{M} = (\mathcal{V}, \mathcal{P})$ with vertices $\mathcal{V}$ and faces $\mathcal{P}$ and a set of basis functions $\{\varphi_1, ..., \varphi_n\}$ defined on this tessellation, the mass and stiffness matrices, $\mathbf{M}, \mathbf{S} \in \mathbb{R}^{n \times n}$, are discretized as

$$\mathbf{M}_{ij} = \int_{\mathcal{M}} \varphi_i \varphi_j \qquad \text{and} \qquad \mathbf{S}_{ij} = -\int_{\mathcal{M}} \langle \nabla \varphi_i, \nabla \varphi_j \rangle.$$

We measure the deviation of the solution u from the true function values, displayed in Figure 11 and Figure 13 for surface meshes and in Figure 12 and Figure 14 for volumetric meshes. The convergence plots demonstrate that our method achieves the desired cubic convergence rate for all tessellations, for both surfaces and volumes, in contrast to the quadratic slope of the linear discretizations. Bi/triquadratic quadrilateral/hexahedral $Q2$ elements and Poly-Spline basis functions also provide cubic convergence. The slightly lower error of the traditional $Q2$ basis is to be expected, since they impose more degrees of freedom per element. Regarding the Poly-Spline basis [Schneider et al. 2019], a major advantage of this method are the few degrees of freedom needed to obtain their level of accuracy. Unfortunately, this feature only holds for specific meshes with large regular quad/hex regions. Furthermore, their basis construction requires an initial subdivision step whenever the input mesh does not meet their requirements. It ensures that for any quad/hex only one edge/face is adjacent to a general polygon/polyhedron. Additionally, they enforce that non-quad/non-hex cells are not adjacent to each other or at the boundary of the mesh. Even in the case of these constrained tessellations our basis functions still give better results than the Poly-Spline method for surface meshes. However, we yield slightly higher errors on the volume meshes.

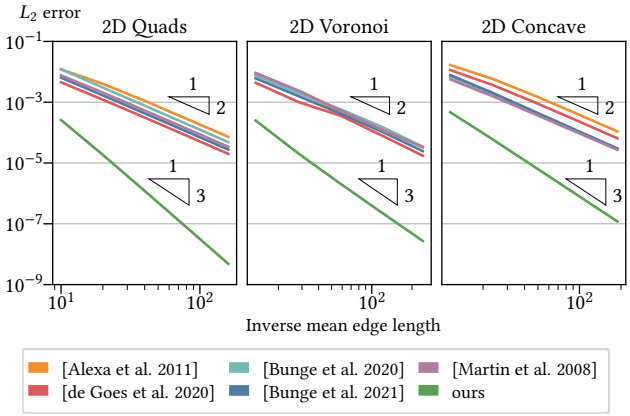


Fig. 11. L^2 error in log-log scale of the Poisson system solved for Franke's test function on planar meshes with quads (left), Voronoi cells (center), and concave faces (right).

Eigenvalue Reproduction. The eigenfunctions of the Laplacian form an orthonormal basis. In the case of a unit sphere, these are the spherical harmonics $Y_l^m \colon S \to \mathbb{R}$ with eigenvalues $-l(l+1)$. We assess different finite elements and discrete exterior calculus discretizations of the mass $\mathbf{M}$ and stiffness $\mathbf{S}$ matrices by comparing the (generalized) eigenvalues of the system $-\mathbf{Su} = \lambda \mathbf{Mu}$ with the analytic eigenvalues. Results for the hexagon sphere can be seen in Figure 15, showing that our basis functions generate more accurate eigenvalues, with significantly less deviation from the ground truth than those produced by competing approaches.

