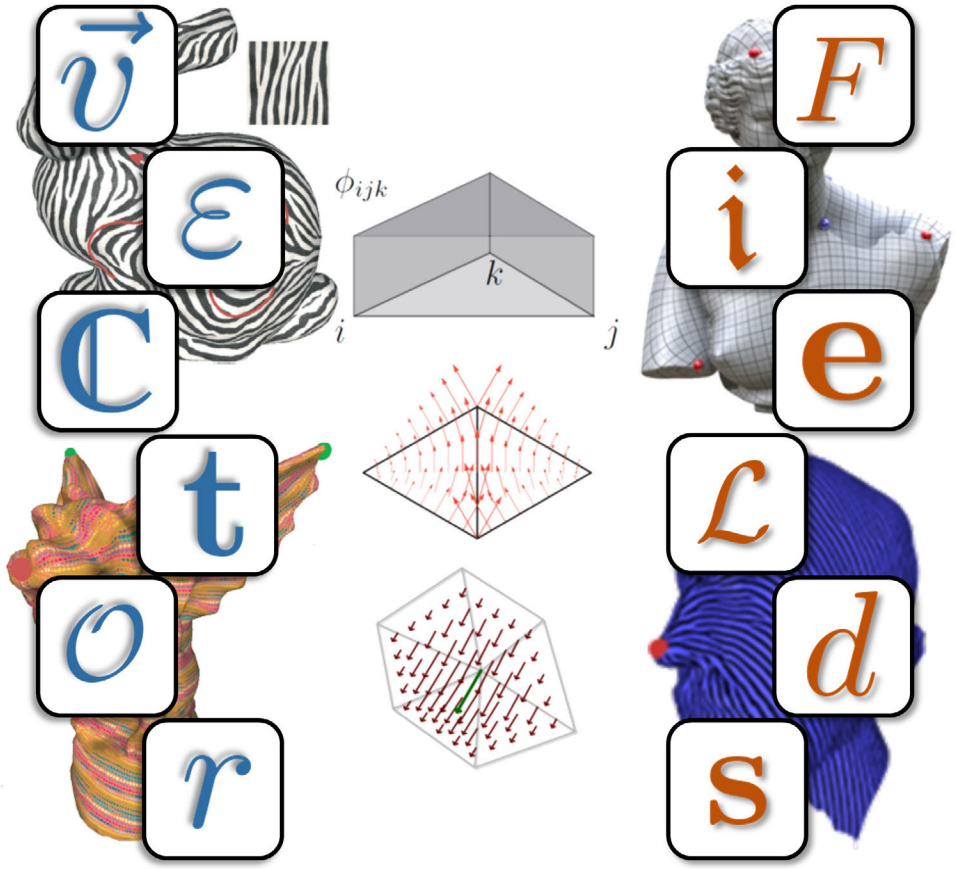
Vector Field Processing on Triangle Meshes



Course Notes

ACM SIGGRAPH 2016

Anaheim, CA

ORGANIZERS:

FERNANDO DE GOES
Pixar Animations Studios

MATHIEU DESBRUN
Caltech

YIYING TONG
Michigan State University

Abstract

While scalar fields on surfaces have been staples of geometry processing, the use of tangent vector fields has steadily grown in geometry processing over the last two decades: they are crucial to encode both directions and sizing on surfaces as commonly required in tasks such as texture synthesis, non-photorealistic rendering, digital grooming, and meshing. There are, however, a variety of discrete representations of tangent vector fields on triangle meshes, and each approach offers different trade-offs among simplicity, efficiency, and accuracy depending on the targeted application.

This course reviews the three main families of discretizations used to design computational tools for vector field processing on triangle meshes: face-based, edge-based, and vertex-based representations. In the process of reviewing the computational tools offered by these representations, we go over a large body of recent developments in vector field processing in the area of discrete differential geometry. We also discuss the theoretical and practical limitations of each type of discretization, and cover increasingly-common extensions such as n-direction and n-vector fields.

While the course will focus on explaining the key approaches to practical encoding (including data structures) and manipulation (including discrete operators) of finite-dimensional vector fields, important differential geometric notions will also be covered: as often in Discrete Differential Geometry, the discrete picture will be used to illustrate deep continuous concepts such as covariant derivatives, metric connections, or Bochner Laplacians.

Preface

These course notes provide a review of the various representations of tangent vector fields on triangulated surfaces. Over the past decades, a number of approaches to express, design, and analyze vector fields on arbitrary meshes have been proposed in geometry processing, targeting a series of applications varying from rendering to animation. Some describe vector fields through values on vertices, some on edges, and some on faces; some treat vector fields as piecewise constant, some as piecewise linear, and some even involve non-linear finite elements. This lack of a unified approach to vector field representation and manipulation can be confusing for practitioners, and rare are the references that discuss the pros and cons of these various representations. The chapters in this document were written with this fact in mind, as a way to offer an introduction to the motivations, benefits, and shortcomings of the most common discretizations of vector fields in graphics—along with their associated discrete differential operators (curl, divergence, Laplacian, etc). Emphasis is placed on the tradeoffs between simplicity, efficiency, and accuracy of these geometry processing tools for applications ranging from texture synthesis to meshing.

Course rationale. In many ways, this course is a continuation of ACM SIGGRAPH courses on Discrete Differential Geometry (SIGGRAPH '05 and '06, and SIGGRAPH Asia '09) and Discrete Exterior Calculus (SIGGRAPH '13) taught over the past decade. It is also a revised version of the vector field processing course taught at SIGGRAPH Asia 2015 in Kobe, Japan. Nevertheless, it does not require familiarity with these former courses. The course will, instead, introduce topics absent from previous offerings such as connections, covariant derivatives, and interpolating basis functions for vector fields—which are gaining popularity among practitioners, but for which no tutorials are available. The notes are intended for graduate students, researchers, and developers interested in geometry processing. Practitioners will find concrete data structures and numerical tools for the manipulation of vector fields on triangle surfaces, while scholars will learn how to preserve geometrical and topological continuous properties in the finite-dimensional realm.

Pedagogical Intentions. These course notes were conceived to complement a half-day course at SIGGRAPH '16, so we mostly mirrored its three-part structure, reviewing works where vector fields are stored per face first, then describing those using per-edge representation, then finally going over the more difficult case of per-vertex encoding. For each representation, we provide its essential mathematical background as well as a few examples of real-world applications. The course material aims at providing not only a literature review of the field of vector field processing, but also to introduce basic geometric concepts that the field of differential geometry has developed to provide a principled approach to vector calculus on arbitrary manifolds. The theme of discrete differential geometry (i.e., finite dimensional approximations that respect key properties of their infinite-dimensional counterparts) will be omnipresent in the course, and numerical advantages and limitations will be highlighted, often using concrete applications for better illustration. As a consequence, the material should be accessible to anyone who has had exposure to basic linear algebra and vector calculus. A basic understanding of differential geometry is also recommended, but not mandatory. Throughout these notes, we strived to present each mathematical concept using a concrete geometric picture; we punt most of the associated abstract or algebraic arguments to existing references for the readers who want to get a more thorough exposition.

Credit where credit is due. We are indebted to our coauthors for the material included in [Tong et al. 2003; Wang et al. 2006; Fisher et al. 2007; Hertzmann and Zorin 2000; Crane et al. 2010; Crane et al. 2013; de Goes et al. 2014; Liu et al. 2016]. We also thank the attendants of our previous classes for their feedback by which our approach was informed.

Introduction to Discrete Vector Fields

1 Vector fields in geometry processing

Before delving into the technical details of how vector fields are encoded in traditional geometry processing methods, we provide some background on the relevance and difficulties of discrete vector fields in graphics.

1.1 Motivation

Numerous applications in computer graphics need to manipulate tangent vector fields on triangle meshes. For instance, example-based texture synthesis needs a vector field to define the texture's local orientation and scale. Non-photorealistic rendering often renders brush strokes and hatches along a vector field as well. Fur in computer-generated movies is typically derived from an artist-designed vector field to indicate hair direction and length (see inset). Vector fields can also be used to derive a quad mesh for which edges align to the local direction and scale of the field. Even animation can benefit from the notion of vector fields since they are the main physical variable to simulate all sorts of fluid motion on geometric domains. Finally, it is important to point out that vector fields are the simplest members of the more general notion of n-vector fields (also called n-rotational symmetry or n-RoSy fields) and n-direction fields (unit n-vector fields), which include direction ($n=1$), line ($n=2$) and cross ($n=4$) fields. Consequently, knowing how to manipulate vector fields on surfaces is the necessary portal to many additional uses.

©Pixar/Disney

1.2 A first look at discrete vector fields

Defining a tangent vector field on a triangulated surface is indubitably more involved than a scalar function. First, the very fact that each vector is supposed to be within a *tangent plane* is at odds with the fact that a triangle mesh does not even have a clear, canonical definition of a tangent space for points along an edge or at a vertex. Second, one could think that it is as simple as storing two scalar fields, one for each coordinate; but that would be assuming that one has previously defined a notion of *local coordinate frame* at every point on the surface—a difficult task in itself as it involves parameterization and defining two basis vector fields over the parameterization chart! No surprise, then, that there exist multiple ways of encoding a vector field over a mesh.

Moreover, a vector field can contain a number of *singularities*, which often require careful handling in practical applications. For a scalar field, there are also special points, easily identified as local maxima, local minima, or saddle points; but the catalog of singularities for vector fields is now more involved. In fact one can define the *index* for a singularity that basically measures the angular variation of the vector field around a singularity (i.e., the number of times it coils around). Moreover, the sum of these indices for an arbitrary vector field must be *fixed* for a given surface. That is to say, you cannot ask a vector field for too much: the topology of the surface over which the tangent vector field is defined gets in the way, forcing global conditions to hold. This is what the "hairy ball theorem" [Eisenberg and Guy 1979] states: you simply cannot comb a sphere without creating at least one cowlick somewhere; yet you can

comb a doughnut, as these two geometric objects differ in topology. Geometry and topology of vector fields are thus intimately linked.

Since we usually seek smooth-looking vector fields, we also need to define a notion of smoothness that is intuitive, both quantitatively (so that we can objectively refer to a vector field as being smoother than another) and visually (so that it looks nice in the "eyeball norm" as well). While smoothness and derivatives of scalar functions are typically high school topics, derivatives of vector fields on non-flat surfaces are more significantly involved, and a few specific definitions of derivatives will be relevant for the design of vector fields. Only then can a user expect to turn a sparse set of constraints (prescribing alignment and magnitude) into a smooth vector field automatically fitting these input constraints—thus making the process of designing a vector field simple, intuitive, and hopefully fast.

1.3 Discrete Representations

Since the choice of smoothness and input constraints depends very much on how vector fields are defined over a triangle mesh, we must first and foremost decide on a finite-dimensional representation of vector fields over the underlying surface. Such representations are often grounded in differential geometry, and will serve as the principles for the discretization of tangent vector fields on triangle meshes.

In $\mathbb{R}^2$, we are used to manipulate vector fields as a set of two coordinates per point associated to a *global* coordinate system. Note that these coordinates depend on the choice of a basis, so the vector field is only *encoded* via coordinates. Yet the field itself is a geometrical object that retains its own identity regardless of how one chooses to describe it in a basis. For tangent vector fields on a non-flat surface, one cannot use a single coordinate frame; instead, one must define a coordinate frame for each tangent plane (possibly through charts) if one wishes to retain the two-component representation of vector fields—and fortunately, we will see that a triangle mesh provides ample opportunities to define local frames. A vector field then becomes an assignment of a tangent vector to each point on a manifold surface. In the language of fiber bundles[1], a vector field is thus a *section of the tangent bundle*, in the sense that from all the tangent vectors associated to each point of the manifold, we pick a particular one, and this choice of tangent vector per surface point defines a vector field. Existing discrete representations of vector fields mimic this fiber bundle point of view as we will review in these notes.

One can also do away with coordinates altogether. We will show that one can sample a vector field, much like one can sample a scalar field and, from these samples, a reconstruction of the original field can be found. Other ways to think of vector fields on manifolds in the continuous setting can also be leveraged to derive discrete representations. For instance, vector fields can be characterized as derivations of smooth functions on a manifold using directional derivative; this has led to an operator representation of vector fields, used first in fluid animation [Mullen et al. 2009; Pavlov et al. 2011; Gawlik et al. 2011], and more recently in geometry processing [Azencot et al. 2013; Azencot et al. 2015].

1.4 Vector calculus

Many of the elements of differential and integral calculus extend to vector fields quite naturally, but can involve advanced geometric notions. A particularly difficult endeavor in vector calculus over surfaces is how to talk about the *difference* between two nearby tangent vectors: while this notion is trivial in a flat space where coordinates in a common basis can be used, we have to find a proper geometric approach to define it on a curved domain. A covariant derivative introduces such an extra geometric structure on a manifold which allows vectors in neighboring tangent spaces to be compared. This extra structure is

[1]The term *fiber bundle* can be intuitively understood by thinking of it as a hairbrush: for each point of a base space (cylinder), there is a vector space (bristle) associated to it. A fiber bundle is thus a way of knitting together several spaces to create a bigger space. By picking a particular point on the bristle at each point of the base manifold in a continuous fashion, we construct a (cross-)section of the hairbrush (in our context, a vector field—but the same holds for any fiber field).

necessary since there is no canonical way to compare vectors from *different* vector spaces. Covariant differentiation also helps to define the notion of parallel transport along a path, the analog of a translation in $\mathbb{R}^2$. Working with a *connection* on the tangent bundle (to compare two nearby tangent vectors, and thus induce a covariant derivative) will eliminate the need for awkward manipulations of Christoffel symbols traditionally found in differential geometry. Our course notes review various approximations of these notions, and as one might expect from any numerical treatment, various tradeoffs between simplicity and accuracy will be provided.

2 Overview

The core of these course notes is made of three chapters, each reviewing one particular vector field representation on triangle meshes. By design, each chapter is essentially independent of the others. However, the order in which these chapters are presented was decided based on the complexity of the material. The reader can thus decide whether to read the chapters sequentially, or directly skip to the material that is most relevant to her.

2.1 A chapter-by-chapter synopsis

Existing geometry processing methods generally use one of three different encoding of vector fields. We review them one by one, contrasting their pluses and minuses.

Face-based vector fields. The first part of the course focuses on piecewise-constant vector fields per triangle. As a triangle is flat, it is particularly convenient to define a tangent vector on its supporting plane. Even with jumps between faces, we will be able to define discrete notions of divergence, curl, or even Laplacian of these vector fields, and we will point out that each vector field can be decomposed into curl- and divergence-free components using (plain or mixed) finite elements. We will also introduce a very simple discrete notion of connection, corresponding to a single angle for every pair of adjacent triangles, as an alternative way to think of the vector field which extends naturally to arbitrary n-vector fields.

Edge-based vector fields. The second part of the course discusses the use of edge integration (i.e., line integral) to represent a discrete vector field. We review how these edge values can, through piecewise-linear Whitney basis functions, be interpolated into vector fields or, equivalently, covector fields. This approach produces locally linear vector fields, although normal continuity across edges is not guaranteed. Nevertheless, discrete differential operators can be easily derived and computed just like in the previous case, from which one can easily design vector fields in realtime on triangle meshes.

Vertex-based vector fields. The third and last part reviews recent approaches encoding vector fields using vertex-based non-linear basis functions. By thinking of the one-ring of a vertex as a chart, a continuous vector field can be defined on an arbitrary triangle mesh, from which operators such as the divergence, curl, and covariant derivative can be directly derived. This will even allow us to define the notion of *smoothest* n-vector and n-direction field for a surface, which will correspond to a generalized eigenvalue problem.

The reader will have noticed that the ordering also corresponds to various orders of continuity—from discontinuous to continuous—of the discrete representations. It also, coincidentally, involves increasingly complex computational tools to evaluate differential operators in a weak form on these vector fields.

2.2 Applications

Applications using vector fields as input or guidance are legion. As our course focuses on the underlying representation of vector fields, we made no effort to review thoroughly the numerous geometry processing applications that each representation has been used in. Instead, we weaved examples of non-photorealistic rendering, quad meshing, or vector field design in every chapter to provide a more thorough description of the use of each representation. The reader interested in a broader perspective on these applications is referred to the numerous books and courses on geometry processing—a recent EG course [Vaxman et al. 2016] is particularly relevant in that respect.

2.3 Notations

Throughout these notes, we assume that we are working with a triangle mesh, approximating a smooth surface, that is of arbitrary topology, orientable, compact, and 2-manifold (possibly with boundary). We denote its vertex set $V = \{v_i\}$, its edge set $E = \{e_{ij}\}$ and its triangle set $T = \{t_{ijk}\}$ ($1 \leq i, j, k \leq n = |V|$). Each triangle is counterclockwise oriented and each edge carries an arbitrary but fixed intrinsic orientation. Note that index order matters since e_{ij} has opposite orientation from e_{ji}, and similarly for triangles. Vertices are given positions $P = \{\mathbf{p}_v \in \mathbb{R}^3 | v \in V\}$, which define the surface through piecewise linear interpolation over each triangle, so that $\mathbf{p}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ represents the edge as a segment in $\mathbb{R}^3$. The intrinsic volumes of edges and triangles will be denoted $|e|$ (length) and $|t|$ (area), and we assume these are all nonzero, i.e., there are no degenerate edges or triangles. For a vertex, we define $|v| = 1$. We also assign a lumped area per edge and vertex: a_{ij} is one third of the triangle areas incident to e_{ij}, and a_i is one third of the one-ring area. Moreover, we use α_{jki} to indicate the angle of the triangle t_{ijk} at vertex v_k, so that the discrete Gaussian curvature κ_i of vertex v_i is the angle defect $\kappa_i = 2\pi - \sum_{t_{ijk}} \alpha_{kij}$. Finally, we denote the imaginary unit by i when we use complex numbers.

3 Differential geometry primer for vector fields

In this section, we tersely review some key continuous geometric notions that will come up in these notes. While these notions can be introduced in various ways, we focus as much as possible on intrinsic definitions to provide a very geometric explanation of their meaning. For more details on our geometric descriptions, see [Abraham et al. 1988]. The reader is invited to refer back to this primer when the discretization of operators is discussed during the class, to better understand how the discrete and differential viewpoints often coincide—but sometimes differ.

3.1 Tangent Vector Fields

Consider a compact topological 2-manifold M, covered by a collection (atlas) of charts that have C^∞ smooth transition functions between each overlapping pair (which always exists [Grimm and Hughes 1995]). The notion of tangent planes and vectors can be defined intrinsically (*i.e.*, independent of the embedding of the manifold in a Euclidean space) via, for instance, the tangency among smooth curves passing through a common point.

Definition 1 ([Abraham et al. 1988])**.** *Let $\mathbf{x} = (x^1, x^2)$ be a local chart mapping an open set $U \subset M$ to $\mathbb{R}^2$. A smooth* curve *c passing through a point $\mathbf{p} \in U$ is a map $c \colon I \to U$ for which the interval $I \subset \mathbb{R}$ contains 0, $c(0) = \mathbf{p}$, and $\mathbf{x} \circ c$ is C^2. Two smooth curves c_1 and c_2 are said to be* **tangent** *at $\mathbf{p}$ if and only if*

$$(\mathbf{x} \circ c_1)'(0) = (\mathbf{x} \circ c_2)'(0). \tag{1}$$

Note that this definition of tangency is independent of the choice of charts. Tangent curves can thus be used as an equivalence relation defining intrinsic vector spaces tangent to M.

Definition 2 ([Abraham et al. 1988]). *A* **tangent vector** *at* $\mathbf{p} \in M$ *is the equivalence class* $[c]_{\mathbf{p}}$ *of curves tangent to curve* c *at* $\mathbf{p}$*. The space of tangent vectors is called the* **tangent space** *at* $\mathbf{p}$*, denoted as* $T_{\mathbf{p}}M$*. The* **tangent bundle** *is the (disjoint) union of tangent spaces* $TM = \cup_{\mathbf{p}\in M} T_{\mathbf{p}}M$*.*

When the surface M has an embedding in $\mathbb{R}^3$, one can further express the tangent vectors as 3D vectors orthogonal to the surface normal, as classically explained in differential geometry of surfaces. Observe that the tangent space $T_{\mathbf{p}}M$ at any point $\mathbf{p} \in M$ is two dimensional and a tangent vector $\mathbf{u} = [c]_{\mathbf{p}}$ can be represented in components as $(u^1, u^2) = ((x^1 \circ c)'(0), (x^2 \circ c)'(0))$ given a chart $\mathbf{x}$. Thus, the tangent bundle TM admits the structure of a 4-manifold with charts (x^1, x^2, u^1, u^2) induced by the atlas of M.

Definition 3 ([Abraham et al. 1988]). *A (tangent)* **vector field** $\mathbf{u}$ *is a continuous map* $M \to TM$ *from a point* $\mathbf{p} \in M$ *to a vector* $\mathbf{u}(\mathbf{p}) \in T_{\mathbf{p}}M$*. A* **local frame field** *of* M *on a chart is defined as two vector fields* $(\mathbf{e}_1, \mathbf{e}_2)$ *that are linearly independent pointwise.*

Global frame fields do not exist in general, otherwise one could build a continuous vector field that is nonzero everywhere on a genus-0 surface, thus contradicting the hairy ball theorem [Eisenberg and Guy 1979]. Consequently, TM does not usually have the structure of $M \times \mathbb{R}^2$, even though on a chart U, TU does have the structure of $U \times \mathbb{R}^2$. On a chart with a local frame field, a vector field $\mathbf{u}$ can be expressed in components as

$$\mathbf{u} = u^1 \mathbf{e}_1 + u^2 \mathbf{e}_2. \tag{2}$$

The aforementioned chart of TM can be seen as a special case of the component representation, with $\mathbf{e}_i$ (often denoted as $\partial/\partial x^i$) being the equivalence class of the curves generated by varying coordinate x^i while keeping the other coordinate fixed.

