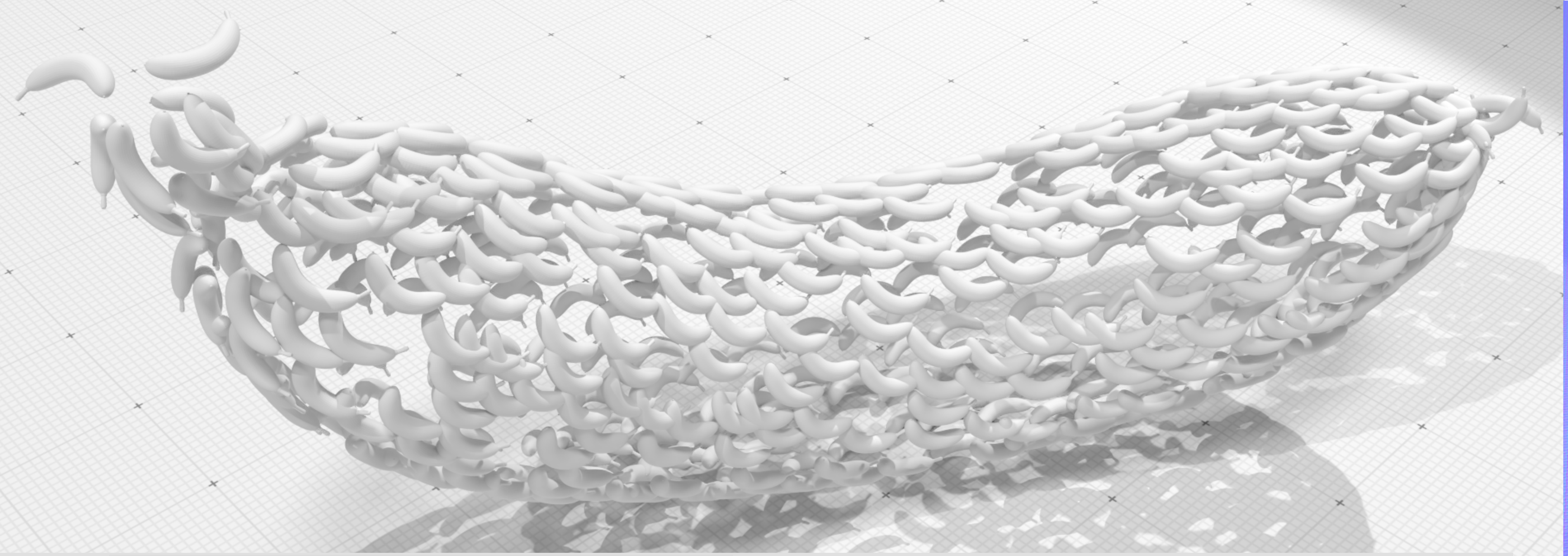
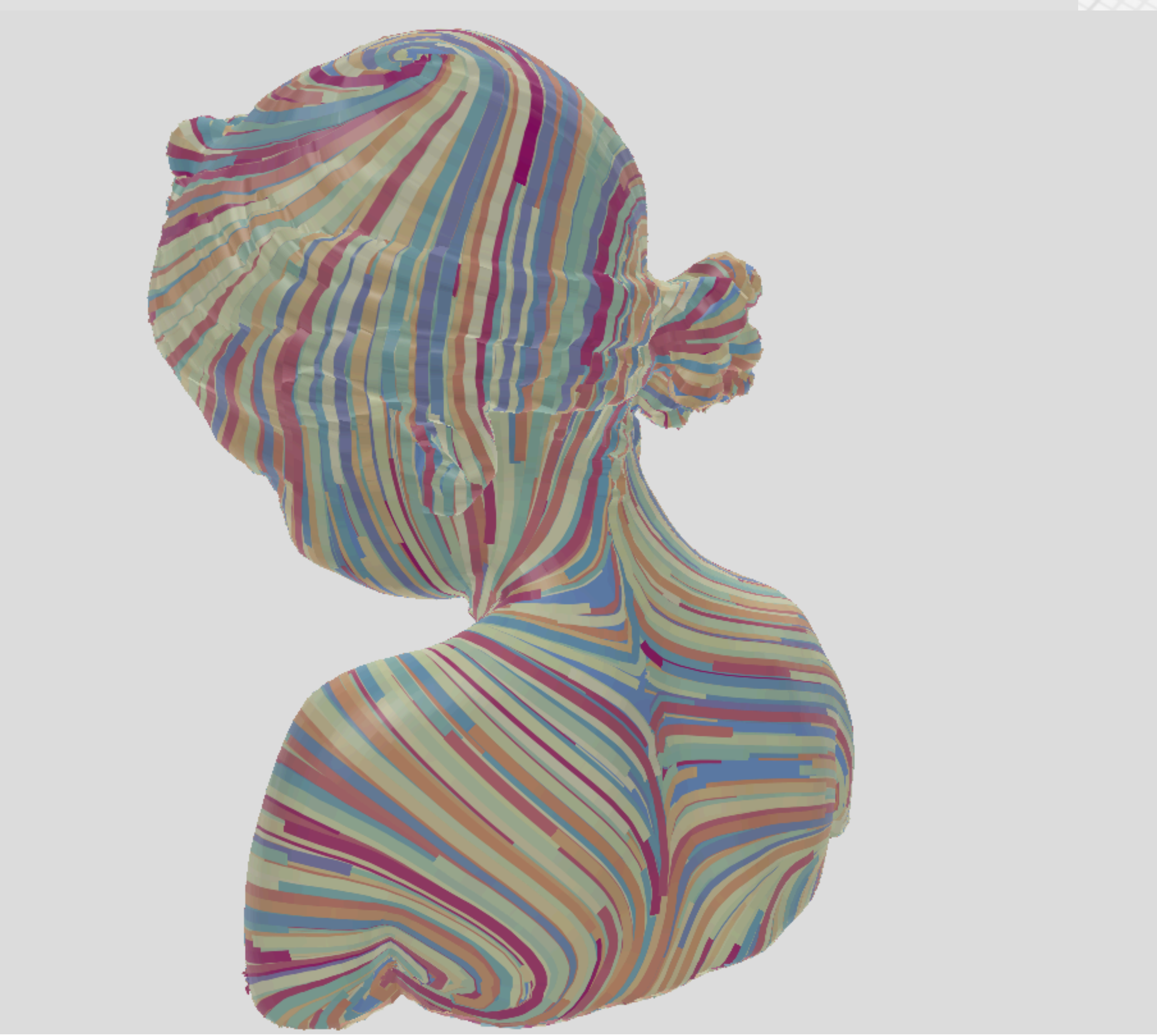
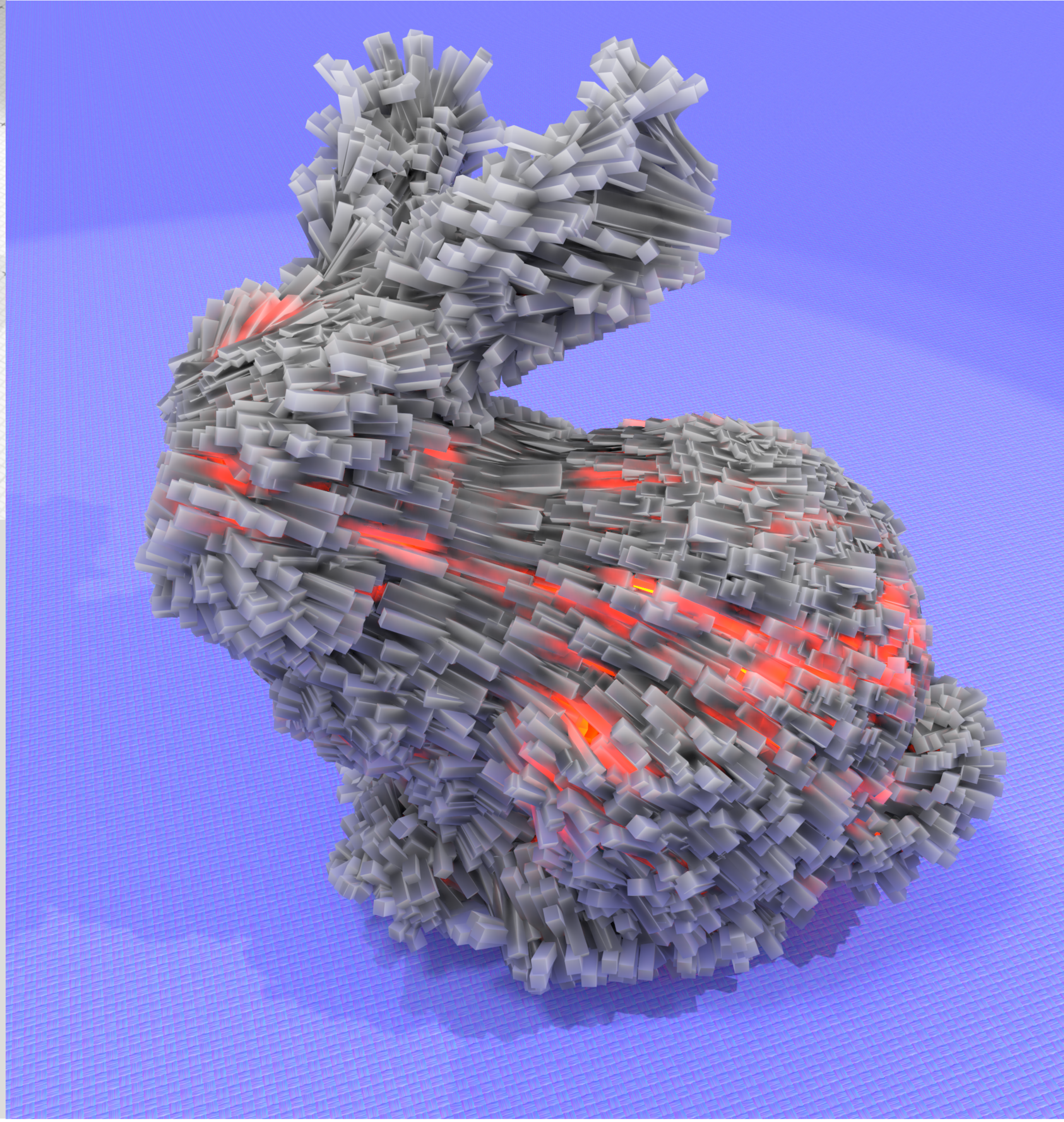




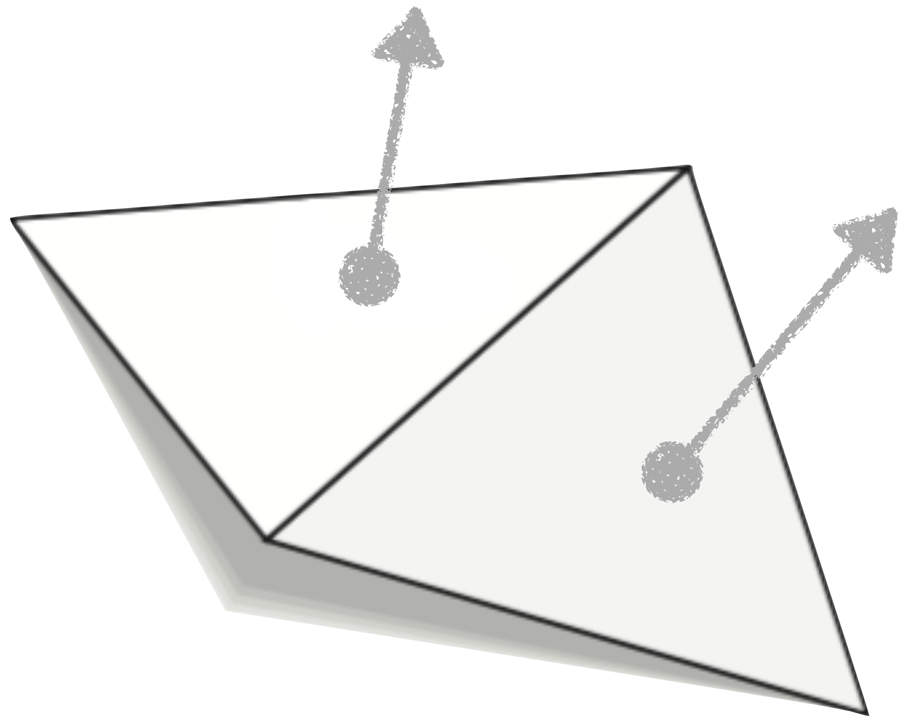


Vector Fields in Computer Graphics

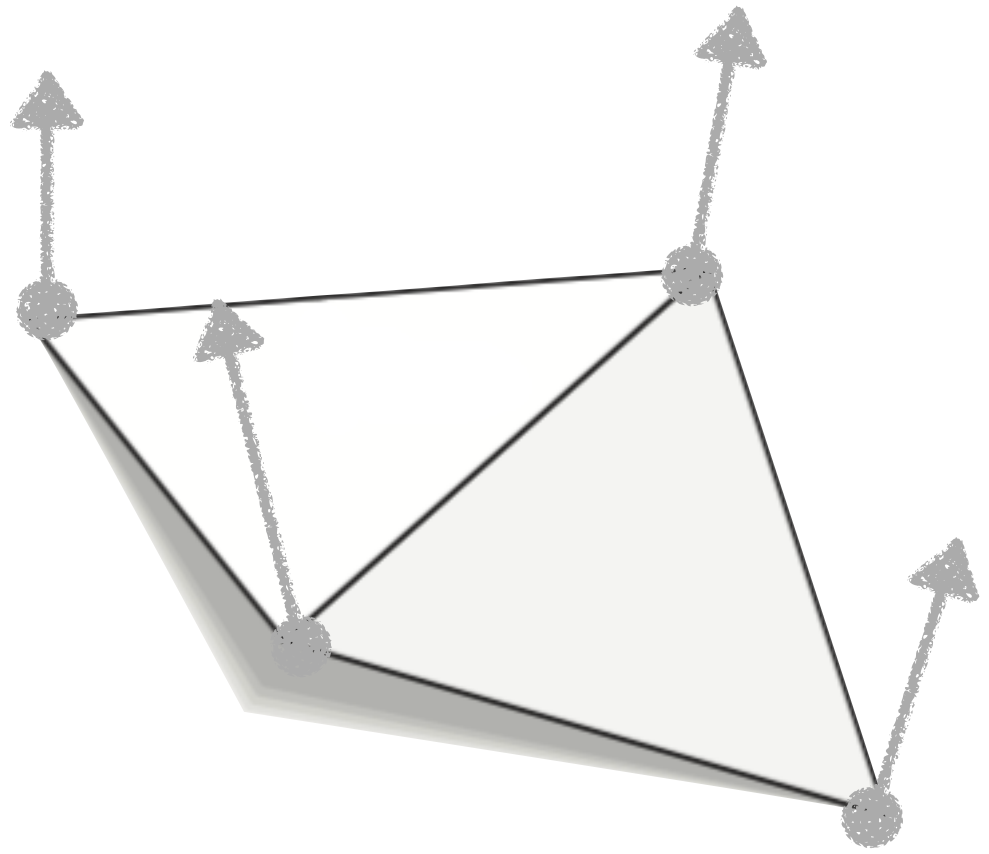
D. Coeurjolly

Introduction

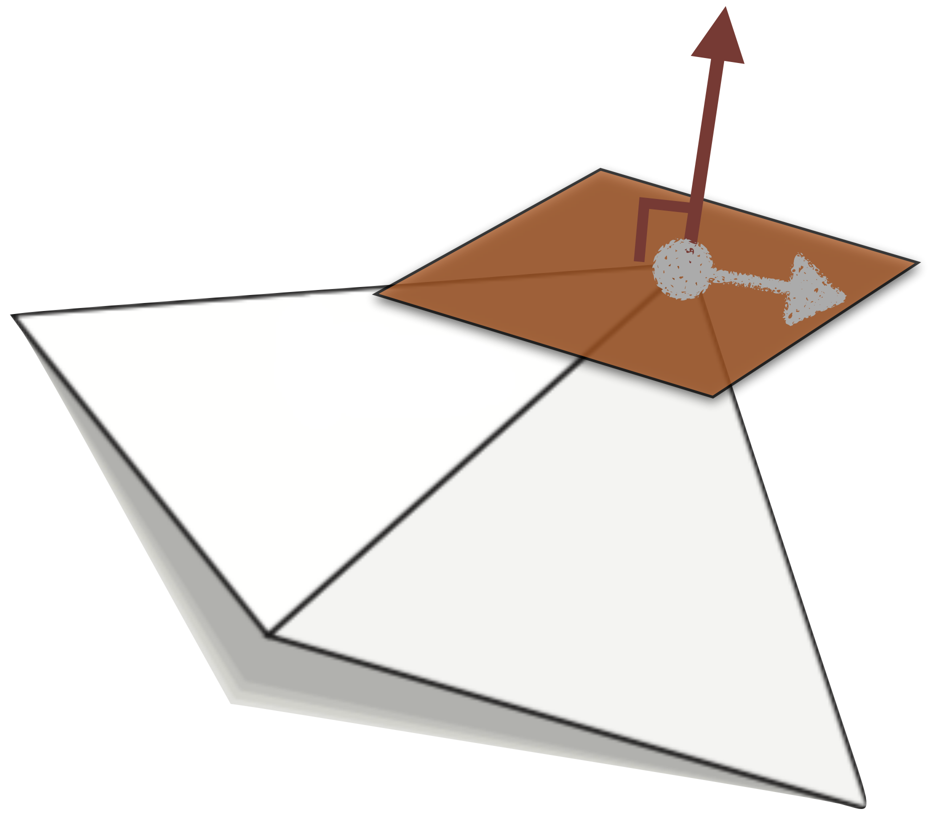
The Bestiary:
ambient vs. Intrinsinc Vector Fields, Direction fields, k-RoSy fields….