As with spherical harmonics, the solutions to the Laplacian eigenvalue problem on the unit 3-ball $\mathcal{B}^3$ with zero Dirichlet boundary conditions

$$-\Delta u = \lambda u \;\text{ in }\; \mathcal{B}^3 \quad \text{with} \quad u = 0 \;\text{ on }\; \partial \mathcal{B}^3,$$

have analytical expressions. As above, these provide the ground truth for comparing the generalized eigenvalues computed using

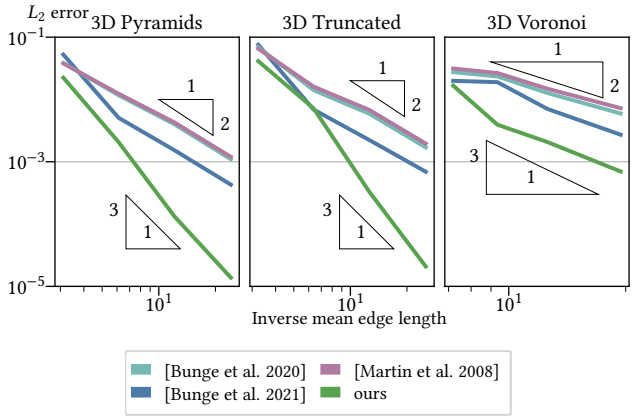


Fig. 12. L^2 error in log-log scale of the Poisson system solved for Franke's test function on different tessellations of the unit cube. The tesselations are as follows: Pyramids (left), truncated cells (center), and Voronoi cells (right).

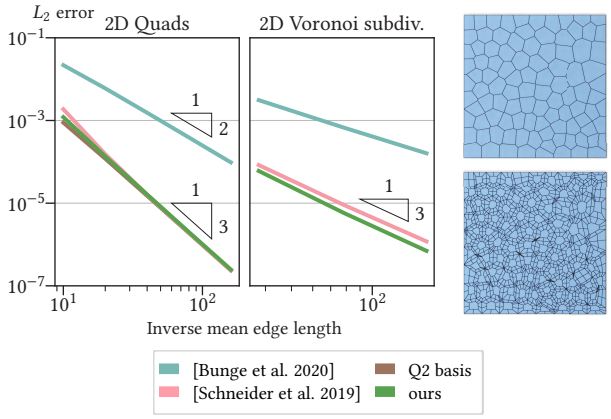


Fig. 13. L^2 error in log-log scale of the Poisson system solved for Franke's test function on planar grids with quads (left) and subdivided Voronoi cells (center) in accordance to the needs of the Poly-Spline method (right).

discrete mass **M** and stiffness **S**. As in the 2D case, we solve the generalized eigenvalue problem, this time locking the boundary values to enforce the Dirichlet constraints. Figure 16 shows the 34 smallest nonzero eigenvalues obtained with the different polyhedral Laplacians on a truncated unit ball. The Poly-Spline method is not included in the comparison because the polygon/polyhedral meshes do not meet their compatibility conditions (see description above).

Linear Elasticity. To go beyond Laplacian systems, we also evaluate our method on a static linear elasticity simulation. The governing PDE for the body Ω and a displacement vector field $\mathbf{u}$ are

$$\nabla \cdot \boldsymbol{\sigma} = \mathbf{f} \quad \text{in } \Omega,$$

with linear Cauchy strain $\boldsymbol{\epsilon} = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathsf{T}}\right)$ and linear material behavior $\boldsymbol{\sigma} = \mathbf{E} : \boldsymbol{\epsilon}$. The material matrix $\mathbf{E}$ is built from the Young's modulus E and Poisson ratio ν.

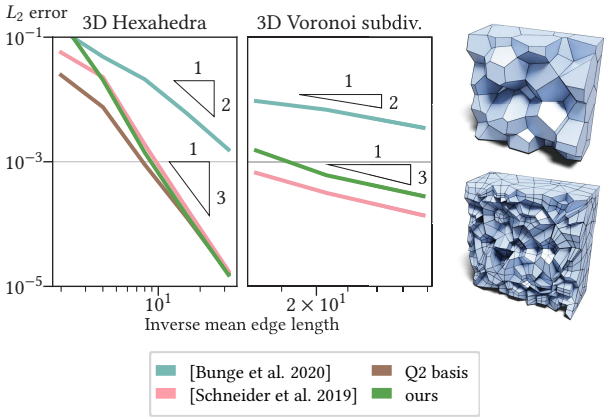


Fig. 14. $L2$ error in log-log scale of the Poisson system solved for Franke's test function on unit cubes with regular hexahedra (left) and subdivided Voronoi cells (center) in accordance to the Poly-Spline refinement (right).