Definition 4 ([Abraham et al. 1988]). *A* **covector** ω *at* $\mathbf{p}$ *is defined as a linear map* $\omega : T_{\mathbf{p}}M \to \mathbb{R}$*. The space of covectors is a linear space itself, denoted as* $T^*_{\mathbf{p}}M$*.*

One can likewise define smooth fields of covectors, which are also called (differential) 1-forms. They can be represented in local bases (η^1, η^2) defined by $\eta^i(\mathbf{e}_j) = \delta^i_j$ given a frame field $(\mathbf{e}_1, \mathbf{e}_2)$.

One can also augment a surface M with a metric by assigning an inner product (symmetric positive definite bilinear mapping) $\langle \cdot, \cdot \rangle_{\mathbf{p}}$ for every tangent space $T_{\mathbf{p}}M$—e.g., for an embedded surface, it can be defined by the inner product of the corresponding 3D vectors in the 3D Euclidean space.

Finally, we point out that the directional derivative of a function f over M w.r.t. a vector $\mathbf{u} = [c]_{\mathbf{p}} \in T_{\mathbf{p}}M$ is defined as $(f \circ c)'(0)$, corresponding to $df(\mathbf{u})$ in the language of differential forms and to the more familiar inner product $\langle \nabla f, \mathbf{u} \rangle$ when a metric is available [Abraham et al. 1988].

3.2 Covariant Derivative

In order to take derivatives of vector fields, one must account for the fact that vectors in nearby tangent spaces are expressed on different local frames. The concept of covariant differentiation, denoted ∇, provides a principled way to compare nearby tangent vectors and measure their differences. The basic geometric intuition behind the covariant derivative of a vector field $\mathbf{u}$ at a point $\mathbf{p}$ is that $\nabla \mathbf{u}$ encodes the rate of change of $\mathbf{u}$ around $\mathbf{p}$. Projecting the derivative of a vector field $\mathbf{u}$ along a vector $\mathbf{w}$ leads to a vector $\nabla_{\mathbf{w}}\mathbf{u}$, which indicates the difference between vectors $\mathbf{u}(\mathbf{p})$ at $\mathbf{p}$ and $\mathbf{u}(\mathbf{q})$ at a nearby point $\mathbf{q} \equiv c(\epsilon)$, where c is a curve passing through $\mathbf{p}$ in the equivalence class $\mathbf{w}$, and $\epsilon \in \mathbb{R}$ is approaching zero (Fig. 1). However, these vectors live in different tangent spaces, so the component-wise differences depend on the choice of local basis frames, and taking their differences in a manner that is purely intrinsic (i.e., coordinate/frame independent) requires the additional notion of connection as we now review.

Definition 5 ([Spivak 1979]). *A* **covariant derivative** *(or an* **affine connection***) is an operator* ∇ *mapping a vector* $\mathbf{w} \in T_{\mathbf{p}}M$ *and a vector field* $\mathbf{u}$ *to a vector* $\nabla_{\mathbf{w}}\mathbf{u} \in T_{\mathbf{p}}M$*, so that it is linear in both* $\mathbf{u}$

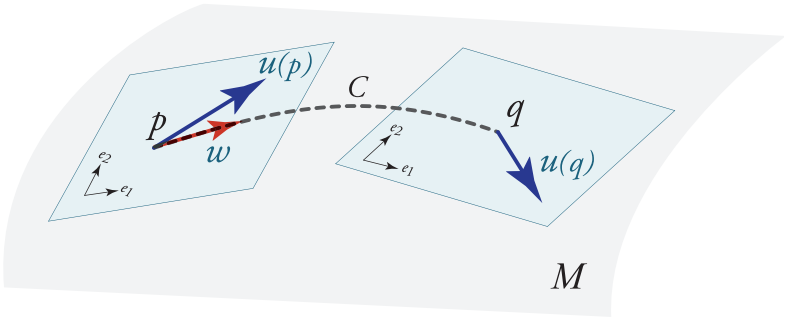


Figure 1: ***Smooth connection.*** *On a smooth manifold, a connection indicates how a tangent vector at point* $\mathbf{p}$ *is parallel transported along a path* C *to a nearby point* $\mathbf{q}$*, accounting for the change of frame between the two tangent spaces. From a connection the notion of (covariant) derivative of vector fields is deduced, as nearby vectors can now be compared.*

and $\mathbf{w}$ *and satisfies Leibniz's product rule, i.e., for a vector field* $\mathbf{u}$ *and a smooth function* f*, one has*

$$\nabla_{\mathbf{w}}(f\mathbf{u}) = df(\mathbf{w})\mathbf{u} + f\nabla_{\mathbf{w}}\mathbf{u}.$$

Using the representation of the vector field $\mathbf{u}$ in a local frame field $(\mathbf{e}_1, \mathbf{e}_2)$, we can expand the covariant derivative through linearity and product rule in $\mathbf{u}$ as:

$$\nabla_{\mathbf{w}}\mathbf{u} = \sum_{i=1,2} \left[du^i(\mathbf{w})\mathbf{e}_i + u^i \nabla_{\mathbf{w}}\mathbf{e}_i \right],$$

where the second term of this derivative accounts for the alignment of the local frame at a point to a nearby local frame along a curve having $\mathbf{w}$ as its tangent vector (Fig. 1). By linearity in $\mathbf{w}$, we can rewrite $\nabla_{\mathbf{w}}\mathbf{e}_i = w^1\nabla_{\mathbf{e}_1}\mathbf{e}_i + w^2\nabla_{\mathbf{e}_2}\mathbf{e}_i$, and introduce coefficients ω_{ji}^k satisfying

$$\nabla_{\mathbf{e}_j}\mathbf{e}_i = \omega_{ji}^1\mathbf{e}_1 + \omega_{ji}^2\mathbf{e}_2.$$

In the dual basis (η^1, η^2) of $T_{\mathbf{p}}^*M$, we can group these coefficients as *local* 1-forms $\omega_j^i \equiv \omega_{1j}^i\eta^1 + \omega_{2j}^i\eta^2$, to encode the alignment of nearby local frames as a local *matrix-valued* 1*-form*:

$$\Omega(\mathbf{w}) = \begin{pmatrix} \omega_1^1(\mathbf{w}) & \omega_2^1(\mathbf{w}) \\ \omega_1^2(\mathbf{w}) & \omega_2^2(\mathbf{w}) \end{pmatrix}, \forall \mathbf{w} \in T_{\mathbf{p}}M.$$

Using Ω, we can then reformulate the covariant derivative as:

$$\nabla_{\mathbf{w}}\mathbf{u} = (\mathbf{e}_1\ \mathbf{e}_2) \begin{pmatrix} du^1(\mathbf{w}) \\ du^2(\mathbf{w}) \end{pmatrix} + (\mathbf{e}_1\ \mathbf{e}_2)\ \Omega(\mathbf{w}) \begin{pmatrix} u^1 \\ u^2 \end{pmatrix}.$$

Note that if one considers a different local frame field $(\tilde{\mathbf{e}}_1, \tilde{\mathbf{e}}_2)$ at $\mathbf{q}$ satisfying $(\tilde{\mathbf{e}}_1(\mathbf{q}), \tilde{\mathbf{e}}_2(\mathbf{q})) = (\mathbf{e}_1(\mathbf{q}), \mathbf{e}_2(\mathbf{q}))(I - \epsilon\Omega(\mathbf{w}))$, where $\mathbf{q} = \mathbf{x}^{-1}(\mathbf{x}(\mathbf{p}) + \epsilon\mathbf{w})$ is a point ϵ-away from $\mathbf{p}$ along $\mathbf{w}$ (expressed in a chart $\mathbf{x}$), then the corresponding matrix-valued 1-form satisfies $\tilde{\Omega}(\mathbf{w}) = 0$, and $\epsilon\nabla_{\mathbf{w}}\mathbf{u}$ becomes a direct comparison of components $(\tilde{u}^1, \tilde{u}^2)$ at $\mathbf{q}$ and $\mathbf{p}$; in other words, these frames are aligned along the path c. It is also worth pointing out that, even though the matrix-based 1-form Ω is dependent on the choice of frame field, $\nabla\mathbf{u}$ is instead a proper, globally-defined tensor field.

3.3 Metric Connections

While the definitions above are valid for arbitrary connections, we restrict our attention from now on to metric affine connections.

Definition 6 ([Spivak 1979])**.** *For a smooth* 2*-manifold* M *equipped with a metric* $\langle \cdot, \cdot \rangle$*, a* **metric affine connection** *is a connection that preserves the metric, i.e., that satisfies*

$$d\langle \mathbf{u}_1, \mathbf{u}_2 \rangle (\mathbf{w}) = \langle \nabla_{\mathbf{w}} \mathbf{u}_1, \mathbf{u}_2 \rangle + \langle \mathbf{u}_1, \nabla_{\mathbf{w}} \mathbf{u}_2 \rangle \, , \; \forall \mathbf{w}, \mathbf{u}_1, \mathbf{u}_2 \in TM.$$

Note that an orthonormal frame field $(\mathbf{e}_1, \mathbf{e}_2) \equiv (\mathbf{e}, \mathbf{e}^\perp)$ is uniquely defined through a unit vector $\mathbf{e}$ and its $\frac{\pi}{2}$-rotation $\mathbf{e}^\perp$ in the given metric; we thus refer to $\mathbf{e}$ as a local frame field. With the compatibility condition that metric connections must verify, one can show that the local 1-form Ω on an orthonormal frame simplifies to:

$$\Omega = \begin{pmatrix} 0 & -\omega \\ \omega & 0 \end{pmatrix} = \omega J,$$

where J is the $\frac{\pi}{2}$-rotation matrix

$$J = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},$$

and ω is a local, real-valued 1-form encoding infinitesimal angular velocity with which a local frame needs to rotate to align to nearby frames when moving along a given vector. We will refer to ω as the (metric) connection 1-form.

An important special case of metric connection is the Levi-Civita connection: for a given metric defined over a 2-manifold M, this is the *unique* metric connection simultaneously preserving this metric and satisfying $\omega^i_{jk} = \omega^i_{kj}$ in frame field $(\partial/\partial x^1, \partial/\partial x^2)$. In particular, for a surface embedded in $\mathbb{R}^3$, the Levi-Civita connection induced by the metric inherited from the Euclidean space corresponds to the tangential component of the traditional (3D) component-wise derivatives of a vector field. For definitions of other connections defined on vector or frame bundles, we refer the reader to [Spivak 1979].

3.4 Related Concepts

We end this section with a few key geometric definitions which we will refer to throughout the course.

Parallel transport. The notion of connection provides a natural definition of parallel transport: given a connection 1-form ω and its covariant derivative ∇, the parallel transport of a vector $\mathbf{u}(\mathbf{p})$ along a curve c is defined as vectors along the curve such that $\nabla_{c'(s)} \mathbf{u} = 0$, where $c'(s)$ is the tangent vector $[c]_{c(s)}$. Using components, one can show that any vector that is parallel-transported along c undergoes a series of infinitesimal rotations in the basis $(\mathbf{e}, \mathbf{e}^\perp)$, leading to

$$\begin{pmatrix} u^1(s) \\ u^2(s) \end{pmatrix} = \exp\left(-J \int_0^s \omega(c'(\alpha)) d\alpha \right) \begin{pmatrix} u^1(0) \\ u^2(0) \end{pmatrix}, \tag{3}$$

where the matrix exponential $\exp(\theta J) = \cos\theta I + \sin\theta J$ is the resulting rotation matrix induced by the connection ω in order to align $T_{c(0)}M$ to $T_{c(s)}M$ (with I denoting the 2×2 identity matrix).

Curvature of connection. Any parallel-transported vector along a closed path ∂R around a simply connected region $R \subset M$ accumulates a rotation angle called the holonomy of the closed path. Given a connection 1-form ω, one can use the Stokes' theorem to express the holonomy as the integral of $-d\omega$ over R, independent of the choice of the local frames. This quantity $-d\omega$ is often called the curvature K of the connection and, in the particular case of Levi-Civita connection, it becomes the conventional notion of (2-form) Gaussian curvature.

Geometric decomposition. Due to the linearity of the covariant derivative, the operation $\nabla \mathbf{u}$ represents a 2-tensor field on M. By denoting the reflection matrix through

$$F = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

and omitting local bases for clarity, the matrix representation of $\nabla \mathbf{u}$ can be rearranged into *four geometrically relevant terms*:

$$\nabla \mathbf{u} = \frac{1}{2}[I\nabla \cdot \mathbf{u} + J\nabla \times \mathbf{u} + F\nabla \cdot (F\mathbf{u}) + JF\nabla \times (F\mathbf{u})], \tag{4}$$

where $J\nabla \times \mathbf{u}$ (measuring the curl of $\mathbf{u}$) is the only antisymmetric term. Moreover, we can rewrite this decomposition as a function of two other relevant derivatives:

$$\nabla \mathbf{u} = \partial \mathbf{u} + F\bar{\partial}\mathbf{u},$$

where the holomorphic derivative $\partial \equiv \frac{1}{2}(I\nabla \cdot + J\nabla \times)$ contains divergence and curl of the vector field, neither of which depends on the choice of local frame; whereas the Cauchy-Riemann operator (or complex conjugate derivative) $\bar{\partial} \equiv \frac{1}{2}(I\nabla \cdot F + J\nabla \times F)$ depends on the choice of frame. Due to the use of reflection, $\bar{\partial}\mathbf{u}$ behaves as a 2-vector (also called a 2-RoSy) field.

Relevant Energies. Based on the decomposition of the covariant derivative operator in Eq. 4, we can also express the *Dirichlet energy* E_{D} of vector field as the sum of two meaningful energies:

$$E_{\mathrm{D}}(\mathbf{u}) = \frac{1}{2}\int_M |\nabla \mathbf{u}|^2 dA = \frac{1}{2}\left(E_{\mathrm{A}}(\mathbf{u}) + E_{\mathrm{H}}(\mathbf{u})\right).$$

The *antiholomorphic energy* E_{A} measures how much the vector field deviates from being harmonic, and the *holomorphic energy* E_{H} measures how much the field deviates from satisfying the Cauchy-Riemann equations:

$$\begin{cases} E_{\mathrm{A}}(\mathbf{u}) = \dfrac{1}{2}\displaystyle\int_M [(\nabla \cdot \mathbf{u})^2 + (\nabla \times \mathbf{u})^2] dA, \\ E_{\mathrm{H}}(\mathbf{u}) = \displaystyle\int_M (\bar{\partial}\mathbf{u})^2 dA. \end{cases} \tag{5}$$

As shown in [Knöppel et al. 2013], the difference between E_{H} and E_{A} leads to a boundary term and a term related to the connection curvature $K = -d\omega$:

$$E_{\mathrm{A}}(\mathbf{u}) - E_{\mathrm{H}}(\mathbf{u}) = \int_{\partial M} \mathbf{u} \times (\nabla \mathbf{u})\, dA + \int_M K|\mathbf{u}|^2 dA. \tag{6}$$

Note that complex numbers can also be used to express these equations by replacing the rotation matrix J by the complex number i, and the reflection matrix F by complex conjugation.

Part I: Face-based Representation of Vector Fields

The Basics

This chapter addresses the discretization of tangent vector fields on triangulations using a piecewise constant vector field per face. This representation leverages the fact that tangent planes are well defined on a flat triangle, even if the whole triangle mesh forms a non-flat manifold. Discrete tangent vectors are thus naturally encoded by assigning a single vector per face. After reviewing basic definitions, we describe the construction of discrete differential operators acting on face-based vector fields, present extensions to n-vector and polyvector fields, and discuss their applications to geometry processing.

1 Piecewise constant vector fields

Discretizing tangent vector fields first requires defining the notion of tangent spaces over triangulations. Despite the use of simplicial elements, the piecewise linear structure of triangle meshes is not particularly amenable to a canonical definition of tangent planes: in particular, two-dimensional meshes embedded in $\mathbb{R}^3$ present hinge and cone discontinuities along edges and at vertices respectively, making the concept of tangency over a discrete surface seemingly ambiguous and ill-defined. There is a silver lining, however: for a triangle, one can precisely and unambiguously determine a tangent plane on each point of the face as being the plane of the triangle itself. This observation motivates the representation of discrete tangent vector fields as a piecewise constant field, determined through a single vector per face. More concretely, our first representation of discrete tangent vector fields encodes a tangent vector $\mathbf{u}_t$ in each triangle t as a representative of the constant vector field lying in the supporting plane of t. This simple (and to certain extent, simplest) discretization was recognized as very convenient in, e.g., [Polthier and Preuss 2003; Tong et al. 2003; Wardetzky 2006], and it has been by now widely adopted in graphics applications ranging from texture synthesis to quad meshing. In many of these applications, the fact that the resulting vector field is clearly not continuous—and worse, not even properly defined on edges and vertices—is not an issue, as it is defined *almost* everywhere. Yet, this low-order, discontinuous description of vector fields will obviously be limited in accuracy, in particular for local derivatives.

Encoding. The formulation of piecewise constant tangent vectors requires the assignment of a local coordinate frame per triangle t to encode the representative tangent vector $\mathbf{u}_t$. Conceptually, these local frames can be arbitrarily chosen as long as they lie within the supporting plane of their respective triangle. A canonical choice is to set the frame by picking the unit vector $\mathbf{e}_t^1$ along one of the triangle's edges, and its orthogonal direction $\mathbf{e}_t^2 \equiv \mathbf{n}_t \times \mathbf{e}_t^1$, where $\mathbf{n}_t$ is the normal of the triangle. Then any tangent vector $\mathbf{u}_t$ on face t can be expressed as $\mathbf{u}_t = u_1\mathbf{e}_t^1 + u_2\mathbf{e}_t^2$, where scalars u_1 and u_2 indicate the components of the vector $\mathbf{u}_t$ in the frame $(\mathbf{e}_t^1, \mathbf{e}_t^2)$. Once a choice of frame per triangle has been made, we have a total of $2|F|$ degrees of freedom (DoFs) to define a face-based vector field.

Vectors as complex numbers. An alternative implementation treats face-based vectors as a complex number per face. In this notation, manipulating a vector $\mathbf{u}_t$ at triangle t can be succinctly written via complex number multiplication. For example, a rotation of the vector by an angle θ is simply expressed as $\mathbf{u}_t e^{\mathrm{i}\theta}$, where i denotes the imaginary complex number. In components, this is equivalent to:

$$\mathbf{u}_t e^{\mathrm{i}\theta} = (u_1 \cos\theta - u_2 \sin\theta)\mathbf{e}_t^1 + (u_1 \sin\theta + u_2 \cos\theta)\mathbf{e}_t^2.$$

Discrete inner product. We can also leverage the Euclidean norm $\langle \cdot, \cdot \rangle$ inside each mesh face to define a discrete notion of the L_2 inner product between pairs of tangent vector fields $\mathbf{u}$ and $\mathbf{v}$; more specifically:

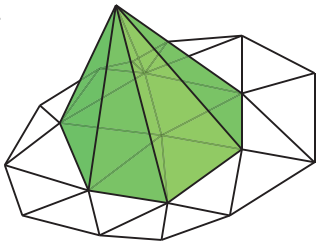
$$\langle\langle \mathbf{u}, \mathbf{v} \rangle\rangle = \sum_t \langle \mathbf{u}_t, \mathbf{v}_t \rangle |t|. \tag{7}$$

We further denote two tangent vector fields $\mathbf{u}$ and $\mathbf{v}$ *orthogonal* to each other *iff* $\langle\langle \mathbf{u}, \mathbf{v} \rangle\rangle = 0$. Note that this is a notion of orthogonality for *fields*, not between pointwise vectors. Nonetheless, it will be useful as it will allow us to decompose a tangent vector field into meaningful components later in this chapter.

2 Gradient vector fields

A special (and important) type of vector fields is what is known as a "gradient" field. It corresponds to tangent vector fields that are generated from scalar functions. In our discrete setting, it is not surprising that a constant vector $\mathbf{u}_t$ in an individual triangle t can be associated to the gradient of a linear function defined over t: by integration, we can construct a linear function over t, up to a constant, whose gradient matches $\mathbf{u}_t$. More generally, we can define *piecewise linear* functions over a triangle mesh by fitting together linear functions per triangle that share common values along edges. With these functions, we can now compute their gradients which define *piecewise constant* tangent vector fields. Next we recap two approaches to construct piecewise linear functions on triangle meshes and describe their respective discrete gradient operators.

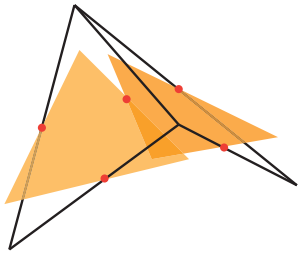
Vertex-based gradient. The first approach to define piecewise linear functions consists in sampling a function with one value per mesh vertex and interpolating these values linearly inside each triangle using barycentric coordinates. The barycentric coordinates for every triangle incident to a vertex v_i determines a piecewise linear basis function ϕ_i (also known as the *hat* basis function) whose support conforms to the vertex one-ring (see inset). The evaluation of a vertex-based piecewise linear function f at a location $\mathbf{p}$ with barycentric coordinates $\{\phi_i(\mathbf{p}), \phi_j(\mathbf{p}), \phi_k(\mathbf{p})\}$ inside a triangle ijk is thus given as:



$$f(\mathbf{p}) = f_i \phi_i(\mathbf{p}) + f_j \phi_j(\mathbf{p}) + f_k \phi_k(\mathbf{p}).$$

This scheme is commonly used in *conforming* finite element methods, since the interpolation at any point along an edge is exactly the same computed from either triangle adjacent to the edge. By differentiating these vertex-based bases, we obtain a family of face-based tangent vector fields spanned by $|V|-1$ DoFs, where one DoF was lost since a constant can be added to a function without changing its resulting gradient.

