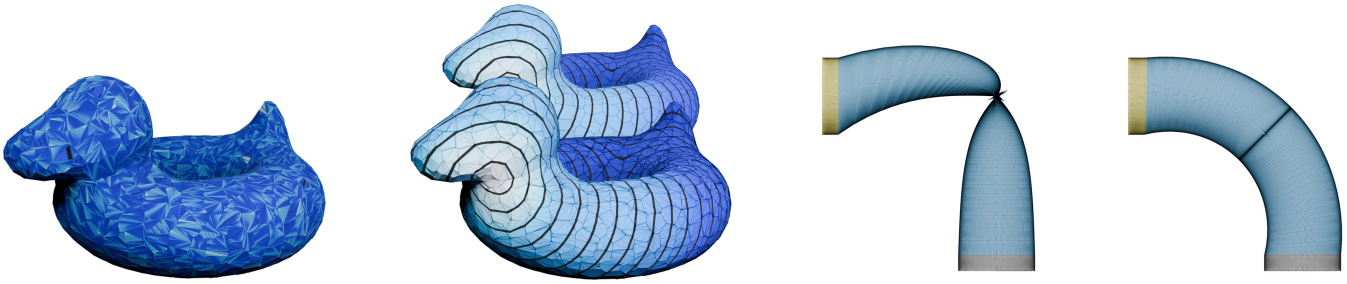
Vision, Modeling, and Visualization (2025)
B. Egger and T. Günther (Eds.)

Robust Discrete Differential Operators for Wild Geometry

S. D. Wagner[1] and M. Botsch[1,2]

[1]TU Dortmund University, Germany
[2]Lamarr Institute for Machine Learning and Artificial Intelligence, Dortmund, Germany



(a) *Standard differential operators* **(b)** *Our robust differential operators* **(c)** *Standard operators* **(d)** *Our robust operators*

Figure 1: *Standard differential operators [MDSB03] fail for degenerate or near-degenerate elements, shown here for geodesic distances [CWW13] on a model generated with VoroMesh [MKO*23] (a) and for gradient-based bending [YZX*04] of a cylinder with degenerate mid-section (c). In contrast, our proposed differential operators work robustly in these challenging cases (b, d).*

Abstract

Many geometry processing algorithms rely on solving PDEs on discrete surface meshes. Their accuracy and robustness crucially depend on the mesh quality, which oftentimes cannot be guaranteed – in particular when automatically processing geometries extracted from arbitrary implicit representations. Through extensive numerical experiments, we evaluate the robustness of various Laplacian implementations across geometry processing libraries on synthetic and "in-the-wild" surface meshes with degenerate or near-degenerate elements, revealing their strengths, weaknesses, and failure cases. To improve numerical stability, we extend the recently proposed tempered finite elements method (TFEM) to meshes with strongly varying element sizes, to arbitrary polygonal elements, and to gradient and divergence operators. Our resulting differential operators are simple to implement, efficient to compute, and robust even in the presence of fully degenerate mesh elements.

CCS Concepts

• ***Mathematics of computing*** → ***Discretization;*** • ***Computing methodologies*** → ***Mesh geometry models;***

1. Introduction

The discrete Laplace-Beltrami operator is one of the most ubiquitous operators in geometry processing, being employed in smoothing, distance computation, deformation, and many more [BKP*10]. It can be used to model many physical phenomena on two-manifolds, such as heat diffusion or wave propagation. Because of its geometric importance, there exists much research regarding properties of the operator [WMKG07] and many different discretization schemes for triangular and polygonal surfaces [Dzi88, PP93, MDSB03, BS07, SC20, BB23].

With the emergence of deep learning, the need for unsupervised processing of massive amounts of data has become a prevalent topic. However, the quality of the data to be processed cannot always be guaranteed. For instance, when processing datasets of meshes, even the most basic preconditions, such as non-degenerate geometry, can often not be guaranteed [ZJ16]. Furthermore, learning implicit representations of shapes [PFS*19, BRV*24] can yield unpredictable results when extracting meshes with Marching Cubes [LC87], resulting in them not necessarily conforming to the preconditions of typical geometric algorithms.

To investigate the resilience of modern general-purpose geometry processing libraries to degenerate geometry, we benchmark the robustness of the standard cotangent Laplace-Beltrami discretization [PP93, MDSB03] as implemented in multiple geometry

processing libraries [BSBK02, SC*19, SB23, The24, Vis23, Liv19, L*25, JP*18] and distill a robust implementation. Furthermore, we go beyond linear FEM and review alternative, more robust Laplace discretizations. We eventually build upon the recent tempered finite elements method (TFEM) [QKL*24], which we extend

- to be locally and globally scaling invariant, thereby being applicable to irregular, non-homogeneous meshes,
- to a consistent discretization of gradient and divergence operators, thereby enabling a wider range of geometry processing algorithms, and
- to work for general polygon meshes [BHKB20], thereby lifting the restriction to pure triangle meshes.

Our proposed operators act as a drop-in replacement of the standard cotangent discretization, are easy to implement, efficient to compute, and robustly work when all others fail. The code for this benchmark and the improved differential operators is available at `https://github.com/sdwagner/wildDDG`.

2. Related Work

Laplacians on Triangle Meshes Discretizing smooth differential operators, such as the Laplace-Beltrami operator, on discrete surfaces has been an active area of research in the last decades. Each approach tries to retain crucial properties of the smooth operators, such as linear precision, symmetry, or locality [WMKG07]. The most ubiquitous discretization on triangle meshes, the cotangent Laplacian, can be derived using many different techniques, such as the linear finite elements method (FEM) [Dzi88], discrete exterior calculus (DEC) [CdGDS13], minimal surfaces [PP93], or mean curvature flow [DMSB99]. Although it fulfills most of the desired properties, it does not always satisfy the maximum principle, as indicated by negative cotangent weights, which can result in non-physical behavior or flipped faces in discrete harmonic parameterizations [FH05].

To guarantee the maximum principle, Bobenko and Springborn define a Laplacian on the intrinsic Delaunay triangulation (iDT) [BS07], which is built using intrinsic edge flips. These edge flips guarantee that the input geometry stays the same, only combinatorially changing the mesh, while providing positive cotangent weights. This can be efficiently calculated and stored using, e.g., the signpost data structure [SSC19]. To improve upon the iDT Laplacian, Sharp and Crane [SC20] generalize it to non-manifold triangle meshes, constructing a manifold shell around it and then computing the iDT Laplacian. They further propose intrinsic mollification to reduce the impact of (near-)degenerate geometry, intrinsically elongating all edges until every triangle is non-degenerate.