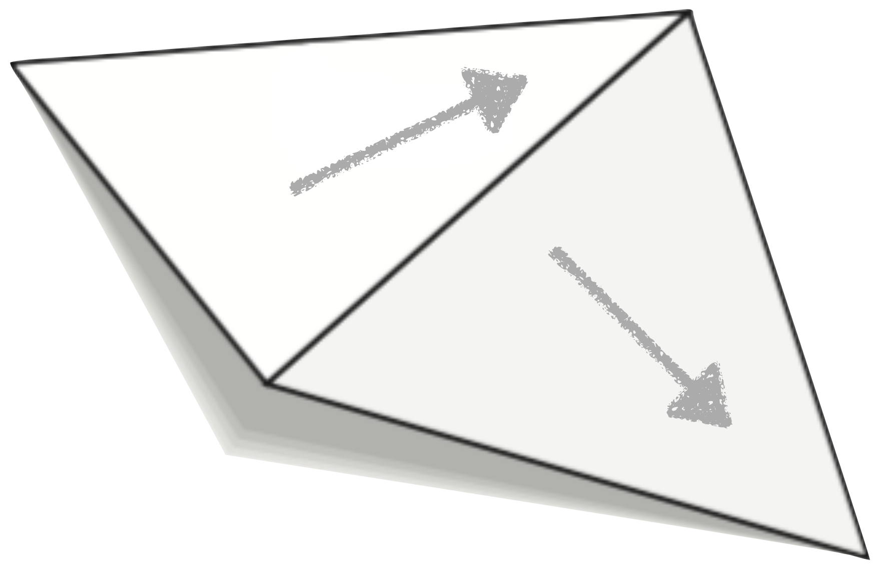
ambient per face



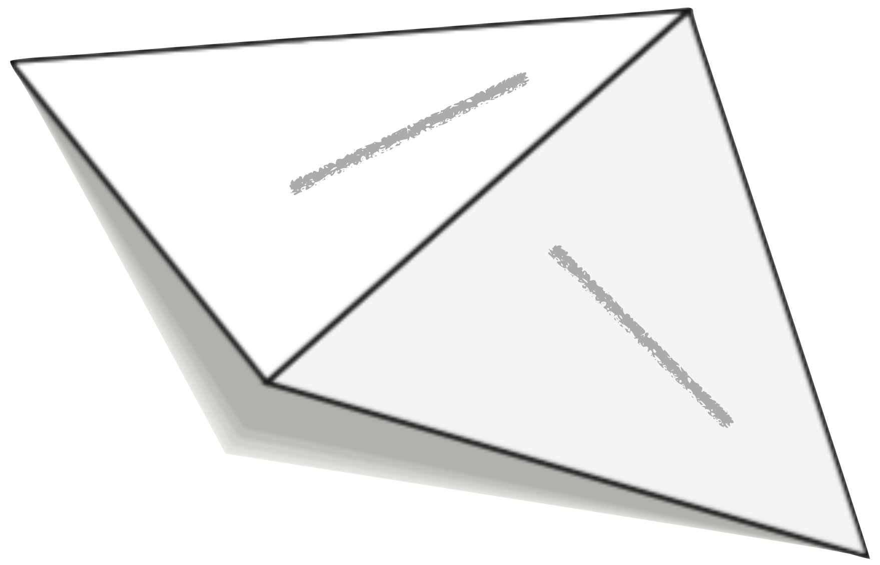
ambient per vertex



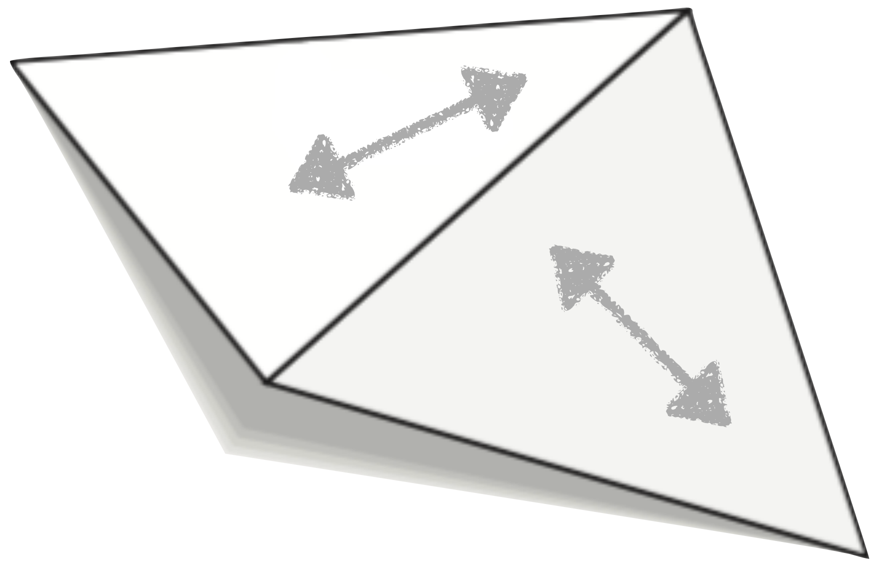
intrinsic per vertex



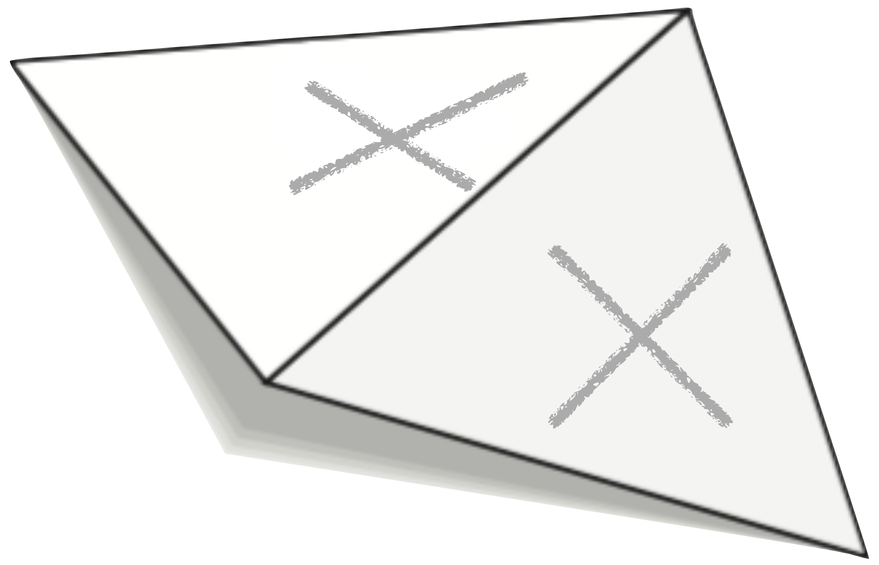
intrinsic per face



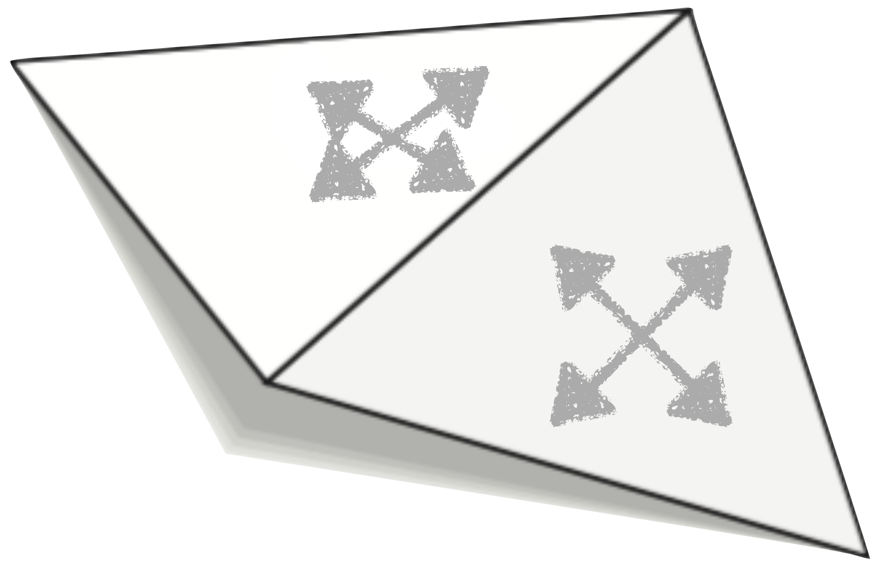
1-RoSy direction field



2-RoSy vector field



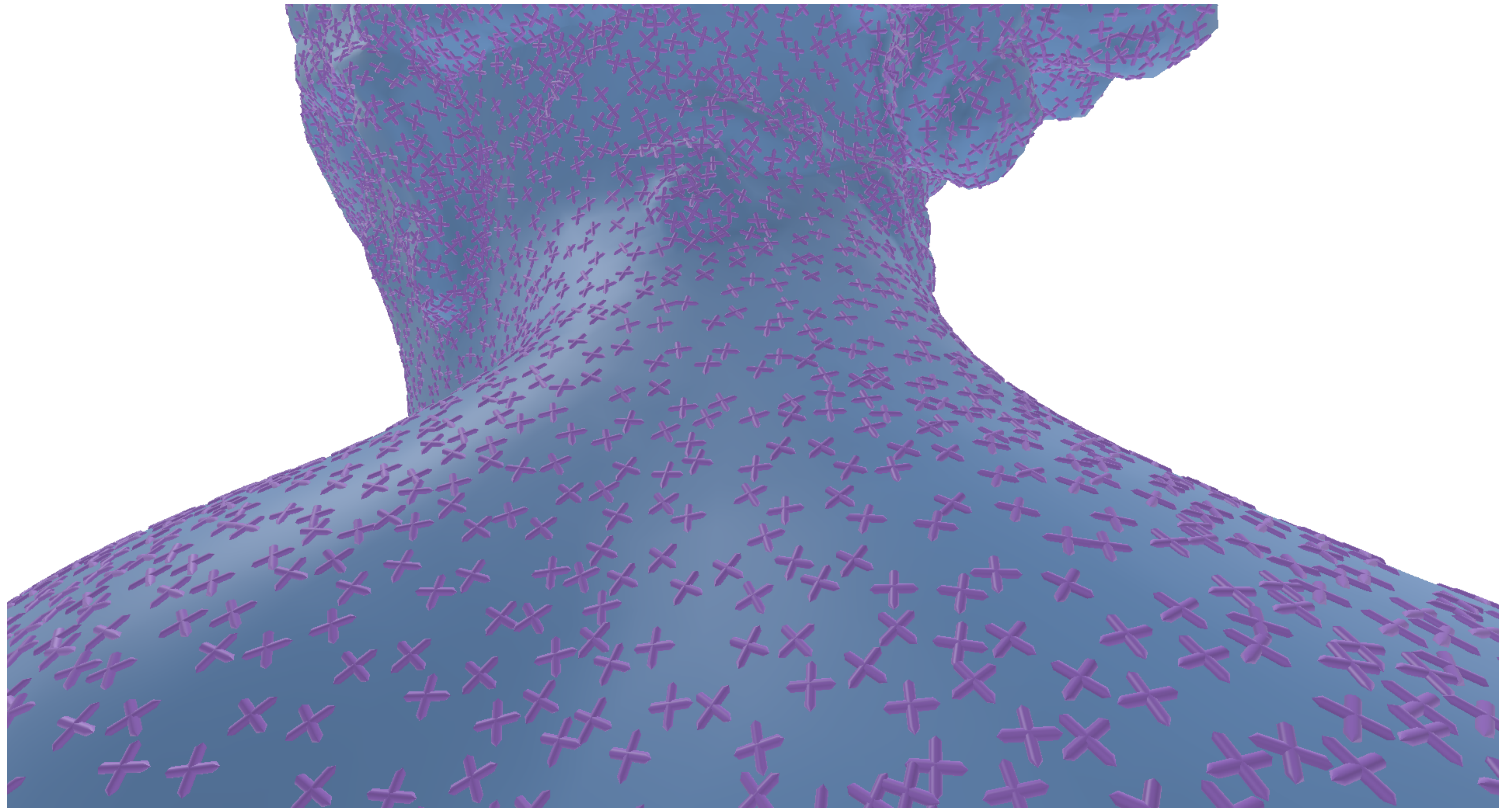
4-RoSy direction field



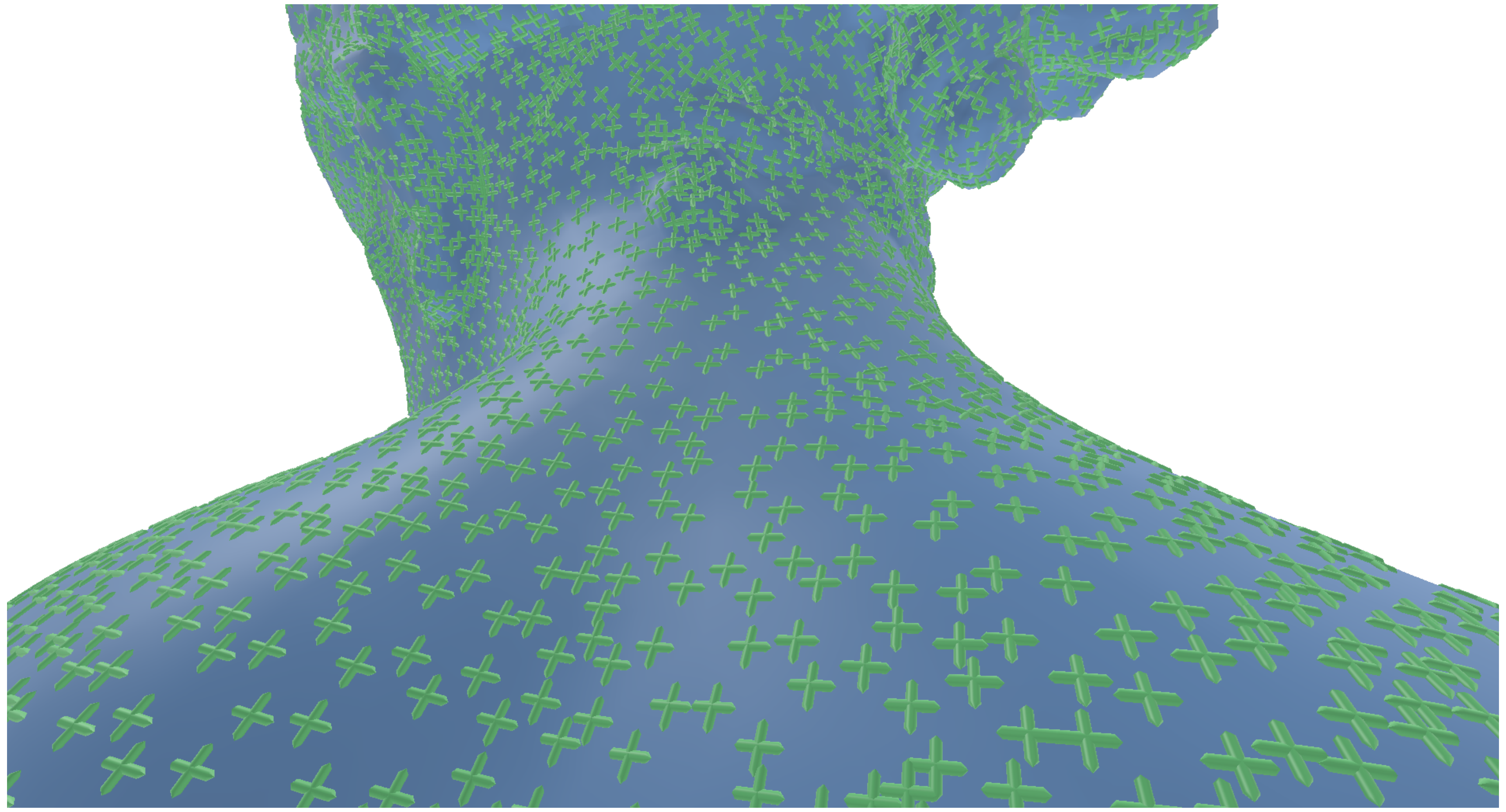