Edge-based gradient. Piecewise linear functions can also be constructed by assigning a value per edge midpoint, instead of vertices. In this less common discretization of scalar functions, the interpolation is still based on barycentric coordinates, but now associated with the interior triangle formed by edge midpoints [Wardetzky 2006]. As a consequence, linear functions defined in neighboring triangles coincide at the edge midpoint, but their evaluation at any other location along the common edge generally differs (see inset). Due to these discontinuities, piecewise linear basis functions ψ_{ij} associated to edge midpoints are commonly referred to as *non-conforming* finite element bases, with support restricted to the stencil of their respective edges. The evaluation of a piecewise linear function f at a point $\mathbf{p}$ with barycentric coordinates $\{\psi_{jk}(\mathbf{p}), \psi_{ki}(\mathbf{p}), \psi_{ij}(\mathbf{p})\}$ inside triangle ijk is then expressed as:



$$f(x) = f_{jk} \psi_{jk}(\mathbf{p}) + f_{ki} \psi_{ki}(\mathbf{p}) + f_{ij} \psi_{ij}(\mathbf{p}).$$

Similar to the vertex case, edge-based gradients also span a set of face-based tangent vector fields, now with $|E|-1$ DoFs.

Vertex vs. edges bases. At this point, the reader may have noticed that the ubiquitous use of barycentric coordinates ties together the gradient of conforming bases $\nabla\phi_i$ to the gradient of non-conforming bases $\nabla\psi_{ij}$. In fact, one can show that $\psi_{ij} = \phi_i + \phi_j - \phi_k$ within a triangle t_{ijk} and consequently:

$$\nabla\phi_i = -\frac{1}{2}\nabla\psi_{jk} = \mathbf{n}_{ijk} \times \mathbf{e}_{jk}/2|t_{ijk}|,$$

As an implication, any vertex-based piecewise linear function can be expressed as an edge-based piecewise linear function: given vertex values f_i, the edge values $f_{ij} = (f_i + f_j)/2$ result in the same piecewise linear function. Hereafter, we denote by A the averaging operator from vertex to edge values, encoded as a $|E| \times |V|$ matrix with non-zero entries $A_{ij,i} = A_{ij,j} = 1$ for each edge e_{ij}.

Rotated gradient fields. Starting from either a vertex-based or edge-based gradient field, one can also generate piecewise constant tangent vector fields by computing the orthogonal complement of each gradient vector within its own triangle. More formally, we define the 90° rotation of a vector $\mathbf{v}_t$ within triangle t to be the vector $J\mathbf{u}_t \equiv \mathbf{n}_t \times \mathbf{u}_t$. For any function f, the set of rotated gradient vectors $J(\nabla f)_t$ defines a piecewise constant field, that we will abusively denote as $J\nabla f$ for convenience. Note that a rotated gradient field *cannot* be a gradient field itself: it is a simple algebra exercise to show that rotated gradient fields are, in fact, orthogonal to gradient fields since $\langle\langle \nabla f, J\nabla g \rangle\rangle = 0$, for any conforming or non-conforming functions f and g.

3 Discrete Differential Operators

Equipped with two types of discrete scalar functions and our discrete, face-based tangent vector fields, we can now discuss the discretization of local derivative operators on triangulated surfaces, including divergence, curl and the Laplacian. Since our vector fields are not even continuous, one can only define derivatives in a weak form, i.e., as local integrations of derivatives that represent local estimates of derivative quantities. Such operators were previously derived using conforming finite element methods (see, e.g., [Polthier and Preuss 2000; Tong et al. 2003]). Instead, we mostly follow the work of [Wardetzky 2006] and provide a geometric interpretation based on a finite volume formulation.

Fluxes and circulations. Divergence and curl of vector fields are basic tools in vector calculus that measure, respectively, the flux and circulation of a tangent vector field around infinitesimal loops. This geometric interpretation serves as the foundation for a finite volume discretization of these operators on triangle meshes. As illustrated in the Fig. 2, the divergence (resp., curl) can be discretized as the *integrated* flux (resp., circulation) of the vector-per-face field across (resp., around) each edge stencil.

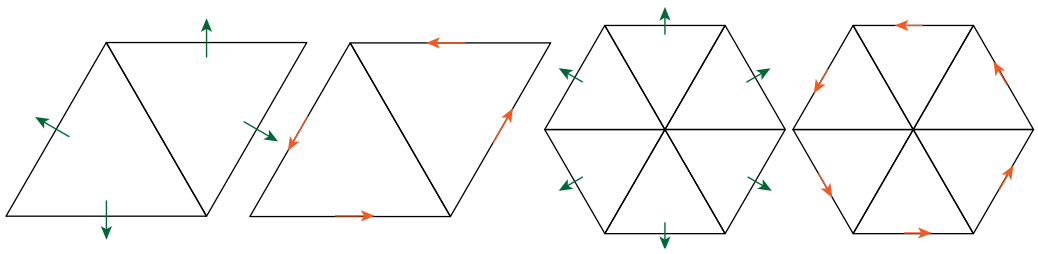


Figure 2: ***Flux & Circulation:*** *Discrete divergence and curl operators can be computed by integrating the total flux and circulation, respectively, of a face-based vector field around edge stencils (left) and vertex one-rings (right).*

These discrete operators are thus implemented via the following expressions:

$$\begin{cases} [\nabla \cdot \mathbf{u}]_{ij} = \langle \mathbf{u}_{ijk}, \mathbf{n}_{ijk} \times \mathbf{e}_{ij} \rangle - \langle \mathbf{u}_{jil}, \mathbf{n}_{jil} \times \mathbf{e}_{ij} \rangle, \\ [\nabla \times \mathbf{u}]_{ij} = \langle \mathbf{u}_{jil}, \mathbf{e}_{ij} \rangle - \langle \mathbf{u}_{ijk}, \mathbf{e}_{ij} \rangle. \end{cases} \tag{8}$$

Intuitively speaking, these edge-based discretizations quantify the (dis)continuity of face-based vectors *across* and *along* mesh edges. Similarly, one can also use finite volumes to discretize the divergence and curl as the total flux and circulation around vertex stencils this time, resulting in:

$$\begin{cases} [\nabla \cdot \mathbf{u}]_{i} = -\sum_{t_{ijk} \supset v_i} \langle \mathbf{u}_{ijk}, \mathbf{n}_{ijk} \times \mathbf{e}_{jk} \rangle, \\ [\nabla \times \mathbf{u}]_{i} = \sum_{t_{ijk} \supset v_i} \langle \mathbf{u}_{ijk}, \mathbf{e}_{jk} \rangle. \end{cases} \tag{9}$$

Discrete Laplacians of scalar fields. Since the continuous Laplacian operator is the divergence of the gradient operator, we discretize the Laplacian operator on a triangle mesh by composing negated discrete divergence and gradient operators. (The negation is introduced to make the operator positive-definite, thus easier to handle by solvers.) In the case of non-conforming bases, we obtain the edge-based Laplacian expressed per edge e_{ij} as:

$$[\Delta f]_{ij} = \sum_{t_{ijk} \supset e_{ij}} 2 \cot \alpha_{ijk} (f_{ij} - f_{jk}) + 2 \cot \alpha_{kij} (f_{ij} - f_{ki}). \tag{10}$$

Similarly, the vertex-based Laplacian operator leads to the usual cotan formula [MacNeal 1949]:

$$[\Delta f]_{i} = \sum_{e_{ij} \supset v_i} \tfrac{1}{2} \left(\cot \alpha_{jki} + \cot \alpha_{ilj} \right) (f_i - f_j). \tag{11}$$

Properties. Our edge-based and vertex-based discretizations verify basic vector calculus identities. For instance, it is easy to show that our discrete operators satisfy $\nabla \times (\nabla f) = \nabla \cdot (J \nabla f) = 0$, for either conforming or non-conforming functions f. Consequently, we can mimic the continuous picture and classify gradient vectors as curl-free fields, and rotated gradients as divergence-free fields. This gives a very concrete, physical meaning to gradient and rotated gradient fields: the former controls the sources and sinks of a flow, while the latter identifies vortical structures. Moreover, the formulations of divergence and curl satisfy Stokes' theorem by construction. In particular, we can use this discrete version of Stokes' theorem to derive a discrete Dirichlet energy for conforming piecewise linear functions (ignoring the boundary terms for simplicity):

$$E_D(f) = \langle\langle \nabla f, \nabla f \rangle\rangle = -\sum_i f_i \left[\nabla \cdot (\nabla f) \right]_i = \sum_i f_i \left[\Delta f \right]_i .$$

A similar expression holds for non-conforming piecewise linear functions too. Finally, since conforming functions span a subspace of non-conforming functions, we can expand vertex-based operators as a linear combination of edge-based operators via the averaging matrix A:

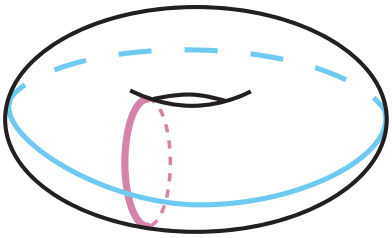
$$\begin{cases} \left[\nabla \times \mathbf{u} \right]_v = A^T \left[\nabla \times \mathbf{u} \right]_e , \\ \left[\nabla \cdot \mathbf{u} \right]_v = A^T \left[\nabla \cdot \mathbf{u} \right]_e , \\ \left[\Delta f \right]_v = A^T \left[\Delta f \right]_e A. \end{cases} \tag{12}$$

4 Orthogonal decomposition

So far we have discussed the construction of piecewise constant vector fields via discrete gradient and rotated gradient fields. We also pointed out that vertex-based gradient fields are rather limited in the

sense that they are just a special case of edge-based gradient fields. On the other hand, we mentioned in Sec. 1 that the total space of vector-per-face fields is completely encoded by $2|F|$ DoFs. In this section, we will show how these different families of piecewise constant vectors fit together in order to span the whole space of face-based vector fields. And we will see that the topology of the domain now matters quite dramatically.

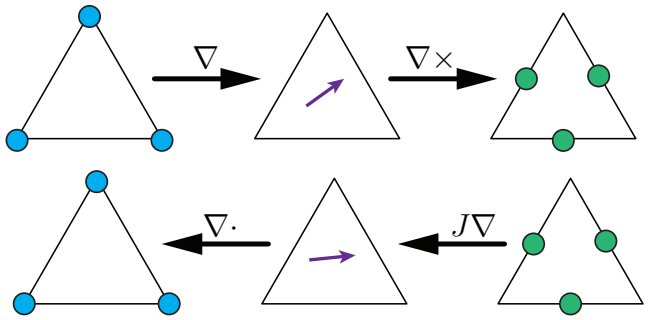
Helmholtz-Hodge decomposition. The notion of L_2 inner product of tangent vector fields given in Eq. (7) provides a very natural (and well known!) linear splitting of the space of vector fields on a smooth manifold. In the case of surfaces, this splitting implies that a tangent vector field can be orthogonally decomposed into a gradient, a rotated gradient, and a harmonic term, where gradients are curl-free, rotated gradients are divergence-free, and harmonic fields are both curl and divergence-free. The extra harmonic term corresponds to non-integrable vector fields, i.e., fields that are not associated to functions. A direct implication of this result is that an arbitrary surface with genus g must have a linear space of harmonic vector fields of dimension $2g$, one for each non-contractible loop generator (e.g., colored loops in the inset), thus exhibiting a *topological* constraint on the space of vector fields. It is far from obvious that this continuous property carries over to the discrete setting. In fact, we did not even mention topology when we defined our discrete operators.



Inadequacy of unified bases. In many discretization methods of vector fields, gradient and rotated gradient fields are assumed to be of the same dimension, both coming from *either* vertex-based, *or* edge-based scalar basis functions. This is the approach adopted, for instance, in plain (conforming) linear finite element methods for vector fields such as [Tong et al. 2003]. This simple space for scalar functions, however, comes at the cost of invalidating the Helmholtz-Hodge decomposition in the discrete setting. For instance, if we restrict the discretization to conforming scalar functions, gradient and rotated gradient vectors cover a space of dimension $2(|V|-1)$, thus inflating the size of the space of harmonic fields to $2(|F|-|V|+1)$, which is always more than $2g$; a similar issue appears if one picks non-conforming scalar functions. This indicates that, in order to mimic the smooth structure of tangent vector fields, one must be much more careful in how we represent gradient and rotated gradient fields.

Mixed finite elements. The work of [Polthier and Preuss 2003; Wardetzky 2006] pointed out that, for a boundaryless triangle mesh with genus g, the dimension of the per-face vector fields can be rewritten as $2|F| = (|V|-1) + (|E|-1) + 2g$ — using Euler's formula $|V|-|E|+|F|=2-2g$ and the fact that each edge is shared by two triangles, implying $3|F|=2|E|$. Therefore, a discrete notion of Helmholtz-Hodge decomposition can be actually formulated by properly mixing the space of constant vectors generated by conforming *and* non-conforming functions. More precisely, if conforming functions are chosen to span gradient fields, then non-conforming functions must be used to span rotated gradient fields (or vice-versa) to make the Hodge decomposition hold in the discrete setting. This implies that vertex and edge-based discrete operators must be interleaved: for gradient fields derived from conforming functions, the divergence must be defined per vertex and the curl per edge (or vice-versa). The diagram below summarizes the chain complex formed by these mixed discrete operators.

Extracting orthogonal components. We now have the discrete tools to decompose an arbitrary piecewise constant vector field into gradient, rotated gradient and harmonic terms properly. Following [Polthier and Preuss 2000; Tong et al. 2003], we make use of a variational formulation to break down an arbitrary piecewise constant vector field $\mathbf{u}$ into orthogonal components $\{\nabla f, J\nabla g, \mathbf{h}\}$. We first extract the gradient part by solving for the conforming function f whose gradient best approximates $\mathbf{u}$



in the L_2 sense:

$$\min_f \sum_t |t| \|(\nabla f)_t - \mathbf{u}_t\|^2.$$

Due to Stokes' theorem, the minimizer f is found via the discrete, vertex-based Poisson equation:

$$[\Delta f]_i = -[\nabla \cdot \mathbf{u}]_i . \tag{13}$$

Similarly, we can extract the rotated gradient by finding the non-conforming function g whose rotated gradient best approximates $\mathbf{u}$ in the L_2 sense:

$$\min_g \sum_t |t| \|(J\nabla g)_t - \mathbf{u}_t\|^2,$$

which now leads to an edge-based Poisson equation:

$$[\Delta g]_{ij} = -[\nabla \times \mathbf{u}]_{ij} . \tag{14}$$

Lastly, the remaining part determines the harmonic term: $\mathbf{h} = \mathbf{u} - \nabla f - J\nabla g$. It is worth pointing out that the orthogonality among these three components via the discrete inner product ensures their uniqueness once boundary conditions are chosen for the Poisson solves (see below). Fig. 3 shows an example of a vector field decomposed into gradient and rotated gradient fields.

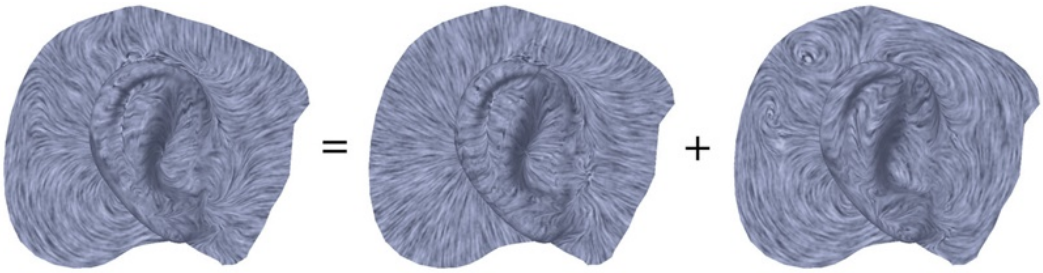


Figure 3: ***Helmholtz-Hodge decomposition:*** *A face-based vector field (left) in a simply-connected triangle mesh is orthogonally split into a curl-free (center) and a divergence-free component (right). Vector fields visualized using LIC [Cabral and Leedom 1993].*

Vector Laplacian. Piecewise constant harmonic vector fields can also be identified as the minimizers of the anti-holomorphic energy E_{A} in Eq. (5). This energy penalizes the squared norm of the divergence and curl of a face-based field $\mathbf{u}$, and is discretized as:

$$E_A(\mathbf{u}) = \sum_{v_i} \frac{[\nabla \cdot \mathbf{u}]_i^2}{a_i} + \sum_{e_{ij}} \frac{[\nabla \times \mathbf{u}]_{ij}^2}{a_{ij}}, \tag{15}$$

where a_i and a_{ij} indicate the (lumped) area per vertex v_i and edge e_{ij}, respectively (see Sec. 2.3 of the Introduction chapter). Due to the linearity of the operators, this energy is a quadratic function $\mathbf{u}^t L\mathbf{u}$, and the matrix L is called the (vector) de-Rham Laplacian. Even though the entries L depend directly on the frames of triangles, the solution is invariant to the choice of local frames. Moreover, it is worth pointing out that the de-Rham Laplacian is *not* the same as the Bochner Laplacian that derives from the norm of the covariant derivative reviewed in Sec. 3.4 of the Introduction chapter.

Boundary conditions. The flux and circulation of a face-based vector field at the mesh boundary are prescribed by controlling boundary vertex and edge values of the piecewise linear functions spanning the gradient and rotated gradient fields in the Poisson solves. Setting Neumann boundary condition for rotated gradient fields, for instance, corresponds to fixing edge boundary values, while forcing tangential components at boundaries for gradient fields is achieved via fixing boundary nodal values.

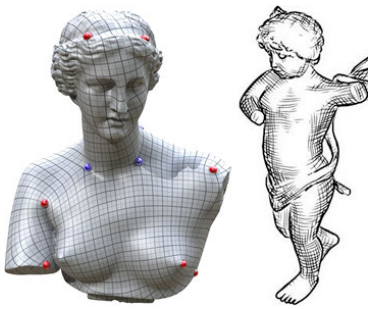
5 Generalizations of vector fields

Tangent vector fields have been extended to more general notions of fields, such as n-direction, n-RoSy, and n-polyvector fields, that are particularly useful in various geometry processing contexts. Next we recapitulate how to represent these generalizations based on piecewise constant vectors per face.

N-RoSy fields. Of particular importance for geometry processing is the case of n-vector fields, that indicate n *rotationally symmetric* fields, or n*-RoSy* fields for short. More concretely, an n-RoSy field describes, at any point $\mathbf{p}$ of a smooth surface, n replicates of a vector $\mathbf{u_p}$ rotated by angles that are multiples of $2\pi/n$, i.e., the set:

$$\{\mathbf{u_p}, \mathbf{u_p}e^{\mathrm{i}(2\pi/n)}, \ldots, \mathbf{u_p}e^{\mathrm{i}(2\pi/n)j}, \ldots, \mathbf{u_p}e^{\mathrm{i}(2\pi/n)(n-1)}\},$$

where we used complex numbers to denote rotations within the triangle's supporting plane. These fields are commonly used, for instance, to guide hatching lines and crosses on surfaces [Hertzmann and Zorin 2000], or in quad meshing by prescribing the local alignment and scale of the quad layout [Bommes et al. 2009; Panozzo et al. 2012] (see inset). Notice that any RoSy field at a point $\mathbf{p}$ can be completely identified with a representative vector $\mathbf{u_p}$. RoSy fields are thus a generalization of vector fields, and a piecewise constant description is, once again, particularly convenient since the flatness of the triangles makes it easy to define multiple vectors in any point inside a triangle. We also point out that the special case of *unit* n-RoSy fields corresponds to the notion of n*-direction fields*.



N-polyvector fields. A more recent extension of n-RoSy fields, called n*-polyvector* fields, consists of an *arbitrary* set of n vectors per triangle [Diamanti et al. 2014]: an n-polyvector at a triangle t is formed by the list of vectors $\{\mathbf{u}_t^1, ..., \mathbf{u}_t^n\}$ identified as the roots of an n-th order complex polynomial

$$p_t(\mathbf{z}) = (\mathbf{z} - \mathbf{u}_t^1)...(\mathbf{z} - \mathbf{u}_t^n).$$

Note that an n-RoSy field becomes just a special case of n-polyvector; but in contrast to RoSy fields, polyvector fields do not restrict the lengths and the angles between vectors at the given location. A measurement of the differences between two polyvector fields can now be carried out through a function of the coefficients of the two associated complex polynomials.

6 Connection, Parallel Transport and Singularities

The covariant derivative measures the rate of change of a vector field from one point to a nearby point on a surface. This is the geometric notion of "local derivative" of a vector field on a surface, since it

can no longer be as simple as a direct comparison of components as it involves different tangent planes. Divergence and curl are parts of this notion of derivative. In order to compute the covariant derivative, one first need a way to turn a vector $\mathbf{u}(\mathbf{p})$ stored in the frame at point $\mathbf{p}$ into an "equivalent" vector $\tilde{\mathbf{u}}(\mathbf{q})$ in the frame of a nearby point $\mathbf{q}$; only then can the similarity of nearby vectors be quantified through the differences of their components in a common frame at $\mathbf{q}$. The process of transporting a vector between nearby frames is known as *parallel transport*. Given an orthonormal frame $(\mathbf{e}, \mathbf{e}^{\perp})$ at $\mathbf{p}$, it amounts to prescribing an infinitesimal rotation in the tangent plane that will align this frame to the frame at $\mathbf{q}$. The rotation to align the frames is called a *connection* (see Sec 3.3 in the Introduction chapter) as it "connects" nearby tangent planes. The connection that corresponds to a notion of parallelism induced by the embedding space turns out to depend only on the surface metric: this metric-induced connection is called the Levi-Civita connection [Spivak 1979].