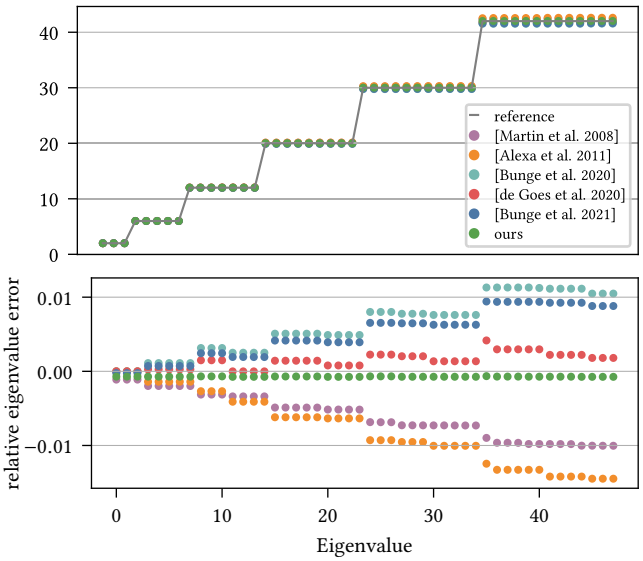


Fig. 15. The 48 smallest non-zero eigenvalues of the Laplacian, computed on a unit sphere tessellated by hexagons. The top plot shows the eigenvalues, the bottom shows the deviation from the ground truth. Our method outperforms all other discretizations, with results barely deviating from the desired values.

A challenging issue is the *locking phenomenon*, which can be observed for linear elements when setting ν close to 0.5. This phenomenon is shown for the rest state of a bar, acted on by gravity, in the top row of Figure 17 and on the left of Figure 20. This problem can be overcome by using higher-order basis functions as shown in the bottom row and right column of the respective figures. For all tessellations, our basis functions avoid locking artifacts.

Geodesics in Heat. Because our basis functions, and specifically their gradients, can be evaluated at any point on the mesh, we can

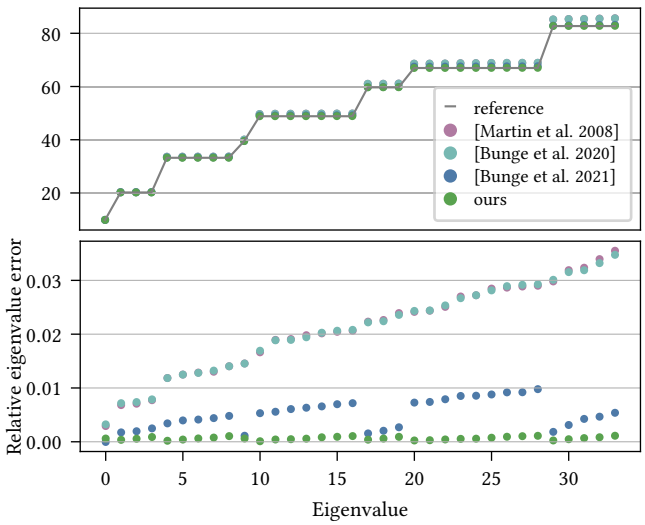


Fig. 16. The 34 smallest non-zero eigenvalues of the Laplacian, computed on a truncated polyhedral tessellation of the unit ball. The top plot shows the eigenvalues, the bottom shows the deviation from the ground truth. Our method outperforms all other discretizations, with results barely deviating from the desired values.

