DISCRETE LAPLACIANS FOR GENERAL POLYGONAL AND POLYHEDRAL MESHES

DISSERTATION

zur erlangung des grades eines

doktors der naturwissenschaften

der technischen universität dortmund
an der fakultät informatik

von

ASTRID PONTZEN

dortmund
2024

Tag der mündlichen Prüfung:	17. Oktober 2024
Dekan:	Prof. Dr. Jens Teubner
Gutachter:	Prof. Dr. Mario Botsch
	Prof. Dr. Klaus Hildebrandt

ACKNOWLEDGMENTS

Completing this thesis has been a journey marked by moments of joy and challenge, and it certainly was a journey I could not have undertaken alone. This work would not have been possible without the people who have supported and guided me these past six years. While the following list is not exhaustive and lacks any particular order, I want to express my heartfelt gratitude to each of them.

I will always be thankful to my supervisor, Mario Botsch, whose impact on my academic and personal growth goes far beyond words. Thank you so much for believing in me. You helped me to challenge my self-doubt and to see this journey through until the end. I am thankful for all the laughter, mischief, and wisdom you brought into my life. You were the best mentor and will always be the best teacher I ever met.

I also want to give special thanks to Philipp Herholz, Misha Kazhdan, and Marc Alexa. Your endless patience, clever hints, and helpful advice made even the most complex topics easy and joyful to understand.

To my colleagues, I cannot stress enough how much I appreciate your unwavering support and camaraderie. During the difficult times of COVID-19 and its quarantine, your presence felt as close as ever, even with the hours of Zoom meetings we had to endure. Whether sharing laughter over virtual coffee chats or offering a listening ear during moments of uncertainty, you made even the darkest days brighter. I want to give a special thanks to Martin Komaritzan for his incredible wisdom, guidance, and helpful words during our time together. With your help, even the smallest worm could grow to new heights.

I want to express my heartfelt gratitude to my family. Your steadfast love and understanding have kept me afloat even when I thought my path seemed uncertain or the workload felt too heavy to bear. I am grateful for your boundless patience and for cheering me on from afar, even when my work consumed me completely. Thank you so much.

Thank you to my friends, whose advice has been invaluable in times of doubt. I appreciate your patience in having the same conversations without ever getting tired. Your unconditional acceptance has brought me immense comfort and happiness.

And to my husband, Tim, who has been my rock through every twist and turn: Your patience, understanding, and unwavering support have carried me through the darkest moments, reminding me that I am never alone.

ABSTRACT

This thesis presents several approaches that generalize the Laplace-Beltrami operator and its closely related gradient and divergence operators to arbitrary polygonal and polyhedral meshes. We start by introducing the *linear virtual refinement method*, which provides a simple yet effective discretization of the Laplacian with the help of the Galerkin method from a Finite Element perspective. Its flexibility allows us to explore alternative numerical schemes in this setting and to derive a second Laplacian, called the *Diamond Laplacian* with a similar approach, but this time combined with the Discrete Duality Finite Volume method. It offers enhanced accuracy but comes at the cost of denser matrices and slightly longer solving times.
In the second part of the thesis, we extend the linear virtual refinement to higher-order discretizations. This method is called the *quadratic virtual refinement method*. It introduces variational quadratic shape functions for arbitrary polygons and polyhedra. We also present a custom multigrid approach to address the computational challenges of higher-order discretizations, making the faster convergence rates and higher accuracy of these polygon shape functions more affordable for the user.
The final part of this thesis focuses on the open degrees of freedom of the linear virtual refinement method. By uncovering connections between our operator and the underlying tessellations, we can enhance the accuracy and stability of our initial method and improve its overall performance. These connections equally allow us to define what a "good" polygon would be in the context of our Laplacian. We present a smoothing approach that alters the shape of the polygons (while retaining the original surface as much as possible) to allow for even better performance.

CONTENTS

1 INTRODUCTION

The discrete Laplace-Beltrami operator, or Laplacian for short, plays a prominent role in geometric modeling and related fields. As a generalization of the second derivative to functions defined on surfaces, it is intimately related to the notion of curvature and signal frequencies. Furthermore, it allows us to solve numerous partial differential equations on discrete surface and volume meshes, which are essential for various computer graphics applications, such as mesh smoothing, parameterization, or fairing. With triangle and tetrahedral meshes being the standard surface and volume representations in computer graphics and geometry processing, the discretization of the Laplace operator for these types of tessellations has received much attention over the years, with the classical cotangent discretization [PP93; MDS+03; DMS+99; Dzi88] being the de-facto standard.

However, while discrete operators for *triangle* meshes are well understood, this is not the case for general *polygon* meshes, despite the growing demand for the latter in many modeling and engineering applications. Recent papers point out that the restriction to triangle or tetrahedral meshes, while simple and convenient, is no longer sufficient. Many users require more general shapes to express geometric properties and features in their models. Applications benefiting from a more flexible range of elements are, for example, fracture modeling [TS08; Bis09; OSTL+12] or linear elasticity problems [TS06]. Lu et al. [LSZ+14] equally made use of polyhedra, by introducing a hollowing optimization algorithm based on the concept of honeycomb-structures. Additionally, one of the necessary steps in their algorithm requires the decomposition of a volume into polyhedral Voronoi cells. In general, structures containing honeycomb geometry have been used in engineering for a long time, since they minimize the required material without losing structural strength [JWW+14]. In the meshing community, as noted by Peng et al. [PPW18], meshing algorithms commonly produce tessellations that incorporate non-triangular elements, such as quad-dominant meshes, mixed tri/quad meshes, or Voronoi-based polygon meshes. Peng et al. further emphasize that these patterns have various applications in architecture, industrial design, and art. Polygon meshes and the differential operators defined on them are also used in the computer animation industry, as they effectively capture geometric features and facilitate both artistic design and fabrication processes [GBD20]. For example, Narayan [Nar23] introduced an approach to generate hair from arbitrary polygon meshes using a generalized Laplace operator.

This progress was only possible since several strategies to extend the Laplacian to general polygon meshes were proposed in recent years [AW11; GBD20;

BB23]. However, this generalization is not without challenges. For instance, arbitrary polygons might not be inherently planar, potentially resulting in twisted surfaces in 3D. In this thesis, we expand those generalizations by proposing a surprisingly simple yet very effective discretization of the polygon Laplace operator, called the "linear virtual refinement method". Inserting a carefully placed vertex for every polygon allows us to define a refined triangulation that retains the symmetries and defines a surface consistent with the polygon mesh. Using the Galerkin method, we then coarsen the cotangent Laplacian defined on the triangulation to obtain a Laplacian on the original polygon mesh, completely hiding the refinement from the user. Furthermore, since the linear virtual refinement method allows us to define shape functions in the finite element context, this thesis will further expand this approach and focus on constructing higher-order basis functions defined on arbitrary polygons and polyhedra.

Additionally, we will adapt the virtual refinement by using an alternative discretization to the cotangent Laplacian, called the Discrete Duality Finite Volume method (DDFV). This approach leads to a new definition of gradient and divergence operators, which can be combined to the so-called *Diamond Laplacian*. We incorporate the primal and dual mesh of the Finite Volume Method (FVM) and accommodate the oblique intersection of primal and corresponding dual elements. Specifically, we define discrete gradients, respectively divergences, per *diamond*: the region spanned by a dual edge and corresponding simplicial primal element.

In the last chapter of this thesis, we will leverage knowledge from FEM on how the shape of triangles affects both the error and the operator's condition. We noticed that shape quality can be encapsulated as the trace of the Laplacian and suggest that trace minimization is a helpful tool to improve numerical behavior. We then apply this observation to the linear virtual refinement method to derive optimal parameters per polygon to enhance both the accuracy and stability of the resulting Laplacian.

In order to make the reader more familiar with the general topic, we will first repeat the necessary definitions for the Laplacian's construction at the example of triangles, followed by more detailed explanations of the respective polygonal and polyhedral operators and their different ideas in each chapter. We will also discuss the properties a discrete Laplacian should fulfill based on the work presented by Wardetzky et al. [WMK+07] and analyze the introduced operators in this context. The defined Laplacians will all be subjected to various quantitative comparisons throughout each chapter and compared to other state-of-the-art approaches that are well-established in the graphics community. Ultimately, this thesis aims to give the reader the intuition to choose the optimal polygonal Laplacian for their given situation.

The main contributions of this thesis are:

- Two new discrete Laplace operators that can be used on various polygonal and polyhedral meshes.
- The extension of the linear virtual refinement method to higher-order shape functions.
- A thorough analysis of the optimal degrees of freedom for the linear virtual refinement method.

More detailed lists of contributions can be found in each chapter.

This thesis is based on the following publications:

- Astrid Pontzen (née Bunge), Philipp Herholz, Michael Kazhdan, Mario Botsch "Polygon Laplacian Made Simple", *Computer Graphics Forum 39(2), (Proc. Eurographics), 2020.*
- Astrid Pontzen (née Bunge), Mario Botsch, Marc Alexa, "The Diamond Laplace for Polygonal and Polyhedral Meshes" *Computer Graphics Forum 40(5), (Proc. Symp. on Geometry Processing), 2021.*
- Astrid Pontzen (née Bunge), Philipp Herholz, Olga Sorkine-Hornung, Mario Botsch, Michael Kazhdan, "Variational Quadratic Shape Functions for Polygons and Polyhedra", *ACM Transaction on Graphics 41(4), (Proc. ACM SIGGRAPH), 2022.*
- Astrid Pontzen (née Bunge) and Mario Botsch, "A Survey on Discrete Laplacians for General Polygonal Meshes", *Computer Graphics Forum 42(2), 2023.*
- Astrid Pontzen (née Bunge), Dennis R. Bukenberger, Sven D. Wagner, Marc Alexa, Mario Botsch, "Polygon Laplacian Made Robust", *Computer Graphics Forum 43(2), (Proc. Eurographics), 2024.*

The source code for *all* the individual operators and experiments presented in this thesis can be found here:

- https://github.com/mbotsch/polyLaplace
- https://github.com/mkazhdan/VariationalPolyShapeFunctions

2 FUNDAMENTALS

In this chapter, we present fundamental concepts that provide the theoretical groundwork for this thesis. We start by defining key notations and concepts concerning meshes and the discrete Laplace-Beltrami operator. Following this, we introduce the Finite Element Method (FEM) at the example of the cotangent Laplacian. Lastly, we offer a brief overview and introduction of three existing polygon Laplacians. These three discretizations will serve as the primary benchmark methods for evaluating the discrete Laplacians presented in this thesis, using a test framework that will be established at the end of this chapter.

2.1 BASIC DEFINTIONS

Consider a 2D polygon mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ embedded in 3D, with vertices $\mathcal{V} = \{v_1, \dots, v_m\}$, edges $\mathcal{E}$, and faces $\mathcal{F}$. Each vertex $v_i \in \mathcal{V}$ has an associated 3D position $\mathbf{x}_i = (x_i, y_i, z_i)^\mathsf{T}$ and each face f consists of n_f vertices. We define an additional set of oriented halfedges $\mathcal{H}$, where for each inner edge $e \in \mathcal{E}$ there exist two oppositely oriented halfedges, while each boundary edge has only one. Likewise, a 3D polyhedral mesh has the same structure with only one additional set consisting of the volumetric cells $\mathcal{C}$. The matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times 3}$ refers to the vertex positions of the mesh in its rows.

2.1.1 Properties of a Discrete General Laplace Operator

In the smooth setting the Laplacian of a function f is defined as

$$\Delta f = \operatorname{div} \nabla f. \tag{2.1}$$

We define its discretised equivalent $\mathbf{L} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ as the product of the inverse of a so-called *mass* matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ and *stiffness* matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$:

$$\mathbf{L} = -\mathbf{M}^{-1}\mathbf{S}. \tag{2.2}$$

$\mathbf{L}$ is generally referred to as the strong, pointwise form of the Laplacian and $\mathbf{S}$ is its weak, integrated form. The exact conditions that are imposed on these matrices will be discussed in the next section.
The smooth Laplace-Beltrami operator has a set of key structural properties that each discretization must be able to fulfill. The correlation between these

smooth properties and discrete Laplace operators has been discussed intensively for triangle meshes by Wardetzky et al. [WMK+07] and for tetrahedral meshes by Alexa et al. [AHK+20]. However, these requirements equally hold for general polygon and polyhedral meshes and are therefore important criteria for the numerical quality of a discrete Laplacian. Unfortunately, as pointed out by Wardetzky et al. [WMK+07], most meshes do not allow for Laplacians to satisfy all discrete properties simultaneously. In this section, we will reintroduce them individually to establish characteristics by which the quality of a polygon Laplacian operator can be assessed.
In the continuous setting, consider a single connected manifold Ω, possibly with boundary, that is equipped with a Riemannian metric. We define a function $u : \Omega \rightarrow \mathbb{R}$ and its discrete equivalent $\mathbf{u} \in \mathbb{R}^m$, whose entries are the function values of u sampled at the vertices of the surface mesh $\mathcal{M}$. The strong Laplacian $\mathbf{L} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ defined on $\mathcal{M}$ is given as a $|\mathcal{V}| \times |\mathcal{V}|$ matrix pair $(\mathbf{M}, \mathbf{S})$ consisting of a sparse symmetric matrix $\mathbf{M}$ and a sparse matrix $\mathbf{S}$.

Symmetry Given two functions u and v that are sufficiently smooth and vanish along the boundary of Ω, the smooth Laplacian is self-adjoint with respect to the L^2 inner product of these functions, meaning

$$\langle \Delta u, v\rangle = \langle u, \Delta v\rangle \tag{2.3}$$

with $\langle u, v\rangle = \int_\Omega uv\,\mathrm{d}A$. We therefore request the strong form $\mathbf{L}$ to be a self-adjoint operator with respect to the inner product induced by the symmetric mass matrix $\mathbf{M}$, meaning

$$(\mathbf{Lu})^\mathsf{T}\mathbf{Mv} = \mathbf{u}^\mathsf{T}\mathbf{M}(\mathbf{Lv}) \tag{2.4}$$

$$\Leftrightarrow \quad \mathbf{u}^\mathsf{T}\mathbf{S}^\mathsf{T}\mathbf{v} = \mathbf{u}^\mathsf{T}\mathbf{Sv} \tag{2.5}$$

for any $\mathbf{u}$ and $\mathbf{v}$.

Locality The smooth Laplacian of a function u at a point $\mathbf{p}$ should only depend on the values $u(\mathbf{q})$ of other points $\mathbf{q}$ in an ε-ball around $\mathbf{p}$. This means that the discrete Laplacian should also operate locally in the 1-ring neighborhood of the respective vertex and should not be affected by distant vertices in the mesh.

Linear Precision In the smooth setting, given that u is a linear function, the Laplacian of said function has to be zero in planar regions of the manifold. The discrete equivalent is similar: Given a *planar* mesh $\mathcal{M}$ and any linear function u, we require the strong version of the Laplacian $\mathbf{L}$ to satisfy

$$(\mathbf{Lu})_i = 0 \tag{2.6}$$

for each inner vertex v_i, where $(\cdot)_i$ denotes the i-th entry or row of the vector or matrix within the parenthesis. Alternatively, we can omit the influence of the mass matrix and require the stiffness matrix multiplied with the vertex positions to satisfy

$$(\mathbf{SX})_i = \mathbf{0}. \tag{2.7}$$

Positive Semi-Definiteness and Null Space In the smooth setting, the Dirichlet energy of u on a Surface $\mathcal{S}$, is defined as

$$E_D(u) = \int_{\mathcal{S}} \|\nabla u\|^2 \, \mathrm{d}A \tag{2.8}$$

and has to be greater than or equal to zero. Its discrete version can be expressed with the help of the stiffness matrix as

$$\frac{1}{2}\mathbf{u}^{\mathsf{T}}\mathbf{S}\mathbf{u}. \tag{2.9}$$

Therefore, $\mathbf{S}$ has to be positive semi-definite in order for the energy to remain non-negative. Note that the Laplace-Beltrami itself is a negative semi-definite linear operator, in contrast to the also commonly used Laplace-de Rham operator, which is typically *positive* semi-definite. If applied to scalar functions, both operators are equivalent up to a sign, which makes the "correct" choice for the semi-definiteness of the stiffness matrix a delicate matter. In this thesis, we focus on the Laplace-Beltrami and its definition as the divergence of the gradient. We hence require the stiffness matrix to be positive semi-definite in order to be consistent with Equation (2.2). If an operator is based on the Laplace-de Rham, we omit the negative sign in Equation (2.2) to obtain the positive semi-definite operator. However, the results of these matrices will be negated on our part in order to yield equivalent results to the other discretizations. In this setting, the stiffness matrices are once again positive semi-definite. This will only be relevant for the introduction of the polygon DEC operators in Section 2.3, but we still want to highlight this detail in order to avoid confusion. But in general, after this point, each stiffness matrix within this thesis will be required to be positive semi-definite. A second aspect of this property addresses the kernel of the Laplacian. The smooth Dirichlet energy vanishes for constant functions. Therefore, the kernel of $\mathbf{S}$ has to be one-dimensional as well and can only contain constant functions. If the stiffness matrix can be expressed as

$$(\mathbf{Su})_i = \sum_j w_{ij}(u_i - u_j), \tag{2.10}$$

the discrete Laplacian automatically fulfills this property.

Maximum Principle The smooth maximum principle requires that harmonic functions ($\Delta u = 0$) have no local extremum at interior points of the manifold

Ω. For example, this property assures that approximated solutions of diffusion problems flow from regions with higher potential to regions with lower potential, instead of the other way round. The discrete equivalent can be directly addressed through the entries of the stiffness matrix by the so-called *negative weight* property, which is a sufficient but not necessary condition for the discrete maximum principle. It demands that for each vertex v_i the entries $\mathbf{S}_{ij}$ have to be less than or equal zero if $i \neq j$ (leading to *positive* weights on the diagonals) and at least one entry per row has to be nonzero.

Convergence The convergence property requires that approximate solutions involving the Laplace operator converge to the exact solution of the partial differential equation (PDE) under refinement of the mesh, which was analyzed by Hildebrandt et al. [HPW06] and Wardetzky [War08]. This property will not be proven for the upcoming operators, but analyzed empirically in the respective result sections.

2.2 FINITE ELEMENT METHOD

The finite element method is often used to approximate the solution u to a given PDE on a simplicial mesh with the help of a finite set of basis functions. The exact number of basis functions depends on both the shape of the element and the order of the basis itself. In the linear case, we typically associate an individual shape function $\varphi_i : \mathbb{R}^3 \to \mathbb{R}$ with the vertex v_i and its position $\mathbf{x}_i$, also commonly referred to as node. Now, instead of solving the PDE directly, the objective changes to finding suitable coefficients u_i, $i = 1, \dots |\mathcal{V}|$, that approximate the unknown solution u of the system with

$$u(\mathbf{x}) = \sum_{i=1}^{|\mathcal{V}|} u_i \varphi_i(\mathbf{x}). \tag{2.11}$$

For example, a common problem solved with the finite element method is the Poisson equation $-\Delta u = f$ for a known function f. Given a surface mesh, the discretized PDE leads to a linear system $\mathbf{Su} = \mathbf{f}$ with the stiffness matrix $\mathbf{S}$ being defined as the integrated dot product of the gradients of the basis functions:

$$\mathbf{S}_{ij} = \int_{\mathcal{M}} \langle \nabla \varphi_i, \nabla \varphi_j \rangle. \tag{2.12}$$

While a variety of different bases can be used, we focus on the linear nodal shape functions that are defined piecewise per face and satisfy the so-called Lagrange interpolation property:

$$\varphi_i(\mathbf{x}_j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.13}$$

Now, consider a polygon face $f \in \mathcal{F}$ with n_f vertices. We want the chosen basis to satisfy additional properties within each respective element of the mesh in order to guarantee convergence under refinement [Hug12]:

1. They have to be C^1 continuous within f and C^0 across its boundaries.
2. For the basis to satisfy *constant precision*, meaning that they are able to exactly reproduce constant functions, the φ_i have to form a partition of unity
$$\sum_{i=1}^{n_f} \varphi_i(\mathbf{x}) = 1. \tag{2.14}$$
3. They have to fulfill the linear reproduction property
$$\sum_{i=1}^{n_f} \varphi_i(\mathbf{x})\mathbf{x}_i = \mathbf{x} \tag{2.15}$$
on planar polyons.

The linear Lagrange basis functions meet all these requirements on triangle meshes. They are also known as barycentric coordinates, $P1$ elements or"hat" basis functions, referring to their shape. We will use them in the next section to define the well known cotangent Laplacian.

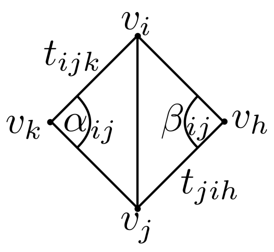
2.2.1 Cotangent Laplacian on Triangle Meshes

Consider a triangle mesh $\mathcal{M}_T = (\mathcal{V}, \mathcal{T})$ defined by a set of vertices $\mathcal{V} = \{v_1, \ldots, v_m\}$ and triangle faces $\mathcal{T}$. Let $\{\psi_1, \ldots, \psi_m\}$ be the piecewise linear Lagrange basis functions defined on $\mathcal{M}_T$. The respective triangle mass and stiffness matrices $\mathbf{M}^{\triangle}, \mathbf{S}^{\triangle} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ of the cotangent Laplace operator $\mathbf{L}^{\triangle}$ are then discretized as

$$\mathbf{M}^{\triangle}_{ij} = \int_{\mathcal{M}_T} \psi_i \, \psi_j = \begin{cases} \dfrac{|t_{ijk}| + |t_{jih}|}{12} & \text{if } v_j \in \mathcal{N}(v_i) \\ \sum\limits_{v_k \in \mathcal{N}(v_i)} \mathbf{M}^{\triangle}_{ik} & \text{if } j = i, \\ 0 & \text{otherwise,} \end{cases} \tag{2.16}$$

$$\mathbf{S}^{\triangle}_{ij} = \int_{\mathcal{M}_T} \langle \nabla\psi_i, \nabla\psi_j \rangle = \begin{cases} -\dfrac{\cot\alpha_{ij} + \cot\beta_{ij}}{2} & \text{if } j \in \mathcal{N}(v_i)\,, \\ -\sum\limits_{v_k \in \mathcal{N}(v_i)} \mathbf{S}^{\triangle}_{ik} & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

Here, the triangles t_{ijk} and t_{jih} are adjacent along the edge $e_{ij} = (v_i, v_j)$ and their areas are denoted by $|t_{ijk}|$ and $|t_{jih}|$. The respective interior angles



of t_{ijk} and t_{jih} opposite to the edge e_{ij} are given by α_{ij} and β_{ij} (see inset), while $\mathcal{N}(v_i)$ denotes the one-ring vertex neighborhood around v_i. The cotangent of the corner angle θ_k at vertex v_k within a triangle t_{ijk} is given as the following ratio between squared triangle edge lengths and triangle area

$$\cot\theta_k = \frac{\left|e_{jk}\right|^2 + \left|e_{ik}\right|^2 - \left|e_{ij}\right|^2}{4\left|t_{ijk}\right|}. \tag{2.18}$$

The scalar $\left|e_{ij}\right|$ denotes the length of edge e_{ij}. As already mentioned, our definition of the cotangent stiffness matrix is *positive* semi-definite in contrast to the *negative* semi-definite Laplacian matrix $\mathbf{L}^{\triangle}$ (see Equation (2.2)). This has the consequence that all eigenvalues of S are non-negative, which makes them more convenient to handle and will become relevant in upcoming chapters.
The key principle of FEM is to divide a large system into smaller, easier to handle *finite elements*. In this spirit, we can obtain the same matrices as defined in Equations (2.16) and (2.17) by constructing local mass and stiffness matrices $\mathbf{M}_t^{\triangle}, \mathbf{S}_t^{\triangle} \in \mathbb{R}^{3\times 3}$ per triangle $t \in \mathcal{T}$:

$$\left(\mathbf{M}_t^{\triangle}\right)_{ij} = \int_t \psi_i \, \psi_j, \tag{2.19}$$

$$\left(\mathbf{S}_t^{\triangle}\right)_{ij} = \int_t \left\langle \nabla\psi_i, \nabla\psi_j \right\rangle. \tag{2.20}$$

These local matrices are then assembled into $\mathbf{M}^{\triangle}$ and $\mathbf{S}^{\triangle}$ by assigning each vertex v_i the i-th row and column of the global matrix. The entries are then the sum over the respective values in the local triangle matrices in which the vertex v_i was involved.
Emulating the continuous setting with the Laplacian being defined as the divergence of the gradient, one can express the gradient operator on triangle meshes as a matrix $\mathbf{G}^{\triangle} \in \mathbb{R}^{3|\mathcal{T}|\times|\mathcal{V}|}$ consisting of local sub-matrices $\mathbf{G}_i^{\triangle} \in \mathbb{R}^{3\times 3}$ per i-th triangle $t = t_{jkl}$. Each column of $\mathbf{G}_i^{\triangle}$ is associated with the gradient of one of the respective vertices. For example, the first column referring to vertex v_j, would be

$$\mathbf{G}_i^{\triangle}(:,1) = \frac{(\mathbf{x}_l - \mathbf{x}_k)^{\perp}}{2\left|t_{jkl}\right|}, \tag{2.21}$$

where $\perp$ denotes a counterclockwise rotation by 90 deg in the triangle plane. The global matrix $\mathbf{G}^{\triangle}$ is then assembled by placing the respective face gradients at the column entries of the individual vertices v_j and setting everything else to zero. This can further be used to discretize the divergence as

$$\mathbf{D}^{\triangle} = \left(\mathbf{G}^{\triangle}\right)^{\mathsf{T}} \hat{\mathbf{M}}, \tag{2.22}$$

with $\hat{\mathbf{M}} \in \mathbb{R}^{3|\mathcal{T}|\times 3|\mathcal{T}|}$ being the diagonal mass matrix containing the area of the i-th triangle in the three associated consecutive diagonal entries [BSP+06].

The product of $\mathbf{D}^{\triangle}$ and $\mathbf{G}^{\triangle}$ gives us the stiffness matrix $\mathbf{S}^{\triangle}$, which is consistent with the continuous setting, but requires a concrete embedding of the mesh in contrast to the intrinsic formulation of $\mathbf{S}^{\triangle}$ itself [Sha21].

2.2.2 Cotangent Laplacian on Tetrahedral Meshes

In the volumetric case, given a tetrahedral mesh $\mathcal{M}$, the stiffness and mass matrices are discretized similarly to Equation (2.16) and Equation (2.17), but with the difference that the volumetric linear Lagrange basis is used instead. This leads to the 3D stiffness matrix

$$\mathbf{S}^{\triangle}_{ij} = \begin{cases} -\frac{1}{6}\sum\limits_{t_{ijkl}} |e_{kl}| \cot \theta^{ij}_{kl} & \text{if } j \in \mathcal{N}(i), \\ -\sum_{k\in\mathcal{N}(i)} \mathbf{S}^{\triangle}_{ik} & \text{if } j = i, \\ 0 & \text{otherwise,} \end{cases} \tag{2.23}$$

The sum is taken over all thetrahedra t_{ijkl} that include the edge connecting the vertices v_i and v_j. The angle θ^{ij}_{kl} is the respective interior angle between the adjacent triangles t_{ikl} and t_{jkl} [Cra19].
A commonly used alternative to the volumetric version of Equation (2.16) is the (lumped) diagonal mass matrix $\mathbf{M}^{\triangle}$ [AHK+20; Cra19] with

$$\mathbf{M}^{\triangle}_{ii} = \frac{1}{4}\sum_{t_{ijkl}} \left|t_{ijkl}\right|. \tag{2.24}$$

The sum is taken over all tetrahedra containing the vertex v_i and $\left|t_{ijkl}\right|$ denotes the volume of the tetrahedron t_{ijkl}.
The discrete local gradient operator for the n-th tetrahedron t_{ijkl} within the mesh can be defined as follows [AHK+20]: Given a vertex v_i within the tetrahedron, let $\mathbf{n}_{jkl}$ be the face normal of the opposite triangle t_{jkl}. The local gradient matrix $\mathbf{G}_n \in \mathbb{R}^{3\times 4}$ can then be constructed column-wise for each vertex within the element. For example, with the first column referring to vertex v_i, it would be defined as

$$\mathbf{G}^{\triangle}_n(:,1) = \frac{\left|t_{jkl}\right|}{3\left|t_{ijkl}\right|}\mathbf{n}_{jkl}. \tag{2.25}$$

The local sub-gradients are then assembled into the global gradient operator $\mathbf{G}^{\triangle} \in \mathbb{R}^{3|\mathcal{C}|\times|\mathcal{V}|}$. Similar to Equation (2.22), the volumetric divergence operator $\mathbf{D}^{\triangle} \in \mathbb{R}^{|\mathcal{V}|\times 3|\mathcal{C}|}$ is then given by

$$\mathbf{D}^{\triangle} = \left(\mathbf{G}^{\triangle}\right)^{\mathsf{T}} \hat{\mathbf{V}}, \tag{2.26}$$

with $\hat{\mathbf{V}} \in \mathbb{R}^{3|\mathcal{C}|\times 3|\mathcal{C}|}$ being the diagonal mass matrix containing the volumes of the tetrahedron i in the three consecutive diagonal entries associated with cell.

2.2.3 FEM on Polygonal and Polyhedral Meshes

For general polygons, there exist a variety of *generalized* barycentric coordinates (GBC) [Flo03; JSW05; JMD+07; HS08; Bis14], which are based on the idea to express any point within the polygon as the weighted sum over its boundary nodes. This defines local shape functions that can be used in the finite element analysis. Extensive surveys [Flo15; CG16] have already discussed the benefits and properties of these shape functions, which were also incorporated in polyhedral finite element methods [MRS14] for volume meshes. We will compare the operators presented in this thesis against one representative set of polygon shape functions, called the harmonic coordinates [JMD+07]. While other methods like the maximum entropy coordinates [HS08] are very present in the FEM analysis on polytopes, we still chose the harmonic shape functions due to their numerous natural mathematical properties that make them so well suited for FEM. This includes smoothness, non-negativity, the mean-value property, and minimization of the Dirichlet energy [MKB+08; CG16]. They can also be defined on arbitrary convex and non-convex polygons and polyhedra [Bis14]. The only real drawback of these shape functions is the lack of a closed form, which requires costly numerical integration. Furthermore, they are only defined on planar elements. Therefore, if an upcoming polygon mesh has non-planar faces, we will consider its planar projection to construct the shape functions. The harmonic coordinates used in this thesis are based on the work of Martin et al. [MKB+08].

2.3 POLYGON LAPLACIANS BASED ON DISCRETE EXTERIOR CALCULUS

The Mimetic Finite Difference method (MFD) [LMS14] is an approximation strategy whose main goal is to define discrete differential operators that try to preserve, or *mimic*, certain critical mathematical and physical properties of an underlying PDE. Its core principle lies in the definition of a so-called *primary* operator, typically gradient, divergence or curl, based on discrete vector and tensor calculus and various forms of Stokes' theorem. The other operators are then derived by using discrete analogs of Green's formulas in order to retain a duality relationship to the primary term. Several authors (e.g. [BLS05; BLS+07]) applied the MFD method to derive mimetic discretizations on polygonal and polyhedral meshes and stressed that one of the key components is the definition of an accurate mimetic inner product matrix. This matrix is a vital part in some derivations of the discrete Laplacian. Although the MFD is not directly focused on the construction of this operator, its theory influenced two polygon Laplacians that are well known in the graphics community. This section will shortly introduce both methods to establish a basic understanding of their core principles. Both operators, besides the discrete harmonic coordinates, will then be used as a main comparison for the upcoming polygon Laplacians introduced in this thesis. As for their construction, both methods focus on a local approach that builds the required matrices per face. We therefore define preliminary:

- $\mathbf{E}_f = (\mathbf{e}_1^f, \dots, \mathbf{e}_{n_f}^f)^\mathsf{T}$ is the $n_f \times 3$ matrix containing in its rows the cyclically ordered edge vectors $\mathbf{e}_i^f = \mathbf{x}_{(i+1) \bmod n_f}^f - \mathbf{x}_i^f$ of the face f.
- $\mathbf{B}_f = (\mathbf{b}_1^f, \dots, \mathbf{b}_{n_f}^f)^\mathsf{T}$ is the $n_f \times 3$ matrix containing in its rows the barycenters $\mathbf{b}_i^f = \frac{1}{2}\left(\mathbf{x}_{(i+1) \bmod n_f}^f + \mathbf{x}_i^f\right)$ of each edge $\mathbf{e}_i^f$.

2.3.1 Algebraic Polygon Laplacian

The operator introduced by Alexa and Wardetzky [AW11] relies on an algebraic approach and extends the MFD-based inner product stabilization [BLS05] to two-dimensional manifolds that even allow for non-planar polygons. Given a polygon surface mesh $\mathcal{M}$ embedded in 3D, the only restrictions are that it has to be *oriented*, meaning that two adjacent faces have to be oppositely oriented on their shared edge, and that the faces are *simple*, meaning that they are not self-intersecting and have boundaries that form a closed loop. The operator is closely tied to the principles involved in Discrete Exterior Calculus (DEC) derivations of the discrete Laplacian.

Algebraic Framework Let $\Gamma^k,\ k \in \{0, 1\}$, be the linear function space of discrete k-forms on $\mathcal{M}$. A k-form can be thought of as a function that takes in k-surfaces and assigns them their integrated value as output, with a 0-surface being a node, a 1-surface an edge, a 2-surface a face and so on. Alexa and Wardetzky derive their polygon Laplacian for 0-forms from the Laplace-de Rahm operator, which for a scalar-valued function u is defined as

$$\Delta u = \mathrm{d}^* \mathrm{d} u. \tag{2.27}$$

In this context $\mathrm{d} : \Gamma^0 \to \Gamma^1$ is the exterior derivative and $\mathrm{d}^* : \Gamma^1 \to \Gamma^0$ the codifferential, which is defined as the adjoint of d with respect to the square integrable inner product [Ros97]. They use the so-called *coboundary* operator as a discrete version of the smooth exterior derivative, with

$$(\mathrm{d}u)(h_{ij}) = u(j) - u(i) \tag{2.28}$$

and h_{ij} being the oriented halfedge from vertex v_i to v_j. The definition of a suitable adjoint operator d^* requires inner products on the k-form function spaces and is therefore, in contrast to the exterior derivative, metric dependent. The inner products can be expressed as two symmetric positive definite matrices $\mathbf{M} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $\mathbf{M}_1 \in \mathbb{R}^{|\mathcal{H}|\times|\mathcal{H}|}$. Any choice of $\mathbf{M}$ and $\mathbf{M}_1$ gives us an expression for the discrete Laplacian

$$\mathbf{L} = \mathrm{d}^*\mathrm{d} = \mathbf{M}^{-1}\mathbf{S}, \tag{2.29}$$

with

$$\mathbf{S} = \mathbf{d}^\mathsf{T}\mathbf{M}_1\mathbf{d}, \tag{2.30}$$

which, negated, becomes equivalent to Equation (2.2). The matrix version of the coboundary operator $\mathbf{d} \in \mathbb{R}^{|\mathcal{H}|\times|\mathcal{V}|}$ is often referred to as the difference operator. Its k-th row associated with the k-th halfedge $h_{ij} \in \mathcal{H}$ can be expressed as

$$\mathbf{d}_{kl} = \begin{cases} -1 & l = i, \\ 1 & l = j, \\ 0 & \text{otherwise,} \end{cases} \tag{2.31}$$

which is only non-zero for the entries $\mathbf{d}_{ki}$ and $\mathbf{d}_{kj}$ associated with the vertices connected by the halfedge.

Choice Of Inner Product Matrices Although in theory any choice for the two inner product matrices would be feasible, not all of them yield the same quality of results. Alexa and Wardetzky therefore motivate their chosen construction by fulfilling the desired criteria discussed in Section 2.1.1. The inner product matrix for 0-forms assigns each vertex a certain mass. In order to retain locality, the matrix $\mathbf{M}$ is given by

$$\mathbf{M}_{ii} = \sum_{f \ni v_i} \frac{|f|}{n_f}, \tag{2.32}$$

where $|f|$ denotes the magnitude of the polygons' vector area. As already mentioned, we also consider non-planar polygons in $\mathbb{R}^3$ that do not necessarily define a surface. Therefore, the vector area

$$|f| = \left\| \frac{1}{2} \sum_{v_i \in f} \mathbf{x}_i \times \mathbf{x}_{(i+1) \bmod n_f} \right\|. \tag{2.33}$$

is defined as the area of the largest orthogonal projection of the polygon onto a plane.

We will first look at the definition of the inner product for 1-forms from a local perspective per face and then later assemble the individual matrices into the global representation, since the process can be repeated per element $f \in \mathcal{F}$. The starting point for the construction is the matrix $\tilde{\mathbf{M}}_f \in \mathbb{R}^{n_f \times n_f}$ given by

$$\tilde{\mathbf{M}}_f = \frac{1}{|f|}\mathbf{B}_f\mathbf{B}_f^\mathsf{T}, \tag{2.34}$$

which was previously defined by Brezzi et al. [BLS05] and is motivated by the Laplacian's connection to mean curvatures. While this choice of the inner product matrix is generally positive semi-definite, in order for the Laplacian itself to fulfill this property, the inner products have to be *positive* definite. Alexa and Wardetzky therefore add a stabilization term to extend Brezzi et al.'s definition to non-planar polygons and give rise to a positive definite inner product

$$\mathbf{M}_f := \tilde{\mathbf{M}}_f + \mathbf{C}_{\bar{f}}\mathbf{U}_{\bar{f}}\mathbf{C}_{\bar{f}}^\mathsf{T}. \tag{2.35}$$

Here, $\bar{f}$ is the maximum orthogonal projection of the polygon f and $\mathbf{C}_{\bar{f}} \in \mathbb{R}^{n_f \times (n_f - 2)}$ is a matrix whose columns span the kernel of $\mathbf{E}_{\bar{f}}^\mathsf{T}$. Combined with any choice of a symmetric positive definite matrix $\mathbf{U}_{\bar{f}} \in \mathbb{R}^{(n_f-2)\times(n_f-2)}$, the stabilization term will lead to a positive definite inner product $\mathbf{M}_f$. Using a parameter $0 < \lambda \in \mathbb{R}$, Alexa and Wardetzky define the matrix $\mathbf{U}_{\bar{f}}$ as

$$\mathbf{U}_{\bar{f}} := \lambda \mathbf{I}_f, \tag{2.36}$$

with $\mathbf{I}_f$ being the $(n_f - 2)$-dimensional identity matrix. Additionally, they choose $\mathbf{C}_{\bar{f}}$ such that its columns are orthonormal, and the final inner product leads to a per-face Laplacian stiffness matrix

$$\mathbf{S}_f = \mathbf{d}_f^\mathsf{T}\mathbf{M}_f\mathbf{d}_f \tag{2.37}$$

that is not affected by scaling, is local and linearly precise. These local matrices are then assembled into the global stiffness matrix $\mathbf{S}$ by assigning each vertex v_i the i-th row and column of $\mathbf{S}$ in which the sum over their respective entries in the local matrices are collected.

2.3.2 Geometric Polygon Laplacian

Alexa and Wardetzky's focus lies solely on the definition of the discrete Laplacian and did not further investigate other operators. This was later addressed by de Goes et al. [GBD20], who defined a variety of discrete differential polygon operators that also serve as a generalization of the MFD, but with a stabilization term for the inner product matrix on 1-forms inspired by the virtual element method (VEM) [VBM13]. The main focus of de Goes et al. [GBD20] was a new linearly precise discretization of the gradient, which allows to define a consistent set of operators, including their own interpretation of the Laplacian.

Polygon Gradient As in the previous section, the definition of the gradient will be applied locally per polygon $f \in \mathcal{F}$, but can be assembled into a global gradient matrix $\mathbf{G} \in \mathbb{R}^{3|\mathcal{F}|\times|\mathcal{V}|}$ acting on the complete mesh. Given a scalar function u defined on f, we want to find a matrix $\mathbf{G}_f$ that simulates the behavior of the gradient ∇u on the polygon. For planar elements, this would normally be achieved by applying Stokes' theorem to ∇u and deriving a matrix discretization through the weak form of the resulting boundary integral. However, since the polygons of the given mesh are not necessarily planar it is not clear how to define the surface normal $\mathbf{n}(\mathbf{x})$ at the boundary points $\mathbf{x}$. Therefore, the standard approach cannot be used. De Goes et al. [GBD20] circumvent this problem by evaluating the co-gradient operator

$$\nabla u^{\perp}(\mathbf{x}) := \mathbf{n}(\mathbf{x}) \times \nabla u(\mathbf{x}), \tag{2.38}$$

on which applying Stoke's theorem leads to

$$\iint_f \nabla u^{\perp}(\mathbf{x}) \, \mathrm{d}A = \oint_{\partial f} u(\mathbf{x})\mathbf{t}(\mathbf{x}) \, \mathrm{d}\mathbf{x}, \tag{2.39}$$

with $\mathbf{t}(\mathbf{x})$ being the unit tangent vector at boundary point $\mathbf{x}$. This expression is independent of the surface of the polygon and only requires the tangent vectors along the boundary, which are uniquely defined. This leads to de Goes et al.'s definition of the the discrete gradient matrix

$$\mathbf{G}_f = -\frac{1}{|f|}[\mathbf{n}_f]\mathbf{E}_f^{\mathsf{T}}\mathbf{Avg}_f \tag{2.40}$$

per polygon f, which is proven to be linearly precise. The matrix $\mathbf{Avg}_f \in \mathbb{R}^{n_f \times n_f}$ yields the average of consecutive vector entries and $[\mathbf{n}_f] \in \mathbb{R}^{3\times 3}$ describes a skew symmetric matrix that, multiplied with a vector, yields the cross product with the face normal $\mathbf{n}_f$ of the planar projection $\bar{f}$. It can be obtained by normalizing the Darboux vector $\mathbf{a}_f \in \mathbb{R}^3$ of the skew symmetric (3×3) matrix

$$\mathbf{A}_f = \mathbf{E}_f^{\mathsf{T}}\mathbf{B}_f. \tag{2.41}$$

The norm of $\mathbf{a}_f$ is exactly the magnitude of the polygons' vector area, therefore leading to

$$\mathbf{n}_f = \frac{\mathbf{a}_f}{|f|}. \tag{2.42}$$

For a more detailed derivation of Equation (2.40) we refer to the original paper [GBD20] or the survey paper by Bunge and Botsch [BB23].

Flat, Sharp and Projection Operator Based on their definition of the gradient operator, de Goes et al. derive an alternative expression to Alexa's and Wardetzky's choice for the inner product matrix on 1-forms. Involved in the process are their discrete polygon extensions of the so-called *sharp* $\sharp$ and *flat* $\flat$ operators, both discretized as $\mathbf{V}_f^{\sharp} \in \mathbb{R}^{3\times n_f}$ and $\mathbf{V}_f^{\flat} \in \mathbb{R}^{n_f \times 3}$ respectively.

As pointed out in Lemma 2 of the original paper, their sharp operator yields

$$\mathbf{G}_f \mathbf{u}_f = \mathbf{V}_f^\sharp \mathbf{d}_f \mathbf{u}_f \tag{2.43}$$

for any scalar function $\mathbf{u}_f$ and is therefore able to reproduce a discrete version of the smooth relation $\nabla u = (\mathrm{d}u)^\sharp$ between sharp and gradient operator.
The matrix $\mathbf{V}^\flat$ maps a 3D vector to its tangential part and then computes its counter-clockwise circulation along the edges of the polygon, giving us a discrete 1-form. The sharp operator $\mathbf{V}^\sharp$ inversely maps the values of a discrete 1-form defined on the polygon back to a single tangent vector per face. However, in contrast to the continuous setting, the operators defined by de Goes et al. are not the exact inverses of each other. Inspired by the virtual element method [VBM13], they mitigate the effect by defining a so-called *projection* operator

$$\mathbf{P}_f := \mathbf{I}_f - \mathbf{V}_f^\flat \mathbf{V}_f^\sharp \quad \in \mathbb{R}^{n_f \times n_f}, \tag{2.44}$$

$\mathbf{I}_f \in \mathbb{R}^{n_f \times n_f}$ being the identity matrix, that measures the error of $\mathbf{V}_f^\flat$ and $\mathbf{V}_f^\sharp$ being inverse to each other. Basically, by first sharpening a 1-form $\mathbf{g}$ to a tangent vector that is then flattened back to a representative 1-form $\hat{\mathbf{g}}$, the projection operator eliminates the components of $\mathbf{g}$ that would result in a tangent vector after applying $\mathbf{V}_f^\sharp$.

Stiffness Matrix Equipped with these operators, de Goes et al. [GBD20] define their local inner product matrices acting on 1-forms as

$$\mathbf{M}_f := |f| \, \mathbf{V}_f^{\sharp\mathsf{T}} \mathbf{V}_f^\sharp + \lambda \, \mathbf{P}_f^\mathsf{T} \mathbf{P}_f, \tag{2.45}$$

which can be assembled into the global inner product matrix $\mathbf{M}_1$ acting on the whole mesh. The matrix $\mathbf{M}_f$ maps the involved 1-forms to their respective tangential vectors with the help of the sharpening operator $\mathbf{V}_f^\sharp$, resulting in their dot product. The potential rank deficiency is mitigated through the second correction term regulated by a parameter $\lambda > 0$. As for Alexa and Wardetzky, this regularization is necessary to guarantee that the inner product matrix is strictly positive definite, which can then be used as before to define the local discrete Laplace operator

$$\mathbf{S}_f = \mathbf{d}_f^\mathsf{T} \mathbf{M}_f \mathbf{d}_f. \tag{2.46}$$

2.3.3 Numerical Evaluation

In this section, we introduce a number of different quantitative computer graphics applications for both surface and volume meshes. This test framework will allow us to analyze the accuracy and quality of the different operators presented in this thesis in a consistent manner. Furthermore, we preliminarily want to evaluate the influence of the respective parameters λ involved in the introduced methods of Alexa and Wardetzky [AW11] and de Goes et al. [GBD20] and how they affect the quality of the inner product matrix. Therefore, we analyze a selection of values besides the recommended choices of the authors. The overall most accurate parameters will then be used in the upcoming evaluation sections of this thesis. Since both DEC operators are not defined for volume meshes, the introduced volumetric tests will only become relevant for the other operators introduced in later chapters of this thesis. Figure A.1 illustrates a selection of the test meshes used in our evaluation.

Poisson Equations We analyze the convergence behavior of the different Laplacians by solving the Poisson equation $-\Delta u = f$ with Dirichlet boundary conditions on various refined tessellations of the unit square and cube. We use the 2D and 3D Franke test functions [Fra79] for the right hand side f and solve the discrete system

$$\mathbf{S}\mathbf{u} = \mathbf{M}\mathbf{b}, \tag{2.47}$$

with $\mathbf{b} \in \mathbb{R}^{|\mathcal{V}|}$ containing the values of the analytic Laplacian Δf of the respective test function sampled at the vertices. The solution $\mathbf{u} \in \mathbb{R}^{|\mathcal{V}|}$ is then compared to the analytic values of f. The formulas of the Franke test functions are

$$\begin{aligned} f_{2\mathrm{D}}(x,y) &= \frac{3}{4}\mathrm{e}^{-\frac{(9x-2)^2+(9y-2)^2}{4}} + \frac{3}{4}\mathrm{e}^{-\frac{(9x+1)^2}{49}-\frac{9y+1}{10}} \\ &+\frac{1}{2}\mathrm{e}^{-\frac{(9x-7)^2+(9y-3)^2}{4}} - \frac{1}{5}\mathrm{e}^{-(9x-4)^2-(9y-7)^2} \end{aligned} \tag{2.48}$$

and

$$\begin{aligned} f_{3\mathrm{D}}(x,y,z) =& \frac{3}{4}\mathrm{e}^{-\frac{(9x-2)^2+(9y-2)^2+(9z-2)^2}{4}} + \frac{3}{4}\mathrm{e}^{-\frac{(9x+1)^2}{49}-\frac{9y+1}{10}-\frac{9z+1}{10}} \\ &+\frac{1}{2}\mathrm{e}^{-\frac{(9x-7)^2+(9y-3)^2+(9z-5)^2}{4}} \\ &-\frac{1}{5}\mathrm{e}^{-(9x-4)^2-(9y-7)^2-(9z-5)^2}. \end{aligned} \tag{2.49}$$

Figure 2.1 showcases the L_2 error rates, computed as the root-mean-square error (RMSE), from solving the Franke Poisson system on different planar surface meshes. Both DEC Laplacians can reproduce the expected quadratic convergence rate across all tessellations in the planar setting. In terms of accuracy, both operators produce high-quality results for $\lambda = 1$ on quad and Voronoi meshes. On concave surfaces, de Goes et al. with parameter $\lambda = 0.5$ yields the best results and surpasses the previous optimal choice of $\lambda = 1$. We want to highlight that both Laplacians, independent of the chosen hyper-parameter, reproduce the cotan Laplacian on triangulated surfaces.

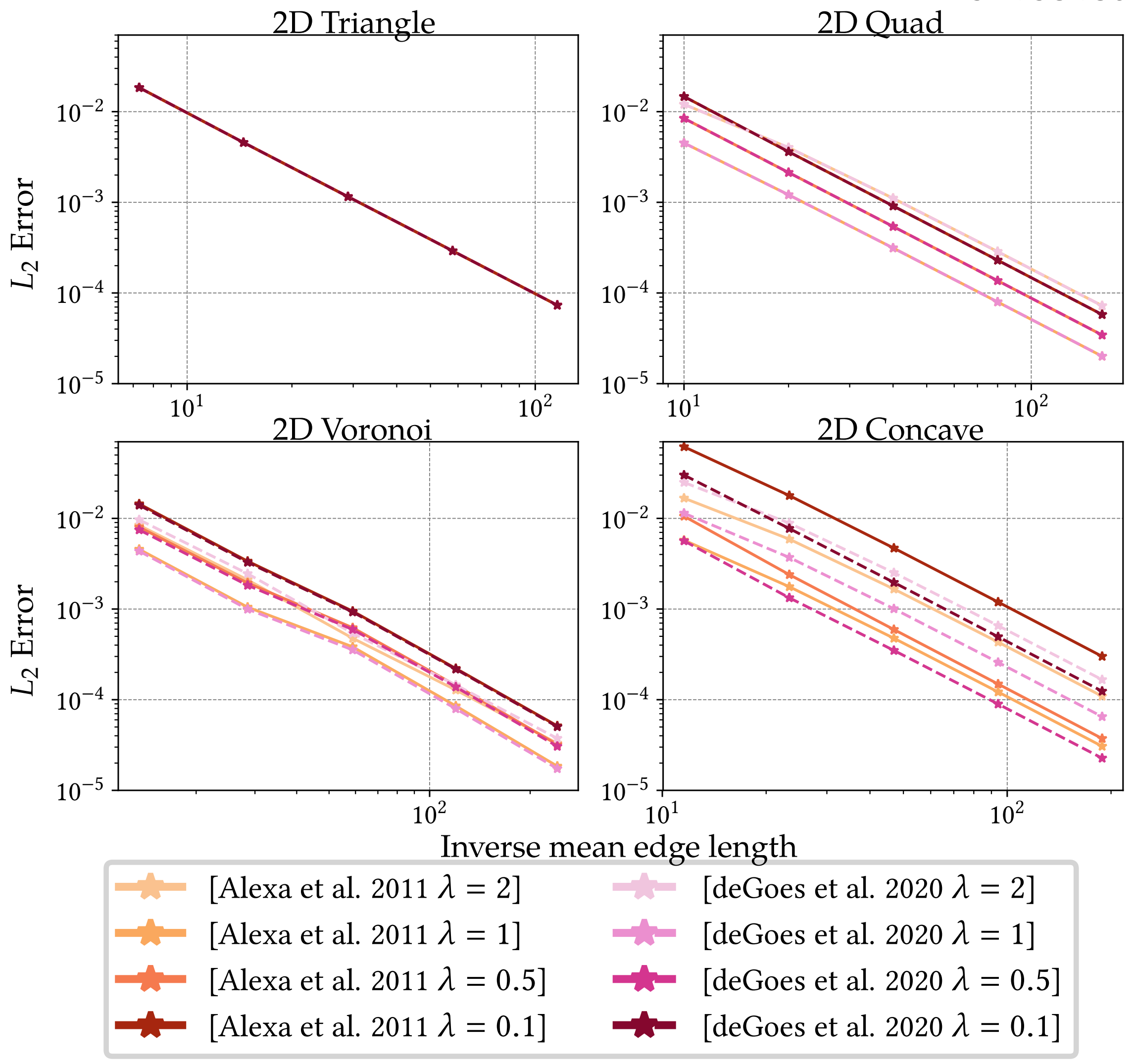


Figure 2.1: *L_2 error in log-log scale of the Poisson system solved for Franke's test function on planar grids. Both DEC Laplacians are equivalent to the cotangent Laplacian on triangles, independent of the chosen hyperparameter. This leads to the overlapping lines for the top left plot.*

Spherical Harmonics The eigenfunctions of the Laplacian on the unit sphere $\mathcal{S}^2$ are called the spherical harmonics $Y_l^m : \mathcal{S}^2 \to \mathbb{R}$ with eigenvalues $-l(l+1)$. Using the fact that Y_l^m are eigenfunctions, we can solve for $\mathbf{u} \in \mathbb{R}^{|\mathcal{V}|}$:

$$\mathbf{u} = \mathbf{M}^{-1}\mathbf{S}\mathbf{y}_l^m \tag{2.50}$$

$$\Leftrightarrow \quad \mathbf{M}\mathbf{u} = \mathbf{S}\mathbf{y}_l^m \tag{2.51}$$

and rescale the solution with the respective eigenvalue. The entries of $\mathbf{y}_l^m \in \mathbb{R}^{|\mathcal{V}|}$ denote the function values of Y_l^m sampled at the vertices. We can measure the error of $\mathbf{u}$ being an eigenfunction to the presented Laplace operators by

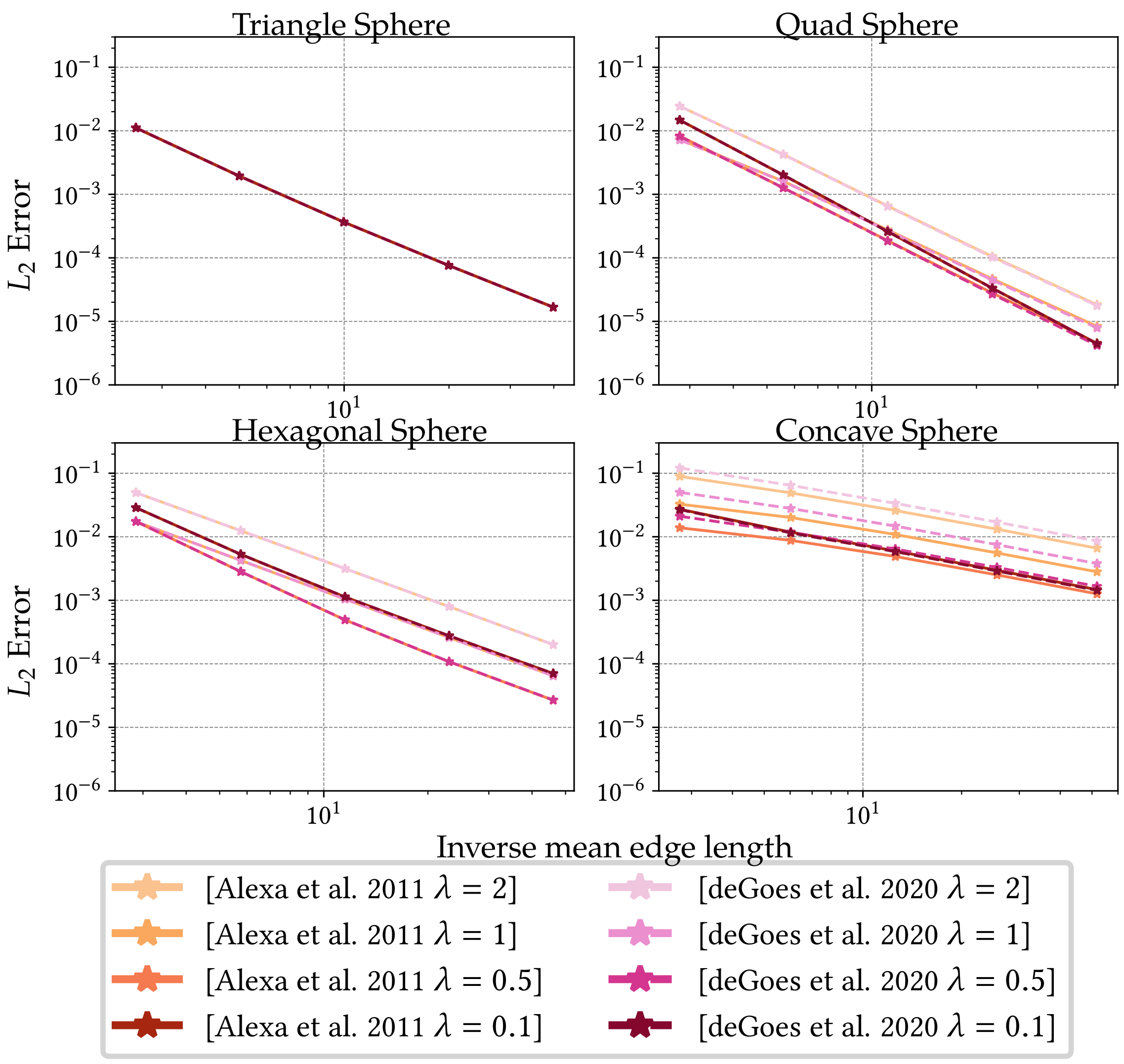


Figure 2.2: *L_2 error in log-log scale for the Poisson solve of the spherical harmonic function Y_3^2 with eigenvalue -12 on different tessellations of the unit sphere.*

evaluating

$$\left\| \mathbf{y}_l^m + \frac{1}{l(l+1)}\mathbf{u} \right\|_{\mathbf{M}}^2 \tag{2.52}$$

for a selected frequency with non-zero eigenvalue. The L^2 norm is computed with respect to the inner product induced by the mass matrix $\mathbf{M}$, which is defined as

$$\|\mathbf{v}\|_{\mathbf{M}}^2 = \mathbf{v}^\top \mathbf{M}\mathbf{v} \tag{2.53}$$

for a vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{V}|}$.

Figure 2.2 displays the deviation of the solution from the analytic function values of

$$Y_3^2(x,y,z) = \frac{1}{4}\sqrt{\frac{105}{\pi}}(x^2 - y^2)z \tag{2.54}$$

with eigenvalue -12. In this setting, choosing lower λ for the stabilization term leads to the most accurate results, with $\lambda = 0.5$ being one of the most consistent options for both operators. For more challenging tessellations like the concave sphere, one can observe that the convergence rate slightly stagnates, but the Laplacians still display a consistent improvement under mesh refinement.