Discrete connection The discretization of the Levi-Civita connection, along with its induced parallel transport, is particularly simple for piecewise constant vector fields on triangle meshes. By leveraging the Euclidean structure of the triangles and the fact that any pair of triangles can be isometrically flattened via the hinge map, the discrete parallel transport from one face to the next reduces to a simple translation after the hinge map brings the two faces to a common plane. We can further define an *arbitrary* connection by simply storing an extra *angle per pair of adjacent triangles* (or equivalently, for each oriented *dual* edge [Crane et al. 2010]): as shown in Fig. 4, the associated parallel transport is thus found by unfolding the pair of triangles, translating the vector from one face to the other now that they are in the same plane, then rotating the resulting vector within the triangle by the prescribed angle, and folding the triangles back to 3D. Therefore, the discrete Levi-Civita connection simply corresponds to an assignment of a zero rotation for each dual edge. Note that this encoding of a discrete connection as the deviation angle from the Levi-Civita connection amounts to defining a discrete dual 1-form as we will discuss in the next chapter.

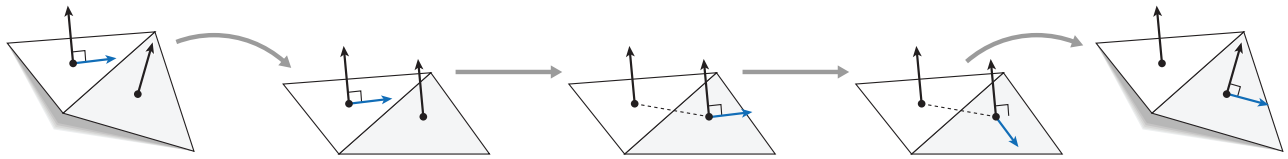


Figure 4: ***Parallel Transport:*** *For piecewise constant vectors, the parallel transport consists of: unfolding a pair of triangles to a common plane via the hinge map aligning the (black) face normals (left), translate the (blue) tangent vector from one triangle to another (left-center), rotating the vector according the the discrete connection (center-right), and then folding back the triangles to 3D (right).*

Discrete holonomy. Starting from an arbitrary vector, the parallel transport around a loop reveals an angle defect between the initial and the final vector known as *holonomy*, which measures the curvature of the connection. In our face-based representation, a discrete notion of holonomy can be computed around the loop of triangles incident to a vertex; it is a simple exercise to see that the holonomy of the Levi-Civita connection returns the angle defect corresponding to the discrete, vertex-based Gaussian curvature. For meshes with non-zero genus, the discrete holonomy can be computed around non-contractible loop generators as well.

Smoothness evaluation. From a face-based vector field, one can directly compute the difference of the vectors between t_{ijk} and t_{jil} based on the components along and across the common edge e_{ij}, namely $\{\langle \mathbf{u}_{ijk}, \mathbf{e}_{ij} \rangle, \langle \mathbf{u}_{ijk}, \mathbf{n}_{ijk} \times \mathbf{e}_{ij} \rangle\}$ and $\{\langle \mathbf{u}_{jil}, \mathbf{e}_{ij} \rangle, \langle \mathbf{u}_{jil}, \mathbf{n}_{jil} \times \mathbf{e}_{ij} \rangle\}$. These terms, from which the divergence and curl operators can be assembled, can be seen as the discrete analogue of covariant derivative for face-based vector fields. This is, however, a limited notion of covariant derivative which ignores curvature at vertices, but the simplicity of these terms is quite attractive in practice. We can also

choose to measure the angle between a vector and a parallel transported vector from a nearby point: these angles reveal how far from parallel transport the field is, thus defining an alternative (nonlinear) notion of smoothness that is particularly suitable for direction fields since magnitude is irrelevant. One may verify that the sum of these deviation angles around a vertex has to be $2\pi m$, where m is an integer. When m differs from 0, the vector field is said to have an *index-m singularity* at the vertex. For more general n-direction fields, the index is a multiple of $1/n$ instead. It is important to notice that the total sum of these singularity indices is constrained by the topology of the surface, as stated by the Poincaré-Hopf index theorem: a design approach for vector fields should take this topological condition into account to avoid surprises in the results!

Optimizing smooth vector field with singularities. In case singularity locations and indices are prescribed, the work of [Crane et al. 2010] showed that the construction of a smooth face-based n-RoSy field can be addressed as a quadratic minimization with linear equality constraints on the deviation angles. This result was reformulated in [de Goes and Crane 2010; Crane et al. 2013] as a series of sparse linear solves. If one wishes to automatically determine the locations and the indices of singularities that lead to a smooth face-based field, various non-convex optimizations can be used, involving a trigonometry-based energy [Hertzmann and Zorin 2000], unit constraints [Ray et al. 2006; Palacios and Zhang 2007; Ray et al. 2009], or integer variables [Ray et al. 2008]. Diamanti et al. [2014] also recently introduced an energy that mixes angle and component differences in order to generate smooth polyvectors fields. Unfortunately, these non-linear approaches not only are computationally intensive, but also lack any guarantee on the global optimality of the results.

7 Limitations

In many ways, the simplicity of the face-based representation of vector field is both its best and worst trait. With very local and intuitive derivatives, implementation of geometry processing algorithms can be exceedingly simple in this framework. The elegant definition of discrete connection is also a major attraction. However, because the vector field is only piecewise constant, some first and second order derivatives are just too ill defined to be reliably computed. As a consequence, finding a globally smoothest field with given user constraints becomes a numerically inefficient process, as expected from the use of low order basis functions. No free lunch!

To close this chapter, we recap the list of the most important pros and cons of the face-based representation of vector fields to consider when one is faced with a decision to pick the simplest vector field discretization that fits his or her needs.

Pros

- Trivial evaluation in triangles.
- Simple differential operators leveraging mixed FEM literature.
- Simple Helmholtz-Hodge decomposition.
- Generalization to n-vector fields is trivial.

Cons

- Requires a choice of frame per triangle.
- No clear definition of vector at vertices and along edges.
- Non-convex and divergent smoothness measures for RoSy fields.
- Limited notion of covariant derivative.

Part II - Edge-based Representation of Vector Fields

The Basics

As discussed in the previous chapter, smoothly varying tangent vector fields can be approximated using one vector per face over a triangulation. This common representation, however, suffers from various shortcomings that makes it not always the most convenient framework to create and manipulate tangent vector fields on surface meshes. In particular, storing a vector in a triangle necessitates a local (but arbitrary) basis frame–a rather simple requirement to address, but an inconvenience nonetheless. Even if one uses the scalar-based Hodge decomposition described in the previous section as a surrogate for this piecewise-constant vector field (via vertex and edge values), the remaining harmonic part makes the establishment of local frames an inevitable necessity for any non-trivial topology. Moreover, the use of piecewise constant basis functions is often an impediment to the reliable estimation of derivatives of the discrete vector field. A different approach to representing tangent vector fields on triangle meshes without any *local frame is to leverage edges as local directions with respect to which a vector field can be encoded. Moreover, the design of vector fields is greatly facilitated by the intuitive relationship between edge coefficients and the resulting vector field. We now review this edge-based alternative, along with its basic differential operators to allow for easy design and editing.*

1 Coordinate-free representation of vector fields

We begin with a brief introduction to the edge-based, frame-invariant way to encode tangent vector fields, which turns out to have its roots in algebraic topology. In order to bypass the use of frames, we will see that one can sample vector fields through integration over edges, thus turning a tangent vector field into one scalar per edge.

1.1 Origins

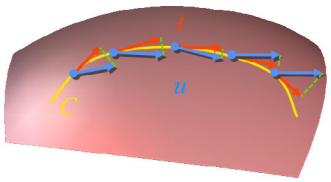
The edge-based representation of vector fields on triangle meshes for computer graphics [Fisher et al. 2007] draws from Cartan's theory of differential forms and integration, an important tool in the modern study of calculus on manifolds. Specifically, a discretization of the mathematical notion of differential forms, called *cochains*, was introduced in algebraic topology [Whitney 1957] as a finite-dimensional approximation space of differential k-forms. Since differential one-forms are uniquely identified to tangent vector fields (and vice-versa) via the musical operators $\flat$ and $\sharp$ [Desbrun et al. 2006], we can utilize this concept for vector field encoding as well. Our exposition in this chapter will, however, largely sidestep the algebraic topological nature of this approach, and focus instead on a hands-on explanation of its practical implementation and benefits.

1.2 Foundations

To understand how to convert vector fields into edge values, one must define a proper sampling procedure, i.e., a way to "measure" the local behavior of a vector field.

Line integrals. A key ingredient of this representation of tangent vector fields on meshes is the use of *line integration*. A line integral (sometimes called path integral) is the integral of some function along a

given curve. Integrating a scalar-valued function over a straight or smooth curve is commonplace in graphics; but integration of a vector-valued function (ore more specifically, of a vector field) along a curve may be less familiar to the reader. Fortunately, this concept is rather simple: the line integral $\int_C \mathbf{u} \cdot ds$ of a vector field $\mathbf{u}$ over an oriented curve C simply *adds up* the component of the vector field that is tangent to the curve. In other words, the line integral of a vector field integrates the scalar function $\mathbf{u} \cdot \mathbf{t}$ where $\mathbf{t}$ is the unit tangent vector to the curve C. One can thus say that this line integral evaluates how locally aligned the vector field is to the curve in addition to how large the magnitude of the vector field is. If the curve C happens to be a closed curve, then the line integral measures how much the vector field "circulates" around the curve C: no wonder then that we call the line integral $\int_C \mathbf{u} \cdot ds$ the *circulation* of $\mathbf{u}$ along C even when C is not closed. What is important in this definition is that this notion of circulation of a vector field along a curve is purely geometric: however one parameterizes the curve C and whichever coordinate patch or frame field is used to express the vector field, the circulation will be the same. That is, this is an objective measurement of a vector field along a curve. Note, however, that this circulation depends on the choice of the unit tangent vector, or equivalently, on the choice of the direction in which we traverse the curve C. Every simple curve has two orientations, one corresponding to one unit tangent vector $\mathbf{t}$ and the other corresponding to the opposite tangent vector $-\mathbf{t}$. Consequently, line integrals do switch sign if one reverses the orientation of C.



Edge circulations. An edge is but a curve: if you assume that a triangle mesh is an approximation of a smooth surface, then an edge represents a curve on this surface, and the union of all these curves form a simplicial decomposition of the surface. Even if you think of the triangle mesh as a surface itself, then this "edge as curve" idea still makes sense. Consequently, a vector field $\mathbf{u}$ can be turned into a series of local scalar values by simply computing the circulation of $\mathbf{u}$ over each (oriented) edge of the mesh. Knowing the circulation of the vector field on each edge is of course not as accurate as knowing the vector field itself: this circulation-based sampling will be oblivious to local variations of the vector field that are smaller in scale than the edge size. Worse, each circulation is locally unaware of the normal component of the vector field along the edge! However, the circulations on nearby edges provide a good approximation of this component. In fact, we will see that the aggregate of all circulations allows us to reconstruct explicitly a vector field that will exactly match the correct circulations. Circulation sampling is thus a convenient way to turn a vector field into a finite dimensional array of edge (scalar) values, which can then be stored without requiring basis frames. This is in sharp contrast with the vector-per-face discretization (described in the previous chapter), or the vertex-based coordinate-wise encoding (which will be reviewed in the next chapter).

Discrete forms. Edge values representing circulations are actually part of a more general notion of geometric sampling based on discrete differential forms and the associated Discrete Exterior Calculus (DEC) [Desbrun et al. 2006]. The central concept in this approach is the representation of antisymmetric tensor fields (also called differential forms) through *measurements on mesh cells*: a discrete 0-form represents a scalar function through its values at vertices (0-dimensional cells), while a discrete 2-form represents a density through its area integral over triangles (2-dimensional cells)—and as we just discussed above, a discrete 1-form represents a *tangent vector field* through its line integral along edges (1-dimensional cells)[2]. Since measurements are by their very nature independent of the basis frame in which they were represented, so are these scalar coefficients. One can even reconstruct continuous

[2]Note that we will interchangeably use the terms "1-form" and "tangent vector field", even if they are actually quite distinct in nature: they are in fact dual to each other, in the sense that a 1-form maps a vector field to a scalar via pairing, so a vector field can be turned into a 1-form (and vice versa) using the raising/lowering of their indices once a metric is given. In our context, we will simply identify vector fields and 1-forms as we can convert between them back and forth using the Euclidean metric of the embedding space.

fields through simple finite-element based interpolation. Finally, one can easily compute (discrete, or weak) derivatives on these discrete representations: we will see, for instance, that divergence and curl of discrete vector fields can be defined as discrete forms themselves that are simple linear combination of surrounding edge values, making this discrete form based representation particularly convenient.

Remark: Note that a regular 1-form is of a very different nature than the notion of discrete connection from the previous and the next chapter: while they both are encoded with a single scalar per edge, one is a discretization of a 1-form (or vector field), while the other is dependent on the choice of frames since it describes a way to align the frames of adjacent faces, and thus, to parallel transport a vector. As frame field cannot be continuously and *globally* prescribed in general, it is impossible to represent a connection globally as a 1-form. In addition, even locally, the two representations are only similar in 2-manifolds: while a vector field on a tetrahedral mesh can still be encoded with one value per edge, a connection on a 3D tetrahedralized manifold would now become a 3D rotation matrix per tetrahedron face.

1.3 Outline

In the remainder of this chapter, we review this edge-based alternative to represent vector fields on triangle meshes. We begin with a discussion of sampling and reconstruction of vector fields using edge values in Sec. 2, before describing in Sec. 3 some of the basic differential operators and functionals that are often useful when working with vector fields. We then detail in Sec. 4 how this edge representation can be utilized for easy design and editing of tangent vector fields on arbitrary triangulated surfaces. We conclude this chapter with a brief discussion in Sec. 5 of the various extensions of this representation that have been proposed in recent years.

2 Edge-based vector field representation

The edge-based approach to represent vector fields, introduced in graphics by [Wang et al. 2006; Fisher et al. 2007], has its mathematical foundations in Discrete Exterior Calculus (DEC) [Hirani 2003; Desbrun et al. 2006; Crane et al. 2013]. DEC defines discrete differential k-forms (here $k = 0, 1, 2$) on triangle meshes and expresses relevant operators such as divergence, curl, gradient, and Laplacian, as simple sparse matrices acting on intrinsic (coordinate-free) coefficients "living" on vertices, edges, and triangles. While we will show that continuous notions can be very naturally recast as discrete quantities on triangle meshes, we will reserve bold symbols for differential objects while discrete objects will be typeset non-bold to help keep them clearly separated.

2.1 From continuous to discrete vector fields

While the vector field representation will only use values on edges, it is convenient at this point to broaden the discussion to any k-form as it will become relevant when we define vector calculus (for instance, we will see that the notions of divergence and curl will be 0- and 2-forms, respectively). Converting a continuous k-form into a set of values on k-cells is done via *integration*, as we now review.

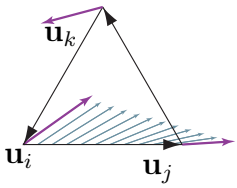
Sampling of forms. Discrete k-forms are given as scalars on k-cells, which represent discrete measurements of continuous k-forms

$$c_i = \boldsymbol{\omega}^0(\mathbf{p}_i), \quad c_{ij} = \int_{\mathbf{p}_{ij}} \boldsymbol{\omega}^1, \quad c_{ijk} = \int_{\mathbf{p}_{ijk}} \boldsymbol{\omega}^2.$$

Here $\boldsymbol{\omega}^k$ denotes a continuous k-form ($k = 0, 1, 2$) and the scalars are the corresponding measurements; i.e., c_i is the value of a scalar function ($k = 0$) at position $\mathbf{p}_i$ belonging to vertex v_i; c_{ij} is the line integral of a vector field $(\boldsymbol{\omega}^1)^\sharp$ ($k = 1$) along the segment $\mathbf{p}_{ij}$ belonging to edge e_{ij}; and c_{ijk} is an area integral of a density ($k = 2$) over $\mathbf{p}_{ijk}$ belonging to triangle t_{ijk}. We treat these coefficients as arrays c_v,

c_e, and c_t using an arbitrary but fixed indexing for the vertices ($v=1,\ldots,|V|$), edges ($e=1,\ldots,|E|$) and triangles ($t=1,\ldots,|T|$). In fact, the process of integration over simplices amounts to sample the continuous k-forms, effectively turning them into just a series of values. Note finally that vertex, edge, or face coefficients change sign if the orientation of their respective integration cells is changed.

Integral evaluation. Sampling 0-forms amounts to simply storing the value of the 0-form (function) at vertices. For 1-forms (vector fields) and 2-forms (density fields), the integration typically requires quadrature. Depending on the desired accuracy, approximations based on the evaluation of the form at one or multiple points can be used. One can therefore convert an arbitrary vector field into an edge-based vector field through simple dot products that generate the $|E|$ values required to store it, with for instance:



$$\int_{\mathbf{p}_{ij}} \mathbf{u}\cdot\mathbf{ds} = \frac{\mathbf{u}_i+\mathbf{u}_j}{2}\cdot\mathbf{p}_{ij}.$$

2.2 From discrete to continuous vector fields

Conversely, it is convenient to turn a set of edge values into a representable vector field. Fortunately, discrete k-forms can be interpolated with the aid of Whitney elements [Whitney 1957]. For 0-forms these are the standard piecewise-linear hat functions $\phi^v = \{\phi_i | v_i \in V\}$. For discrete 2-forms the corresponding Whitney interpolators are the constant functions $\phi^t = \{\phi_{ijk} | t_{ijk} \in T\}$ supported on individual triangles (Fig. 5). These piecewise-linear and piecewise constant interpolating functions are commonly used in graphics since its inception.

Vector field reconstruction. Reconstructing a vector field from edge values is, however, less common; yet it fits neatly within the usual finite-element view of field reconstruction from scalar data. Appropriate 1-form interpolators $\phi^e = \{\phi_{ij} | e_{ij} \in E\}$ can be defined as

$$\phi_{ij}(\mathbf{p}) = \phi_i(\mathbf{p})\boldsymbol{d}\phi_j - \phi_j(\mathbf{p})\boldsymbol{d}\phi_i.$$

Since $(\boldsymbol{d}\phi_i)^\sharp = \nabla\phi_i$ for the typical Euclidean metric, we can rewrite a Whitney basis function without exterior calculus operators as the local vector field:

$$\boldsymbol{\phi}_{ij}(\mathbf{p}) = \phi_i(\mathbf{p})\nabla\phi_j - \phi_j(\mathbf{p})\nabla\phi_i.$$

Each Whitney basis element ϕ_{ij}, or equivalent $\boldsymbol{\phi}_{ij}$, is supported on the two triangles incident to the given edge and varies linearly within them. (Note that basis elements associated with boundary edges are only supported on one triangle.) These edge basis functions are, indeed, interpolators for discrete 1-form

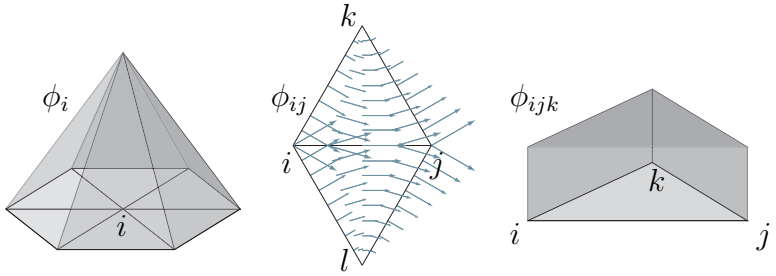


Figure 5: *The Whitney bases for 0- and 2-forms are standard piecewise linear hat functions and constant functions, respectively. The 1-form bases correspond to locally-supported vector field bases.*

data: the line integral of ϕ_{ij} is 1 along e_{ij} and 0 along all other edges. Therefore, from a discrete 1-form c_e given through edge coefficient c_{ij} one can generate a piecewise linear vector field $\mathbf{u}$ through:

$$\mathbf{u} = \sum_{e_{ij} \in E} c_{ij} \boldsymbol{\phi}_{ij},$$

and this vector field matches exactly the circulation c_{ij} over edge e_{ij}. It is in that sense that this reconstruction is considered an "interpolation".

Local evaluation. Given the expressions of ϕ_{ij}, the interpolated vector field $\mathbf{u}$ in a given triangle t_{ijk} is determined by only three incident edge coefficients c_{ij}, c_{jk}, and c_{ki}. With barycentric coordinates α_i, α_j, α_k (associated to vertices i, j, and k respectively) we can express within the triangle the local vector field $\mathbf{u} \in \mathbb{R}^2$ through

$$\mathbf{u}(\alpha_i, \alpha_j, \alpha_k) = \left((c_{ki}\alpha_k - c_{ij}\alpha_j)\ \mathbf{p}_{jk}^{\perp} + (c_{ij}\alpha_i - c_{jk}\alpha_k)\ \mathbf{p}_{ki}^{\perp} + (c_{jk}\alpha_j - c_{ki}\alpha_i)\ \mathbf{p}_{ij}^{\perp} \right) / 2\ |t_{ijk}|\ , \tag{16}$$

where $\perp$ indicates a $\pi/2$ rotation in the plane of t_{ijk}; we also used the property $\nabla\phi_i = \mathbf{p}_{jk}^{\perp}/(2|t_{ijk}|)$. This resulting expression can be understood as an equation returning a vector in $\mathbb{R}^3$ as well; however, since the vector lives in the plane of t_{ijk} it can equally well be treated as a 2D-vector equation when expressing it in a tangent plane frame. Moreover, in some applications vector fields computed on the surface are used in the texture (parameter) domain. In that case tangent vectors can be pushed forward through the Jacobian of the mapping from the surface to the texture domain. With the edge-based representation there is an even simpler approach: it suffices to replace all spatial quantities in Eq. (16) with their corresponding texture domain counterparts

$$\tilde{\mathbf{u}}(\alpha_i, \alpha_j, \alpha_k) = \left((c_{ki}\alpha_k - c_{ij}\alpha_j)\ \tilde{\mathbf{p}}_{jk}^{\perp} + (c_{ij}\alpha_i - c_{jk}\alpha_k)\ \tilde{\mathbf{p}}_{ki}^{\perp} + (c_{jk}\alpha_j - c_{ki}\alpha_i)\ \tilde{\mathbf{p}}_{ij}^{\perp} \right) / 2\ |\tilde{t}_{ijk}|\ ,$$

where we used $\tilde{}$ to indicate that these quantities are to be taken in the texture domain rather than on the mesh in 3D. This automatically accounts for the effects of the Jacobian. Note that this gives you a glimpse on the reason why the notion of forms is so important and beneficial in practical computations!

