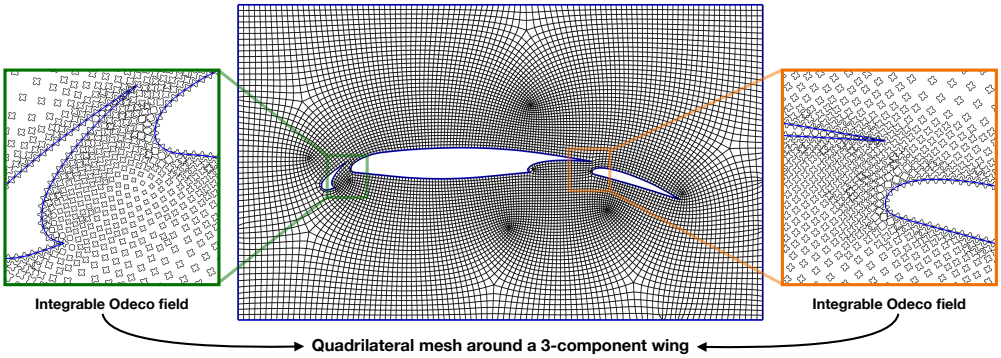
Graphical Abstract

Size-Controlled Quadrilateral Meshing using Integrable Odeco Fields

Mattéo Couplet, Alexandre Chemin, Jean-François Remacle



Highlights

Size-Controlled Quadrilateral Meshing using Integrable Odeco Fields

Mattéo Couplet, Alexandre Chemin, Jean-François Remacle

- Novel method for computing planar quadrilateral meshes with sizing constraints
- Uses integrable orthogonal frame fields represented by Odeco tensors
- Introduces a Lie bracket formulation for integrability in tensor representation
- Optimizes smooth, integrable frame fields with correct singularity placement
- Proposes a greedy quantization approach for mesh extraction from parametrization

Size-Controlled Quadrilateral Meshing using Integrable Odeco Fields

Mattéo Couplet[a], Alexandre Chemin[b], Jean-François Remacle[b]

[a] *Boston University Department of Computer Science, 665 Commonwealth Avenue, Boston, MA, United States*
[b] *UCLouvain Institute of Mechanics, Materials and Civil Engineering, Avenue Georges Lemaître 4, Louvain-la-Neuve, Belgium*

Abstract

This paper proposes a novel approach for computing planar quadrilateral meshes complying with sizing prescriptions on boundary and feature curves. The method relies on computing *integrable* orthogonal frame fields, whose symmetries are implicitly represe nted using orthogonally decomposable (*odeco*) tensors. To formulate an integrability criterion, we express the frame field's Lie bracket solely in terms of the tensor representation; this is made possible by studying the sensitivity of the frame with respect to perturbations in the tensor. We construct an energy formulation that computes smooth and integrable frame fields in both isotropic and anisotropic settings. The solver creates and places the singularities required to fit the sizing constraints with the correct topology. The computed frame field is integrated to a seamless parametrization that is aligned with the frame field, and we propose a mesh extraction method that relies on a greedy quantization of the parametrization.

Keywords:
mesh generation, anisotropic meshing, parametrization, frame field design

1. Introduction

Meshes composed of quadrilaterals are known to offer superior performance than their triangular counterpart when used as a support in numerical simulations, however quad meshers still do not meet the same level of robustness, flexibility and quality required for industrial applications as triangular meshes. In three dimensions, generating fully hexahedral meshes

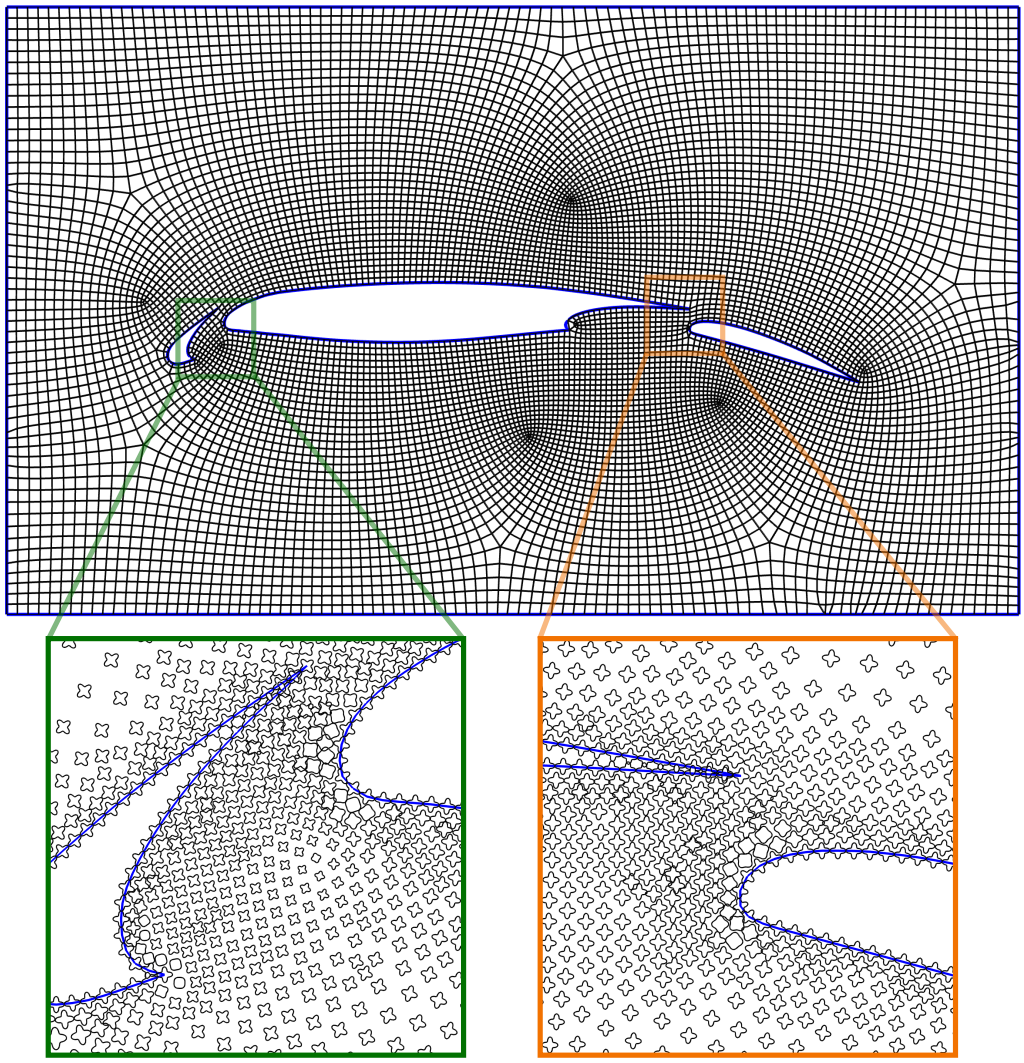


Figure 1: Overview of the approach on a three-component wing cross-section. An integrable odeco field (close-up views) is computed that finds a frame field topology to comply with user-provided sizing constraints (ratio of 1:2 between wing boundaries and outer box). From this frame field a quadrilateral mesh is computed that closely matches the orientation and sizing of the frames.

is an even more difficult task, and their industrial use is consequently very limited despite a persistent demand from practitioners.

Quadrilateral and hexahedral meshing are challenging because they couple (a) a *geometric* problem, minimizing the distortion of the elements, and (b) a *combinatorial* problem, achieving a conforming connectivity structure that is free of hanging nodes. If one also desires a *multi-block* structure, a layer of complexity is added, requiring (c) an adequately coarse block structure. The last two decades have seen the emergence of *field-based approaches*, which divide the problem in two main steps. (1) The combinatorial constraints are ignored and a *frame field* is computed; a frame is a set of 2/3 directions representing the orientation (and sometimes the size) of a quad/hex. This frame field can be seen as a continuous representation of a mesh. (2) A mesh is generated using guidance from the frame field. This can be done through numerous approaches which we review below.

An issue shared by most existing works is that no quad/hex mesh exactly follows the frame field computed a priori. This is due to the frame field not being *integrable*; we elaborate on this in section 3. This limitation of frame field-based methods means that the mesh deviates from the frame field and the user must compromise on element quality, control over element size and orientation, and control over the topology of the mesh. A more fundamental issue arises when the computed frame field has a topology (i.e., a singularity configuration) that is not meshable, making it unusable. This happens in the 2D case when limit cycles appear; in 3D the singular structure is very often invalid due to the presence of non-meshable singular edges and nodes.

Our contribution overcomes this limitation in 2D and allows to generate frame fields on planar surfaces that are *integrable* (fig. 1, bottom); during this process the user can prescribe any orientation or size constraints along feature curves. Integrability guarantees that the frame field can be integrated to a seamless parametrization that is exactly aligned with the frame field and respects the prescribed constraints The parametrization is then quantized and a quad mesh is extracted (fig. 1, top); for this we propose a greedy strategy that is free of any costly integer optimization. The quality of this greedy quantization is made possible by the absence of limit cycles in the integrable frame field.

Our frame field approach relies on orthogonally decomposable (*odeco*) tensors which serve as an algebraic representation for the frames. By studying the so-called *eigenvalue sensitivity problem* for tensors, we are able to formulate the problem completely in terms of this implicit algebraic repre-

sentation. This work is the first achieving integrability using odeco tensors. As they can be extended to three dimensions, this contribution paves the way for hex-meshable frame fields, which are allegedly the key to the long-standing problem of robustly generating optimal hexahedral meshes.

2. Related work

Frame field design. 4-direction fields, i.e., assignments of four directions to every point of a surface, and their application to quad meshing have been studied extensively in the computer graphics community; Vaxman et al. (2016) provide a review. Some important works (Ray et al., 2008; Crane et al., 2010) assume the field topology to be known in advance and optimize for cross field smoothness. If the topology is unknown, it can be represented explicitly and optimized through integer variables such as in (Bommes et al., 2009), which often leads to costly mixed-integer formulations. Alternatively, the topology can also be implicitly encoded in the field representation; our work fits in this category. Earlier works in this line of research include (Ray et al., 2006; Knöppel et al., 2013). A key challenge is being able to represent singularities while maintaining unit norm frames. More recently, this problem has been re-framed in the Ginzburg-Landau framework, which replaces the ill-posed unit norm constraint by a penalty term taken to the limit (Beaufort et al., 2017; Viertel and Osting, 2019).

Field-guided meshing. As an intermediate step before generating a quad mesh, a global parametrization is often computed on the domain using the guidance from the frame field. Among notable works we can cite (Kälberer et al., 2007; Bommes et al., 2009) who integrate the frame field in a least-squares sense to find the parametrization that best aligns to the frame field, or (Ray et al., 2006), who perform a curl-reduction procedure a posteriori. Another approach, e.g., in (Myles et al., 2014), is to trace out parametric lines from the frame field to form quadrilateral patches. Getting a quad mesh from a global parametrization is typically done using quantization methods, where a T-mesh is traced out and its edges are given integer lengths; see, e.g., (Campen et al., 2015; Lyon et al., 2021). Obtaining a quad mesh from a frame field can also be achieved robustly using frontal methods, by inserting points with guidance from the frame field. However, the topology of the frame field is usually not preserved, and remeshing and smoothing is often necessary a posteriori to fix regions of bad quality; see, e.g., (Reberol et al., 2021).

In this work, the parametrization step is made trivial as an integrable frame field is equivalent to a seamless parametrization. Hence, integrating the frame field provides a parametrization that is exactly aligned and matches the sizing of the frame field. This confers the user a complete control over the sizing and orientation of the mesh, a desirable property that cannot be achieved through conventional cross field guided meshing. For extracting a mesh from this seamless parametrization, we leverage the property that our geometries are planar and that the parametrizations are free of any limit cycles. Thanks to this, we can build a greedy quantization strategy that works directly on the parametrization's quadrilateral layout.