Numerical Stability on Triangle Meshes Investigating the numerical stability of linear FEM has been a key research interest, with several works suggesting error bounds, quality measures [She02, SBMS23] and angle conditions [Zlá68, BA76] to estimate the error of FEM schemes via the shape of the underlying triangles. Generally, triangles with small angles (needles), large angles (caps), and very small areas should be avoided. Yet, the small angle condition derived by Zlámal [Zlá68] and the large angle condition derived by Babuška and Aziz [BA76] were shown not to be necessary conditions for FEM convergence [BA76, HKK12] and using needle elements can in some cases be beneficial for solving PDEs with highly anisotropic coefficients [Rip92]. Still, avoiding small and large angles can have other benefits related to the stiffness matrix conditioning, gradient error convergence [She02], or ensuring the maximum principle [WMKG07]. Furthermore, Kučera [Kuč16] shows that large clusters or bands of degenerate elements, i.e., many adjacent degenerate elements forming a strip, can become problematic for FEM performance.

Handling Degenerate Triangles As poorly shaped triangles are detrimental for many applications, several strategies exist to cope with these elements. While there are many remeshing strategies to improve the quality of triangular meshes [BKP*10, Chapter 6], most approaches change the complexity of the mesh or shift vertices, resulting in a need for non-trivial interpolation of vertex-based quantities. Furthermore, applying global remeshing can be very expensive depending on the quality criterion [SC20]. Another possibility to cope with poorly shaped elements is to use higher-order FEM approaches, as they are better suited for these elements [SHD*18]. Schneider et al. propose to adaptively increase the order of the FEM bases per element based on an a priori error estimate. While this approach can significantly reduce the error introduced by poorly shaped elements, the required computation time scales with the degree of degeneracy and results in the need to solve a considerably larger linear system. Furthermore, very degenerate triangles can still become problematic. In a recent paper, Quiriny et al. [QKL*24] propose to modify the standard FEM scheme and introduce the tempered finite elements method (TFEM), reducing the impact of multiple adjacent cap-like elements to the convergence of FEM by applying clamping to the Jacobian determinant of the geometric map used when defining the stiffness matrix. This approach and our extensions will be discussed in greater detail in Section 4.

Laplacians on Polygon Meshes When extending the scope to include arbitrary two-manifold polygon meshes, there exist several different discretizations, since there is no canonical definition of a surface, as is the case for triangle meshes. We refer to the comprehensive survey of Bunge and Botsch [BB23] for more details on the different approaches. The Laplacian of Bunge et al. [BHKB20] provides a relatively straightforward approach, defining a virtual triangulation and using the cotangent Laplacian as a basis to define the polygonal operator. In a follow-up paper, Bunge et al. [BBW*24] improve the numerical stability of this polygonal operator. We will discuss how we adapt these approaches to generalize our TFEM discretization to polygons in Section 4.3.

3. Laplacian on Triangle Meshes

This section provides a short review of the basic definitions regarding the cotangent Laplacian on triangle meshes [Dzi88, PP93, DMSB99]. Let $\mathcal{M} = (\mathcal{V}, \mathcal{T})$ be a triangle mesh consisting of vertices $\mathcal{V}$ and triangles $\mathcal{T}$ with their respective cardinalities $|\mathcal{V}|$ and $|\mathcal{T}|$. For each vertex $v_i \in \mathcal{V}$, we define a position $\mathbf{x}_i \in \mathbb{R}^3$. In linear FEM, the strong form of the cotangent Laplacian

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

can be defined by using a mass matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ and stiffness matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. The matrices are defined as

$$\mathbf{S}_{ij} = \begin{cases} -\dfrac{\cot\alpha_{ij} + \cot\beta_{ij}}{2} & \text{if } v_j \in \mathcal{N}(v_i), \\ -\displaystyle\sum_{v_k \in \mathcal{N}(v_i)} \mathbf{S}_{ik} & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

$$\mathbf{M}_{ij} = \begin{cases} \dfrac{|t_{ijk}| + |t_{ijl}|}{12} & \text{if } v_j \in \mathcal{N}(v_i), \\ \displaystyle\sum_{v_k \in \mathcal{N}(v_i)} \mathbf{M}_{ik} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Here, α_{ij} and β_{ij} are the two angles opposing the edge (v_i, v_j), $|t_{ijk}|$ and $|t_{ijl}|$ are the areas of the triangles adjacent to this edge, and $\mathcal{N}(v_i)$ refers to the one-ring vertex neighborhood of vertex v_i. Note that the mass matrix is typically lumped by summing up the entries per row, resulting in the usual barycentric mass matrix.

Since the continuous Laplacian is defined as divergence of gradient, we can also decompose the stiffness matrix into the respective gradient and divergence matrices $\mathbf{G} \in \mathbb{R}^{3|\mathcal{T}|\times|\mathcal{V}|}$, $\mathbf{D} \in \mathbb{R}^{|\mathcal{V}|\times 3|\mathcal{T}|}$. The gradient matrix can be defined blockwise using the matrices $\mathbf{G}^i \in \mathbb{R}^{3\times|\mathcal{V}|}$ for each triangle $t_i = (v_j, v_k, v_l) \in \mathcal{T}$ according to

$$\mathbf{G}^i_{:j} = \begin{cases} \dfrac{(\mathbf{x}_l - \mathbf{x}_k)^\perp}{2|t_i|} & \text{if } v_j \in t_i, \\ \mathbf{0} & \text{otherwise,} \end{cases} \tag{3}$$

$$\mathbf{D} = \mathbf{G}^\top \mathbf{M}_D, \tag{4}$$

where $\mathbf{v}^\perp$ denotes a counterclockwise rotation of the vector $\mathbf{v}$ by 90° in the triangle plane and $\mathbf{M}_D \in \mathbb{R}^{3|\mathcal{T}|\times 3|\mathcal{T}|}$ is the diagonal divergence mass matrix consisting of the triangle areas, each repeated three times. Multiplying these definitions results in the cotangent stiffness matrix $\mathbf{S} = \mathbf{DG}$, and thus $\mathbf{L} = -\mathbf{M}^{-1}\mathbf{DG}$.

3.1. Ways to Compute the Laplacian

While calculating the corner angles of the triangles and applying the cotangent is the easiest way to compute the cotangent weights, several mathematical identities can make the calculations more performant and numerically stable. This section investigates the different ways to calculate the cotangent Laplacian present in modern geometry processing libraries and compares their robustness. We focus exclusively on general-purpose libraries, as these are commonly used to process geometric data.