Geodesics in Heat In order to assess the quality of the divergence and gradient operators, we evaluate them in the context of the geodesics in heat method presented by Crane et al. [CWW13]. Given the i-th unit vector $\mathbf{e}_i \in \mathbb{R}^{|\mathcal{V}|}$, we can obtain the geodesic distances from a vertex v_i to all other vertices in the mesh in three steps: First we solve the heat flow with a fixed small time-step ε for the vector $\mathbf{u} \in \mathbb{R}^{|\mathcal{V}|}$:

$$\begin{aligned} &(\mathbf{I} - \varepsilon \mathbf{L})\mathbf{u} = \mathbf{e}_i \\ \Leftrightarrow\ &(\mathbf{I} + \varepsilon \mathbf{M}^{-1}\mathbf{S})\mathbf{u} = \mathbf{e}_i \\ \Leftrightarrow\ &(\mathbf{M} + \varepsilon \mathbf{S})\mathbf{u} = \mathbf{M}\mathbf{e}_i. \end{aligned} \tag{2.55}$$

Then we compute the normalized gradients of the solution vector through

$$\mathbf{g}_j = \frac{(\mathbf{Gu})_j}{\|(\mathbf{Gu})_j\|}. \tag{2.56}$$

In the last step, we solve the Poisson equation

$$\mathbf{Sv} = \mathbf{Dg} \tag{2.57}$$

for the geodesic distances $\mathbf{v} \in \mathbb{R}^{|\mathcal{V}|}$ and shift the solution by the offset of the value associated with vertex v_i to zero. Note that, depending on the employed Laplacian, the number and dimension of the gradient vectors will vary in the upcoming chapters. For example, the methods introduced by Alexa and Wardetzky and de Goes et al. [AW11; GBD20] obtain both three-dimensional gradient vectors per polygon face. Additionally, the normalization step in Equation (2.56) changes depending on the chosen method. While methods with a geometrically motivated gradient can normalize the vectors by their respective Euclidean length, like de Goes et al.[GBD20], the method by Alexa and Wardetzky [AW11] needs an alternative approach. As pointed out by Crane et al. [CWW13], interpreting the coboundary operator $\mathbf{d}$ as gradient leads to discrete 1-forms associated with the halfedges, which cannot be directly normalized. However, since $\mathbf{M}_1$ (see Equation (2.35)) gives us an inner product matrix for 1-forms, they propose to use

$$\|\nabla u\|_f = \sqrt{\frac{\mathbf{u}_f^\mathsf{T}\mathbf{S}_f\mathbf{u}_f}{|f|}} \tag{2.58}$$

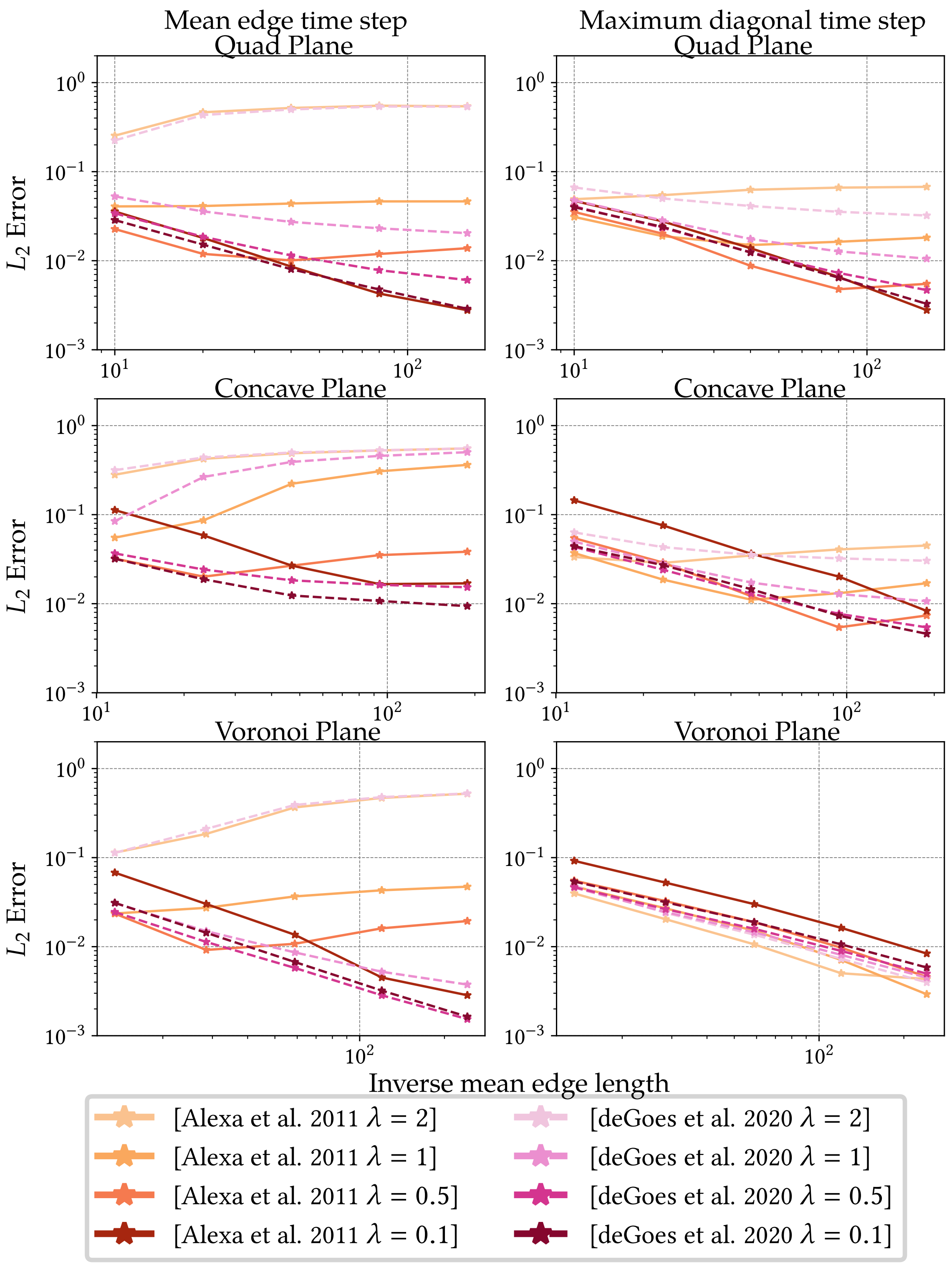


Figure 2.3: *L_2 error in log-log scale of the Geodesics in heat method on planar grids with quads (top), concave polygons (center) and Voronoi faces (bottom).*

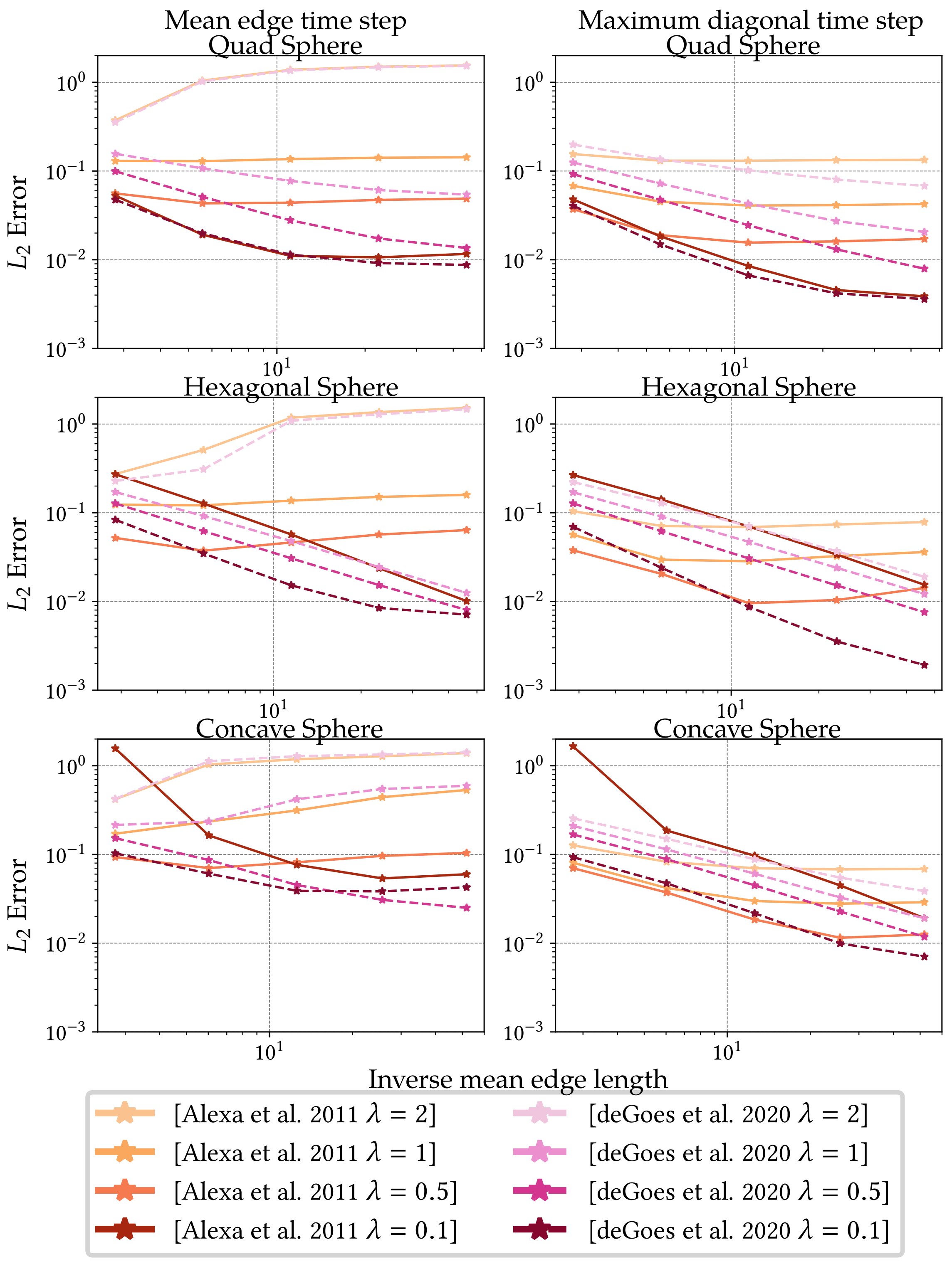


Figure 2.4: *L_2 error in log-log scale of the Geodesics in heat method on unit spheres with quad (top), hexagon (center) and concave faces (bottom).*

as normalization term by assuming that ∇u is constant over each face and therefore

$$\mathbf{u}_f^\mathsf{T} \mathbf{S}_f \mathbf{u}_f = \int_f \|\nabla u\|^2 \, \mathrm{d}A = \|\nabla u\|^2 \, |f| \, . \tag{2.59}$$

The time step ε involved in the first step of the heat method (Equation (2.55)) is a debated subject. As pointed out by Crane et al. [CWW13], the discrete setting does not follow the expected rule that smaller time steps necessarily lead to more accurate results. However, too large time steps lead to a smoothed approximation of the distances. We therefore compare the behavior of the two most common choices:

- The squared mean edge length, as proposed by Crane et al. [CWW13].
- The squared length of the longest face diagonal, as suggested by de Goes et al. [GBD20; GDM+16].

Figure 2.3 and 2.4 show the deviation of the obtained geodesic distances to the Euclidean distance in the plane and the great-circle distance on the unit sphere. In order to provide a less biased evaluation, we compute the L_2 errors for 16^2 (planes) and 8^3 (spheres) uniformly sampled points and then report the mean of these errors. Using the mean edge length as time-step ε leads to more significant error fluctuations for both DEC methods, especially for progressively larger λ. The maximum face diagonal stabilizes these deviations, achieving the most impact for a high hyper-parameter. In general, de Goes et al.'s method for $\lambda = 0.1$ has the lowest error rates, independent of the chosen time step. Additionally, the definition of a geometric gradient operator improves the accuracy of de Goes et al.'s method compared to the algebraic coboundary operator used for Alexa and Wardetzky's Laplacian. However, choosing larger values for λ affects both methods negatively. Still, given that λ controls the influence of the stabilization term for both methods, it cannot be chosen to be indefinitely close to zero since this would lead to Laplacians with too large kernels and, therefore, spurious modes.

In light of the above, we can say that the suggested hyper-parameter from the original paper and the parameter that yielded the best results in our test cases slightly differ for both methods. For Alexa and Wardetzky, this corresponds to $\lambda = 2$ (suggested) and $\lambda = 0.5$ (optimal). In the case of de Goes et al., we consider $\lambda = 1$ (suggested) and $\lambda = 0.1$ (optimal). The results vary depending on the selected test mesh or application, suggesting that alternative hyperparameter configurations tailored to specific mesh types or applications could, of course, further improve the results for the respective DEC operator. However, in the upcoming chapters, we will compare the derived polygon operators to these four specific choices, to provide a slightly less crowded numerical evaluation of the different Laplacians.

3 THE LINEAR VIRTUAL REFINEMENT METHOD

When working with general polygon meshes, the finite element method (Section 2.2) cannot be applied directly for two reasons. First, when the polygon is not planar, it is unclear what the underlying surface is over which functions should be integrated. Second, even for simple planar polygons, it is unclear how to associate basis functions with the mesh's vertices.

A simple approach would be to triangulate all the polygons, define a piecewise linear integration domain, and use the linear Lagrange basis functions to define the finite elements. This would have the advantage of defining a finite elements system whose dimension equals the number of vertices in the input mesh. Unfortunately, introducing diagonal edges would also break the symmetry structure of the polygonization (e.g., in quad meshes where edges are aligned with principal curvature directions).

An alternative approach would be to refine the polygon mesh by introducing a new vertex in the middle of each face. This would preserve the symmetry structure but would come at the cost of an increase in the dimension of the finite elements system. Additionally, as we show in Section 3.3, such an approach can result in poor performance due to the introduction of positive edge weights in the stiffness matrix.

We propose an in-between approach – introducing a *virtual* vertex in the middle of each face, expressed as an affine combination of the face's vertices. Naively, the new vertex produces a refined triangle mesh with a finite elements system given by the hat basis functions, as before. However, we then coarsen the refined system, expressing the vertex functions on the coarse mesh as linear combinations of the linear Lagrange basis functions on the finer one.

This new system has a dimension equal to the number of vertices in the original polygon mesh (preserving the finite elements system dimension). It has the property that basis functions have overlapping support only if the associated vertices lie on a common face (defining a sparse system that preserves the symmetry structure). A further advantage of our approach is that we easily obtain a consistent factorization of the stiffness matrix as the product of divergence and gradient matrices.

Individual Contribution *I developed the linear virtual refinement method in close cooperation with Philipp Herholz, supervised by Mario Botsch and Misha Kazhdan. While the theoretical derivations were a collective effort, I was responsible for implementing the linear virtual refinement method and all quantitative and qualitative evaluations. Additionally, I am responsible for extending the method to polyhedral elements. Misha Kazhdan made the connection between the linear virtual refinement method and* *discrete exterior calculus, and Philipp Herholz provided insights into the relationship between our method and the operator of Alexa and Wardetzky.*

Corresponding Publication *This chapter is based on the following publication:*

Bunge, A., Herholz, P., Botsch, M., and Kazhdan M. (2020). "Polygon Laplacian Made Simple." Computer Graphics Forum, 39(2):303-313.

3.1 RELATED WORK

Laplacians for Triangle Meshes The main goal when constructing a discrete Laplacian is to retain as many properties from the smooth setting as possible. While the classical cotangent Laplace operator [Mac49] is positive semi-definite, symmetric, and has linear precision, it fails to maintain the maximum principle. As a consequence, parametrizations obtained with this discretization can suffer from flipped triangles. In contrast, the combinatorial Laplacian [Tau95; Zha04] guarantees the maximum principle while failing to have linear precision. Bobenko and Springborn [BS07] introduced a discrete Laplacian based on the intrinsic Delaunay triangulation. While guaranteeing the maximum principle, this operator is defined over an intrinsic mesh with different connectivity. Recently, an efficient data structure for the representation of these meshes has been introduced [SSC19a]. The idea of intrinsic triangulations does not extend to the case of non-planar polygon meshes due to the lack of a well defined surface. Other discretizations include the Laplacian of Belkin et al. [BSW08] that provides point-wise convergence and the octree-based Laplacian [CLB+09] defined to support multigrid solvers. In general, Wardetzky et al. [WMK+07] have shown that there cannot exist a discretization that fulfills a certain set of properties for all meshes, explaining the variety of approaches in the literature.

Laplacians for Polygon Meshes There have been a couple of approaches for discretizing differential operators on polygonal meshes. As discussed in Section 2.3, Alexa and Wardetzky [AW11] circumvent the problem that (non-planar) polygons in 3D do not bound a canonical surface patch by considering the projection of the polygon onto the plane that yields the largest projection area. Their polygon Laplace combines an MFD-based inner product [BLS05] with an algebraic kernel stabilization, regulated with a scalar hyper-parameter. A drawback of this discretization is that it requires the suitable choice of a scalar parameter, and that the potentially large number of positive non-diagonal entries in the stiffness matrix violates the discrete maximum principle [WMK+07].
Herholz et al. [HKA15] extend the definition of desirable properties of discrete Laplacians to polygon meshes, where they characterize polygon tessel-

lations that admit a "perfect" operator with only non-negative weights. Sharp et al. [SSC19b] handle potentially non-planar polygons by deriving a version of the "connection" Laplacian, which, in contrast to the standard expression as the divergence of the gradient, is given by the trace of the second covariant derivative. This operator fulfills many of the same properties as the cotan Laplacian and enables the diffusion of vectors from one tangent space to another on curved domains.

De Goes et al. [GBD20] go beyond the Laplace operator and generalize a whole set of discrete differential operators to general polygonal meshes. They extend the approach of [AW11] by defining a new discrete gradient operator, which can be interpreted as a generalization of mimetic finite differences [BLS05] and the virtual element method [VBM13]. Using the weak form of the cogradient operator and the divergence theorem, they bypass the need for an interpolation of the non-planar polygon surface.

Similarly, Ptáčková and Velho [PV21] presented an extended version of discrete exterior calculus for general polygon meshes. They define essential operators like a polygonal wedge product and Hodge star operator, allowing them to derive new versions of the Lie derivative, codifferential, and Laplace-de Rahm operator.

Virtual Refinement in Geometry Processing To extend the cotangent Laplacian to polygon meshes, we refine the mesh by inserting a virtual vertex for each face and using these to tessellate each polygon into a triangle fan. Similar to recent work on Subdivision Exterior Calculus [GDM+16], we define a *prolongation* operator expressing functions on the coarser polygonal mesh as linear combinations of the triangle hat basis functions on the finer mesh. Then, leveraging the Galerkin method [Fle84], we define the Laplacian on the polygon mesh by coarsening the Laplacian from the triangle mesh. As in the method of de Goes et al. [GDM+16], this gives us the benefit of working over a refined triangle mesh (where discrete Laplacians are well understood) without incurring the computational complexity of working on a finer mesh.

Sample Applications Applications of Laplacians include the approximation of conformal parametrizations [DMA02], mesh deformation [SCOL+04], and signal processing on meshes [CRK16], to name a few. Replacing the usual cotangent Laplacian with a Laplacian defined on polygon meshes directly enables many of these algorithms to work in this more general setting. However, the quality of the results depends on the specific construction and properties of the polygon Laplacian. In this chapter we compare different variants with respect to a set of applications including parametrization [Flo97; GGT06], mean curvature flow [DMA02; KSBC12], spectral analysis [LZ10], and geodesics in heat [CWW13].

3.2 POLYGON LAPLACIAN

Our approach for defining a polygon Laplacian proceeds in two steps. First, we refine the polygon mesh to define a triangle mesh on which the standard cotangent Laplacian is defined. Then, to obtain a Laplacian on the initial polygon mesh, we use the Galerkin method to coarsen the cotangent Laplacian, which was previously introduced in Section 2.2.1. In the following we briefly review the Galerkin method [Fle84].

3.2.1 The Galerkin Method

Assume that we are given two finite-dimensional nesting function spaces $F^c \subset F^f$. Given a *prolongation* operator P injecting the coarser space into the finer, $P\colon F^c \hookrightarrow F^f$, and given a symmetric positive semi-definite operator Q defining a quadratic energy on the space of functions, we can define a symmetric positive semi-definite operator on the space F^f through restriction

$$Q^f(\phi, \psi) := Q(\phi, \psi), \qquad \forall \phi, \psi \in F^f. \tag{3.1}$$

In a similar manner, we can define a symmetric positive semi-definite operator Q^c on the space F^c.
The Galerkin method tells us that the operators are related by

$$Q^c = P^* \circ Q^f \circ P, \tag{3.2}$$

where P^* is the dual of P. In particular, given bases for F^c and F^f and letting $\mathbf{P}$, $\mathbf{Q}^c$, and $\mathbf{Q}^f$ be the matrices representing the operators with respect to these bases, we have

$$\mathbf{Q}^c = \mathbf{P}^\mathsf{T} \cdot \mathbf{Q}^f \cdot \mathbf{P}. \tag{3.3}$$

3.2.2 Construction of the Finite Elements System

Given a *polygon* mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, we construct our finite elements system by defining a *refined triangle* mesh $\mathcal{M}_T = (\mathcal{V}_T, \mathcal{T})$. Vertices in the refined mesh, $\mathcal{V}_T$, are obtained by introducing a new vertex, $v_{|\mathcal{V}|+j}$, for every polygon $f_j \in \mathcal{F}$ and setting the position of $v_{|\mathcal{V}|+j}$ to an affine combination of the positions of vertices in f_j

$$\mathbf{x}_{|\mathcal{V}|+j} = \sum_{v_i \in f_j} w_{ji} \mathbf{x}_i, \quad \text{with} \quad \sum_{v_i \in f_j} w_{ji} = 1. \tag{3.4}$$

Triangles in the refined mesh, $\mathcal{T}$, are obtained by replacing each face $f_j \in \mathcal{F}$ with the triangle fan connecting the inserted vertex $v_{|\mathcal{V}|+j}$ to the edges in f_j, as can be seen in Figure 3.1. Using the refined triangle mesh, we can define

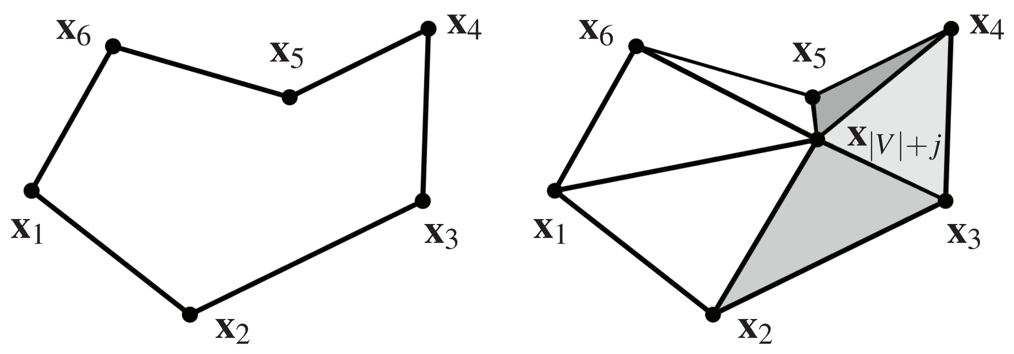


Figure 3.1: *Spanned triangle fan on the virtual mesh after inserting the virtual vertex $\mathbf{x}_{|\mathcal{V}|+j}$ at the j-th face.*

the stiffness matrix, $\mathbf{S}^{\triangle} \in \mathbb{R}^{|\mathcal{V}_T|\times|\mathcal{V}_T|}$, using the standard cotangent weights of Equation (2.17). The affine weights $\mathbf{w}_f = (w_1, \ldots, w_{n_f})$ of each face can be aggregated into a local $(n_f + 1) \times n_f$ matrix

$$\mathbf{P}^f_{ij} = \begin{cases} w_j & \text{for } i = n_f + 1 \\ \delta_{ij} & \text{otherwise,} \end{cases} \tag{3.5}$$

which can be assembled into a global *prolongation* matrix $\mathbf{P} \in \mathbb{R}^{|\mathcal{V}_T|\times|\mathcal{V}|}$, with

$$\mathbf{P}_{ij} = \begin{cases} 1 & \text{if } i = j\,, \\ w_{ki} & \text{if } i = |\mathcal{V}| + k \text{ and } v_j \in f_k\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{3.6}$$

And finally, using the Galerkin method, we obtain an expression for the coarsened polygonal stiffness matrix as

$$\mathbf{S}^{\pentagon} = \mathbf{P}^{\mathsf{T}}\mathbf{S}^{\triangle}\mathbf{P}. \tag{3.7}$$

We use the same approach to obtain the coarsened polygonal mass matrix

$$\mathbf{M}^{\pentagon} = \mathbf{P}^{\mathsf{T}}\mathbf{M}^{\triangle}\mathbf{P}, \tag{3.8}$$

but in addition to restricting the refined triangle mass matrix $\mathbf{M}^{\triangle}$, we also lump it to a diagonal matrix:

$$\mathbf{M}^{\pentagon} = \text{lump}\left(\mathbf{P}^{\mathsf{T}}\mathbf{M}^{\triangle}\mathbf{P}\right) \quad \text{with} \tag{3.9}$$

$$\text{lump}(\mathbf{M}^{\pentagon})_{ij} = \begin{cases} \sum_k \mathbf{M}^{\pentagon}_{ik} & \text{if } i = j\,, \\ 0 & \text{otherwise}\,, \end{cases} \tag{3.10}$$

setting the diagonal entry to the sum of the entries in the row.
Note that our stiffness matrix has non-zero entries if the corresponding vertices share a face, not just an edge. Also note that solving a linear system using our

operators differs from solving the system on the refined mesh and only using the solution values at the original (i.e., non-virtual) vertices. Our approach corresponds to solving the system using a coarser function space. The alternative corresponds to sub-sampling a solution obtained over a refined function space, which could result in aliasing.

3.2.3 Choice of Virtual Vertex Position

Choosing different virtual vertex positions, we provide a framework for defining a whole family of polygon Laplace operators. While many choices are possible we would like the virtual vertex to fulfill a set of properties in order to yield a geometric Laplacian:

- The vertex should be efficient to compute and uniquely defined.
- Since the choice of virtual vertex defines the surface of the polygon, we would like to find a point giving small surface area.
- For planar polygons, the vertex should be inside the polygon.
- To achieve linear precision, the virtual vertex should be an affine combination of polygon vertices (see Equation (3.4)).

A straightforward choice is to use the *centroid* of the polygon vertices. However, for non-convex (planar) polygons this point can be located outside the polygon. Moreover, it will be biased by an uneven distribution of polygon vertices.

Another choice is to find a point *minimizing the area* of the induced triangle fan, which is related to the minimization of the Dirichlet energy. Unfortunately, this point is not uniquely defined. For example, for convex planar polygons, the total area will be identical for every virtual vertex inside the polygon. Furthermore, since area is non-linear in the position of the virtual vertex, less efficient iterative solvers, e.g. Newton's method, are required to compute the position of the virtual vertex.

Instead, we opt for the minimizer of the sum of *squared triangle areas* of the induced triangle fan. Introducing a virtual vertex with position $\mathbf{x}_f$ into an n_f-gon with vertex positions $(\mathbf{x}_1, \ldots, \mathbf{x}_{n_f})$ allows us to define the triangle fan with n_f triangles $(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_f)$, where indices are interpreted modulo n_f. The position of the virtual vertex, $\mathbf{x}_f \in \mathbb{R}^3$, is then defined as the minimizer

$$\mathbf{x}_f = \arg\min_{\mathbf{x}} \sum_{i=1}^{n_f} \operatorname{area}(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x})^2 . \tag{3.11}$$

Compared to the minimizer of the area, the squared area minimizer has two advantages. First, the solution is unique even for planar convex polygons. Second, using the squared area, the objective function becomes quadratic in $\mathbf{x}_f$. Also, in contrast to the centroid, the minimizer of the squared area is almost always positioned in the interior of a planar, star-shaped polygon, even when the polygon is not convex.

While the position of our virtual vertex is unique, its expression as the linear combination of polygon vertices usually is not. Since linear precision requires $\mathbf{x}_f$ to be an affine combination of the polygon's vertices (see Section 3.2.8), finding the position of the virtual vertex can be formulated as a minimization directly over the weights $\mathbf{w} = (w_1, \ldots, w_{n_f})^\mathsf{T}$:

$$\begin{aligned} \mathbf{w}_f \; &= \; \arg\min_{\mathbf{w}} \sum_{i=1}^{n_f} \operatorname{area}\left(\mathbf{x}_i, \mathbf{x}_{i+1}, \sum_{j=1}^{n_f} w_j \mathbf{x}_j \right)^2 \\ &\text{such that} \sum_{j=1}^{n_f} w_j = 1. \end{aligned} \tag{3.12}$$

Noting that when $n_f > 3$ the system is under-constrained (with multiple affine weights defining the same unique squared-area minimizer $\mathbf{x}_f$), we add the constraint that, of all the weights $\mathbf{w}$ defining the minimizing position $\mathbf{x}_f$, we prefer the one with minimal L_2-norm $\|\mathbf{w}\|$. This encourages the weights to be as uniform as possible, allowing each polygon vertex to contribute equally. These affine weights can be obtained by solving a single linear system, derived in Appendix A.2.

It is tempting to add non-negativity constraints $w_j \geq 0$ to Equation (3.12) since this would yield *convex weights* that guarantee a maximum principle for the virtual vertex and prevent positive coefficients in the coarsened stiffness matrix $\mathbf{S}^{\bigcirc}$ (see Equation (3.7)) as long as they do not appear in $\mathbf{S}^{\triangle}$. However, this comes at the cost of solving a quadratic program with inequality constraints.

We compare our choice of virtual vertex—minimizing the sum of squared triangle areas through affine weights—to the other options (centroid, min. area, convex weights) in Section 3.3.2.

3.2.4 Implementation

Implementing our system is simple, especially if code for the computation of the cotangent Laplacian is already available. Our implementation is based on the PMP-Library [SB19]. Algorithm 1 illustrates the procedure. Initially we set the prolongation matrix $\mathbf{P}$ to the identity of size $|\mathcal{V}| \times |\mathcal{V}|$ (line 3), reflecting the fact that all original vertices appear in the refined mesh. We then loop over all faces, collecting all vertex positions of the polygon, finding affine weights for the optimal virtual vertex, and constructing this point (lines 5–7, Section 3.2.3). Next we compute the area and cotangent weights for the resulting triangle fan and aggregate them in $\mathbf{M}^{\triangle}$ and $\mathbf{S}^{\triangle}$ (lines 9–10, Section 2.2.1). The final operators are constructed by multiplying the matrices $\mathbf{M}^{\triangle}$ and $\mathbf{S}^{\triangle}$ defined on the refined mesh with the prolongation matrix and its transpose (line 12, Section 3.2.2).

```
Data: Mesh vertices $\mathcal{V}$ and faces $\mathcal{F}$
Result: Mass matrix $\mathbf{M}^{\pentagon}$, stiffness matrix $\mathbf{S}^{\pentagon}$.
$\mathbf{S}^{\triangle} = \mathbf{0}^{|\mathcal{V}_T| \times |\mathcal{V}_T|}$
$\mathbf{M}^{\triangle} = \mathbf{0}^{|\mathcal{V}_T| \times |\mathcal{V}_T|}$
$\mathbf{P} = \mathbf{I}^{|\mathcal{V}| \times |\mathcal{V}|}$
foreach $f \in \mathcal{F}$ do
    $\mathbf{X}_f \leftarrow \text{getVertexPositions}(f)$;
    $\mathbf{w}_f \leftarrow \text{findVirtualVertexWeights}(\mathbf{X}_f)$;
    $\mathbf{x}_f \leftarrow \mathbf{X}_f^{\mathsf{T}} \mathbf{w}_f$;
    $\mathbf{P} \leftarrow \text{appendWeightRow}(\mathbf{P}, f, \mathbf{w}_f)$;
    $\mathbf{M}^{\triangle} \leftarrow \mathbf{M}^{\triangle} + \text{buildTriangleFanMassMatrix}(f, \mathbf{X}_f, \mathbf{x}_f)$;
    $\mathbf{S}^{\triangle} \leftarrow \mathbf{S}^{\triangle} + \text{buildTriangleFanCotan}(f, \mathbf{X}_f, \mathbf{x}_f)$;
end
return $\mathbf{M}^{\pentagon} = \mathbf{P}^{\mathsf{T}} \mathbf{M}^{\triangle} \mathbf{P}, \quad \mathbf{S}^{\pentagon} = \mathbf{P}^{\mathsf{T}} \mathbf{S}^{\triangle} \mathbf{P}$
```

Algorithm 1: Assemble polygon stiffness and mass matrix.

3.2.5 Gradient and Divergence

We are also able to factor the stiffness matrix as the product of divergence and gradient matrices. As described in Section 2.2.1, we can obtain a matrix expression for the gradient and divergence operators over the refined mesh, $\mathbf{G}^{\triangle}$ and $\mathbf{D}^{\triangle}$ and use those to factor the refined triangle stiffness matrix

$$\mathbf{S}^{\triangle} = \mathbf{D}^{\triangle} \mathbf{G}^{\triangle}. \tag{3.13}$$

Coarsening, we obtain a factorization of the polygon stiffness matrix in terms of coarse polygon divergence and gradient matrices

$$\mathbf{S}^{\pentagon} = \underbrace{\mathbf{P}^{\mathsf{T}} \mathbf{D}^{\triangle}}_{=\mathbf{D}^{\pentagon}} \underbrace{\mathbf{G}^{\triangle} \mathbf{P}}_{=\mathbf{G}^{\pentagon}}. \tag{3.14}$$

Note that the derived gradient operator, $\mathbf{G}^{\pentagon} = \mathbf{G}^{\triangle}\mathbf{P}$, is a map from functions defined on the vertices of the *original* polygon mesh to tangent vector fields that are constant on triangles of the *refined* mesh. These refined triangles, however, can uniquely be identified with half-edges of the original polygon mesh, so that the refined triangles never have to be constructed explicitly. Hence, the gradient operator maps from function values at vertices to vectors at half-edges (and conversely for the divergence operator, $\mathbf{D}^{\pentagon} = \mathbf{P}^{\mathsf{T}}\mathbf{D}^{\triangle}$).

3.2.6 Finite Elements Exterior Calculus

An alternative approach is to define differential operators by extending Finite Elements Exterior Calculus to polygon meshes. We do this by using our coarsened basis to define Whitney bases for higher-order forms and then using these higher-order bases to define the Hodge star operators.

Recall Given a basis of zero-forms $\{\varphi_i\}$ forming a partition of unity, one can define Whitney bases [Whi57; AFW06] for 1-forms $\{\varphi_{ij}\}$ (with $i < j$ and $\mathrm{supp}(\varphi_i) \cap \mathrm{supp}(\varphi_j) \neq \emptyset$) and 2-forms $\{\varphi_{ijk}\}$ (with $i < j < k$ and $\mathrm{supp}(\varphi_i) \cap \mathrm{supp}(\varphi_j) \cap \mathrm{supp}(\varphi_k) \neq \emptyset$) by setting:

$$\begin{aligned}\varphi_{ij} &= \varphi_i \cdot d\varphi_j - \varphi_j \cdot d\varphi_i\,,\\ \varphi_{ijk} &= 2\left(\varphi_i \cdot d\varphi_j \wedge \psi_k - \varphi_j \cdot d\varphi_i \wedge d\varphi_k + \psi_k \cdot d\varphi_i \wedge d\varphi_j\right).\end{aligned}$$

The exterior derivatives are then defined using the combinatorics

$$\mathbf{d}^0_{(ij)k} = \begin{cases} -1 & i = k \\ 1 & j = k \\ 0 & \text{otherwise} \end{cases}, \quad \mathbf{d}^1_{(ijk)(lm)} = \begin{cases} 1 & i = l,\ j = m \\ 1 & j = l,\ k = m \\ -1 & i = l,\ k = m \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

and the discrete Hodge stars are defined using the geometry

$$\star^0_{ij} = \langle\langle \varphi_i, \varphi_j \rangle\rangle, \tag{3.16}$$

$$\star^1_{(ij)(kl)} = \langle\langle \varphi_{ij}, \varphi_{kl} \rangle\rangle, \tag{3.17}$$

$$\star^2_{(ijk)(lmn)} = \langle\langle \varphi_{ijk}, \varphi_{lmn} \rangle\rangle, \tag{3.18}$$

with $\langle\langle \cdot, \cdot \rangle\rangle$ denoting the integral of the (dot-)product of k-forms. Using the linear Lagrange basis on a triangle mesh, the basis functions are linear within each triangle so computing the coefficients reduces to integrating quadratic polynomials over a triangle.

Prolonging Higher-order Forms Given the linear Lagrange basis on the refined triangle mesh $\{\psi_1, \ldots, \psi_{|\mathcal{V}_T|}\}$, defining a prolongation operator $\mathbf{P}$ is equivalent to defining a coarsened basis $\{\varphi_1, \ldots, \varphi_{|\mathcal{V}|}\}$ on the triangle mesh, with

$$\varphi_j = \sum_i \mathbf{P}_{ij} \psi_i. \tag{3.19}$$

As both bases form a partition of unity, we can define Whitney bases for higher-order forms. In doing so we get:

$$\varphi_{ij} = \sum_{k,l} \mathbf{P}_{ki} \mathbf{P}_{lj} \psi_{kl} \quad \text{and} \quad \varphi_{ijk} = \sum_{l,m,n} \mathbf{P}_{li} \mathbf{P}_{mj} \mathbf{P}_{nk} \psi_{lmn}, \tag{3.20}$$

which gives prolongation operators for 0-, 1-, and 2-forms:

$$\mathbf{P}^0_{ij} = \mathbf{P}_{ij}, \tag{3.21}$$

$$\mathbf{P}^1_{(ij)(kl)} = \mathbf{P}_{ik}\mathbf{P}_{jl}, \tag{3.22}$$

$$\mathbf{P}^2_{(ijk)(lmn)} = \mathbf{P}_{il}\mathbf{P}_{jm}\mathbf{P}_{kn}. \tag{3.23}$$

This allows us to define Hodge star operators for the coarsened basis through prolongation:

$$\star^k = (\mathbf{P}^k)^\top \star^{k,\triangle} \mathbf{P}^k, \tag{3.24}$$

where $\star^k$ is the discrete Hodge star for k-forms defined using the coarsened basis and $\star^{k,\triangle}$ is the discrete Hodge star for k-forms defined using the refined linear Lagrange basis.
In particular, this gives a factorization of the stiffness matrix as

$$\mathbf{S}^{\bigcirc} = (\mathbf{d}^0)^\top \cdot \star^1 \cdot \mathbf{d}^0, \tag{3.25}$$

with gradients represented in terms of differences across polygon edges/diagonals.

3.2.7 Laplacian on Volume Meshes

The previously described method can be intuitively extended to arbitrary polyhedral meshes, but instead of virtual triangles, the mesh $\mathcal{M}$ will be refined into virtual *tetrahedra*. The first steps are analogous to the surface case, meaning that all faces $f \in \mathcal{F}$ of a given polyhedron $c \in \mathcal{C}$ are refined into triangles with virtual vertices placed at the point that minimizes the sum of squared triangle areas (see Equation (3.11)). To span the virtual tetrahedra, we introduce an additional vertex $\mathbf{x}_c$ inside of each cell, which is, similar to the surface case, the affine combination of the cells' refined faces vertex positions

$$\mathbf{x}_c = \sum_{v_i \in \mathcal{V}(\partial c)} w_i \, \mathbf{x}_i, \quad \text{with} \sum_{v_i \in \mathcal{V}(\partial c)} w_i = 1. \tag{3.26}$$

Here, $\mathcal{V}(\partial c)$ refers to the set of vertex indices that lie on the refined boundary of the polyhedron c. The position of $\mathbf{x}_c$ is defined as the minimizer of the sum of squared tetrahedron volumes

$$\mathbf{x}_c = \arg\min_{\mathbf{x}} \sum_{t_{ijk} \in \partial c} \mathrm{vol}\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}\right)^2, \tag{3.27}$$

with t_{ijk} being the refined triangles along the cell's boundary. As for surfaces, this minimization problem can be expressed with respect to a set of affine weights $\mathbf{w}_c \in \mathbb{R}^{|\mathcal{V}(\partial c)|}$ and assembled into a local prolongation matrix $\mathbf{P}^c$. The only real difference of this approach is that the global prolongation $\mathbf{P}$ is now divided into a two-step process, with the "surface" prolongation matrix $\mathbf{P}_F$ inserting the virtual face points for all $f \in \mathcal{F}$ and $\mathbf{P}_C$ the cell points for all $c \in \mathcal{C}$, respectively, giving us

$$\mathbf{P} = \mathbf{P}_C \mathbf{P}_F. \tag{3.28}$$

The polyhedral stiffness and mass matrix are then obtained as in Equation (3.7) and Equation (3.8), with the slight change that the refined matrices are the volumetric discretizations of the cotangent formula (see Section 2.2.2).

3.2.8 Properties of the Operator

One goal of our construction is to preserve the beneficial numerical properties of the cotangent Laplacian. The **symmetry**, **positive semi-definiteness** and **null space** properties follow for both surface and volume methods directly from our construction of the stiffness matrix $\mathbf{S}^{\circ} = \mathbf{P}^{\mathsf{T}}\mathbf{S}^{\triangle}\mathbf{P}$, since the refined (cotangent) stiffness matrix $\mathbf{S}^{\triangle}$ has these properties and since the prolongation matrix $\mathbf{P}$ has full rank. For **linear precision** we require $(\mathbf{S}^{\circ}\mathbf{u})_i = 0$ whenever vertex v_i is not part of the boundary, all incident polygons $f_j \ni v_i$ are coplanar, and the values of $\mathbf{u}$ in the one-ring of v_i are obtained by sampling a linear function on the plane. To see that this is the case, we note that by constraining ourselves to use affine weights in defining the prolongation, we ensure that prolonging $\mathbf{u}$ to the finer mesh, the values of $\mathbf{u}^{\triangle} = \mathbf{P}\mathbf{u}$ sample a linear function at all vertices of the fan triangulations incident to v_i. Since the cotangent Laplacian has linear precision, this implies that $\mathbf{S}^{\triangle}\mathbf{u}^{\triangle}$ is zero at v_i *and* all virtual centers $v_{|\mathcal{V}|+j}$ with $f_j \ni v_i$. Since these are precisely the entries at which the i-th column of $\mathbf{P}$ is non-zero, it follows that $(\mathbf{P}^{\mathsf{T}}\mathbf{S}^{\triangle}\mathbf{u}^{\triangle})_i = 0$. Or, equivalently, $(\mathbf{S}^{\circ}\mathbf{u})_i = 0$. Given that the cotan Laplacian for volume meshes also satisfies linear precision, the same arguments apply for polyhedral meshes. It follows that if the initial mesh $\mathcal{M}$ is a **triangle** or **tetrahedral** mesh, then the derived stiffness matrix $\mathbf{S}^{\circ}$ is the standard surface of volume cotangent Laplacian. This is because using affine weights, the finite elements basis defined on the coarse mesh through prolongation are *precisely* the linear Lagrange elements.

The property of strictly negative off-diagonal values does not hold for the cotangent Laplacian and consequently cannot extend to our construction, resulting in the potential violation of the **maximum principle**. The lack of negativity is also present in the definition of [AW11] and [GBD20], however, our operator tends to contain fewer positive off-diagonal coefficients, which will be discussed in the upcoming evaluation.

3.3 EVALUATION

In this section, we evaluate the Laplacian based on the linear virtual refinement method in a variety of different qualitative and quantitative geometry processing operations. After providing some visual examples in which a polygon Laplacian can be used, we will analyze the effect of the placement of the virtual vertex applied to a selection of polygon meshes (Figures 3.2 and 3.3). Additionally, we evaluate the difference in accuracy between our method and the triangulation of the original polygon mesh. For this, we use a selection of different approaches. First, we triangulate each (potentially non-planar, non-convex) n_f-gon into $n_f - 2$ triangles without inserting an extra point, and then use the standard cotangent Laplacian for triangle meshes. Secondly, using the dynamic-programming approach of Liepa [Lie03], we find the polygon triangulation that minimizes the sum of squared triangle areas (similar in concept to our squared area minimization). We also experimented with the triangulation that maximizes the minimum triangle angle (similar to planar Delaunay triangulations). While the latter yields better-behaved cotangent weights, it tends to produce flips or fold-overs for non-convex polygons, so we used the former for most experiments.
In the second part of this chapter, we compare our results to the ones obtained with the polygon Laplacians based on the work of Alexa and Wardetzky [AW11], de Goes et al. [GBD20] and Martin et al. [MKB+08], using the values for their respective parameters λ evaluated in Section 2.3.3. For the harmonic shape functions [MKB+08], the number of chosen kernels and control points strongly affects the results on our chosen test meshes. We follow the evaluation of our survey paper [BB23] and use a ratio of 20/80 for edge kernel/collocation points for surface meshes and the recommended 3/9 points per edge and 10/30 per face for the volumetric tessellations. The choice was also influenced by the numerical costs involved in using more samples, explaining the lower sample sizes for volume meshes. The triangles and tetrahedra used for the numerical integration are the same as the virtual elements used for the virtual refinement method. Given that the integration of the shape functions is not exact, using a more elaborate tessellation technique could further improve the results.

3.3.1 Qualitative Examples

Before we analyze the varying numerical aspects of our operator, we provide some qualitative examples in which a polygon Laplacian can be used, to highlight the flexibilty and stability of our approach. All examples use the mentioned minimizer of squared triangle areas as virtual vertex and the norm minimizing affine weights for the prolongation.

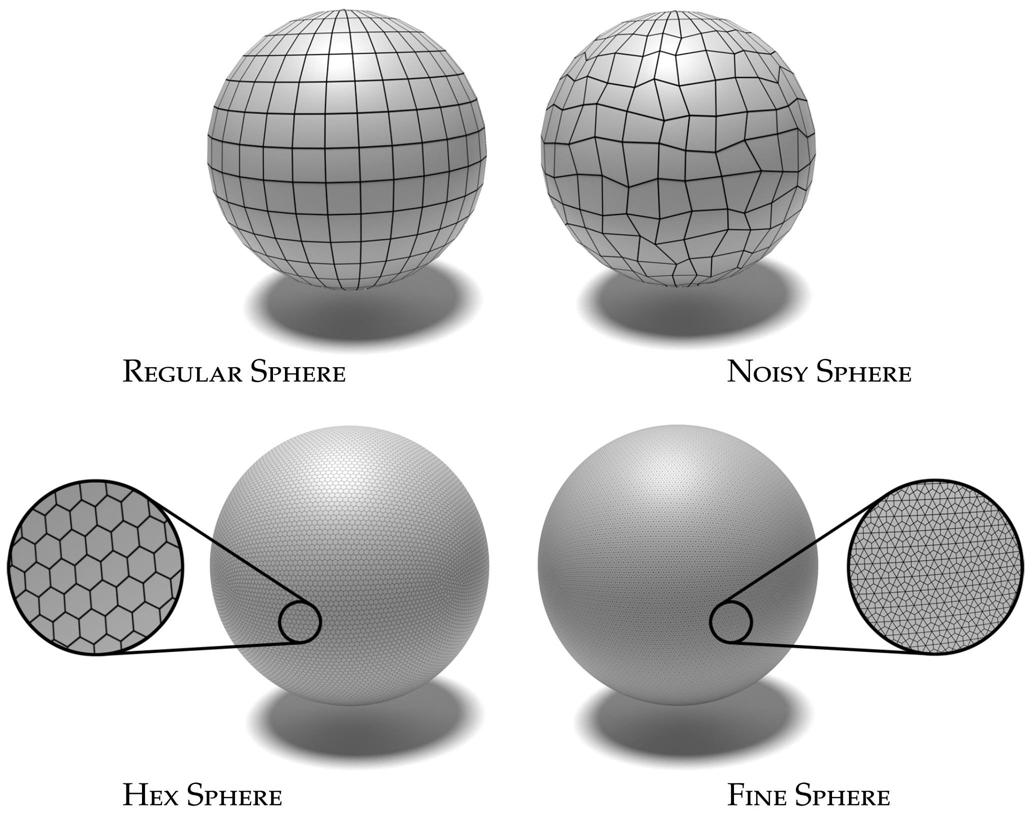


Figure 3.2: *Spherical meshes used for within the evaluation.*

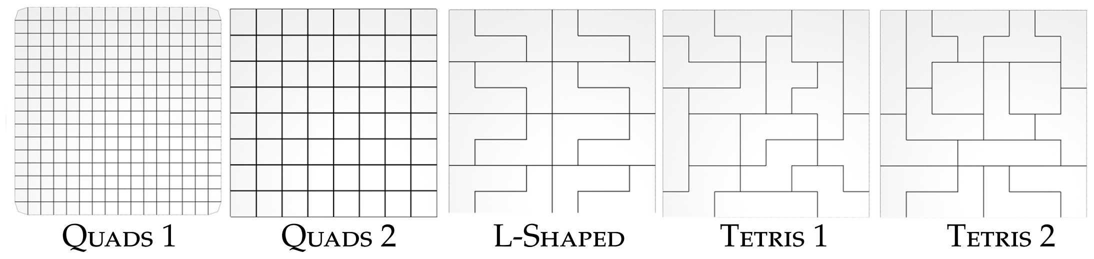


Figure 3.3: *Planar meshes used for the evaluation of geodesic distances, including non-convex and non-star shaped tessellations.*

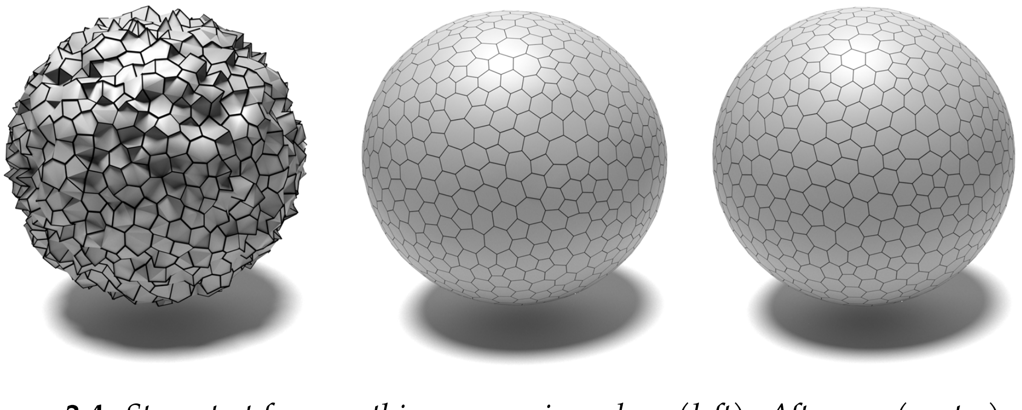


Figure 3.4: *Stress test for smoothing on a noisy sphere (left). After one (center) and ten iterations (right).*

Conformalized Mean Curvature Flow A common approach for smoothing meshes is to use implicit integration to solve the diffusion equation [DMS+99]. In our application we use the conformalized Mean Curvature Flow introduced by Kazhdan et al.[KSBC12], which obtains the coordinates of the mesh vertices at the next time-step, $\mathbf{X}^{t+\varepsilon}$ from the coordinates at the current time-step, $\mathbf{X}^{t}$, iteratively solving the linear system

$$\left(\mathbf{M}_t^{\pentagon} + \varepsilon \mathbf{S}_0^{\pentagon}\right) \mathbf{X}^{t+\varepsilon} = \mathbf{M}_t^{\pentagon} \mathbf{X}^{t} \tag{3.29}$$

with ε the time-step, $\mathbf{S}_0^{\pentagon}$ the initial stiffness matrix, and $\mathbf{M}_t^{\pentagon}$ the mass matrix at time t. After each iteration the mesh is translated back to the origin and re-scaled to its original surface area. Figure 3.4 demonstrates the resilience of our Laplacian to noise and non-planar as well as non-convex faces. The flow can recover the spherical shape after one iteration and converges after 10 iterations. Figure 3.5 shows a second example of this flow after one and ten iterations respectively. The mean curvature is color-coded and shows that the mesh correctly converges to a sphere when using our operator. This example shows that even extreme differences in polygon scale are handled correctly.

Parametrization Another potential application for the Laplacian is mesh parameterization. Traditionally, the goal is to find a correspondence between a discrete surface patch (possibly with holes) and a homeomorphic planar mesh through a piecewise linear map. Figure 3.6 illustrates an example of a conformal parameterization based on Mullen et al.'s [MTA+08] spectral free-boundary parametrization, extended to polygon meshes. We compare our operator to a result obtained with the Laplacian based on Alexa and Wardetzky [AW11]. In general, both operators perform well in this application, but visual differences can still be observed.

Geodescis in Heat As described in Section 2.3.3, Crane et al. [CWW13] proposed the heat method for computing geodesic distances from a selected vertex v_i to all others on the mesh. Figure 3.7 shows a result of geodesics in heat with highly anisotropic polygons, on which our operator yields smooth geodesic distances.

3.3.2 Comparison of Virtual Vertex Choices

As stated in Section 3.2.3, there are numerous options to compute the virtual vertices and their weights needed for the linear virutal refinement method. We compare the performance of several alternative constructions, using both lumped and un-lumped mass matrices, in a number of applications. Additionally, we compare to *explicit* triangulations of the polygon meshes such that that

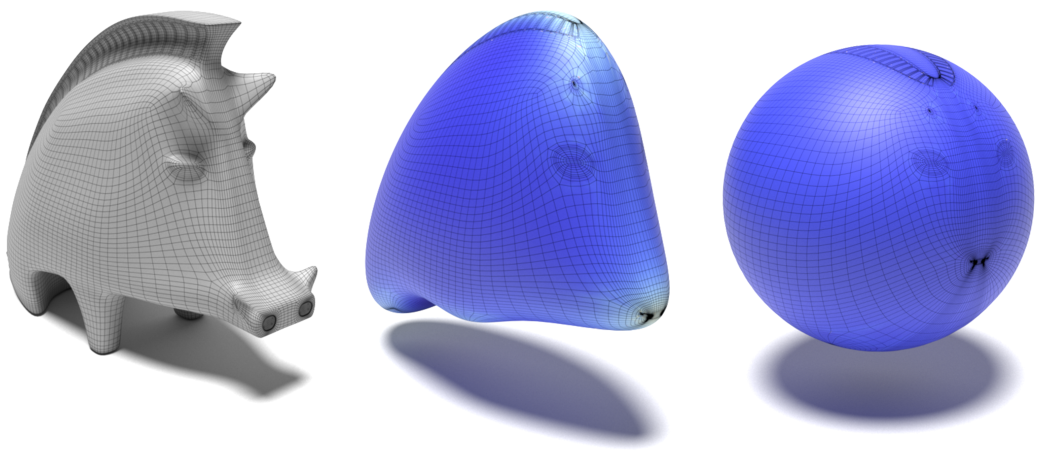


Figure 3.5: *Visualization of the mean curvature flow on a quad mesh. Mean curvature is color coded.*

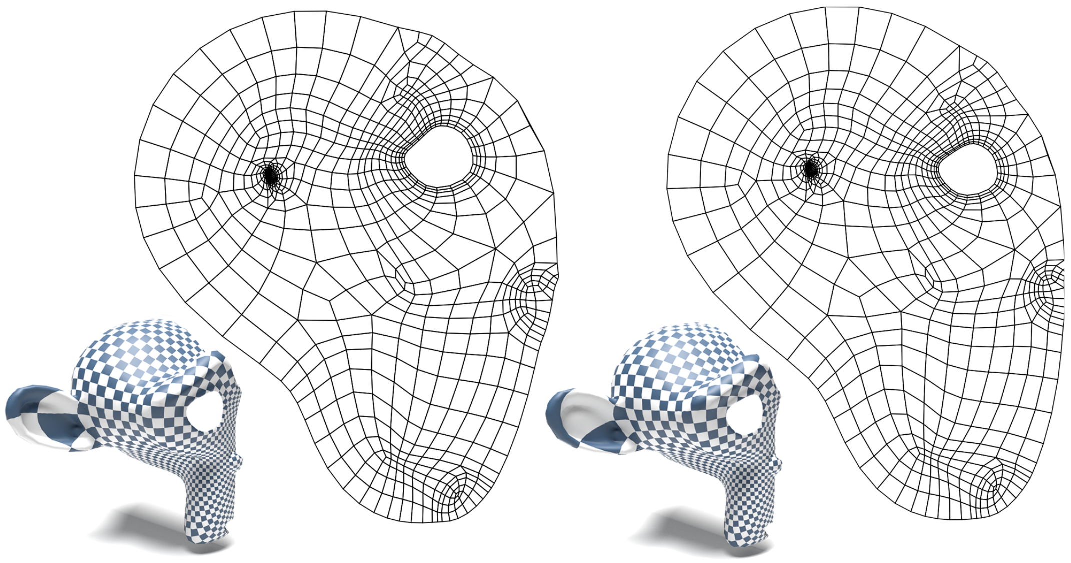


Figure 3.6: *Parametrization of the right half of a quadrangulated monkey head. The result of [AW11] (left) is similar to ours (right), but our operator has slightly lower conformal distortion.*

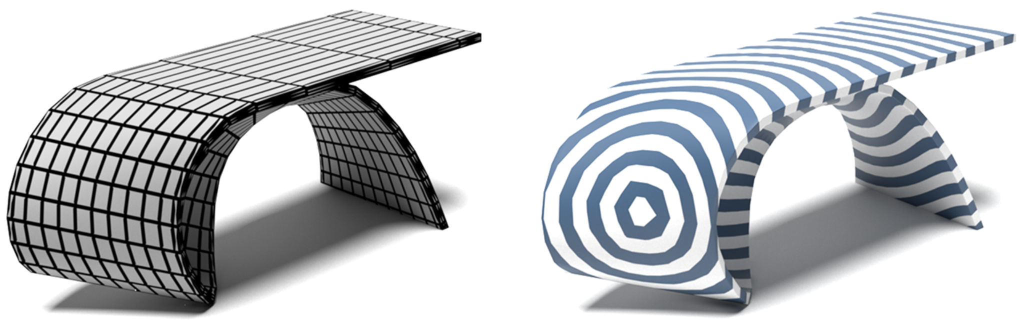


Figure 3.7: *Geodesic distances computed on a quad mesh (left) with our Laplacian (right) and a timestep set to the squared mean of edge lengths.*

the sum of squared triangle areas is minimized, using the dynamic programming approach of Liepa [Lie03]. For geodesic distances, we also provide results for computing the Laplacian based on the intrinsic Delaunay triangulation [BS07] of this minimum area triangulation.

Mean Curvature Estimation Noting that the Laplacian of the coordinate function is the mean-curvature normal vector, we can use our Laplace operator to approximate the mean curvature H at vertex v_i

$$H(v_i) := \frac{1}{2} \|(\mathbf{LX})_i\| \cdot \operatorname{sign}(\langle(\mathbf{LX})_i\,,\,\mathbf{n}_i\rangle) \tag{3.30}$$

with $\mathbf{n}_i$ being the normal at v_i.
Since the mean curvature is one at each point on the unit sphere, we can measure the accuracy of our Laplacian by comparing the estimated mean curvature to the true one. Table 3.1 gives the root mean square error (RMSE), comparing curvature estimates over different polygonizations of the sphere, and using different definitions of the Laplace operator. As the table shows, using the lumped mass matrix increases the accuracy of our operator significantly. While an un-lumped matrix yields results that are generally surpassed by the triangulation approach, using the diagonalized matrix changes the outcome for most meshes, with the exception of the fine sphere. Overall, the squared triangle area minimizer leads to the most accurate results, for both affine and convex prolongation weights.

Reproducing the Spherical Harmonics The eigenfunctions of the Laplacian form an orthonormal basis known as the “manifold harmonics” [VL08]. As described in Section 2.3.3, in the case that the surface is a sphere, these eigenfunctions are known to be the spherical harmonics. We evaluate the quality of our Laplacian by measuring the extent to which the true spherical harmonics are eigenvectors of the Laplacian. Table 3.2 compares the spectral error, giving the sum of errors over the spherical harmonics in the first nine frequencies

	un-lumped mass matrix				
	Sqr. Area				
Mesh	Affine	Convex	Centroid	Abs. Area	[Lie03]
Hex Sphere	**0.0039**	0.0159	0.0159	0.0100	0.3711
Fine Sphere	0.3669	0.4181	0.4181	0.5454	**0.0623**
Regular Sphere	0.0515	0.0515	0.0515	0.0537	**0.0469**
Noisy Sphere	0.1520	0.1463	0.1463	0.1641	**0.1053**
	lumped mass matrix				
	Sqr. Area				
Mesh	Affine	Convex	Centroid	Abs. Area	[Lie03]
Hex Sphere	**0.0016**	0.0049	0.0049	0.0026	0.3711
Fine Sphere	0.1334	0.1332	0.1332	0.1630	0.0623
Regular Sphere	**0.0168**	**0.0168**	**0.0168**	0.0172	0.0469
Noisy Sphere	0.0470	**0.0440**	**0.0440**	0.0493	0.1053

Table 3.1: *RMSE of mean curvature computation on the spherical meshes in Figure 3.2.*

	un-lumped mass matrix				
	Sqr. Area				
Mesh	Affine	Convex	Centroid	Abs. Area	[Lie03]
Hex Sphere	**9.18e-7**	4.56e-6	4.56e-6	2.10e-5	0.0037
Fine Sphere	0.0009	0.0009	0.0009	0.0011	**0.0005**
Regular Sphere	0.0256	0.0258	0.0258	0.0272	**0.0200**
Noisy Sphere	0.0636	0.0804	0.0804	**0.0623**	0.0722
	lumped mass matrix				
	Sqr. Area				
Mesh	Affine	Convex	Centroid	Abs. Area	[Lie03]
Hex Sphere	**7.41e-7**	1.15e-6	1.15e-6	4.30e-6	0.0037
Fine Sphere	**0.0003**	**0.0003**	**0.0003**	0.0004	0.0005
Regular Sphere	0.0393	0.0398	0.0398	0.0398	**0.0200**
Noisy Sphere	**0.0643**	0.0644	0.0644	0.0655	0.0722

Table 3.2: *RSME of spherical harmonics on the meshes in Figure 3.2.*

$(1 \leq l \leq 9)$. As the table shows, using the squared triangle minimizer as virtual vertex, combined with affine weigths, yields the most accurate results for our method. In only one occasion it is surpassed by the absolute area minimizer. As for the mean curvature estimation, using the lumped mass matrix leads to an overall better accuracy, one again surpassing the triangulation approach on most tessellations.

un-lumped mass matrix						
	Sqr. Area					
Mesh	Affine	Convex	Centroid	Abs. Area	[Lie03]	[BS07]
QUADS 1	**0.025**	**0.025**	**0.025**	**0.025**	0.039	0.039
QUADS 2	**0.031**	**0.031**	**0.031**	**0.031**	0.086	0.086
L-SHAPED	0.141	0.165	0.134	0.134	**0.071**	0.074
TETRIS 1	0.465	0.490	0.490	0.491	0.168	**0.141**
TETRIS 2	0.406	0.346	0.151	0.153	0.081	**0.067**
lumped mass matrix						
	Sqr. Area					
Mesh	Affine	Convex	Centroid	Abs. Area	[Lie03]	[BS07]
QUADS 1	**0.026**	0.027	0.027	0.027	0.039	0.039
QUADS 2	**0.036**	**0.036**	**0.036**	**0.036**	0.086	0.086
L-SHAPED	**0.057**	0.063	0.068	0.068	0.071	0.074
TETRIS 1	0.218	0.181	0.186	0.185	0.168	**0.141**
TETRIS 2	0.082	0.089	0.089	0.088	0.081	**0.067**

Table 3.3: *RMSE of geodesic distances for the planar meshes in Figure 3.3 computed with different choices of virtual vertices and using un-lumped and lumped mass matrices. For each planar mesh the selected vertex was the one with the least norm to the center of the plane.*

Geodesics in Heat We qualitatively compare the results using our operator and several polygon triangulation strategies in Figure 3.8. The selected time step was the squared mean edge length of the mesh. Our construction (a) gives a result with considerably fewer artifacts compared to the other approaches. Triangulating polygons to maximize the minimal angle (c) also gives good results, but this approach is not suitable for arbitrary meshes since it fails on non-convex polygons. Minimum-area triangulations avoid this problem (d), but give worse results due to poor triangle shapes. Combining minimum-area triangulations with the Laplacian based on the intrinsic Delaunay triangulation [BS07] (e) fixes this problem, but is more complex to compute. In (b) we show the result obtained by using the cotangent Laplacian on the mesh explicitly refined by inserting the virtual vertices (this is equivalent to $\mathbf{S}^{\triangle}$). Our construction is clearly different from just refining polygons.
The quality of geodesics is linked to the number of positive off-diagonal coefficients in the stiffness matrix. Analyzing the ratio of these entries for Figure 3.8 confirms this correlation:

a) ours	b) refinement	c) max-angle	d) min-area	e) intr. Delaunay
11%	17%	5%	15%	0%

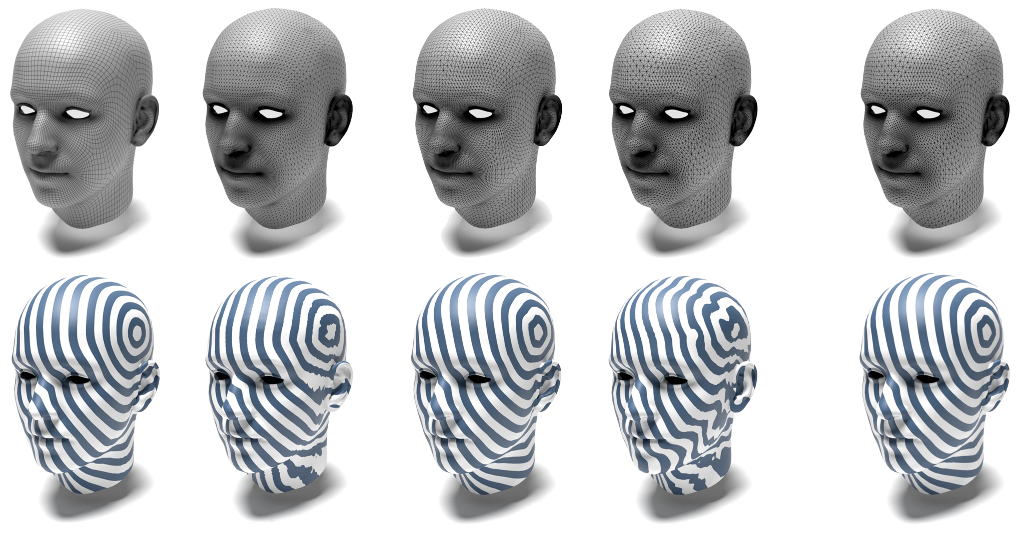


Figure 3.8: *Computing geodesic distances on a quad mesh (top left) following the approach of Crane et al. [CWW13] with the timestep set to the squared mean of edge lengths. Column (a) depicts the result of our operator, while the images (b–e) show results using different triangulation strategies: refining the mesh by inserting the virtual point (b), triangulating polygons to maximize the minimum angle (c), and triangulating polygons to minimize squared triangle areas (d). To improve the results of (d), we employ the Laplacian based on the intrinsic Delaunay triangulation [BS07] (e). The triangles of (c) are already Delaunay, therefore using the intrinsic triangulation does not further improve the result in (c).*

The smaller the number of positive non-diagonal entries is, the higher the quality of the result becomes. Table 3.3 provides a quantitative evaluation of geodesic distances for the different point choices. We compare them to Euclidean distances on different planar meshes (Figure 3.3). Our operator yields smaller root-mean-square errors for most models, including the geodesic distances computed via intrinsic Delaunay triangulation [BS07; SSC19a] (based on the implementation provided in libigl [JP+18]), but is bested on the tetris tessellations.