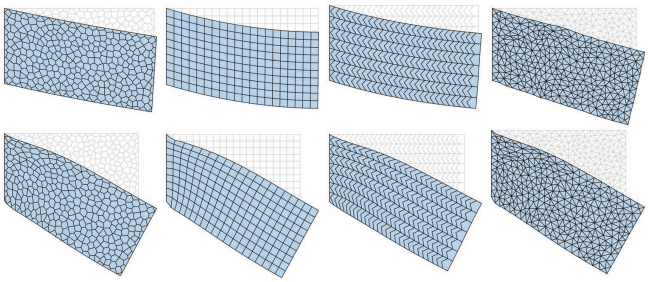


Fig. 17. Displacements due to gravity computed by linear elasticity with Young's modulus 1e+10 and Poisson ratio 0.4999 on differently tessellated 2D bars. The top row shows the results for linear elements [Bunge et al. 2020] and the bottom row for our quadratic basis functions.

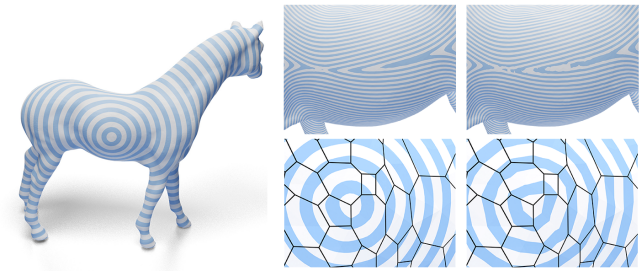


Fig. 18. Geodesics in heat [Crane et al. 2013] on a polygonal mesh. Quadratic basis functions (left, center) lead to smoother results with less artifacts compared to the linear version (right). All images have been rendered by evaluating the result on triangulated and refined polygons.

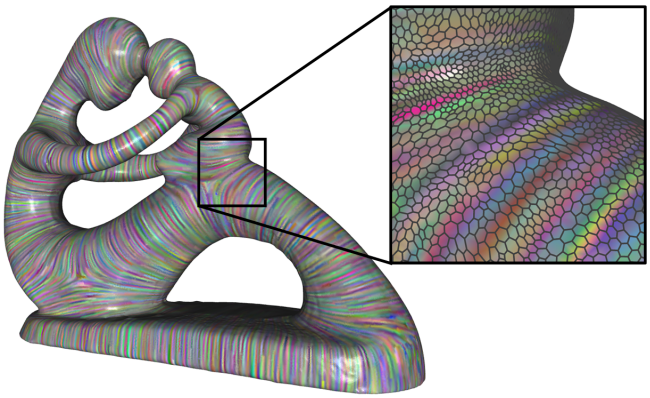


Fig. 19. Line Integral Convolution visualization obtained by anisotropically diffusing a random color signal along the maximum curvature direction.

straightforwardly implement Geodesics in Heat [Crane et al. 2013] for polygon meshes. In the original method a constant gradient per face is normalized and integrated. With quadratic elements, we use numerical quadrature to integrate the point-wise dot product of the normalized gradient of the diffused delta function and the gradient of the basis vectors. Figure 18 compares results for linear and quadratic polygon basis functions. In addition to supporting shorter diffusion time-steps (due to the effective refinement that comes from adding degrees of freedom at edge midpoints) the quadratic basis also produces smoother functions.

Anisotropic Smoothing. Typically, the metric tensor is defined by the positions of the mesh vertices and the virtual vertices. However, our implementation allows us to to modify the tensor for each triangle in the virtual refinement to support anisotropic diffusion [Clarenz et al. [n.d.]]. For example, this allows us to implement Line Integral Convolution as in [Prada et al. 2018]. Starting with a vector field, we scale the metric tensor so as to shrink distances in directions parallel to the vector field while preserving distances along perpendicular directions. Using this metric, we anistotropically diffuse a random signal on the mesh, smoothing the signal along the vector field's streamlines. Figure 19 shows the resulting visualization of the maximal curvature directions field on the fertility model.

5.2 Multigrid and Timings

A specific choice of basis determines the structure of the stiffness matrix and consequently the performance of a linear system solve. Here we evaluate this relationship and assess the convergence behavior of our multigrid solver.