Trigonometry Trigonometry is still the most widely used method of calculation, but there are several different variations. OpenMesh [BSBK02], Geogram [L*25], VCGLib [Vis23], and CinoLib [Liv19] each first calculate the corner angles using the standard cosine identity

$$\alpha_i = \arccos\left(\frac{\langle \mathbf{e}_{ij}, \mathbf{e}_{ik}\rangle}{\|\mathbf{e}_{ij}\| \cdot \|\mathbf{e}_{ik}\|}\right) \tag{5}$$

at vertex v_i and triangle $t_{ijk} = (v_i, v_j, v_k)$, where $\mathbf{e}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ is the edge vector from vertex v_i to v_j. Following this, there exist several variants to compute the cotangent using this angle. For instance, OpenMesh and Geogram compute the cotangent as $\cot\alpha_i = 1/\tan\alpha_i$, while VCGLib uses the equivalent expression $\cot\alpha_i = \cos\alpha_i/\sin\alpha_i$. CinoLib, on the other hand, computes the cotangent via the identity $\cot\alpha_i = \tan(\pi/2 - \alpha_i)$.

Extrinsic Computation Expanding the definitions of the trigonometric functions to vector calculus leads to

$$\cot\alpha_i = \frac{\langle \mathbf{e}_{ij}, \mathbf{e}_{ik}\rangle}{\|\mathbf{e}_{ij} \times \mathbf{e}_{ik}\|}, \tag{6}$$

which is the definition used by CGAL [The24] and geometry-central [SC*19] to calculate the cotangent weights.

Intrinsic Computation As the cotangent Laplacian is an intrinsic operator, only depending on edge lengths and connectivity, the above equation can also be rewritten further:

$$\langle \mathbf{e}_{ij}, \mathbf{e}_{ik}\rangle = \frac{a^2 + b^2 - c^2}{2}, \tag{7}$$

$$\|\mathbf{e}_{ij} \times \mathbf{e}_{ik}\| = 2|t_{ijk}| = 2\sqrt{s\cdot(s-a)\cdot(s-b)\cdot(s-c)}, \tag{8}$$

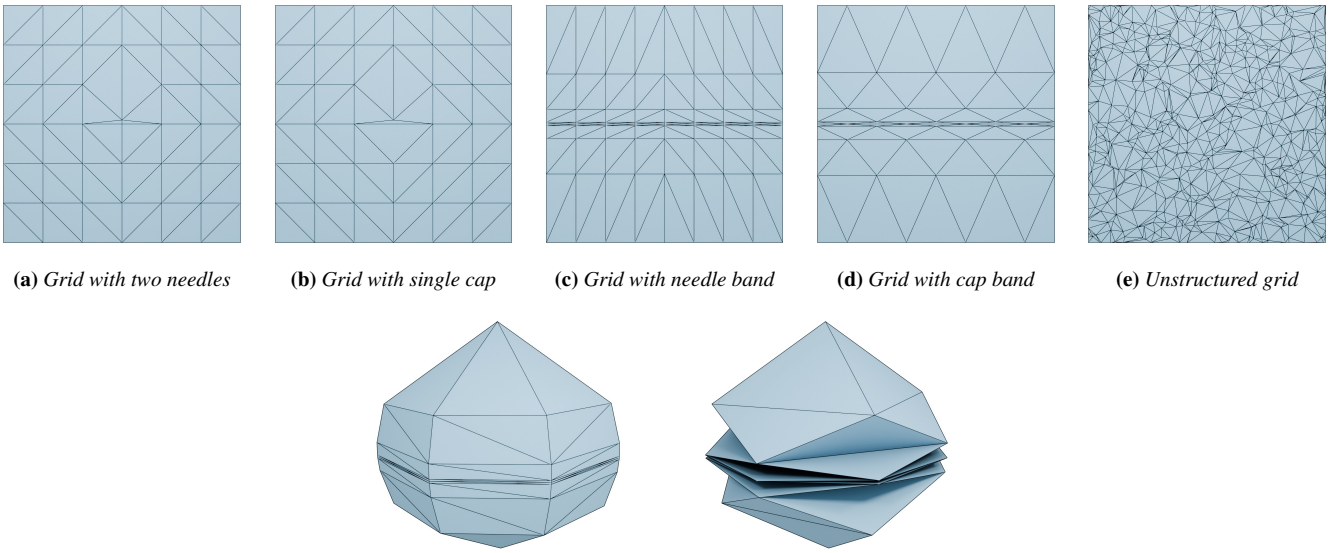
with $a = \|\mathbf{e}_{ij}\|$, $b = \|\mathbf{e}_{ik}\|$, $c = \|\mathbf{e}_{jk}\|$, and $s = \frac{a+b+c}{2}$. This computation is used by PMP [SB23], libigl [JP*18], and by geometry-central [SC*19] when explicitly using intrinsic meshes. To improve the numerical stability of the area computation, libigl further uses a sorted variant of Heron's formula (8), such that $a \geq b \geq c$ [Kah97].

Clamping Although most libraries do not clamp any weights, several options exist to remove unwanted entries from the stiffness matrix. CGAL and PMP ignore cotangent computations whenever the area of the corresponding triangle is zero to avoid dividing by zero. This improves stability in the case of truly degenerate faces. In addition, PMP and CinoLib implement another approach, the clamping of negative entries, which is a sufficient condition to guarantee the maximum principle, but might result in the violation of other properties (e.g., linear precision) [WMKG07]. While PMP provides this optionally and per matrix entry, CinoLib clamps each cotangent computation to 10^{-10}, thus resulting in a more conservative and less correct result.

Building our own We also condense the above-mentioned methods into one extrinsic implementation (6) and one intrinsic implementation (7), (8). Generally, we ignore all cotangent computations on zero-area triangles to prevent division by zero. Furthermore, we use the sorted version of Heron's formula (8) in our intrinsic implementation.

3.2. Comparison

To assess the robustness of the different implementations to poorly shaped and near-degenerate elements, we solve several kinds of PDEs on various mesh types and evaluate the deviation from the respective true analytical solutions.



(a) *Grid with two needles* **(b)** *Grid with single cap* **(c)** *Grid with needle band* **(d)** *Grid with cap band* **(e)** *Unstructured grid*

(f) *Sphere with needle band* **(g)** *Sphere with cap band*

Figure 2: *The different types of triangular test meshes that were used in the numerical benchmarks.*

Experimental Setup To provide a stable testing ground, we build a synthetic dataset of relatively regular meshes, each including isolated, banded, or clustered degeneracies, as seen in Figure 2. These meshes can be adjusted to increase the number of elements (i.e., increase the grid resolution or number of vertices) and degree of degeneracy (i.e., smaller/bigger angles or degree of compression). We choose these types of degeneracies to simulate typical points of failure found in "in-the-wild" meshes, as seen in Figure 1, 5, and 4, while keeping a simple mesh shape. They are categorized into the following types:

(a) The **grid with two needles** contains two adjacent needles in the middle of the mesh. We adjust the edge length between the needles to be between 1 and 10^{-30} times the original edge length.
(b) The **grid with single cap** contains one cap in the middle and is similar to the previous case, but we decrease the distance between the vertex and edge instead.
(c) The **grid with needle band** contains multiple bands of needles. The angles can be adjusted by increasing the vertical compression (i.e., shifting the vertices to the middle).
(d) The **grid with cap band** contains multiple bands of caps and single needles at the edges. The angles can be adjusted similarly to the previous case.
(e) The **unstructured grid** contains randomly sampled vertices on $[-1,1]^2$, which are triangulated via the Delaunay triangulation. Afterwards, vertices are shifted to produce needles and caps in a ratio of 1:1. The degree of degeneracy can also be adjusted as in the first two cases.
(f) The **sphere with needle band** contains the grid with needle band wrapped into a sphere. The top and bottom vertices are designed to have a high degree.
(g) The **sphere with cap band** is analogous to the previous one using the grid with cap band.