Timings We evaluated the computational costs of the different virtual vertex options on the Hex Sphere (16070 faces) and the Fine Sphere (96408 faces) shown in Figure 3.2. All timings were measured on a standard workstation with a six-core Intel Xeon 3.6 GHz CPU; no experiment exploited multi-threading. We analyze the construction of our approach with the different virtual vertex versions described in Section 3.2.3: *centroid* of polygon vertices, minimizing the sum of (absolute) triangle areas (*Abs. Area*), minimizing the sum of squared triangle areas using either *affine* weights or *convex* weights. We compare these methods to the minimum area polygon triangulation [Lie03]. The timings are given in Table 3.4.

Mesh	Affine	Convex	Centroid	Abs. Area	[Lie03]
Hexagon	55	142	13	357	21
Quads	171	573	50	1460	82
Hexagon	472+25	483+25	476+26	469+25	77+8
Quads	596+35	596+35	599+35	600+36	498+29

Table 3.4: *Timing (in ms) for constructing the Laplace matrix (top) and solving the linear system (bottom). The latter is split into the time needed for Cholesky factorization and for back-substitution.*

In terms of the construction time for the Laplace matrix, our approach, with the exception of *centroid*, is the fastest of the different virtual vertex choices, since the Newton optimization of *Abs. Area* (based on Eigen) and the QP solver of *convex* (based on CGAL) are computationally expensive. However, triangulating the mesh and constructing the cotangent Laplacian is faster than defining a polygon Laplacian. Also, since a vertex on the polygon mesh has at least as many face-adjacent neighbors as the same vertex on the triangulated mesh, the polygon Laplacians are less sparse, resulting in an increased solver time.

Conclusion To conclude, in terms of accuracy, our proposed version using affine weights and the squared triangle area minimizer gives the overall best results. However, in some settings the strictly convex weights can lead to smaller errors. For most applications, using the lumped mass matrix instead of the traditional mass matrix defined in Equation (3.8) improves the results. Therefore, balancing numerical performance with efficiency, we found the minimizer of the squared triangle areas through affine weights to be the best choice.

3.3.3 Comparison to Other Polygon Laplacians

In this section, we compare the linear virtual refinement method to the polygon Laplacians defined by Alexa and Wardetzky [AW11], de Goes et al. [GBD20] (see Section 2.3) and the harmonic shape functions by Martin et al. [MKB+08]. We use the same test setting introduced in Section 2.3.3 and will, therefore, omit a detailed description of the different applications. The figures in the evaluation will feature specific labels for some methods to improve clarity. We will refer to the harmonic shape functions as "Harmonic" and the linear virtual refinement method as "Lin. Virt. Ref." respectively.

Poisson Equations The convergence behavior of the linear virtual refinement method is depicted in Figure 3.9 for planar meshes and in Figure 3.10 for spherical tessellations. The identical slopes of the log-log plots for the unit grids reveal that our operator inherits the desired quadratic convergence order of the triangle-based cotangent Laplacian. On challenging meshes, the convergence rate stagnates similarly to that of the other polygon Laplacians. While competitive in terms of accuracy, it is bested by the DEC operators for one selection of the considered parameter choices on two of the three planar polygon meshes. However, depending on the mesh, the other parameter choice typically yields one of the highest error rates. Our method yields the best results for concave faces and some of the lowest error rates for spherical meshes, especially for hexagons. At the same time, the harmonic shape functions surpass the other operators on the concave tessellation.

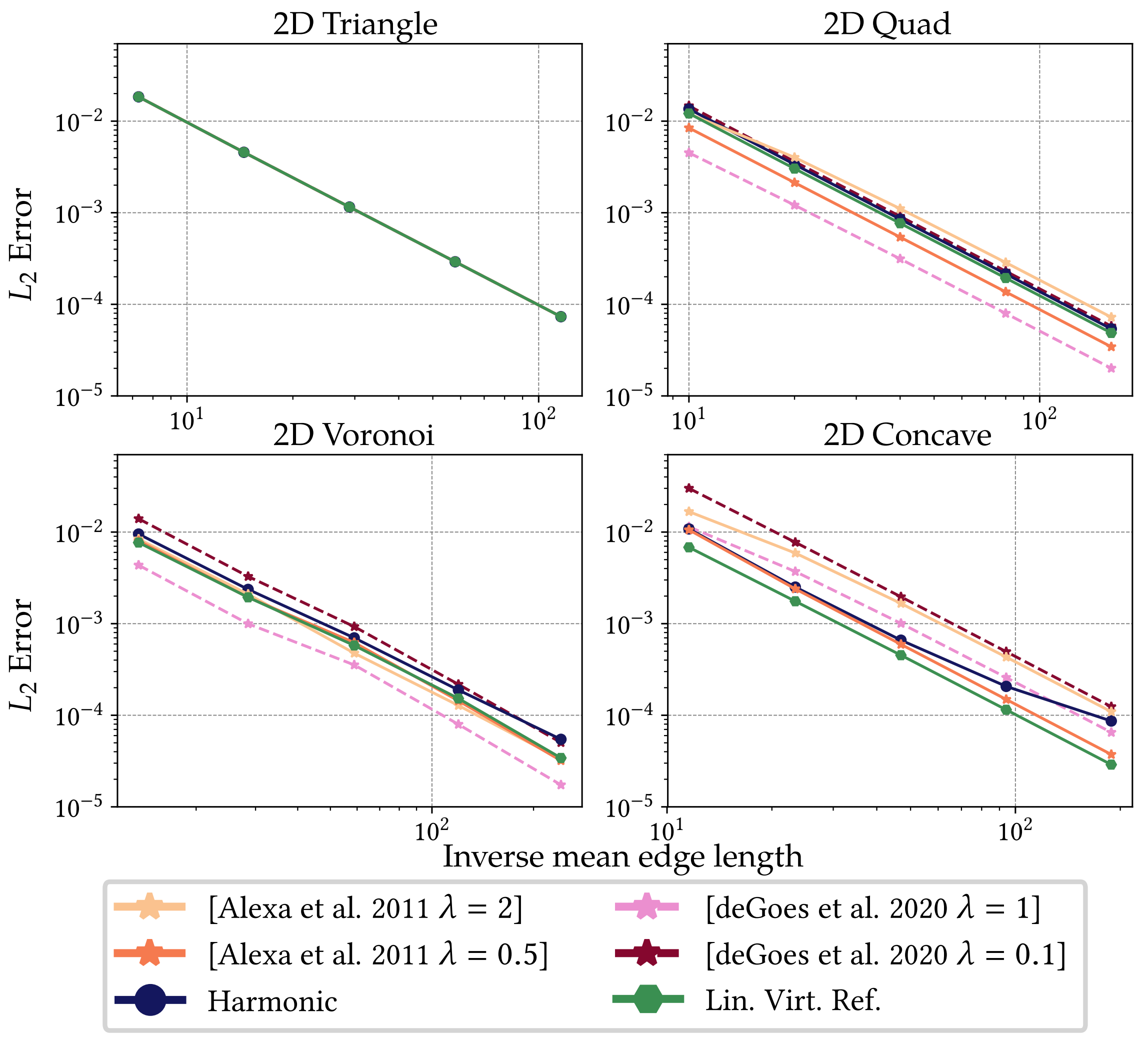


Figure 3.9: *L_2 convergence plots for the solution of the Poisson equation on planar grids with triangle, quad, Voronoi and concave faces of increasing resolution.*

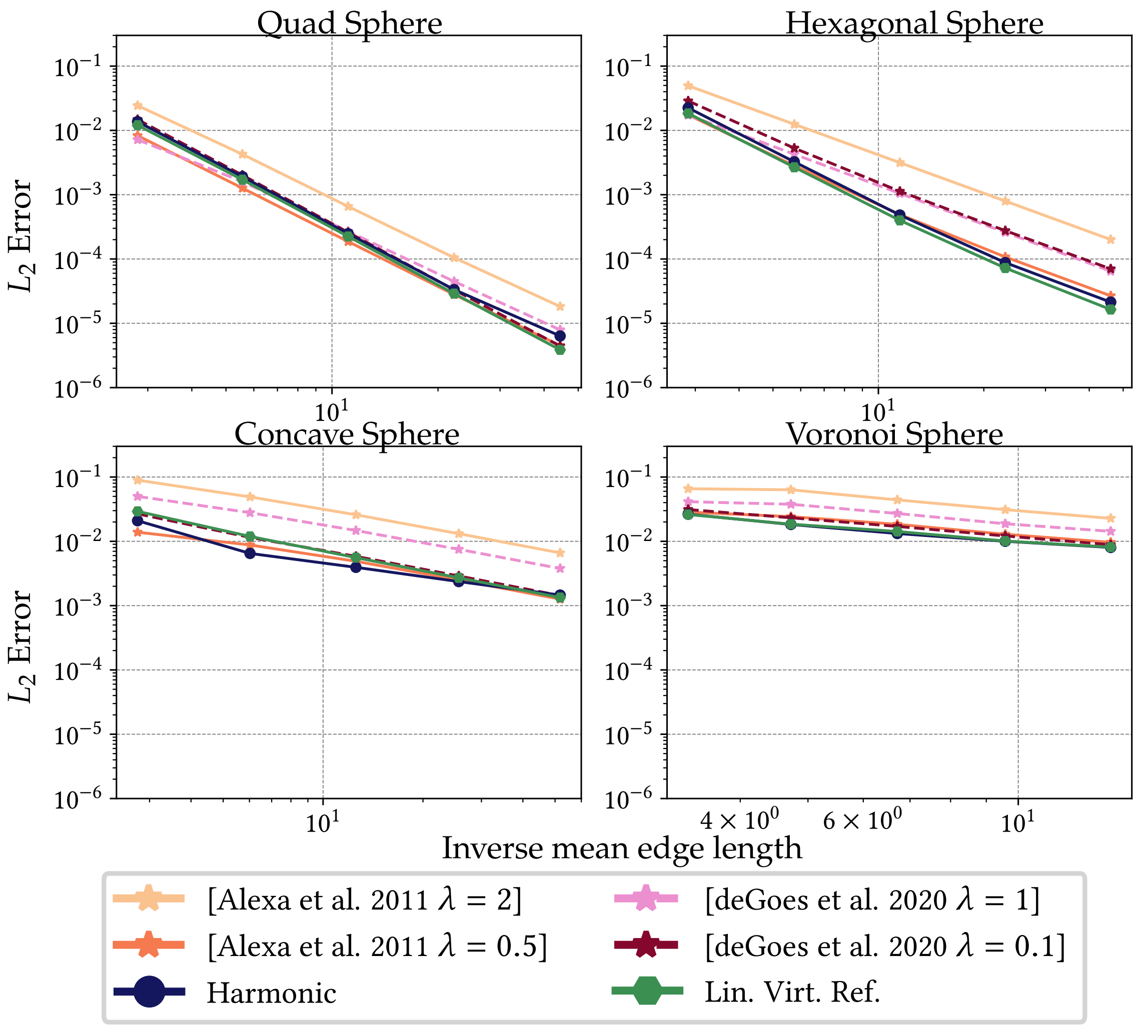


Figure 3.10: *L_2 error in log-log scale for the Poisson solve of the spherical harmonic function Y_3^2 with eigenvalue -12 on different tessellations of the unit sphere.*

Geodesics in Heat Figure 3.11 and 3.12 show the deviation of the obtained geodesic distances, including the results of our linear virtual refinement method. While we already established that the DEC operators tend to have more significant error fluctuations for different parameter choices λ, the Laplacian obtained with our proposed method remains relatively unaffected, even if we choose different time steps for the computation. However, while yielding good results, the linear virtual refinement method is often outperformed by the Laplacian of deGoes et al. [GBD20] for the optimized parameter $\lambda = 0.1$.

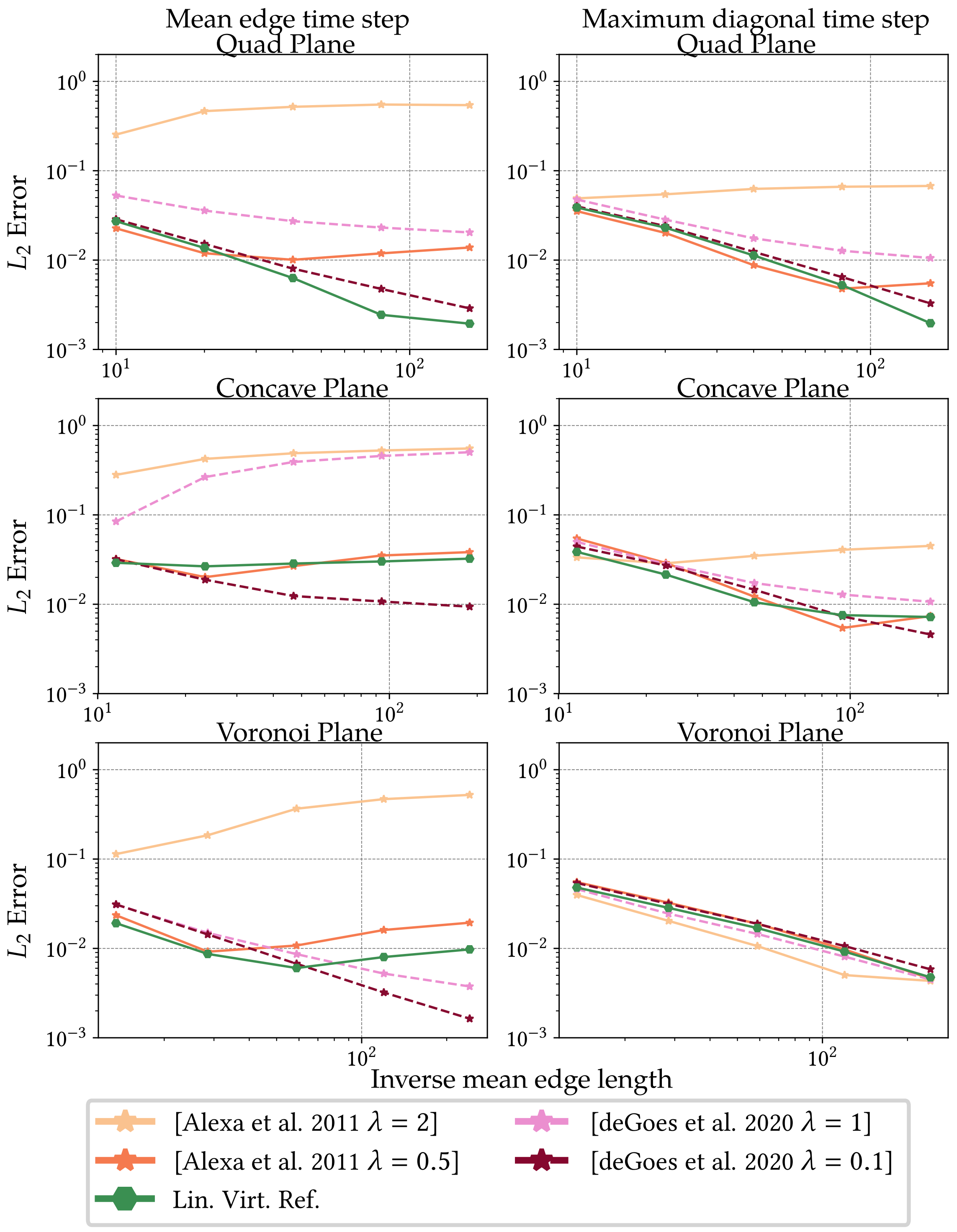


Figure 3.11: *L_2 error in log-log scale of the Geodesics in heat method on planar grids with quads (top), concave polygons (center) and Voronoi faces (bottom).*

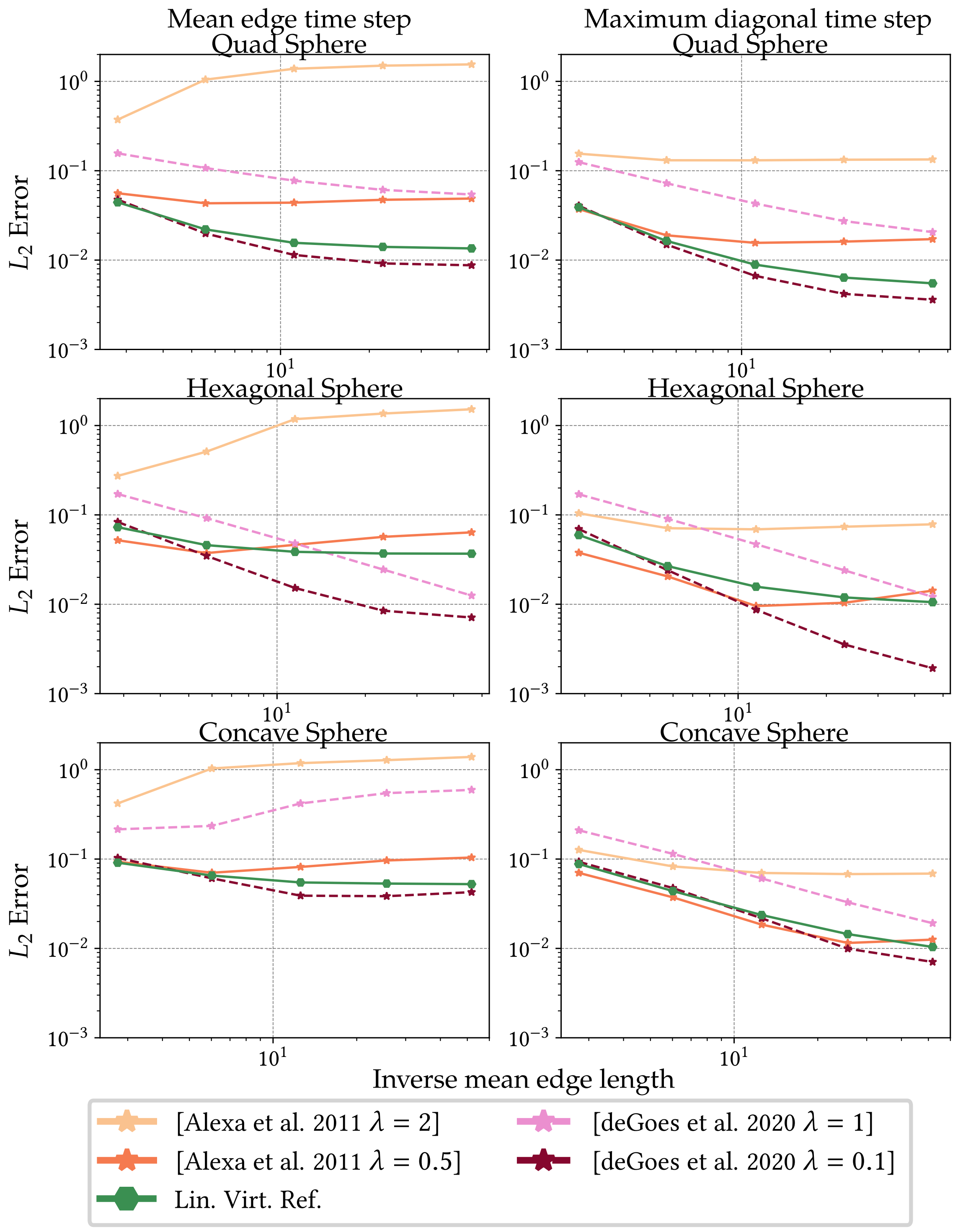


Figure 3.12: *L_2 error in log-log scale of the Geodesics in heat method on unit spheres with quad (top), hexagon (center) and concave faces (bottom).*

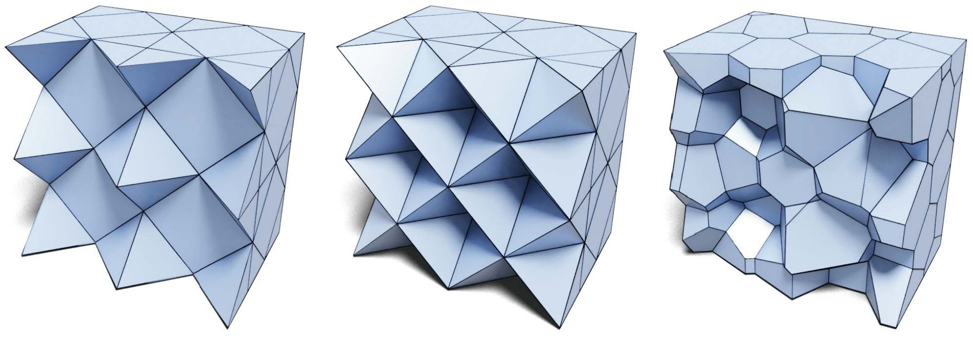


Figure 3.13: *The types of non-simplicial polyhedral meshes used for evaluation in the volumetric case. They will be referred to as Pyramids (left), Truncated (center), and Voronoi (right).*

3.3.4 Linear Virtual Refinement on Volumes

As previously mentioned, the DEC operators are only defined on surface meshes. The volumetric evaluation of the linear virtual refinement method will therefore only be compared against the harmonic coordinates. Examples for the different types of polyhedral tessellations used in the following evaluation are depicted in Figure 3.13.

Poisson Equations Similar to surface meshes, the convergence behavior of the Laplacian can be analyzed on different volumetric tessellations of the unit cube. The process is similar to Section 2.3.3, but now we chose the 3D Franke test function [Fra79] (see Equation (2.49)) for the right hand side of the system. The error rates are depicted in Figure 3.14. Both methods, the linear virtual refinement method and the harmonic shape functions by Martin et al. [MKB+08] yield qualitatively similar results. However, while slightly more accurate, the harmonic coordinates are very expensive due to the solving process involved in their construction, especially for volume meshes, while our method is significantly faster (see paragraph "Timings and Sparsity").

Eigenmodes Given the volumetric unit 3-ball $\mathcal{B}^3$, the eigenfunctions u and eigenvalues λ of the Laplacian can be obtained with the help of the Helmholtz equation:

$$\Delta u = -\lambda u \quad \text{in } \mathcal{B}^3 \tag{3.31}$$

$$\text{s.t.} \quad u = 0 \quad \text{on } \partial\mathcal{B}^3. \tag{3.32}$$

The discrete solution can be expressed by the spherical Bessel functions, which allows us to solve the generalized eigenvalue problem

$$\mathbf{Su} = \lambda\mathbf{Mu} \tag{3.33}$$

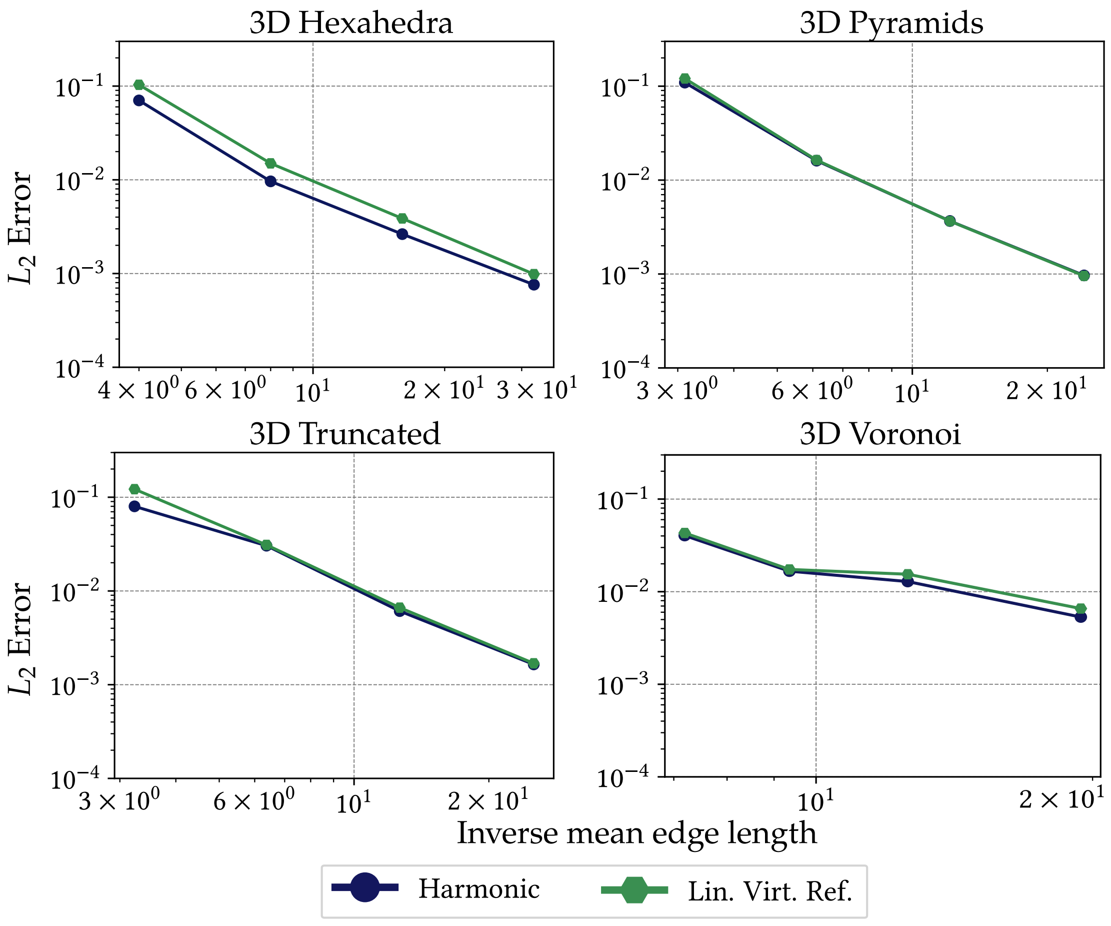


Figure 3.14: *L_2 error in log-log scale of the Poisson system solved for Franke's test function on unit cubes tessellated with hexahedra (left), pyramids (center left), truncated cells (center right), and Voronoi cells (right).*

Table 3.5: *Different statistics involved in the solution process of a Poisson problem with the presented polygon and polyhedral Laplacians.*

Mesh	$\lvert\mathcal{V}\rvert$	Harmonic			[AW11]			[GBD20]			Lin. Virt. Ref.		
		build	nnz	solve	build	nnz	solve	build	nnz	solve	build	nnz	solve
Quads 2D	26k	92s	231k	8ms	44ms	231k	8ms	47ms	231k	8ms	10ms	231k	8ms
Voronoi 2D	51k	78s	616k	25ms	94ms	616k	25ms	76ms	616k	25ms	29ms	616k	25ms
Hexahedra 3D	4913	465s	117k	3ms	—	—	—	—	—	—	190ms	117k	3ms
Voronoi 3D	5183	482s	324k	5ms	—	—	—	—	—	—	140ms	324k	5ms

with the stiffness and mass matrix obtained on a polyhedral tessellation of $\mathcal{B}^3$. Figure 3.15 shows the results for the eigenvalues on the unit ball. Once again, the harmonic coordinates yield slightly more accurate results, but both methods display the desired constant eigenvalues for the respective frequencies, with only slight deviations.

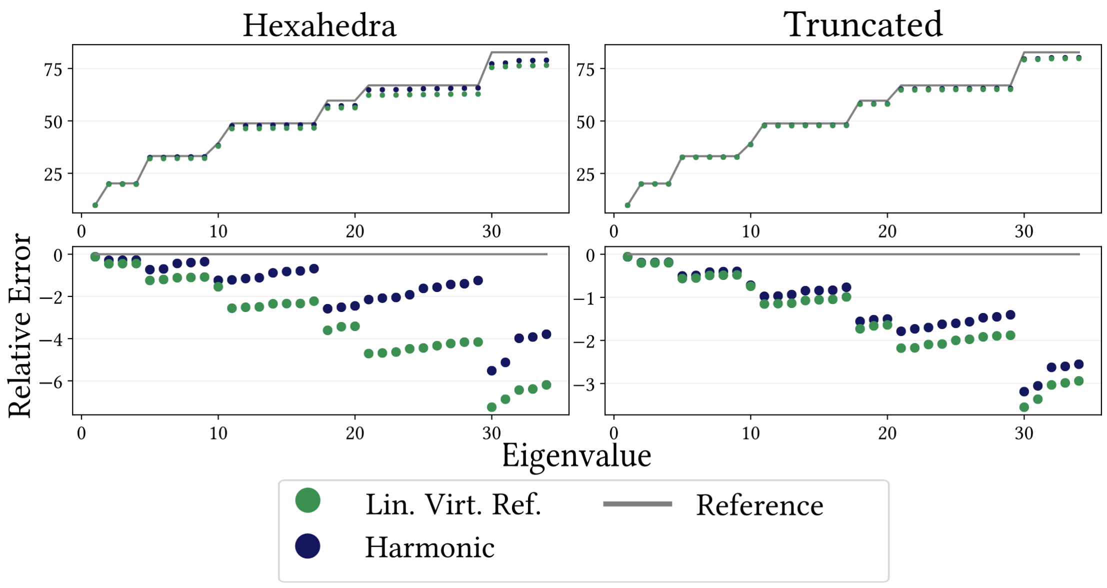


Figure 3.15: *The smallest 34 non-zero eigenvalues of the Laplacian on two unit balls consisting of hexhedra (left) and truncated cells (right). The individual top plots shows the computed eigenvalues and the lower ones the relative deviation from the ground truth.*

Timings and Sparsity In this section, we compare different statistics involved in the solution process of a Poisson problem for both 2D and 3D meshes. Table 3.5 lists the respective timings to construct the stiffness matrix (build), the number of its non-zero entries (nnz), and the time it takes to solve the system (solve) with Eigen's SimplicialLLT solver [GJ+10]. The timings were measured on a standard workstation with a six-core Intel Xeon 3.6 GHz CPU. All Laplacians have the same sparsity pattern, leading to roughly the same solving times. However, while the implementation of the respective methods has not been extensively optimized for efficiency, it is very apparent that the construction time of the harmonic shape functions by Martin et al. [MKB+08] exceeds the other operators by a tremendous amount. Especially for volume meshes, the time it takes to build the involved matrices makes the method not competitive, since its accuracy is on par with the linear virtual refinement method and does not justify the large costs.

Summary While our evaluations do not demonstrate that our Laplace operator is superior to existing state-of-the-art methods, it does show that it is competitive. We perform slightly better than triangulation [Lie03] and, while typically on par with the harmonic shape functions, are significantly more efficent regarding the construction costs of our operator. While the DEC polygon Laplacians outperform our method in some cases, their behavior depends on the choice of the parameter λ, which cannot be fixed so as to perform well in all cases. In contrast, our method is parameter-free.

3.4 LIMITATIONS

The presented virtual refinement method can handle both non-planar and non-convex polygons, but there are some aspects that must be considered. In the planar setting, our method is limited to star-shaped polygons. A planar polygon is called star-shaped, if there exists a point $\mathbf{p} \in f$, such that for each other point $\mathbf{x} \in f$ the direct line $\overline{\mathbf{px}}$ between the points lies entirely within the polygon's boundary. The set of all points that satisfy this property is referred to as *kernel*, or $Ker(f)$ of the polygon. Therefore, we can only achieve virtual triangles with completely positive areas if the virtual vertex lies within the kernel of the shape. Otherwise, triangle flips will occur due to the boundary intersections of the virtual triangle edges. Furthermore, while the squared area minimizer generally lies within the kernel of the polygon, there exist star-shaped elements where this is not the case. While this is more of a theoretical issue, as these examples had to be carefully constructed, it should be kept in mind. Additionally, if the polygon is non-planar, the virtual triangulation will only yield an approximation of the shape's original surface. Challenging configurations may lead to less accurate results since the virtual triangulation might not be the best surface representation for the given polygon.

3.5 CONCLUSION

This chapter presented a novel polygon Laplacian, defined by first virtually refining the polygon mesh to a triangle mesh and then coarsening the cotangent Laplacian from the triangle mesh back to the original polygon mesh. The method can be extended to polyhedral meshes by adding a virtual vertex within each cell and refining the tessellation into virtual tetrahedra. The derived Laplace operator exhibits numerous desirable properties, including sparsity, symmetry, positive semi-definiteness, linear precision, and consistency with the divergence and gradient operators, without suffering from an increase in the dimensionality of the linear system. We have evaluated our Laplacian against other state-of-the-art methods and have demonstrated that it performs competitively, providing efficient and high-quality solutions without requiring parameter tuning. However, there are still several open questions that remain to be addressed. For simplicity, we used the omnipresent cotangent Laplacian on the refined tessellation. Nevertheless, what would happen if we use different operators from alternative numerical discretization schemes? Furthermore, various applications require higher accuracy and faster convergence rates that can seldom be achieved with basis functions defined on linear degrees of freedom. Therefore, extending the linear virtual refinement method to higher-order basis functions would extend this flexibility to arbitrary polygonal and polyhedral meshes. Both of these questions will be explored in the following two chapters of this thesis.

4 THE DIAMOND LAPLACIAN

The previous chapter of this thesis introduced the linear virtual refinement method. Now, we shift our focus slightly and investigate using an alternative numerical discretization scheme on the refined tessellation, instead of the previously employed cotangent Laplacian.

The main idea of this chapter focuses on the Discrete Duality Finite Volume approach (DDFV [Her00; Her09; DO05; CH11], detailed in Section 4.2.2), which is part of the Finite Volume Method (FVM). The FVM was initially introduced by Dusinberre [Dus55; Dus61] for the heat equation and can be used on all differential equations that can be expressed through the divergence operator. The fundamental principle of the FVM is that the integral of a differential over a small volume can be expressed as a surface integral of fluxes over the boundary of the same cell [Rap17]. Similar to MFD, Finite Volume discretizations can be considered mimetic as they seek to enforce balance equations for mass, momentum, and energy within each cell, making them particularly well-suited for fluid mechanics problems [LMS14].

However, the conventional derivation of the 2D Laplace operator with Finite Volumes (FV) assumes a Delaunay triangle mesh, more specifically orthogonal dual and primal edges, to prevent positive coefficients on the off-diagonal entries of the stiffness matrix. This is a very strong restriction for our virtual triangulation and one of the many reasons we use DDFV. It is a particular polygonal variant of the FV that allows non-Delaunay meshes and inspired several other generalizations of a similar kind, like the Nodal Discrete Duality scheme by Andreianov and Quenjel [AHQ23] or the Node-Diamond scheme by Quenjel and Beljadid [QB23].

For our polygon Laplacian, we incorporate the primal as well as the (typically non-orthogonal) dual mesh of the DDFV method and accommodate the oblique intersection of primal and corresponding dual elements. Specifically, we define discrete gradients, respectively divergences, per so-called *diamond*: the region spanned by a dual edge and corresponding simplicial primal element. In 2D, the corresponding primal element is an edge; in 3D, it is a triangle. If facets have degrees higher than three, we insert an additional virtual vertex and triangulate the facet. In all cases, the primal element is incident on two cells, and the dual vertices in these cells define two simplices. In other words, in 2D, a diamond is spanned by two triangles; in 3D, it is spanned by two tetrahedra.

In the spirit of the original mesh defining the domain for discretization, we also want the new polygon Laplace operator to map from values at vertices to values at vertices. We achieve this by combining the DDFV approach with the linear

virtual refinement from the previous chapter. This means all vertices except the primal ones are *virtual*: They are defined as affine combinations of primal vertices. These affine combinations are then once again encoded as prolongation matrices and multiplied onto the DDFV Laplacian, thereby effectively hiding the involved diamonds and virtual vertices from the user.
The resulting construction is comparatively straightforward and seamlessly generalizes from arbitrary polygons (Section 4.3) to arbitrary polyhedra (Section 4.4), on which we will further elaborate in the upcoming chapter.

Individual Contribution *I developed the Diamond Laplacian presented in this chapter in collaboration with and under the supervision of Mario Botsch and Marc Alexa. Although the theoretical derivations were a collective effort, I was responsible for implementing the method and conducting all quantitative and qualitative evaluations. The proof as to why the minimum diamond cell is necessary to avoid spurious modes in 3D was provided by Marc Alexa.*

Corresponding Publication *This chapter is based on the following publications:*

Bunge, A., Botsch, M., and Alexa M. (2021). "The Diamond Laplace for Polygonal and Polyhedral Meshes." Computer Graphics Forum, 40(5):217-230.

4.1 PROBLEM STATEMENT

We assume a mesh $\mathcal{M}$ is given. The particular types of meshes we consider are two-dimensional surface meshes immersed in 3D and volumetric meshes embedded in 3D. Because they are the most common operators, we focus on the gradient, the divergence, and the Laplacian.
As in the previous chapters, we denote the elements within the mesh as ordered vertex sets. Edges are denoted as $e_{ij} = (v_i, v_j)$, triangles as $t_{ijk} = (v_i, v_j, v_k)$, and tetrahedra as $t_{ijkl} = (v_i, v_j, v_k, v_l)$, with $|e_{ij}|$, $|t_{ijk}|$, and $|t_{ijkl}|$ referring to their length, (signed) area, and (signed) volume, respectively. We use the operator $*$ to map between primal and dual entities, such that, e.g., v_i^* denotes the dual region of a vertex v_i and e_{ij}^* denotes the dual edge of the primal edge e_{ij}.
Given a polygonal or polyhedral mesh with vertices $\mathcal{V}$, edges $\mathcal{E}$, faces $\mathcal{F}$, cells $\mathcal{C}$, and diamonds $\mathcal{D}$, our aim is to generate the following matrices, representing discrete linear approximation of differential operators:

- The gradient $\mathbf{G}^\diamond \in \mathbb{R}^{d \cdot |\mathcal{D}| \times |\mathcal{V}|}$, where $d \in 2, 3$ is the intrinsic dimension of the mesh and $|\mathcal{D}|$ is the number of diamonds to which the operator assigns constant gradients.
- The divergence $\mathbf{D}^\diamond \in \mathbb{R}^{|\mathcal{V}| \times d \cdot |\mathcal{D}|}$, which we assume to be constructed as

$$\mathbf{D}^\diamond = \mathbf{G}^{\diamond\mathsf{T}} \mathbf{M}_\diamond.$$

Here, $\mathbf{M}_\diamond \in \mathbb{R}^{d\cdot|\mathcal{D}|\times d\cdot|\mathcal{D}|}$ is a diagonal matrix containing d-times the diamond masses.

- The stiffness matrix for the Laplacian is then constructed as

$$\mathbf{D}^{\diamond\mathsf{T}}\mathbf{G}^\diamond = \mathbf{S}^\diamond \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}.$$

This construction ensures that stiffness matrix is symmetric positive semi-definite and that the the discrete operator is consisten with Equation (2.2). In addition, we ask that the Laplacian operator maps constant vectors to zero and has linear precision, i.e. maps linear functions to themselves, which will be further discussed in Section 4.5.

4.2 FINITE VOLUME DISCRETIZATIONS

We believe that FV discretizations [Dro14], in particular the Discrete Duality Finite Volume method [Her00; DO05; Her09; CH11], offer an interesting alternative to deriving discrete differential operators for geometry processing applications. As our approach is inspired by and extends upon DDFV, this chapter will describe both concepts, FV and DDFV, in more detail.

4.2.1 Finite Volume (FV)

Finite Volume methods are based on the idea to consider the *integral* of a *differential* in a small region. There exist a number of identities that allows expressing such integrals of a differential as an integral over only the boundary of the region. For the *divergence* of a vector-valued function $\mathbf{u}$ over a flat two-dimensional region Ω we have

$$\iint_\Omega \operatorname{div}\mathbf{u}\ \mathrm{d}A = \oint_{\partial\Omega} \mathbf{u}^\mathsf{T}\mathbf{n}\,\mathrm{d}s, \tag{4.1}$$

where $\mathbf{n}$ is the outward normal along the boundary $\partial\Omega$. For $\mathbf{u} = u\mathbf{c}$ with constant vector $\mathbf{c}$ and a scalar function u we can exploit the "product rule"

$$\operatorname{div}(u\mathbf{c}) = u\underbrace{\operatorname{div}\mathbf{c}}_{=0} + \mathbf{c}^\mathsf{T}\nabla u = \mathbf{c}^\mathsf{T}\nabla u. \tag{4.2}$$

Plugging this into the divergence theorem above for $\mathbf{c} = \mathbf{e}_k$ (the canonical unit vectors) and combining the results we find the vector-valued identity

$$\iint_\Omega \nabla u\,\mathrm{d}A = \oint_{\partial\Omega} u\,\mathbf{n}\,\mathrm{d}s. \tag{4.3}$$

Applying the divergence theorem to the vector field ∇u we get

$$\iint_{\Omega} \Delta u \, \mathrm{d}A = \oint_{\partial\Omega} \nabla u^{\mathsf{T}} \mathbf{n} \, \mathrm{d}s. \tag{4.4}$$

All identities straightforwardly extend to higher dimensions.
The basic derivation of the Laplace operator with FVs in 2D makes the assumption that the mesh is Delaunay. This means the dual mesh is the Voronoi diagram, such that primal and dual edges are orthogonal. Consider a vertex v_i with position $\mathbf{x}_i$, the dual region associated to it is its Voronoi cell Ω_i. The function values of the unknown piecewise linear function u at vertex v_i are $u_i = u(\mathbf{x}_i)$. For the integrated Laplacian over the region Ω_i, the boundary $\partial\Omega_i$ is piecewise linear and consists of the dual edges e^*_{ij}, with $v_j \in N(v_i)$ denoting the one-ring neighbors of vertex v_i. If we denote by $\mathbf{e}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ and $\mathbf{e}^*_{ij} = \mathbf{x}^*_j - \mathbf{x}^*_i$ the primal/dual edge vectors, respectively, and exploit that the normal $\mathbf{n}$ on the dual edge e^*_{ij} is parallel to $\mathbf{e}_{ij}$ and that the gradient of the piecewise linear function on the vertices points along this edge, we get

$$\begin{aligned}
\iint_{\Omega_i} \Delta u \, \mathrm{d}A &= \sum_{v_j \in N(v_i)} \int_{e^*_{ij}} \nabla u^{\mathsf{T}} \mathbf{n} \, \mathrm{d}s \\
&= \sum_{v_j \in N(v_i)} \int_{e^*_{ij}} \nabla u^{\mathsf{T}} \frac{\mathbf{e}_{ij}}{\|\mathbf{e}_{ij}\|} \, \mathrm{d}s \\
&= \sum_{v_j \in N(v_i)} \int_{e^*_{ij}} (u_j - u_i) \frac{1}{\|\mathbf{e}_{ij}\|} \, \mathrm{d}s \\
&= \sum_{v_j \in N(v_i)} \frac{\left|\mathbf{e}^*_{ij}\right|}{\|\mathbf{e}_{ij}\|} (u_j - u_i) .
\end{aligned} \tag{4.5}$$

The last expression directly describes the construction of a linear operator that maps values at vertices $\{u_i\}$ to the integral over the region associated to vertex v_i of the Laplacian. Note that the (i, j) entry in the matrix $\mathbf{S}$ is given by the (signed) length of the dual edge divided by the length of the primal edges. One obtains exactly the same result with the DEC approach [Hir03], which is based on similar arguments. Interestingly, also the Finite Element Method applied to triangles leads to these weights [PP93]. This suggests that the derivation extends to arbitrary triangulations, albeit carefully assigning *signed* lengths to the dual edges. The resulting positive coefficients for edges without the Delaunay property have undesirable consequences (see, for example, the discussion in [SSC19b]). While this is often accepted for applications in graphics, in the FV community it is not considered admissible, which restricts the meshes to be (weighted) Delaunay. From a practical perspective, however, one is often given a primal mesh and it is costly to generate an orthogonal dual, if one exists [Aur87; Ale20].

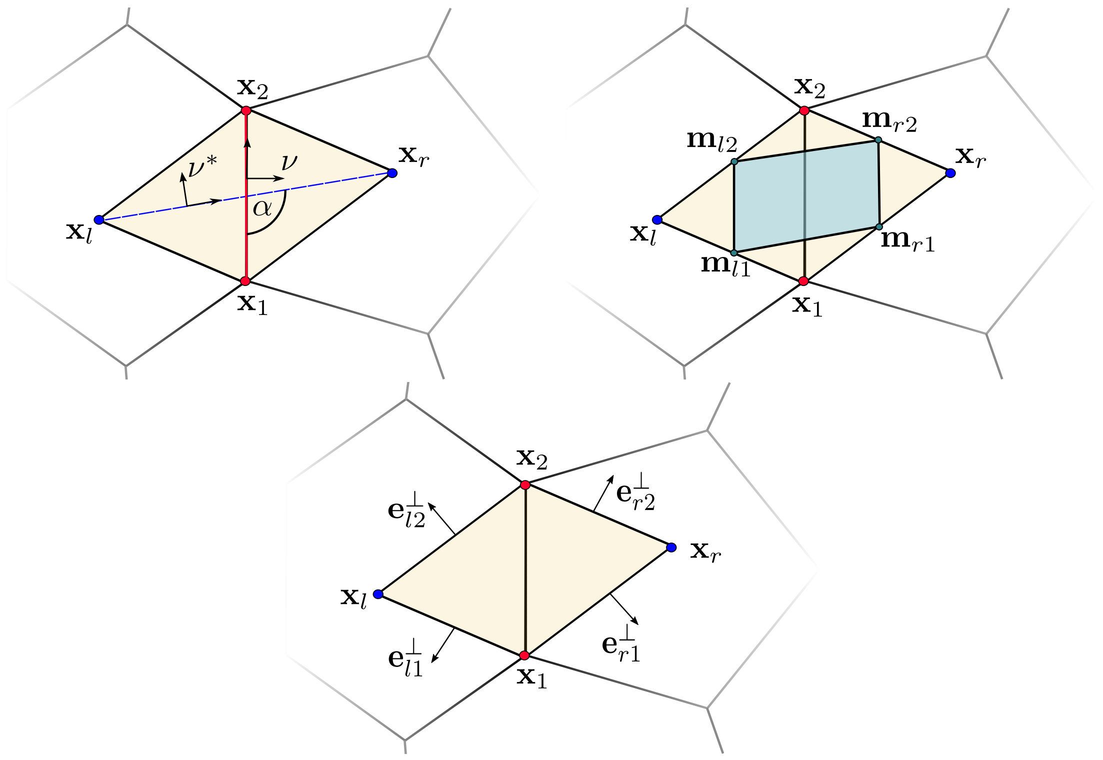


Figure 4.1: *Different formulae and interpretations of the per-diamond gradient in 2D DDFV. Top left: Constructing gradients from primal and dual axes ν, ν^* and their enclosed angle α. Top right: The midpoints $\mathbf{m}_{ij}$ of the diamond edges and their enclosed subarea. Fitting an affine function to the function values at these midpoints is another possibility to obtain the 2D derivation of the diamond gradient. Bottom center: The vectors $\mathbf{e}_{ij}^{\perp}$ orthogonal to the diamond edges $\mathbf{e}_{ij}$ needed to compute our gradient for the diamond cell.*

4.2.2 Discrete Duality Finite Volume (DDFV)

The DDFV method deals with this problem by giving up orthogonality. Rather, the idea is to construct a gradient operator and associate it to the region spanned by a pair of a primal edge (with endpoints $\mathbf{x}_1$ and $\mathbf{x}_2$) and its corresponding dual edge (with endpoints $\mathbf{x}_l$ and $\mathbf{x}_r$). This region, depicted in Figure 4.1, is called a *diamond*.

Notice that a diamond is always a quadrilateral, regardless of the degree of the faces. The DDFV approach is to associate function values u_l and u_r to the dual vertices – thereby introducing a second set of degrees of freedom – and to associate a constant gradient with the diamond. As shown in Figure 4.1, left, we denote by D the diamond built from the four points $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_l, \mathbf{x}_r)$, by $e_{ij} \in D$ its edges, and by $\mathbf{e}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ the corresponding edge vectors. Making use of

Stokes' theorem gives

$$\begin{aligned}\iint_D \nabla u \, \mathrm{d}A &= \oint_{\partial D} u \, \mathbf{n} \, \mathrm{d}s \\ &= \sum_{e_{ij} \in \partial D} \frac{\mathbf{e}_{ij}^{\perp}}{\|\mathbf{e}_{ij}\|} \int_0^1 \|\mathbf{e}_{ij}\| \left((1-t)u_i + tu_j\right) \mathrm{d}t \\ &= \sum_{e_{ij} \in \partial D} \mathbf{e}_{ij}^{\perp} \frac{u_i + u_j}{2}.\end{aligned} \tag{4.6}$$

For the discrete gradient of the diamond we take the mean over the region, so we divide the integral by the area $|D|$, leading to

$$\nabla u|_D = \frac{1}{2\,|D|} \sum_{e_{ij} \in \partial D} \mathbf{e}_{ij}^{\perp} \left(u_i + u_j\right). \tag{4.7}$$

The literature on DDFV [Her00; DO05; Her09; CH11] provides alternative derivations for the per-diamond gradient $\nabla u|_D$ and several interpretations and corresponding formulae:

- Fitting the gradient to directional derivatives along the primal and dual edges:

$$\begin{aligned}\nabla u|_D \cdot (\mathbf{x}_l - \mathbf{x}_r) &= u_l - u_r, \\ \nabla u|_D \cdot (\mathbf{x}_1 - \mathbf{x}_2) &= u_1 - u_2.\end{aligned} \tag{4.8}$$

- Fitting an affine function $w(x, y)$ to the function values $u_{ij} = \frac{1}{2}(u_i + u_j)$ at the midpoints $\mathbf{m}_{ij} = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j)$ of the four diamond edges $e_{ij} \in \partial D$, and taking the gradient ∇w of this affine function (see Figure 4.1, top right). Note that fitting an affine function to the four diamond vertices is an overdetermined problem, while the midpoint fit is uniquely determined.
- A formulation based on the primal/dual axes $\boldsymbol{\nu} = (\mathbf{x}_2 - \mathbf{x}_1)^{\perp} / \|\mathbf{x}_2 - \mathbf{x}_1\|$ and $\boldsymbol{\nu}^* = (\mathbf{x}_r - \mathbf{x}_l)^{\perp} / \|\mathbf{x}_r - \mathbf{x}_l\|$ as well as their enclosed angle α (Figure 4.1, top left):

$$\nabla u|_D = \frac{1}{\sin\alpha} \left(\frac{u_l - u_r}{\|\mathbf{x}_l - \mathbf{x}_r\|} \boldsymbol{\nu} + \frac{u_1 - u_2}{\|\mathbf{x}_1 - \mathbf{x}_2\|} \boldsymbol{\nu}^* \right). \tag{4.9}$$

Although these formulations are all equivalent, we believe our formulation (4.6) to be more intuitive when handling boundary cases and when generalizing to polyhedral meshes in Section 4.4. For boundary edges, the diamond consists of a single triangle $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_l)$ only. The typical DDFV approach is to replace $\mathbf{x}_r$ by the edge midpoint $\frac{1}{2}(\mathbf{x}_1 + \mathbf{x}_2)$ and to properly deal with degenerate edges/faces. In contrast, our formulation (4.7) remains unchanged.
The divergence $\operatorname{div} u$ and the Laplacian $\Delta u = -\operatorname{div} \nabla u$ can be obtained through very similar derivations. The resulting DDFV gradient operator maps

from function values at primal *and* dual vertices to gradients at diamonds, the divergence operator from vectors at diamonds to scalars at primal/dual vertices. The DDFV Laplacian therefore maps functions values at primal/dual vertices to their Laplacians sampled at primal/dual vertices.

4.3 DIAMOND LAPLACE FOR SURFACE MESHES

In its standard formulation, the DDFV operators are not directly useful for applications in computer graphics and geometry processing, since they have two main drawbacks: First, by introducing function values at dual vertices it significantly increases the number of degrees of freedom (DoF) to be solved for. For instance, the DoFs are roughly tripled for triangle meshes and roughly doubled for quad meshes. Second, the approach is defined for planar 2D meshes only, but not for two-manifold surface meshes embedded in 3D, which we are mostly interested in.
In this section, we address both problems. Replacing the extrinsic per-diamond gradient with an intrinsic version w.r.t. the polygonal mesh allows us to generalize the 2D DDFV scheme to embedded surface meshes (Section 4.3.1). For the second part, we follow the idea of our linear virtual refinement method and represent the dual DoFs as interpolations of the primal DoFs. These vertices are then incorporated through prolongation and restriction matrices that remove the dual DoFs and keep only the primal ones.

4.3.1 **Intrinsic Gradient**

Compared to mesh faces, diamonds are the better entity to associate gradients with, since for general polygonal meshes higher-degree faces might not be planar and typically cannot be flattened without introducing distortion. Diamonds spanned by a pair of primal vertices $\mathbf{x}_1, \mathbf{x}_2$ and dual vertices $\mathbf{x}_l, \mathbf{x}_r$ are not necessarily planar in the 3D embedding, but can be isometrically unfolded into the plane around their primal "hinge" edge $(\mathbf{x}_1, \mathbf{x}_2)$. We can therefore represent the diamond in an intrinsic 2D coordinate system, which allows us to then directly apply the gradient construction of (4.7).
It is convenient to choose the primal edge as the first coordinate axis, i.e.,

$$\mathbf{r} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}. \tag{4.10}$$

The second coordinate axis has to be orthogonal to this axis, contained in the planes spanned by the two triangles $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_l)$ and $(\mathbf{x}_2, \mathbf{x}_1, \mathbf{x}_r)$, and consistently oriented w.r.t. $\mathbf{r}$. We achieve this by projecting the edges e_{1l} and e_{1r} onto the orthogonal complement of the first axis $\mathbf{r}$ and normalizing the result:

$$\begin{aligned}\tilde{\mathbf{v}}_l &= \left(\mathbf{I} - \mathbf{r}\mathbf{r}^\mathsf{T}\right)(\mathbf{x}_l - \mathbf{x}_1), \quad \mathbf{v}_l = \tilde{\mathbf{v}}_l / \|\tilde{\mathbf{v}}_l\|, \\ \tilde{\mathbf{v}}_r &= \left(\mathbf{I} - \mathbf{r}\mathbf{r}^\mathsf{T}\right)(\mathbf{x}_1 - \mathbf{x}_r), \quad \mathbf{v}_r = \tilde{\mathbf{v}}_r / \|\tilde{\mathbf{v}}_r\|.\end{aligned} \tag{4.11}$$

Notice that both directions $\mathbf{v}_l$ and $\mathbf{v}_r$ are consistently oriented, are intrinsically in one plane, and form an orthonormal frame with $\mathbf{r}$. In this frame, the 2D coordinates of the four diamond vertices are

$$\begin{aligned}\mathbf{x}_1^{2\mathrm{D}} &= (0,0)^\mathsf{T}, \\ \mathbf{x}_2^{2\mathrm{D}} &= (\|\mathbf{x}_2 - \mathbf{x}_1\|, 0)^\mathsf{T}, \\ \mathbf{x}_l^{2\mathrm{D}} &= (\mathbf{r}, \mathbf{v}_l)^\mathsf{T}(\mathbf{x}_l - \mathbf{x}_1), \\ \mathbf{x}_r^{2\mathrm{D}} &= (\mathbf{r}, \mathbf{v}_r)^\mathsf{T}(\mathbf{x}_r - \mathbf{x}_1).\end{aligned} \tag{4.12}$$

These 2D coordinates can now be used in the gradient construction of Equation (4.7), yielding an *intrinsic* 2D gradient per diamond.
From (4.7) we can then directly read off the entries for the diamond's *gradient operator* matrix $\mathbf{G}_D^\diamond \in \mathbb{R}^{2\times 4}$ by noticing that the value i depends only on the two diamond edges incident on it. Therefore, the i-th column of $\mathbf{G}_D^\diamond$ is

$$\mathbf{G}_D^\diamond(:, i) = \frac{1}{2\,|D|} \sum_{e_{ij} \in \partial D} \mathbf{e}_{ij}^\perp. \tag{4.13}$$

The matrix $\hat{\mathbf{G}}^\diamond$ for the global gradient operator, mapping from function values at primal and dual vertices to gradients at diamonds, is then assembled from all diamond gradient matrices

$$\hat{\mathbf{G}}^\diamond = \bigoplus_{D \in \mathcal{D}} \mathbf{G}_D^\diamond, \tag{4.14}$$

where $\bigoplus$ is the assembly operator that scatters and accumulates the entries of the local matrices into the global matrix.

4.3.2 Dual Vertices as Affine Combinations

As previously mentioned, the approach to remove the dual DoFs from the DDFV formulation is inspired by the linear virtual refinement method (Chapter 3). It also introduces dual vertices into primal faces, but represents their position and function values as affine combinations of the positions/values of the face's vertices.
Consider a general polygonal face f with n_f vertices $v_i \in f$ and index j. We construct the dual face point $\mathbf{x}_f$, which takes the role of $\mathbf{x}_l$ or $\mathbf{x}_r$ in the gradient construction described above, as an affine combination of the face vertices

$$\mathbf{x}_f = \sum_{v_i \in f} w_{ji}\, \mathbf{x}_i \quad \text{with} \quad \sum_{v_i \in f} w_{ji} = 1. \tag{4.15}$$

The dual DoFs, i.e., the function values at dual face vertices $\mathbf{x}_f$, are then represented in terms of the primal DoFs by the same affine combination:

$$u(\mathbf{x}_f) = \sum_{v_i \in f} w_{ji}\, u_i. \tag{4.16}$$

For a polygonal mesh with $|\mathcal{V}|$ vertices and $|\mathcal{F}|$ faces, this construction can be packed into an $(|\mathcal{V}| + |\mathcal{F}|) \times |\mathcal{V}|$ prolongation matrix $\mathbf{P}$ with entries

$$\mathbf{P}_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and } i < |\mathcal{V}|, \\ w_{ki} & \text{if } i = |\mathcal{V}| + k \text{ and vertex } v_j \in f_k, \\ 0 & \text{otherwise,} \end{cases} \tag{4.17}$$

just as in Equation (3.6). Combining the gradient matrix (4.14) and the prolongation matrix (4.17) yields our gradient operator for polygonal meshes

$$\mathbf{G}^{\diamond} = \hat{\mathbf{G}}^{\diamond}\, \mathbf{P} \,\in \mathbb{R}^{2|\mathcal{E}| \times |\mathcal{V}|}, \tag{4.18}$$

where $|\mathcal{E}|$ denotes the number of edges (and therefore of diamonds) in the mesh. This matrix maps scalar function values u_i at *primal vertices* to 2D intrinsic gradient vectors $\nabla u|_D$ at diamonds.
While for standard FV methods with orthogonal duals the dual point $\mathbf{x}_f$ is the circum-center of triangle f, the canonical choice for general polygonal meshes in the DDFV literature [Her00; DO05; Her09; CH11] is the face's barycenter. However, as the barycenter is not necessarily inside a non-convex (planar) face, we instead follow the suggested virtual vertex placement of Section 3.2.3 and compute the virtual dual vertex $\mathbf{x}_{f_j}$ associated with a polygonal face f_j (and its affine weights w_{ji} respectively) by minimizing the sum of squared triangle areas (see Equation (3.11)).