Frame representation. Computing frame fields with implicit topology requires a representation that is invariant to the ordering and symmetries of the frame vectors. The majority of frame field-driven methods use representations of *unit* frames, or *crosses*, as they only care about the orientation of the mesh; a review can be found in (Vaxman et al., 2016). One of the most prominent representations is the trigonometric pair $(\cos(4\theta), \sin(4\theta))$ (or, equivalently, the complex exponential $e^{i4\theta}$) which effectively encodes rotational symmetries and is trivially normalized to unit norm (Ray et al., 2006; Palacios and Zhang, 2007; Kowalski et al., 2013; Knöppel et al., 2013). However, this representation cannot encode sizes along with the directions. On the other hand, PolyVectors, introduced by (Diamanti et al., 2014), encode the frame vectors as the complex roots of a degree 4 polynomial. They have the advantage that they can represent non-orthogonal frames (as done in (Sageman-Furnas et al., 2019)), but they offer no straightforward extension to volumetric frames. For representing 3D frames, a spherical function is often used that is maximal in the directions of the frame. This function is encoded by its coefficients in the basis of spherical harmonics (Huang et al., 2011). Zhang et al. (2020) use this octahedral representation to design smooth cross fields on surfaces aligning to any specified extrinsic feature curves. As highlighted by (Chemin et al., 2019), the spherical harmonics representation can also be interpreted as a 4th-order tensor. Recently, Palmer et al. (2020) have proposed an extension of these representations that encodes sizes along the directions, relying on the so-called orthogonally decomposable (Odeco) tensors. As they are the only known representation that can encode three-dimensional orthogonal and scaled frames, we propose to use them in the 2D setting so that they can offer a 3D extension for later work.

Integrable frame fields. Motivated by the issue where the mesh is not aligned to the computed frame field, a handful of works have focused on computing frame fields that are *integrable*, i.e., the direction vectors are the gradients of a seamless parametrization. This property holds if the two vector fields are *curl-free*. Diamanti et al. (Diamanti et al., 2015) achieve this using PolyVector fields. Unlike our approach, the integrability condition cannot be expressed in terms of the PolyVector coefficients, and the optimization is done on the explicit direction vectors. In (Sageman-Furnas et al., 2019), PolyVector fields are used to construct *Chebyshev nets*, i.e., quadrilateral meshes with a uniform size but not necessarily orthogonal. Our goal is slightly different (orthogonal quad meshing) but we address the same challenges regarding control of size and orientation. Like us, they optimize a frame field instead of a parametrization Jacobian, and express the integrability through the Lie bracket. The main advantage of odeco tensors compared to PolyVectors is that the former offer a natural extension to 3D frame fields. In (Jezdimirović et al., 2022), an integrable frame field is computed given a user-imposed (or pre-computed) singularity configuration. The frame fields are exactly integrable and the method can even remove limit cycles that appear in most field-based approaches. Having to specify singularities however is a significant limitation, especially in the 3D case where almost all existing frame field solvers produce invalid singular configurations. (Vekhter et al., 2025) are the first to address integrability optimization for volumetric frame fields by directly optimizing over frame vectors defined on tetrahedra, and matching *tensor moments* on triangle faces. Their approach does not guarantee hex-meshability however, and importantly does not address the fundamental issue of discouraging "zipper nodes", which shows that integrability of 3D frame fields remains an open problem.

2.1. Overview

In section 3 we review the mathematical formalism behind parametrizations, their Jacobians and frame fields. We define the integrability property for frame fields and show how they are equivalent to seamless parametrizations. In section 4, we review the algebraic representation for our frames, which is the two-dimensional version of the odeco tensors proposed by Palmer et al. (2020). In section 5, we present two contributions: (a) an expression for the sensitivity of a higher-order tensor's eigenvectors with respect to small perturbations of the tensor, and we use this result to show (b) an expression of a frame field's Lie bracket only in terms of the tensor field coefficients

and their derivatives. In section 6 we present a last contribution, an energy formulation that optimizes for smooth integrable frame fields, for both the isotropic and anisotropic cases. In section 7 we present our pipeline to compute a seamless parametrization from an integrable frame field, and our greedy strategy to extract a quad mesh from the parametrization. Section 8 demonstrates the effectiveness of the algorithm.

3. Integrable frame fields

Seamless parametrizations. The idea of parametrization-based quadrilateral meshing is to map a coordinate system onto the domain to be meshed. The coordinate lines of this map then provide, at least locally, a quadrilateral mesh on the domain. Consider a planar domain Ω; we wish to parametrize Ω by assigning to each point $\mathbf{p}$ of Ω a pair of coordinates $(u(\mathbf{p}), v(\mathbf{p}))$. In general, this parametrization cannot be defined as a global continuous function; *cuts* need to be introduced to obtain a disk topology on which u and v can be defined continuously. The parametrization is said to be *seamless* if across every cut, the coordinates transform through a rigid rotation of some multiple of 90°; if u' and v' are the coordinates on the other side of the cut, then the transformation is seamless if

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}}_{\mathbf{R}}^{k} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} s \\ t \end{pmatrix}, \tag{1}$$

for some $k \in \{0, 1, 2, 3\}$ and fixed translation (s, t). If, on top of being seamless, the translation (s, t) is integer and the singularities lie at integer coordinates, then the parametrization is called an *integer-grid map* and exactly corresponds to a quadrilateral mesh.

Computing seamless parametrizations is a challenging task due to the discrete nature of the cut graph involved, which is unknown a priori. Instead, most existing works focus on computing the Jacobian $(\boldsymbol{\nabla} u, \boldsymbol{\nabla} v)$ of a parametrization. Let $(\mathbf{d}u, \mathbf{d}v)$ be a pair of vector fields on Ω; they form the Jacobian of a parametrization (and are said to be *integrable*) if they are curl-free:

$$\boldsymbol{\nabla} \times \mathbf{d}u = \mathbf{0}, \quad \boldsymbol{\nabla} \times \mathbf{d}v = \mathbf{0}, \tag{2}$$

and they transform through a 90°-multiple rotation across cuts (which amounts

to differentiating (1)):

$$\begin{pmatrix} \mathbf{d}u' \\ \mathbf{d}v' \end{pmatrix} = \mathbf{R}^k \begin{pmatrix} \mathbf{d}u \\ \mathbf{d}v \end{pmatrix}. \tag{3}$$

If these properties are verified then the vector fields $(\mathbf{d}u, \mathbf{d}v)$ are essentially equivalent to a seamless parametrization, up to a global rigid rotation of the coordinate map. Computing a parametrization's Jacobian is a more doable task, provided one can appropriately encode the symmetries in (3); we elaborate on that in section 4.

Frame fields. Consider now the inverse of the parametrization $\mathbf{r}$ that maps a pair of coordinates to a point of the domain: $\mathbf{p} = \mathbf{r}(u, v)$. This map can only be defined locally since the parametrization can map different points to the same coordinates. The Jacobian matrix of $\mathbf{r}$ defines the *coordinate vectors* $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{J}_\mathbf{r} = \begin{pmatrix} \dfrac{\partial \mathbf{r}}{\partial u} & \dfrac{\partial \mathbf{r}}{\partial v} \end{pmatrix} = \begin{pmatrix} \mathbf{u} & \mathbf{v} \end{pmatrix}, \tag{4}$$

forming a *coordinate frame* $\mathbf{F} = \begin{pmatrix} \mathbf{u} & \mathbf{v} \end{pmatrix}$. Since $\mathbf{r}$ is the inverse of the parametrization, their Jacobians are inverse of one another, meaning that the coordinate frame is the inverse of the parametrization's Jacobian:

$$\mathbf{J}_{\mathbf{r}^{-1}} = (\mathbf{J}_\mathbf{r})^{-1} \quad \Rightarrow \quad \begin{pmatrix} \boldsymbol{\nabla} u \\ \boldsymbol{\nabla} v \end{pmatrix} = \mathbf{F}^{-1}. \tag{5}$$

Note that, looking at the off-diagonal terms in $\mathbf{F}^{-1}\mathbf{F} = \mathbf{I}$, we have $\boldsymbol{\nabla} u \cdot \mathbf{v} = \boldsymbol{\nabla} v \cdot \mathbf{u} = 0$, meaning that the coordinate vectors $\mathbf{u}, \mathbf{v}$ are orthogonal to the gradients of v, u, respectively, and therefore parallel to the isolines of v, u respectively. This shows how the frame field corresponds to a quad mesh through discrete isolines of the coordinate map u, v. Looking at the diagonal terms, we have $\boldsymbol{\nabla} u \cdot \mathbf{u} = \boldsymbol{\nabla} v \cdot \mathbf{v} = 1$, which indicates that the integer isolines are spaced out according to $\|\mathbf{u}\|$ and $\|\mathbf{v}\|$. We illustrate the introduced concepts and notations on fig. 2.

Orthogonal case. In this work we are interested in frames that are orthogonal: $\mathbf{u} \perp \mathbf{v}$. Let $\hat{\mathbf{u}} = \mathbf{u}/\|\mathbf{u}\|$ and $\hat{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|$; the frame matrix $\mathbf{F}$ is then diagonal in the local orthonormal basis $(\hat{\mathbf{u}}, \hat{\mathbf{v}})$, and is easily inverted:

$$\begin{pmatrix} \boldsymbol{\nabla} u \\ \boldsymbol{\nabla} v \end{pmatrix} = \begin{pmatrix} \mathbf{u} & \mathbf{v} \end{pmatrix}^{-1} = \begin{pmatrix} \|\mathbf{u}\| & 0 \\ 0 & \|\mathbf{v}\| \end{pmatrix}^{-1} = \begin{pmatrix} 1/\|\mathbf{u}\| & 0 \\ 0 & 1/\|\mathbf{v}\| \end{pmatrix}. \tag{6}$$

The Jacobian of the coordinate map can thus be computed by simply scaling the frame's coordinate vectors: $\boldsymbol{\nabla} u = \mathbf{u}/\|\mathbf{u}\|^2$ and $\boldsymbol{\nabla} v = \mathbf{v}/\|\mathbf{v}\|^2$.

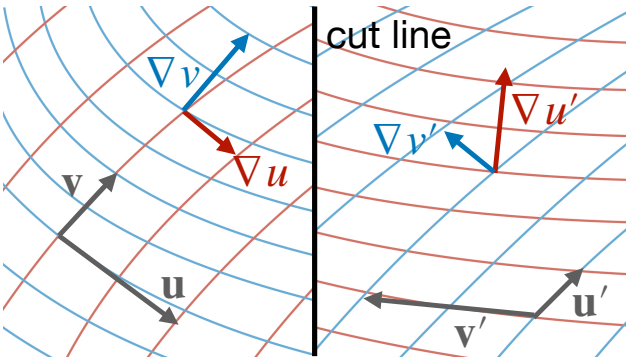


Figure 2: A seamless parametrization across a cut. The isolines of the coordinate maps $u(\mathbf{p})$ and $v(\mathbf{p})$ are depicted in red and blue, respectively. Notice how the gradients $\boldsymbol{\nabla} u, \boldsymbol{\nabla} v$ and the frame vectors $\mathbf{u}, \mathbf{v}$ undergo a 90° rotation as they cross the cut.

Frame field integrability. Several previous works, e.g., (Diamanti et al., 2015), build upon the curl-free condition (2) and compute parametrizations by computing vector fields $(\mathbf{d}u, \mathbf{d}v)$. However, in our case we want to be able to express the integrability condition directly in terms of our implicit frame representation, and expressing the individual curls of $\mathbf{d}u$ and $\mathbf{d}v$ is not possible as the two vectors are mixed in a single algebraic representation. Instead, we use the following property: the vector fields $(\mathbf{d}u, \mathbf{d}v)$ are *curl-free* if and only if the corresponding frame vectors $\begin{pmatrix}\mathbf{u} & \mathbf{v}\end{pmatrix} = \begin{pmatrix}\mathbf{d}u \\ \mathbf{d}v\end{pmatrix}^{-1}$ have *zero Lie bracket*:

$$[\mathbf{u}, \mathbf{v}] = \boldsymbol{\nabla}_{\mathbf{u}}\mathbf{v} - \boldsymbol{\nabla}_{\mathbf{v}}\mathbf{u} = \mathbf{0}, \tag{7}$$

where $\boldsymbol{\nabla}_{\mathbf{u}}$ and $\boldsymbol{\nabla}_{\mathbf{v}}$ are directional derivatives. Note that this is true even if the frame field is not orthogonal. We refer to (Sageman-Furnas et al., 2019) for a proof, which calls for theory of differential forms on manifolds. Hence, a frame field $\mathbf{F} = \begin{pmatrix}\mathbf{u} & \mathbf{v}\end{pmatrix}$ guides a parametrization if its Lie bracket is zero, and this parametrization is seamless if the frame vectors transform through 90° rotations across cuts, as in (3). We will see in section 5 how the Lie bracket can be expressed in terms of the implicit frame representation.