Multigrid vs. Direct Solve. Our multigrid approach, introduced in Section 4.4, sidesteps the direct solution of the full system and employs a direct solver at the coarse level only. In Figure 21 we illustrate its convergence behavior using four polyhedral meshes (all hex-dominant except for the first one which contains pyramids, see Figure 10). We report timings of our multigrid approach for the solution of a volumetric Poisson problem, relative to the time needed to solve the system using the direct supernodal Cholesky

Table 1. We compare statistics for the solution of a Poisson problem using different basis constructions. The timings include solving times using supernodal Cholesky decomposition and back substitution. For our method we additionally include timings for our multigrid approach and report the time it takes to reduce the residual error to $\|\mathbf{x} - \mathbf{x}_{\text{ref}}\| / \|\mathbf{x}_{\text{ref}}\| < 10^{-8}$ with respect to the direct solution.

Mesh	$\lvert\mathcal{V}\rvert$	[Martin et al. 2008]			[Bunge et al. 2020]			[Bunge et al. 2021]			ours			
		time	dof	nnz	time	dof	nnz	time	dof	nnz	time	MG time	dof	nnz
Voronoi 2D	200k	0.57s	200k	2.6M	0.73s	200k	2.6M	4.3s	200k	7.6M	7.7s	3.2s	507k	12.1M
Voronoi 2D	800k	9.4s	800k	10.4M	9.8s	800k	10.4M	21s	800k	29.6M	36.7s	24.2s	2M	48M
Bunny 3D	80k	1.8s	80k	2M	1.6s	80k	2M	6.2s	80k	5.9M	26s	3.6s	316k	17.6M
Kong 3D	160k	5.45s	167k	4.3M	4.44s	167k	4.3M	25.9s	167k	12M	97.6s	11.9s	662k	38M

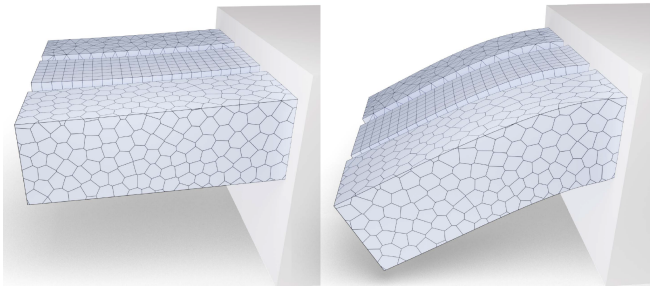


Fig. 20. Displacements due to gravity by linear elasticity with Young's modulus 5e+10 and Poisson ratio 0.4999 on differently tessellated bars consisting of tetrahedra, hexahedra, and Voronoi cells. The left figure shows the results for linear basis functions [Bunge et al. 2021] and the right for our quadratic basis.

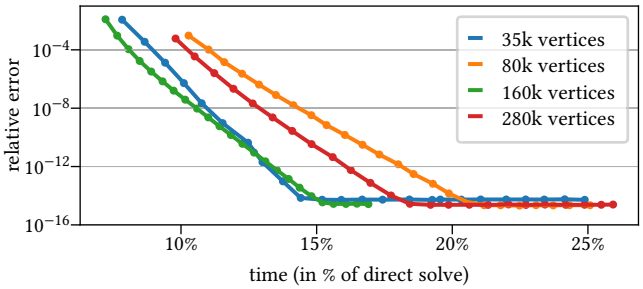


Fig. 21. Multigrid convergence for four polyhedral meshes of different size. Each point represents a V-cycle and we report time relative to a direct solve of the full system for each mesh. The relative error is measured by $\|\mathbf{x} - \mathbf{x}_{\text{ref}}\| / \|\mathbf{x}_{\text{ref}}\|$ where $\mathbf{x}_{\text{ref}}$ is the direct solution.

solver implemented in Cholmod [Chen et al. 2008]. The solver exhibits the expected convergence rate, with relative errors reducing exponentially until floating-point precision is reached. In addition, it produces an accurate solution, with error smaller than 10^{-8} relative to the solution of the direct solve, but in a fraction of the time needed by the direct solver. Setup times for the multigrid method are included in the measurements, specifically the factorization of the stiffness matrix at the coarse level.