4.3.3 Divergence and Laplacian

With the intrinsic gradient (4.18) in place, we can now define the discrete divergence and Laplacian. The DDFV discretization of the divergence operator leads to a matrix $\hat{\mathbf{G}}^{\diamond\mathsf{T}}\, \mathbf{M}_{\diamond}$, which we combine with the transposed prolongation (or restriction) matrix to get the diamond divergence matrix

$$\mathbf{D}^{\diamond} = \mathbf{P}^{\mathsf{T}} \hat{\mathbf{G}}^{\diamond\mathsf{T}}\, \mathbf{M}_{\diamond}. \tag{4.19}$$

Here, $\hat{\mathbf{G}}^{\diamond}$ is the gradient matrix of (4.14) and $\mathbf{M}_{\diamond}$ is a $2\,|\mathcal{E}| \times 2\,|\mathcal{E}|$ diagonal matrix with the area $|D_i|$ of diamond D_i in its entries $(2i-1, 2i-1)$ and $(2i, 2i)$. This operator maps intrinsic 2D vectors at diamonds to scalar values at primal vertices.

The (integrated) diamond Laplace operator is finally defined as the diamond divergence of the diamond gradient, i.e.,

$$\mathbf{S}^{\diamond} \;=\; \mathbf{D}^{\diamond}\,\mathbf{G}^{\diamond} \;=\; \mathbf{P}^{\mathsf{T}}\hat{\mathbf{G}}^{\diamond\mathsf{T}}\,\mathbf{M}_{\diamond}\,\hat{\mathbf{G}}^{\diamond}\mathbf{P}, \tag{4.20}$$

and maps from vertices to vertices. The pointwise Laplacian is obtained as $-(\mathbf{M}^{\diamond})^{-1}\mathbf{S}^{\diamond}$ by multiplying with the inverse of the mass matrix $\mathbf{M}^{\diamond}$. This mass matrix is defined as

$$\mathbf{M}^{\diamond} = \mathbf{P}^{\mathsf{T}}\,\hat{\mathbf{M}}^{\diamond}\,\mathbf{P} \tag{4.21}$$

from the standard DDFV diagonal mass matrix $\hat{\mathbf{M}}^{\diamond}$

$$\hat{\mathbf{M}}^{\diamond}_{ii} = \begin{cases} \sum_{D \ni v_i} \frac{1}{4}\,|D| & \text{if } v_i \text{ is a primal vertex,} \\ \sum_{D \ni v_i} \frac{1}{4}\,|D| & \text{if } v_i \text{ is a dual face vertex,} \\ 0 & \text{otherwise,} \end{cases} \tag{4.22}$$

which assigns to the four (primal and dual) vertices of a diamond one fourth of its area. The "sandwiching" with $\mathbf{P}^{\mathsf{T}}$ and $\mathbf{P}$ distributes the mass from primal and dual vertices to the primal vertices only. Note that the sandwiching leads to a non-diagonal mass matrix $\mathbf{M}^{\diamond}$. We avoid lumping this matrix to a diagonal matrix, since numerical results have shown that the initial matrix leads to higher accuracy of our operator. Notice that in above construction, the dual points do not have to be inserted into the mesh explicitly, nor do the matrices $\hat{\mathbf{G}}^{\diamond}, \mathbf{M}_{\diamond}, \mathbf{P}$ have to be built explicitly. Instead, the matrices $\mathbf{G}^{\diamond}$ and $\mathbf{M}^{\diamond}$ can directly be constructed through a diamond-based matrix assembly, which *implicitly* computes the virtual vertices and their affine weights, similar to the construction described in Algorithm 1 of Chapter 3.

4.4 DIAMOND LAPLACE FOR VOLUME MESHES

One particular advantage of our Diamond Laplacian is that it can be generalized to 3D polyhedral meshes in an intuitive and consistent manner. Given a general polyhedral mesh with vertices $\mathcal{V}$, edges $\mathcal{E}$, faces $\mathcal{F}$, and cells $\mathcal{C}$, we will define diamonds $\mathcal{D}$ from primal and dual vertices. The starting point of our construction is the generalization of the (integrated) gradient of a function u over a diamond. Analogous to the 2D case, given the gradient operator $\mathbf{G}^{\diamond}$, we also have a divergence operator $\mathbf{D}^{\diamond}$ and can then assemble the weak form of the Laplacian $\mathbf{S}^{\diamond} = \mathbf{D}^{\diamond}\,\mathbf{G}^{\diamond}$. Representing the dual vertices as affine combinations of primal vertices will again define the sandwiching operator $\mathbf{P}^{\mathsf{T}}(\cdot)\mathbf{P}$ that removes the dual DoFs. In the following we provide the details of these steps, in particular where they deviate from the case for surface meshes.

4.4.1 Integrated Gradient

Before we focus on the particular shape and construction of the diamonds, we derive the integral of the gradient (and, by analogy, divergence) for an arbitrary region Ω bounded by a *triangulated* surface. We assume the function over the triangulated boundary to be linear on the triangles, defined by values u_i at vertices v_i. If the triangles are given as triples of vertices $t_{ijk} = (v_i, v_j, v_k) \in \partial\Omega$, we get

$$\begin{aligned}\iiint_{\Omega} \nabla u \, \mathrm{d}V &= \oiint_{\partial\Omega} u \mathbf{n} \, \mathrm{d}A \\ &= \sum_{t_{ijk}\in\partial\Omega} \frac{\mathbf{a}_{ijk}}{\|\mathbf{a}_{ijk}\|} \int_0^1 \int_0^t \|\mathbf{a}_{ijk}\| \left((1-s-t)u_i + su_j + tu_k\right) \, \mathrm{d}s \, \mathrm{d}t \\ &= \sum_{t_{ijk}\in\partial\Omega} \mathbf{a}_{ijk} \frac{u_i + u_j + u_k}{3},\end{aligned} \tag{4.23}$$

where

$$\mathbf{a}_{ijk} = \frac{1}{2}(\mathbf{x}_j - \mathbf{x}_i) \times (\mathbf{x}_k - \mathbf{x}_i) \tag{4.24}$$

is the *area vector* of triangle (v_i, v_j, v_k), i.e., the vector pointing in outward normal direction and with magnitude equal to the area of the triangle (see Figure 4.3, left). Taking the mean over the region by dividing the integral by the volume $|\Omega|$ leads to

$$\nabla u|_{\Omega} = \frac{1}{3\,|\Omega|} \sum_{t_{ijk}\in\partial\Omega} \mathbf{a}_{ijk} \left(u_i + u_j + u_k\right). \tag{4.25}$$

The local gradient operator for the region Ω, mapping values at the vertices $i \in \partial\Omega$ to a constant 3D gradient vector, is then built in a column-wise manner as

$$\mathbf{G}^{\diamond}_{\Omega}(:, i) = \frac{1}{3\,|\Omega|} \sum_{t_{ijk}\in\partial\Omega} \mathbf{a}_{ijk}, \tag{4.26}$$

which is consistent with the 2D version in Equation (4.13).

4.4.2 Diamond Rings and Minimal Diamonds

The canonical choice for a diamond in a volumetric mesh would be associated with a dual edge e_{lr} with endpoints $\mathbf{x}_l, \mathbf{x}_r$. These two dual vertices, together with the primal vertices $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ of the face $f = e^*_{lr}$ that is dual to e_{lr}, define a region that is bounded by two triangle fans spanned by $\mathbf{x}_l$ or $\mathbf{x}_r$ and two neighboring vertices $\mathbf{x}_i, \mathbf{x}_{i+1}$ of the face f. Given that the integrated gradient

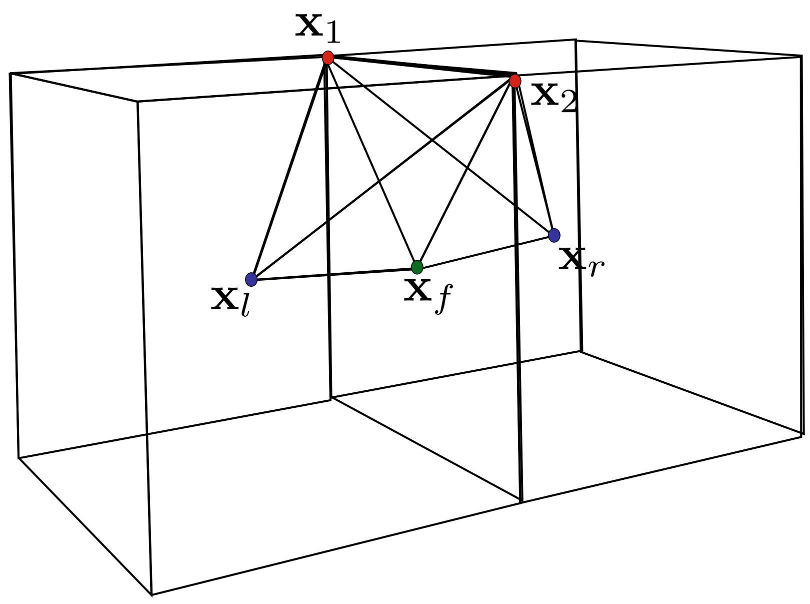


Figure 4.2: *A minimal diamond spanned by two cell points* $\mathbf{x}_l, \mathbf{x}_r$*, a face point* $\mathbf{x}_f$*, and a primal edge* $\mathbf{x}_1, \mathbf{x}_2$*.*

can be computed easily for this region as shown above, it may be tempting to assign a gradient to each such diamond (as done in [Her09]). Yet, similar to other constructions for non-simplicial meshes [AW11; GBD20], constructing the gradient, divergence, and Laplacian in this way and then sandwiching the resulting matrices leads to *spurious modes*, i.e., a Laplacian operator with more than the constant functions in its kernel. This would be a serious drawback, and is a known limitation of the CeVe DDFV method [Her09], as discussed for instance in [ABH13].

Since we are adding a dual vertex to each cell, all vertices in a cell become connected in the operator. Consequently, adding a dual vertex $\mathbf{x}_f$ to each face f introduces no additional non-zeros in the operator. Based on this virtual face vertex, the diamond is decomposed into a *ring of diamonds*, where each individual diamond is *minimal*, i.e., consists of two tetrahedra with tips $\mathbf{x}_l, \mathbf{x}_r$ and a base triangle $(\mathbf{x}_i, \mathbf{x}_{i+1}, \mathbf{x}_f)$, as shown in Figure 4.2. Basing the construction on these minimal elements ensures that the kernel of the Laplace operator only contains the constants.

Incidentally, minimal diamonds are the right analogy to 2D diamonds, in the following sense: Consider a minimal diamond defined by $\mathbf{x}_l, \mathbf{x}_r$ and $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ and the midpoints of the 6 edges emanating from $\mathbf{x}_l$ and $\mathbf{x}_r$ (see Figure 4.3, right):

$$\mathbf{m}_{li} = \frac{1}{2}(\mathbf{x}_l + \mathbf{x}_i), \quad \mathbf{m}_{ri} = \frac{1}{2}(\mathbf{x}_r + \mathbf{x}_i), \quad i = 1, 2, 3. \tag{4.27}$$

We observe that these six midpoints form a *parallelotope*:

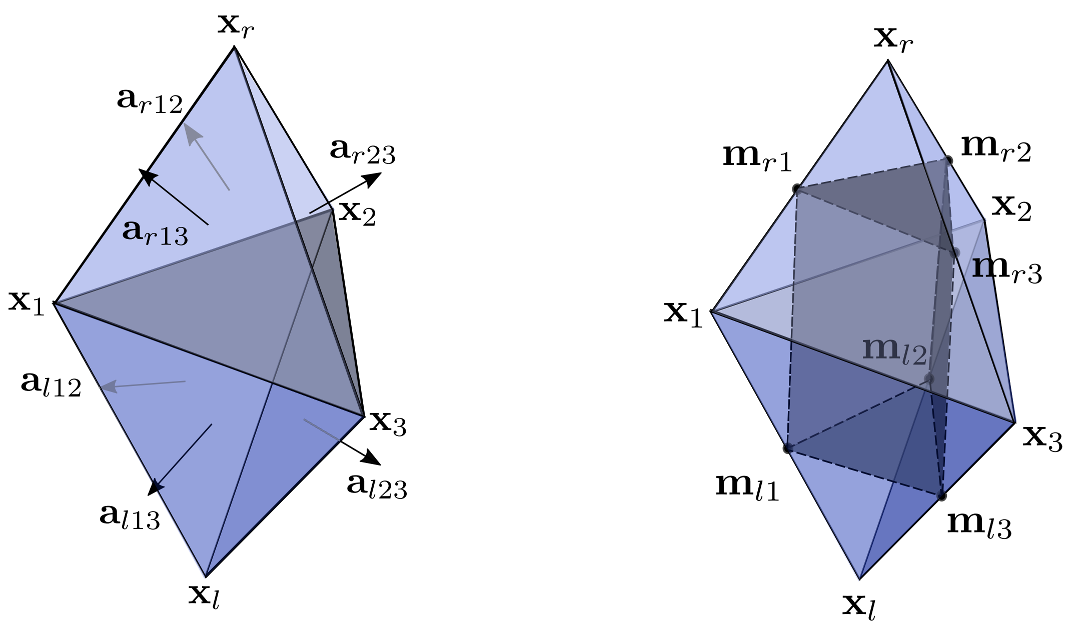


Figure 4.3: *For a minimal diamond consisting of two tetrahedra, the gradient can be computed from the area vectors $\mathbf{a}_{ijk}$ of its faces (left) or by fitting an affine function to edge midpoints $\mathbf{m}_{ij}$ (right).*

1. The two triangles $(\mathbf{m}_{l1}, \mathbf{m}_{l2}, \mathbf{m}_{l3})$ and $(\mathbf{m}_{r1}, \mathbf{m}_{r2}, \mathbf{m}_{r3})$ are parallel to the triangle $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, in fact, translates scaled by a factor of $\frac{1}{2}$.
2. The quad $\mathbf{m}_{l1}, \mathbf{m}_{r1}, \mathbf{m}_{r2}, \mathbf{m}_{l2}$ is a planar parallelogram, and likewise for the other two quads. All edges $\mathbf{m}_{li} - \mathbf{m}_{ri}$ connecting corresponding points on opposite sides are copies of the vector $\mathbf{x}_l - \mathbf{x}_r$, scaled by a factor of $\frac{1}{2}$.

This means any two edges of the triangle $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ together with the vector $\mathbf{x}_l - \mathbf{x}_r$ span the linear part of the affine space defined by the six points. Hence, an affine function can uniquely be fitted to these midpoints – analogously to the 2D parallelogram version show in Figure 4.1, bottom center – and the gradient of this function can be used as the diamond gradient (giving the same result as Equation (4.26)).

4.4.3 Dual Vertices as Affine Combinations

There have been several extensions of the DDFV scheme to volumetric meshes, see Hubert and colleagues [CH11; ABH+12] for a good overview. Most 3D DDFV methods define the gradient on minimal diamonds, as proposed above, but require the insertion of additional vertices (and their associated DoFs) per cell, face, and edge, thereby significantly increasing the number of degrees of freedom. As previously described in Section 3.2.7, our construction requires

virtual vertices per cell and face only, but their DoFs are eventually removed by the sandwiching operator.
Analogous to the surface case, we first insert for each face the point that minimizes the sum of squared triangle areas (see Equation (3.11)), turning each face into a fan of (virtual) triangles. For a polyhedral cell c, the virtual point $\mathbf{x}_c$ is then computed to minimize the sum of squared tetrahedron volumes:

$$\mathbf{x}_c = \arg\min_{\mathbf{x}} \sum_{t_{ijk}\in\partial c} \operatorname{vol}\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}\right)^2 \tag{4.28}$$

For a cell c with m vertices (consisting of primal vertices and virtual face vertices), the above minimization requires to solve an $m \times m$ linear system for the affine weights defining $\mathbf{x}_c$.
This two-step sandwiching procedure results in the same two prolongation matrices $\mathbf{P}_F$ and $\mathbf{P}_C$ previously described in Equation (3.28). As already mentioned, they insert face and cell points, respectively, and are then combined into the final prolongation matrix

$$\mathbf{P} = \mathbf{P}_C\, \mathbf{P}_F. \tag{4.29}$$

4.4.4 Gradient, Divergence, Laplace

The global gradient operator $\hat{\mathbf{G}}^\diamond$, mapping values at primal vertices and dual face/cell points, is again constructed by assembling per-diamond gradient matrices $\mathbf{G}_D^\diamond$, and is then combined with the prolongation matrix to yield $\mathbf{G}^\diamond$

$$\hat{\mathbf{G}}^\diamond = \bigoplus_{D\in\mathcal{D}} \mathbf{G}_D^\diamond, \qquad \mathbf{G}^\diamond = \hat{\mathbf{G}}^\diamond\, \mathbf{P}, \tag{4.30}$$

where $\bigoplus$ again is the matrix assembly operator. Following the 2D derivation, the divergence and Laplacian operators for polyhedral meshes become

$$\mathbf{D}^\diamond = \mathbf{P}^\mathsf{T} (\hat{\mathbf{G}}^\diamond)^\mathsf{T}\, \mathbf{M}_\diamond, \qquad \mathbf{S}^\diamond = \mathbf{D}^\diamond\, \mathbf{G}^\diamond, \tag{4.31}$$

with a diagonal matrix $\mathbf{M}_\diamond$ containing diamond volumes. The mass matrix $\mathbf{M}^\diamond = \mathbf{P}^\mathsf{T}\hat{\mathbf{M}}^\diamond\mathbf{P}$, required for the point-wise Laplacian $(\mathbf{M}^\diamond)^{-1}\mathbf{S}^\diamond$, is defined in terms of the diagonal matrix $\hat{\mathbf{M}}^\diamond$, which distributes the volumes of the (minimal) diamonds D to the primal vertices, face vertices, and cell vertices:

$$\hat{\mathbf{M}}_{ii}^\diamond = \begin{cases} \sum_{D\ni i} \frac{1}{6}\,|D| & \text{if } v_i \text{ is a primal vertex,} \\ \sum_{D\ni i} \frac{1}{6}\,|D| & \text{if } v_i \text{ is a face vertex,} \\ \sum_{D\ni i} \frac{1}{4}\,|D| & \text{if } v_i \text{ is a cell vertex,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.32}$$

Analogous to the surface construction, we avoid lumping the mass matrix and instead work with the non-diagonal matrix $\mathbf{M}^\diamond$.

4.5 PROPERTIES

We analyze the properties of our Diamond Laplacian with respect to the criteria listed in Section 2.1.1.

Symmetry, Definiteness By construction, the stiffness matrix of the Diamond Laplacian $\mathbf{S}^{\diamond} = \mathbf{D}^{\diamond}\mathbf{G}^{\diamond} = \mathbf{P}^{\mathsf{T}}(\hat{\mathbf{G}}^{\diamond})^{\mathsf{T}}\mathbf{M}_{\diamond}\hat{\mathbf{G}}^{\diamond}\mathbf{P}$ is a real-valued symmetric positive semi-definite matrix, since the diagonal matrix $\mathbf{M}_{\diamond}$ (containing diamond areas/volumes) is also symmetric positive definite.

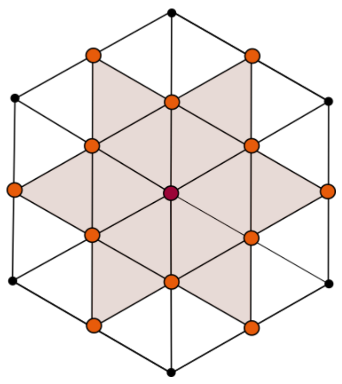
Linear Precision If the mesh is flat, i.e., a polygon mesh embedded in a plane, respectively a polyhedral mesh embedded in 3D, then we expect the discrete Laplacian to vanish on linear functions away from the boundary of the domain. The DDFV gradient $\hat{\mathbf{G}}^{\diamond}$ of a linear function over a closed region with polygonal or polyhedral boundary reproduces the constant gradient of this function. The divergence operator $(\hat{\mathbf{G}}^{\diamond})^{\mathsf{T}}\mathbf{M}_{\diamond}$ is exact on the resulting constant vector fields, leading to linear precision of the refined DDFV stiffness matrix $(\hat{\mathbf{G}}^{\diamond})^{\mathsf{T}}\mathbf{M}_{\diamond}\hat{\mathbf{G}}^{\diamond}$ [DO05; Her09; CH11]. The sandwiching $\mathbf{P}^{\mathsf{T}}(\cdot)\,\mathbf{P}$ preserves this linear precision, as discussed in Section 3.2.8 of the previous chapter.

Null-Space The Diamond Laplacian has a one-dimensional kernel containing only constant functions. It is obvious that constant functions are sufficient for the gradient to vanish, implying that they are in the kernel of the Laplacian. It remains to show that constant values are necessary for the gradient to vanish.
We explain the situation for minimal diamonds in 3D – the case for surface meshes works analogously. The gradient of a minimal diamond can be interpreted as the gradient of the affine function interpolating the values on the edge midpoints $\mathbf{m}_{li}$, $\mathbf{m}_{ri}$, $i = 1, 2, 3$ (Section 4.4.2). For these values to be identical it is necessary that (i) the values at $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ are identical and (ii) the values at $\mathbf{x}_l$ and $\mathbf{x}_r$ are identical. Because the mesh is connected, it follows that the values at all primal vertices and dual vertices need to be identical. Notice that this argument cannot be extended to diamonds with a non-triangular base: If the base is a polygon with more than three vertices already the gradient within this polygon may vanish for non-constant values on the vertices. This problem for diamonds with non-triangular base has also been described in the DDFV literature (cf. [ABH13]).
It remains to explain why the constant values on primal vertices and the constant values on dual vertices are identical. In our setup this follows directly from the fact that values on dual vertices are affine combinations of the values in primal vertices. In other words, while $\hat{\mathbf{G}}^{\diamond}$ may have a two-dimensional kernel, the kernel of $\hat{\mathbf{G}}^{\diamond}\mathbf{P}$ is guaranteed to contain only the constant functions. As

long as the diamond mass matrix $\mathbf{M}_\diamond$ has full rank this implies that also $\mathbf{S}^\diamond$ has the desired kernel.

Lastly, note that the DDFV literature only considers meshes with boundary, where values on primal and dual vertices are connected through identification on boundary edges. In this case, $\hat{\mathbf{G}}^\diamond$ already has the desired kernel [ABH13]. For meshes without boundary this fails. Our sandwiching approach rectifies the situation.

Locality The Diamond Laplacian is local, but less local than related existing schemes, since the diamond gradient couples neighboring cells and the sandwiching couples all primal vertices incident on a cell. For simplicial meshes, the cotan Laplacian of a vertex v_i depends on all vertices sharing an edge with v_i. For the previously introduced polygonal Laplacians [AW11; BHK+20; GBD20] it depends on all vertices sharing a face with v_i. For the Diamond Laplacian it depends on the vertices of (i) the cells incident on vertex v_i and (ii) the cells sharing a face with these cells. This set is larger than the vertices in the edge-based or cell-based one-ring neighborhood, but generally smaller than edge-based two-ring neighborhood. For instance, on a regular triangle mesh the Laplacian of vertex v_i depends on 12 neighbors, which is in between the 6 and 18 vertices of the one-ring and two-ring neighborhoods, respectively (see inset figure).



Maximum Principle The maximum principle for discrete Laplace operators is commonly derived from the sign structure of the operator matrix. If the diagonal elements are all positive and the off-diagonal elements are all negative, then the operator is an *M-matrix*, implying the maximum principle. The sign of the off-diagonal entries in $\mathbf{S}^\diamond$ depends on the input mesh, and like other Laplacians the Diamond Laplacian, in general, has no maximum principle. A Delaunay *triangle* mesh guarantees the desired signs for the coefficients of the cotangent operator. For Delaunay *tetrahedral* meshes, the cotangent operator has no maximum principle. The DEC construction does provide the maximum principle for Delaunay meshes, but may lack semi-definiteness if the mesh is not Delaunay [AHK+20]. The Diamond Laplacian, in contrast, has no maximum principle even for Delaunay triangle or tetrahedral meshes.

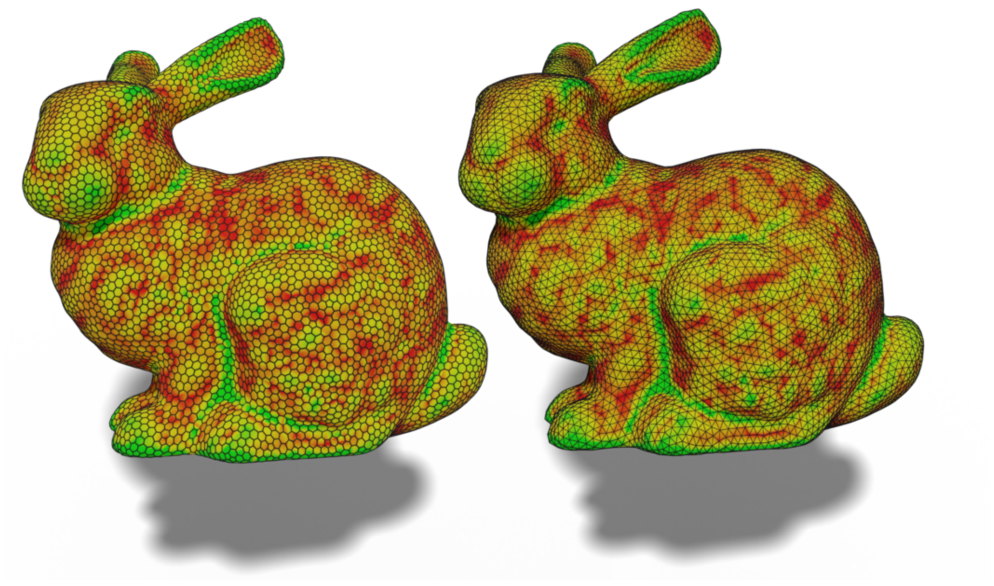


Figure 4.4: *Color-coded absolute mean curvature for two different tessellations computed with the Diamond Laplacian (left: hexagon-dominant, right: triangles). The results are visually very similar.*

4.6 EVALUATION

In this chapter, we will expand the introduced comparisons for non-simplicial meshes of Section 3.3 with the results of the newly defined Diamond Laplacian. As in the evaluation of the linear virtual refinement method, we will first highlight some qualitative results obtained with the Diamond Laplacian, but the main focus of this chapter will lie on the discussion of the quantitative tests.

4.6.1 Surface Meshes

Mean Curvature When applied to the coordinate function of the surface mesh, the Laplace operator yields an approximation of the integrated mean curvature vector (see Equation (3.30)). We approximate the point-wise mean curvature H at a vertex v_i with the Diamond Laplacian, as visualized for a triangle mesh and a general polygon mesh in Figure 4.4. Note that in contrast to the other polygon Laplacians presented in this thesis [AW11; BHK+20; GBD20; MKB+08], the Diamond Laplacian does *not* reduce to the cotangent Laplacian in case of triangle meshes, but instead provides a more accurate discretization. This will become more obvious in the quantitative evaluation.

Geodesic in Heat Since the gradient and divergence operators are directly involved in several computation steps, the gradient defined on the diamonds

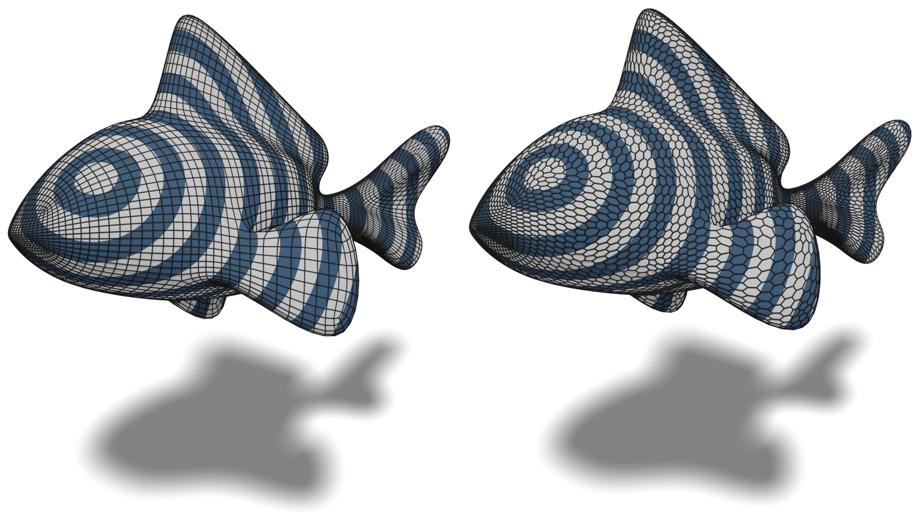


Figure 4.5: *Geodesic distances obtained with the Diamond Laplacian, being visually identical despite different tessellations.*

makes the application of this method natural. We already mentioned that, depending on the employed Laplacian, the number and dimension of the gradient vectors in the geodesic in heat method vary. As described in Section 4.3.1 the Diamond gradient operator leads to intrinsic two-dimensional gradients that are associated with the virtual diamond cells and therefore with the edges of the original mesh. The distances on different tessellations are qualitatively compared in Figure 4.5. The quantitative evaluation displayed in Figure 4.6 and Figure 4.7 shows the results for different unit grid and unit sphere tessellations. In general, the Diamond Laplace and de Goes et al.'s method [GBD20] for $\lambda = 0.1$ yield the lowest error rates for the spherical tessellations, independent of the chosen time step. This is a good indicator of the quality of the gradient and divergence operators defined on diamonds compared to other constructions. On planar grids, the Diamond Laplacian retains its accuracy for the mean edge time step but is negatively affected if ε is increased to the maximum diagonal time step. In this setting, other operators like those obtained with the linear virtual refinement method or Alexa and Wardetzky's method [AW11] for $\lambda = 0.5$ can surpass it.

We also evaluated the accuracy of the Diamond operator constructed by placing dual vertices $\mathbf{x}_f$ at face *centroids* instead of the minimizer of the squared triangle areas. While the centroid is typically employed in the DDFV literature, its performance on non-convex tessellations (similar to the L-plane and Tetris meshes displayed in Figure 3.3) yields worse results than the area minimizer, since flipped triangles lead to unfavorable diamond cells, which in turn reduce the overall performance of the operator.

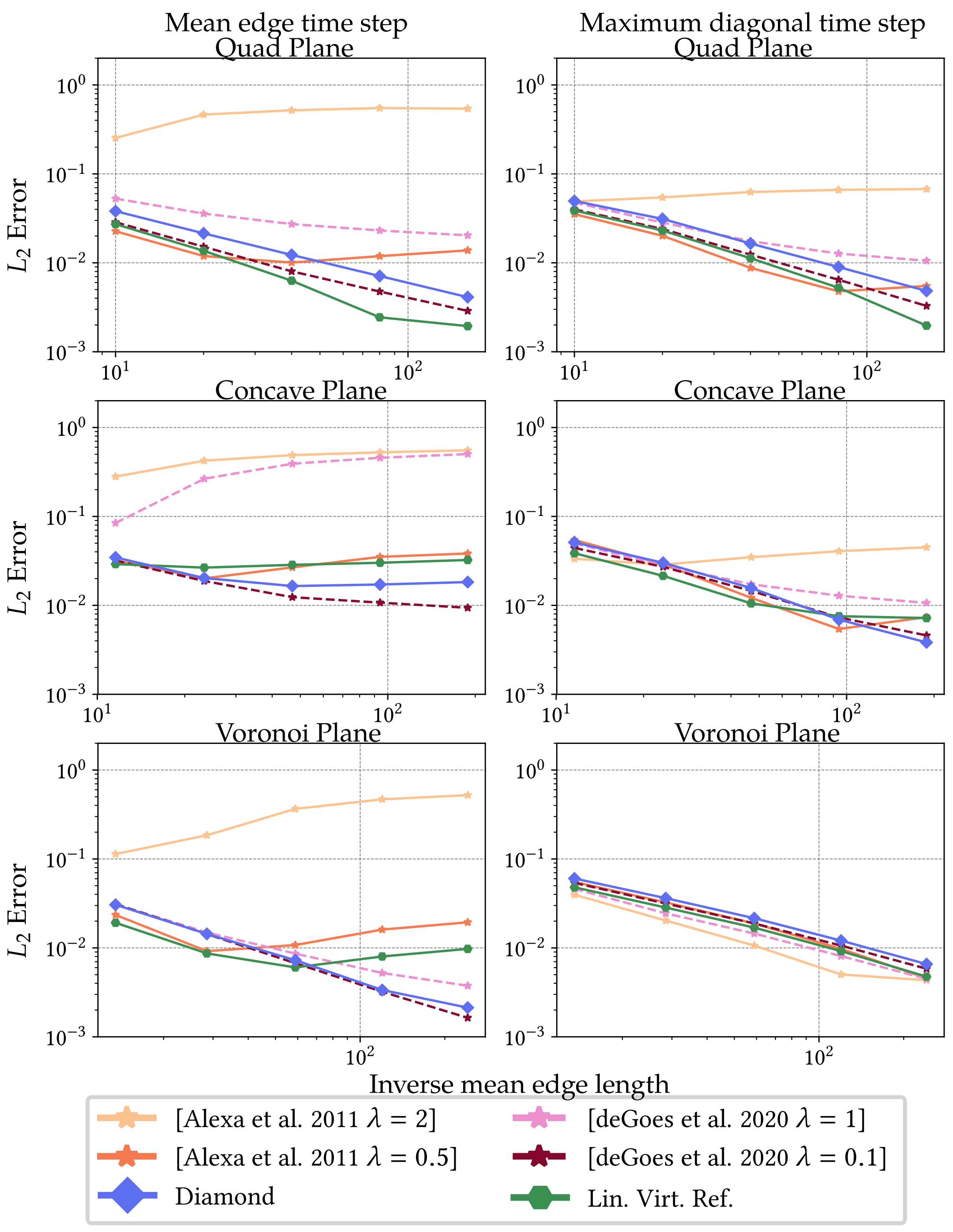


Figure 4.6: *L_2 error of geodesic distances for different tessellations of the unit grid.*

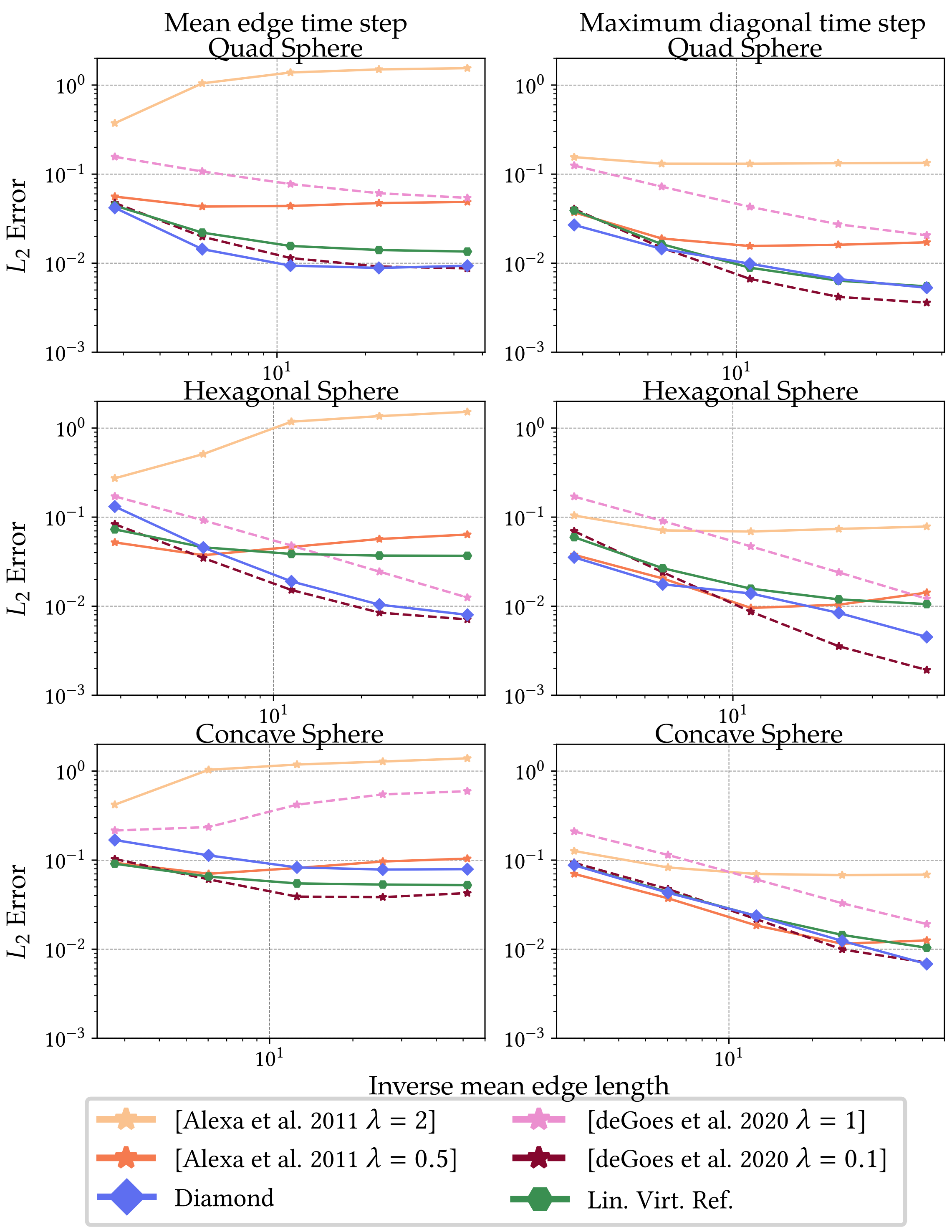


Figure 4.7: *L_2 error of geodesic distances for different tessellations of the unit sphere.*

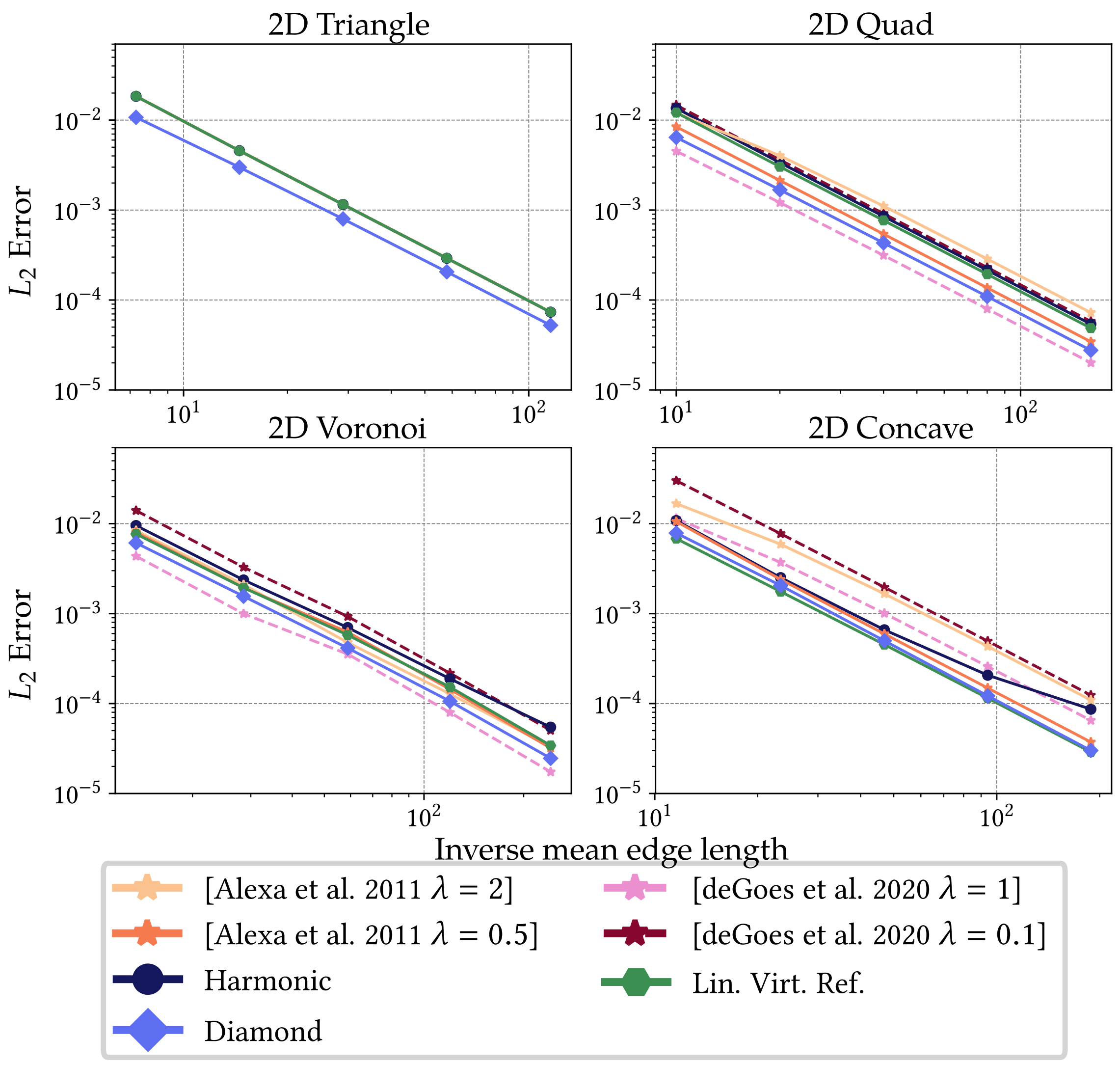


Figure 4.8: *L_2 error in log-log scale of the Poisson system solved for Franke's test function on different tesselations of the unit square. On triangle meshes, the operators [AW11; BHK+20; GBD20; MKB+08] reduce to the cotangent Laplacian and hence yield the same results.*

Poisson Equations We analyze the convergence rate of the Diamond Laplacian by solving the Poisson equation on different tessellations of the unit square and sphere. As in previous chapters, we employ the Laplacian of Franke's test function [Fra79] and the values of the scaled spherical harmonic function $Y_3^2(x, y, z)$ with eigenvalue $\lambda = -12$. As can be seen in Figure 4.8 and Figure 4.9, the Diamond Laplacian has the expected convergence rate and consistently maintains lower errors than most of the other discretizations for the majority of test cases. On spherical meshes, it consistently leads to the most accurate results. On planar meshes, de Goes et al. [GBD20] yields lower error rates on two tessellations for $\lambda = 1$. However, this hyper-parameter configuration would lead to one of the least accurate results on the spherical tessellations. The linear virtual refinement method slightly surpasses our new operator on

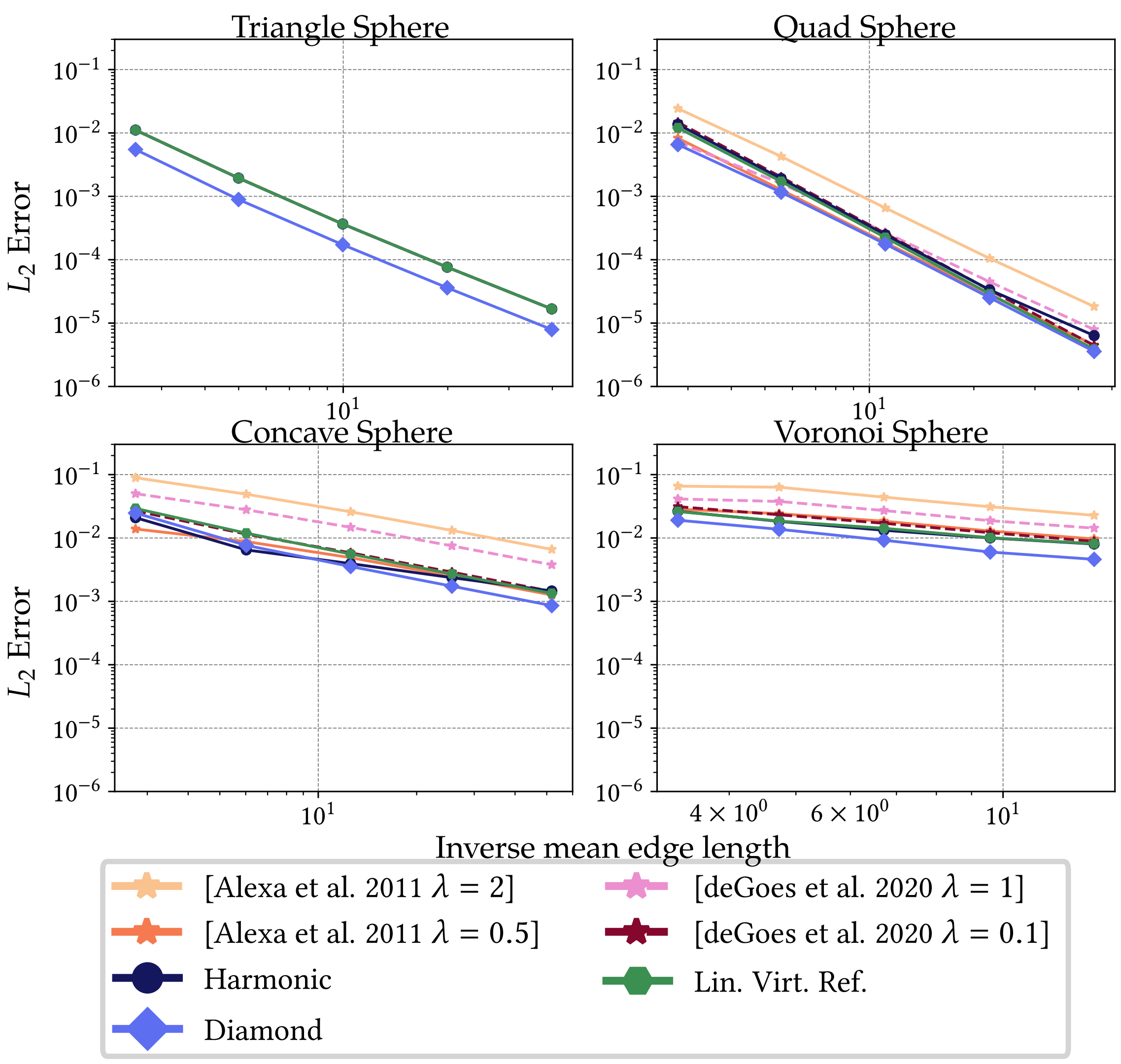


Figure 4.9: *L_2 error in log-log scale for the Poisson solve for the scaled real-valued spherical harmonic function Y_2^3 with eigenvalue -12 on different tessellations of the unit sphere.*

the concave planes. However, the Diamond Laplacian is empirically still one of the most consistent operators with respect to its accuracy. On triangle meshes, it performs favorably for both problems and is more accurate than the cotangent operator, to which the other polygon Laplacians are reduced.

4.6.2 Volume Meshes

Poisson Equations We solve the Poisson system on different tessellations of the 3D unit cube, as depicted in Figure 3.13, by fixing boundary vertices to the values of F_{3D} (see Equation (2.49)). As shown in Figure 4.10, the Diamond Laplacian has the expected convergence behavior and yields lower error rates than the linear virtual refinement method or the harmonic coordinates.

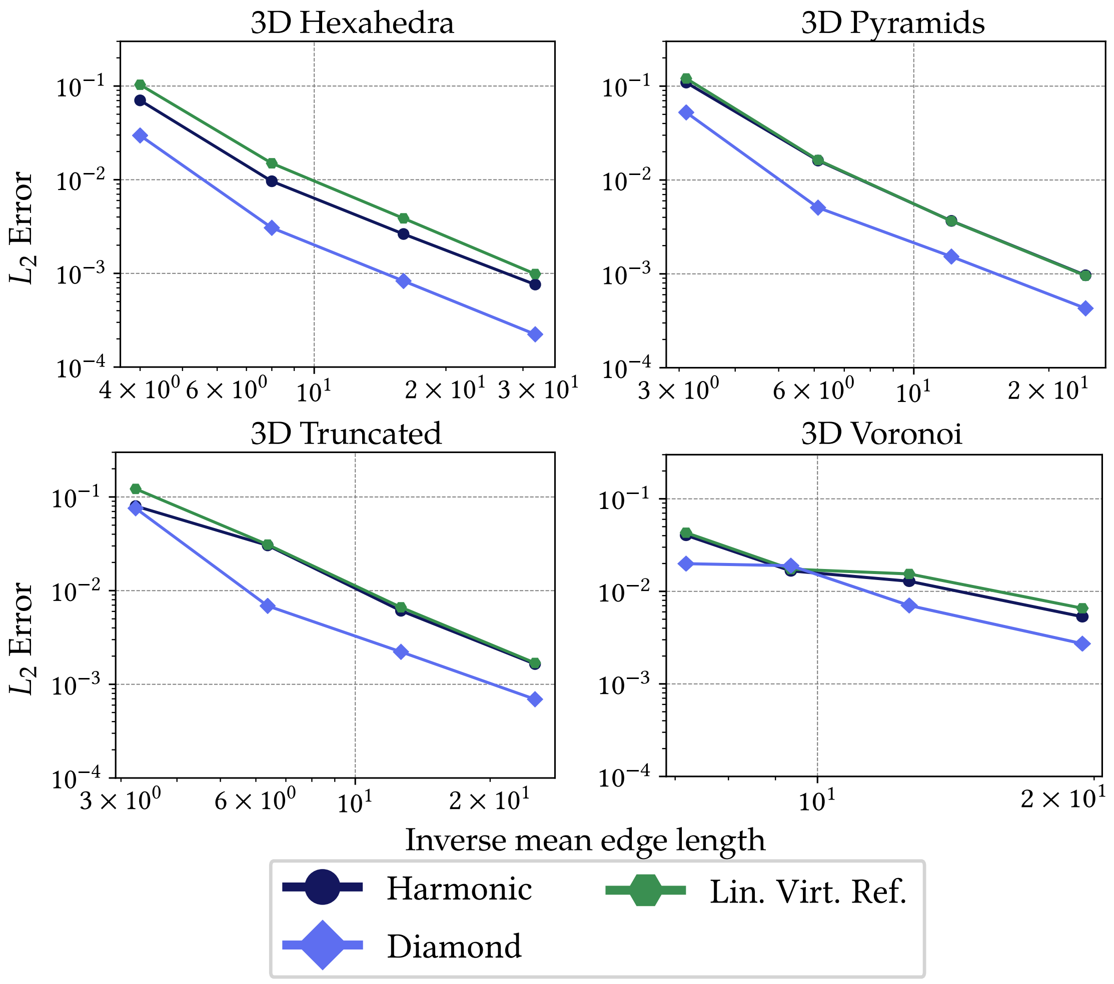


Figure 4.10: *L_2 error in log-log scale of the Poisson system solved for Franke's test function on unit cubes tessellated with hexahedra (left), pyramids (center left), truncated cells (center right), and Voronoi cells (right).*

Eigenmodes The plots in Figure 4.11 show that the Diamond Laplacian successfully recovers the desired eigenvalues, independent of the tessellation. It also significantly surpasses both the linear virtual refinement method and the harmonic coordinates in accuracy. For tetrahedral meshes, we can compare the Diamond Laplacian to the volume cotangent Laplacian (see Section 2.2.2), in this setting referred to as *Primal Laplacian*, and to the *Dual Laplacian*, which was introduced in [AHK+20]. The results are displayed in Figure 4.12. As for the polyhedral tessellations, the Diamond Laplacian is considerably more accurate than both the primal and dual tetrahedral Laplacians.

Sparsity and Timings The operator matrix for polygon Laplacians has more non-zero elements than the corresponding adjacency matrix. These entries reflect that at least the vertices belonging to the same face are connected since they all influence the (integrated) differential properties of the face. The choice of

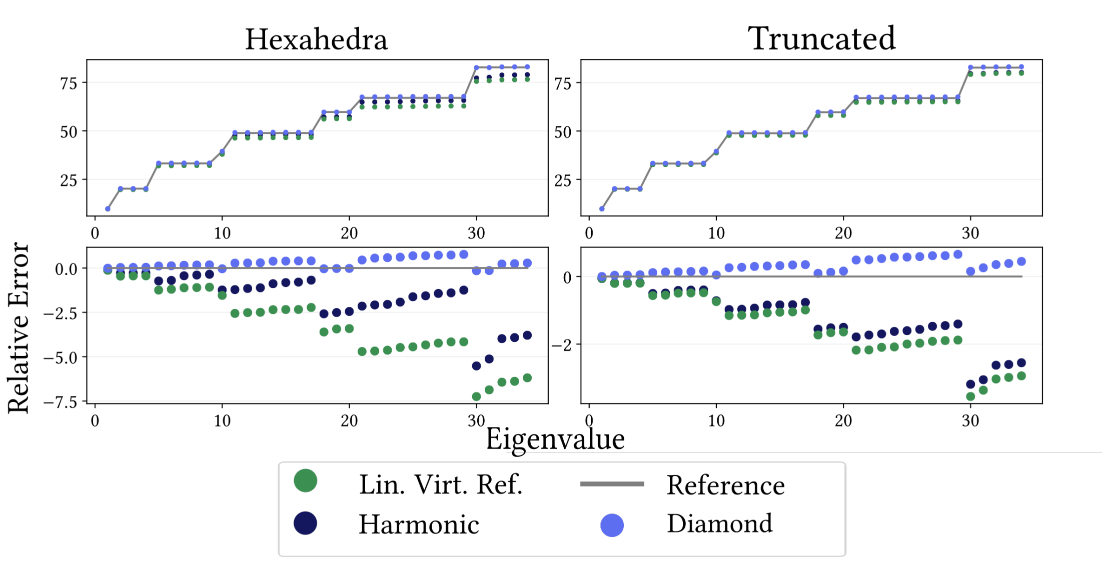
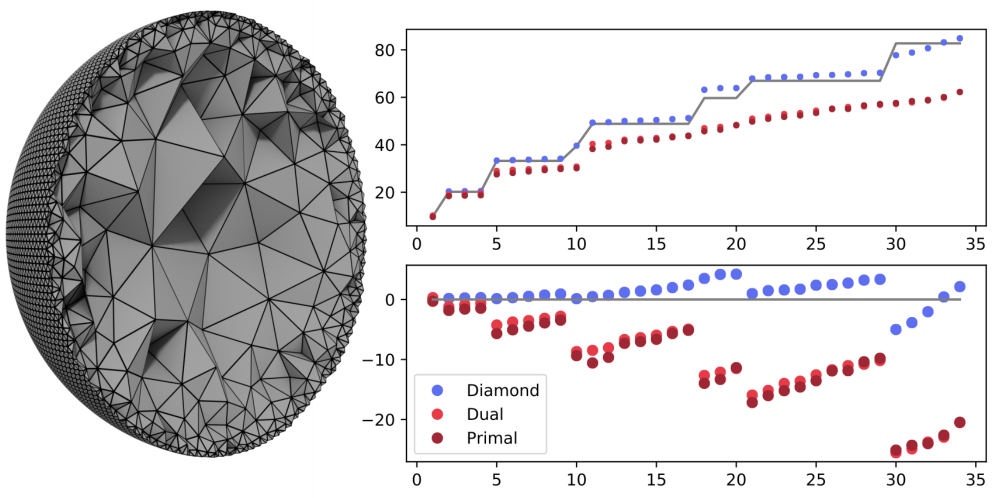


Figure 4.11: *The smallest 34 non-zero eigenvalues of the Laplacian on two unit balls consisting of hexhedra (left) and truncated cells (right). The top row shows the obtained eigenvalues, the bottom row the deviation from the true values. In all cases, the Diamond Laplacian is more accurate than the linear virtual refinement method*[BHK+20] *or the harmonic coordinates* [MKB+08].

Table 4.1: *Different statistics involved in the solution process of a Poisson problem expanded with the results of the Diamond Laplacian. Since the operators of Alexa and Wardetzky* [AW11] *and de Goes et al.* [GBD20] *have similar construction times, work only in 2D, and have the same sparsity pattern, we use the results obtained with Alexa and Wardetzky's operator as a representative for the DEC discretizations.*

Mesh	$\lvert\mathcal{V}\rvert$	Harmonic			DEC			Lin. Virt. Ref.			Diamond		
		build	nnz	solve	build	nnz	solve	build	nnz	solve	build	nnz	solve
Quads 2D	26k	92s	231k	8ms	44ms	231k	8ms	10ms	231k	8ms	43ms	537k	21ms
Voronoi 2D	51k	78s	616k	25ms	94ms	616k	25ms	29ms	616k	25ms	224ms	1723k	74ms
Hexahedra 3D	4913	465s	117k	3ms	—	—	—	190ms	117k	3ms	250ms	333k	6ms
Voronoi 3D	5183	482s	324k	5ms	—	—	—	140ms	324k	5ms	280ms	1497k	11ms

diamonds as regions for estimating the differentials allows a more accurate estimation of the gradient across (primal) edges. However, using these cells comes at the expense of introducing additional non-zero entries in the operator matrix that reflect this connection. The same holds for polyhedral meshes, where the additional cell and face vertices we (virtually) insert to construct the minimal diamonds lead to a higher density in the Diamond stiffness matrix than that of the matrices constructed with the linear virtual refinement method and the harmonic coordinates. The reduced sparsity results in higher computational complexity for solving the involved linear systems, as can be seen in Table 4.1, which expands the results from Table 3.5 by the respective timings obtained with the Diamond Laplacian.



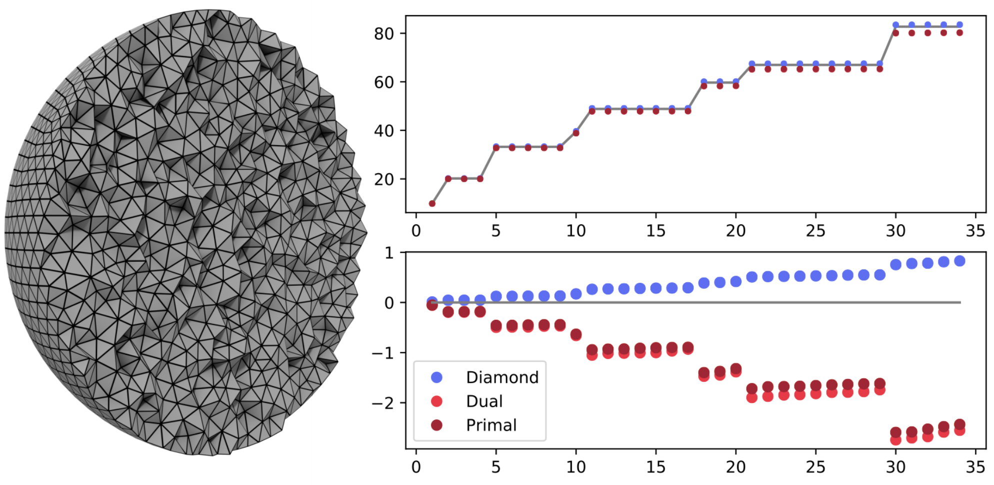


Figure 4.12: *The 34 smallest eigenvalues of the Laplacian, computed on uniform and adaptive tetrahedral tessellations of the unit ball. For tetrahedral meshes the harmonic coordinats [MKB+08] and the linear virtual refinement method both reduce to the Primal cotangent operator, therefore yielding the same results. The Diamond Laplacian is not affected by adaptive tessellations and achieves a higher accuracy.*

4.7 LIMITATIONS

The virtual refinement for the Diamond Laplacian is limited by the same aspects as discussed in Section 3.4, meaning that, depending on the shape of the respective primal cell and the placement of the virtual dual vertices, parts of the diamond cells might be flipped.
Furthermore, by construction, the operator does not allow for local optimizations per polygon. Finite Element approaches enable us to optimize its shape functions, for example by altering the position of the virtual vertex, individually for every single cell. For the Diamond Laplacian, similar optimizations require global operations since the diamond cells depend on the values of two adjacent polygons.

4.8 CONCLUSION

In this chapter, we improved on DDFV methods by generalizing the 2D DDFV formulation to surface meshes immersed in 3D (*intrinsic gradients*) and by providing a formulation for polyhedral meshes (*ring of minimal diamonds*) that avoids the spurious modes of CeVe [Her09] and is considerably simpler than CeVeFE [CH11]. We combine this with the linear virtual refinement method's one- and two-step prolongation matrices for polygonal and polyhedral meshes. The resulting prolongation allows us to remove the additional degrees of freedom of dual cell and face vertices from the improved DDFV operators. The Diamond Laplacian provides a simple and accurate Laplacian for general polygonal and polyhedral meshes that maps values at vertices to values at vertices while having the appropriate kernel, linear precision, and the desired semi-definiteness.
In extensive numerical evaluations of prototypical geometry processing applications, we compare the Diamond Laplacian to several existing methods from the graphics community. The Diamond Laplacian is superior to all its competitors in almost all experiments, especially on volume methods. Otherwise, it is one of the most accurate contenders. In contrast to existing polygon Laplacians, it is not reduced to the cotangent formulation on triangle or tetrahedral meshes. It is, therefore, a viable alternative for pure simplicial meshes since it provides more accurate results, particularly if the gradient operator is involved.
The price for generality and accuracy is a higher number of non-zero elements, leading to a slight increase in computation time. We believe that this is a useful trade-off in graphics and geometric modeling, where meshes are mainly processed without altering them. We cannot resist making the obvious remark: Diamonds are attractive but somewhat expensive.

5 VARIATIONAL QUADRATIC SHAPE FUNCTIONS FOR POLYGONS AND POLYHEDRA

In the previous chapters, we considered polygonal and polyhedral Laplacians that provide a similar level of numerical accuracy for a given mesh resolution and inherit the desired quadratic convergence rate under element refinement. For the FEM methods specifically (linear virtual refinement method (Chapter 3) and harmonic coordinates [WBG07; MKB+08]), this implies a common characteristic: They define piecewise shape functions that are associated with the vertex nodes of the given polygon. Using this kind of bases is common practice in geometry processing due to their balanced trade-off between accuracy and efficiency. Nevertheless, 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., [MTP+08; SDG+19; SHG+22]), but they are currently restricted to simplicial or hexahedral meshes. In this chapter, we generalize quadratic shape functions to arbitrary polygonal/polyhedral meshes and demonstrate their superior numerical behavior in various experiments.