4-RoSy vector field

RoSy = Rotational Symmetry

Integrability / Smooth (tangent) vector fields



😟



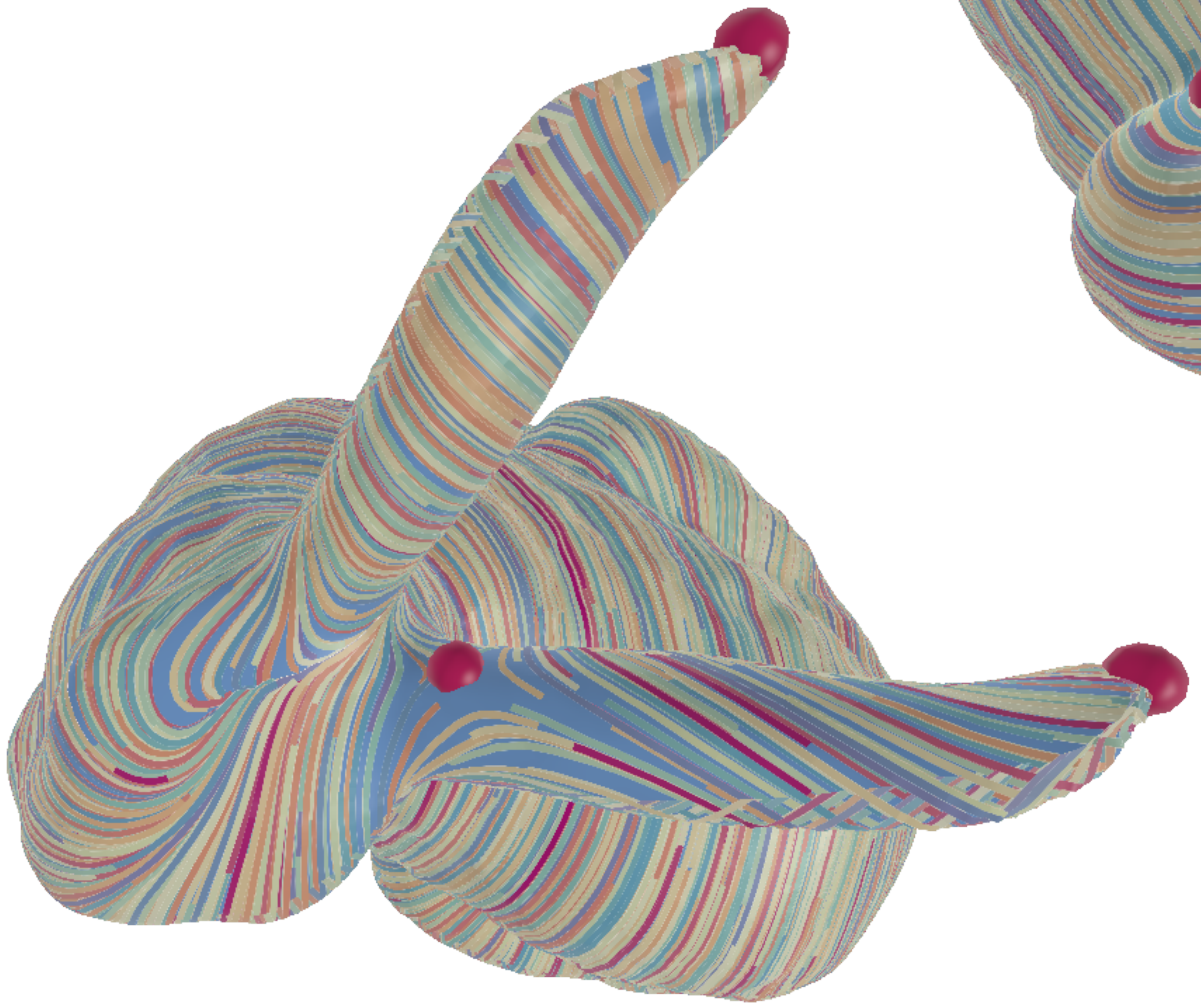
😍

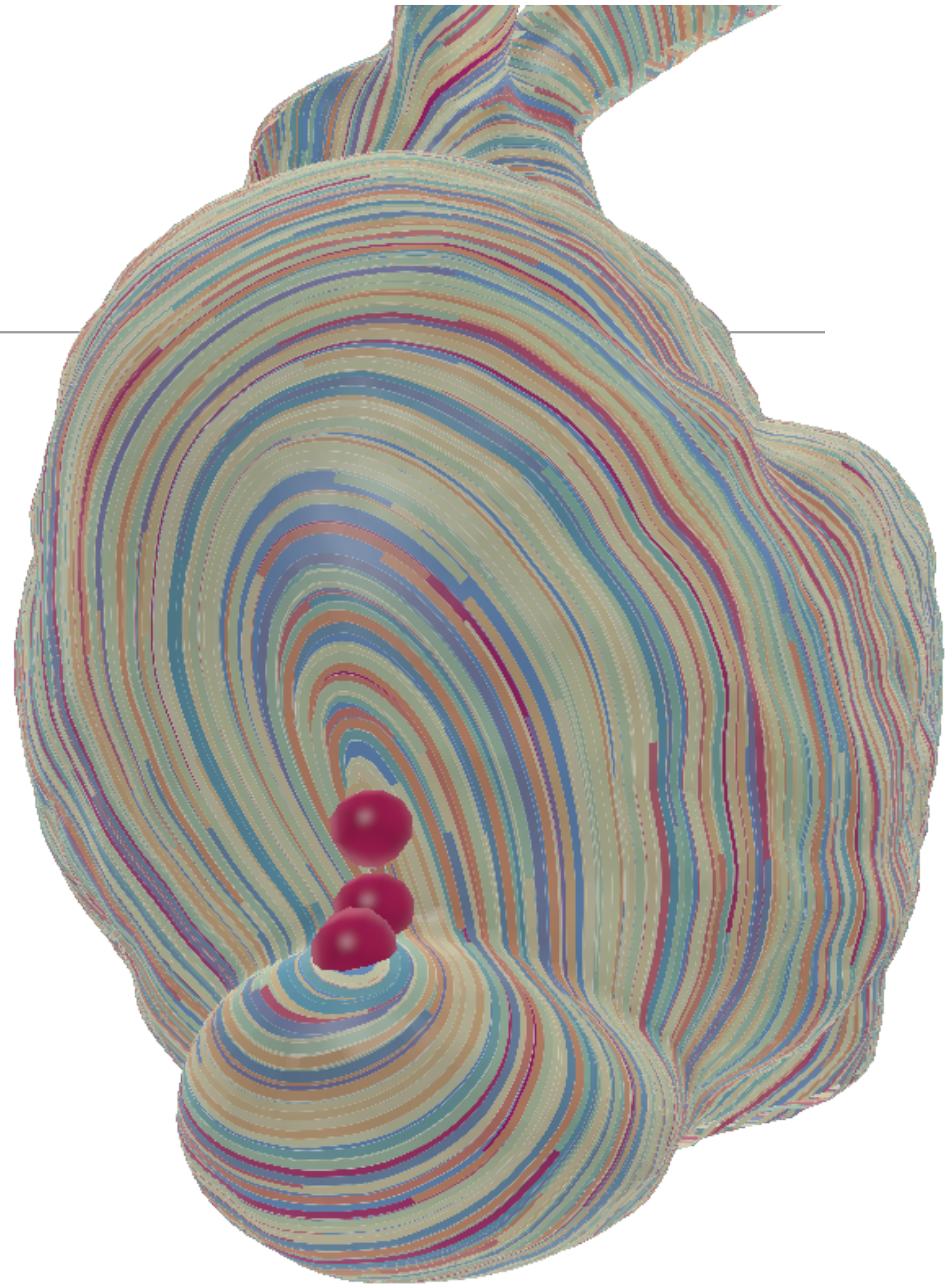
⇒ Notion of parallel transport of vectors on a manifold

⇒ Energy model

⚠ Singularities!!

Let’s comb some bunnies…





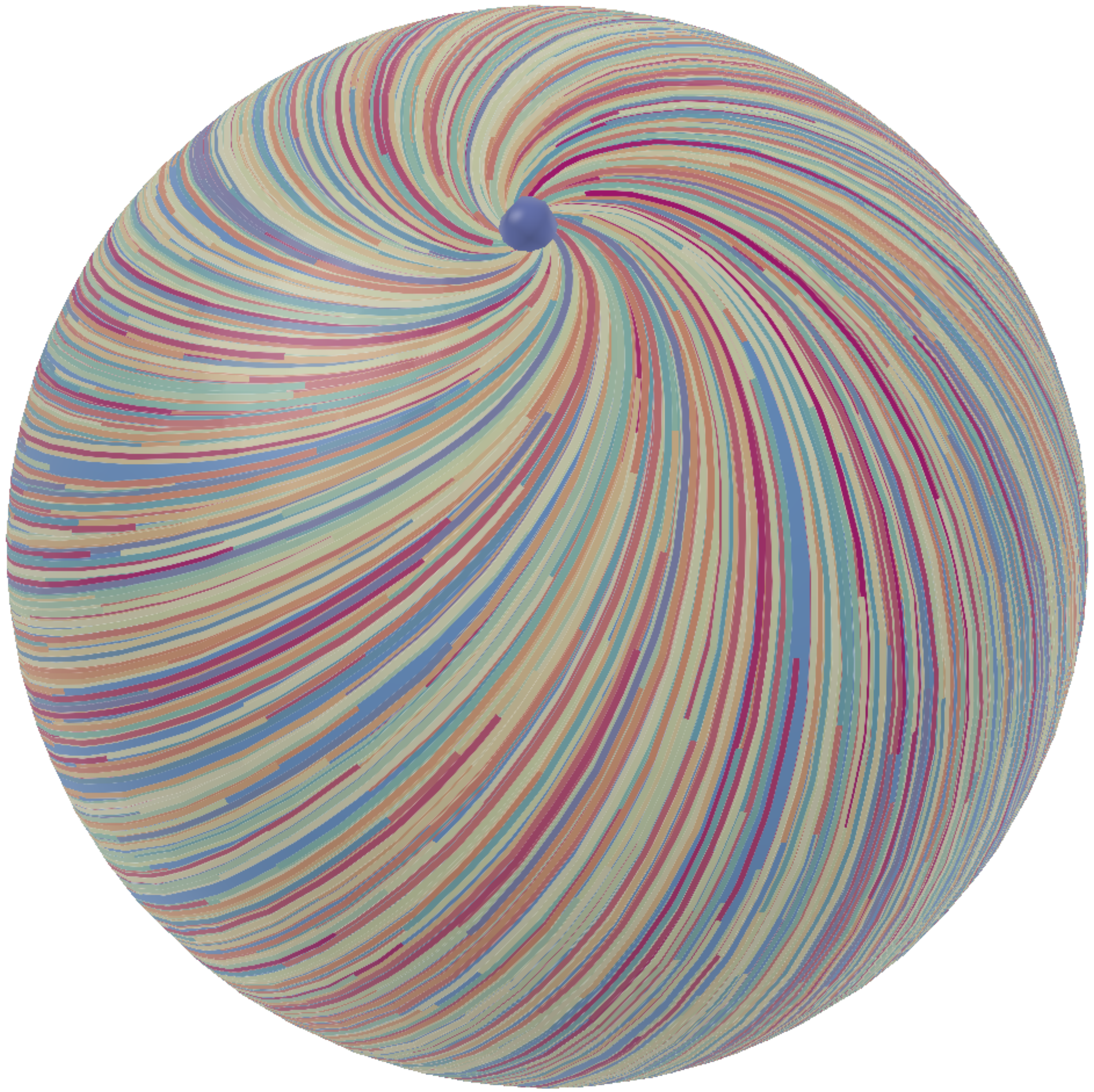
Let's comb some bunnies…

Thm: Bunnies cannot be combed properly!

Coro: Bunnies have at least 2 cowlicks.