Solving times. In Table 1 we compare statistics for the solution of a Poisson problem using different basis constructions. The time it takes to solve the system using a direct solver depends on the degrees of freedom (which manifest as number of rows and columns) and the number of non-zeros (nnz). To define our quadratic basis functions we need to introduce additional nodes, which leads to larger and denser systems. The superior convergence behavior of our method therefore comes at the price of a more costly Poisson system solve compared to methods that only use vertex nodes [Alexa and Wardetzky 2011; Bunge et al. 2020; Martin et al. 2008]. As we use a Cholesky solver, discrete Laplacians with the same non-zero structure give the same solve times. The approach of Bunge et al. [2021] introduces coefficients relating vertex nodes of adjacent elements leading to denser matrices.

As expected, the solve times of our quadratic method are higher for all examples due to the larger number of degrees of freedom and denser matrices. However, using the multigrid construction significantly lowers the computational time. In Table 1 we report the time it takes to achieve a relative accuracy of 10^{-8}, typically requiring between 5 and 10 V-cycles. Since we have already validated that our method generally yields superior accuracy, and since the multigrid solver only requires factoring a matrix whose sparsity structure matches that of matrices defined using linear elements, this leads towards an overall improvement in quality at negligible increase in computation time.

6 CONCLUSION AND FUTURE WORK

We presented a new approach for defining finite elements shape functions for general polygonal and polyhedral meshes. In contrast to previous linear approaches, the basis we propose is quadratic and exhibits the commensurate convergence properties. The key is defining continuous basis functions that are linear combinations of standard quadratic Lagrange functions on a virtual simplicial refinement, with weights defined by solving a variational optimization problem encouraging the basis functions to minimize gradient discontinuities along virtual edges. Leveraging the natural hierarchical structure within our construction, we define a multigrid solver that mitigates the loss of sparsity associated with higher-order shape functions. We demonstrate the efficacy of our approach comparing numerical performance for standard geometry processing applications requiring a discretization of the Laplacian, applications in simulation requiring a more general stiffness matrices, as well as

applications that benefit from pointwise evaluation. Empirically, our approach provides cubic convergence for general polygonal and polyhedral meshes without the ensuing increase in computational complexity.

By focusing on the general construction of finite element shape functions our approach provides a general framework that can be directly incorporated into many applications in geometry processing and simulation.The source code is available at https://github.com/mkazhdan/VariationalPolyShapeFunctions.

In the future, we will consider higher-order extensions of our approach, with higher derivative continuity constraints across elements – enabling thin shell simulations on polygonal meshes. We would also like to extend our approach to the construction of 1-form bases, supporting the large body of work on vector field processing.

ACKNOWLEDGMENTS

This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101003104).

REFERENCES

Marc Alexa and Max Wardetzky. 2011. Discrete Laplacians on General Polygonal Meshes. *ACM Transactions on Graphics* 30, 4 (2011), 102:1–102:10.

Prusty Aurojyoti, Piska Raghu, Amirtham Rajagopal, and Jn Reddy. 2019. An n-sided polygonal finite element for nonlocal nonlinear analysis of plates and laminates. *Internat. J. Numer. Methods Engrg.* 120 (2019), 1071–1107.

Lourenço Beirão da Veiga, Franco Brezzi, Andrea Cangiani, Gianmarco Manzini, Luisa Donatella Marini, and Alessandro Russo. 2013b. Basic principles of Virtual Element Methods. *Mathematical Models and Methods in Applied Sciences* 23, 1 (2013), 199–214.

Lourenço Beirão da Veiga, Franco Brezzi, and Luisa Donatella Marini. 2013a. Virtual Elements for Linear Elasticity Problems. *SIAM J. Numer. Anal.* 51, 2 (2013), 794–812.