Inspired by the linear virtual refinement method and the Diamond Laplacian, 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 is hidden from the user through the 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 Chapter 3), 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 second-order Lagrange shape functions, also known as $P2$ 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.

In the following, we review related work (Section 5.1) before proposing our variational quadratic shape functions and multigrid scheme for polygonal/polyhedral meshes (Section 5.3). In Section 5.4, we evaluate our method's numerical performance for various experiments on surface and volume meshes, demonstrating that our method compares favorably to existing approaches.

Individual Contribution *I developed the generalization of the linear virtual refinement method to higher-order shape functions in collaboration with Philipp Herholz, Olga Sorkine-Hornung, Mario Botsch, and Misha Kazhdan. The generalization of quadratic P2 elements to arbitrary polygons and polyhedra was a collective effort. The proofs of the shape functions various properties were written by Misha Kazhdan and extended upon by me. The numerical evaluation and comparison to other polygonal operators and higher order shape function in different computer graphics applications were implemented by Misha Kazhdan and me. The custom multrigrid solver that sigificantly improves computational times were designed and implemented by Misha Kazhdan, Mario Botsch and Philipp Herholz.*

Corresponding Publication *This chapter is based on the following publications:*

Bunge, A., Herholz, P., Sorkine-Hornung, O., Botsch, M., and Kazhdan, M. (2022). "Variational Quadratic Shape Functions for Polygons and Polyhedra." ACM Transactions on Graphics, 41(4) 54:1–54:14.

5.1 RELATED WORK

Although the discretization of a respective Laplace, divergence, and gradient operator is a valuable tool in geometry processing, an explicit construction of finite element shape functions further extends the possible set of applications we are able to address. Therefore, while briefly touching upon this subject in Section 2.2.3 and highlighting the construction of polygon linear shape functions in Section 3.2.2, we will further expand on this topic and other existing basis functions in the upcoming section.

5.1.1 Shape Functions for Polygons and Polyhedra

A displacement-based finite element method for polygons and polyhedra was presented by Rashid et al. [RS06]. 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 [HS17]. 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 [Wac75], mean value coordinates [Flo03; JSW05], harmonic coordinates [JMD+07], or maximum entropy coordinates [Suk04; HS08].

These coordinates have been mainly used for cage-based deformation, which is also the motivation for higher-order constructions [LS08]. Recently Longva et al. [LLK+20] 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 [VBC+13] 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. [MRS14]. The work of Gilette et al. [GRB16] uses generalized barycentric coordinates to define conforming scalar-valued and vector-valued basis functions of differential form order $k = 0, \dots, 2$ for polygons and $k = 0, \dots, 3$ for polyhedra, inspired by Whitney differential forms. A novel method to define barycentric coordinates in the context of cage-based deformation was presented by Dodik et al. [DSS+23], who are able to learn different sets of shape funtions with the help of neural fields. Wicke et al. [WBG07] 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. [MKB+08] through the use of harmonic coordinates. Schneider et al. [SDG+19] 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. [Bis14] also worked with harmonic coordinates to define shape functions on star-shaped polygons and polyhedra. Similar to Rashid and Selimotic [RS06], 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, the shape functions for polygons/polyhedra we are going to describe in the next chapter of this thesis are considerably simpler, since they are piecewise quadratic, and therefore can more easily and efficiently be constructed, differentiated, and integrated.

5.1.2 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 [SHG+22] 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. [MTP+08] 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. [SHD+18] 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 [VDR17], 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. [ARR+19] 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 do not require a canonical base domain, which is hard to define, especially for arbitrary polyhedra.

5.2 BACKGROUND

In this section we will briefly extend the notion of Lagrange basis functions presented in Section 2.2 to higher-order elements. We will also revisit the linear virtual refinement method introduced in Chapter 3 and delve deeper into the linear shape functions derived from this method, with the intention to lay the theoretical groundwork for extending them to quadratic basis functions on polygonal and polyhedral domains.

5.2.1 Lagrange Basis Functions

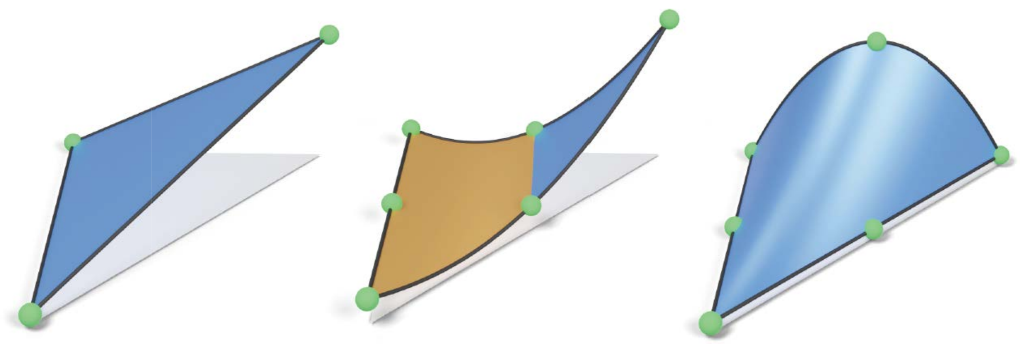


Figure 5.1: *Lagrange basis functions restricted to a single triangle. Linear basis (left) and two quadratic basis functions, one associated with a vertex (center) and another one with an edge midpoint (right). The functions are plotted as height fields over their domain (gray triangle). Gold and blue regions indicate negative and positive values respectively.*

Consider a simplicial mesh $\mathcal{M} = (\mathcal{V}, \mathcal{S})$ with vertices $\mathcal{V} = \{v_1, \ldots, v_m\}$ and simplex set $\mathcal{S}$. The linear Lagrange basis function at the vertex position $\mathbf{x}_i$ is defined as the unique C^0 continuous and piecewise linear function ψ_i with the Lagrange interpolation property

$$\psi_i(\mathbf{x}_j) = \delta_{ij} := \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

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 5.1 (left). Degrees of freedom u_i are associated to vertices v_i, spanning the space of continuous and piecewise linear functions

$$f(\mathbf{x}) = \sum_i u_i \psi_i(\mathbf{x}). \tag{5.2}$$

Quadratic Lagrange basis functions (Figure 5.1 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 $f(\mathbf{x}) \equiv 1$.

5.2.2 The Shape Functions of the Linear Virtual Refinement Method

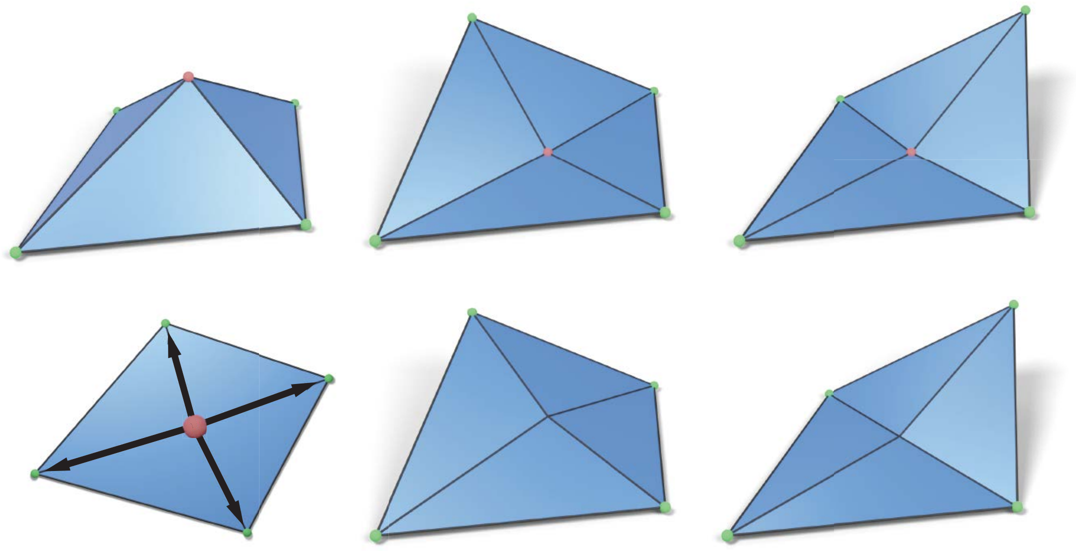


Figure 5.2: *Linear shape functions ψ_j on a refined quad element (top row) are combined to form a basis φ_i on the original polygon (bottom row).*

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 (5.1). 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.

As touched upon in Chapter 3, the core concept of the linear virtual refinement method 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 5.2), 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 original mesh's $|\mathcal{V}|$ vertices. Therefore, the shape functions associated with the virtual vertices are distributed to the n_f vertices of the original polygon using weights $w_1, \ldots, w_{n_f}$ with $\sum_i w_i = 1$. In other words, if the ψ_i are the $(n_f + 1)$ linear Lagrange basis functions on the refined polygon (Figure 5.2, top row), the n_f functions at the original vertices are

$$\varphi_i = \psi_i + w_i \psi_f \quad \text{for } i = 1, \ldots, n_f \tag{5.3}$$

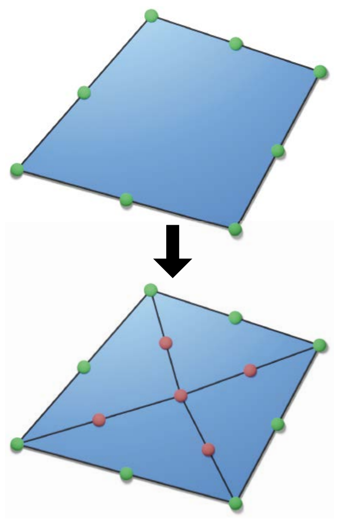
(see Figure 5.2, bottom row), with ψ_f being the basis function associated to the virtual vertex. This construction guarantees that the piecewise linear polygon shape functions φ_i obey the Lagrange interpolation and partition of unity property.

As discussed in Section 3.2.3, we define that the position of the virtual vertex $\mathbf{x}_f \in \mathbb{R}^3$ for a polygon $\left(\mathbf{x}_1, \ldots, \mathbf{x}_{n_f}\right)$, should be the minimizer of the sum of squared triangle areas. The virtual vertex is then expressed as an affine combination of polygon vertices $\mathbf{x}_f = \sum_i w_i \mathbf{x}_i$, since choosing a set of weights with this property guarantees linear precision for the shape functions. For the linear shape functions, we use the L_2 regularization described in Appendix A.2 and solve a linearly constrained quadratic optimization problem to determine the weights for each polygon. Using these weights we express shape functions φ_i as linear combination of the shape functions ψ_i. Unfortunately, these ideas do not directly extend to higher-order basis functions, as we detail in the next section.

5.3 METHOD

5.3.1 Quadratic Basis Functions for Polygons

Suppose we are given a polygon $\left(\mathbf{x}_1, \ldots, \mathbf{x}_{n_f}\right) \subset \mathbb{R}^3$. As in Chapter 3, we first introduce the minimizer of the sum of squared triangle areas as virtual vertex $\mathbf{x}_f \in \mathbb{R}^3$, splitting the polygon into n_f 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 linear refinement method 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 $2n_f$ degrees of freedom on the polygon boundary (green), which we call *coarse* nodes $\mathcal{C}_N$, and $n_f + 1$ virtual degrees of freedom $\mathcal{K}_N$ (red). We call the union of both sets the set of *fine* nodes $\mathcal{F}_N = \mathcal{C}_N \bigcup \mathcal{K}_N$. The location of the fine nodes are denoted by $\mathbf{x}_i$. 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



$$\varphi_i = \psi_i + \sum_{j \in \mathcal{K}_N} w_{ij} \psi_j \quad \text{for } i \in \mathcal{C}_N. \tag{5.4}$$

Interpolation The shape functions constructed according to Equation (5.4) obey the Lagrange interpolation property $\varphi_i(\mathbf{x}_j) = \delta_{ij}$ at all coarse nodes $j \in \mathcal{C}_N$ by construction, because the fine basis obeys $\psi_i(\mathbf{x}_j) = 0$ for all $i \in \mathcal{K}_N$ and $j \in \mathcal{C}_N$. 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 3.2.2. The only difference is that we have more than one virtual vertex. For a face f the local prolongation matrix is

$$\mathbf{P}^f_{ij} = \begin{cases} \delta_{ij} & i \in \mathcal{C}_N, \\ w_{ji} & i \in \mathcal{K}_N, \end{cases} \tag{5.5}$$

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

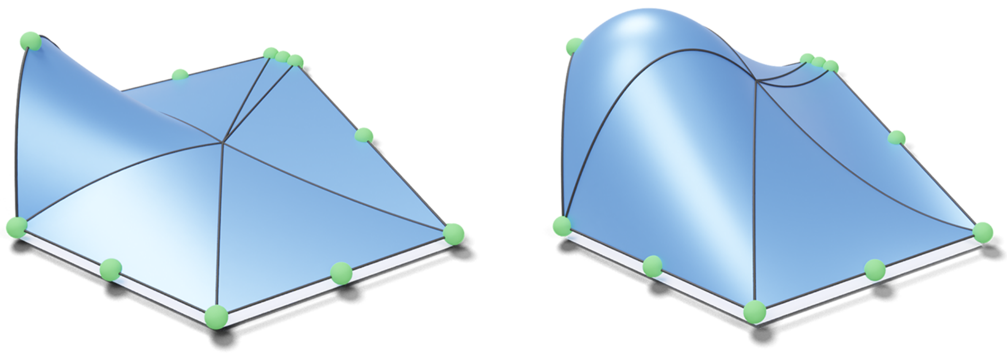


Figure 5.3: *Prolongation weights computed using a regularizer on their norm lead to very local basis functions at the cost of smoothness (left). Our proposed energy explicitly leads to basis functions that prioritize smoothness across internal edges (right).*

Naive Extension The challenge is choosing weights so that the resulting coarse basis is 'nice' – resulting in favorable numerical behavior. A direct generalization of the linear virtual refinement method could look as follows: For a polygon with n_f vertices we have $2n_f$ coarse and $n_f + 1$ virtual degrees of freedom. Find the $|\mathcal{C}_N| \cdot |\mathcal{K}_N| = 2n_f \cdot (n_f + 1)$ weights w_{ij} such that:

1. The partition of unity property holds:

$$\sum_{i \in \mathcal{C}_N} w_{ij} = 1 \quad \text{for } j \in \mathcal{K}_N. \tag{5.6}$$

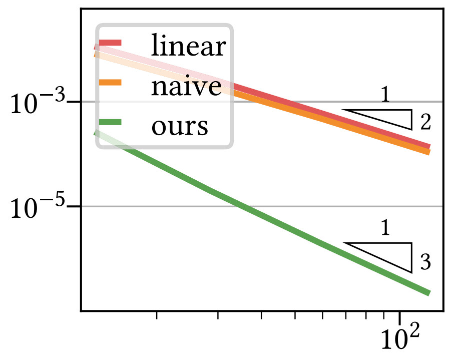
2. The virtual node positions can be expressed as affine combination of coarse nodes using the weights:

$$\mathbf{x}_j = \sum_{i \in \mathcal{C}_N} w_{ij}\mathbf{x}_i \quad \text{for } j \in \mathcal{K}_N. \tag{5.7}$$

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

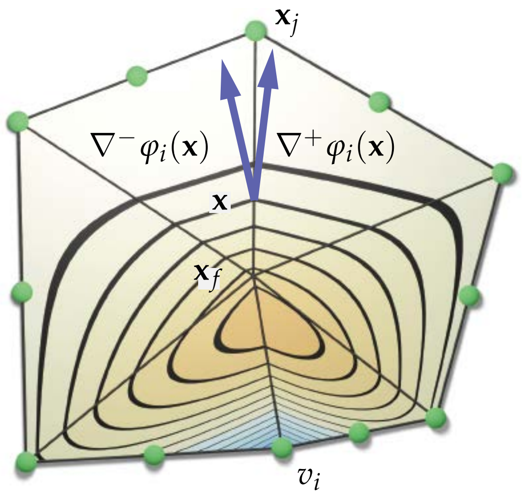
$$\{w_{ij}\} = \arg\min_{\{w_{ij}\}} \sum_{i,j} w_{ij}^2. \tag{5.8}$$

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 5.3 (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, which led to the final quadratic virtual refinement method.



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 (5.8) of the naive approach with the squared gradient difference integrated along all virtual edges, summed over all coarse basis functions. Figure 5.3 (right) demonstrates that optimizing this objective subject to constraints (5.6) and (5.7) produces shape functions that are significantly smoother compared to the naive approach. Specifically, we solve the following quadratic optimization problem

$$W = \arg\min_{W} \sum_{i\in\mathcal{C}_N} \sum_{\sigma\in\mathcal{E}^*} \int_\sigma \|\nabla_\sigma^+ \varphi_i - \nabla_\sigma^- \varphi_i\|^2 \,\mathrm{d}\sigma \qquad \text{s.t. constraints (5.6) and (5.7) are satisified} \tag{5.9}$$


$\mathbf{x}_j$
$\nabla^-\varphi_i(\mathbf{x})$
$\nabla^+\varphi_i(\mathbf{x})$
$\mathbf{x}$
$\mathbf{x}_f$
v_i

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 ($\mathcal{E}^* = \{(\mathbf{x}_f, \mathbf{x}_i)\}_{1\le i\le n_f}$). 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. 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 5.4 shows a set of quadratic Lagrange shape functions on the refined triangle mesh (top row). Minimizing the cross-edge gradient difference (Equation (5.9)) 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 5.3, right).

Partition of Unity As in the linear case, satisfying Equation (5.6) ensures a partition of unity for the quadratic basis. Specifically, since the Lagrange basis ψ_i sums to one, we have

$$\sum_{i\in\mathcal{C}_N} \varphi_i = \sum_{i\in\mathcal{C}_N}\left(\psi_i + \sum_{j\in\mathcal{K}_N} w_{ij}\psi_j\right) = \sum_{i\in\mathcal{C}_N}\psi_i + \sum_{\substack{i\in\mathcal{C}_N\\ j\in\mathcal{K}_N}} w_{ij}\psi_j = \sum_{i\in\mathcal{F}_N}\psi_i \equiv 1. \tag{5.10}$$

Linear Precision Our shape functions are linearly precise, which is a direct consequence of the reproduction property enforced as constraint (5.7). 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 $u\colon \mathbb{R}^3 \to \mathbb{R}$, let $u_i = u(\mathbf{x}_i)$ be the evaluation of u at node $\mathbf{x}_i$, and consider the sum $\sum_{i\in\mathcal{C}_N} u_i\varphi_i$. By the reproduction property we have

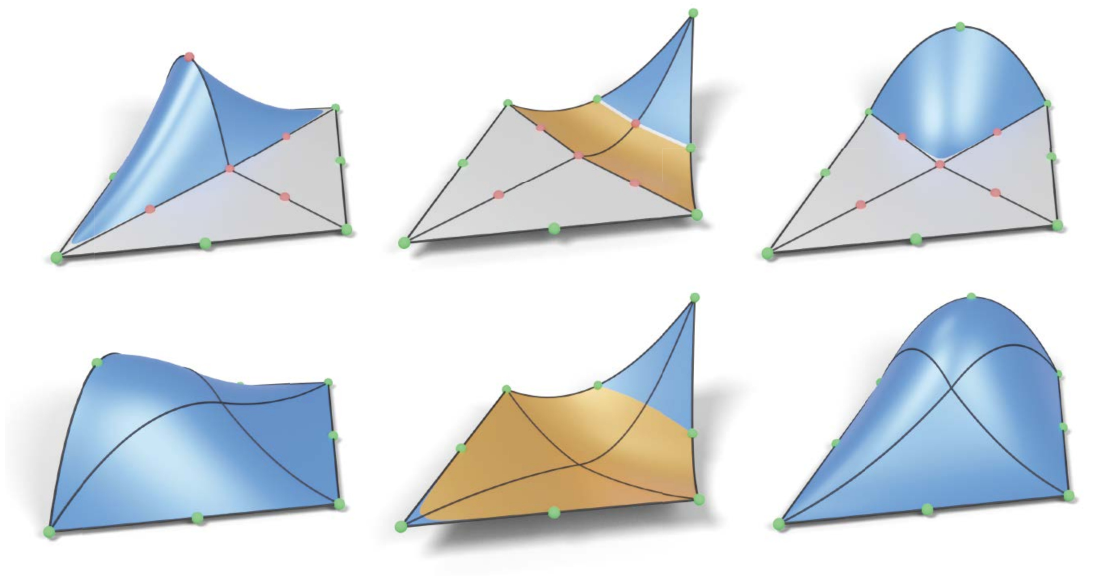


Figure 5.4: *Quadratic shape functions on a refined triangle mesh (top row) are combined to form shape functions on the original polygon. Regions colored in orange have negative function values. The functions are C^∞ everywhere except at the edges where they only have C^0 continuity (like the standard quadratic Lagrange basis).*

$$\sum_{i\in\mathcal{C}_N} u_i\varphi_i = \sum_{i\in\mathcal{C}_N} u_i\left(\psi_i + \sum_{j\in\mathcal{K}_N} w_{ij}\psi_j\right) \tag{5.11}$$

$$= \sum_{i\in\mathcal{C}_N} u_i\psi_i + \sum_{j\in\mathcal{K}_N} \underbrace{\left(\sum_{i\in\mathcal{C}_N} u_i w_{ij}\right)}_{u_j} \psi_j = \sum_{j\in\mathcal{F}_N} u_j\psi_j = u, \tag{5.12}$$

because the reproduction property (Equation (5.7)) extends to arbitrary linear functions. On the one hand, the sum is in the span of the φ_i. On the other hand, it can be expressed as the sum of the Lagrange basis functions ψ_j, weighted by the values of u at nodes $\mathbf{x}_j$. 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. The proof is provided in the Appendix section B.1.

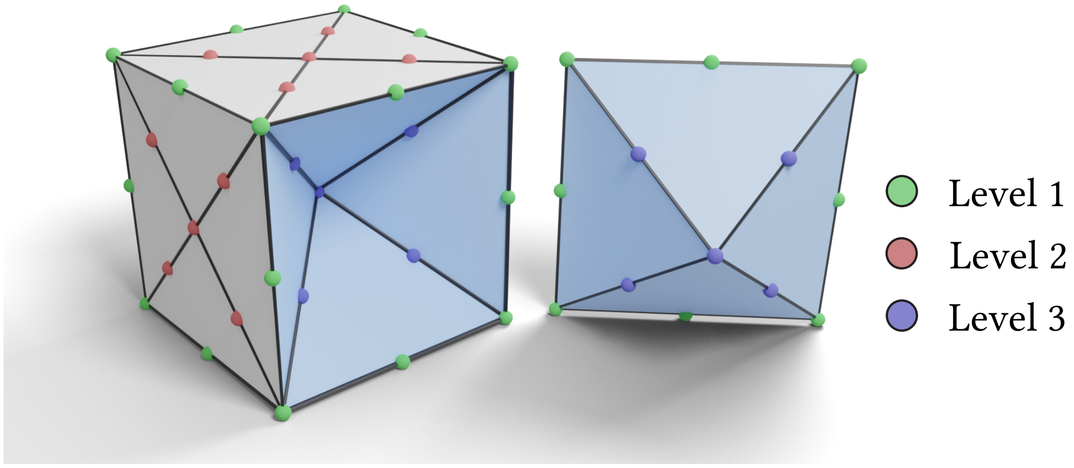


Figure 5.5: *Node positions for a refined cube. Parts of the mesh are detached for visualization purposes. All 2-faces are split into triangles using a virtual vertices. The volume is decomposed into tetrahedra by introducing a central virtual vertex which is connected to all face triangles. The nodes are colored according to their type.*

5.3.2 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 the volumetric linear refinement method (see Section 3.2.7). 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 5.5 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 5.5. 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 (Equation (5.9)). 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 5.5 in light blue. We again integrate the squared gradient difference of the shape functions, this time over triangles of the virtual

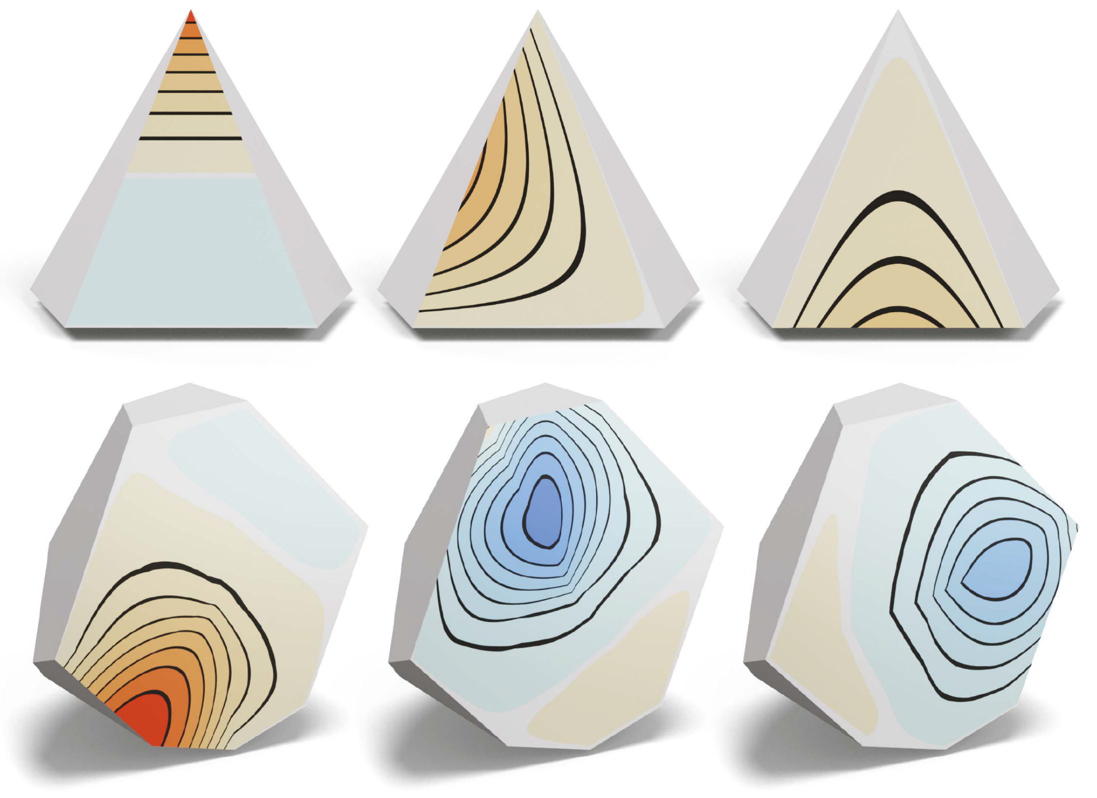


Figure 5.6: *Cross-sections showing our quadratic shape functions, for different polyhedra: While the shape functions are optimized for smoothness, they may exhibit small gradient discontinuities across virtual faces, visible as sharp corners along iso-curves. For tetrahedra (top), where we reproduce quadratic Lagrange shape functions, the functions are* C^1 *inside the simplex.*

tetrahedra that do not tessellate the boundary of the polyhedron:

$$W = \arg\min_{W} \sum_{i \in \mathcal{C}_N} \sum_{\sigma \in \mathcal{T}^*} \int_{\sigma} \|\nabla_{\sigma}^{+} \varphi_i - \nabla_{\sigma}^{-} \varphi_i\|^2 \,\mathrm{d}\sigma \qquad \text{s.t. constraints (5.6) and (5.7) are satisified,} \tag{5.13}$$

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 5.6 and Figure 5.7 show 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.

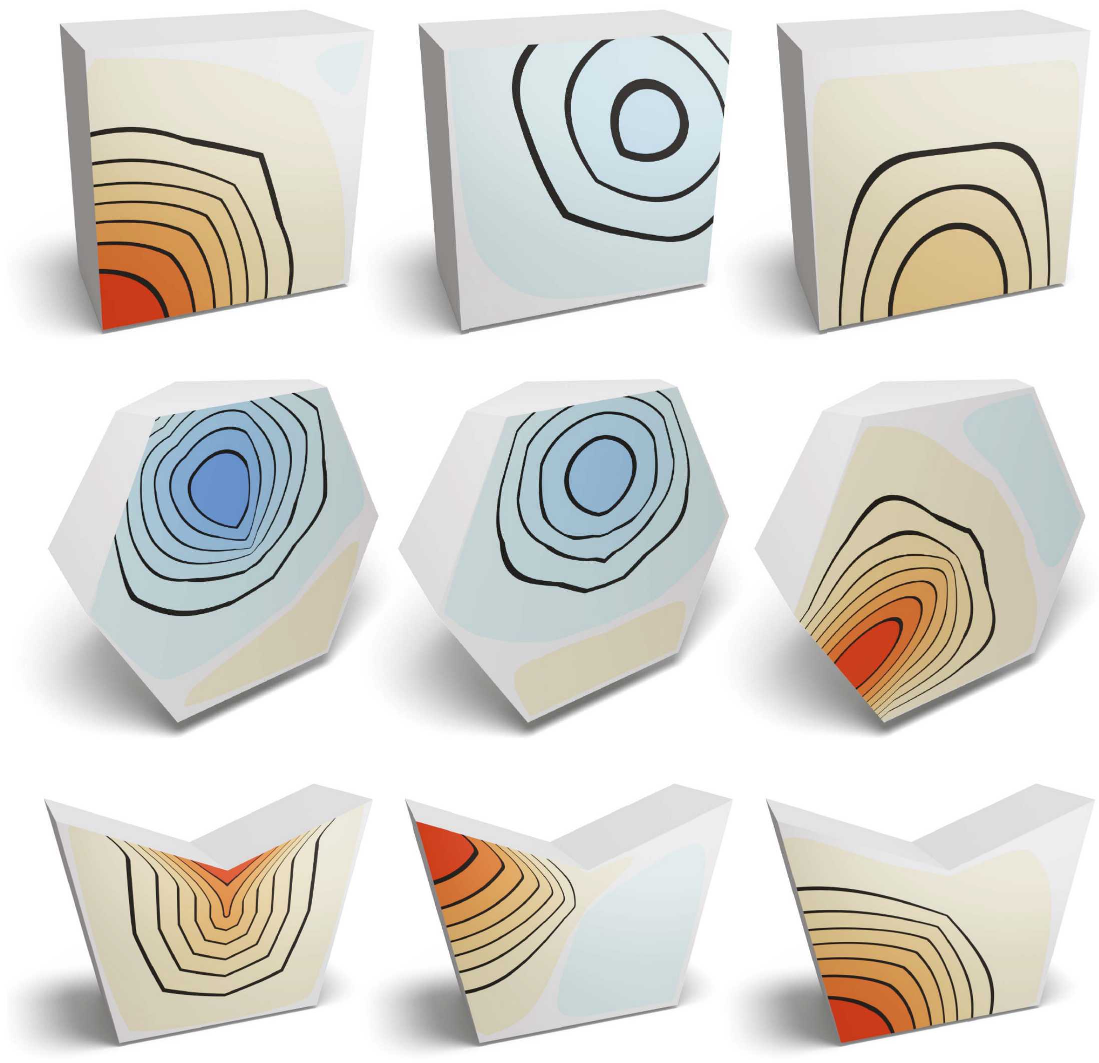


Figure 5.7: *Additional cross-sections showing our quadratic shape functions for different polyhedra.*

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 function associated with a virtual node is supported within a single polygon and not shared through a common edge.

Figure 5.8 (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

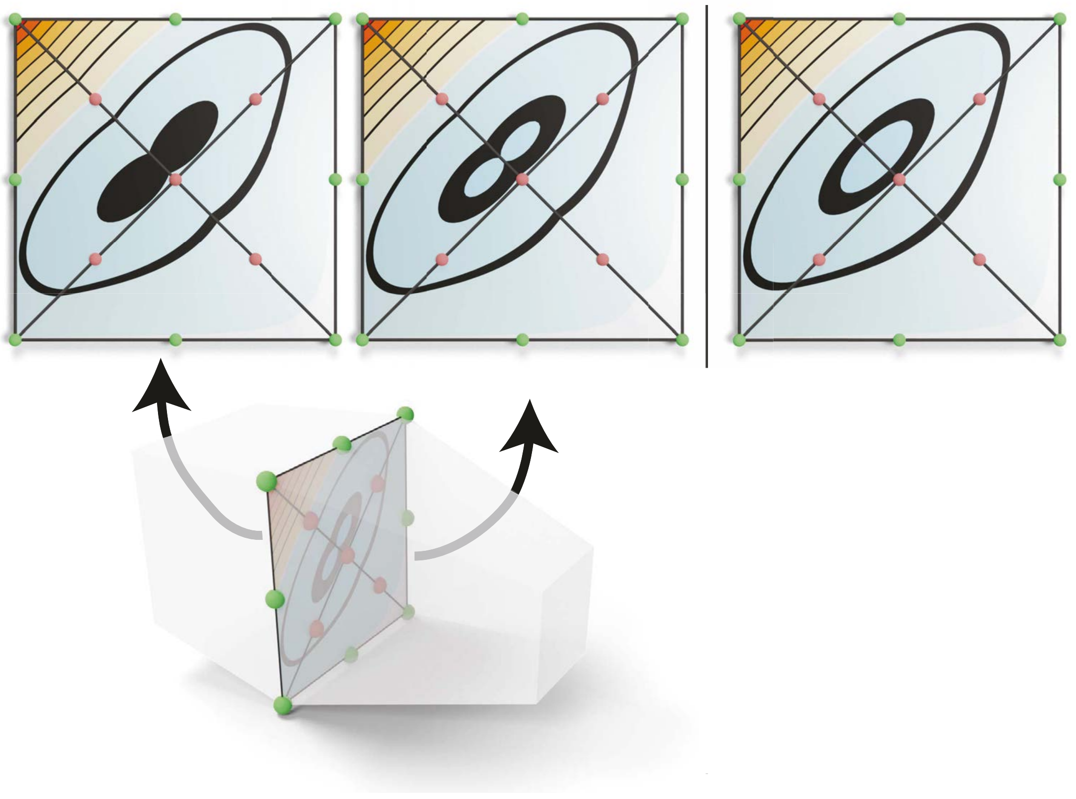


Figure 5.8: *Direct prolongation of all virtual degrees of freedom will result in shape functions that are not C^0 across shared faces (left). The values of the shape functions computed for each cell individually differ on the interface between both cells. To remedy this problem we compute the prolongation 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.*

single step, the shape functions defined by the two polyhedra sharing the quad do not agree on the quad (Figure 5.8 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 (5.9), distributing level 2 nodes to level 1 nodes.
2. Then, we solve for the prolongation matrix **P**, distributing level 2 and level 3 nodes to level 1 nodes, solving Equation (5.13). 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 5.8 (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 Appendix B) .

5.3.3 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 (5.9) and (5.13). For each cell, we begin by computing the $|\mathcal{F}_N| \times |\mathcal{F}_N|$ matrix giving the 'gradient discontinuity mass' of the finer Lagrange basis functions:

$$\mathbf{K}_{ij} = \sum_{\sigma \in \mathcal{S}^*} \int_\sigma \left\langle \nabla_\sigma^+ \psi_i - \nabla_\sigma^- \psi_i, \nabla_\sigma^+ \psi_j - \nabla_\sigma^- \psi_j \right\rangle \, d\sigma, \tag{5.14}$$

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

$$E(\mathbf{P}) = \mathrm{Tr}\left(\mathbf{P}^\top (\mathbf{K} + \varepsilon \mathbf{S}^\triangle)\mathbf{P}\right) \tag{5.15}$$

where $\mathbf{S}^\triangle$ is the stiffness matrix defined on the cell ($\mathbf{S}^\triangle_{ij} = \int \langle \nabla \psi_i, \nabla \psi_j \rangle$, 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. We show in Appendix B 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 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 (5.6) and linear reproduction (5.7) constraints using Lagrange multipliers and solve the resulting KKT system.

5.3.4 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 generating an accurate solution.

Our multigrid hierarchy consists of two levels: The coarse level contains nodes at vertices (Figure 5.9, 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 5.3. We perform multiple V-cycles as depicted in Figure 5.9 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:

$$\mathbf{R}_{ij} = \begin{cases} 1 & j \text{ is a vertex node and } i = j, \\ \frac{1}{2} & j \text{ is an edge node and } i \text{ is indicent to } j, \\ 0 & \text{otherwise.} \end{cases} \tag{5.16}$$

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

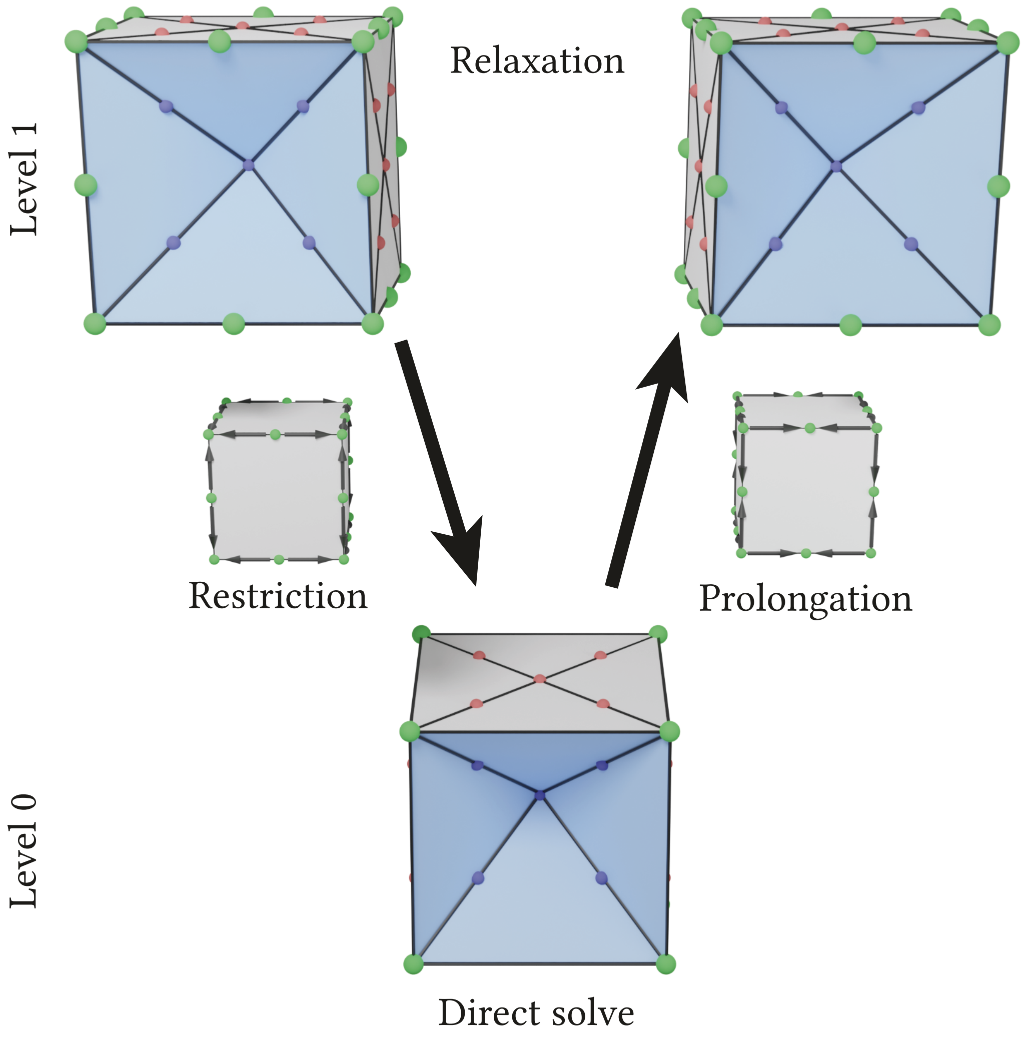


Figure 5.9: *A V-cycle of our multigrid implementation. The fine level relaxes the linear system using degrees of freedom at the vertices and edge-midpoints (top). The restriction/prolongation operators distribute information from/to edge-midpoints to/from vertices (middle). At the coarse level, we use a direct solve to solve the system discretized over the vertices (bottom).*

Direct Solve At the coarse level we employ the sparse supernodal Cholesky solver implemented in Cholmod [CDH+08] to solve

$$\mathbf{RAR}^\top \mathbf{x}^C = \mathbf{b}^C. \tag{5.17}$$

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.

5.4 EVALUATION

This evaluation will slightly deviate from the preceding result sections of this thesis. While we will keep expanding upon the comparisons of the different Laplacians within the introduced applications, we will shorten some of the tests due to the lack of direct comparability between linear and quadratic discretizations. As previously discussed, quadratic shape functions promise enhanced accuracy with the trade-off of longer solving times. Therefore, we will put more emphasis on comparing our approach to other higher-order basis functions designed for general volume and surface meshes. These include the standard bi/triquadratic quadrilateral/hexahedral $Q2$ shape functions for quadrilateral/hexahedral meshes and Schneider et al.'s [SDG+19] Poly-Spline basis functions, which we will refer to as PolyFEM in the legends of the plots. Similarly, our quadratic shape functions will from now on be labeled as the quadratic virtual refinement method, abbreviated as "Quad. Virt. Ref". Furthermore, we will provide qualitative examples to illustrate the visible advantages of using quadratic basis functions over the linear elements defined in the earlier chapters of this thesis.

5.4.1 Numerical Accuracy

Poisson Equation We measure the deviation of the solution u from the true function values, displayed in Figure 5.10 and Figure 5.11 for surface meshes and in Figure 5.12 and Figure 5.13 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. $Q2$ 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 [SDG+19], 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 (see example tessellations in Figure 5.11 and Figure 5.13). 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.

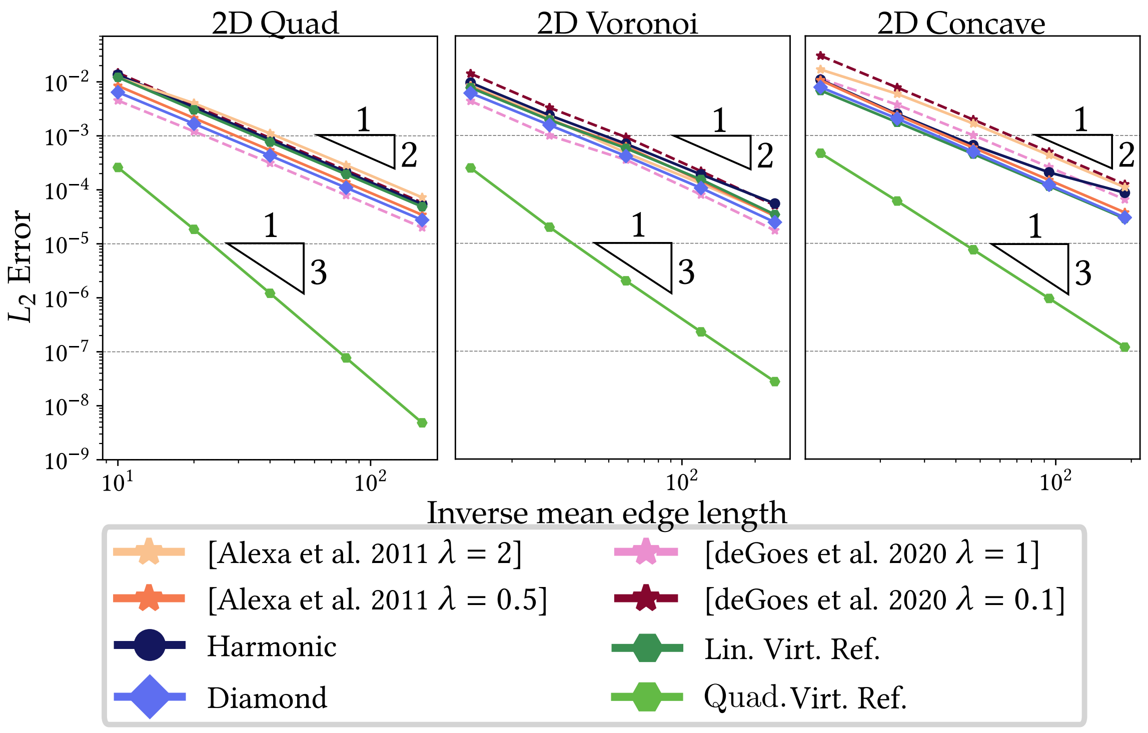


Figure 5.10: *L^2 error in log-log scale of the Poisson system solved for Franke's test function on planar meshes with quads (left), Voronoi cells (center), and concave faces (right).*

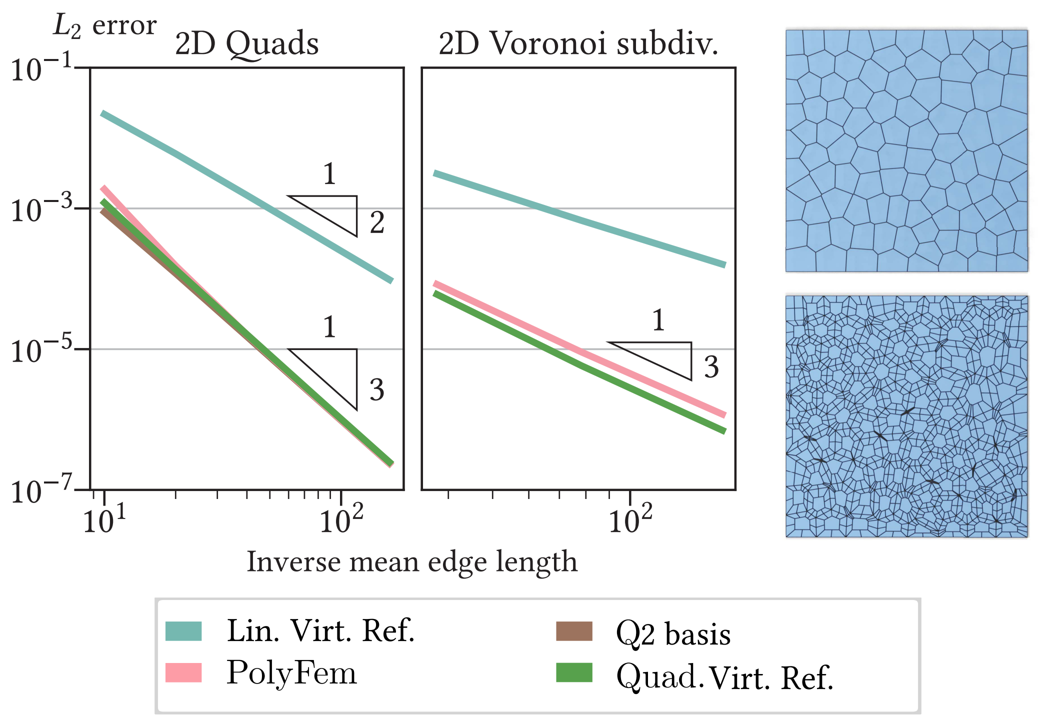


Figure 5.11: *L^2 error in log-log scale of the Poisson system solved for Franke's test function on planar grids with quads (left) and subdivided Voronoi cells (center) in accordance to the needs of the Poly-Spline method (right).*

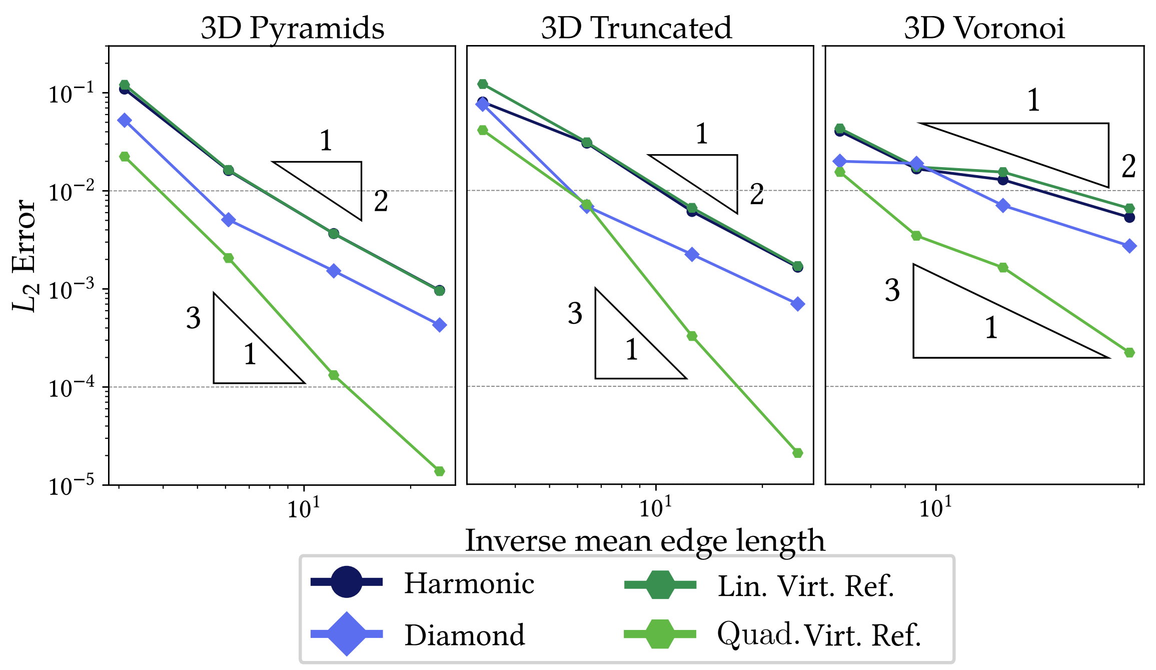


Figure 5.12: *L^2 error in log-log scale of the Poisson system solved for Franke's test function on different tessellations of the unit cube. The tesselations are as follows: Pyramids (left), truncated cells (center), and Voronoi cells (right).*

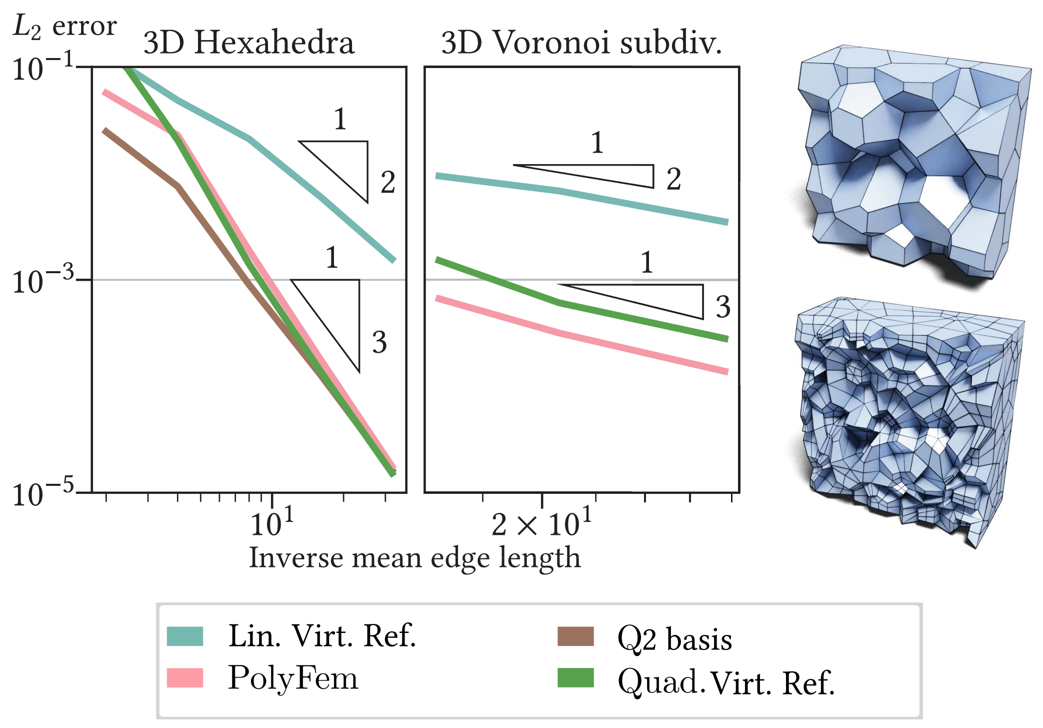


Figure 5.13: *L2 error in log-log scale of the Poisson system solved for Franke's test function on unit cubes with regular hexahedra (left) and subdivided Voronoi cells (center) in accordance to the Poly-Spline refinement (right).*

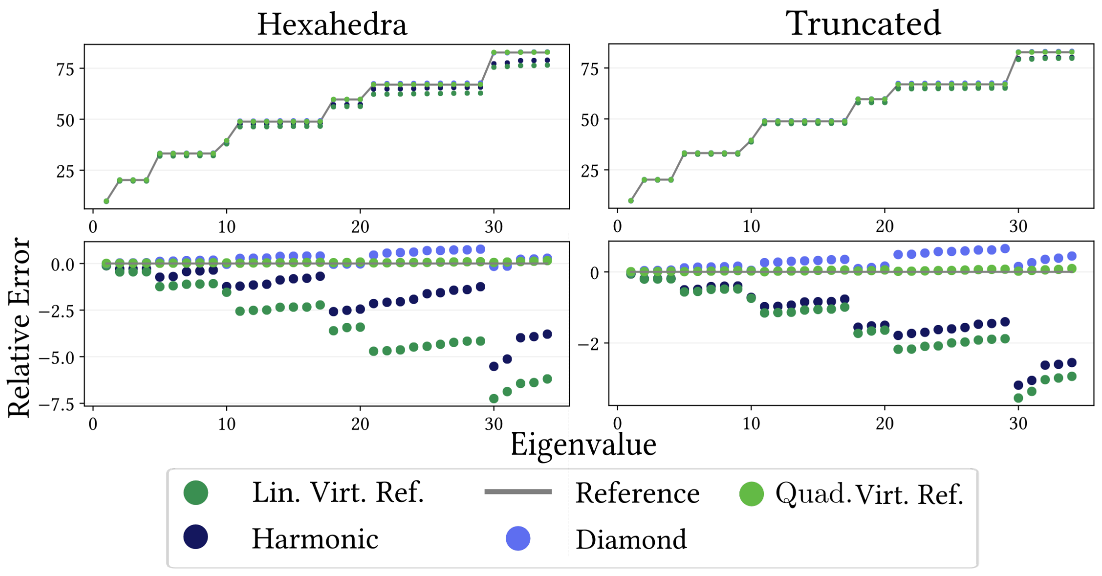


Figure 5.14: *The 34 smallest non-zero eigenvalues of the Laplacian, computed on a truncated polyhedral tessellation of the unit ball. The top plot shows the eigenvalues, the bottom shows the deviation from the ground truth. Our method outperforms all other discretizations, with results barely deviating from the desired values.*

Eigenvalue Reproduction Figure 5.14 shows the 34 smallest nonzero eigenvalues obtained with the different polyhedral Laplacians on hexahedral and truncated tessellations of the 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). The results in Figure 5.14 demonstrate that our basis functions generate more accurate eigenvalues, with significantly less deviation from the ground truth than those produced by competing approaches.

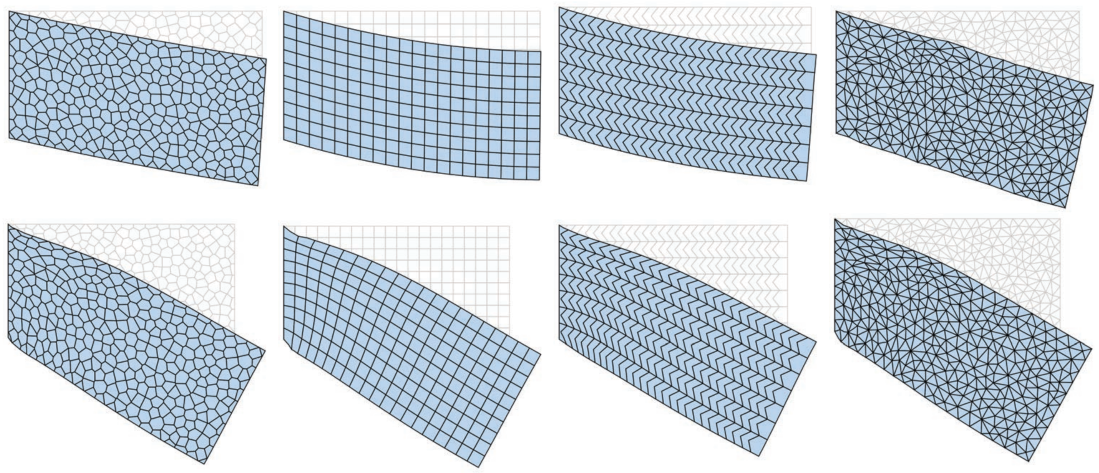


Figure 5.15: *Displacements due to gravity computed by linear elasticity with Young's modulus 1e+10 and Poisson ratio 0.4999 on differently tessellated 2D bars. The top row shows the results for our linear elements and the bottom row for our quadratic basis functions.*

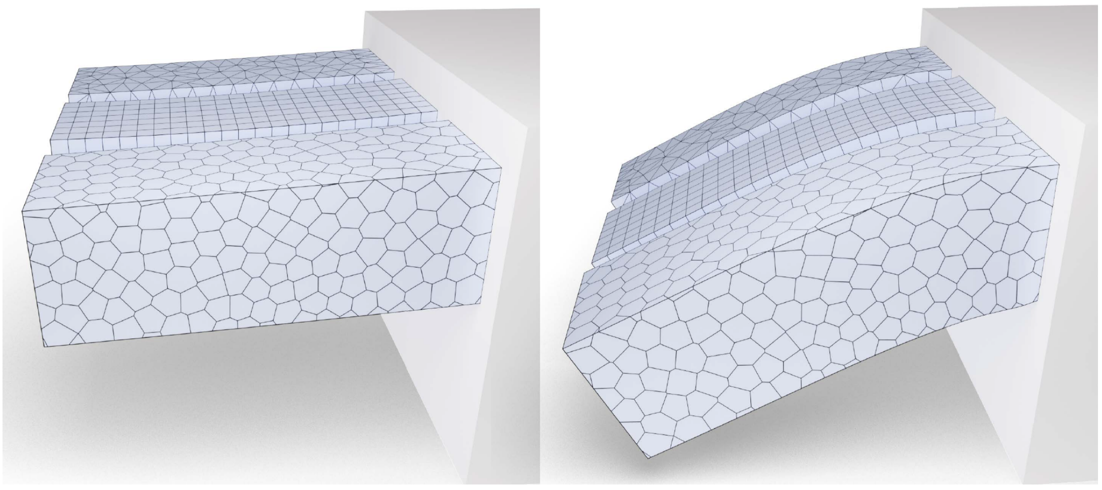


Figure 5.16: *Displacements due to gravity by linear elasticity with Young's modulus 5e+10 and Poisson ratio 0.4999 on differently tessellated bars consisting of tetrahedra, hexahedra, and Voronoi cells. The left figure shows the results for our linear basis functions and the right for our quadratic basis.*

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

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

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

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 5.15 and on the left of Figure 5.16. This problem can be reduced 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.

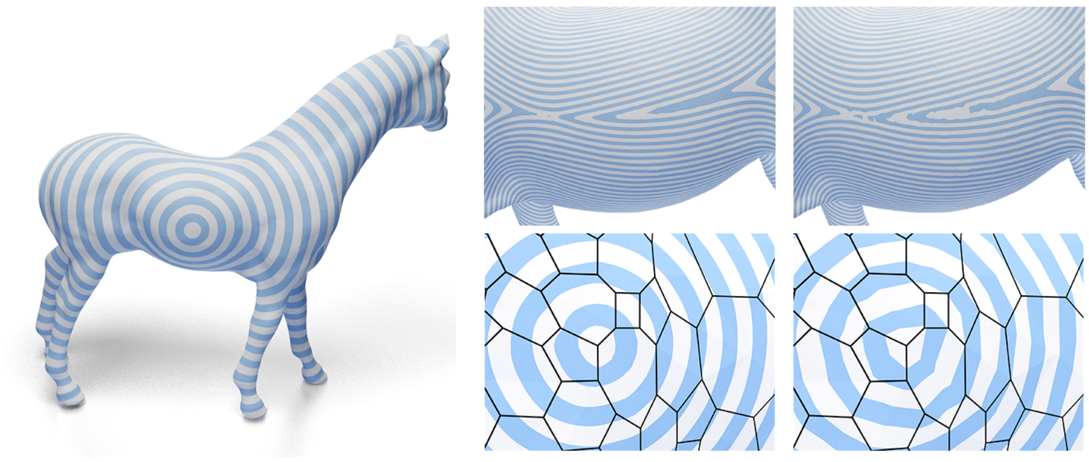


Figure 5.17: *Geodesics in heat [CWW13] on a polygonal mesh. Quadratic basis functions (left, center) lead to smoother results with less artifacts compared to the linear version (right). All images have been rendered by evaluating the result on triangulated and refined polygons.*

Geodesics in Heat Because our basis functions, and specifically their gradients, can be evaluated at any point on the mesh, we can straightforwardly implement Geodesics in Heat [CWW13] 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 5.17 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 [CDR00]. For example, this allows us to implement Line Integral Convolution as in [PKC+18]. 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 5.18 shows the resulting visualization of the maximal curvature directions field on the fertility model.