We conduct the following experiments on several of these mesh types, each adjusted in vertex count and degree of degeneracy, resulting in hundreds of datapoints per experiment. As such, we first individually calculate a root mean squared error (RMSE) between the computed solution $\mathbf{s} \in \mathbb{R}^{|\mathcal{V}|}$ and analytical reference solution $\mathbf{a} \in \mathbb{R}^{|\mathcal{V}|}$ per mesh $\mathcal{M} = (\mathcal{V}, \mathcal{T})$, as

$$\mathrm{RMSE}(\mu, e, \mathcal{M}) = \sqrt{\frac{1}{|\mathcal{V}|} ||\mathbf{s}(\mu, e, \mathcal{M}) - \mathbf{a}(e, \mathcal{M})||_2^2}. \tag{9}$$

Here, μ and e are the current method and experiment, respectively. These per-mesh errors are then consolidated into a per-method mean relative error

$$E(\mu, e) = \frac{1}{M} \sum_{i=1}^{M} \frac{\mathrm{RMSE}(\mu, e, \mathcal{M}_i)}{\mathrm{RMSE}(\rho, e, \mathcal{M}_i)}, \tag{10}$$

over the meshes $\{\mathcal{M}_1, \ldots, \mathcal{M}_M\}$ in relation to our recommended method ρ (i.e., the D-TFEM approach introduced in Section 4.4). This results in one measure per method and experiment, indicating whether the method performs better (<1) or worse (>1) than our recommended approach. Further, as we encounter non-solvable linear systems and very high errors, we also report a NaN (not a number) percentage for non-solvable systems, and a failure percentage, meaning that the error of this method is at least three orders of magnitude higher than that of our recommended approach. The resulting mean relative error is only computed on the remaining data. We always use the same standard barycentric mass matrix.

Poisson Equation on Planar Grids To investigate the convergence property [WMKG07] of the different methods, we follow the evaluation of Bunge et al. [BB23] and solve a Poisson system for the Franke 2D function [Fra79] using Dirichlet boundary constraints. To get a broad spectrum of meshes, we use mesh types

	Library/Approach	Poisson				Spherical Harmonics				Bi-Poisson			
		NaN	Fail	Fine	Error	NaN	Fail	Fine	Error	NaN	Fail	Fine	Error
Trigonometric	CinoLib [Liv19]	2.4	10.4	87.3	103.67	0.0	60.3	39.7	6.65	27.2	23.4	49.4	127.08
	Geogram [L*25]	91.0	0.0	9.0	1.30	91.2	0.0	8.8	0.99	91.0	0.4	8.6	18.12
	OpenMesh [BSBK02]	91.0	0.0	9.0	1.26	87.8	0.0	12.2	1.00	91.0	0.4	8.6	23.70
	VCGLib [Vis23]	91.0	0.0	9.0	1.28	61.9	26.6	11.6	1.00	91.0	0.6	8.4	9.05
Intrinsic	libigl [JP*18]	63.0	2.2	34.8	50.29	70.0	9.4	20.6	59.89	59.9	1.8	38.3	88.35
	PMP [SB23]	35.8	0.9	63.3	26.95	0.0	66.6	33.4	135.23	55.5	1.0	43.5	76.54
	Intrinsic (ours)	28.3	2.8	68.9	22.74	0.0	65.6	34.4	158.39	46.4	1.6	51.9	64.88
Extrinsic	CGAL [The24]	8.3	7.0	84.8	58.18	0.0	66.2	33.8	148.31	30.0	24.0	46.0	116.36
	Geometry-Central [SC*19]	10.7	6.7	82.7	62.15	70.0	9.7	20.3	43.93	32.5	24.3	43.2	113.58
	Extrinsic (ours)	5.2	9.5	85.4	56.12	0.0	79.7	20.3	39.60	33.0	23.5	43.5	109.62
Non-Standard	Mollification [SC20]	**0.0**	**0.0**	**100.0**	**0.96**	**0.0**	**0.0**	**100.0**	0.95	0.0	0.1	99.9	10.95
	iDT+Molli. [BS07, SC20]	1.3	0.0	98.7	0.88†	**0.0**	**0.0**	**100.0**	**0.52**	5.9	0.0	94.1	9.64
	TFEM [QKL*24]	**0.0**	**0.0**	**100.0**	1.40	**0.0**	**0.0**	**100.0**	0.80	4.4	10.4	85.2	7.37
	D-TFEM (ours)	**0.0**	**0.0**	**100.0**	1.00	**0.0**	**0.0**	**100.0**	1.00	**0.0**	**0.0**	**100.0**	**1.00**

Table 1: *Statistics for the Poisson, Spherical Harmonics, and Bi-Poisson experiments. NaN: percentage of non-solvable systems; Fail: percentage of systems where the error is at least three orders of magnitude higher than reference; Fine: percentage of all other cases; Error: Mean relative error, as defined in (10).* †*Error is lower than Molli. but has NaN cases.*

(a–e). The combined results can be seen in Table 1 (left) and in the interactive convergence plots provided as supplementary material. For most approaches, we see typical quadratic convergence behavior under refinement on the meshes with isolated degenerate triangles and, to a certain degree, for the needle bands, but do not notice convergence behavior on the cap band and unstructured meshes.

Generally, the trigonometric implementations (except for CinoLib) perform poorly, resulting in 90% NaN values across all tests. This can be attributed to the fact that the angle computation becomes very unstable for small angles, thus resulting in 0° angles, which lead to the cotangent being undefined. CinoLib circumvents this problem by using a different computation of the cotangent, which is robust (i.e., $< \infty$) for 0° angles, leading to a stable performance on needle meshes. Yet, the clamping of angles $> 90°$ hurts its performance for cap meshes, as it does not show the convergence property on the single cap mesh under refinement. We also do not observe convergence for the other approaches on the single-cap meshes, as they do not produce solvable systems.