Illustration. We provide a concrete example of an integrable frame field to illustrate the concepts introduced. Consider a frame field $(\mathbf{u}, \mathbf{v})$ on a domain described by polar coordinates:

$$\mathbf{u}(r, \theta) = r\hat{\mathbf{e}}_r, \quad \mathbf{v}(r, \theta) = r\hat{\mathbf{e}}_\theta, \tag{8}$$

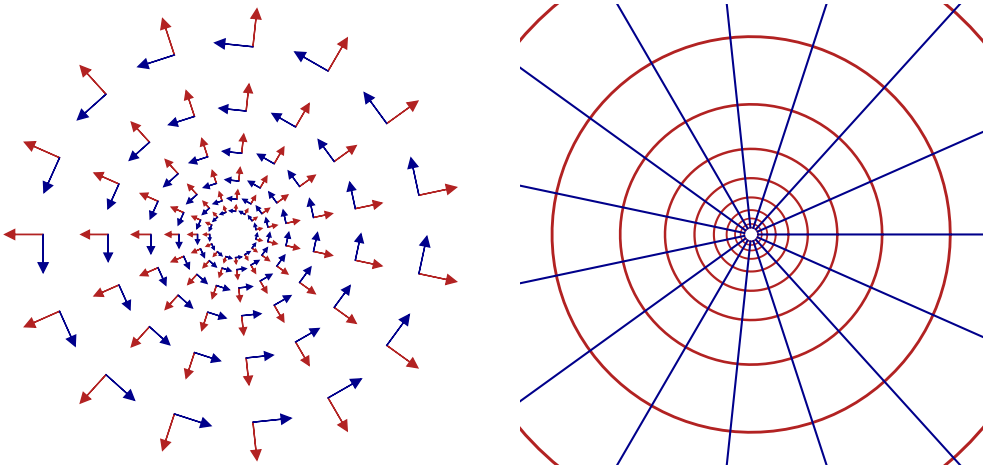


Figure 3: Example of an integrable frame field $(r\hat{\mathbf{e}}_r, r\hat{\mathbf{e}}_\theta)$ and its corresponding parametrization $(\log r, \theta)$, illustrated with a set of isolines of u and v.

where $\hat{\mathbf{e}}_r$ and $\hat{\mathbf{e}}_\theta$ are the radial and tangential unit vectors, respectively. One can check that the directional derivatives coincide: $\boldsymbol{\nabla}_\mathbf{u}\mathbf{v} = \boldsymbol{\nabla}_\mathbf{v}\mathbf{u} = r\hat{\mathbf{e}}_\theta$, and their Lie bracket is zero. The parametrization gradient is obtained by inverting the frame matrix $\mathbf{F}$:

$$\begin{pmatrix}\mathbf{d}u\\ \mathbf{d}v\end{pmatrix} = \begin{pmatrix}\mathbf{u} & \mathbf{v}\end{pmatrix}^{-1} = \begin{pmatrix}r & 0\\ 0 & r\end{pmatrix}^{-1} = \begin{pmatrix}1/r & 0\\ 0 & 1/r\end{pmatrix}. \tag{9}$$

Hence, $\mathbf{d}u = \hat{\mathbf{e}}_r/r$ and $\mathbf{d}v = \hat{\mathbf{e}}_\theta/r$, and one can check that these vector fields are curl-free: $\boldsymbol{\nabla}\times\mathbf{d}u = \boldsymbol{\nabla}\times\mathbf{d}v = \mathbf{0}$. Integatring $\mathbf{d}u$ and $\mathbf{d}v$ then gives rise to a parametrization $(u, v) = (\log r, \theta)$, defined everywhere except at $r = 0$ and with a cut along the half-line $(r, 0)$ with $r > 0$. This integrable frame field and its parametrization are illustrated in fig. 3.

4. Algebraic representation of frames

Due to the topology of the domain Ω and the presence of singularities in the frame field, it is not possible to define a globally continuous frame field $(\mathbf{u}, \mathbf{v})$. One needs to introduce *cuts* in the domain across which the frame vectors are symmetric according the rotations defined by (3). In order to compute a frame field without the need to introduce cuts, we extend the

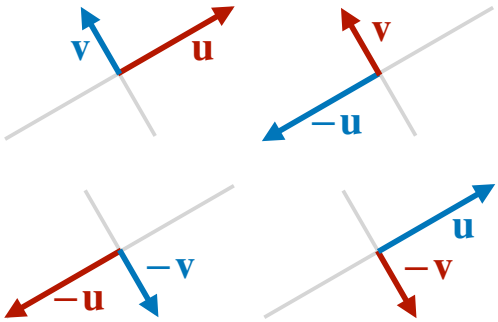


Figure 4: Equivalence class of the 90° rotations of an orthogonal frame $(\mathbf{u}, \mathbf{v})$.

notion of frame such that it is invariant up to 90° rotations:

$$\mathbf{F} = \{(\mathbf{u}, \mathbf{v}), (\mathbf{v}, -\mathbf{u}), (-\mathbf{u}, -\mathbf{v}), (-\mathbf{v}, \mathbf{u})\}. \tag{10}$$

These symmetries are illustrated in fig. 4. Given this equivalence class, one needs to introduce an algebraic representation that unambiguously represents a frame and supports arithmetic operations. To this end we resort to the class of two-dimensional orthogonally decomposable (or Odeco) tensors, which were introduced by Palmer et al. (Palmer et al., 2020) in the three-dimensional case. In this section we briefly lay out the theory of odeco tensors, and refer the reader to the paper for more details.

Odeco tensors. Although the odeco theory applies to tensors of arbitrary order, we restrict the definitions to fourth-order tensors, as they are sufficient to represent frames. Let $S^4(\mathbb{R}^n)$ be the space of $n \times n \times n \times n$ *fully symmetric tensors*, i.e., their real entries T_{i_1,i_2,i_3,i_4} are invariant up to permutations of the indices i_1, i_2, i_3, i_4. n is the *dimension* of the tensor and corresponds to the dimension of the frame we wish to represent (2 or 3). A combinatorial inspection shows that tensor $\mathbf{T}$ has 5 independent components for $n = 2$, and 15 components for $n = 3$.

Robeva et al. (Robeva, 2016) have studied a special class of tensors $\mathbf{T}$ that are said to be *orthogonally decomposable*, or *odeco* for short, if they can be written as

$$\mathbf{T} = \lambda_1 \mathbf{v}_1^{\otimes 4} + \cdots + \lambda_n \mathbf{v}_n^{\otimes 4}, \tag{11}$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_n$ form an orthonormal basis of $\mathbb{R}^n$. Note that for second-order tensors, $S^2(\mathbb{R}^n)$ is the space of real symmetric matrices; these are always orthogonally decomposable according to the spectral theorem. Higher-order

tensors (≥ 3) however, admit rank-one decompositions with more than n terms.

The notion of eigenvector can be generalized to higher-order tensors: $\mathbf{w}$ is an eigenvector of $\mathbf{T}$ with eigenvalue λ if

$$\begin{aligned} &\mathbf{T}\mathbf{w}^3 = \lambda \mathbf{w}, \quad \text{or, in Einstein notation,} \\ &T_{i,j_2,j_3,j_4} w_{j_2} w_{j_3} w_{j_4} = \lambda w_i. \end{aligned} \tag{12}$$

One can easily see that, for an odeco tensor $\mathbf{T}$, the vectors $\mathbf{v}_k$ in (11) are eigenvectors of $\mathbf{T}$ with corresponding eigenvalue λ_k.

As odeco tensors only form a small subset of the space of fully symmetric tensors, one needs to characterize what makes a tensor odeco. The major result of Robeva et al. (Robeva, 2016) is the fact that odeco tensors form an algebraic variety defined by a set of homogeneous quadratic equations in the tensor coefficients. More precisely, $\mathbf{T}$ is odeco if the contraction $(\mathbf{T} * \mathbf{T})$ defined by

$$(\mathbf{T} * \mathbf{T})_{i_1,i_2,i_3,j_1,j_2,j_3} = T_{i_1,i_2,i_3,s} T_{j_1,j_2,j_3,s} \tag{13}$$

is a fully symmetric tensor, i.e.,

$$\mathbf{T} * \mathbf{T} \in S^6(\mathbb{R}^n). \tag{14}$$

Frames as odeco tensors. To represent a 2D orthogonal frame in a way that is invariant to the permutations in (10), we use an odeco tensor of which the eigenvalues and eigenvectors, as defined in (12), match the frame directions and sizes. Specifically, let $\mathbf{u}, \mathbf{v}$ be the frame vectors, λ, μ their norms and $\hat{\mathbf{u}}, \hat{\mathbf{v}}$ their normalization such that $\mathbf{u} = \lambda \hat{\mathbf{u}},\ \mathbf{v} = \mu \hat{\mathbf{v}}$. Then the corresponding tensor is defined as

$$\mathbf{T} = \lambda \hat{\mathbf{u}}^{\otimes 4} + \mu \hat{\mathbf{v}}^{\otimes 4}. \tag{15}$$

Notice how this tensor remains the same when plugging in any of the rotations of (10). We can now justify the choice for a fourth-order tensor: an even order is required for the tensor to be invariant to the signs of the vectors, and order 2 cannot be chosen since, in the case $\lambda = \mu$, the tensor degenerates to a multiple of the identity matrix, $\mathbf{T} = \lambda \mathbf{I}$, and then any vector of $\mathbb{R}^2$ is an eigenvector of $\mathbf{T}$ and one loses the direction vectors $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$. Order 4 is therefore the lowest possible order for our purpose.

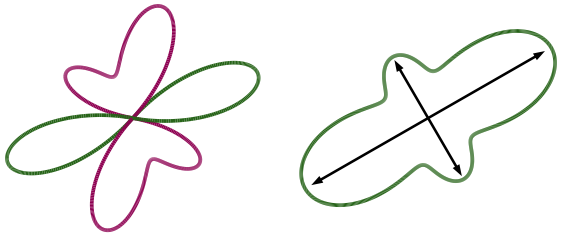


Figure 5: Polynomial representation $p_{\mathbf{T}}(\theta)$ of (left) an arbitrary tensor, and (right) an odeco tensor, with the corresponding frame vectors. Green represents positive values for $p_{\mathbf{T}}(\theta)$ and magenta represents negative values.

Polynomial representation of tensors. There is a one-to-one correspondence between 4th-order fully symmetric tensors of dimension n, and degree 4 homogeneous polynomials of n variables, given by

$$p_{\mathbf{T}}(x_1, \ldots, x_n) = T_{i_1,i_2,i_3,i_4} x_{i_1} x_{i_2} x_{i_3} x_{i_4}. \tag{16}$$

Since it is homogeneous, one can restrict this polynomial to the sphere $\mathbb{S}_{n-1}$, since

$$p_{\mathbf{T}}(\mathbf{x}) = \|\mathbf{x}\|^4 \, p_{\mathbf{T}}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right). \tag{17}$$

We can therefore define a function $p_{\mathbf{T}}(\theta)$ such that $p_{\mathbf{T}}(\mathbf{x}) = \|\mathbf{x}\|^4 \, p_{\mathbf{T}}(\theta)$; this function is periodic with period π, since $p_{\mathbf{T}}(\mathbf{x}) = p_{\mathbf{T}}(-\mathbf{x})$. Such a function can be conveniently visualized by taking a unit circle and virtually stretching it according to the value that $p_{\mathbf{T}}$ takes on the circle; formally, one draws the parametric curve given by $r(\theta) = p_{\mathbf{T}}(\theta)$ for $\theta \in [0, 2\pi[$. We show this on fig. 5.