Continuity. The astute reader will have noticed that vector fields reconstructed with Whitney 1-form bases are piecewise linear, but in general not continuous along edges or at vertices. Indeed, the edge basis functions are not continuous at edges: only their projections along an edge are continuous, while the component normal to the edge may contain a discontinuity. While it is already an improvement compared to face-based representation (where tangential continuity at edges was not even present), this could be considered as a flaw, in particular on coarse meshes—and in fact, the next chapter will provide a final approach enforcing continuity. Yet, these edge elements have been used successfully for years ever since they were introduced in the engineering community by Nédélec [1980]. Their connection with finite element theory ensures that the Whitney-interpolated fields approximate the underlying smooth solution arbitrarily well as long as the mesh is suitably fine: Dodziuk et al. [1976] show that Whitney 1-forms can approximate any smooth vector field arbitrarily well in the L_∞ norm (see also [Arnold et al. 2006]). For very coarse triangulations, however, the normal jumps can be a potential issue in applications requiring continuity. We will point to recent work in Sec. 5 that can derive smoother basis functions through simple stationary subdivision rules, leading to reconstructed vector fields that are always continuous.

2.3 Discussion

Now that we have defined how to project a continuous form to its discrete realization via integration (this reduction is technically called the de Rham map) and how to reconstruct a continuous form from its discrete realization, we are set: since vector fields correspond to 1-forms, they are thus encoded as

one value per edge and can be easily reconstructed as actual vector fields if needed. Note that as we discussed early on, the discrete encoding via edge values does not need (or depend on) the definition of local frames. Even the edge orientation is not an inconvenience in practice: a mesh data structure always has a canonical direction for an edge—for instance, an edge orientation can be based on which vertex has the lower index.

While a convenient discrete encoding of vector fields is a nice thing to have, it serves very little purpose if it is not associated with simple operators that can compute some notion of derivatives. It allows us to measure, for instance, how "smooth" a discrete vector field is. This is where the edge-based representation shines: it comes with very simple operators that can be leveraged for a number of applications, as we now explain.

3 Operators on edge-based vector fields

The edge-based representation of vector fields also comes with discrete versions of important differential operators such as the Laplace-Beltrami operator, as well as a full discrete Hodge decomposition. Note that this machinery has been shown useful in settings such as surface parameterization (e.g., [Mercat 2001; Gu and Yau 2003; Gortler et al. 2006; Tong et al. 2006]) and physical simulation in 2D and 3D [Elcott et al. 2007]. A rigorous treatment of the connection between discrete and continuous settings can be found in the survey of Arnold et al. [2006], where finite element techniques are used to establish such essential properties as consistency, stability, and convergence of DEC-based approaches.

3.1 Basic operators

The main ingredients we need to compute differential operators on discrete k-forms are the discrete differential d and its L_2-adjoint δ, along with the Hodge star $\star$.

Discrete differential. The operator d, which maps k-forms to $(k+1)$-forms, is given by the transpose of the signed incidence matrices of the triangle mesh: $d_0 = (\partial^1)^T$ maps 0-forms (coefficients at vertices) to 1-forms (coefficients at edges), while $d_1 = (\partial^2)^T$ maps 1-forms (coefficients at edges) to 2-forms (coefficients at triangles). Here $(\partial^2)_{et} = \pm 1$ if edge e is incident on triangle t and their intrinsic orientations agree/disagree, and zero otherwise (and correspondingly for ∂^1). In standard vector calculus $d_0 \equiv \nabla$ and $d_1 \equiv \nabla\times$ and the fact that the boundary of a boundary is empty results in $d_1 d_0 = 0$, which in turn corresponds to the vector calculus fact that $\nabla \times (\nabla f) = 0$, for any function f. To simplify notation we will generally drop the subscript on d since the type of d, i.e., d_0 or d_1, follows from its argument—think of it as operator overloading in C++.

Discrete Hodge star. We also need inner products for discrete forms c_v, c_e, resp. c_t, which correspond to the L_2 inner products of continuous forms $\boldsymbol{\omega}^0$, $\boldsymbol{\omega}^1$, resp. $\boldsymbol{\omega}^2$. For our purposes the diagonal matrices

$$(\star_0)_{vv} = |v^*|/|v|, \quad (\star_1)_{ee} = |e^*|/|e|, \quad (\star_2)_{tt} = |t^*|/|t|,$$

are sufficient (the superscript * denotes the Voronoi dual of a given cell). These so called *diagonal Hodge star* matrices may be regarded as lumped mass matrices [Bossavit and Kettunen 1999] and are nothing more than ratios of dual to primal intrinsic volumes. Just like in the case of the operator d, we will use the symbol $\star$ to denote either $\star_0$, $\star_1$, or $\star_2$ depending on the type of its argument. The L_2 inner product between two discrete forms can then be represented as $\langle c_1, c_2 \rangle = c_1^T \star c_2$.

Discrete codifferential. We can now consider the L_2-adjoint of the continuous exterior derivative $\boldsymbol{d}$. This *co-differential* $\boldsymbol{\delta}$ maps $(k+1)$-forms to k-forms, and is defined in the absence of boundaries (see [Desbrun et al. 2006] for details) via Stokes' theorem:

$$\langle \boldsymbol{d}\boldsymbol{\omega}^k, \boldsymbol{\xi}^{k+1} \rangle = \langle \boldsymbol{\omega}^k, \boldsymbol{\delta}\boldsymbol{\xi}^{k+1} \rangle.$$

In the language of vector calculus, this codifferential operator corresponds to the well-known notion of divergence $\boldsymbol{\delta}_1 \equiv \nabla\cdot$. Now, the *discrete* co-differential operator is simply written as: $\delta_2 = \star_1^{-1} d_1^T \star_2$, which is the adjoint of d_1, and $\delta_1 = \star_0^{-1} d_0^T \star_1$, which is the adjoint of d_0.

These basic operators are rather simple to assemble based on an input mesh. We refer the reader to [Elcott and Schröder 2006] for a discussion of data structures to use to facilitate implementation.

3.2 Variational definition of harmonic vector fields

A vector field is called harmonic iff it is both curl- and divergence-free; in the language of 1-forms this corresponds in the continuous realm to requiring

$$\boldsymbol{d\omega}^1 = 0 \quad \text{and} \quad \boldsymbol{\delta\omega}^1 = 0.$$

(We ignore boundary issues for now and postpone their discussion to Sec. 4.4.) A harmonic vector field is thus particularly canonical in the sense that it contains no vortices or sinks or sources. One can "quantify" how close to harmonic the vector field corresponding to a given 1-form is by considering the bilinear form

$$\boldsymbol{E}(\boldsymbol{\omega}^1, \boldsymbol{\xi}^1) = \langle \boldsymbol{d\omega}^1, \boldsymbol{d\xi}^1 \rangle + \langle \boldsymbol{\delta\omega}^1, \boldsymbol{\delta\xi}^1 \rangle.$$

Indeed, a harmonic 1-form $\boldsymbol{\omega}^1$ satisfies $\boldsymbol{E}(\boldsymbol{\omega}^1, \boldsymbol{\omega}^1) = 0$ by construction, and only harmonic 1-forms have this property. The connection with the Laplace operator, $\Delta = \boldsymbol{\delta d} + \boldsymbol{d\delta}$, is established through its weak formulation: a form $\boldsymbol{\omega}$ has vanishing Laplacian if its inner product with any of a set of test forms $\boldsymbol{\xi}$ vanishes

$$0 = \langle \Delta\boldsymbol{\omega}, \boldsymbol{\xi} \rangle = \langle \boldsymbol{\delta d\omega}, \boldsymbol{\xi} \rangle + \langle \boldsymbol{d\delta\omega}, \boldsymbol{\xi} \rangle = \langle \boldsymbol{d\omega}, \boldsymbol{d\xi} \rangle + \langle \boldsymbol{\delta\omega}, \boldsymbol{\delta\xi} \rangle.$$

In other words, $\boldsymbol{E}$ describes the bilinear form appearing in the weak formulation of the (Hodge) Laplacian. It is, in fact, well known that $\boldsymbol{E}(\boldsymbol{\omega}^1, \boldsymbol{\omega}^1)$ measures the *Dirichlet energy* of the 1-form $\boldsymbol{\omega}^1$.

Translating the expression for $\boldsymbol{E}$ into the setting of discrete forms is now trivial, leading to:

$$E(c_e, c_e) = c_e^T M c_e = c_e^T \left(d_1^T \star_2 d_1 + \star_1 d_0 \star_0^{-1} d_0^T \star_1 \right) c_e, \tag{17}$$

where M is the discrete version of the 1-form Laplace operator. Note that $M = M^{\nabla\times} + M^{\nabla\cdot}$ consists of a first summand encoding the squared curl magnitude, while the second summand encodes the squared divergence magnitude.

Implementation. Practically speaking M is a straightforward assembly of diagonal matrices and signed incidence matrices. More concretely, for a given edge e_{ij} with associated 1-form coefficient c_{ij} a

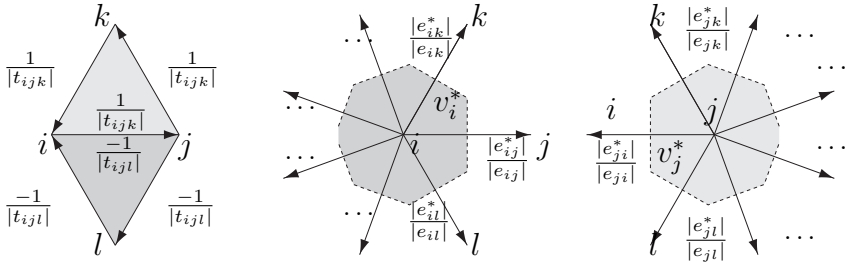


Figure 6: *The 1-form Laplace stencil, Eq.* (18)*, for a given e_{ij} depends on data at the two incident triangles (left) as well as the edges incident on v_i and v_j (middle and right).*

row of M reads as (see also Fig. 6)

$$M_{e_{ij}} = \frac{c_{ij} + c_{jk} + c_{ki}}{|t_{ijk}|} - \frac{c_{ij} + c_{jl} + c_{li}}{|t_{ijl}|} + \frac{|e^*_{ij}|}{|e_{ij}|} \left(\frac{|v_j|}{|v^*_j|} \sum_{e_{jl} \ni v_j} \frac{|e^*_{jl}|}{|e_{jl}|} c_{jl} - \frac{|v_i|}{|v^*_i|} \sum_{e_{ik} \ni v_i} \frac{|e^*_{ik}|}{|e_{ik}|} c_{ik} \right). \quad (18)$$

Note that M has an average of ≈ 11 nonzero entries per row.

3.3 Discrete Hodge Decomposition of Vector Fields

Given proper boundary conditions, any continuous 1-form can be written uniquely as an orthogonal sum

$$\boldsymbol{\omega}^1 = \boldsymbol{d\omega}^0 + \boldsymbol{\delta\omega}^2 + \boldsymbol{h}$$

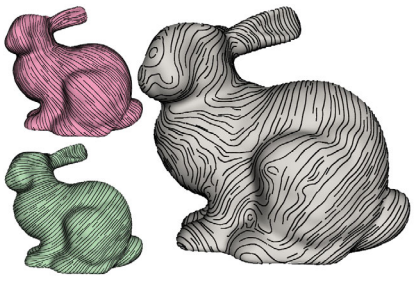
where $\boldsymbol{\omega}^0$ and $\boldsymbol{\omega}^2$ are scalar potentials (0- and 2-forms respectively), and $\boldsymbol{h}$ a harmonic 1-form. The latter is nontrivial only in the case of surfaces with genus $g > 0$, for which the space of such harmonic 1-forms is $2g$ dimensional. The corresponding decomposition for discrete 1-forms holds as well:

$$c_e = dc_v + \delta c_t + h. \quad (19)$$

Note that it mimics the continuous case quite neatly: the dimension of the discrete harmonic part h is also $2g$ for a surface of genus g as explained in [Desbrun et al. 2006]. This simple decomposition gives us access to the curl-free (dc_v) and divergence-free (δc_t) parts of a discrete tangent vector field—and this property will be key to the efficient design of vector fields on arbitrary surfaces as we will explain in Sec. 4.

3.4 More advanced operators

One major shortcoming of many techniques relying on discrete forms is their inability to also handle *symmetric* tensors. Since forms are by definition antisymmetric tensors, they do not offer a canonical way to deal with, e.g., metric fields which are symmetric, positive-definite tensor fields. This means that one can only handle the metric induced by the Euclidean embedding space of a mesh. In fact, the Hodge operator described above derived from this canonical metric since all volume measurements are performed in the Euclidean metric. Of course, altering the Hodge star is a way to indirectly work with arbitrary metrics, but it only skirts the issue. Recent work from de Goes et al. [2014] provides a discrete encoding of tensors that encompasses both symmetric and antisymmetric tensors in a consistent framework, thus extending the DEC methodology on triangle meshes. By leveraging a coordinate-free tensor decomposition extending the Hodge splitting of forms that we discussed in Sec. 3.3, they extend the DEC machinery to provide, e.g., generalized inner products and generalized Laplacians based on arbitrary Riemannian metrics. In the process, they define discrete versions of the *covariant derivative* and *Lie bracket* of vector fields represented by 1-forms—discrete operators that are not available for a face-based representation of vector fields. Indeed, the covariant derivative $\nabla \mathbf{u}$ of a vector field $\mathbf{u}$ can be decomposed (see Eq. (4) in the Introduction chapter) into two components: $\frac{1}{2}(\nabla \mathbf{u} - \nabla \mathbf{u}^t)$ that is antisymmetric (and corresponds to the curl of $\mathbf{u}$), and $\frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^t)$ that is symmetric (and called the Killing operator). So a clean expression of these two components can be expressed within the DEC context, which leads a discrete notion of covariant and Lie derivative. The inset illustrates the DEC-based covariant derivative for the pink/top vector field with respect to the green/bottom field, resulting in the right/grey vector field, visualized as streamlines. Interestingly, this generalization was also shown to



provide, a posteriori, the foundations behind the recent use of non-circumcentric orthogonal duals [Desbrun and de Goes 2014]. We also point out that the authors of [Ben-Chen et al. 2010] proposed to reformulate the Killing operator in terms of the well known Hodge Laplacian and an average Gaussian curvature per edge instead, using what is known as the Bochner technique. While finding better representations of discrete metrics is still an active field of research, it is comforting to see that the edge-based vector field representation is compatible with a simplex-based discretization of metric, hinting at its broad applicability.

4 Edge-based Vector Field Design

With the edge-based representation, the design of tangent vector fields on meshes can be achieved quite naturally. In this section, we discuss the approach of [Fisher et al. 2007] which leveraged the per-edge encoding of vector fields. In particular, we discuss the different constraints that are useful during design, and the linear equations that need to be solved.

4.1 Variational approach

Design requires immediate visual updates based on intuitive user interaction. With this important goal in mind, one can cast the design of vector field as the minimization of a quadratic energy of edge coefficients c_e given user constraints. Such a simple variational formulation will allow for fast updates based on user inputs. Moreover, the notion of curl and divergence of a tangent vector fields is visually important, as it leads to particularly noticeable features in the field. So one must derive a quadratic functional which, up to additional user inputs, provide good control over local curl and divergence. We will see now that our setup of edge values and their associated operators are particularly relevant in practice.

Consider first an arbitrary discrete 1-form $c_e = dc_v + \delta c_t + h$ (Eq. (19)). Its curl per triangle is $r_t := dc_e = d\delta c_t$ while its divergence per vertex is $s_v := \delta c_e = \delta d c_v$ since h is both curl- and divergence-free. Allowing arbitrary linear functionals Zc_e—which will account for constraints as discussed below—we arrive at a "stack" of matrices acting on the unknown vector of edge coefficients c_e

$$\begin{pmatrix} d \\ \delta \\ Z \end{pmatrix} c_e = \begin{pmatrix} r_t \\ s_v \\ c_z \end{pmatrix}$$

where the right hand side encodes the desired curl at triangles, divergence at vertices, and the inhomogeneous part of the linear functionals (c_z). The solution with minimum residual norm (in other words, the vector field that "best" fits the requirements) satisfies the associated normal equations

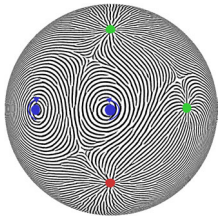
$$(\star\delta \;\; \star d \;\; Z^T W) \begin{pmatrix} d \\ \delta \\ Z \end{pmatrix} c_e = (\star\delta \;\; \star d \;\; Z^T W) \begin{pmatrix} r_t \\ s_v \\ c_z \end{pmatrix} \iff (M + Z^T W Z) c_e \;\; = \star\delta r_t + \star d s_v + Z^T W c_z.$$

where W is a diagonal weight matrix containing non-negative weights indicating the strengths of the constraints, i.e., how much each constraint should be enforced relatively to others.

4.2 Control of Sources, Sinks and Vortices

We first examine the setting where no additional constraints are given by the user (absence of any Z). In that case the right hand side can encode desired locations for sources and sinks through s_v and similarly for vortices through r_t as follows. Let s_v be a vector of vertex coefficients with positive entries for selected sink vertices and negative entries for selected source vertices with their magnitude representing strengths and satisfying a weighted (by Voronoi area) zero mean condition, i.e., $\bar{s}_v = A^{-1} \sum_i s_i |v_i^*| = 0$, where A is the total area of the mesh. In practice, if the user specifies

a vector s_v with $\bar{s}_v \neq 0$ we perturb it as $\hat{s}_i = s_i - \bar{s}_v$. This deals with "impossible" right hand sides and ensures that the computed solution c_e is optimal in the least-squares sense: the perturbation from s_v to $\hat{s}_v$ minimizes $\langle \delta c_e - s_v, \delta c_e - s_v \rangle$. For vortices we proceed similarly. Let r_t be a vector of (selected) triangle coefficients with positive entries for vortices whose orientation coincides with the triangle orientation and negative entries for opposing orientation (with $r_{ijk}/|t_{ijk}|$ indicating the strength). This time we enforce an unweighted zero mean for r_t, if not already satisfied, through a perturbation $\hat{r}_{ijk} = r_{ijk} - \bar{r}_t |t_{ijk}|$ for $\bar{r}_t = A^{-1} \sum_{ijk} r_{ijk}$, minimizing $\langle dc_e - r_t, dc_e - r_t \rangle$. If the user-supplied s_v and r_t satisfy the zero mean condition up front then the resulting vector field has *only* those sources, sinks and vortices—this follows from the existence and uniqueness of the solution to the underlying 0-form and 2-form Poisson problem, see [Desbrun et al. 2006]. The inset demonstrates this property for a sphere where two vortices (blue marks), one source (red mark) and two sinks (green marks) were set. For the horse (genus 0) in Fig. 7, when only a single source placed by the user, adjustment of s_v leads to a solution with additional singularities at the hoofs (left).



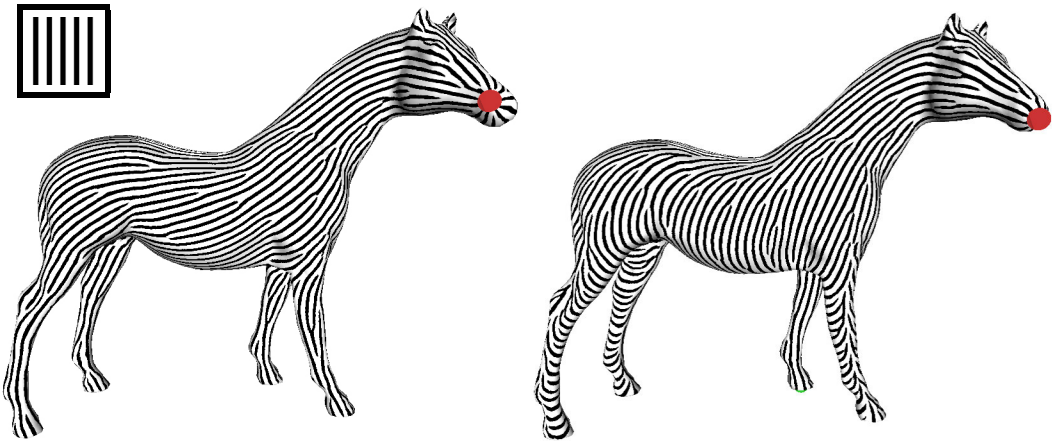


Figure 7: *Example of user-specified single source (left) and matching source/sink (right) on the horse. The least-squares solution in the case of only a single source places sinks at the hoofs. The inset shows the texture synthesis exemplar used for all visualizations in this section.*

4.3 Edge Constraints

While vortices, sources and sinks are important features in a vector field and can produce an overall field with little user supplied data, they do not provide enough detail control. For this we need direct constraints on the edges. This is where the extra Z terms become handy.