The intrinsic definitions perform noticeably better, with our implementation performing slightly better than the PMP implementation. Yet, the libigl implementation performs badly in comparison, which we attribute to the lack of ignoring zero areas. The extrinsic definitions perform best overall, wherein our implementation achieves the best results again, but all libraries perform similarly, having about 15–17% NaN/Fail values.

Spherical Harmonics To test the extension from two dimensions to closed manifold surfaces in three dimensions, we again adopt the evaluation of Bunge et al. [BB23] and solve for the spherical harmonics with $l = 4$ and $m = 2$. We use the test meshes (f, g) for this experiment. Generally, as seen in Table 1 (middle), all methods perform poorly on these tests, often not breaking entirely but leading to a very high error. CinoLib performs best, having a higher initial error than other methods but staying stable on this higher error. Furthermore, the methods using clamping of zero areas overall perform better, as they do not produce singular systems. However, they still result in the relative error being over 100.

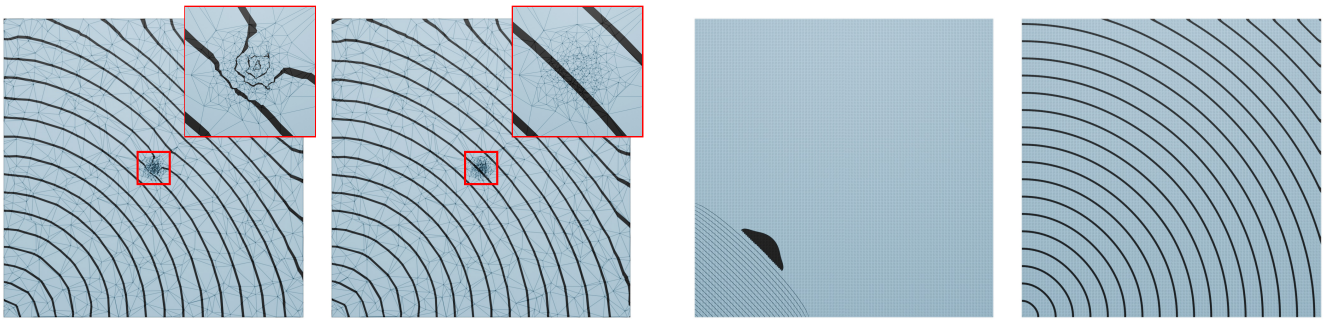
Bi-Poisson Equation To evaluate the impact of inverting the mass matrix of degenerate meshes and solving a higher-order PDE, we solve a biharmonic equation with non-zero right-hand side, which we call the Bi-Poisson equation:

$$\mathbf{L}^2\mathbf{u} = \mathbf{f} \quad \Leftrightarrow \quad \mathbf{S}\mathbf{M}^{-1}\mathbf{S}\mathbf{u} = \mathbf{M}\mathbf{f}, \tag{11}$$

on a planar grid with boundary constraints on the two outer rings of vertices. We again solve for the Franke 2D function, setting $\mathbf{f}$ to the bi-Laplacian of the Franke function, and use the test meshes (a–e). Generally, as seen in Table 1 (right), solving a fourth-order PDE results in very high errors, and nearly all approaches have NaN/Fail percentages of above 50%. Typically, the extrinsic approaches perform better on cap meshes, while the intrinsic approaches perform slightly better on needle meshes. Again, the trigonometric approaches, except for CinoLib, perform the worst across the board.

4. Don't Lose Your Temper(ing)!

While the presented methods each have their strengths and weaknesses, all of them have constraints set forth by the linear FEM discretization scheme. When looking beyond the conventional cotangent implementations, several different approaches exist, as discussed in Section 2. In this section, we investigate the recent tempered finite elements method (TFEM) by Quiriny et al. [QKL*24].



(a) *Local Refinement: [QKL*24]* **(b)** *Local Refinement: ours* **(c)** *Up-scaling: [QKL*24]* **(d)** *Up-scaling: ours*

Figure 3: *The lack of scaling invariance of the original TFEM approach [QKL*24] leads to artifacts (shown here for geodesic distance computations [CWW13]) when locally refining a coarse irregular mesh in the center region (left) and when globally up-scaling a regular regular grid of the unit square by a factor of* 10^3 *(right). In contrast, our locally and globally scaling-invariant method works robustly in both cases.*

Let $\mathbf{S}$ be the triangle stiffness matrix as defined in (1). This definition is equivalent to assembling it from local matrices $\mathbf{S}^t \in \mathbb{R}^{3\times 3}$ per triangle $t = (v_i, v_j, v_k) \in \mathcal{T}$, which are defined as

$$\mathbf{S}^t_{ij} = \frac{\langle \mathbf{x}_j - \mathbf{x}_k, \mathbf{x}_i - \mathbf{x}_k \rangle}{4|t|}, \tag{12}$$

where $|t|$ is the area of triangle t [BKP*10]. Quiriny et al. argue that this formulation creates a locking phenomenon, where bands of caps that are (nearly) degenerate are only able to interpolate linearly across the band, thus losing the convergence property of the operator [WMKG07]. To avoid this, they propose to use a mesh-dependent constant $C_{\mathcal{T}} \in \mathbb{R}$ to clamp the Jacobian determinant of the geometric map, which is equal to the double triangle area for linear FEM, resulting in the local tempered stiffness matrix

$$\overline{\mathbf{S}}^t_{ij} = \frac{\langle \mathbf{x}_j - \mathbf{x}_k, \mathbf{x}_i - \mathbf{x}_k \rangle}{2 \cdot \max\{2|t|, C_{\mathcal{T}}\}}. \tag{13}$$

This constant is chosen in relation to the mean edge length h of the mesh, where the authors propose $C_{\mathcal{T}} = C \cdot h^3$, with $C \in \mathbb{R}$ being a mesh-independent constant. While this approach is more resilient to degenerate elements, it does not necessarily fulfill the linear precision property [WMKG07] (see Section 4.4), which might be problematic in some applications.

4.1. Extension to Scaling Invariance

While the constants proposed by Quiriny et al. work well on simple caps bands, they turned out problematic in some of our tests and performed slightly worse on the more general meshes discussed in Section 3.2, as shown in Table 1 and Figure 3.

We therefore propose using a *local* element-dependent constant C_t instead of the global mesh-dependent constant $C_{\mathcal{T}}$, based on the mean edge length h_t of the element t. This allows for more flexibility when using both well-shaped large and small elements in a mesh. As this formulation is problematic for truly degenerate triangles, where the mean edge length is (nearly) zero, we clamp h_t to 10^{-10}. Moreover, we propose to use h_t^2 instead of h_t^3 to guarantee scaling invariance locally per triangle and globally for the mesh. This is the case, as C_t and $2|t|$ behave identically under uniform scaling of the triangle. Overall, this results in the formulation

$$C_t = C \cdot \max\left\{h_t, 10^{-10}\right\}^2, \tag{14}$$

$$c(t) = \max\{2|t|, C_t\}, \tag{15}$$

where we define $c(t)$ as the tempering function and $C = 10^{-3}$ as the mesh-independent constant, which we determined using a hyper-parameter sweep between 1 and 10^{-10} on our test meshes. Besides tempering the stiffness matrix $\mathbf{S}$, we also temper the mass matrix $\mathbf{M}$ by using $c(t)/2$ instead of the triangle area.