A natural way to encode this polynomial is to decompose it into an orthonormal basis of functions on the sphere $\mathbb{S}_{n-1}$. Considering the two-dimensional case $n = 2$, $p_{\mathbf{T}}(\theta)$ can be decomposed into the Fourier series

$$\begin{aligned} p_{\mathbf{T}}(\theta) = \frac{1}{\sqrt{2\pi}} \, q_0 + \frac{1}{\sqrt{\pi}} \cos(2\theta) \, q_1 + \frac{1}{\sqrt{\pi}} \sin(2\theta) \, q_2 \\ + \frac{1}{\sqrt{\pi}} \cos(4\theta) \, q_3 + \frac{1}{\sqrt{\pi}} \sin(4\theta) \, q_4, \end{aligned} \tag{18}$$

where $\mathbf{q} = (q_0, \ldots, q_4)$ fully describes the polynomial and its corresponding tensor. These basis functions, shown in fig. 6, correspond to two-dimensional

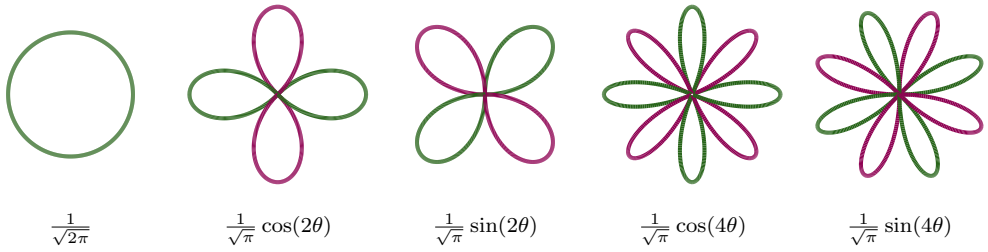


Figure 6: Orthonormal basis of circular harmonics used to represent odeco tensor polynomials. Green represents positive values for $p_{\mathbf{T}}(\theta)$ and magenta represents negative values.

spherical harmonics and are sometimes called *circular harmonics*. The number of five is not surprising since the original tensor has five independent components: T_{1111}, T_{1112}, T_{1122}, T_{1222} and T_{2222}. The change of basis from $\mathbf{T}$ to $\mathbf{q}$ is done through a linear transformation, which can be derived by matching expressions (16) and (18).

Eigenvectors and eigenvalues of higher-order tensors, defined in (12), can be interpreted in the polynomial representation through the following property: $\mathbf{w}$ is an eigenvector of $\mathbf{T}$ with eigenvalue λ if and only if

$$\boldsymbol{\nabla} p_{\mathbf{T}}(\mathbf{w}) = 4\ \lambda\ \mathbf{w}. \tag{19}$$

In other words, the eigenvectors of $\mathbf{T}$ correspond to the stationary points of $p_{\mathbf{T}}(\theta)$.

Algebraic variety in the 2D case. A 2D fourth-order tensor is completely defined by a set of five coefficients $(q_0, \ldots, q_4)$. This tensor corresponds to a frame if the tensor is odeco, i.e., it lies on the algebraic variety defined by the set of quadratic equations (14). In the 2D case ($n = 2$), we can write out this set of equations and change the basis to $\mathbf{q}$, and derive the following set of linearly independent equations that define the variety:

$$\begin{cases} c_1(\mathbf{q}) = & q_0^2 - 18(q_3^2 + q_4^2) = 0 \\ c_2(\mathbf{q}) = & \sqrt{2}q_0q_1 - 6q_1q_3 - 6q_2q_4 = 0 \\ c_3(\mathbf{q}) = & \sqrt{2}q_0q_2 - 6q_1q_4 + 6q_2q_3 = 0 \end{cases} \tag{20}$$

Area of a frame. The optimization we perform on the frame field requires to normalize the integrability condition to make it independent of the size of the frame. We choose to do this normalization with the *area* of the frame,

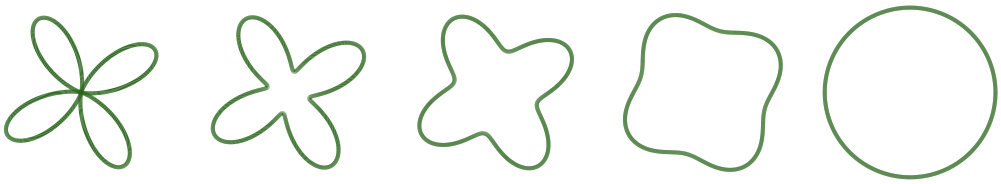


Figure 7: Range of isotropic tensors, the middle one being odeco. All tensors have a set of 4 eigenvectors with the same eigenvalues.

i.e., the product of the sizes $a = \lambda\mu$. This can be interpreted as the local area of the quad elements in the frame's neighborhood. One can show that it can be expressed only in terms of the circular harmonics coefficients $\mathbf{q}$:

$$a(\mathbf{q}) = \lambda\mu = \frac{1}{\pi}\left(\frac{8}{9}q_0^2 - (q_1^2 + q_2^2)\right). \tag{21}$$

We show later in Section 6 how this expression is used to normalize the integrability condition.

Isotropic case. A frame is *isotropic*, i.e., its vectors have equal length $\lambda = \mu$, if its degree 2 coefficients are zero: $q_1 = q_2 = 0$. The algebraic variety reduces to a single quadratic equation $c_1(\mathbf{q}) = 0$. When the degree 2 coefficients are zero, the tensor eigenvectors are always orthogonal, even if the tensor is not odeco. A range of isotropic tensors is illustrated on fig. 7. We see that in the general non-odeco case, this class has an additional degree of freedom depending on how close it is to a sphere. We explain in section 8 how this extra degree of freedom is beneficial as it allows to represent singularities.

5. Integrability condition

Since there is a one-to-one correspondence between frames (as defined in (10)) and orthogonally decomposable (odeco) tensors with positive eigenvalues (as defined in (15)), one can express the integrability condition (7) solely in terms of the odeco tensor coefficients and their spatial derivatives. In this section we show how we derive the closed-form expression for the Lie bracket.

Consider a frame $\mathbf{F} = \left\{\mathbf{R}^k(\mathbf{u}, \mathbf{v}),\ k = 0, 1, 2, 3\right\}$, where $\mathbf{R}$ performs a 90° rotation as defined in (1) and illustrated in fig. 4. Every pair of vectors in $\mathbf{F}$

has the same Lie bracket, which we denote Lie($\mathbf{F}$):

$$\text{Lie}(\mathbf{F}) = [\mathbf{R}^k\mathbf{u}, \mathbf{R}^k\mathbf{v}], \quad k = 0, 1, 2, 3. \tag{22}$$

First we write the Lie bracket $[\mathbf{u}, \mathbf{v}]$ in index notation:

$$[\mathbf{u}, \mathbf{v}]_i = u_\alpha \frac{\partial v_i}{\partial x_\alpha} - v_\alpha \frac{\partial u_i}{\partial x_\alpha}. \tag{23}$$

Let $\mathbf{T}$ denote the odeco tensor corresponding to $\mathbf{F}$. We apply the chain rule to express the spatial derivatives on the tensor coefficients:

$$[\mathbf{u}, \mathbf{v}]_i = u_\alpha \frac{\partial v_i}{\partial T_{j_1 j_2 j_3 j_4}} \frac{\partial T_{j_1 j_2 j_3 j_4}}{\partial x_\alpha} - v_\alpha \frac{\partial u_i}{\partial T_{j_1 j_2 j_3 j_4}} \frac{\partial T_{j_1 j_2 j_3 j_4}}{\partial x_\alpha}. \tag{24}$$

In the remaining of the section we show how (a) the sensitivity terms $\partial v_i / \partial T_j$ and $\partial u_i / \partial T_j$ are derived, and (b) how the Lie bracket is completely expressed in terms of the tensor coefficients $\mathbf{T}$.

Eigenvalue sensitivity for tensors. Recall that $\mathbf{u}, \mathbf{v}$ are eigenvectors of $\mathbf{T}$ as defined by (12). Finding the sensitivity terms $\partial v_i / \partial T_j$ and $\partial u_i / \partial T_j$ can be done by analyzing how the eigenvectors $\mathbf{u}, \mathbf{v}$ change when adding an infinitesimal perturbation $\delta\mathbf{T}$ to the tensor. This analysis is well-known for matrices as the *eigenvalue perturbation problem*, but can be generalized to higher-order tensors, provided they are odeco. We leave the complete derivation in appendix Appendix A and retain the main result: if $(\lambda, \hat{\mathbf{w}})$ is an eigenpair of odeco tensor $\mathbf{T}$, then the sensitivity of $\mathbf{w} = \lambda\hat{\mathbf{w}}$ with respect to a perturbation $\delta\mathbf{T}$ is given by

$$\delta\mathbf{w} = (\delta\mathbf{T})\hat{\mathbf{w}}^3. \tag{25}$$

From this result we find that the partial derivatives are given by

$$\frac{\partial w_i}{\partial T_{j_1 j_2 j_3 j_4}} = \delta_{ij_1}\hat{w}_{j_2}\hat{w}_{j_3}\hat{w}_{j_4}, \tag{26}$$

where δ_{ij} is the Kronecker delta symbol. Note that an eigenvector's sensitivity only depends on itself and not on the other eigenvectors of the odeco tensor.

Lie bracket in terms of tensor representation. We now wish to find an expression for the Lie bracket (24) so that it only depends on the tensor coefficients $\mathbf{T}$ and not on the frame directions $\mathbf{u}, \mathbf{v}$. Plugging in the sensitivity result (26) yields

$$[\mathbf{u}, \mathbf{v}]_i = (\|\mathbf{u}\|\hat{u}_\alpha \hat{v}_{j_2} \hat{v}_{j_3} \hat{v}_{j_4} - \|\mathbf{v}\|\hat{v}_k \hat{u}_{j_2} \hat{u}_{j_3} \hat{u}_{j_4}) \frac{\partial T_{ij_2j_3j_4}}{\partial x_\alpha}. \tag{27}$$

We see that an expression close to the odeco tensor definition (15) starts to appear. The last ingredient is to note that $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ are related by a 90° rotation, which can be written an $\hat{v}_i = \epsilon_{ij}\hat{u}_j$ and $\hat{u}_i = -\epsilon_{ij}\hat{v}_j$, where ϵ_{ij} is the two-dimensional Levi-Civita symbol. Plugging this into the Lie bracket expression and identifying the tensor coefficients yields

$$\mathrm{Lie}(\mathbf{T})_i = \epsilon_{j_2k_2}\epsilon_{j_3k_3}\epsilon_{j_4k_4} T_{\alpha k_2 k_3 k_4} \frac{\partial T_{ij_2j_3j_4}}{\partial x_\alpha}. \tag{28}$$

As the tensor coefficients $\mathbf{T}$ are a linear transformation of the polynomial coefficients $\mathbf{q}$, one can express the Lie bracket in terms of $\mathbf{q}$ only:

$$\mathrm{Lie}(\mathbf{q})_i = C_{ijk\alpha}\ q_k\ \frac{\partial q_j}{\partial x_\alpha}, \tag{29}$$

where the $C_{ijk\alpha}$ are known coefficients. This expression is a key result of this work as it allows to compute an integrability metric that does not involve the frame vectors, but instead a quadratic function of the implicit tensor representation.