Lourenço Beirão da Veiga, Franco Dassi, and Alessandro Russo. 2017. High-order Virtual Element Method on polyhedral meshes. *Computers and Mathematics with Applications* 74 (2017), 1110–1122.

J.E. Bishop. 2014. A displacement-based finite element formulation for general polyhedra using harmonic shape functions. *Internat. J. Numer. Methods Engrg.* 97 (2014), 1–31.

Franco Brezzi, Konstantin Lipnikov, and Valeria Simoncini. 2005. A Family of Mimetic Finite Difference Methods on Polygonal and Polyhedral Meshes. *Mathematical Models and Methods in Applied Sciences* 15, 10 (2005), 1533–1551.

Astrid Bunge, Mario Botsch, and Marc Alexa. 2021. The Diamond Laplace for Polygonal and Polyhedral Meshes. *Computer Graphics Forum* 40, 5 (2021), 217–230.

Astrid Bunge, Philipp Herholz, Misha Kazhdan, and Mario Botsch. 2020. Polygon Laplacian Made Simple. *Computer Graphics Forum* 39, 2 (2020), 303–313.

Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.* 35, 3 (2008), 1–14.

U. Clarenz, U. Diewald, and M. Rumpf. [n.d.]. Anisotropic geometric diffusion in surface processing. In *Proceedings Visualization 2000.* 397–405.

Yves Coudière and Florence Hubert. 2011. A 3D Discrete Duality Finite Volume Method for Nonlinear Elliptic Equations. *SIAM J. Sci. Comput.* 33, 4 (2011), 1739–1764.

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013. Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow. *ACM Transactions on Graphics* 32, 5 (2013), 152:1–152:11.

Fernando de Goes, Andrew Butts, and Mathieu Desbrun. 2020. Discrete Differential Operators on Polygonal Meshes. *ACM Transactions on Graphics* 39, 4 (2020), 110:1–110:14.

Fernando de Goes, Mathieu Desbrun, Mark Meyer, and Tony DeRose. 2016. Subdivision Exterior Calculus for Geometry Processing. *ACM Transactions on Graphics* 35, 4 (2016), 133:1–133:11.

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. 1999. Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. In *Proceedings of ACM SIGGRAPH.* 317–324.

Gerhard Dziuk. 1988. *Finite Elements for the Beltrami operator on arbitrary surfaces.* Springer Berlin Heidelberg, 142–155.

Michael S. Floater. 2003. Mean value coordinates. *Computer Aided Geometric Design* 20, 1 (2003), 19–27.

Richard Franke. 1979. *A critical comparison of some methods for interpolation of scattered data.* Technical Report. Naval Postgraduate School.

Xifeng Gao, Daniele Panozzo, Wenping Wang, Zhigang Deng, and Guoning Chen. 2017. Robust Structure Simplification for Hex Re-Meshing. *ACM Transactions on Graphics* 36, 6 (2017), 185:1–185:13.

Andrew Gillette, Alexander Rand, and Chandrajit Bajaj. 2016. Construction of Scalar and Vector Finite Element Families on Polygonal and Polyhedral Meshes. *Computational Methods in Applied Mathematics* 16, 4 (2016), 667–683.

F. Hermeline. 2009. A Finite Volume Method for Approximating 3D Diffusion Operators on General Meshes. *J. Comput. Phys.* 228, 16 (2009), 5763–5786.

K. Hormann and N. Sukumar. 2008. Maximum Entropy Coordinates for Arbitrary Polytopes. *Computer Graphics Forum* 27, 5 (2008), 1513–1520.

Kai Hormann and N. Sukumar. 2017. *Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics.* Taylor & Francis.

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic Coordinates for Character Articulation. *ACM Transactions on Graphics* 26, 3 (2007), 71—-es.

Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean Value Coordinates for Closed Triangular Meshes. *ACM Transactions on Graphics* 24, 3 (2005), 561–566.