Figure 3 qualitatively shows the importance of our global/local scaling invariance and the artifacts caused by the lack thereof in the original TFEM method [QKL*24].

4.2. Extension to Gradient and Divergence

Although gradient and divergence matrices are not as prominent as the Laplacian in the geometry processing literature, both play an important role for, e.g., geodesic distances [CWW13], anisotropic smoothing [CDR00], or shape editing [YZX*04].

To extend the notion of tempering to both differential operators, we only set the simple constraint

$$\overline{\mathbf{D}} \cdot \overline{\mathbf{G}} \;=\; \overline{\mathbf{G}}^\top \overline{\mathbf{M}}_D \overline{\mathbf{G}} \;=\; \overline{\mathbf{S}}, \tag{16}$$

where $\overline{\mathbf{D}}, \overline{\mathbf{G}}$, and $\overline{\mathbf{M}}_D$ are the tempered matrices. One possible solution to this is using the tempering function $c(t_i)$ instead of $2|t_i|$ in Equation (3) and $c(t_i)/2$ instead of $|t_i|$ in $\mathbf{M}_D$, which can be easily verified when using the definitions from Equations (3) and (4).

4.3. Extension to Polygon Meshes

We generalize the TFEM discretization to general polygon meshes by following the approach of Bunge et al. [BHKB20]: We *virtually* refine polygons to triangle fans by inserting a virtual vertex, employ the above TFEM operators on the refined triangulation, and use the prolongation/restriction of Bunge et al. [BHKB20] to construct the

discrete polygon operator matrices. Since the virtual vertex optimizing the harmonic index [BBW*24] did not converge robustly in our experiments, we use the original definition as the minimizer of squared (virtual) triangle areas [BHKB20].

4.4. Comparison

We first evaluate our dynamically-tempered differential operators on triangle meshes by employing the same tests as conducted in Section 3.2, but adding one additional experiment to investigate the linear precision property and two experiments to evaluate the quality of the divergence and gradient operators. We compare our results (D-TFEM) to those obtained with the original TFEM [QKL*24], with the intrinsic Laplacian with mollification [SC20], and with intrinsic Delaunay triangulation (iDT) [BS07] combined with mollification. For the latter two cases, we employ the implementation provided in [SC*19], use the proposed mollification factor of 10^{-6}, and use the provided mass matrix based on mollification. We do not add a comparison to the regular iDT Laplacian [BS07], as building the intrinsic Delaunay triangulation without mollification did not converge on several meshes.

Matrix Assembly on Thingi10k To evaluate the robustness on real-world datasets, we build the stiffness matrix for every manifold mesh of the Thingi10k dataset [ZJ16] and evaluate if the matrix contains NaN or ∞. Generally, we found that regular Laplacians fail to produce a valid Laplacian on $\sim$10% of the meshes. Yet, approaches ignoring zero-area triangles, CinoLib, and all non-standard approaches produce valid Laplacians for all meshes.

Poisson Equation on Planar Grids Results of the Poisson test can be seen in Table 1 (left) and in the interactive convergence plots provided as supplementary material. Generally, the iDT with mollification performs best on *nearly* all triangulations, only struggling on the unstructured meshes. Overall, the tempering methods are stable for *every* triangulation, with our dynamic tempering performing best. All non-standard approaches provide a better convergence behavior than the cotangent Laplacian (see Section 3.2).

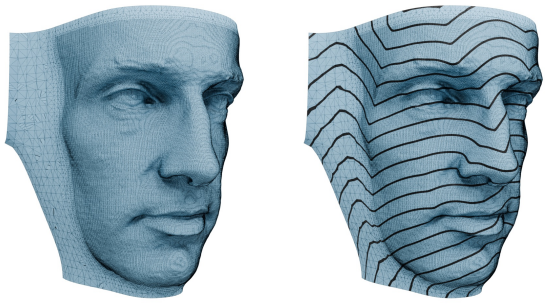
Spherical Harmonics A similar effect can be observed for the spherical harmonics test in Table 1 (middle). The iDT with mollification consistently performs better than the tempering-based approaches, but all non-standard approaches deliver stable results.

Bi-Poisson Equation The reason for this test is to investigate the impact of the mass matrix inversion (Equation (11)) on the system's stability. Table 1 (right) shows that not tempering the mass matrix can adversely affect the quality of the results, as the approach using our dynamic tempering performs best with a considerable margin. Mollification, too, alleviates these effects, but still performs subpar in some cases, especially on the unstructured triangulations.

Linear Precision A limitation of mollification and tempering is that they do not fulfill the linear precision property [WMKG07]. To validate this, we compute mean curvature (through norm of the Laplacian) for the inner vertices of a planar grid with a slightly degenerate edge. Subsequently, we compute the mean deviation from zero. While the standard cotangent Laplacians obtain errors of $\sim 10^{-12} - 10^{-16}$ depending on the discretization, both the mollification and the tempering approaches get errors of $\sim 10^{-2}$, thus showing that they do not fulfill the linear precision property. Note that while these operators technically do not satisfy the property, the regular cotangent Laplacians typically do not fulfill the property on more degenerate meshes because of numerical instabilities.

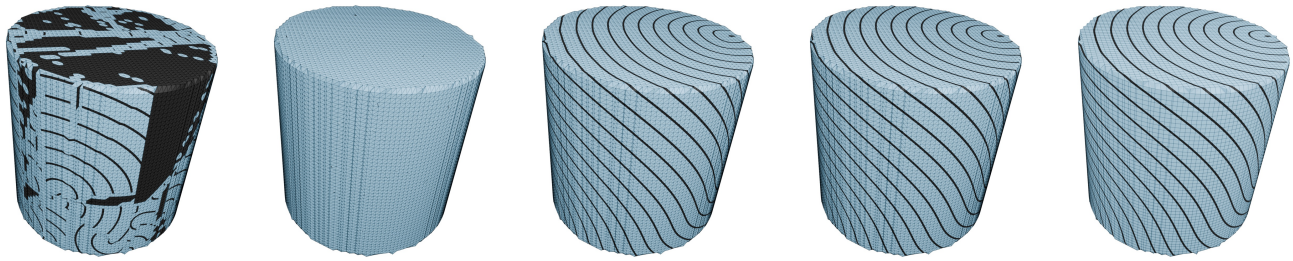
Stiffness Matrix Conditioning When solving linear systems depending on the stiffness matrix, the matrix conditioning plays a vital role in improving the speed of solvers and preventing round-off errors [She02]. The condition number strongly depends on the element quality and can grow arbitrarily large for degenerate faces. When evaluating the condition numbers of the different stiffness matrices, we find that for the standard cotangent matrices, the condition number can grow to 10^{70} or even higher, depending on the degree of degeneracy. Using the iDT approach with mollification yields much lower condition numbers, resulting in 10^{10} in the worst cases. Our dynamic tempering approach produces significantly lower condition numbers of 10^{7} in the worst cases.