6. Optimization

We design an optimization formulation that aims at making the Lie bracket (29) as close to zero as possible, with close-to-odeco tensors (20) having positive sizes ($\lambda, \mu > 0$). To do so we define the following energy functionals on the domain Ω:

$$E_{\mathrm{Lie}}[\mathbf{q}] = \int_\Omega \frac{\|\mathrm{Lie}(\mathbf{q})\|^2}{a(\mathbf{q})^2}\ \mathrm{d}\mathbf{x}\,, \tag{30}$$

$$E_{\mathrm{odeco}}[\mathbf{q}] = \sum_{i=1}^{3} \int_\Omega \frac{c_i(\mathbf{q})^2}{a(\mathbf{q})^2}\ \mathrm{d}\mathbf{x}, \tag{31}$$

$$E_{\mathrm{Dir}}[\mathbf{q}] = \sum_{j=1}^{5} \frac{1}{2} \int_\Omega \|\boldsymbol{\nabla} q_j\|^2\ \mathrm{d}\mathbf{x}\,. \tag{32}$$

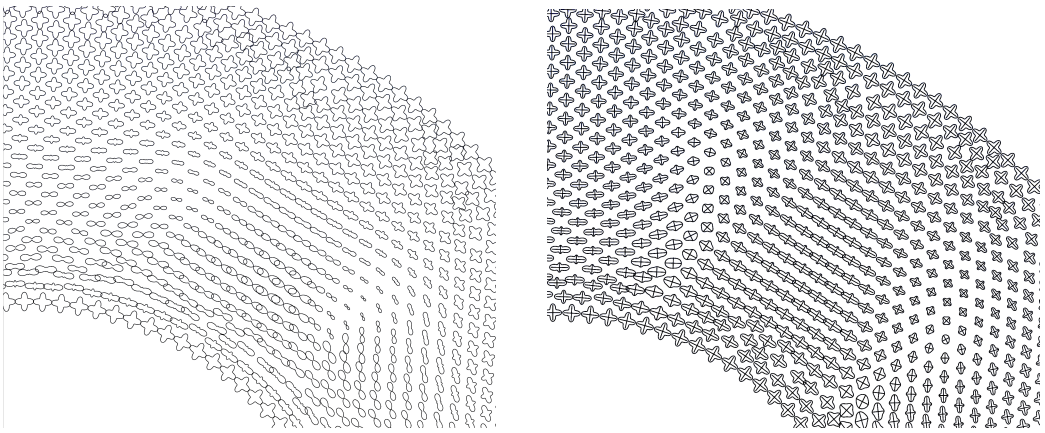


Figure 8: Energy normalization acts as a barrier preventing sizes from going negative. Left: a frame field optimized with the unnormalized Lie bracket produces frames with negative sizes, violating the local injectivity. Right: using the normalized energies forces sizes to remain positive.

The normalized Lie bracket energy E_{Lie} aims at making the frame field integrable, the normalized odeco penalty E_{odeco} at keeping the tensors odeco, and the Dirichlet energy E_{Dir} strives for smoothness of the frame field (Palmer et al. (2020) have shown that, since the L^2 distance on $\mathbf{q}$ corresponds to the L^2 distance between the polynomials $p_{\mathbf{T}}$, the Dirichlet energy is an adequate proxy for the smoothness of the frame field). The area expression $a(\mathbf{q})$ in the denominator of the Lie bracket energy and the odeco energy acts as a normalization: E_{Lie} and E_{odeco} do not depend on the global scale of the frame field, i.e., for a constant factor α, $E_{\text{Lie/odeco}}(\alpha\mathbf{q}) = E_{\text{Lie/odeco}}(\mathbf{q})$. It also acts as a barrier, preventing the frame sizes λ and μ from becoming zero or negative, provided the initial solution has positive sizes; this is illustrated in fig. 8. We assemble these energies in a Ginzburg-Landau fashion, to define a *smoothness energy* and an *integrability energy*:

$$\begin{aligned} E_{\text{smooth}}[\mathbf{q}] &= E_{\text{Dir}}[\mathbf{q}] + \frac{1}{\epsilon^2} E_{\text{odeco}}[\mathbf{q}], \\ E_{\text{integ}}[\mathbf{q}] &= E_{\text{Lie}}[\mathbf{q}] + \frac{1}{\epsilon^2} E_{\text{odeco}}[\mathbf{q}]. \end{aligned} \tag{33}$$

Parameter ϵ has units of length and, like in the Ginzburg-Landau functional, controls the size of the singularity neighborhoods where the frames drift away from the odeco variety. This behavior is illustrated in fig. 9. In our method, minimizing E_{smooth} will serve as a initial solution to the minimization of of

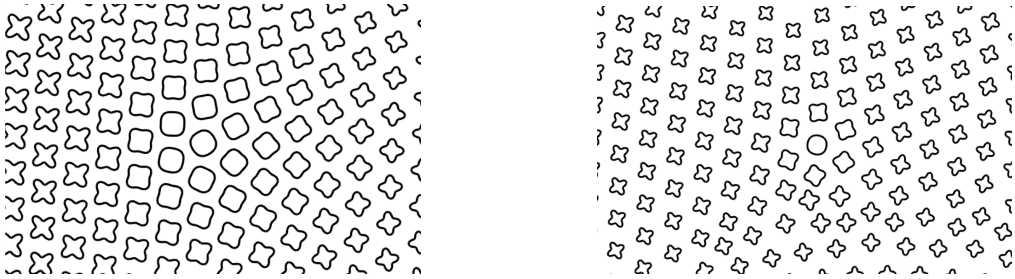


Figure 9: Minimizers of $E_{\text{integ}}[\mathbf{q}]$ producing a valence 5 singularity with different ϵ parameters. Left: a large ϵ produces a diffuse singularity. Right: a small ϵ locates the singularity more precisely. Setting an even smaller ϵ might destroy the singularity if it is not topologically necessary.

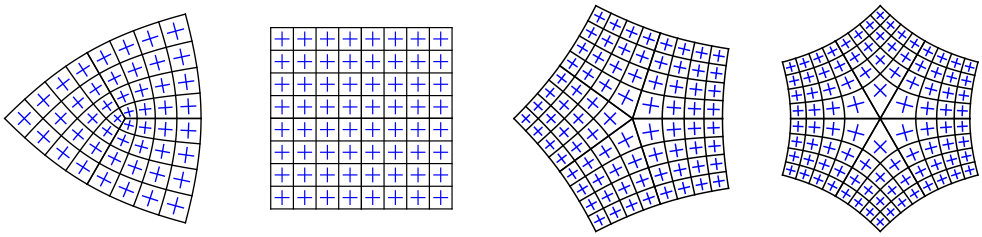
E_{integ}.

Discretization. The frame field is discretized on a standard triangle mesh over Ω. At each node we store the five circular harmonics coefficients $\mathbf{q} = (q_0, \ldots, q_4)$. The coefficients are linearly interpolated on the mesh using continuous P1 triangular finite elements. The energy functionals are then evaluated using a standard 3-point quadrature rule on the triangles. Note that we are using a continuous solution space for the frame field even though the actual field is not defined at the singularities and therefore cannot be represented by our discretization. This justifies our weak enforcement of the odeco constraint through a penalty term: it allows the creation of singularities in a way that does not blow up the integrability energy E_{Lie}, by using tensors which have no preferential directions, i.e., a circle as in fig. 6, left.

Behavior of E_{Lie} at singularities. Since we are using a continuous representation for the frame field, a legitimate concern is whether the Lie bracket energy E_{Lie} remains bounded around singularities. To analyze this we compute an exactly integrable frame field around a singularity of index i; this frame field is given in polar coordinates by

$$\begin{aligned} \mathbf{u}(r,\theta) &= r^i \ (\cos((1-i)\theta)\ \hat{\mathbf{e}}_r - \sin((1-i)\theta)\ \hat{\mathbf{e}}_\theta), \\ \mathbf{v}(r,\theta) &= r^i \ (\sin((1-i)\theta)\ \hat{\mathbf{e}}_r + \cos((1-i)\theta)\ \hat{\mathbf{e}}_\theta). \end{aligned} \tag{34}$$

Notice that the size r^i vanishes at the singularity for $i > 0$ (valence 3 and less) and blows up for $i < 0$ (valence 5 and more). One can check that this frame field indeed has zero Lie bracket. These integrable frame fields



(a) index $= +1/4$ (b) index $= 0$ (c) index $= -1/4$ (d) index $= -1/2$

Figure 10: Parametrization isolines (in black) and their coordinate frames (in blue) defined by eq. (34) for a range of singular indices. Notice how the frame field grows or shrinks according to the singularity index.

along with their corresponding parametrization are illustrated in fig. 10. To evaluate how our Lie bracket energy fairs on a discretized mesh, we represent this frame field on finite element meshes of varying sizes and evaluate both the integrated energy E_{Lie} as well as the maximum value of the integrand; this is shown on fig. 11. We see that even though the value of the integrand blows up when refining the mesh, the energy remains bounded and converges to a fixed value for both singularities. The energy is higher for a valence 5 singularity than a valence 3 since the size is unbounded. This property makes it possible for our solver to introduce singularities whatever the mesh size, if they make the frame field more integrable.

Recovering a frame field. The minimization of (33) provides a field of tensors that is odeco everywhere except in the vicinity of singularities. We recover a frame at every node of the mesh in two steps: first, the tensors are projected onto the odeco variety using the projector of Palmer et al. (2020), using a semidefinite relaxation of the exact projection problem. This operation results in a set of frames of which we only keep the direction vectors $\hat{\mathbf{u}}, \hat{\mathbf{v}}$. In a second step, the sizes of the direction vectors are computed by contracting the tensor 4 times, as if they were eigenvectors:

$$\lambda = \mathbf{T}\hat{\mathbf{u}}^4, \quad \mu = \mathbf{T}\hat{\mathbf{v}}^4. \tag{35}$$

We found that this approach for recovering sizes provided better results than using the sizes given by the projector; this is because the Lie bracket is expressed in terms of the eigenvectors and eigenvalues of the odeco tensor field.

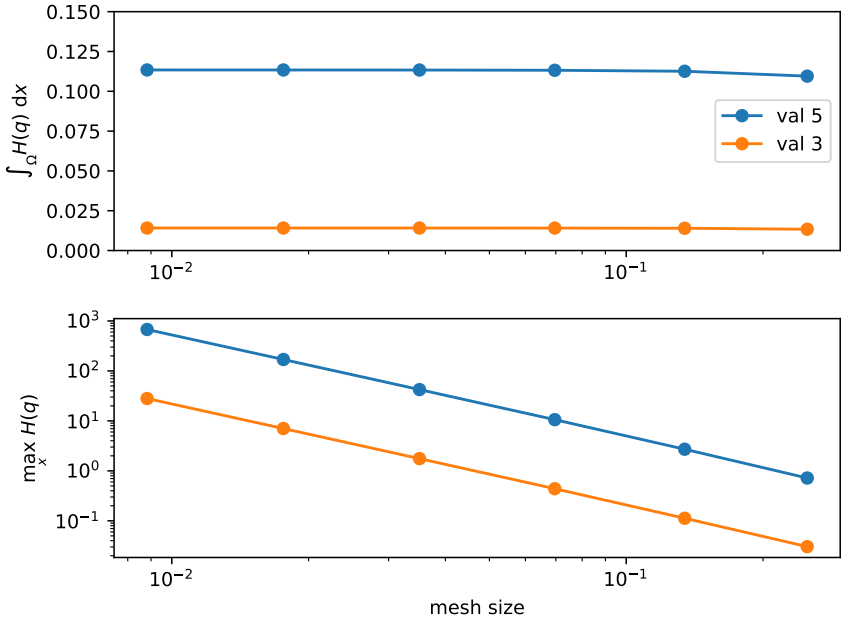


Figure 11: Convergence of the Lie bracket energy E_{Lie} around singularities of valence 5 and 3. $H(\mathbf{q})$ denotes the integrand of E_{Lie}: $H(\mathbf{q}) = \|\text{Lie}(\mathbf{q})\|^2/a(\mathbf{q})^2$.

The optimization procedure to compute an integrable frame field is summarized in Algorithm 1.

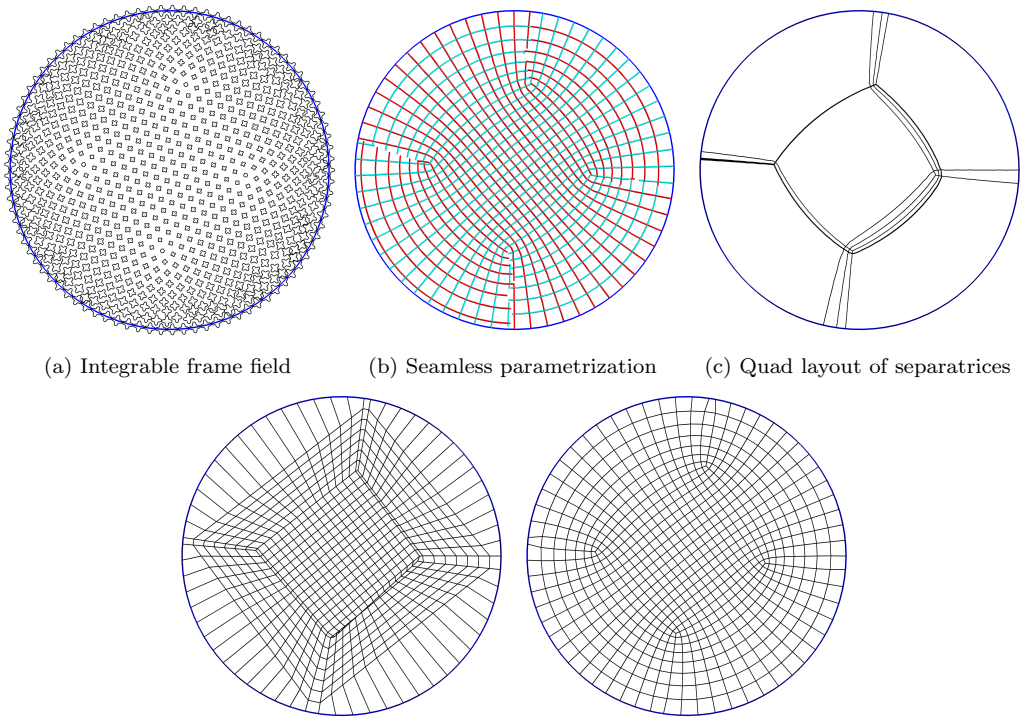
7. Frame field guided parametrization and meshing

Once an integrable frame field is obtained, we extract a fully quadrilateral mesh in two main steps described in following subsections. First, a seamless parametrization is computed; this step is made easy by the integrability property of the frame field. Second, the parametrization is quantized and a quad mesh is extracted and smoothened. An overview of the pipeline is depicted in fig. 12.