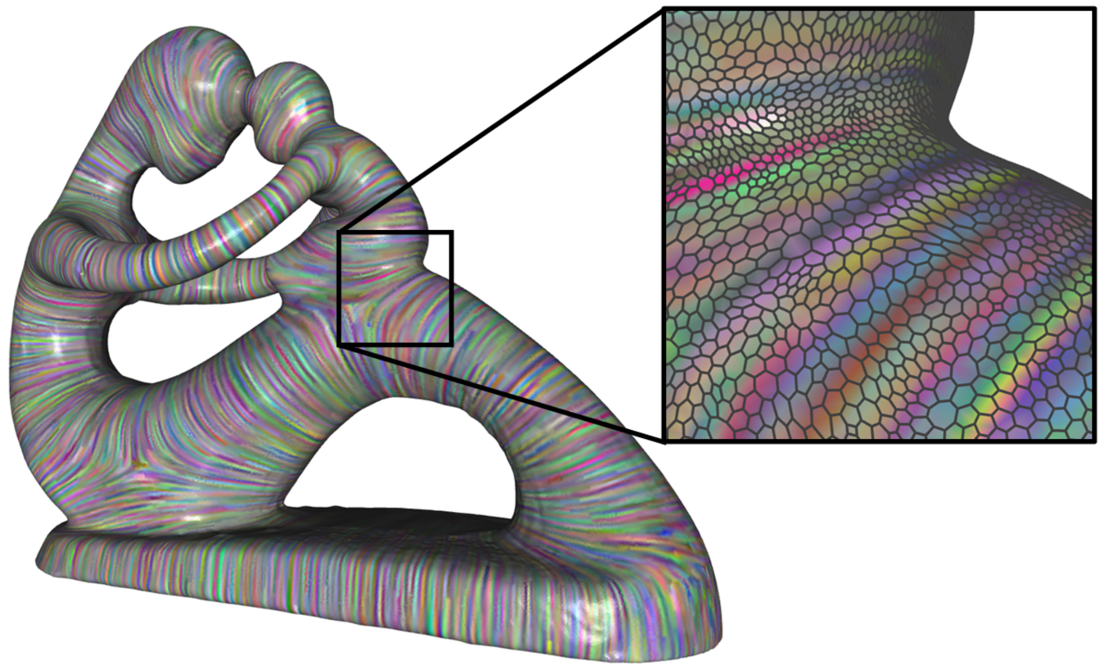


Figure 5.18: *Line Integral Convolution visualization obtained by anisotropically diffusing a random color signal along the maximum curvature direction.*

5.4.2 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 5.3.4, sidesteps the direct solution of the full system and employs a direct solver at the coarse level only. In Figure 5.19 we illustrate its convergence behavior using four polyhedral meshes (all hex-dominant except for the first one which

Table 5.1: *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* $\|\mathbf{x} - \mathbf{x}_{ref}\| \,/\, \|\mathbf{x}_{ref}\| < 10^{-8}$ *with respect to the direct solution.*

Mesh	$\lvert\mathcal{V}\rvert$	Harmonic			Lin. Virt. Ref.			Diamond			Quad. Virt. Ref.			
		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

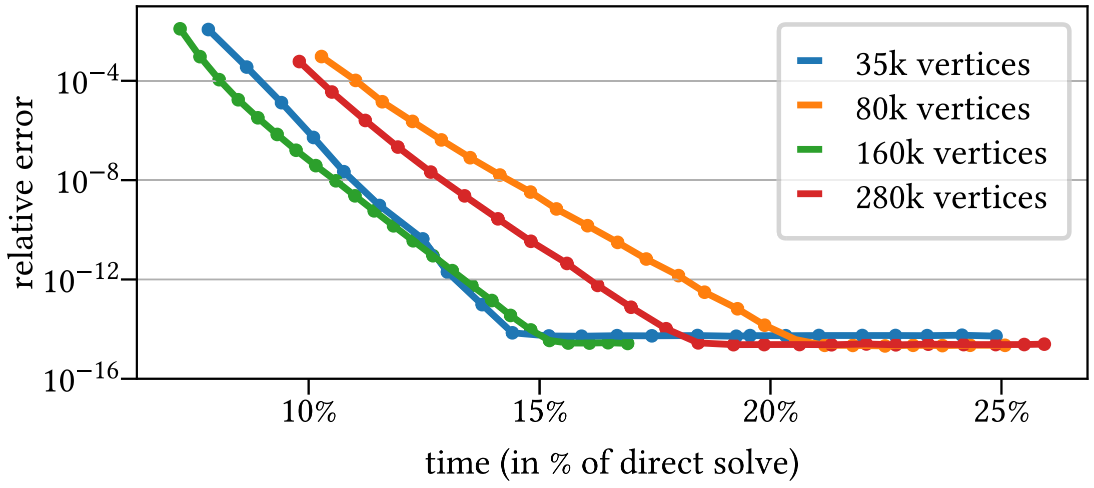


Figure 5.19: *Multigrid convergence for four polyhedral meshes of different size. Each point represents a V-cycle and we report time relative to a direct solve of the full system for each mesh. The relative error is measured by* $\|\mathbf{x} - \mathbf{x}_{ref}\| \,/\, \|\mathbf{x}_{ref}\|$ *where* $\mathbf{x}_{ref}$ *is the direct solution.*

contains pyramids, see Figure 3.13). 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 solver implemented in CHOLMOD [CDH+08]. 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 5.1 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 [AW11; GBD20; MKB+08; BHK+20]. As we use a direct sparse Cholesky solver, discrete Laplacians with the same non-zero structure give the same solve times. The Diamond Laplacian introduces coefficients relating vertex nodes of adjacent elements (see Section 4.5) and therefore leads 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. The timings listed in Table 5.1 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.

5.5 LIMITATIONS

As discussed in Section 5.3.3, we incorporate a Dirichlet regularizer scaled with a very small $\varepsilon = 10^{-8}$ into our energy. This regularizer guarantees quadratic reproduction on triangles and tetrahedra and other beneficial properties for our system. However, it also affects the quadratic precision property of our shape functions. While quadratic functions lie in the span of our basis, they do have a non-zero Dirichlet energy. Therefore, it remains to be shown that the chosen prolongation weights of our energy do not differ from those needed to interpolate the quadratic function. Since we have to choose a ε to guarantee the properties listed in Appendix B, it generates the possibility to interpolate a function with non-continuous gradients but lower Dirichlet energies. While parameters close to zero make the space of potentially interpolated functions arbitrarily small, we cannot guarantee the quadratic precision property. A more detailed discussion on this matter can be found in Appendix C.1.

5.6 CONCLUSION

In this chapter we presented a new approach, called the quadratic virtual refinement mehthod, for defining finite elements shape functions for general polygonal and polyhedral meshes. In contrast to the linear approaches presented in the previous chapters of this thesis, 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.

6 POLYGON LAPLACIAN MADE ROBUST

At this point, we have thoroughly investigated the potential of the linear virtual refinement method to be extended to higher-order shape functions and explored alternative numerical discretization schemes for the refined tessellation. As we approach the conclusion of this thesis, our attention returns to the original method introduced in Chapter 3. While we have investigated many avenues and possible extensions of the linear virtual refinement method, we have not yet extensively touched upon its degrees of freedom, particularly the placement of the virtual vertex and the selection of the prolongation weights. Our initial choices were based on empirical observations, as described in Section 3.2.3, but, as the alternative weight computation from Chapter 5 demonstrates, there might remain a lot of room for improvement. Thus, we reconsider what the ideal selection for the degrees of freedom for the linear virtual refinement method might be.

We made the central observation that the virtual triangulation of the polygons allows us to utilize the extensive knowledge from the linear Finite Element Method regarding *triangle* shapes. This, in turn, allows us to construct an optimized version of our polygon Laplace operator. From linear FEM, it is known that (and how) the shape of the elements (i.e., triangles) affects the accuracy and numerical stability of the cotangent Laplacian [She02]: In general, the ratio of squared edge lengths to triangle areas should be small. In this chapter, we want to create virtual triangulations that minimize this ratio. Notably, this measure has appeared before as the *harmonic index* of a triangle and has been linked to the Delaunay triangulation [Mus97] and the discrete Laplace operator [BS07]. In particular, the sum of the harmonic indices across all triangles in the mesh corresponds to the *trace* of the cotangent stiffness matrix [Ale19]. Using the minimization of the trace as the guiding principle, we derive optimal positions for the virtual vertices within the polygons and optimal weights to express these points as affine combinations of the polygon vertices. Additionally, we derive a smoothing scheme that optimizes the vertex locations (for applications where this is permissible) based on the same principles.

We analyze how minimizing the trace affects the spectrum of the discrete Laplace operator and compare this optimized Laplacian to the previously introduced operators within the established test scenarios. Through our experiments, we demonstrate that for low-quality polygon meshes, the optimized Laplacian consistently and significantly enhances robustness and accuracy – with only negligible computational cost for constructing the discrete operator and no additional computational overhead for solving the involved linear systems.

Individual Contribution *I investigated and optimized the linear virtual refinement method in close cooperation with Dennis Bukenberger and Sven Wagner, supervised by Marc Alexa and Mario Botsch. I am responsible for the mathematical proofs and derivations for the optimized degrees of freedom of the method, as well as the energy minimizations provided in this chapter. Many ideas were motivated by the thorough results provided by Sven Wagner in his Bachelor's thesis, which I supervised. Dennis Bukenberger implemented and developed the proposed smoothing algorithm. The implementation of the Laplace optimization was a collective effort by Sven Wagner, Dennis Bukenberger, and me.*

Corresponding Publication *This chapter is based on the following publication:*

Bunge, A., Bukenberger, D. R., Wagner, S. D., Alexa, M., and Botsch, M. (2024) "Polygon Laplacian Made Robust." Computer Graphics Forum 43(2).

6.1 RELATED WORK

As established in the introduction, this chapter aims to enhance the numerical accuracy and stability of the linear virtual refinement method. Consequently, it becomes necessary to explore the nuances of numerical analysis on polygon meshes and their associated challenges. Moreover, although the concept of polygon Laplacians has been thoroughly investigated up to this point in the thesis, the idea of polygon mesh smoothing approaches has yet to be discussed. Therefore, we want to address this gap by examining relevant research in this area.

Numerical Accuracy and Robustness On Polygons The shape of a simplex can affect the quality of linear FEM discretizations for triangles and tetrahedra. Shewchuk [She02] analyzed various aspects of this phenomenon, which were later expanded on in the exhaustive survey by Sorgente et al. [SBM+23]. However, Sorgente et al. also noted that many concepts used to evaluate the geometric quality of lower-degree polytopes cannot be easily transferred to generic polygons. To address this issue, several works developed shape quality metrics specifically for surface and volume polytopes, such as Lipnikov [Lip13], Mu et al. [MWW15], and Gillette and Rand [GR17]. In a recent study, Attene et al. [ABB+21] investigated the correlation between the performance of VEM [VBM13] and the underlying polygonal tessellation. This chapter expands on this type of analysis and establishes a geometric link between polygons and the linear virtual refinement method.

Polygon Mesh Smoothing The ability to enhance the quality of a mesh by improving, or *smoothing* the shapes of the individual faces, is a valuable tool in geometry processing. The effectiveness of these algorithms has been extensively studied, particularly in 2D. This extends to meshes in 3D when the placement of (interior) vertices is less critical, e.g., tessellations of planes in CAD-like objects as long as sharp feature edges and corners are retained. Mesh smoothing impacts numerical stability and discretization error, a crucial consideration in computational simulations. Numerous approaches have been introduced for triangle meshes, with some extending to quads or polygonal meshes.
Zhou and Shimada [ZS00] focus on optimizing the inner angles within mixed 2D meshes. It has been shown to be effective for triangles, as adjusting the angles generally improves the ratio of edge lengths to area (Section 6.2). The extension to quads is based on introducing virtual diagonals, and it is unclear how to extend this concept to general polygons. The works of Garimella et al. [GSK02; GSK04; GS04] aim for optimizing triangle, quad, mixed and polygon manifold surface meshes. They formulate an objective function to minimize the condition numbers of local Jacobian matrices, which then indirectly improve the global mesh quality. However, by construction, this approach can only incorporate direct vertex neighbors and, therefore, does not generalize well for larger polygons, limiting the effect on the condition of the global stiffness matrix. Further, Knupp et al. [KMS02] proposed mesh optimization using Reference Jacobian Matrices, which helps preserving mesh details but is not suitable for polygon meshes due to ambiguous local Jacobian definitions. Vartziotis et al. [VAG+08] propose the Geometric Element Transformation Method (GETMe) to successively regularize low quality triangles by iteratively applying a geometric transformation until a user defined quality level is reached. Generalizations for general polygonal, tetrahedral [VW12] and later general polyhedral meshes are suggested, but the official GETMe implementation currently only supports smoothing 2D polygonal shapes with fixed boundary vertices. Our approach sets itself apart by optimizing the complete polygonal shape, and not only the interior angles at vertices. It is specifically tailored for the use of our proposed Laplace operator, taking into account the entire polygonal configuration in relation to the virtual vertex.

6.2 ACCURACY AND ROBUSTNESS ON TRIANGLE MESHES

For linear FEM, Shewchuk has analyzed how the shape of triangles affects various aspects of the discretization [She02]. He argues that for the interpolation error it is more important to control the error in the gradients and gives a smooth, *size-independent* measure of the quality of an element in this respect [She02, Tab. 4:]

$$\frac{\left|t_{ijk}\right|}{\left(\left|e_{ij}\right|^2 \left|e_{jk}\right|^2 \left|e_{ik}\right|^2\right)^{\frac{1}{3}}}, \tag{6.1}$$

where larger ratios indicate better shaped triangles. As before, the scalar $\left|e_{ij}\right|$ denotes the length of an edge e_{ij} and $\left|t_{ijk}\right|$ the area of a triangle t_{ijk}. For analyzing (or improving) the stiffness matrix' condition number, again with a smooth and scale-independent measure, he suggests [She02, Tab. 4:]

$$\frac{\left|t_{ijk}\right|}{\frac{1}{3}\left(\left|e_{ij}\right|^2 + \left|e_{jk}\right|^2 + \left|e_{ik}\right|^2\right)}. \tag{6.2}$$

The condition number κ of a quadratic matrix $\mathbf{A}$ is defined as the ratio between its largest and smallest eigenvalue

$$\kappa(\mathbf{A}) = \frac{\lambda_{max}(\mathbf{A})}{\lambda_{min}(\mathbf{A})} \tag{6.3}$$

and is a common measure to quantify numerical stability. Equation (6.2) is inversely proportional to the *harmonic index* of the triangle [Mus97; BS07; CXG+10]:

$$\eta(t_{ijk}) = \frac{\left|e_{ij}\right|^2 + \left|e_{jk}\right|^2 + \left|e_{ik}\right|^2}{\left|t_{ijk}\right|} = 4 \sum_{v_i \in t_{ijk}} \cot\theta_i. \tag{6.4}$$

Notice that the two quality measures in Equation (6.1) and Equation (6.2) differ only by how the average is taken over the squares of the three edge lengths. By the inequality of arithmetic and geometric means (AM-GM inequality), the arithmetic mean in the denominator of the condition quality measure (6.2) is not smaller than the geometric mean in the denominator of the interpolation error measure (6.1). This means that *minimizing* the harmonic index $\eta(t_{ijk})$ improves the condition of the stiffness matrix w.r.t. the triangle t_{ijk}, and generally also improves the accuracy of the gradient in the solution. This is the case, because triangles with large angles contain points where the gradient interpolation error increases immensely [She02]. Since the minimization of the harmonic index avoids angles close to 0 and 180 degrees, these shapes do not occur, leading to well-behaved gradients.

The work of Alexa [2019] shows that the trace of the stiffness matrix (per element or for the whole mesh) is proportional to the sum of the harmonic indices of the triangle(s):

$$\begin{aligned} H(\mathcal{T}) &= \sum_{t_{ijk}\in\mathcal{T}} \eta(t_{ijk}) && (6.5)\\ &= 4 \sum_{t_{ijk}\in\mathcal{T}} (\cot\theta_i + \cot\theta_j + \cot\theta_k) && (6.6)\\ &= 4\,\mathrm{tr}(\mathbf{S}^{\triangle}). && (6.7) \end{aligned}$$

In the following, we will use the idea of minimizing the trace of the triangle cotangent operator for optimizing the operators for polygon meshes.

6.3 ACCURACY AND ROBUSTNESS ON POLYGON MESHES

In Section 6.2 we established that a smaller trace of the cotangent stiffness matrix generally indicates a numerically preferable triangle mesh. In this section, we aim to investigate whether the same observation holds for the trace of the polygon stiffness matrix given by the linear virtual refinement method (see Equation (3.7)) and whether we can establish a similar connection to the geometry of the polygon.
When dealing with triangles, the cotangent Laplacian is solely based on the tessellation and cannot be altered unless the triangle mesh is modified. However, when working with polygons, we have two degrees of freedom to consider: the affine weights used in the prolongation matrix and the placement of the virtual vertex. As a result, we can optimize these parameters with respect to the trace of the polygon stiffness matrix *without* changing the polygon mesh and examine how it affects the Laplacian.

6.3.1 Optimal Prolongation Weights

We will start with the affine weights for the virtual vertex. As in Section 3.2.3, assume we are given a polygon $f \in \mathcal{F}$ with vertices $(v_1, \dots, v_{n_f})$ and its corresponding virtual face vertex v_f. Let $\mathbf{S}^{\triangle} \in \mathbb{R}^{(n_f+1)\times(n_f+1)}$ denote the local cotangent stiffness matrix constructed on the virtual triangulation and $\mathbf{w}_f = \left(w_1, \dots, w_{n_f}\right)^{\mathsf{T}} \in \mathbb{R}^{n_f}$ the set of affine weights. We build the local prolongation matrix $\mathbf{P} \in \mathbb{R}^{(n_f+1)\times n_f}$ as defined in Equation (3.5) and obtain the local polygon stiffness matrix $\mathbf{S}^{\pentagon}$ as established in Equation (3.7).

The objective is to minimize the trace of $\mathbf{S}^{\bigcirc}$ with respect to the individual prolongation weights $w_i \in \mathbf{w}_f$. We first rearrange the polygon stiffness matrix into a similar pattern as in Equation (5.3): We divide the matrix into the cotangent entries related to the existing polygon vertices and a matrix that redistributes the values associated with the virtual vertex v_f:

$$\mathbf{S}^{\bigcirc} \;=\; \mathbf{P}^{\mathsf{T}}\mathbf{S}^{\triangle}\mathbf{P} \;=\; \mathbf{S}^{\triangle}_{\text{sub}} + \mathbf{W}. \tag{6.8}$$

The matrix $\mathbf{S}^{\triangle}_{\text{sub}}$ is the symmetric $n_f \times n_f$ submatrix of $\mathbf{S}^{\triangle}$ excluding the row and column associated with the virtual vertex. In our setting, the cotangent weights associated with the virtual vertex are located at row/column $(n_f + 1)$ of $\mathbf{S}^{\triangle}$. We will use the vector

$$\mathbf{s} = \left(\mathbf{S}^{\triangle}_{1,n_f+1}, \dots, \mathbf{S}^{\triangle}_{n_f,n_f+1}\right)^{\mathsf{T}} \in \mathbb{R}^{n_f}$$

and its respective entries $(s_1, \dots, s_{n_f})$ to refer to the first n_f entries of the virtual vertex column. The matrix $\mathbf{W} \in \mathbb{R}^{n_f \times n_f}$ can be defined as

$$\mathbf{W}_{ij} = \begin{cases} 2w_i s_i - w_i^2 \sum\limits_{k=1}^{n_f} s_k & \text{if } i = j, \\ w_i s_j + w_j s_i - w_i w_j \sum\limits_{k=1}^{n_f} s_k & \text{otherwise,} \end{cases} \tag{6.9}$$

and redistributes the values of the virtual vertex onto the original n_f nodes of the polygon.
We minimize the trace of the polygon stiffness matrix with respect to the affine weights $w_1, \dots, w_{n_f}$ by setting the respective partial derivatives to zero. Here we only have to consider the trace of the matrix $\mathbf{W}$, since $\mathbf{S}^{\triangle}_{\text{sub}}$ does not depend on the weights w_i:

$$\frac{\partial \operatorname{tr}(\mathbf{S}^{\bigcirc})}{\partial w_i} = \frac{\partial \operatorname{tr}(\mathbf{W})}{\partial w_i} = 2s_i - 2w_i \sum_{j=1}^{n_f} s_j. \tag{6.10}$$

Setting Equation (6.10) to zero leads to the trace-optimal weights

$$w_i = \frac{-(\cot \alpha_{i,n_f+1} + \cot \beta_{i,n_f+1})}{-\sum_{j=1}^{n_f} \cot \alpha_{j,n_f+1} + \cot \beta_{j,n_f+1}} = \frac{s_i}{\sum_{j=1}^{n_f} s_j} =: \frac{s_i}{\omega} \tag{6.11}$$

with $\omega \in \mathbb{R}$ referring to the negated $(n_f + 1)$-th diagonal entry of $\mathbf{S}^{\triangle}$ associated with the virtual vertex.
These trace-minimizing weights are well-known as the *discrete harmonic coordinates* [PP93; EDD+95] and can be computed through Equation (6.11) without numerical optimization, in contrast to the least norm weights suggested in the original linear virtual refinement method (see Equation (3.11)). Note that

these weights are affine, since they by construction sum to one, and that the denominator ω is guaranteed to be non-zero as long as the virtual vertex v_f is positioned in the kernel of a star-shaped polygon. We point out that using these weights in the prolongation step simplifies the entries of the matrix **W** to

$$\mathbf{W}_{ij} = \begin{Bmatrix} s_i^2/\omega & \text{if } i=j \\ s_i s_j/\omega & \text{otherwise} \end{Bmatrix} = \frac{1}{\omega}\mathbf{s}\mathbf{s}^{\mathsf{T}}, \tag{6.12}$$

which we will exploit in the upcoming section.

6.3.2 Eigenvalues of the Polygon Stiffness Matrix

In this section, we uncover a geometric link between the local eigenvalues of the polygon stiffness matrix and the polygon itself. Assume the local prolongation matrix **P** on the polygon f is obtained with the discrete harmonic weights established in the previous section. Given that $\mathbf{S}^{\pentagon}$, $\mathbf{S}^{\triangle}_{\text{sub}}$, and **W** are all $n_f \times n_f$ Hermitian matrices and $\mathbf{S}^{\pentagon} = \mathbf{S}^{\triangle}_{\text{sub}} + \mathbf{W}$, we can make the following observation: Let μ_i, ν_i and ρ_i be the respective eigenvalues of $\mathbf{S}^{\pentagon}$, $\mathbf{S}^{\triangle}_{\text{sub}}$, and **W**, ordered as follows:

$$\mathbf{S}^{\pentagon}: \quad \mu_1 \geq \cdots \geq \mu_{n_f-1} > \mu_{n_f} = 0, \tag{6.13}$$

$$\mathbf{S}^{\triangle}_{\text{sub}}: \quad \nu_1 \geq \cdots \geq \nu_{n_f-1} \geq \nu_{n_f}, \tag{6.14}$$

$$\mathbf{W}: \quad \rho_1 \geq \cdots \geq \rho_{n_f-1} \geq \rho_{n_f}. \tag{6.15}$$

Applying Weyl's inequality [Wey12] yields

$$\nu_i + \rho_{n_f} \leq \mu_i \leq \nu_i + \rho_1, \quad i = 1, \ldots, n_f. \tag{6.16}$$

which links the individual eigenvalues of the polygon stiffness matrix $\mathbf{S}^{\pentagon}$ to the eigenvalues of the submatrix $\mathbf{S}^{\triangle}_{\text{sub}}$.
But what about the eigenvalues of **W**? Using the discrete harmonic coordinates as prolongation weights allows us to directly determine the eigenvalues ρ_i. As an outer-product matrix (see Equation (6.12)) **W** has rank one and a kernel of dimension $n_f - 1$, leading to $n_f - 1$ vanishing eigenvalues. The only remaining non-zero eigenvalue is

$$\rho = \frac{\|\mathbf{s}\|^2}{\omega} \tag{6.17}$$

with the eigenvector **s**, since

$$\mathbf{W}\mathbf{s} = \left(\frac{1}{\omega}\mathbf{s}\mathbf{s}^{\mathsf{T}}\right)\mathbf{s} = \frac{\|\mathbf{s}\|^2}{\omega}\mathbf{s}. \tag{6.18}$$

The question remains whether ρ is the largest or the smallest eigenvalue of **W**. Since $\mathbf{S}^{\triangle}$ is positive semi-definite, ω is negative, which means that $\rho_{n_f} = \rho$ is negative as well and is therefore the smallest eigenvalue of $\mathbf{W}$. This allows us to simplify Equation (6.16) to

$$\nu_i + \frac{\|\mathbf{s}\|^2}{\omega} \leq \mu_i \leq \nu_i \quad \text{for } i = 1, \ldots, n_f. \tag{6.19}$$

Therefore, as a first conclusion, we can observe that the i-th eigenvalue μ_i of the polygon stiffness matrix is confined by the i-th eigenvalue ν_i of the submatrix $\mathbf{S}^{\triangle}_{\text{sub}}$.
Conveniently, the matrix $\mathbf{S}^{\triangle}_{\text{sub}}$ is an n_f-dimensional principal submatrix of the $(n_f + 1)$-dimensional symmetric cotangent stiffness matrix $\mathbf{S}^{\triangle}$, allowing us to apply Cauchy's Interlace Theorem [Hwa04], which reveals the following eigenvalue relationship:

$$0 = \lambda_{n_f+1} \leq \nu_{n_f} \leq \lambda_{n_f} \leq \cdots \leq \lambda_2 \leq \nu_1 \leq \lambda_1, \tag{6.20}$$

with λ_i denoting the $n_f + 1$ eigenvalues of the cotangent stiffness matrix $\mathbf{S}^{\triangle}$. Merging this inequality with the upper and lower bounds from Equation (6.19) results in the final bounds

$$\lambda_{i+1} + \frac{\|\mathbf{s}\|^2}{\omega} \leq \nu_i + \frac{\|\mathbf{s}\|^2}{\omega} \leq \mu_i \leq \nu_i \leq \lambda_i, \quad i = 1, \ldots, n_f. \tag{6.21}$$

We are now able to draw a powerful conclusion: All eigenvalues of the polygon stiffness matrix $\mathbf{S}^{\pentagon}$ are less than or equal to the corresponding eigenvalues of the cotangent stiffness matrix $\mathbf{S}^{\triangle}$.
Therefore, the upper bound in Equation (6.21) directly establishes a geometric connection between the stiffness matrix and the polygon, albeit indirectly through the virtual triangle fan. Polygons that allow their virtual vertex to form a high-quality triangle fan (by the standards established in Section 6.2) can minimize the trace of the virtual cotangent stiffness matrix and, consequently, improve that of the polygon stiffness matrix. Not only that, but a good virtual triangulation also implies a cotangent matrix of higher quality, which also improves the performance of the polygon Laplacian. Furthermore, the condition number of the polygon stiffness matrix is expected to improve proportionally to that of the cotangent Laplacian.

6.3.3 Optimal Choice for Virtual Vertex Placement

The findings from the previous section establish a suitable objective function for the remaining degree of freedom: the placement of the virtual vertex. The point that minimizes (within the kernel of the polygon) the trace of the cotangent stiffness matrix will also minimize the harmonic index of the virtual triangle fan (see Equation (6.5)) and consequently further improve the quality of individual (virtual) triangles.

We start with the case of a *planar polygon* and consider the cotangent stiffness matrix $\mathbf{S}^{\triangle}$ on the virtual triangles $\mathcal{T}_f$ of a polygon f with virtual vertex v_f. We want to minimize the following energy with respect to the virtual vertex position $\mathbf{x}_f$:

$$E(\mathbf{x}_f) = \operatorname{tr}(\mathbf{S}^{\triangle}) = \sum_{t \in \mathcal{T}_f} \sum_{v_j \in t} \cot(\theta_j). \tag{6.22}$$

We minimize this energy using the projected Newton method, which ensures a positive definite Hessian. In order to prevent virtual triangles from flipping, we start the minimization from the squared area minimizer (see Equation (3.11)), which is guaranteed to lie in the polygon's kernel, and return an infinite cost during the line search in case of negative or degenerate triangle areas. This avoids artifacts caused by the flipped triangles and guarantees the validity of our prolongation weights, since ω cannot become zero. Combining Equation (6.22) and the penalty term leads to a virtual vertex position that minimizes the trace of the stiffness matrix within the polygon's kernel.
As mentioned by Hormann and Floater [HF06], the discrete harmonic coordinates can represent points in the interior of a polygon as long as the resulting virtual triangle areas are positive and the edges connecting the virtual point to the polygon vertices remain non-degenerate. While a convex polygon satisfies these criteria for any point in its interior, star-shaped polygons guarantee these properties solely within the element's kernel. This condition is non-negotiable since reproducing the virtual vertex through the prolongation weights is necessary to retain the linear precision property for the polygon Laplacian (see Section 3.2.8).
The above approach, however, does not extend to polygon meshes embedded in $\mathbb{R}^3$ with potentially non-planar faces. First, we lose the notion of signed triangle area and flipped triangles. Second, for non-planar polygons only a limited set of points can be represented by discrete harmonic coordinates: Assuming that the virtual point $\mathbf{x}_f$ is represented by the weights w_i, and exploiting that the discrete harmonic weights are just the (normalized) cotangent weights for discretizing the Laplacian, we get

$$\begin{aligned} & \sum_{j=1}^{n_f} w_j \mathbf{x}_j = \mathbf{x}_f \\ \Leftrightarrow \quad & \sum_{j=1}^{n_f} \frac{s_j}{\omega} \mathbf{x}_j = \sum_{j=1}^{n_f} \frac{s_j}{\omega} \mathbf{x}_f \\ \Leftrightarrow \quad & \sum_{j=1}^{n_f} s_j \left(\mathbf{x}_j - \mathbf{x}_f\right) = \mathbf{0} \\ \Leftrightarrow \quad & \Delta \mathbf{x}_f = \mathbf{0}. \end{aligned} \tag{6.23}$$

As the cotangent Laplacian corresponds exactly to the gradient of surface area [DMS+99], the last condition means that the virtual vertex has to minimize

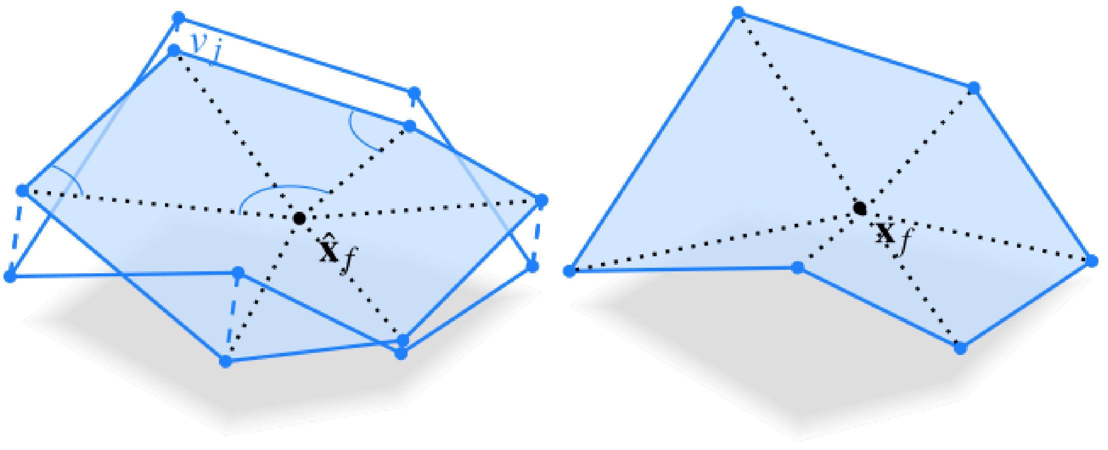


Figure 6.1: *Left: Non-planar polygon and its orthogonal projection with maximum area. The projection's virtual vertex position $\hat{\mathbf{x}}_f$ is the trace minimizer of the planar cotangent stiffness matrix. Right: The virtual vertex position $\mathbf{x}_f$ is obtained by multiplying the original vertices with the discrete harmonic weights of the projection.*

the sum of virtual triangle areas for the weights (6.11) to be able to reproduce the point. This is not an issue in the planar setting, where any point leads to the same area. For non-planar polygons, however, the trace minimizer and the area minimizer do in general not coincide, leading to a discontinuous energy when slightly deviating from a planar configuration.

We suggest an alternative approach that is able to retain linear precision: Given a non-planar polygon f, we consider its orthogonal planar projection with maximum surface area, as proposed by Alexa and Wardetzky [AW11]. To this end we project the polygon vertices $\mathbf{x}_i$ along the polygon's normal $\mathbf{n}_f$, which itself is defined in terms of the vector area $\mathbf{a}_f$:

$$\mathbf{a}_f = \frac{1}{2}\sum_{v_i \in f}\left(\mathbf{x}_i \times \mathbf{x}_{(i+1) \bmod n_f}\right), \quad \mathbf{n}_f = \frac{\mathbf{a}_f}{\|\mathbf{a}_f\|}, \quad \hat{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{n}_f\mathbf{n}_f^{\mathsf{T}}\mathbf{x}_i. \tag{6.24}$$

For the planar configuration $(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$ we minimize the trace of the stiffness matrix with respect to $\hat{\mathbf{x}}_f$ as described above. The resulting discrete harmonic weights $\hat{\mathbf{w}}_f$ are then used to define the virtual point for the original, non-planar polygon as $\mathbf{x}_f := \sum_j \hat{w}_j \mathbf{x}_j$.

For non-planar polygons that are still relatively close to a planar configuration, the virtual vertex will only slightly deviate from the trace minimizer of the orthogonal projection, as illustrated in Figure 6.1. This approach, therefore, avoids the discontinuity between the trace minimizer and the area minimizer. However, as the polygons become increasingly distorted, the results from this approach may become less favorable as the positions of the original and projected vertices increasingly differ. To further stabilize our approach, we integrate the fallback of comparing the trace of the optimized local polygon stiffness matrix to that of the original method. If the new trace is higher, we use the original point and weights instead. However, this is only necessary if the

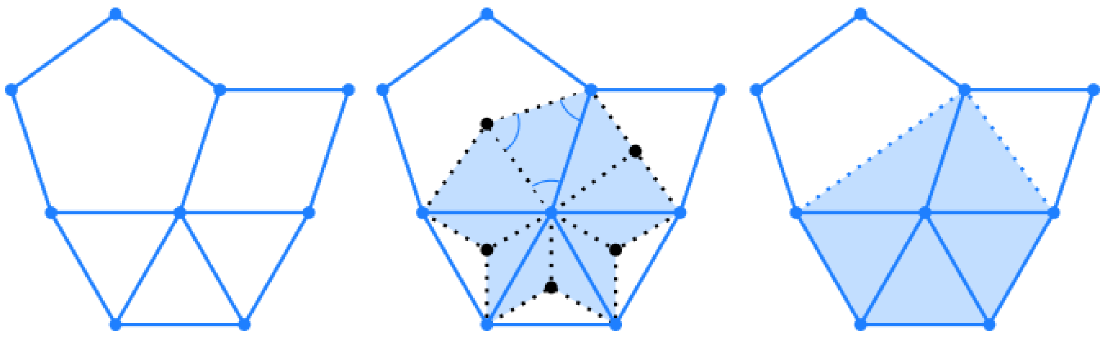


Figure 6.2: *In a mixed polygonal mesh (left) we also incorporate virtual vertices in the one-ring (center) whereas other methods often only consider real vertices (right) [Knu00; GS04; VAG+08].*

orthogonal projection of the polygon does not satisfy our initial assumption of being star-shaped, and almost never occurred in our experiments.

Although our energy minimizing requires an iterative approach, the implementation of is quite straightforward. Using the maximum orthogonal projection, we can express everything in an intrinsic 2D coordinate system, making the 2×2 Newton solver converge in just a few steps. The gradients and Hessian of Equation (6.22) can be derived analytically [Cra18], as well as the Hessian's eigenvalues required for the projection step.

6.4 POLYGON MESH SMOOTHING

In the previous sections we explored the intricate relationship between virtual vertex positions, affine prolongation weights, and the eigenvalues of the polygon Laplacian. We established that reducing the trace of the stiffness matrix improves the numerical quality of the Laplacian. So far, however, we have only focused on the degrees of freedom of the discrete Laplace operator itself, not on the "real" non-virtual vertices of the polygon mesh. If we also take these into account when optimizing the trace of the stiffness matrix, we can improve the numerical conditioning much further. Consider a polygon mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ with m vertices and their associated positions $\mathbf{x}_1, \ldots, \mathbf{x}_m$, as well as a virtual triangulation $\mathcal{M}_T = (\mathcal{V}^\triangle, \mathcal{T})$ generated by inserting a virtual vertex into each polygonal face. Following the insights from Sections 6.2 and 6.3, we want to optimize the trace of the cotan Laplacian on the virtual triangulation $\mathcal{T}$ with respect to the positions of the polygon vertices $v_i \in \mathcal{V}$, as shown in Figure 6.2, center. The trace of the global cotan stiffness matrix can be formulated as the energy

$$E(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \sum_{t \in \mathcal{T}} \sum_{v_j \in t} \cot(\theta_j), \tag{6.25}$$

which sums up local cotangent components of the corner angles θ_j at the vertices v_j within all virtual triangles $t \in \mathcal{T}$. We minimize this energy in a Newton-like manner, but decouple the relationship of real and virtual vertices in an alternating optimization scheme, such that each Newton iteration has these two steps:

1. Minimize trace (as in Equation (6.25)) with respect to all real vertex positions $\mathbf{x}_1, \ldots, \mathbf{x}_m$, while keeping (the affine weights of) virtual vertices fixed.
2. For each face f, minimize trace (as in Equation (6.22)) with respect to virtual vertex positions $\mathbf{x}_f$ and weights $\mathbf{w}_f$, while keeping the real vertices fixed.

This formulation not only simplifies the implementation, it also increases the robustness of the numerical optimization. We implement the smoothing (Step 1) using TinyAD [SBB+22].

When optimizing polygon meshes embedded in $\mathbb{R}^3$, we have to restrict vertex movement to the mesh surface to prevent deviation of the optimized mesh from the original geometry. This is typically done by restricting vertex movement to their tangent planes and/or re-projecting vertices to the original mesh after each iteration. For technical models with sharp features, however, this will not accurately preserve feature edges and corners.

We propose to instead adapt the quadric error metric [GH97] to our setting and to directly incorporate it into the optimization. At initialization time, each vertex v_i is assigned a 4×4 quadric $\mathbf{Q}_i$ built by summing up the quadrics of v_i's incident faces f. As proposed by Garland and Heckbert [GH97], the face quadrics are constructed from area-weighted normal vectors, i.e., from the vector areas $\mathbf{a}_f$. To optimize the trace of the stiffness matrix while keeping vertices on the surface and on the features, we minimize the energy

$$E(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \sum_{t \in \mathcal{T}} \sum_{v_j \in t} \cot(\theta_j) + \tau \sum_{v_j \in \mathcal{V}} Q_j(\mathbf{x}_j)\,, \tag{6.26}$$

where

$$Q(\mathbf{x}) = (\mathbf{x}^\mathsf{T}, 1)\, \mathbf{Q} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} \tag{6.27}$$

denotes the quadric error of vertex position $\mathbf{x}$ with respect to the quadric $\mathbf{Q}$. As the τ parameter can be adjusted to fine-tune the optimization process, it is important to note that the sum of cotan values often tends to become relatively large. Consequently, to effectively balance the influence of cotan sums and quadric errors, higher τ values perform better and we used $\tau = 10^5$ for all results. While the quadrics could be updated after each Newton iteration, we typically keep them fixed to restrict the mesh optimization to rather local modifications. Figure 6.3 compares the results of unconstrained ($\tau = 0$) and quadric-constrained ($\tau = 10^5$) optimization. While in the unconstrained case

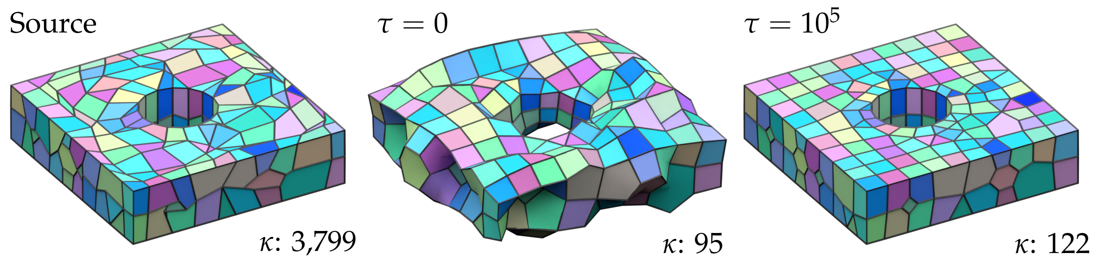


Figure 6.3: *Without constraints, surface meshes may get disfigured (center). Due to the quadrics energy, vertices move only on the surface and sharp features are preserved automatically (right). The condition number κ of the polygon stiffness matrix $\mathbf{S}^{\varhexagon}$ significantly improves on our smoothed tessellation compared to the input mesh.*

a lower condition number κ is achieved, this comes at the price of an unacceptable deviation from the original shape. Figure 6.4 compares to the polygon mesh smoothing of Garimella et al. [GS04]. Their energy formulation for the apex vertex has multiple equivalent minima on the lateral sides of the pyramid, which crumples down the pyramid in one iteration and would eventually flatten it completely. Our energy is minimal at the apex itself due to the symmetry of the constellation, thus the vertex is not moved (even without quadric constraints).

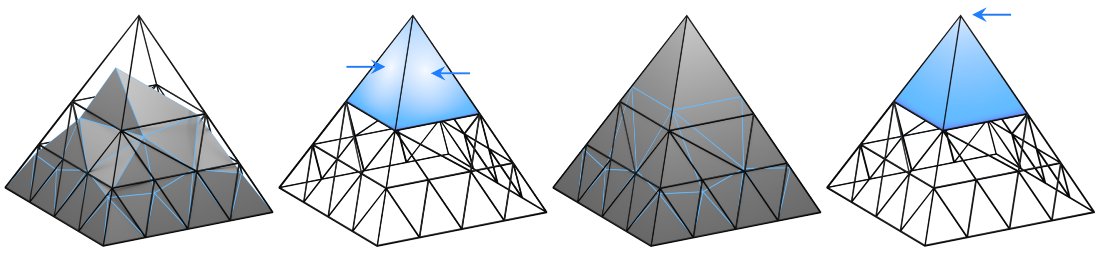


Figure 6.4: *Results of Garimella [GS04] (left) and ours (right). As highlighted, their energy function has multiple equivalent minima for the apex vertex on all four sides, thus crumples the pyramid, whereas ours remains straight as the minimum is exactly at the top.*

6.5 RESULTS AND DISCUSSION

In this section we present experiments, comprehensive comparisons, and discussion of the results, highlighting the effectiveness of our approach on mixed polygon meshes.

6.5.1 Optimized Polygon Laplacian

We expand our test setting with the final polygon Laplacian of this thesis and analyze its performance and stability. As before, we compare our results to the previously introduced state-of-the-art operators.

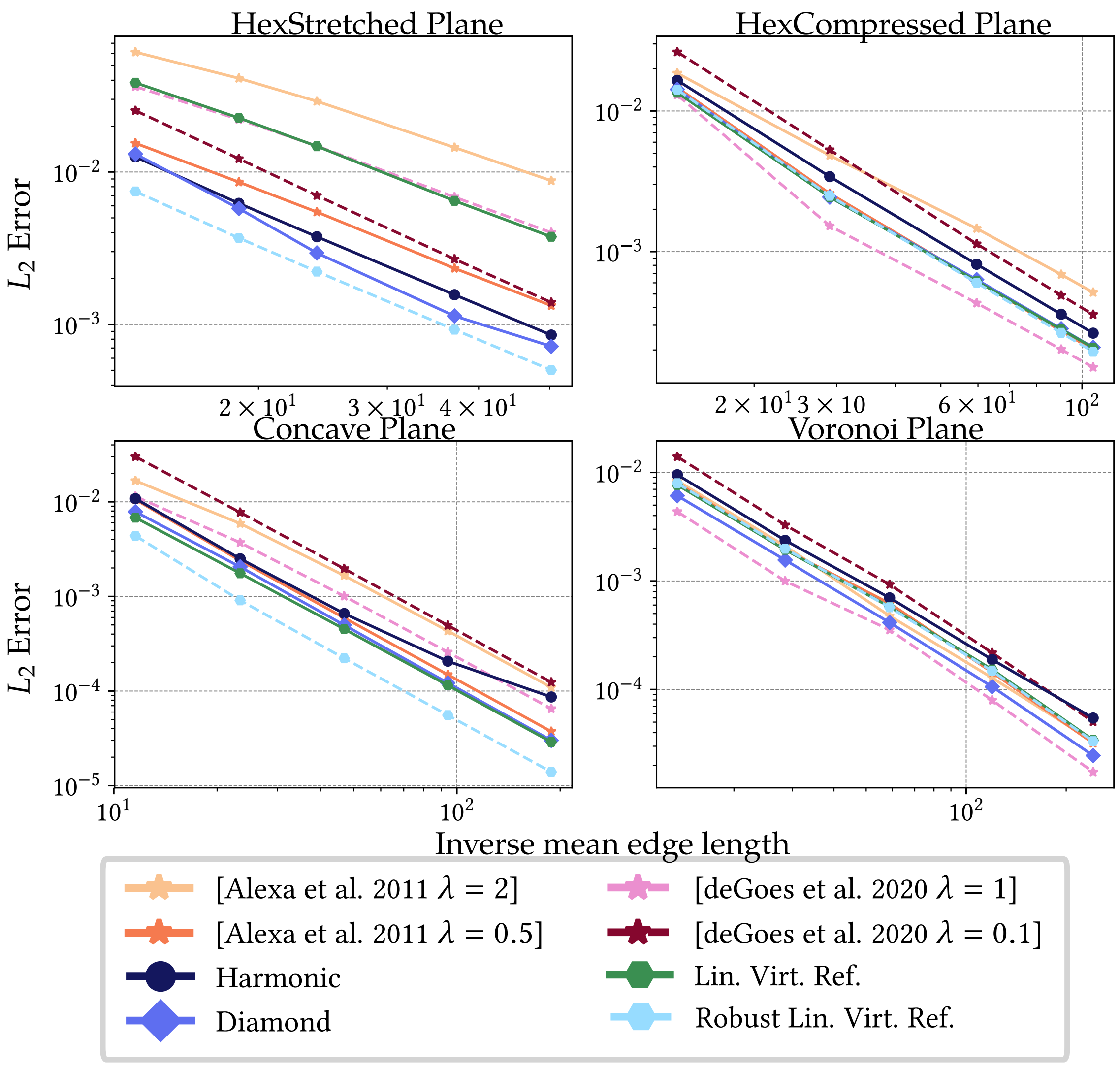


Figure 6.5: *Poisson solve for the 2D Franke test function in log-log scale. The errors were evaluated on different polygon tessellations of the unit plane.*

Poisson Equation We consider the discretized Poisson equation for the Laplacian matrix with Dirichlet boundary conditions. We solve for the discretization of the analytically calculated Laplacian of the 2D Franke test function [Fra79], sampled at the vertices. The convergence plots in Figure 6.5 show the L_2 errors rates on different planar tessellations. Our method consistently maintains low error rates and outperforms all other operators on several meshes. However, the accuracy of more regular shapes, like the Voronoi mesh, does not significantly differ from that of the original method. The operator of de Goes et al. [GBD20] outperforms our method on two tessellations for the hyper-parameter $\lambda = 1$. However, on the other tessellations this parameter leads to less favorable results, while our method remains one of the most accurate.

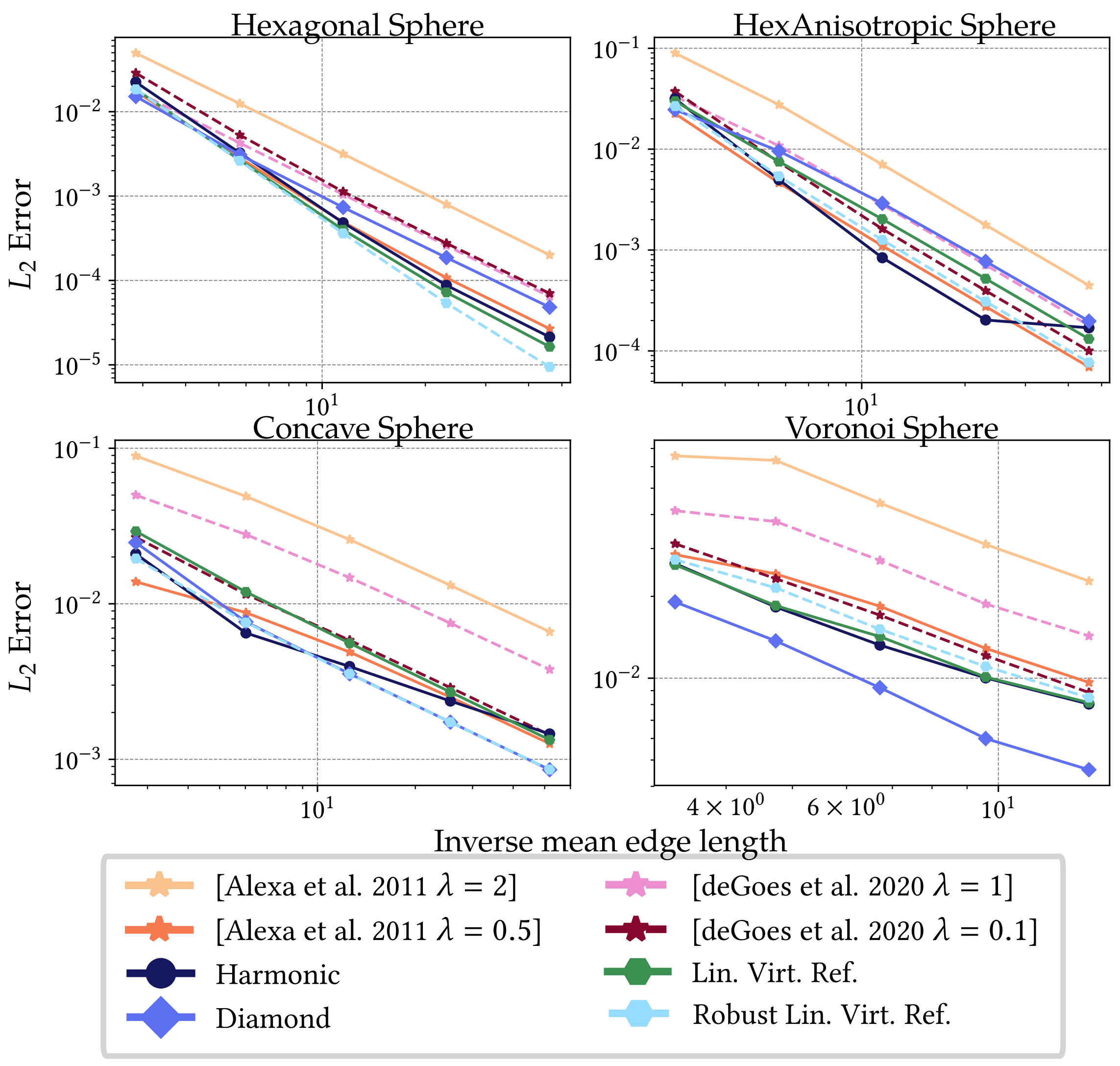


Figure 6.6: *Poisson solve for the spherical harmonic function Y_3^2 in log-log scale on different tessellations of the unit sphere.*

Spherical Harmonics Figure 6.6 shows the obtained results. As for the planar meshes, our operator is able to outperform the other operators on several meshes. In the other cases, it is either on par with the optimized version of Alexa and Wardetzky's operator [AW11] or outperforms the DEC Laplacians. The Diamond operator is able to yield the lowest error on Voronoi spheres.

Geodesics in Heat For this evaluation, we additionally compare the results of the introduced polygon Laplacians to those of a triangle approach. We refine the mesh by triangulating polygons to minimize squared triangle areas

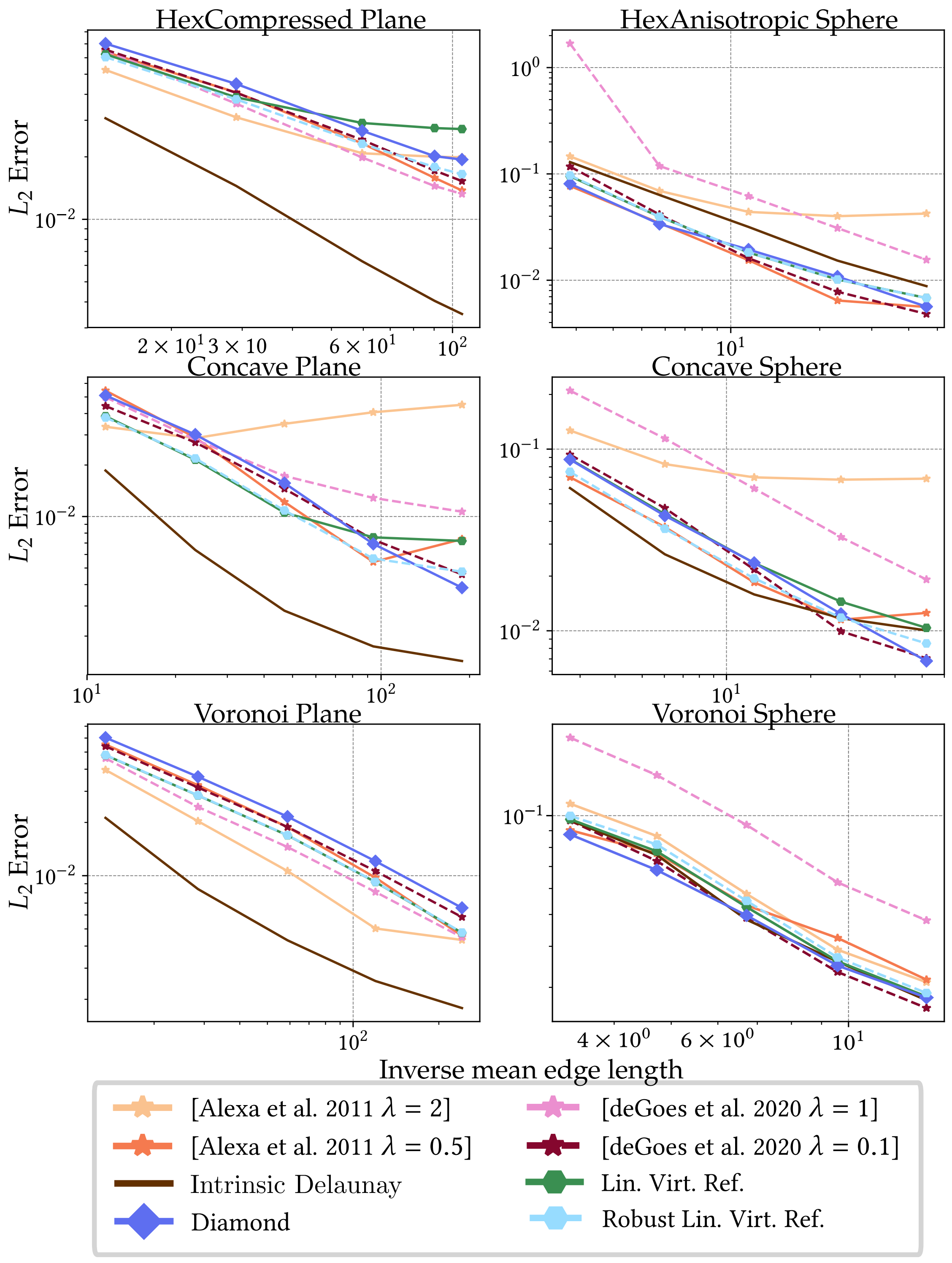


Figure 6.7: *Geodesics in heat method in log-log scale for planes (left) and spheres (right).*

[Lie03] and employ the Laplacian based on the intrinsic Delaunay triangulation [BS07], using the geodesic in heat implementation of libigl [JP+18]. The results are presented in Figure 6.7. Comparing the polygon Laplacians, our method and the two DEC operators with optimized hyper-parameters generally perform best, except in the case of the Voronoi Plane, where the Laplacians obtained with recommended DEC parameters outperform our method. However, these Laplacians had higher errors on other tessellations. The Diamond Laplacian also achieves competitive accuracy, especially for spherical meshes. All polygon approaches are bested by the intrinsic Delaunay approach on planar meshes but remain competitive on spherical tessellations. Comparing our approach to the original linear virtual refinement method, we found that our method consistently improves the accuracy, with only a few exceptions on very ill-conditioned meshes like the Voronoi sphere.

Numerical Stability We evaluate the numerical stability of our proposed polygon Laplacian by analyzing the condition number of its involved components, defined in Equation (6.3). Note that the smallest eigenvalue of the stiffness and Laplacian matrices are zero due to their one-dimensional kernel, so we consider the minimal nonzero eigenvalue. Figure 6.8 displays the condition numbers of the respective stiffness matrices of the different Laplace operators on the previously presented test meshes. Figure 6.9 shows the condition numbers of the strong form of the respective Laplacians, which includes the individual mass matrices. Further examples regarding the condition number of our approach and the original linear virtual refinement method can be found in Figure 6.13. Our approach consistently improves the conditioning of the polygon stiffness matrix compared to the original method. However, we observe that the DEC operators yield lower condition numbers for lower hyper-parameters. An improvement in the conditioning of the *strong form* $(\mathbf{M}^{\pentagon})^{-1}\,\mathbf{S}^{\pentagon}$ of the Laplacian cannot be guaranteed. While often observed, as shown in Figure 6.9, our optimization primarily focuses on angle quality. This can lead to smaller virtual triangles, for example, when the trace minimizer avoids acute angles at the virtual vertex by moving it closer to a polygon edge. These smaller triangle areas affect both the smallest eigenvalue of the stiffness matrix $\mathbf{S}^{\pentagon}$ and the eigenvalues of the mass matrix $\mathbf{M}^{\pentagon}$, which can lead to potentially increased condition numbers of the combined matrix $(\mathbf{M}^{\pentagon})^{-1}\,\mathbf{S}^{\pentagon}$. Since we are using the *lumped* version of the mass matrix, the individual eigenvalues of $\mathbf{M}^{\pentagon}$ are the vertex areas on the diagonal, i.e., the sum of incident faces' areas. The conditioning of the stiffness matrix is closely related to the convergence rate of iterative solvers like conjugate gradients [She02], meaning that well-conditioned matrices should cause a faster convergence. We verify this correlation by solving the Poisson equations mentioned above with the conjugate gradient method and compare the number of iterations until convergence with the condition numbers of the global stiffness matrices.

We observed that when condition numbers improved by 10–50 % with the help of our new method in comparison to the original linear virtual refinement method, the number of iterations decreased by 8–15 %.