Geodesics in Heat To evaluate our newly tempered gradient and divergence, we qualitatively compare the geodesic distances computed using the geodesics in heat approach [CWW13] over several test meshes, which are shown in Figure 5, 4, and 1 (left). All three meshes are real-world examples automatically generated and triangulated using Marching Cubes [LC87], Poisson surface reconstruction [KBH06], or VoroMesh [MKO*23], featuring several degeneracies similar to the synthetic ones, such as needle bands in Figure 5, and isolated/clustered degeneracies in Figure 4 and 1 (left). The cylinder even features faces with three collocated vertices, thus resulting in truly degenerate geometry. While the exact geodesic distance computation of geometry-central [MMP87, SC*19] and standard linear FEM operators cannot handle this, both the iDT with mollification and our triangular D-TFEM approach can. Similar behavior can also be seen on the *Bob* representation generated with VoroMesh [MKO*23] (Figure 1b) and the laser-scanned face reconstructed with Poisson surface reconstruction [KBH06] (Figure 4b). Yet in relation to iDT with mollification, our D-TFEM approach results in a more efficient and straightforward scheme.



(a) *Standard FEM* **(b)** *Our D-TFEM*

Figure 4: *Comparison of geodesics in heat [CWW13] for a laser-scanned face, reconstructed using triangular Poisson surface reconstruction [KBH06]: While the standard differential operators (a) fail, our approach (b) robustly obtains good results.*



(a) *Exact Geodesic Distances* **(b)** *Standard FEM (14 ms)* **(c)** *iDT + Molli. (22 ms)* **(d)** *Tri. D-TFEM (15 ms)* **(e)** *Poly. D-TFEM (88 ms)*

Figure 5: *Comparison of geodesic distances and matrix assembly timings for an implicit cylinder extracted using Marching Cubes [LC87]: (a) the exact geodesic approach [MMP87] fails; (b–e) show geodesics in heat [CWW13] using (b) the standard differential operators, (c) iDT operators with mollification [BS07, SC20], and our D-TFEM for triangle meshes (d) and polygon meshes (e). While the first two approaches fail, iDT with mollification and our approach both obtain good results, but our approach is simpler to implement and more efficient.*

Gradient-based Editing We also evaluate our gradient and divergence operators for gradient-based editing [YZX*04]. As this requires an *extrinsic* gradient and divergence definition, this can not be computed using *intrinsic* Delaunay methods. We utilize a test mesh of type (c) wrapped into a cylinder for this test. The results can be seen in Figure 1 (right). While standard operators fail at the densely triangulated center, our approach yields the expected result.

General Polygon Meshes By conducting the same experiments on quad meshes analogous to mesh types (c) and (f), we verify that the above findings generalize to polygon meshes. We improve the robustness of the polygon Laplacian to a similar extent as for the triangle operator for the Poisson, Spherical Harmonics, and Bi-Poisson tests. The geodesic distances on polygon meshes, which fail for standard operators, work robustly when using our D-TFEM approach, as shown in Figures 5e and 1b (top).

5. Recommendation and Conclusion

Based on the numerical experiments in Section 3.2 we can make the following recommendations on how to robustly compute the standard cotangent Laplacian. While the robustness of different implementations varies considerably between use cases and mesh types, *trigonometric* computations should be avoided, as they are the least robust. In our tests, *extrinsic* computations performed better on cap meshes, while *intrinsic* computations performed better on needle meshes and mixed unstructured meshes. Overall, there is no clear winner between the extrinsic and intrinsic approaches, but trigonometric computation is the clear loser. Regardless, clamping of negative weights should be avoided, while ignoring (close-to-)zero-area triangles typically is advantageous.

Our analysis of non-standard approaches in Section 4.4 clearly demonstrates that the operator stability can be significantly improved when (slightly) violating the linear precision property. Mollification [SC20], intrinsic Delaunay triangulation [BS07] (with mollification), TFEM [QKL*24], and our D-TFEM can all dramatically reduce errors and almost always provide a solvable system. These methods still depend on the stability of the underlying cotangent implementation, as a robust implementation allows to decrease the amount of mollification/tempering (and thereby reduce the violation of the linear precision property).

While mollification is a simple and effective strategy to improve the robustness of the (intrinsic) Laplacian operator, combining it with the intrinsic Delaunay triangulation gives more accurate results – actually the lowest overall errors in the Poisson and Spherical Harmonics experiments. Computing the iDT, however, requires a more complex implementation and is computationally more expensive (50–300% overhead during matrix assembly). The major drawback of mollification and iDT, however, is that both methods by construction cannot provide extrinsic gradient and divergence operators, thereby limiting the range of applications. The recent TFEM [QKL*24] is a very simple, very efficient, and very robust approach, but it suffers from severe artifacts for general, non-homogeneous irregular meshes due to its lack of scaling invariance.

Our proposed D-TFEM approach, in contrast, is applicable to a wide range of meshes thanks to its scaling invariance and our generalization to arbitrary polygon meshes. Our extension to extrinsic operators for discrete gradient and divergence allows it to be employed for a wider range of geometry processing algorithms. Overall, our method is a simple drop-in replacement for the standard operators, comes with negligible computational overhead, and robustly works where all standard operators fail. However, it can still fail in extreme cases where vertices become unconstrained, e.g., when all triangles surrounding a vertex have zero area, resulting in zero cotangent weights for all incident edges.

While our scheme works rather well on surface meshes, it might be interesting to look into a generalization to volume meshes, as Quiriny et al. also derive a similar scheme for 3D [QKL*24]. Further, the selected meshes are all manifold, and as such, it might be interesting to also consider more general surface meshes, allowing non-manifold edges or vertices, as the cotangent Laplacian can also be extended to non-manifold edges by summing over the adjacent triangles [PP93] or building a tufted cover [SC20].