Let’s try simpler shapes…

😍



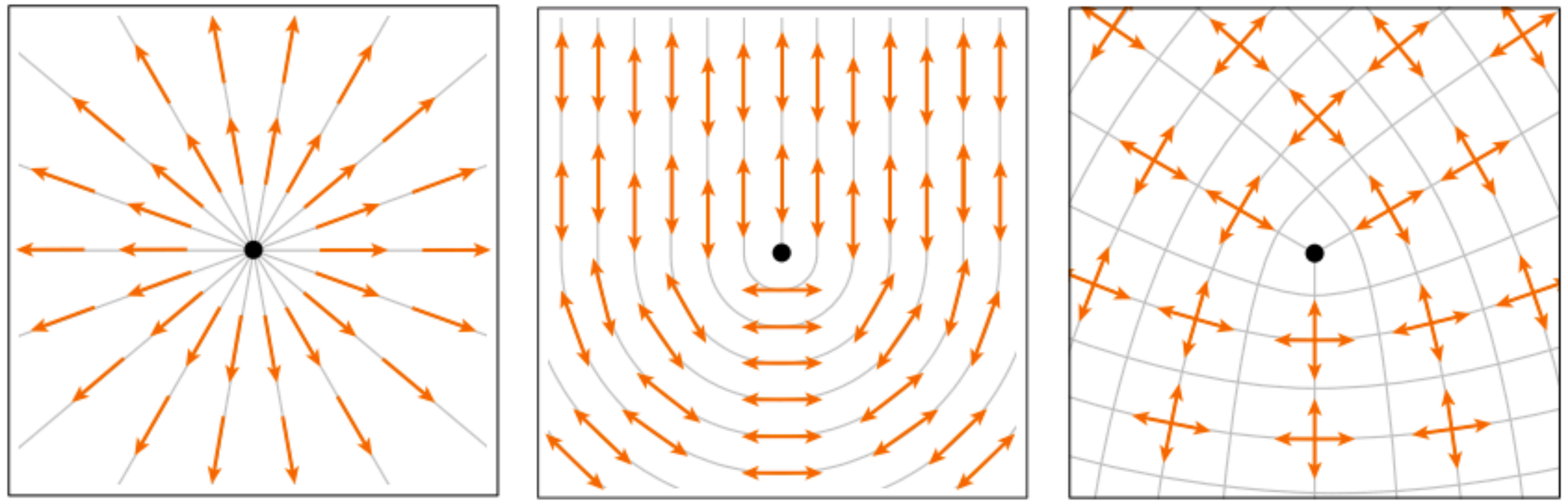
😟

Singularities encode the topology

Singularities have rational indices

Sum of indices == Euler characteristics

$$\sum_i k_i = \chi$$



+1 +1/2 +1/4

N-RoSy fields: $k_i = n_i / N,\ n_i \in \mathbb{Z}$

Singularities encode the topology

Singularities have rational indices

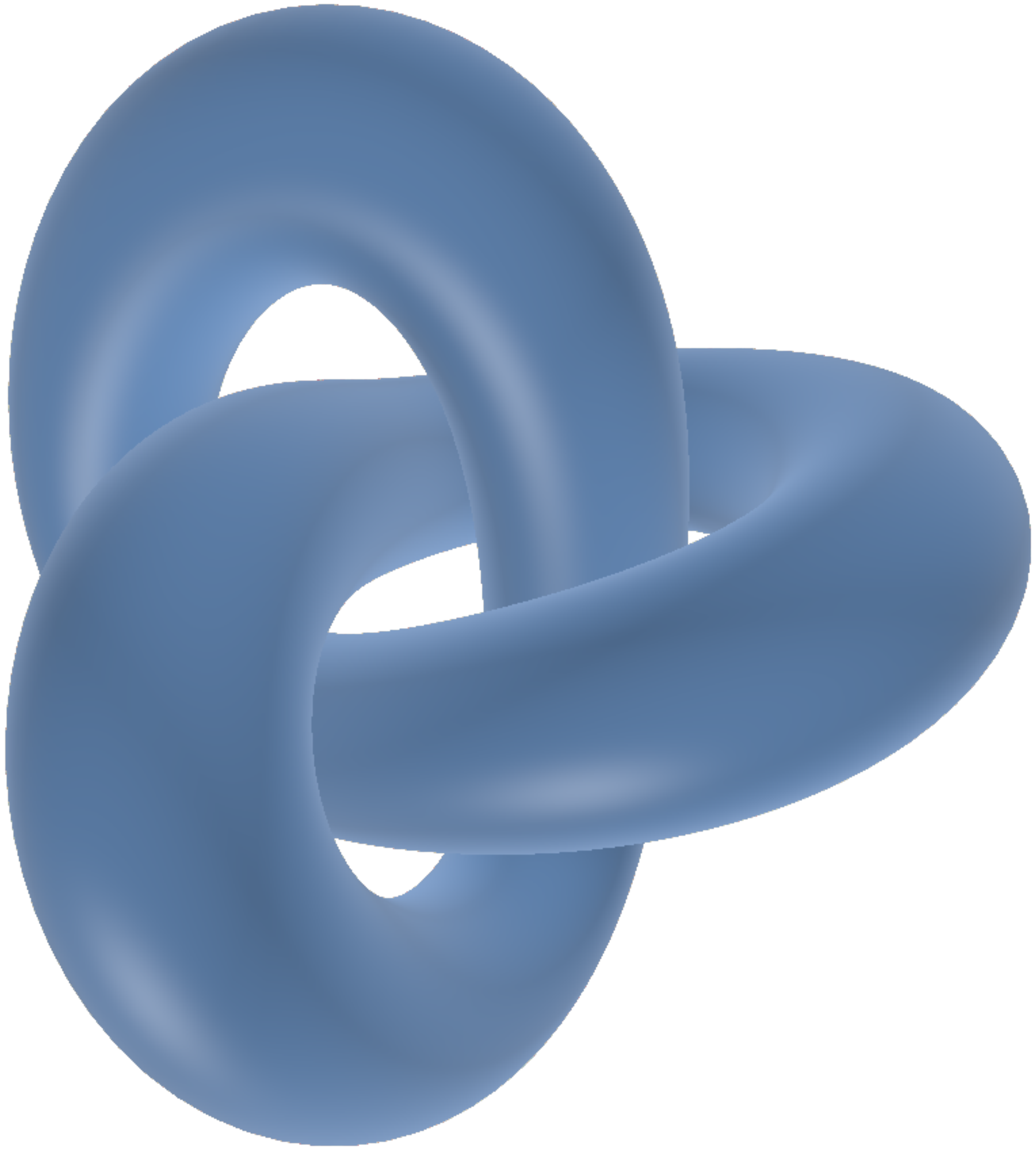
Sum of indices == E

$\sum_i k_i = \chi$

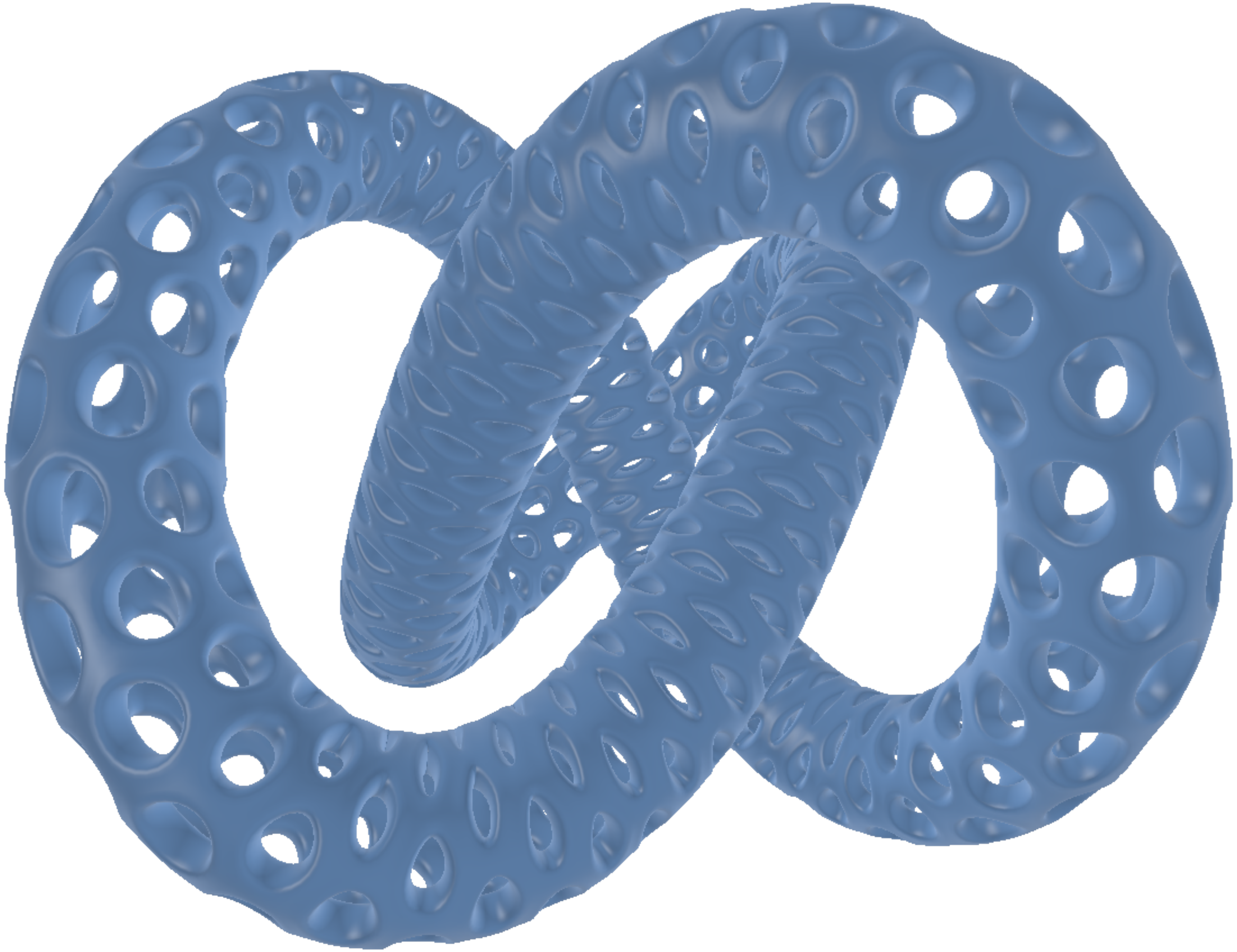
-1/4

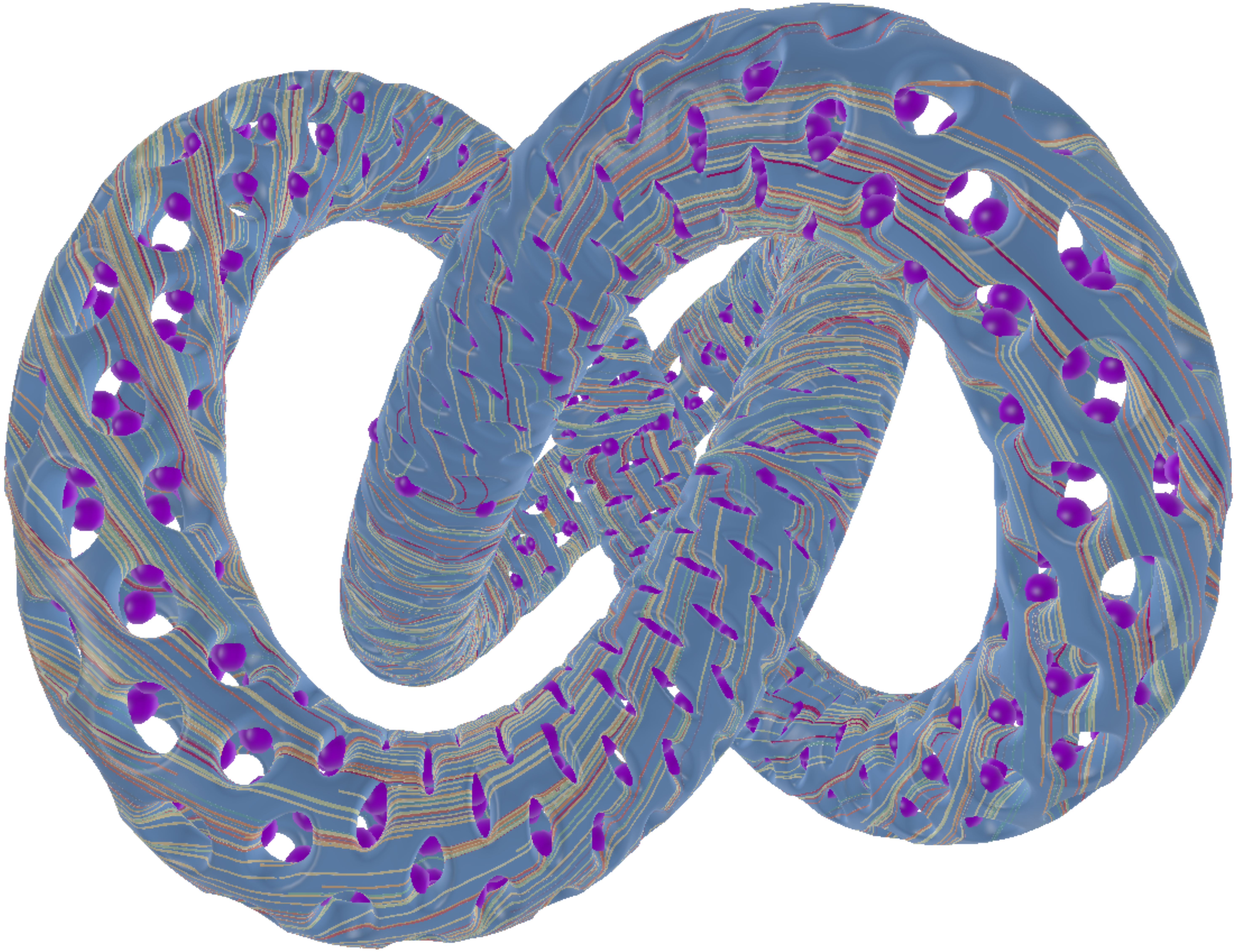
Proof Punk Bunnies Theorem: Bunnies are homeomorphic to spheres (😶) whose Euler characteristics are 2. At least two +1 singularities for 1-fields, or a single +2 one. QED:=)

N-RoSy fields: $k_i = n_i / N,\ n_i \in \mathbb{Z}$



$\chi = 0$



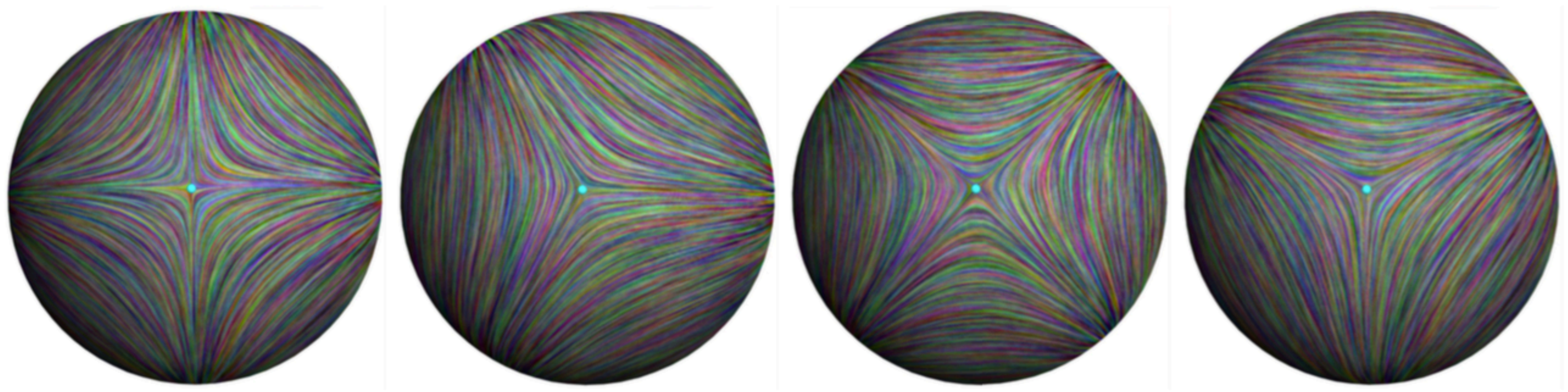


$\chi = -1440$

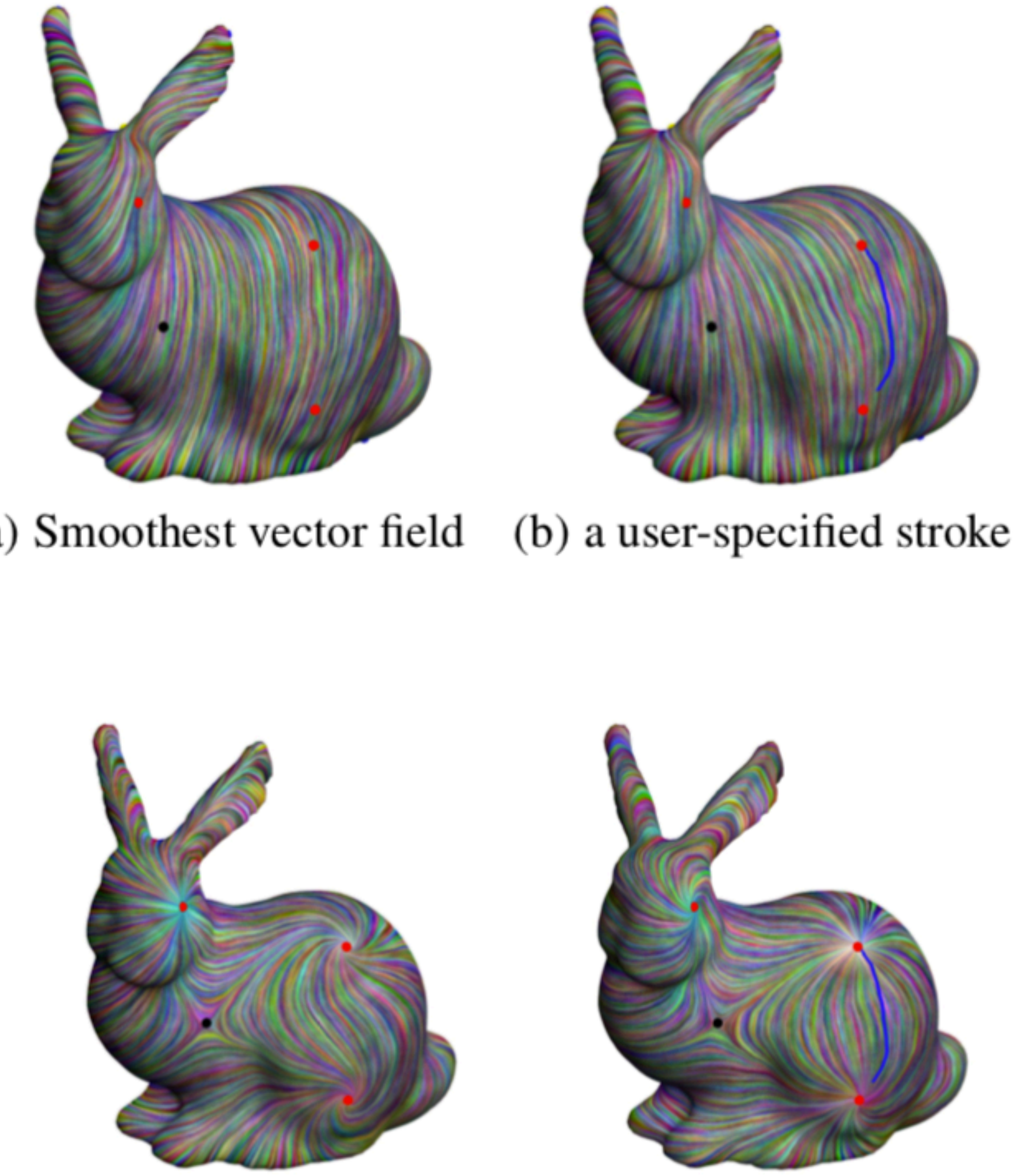
Controlling a vector field ?