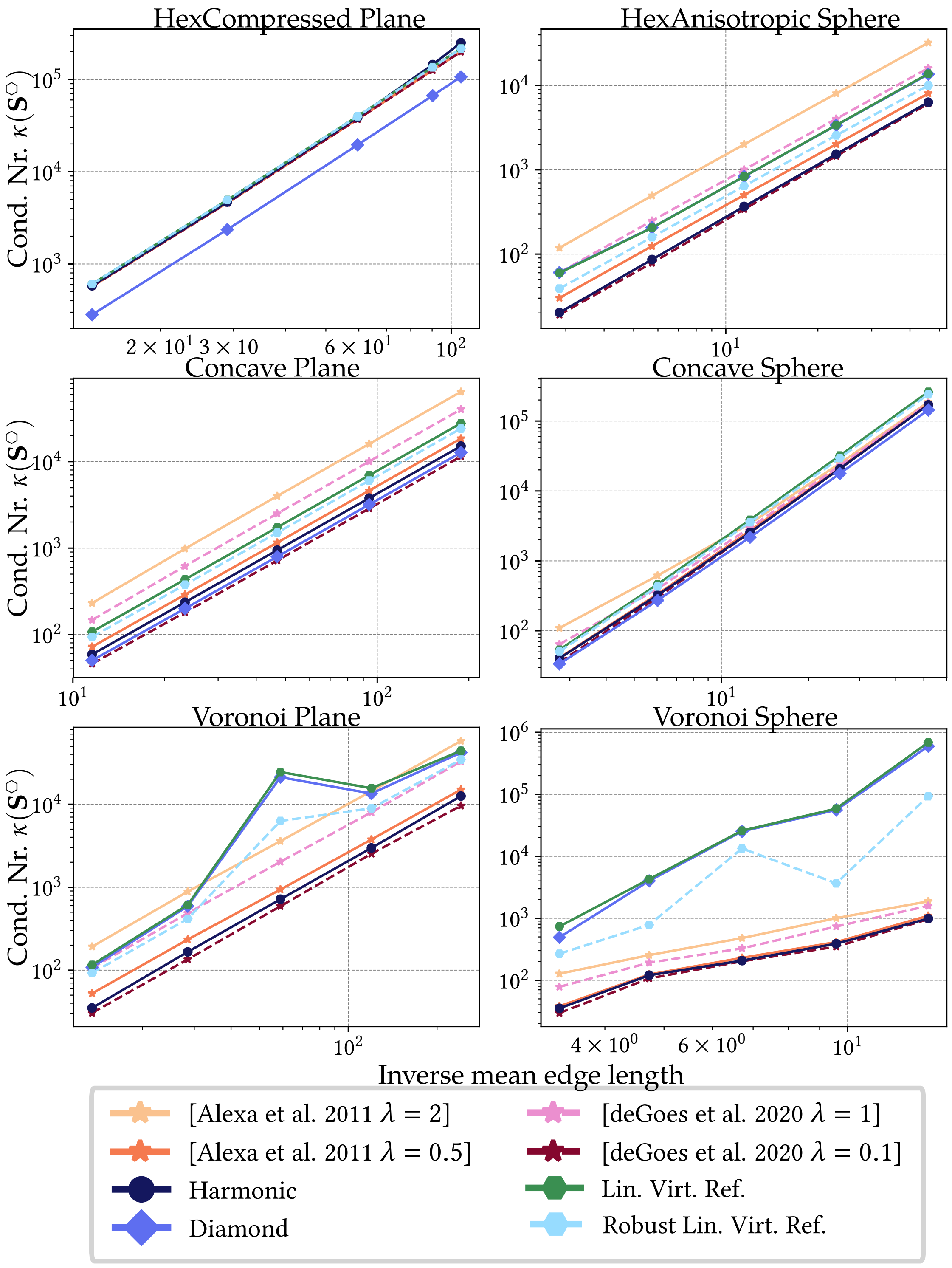


Figure 6.8: *The condition number $\kappa(\mathbf{S}^{\circ})$ of the respective polygon stiffness matrices on planar (left) and spherical (right) polygon meshes.*

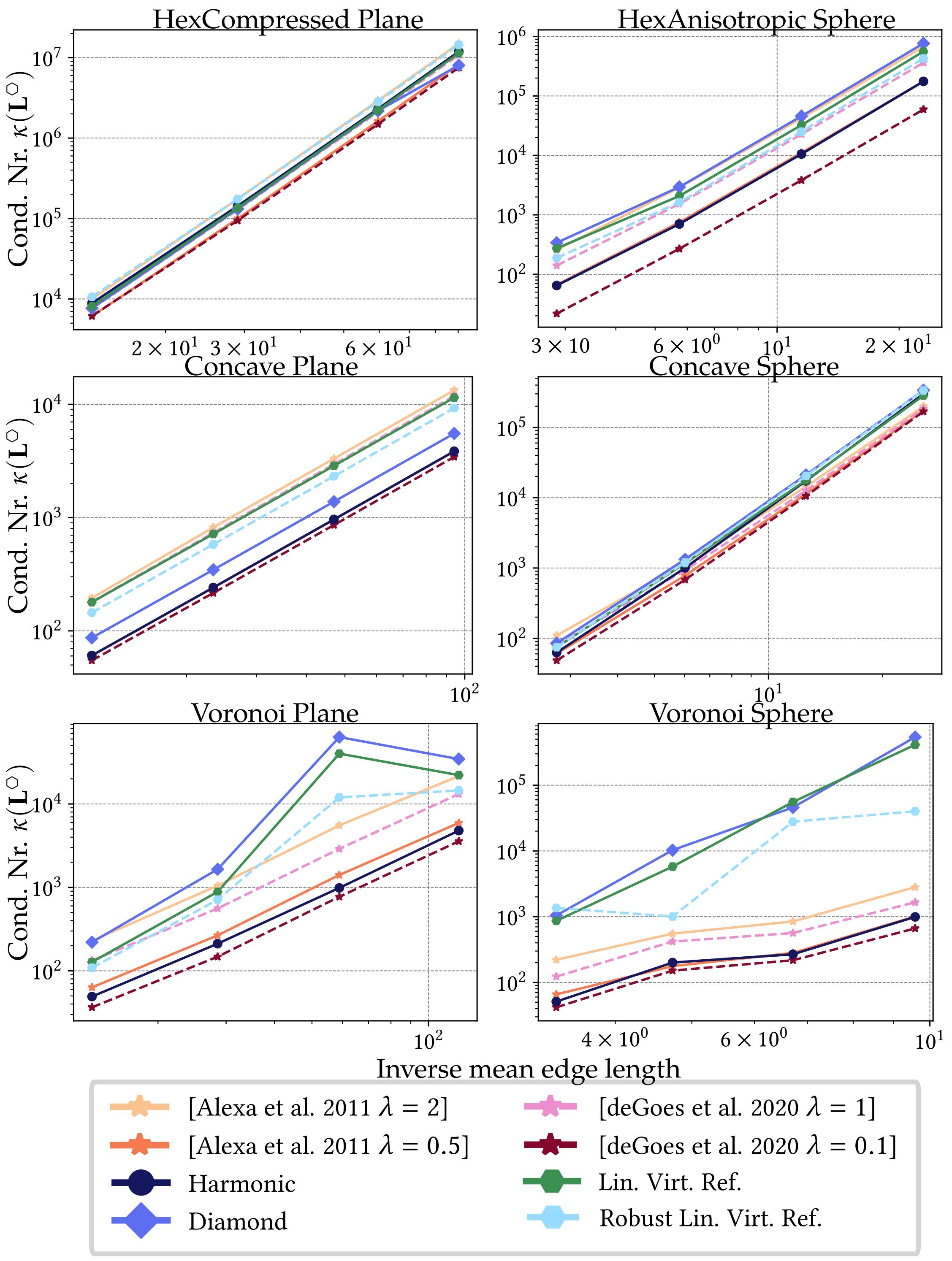


Figure 6.9: *The condition number $\kappa(\mathbf{L}^{\circ})$ of the respective point-wise polygon Laplacians on planar (top) and spherical (bottom) polygon meshes.*

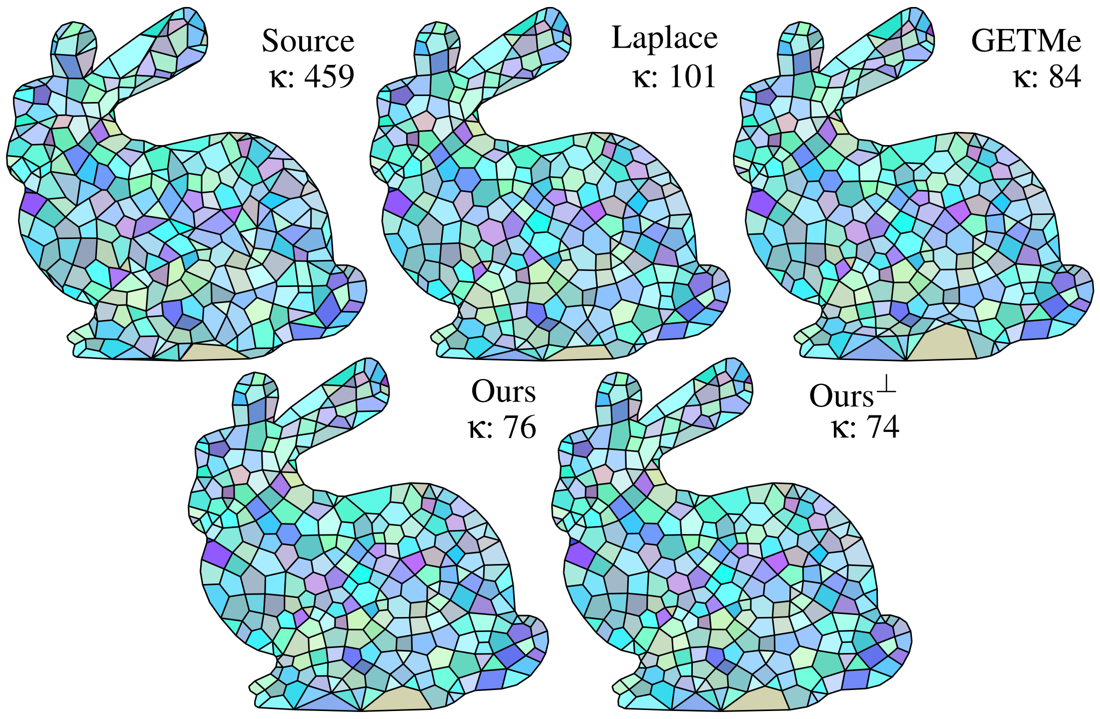


Figure 6.10: *2D smoothing results on a mixed polygon mesh using Smart Laplace [VAG+08], GETMe [VAG+08], and our method. Further included is a version (⊥) where boundary vertices are also allowed to move along the boundary (not possible with GETMe).*

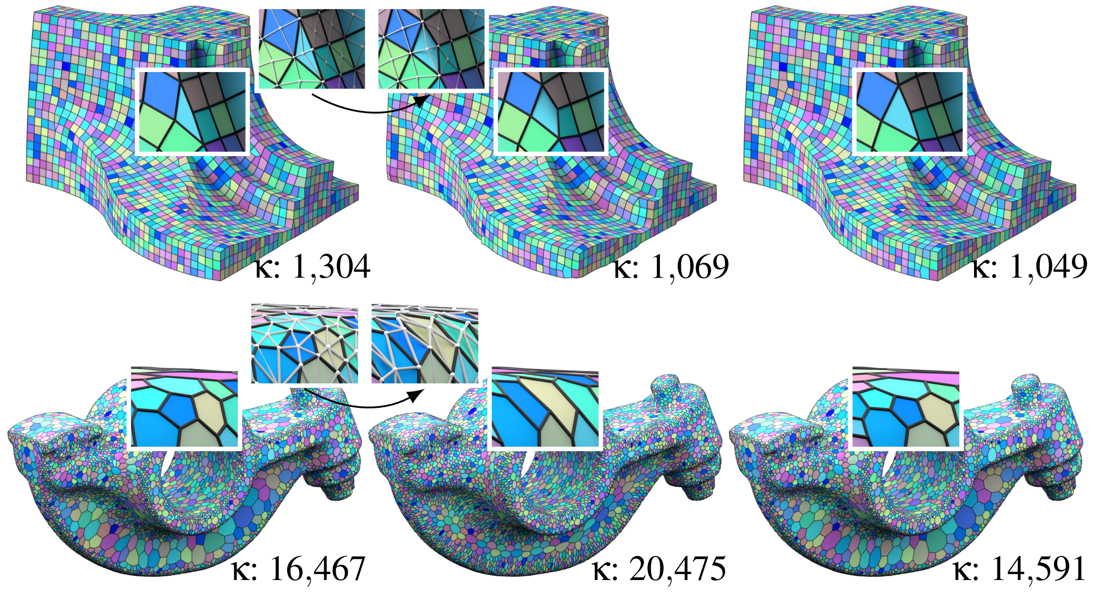


Figure 6.11: *The polygonal input mesh on the left and our smoothed result on the right. Applying tangential smoothing [DVB+13] on the virtual triangulation may lead to bent polygons (center).*

6.5.2 Polygon Mesh Smoothing

Figure 6.10 illustrates a comparison of smoothing results on a mixed polygon mesh generated with Smart Laplacian, GETMe, and our method. Smart Laplacian and GETMe are both by Vartziotis et al. [VAG+08], with GETMe being considered as a state-of-the-art method in mesh smoothing [Lo14; SBM+23]. However, the official implementation currently only supports 2D input. Further, the boundary vertices are fixed during the optimization. Our approach easily allows for the integration of boundary adherence conditions, such that these vertices are also allowed to move. For the optimization, this additional degree of freedom can further improve the polygon constellations, shown by the condition numbers of our polygon stiffness matrix $\kappa(\mathbf{S}^{\bigcirc})$, which we refer to as κ.

In Figure 6.12 we compare our methods to simple explicit Laplacian smoothing and the methods of Knupp [Knu00] and Garimella et al. [GS04]. While all of the other methods were also intended for polygons, the energy term for a respective vertex only incorporates its direct one-ring neighbors and neglects the rest of the polygon (see Figure 6.2). Especially on challenging inputs, featuring non-convex polygons, this often breaks the optimization for these methods. Therefore, the results for Knupp and Garimella had to be generated with a preceding step to repair non-convex faces.

A simpler approach on smoothing could be to apply established triangle smoothing algorithms on the virtual triangulation and unrefine the mesh afterwards. However, common techniques for triangles, such as the tangential smoothing approach by Dunyach et al. [DVB+13], can result in scenarios where vertices slide over sharp edges or high curvature features, as shown in Figure 6.11 (center). While the feature is preserved in the smoothed triangle mesh, unrefining it (removing the inserted vertices) leaves us with bent non-planar polygons. Increased non-planarity and ill-shaped polygons may also negatively impact the mesh conditioning, as shown with the condition numbers given in the bottom row of Figure 6.11.

In Figure 6.13 and 6.14, we present examples of our procedure's input-output pairs, complemented by bar plots that offer comparisons with alternative smoothing strategies. These comparisons encompass both the outcomes of the optimization processes and the condition numbers of the stiffness matrices of the original linear virtual refinement method and our improved version. Notably, our mesh improvement strategy is meticulously tailored to complement our enhanced Laplace operator, resulting in substantial improvements when applied together. In Table 6.1, we show that in addition to improved condition numbers of the weak and strong form of the polygon Laplacian, the accuracy of Poisson solves on planar and spherical meshes is enhanced on the optimized mesh compared to the original tessellation.

Level		$\kappa(\mathbf{S}^{\bigcirc})$	$\kappa(\mathbf{L}^{\bigcirc})$	L_2 error	Type
Voronoi 2	a	10 312	12 068	0.002 072 91	Plane
	b	2084	2432	0.002 134 34	
	c	226	358	0.002 019 59	
	a	4284	5776	0.020 494 60	Sphere
	b	782	1006	0.022 109 20	
	c	187	323	0.015 656 10	
Voronoi 3	a	43 091	53 099	0.000 584 99	Plane
	b	13 893	18 576	0.000 580 63	
	c	898	1806	0.000 549 82	
	a	25 877	55 648	0.015 578 80	Sphere
	b	13 372	27 876	0.017 322 00	
	c	317	350	0.007 819 03	

Table 6.1: *We analyze condition numbers and accuracy on two Voronoi planes and spheres (see Figure A.1), respectively. Numbers are given on the unaltered mesh for the (a) linear virtual refinement method, (b) our new Laplacian and (c) our Laplacian on the optimized tessellation. Our optimization yields improvements in both condition numbers and accuracy.*

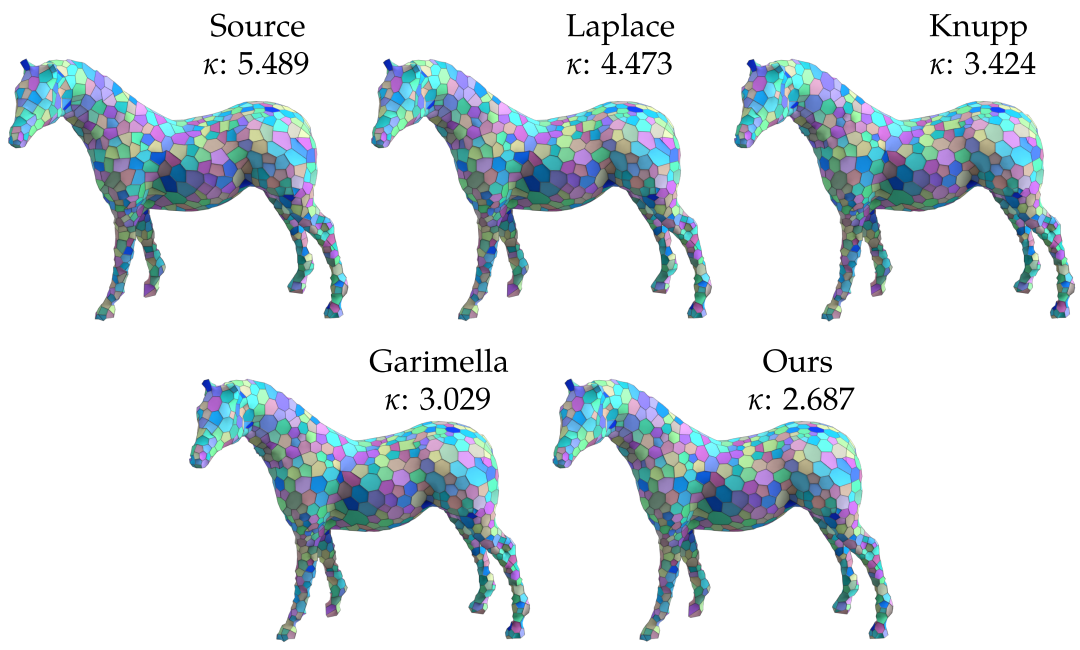
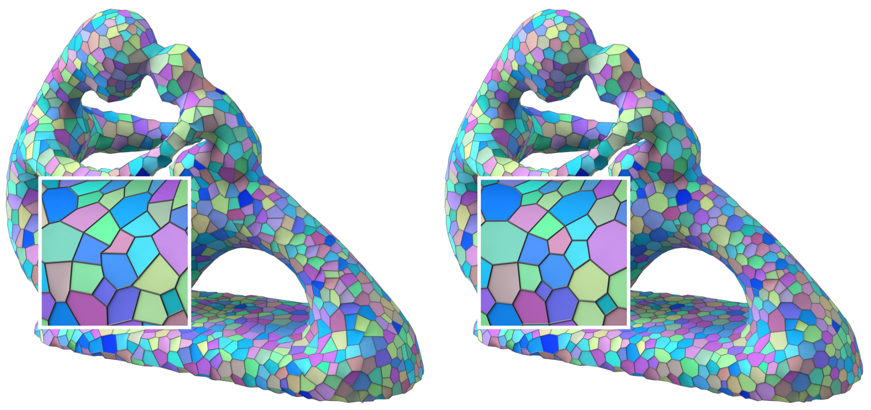
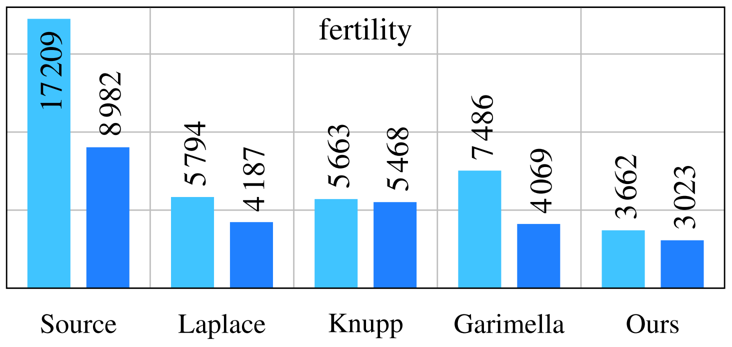
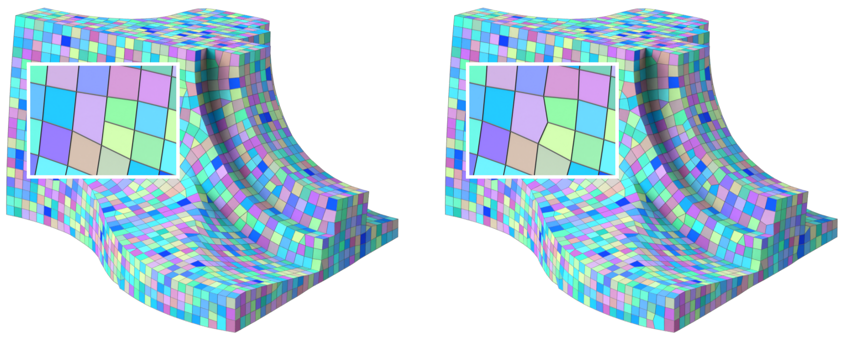


Figure 6.12: *2D manifold smoothing results on a mixed polygon mesh using Laplace smoothing, Knupp's [Knu00], Garimella's [GS04], and our method; all using surface Quadrics, respectively. GETMe is not included as the official implementation only supports 2D input.*







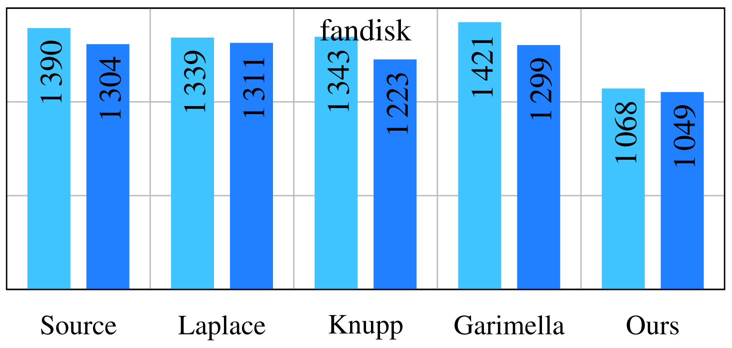


Figure 6.13: *Sources are shown on the left and our smoothed results on the right. The fertility mesh consists of mixed polygons and the fandisk is a quad-dominant mesh. The plots show condition numbers κ on results of different smoothing strategies for the Laplace operator obtained with the initial virtual linear refinement method* 🟦 *and with our improvements* 🟦.

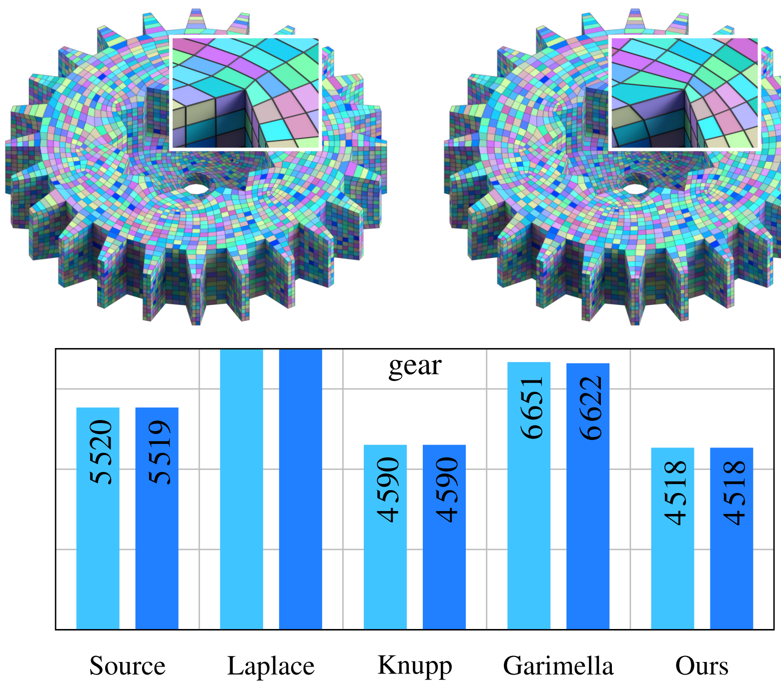


Figure 6.14: *The source of the gear mesh is shown on the left and our smoothed results on the right. The gear wheel is a random pick from the quad-mesh dataset of Pietroni et al. [PNA+21].The plot on the right shows condition numbers κ on results of different smoothing strategies for the Laplace operator obtained with the initial virtual linear refinement method* 🟦 *and with our improvements* 🟦.

6.5.3 Timings

With our current implementation we report the following performance: As elaborated in Section 6.3.3, the optimal position for our proposed virtual vertex is determined via a polygon projection and optimizing for a trace minimizing point. Compared to the original method, which solves for a squared area minimizer and least norm weights, our matrix assembly is on average about 10 % slower. Note however, that assembly costs are negligible (in the range of milliseconds) compared to solving times (several seconds), which stay the same. The time for the smoothing iteration strongly depends on the conditioning and size of the given input mesh. With our single-core implementation (using TinyAD [SBB+22]) the input meshes used in our tests, like Voronoi spheres, the horse, fertility or fandisk terminate with less than 50 iterations in 2 to 10 seconds. For the rocker-arm (Figure 6.11, bottom), 709 iterations finished in 1400 seconds. However, this can be improved by either adjusting the termination criterion, as the most improvement is achieved within the first few steps, or a dedicated parallelized implementation. Additionally, not relying on TinyAD and using the analytic derivations instead will also lower the computation time.

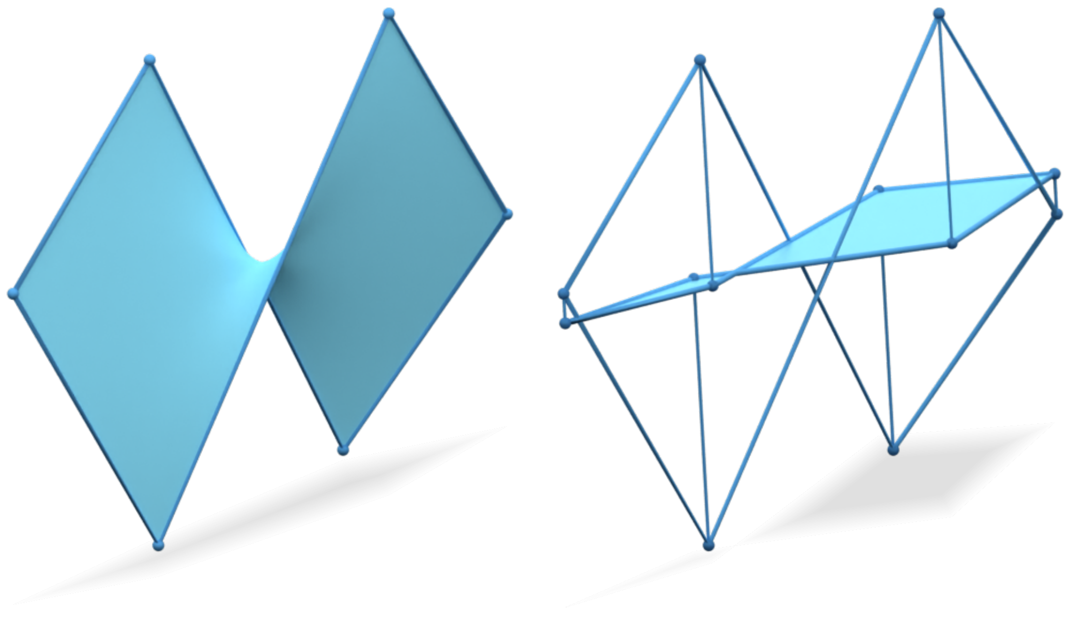


Figure 6.15: *Strongly disfigured polygons, as the saddle shape on the left, may induce a max-area projection (right) that is self-intersecting, thus non-star-shaped.*

6.6 LIMITATIONS

While our method can handle non-planar and non-convex polygons and generally improves the stiffness matrix's accuracy and condition number, there are some limitations. First, the shape of the polygons is limited to those that result in star-shaped faces after the planar projection to avoid negative triangle areas. Challenging configurations are extremely disfigured saddle-shaped polygons that, when projected, result in tangled and self-intersecting shapes. This is, however, more of a theoretical issue as we have yet to find such shapes in the wild, and the example in Figure 6.15 is manually constructed. Secondly, negative discrete harmonic coordinates can cause negative mass matrix entries on ill-conditioned meshes. Notably, such occurrences are infrequent; we observed them only on a single face. While relatively large, this particular polygon contained an almost degenerating edge. The negative weight associated with the virtual vertex overpowered the area of the existing node, resulting in a negative entry despite the absence of flipped triangles. The mass matrix, and consequently the strong form of the Laplacian, are then no longer positive/negative (semi-)definite.

6.7 CONCLUSION

In this chapter, we optimized the numerical quality and accuracy of the polygon Laplacian based on the linear virtual refinement method prestend in Chapter 3. We showed that minimizing the trace of the polygon stiffness matrix establishes a direct link to the geometry of the polygon through the virtual triangle fan. This connection allows us to leverage existing knowledge of the finite element method regarding triangle shapes to find an optimized placement of the virtual vertex. Based on these insights, we presented a smoothing algorithm that can further improve polygon meshes with regard to the trace of the polygon stiffness matrix. The combination of techniques offers a valuable tool set for improving the overall performance of numerical simulations on complex surface geometries.
The experiments showed mostly consistent improvement in both accuracy and condition numbers for the virtual refinement method, with only some exceptions (see table 6.1). Compared to other existing operators, our methods offers an approach that consistently yields good results, whereas the other methods require tuning the available parameter to adapt to the task at hand.

7 SUMMARY

As commonly acknowledged, the classical cotangent discretization of the Laplacian is widely used and a prominent method to handle various algorithms involved in geometry processing. However, the recognized needs of artists and scientists for modeling complex shapes and tessellations demonstrate that this is simply not enough. More flexible approaches are required that extend the discrete Laplacian operator beyond triangle and tetrahedral meshes to meet the evolving demands of geometric modeling and engineering applications. Furthermore, while facing the mathematical difficulties these extensions entail, we still have to find practical solutions accessible to a broader audience.

This thesis addressed these challenges by introducing the *linear virtual refinement method* (Chapter 3). It offers a straightforward yet effective discretization of the Laplace operator for polygonal and polyhedral meshes. The fundamental building block of this method, the virtual refinement based on the Galerkin method, proved to be highly adaptable and allowed the definition of even higher-quality polygon Laplacians across various numerical schemes. For example, the Diamond Laplacian (Chapter 4) based on the DDFV method trades a denser sparsity pattern for more accurate results, even on standard triangle and tetrahedral meshes.

In Chapter 5, we demonstrated that the linear virtual refinement method can be extended to higher-order discretizations. We defined variational quadratic shape functions for arbitrary polygons and polyhedra while retaining beneficial properties such as faster convergence, higher accuracy, and quadratic precision. Using these shape functions was made affordable through our custom multigrid approach, successfully mitigating the expensive solving times of the volumetric case.

Furthermore, Chapter 6 demonstrates that the full potential of the linear virtual refinement method, and consequently other discretizations, can be even further explored. We enhanced the original method by focusing on the degrees of freedom involved and were even able to find a correlation between the shape of the polygons and our Laplacian's performance.

In short, we do not have to limit ourselves to shapes like triangles and tetrahedra. If we further explore arbitrary polygons and the mathematics behind them, we can establish the flexibility and freedom to use whatever shapes a user might desire. We hope this thesis made a meaningful step toward that goal.

7.1 RECOMMENDATIONS

Considering the exhaustive evaluation on different polygon Lapalcians presented in this thesis, we want to summarize the obtained results and give the reader a recommendation in which situation which operator should be used.
In the linear setting, given their overall performance, the DEC Laplacians presented by Alexa and Wardetzky [AW11] and de Goes et al. [GBD20] lead to favorable numerical results if the user is willing to adjust the stabilization parameter for each individual application.
The presented Diamond Laplacian will be the better choice if the reader is looking for a method that works both on surface and volume meshes and leads to accurate results without any adjusting. However, given its denser matrix pattern, this approach leads to longer solving times.
If this is a problem, the operator obtained with the linear virtual refinement method would be a computationally more efficient choice. It works on both surface and volume meshes and, given that many applications already work with the cotangent Laplacian, can be easily integrated since the only missing piece is the prolongation matrix. Initially, it provides slightly less accurate results than the Diamond Laplacian. However, using the optimized weights and placement of the virtual vertex proposed in Chapter 6 further enhances its accuracy and numerical stability. These enhancements lead to an operator that can often surpass all of the other methods. However, currently, these optimized parameters are only provided for surface meshes.
Should the reader's primary goals revolve around achieving high accuracy and faster convergence, with less regard for the computational complexities and potential numerical stability of the underlying system to be solved, then forsaking the linear setting entirely in favor of employing higher-order shape functions as defined in Chapter 5 would be the best choice.
The harmonic shape functions are not competitive compared to the other methods due to their costly construction process. However, they are able to reproduce P1 and Q1 elements on triangles and quads, are C^0 continuous to P1/Q1 at the boundaries of polygons and polyhedra, and can therefore be seamlessly mixed with these standard elements.

7.2 OUTLOOK

Lastly, we would like to conclude this thesis with some promising directions for future work. A possible avenue would be to consider even higher-order extensions of the linear and quadratic virtual refinement method, with higher derivative continuity constraints across elements – enabling thin shell simulations on polygonal meshes. We would also like to extend our approach to constructing 1-form bases, supporting the large body of work on vector field processing. Furthermore, besides the Laplacian, there exist a variety of other discrete differential operators whose extensions could be helpful in the graphics community, as touched upon by de Goes et al. [GBD20], and Lipnikov et al. [LMS14] in their survey on the MFD method. Based on Chapter 6, further research could involve extending the numerical analysis of the linear virtual refinement method to volume meshes and examining whether similar connections can be established. Additionally, expanding our analysis to other Laplacians, such as the Diamond Laplacian or the Discrete Exterior Calculus methods [AW11; GBD20], could be promising. Regarding the linear virtual refinement method itself, it is currently still limited to polygons that can be reasonably refined by introducing a single virtual vertex. The possibility of introducing multiple virtual vertices within a single face could lead to improved triangulations of non-convex polygons, which would further enhance the quality and flexibility of our method.
In conclusion, while this thesis addressed many aspects of polygon and polyhedral Laplacians, some unopened doors for research opportunities remain. Exploring them could further improve our community's range of tools and possibilities to work on arbitrary tesselations, giving polygons the attention they deserve.

BIBLIOGRAPHY

[ABB+21] Marco Attene, Silvia Biasotti, Silvia Bertoluzza, Daniela Cabiddu, Marco Livesu, Giuseppe Patanè, Micol Pennacchio, Daniele Prada, and Michela Spagnuolo. "Benchmarking the Geometrical Robustness of a Virtual Element Poisson Solver". *Mathematics and Computers in Simulation* 190 (2021), pp. 1392–1414.

[ABH+12] B. Andreianov, M. Bendahmane, F. Hubert, and S. Krell. "On 3D DDFV Discretization of Gradient and Divergence Operators. I. Meshing, Operators and Discrete Duality". *IMA Journal of Numerical Analysis* 32.4 (2012), pp. 1574–1603.

[ABH13] Boris Andreianov, Mostafa Bendahmane, and Florence Hubert. "On 3D DDFV Discretization of Gradient and Divergence Operators. II. Discrete Functional Analysis Tools and Applications To Degenerate Parabolic Problems." *Computational Methods in Applied Mathematics* 13.4 (2013), pp. 369–410.

[AFW06] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. "Finite Element Exterior Calculus, Homological Techniques, and Applications". *Acta Numerica* 15 (2006), pp. 1–155.

[AHK+20] Marc Alexa, Philipp Herholz, Maximilian Kohlbrenner, and Olga Sorkine-Hornung. "Properties of Laplace Operators for Tetrahedral Meshes". *Computer Graphics Forum* 39.5 (2020).

[AHQ23] Boris Andreianov and El Houssaine Quenjel. "Nodal Discrete Duality Numerical Scheme for Nonlinear Diffusion Problems on General Meshes". *IMA Journal of Numerical Analysis* 44.3 (June 2023), pp. 1597–1643.

[Ale19] Marc Alexa. "Harmonic Triangulations". *ACM Transactions on Graphics* 38.4 (2019), 54:1–54:14.

[Ale20] Marc Alexa. "Conforming Weighted Delaunay Triangulations". *ACM Transactions on Graphics* 39.6 (2020).

[ARR+19] Prusty Aurojyoti, Piska Raghu, Amirtham Rajagopal, and Jn Reddy. "An n-Sided Polygonal Finite Element for Nonlocal Nonlinear Analysis of Plates and Laminates". *International Journal for Numerical Methods in Engineering* 120 (2019), pp. 1071–1107.

[Aur87] Franz Aurenhammer. "A Criterion for the Affine Equivalence of Cell Complexes In $\mathbb{R}^d$ and Convex Polyhedra In $\mathbb{R}^{d+1}$". *Discrete & Computational Geometry* 2.1 (1987), pp. 49–64.

[AW11] Marc Alexa and Max Wardetzky. "Discrete Laplacians on General Polygonal Meshes". *ACM Transactions on Graphics* 30.4 (2011), 102:1–102:10.

[BB23] Astrid Bunge and Mario Botsch. "A Survey on Discrete Laplacians for General Polygonal Meshes". *Computer Graphics Forum* 42.2 (2023), pp. 521–544.

[BHK+20] Astrid Bunge, Philipp Herholz, Misha Kazhdan, and Mario Botsch. "Polygon Laplacian Made Simple". *Computer Graphics Forum* 39.2 (2020), pp. 303–313.

[Bis09] J.E. Bishop. "Simulating the Pervasive Fracture of Materials and Structures Using Randomly Close Packed Voronoi Tessellations". *Computational Mechanics* 44.4 (2009), pp. 455–471.

[Bis14] J.E. Bishop. "A Displacement-Based Finite Element Formulation for General Polyhedra Using Harmonic Shape Functions". *International Journal for Numerical Methods in Engineering* 97 (2014), pp. 1–31.

[BLS05] Franco Brezzi, Konstantin Lipnikov, and Valeria Simoncini. "A Family of Mimetic Finite Difference Methods on Polygonal and Polyhedral Meshes". *Mathematical Models and Methods in Applied Sciences* 15.10 (2005), pp. 1533–1551.

[BLS+07] Franco Brezzi, Konstantin Lipnikov, Mikhail Shashkov, and Valeria Simoncini. "A new Discretization Methodology for Diffusion Problems on Generalized Polyhedral Meshes". *Computer Methods in Applied Mechanics and Engineering* 196.37 (2007), pp. 3682–3692.

[BS07] Alexander I. Bobenko and Boris A. Springborn. "A Discrete Laplace–Beltrami Operator for Simplicial Surfaces". *Discrete & Computational Geometry* 38.4 (2007), pp. 740–756.

[BSP+06] Mario Botsch, Robert Sumner, Mark Pauly, and Markus Gross. "Deformation Transfer for Detail-Preserving Surface Editing". In *Proc. of Vision, Modeling and Visualization*. 2006, pp. 357–364.

[BSW08] Mikhail Belkin, Jian Sun, and Yusu Wang. "Discrete Laplace Operator on Meshed Surfaces". In *Proceedings of Symposium on Computational Geometry*. 2008, pp. 278–287.

[CDH+08] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. "Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate". *ACM Transactions on Mathematical Software* 35.3 (2008), pp. 1–14.

[CDR00] Ulrich Clarenz, Udo Diewald, and Martin Rumpf. "Anisotropic geometric diffusion in surface processing". In *Proceedings of IEEE Visualization*. 2000, pp. 397–405.

[CG16] Renjie Chen and Craig Gotsman. "On Pseudo-Harmonic Barycentric Coordinates". *Computer Aided Geometric Design* 44 (2016), pp. 15–35.

[CH11] Yves Coudière and Florence Hubert. "A 3D Discrete Duality Finite Volume Method for Nonlinear Elliptic Equations". *SIAM Journal on Scientific Computing* 33.4 (2011), pp. 1739–1764.

[CLB+09] Ming Chuang, Linjie Luo, Benedict J. Brown, Szymon Rusinkiewicz, and Michael Kazhdan. "Estimating the Laplace-Beltrami Operator by Restricting 3D Functions". *Computer Graphics Forum* 28.5 (2009), pp. 1475–1484.

[Cra18] Keenan Crane. *Triangle Mesh Derivatives Cheat Sheet*. https://www.cs.cmu.edu/~kmcrane/Projects/OtherTriangleMeshDerivativesCheatSheet.pdf. 2018.

[Cra19] Keenan Crane. *The n-dimensional Cotangent Formula*. https://www.cs.cmu.edu/~kmcrane/Projects/Other/nDCotanFormula.pdf. 2019.

[CRK16] Ming Chuang, Szymon Rusinkiewicz, and Misha Kazhdan. "Gradient-Domain Processing of Meshes". *Journal of Computer Graphics Techniques* 5.4 (2016), pp. 44–55.

[CWW13] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. "Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow". *ACM Transactions on Graphics* 32.5 (2013), 152:1–152:11.

[CXG+10] Renjie Chen, Yin Xu, Craig Gotsman, and Ligang Liu. "A Spectral Characterization of the Delaunay Triangulation". *Computer Aided Geometric Design* 27.4 (2010), pp. 295–300.

[DMA02] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. "Intrinsic Parameterizations of Surface Meshes". *Computer Graphics Forum* 21.3 (2002), pp. 209–218.

[DMS+99] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. "Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow". In *Proceedings of ACM SIGGRAPH*. 1999, pp. 317–324.

[DO05] K. Domelevo and P. Omnés. "A Finite Volume Method for the Laplace Equation on Almost Arbitrary Two-Dimensional Grids". *Mathematical Modelling and Numerical Analysis (M2AN)* 39.6 (2005), pp. 1203–1249.

[Dro14] Jerome Droniou. “Finite Volume Schemes for Diffusion Equations: Introduction to and Review of Modern Methods”. *Mathematical Models and Methods in Applied Sciences* 24.08 (2014), pp. 1575–1619.

[DSS+23] Ana Dodik, Oded Stein, Vincent Sitzmann, and Justin Solomon. “Variational Barycentric Coordinates”. *ACM Transactions on Graphics* 42.6 (2023), 255:1–255:16.

[Dus55] G. M. Dusinberre. “Heat Transfer Calculations by Numerical Methods”. *Journal of the American Society for Naval Engineers* 67.4 (1955), pp. 991–1002.

[Dus61] G.M. Dusinberre. “Heat-transfer Calculations by Finite Differences”. International textbooks in mechanical engineering. International Textbook Company, 1961.

[DVB+13] Marion Dunyach, David Vanderhaeghe, Loïc Barthe, and Mario Botsch. “Adaptive Remeshing for Real-Time Mesh Deformation”. In *Proceedings of Eurographics Short Papers*. 2013.

[Dzi88] Gerhard Dziuk. “Finite Elements for the Beltrami Operator on Arbitrary Surfaces”. In *Partial Differential Equations and Calculus of Variations*. Ed. by Stefan Hildebrandt and Rolf Leis. Springer Berlin Heidelberg, 1988, pp. 142–155. ISBN: 978-3-540-46024-4.

[EDD+95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. “Multiresolution Analysis of Arbitrary Meshes”. In *Proceedings of ACM SIGGRAPH*. 1995, pp. 173–182.

[Fle84] C.A.J. Fletcher. “Computational Galerkin Methods”. Computational Physics Series. Springer-Verlag, 1984.

[Flo03] Michael S. Floater. “Mean Value Coordinates”. *Computer Aided Geometric Design* 20.1 (2003), pp. 19–27.

[Flo15] Michael S. Floater. “Generalized Barycentric Coordinates and Applications”. *Acta Numerica* 24 (2015), pp. 161–214.

[Flo97] Michael S. Floater. “Parametrization and Smooth Approximation of Surface Triangulations”. *Computer Aided Geometric Design* 14.3 (1997), pp. 231–250.

[Fra79] Richard Franke. *A Critical Comparison of Some Methods for Interpolation of Scattered Data*. Tech. rep. Naval Postgraduate School, 1979.

[GBD20] Fernando de Goes, Andrew Butts, and Mathieu Desbrun. “Discrete Differential Operators on Polygonal Meshes”. *ACM Transactions on Graphics* 39.4 (2020), 110:1–110:14.

[GDM+16] Fernando de Goes, Mathieu Desbrun, Mark Meyer, and Tony DeRose. "Subdivision Exterior Calculus for Geometry Processing". *ACM Transactions on Graphics* 35.4 (2016), 133:1–133:11.

[GGT06] Steven J. Gortler, Craig Gotsman, and Dylan Thurston. "Discrete One-forms on Meshes and Applications to 3D Mesh Parameterization". *Computer Aided Geometric Design* 23.2 (2006), pp. 83–112.

[GH97] Michael Garland and Paul S Heckbert. "Surface Simplification Using Quadric Error Metrics". In *Proceedings of ACM SIGGRAPH*. 1997, pp. 209–216.

[GJ+10] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[GR17] Andrew Gillette and Alexander Rand. "Shape Quality for Generalized Barycentric Interpolation". In *Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics*. CRC Press, 2017, pp. 23–42.

[GRB16] Andrew Gillette, Alexander Rand, and Chandrajit Bajaj. "Construction of Scalar and Vector Finite Element Families on Polygonal and Polyhedral Meshes". *Computational Methods in Applied Mathematics* 16.4 (2016), pp. 667–683.

[GS04] Rao V Garimella and Mikhail J Shashkov. "Polygonal Surface Mesh Optimization". *Engineering with Computers* 20 (2004), pp. 265–272.

[GSK02] Rao V Garimella, Mikhail J Shashkov, and Patrick M Knupp. "Optimization of Surface Mesh Quality Using Local Parametrization." In *Proceedings of the International Meshing Roundtable*. 2002, pp. 41–52.

[GSK04] Rao V Garimella, Mikhail J Shashkov, and Patrick M Knupp. "Triangular and Quadrilateral Surface Mesh Quality Optimization Using Local Parametrization". *Computer Methods in Applied Mechanics and Engineering* 193.9–11 (2004), pp. 913–928.

[GVL96] Gene H. Golub and Charles F. Van Loan. "Matrix Computations". Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996. ISBN: 9780801854149.

[Her00] Francois Hermeline. "A Finite Volume Method for the Approximation of Diffusion Operators on Distorted Meshes". *Journal of Computational Physics* 160.2 (2000), pp. 481–499.

[Her09] Francois Hermeline. "A Finite Volume Method for Approximating 3D Diffusion Operators on General Meshes". *Journal of Computational Physics* 228.16 (2009), pp. 5763–5786.

[HF06] Kai Hormann and Michael S. Floater. “Mean Value Coordinates for Arbitrary Planar Polygons”. *ACM Transactions on Graphics* 25.4 (2006), pp. 1424–1441.

[Hir03] Anil Nirmal Hirani. “Discrete Exterior Calculus”. PhD thesis. California Institute of Technology, 2003.

[HKA15] Philipp Herholz, Jan Eric Kyprianidis, and Marc Alexa. “Perfect Laplacians for Polygon Meshes”. *Computer Graphics Forum* 34.5 (2015), pp. 211–218.

[HPW06] Klaus Hildebrandt, Konrad Polthier, and Max Wardetzky. “On the Convergence of Metric and Geometric Properties of Polyhedral Surfaces”. *Geometriae Dedicata* 123 (2006), pp. 89–112.

[HS08] K. Hormann and N. Sukumar. “Maximum Entropy Coordinates for Arbitrary Polytopes”. *Computer Graphics Forum* 27.5 (2008), pp. 1513–1520.

[HS17] Kai Hormann and N. Sukumar. “Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics”. Taylor & Francis, 2017.

[Hug12] T.J.R. Hughes. “The Finite Element Method: Linear Static and Dynamic Finite Element Analysis”. Dover Civil and Mechanical Engineering. Dover Publications, 2012. ISBN: 9780486135021.

[Hwa04] Suk-Geun Hwang. “Cauchy’s Interlace Theorem for Eigenvalues of Hermitian Matrices”. *The American Mathematical Monthly* 111.2 (2004), pp. 157–159.

[JMD+07] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. “Harmonic Coordinates for Character Articulation”. *ACM Transactions on Graphics* 26.3 (2007), pp. 71–81.

[JP+18] Alec Jacobson, Daniele Panozzo, et al. *libigl: A simple C++ Geometry Processing Library*. https://libigl.github.io/. 2018.

[JSW05] Tao Ju, Scott Schaefer, and Joe Warren. “Mean Value Coordinates for Closed Triangular Meshes”. *ACM Transactions on Graphics* 24.3 (2005), pp. 561–566.

[JWW+14] Caigui Jiang, Jun Wang, Johannes Wallner, and Helmut Pottmann. “Freeform Honeycomb Structures”. *Computer Graphics Forum* 33.5 (2014), pp. 185–194.

[KMS02] Patrick Knupp, Len G Margolin, and Mikhail Shashkov. “Reference Jacobian Optimization-Based Rezone Strategies for Arbitrary Lagrangian Eulerian Methods”. *Journal of Computational Physics* 176.1 (2002), pp. 93–128.

[Knu00] Patrick M Knupp. "Achieving Finite Element Mesh Quality via Optimization of the Jacobian Matrix Norm and Associated Quantities. Part I - a Framework for Surface Mesh Optimization". *International Journal for Numerical Methods in Engineering* 48.3 (2000), pp. 401–420.

[KSBC12] Misha Kazhdan, Justin Solomon, and Mirela Ben-Chen. "Can Mean-Curvature Flow be Modified to be Non-singular?" *Computer Graphics Forum* 31.5 (2012), pp. 1745–1754.

[Lie03] Peter Liepa. "Filling Holes in Meshes". In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 2003, pp. 200–205.

[Lip13] Konstantin Lipnikov. "On Shape-Regularity of Polyhedral Meshes for Solving PDEs". In *Proceedings of the International Meshing Roundtable*. 2013.

[LLK+20] Andreas Longva, Fabian Löschner, Tassilo Kugelstadt, José Antonio Fernández-Fernández, and Jan Bender. "Higher-Order Finite Elements for Embedded Simulation". *ACM Transactions on Graphics* 39.6 (2020), 181:1–181:14.

[LMS14] Konstantin Lipnikov, Gianmarco Manzini, and Mikhail Shashkov. "Mimetic Finite Difference Method". *Journal of Computational Physics* 257 (2014), pp. 1163–1227.

[Lo14] Daniel SH Lo. "Finite Element Mesh Generation". CRC press, 2014.

[LS08] Torsten Langer and Hans-Peter Seidel. "Higher Order Barycentric Coordinates". *Computer Graphics Forum* 27.2 (2008).

[LSZ+14] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. "Build-to-last: Strength to weight 3D printed objects". *ACM Transactions on Graphics* 33.4 (July 2014).

[LZ10] Bruno Lévy and Hao (Richard) Zhang. "Spectral Mesh Processing". In *ACM SIGGRAPH 2010 Courses*. 2010, 8:1–8:312.

[Mac49] Richard MacNeal. "The Solution of Partial Differential Equations by Means of Electrical Networks." PhD thesis. California Institute of Technology, 1949.

[MDS+03] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds". In *Visualization and Mathematics III*. Ed. by Hans-Christian Hege and Konrad Polthier. Springer-Verlag, 2003, pp. 35–57.

[MKB+08] Sebastian Martin, Peter Kaufmann, Mario Botsch, Martin Wicke, and Markus Gross. "Polyhedral Finite Elements Using Harmonic Basis Functions". *Computer Graphics Forum* 27.5 (2008), pp. 1521–1529.

[MRS14] G. Manzini, A. Russo, and N. Sukumar. "New Perspectives on Polygonal and Polyhedral Finite Element Methods". *Mathematical Models and Methods in Applied Sciences* 24 (2014), pp. 1665–1699.

[MTA+08] Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. "Spectral Conformal Parameterization". *Computer Graphics Forum* 27.5 (2008), pp. 1487–1494.

[MTP+08] Johannes Mezger, Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. "Interactive Physically-Based Shape Editing". In *Proceedings of ACM Symposium on Solid and Physical Modeling*. 2008, pp. 79–89.

[Mus97] Oleg R. Musin. "Properties of the Delaunay Triangulation". In *Proceedings of Symposium on Computational Geometry*. 1997, pp. 424–426.

[MWW15] Lin Mu, Xiaoshen Wang, and Yanqiu Wang. "Shape Regularity Conditions for Polygonal/Polyhedral Meshes, Exemplified in a Discontinuous Galerkin Discretization". *Numerical Methods for Partial Differential Equations* 31.1 (2015), pp. 308–325.

[Nar23] Soorya J M Narayan. "Hair Tubes: Stylized Hair from Polygonal Meshes of Arbitrary Topology". In *SIGGRAPH Asia 2023 Technical Communications*. 2023, pp. 1–4.

[OSTL+12] Ean Tat Ooi, Chongmin Song, Francis Tin-Loi, and Zhenjun Yang. "Polygon Scaled Boundary Finite Elements for Crack Propagation Modelling". *International Journal for Numerical Methods in Engineering* 91.3 (2012), pp. 319–342.

[PKC+18] Fabián Prada, Misha Kazhdan, Ming Chuang, and Hugues Hoppe. "Gradient-Domain Processing within a Texture Atlas". *ACM Transactions on Graphics* 37.4 (2018), 154:1–154:14.

[PNA+21] Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. "Reliable Feature-Line Driven Quad-Remeshing". *ACM Transactions on Graphics* 40.4 (2021), 155:1–155:17.

[PP93] Ulrich Pinkall and Konrad Polthier. "Computing Discrete Minimal Surfaces and Their Conjugates". *Experimental Mathematics* 2 (1993), pp. 15–36.

[PPW18] Chi-Han Peng, Helmut Pottmann, and Peter Wonka. “Designing Patterns Using Triangle-Quad Hybrid Meshes”. *ACM Transactions on Graphics* 37.4 (2018), 107:1–107:14.

[PV21] Lenka Ptáčková and Luiz Velho. “A Simple and Complete Discrete Exterior Calculus on General Polygonal Meshes”. *Computer Aided Geometric Design* 88 (2021), p. 102002.

[QB23] El Houssaine Quenjel and Abdelaziz Beljadid. “Node-Diamond Approximation of Heterogeneous and Anisotropic Diffusion Systems on Arbitrary Two-Dimensional Grids”. *Mathematics and Computers in Simulation* 204.C (Feb. 2023), pp. 450–472.

[Rap17] Bastian E. Rapp. “Chapter 31 - Finite Volume Method”. In *Microfluidics: Modelling, Mechanics and Mathematics*. Micro and Nano Technologies. Oxford: Elsevier, 2017, pp. 633–654. ISBN: 978-1-4557-3141-1.

[Ros97] Steven Rosenberg. “The Laplacian on a Riemannian Manifold”. In. *The Laplacian on a Riemannian Manifold: An Introduction to Analysis on Manifolds*. London Mathematical Society Student Texts. Cambridge University Press, 1997, pp. 1–51.

[RS06] M. Rashid and M. Selimotic. “A Three-Dimensional Finite Element Method With Arbitrary Polyhedral Elements”. *International Journal for Numerical Methods in Engineering* 67 (2006), pp. 226 – 252.

[SB19] Daniel Sieger and Mario Botsch. *The Polygon Mesh Processing Library*. http://www.pmp-library.org. 2019.

[SBB+22] Patrick Schmidt, Janis Born, David Bommes, Marcel Campen, and Leif Kobbelt. “TinyAD: Automatic Differentiation in Geometry Processing Made Simple”. *Computer Graphics Forum* 41.5 (2022), pp. 113–124.

[SBM+23] Tommaso Sorgente, Silvia Biasotti, Gianmarco Manzini, and Michela Spagnuolo. “A Survey of Indicators for Mesh Quality Assessment”. *Computer Graphics Forum* 42.2 (2023), pp. 461–483.

[SCOL+04] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. “Laplacian Surface Editing”. In *Proceedings of Eurographics Symposium on Geometry Processing*. 2004, pp. 179–188.

[SDG+19] Teseo Schneider, Jérémie Dumas, Xifeng Gao, Mario Botsch, Daniele Panozzo, and Denis Zorin. “Poly-Spline Finite-Element Method”. *ACM Transactions on Graphics* 38.3 (2019), 19:1–19:16.

[Sha21] Nicholas Sharp. “Intrinsic Triangulations in Geometry Processing”. PhD thesis. Carnegie Mellon University, Dec. 2021.

[SHD+18] Teseo Schneider, Yixin Hu, Jérémie Dumas, Xifeng Gao, Daniele Panozzo, and Denis Zorin. "Decoupling Simulation Accuracy from Mesh Quality". *ACM Transactions on Graphics* 37.6 (2018), 280:1–280:14.

[She02] Jonathan Richard Shewchuk. "What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures." In *Proceedings of the International Meshing Roundtable*. 2002, pp. 115–126.

[SHG+22] Teseo Schneider, Yixin Hu, Xifeng Gao, Jeremie Dumas, Denis Zorin, and Daniele Panozzo. "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.

[SSC19a] Nicholas Sharp, Yousuf Soliman, and Keenan Crane. "Navigating Intrinsic Triangulations". *ACM Transactions on Graphics* 38.4 (2019).

[SSC19b] Nicholas Sharp, Yousuf Soliman, and Keenan Crane. "The Vector Heat Method". *ACM Transactions on Graphics* 38.3 (2019), 24:1–24:19.

[Suk04] N. Sukumar. "Construction of Polygonal Interpolants: A Maximum Entropy Approach". *International Journal for Numerical Methods in Engineering* (2004), pp. 2159–2181.

[Tau95] Gabriel Taubin. "A Signal Processing Approach to Fair Surface Design". In *Proceedings of ACM SIGGRAPH*. 1995, pp. 351–358.

[TS06] Alireza Tabarraei and N. Sukumar. "Application of Polygonal Finite Elements in Linear Elasticity". *International Journal of Computational Methods* 03 (2006), pp. 503–520.

[TS08] Alireza Tabarraei and N. Sukumar. "Extended Finite Element Method on Polygonal and Quadtree Meshes". *Computer Methods in Applied Mechanics and Engineering* 197 (2008), pp. 425–438.

[VAG+08] Dimitris Vartziotis, Theodoros Athanasiadis, Iraklis Goudas, and Joachim Wipper. "Mesh Smoothing Using the Geometric Element Transformation Method". *Computer Methods in Applied Mechanics and Engineering* 197.45–48 (2008), pp. 3760–3767.

[VBC+13] Lourenço Beirão da Veiga, Franco Brezzi, Andrea Cangiani, Gianmarco Manzini, Luisa Donatella Marini, and Alessandro Russo. "Basic Principles of Virtual Element Methods". *Mathematical Models and Methods in Applied Sciences* 23.1 (2013), pp. 199–214.

[VBM13] Lourenço Beirão da Veiga, Franco Brezzi, and Luisa Donatella Marini. “Virtual Elements for Linear Elasticity Problems”. *SIAM J. Numer. Anal.* 51.2 (2013), pp. 794–812.

[VDR17] Lourenço Beirão da Veiga, Franco Dassi, and Alessandro Russo. “High-order Virtual Element Method on Polyhedral Meshes”. *Computers and Mathematics with Applications* 74 (2017), pp. 1110–1122.

[VL08] B. Vallet and B. Lévy. “Spectral Geometry Processing with Manifold Harmonics”. *Computer Graphics Forum* 27.2 (2008), pp. 251–260.

[VW12] Dimitris Vartziotis and Joachim Wipper. “Fast Smoothing of Mixed Volume Meshes Based on the Effective Geometric Element Transformation Method”. *Computer Methods in Applied Mechanics and Engineering* 201–204 (2012), pp. 65–81.

[Wac75] Eugene L. Wachspress. “A Rational Finite Element Basis”. Academic Press, 1975.

[War08] Max Wardetzky. “Convergence of the Cotangent Formula: An Overview”. In. *Discrete Differential Geometry*. Basel: Birkhäuser Basel, 2008, pp. 275–286.

[WBG07] Martin Wicke, Mario Botsch, and Markus Gross. “A Finite Element Method on Convex Polyhedra”. *Computer Graphics Forum* 26.3 (2007), pp. 355–364.

[Wey12] Hermann Weyl. “Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung)”. *Mathematische Annalen* 71 (1912), pp. 441–479.

[Whi57] Hassler Whitney. “Geometric Integration Theory”. Princeton University Press, 1957.

[WMK+07] Max Wardetzky, Saurabh Mathur, Felix Kälberer, and Eitan Grinspun. “Discrete Laplace Operators: No Free Lunch”. In *Proceedings of Eurographics Symposium on Geometry Processing*. 2007, pp. 33–37.

[Zha04] Hao Zhang. “Discrete Combinatorial Laplacian Operators for Digita Geometry Processing”. In *Proceedings of SIAM Conference on Geometric Design and Computing*. 2004, pp. 575–592.

[ZS00] Tian Zhou and Kenji Shimada. “An Angle-Based Approach to Two-Dimensional Mesh Smoothing”. In *Proceedings of the International Meshing Roundtable*. 2000, pp. 373–384.

APPENDIX A

A.1 TEST MESHES

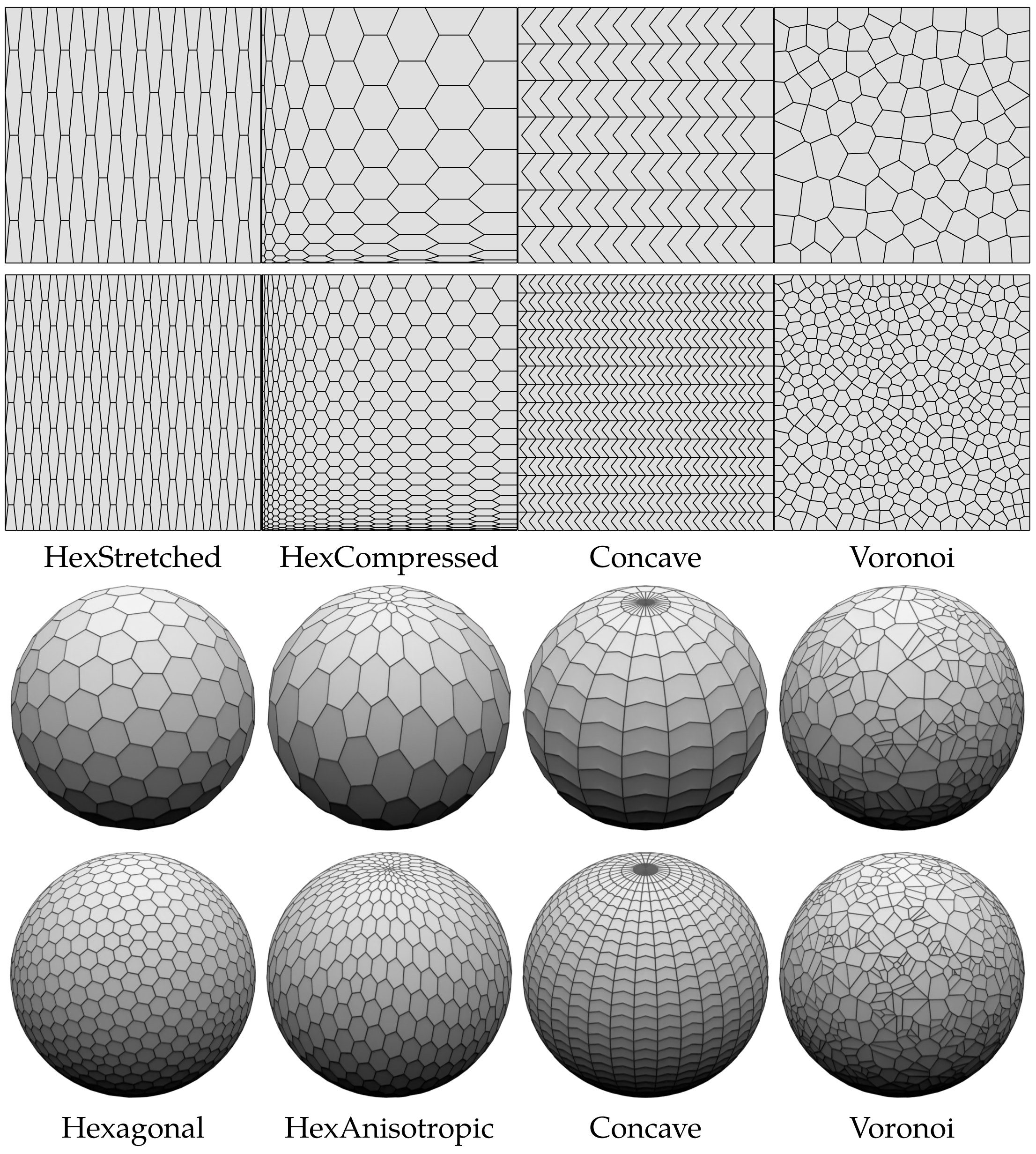


Figure A.1: *Some of the more abstract mesh types used in our evaluation. Shown here are resolutions 2 and 3 of the five levels used.*

A.2 FINDING THE OPTIMAL VIRTUAL POINT

Given a polygon with n_f vertices $(\mathbf{x}_1, \dots, \mathbf{x}_{n_f})$, we want to insert a point $\mathbf{x} = \mathbf{x}(\mathbf{w})$, defined as an affine combination $\mathbf{x}(\mathbf{w}) = \sum_{j=1}^{n_f} w_j \mathbf{x}_j$ with $\sum_j w_j = 1$, such that the sum of squared triangle areas over the resulting triangle fan is minimized. This leads to the following optimization problem in the weights $\mathbf{w} = (w_1, \dots, w_{n_f})^\mathsf{T}$:

$$\min_{\mathbf{w}} \sum_{i=1}^{n_f} \operatorname{area}\left(\mathbf{x}_i, \mathbf{x}_{i+1}, \sum_{j=1}^{n} w_j \mathbf{x}_j\right)^2 \quad \text{s.t.} \quad \sum_{j=1}^{n_f} w_j = 1. \tag{A.1}$$

The objective function can be rewritten as

$$E(\mathbf{w}) = \sum_{i=1}^{n_f} \frac{1}{2} \left\| (\mathbf{x}_i - \mathbf{x}_{i+1}) \times (\mathbf{x}(\mathbf{w}) - \mathbf{x}_i) \right\|^2. \tag{A.2}$$

Since E is quadratic in $\mathbf{x}(\mathbf{w})$ and therefore also quadratic in $\mathbf{w}$, it can be written as

$$E(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{A}\mathbf{w} + \mathbf{b}^\mathsf{T}\mathbf{w} + c. \tag{A.3}$$

Minimizing with respect to $\mathbf{w}$, i.e., setting $\frac{\partial E}{\partial \mathbf{w}}$ to zero, leads to

$$\begin{aligned} &\mathbf{A}\mathbf{w} = -\mathbf{b} \quad \text{with} \\ &\mathbf{A}_{ij} = 2\sum_{k=1}^{n_f} \left(\mathbf{x}_j \times (\mathbf{x}_{k+1} - \mathbf{x}_k)\right) \cdot \left(\mathbf{x}_i \times (\mathbf{x}_{k+1} - \mathbf{x}_k)\right), \\ &\mathbf{b}_i = 2\sum_{k=1}^{n_f} \left(\mathbf{x}_i \times (\mathbf{x}_{k+1} - \mathbf{x}_k)\right) \cdot \left((\mathbf{x}_{k+1} - \mathbf{x}_k) \times \mathbf{x}_k\right) \end{aligned} \tag{A.4}$$

We add one row to the matrix to enforce the partition of unity constraint $\sum_j w_j = 1$, turning it into the $(n_f + 1) \times n_f$ linear system

$$\begin{pmatrix} \mathbf{A} \\ 1 \cdots 1 \end{pmatrix} \mathbf{w} = \begin{pmatrix} -\mathbf{b} \\ 1 \end{pmatrix}. \tag{A.5}$$

The matrix $\mathbf{A}$ has rank 2 or 3 for planar or non-planar polygons, respectively. Hence, the system in Equation (A.5) has rank 3 or 4 for planar/non-planar polygons. It is therefore fully determined for either (planar) triangles or non-planar quads, and is underdetermined otherwise. In the latter case, we aim for the least-norm solution, i.e., the solution $\mathbf{w}$ with minimal $\|\mathbf{w}\|$, because it distributes the influence of polygon vertices $\mathbf{x}_i$ equally. We handle both the fully-determined and the under-determined cases in a robust and unified manner through the matrix pseudo-inverse [GVL96], which we compute through Eigen's complete orthogonal decomposition [GJ+10].