Consider now the case of using constraints Z. If both s_v and r_t are zero then we are seeking a field which is harmonic, i.e., free of curl and divergence, while approximating the given Z constraints. What constraints are useful? In the least squares setup, constraining an individual edge value amounts to the trivial row

$$Z c_e := c_{ij} = \text{chosen value}. \tag{20}$$

Fixing an edge coefficient in this manner leads to a vector field whose line integral along the edge matches the given value, but does not directly imply that the vector field is parallel to the edge. Stronger control though can be imposed by using several edge constraints in concert, see Fig. 8(left).

Vector constraints. At times it is desirable to specify a particular vector at a specific location as a constraint. Suppose we want to specify a vector $\mathbf{u}$ on some triangle t_{ijk}. Taking advantage of

Figure 8: *On the gargoyle model, a single edge constraint induces a global field; adding 3 closely spaced edge constraints creates a spiral. On the bunny model, a single source/sink pair produces the field on the left which is then reshaped with a curve constraint drawn on the body.*

$dc_e = c_{ij} + c_{jk} + c_{ki} = 0$ on t_{ijk}, i.e., the zero curl condition, and using Eq. (16) we find the desired constraints as

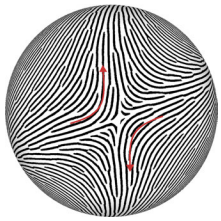
$$Z_1 c_e := c_{ij} = \mathbf{u} \cdot \mathbf{p}_{ij}, \quad Z_2 c_e := c_{jk} = \mathbf{u} \cdot \mathbf{p}_{jk}, \quad Z_3 c_e := c_{ki} = \mathbf{u} \cdot \mathbf{p}_{ki},$$

where $\mathbf{p}_{ab}$ refers to the edge ab seen as a vector and the inner products are taken in the plane of t_{ijk}.

Strokes. The most important means to control the appearance of a vector field are sequences of edge constraints given through a curve drawn on the mesh, signifying the vector field should follow it. Let $\mathbf{t}$ be the velocity vector to the curve as it crosses edge $\mathbf{p}_{ij}$, then

$$Z c_e := c_{ij} = \mathbf{t} \cdot \mathbf{p}_{ij}$$

gives the constraint for each edge crossed. In this manner, it is easy to place a saddle (inset image) or perform editing operations such as on the bunny in Fig. 8 (right).



4.4 Boundaries

To properly deal with boundaries the matrix M (Eq. (18)) must be modified according to the type of boundary condition we wish to enforce. Fisher et al. [2007] discussed two different types: *free* and *fixed angle* (see Fig. 9). For free boundaries neither fluxes across boundary edges, nor line integrals along them are constrained. The fixed angle boundary conditions, instead, enforce a zero line integral in a freely chosen direction relative to each boundary edge; special cases of this setting include *tangential* boundary conditions, which force the vector field to have zero flux across the boundary edges, and *normal* boundary conditions, which force the field to meet the boundary edges orthogonally. Since we control the circulation on edges, and from these circulations, we can even evaluate the flux through any edge, it is quite obvious that all these boundary conditions are rather simple to enforce. We refer the reader to the paper for detailed explanations of how to locally modify the matrix M near boundaries based on the desired boundary condition; note however that the free boundary condition was not quite correct in this original paper, and a more robust version of this enforcement is described in [Liu et al. 2013]. Fig. 9 demonstrates the effect of $\beta = \pi/2$ (tangential), $\beta = 0$ (normal), and free boundary conditions on two prototypical fields specified with a curve constraint. Shown are the integral curves of the underlying vector fields. Fig. 10 also shows the effect of different boundary conditions, this time on the (damaged) neck of the Planck dataset. In the rightmost image the desired angle for each boundary edge is achieved by setting the values β_e on a per edge basis to match a global down direction.

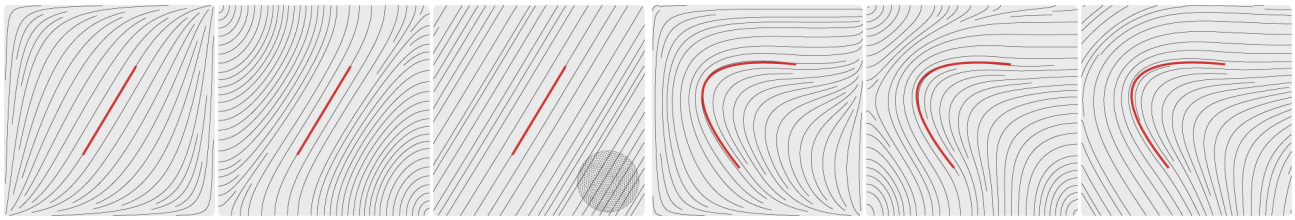


Figure 9: *Comparison of tangential, normal, and free boundary conditions with two different strokes as constraints, rendered with the method of [Mebarki et al. 2005] (inset shows mesh resolution).*

4.5 Implementation

While the setup of this approach to vector field design is rather straightforward, various implementation and numerical details are important to guarantee efficiency. Again, we point the reader to [Fisher et al. 2007] where an efficient (low rank) incremental update of the Cholesky factorization of M is proposed for each change of the Z constraints to offer realtime design: once a factorization of the original matrix M (without constraints) into $M = CC^T$ is found as a preprocess, a user can input any constraint and the update of the vector field is done via backward/forward substitution and a simple update of C. A gallery of results on models of various sizes with diverse user-input constraints can be seen in Fig. 11.

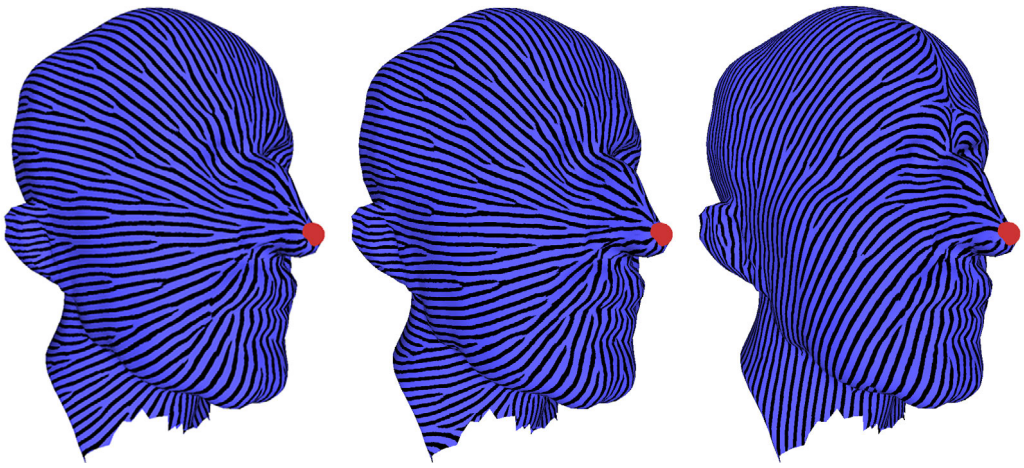


Figure 10: *The Planck dataset with a single source and varying boundary conditions: free (left), normal (middle), and fixed angle (right). In the latter case the desired angle was set on a per edge basis ensuring that the field flows off the boundary in the global down direction.*

5 Recent Extensions

While the edge-based encoding of vector fields have found a number of applications throughout the last decade, we end this chapter by reviewing a few notable extensions that enrich the applicability or accuracy of this representation.

5.1 Higher-order Whitney bases

We discussed the lack of continuity of the Whitney 1-form bases in Sec. 2.2, along with its consequences. While these are usually minor in most graphics applications, having higher-order 1-form bases would definitely remove all discontinuities, and thus allow for accurate approximations of higher order derivatives. One simple construction of higher-order Whitney bases for k-forms was proposed through subdivision

schemes [Wang et al. 2006], where 0-form bases are found using the traditional Loop scheme, and 2-form bases use half-box splines. In particular, it was shown how to derive the corresponding 1-form bases such that the "formule de commutation" is exactly enforced—ensuring that the reconstructed k-form of the discrete exterior derivative d of a $(k-1)$-form matches the continuous exterior derivative $\mathbf{d}$ of a reconstructed $(k-1)$-form. This property, valid for Whitney's finite elements, is important to guarantee the consistency of the discrete calculus induced by the operator d and consequently a valid Helmholtz decomposition. The recent work of de Goes et al [2016] also extends the approach of vector design from Whitney bases to subdivision-based higher-order Whitney bases by introducing a Subdivision Exterior Calculus. Note that the limit case of non-local bases was also proposed to ensure a spectral convergence of the discrete Hodge star [Rufat et al. 2014]—but is limited to logically rectangular grids so that fast transforms can be efficiently computed.

Figure 11: *Gallery of surface texture synthesis results based on vector fields specified with a variety of constraints, demonstrating that even just a few constraints can quickly build overall fields with pleasing flows.*

5.2 Height and tilt

Edge-based discretization can also be useful for other geometric quantities. For instance, the authors of [Andersen et al. 2009] used edge values to encode a "normal tilt" field over a surface: this is a vector field that defines a tilt (rotation) of the displacement direction with respect to the base normal direction. A tilt is thus a vector in the tangent space of the base shape, and its direction is the rotation axis for a rotation that transforms the displacement direction into the normal direction, and the magnitude of the tilt is the sine of the rotation angle. Along with a height field (with values sampled at vertices then linearly interpolated across triangles), using a tilt instead of a simple tangential displacement offers an intuitive description of a surface texture: the height truly represents the magnitude of the displacement, while the tilt indicates the local rotation of the normal field. This particular decomposition allows for very simple editing of geometric textures, see Fig. 12.

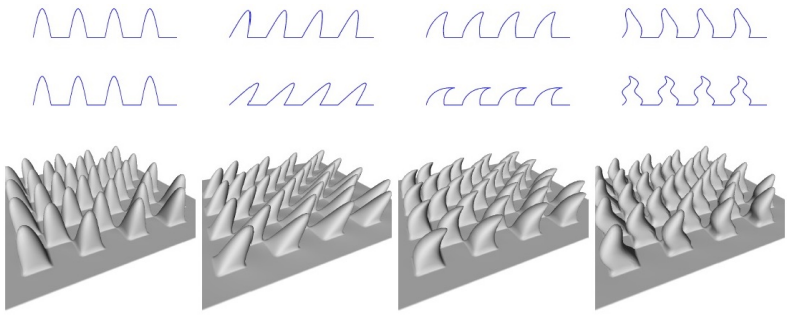


Figure 12: *Tilt textures use a discrete* 1*-form to geometrically define the texture in a thin shell around a surface, to be able to manipulate the 3D texture quite intuitively.*

5.3 Link to discrete diffeomorphisms

The edge-based discretization of tangent vector fields is also linked to a recent proposal to use a *functional* encoding of vector fields. The term "functional" is meant to refer to the effect of a vector field on a scalar function: if one think of a vector field $\mathbf{v}$ as a wind, a scalar function f will be advected (i.e., pushed infinitesimally by the wind) through $\langle \mathbf{v}, \nabla f \rangle$. This advection operator is actually called the Lie derivative $\mathcal{L}_{\mathbf{v}} f$ of f in $\mathbf{v}$, and is *linear* in $\mathbf{v}$ in the sense that $\mathcal{L}_{\lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2} f = \lambda_1 \mathcal{L}_{\mathbf{v}_1} f + \lambda_2 \mathcal{L}_{\mathbf{v}_2} f$. Therefore, one can *define* a vector field directly by how it acts on functions, i.e., as the operator $\langle \mathbf{v}, \nabla(\cdot) \rangle$. Moreover, since a vector field represents an infinitesimal flow, this Lie derivative operator can be seen as an infinitesimal action of a Lie algebra element of the diffeomorphism group on the space of scalar functions (in the study of dynamical systems, this idea was brought forth nearly a century ago by Koopman [1931]). In the discrete setting, Pavlov et al. [2011] showed that if one defines a discrete notion of diffeormorphism that transfers scalar values between dual cells, then the resulting Lie derivative turns out to be (the Hodge star of) the discrete 1-form we used above, since it represents fluxes across dual cells. Since then, this functional point of view was shown to be very relevant to, e.g., the problem of finding correspondences between meshes [Azencot et al. 2013; Azencot et al. 2015]. We refer the reader to [Gawlik et al. 2011] for a complete treatment of the dynamics of fluids using discrete diffeomorphisms.

6 Limitations

The edge-based representation of vector fields is quite powerful: it is actually simpler to use than the piecewise-constant alternative since no explicit frames need to be defined, and the operators associated to this representation are nearly as simple as the face-based case. However, due to its increased continuity and the consistent use of Whitney bases, derivatives are generally more reliable on arbitrary meshes. The possibility to extend this representation to the case of Riemannian metrics is also a plus, although we have to pay a price in the complexity of the resulting operators.

To close this chapter, we recap the list of the most important pros and cons of the edge-based representation of vector fields to consider when one is faced with a decision to pick the simplest vector field discretization that fits his or her needs.

Pros

- Coordinate-free representation using only one scalar value per edge.
- Simple interpolation of edge values.
- Simple differential operators leveraging the DEC literature.

Cons

- Discontinuous reconstruction for low-order Whitney basis functions.
- No clear vector at vertices, so incompatible with vertex-based deformation of meshes.
- Generalization to n-vector fields has not been studied.

Part III - Vertex-based Representation of Vector Fields

The Basics

While the most useful vector fields on surfaces for graphics applications are usually quite smooth, neither face-based or edge-based vector fields are even continuous unless interpolated with nonlinear bases (such as higher-order Whitney forms). In particular, these representations produce conflicting values at mesh vertices, rendering them inappropriate to describe the motion of vertices during, e.g., a deformation of the surface. A seemingly easy work-around consists in picking a normal per vertex to define vertex-based tangent planes over the surface, in which 2D tangent vectors (still one per vertex) can be set and interpolated over the whole mesh via simple piecewise linear basis functions over triangles. The choice in normals is far from canonical, however, and it clashes with face normals, making the whole construction just as inadequate as the face-based and edge-based representations, specially if one wishes to define a continuous tangent vector field. Fortunately, one can construct, in a fully intrinsic way, vertex-based vector fields on surfaces by designing specific (nonlinear) vector-valued basis functions. In this chapter, we review their basic constructions, and derive their associated discrete differential operators. Due to the continuity of the construction and its preservation of the underlying intrinsic parallel transport and metric structure, we will be able to provide definitions of nonlinear operators such as the covariant derivative that will have guaranteed convergence under refinement and good accuracy even on coarse meshes. Our geometric treatment will also be shown to extend to vertex-based representations of n-vector and n-direction fields just as easily.

1 Local Representation for Vertex-based Vector Fields

In classical differential geometry, vector fields are represented by their components in local charts. It thus seems logical to extend this representation to meshes, where the vector fields would be sampled at vertices. To ensure locality, the interpolation function from a vertex to the rest of the surface should have a small support—ideally, the same support as the standard piecewise-linear hat function ϕ_i for each vertex v_i, i.e., the one-ring neighborhood around the vertex. This construction turns out to be, in fact, simple to achieve: one first needs to parameterize these one-ring charts so that a local representation of vector fields in these charts can be established, as we now review.

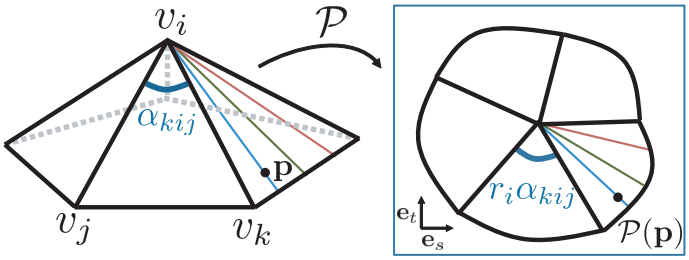
1.1 Geodesic polar map

One common method to get a parameterization of one-rings is through the geodesic polar map, mimicking the typical 2D geographical maps of Arctic or Antarctic regions where the meridians are straight lines leaving the pole. The geodesic polar map of a vertex v_i flattens its one-ring into a planar one-ring by enforcing that the geodesics from v_i (i.e., line segments on adjacent triangles starting at v_i) remain straight in the plane, while rescaling the angles between these geodesics by

$$r_i = 2\pi / \sum_{t_{ijk}} \alpha_{kij} \equiv 2\pi/(2\pi - \kappa_i), \tag{21}$$

where α_{kij} is the angle at vertex i within triangle t_{ijk}, and κ_i is the commonly used discrete Gaussian curvature integral for v_i. In other words, we find a particular map between an intrinsically non-flat one-ring of the surface into a flat domain by "stretching" the angles around $\mathbf{p}_i$ uniformly [Zhang et al. 2006].

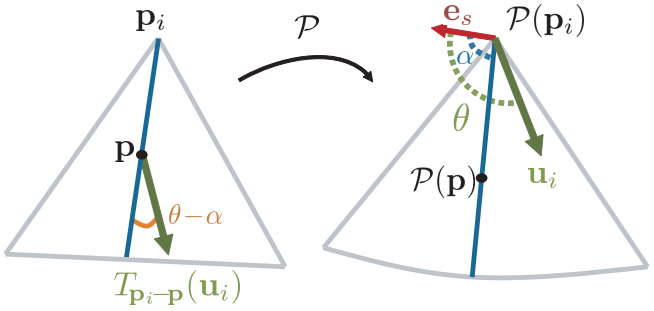
Note that rim edges of the one-ring of vertex v_i will not remain as straight lines in the parameterization of v_i, nor would the edges emanating from v_i remain straight in the parameterization of the neighboring vertex v_j. But no worries, we are not done yet: we have just constructed a set of charts so that we can define a proper interpolation between vertex vectors. In these local charts, we can now choose a standard Cartesian coordinate system (s, t) with the direction of s-axis aligned to one of the edges incident to v_i. As shown in the inset, we denote the geodesic polar map as $\mathcal{P}$ which maps a point $\mathbf{p}$ on the mesh to a pair of coordinates (s, t).



1.2 Interpolation bases through parallel transport

Given the parameterization of a one-ring based on the geodesic polar map $\mathcal{P}$, we have a notion of tangency at each vertex v_i that identifies tangent planes to the 2D space spanned by the local directions of the isocurves of s and t respectively. With this local parameterization, there is no need to assign a normal at a vertex; therefore, we say that the tangent space defined by the parametrization is *intrinsic*. In order to create an interpolation basis for a vector stored in the tangent space of v_i, one must establish the relationship between the tangent spaces in each adjacent triangles and the induced tangent space at v_i. Parallel transport of the vertex-based vector along geodesics starting at v_i is one way to derive such a relation, as initially proposed in [Zhang et al. 2006].

In the continuous setting, a vector is said to be parallel transported along a geodesic if it maintains the same angle with the tangent vectors of the geodesic curve as it is transported: this is a generalization of the notion of translation along a line in $\mathbb{R}^2$. This concept of parallel transport is relatively simple to discretize in the case of face-based representations [Crane et al. 2010]. The discretized version of parallel transport for vertex-based representation can be also defined now that we have a notion of geodesic paths in a one-ring: for a point $\mathbf{p} \in t_{ijk}$, assuming the image of line segment $\mathcal{P}(\overline{\mathbf{p}_i\mathbf{p}})$ makes an angle α with the s-axis for this one-ring, and the vector $\mathbf{u}_i$ of magnitude $|\mathbf{u}_i|$ stored at v_i (located at $\mathbf{p}_i$) makes an angle θ with the same s-axis, the parallel transported vector from $\mathbf{p}_i$ to $\mathbf{p}$ is the tangent vector $(|\mathbf{u}_i|/|\overline{\mathbf{p}_i\mathbf{p}}|)\overline{\mathbf{p}_i\mathbf{p}}$ rotated around the triangle normal by $\theta - \alpha$. Denoting the parallel transport from $\mathbf{p}_i$ to $\mathbf{p}$ in the one-ring by $T_{\mathbf{p}_i \to \mathbf{p}}$, and the coordinate basis at v_i as $\mathbf{e}_s$ and $\mathbf{e}_t$ respectively for the s-axis and the t-axis, we can formulate the interpolation for $\mathbf{u}_i = u_s\mathbf{e}_s + u_t\mathbf{e}_t$ using the piecewise linear hat functions ϕ_i to extend the vector through parallel transport as



$$\mathbf{u}(\mathbf{p}) = u_s T_{\mathbf{p}_i \to \mathbf{p}}(\mathbf{e}_s)\phi_i(\mathbf{p}) + u_t T_{\mathbf{p}_i \to \mathbf{p}}(\mathbf{e}_t)\phi_i(\mathbf{p}). \tag{22}$$

This means that we have created a coordinate frame at each point $\mathbf{p}$ in the one-ring through parallel transport from $\mathbf{p}_i$, thus defining a two-vector-valued basis function:

$$\Psi_i(\mathbf{p}) = (T_{\mathbf{p}_i \to \mathbf{p}}(\mathbf{e}_s)\phi_i(\mathbf{p}), T_{\mathbf{p}_i \to \mathbf{p}}(\mathbf{e}_t)\phi_i(\mathbf{p})). \tag{23}$$

Assuming that each $\mathbf{u}_i$ is stored as a column vector, we can thus represent a vector field over the entire mesh as

$$\mathbf{u}(\mathbf{p}) = \sum_{v_i \in V} \Psi_i(\mathbf{p})\mathbf{u}_i. \tag{24}$$

Note that due to the partition of unity that the hat functions create over the mesh, this (now finite-dimensional) representation of vector fields exactly interpolate the vectors given at each vertex.