1. Add hard / soft constraints on the field
   - ➡ per triangle constraints
   - ➡ via brushing tools

2. Play with singularities (location+indices)



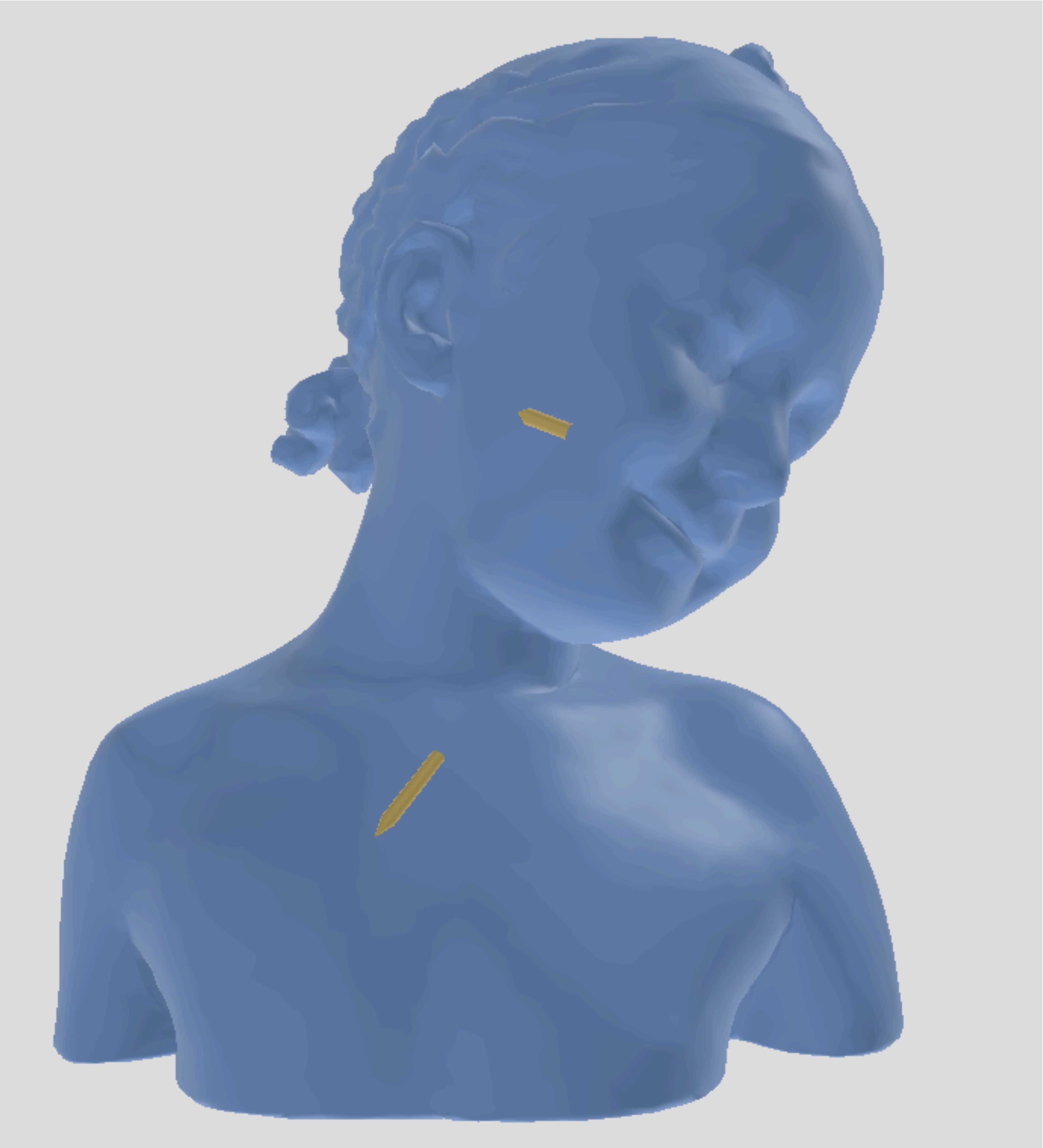
(a) (b) (c) (d)



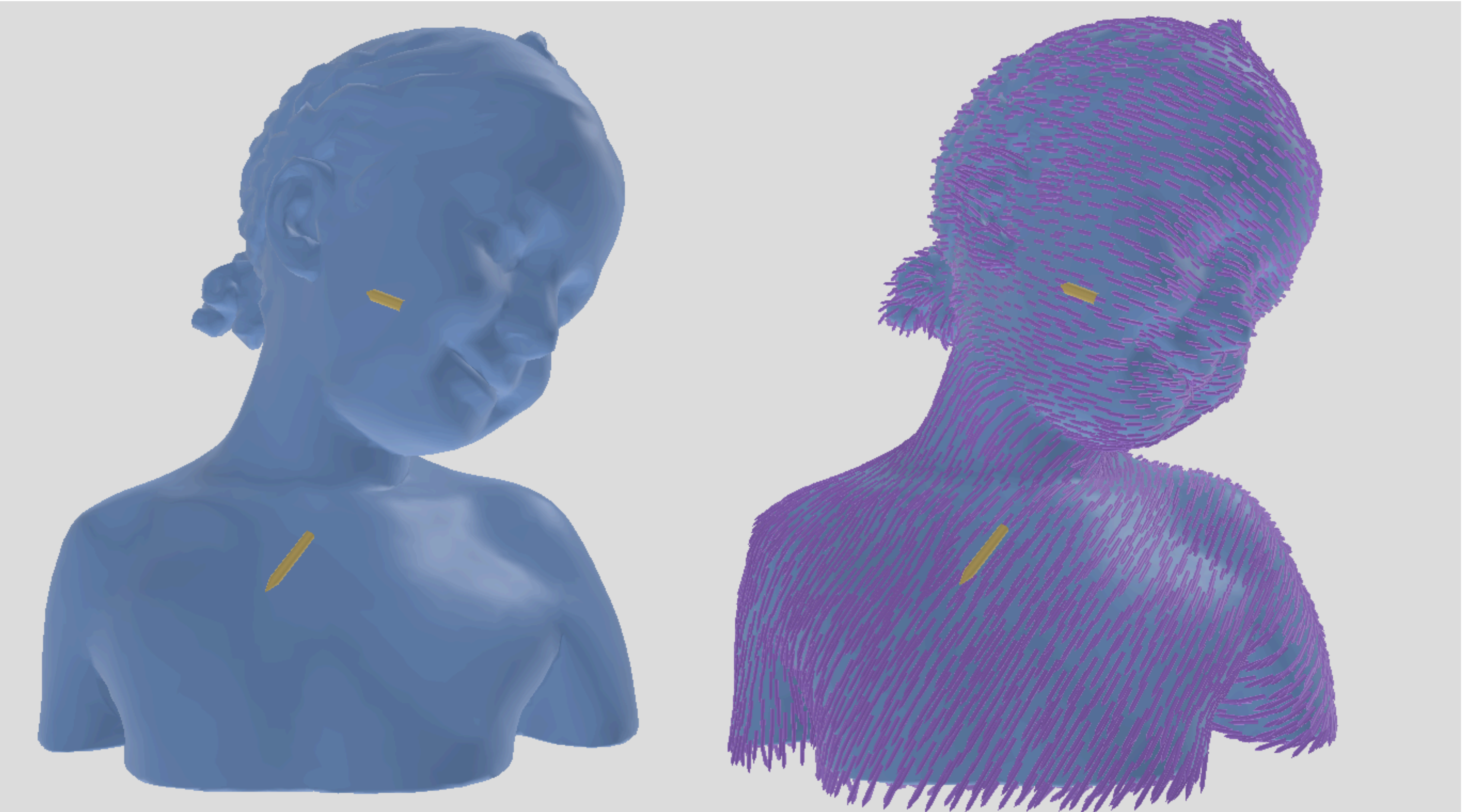
(a) Smoothest vector field (b) a user-specified stroke

(c) prescribed singularities (d) both

Objectives: authoring vector fields



Objectives: authoring vector fields

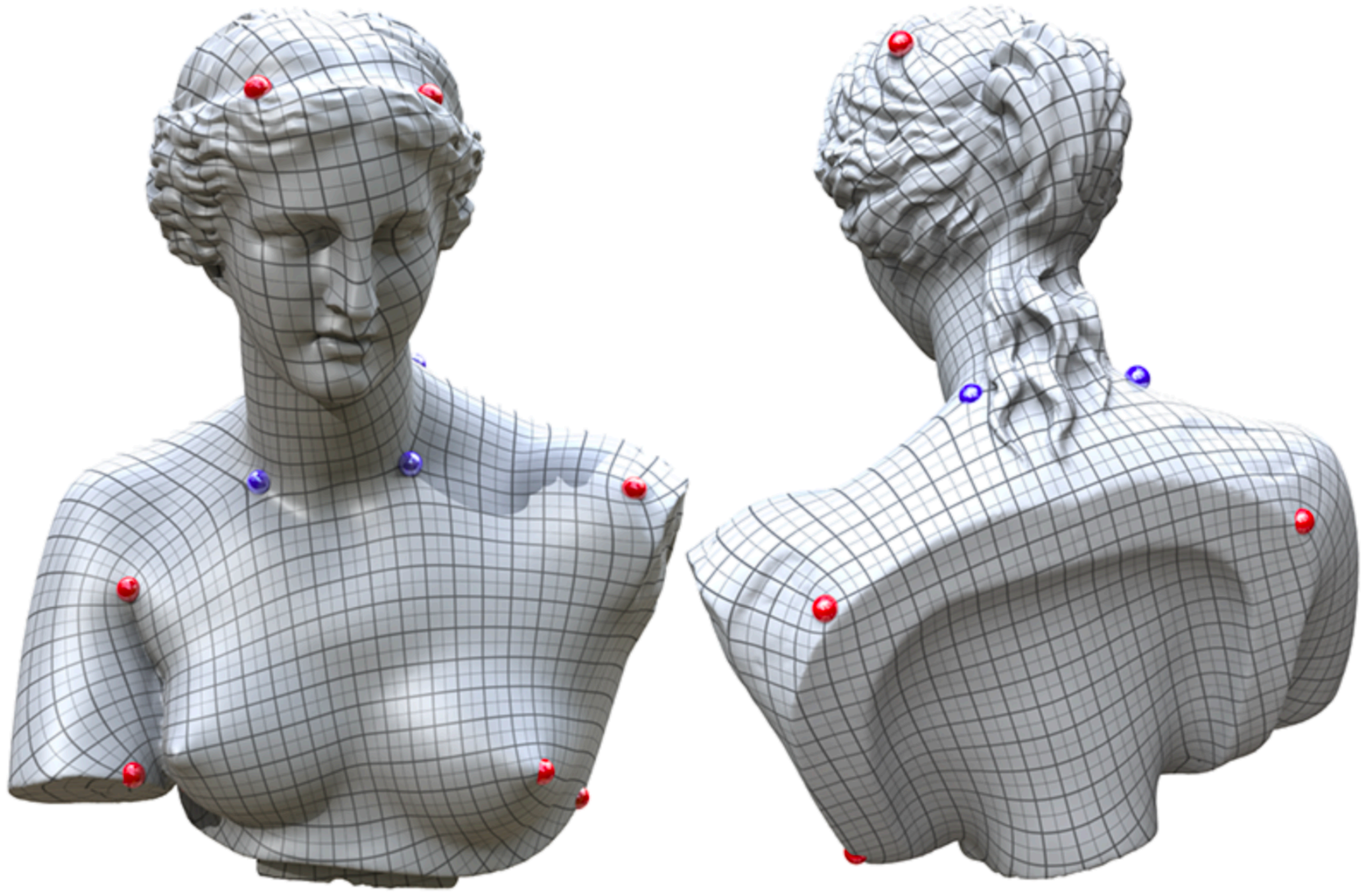


Objectives: authoring vector fields

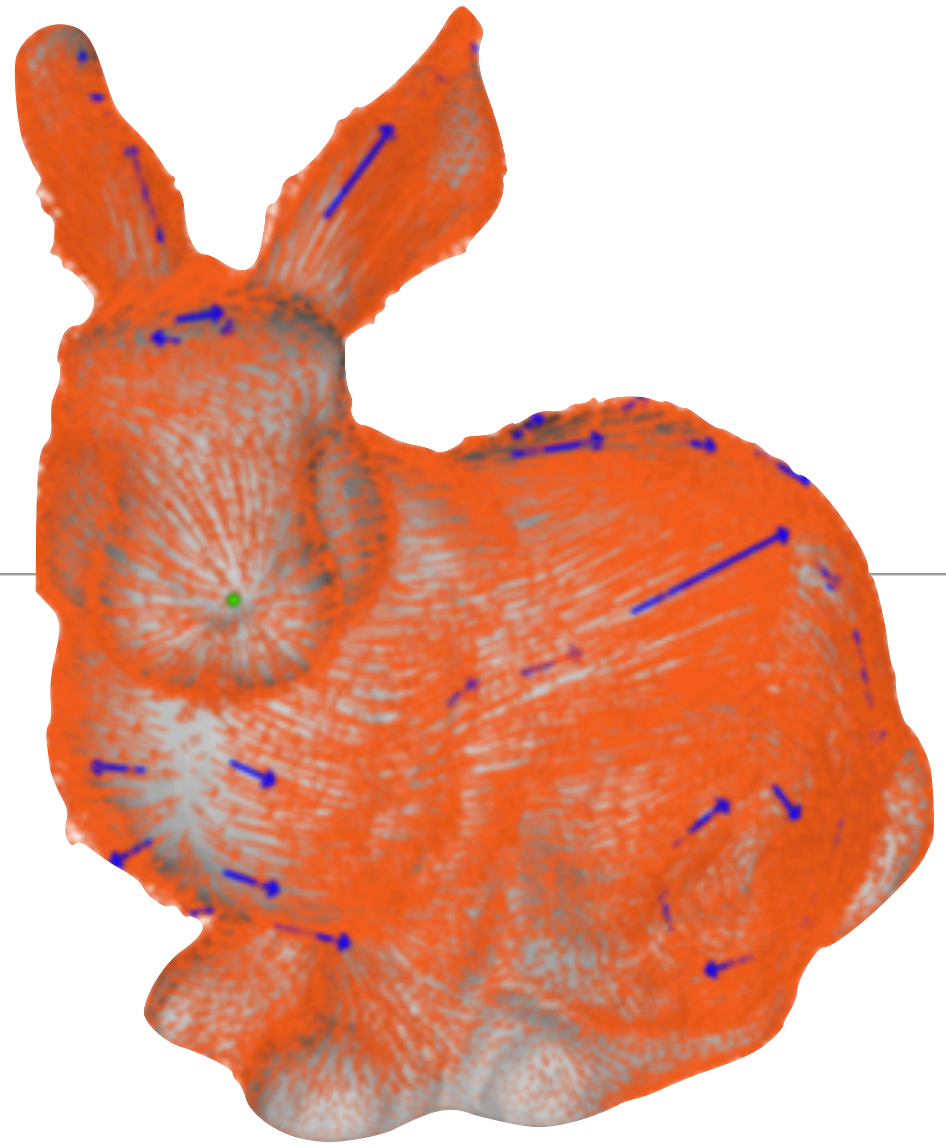
QuadMeshes

4-RoSy field + meshing

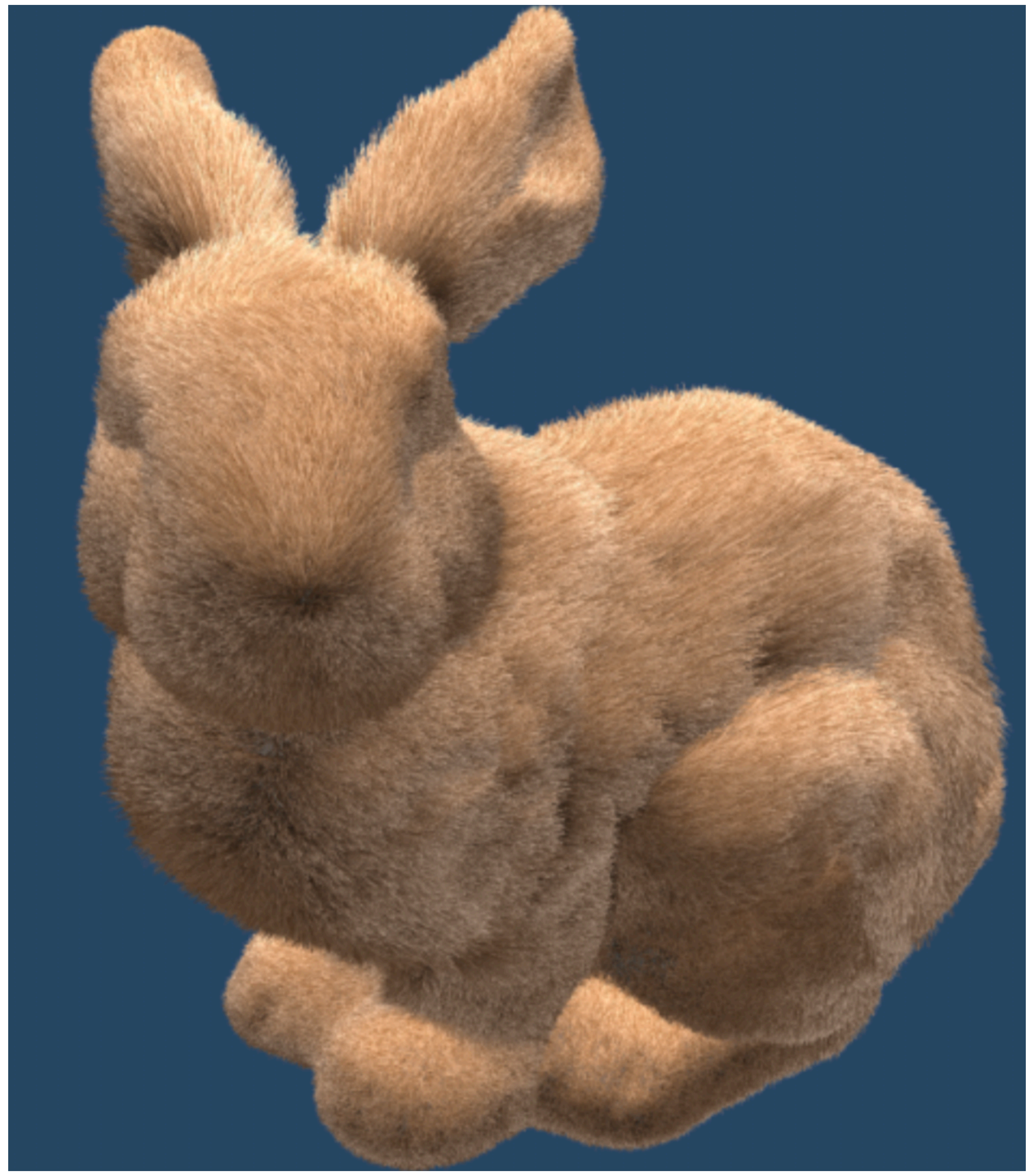
(usually curvature guided, singularities matter)