APPENDIX B

This appendix lists proofs for some of the properties we observed empirically in Chapter 5 and leveraged to develop a more efficient implementation. These proofs were all authored by Misha Kazhdan, and the author of this thesis takes no credit for this section. However, it is necessary for a better understanding of Chapter 5 and therefore included in this thesis. In particular, we will show that when an arbitrarily small Dirichlet energy is incorporated into the gradient continuity energy, the derived basis functions are uniquely defined, the partition of unity property is satisfied automatically (and does not to be enforced explicitly), and the linear precision property is satisfied automatically for faces that are planar. We start with the proof that our shape functions reproduce the quadratic Lagrange elements on triangle and tetrahedral meshes. In the second part of this appendix, we address the properties that can only be proven by adding the Dirchilet constraint to our energy. We start by formalizing terminology and notation (Section B.2), to then prove that the function basis is unique, that the partition of unity property is satisfied automatically, and that the linear precision property automatically holds for planar cells for both the single-level system (Section B.3) and for the hierarchical construction (Section B.4). We conclude with a short discussion about the generalizability of the approach (Section B.5).

b.1 LAGRANGE ELEMENT REPRODUCTION

Given a simplex σ in 2D or 3D, let $\{\phi_i^*\}$ be the standard quadratic Lagrange elements, let $\{\psi_i\}$ be Lagrange elements on the virtually refined simplex, and let $\{\phi_i\}$ be the linear combinations of $\{\psi_i\}$ that satisfy the Lagrange interpolation property and minimize the gradient discontinuity energy.

Claim 1. *The two sets of functions are identical $\{\phi_i\} = \{\phi_i^*\}$.*

To prove the claim we formulate several lemmas.

Lemma B.1.1. *Given polynomials P_1 and P_2 of degree d in $\mathbb{R}^D$ whose values and partial derivatives up to order $d-1$ agree on the hyperplane $x_1 = 0$, there exists $\alpha \in \mathbb{R}$ such that:*

$$P_1(x_1, \dots, x_D) = P_2(x_1, \dots, x_D) + \alpha \cdot x_1^d.$$

Proof. Expanding the polynomials as:

$$P_i(x_1, \dots, x_D) = \sum_{j_1+\dots+j_D \le d} a^i_{j_1 \dots j_D} \cdot x_1^{j_1} \cdots x_D^{j_D}$$

we note that the coefficients $a^i_{j_1\cdots j_D}$ must agree whenever $j_1 < d$. To see this, take the j_1-th partial derivative with respect to x_1 and consider the resulting polynomials Q_i in $D-1$ variables:

$$\begin{aligned} Q_i(x_2,\ldots,x_D) &= \frac{\partial^{j_1}}{\partial x_1^{j_1}} P_i(0,x_2,\ldots,x_D) \\ &= \sum_{j_2+\ldots+j_D \le d-j_1} (j_1!)\cdot a^i_{j_1\cdots j_D}\cdot x_2^{j_2}\cdots x_D^{j_D}. \end{aligned}$$

Since we take $j_1 < d$ derivatives, the polynomials Q_1 and Q_2 are equal, implying that their coefficients $a^i_{j_1\cdots j_D}$ are the same. Thus, the polynomials P_1 and P_2 can only differ in their x_1^d term. □

Corollary B.1.1.1. *Although we required the values and derivatives (up to order $d-1$) of P_1 and P_2 to agree on the hyperplane $x_1 = 0$, Lemma 1 holds if they agree on any open (non-empty) subset of the hyperplane. (This follows from the fact that polynomials are analytic.)*

Corollary B.1.1.2. *We can replace the condition that the polynomials P_1 and P_2 agree on the hyperplane $x_1 = 0$ with the condition that P_1 and P_2 agree on a hyperplane defined by the linear system $\langle \mathbf{x},\mathbf{v}\rangle = 0$ for any $\mathbf{v}\in\mathbb{R}^D$. In this case there exists $\alpha\in\mathbb{R}$ such that:*

$$P_1(\mathbf{x}) = P_2(\mathbf{x}) + \alpha\cdot\langle \mathbf{x},\mathbf{v}\rangle^d.$$

Lemma B.1.2. *Given a vector $\mathbf{v}$, denote by $\Lambda_{\mathbf{v}} : \mathbb{R}^D \to \mathbb{R}$ the linear function $\Lambda_{\mathbf{v}}(\mathbf{x}) \equiv \langle \mathbf{v},\mathbf{x}\rangle$. Given $D+1$ vectors $\{\mathbf{v}_0,\ldots,\mathbf{v}_D\}$ in $\mathbb{R}^D$, no D of which are linearly dependent, the set of functions:*

$$\left\{\Lambda^d_{\mathbf{v}_i}(\mathbf{x}) = \langle \mathbf{v}_i,\mathbf{x}\rangle^d\right\}$$

are linearly independent for all $d > 1$.

Proof. The statement is invariant under linear transformation so we can assume, without loss of generality, that $\{\mathbf{v}_1,\ldots,\mathbf{v}_D\}$ are the coordinate axes. By linear independence, we have: $\mathbf{v}_0 = (v_1,\ldots,v_D)$ where *all* of the v_i are non-zero. In this coordinate system we have:

$$\begin{aligned} \Lambda_{\mathbf{v}_i}(x_1,\ldots,x_D) &= x_i^d, \quad \forall 1\le i\le D \\ \Lambda_{\mathbf{v}_0}(x_1,\ldots,x_D) &= \left(\sum_{i=1}^{D} v_i x_i\right)^d. \end{aligned}$$

The $\Lambda_{\mathbf{v}_i}$ (w/ $1\le i\le D$) are linearly independent as they are functions of different variables. In addition $\Lambda_{\mathbf{v}_0}$ cannot be expressed as the linear combination of the $\Lambda_{\mathbf{v}_i}$ (w/ $1\le i\le D$) since $\Lambda_{\mathbf{v}_0}$ contains mixed terms while the monomials $\Lambda_{\mathbf{v}_i}$ (w/ $1\le i\le D$) do not. □

We can now prove the claim. We proceed in two steps. First we show that the Lagrange basis function, $\{\phi_i^*\}$, are in the span of $\{\psi_i\}$ and that they have zero energy. Then we show that these are the only functions that have zero energy and satisfy the Lagrange interpolation property, implying that they are the unique minimizers. Proving the first statement is straight-forward. Proving the second amounts to associating functions with edges in the adjacency graph of the simplicial refinement and showing that the functions associated with cycles in the graph are linearly independent.

Proof. To prove that the $\{\phi_i^*\}$ are a solution, it suffices to show that each ϕ_j^* is in the span of $\{\psi_i\}$ and that the energy of each ϕ_j^* is zero. To prove that ϕ_j^* is in the span of $\{\psi_i\}$, let $\{\mathbf{p}_k\} \subset \sigma$ be the Lagrange nodes in the refined complex Σ_σ. Setting

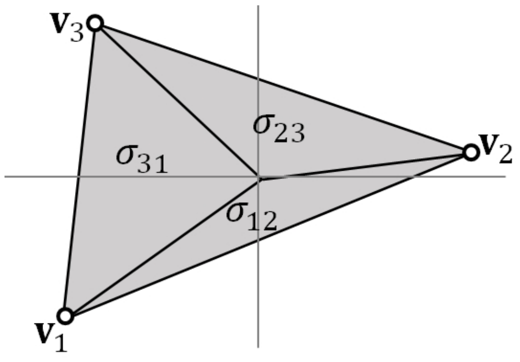
$$\phi_j(\mathbf{x}) = \sum_k \phi_j^*(\mathbf{p}_k) \cdot \psi_k(\mathbf{x})$$

we get a piecewise quadratic polynomial that agrees with the Lagrange basis functions ϕ_j^* at all refined nodes. By the Lagrange interpolation property, this implies that ϕ_j and ϕ_j^* agree on each $\sigma' \in \Sigma_\sigma$. Thus ϕ_j and ϕ_j^* agree on all of σ. Additionally, since the $\{\phi_i^*\}$ are strictly quadratic, they have no derivative discontinuity and the discontinuity energy is trivially zero.

To prove that $\{\phi_i^*\}$ is the unique minimizer we show that if the $\{\phi_i\}$ have zero energy then they must be strictly quadratic in σ. Then the proof follows from the fact that Lagrange basis functions are the only quadratic functions satisfying the interpolation property.

Since the claim is invariant to affine transformation, we assume that the simplex σ is translated so that the virtual vertex introduced in the interior of σ is at the origin, and consider two cases.

2D We start with the case of a triangle $\sigma = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$. After virtual refinement, σ is partitioned into three subtriangles, $\Sigma_\sigma = \{\sigma_{12}, \sigma_{23}, \sigma_{31}\}$, with indices chosen so that vertex $\mathbf{v}_i$ is opposite triangle σ_{jk} for $i \neq j, k$.



Now, given a piecewise quadratic polynomial P (strictly quadratic within each σ_{ij}) whose gradient is continuous across σ, we denote the restriction of P to σ_{ij}

as P_{ij}. By Corollary B.1.1.2 we have:

$$\begin{aligned} P_{12}(\mathbf{x}) &= P_{23}(\mathbf{x}) + \alpha_2 \cdot \Lambda^2_{\mathbf{v}_2^\perp}(\mathbf{x}) \\ &= P_{31}(\mathbf{x}) + \alpha_3 \cdot \Lambda^2_{\mathbf{v}_3^\perp}(\mathbf{x}) + \alpha_2 \cdot \Lambda^2_{\mathbf{v}_2^\perp}(\mathbf{x}) \\ &= P_{12}(\mathbf{x}) + \alpha_1 \cdot \Lambda^2_{\mathbf{v}_1^\perp}(\mathbf{x}) + \alpha_3 \cdot \Lambda^2_{\mathbf{v}_3^\perp}(\mathbf{x}) + \alpha_2 \cdot \Lambda^2_{\mathbf{v}_2^\perp}(\mathbf{x}) \end{aligned}$$

for real values $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{R}$. Or, equivalently:

$$0 = \alpha_1 \cdot \Lambda^2_{\mathbf{v}_1^\perp}(\mathbf{x}) + \alpha_2 \cdot \Lambda^2_{\mathbf{v}_2^\perp}(\mathbf{x}) + \alpha_3 \cdot \Lambda^2_{\mathbf{v}_3^\perp}(\mathbf{x}).$$

Since no two of the $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ are collinear, by Lemma B.1.2 the functions $\{\Lambda_{\mathbf{v}_i^\perp}\}$ are linearly independent so that $\alpha_1 = \alpha_2 = \alpha_3 = 0$ and hence P is strictly quadratic in σ.

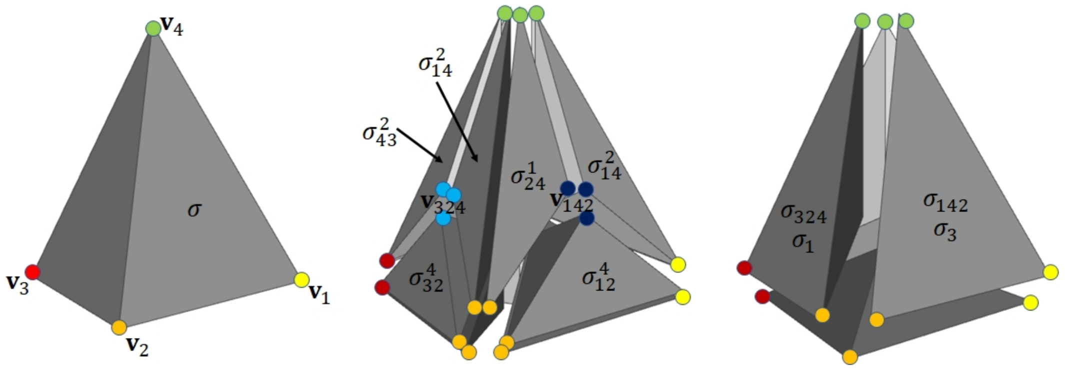


Figure B.1: *Visualization of the domains considered in the two steps of the volumetric proof.*

3D Next, we consider the case of a tetrahedron $\sigma = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4\}$ and assume we have a piecewise quadratic polynomial P (strictly quadratic within each $\sigma' \in \Sigma_\sigma$) whose gradient is continuous across σ. We proceed in two steps. First we show that P must be strictly quadratic within the tetrahedra defined by joining the faces of σ with the origin (i.e. the virtual vertex defined in the interior of σ). Then we show that P is strictly quadratic in σ.
Denote by $\sigma_{ijk} = \{\mathbf{0}, \mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k\}$ the tetrahedron defined by joining the face $\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k\}$ with the origin. Note that σ_{ijk} is not in the refined complex Σ_σ but we can express σ_{ijk} as the union of three simplices in Σ_σ. Specifically, if we denote by $\mathbf{v}_{ijk}$ the virtual vertex in the interior of the face $\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k\}$ and set σ^k_{ij} to be the simplex $\sigma^k_{ij} = \{\mathbf{0}, \mathbf{v}_{ijk}, \mathbf{v}_i, \mathbf{v}_j\}$ which is in the refined complex, then we have:

$$\sigma_{ijk} = \sigma^k_{ij} \cup \sigma^i_{jk} \cup \sigma^j_{ki}.$$

Again, noting that P is strictly quadratic within each each σ^k_{ij}, we denote by P^k_{ij} the restriction of P to σ^k_{ij}. Proceeding as before, cycling around edge $\{\mathbf{0}, \mathbf{v}_{ijk}\}$, we have:

$$P_{ij}^{k}(\mathbf{x}) = P_{ij}^{k}(\mathbf{x}) + \alpha_i^{jk} \cdot \Lambda^2_{\mathbf{v}_i^{jk}}(\mathbf{x}) + \alpha_k^{ij} \cdot \Lambda^2_{\mathbf{v}_k^{ij}}(\mathbf{x}) + \alpha_j^{ik} \cdot \Lambda^2_{\mathbf{v}_j^{ik}}(\mathbf{x}).$$

where $\mathbf{v}_j^{ik} = \mathbf{v}_j \times \mathbf{v}_{ijk}$ is the vector perpendicular to the face separating simplices σ_{ij}^k and σ_{jk}^i. Or, equivalently:

$$0 = \alpha_i^{jk} \cdot \Lambda^2_{\mathbf{v}_i^{jk}}(\mathbf{x}) + \alpha_j^{ik} \cdot \Lambda^2_{\mathbf{v}_j^{ik}}(\mathbf{x}) + \alpha_k^{ij} \cdot \Lambda^2_{\mathbf{v}_k^{ij}}(\mathbf{x}).$$

As before, using the linear independence of $\mathbf{v}_i^{jk}$, $\mathbf{v}_j^{ki}$, and $\mathbf{v}_k^{ij}$, and applying Lemma B.1.2, it follows that $\alpha_i^{jk} = \alpha_j^{ki} = \alpha_k^{ij} = 0$ so that P is strictly quadratic in σ_{ijk}.
Finally, we show that P is strictly quadratic in σ. For simplicity, we will denote by σ_i the simplex opposite vertex $\mathbf{v}_i$:

$$\sigma_i = \{\mathbf{0}, \mathbf{v}_{i+1}, \mathbf{v}_{i+2}, \mathbf{v}_{i+3}\}$$

and we will denote by P_i the restriction of P to σ_i (which we have shown is strictly quadratic). Without loss of generality, fixing vertex $\mathbf{v}_4$, we consider simplices σ_1, σ_2, and σ_3. These meet at edge $\{\mathbf{0}, \mathbf{v}_4\}$ and, again, cycling through the simplices around the edge we get:

$$P_1(\mathbf{x}) = P_1(\mathbf{x}) + \alpha_2^4 \cdot \Lambda_{\mathbf{v}_4 \times \mathbf{v}_2}(\mathbf{x}) + \alpha_1^4 \cdot \Lambda_{\mathbf{v}_4 \times \mathbf{v}_1}(\mathbf{x}) + \alpha_3^4 \cdot \Lambda_{\mathbf{v}_4 \times \mathbf{v}_3}(\mathbf{x}).$$

Noting that $\mathbf{v}_4 \times \mathbf{v}_1$, $\mathbf{v}_4 \times \mathbf{v}_2$, and $\mathbf{v}_4 \times \mathbf{v}_3$ are linearly independent, it follows that $\alpha_1^4 = \alpha_2^4 = \alpha_3^4 = 0$ and hence P is strictly quadratic on $\sigma_1 \cup \sigma_2 \cup \sigma_3$. Repeating this argument for the $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$, it follows that P is strictly quadratic on $\sigma_2 \cup \sigma_3 \cup \sigma_4$, on $\sigma_3 \cup \sigma_4 \cup \sigma_1$, and on $\sigma_4 \cup \sigma_1 \cup \sigma_2$. Thus P is strictly quadratic on all of σ. $\square$

B.2 TERMINOLOGY AND NOTATION

In what follows, we assume that all complexes are *pure*. That is, if the complex is D-dimensional, then every d-dimensional simplex/cell (with $d < D$) is on the boundary of some $(d+1)$-dimensional simplex/cell
We begin by presenting the terminology and notation used in the proofs. A brief description can be found in Table B.1.

Simplicial Complexes

- We denote by Σ a D-dimensional simplicial complex.

Symbol	Definition
Σ	a D-dimensional simplicial complex
Σ_d	the subset of d-dimensional simplices in Σ
$\mathcal{N}(\Sigma)$	the set of Lagrange nodes of Σ
$\mathcal{B}(\Sigma)$	the set of Lagrange basis functions on Σ
$\mathcal{C}$	a D-dimensional cell complex
$\mathcal{C}^d$	the d-dimensional sub-complex of $\mathcal{C}$
$\Sigma(\mathcal{C})$	the simplicial complex obtained by refining $\mathcal{C}$
$\mathcal{N}^d$	the set of Lagrange nodes on $\Sigma(\mathcal{C}^d)$
$\mathcal{B}^d$	the set of Lagrange basis functions on $\Sigma(\mathcal{C}^d)$
$\mathring{\mathcal{B}}^d$	the subset of $\mathcal{B}^d$ indexed by interior nodes
$\mathcal{Q}_d$	the quadratic energy on $\text{Span}(\mathcal{B}^d)$
$\langle\cdot,\cdot\rangle_d$	the symmetric bilinear form defined by $\mathcal{Q}_d$
$\|\cdot\|_d^2$	the squared norm $\mathcal{Q}_d$
$\mathcal{B}^d_{d'}$	the function basis on $\Sigma(\mathcal{C}^d)$ indexed by $\mathcal{N}^{d'}$ $(d' \leq d)$
$\psi^d_{\eta,d'}$	the function in $\mathcal{B}^d_{d'}$ indexed by node $\eta \in \mathcal{N}^{d'}$

Table B.1: *Summary of notation*

- We denote by Σ_d (for $0 \leq d \leq D$) the set of d-dimensional simplices in Σ.
- Given a D-dimensional simplicial complex Σ, we denote by $\mathcal{N}(\Sigma)$ the set of quadratic Lagrange nodes of Σ.
- Given a D-dimensional simplicial complex Σ, we denote by $\mathcal{B}(\Sigma)$ the set of quadratic Lagrange basis functions on Σ. Specifically $\mathcal{B}(\Sigma) = \{\psi^d_{\eta,d}\}_{\eta\in\mathcal{N}(\Sigma)}$ where each $\psi^d_{\eta,d}$ is strictly quadratic within each simplex $\sigma \in \Sigma$ and satisfies the interpolation condition, $\psi^d_{\eta,d}(\eta') = \delta_{\eta\eta'}$ for all $\eta' \in \mathcal{N}(\Sigma)$.

Linear Precision

- Given a domain embedded in Euclidean space, $\Omega \subset \mathbb{R}^n$, we say that a set of functions $\{\varphi_i\} : \Omega \to \mathbb{R}$, has <u>linear precision</u> if for every linear function $L : \mathbb{R}^n \to \mathbb{R}$, there exist coefficients $\{x_i\} \subset \mathbb{R}$ with:

$$L\big|_\Omega = \sum_i x_i \cdot \varphi_i.$$

Cell Complexes and Their Refinement

- We denote by $\mathcal{C}$ a D-dimensional cell complex.

- We denote by $\mathcal{C}^d$ (for $0 \leq d \leq D$) the d-dimensional sub-complex obtained by only considering cells in $\mathcal{C}$ with dimension less than or equal to d. In particular, $\mathcal{C}^0 \subset \cdots \subset \mathcal{C}^D = \mathcal{C}$.
- Assuming that there is a virtual vertex associated with each d-dimensional cell in $\mathcal{C}$ (for all $d > 1$), we denote by $\Sigma(\mathcal{C})$ the simplicial complex obtained by refining $\mathcal{C}$.
- We set $\mathcal{N}^d \equiv \mathcal{N}\left(\Sigma(\mathcal{C}^d)\right)$ to be the set of quadratic Lagrange nodes defined on the simplicial refinement of $\mathcal{C}^d$. Because the complex $\mathcal{C}$ is pure, we have $\mathcal{N}^0 \subset \cdots \subset \mathcal{N}^D$.
- We set $\mathcal{B}^d \equiv \mathcal{B}\left(\Sigma(\mathcal{C}^d)\right) = \{\psi^d_{\eta,d}\}_{\eta\in\mathcal{N}^d}$ to be the set of Lagrange basis functions defined over the simplicial refinement of the sub-complex $\mathcal{C}^d$.
- We set $\mathring{\mathcal{B}}^d \subset \mathcal{B}^d$ to be the subset of Lagrange basis functions on $\Sigma(\mathcal{C}^d)$ indexed by nodes interior to the d-dimensional cells in $\mathcal{C}^d$:

$$\mathring{\mathcal{B}}^d = \{\psi^d_{\eta,d}\}_{\eta\in\mathcal{N}^d\setminus\mathcal{N}^{d-1}}.$$

The Quadratic Energy

- Given a cell complex $\mathcal{C}$, given a dimension $1 \leq d \leq D$, and given $\varepsilon > 0$, we define a quadratic energy $\mathcal{Q}_d(\cdot)$ on the space of functions on $\Sigma(\mathcal{C}^d)$:

$$\mathcal{Q}_d\left(\psi^d\right) = \sum_{\sigma\in\Sigma_{d-1}(\mathcal{C}^d)\setminus\Sigma_{d-1}(\mathcal{C}^{d-1})} \int_\sigma \left\|\nabla^+_\sigma\psi^d - \nabla^-_\sigma\psi^d\right\|^2 d\sigma + \sum_{\sigma\in\Sigma_d(\mathcal{C}^d)} \varepsilon\cdot\int_\sigma \left\|\nabla\psi^d\right\|^2 d\sigma$$

  The first integral measures the C^1-continuity of ψ^d, and is taken over all $(d-1)$-dimensional simplices in the refinement of $\mathcal{C}^d$ that are not in the refinement of $\mathcal{C}^{d-1}$. The second integral is a Dirichlet regularizer and is taken over all d-dimensional simplices in the refinement of $\mathcal{C}^d$.
- We denote by $\langle\cdot,\cdot\rangle_d$ the symmetric, positive, semi-definite form defined by $\mathcal{Q}_d$ and write the energy as $\|\cdot\|^2_d \equiv \mathcal{Q}_d(\cdot)$.

Given the quadratic energy, we denote by $\mathcal{B}^d_{d-1} \subset \text{Span}(\mathcal{B}^d)$ the "coarse" basis on $\Sigma(\mathcal{C}^d)$. This is the set of functions $\mathcal{B}^d_{d-1} = \{\psi^d_{\eta,d-1}\}$, defined on $\Sigma(\mathcal{C}^d)$ but indexed by nodes $\eta \in \mathcal{N}^{d-1}$, where each $\psi^d_{\eta,d-1}$ minimizes the energy, subject to the interpolation constraint:

$$\psi^d_{\eta,d-1} = \underset{\psi^d\in\text{Span}(\mathcal{B}^d)}{\arg\min} \left\|\psi^d\right\|^2_d \quad \text{s.t.} \quad \psi^d_{\eta,d-1}(\eta') = \delta_{\eta\eta'}, \quad \forall\eta'\in\mathcal{N}^{d-1}.$$

B.3 PROPERTIES OF THE BASIS $\mathcal{B}^d_{d-1}$

In what follows, we demonstrate that the functions in $\mathcal{B}^d_{d-1}$ are unique, satisfy the partition of unity property, and (under appropriate conditions) have linear precision.
In doing so, we make use of the following lemmas.

Lemma B.3.1. *The basis $\mathring{\mathcal{B}}^d$ spans the subspace of functions in Span$(\mathcal{B}^d)$ that vanish on $\Sigma(\mathcal{C}^{d-1})$.*

Proof. Suppose we are given a function $\psi^d \in \text{Span}(\mathcal{B}^d)$:

$$\psi^d = \sum_{\eta \in \mathcal{N}^d} \alpha_\eta \cdot \psi^d_{\eta,d}.$$

If ψ^d vanishes on $\Sigma(\mathcal{C}^{d-1})$ then, in particular, it must vanish at every node $\eta' \in \mathcal{N}^{d-1}$. But because the $\mathcal{B}^d$ satisfy the Lagrange interpolation property, we have $\psi^d(\eta') = \alpha_{\eta'}$. In particular, this implies that if ψ^d vanishes on $\Sigma(\mathcal{C}^{d-1})$ then $\psi^d \in \mathring{\mathcal{B}}^d$.
Conversely, as $\mathcal{N}^{d-1} \subset \mathcal{N}^d$, given $\psi^d_{\eta,d}$ with $\eta \in \mathcal{N}^d \backslash \mathcal{N}^{d-1}$ we must have $\psi^d_{\eta,d}(\eta') = 0$ for all $\eta' \in \mathcal{N}^{d-1}$ since $\psi^d_{\eta,d}$ satisfies the interpolation property. On the other hand, since $\psi^d_{\eta,d}$ is strictly quadratic, its restriction to $\Sigma(\mathcal{C}^{d-1})$ will also be strictly quadratic, so that the fact that it vanishes at all quadratic Lagrange nodes $\mathcal{N}^{d-1}$ implies that it must be constantly zero on $\Sigma(\mathcal{C}^{d-1})$. □

Lemma B.3.2. *Though the symmetric bilinear form $\langle \cdot, \cdot \rangle_d$ is only <u>semi</u>-definite on Span$(\mathcal{B}^d)$ it is strictly definite on Span$(\mathring{\mathcal{B}}^d)$.*

Proof. Suppose that we have $\mathring{\psi}^d \in \text{Span}(\mathring{\mathcal{B}}^d)$ such that $\|\mathring{\psi}^d\|^2_d = 0$. Since $\mathring{\psi}^d \in \text{Span}(\mathcal{B}^d)$ it must be continuous. Let $c \in \mathcal{C}^d \backslash \mathcal{C}^{d-1}$ be a d-dimensional cell in $\mathcal{C}$. Since $\|\mathring{\psi}^d\|^2_d = 0$, it must be constant on c. And since $\mathring{\psi}^d \in \text{Span}(\mathring{\mathcal{B}}^d)$ it vanishes on the boundary $\partial\Sigma(c) \subset \Sigma(\mathcal{C}^{d-1})$. Thus it must be the case that $\mathring{\psi}^d = 0$. □

Lemma B.3.3. *The spaces Span$(\mathcal{B}^d_{d-1})$ and Span$(\mathring{\mathcal{B}}^d)$ are orthogonal with respect to $\langle \cdot, \cdot \rangle_d$.*

Proof. It suffices to show that $\langle \psi^d_{\eta,d-1}, \mathring{\psi}^d \rangle_d = 0$ for all $\eta \in \mathcal{N}^{d-1}$ and all $\mathring{\psi}^d \in \text{Span}(\mathring{\mathcal{B}}^d)$.
Suppose to the contrary that there exists $\eta \in \mathcal{N}^{d-1}$ and $\mathring{\psi}^d \in \text{Span}(\mathring{\mathcal{B}}^d)$ with $\langle \psi^d_{\eta,d-1}, \mathring{\psi}^d \rangle_d \neq 0$. Consider the function

$$\psi^{d,\alpha}_{\eta,d-1} = \psi^d_{\eta,d-1} + \alpha \cdot \mathring{\psi}^d.$$

On the one hand we have $\psi^{d,\alpha}_{\eta,d-1}(\eta') = \psi^d_{\eta,d-1}(\eta') = \delta_{\eta\eta'}$ for all $\eta' \in \mathcal{N}^{d-1}$. On the other, the energy of $\psi^{d,\alpha}_{\eta,d-1}$ can be expressed as:

$$\begin{aligned}\left\|\psi^{d,\alpha}_{\eta,d-1}\right\|^2_d &= \left\|\psi^d_{\eta,d-1} + \alpha\cdot\mathring{\psi}^d\right\|^2_d \\ &= \left\|\psi^d_{\eta,d-1}\right\|^2_d + \underbrace{2\alpha\left\langle\psi^d_{\eta,d-1},\mathring{\psi}^d\right\rangle_d + \alpha^2\left\|\mathring{\psi}^d\right\|^2_d}_{P(\alpha)}.\end{aligned}$$

At $\alpha = 0$, the polynomial $P(\alpha)$ vanishes but its derivative does not. Thus, there exists a small value α^* (less than zero if $P'(0) > 0$ and greater than zero if $P'(0) < 0$) such that $P(\alpha^*)$ is negative. Setting $\tilde{\psi}^d_{\eta,d-1} = \psi^d_{\eta,d-1} + \alpha^*\cdot\mathring{\psi}^d$ we get a function with $\tilde{\psi}^d_{\eta,d-1}(\eta') = \delta_{\eta\eta'}$ for all $\eta' \in \mathcal{N}^{d-1}$, such that:

$$\left\|\tilde{\psi}^d_{\eta,d-1}\right\|^2_d < \left\|\psi^d_{\eta,d-1}\right\|^2_d,$$

contradicting the fact that $\psi^d_{\eta,d-1}$ is the interpolation-constrained minimizer of the energy. □

Claim 2 (Well-Defined)**.** *For a given $\eta \in \mathcal{N}^{d-1}$, the function $\psi^d_{\eta,d-1}$ that satisfies the interpolation conditions $\psi^d_{\eta,d-1}(\eta') = \delta_{\eta\eta'}$ for all $\eta' \in \mathcal{N}^{d-1}$ and minimizes the energy $\|\cdot\|^2_d$ is unique.*

Proof. Suppose that there are two functions $\psi^d_{\eta,d-1}$ and $\tilde{\psi}^d_{\eta,d-1}$ that satisfy the interpolation condition and minimize the energy, with $\|\psi^d_{\eta,d-1}\|^2_d = \|\tilde{\psi}^d_{\eta,d-1}\|^2_d$. We would like to invoke Lemma B.3.3 using $\psi^d = \psi^d_{\eta,d-1} - \tilde{\psi}^d_{\eta,d-1}$. As both functions satisfy $\psi^d_{\eta,d-1}(\eta') = \tilde{\psi}^d_{\eta,d-1}(\eta') = \delta_{\eta\eta'}$ for all $\eta' \in \mathcal{N}^{d-1}$, their difference is equal to zero at all nodes in $\mathcal{N}^{d-1}$ and so by Lemma B.3.1 we have $\psi^d \in \mathrm{Span}(\mathring{\mathcal{B}}^d)$. Thus, if it were the case that $\langle\psi^d_{\eta,d-1},\psi^d\rangle_d \neq 0$, we would have a contradiction to the optimality of $\psi^d_{\eta,d-1}$.
Suppose that $0 = \langle\psi^d_{\eta,d-1},\psi^d\rangle_d = \langle\psi^d_{\eta,d-1},\psi^d_{\eta,d-1} - \tilde{\psi}^d_{\eta,d-1}\rangle_d$. Then we have:

$$\left\|\psi^d_{\eta,d-1}\right\|^2_d = \left\langle\psi^d_{\eta,d-1},\tilde{\psi}^d_{\eta,d-1}\right\rangle_d.$$

But under the assumption that $\psi^d_{\eta,d-1}$ and $\tilde{\psi}^d_{\eta,d-1}$ are both minimizers, we also have $\|\psi^d_{\eta,d-1}\|^2_d = \|\tilde{\psi}^d_{\eta,d-1}\|^2_d$ so that:

$$\begin{aligned}\left\|\psi^d\right\|^2_d &= \left\|\psi^d_{\eta,d-1} - \tilde{\psi}^d_{\eta,d-1}\right\|^2_d \\ &= \left\|\psi^d_{\eta,d-1}\right\|^2_d + \left\|\tilde{\psi}^d_{\eta,d-1}\right\|^2_d - 2\left\langle\psi^d_{\eta,d-1},\tilde{\psi}^d_{\eta,d-1}\right\rangle_d \\ &= 0.\end{aligned}$$

Thus, by Lemma B.3.2, $\psi^d = 0$. Or, equivalently, that $\psi^d_{\eta,d-1} = \tilde{\psi}^d_{\eta,d-1}$. □

Claim 3 (Partition of Unity)**.** *The functions $\mathcal{B}^d_{d-1}$ satisfy the partition of unity property on $\Sigma(\mathcal{C}^d)$. In particular, setting:*

$$\psi^d_{1,d-1} \equiv \sum_{\eta \in \mathcal{N}^{d-1}} \psi^d_{\eta,d-1}$$

we have $\psi^d_{1,d-1} = 1$.

Proof. Recall that the Lagrange basis $\mathcal{B}^d$ satisfies the partition of unity property, so that setting

$$\psi^d_{1,d} \equiv \sum_{\eta \in \mathcal{N}^d} \psi^d_{\eta,d}$$

we have $\psi^d_{1,d} = 1$.
We would like to apply Lemma B.3.3 with $\psi^d = \psi^d_{1,d-1} - \psi^d_{1,d}$. As above, $\psi^d \in \text{Span}(\mathring{\mathcal{B}}^d)$ so that if $\langle \psi^d_{\eta,d-1}, \psi^d \rangle_d \neq 0$ for some $\eta \in \mathcal{N}^{d-1}$ we would have a contradiction to the optimality of $\psi^d_{\eta,d-1}$.
Suppose, to the contrary, that $\langle \psi^d_{\eta,d-1}, \psi^d \rangle_d = 0$ for all $\eta \in \mathcal{N}^{d-1}$. Summing, we get:

$$0 = \sum_{\eta \in \mathcal{N}^{d-1}} \left\langle \psi^d_{\eta,d-1}, \psi^d \right\rangle_d = \left\langle \psi^d_{1,d-1}, \psi^d \right\rangle_d = \left\| \psi^d \right\|^2_d,$$

where the last equality follows from the fact that the $\|\psi^d_{1,d}\|^2_d = 0$. Then using Lemma B.3.2 it follows that $\psi^d = 0$ or, equivalently, that $\psi^d_{1,d-1} = \psi^d_{1,d} = 1$. □

Claim 4 (Linear Precision)**.** *If the cell complex $\mathcal{C}$ (together with the virtual vertices) is embedded in Euclidean space, $\Sigma_0(\mathcal{C}) \subset \mathbb{R}^n$ and if for each d-dimensional cell $c \in \mathcal{C}^d \backslash \mathcal{C}^{d-1}$ the vertices $\Sigma_0(c)$ all lie within a d-dimensional plane, then the basis $\mathcal{B}^d_{d-1}$ has linear precision.*

Proof. We start by observing that, since the functions in $\mathcal{B}^d_{d-1}$ satisfy the interpolation condition $\psi^d_{\eta,d-1}(\eta') = \delta_{\eta\eta'}$ for all $\eta, \eta' \in \mathcal{N}^{d-1}$, the basis has linear precision if and only if for all linear functions $L : \mathbb{R}^n \to \mathbb{R}$, we have:

$$L\big|_{\Sigma(\mathcal{C}^d)} = \psi^d_{L,d-1} \equiv \sum_{\eta \in \mathcal{N}^{d-1}} L(\eta) \cdot \psi^d_{\eta,d-1}.$$

Because the functions in the Lagrange basis $\mathcal{B}^d$ have linear precision, we have:

$$L\big|_{\Sigma(\mathcal{C}^d)} = \psi^d_{L,d} \equiv \sum_{\eta \in \mathcal{N}^d} L(\eta) \cdot \psi^d_{\eta,d}.$$

We would like to invoke Lemma B.3.3 using $\psi^d = \psi^d_{L,d-1} - \psi^d_{L,d}$. As above, $\psi^d \in \text{Span}(\mathring{\mathcal{B}}^d)$ so that if $\langle \psi^d_{\eta,d-1}, \psi^d \rangle_d \neq 0$ for some $\eta \in \mathcal{N}^{d-1}$ we would have a contradiction to the optimality of $\psi^d_{\eta,d-1}$.

Again, supposing to the contrary that $\langle \psi^d_{\eta,d-1}, \psi^d \rangle_d = 0$ for all $\eta \in \mathcal{N}^{d-1}$, we take the weighted sum:

$$\begin{aligned} 0 &= \sum_{\eta \in \mathcal{N}^{d-1}} L(\eta) \cdot \left\langle \psi^d_{\eta,d-1}, \psi^d \right\rangle_d \\ &= \sum_{\eta \in \mathcal{N}^{d-1}} L(\eta) \cdot \left\langle \psi^d_{\eta,d-1}, \psi^d_{L,d-1} - \psi^d_{L,d} \right\rangle_d \\ &= \left\langle \psi^d_{L,d-1}, \psi^d_{L,d-1} - \psi^d_{L,d} \right\rangle_d . \end{aligned}$$

Or, equivalently:

$$\left\| \psi^d_{L,d-1} \right\|^2_d = \left\langle \psi^d_{L,d-1}, \psi^d_{L,d} \right\rangle_d .$$

Using the Cauchy-Schwarz inequality it follows that:

$$\left\| \psi^d_{L,d-1} \right\|^2_d \leq \left\| \psi^d_{L,d} \right\|^2_d .$$

On the other hand, since each d-dimensional cell $c \in \mathcal{C}^d \backslash \mathcal{C}^{d-1}$ lies within a d-dimensional plane, we know that the restriction of the function L to c has continuous gradient. Thus, we have:

$$\left\| \psi^d_{L,d} \right\|^2_d = \sum_{\sigma \in \Sigma_d(\mathcal{C}^d)} \varepsilon \cdot \int_\sigma \left\| \nabla \psi^d_{L,d} \right\|^2 d\sigma.$$

Furthermore, since $\psi^d_{L,d}$ is linear, it is harmonic within each cell $c \in \mathcal{C}^d \backslash \mathcal{C}^{d-1}$. Thus, of all continuous functions that agree with L on $\Sigma(\mathcal{C}^{d-1})$, the function $\psi^d_{L,d}$ is the one minimizing the Dirichlet energy. In particular, since $\psi^d_{L,d-1}$ and $\psi^d_{L,d}$ agree with L on $\Sigma(\mathcal{C}^{d-1})$, we have:

$$\begin{aligned} \left\| \psi^d_{L,d} \right\|^2_d &= \sum_{\sigma \in \Sigma_d(\mathcal{C}^d)} \varepsilon \cdot \int_\sigma \left\| \nabla \psi^d_{L,d} \right\|^2 \\ &\leq \sum_{\sigma \in \Sigma_d(\mathcal{C}^d)} \varepsilon \cdot \int_\sigma \left\| \nabla \psi^d_{L,d-1} \right\|^2 \leq \left\| \psi^d_{L,d-1} \right\|^2_d . \end{aligned}$$

Combining the two inequalities, we get:

$$\left\| \psi^d_{L,d-1} \right\|^2_d = \left\| \psi^d_{L,d} \right\|^2_d = \left\langle \psi^d_{L,d-1}, \psi^d_{L,d} \right\rangle_d .$$

As above, this implies that:

$$\left\| \psi^d_{L,d-1} - \psi^d_{L,d} \right\|^2_d = 0$$

and using Lemma B.3.2 it follows that $\psi^d_{L,d-1} = \psi^d_{L,d} = L|_{\Sigma(\mathcal{C}^d)}$. □

B.4 GENERAL PROLONGATION

The basis $\mathcal{B}^d_{d-1}$ consists of functions indexed by nodes $\eta \in \mathcal{N}^{d-1}$ that are defined on $\Sigma(\mathcal{C}^d)$. We now describe an approach for generalizing this formulation, defining a basis $\mathcal{B}^d_{d'}$ for all $0 \leq d' < d \leq D$, where the basis functions are indexed by nodes $\eta' \in \mathcal{N}^{d'}$ and defined on $\Sigma(\mathcal{C}^d)$. As above, we show that these bases can be expressed as the solution to a constrained minimization problem, satisfy the partition of unity property, and (under appropriate conditions) satisfy the partition of unity property.

Definition. Given the basis $\mathcal{B}^d_{d-1}$, we define the prolongation matrix $\mathbf{P}^d_{d-1} \in \mathbb{R}^{|\mathcal{N}^d| \times |\mathcal{N}^{d-1}|}$ to be the matrix whose coefficients give the expression of $\psi^d_{\eta',d-1}$ as a linear combination of the $\psi^d_{\eta,d}$:

$$\psi^d_{\eta',d-1} = \sum_{\eta \in \mathcal{N}^d} \left(\mathbf{P}^d_{d-1}\right)_{\eta\eta'} \psi^d_{\eta,d}.$$

Definition. For $0 \leq d' < d \leq D$, we define the prolongation matrix $\mathbf{P}^d_{d'}$ to be the composition:

$$\mathbf{P}^d_{d'} = \mathbf{P}^d_{d-1} \cdots \mathbf{P}^{d'+1}_{d'}.$$

Definition. For $0 \leq d' < d \leq D$ we define $\mathcal{B}^d_{d'} = \{\psi^d_{\eta',d'}\}_{\eta' \in \mathcal{N}^{d'}}$ to be the subset of Span$(\mathcal{B}^d)$ such that:

$$\psi^d_{\eta',d'} \equiv \sum_{\eta \in \mathcal{N}^d} \left(\mathbf{P}^d_{d'}\right)_{\eta\eta'} \cdot \psi^d_{\eta,d}.$$

We note that for all d' with $0 \leq d' < d$ we have $\mathcal{B}^d_{d'} \subset \text{Span}(\mathcal{B}^d_{d-1})$. This motivates the following claim.

Claim 5 (Optimality). *Given $0 \leq d' < d \leq D$ and given $\eta' \in \mathcal{N}^{d'}$, if $\tilde{\psi}^d_{\eta',d'} \in$ Span$(\mathcal{B}^d)$ is a function that agrees with $\psi^d_{\eta',d'}$ on $\Sigma(\mathcal{C}^{d-1})$ then we have:*

$$\left\|\psi^d_{\eta',d'}\right\|^2_d \leq \left\|\tilde{\psi}^d_{\eta',d'}\right\|^2_d,$$

with equality if and only if $\psi^d_{\eta',d'} = \tilde{\psi}^d_{\eta',d'}$. Thus, of all functions in Span$(\mathcal{B}^d)$ that agree with $\psi^d_{\eta',d'}$ on $\Sigma(\mathcal{C}^{d-1})$ the function $\psi^d_{\eta',d'}$ is the unique minimizer of the energy $\|\cdot\|^2_d$.

Proof. Consider the difference $\psi^d = \tilde{\psi}^d_{\eta',d'} - \psi^d_{\eta',d'}$. As $\tilde{\psi}^d_{\eta',d'}$ and $\psi^d_{\eta',d'}$ agree on $\Sigma(\mathcal{C}^{d-1})$ we have $\psi^d \in \text{Span}(\mathring{\mathcal{B}}^d)$.

Expanding the energy of $\tilde{\psi}^d_{\eta',d'}$ we get:

$$\begin{aligned}\left\|\tilde{\psi}^d_{\eta',d'}\right\|^2_d &= \left\|\psi^d_{\eta',d'} + \psi^d\right\|^2_d \\ &= \left\|\psi^d_{\eta',d'}\right\|^2_d + \left\|\psi^d\right\|^2_d + 2\left\langle\psi^d_{\eta',d'}, \psi^d\right\rangle_d \\ &= \left\|\psi^d_{\eta',d'}\right\|^2_d + \left\|\psi^d\right\|^2_d\end{aligned}$$

where the last equality follows from Lemma B.3.3 – using the fact that $\psi^d_{\eta',d'} \in \text{Span}(\mathcal{B}^d_{d-1})$ and $\psi^d \in \text{Span}(\mathring{\mathcal{B}}^d)$. Thus the energy of $\psi^d_{\eta',d'}$ is no greater than the energy of $\tilde{\psi}^d_{\eta',d'}$, with equality if and only if the energy of ψ^d is zero. But since $\psi^d \in \text{Span}(\mathring{\mathcal{B}}^d)$, Lemma B.3.2 implies that the energy of ψ^d vanishes if and only if $\psi^d = 0$ or, equivalently, if and only if $\psi^d_{\eta',d'} = \tilde{\psi}^d_{\eta',d'}$. □

Claim 6 (Partition of Unity)**.** *If the basis $\mathcal{B}^{d-1}_{d'}$ satisfies the partition of unity property, then so does the basis $\mathcal{B}^d_{d'}$.*

Proof. Consider the functions:

$$\psi^d_{1,d'} \equiv \sum_{\eta' \in \mathcal{N}^{d'}} \psi^d_{\eta',d'} \in \mathcal{B}^d \qquad \text{and} \qquad \psi^d_{1,d} \equiv \sum_{\eta \in \mathcal{N}^d} \psi^d_{\eta,d}.$$

We have $\psi^d_{1,d'} \in \text{Span}(\mathcal{B}^d_{d-1})$ and, by the assumption of the claim, the functions $\psi^d_{1,d'}$ and $\psi^d_{1,d}$ are both constantly equal to one on $\Sigma(\mathcal{C}^{d-1})$. By Claim 5 we have:

$$\left\|\psi^d_{1,d'}\right\|^2_d \leq \left\|\psi^d_{1,d}\right\|^2_d = 0.$$

Thus, we must have $\|\psi^d_{1,d'}\|^2_d = \|\psi^d_{1,d}\|^2_d$ so, by the claim, the two functions are equal – $\psi^d_{1,d'} = \psi^d_{1,d} = 1$. □

Corollary B.4.0.1. *Using the fact that $\mathcal{B}^{d'+1}_{d'}$ satisfies the partition of unity property, it follows that $\mathcal{B}^d_{d'}$ satisfies the partition of unity property.*

Claim 7 (Linear Precision)**.** *If the basis $\mathcal{B}^{d-1}_{d'}$ satisfies the linear precision property and every d-dimensional cell $c \in \mathcal{C}^d \backslash \mathcal{C}^{d-1}$ has the property that the vertices of its simplicial refinement $\Sigma_0(c)$ lie in a d-dimensional plane, then the basis $\mathcal{B}^d_{d'}$ also has linear precision.*

Proof. Let $L : \mathbb{R}^n \to \mathbb{R}$ be a linear function and consider the functions $\psi^d_{L,d'}$ and $\psi^d_{L,d}$:

$$\begin{aligned}\psi^d_{L,d'} &\equiv \sum_{\eta' \in \mathcal{N}^{d'}} L(\eta') \cdot \psi^d_{\eta',d'} \in \mathcal{B}^d_{d'} \\ \psi^d_{L,d} &\equiv \sum_{\eta \in \mathcal{N}^d} L(\eta) \cdot \psi^d_{\eta,d} = L|_{\Sigma(\mathcal{C}^d)}.\end{aligned}$$

By the assumption of the claim we know that $\psi^d_{L,d'}$ agrees with $\psi^d_{L,d}$ on the simplicial complex $\Sigma(\mathcal{C}^{d-1})$. Thus, by Claim 5 we know that:

$$\left\|\psi^d_{L,d'}\right\|^2_d \leq \left\|\psi^d_{L,d}\right\|^2_d .$$

On the other hand, we know that of all functions agreeing with L on $\Sigma(\mathcal{C}^{d-1})$, the function $\psi^d_{L,d}$ minimizes $\|\cdot\|^2_d$ since it is C^1 and harmonic. Thus the two functions must have the same energy, $\|\psi^d_{L,d'}\|^2_d = \|\psi^d_{L,d}\|^2_d$ so, by the claim, the functions are equal – $\psi^d_{L,d'} = \psi^d_{L,d} = L|_{\Sigma(\mathcal{C}^d)}$. $\square$

Corollary B.4.0.2. *Given $1 \leq d \leq D$, if for every $1 \leq d' \leq d$, every d'-th dimensional cell $c \in \mathcal{C}^{d'} \backslash \mathcal{C}^{d'-1}$ has the property that the vertices of the simplicial refinement $\Sigma_0(c)$ lie in a d'-dimensional plane, then the basis $\mathcal{B}^d_{d'}$ has linear precision.*

b.5 DISCUSSION

The Need for gradient continuity

In these derivations we only used the Dirichlet regularizer in the definition of the energies $\mathcal{Q}_d(\cdot)$ and could have foregone the gradient-continuity term, as all the functions we ended up considering had continuous gradients in any case. The reason we incorporate the gradient continuity energy is to prove that the basis $\mathcal{B}^d_{d-1}$ reproduces the Lagrange basis in the case that $\mathcal{C}$ is a simplicial complex. That is, that our work generalizes the standard Lagrange basis. In that case the reproduction proof holds when the energy is defined entirely in terms of gradient continuity (i.e. when $\varepsilon = 0$). Thus, in practice our works assumes that ε is an arbitrarily small, but non-zero, value. It needs to be non-zero for the proofs described here to hold. It needs to be arbitrary small so that we come arbitrarily close to reproducing the Lagrange basis in the case that the cell complex is a simplicial complex. (One could make the reproduction exact by identifying the kernel of the gradient continuity energy and only adding the Dirichlet energy of the projection onto the kernel. However, this would require identifying the kernel for each cell, making the implementation slower in practice.)

The Order of the Shape Functions

Though our research focuses specifically on quadratic Lagrange elements, the proofs above only assume that the order is at least one, so that we are working within a continuous function space.

The Choice of Metric

The definition of $\mathcal{Q}_d$ requires the choice of a metric in order to define gradients and compute integrals. In general, our implementation only requires an assignment of a symmetric positive definite matrix with each D-dimensional simplex of the refinement, $\sigma \in \Sigma_D(\mathcal{C})$.
However, in the case that the cell complex $\mathcal{C}$ is embedded in $\mathbb{R}^n$ and we require linear precision, we assume that the metric is induced by the embedding. This ensures that the restriction of linear functions to the cell complex will have continuous gradients.
We note that the piecewise-constant assignment of metric tensors to simplices does not ensure that the induced metric on boundary simplices is consistently defined for face-adjacent simplices. This *is* the case when the metric is induced by the embedding of cell complex in $\mathbb{R}^n$ but need not be true in general. In our application of anisotropic diffusion (for line integral convolution) this is not an issue, but it could be in other contexts.[1] One can bypass this problem by defining the metric by assigning lengths to all edges of the simplicial complex. This ensures metrics are consistently defined on boundary faces, but has the limitation that the assignment of positive edge weights must satisfy the triangle inequality – a nonlinear constraint on the set of edge weights.

Planarity Testing

As discussed above, if the cell complex is embedded in Euclidean space, $\Sigma_0(\mathcal{C}) \subset \mathbb{R}^N$, and we would like to have linear precision, we only need to explicitly impose linear precision constraints for those d-dimensional cells $c \in \mathcal{C}$ whose vertices do not lie within a d-dimensional plane. This can be checked by constructing the characteristic polynomial of the covariance matrix of the vertices of c and checking that the lowest $N - d$ coefficients are zero. That is that is, that the characteristic polynomial $P(x)$ is divisible by x^{N-d}.

1 Our approach supports this metric discontinuity, with the implication being that gradient discontinuity can be manifest along the virtual face, not just across it.

APPENDIX C

C.1 QUADRATIC PRECISION OF THE BASIS $\mathcal{B}^d_{d-1}$

This section is an addition to the proofs listed in Appendix B. We follow the line of argumentation that was used to prove the linear precision property of our basis functions (see Chapter 5) to show that we also retain the desired quadratic precision. Given a domain embedded in Euclidean space, $\Omega \subset \mathbb{R}^n$, we say that a set of functions $\{\varphi_i\} : \Omega \to \mathbb{R}$, has quadratic precision if for every quadratic function $Q : \mathbb{R}^n \to \mathbb{R}$, there exist coefficients $\{a_i\} \subset \mathbb{R}$ with:

$$Q\big|_\Omega = \sum_i a_i \cdot \varphi_i.$$

In what follows, we once again assume that all complexes are *pure*. In other words, if the complex is D-dimensional, then every d-dimensional simplex/cell (with $d < D$) is on the boundary of some $(d+1)$-dimensional simplex/cell.

Claim 8 (Quadratic Precision)**.** *If the cell complex $\mathcal{C}$ (together with the virtual vertices) is embedded in Euclidean space, $\Sigma_0(\mathcal{C}) \subset \mathbb{R}^n$ and if for each d-dimensional cell $c \in \mathcal{C}^d \backslash \mathcal{C}^{d-1}$ the vertices $\Sigma_0(c)$ all lie within a d-dimensional plane, then the basis $\mathcal{B}^d_{d-1}$ has quadratic precision.*

Proof. The first part of the proof is almost analogous to the one provided for the linear precision property. Since the functions in $\mathcal{B}^d_{d-1}$ satisfy the interpolation condition $\psi^d_{\eta,d-1}(\eta') = \delta_{\eta\eta'}$ for all $\eta, \eta' \in \mathcal{N}^{d-1}$, the basis has quadratic precision if and only if for all quadratic functions $Q : \mathbb{R}^n \to \mathbb{R}$, we have:

$$Q\big|_{\Sigma(\mathcal{C}^d)} = \psi^d_{Q,d-1} \equiv \sum_{\eta\in\mathcal{N}^{d-1}} Q(\eta) \cdot \psi^d_{\eta,d-1}.$$

The functions in the Lagrange basis $\mathcal{B}^d$ have quadratic precision, and therefore satisfy:

$$Q\big|_{\Sigma(\mathcal{C}^d)} = \psi^d_{Q,d} \equiv \sum_{\eta\in\mathcal{N}^{d}} Q(\eta) \cdot \psi^d_{\eta,d}.$$

We apply Lemma B.3.3 to the difference between the two interpolated quadratic functions $\psi^d = \psi^d_{Q,d-1} - \psi^d_{Q,d}$. As before, since both interpolations agree on the boundary vertices, we have $\psi^d \in \text{Span}(\mathring{\mathcal{B}}^d)$ so that $\langle \psi^d_{\eta,d-1}, \psi^d \rangle_d = 0$ for all $\eta \in \mathcal{N}^{d-1}$. If that were not the case, meaning $\langle \psi^d_{\eta,d-1}, \psi^d \rangle_d \neq 0$ for some $\eta \in \mathcal{N}^{d-1}$, we would have a contradiction to the optimality of $\psi^d_{\eta,d-1}$.

Therefore, supposing that $\langle \psi^d_{\eta,d-1}, \psi^d \rangle_d = 0$ for all $\eta \in \mathcal{N}^{d-1}$, we take the weighted sum:

$$\begin{aligned} 0 &= \sum_{\eta \in \mathcal{N}^{d-1}} Q(\eta) \cdot \left\langle \psi^d_{\eta,d-1}, \psi^d \right\rangle_d \\ &= \sum_{\eta \in \mathcal{N}^{d-1}} Q(\eta) \cdot \left\langle \psi^d_{\eta,d-1}, \psi^d_{Q,d-1} - \psi^d_{Q,d} \right\rangle_d \\ &= \left\langle \psi^d_{Q,d-1}, \psi^d_{Q,d-1} - \psi^d_{Q,d} \right\rangle_d \\ &= \left\langle \psi^d_{Q,d-1}, \psi^d_{Q,d-1} \right\rangle_d - \left\langle \psi^d_{Q,d-1}, \psi^d_{Q,d} \right\rangle_d \\ &= \left\| \psi^d_{Q,d-1} \right\|^2_d - \left\langle \psi^d_{Q,d-1}, \psi^d_{Q,d} \right\rangle_d . \end{aligned}$$

Or, equivalently:

$$\left\| \psi^d_{Q,d-1} \right\|^2_d = \left\langle \psi^d_{Q,d-1}, \psi^d_{Q,d} \right\rangle_d .$$

Using the Cauchy-Schwarz inequality

$$| \langle \mathbf{u}, \mathbf{v} \rangle_d | \leq \| \mathbf{u} \|_d \, \| \mathbf{v} \|_d$$

it follows that:

$$\begin{aligned} & \left\| \psi^d_{Q,d-1} \right\|^2_d = \left\langle \psi^d_{Q,d-1}, \psi^d_{Q,d} \right\rangle_d \leq \left\| \psi^d_{Q,d-1} \right\|_d \left\| \psi^d_{Q,d} \right\|_d \\ \Leftrightarrow & \left\| \psi^d_{Q,d-1} \right\|^2_d \leq \left\| \psi^d_{Q,d} \right\|^2_d \end{aligned}$$

On the other hand, since each d-dimensional cell $c \in \mathcal{C}^d \backslash \mathcal{C}^{d-1}$ lies within a d-dimensional plane, we know that the restriction of the function Q to c has continuous gradients. Thus, we have:

$$\left\| \psi^d_{Q,d} \right\|^2_d = \sum_{\sigma \in \Sigma_d(\mathcal{C}^d)} \varepsilon \cdot \int_\sigma \left\| \nabla \psi^d_{Q,d} \right\|^2 d\sigma.$$

So there exists an interpolated quadratic function $\psi^d_{Q,d-1}$ with lower energy than the exact interpolation $\psi^d_{Q,d}$. However, the Dirichlet energy does not vanish for quadratic functions. Therefore, it remains to be shown that no other interpolation defined by the coarse nodes of the polygon yields a lower Dirichlet energy than the exact function. However, it potentially has a non-zero gradient-continuity energy instead. In this case, the chosen prolongation weights might differ from those needed for interpolating the exact quadratic function. Here, we run into a similar problem as discussed in Section B.5. If our energy were entirely defined in terms of gradient continuity, meaning $\varepsilon = 0$, we could finish the proof, and the interpolated function would be exact. However, we must choose a small $\varepsilon = 10^{-8}$ to guarantee the properties listed in Appendix B.

While this also makes the space of potentially interpolated functions with non-continuous gradients but lower Dirichlet energies arbitrarily small, we come only arbitrarily close to the quadratic precision property, depending on the chosen ε. So, this thesis has no *exact* proof of quadratic precision. Regardless, the potential is there. A different set of constraints instead of the Dirichlet regularizer might exist that can yield functions that satisfy all the properties presented in Appendix B and C. However, this remains to be shown.

$\square$