7.1. Seamless parametrization

We briefly review how a seamless parametrization is computed from the frame field $\mathbf{F}(\mathbf{x})$ obtained through the optimization of (33); for a more detailed treatment we refer to, e.g., (Bommes et al., 2009).

We compute a field-guided seamless parametrization with the following steps:



(a) Integrable frame field (b) Seamless parametrization (c) Quad layout of separatrices

(d) Raw quad mesh (e) After smoothing

Figure 12: Overview of the parametrization and meshing pipeline. An integrable frame field $\mathbf{q}(\mathbf{x})$ (a) is computed respecting user-prescribed boundary conditions. The frame field is integrated to a seamless parametrization $(u(\mathbf{x}), v(\mathbf{x}))$ (b) that is continuous everywhere but across a cut graph. Separatrices are traced as isolines from singularities to form a quadrilateral layout (c). This quad layout is quantized and a rough, initial quad mesh is generated through transfinite interpolation (d). Finally, a smoothing step is performed to obtain the final quad mesh (e).

Algorithm 1 Integrable frame field solver

Input: A triangle mesh on a planar geometry, with size and/or orientation constraints on boundaries
Output: A smooth integrable frame field

On boundaries, assign frames respecting constraints.
In interior, assign zero frames.
Compute $\mathbf{T}$ and $\mathbf{q}$ using (15).
$\mathbf{q} \leftarrow \arg\min_{\mathbf{q}} E_{\text{smooth}}[\mathbf{q}]$ using L-BFGS. (eq. (33))
$\mathbf{q} \leftarrow \arg\min_{\mathbf{q}} E_{\text{integ}}[\mathbf{q}]$ using L-BFGS. (eq. (33))
for each node **do**
    Recover directions $\hat{\mathbf{u}}, \hat{\mathbf{v}}$ using projection of (Palmer et al., 2020).
    Recover frame sizes λ, μ using (35).
end for

Interpolation on triangles. The tensor field $\mathbf{q}$, defined at the mesh vertices, is interpolated on each triangle $t = (v_1, v_2, v_3)$, and then projected onto the odeco variety $\mathcal{V}$:

$$\mathbf{q}_t = \Pi_{\mathcal{V}}\left(\frac{1}{3}(\mathbf{q}_{v_1} + \mathbf{q}_{v_2} + \mathbf{q}_{v_3})\right).$$

Singularity detection. Singular vertices are identified by calculating the turning number of the frames around a vertex' 1-ring $N_1(v)$; this number is $+1/4$ or $-1/4$ for a singularity of valence 3 and 5, respectively.

Cut graph. The triangle mesh is "cut open" such that a closed curve cannot turn around a singularity without crossing the cut graph.

Frame vectors assignment. Thanks to the cut graph, two vectors $\mathbf{u}$ and $\mathbf{v}$ can be chosen from each frame to form two piecewise constant vector fields $\mathbf{u}(\mathbf{x})$ and $\mathbf{v}(\mathbf{x})$. Each cut in the cut graph has a *matching* that indicates how to match frame vectors across the cut.

Integration. The frame vector fields $\mathbf{u}(\mathbf{x})$ and $\mathbf{v}(\mathbf{x})$ are integrated to piecewise linear scalar potentials $u(\mathbf{x})$ and $v(\mathbf{x})$ by solving the least-squares

problem

$$\min_{\substack{u(\mathbf{x}) \\ v(\mathbf{x})}} \int_\Omega \left(\left\| \boldsymbol{\nabla} u - \frac{\mathbf{u}}{\|\mathbf{u}\|^2} \right\|^2 + \left\| \boldsymbol{\nabla} v - \frac{\mathbf{v}}{\|\mathbf{v}\|^2} \right\|^2 \right) \, \mathrm{d}\mathbf{x} \,. \tag{36}$$

During this optimization, the parametrization is constrained to be seamless, meaning that (i) on the boundary, one of the potentials u or v is constant, and (ii) along each cut, the matching potential gradients are equal (which amounts to impose (3)). These seamlessness constraints can be written as linear constraint on the unknowns which are straightforward to impose.

We call the residual of the optimization objective (36) the *integration error* e_{integ}; it quantifies how integrable the frame field was, and in general how well the parametrization aligns to the prescribed frame field. Achieving a small integration error means we can adequately control the size and orientation constraints in the seamless parametrization process.

7.2. Quad mesh extraction

A quadrilateral mesh cannot be directly extracted from a seamless parametrization, due to the integer parametric isolines not agreeing on the cut lines and singularities not lying at integer locations; see fig. 12b. An "integer rounding" of sorts is necessary to eliminate these discontinuities. As reviewed in section 2, this usually requires a tedious and costly integer optimization process. Being in the planar setting, and our frame fields being free of limit cycles, all separatrices (parametric isolines traced from singularities) have a finite length. This means we can forgo any T-mesh, and work directly on a quadrilateral layout (sometimes called the *base complex*). We designed a robust greedy quantization strategy that is free of any costly integer optimization. We outline this strategy below, as well as the post-processing needed to extract and smooth the final quad mesh.

Separatrix tracing. For each singular vertex $\mathbf{p}$, we trace and propagate 3 or 5 separatrices, depending on the vertex' singularity index. Let $u_\mathbf{p}, v_\mathbf{p}$ be the (unique) pair of scalar potentials at the singularity. For each triangle in the 1-ring $N_1(v)$, we look if either isoline $u(\mathbf{x}) = u_\mathbf{p}$ and/or $v(\mathbf{x}) = v_\mathbf{p}$ traverses the triangle. The isolines are then propagated in the mesh by marching on triangles, taking special care when

crossing the cut graph to apply the adequate matching. The propagation stops when hitting a boundary or another singular vertex, which is guaranteed to happen as long as the parametrization is free of limit cycles.

In order to comply with user-described features, separatrices are also traced from concave corners (2 separatrices) and feature line endpoints (3 separatrices).

Quantization. The traced separatrices partition the domain into quadrilateral patches (fig. 12c). Quantization consists in assigning an integer length to every edge in the layout. First, we find the *quad strips* of the layout; a quad strip is a sequence of consecutive quad patches, and can either be closed or start and end on a boundary. Each patch belongs to exactly two strips, one for each orthogonal direction. Every quad strip s has an initial (non-integer) parametric width l_s, and the quantization problem consists in choosing an integer width for every strip (which can be zero). At first, rounding every l_s to the nearest integer seems evident, however this usually leads to a poor quantization: the quad layout usually contains a large number of thin ($l_s < 0.5$) quad strips, and collapsing them causes the total parametric area to decrease, leading to poorly shaped quads in the final mesh. Instead, we designed a novel greedy strategy that aims to preserve the total parametric area during the quantization process. At a high level, each edge e has a modified parametric length $l(e)$, and when quantizing a quad strip, the lost or gained parametric area is transferred to the neighboring left and right patches along the strip. This method is detailed in algorithm 2. Experimentally, we found that using this strategy over the naive rounding reduces the lost parametric area from 30-40 % down to under 5 %, providing a high-fidelity quantization without any costly integer optimization.

Collapse of quadrilateral strips. As some quad strips have been collapsed to zero integer width, some vertices of the quadrilateral layout have to be merged before proceeding to the meshing step. Vertices that are joined through a collapsed edge are grouped into *vertex clusters*, and a representative vertex is chosen. If a singularity or corner is present in the cluster, it is set as the representative vertex; this allows to preserve the singularity locations from the frame field.

Algorithm 2 Greedy quantization

Input: A quadrilateral layout made of n *quad strips*; each strip s has a (non-integer) target parametric length l_s^{target}.
Output: A quantized quadrilateral layout; each strip s has an integer length l_s

Set initial parametric length $l(e) \leftarrow l_s^{\text{target}}$
for n times **do**
  for each non-quantized quad strip s **do**
    $q_s \leftarrow \arg\min_{q\in\mathbb{Z}} C(q) = \sum_{(e_1,e_2)\in s} |l(e_1) - q|\ l(e_2)$
  end for
  Choose strip s^* with minimal $C(q_{s^*})$ and quantize it: $l_{s^*} \leftarrow q_{s^*}$
  for each patch (e_1, e_2) in s^* **do**
    Transfer parametric area $(l(e_1) - l(s^*))\ l(e_2)$ to neighboring patches:
    $l(e_{\text{left}}) \leftarrow l(e_{\text{left}}) + \frac{l(e_1)-l(s^*)}{2}$
    $l(e_{\text{right}}) \leftarrow l(e_{\text{right}}) + \frac{l(e_1)-l(s^*)}{2}$
  end for
end for

Quadrilateral meshing and smoothing. One can finally mesh the patches with a positive integer area. For interior (non-feature) patch edges, we have lost the exact geometric representation because of the edge collapses, hence we simply draw straight lines; boundary and feature edges, however, are exactly preserved. The patches are meshed at the adequate quantization using a transfinite interpolation (Gordon and Hall, 1973) (fig. 12d). A final smoothing is performed on the mesh using the untangler of Garanzha et al. (2021) (fig. 12e). This algorithm takes as input a target shape for each quad element; we feed as a target the frame field evaluated on the original patch, before the collapses are performed.

Note that the quadrilateral mesh we extract does not necessarily have the coarsest quad layout that is achievable given the singularity configuration. Indeed, integrability of the frame field does not guarantee that the singularities connect through their separatrices. A user wishing to obtain a coarse multi-block layout can resort to, e.g., the method of Couplet et al. (2021), who perform a quad layout simplification directly on an input quad mesh.

8. Results and Discussion

To demonstrate the validity of our method, we first compare smooth frame fields against integrable frame fields for a set of geometries where the frame orientations and sizes are fixed on the boundaries. For each frame field we extract a quad mesh as described in the previous section, and measure the integration error committed when computing the seamless parametrization.

Parameter settings. For each test case we use a fixed odeco parameter ϵ that has the order of magnitude of the local mesh size. As described in algorithm 1, we consecutively solve for a smooth frame field using E_{smooth}, then an integrable one using E_{integ}. To solve each optimization problem we use a standard quasi-Newton L-BFGS solver. The integral calculations are done in parallel over the triangles. Computations were done on a MacBook Pro equipped with an M3 Pro using 6 threads for the parallelization. Information on the input and output mesh sizes as well as the run times of the different steps are detailed in table 1.

Smooth vs integrable frames. The results of this comparison are illustrated in fig. 13. On average, the integrable frame field achieves an integration error that is 1 or 2 orders of magnitude smaller than the smooth frame field. The smooth frame fields do not have the correct singularity configuration to perform the required size transition. This causes the corresponding meshes to have poor quality and completely violate the user-prescribed sizing constraints, even though the frames are fully isotropic. On the other hand, the integrable frame fields have a set of singularities that ensure the size transitions, and this is reflected in the meshes where the elements remain isotropic and have the correct sizes at the boundaries.