1.3 Shortcomings

While the construction above seems natural and sufficient to define vector fields on triangle meshes, the use of a per-vertex geodesic polar map does *not* produce continuous vector fields, which lead to infinite covariant derivatives (see Sec. 3.2 in the Introduction chapter) of the vector fields. In fact, this practical issue was dealt with in [Zhang et al. 2006] by treating each triangle as a trapezoid created by removing a small triangle near the vertex, which still leads to inconsistent derivatives values compared to the continuous limit. More recently, a piecewise-constant redistribution of the discrete Gaussian curvature (angle deficit) at vertices was proposed in [Knöppel et al. 2013] to alleviate the issue. However, their approach was only able to provide the L_2 energy of the first-order derivatives, and no closed-form expression of a smooth vector field was offered. Very recently, a solution to this issue was brought forth by [Liu et al. 2016] while sticking to piecewise-linear blending function, i.e., without having recourse to high-order basis functions to define smoother transition between the one-ring charts. Their method requires the proper definition of a simplicial connection on the mesh, which we review next.

2 Vertex-based vector fields through simplicial connections

We now revisit the local interpolation we introduced above by adding an additional twist, namely, a simplicial connection that allows all these local one-ring parameterizations to form a valid atlas for the surface—and thus, a continuous notion of vector fields over the surface. In the following exposition, we loosely follow the notation from [Liu et al. 2016], but provide a more didactic explanation of the construction.

2.1 Piecewise linear charts

As mentioned early on, smooth vector fields need to be represented through coordinates in various parameterization charts: while it would be convenient to use a single, global parameterization chart, general manifolds (other than those with trivial topology) cannot be described by a single chart. Instead, a collection of charts (called an atlas) is typically used to describe the entire surface. For computational efficiency, piecewise linear parameterization charts are often used: it can be seen as creating parameterizations that are linear functions within each triangle, and then stitched together on shared edges to form larger patches. On each patch, we can find the parameter values through piecewise-linear chart functions $(s(\mathbf{p}), t(\mathbf{p})) = \sum (s_i, t_i)\phi_i(\mathbf{p})$, the inverse of which is the parameterization $\mathbf{p}(s, t)$. In addition to denoting each point on the surface by a pair of parameters, such a parameterization induces local basis frames $(\partial/\partial s, \partial/\partial t)$ for the tangent space at each point (as described in Sec. 1.2), which can be seen as the vectors corresponding to moving along s-isocurves and t-isocurves at unit speed, or simply as $(\partial\mathbf{p}/\partial s, \partial\mathbf{p}/\partial t)$. Note that this per-face linear treatment corresponds to basis frames that are piecewise-constant; alas, the frame is not well-defined at vertices. A straightforward way of dealing with this lack of continuity at vertices is to introduce a smoother parameterization, using higher order basis functions. However, even with well-defined pointwise tangent bases, defining a notion of derivatives of vector fields is not as simple as taking partial derivatives of its components, as the resulting 2×2 Jacobian matrix would depend on the chart used!

The Introduction chapter introduced a geometrically-motivated derivative called the *covariant derivative*, still forming a 2×2-matrix but now in a chart-independent way. It involves an entity called a *connection*, whose role is to align the frame basis at one point to the frame at a nearby point along a given direction. If we use *orthonormal* basis frames, the connection corresponds to a rotation rate per direction, which, within a local chart, can be represented as a 1-form (see Sec. 3.3 in the Introduction chapter). In the remainder of this section, we describe a unified description based on a piecewise-linear connection 1-form and a way to compare the tangent spaces within each triangle with a 2D tangent space associated with each *vertex*. This description is compatible with the commonly used treatment of vector field derivatives, but extends it to allow evaluation of the derivatives everywhere on the surface without

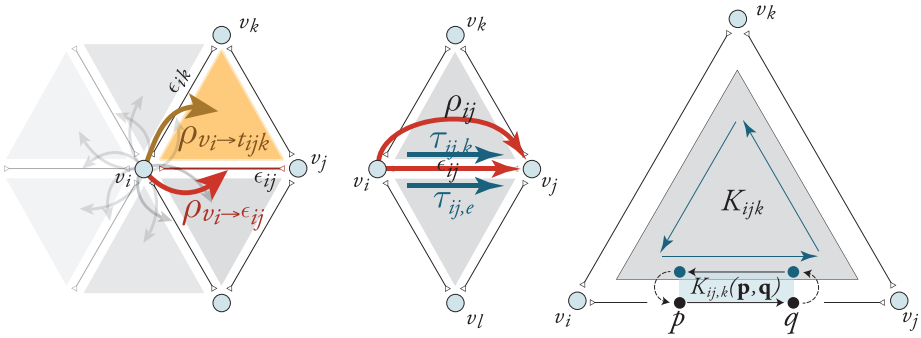


Figure 13: ***Discrete simplicial connection.*** *(left) Each vertex v_i is given a transition rotation angle $\rho_{v_i \to e_{ij}}$ to edge e_{ij} and $\rho_{v_i \to t_{ijk}}$ to triangle t_{ijk}. (middle) A continuous connection within simplices is encoded through edge rotation ϵ_{ij} and half-edge rotation $\tau_{ij,k}$ interpolated over edges and faces respectively via Whitney basis functions. (right) Connection curvature can be evaluated with circulation of local 1-form representation of the connection.*

resorting to parameterizations with high-order smoothness.

2.2 Putting the puzzle back together

As a triangle mesh is the union of vertices, edges, and triangles, a general treatment of tangent vector fields requires the proper definition of tangent planes at all points in these simplices. For simplicity, one can choose an orthonormal frame at vertex v by first picking a tangent plane through, e.g., the area weighted average normal of the one-ring, and projecting one of the incident edges to the tangent plane: the unit vector along that direction $\mathbf{e}_v$ and its 90° rotation $\mathbf{e}_v^\perp$ in the tangent plane then define a vertex frame. Likewise, we can pick the unit vector $\mathbf{e}_e$ along the edge e along with $\mathbf{e}_e^\perp$ equal to the cross product of the average normal of the two incident triangles with $\mathbf{e}_e$ to define an edge frame. For each triangle t, we can pick one arbitrary unit tangent vector $\mathbf{e}_t$ along with $\mathbf{e}_t^\perp$, its 90° rotation within the triangle. Note that these frames are rather arbitrary (in the sense that they locally do not align), but they properly define a frame for each of the tangent spaces of the triangle mesh in which to express tangent vectors. We will, from now on, adopt a complex number representation for vectors in any of these tangent spaces: for a frame $(\mathbf{e}, \mathbf{e}^\perp)$, the vector $\mathbf{u} = u_1\mathbf{e} + u_2\mathbf{e}^\perp$ will be denoted as $\mathbf{u} = (u_1 + \mathrm{i}\, u_2)\mathbf{e}$. Thus, we will encode a rotation in the tangent plane by an angle of θ through the multiplication by $e^{\mathrm{i}\theta}$.

With these frames given, the connection 1-form ω in a triangle t_{ijk} can be defined in order to align frame bases between nearby tangent spaces, thus inducing the parallel transport of an arbitrary tangent vector: for a segment c between two points $\mathbf{p}_1$ and $\mathbf{p}_2$ within this triangle, one needs to perform a rotation by an angle $\int_c \omega$ (i.e., multiplication by the complex number $e^{\mathrm{i}\int_c \omega}$) to transport a vector from $\mathbf{p}_1$ to $\mathbf{p}_2$ since ω measures the rotation angle between infinitesimally nearby tangent frames. A natural discretization of $\omega_{t_{ijk}}$ within a triangle is a discrete Whitney 1-form introduced in the previous chapter, with edge values:

$$\omega_{t_{ijk}} = \tau_{ij,k}\varphi_{ij} + \tau_{jk,i}\varphi_{jk} + \tau_{ki,j}\varphi_{ki},$$

Similarly, within an edge, the connection $\omega_{e_{ij}}$ can be defined as the discrete Whitney 1-form using a single edge value:

$$\omega_{e_{ij}} = \epsilon_{ij}\varphi_{ij} = \epsilon_{ij}[\varphi_i d\varphi_j - \varphi_j d\varphi_i] \overset{(\varphi_i+\varphi_j=1)}{=} \epsilon_{ij} d\varphi_j.$$

For a path connecting two points not on the same simplex, we need to also deal with the frame (mis)alignment across edges or even through vertices. To handle such cases, we can define "transition" rotations that compensate the difference in frame choice that we made at different incident simplex pairs (σ_1, σ_2) at the intersection $\sigma_1 \cap \sigma_2$ of the two simplices. We will denote these finite rotations as $\rho_{\sigma_1 \to \sigma_2}$.

Note that the transition rotation between a vertex and an edge (or equally, a vertex and a face) is a single angle, while the transition angle between an edge and a face depends on the location along the edge, as shown in Fig. 13. Moreover, these alignment angles must satisfy the following properties:

$$\rho_{\sigma_1\to\sigma_2}(\mathbf{p}) = -\rho_{\sigma_2\to\sigma_1}(\mathbf{p}) \quad \forall \mathbf{p} \in \sigma_1 \cap \sigma_2;$$

$$\rho_{v_i\to e_{ij}} + \rho_{e_{ij}\to t_{ijk}}(\mathbf{p}_i) = \rho_{v_i\to t_{ijk}};$$

$$\rho_{v_i\to e_{ij}} + \epsilon_{ij} + \rho_{e_{ij}\to v_j} = \rho_{v_i\to t_{ijk}} + \tau_{ij,k} + \rho_{t_{ijk}\to v_j};$$

$$\rho_{v_i\to e_{ji}} = \pi + \rho_{v_i\to e_{ij}} + 2\pi n_{ij},$$

where n_{ij} is an integer per edge determined by the choice of simplicial frames—a detailed proof can be found in [Liu et al. 2016]. Given these conditions, it turns out that the whole set of alignment angles among simplices, the edge connection 1-forms, and the face connection 1-forms can be completely determined by just the vertex-to-triangle transition rotations $\rho_{v_i\to t_{ijk}}$, vertex-to-edge transition rotations $\rho_{v_i\to e_{ij}}$, and $|E|$ vertex-to-vertex rotations ρ_{ij}. Indeed, once these independent parameters are chosen, one can deduce all the other transition angles and 1-forms via:

$$\begin{aligned}\epsilon_{ij} &= -\rho_{v_i\to e_{ij}} + \rho_{ij} + \rho_{v_j\to e_{ij}},\\ \tau_{ij,k} &= -\rho_{v_i\to t_{ijk}} + \rho_{ij} + \rho_{v_j\to t_{ijk}}.\end{aligned} \tag{25}$$

2.3 Interpolation basis

Now that we have defined above a path-dependent alignment between frame bases at different points, we can construct interpolation basis that are geometrically continuous. First, we transport the basis $\mathbf{e}_v$ at vertex v to its one-ring by

$$\Phi_i\Big|_t(\mathbf{p}) = \mathbf{e}_t \exp\left[-\mathfrak{i}\,(\rho_{v_i\to t} + \int_{\mathbf{p}_i\to\mathbf{p}} \omega_t)\right],$$

which accounts both for the transition angle and the path within the triangle t to $\mathbf{p}$. Then, we create an interpolation (partition of unity) by attenuating this field linearly to zero at the border of the one-ring using the hat function φ_i:

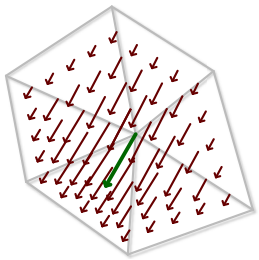
$$\Psi_i\Big|_{t_{ijk}}(\mathbf{p}) = \varphi_i(\mathbf{p})\Phi_i\Big|_{t_{ijk}}(\mathbf{p}).$$

Now, we can evaluate the vector $\mathbf{u}(\mathbf{p})$ parallel transported from the vector $\mathbf{u}_i = (u_i^1, u_i^2)$ given at vertex v_i through:

$$\mathbf{u}(\mathbf{p}) = \sum_i \Psi_i(\mathbf{p})(u_i^1 + \mathfrak{i}\,u_i^2).$$

By construction, this vector field is continuous everywhere, zero beyond the one-ring of v_i, and interpolate $\mathbf{u}_i$ at v_i. Consequently, a continuous vector field can be generated by defining a vector $\mathbf{u}_i$ per vertex v_i and summing all the corresponding local vector fields $\Psi_i(\mathbf{p})(u_i^1 + \mathfrak{i}\,u_i^2)$.

Continuity. Note that while the constructed finite-dimensional vector fields are continuous, the *components* (u^1, u^2) of such interpolated vector fields are not necessarily continuous across incident simplices. However, the transition angles exactly provide the necessary rotation that produce the change of coordinates to get these components continuous after alignment to a common frame (see inset), while continuous component values may lead to discontinuity in a given choice on local frames. We elaborate on how derivatives are evaluated in the next section.



3 Operators on vertex-based vector fields

Analyzing and designing vector fields involves not just a representation for the vector fields, but it also requires evaluations of the local changes (derivatives) of these vector fields. Several types of geometrically-defined (meaning, invariant under change of coordinates) were provided in previous chapters. However, the full-blown 2×2-matrix representing the covariant derivative of vector fields is more conveniently evaluated on vertex-based vector fields. In the following, we first show how the covariant derivative is evaluated, then we examine its relation to differential operators such as divergence and curl, and finally we formulate L_2 energies associated with these first-order operators to complete our computational toolkit for vector fields.

3.1 First-derivative operators

Recall that we denote the local basis of a tangent space by $\mathbf{e}$ and $\mathbf{e}^\perp$ ($=\mathbf{i}\,\mathbf{e}$). Still using the complex number representation, we can compute the covariant derivative ∇ of a vector field $\mathbf{u}=u\,\mathbf{e}=(u^1+\mathbf{i}\,u^2)\,\mathbf{e}$ along a direction $\mathbf{w}$ (living also in the same tangent space) by combining the differential of components, du, and the angular velocity ω for aligning the bases through

$$\nabla_{\mathbf{w}}\mathbf{u} = du(\mathbf{w})\;\mathbf{e} + u\nabla_{\mathbf{w}}\mathbf{e} = du(\mathbf{w})\;\mathbf{e} + u\;\mathbf{i}\omega(\mathbf{w})\;\mathbf{e}.$$

Note that the derivative is linear with respect to $\mathbf{w}$ and obeys Leibnitz product rule on $\mathbf{u}$ as required, and it can be expressed as a 2×2 matrix acting on $\mathbf{w}$:

$$\nabla_{\mathbf{w}}\mathbf{u} = \nabla\mathbf{u}(\mathbf{w}) = (\mathbf{e}\otimes du + u\,\mathbf{i}\,\mathbf{e}\otimes\omega)\,(\mathbf{w}).$$

The four components of this matrix defining the covariant derivative are thus

$$\begin{bmatrix} \nabla_{\mathbf{e}}(u^1\mathbf{e}) & \nabla_{\mathbf{e}^\perp}(u^1\mathbf{e}) \\ \nabla_{\mathbf{e}}(u^2\mathbf{e}^\perp), & \nabla_{\mathbf{e}^\perp}(u^2\mathbf{e}^\perp) \end{bmatrix}.$$

Limitations of the geodesic polar map approach. In the geodesic polar map induced alignment using parallel transport along geodesics [Zhang et al. 2006], the derivatives are evaluated through the Jacobian matrix of the vector field's components, which can be constructed pointwise inside each triangle through the basis $\Psi(\mathbf{p})$ except near vertices with non-zero Gaussian curvature. To handle such vertices easily, one may subdivide the mesh topologically once by inserting midpoint of each edge and splitting each triangle into four, thus each triangle contains only one non-zero Gaussian curvature vertex. The Jacobian evaluation at $\mathbf{p}$ is then carried out in the geodesic polar map of that vertex by evaluating a finite difference approximation of the interpolated vector field at $\mathbf{p}$ and two nearby points, e.g., two points along $\mathbf{e}$ and $\mathbf{e}^\perp$ direction. Note that this is actually "more" than just an approximation of the Jacobian of the components (u^1, u^2), since the interpolation indirectly introduces an alignment described by the non-linear connection 1-form ω. Yet the resulting covariant derivative can be extremely large near vertices with angle deficit; therefore, this approximate covariant derivative is limited in its usefulness, and can only be used for quadrature rules at locations away from non-flat vertices to estimate divergence or curl.

Discrete connection approach. Through the connection 1-form defined within triangles and edges and the alignment angles between incident simplices, the covariant derivative for $\Psi_i(\mathbf{p})$ is much more straightforward to calculate, and is *finite everywhere* on the mesh. In particular, the expression within triangle t_{ijk} can be expressed analytically as

$$\nabla\Psi_i = \nabla(\phi\Phi) = \Phi\otimes d\phi_i - K_{ijk}\mathbf{i}\,\Psi_i\otimes\phi_{jk},$$

where K_{ijk} is the (integral of the) connection curvature (i.e., $=-d\tau_{t_{ijk}}$) over the triangle.

Given the covariant derivative, one may reassemble its four components into geometrically intuitive operators (mimicking the continuous case described in Sec. 3.4 in the Introduction chapter) as

$$\begin{aligned}
\text{div } \Psi_i &= \mathbf{e} \cdot \nabla_{\mathbf{e}} \Psi_i + \mathbf{e}^{\perp} \cdot \nabla_{\mathbf{e}^{\perp}} \Psi_i, \\
\text{curl } \Psi_i &= \mathbf{e} \cdot \nabla_{\mathbf{e}^{\perp}} \Psi_i - \mathbf{e}^{\perp} \cdot \nabla_{\mathbf{e}} \Psi_i, \\
\overline{\text{div}} \ \Psi_i &= \mathbf{e} \cdot \nabla_{\mathbf{e}} \Psi_i - \mathbf{e}^{\perp} \cdot \nabla_{\mathbf{e}^{\perp}} \Psi_i, \\
\overline{\text{curl}} \ \Psi_i &= \mathbf{e} \cdot \nabla_{\mathbf{e}^{\perp}} \Psi_i + \mathbf{e}^{\perp} \cdot \nabla_{\mathbf{e}} \Psi_i,
\end{aligned}$$

which represent respectively the divergence, curl, reflected divergence, and reflected curl operators. Among these four operators useful for vector field analysis and design, the divergence and the curl of a vector field are well defined scalar fields that do not depend on the choice of the frame basis $\mathbf{e}$. On the other hand, the reflected divergence (resp., curl), which is the divergence (resp., curl) of the vector field reflected with respect to the frame basis $\mathbf{e}$, is apparently dependent on the choice of bases, thus not a geometric quantity. However, these two terms can be assembled into the Cauchy-Riemann operator $\bar{\partial} = 1/2(\overline{\text{div}} \ , \overline{\text{curl}} \ )$, which, when applied to $\mathbf{u}$, turns out to produce a well-defined geometric quantity: a 2-vector field!

3.2 Discrete vector Laplacians

In the edge-based formulation, we have already seen that the Hodge Laplacian operator $\Delta = \star d \star d + d \star d \star$ of the 1-form version $\mathbf{u}^\flat$ of the vector field $\mathbf{u}$ can be evaluated through gradient, divergence, curl and Hodge star. In other words, it can be defined through $\langle \Delta \mathbf{u}^\flat, \mathbf{v}^\flat \rangle = \langle d\mathbf{u}^\flat, d\mathbf{v}^\flat \rangle + \langle \delta \mathbf{u}^\flat, \delta \mathbf{v}^\flat \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the inner product between forms. This Laplacian leads to an energy called the anti-holomorphic energy of the vector field, which is actually slightly different from the Dirichlet energy defined by the integral of the squared norm of the covariant derivative $|\nabla \mathbf{u}|^2$, where the norm is now defined by the Frobenius norm of the matrix representation in an orthonormal frame basis. Such a Dirichlet energy induces another notion of Laplacian called the *Bochner Laplacian* for curved Riemannian manifolds.

With the interpolation basis Ψ_i, we can directly evaluate the integral of the squared norm of the covariant derivative since we have an analytical expression for it within each triangle, and the integral over edges and vertices is nil since the covariant derivative is finite on these 0-measure sets. One can then derive that the Bochner Laplacian Δ_B is defined through

$$\begin{aligned}
\langle \Delta_B \mathbf{u}, \mathbf{v} \rangle &= \frac{1}{2} (\langle \text{div } \mathbf{u}, \text{div } \mathbf{v} \rangle + \langle \text{curl } \mathbf{u}, \text{curl } \mathbf{v} \rangle + \langle \overline{\text{div}} \ \mathbf{u}, \overline{\text{div}} \ \mathbf{v} \rangle + \langle \overline{\text{curl}} \ \mathbf{u}, \overline{\text{curl}} \ \mathbf{v} \rangle \qquad (26) \\
&= \langle \nabla_{\mathbf{e}} \mathbf{u}, \nabla_{\mathbf{e}} \mathbf{v} \rangle + \langle \nabla_{\mathbf{e}^{\perp}} \mathbf{u}, \nabla_{\mathbf{e}^{\perp}} \mathbf{v} \rangle. \qquad (27)
\end{aligned}$$

3.3 Implementation.

Given the expression of Ψ_i and $\nabla \Psi_i$ we provided above, explicit symbolic integration of the components of the covariant derivative and associated derivatives can be performed directly. However, in some cases, their computation can be further simplified. For instance, the evaluation of the Dirichlet energy can be done without specifying the transition angle $\rho_{v \to t}$, as pointed out by [Knöppel et al. 2013], since only $\int_\Omega \langle \Psi_i, \Psi_j \rangle$ and $\int_\Omega \langle \nabla \Psi_i, \nabla \Psi_j \rangle$ need to be evaluated, and they only depend on the curvature induced by the transition rotation between adjacent vertices ρ_{ij}.