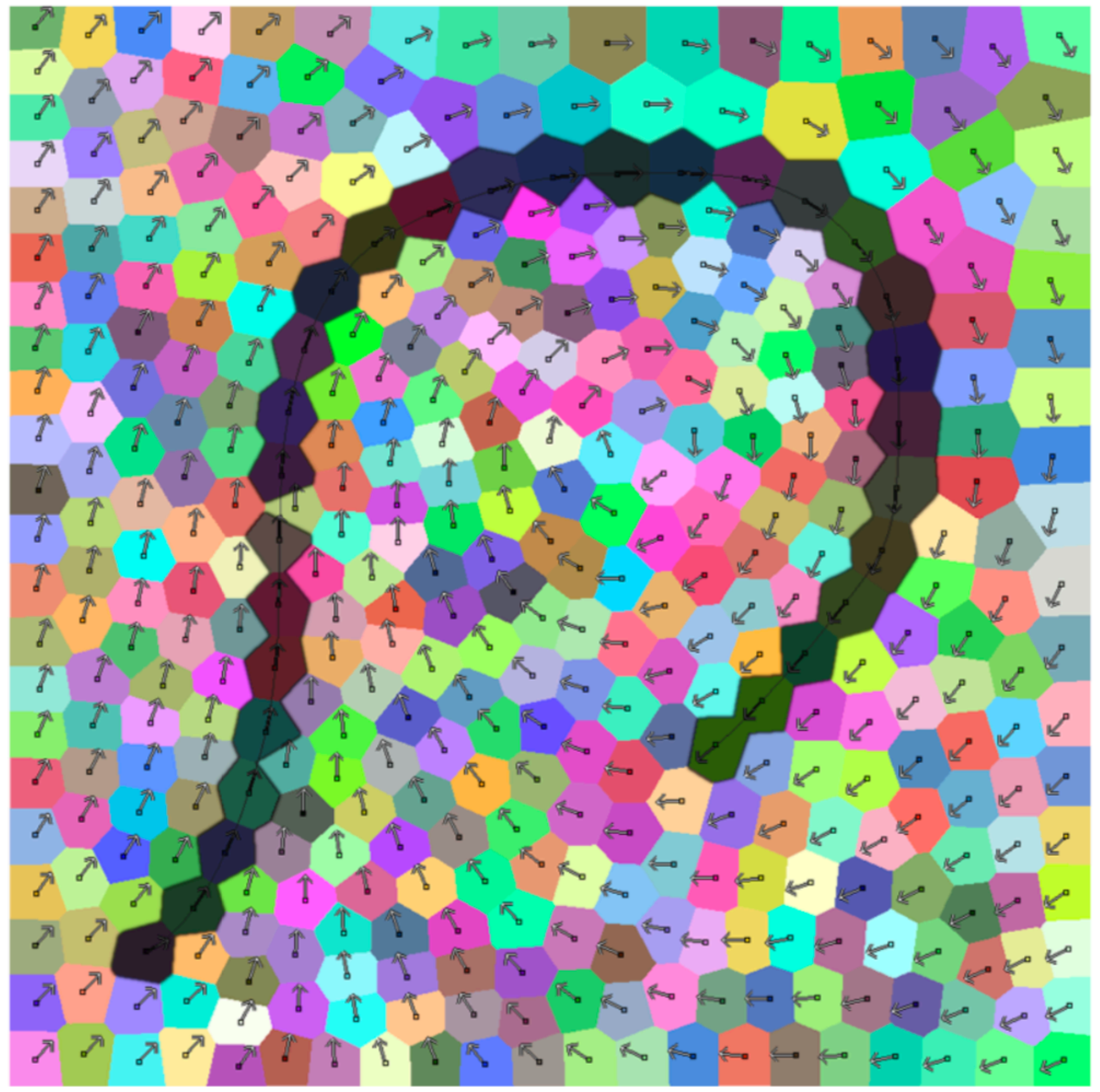
Sweet kitty warm kitty little ball of fur…



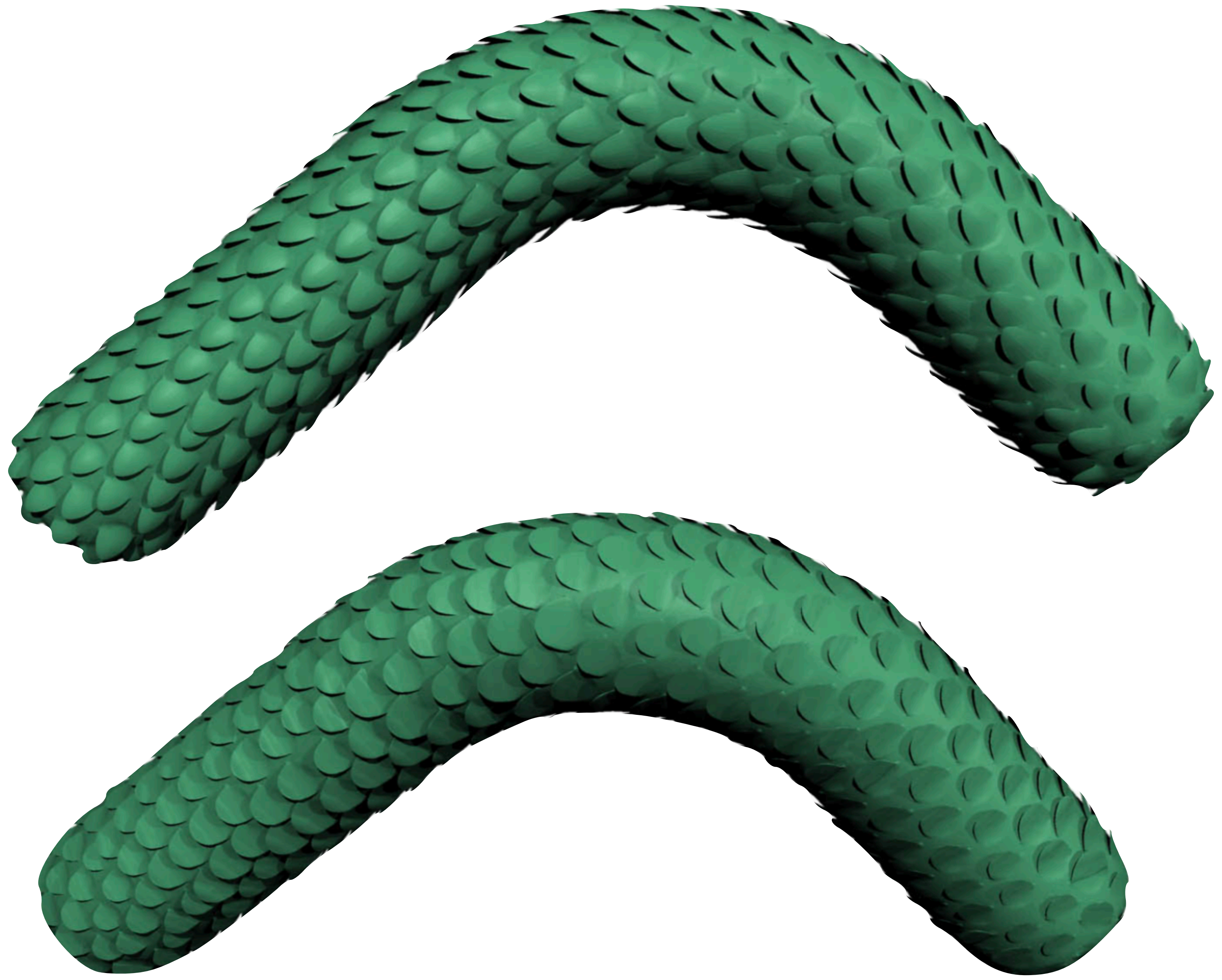
1-RoSy field to control the hair in the tangent plane + extrusion to 3D



Scales, Feathers, geometries…



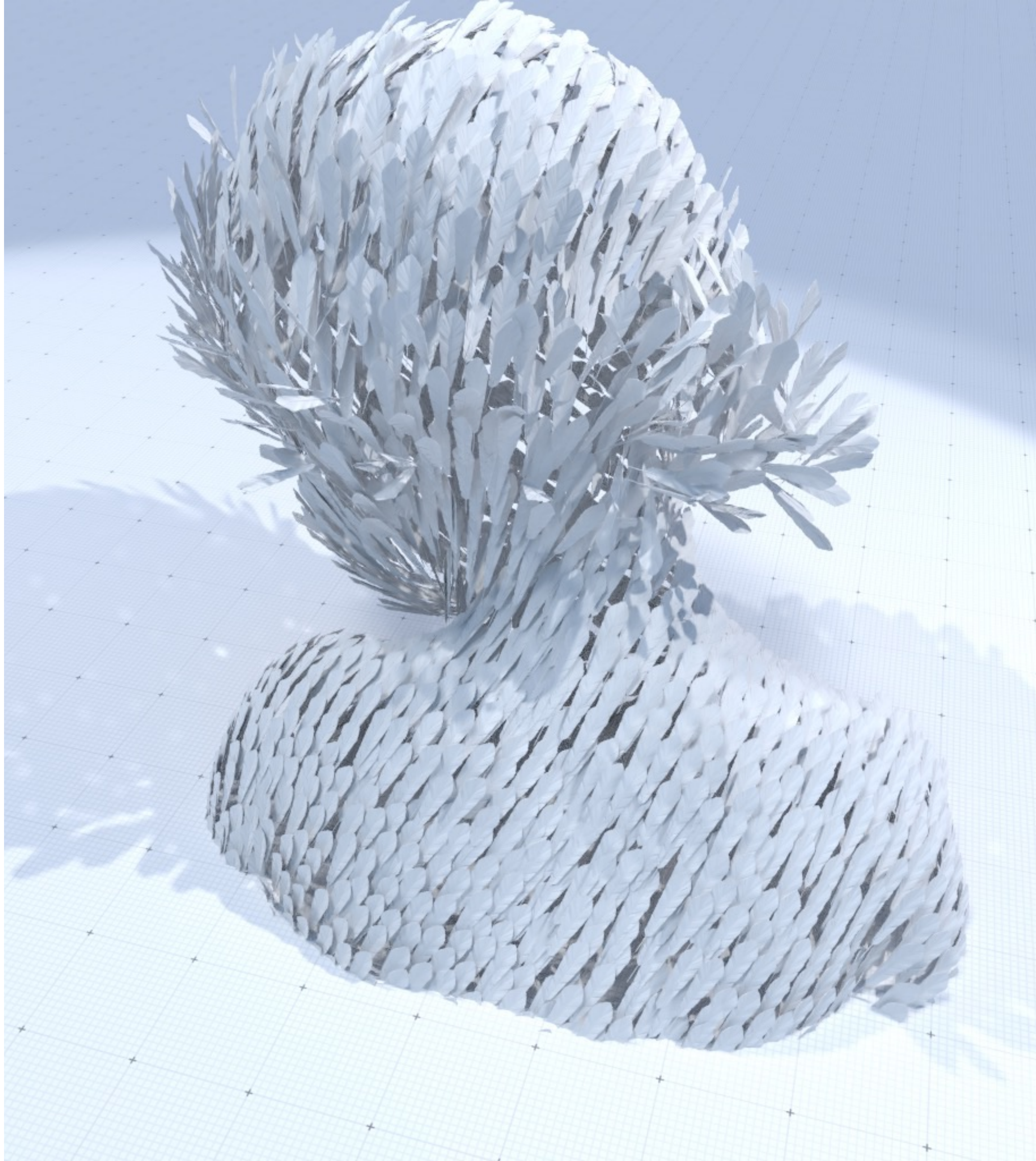
1-RoSy field to have a consistent local frame + patterns glued to the base maesh



Example: VF controlled micro-geometries

Base mesh + pattern

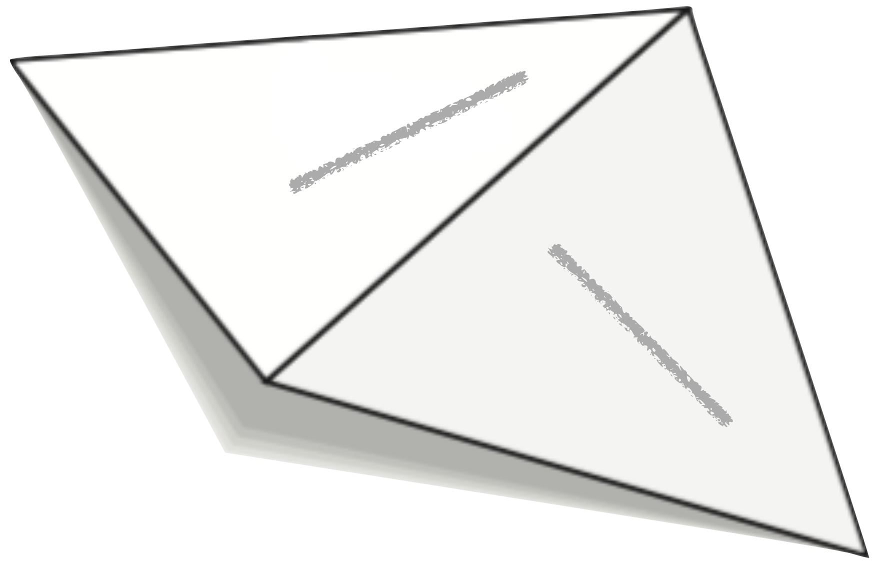
- Compute consistent frames using a smooth 1-field with smooth constraints
- Surface sampling
- Oriented instances
- Rendering