Singularities. These results also exhibit how the field of tensors behaves in the vicinity of singularities. As expected, the tensors become non-odeco as they narrow the singularities; this allows the solver to represent singularities in a way that is not too costly in terms of integrability energy. However it also brings a limitation: as the tensors are not odeco, the assumptions needed for the integrability energy fail, causing the frame field to not be exactly integrable close to singularities. In practice, one needs to tune parameter ϵ to achieve an appropriate trade-off between singularity cost and integrability.

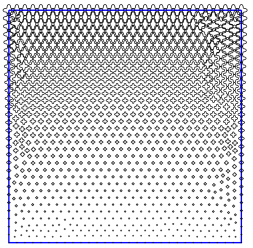
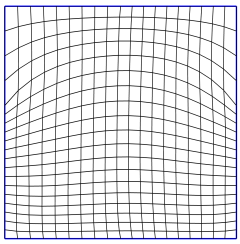
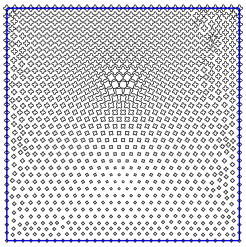
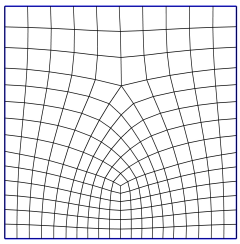
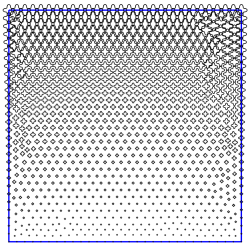
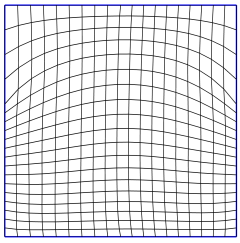
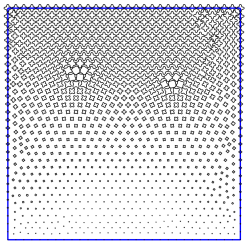
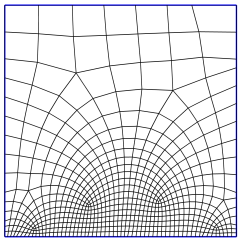
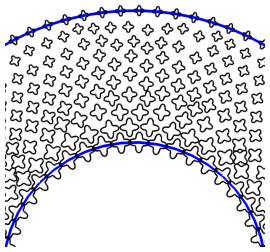
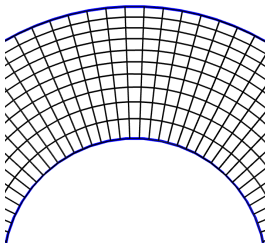
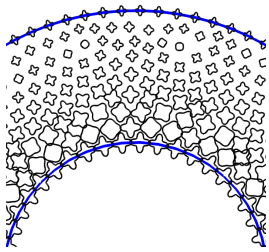
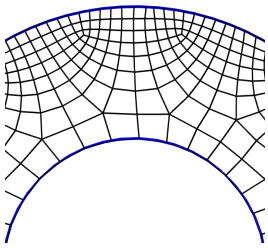
Smooth frame field	**Quad mesh.**	**Integ. frame field**	**Quad mesh**
(a) Size is 1 at bottom, 2 at top, and varies linearly on the sides.			
			
integ. error: 1.44×10^{-2}		integ. error: 7.96×10^{-4}	
(b) Size is 1 at bottom, 10 at top, and varies linearly on the sides.			
			
integ. error: 3.12×10^{-1}		integ. error: 3.31×10^{-3}	
(c) Size is 2 on the inside and 1 on the outside.			
			
integ. error: 6.03×10^{-2}		integ. error: 1.40×10^{-2}	

Figure 13: Comparison of smooth frame fields and integrable frame fields when sizes are imposed on boundaries. For each frame field the corresponding quad mesh is illustrated. All frame fields are isotropic.

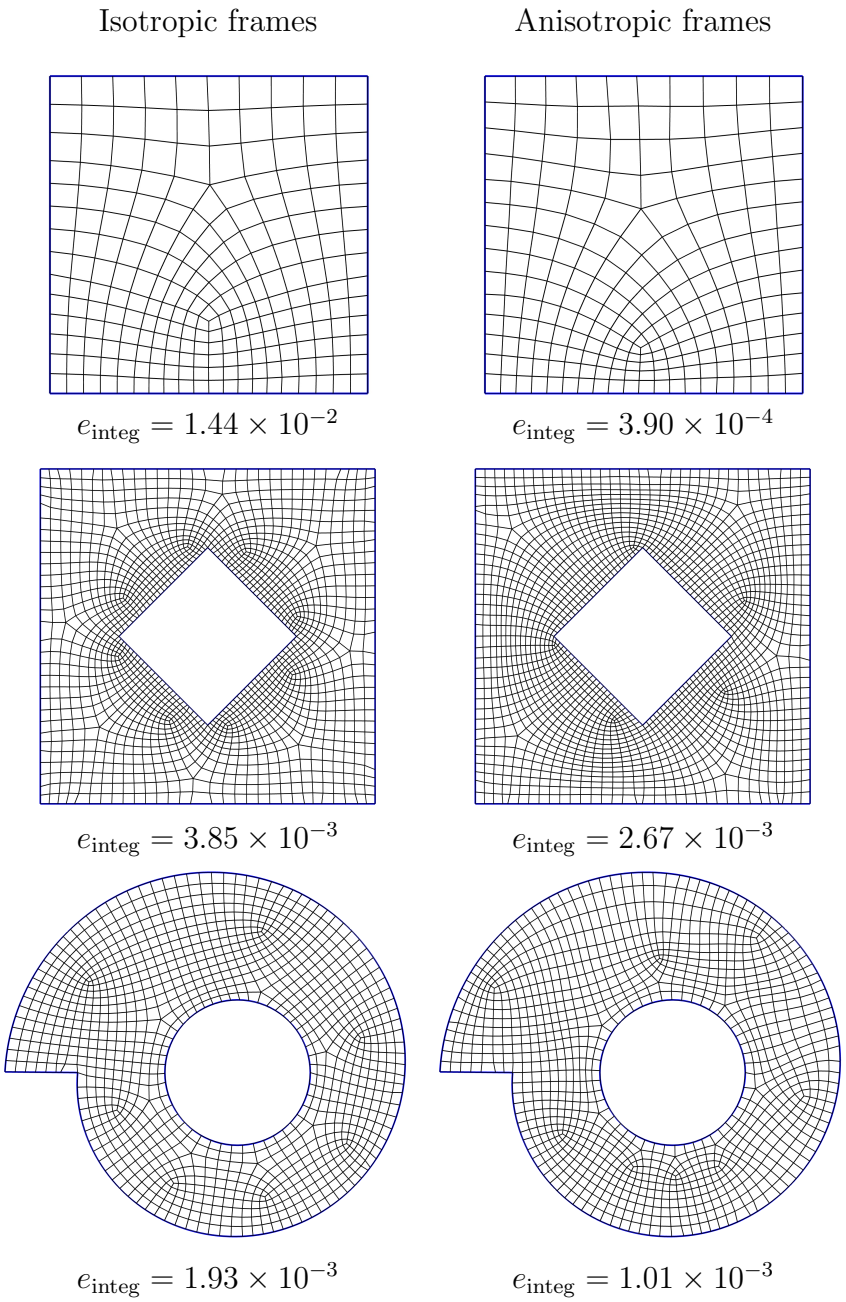


Figure 14: Quadrilateral meshes computed for various models in both isotropic and anisotropic settings. e_{integ} denotes the integration error. Boundary size conditions are: (top) 1 at bottom and 2 at top, (middle) 1 on inner boundary and 2 on outer boundary, (bottom) 1 on inner and outer boundaries.

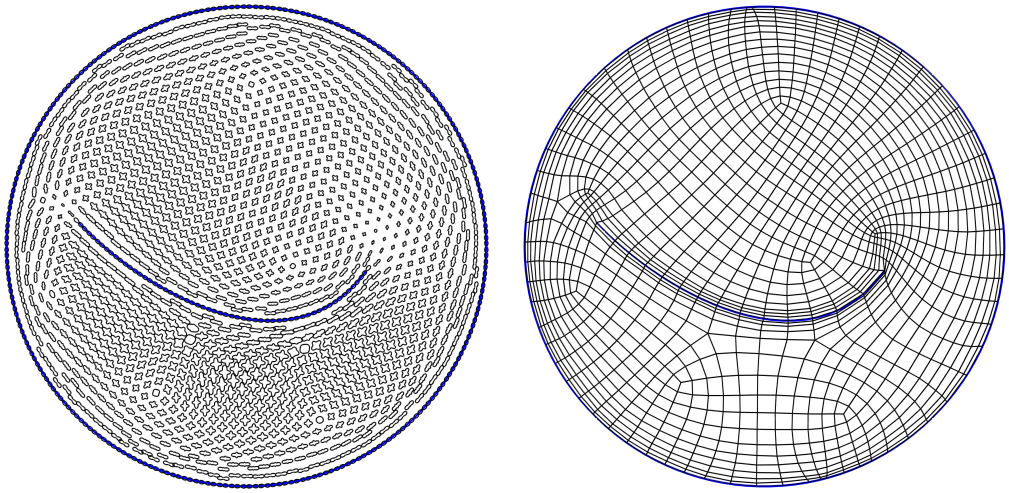


Figure 15: Frame field (left) and quad mesh (right) complying with a user-provided feature line. On the boundary and feature line, a ratio of 1:10 is prescribed between the normal and tangential directions.

Isotropic vs anisotropic frames. Figure 14 compares the isotropic and anisotropic frame fields obtained with our solvers on additional examples. The anisotropic solver produces a very distinct singularity configuration compared to the isotropic solver, and achieves a much smaller integration error thanks to the additional degrees of freedom. Figure 15 presents a testcase with highly anisotropic sizing prescription. Elements lying on boundaries and internal features must have an aspect ratio of 1:10, respectively in the directions normal and tangent to the feature curve. We see that the odeco representation is able to capture this high degree of anisotropy, and the integrability criterion successfully inserts the correct singularities to make the size transition happen.

Limit cycles. Conventional frame field-driven quad meshing methods fail due to the presence of *limit cycles* in the frame field, which are the consequence of a non-meshable topology. A standard test case where a limit cycle appears is the *nautilus* model, for which we show our results on fig. 14, bottom. Our method successfully produces singularity configurations the are free of limit cycles, and a valid quad mesh can be extracted.

Existence of integrable frame fields. Given a set of boundary conditions, an exactly integrable frame field does not exist in general; in our case, by im-

Model	Size	#T	Run times (s) FF	P+MB+Q	Smooth	#Q
3-comp. wing (fig. 1)	iso	6.9k	720	< 1	208	9630
Disk (fig. 12)	iso	1.6k	18	< 1	< 1	704
Square 1-2 (fig. 13a)	iso	2.1k	9	< 1	< 1	196
Square 1-10 (fig. 13b)	iso	2.1k	242	< 1	6	652
Annulus (fig. 13c)	iso	1.3k	20	< 1	3	831
Square 1-2 (fig. 14)	aniso	2.1k	298	< 1	< 1	215
Tilted squares (fig. 14)	iso	1.9k	36	< 1	12	1704
Tilted squares (fig. 14)	aniso	1.9k	237	< 1	11	1788
Nautilus (fig. 14)	iso	7.2k	157	< 1	38	2152
Nautilus (fig. 14)	aniso	7.2k	1397	< 1	6	
Disk + feature (fig. 15)	aniso	3.2k	997	< 1	40	1209

Table 1: Statistics and run times for the various models in previous figures. #T denotes the number of triangle elements in the input mesh, #Q the number of quad elements in the output mesh. Run times are detailed for the frame field optimization (FF), parametrization (P), multi-block decomposition (MB), quantization (Q) and smoothing.

posing sizes strongly on the boundaries, an integrable isotropic frame field very often does not exist. We refer to (Jezdimirović et al., 2022) and the Abel-Jacobi framework for a complete treatment of these aspects. The non-existence of an integrable frame field can explain why a Lie bracket of zero is not achievable. Moreover, the smaller integration error achieved by allowing anisotropic frames (as illustrated in fig. 14) shows that the boundary conditions truly obstruct the integrability. This can be alleviated either by relaxing the boundary conditions or the isotropy constraint.