References

[BA76] BABUŠKA I., AZIZ A. K.: On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis 13*, 2 (1976), 214–226. 2

[BB23] BUNGE A., BOTSCH M.: A survey on discrete Laplacians for general polygonal meshes. *Computer Graphics Forum 42*, 2 (2023), 521–544. 1, 2, 4, 5

[BBW*24] BUNGE A., BUKENBERGER D. R., WAGNER S. D., ALEXA M., BOTSCH M.: Polygon Laplacian made robust. *Computer Graphics Forum 43*, 2 (2024). 2, 7

[BHKB20] BUNGE A., HERHOLZ P., KAZHDAN M., BOTSCH M.: Polygon Laplacian made simple. *Computer Graphics Forum 39*, 2 (2020), 303–313. 2, 6, 7

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LÉVY B.: *Polygon Mesh Processing*. CRC Press, 2010. 1, 2, 6

[BRV*24] BERZINS A., RADLER A., VOLKMANN E., SANOKOWSKI S., HOCHREITER S., BRANDSTETTER J.: Geometry-informed neural networks. *arXiv preprint arXiv:2402.14009* (2024). 1

[BS07] BOBENKO A. I., SPRINGBORN B. A.: A discrete Laplace–Beltrami operator for simplicial surfaces. *Discrete & Computational Geometry 38*, 4 (2007), 740–756. 1, 2, 5, 7, 8

[BSBK02] BOTSCH M., STEINBERG S., BISCHOFF S., KOBBELT L.: OpenMesh: A generic and efficient polygon mesh data structure. In *OpenSG Symposium* (2002). https://graphics.rwth-aachen.de/software/openmesh. 2, 3, 5

[CdGDS13] CRANE K., DE GOES F., DESBRUN M., SCHRÖDER P.: Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses* (2013). 2

[CDR00] CLARENZ U., DIEWALD U., RUMPF M.: Anisotropic geometric diffusion in surface processing. In *Proceedings of the IEE Conference on Visualization '00* (2000), pp. 397–405. 6

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG) 32*, 5 (2013), 1–11. 1, 6, 7, 8

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of ACM SIGGRAPH* (1999), pp. 317–324. 2

[Dzi88] DZIUK G.: Finite elements for the Beltrami operator on arbitrary surfaces. In *Partial Differential Equations and Calculus of Variations* (1988), pp. 142–155. 1, 2

[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: a tutorial and survey. *Advances in Multiresolution for Geometric Modelling* (2005), 157–186. 2

[Fra79] FRANKE R.: *A Critical Comparison of Some Methods for Interpolation of Scattered Data*. Naval Postgraduate School Monterey, CA, 1979. 4

[HKK12] HANNUKAINEN A., KOROTOV S., KŘÍŽEK M.: The maximum angle condition is not necessary for convergence of the finite element method. *Numerische Mathematik 120* (2012), 79–88. 2

[JP*18] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library. https://libigl.github.io, 2018. 2, 3, 5

[Kah97] KAHAN W.: *Miscalculating Area and Angles of a Needle-like Triangle*. Tech. rep., University of California, Berkeley, 1997. 3

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Symposium on Geometry Processing* (2006), pp. 61–70. 7

[Kuč16] KUČERA V.: On necessary and sufficient conditions for finite element convergence. *arXiv preprint arXiv:1601.02942* (2016). 2

[L*25] LÉVY B., ET AL.: Geogram. https://github.com/BrunoLevy/geogram, 2025. 2, 3, 5

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Computer Graphics 21*, 4 (1987), 163–169. 1, 7, 8

[Liv19] LIVESU M.: Cinolib: A generic programming header only C++ library for processing polygonal and polyhedral meshes. *Transactions on Computational Science XXXIV* (2019). https://github.com/mlivesu/cinolib/. 2, 3, 5

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*. Springer-Verlag, 2003, pp. 35–57. 1

[MKO*23] MARUANI N., KLOKOV R., OVSJANIKOV M., ALLIEZ P., DESBRUN M.: VoroMesh: Learning watertight surface meshes with Voronoi diagrams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2023), pp. 14565–14574. 1, 7

[MMP87] MITCHELL J. S., MOUNT D. M., PAPADIMITRIOU C. H.: The discrete geodesic problem. *SIAM Journal on Computing 16*, 4 (1987), 647–668. 7, 8

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 165–174. 1

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2*, 1 (1993), 15–36. 1, 2, 8

[QKL*24] QUIRINY A., KUČERA V., LAMBRECHTS J., MOËS N., REMACLE J.-F.: The tempered finite element method. *arXiv preprint arXiv:2411.17564* (2024). 2, 5, 6, 7, 8

[Rip92] RIPPA S.: Long and thin triangles can be good for linear interpolation. *SIAM Journal on Numerical Analysis 29*, 1 (1992), 257–270. 2

[SB23] SIEGER D., BOTSCH M.: The Polygon Mesh Processing Library. https://github.com/pmp-library/pmp-library, 2023. 2, 3, 5

[SBMS23] SORGENTE T., BIASOTTI S., MANZINI G., SPAGNUOLO M.: A survey of indicators for mesh quality assessment. *Computer Graphics Forum 42*, 2 (2023), 461–483. 2

[SC*19] SHARP N., CRANE K., ET AL.: GeometryCentral: A modern C++ library of data structures and algorithms for geometry processing. https://geometry-central.net, 2019. 2, 3, 5, 7

[SC20] SHARP N., CRANE K.: A Laplacian for nonmanifold triangle meshes. *Computer Graphics Forum 39*, 5 (2020), 69–80. 1, 2, 5, 7, 8

[SHD*18] SCHNEIDER T., HU Y., DUMAS J., GAO X., PANOZZO D., ZORIN D.: Decoupling simulation accuracy from mesh quality. *ACM Transactions on Graphics (TOG) 37*, 6 (2018). 2

[She02] SHEWCHUK J. R.: What is a good linear element? interpolation, conditioning, and quality measures. In *International Meshing Roundtable (IMR)* (2002). 2, 7

[SSC19] SHARP N., SOLIMAN Y., CRANE K.: Navigating intrinsic triangulations. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–16. 2

[The24] THE CGAL PROJECT: *CGAL User and Reference Manual*, 6.0.1 ed. CGAL Editorial Board, 2024. 2, 3, 5

[Vis23] VISUAL COMPUTING LAB: The visualization and computer graphics library. https://github.com/cnr-isti-vclab/vcglib, 2023. 2, 3, 5

[WMKG07] WARDETZKY M., MATHUR S., KÄLBERER F., GRINSPUN E.: Discrete Laplace operators: No free lunch. In *Symposium on Geometry Processing* (2007), pp. 33–37. 1, 2, 3, 4, 6, 7

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with Poisson-based gradient field manipulation. In *Proceedings of ACM SIGGRAPH* (2004), pp. 644–651. 1, 6, 8

[ZJ16] ZHOU Q., JACOBSON A.: Thingi10K: A dataset of 10,000 3D-printing models. *arXiv preprint arXiv:1605.04797* (2016). 1, 7

[Zlá68] ZLÁMAL M.: On the finite element method. *Numerische Mathematik 12*, 5 (1968), 394–409. 2