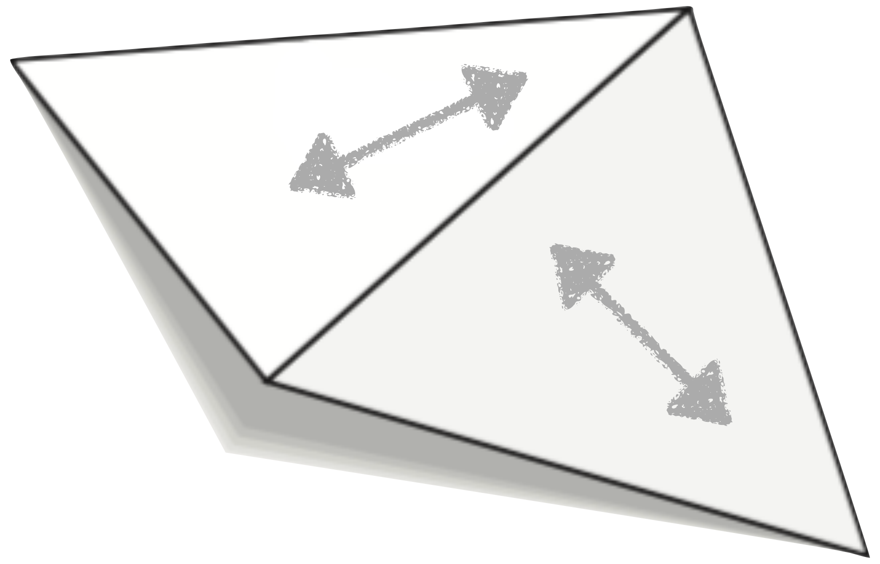
Mathematical background

Directional/vector fields

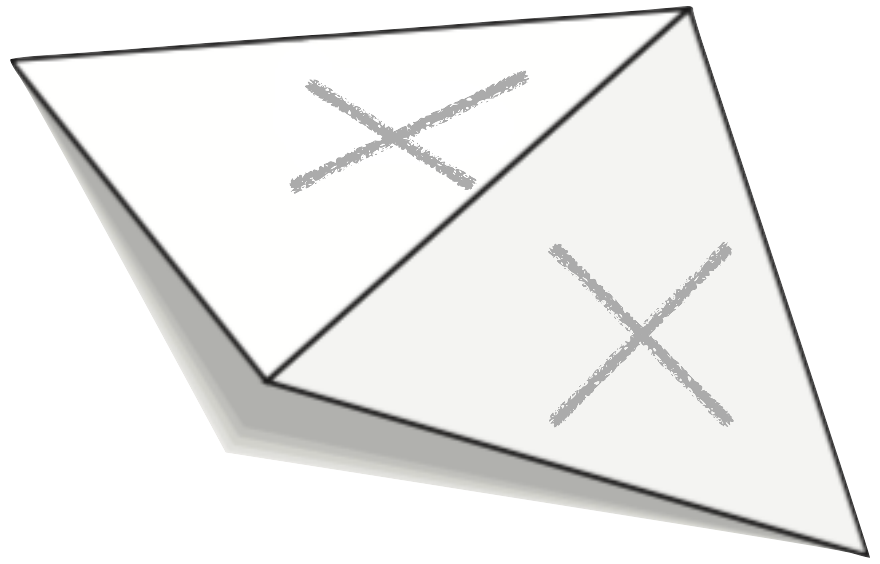
1-vector field	One vector, classical "vector field"
2-direction field	Two directions with π symmetry, "line field", "2-RoSy field"
1^3-vector field	Three independent vectors, "3-polyvector field"
4-vector field	Four vectors with $\pi/2$ symmetry, "non-unit cross field"
4-direction field	Four directions with $\pi/2$ symmetry, "unit cross field", "4-RoSy field"
2^2-vector field	Two pairs of vectors with π symmetry each, "frame field"
2^2-direction field	Two pairs of directions with π symmetry each, "non-ortho. cross field"
6-direction field	Six directions with $\pi/3$ symmetry, "6-RoSy"
2^3-vector field	Three pairs of vectors with π symmetry each



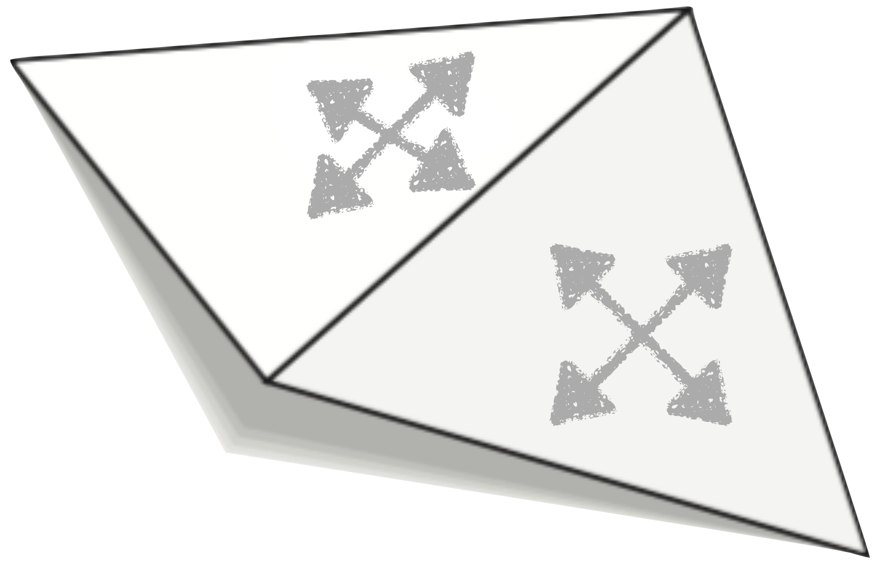
2-RoSy direction field



2-RoSy vector field



4-direction field

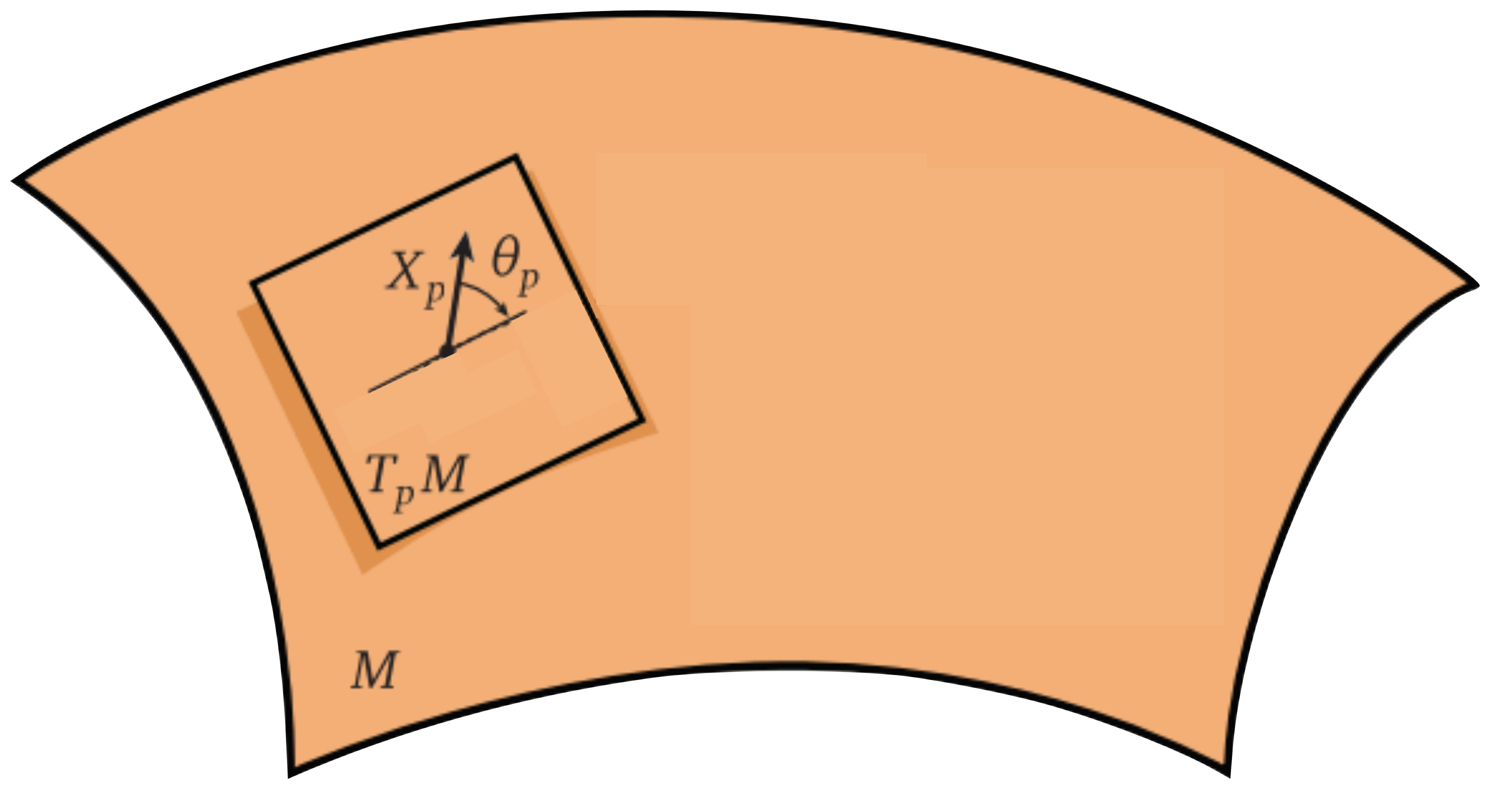


4-vector field

Tangent Bundle

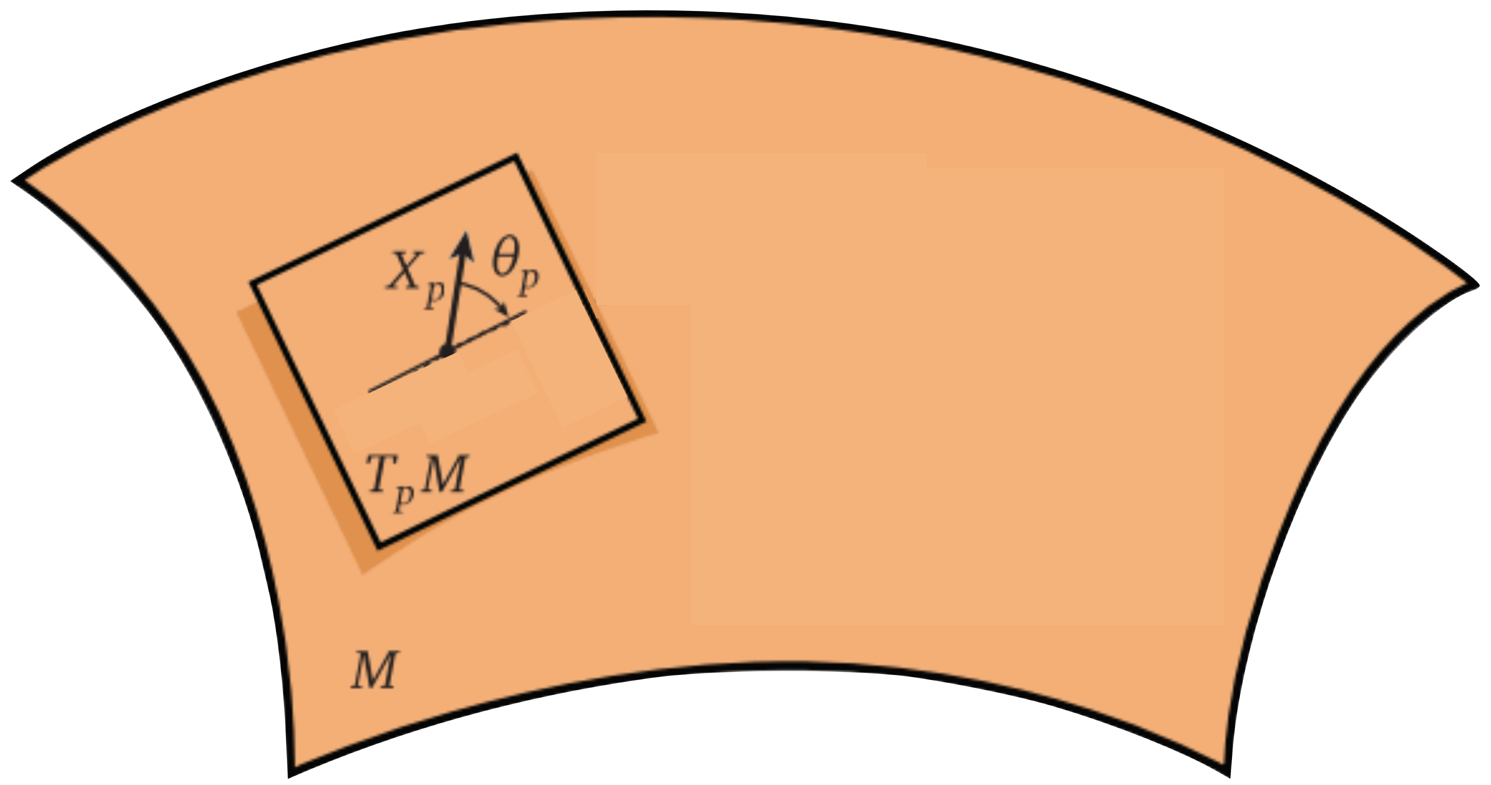
- **Smooth 2-manifold** M, $p \in M$

- **Tangent space** T_pM: orthogonal subspace of the surface normal of M at p



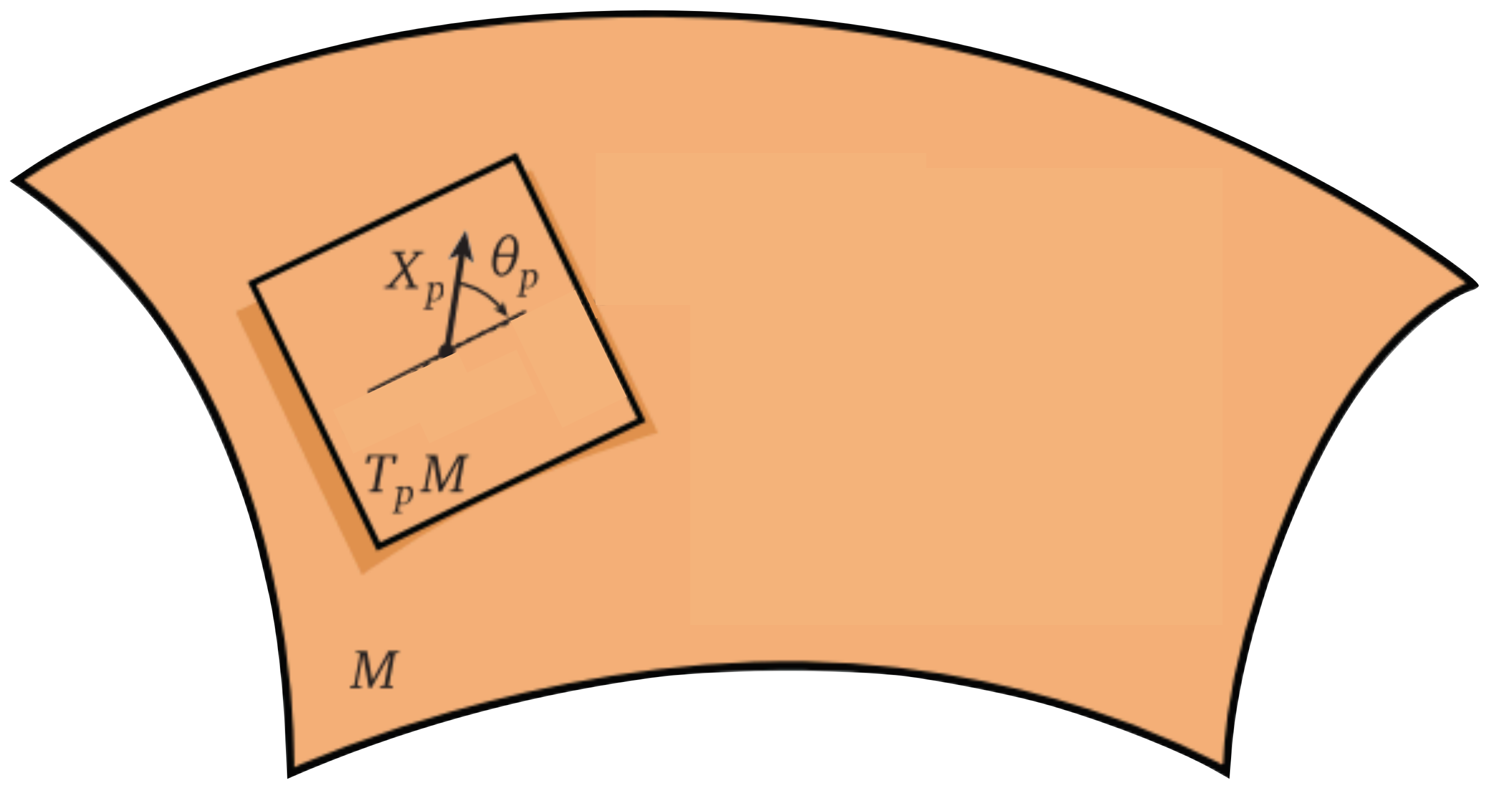
Tangent Bundle

- **Smooth 2-manifold** M, $p \in M$
- **Tangent space** T_pM: orthogonal subspace of the surface normal of M at p
- **Tangent bundle**: $TM = \cup_p T_pM$