Torsten Langer and Hans-Peter Seidel. 2008. Higher Order Barycentric Coordinates. *Computer Graphics Forum* 27, 2 (2008).

Andreas Longva, Fabian Löschner, Tassilo Kugelstadt, José Antonio Fernández-Fernández, and Jan Bender. 2020. Higher-Order Finite Elements for Embedded Simulation. *ACM Transactions on Graphics* 39, 6 (2020), 181:1–181:14.

G. Manzini, A. Russo, and N. Sukumar. 2014. New perspectives on polygonal and polyhedral finite element methods. *Mathematical Models and Methods in Applied Sciences* 24 (2014), 1665–1699.

Sebastian Martin, Peter Kaufmann, Mario Botsch, Martin Wicke, and Markus Gross. 2008. Polyhedral Finite Elements Using Harmonic Basis Functions. *Computer Graphics Forum* 27, 5 (2008), 1521–1529.

Johannes Mezger, Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. 2008. Interactive Physically-Based Shape Editing. In *Proceedings of ACM Symposium on Solid and Physical Modeling.* 79–89.

Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experim. Math.* 2 (1993), 15–36.

Fabián Prada, Misha Kazhdan, Ming Chuang, and Hugues Hoppe. 2018. Gradient-Domain Processing within a Texture Atlas. *ACM Transactions on Graphics* 37, 4 (2018), 154:1–154:14.

M. Rashid and M. Selimotic. 2006. A three-dimensional finite element method with arbitrary polyhedral elements. *Internat. J. Numer. Methods Engrg.* 67 (2006), 226 – 252.

Teseo Schneider, Jérémie Dumas, Xifeng Gao, Mario Botsch, Daniele Panozzo, and Denis Zorin. 2019. Poly-Spline Finite-Element Method. *ACM Transactions on Graphics* 38, 3 (2019), 19:1–19:16.

Teseo Schneider, Yixin Hu, Jérémie Dumas, Xifeng Gao, Daniele Panozzo, and Denis Zorin. 2018. Decoupling Simulation Accuracy from Mesh Quality. *ACM Transactions on Graphics* 37, 6 (2018), 280:1–280:14.

Teseo Schneider, Yixin Hu, Xifeng Gao, Jeremie Dumas, Denis Zorin, and Daniele Panozzo. 2022. A Large Scale Comparison of Tetrahedral and Hexahedral Elements for Solving Elliptic PDEs with the Finite Element Method. *ACM Transactions on Graphics* 41, 3 (2022), 23:1–23:14.

Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019. The Vector Heat Method. *ACM Transactions on Graphics* 38, 3 (2019), 24:1–24:19.

Dmitry Sokolov, Nicolas Ray, Lionel Untereiner, and Bruno Lévy. 2016. Hexahedral-Dominant Meshing. *ACM Transactions on Graphics* 35, 5 (2016), 157:1–157:23.

N. Sukumar. 2004. Construction of polygonal interpolants: A maximum entropy approach. *Internat. J. Numer. Methods Engrg.* (2004), 2159–2181.

Alireza Tabarraei and N. Sukumar. 2006. Application of Polygonal Finite Elements in Linear Elasticity. *International Journal of Computational Methods* 03 (2006), 503–520.

Xu-hai Tang, Sheng-chuan Wu, Chao Zheng, and Jian-hai Zhang. 2009. A novel virtual node method for polygonal elements. *Applied Mathematics and Mechanics* 30, 10 (2009).

Eugene L. Wachspress. 1975. *A Rational Finite Element Basis.* Academic Press.

Wenping Wang and Yang Liu. 2010. A Note on Planar Hexagonal Meshes. In *Nonlinear Computational Geometry*, Ioannis Z. Emiris, Frank Sottile, and Thorsten Theobald (Eds.). Springer New York, 221–233.

Martin Wicke, Mario Botsch, and Markus Gross. 2007. A Finite Element Method on Convex Polyhedra. *Computer Graphics Forum* 26, 3 (2007), 355–364.