One can also evaluate the divergence and curl operators within the triangle through the line integral along edges only by invoking Stokes' theorem. As an example, explicit formulas can be found in [Liu et al. 2016] for the evaluation of the discrete operators based on the integral of the pointwise analytic expression for components of Ψ_i along edge e_{ij}. Similarly the Cauchy-Riemann operator, representing the reflected divergence and reflected curl, can be evaluated on the triangle through:

$$\bar{\partial}_t \Psi_i = \frac{1}{2} \int_{\partial t} ((F\Psi_i) \times d\mathbf{l}, (F\Psi_i) \cdot d\mathbf{l})^T,$$

where F is the reflection defined through $F((u^1 + \mathrm{i}\, u^2)\,\mathbf{e}) = (u^1 - \mathrm{i}\, u^2)\,\mathbf{e}$.

4 Vertex-based n-vector and n-direction fields

As mentioned in the face-based formulation, n-vector and n-direction (or unit n-vector) fields are indispensable in various graphics applications: for instance, 2-vector fields help design stripe patterns [Knöppel et al. 2015], 4-vector fields are useful to compute curvature fields and design quadrangulation [Kalberer et al. 2007], and 6-vector fields can drive triangular remeshing [Nieser et al. 2012]. Such fields can be identified by a representative vector so that any one of the n vectors at a point are just rotationally symmetric replicates. Thus, the design of the n-vector field reduces to the design of a vector field, with the caveat that the notion of smoothness of the field must now account for the n-rotational symmetry nature.

Designing a n-vector field is typically guided by a choice of singularities and directions/magnitudes at user-specified locations. Finding a smooth field that interpolate the user constraints without adding new singularities can resemble a game of whack-a-mole as singularities (i.e., where the representative vector field becomes nil) can pop up all over the place with the previous representations. Since we have a clear definition of connection in this vertex-based formulation, we have the tools to specify precisely where positive *and* negative indexed singularities (sinks/sources and saddle points) should appear.

4.1 Control of singularities

First, observe that n-vector fields can be thought of having only singularities of index $\pm 1/n$, since other singularity types are combinations of these basic singularities. Second, one can show that index-$1/n$ singularities can be created by a nonzero holomorphic derivative $\partial = (\mathrm{div}\ , \mathrm{curl}\ )$, while singularities of index $-1/n$ can be created by a nonzero Cauchy-Riemann operator $\bar{\partial}$. Third, if all the local frames in the neighborhood rotate by $-\alpha$, the representative vector field $\mathbf{v}$ are rotated by $\mathbf{v}' = e^{\mathrm{i}n\alpha}$, and then the operators involved in the covariant derivative become:

$$\nabla \mathbf{v}' = e^{\mathrm{i}n\alpha}(\nabla \mathbf{v})e^{-\mathrm{i}\alpha} = \tfrac{1}{2}e^{\mathrm{i}n\alpha}(\partial \mathbf{v} + F\bar{\partial}\mathbf{v})e^{-\mathrm{i}\alpha} = \tfrac{1}{2}e^{\mathrm{i}(n-1)\alpha}\partial \mathbf{v} + \tfrac{1}{2}e^{\mathrm{i}(n+1)\alpha}F\bar{\partial}\mathbf{v}.$$

Therefore, we can conclude that $\partial \mathbf{v}$ is an $(n-1)$-vector field, and $\bar{\partial}\mathbf{v}$ is an $(n+1)$-vector field [Liu et al. 2016]. From these facts, one realizes that, in order to control the location of the singularity, a local non-zero assignment for ∂ or $\bar{\partial}$ suffices. On the other hand, to control the direction of the singularity (i.e., rotating it by angle θ), one just need to multiply the components by $e^{\mathrm{i}(n-1)\theta}$ and $e^{\mathrm{i}(n+1)\theta}$ respectively, as shown in Fig. 14.

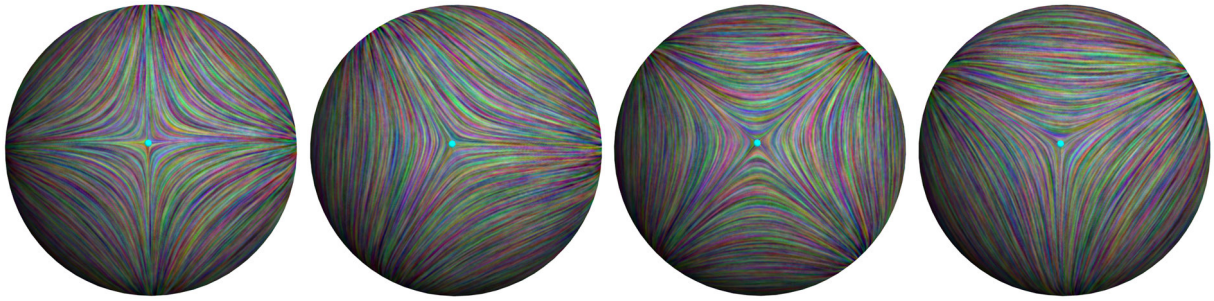
4.2 Control of directions

Controlling the direction of a n-vector field is even simpler: we can simply penalize the mismatch between the representative vector field and a user-specified direction (typically drawn as a stroke) prescribed by an arclength-parametrized curve c. This can be achieved by minimizing

$$\int_c |\nabla_{\dot{c}}(\mathbf{u} - (\mathbf{u}\cdot\dot{c})\dot{c})|^2 ds,$$

i.e., the L_2 squared norm of the covariant derivative of the *difference* between $\mathbf{u}$ and the user-specified direction, while maximizing the alignment of the vector field $\mathbf{u}$ with the user-specified direction

$$\int_c |\mathbf{u}\cdot\dot{c}|^2 ds.$$

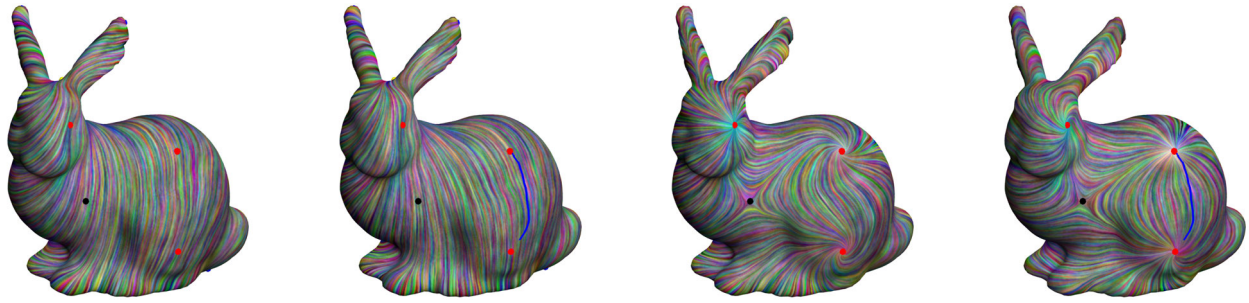


(a) (b) (c) (d)

Figure 14: ***Orientation control for negative index singularities.*** *From a vector field (a) on a sphere with a saddle point with index -1 and its corresponding 2-RoSy field (b) forming a trisector of index $-1/2$), the user can directly control the orientation (c) of the saddle and the respective orientation of the trisector (d) without affecting its position on the surface.*

4.3 Eigen-based design

Since control over singularities and directions is rather simple, we can in fact formulate the problem of vector field design as the following optimization: find the smoothest n-vector field satisfying the user constraints. When there is no user constraints, the problem reduces to a generalized eigenvalue problem $\mathbf{Au} = \lambda\mathbf{Bu}$ as initially pointed in [Knöppel et al. 2013], where $\mathbf{A}$ is the Bochner Laplacian, and $\mathbf{B}$ is the mass matrix corresponding to the quadratic form associated with the L_2-norm of the vector field; the smallest λ corresponds to the smoothest field for a given overall norm. But we can now enrich this approach thanks to our control over both singularities and directions: singularity constraints can be added by rewarding the projection of $\partial\mathbf{u}$ and $\bar{\partial}\mathbf{u}$ on the prescribed orientation as $n \pm 1$-fields, and removing the smoothness penalty of the corresponding part in the covariant derivative norm. Similarly, the direction constraints can be incorporated by adding the aforementioned penalty to $\mathbf{A}$ and adding the projection term to the mass matrix [Liu et al. 2016]. N-vector field design with intuitive user control is thus as simple as a generalized eigenvalue problem. Fig. 15 shows an example with both singularity and direction constraints resolved through an eigen solver.



(a) Smoothest vector field (b) a user-specified stroke (c) prescribed singularities (d) both

Figure 15: ***Eigen design.*** *While unconstrained generalized eigenvalue problems can be used to find the smoothest vector fields with or without stroke constraints (a & b), the user can also prescribe both positive and negative singularities (c) and use strokes to guide the vector field (d).*

5 Limitations

While we had recourse to advanced geometric notions, the vertex-based representation closely resembles the common finite element representation of vector fields for planar meshes, and allows for the evaluation of pointwise covariant derivative and associated differential operators. However, while the derivatives we formulated do not depend on the chosen frames, the representation itself relies on a(n arbitrary) choice of local frames. Moreover, the useful notion of Helmholtz-Hodge decomposition, crucial in solving various differential and integral equations, has not yet been proposed within this representation, and may be difficult to obtain: while the divergence operator can be defined to satisfy Stokes' theorem for integral over each triangle, its adjoint operator does not lead to a gradient operator that satisfies $\mathrm{curl} \circ \mathrm{grad} = 0$, which is crucial to form a space of harmonic vector fields with the correct dimensionality.

To close this chapter, we recap the list of the most important pros and cons of the vertex-based representation of vector fields to consider when one is faced with a decision to pick the simplest vector field discretization that fits his or her needs.

Pros

- Continuous interpolation of vertex-based tangent vectors.
- Pointwise and integral version of first- and second-order differential operators.
- Generalization to n-vector fields is trivial.

Cons

- Requires a choice of frame per simplex.
- Requires a simplicial connection.
- No Hodge decomposition of vector fields.

Conclusion

As summarized in previous chapters, each family of representations for tangent vector fields has its pros and cons. The lack of continuity of the face-based representation and of the normal component of the edge-based representation is outweighed by their simplicity of storage and implementation and the conciseness of their differential operators. On the other hand, the continuity provided by the vertex-based representation offers the flexibility of constructing a full-blown notion of covariant derivative that perfectly mimics its continuous equivalent, at the cost of having to store, preprocess, and access additional structures such as the transition angles between simplices and piecewise linear connection 1-forms. The situation is somewhat analogous to the choice of surface representation in CAGD: triangle meshes are simpler to work with, but higher-order surface representations provide improved smoothness and accuracy at the cost of an increased complexity in implementation.

Therefore, to make an informed decision on which representation to use or to develop, one should first identify the continuous structural properties relevant for the target application, and only then choose the discretization that best matches these requirements. For instance, in texture mapping, the parameter values are often represented by vertex-based scalar fields whose gradients are guided by user-specified vector fields. In such a scenario, it is crucial to rely on a discrete Helmholtz-Hodge decomposition, which both face-based or edge-based representations offer. One can also use different representations in different parts of an application, e.g., using vertex-based representation to design smooth vector fields satisfying user constraints before converting them into a face-based representation to compute a parameterization through a discrete Poisson equation. As a rule of thumb though it is always preferable to keep the same representation throughout an application pipeline, since conversion may introduce numerical artifacts.

There are also properties in the continuous setting that none of the existing representations can easily preserve. For instance, one could argue that the two scalar potentials in the Helmholtz decomposition should have the same number of degrees of freedom, but neither face-based nor edge-based presentations have matching DoFs for the potentials of gradients and rotated gradients. The vertex-based representation, on the other hand, uses collocated basis functions for gradient and rotated gradient fields, but it does not ensure an orthogonal decomposition of vector fields.

In conclusion, despite great advances in the representation and processing of discrete vector fields, there is still a considerable demand for new and improved formulations. Vector fields are such a basic ingredient of geometry processing, as well as a common constituent of the more general family of tensors, that any novel computational framework that offers simplicity of representation and versatility of processing is bound to have wide repercussions and adoption. The application of these generalized computational tools on irregular and regular meshes will also likely improve the whole geometry processing pipeline, from modeling, design, and analysis, to the simulation of physical phenomena on surfaces. We hope that the readers of these notes will be challenged to become the architects of these future developments.

References

ABRAHAM, R., MARSDEN, J. E., AND RATIU, R. 1988. *Manifolds, Tensor Analysis, and Applications: 2nd Edition*. Springer-Verlag.

ANDERSEN, V., DESBRUN, M., BRENTZEN, J., AND AANS, H. 2009. Height and tilt geometric texture. In *Advances in Visual Computing*, vol. 5875 of *Lecture Notes in Computer Science*. Springer, 656–667.

ARNOLD, D. N., FALK, R. S., AND WINTHER, R. 2006. Finite element exterior calculus, homological techniques, and applications. *Acta Numerica 15*, 1–155.

AZENCOT, O., BEN-CHEN, M., CHAZAL, F., AND OVSJANIKOV, M. 2013. An operator approach to tangent vector field processing. *Comp. Graph. Forum 32*, 5, 73–82.

AZENCOT, O., OVSJANIKOV, M., CHAZAL, F., AND BEN-CHEN, M. 2015. Discrete derivatives of vector fields on surfaces – an operator approach. *ACM Trans. Graph. 34*, 3.

BEN-CHEN, M., BUTSCHER, A., SOLOMON, J., AND GUIBAS, L. 2010. On discrete killing vector fields and patterns on surfaces. *Comp. Graph. Forum 29*, 5, 1701–1711.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3.

BOSSAVIT, A., AND KETTUNEN, L. 1999. Yee-schemes on a tetrahedral mesh, with diagonal lumping. *Int. J. Num. Model. 12*, 129–142.

CABRAL, B., AND LEEDOM, L. C. 1993. Imaging vector fields using line integral convolution. In *Proc. ACM/SIGGRAPH Conf.*, 263–270.

CRANE, K., DESBRUN, M., AND SCHRÖDER, P. 2010. Trivial connections on discrete surfaces. *Comp. Graph. Forum 29*, 1525–1533.

CRANE, K., DE GOES, F., DESBRUN, M., AND SCHRDER, P. 2013. *Digital Geometry Processing with Discrete Exterior Calculus*. Course Notes. ACM SIGGRAPH.

DE GOES, F., AND CRANE, K., 2010. A simplified algorithm for simply-connected surfaces. Caltech CMS technical report.

DE GOES, F., LIU, B., BUDNINSKIY, M., TONG, Y., AND DESBRUN, M. 2014. Discrete 2-tensor fields on triangulations. *Comp. Graph. Forum 33*, 5, 13–24.

DE GOES, F., DESBRUN, M., MEYER, M., AND DEROSE, T. 2016. Subdivision exterior calulcus for geometry processing. *ACM Trans. Graph. 35(4)*.

DESBRUN, M., AND DE GOES, F. 2014. The power of orthogonal duals. In *Mathematical Progress in Expressive Image Synthesis I*, vol. 4 of *Mathematics for Industry*.

DESBRUN, M., KANSO, E., AND TONG, Y. 2006. Discrete differential forms for computational modeling. [Grinspun et al. 2006].

DIAMANTI, O., VAXMAN, A., PANOZZO, D., AND SORKINE-HORNUNG, O. 2014. Designing n-polyvector fields with complex polynomials. *Comp. Graph. Forum 33*, 5.

DODZIUK, J., AND PATODI, V. K. 1976. Riemannian structures and triangulations of manifolds. *J. Ind. Math. Soc. 40*, 1–4, 1–52.

EISENBERG, M., AND GUY, R. 1979. A proof of the hairy ball theorem. *The American Mathematical Monthly 86*, 7, 571–574.

ELCOTT, S., AND SCHRÖDER, P. 2006. Building your own DEC at home. [Grinspun et al. 2006].

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26*, 1.

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. *ACM Trans. Graph. 26*, 3.

GAWLIK, E., MULLEN, P., PAVLOV, D., MARSDEN, J., AND DESBRUN, M. 2011. Geometric, variational discretization of continuum theories. *Physica D: Nonlinear Phenomena 240*, 21, 1724–1760.

GORTLER, S. J., GOTSMAN, C., AND THURSTON, D. 2006. Discrete one-forms on meshes and applications to 3d mesh parameterization. *Comput. Aided Geom. Des. 33*, 2, 83–112.

GRIMM, C. M., AND HUGHES, J. F. 1995. Modeling surfaces of arbitrary topology using manifolds. In *Proc. ACM/SIGGRAPH Conf.*, 359–368.

GRINSPUN, E., SCHRÖDER, P., AND DESBRUN, M. 2006. *Discrete Differential Geometry*. Course Notes. ACM SIGGRAPH.

GU, X., AND YAU, S.-T. 2003. Global conformal surface parameterization. In *Proc. Symp. Geom. Proc.*, 127–137.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *Proc. ACM/SIGGRAPH Conf.*, 517–526.

HIRANI, A. N. 2003. *Discrete Exterior Calculus*. PhD thesis, Caltech.

KALBERER, F., NIESER, M., AND POLTHIER, K. 2007. Quadcover: Surface parameterization using branched coverings. *Comp. Graph. Forum 26*, 3, 375–384.

KNÖPPEL, F., CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2013. Globally optimal direction fields. *ACM Trans. Graph. 32*, 4.

KNÖPPEL, F., CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2015. Stripe patterns on surfaces. *ACM Trans. Graph. 34*, 4.

KOOPMAN, B. O. 1931. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences 17*, 315–318.

LIU, B.-B., WENG, Y.-L., WANG, J.-N., AND TONG, Y.-Y. 2013. Orientation field guided texture synthesis. *Journal of Computer Science and Technology 28*, 5.

LIU, B., TONG, Y., DE GOES, F., AND DESBRUN, M. 2016. Discrete connection and covariant derivative for vector field analysis and design. *ACM Trans. Graph. 35(3)*, Art. 23.

MACNEAL, R. 1949. *The Solution of Partial Differential Equations by means of Electrical Networks*. PhD thesis, Caltech.

MEBARKI, A., ALLIEZ, P., AND DEVILLERS, O. 2005. Farthest point seeding for efficient placement of streamlines. In *IEEE Visualization*, 479–486.

MERCAT, C. 2001. Discrete Riemannian surfaces and the Ising model. *Comm. in Math. Physics 218*, 1, 177–216.

MULLEN, P., CRANE, K., PAVLOV, D., TONG, Y., AND DESBRUN, M. 2009. Energy-preserving integrators for fluid animation. *ACM Trans. Graph. 28*, 3.

NÉDÉLEC, J.-C. 1980. Mixed finite elements in $\mathbb{R}^3$. *Numer. Math. 35*, 315–341.

NIESER, M., PALACIOS, J., POLTHIER, K., AND ZHANG, E. 2012. Hexagonal global parameterization of arbitrary surfaces. *IEEE Trans. Vis. Comp. Graph. 18*, 6, 865–878.

PALACIOS, J., AND ZHANG, E. 2007. Rotational symmetry field design on surfaces. *ACM Trans. Graph. 26*, 3.

PANOZZO, D., LIPMAN, Y., PUPPO, E., AND ZORIN, D. 2012. Fields on symmetric surfaces. *ACM Trans. Graph. 31*, 4.

PAVLOV, D., MULLEN, P., TONG, Y., KANSO, E., MARSDEN, J., AND DESBRUN, M. 2011. Structure-preserving discretization of incompressible fluids. *Physica D: Nonlinear Phenomena 240*, 6, 443–458.

POLTHIER, K., AND PREUSS, E. 2000. Variational approach to vector field decomposition. In *Data Visualization*, Eurographics. 147–155.

POLTHIER, K., AND PREUSS, E. 2003. Identifying vector field singularities using a discrete Hodge decomposition. In *Vis. and Math. III*. 113–134.

RAY, N., LI, W. C., LÉVY, B., SHEFFER, A., AND ALLIEZ, P. 2006. Periodic global parameterization. *ACM Trans. Graph. 25*, 4, 1460–1485.

RAY, N., VALLET, B., LI, W. C., AND LÉVY, B. 2008. N-symmetry direction field design. *ACM Trans. Graph. 27*, 2.

RAY, N., VALLET, B., ALONSO, L., AND LEVY, B. 2009. Geometry-aware direction field processing. *ACM Trans. Graph. 29*, 1.

RUFAT, D., MASON, G., MULLEN, P., AND DESBRUN, M. 2014. The chain collocation method: a spectrally accurate calculus of forms. *Journal of Computational Physics 257*, Part B, 1352–1372.

SPIVAK, M. 1979. *A comprehensive introduction to differential geometry. Vol. II (2nd edition)*. Publish or Perish Inc.

TONG, Y., LOMBEYDA, S., HIRANI, A. N., AND DESBRUN, M. 2003. Discrete multiscale vector field decomposition. *ACM Trans. Graph. 22*, 3, 445–452.

TONG, Y., ALLIEZ, P., COHEN-STEINER, D., AND DESBRUN, M. 2006. Designing quadrangulations with discrete harmonic forms. In *Proc. Symp. Geom. Proc.*, 201–210.

VAXMAN, A., CAMPEN, M., DIAMANTI, O., PANOZZO, D., BOMMES, D., HILDEBRANDT, K., AND BEN-CHEN, M. 2016. *Directional Field Synthesis, Design, and Processing*. Course Notes. Eurographics.

WANG, K., WEIWEI, TONG, Y., DESBRUN, M., AND SCHRÖDER, P. 2006. Edge subdivision schemes and the construction of smooth vector fields. *ACM Trans. Graph. 25*, 3, 1041–1048.

WARDETZKY, M. 2006. *Discrete Differential Operators on Polyhedral Surfaces - Convergence and Approximation*. PhD thesis, Freie Universität Berlin.

WHITNEY, H. 1957. *Geometric Integration Theory*. Princeton University Press.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2006. Vector field design on surfaces. *ACM Trans. Graph. 25*, 4, 1294–1326.