Source code. Our implementation of the complete meshing pipeline will be made publicly available upon paper acceptance.

9. Conclusion and Future Work

In this work, we have shown how to leverage the theory of orthogonally decomposable tensors to produce integrable frame fields. We have studied the tensor eigenvalue perturbation problem and found simple expressions for the eigenvalue sensitivity of tensors. This result enables us to write a frame field's Lie bracket (and thus, the integrability optimization problem) solely in terms of its tensor representation. The integrable frame field can be

integrated to a seamless parametrization of which we have full control over size and orientation through the frame field optimization; this is a feature that is usually overlooked in field-based meshing methods. Finally, a quad mesh is extracted using a quantization approach that is free of any T-mesh or integer optimization. We believe this contribution of both of theoretical and practical importance, as it is a first step towards hex-meshable frame fields, and in the planar case provides a high-quality mesher with user control over the mesh sizing.

Integrable non-odeco fields. An important limitation of our approach is that the tensor field is assumed to be odeco for the Lie bracket expression to be valid. This causes the frame field to not be exactly integrable near singularities. A possible solution for this is to remove the odeco assumption when writing out the eigenvalue perturbation problem. This makes it possible to write a Lie bracket that remains valid even if the tensor field is non-odeco (at least in the isotropic case). The price to pay is that the integrability condition becomes a more complex, rational expression of the tensor coefficients. We leave the investigation of this approach for future work.

The surface case. In its current state, the method does not take into account curvature in the geometry, and hence only works in the planar case. The method can still be used on an open parametric surface that is nearly-isometric, by meshing the parametric space and pulling the mesh back onto the geometry. Future work will investigate how to extend the methodology to general surface geometries with curvature.

The 3D case. The major advantage of our methodology is that it offers a natural extension to compute 3D integrable frame fields; indeed, the theory of odeco tensors, as well as the results on eigenvalue sensitivity, remain valid in arbitrary dimensions. The only obstacle remaining is to express the integrability condition in terms of the algebraic frame representation We are confident that a solution can be found in future work.

Acknowledgments

This work is supported by the Belgian American Educational Foundation, the Francqui Foundation, Wallonie-Bruxelles International, and the European Research Council (grant no. 101 071 255).

Appendix A. Eigenvalue sensitivity for odeco tensors

Let $\mathbf{T}$ be a fully symmetric fourth-order orthogonally decomposable tensor of dimension n, i.e., $\mathbf{T} = \sum_{i=1}^{n} \lambda_i \hat{\mathbf{v}}_i^4$, or, in index notation,

$$T_{j_1 j_2 j_3 j_4} = \lambda_i \hat{v}_{i,j_1} \hat{v}_{i,j_2} \hat{v}_{i,j_3} \hat{v}_{i,j_4}, \tag{A.1}$$

with orthogonal unit vectors $\hat{\mathbf{v}}_i$. Consider the contraction of $\mathbf{T}$ with one of its eigenvectors, $\hat{\mathbf{v}}_k$:

$$\begin{aligned} T_{j_1 j_2 j_3 j_4} \hat{v}_{k,j_4} &= \lambda_i \hat{v}_{i,j_1} \hat{v}_{i,j_2} \hat{v}_{i,j_3} \hat{v}_{i,j_4} \hat{v}_{k,j_4}, \\ &= \lambda_i \hat{v}_{i,j_1} \hat{v}_{i,j_2} \hat{v}_{i,j_3} \delta_{ik} \\ &= \lambda_k \hat{v}_{k,j_1} \hat{v}_{k,j_2} \hat{v}_{k,j_3}, \end{aligned} \tag{A.2}$$

or, written compactly, $\mathbf{T}\hat{\mathbf{v}}_k = \lambda_k \hat{\mathbf{v}}_k^3$. Contracting again and following the same procedure we find that $\mathbf{T}\hat{\mathbf{v}}_k^2 = \lambda_k \hat{\mathbf{v}}_k^2$ and $\mathbf{T}\hat{\mathbf{v}}_k^3 = \lambda_k \hat{\mathbf{v}}_k$, the latter being the definition of a tensor eigenvector. Consider now a specific eigenpair $(\lambda, \mathbf{w})$, with $\mathbf{w}$ having unit norm. We wish to express the variation of the eigenpair $(\delta\lambda, \delta\mathbf{w})$ given some perturbation of the tensor $\delta\mathbf{T}$. We write the remainder of the proof in compact notation for brevity but the same steps can be performed in index notation. Since the eigenvectors have unit norm, any eigenvector is orthogonal to its variation:

$$(\mathbf{w} + \delta\mathbf{w}) \cdot (\mathbf{w} + \delta\mathbf{w}) = 1 \;\Rightarrow\; \mathbf{w} \cdot \delta\mathbf{w} = 0, \tag{A.3}$$

neglecting the higher-order terms $(\delta\mathbf{w} \cdot \delta\mathbf{w})$. Writing the eigenvector definition $\mathbf{T}\mathbf{w}^3 = \lambda\mathbf{w}$ for the new eigenpair, we find

$$\begin{aligned} (\mathbf{T} + \delta\mathbf{T})(\mathbf{w} + \delta\mathbf{w})^3 &= \lambda\mathbf{w} + \delta(\lambda\mathbf{w}) \\ \cancel{\mathbf{T}\mathbf{w}^3} + 3\mathbf{T}\mathbf{w}^2\delta\mathbf{w} + \delta\mathbf{T}\mathbf{w}^3 &= \cancel{\lambda\mathbf{w}} + \delta(\lambda\mathbf{w}) \\ \cancel{3\lambda\mathbf{w}^2\delta\mathbf{w}} + \delta\mathbf{T}\mathbf{w}^3 &= \delta(\lambda\mathbf{w}) \end{aligned} \tag{A.4}$$

We have used, respectively, the eigenvector definition, the contraction property $\mathbf{T}\mathbf{w}^2 = \lambda\mathbf{w}^2$ and the orthogonality property $\mathbf{w} \cdot \delta\mathbf{w} = 0$. This gives the desired sensitivity expression.

References

Beaufort, P.A., Lambrechts, J., Henrotte, F., Geuzaine, C., Remacle, J.F., 2017. Computing cross fields A PDE approach based on the Ginzburg-Landau theory. Procedia Engineering 203, 219–231. doi:10.1016/j.proeng.2017.09.799.

Bommes, D., Zimmer, H., Kobbelt, L., 2009. Mixed-integer quadrangulation. ACM Transactions on Graphics 28, 77:1–77:10. doi:10.1145/1531326.1531383.

Campen, M., Bommes, D., Kobbelt, L., 2015. Quantized global parametrization. ACM Transactions on Graphics 34, 1–12. doi:10.1145/2816795.2818140.

Chemin, A., Henrotte, F., Remacle, J.F., Schaftingen, J.V., 2019. Representing Three-Dimensional Cross Fields Using Fourth Order Tensors, in: Roca, X., Loseille, A. (Eds.), 27th International Meshing Roundtable. Springer International Publishing, Cham. volume 127, pp. 89–108. doi:10.1007/978-3-030-13992-6_6.

Couplet, M., Reberol, M., Remacle, J.F., 2021. Generation of High-Order Coarse Quad Meshes on CAD Models via Integer Linear Programming, in: AIAA Aviation Forum. doi:10.2514/6.2021-2991.

Crane, K., Desbrun, M., Schröder, P., 2010. Trivial Connections on Discrete Surfaces. Computer Graphics Forum 29, 1525–1533. doi:10.1111/j.1467-8659.2010.01761.x.

Diamanti, O., Vaxman, A., Panozzo, D., Sorkine-Hornung, O., 2014. Designing N -PolyVector Fields with Complex Polynomials. Computer Graphics Forum 33, 1–11. doi:10.1111/cgf.12426.

Diamanti, O., Vaxman, A., Panozzo, D., Sorkine-Hornung, O., 2015. Integrable PolyVector fields. ACM Transactions on Graphics 34, 1–12. doi:10.1145/2766906.

Garanzha, V., Kaporin, I., Kudryavtseva, L., Protais, F., Ray, N., Sokolov, D., 2021. Foldover-free maps in 50 lines of code. ACM Transactions on Graphics 40, 1–16. doi:10.1145/3450626.3459847.

Gordon, W.J., Hall, C.A., 1973. Construction of curvilinear co-ordinate systems and applications to mesh generation. International Journal for Numerical Methods in Engineering 7, 461–477. doi:10.1002/nme.1620070405.

Huang, J., Tong, Y., Wei, H., Bao, H., 2011. Boundary aligned smooth 3D cross-frame field. ACM Transactions on Graphics 30, 1–8. doi:10.1145/2070781.2024177.

Jezdimirović, J., Chemin, A., Remacle, J.F., 2022. Integrable cross-field generation based on imposed singularity configuration – the 2D manifold case –. arXiv:2210.02563.

Kälberer, F., Nieser, M., Polthier, K., 2007. QuadCover - Surface Parameterization using Branched Coverings. Computer Graphics Forum 26, 375–384. doi:10.1111/j.1467-8659.2007.01060.x.

Knöppel, F., Crane, K., Pinkall, U., Schröder, P., 2013. Globally optimal direction fields. ACM Transactions on Graphics 32, 1–10. doi:10.1145/2461912.2462005.

Kowalski, N., Ledoux, F., Frey, P., 2013. A PDE Based Approach to Multidomain Partitioning and Quadrilateral Meshing, in: Jiao, X., Weill, J.C. (Eds.), Proceedings of the 21st International Meshing Roundtable. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 137–154. doi:10.1007/978-3-642-33573-0_9.

Lyon, M., Campen, M., Kobbelt, L., 2021. Quad Layouts via Constrained T-Mesh Quantization. Computer Graphics Forum 40, 305–314. doi:10.1111/cgf.142634.

Myles, A., Pietroni, N., Zorin, D., 2014. Robust field-aligned global parametrization. ACM Transactions on Graphics 33, 1–14. doi:10.1145/2601097.2601154.

Palacios, J., Zhang, E., 2007. Rotational symmetry field design on surfaces. ACM Transactions on Graphics 26, 55. doi:10.1145/1276377.1276446.

Palmer, D., Bommes, D., Solomon, J., 2020. Algebraic Representations for Volumetric Frame Fields. ACM Transactions on Graphics 39, 1–17. doi:10.1145/3366786.

Ray, N., Li, W.C., Lévy, B., Sheffer, A., Alliez, P., 2006. Periodic global parameterization. ACM Transactions on Graphics 25, 1460–1485. doi:10.1145/1183287.1183297.

Ray, N., Vallet, B., Li, W.C., Lévy, B., 2008. N-symmetry direction field design. ACM Transactions on Graphics 27, 1–13. doi:10.1145/1356682.1356683.

Reberol, M., Georgiadis, C., Remacle, J.F., 2021. Quasi-structured quadrilateral meshing in Gmsh – a robust pipeline for complex CAD models. arXiv:2103.04652 [cs] arXiv:2103.04652.

Robeva, E., 2016. Orthogonal Decomposition of Symmetric Tensors. SIAM Journal on Matrix Analysis and Applications 37, 86–102. doi:10.1137/140989340.

Sageman-Furnas, A.O., Chern, A., Ben-Chen, M., Vaxman, A., 2019. Chebyshev nets from commuting PolyVector fields. ACM Transactions on Graphics 38, 1–16. doi:10.1145/3355089.3356564.

Vaxman, A., Campen, M., Diamanti, O., Panozzo, D., Bommes, D., Hildebrandt, K., Ben-Chen, M., 2016. Directional Field Synthesis, Design, and Processing. Computer Graphics Forum 35, 545–572. doi:10.1111/cgf.12864.

Vekhter, J., Chen, Z., Vouga, E., 2025. Mint: Discretely Integrable Moments for Symmetric Frame Fields .

Viertel, R., Osting, B., 2019. An Approach to Quad Meshing Based on Harmonic Cross-Valued Maps and the Ginzburg–Landau Theory. SIAM Journal on Scientific Computing 41, A452–A479. doi:10.1137/17M1142703.

Zhang, P., Vekhter, J., Chien, E., Bommes, D., Vouga, E., Solomon, J., 2020. Octahedral Frames for Feature-Aligned Cross Fields. ACM Transactions on Graphics 39, 1–13. doi:10.1145/3374209.