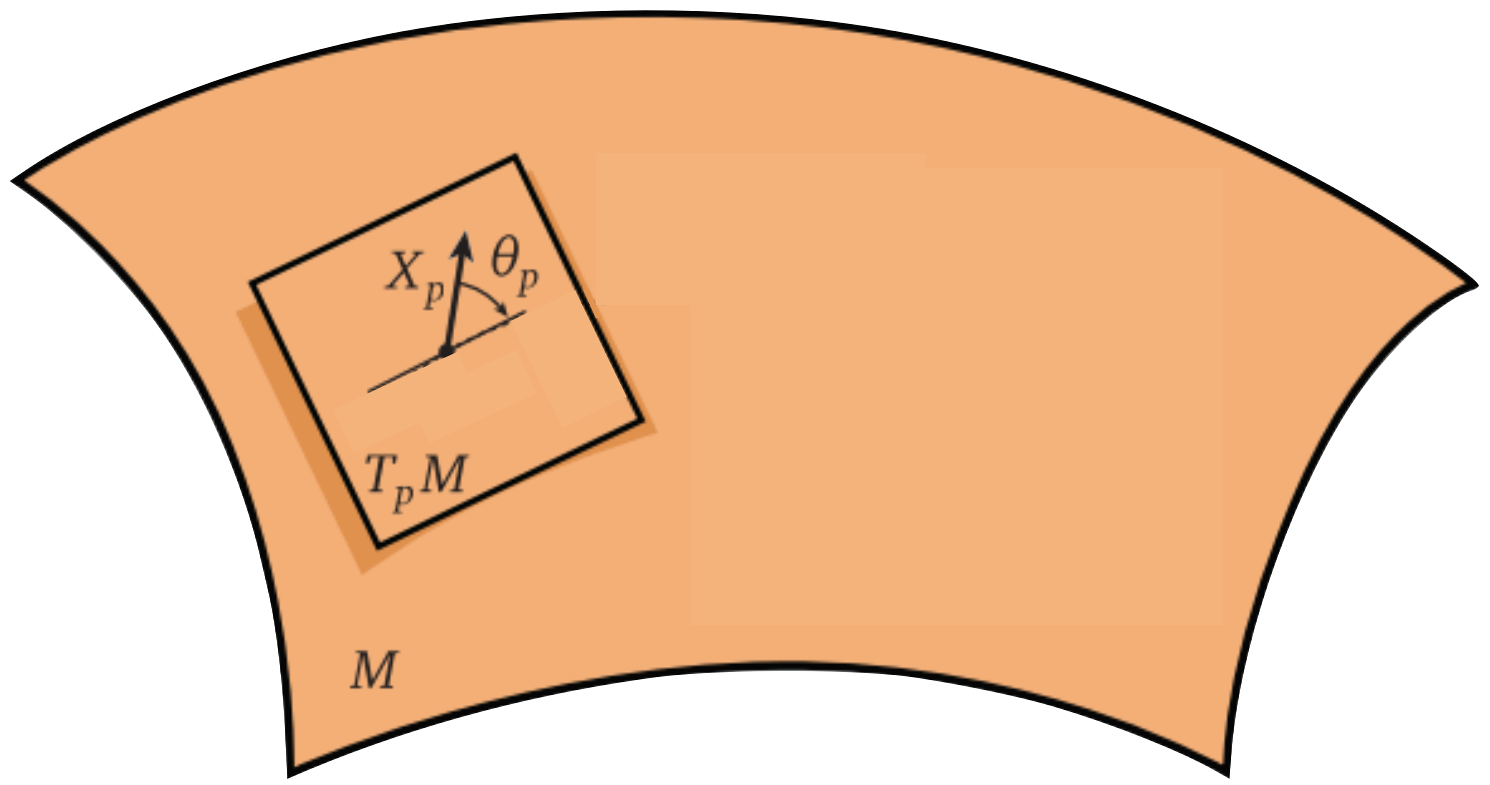
Tangent Bundle

- **Smooth 2-manifold** M, $p \in M$
- **Tangent space** T_pM: orthogonal subspace of the surface normal of M at p
- **Tangent bundle**: $TM = \cup_p T_pM$
- **Projection of a vector** : $\pi : TM \rightarrow M$



Tangent Bundle

- **Smooth 2-manifold** M, $p \in M$
- **Tangent space** T_pM: orthogonal subspace of the surface normal of M at p
- **Tangent bundle**: $TM = \cup_p T_pM$
- **Projection of a vector** : $\pi : TM \to M$
- **Tangent Vector field**: $\mathbf{v} : M \to TM$ such that $\pi \circ \mathbf{v} = Id$



Connections, Parallel Transport…

- **Directional derivative:** $\nabla_{\mathbf{v}} f$

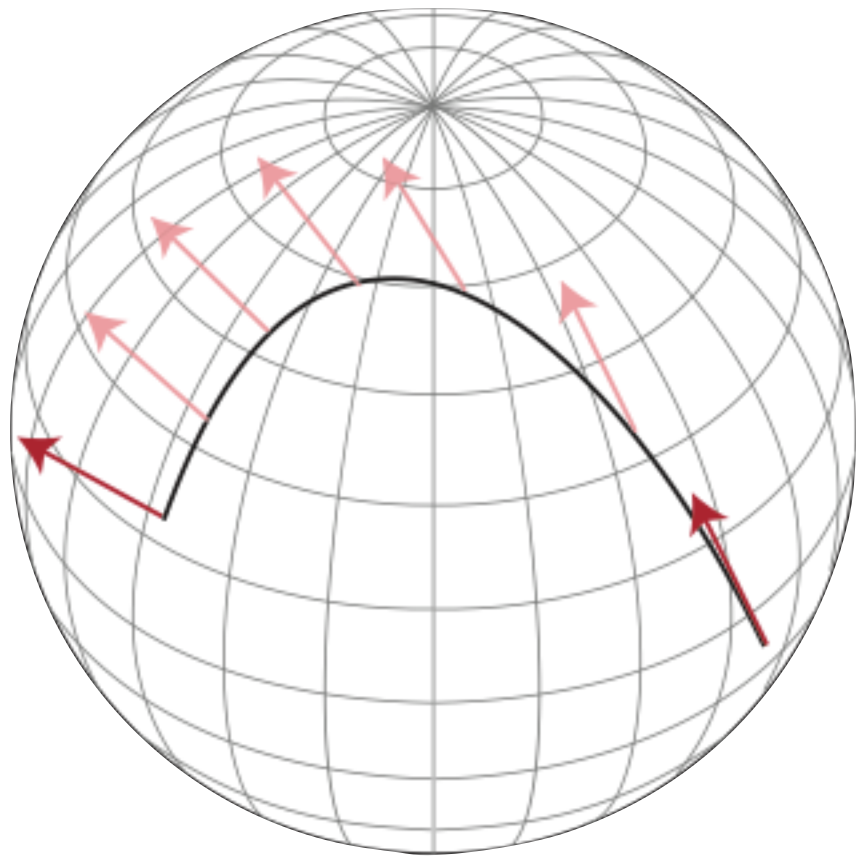
- **Affine Connection:**

$$\nabla_{\mathbf{w}}(f\mathbf{v}_1 + \mathbf{v}_2) = (\nabla_{\mathbf{w}} f)\mathbf{v}_1 + f\,\nabla_{\mathbf{w}}\mathbf{v}_1 + \nabla_{\mathbf{w}}\mathbf{v}_2$$
$$\nabla_{f\mathbf{w}_1+\mathbf{w}_2}\mathbf{v} = f\,\nabla_{\mathbf{w}_1}\mathbf{v} + \nabla_{\mathbf{w}_2}\mathbf{v}$$

- **Parallel Transport** of a vector $\mathbf{v}_0$ along a curve $c : [0,1] \to M$

$$\nabla_{\dot{c}(t)}\mathbf{v}(t) = 0$$

s.t. $\mathbf{v}(0) = \mathbf{v}_0$



Riemannian Metric and Levi-Civita Connection

Riemannian Metric and Levi-Civita Connection

- **Rienmannian metric** g = scalar product $\langle \cdot , \cdot \rangle_p$ on T_pM

Riemannian Metric and Levi-Civita Connection

- **Rienmannian metric** g = scalar product $\langle \cdot , \cdot \rangle_p$ on T_pM
- From g, we can measure the **distance between pair of points** on M, define geodesics, compute angles between vectors, between curves, define intrinsic quantities (e.g. **Gaussian curvature**), define **VF differential operators**…

Riemannian Metric and Levi-Civita Connection

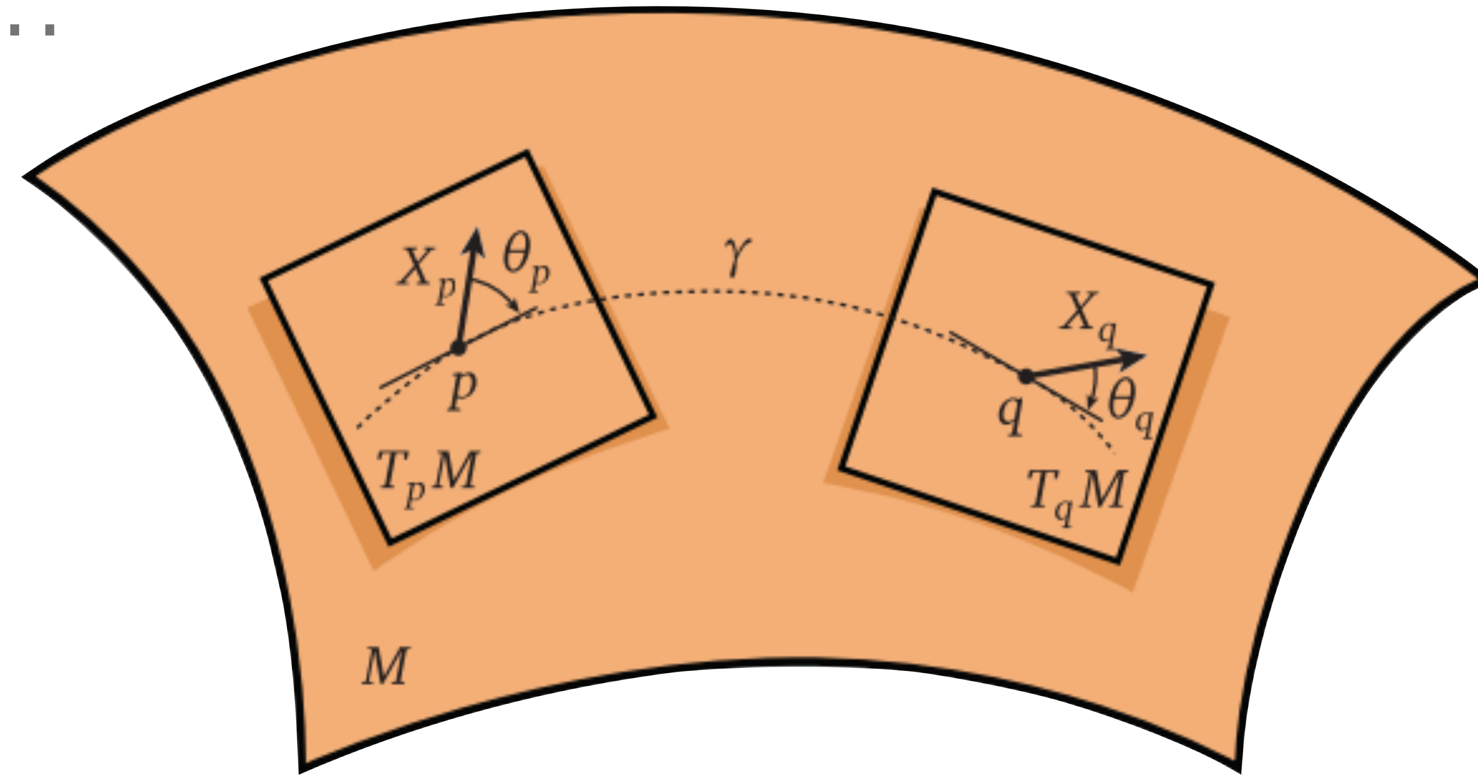
- **Rienmannian metric** g = scalar product $\langle \cdot , \cdot \rangle_p$ on T_pM

- From g, we can measure the **distance between pair of points** on M, define geodesics, compute angles between vectors, between curves, define intrinsic quantities (e.g. **Gaussian curvature**), define **VF differential operators**…

Riemannian Metric and Levi-Civita Connection

- **Rienmannian metric** g = scalar product $\langle \cdot , \cdot \rangle_p$ on T_pM

- From g, we can measure the **distance between pair of points** on M, define geodesics, compute angles between vectors, between curves, define intrinsic quantities (e.g. **Gaussian curvature**), define **VF differential operators**…

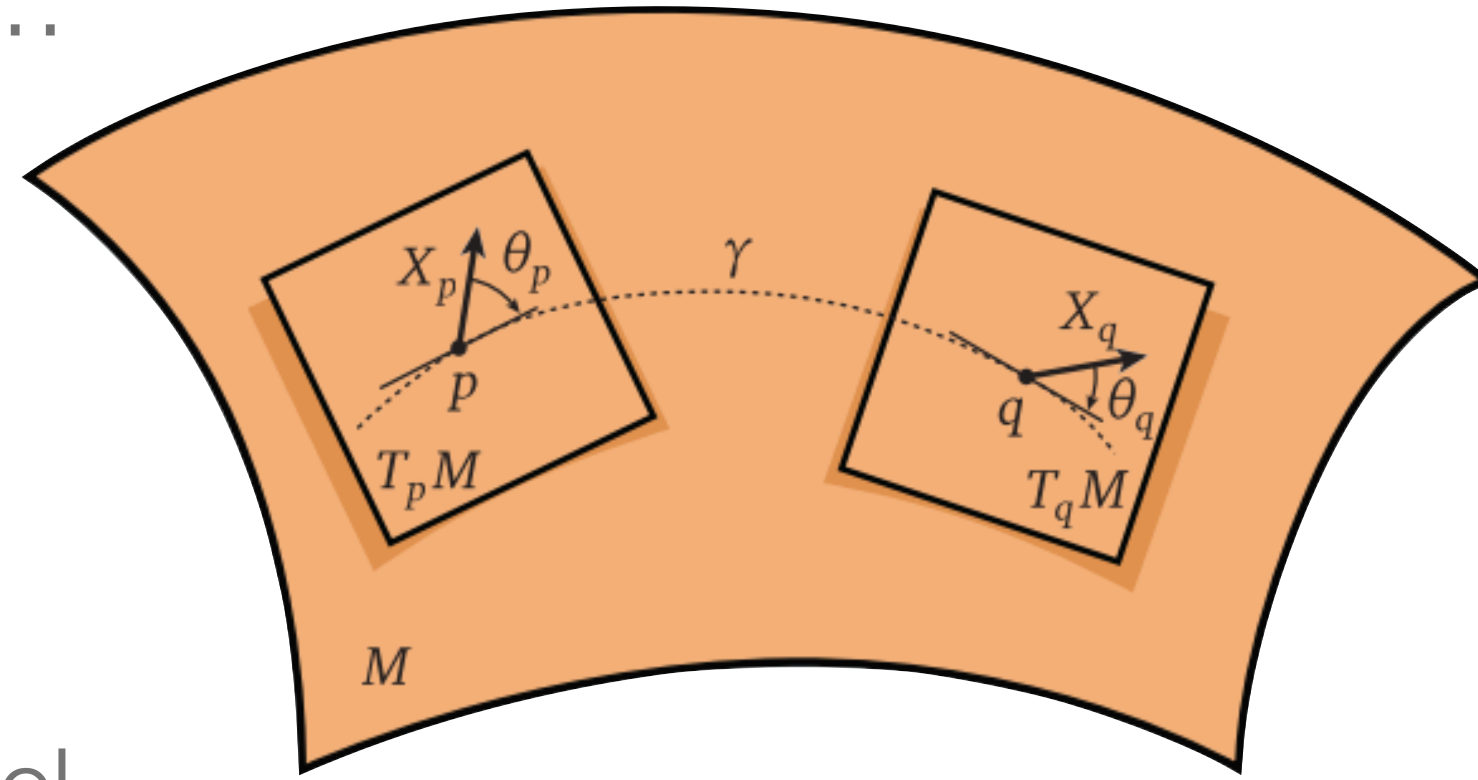


- **Levi-Civita Connection:** $\nabla_{\mathbf{u}} g(\mathbf{v}, \mathbf{w}) = g(\nabla_{\mathbf{u}} \mathbf{v}, \mathbf{w}) + g(\mathbf{v}, \nabla_{\mathbf{u}} \mathbf{w})$

Riemannian Metric and Levi-Civita Connection

- **Rienmannian metric** g = scalar product $\langle \cdot , \cdot \rangle_p$ on T_pM

- From g, we can measure the **distance between pair of points** on M, define geodesics, compute angles between vectors, between curves, define intrinsic quantities (e.g. **Gaussian curvature**), define **VF differential operators**…



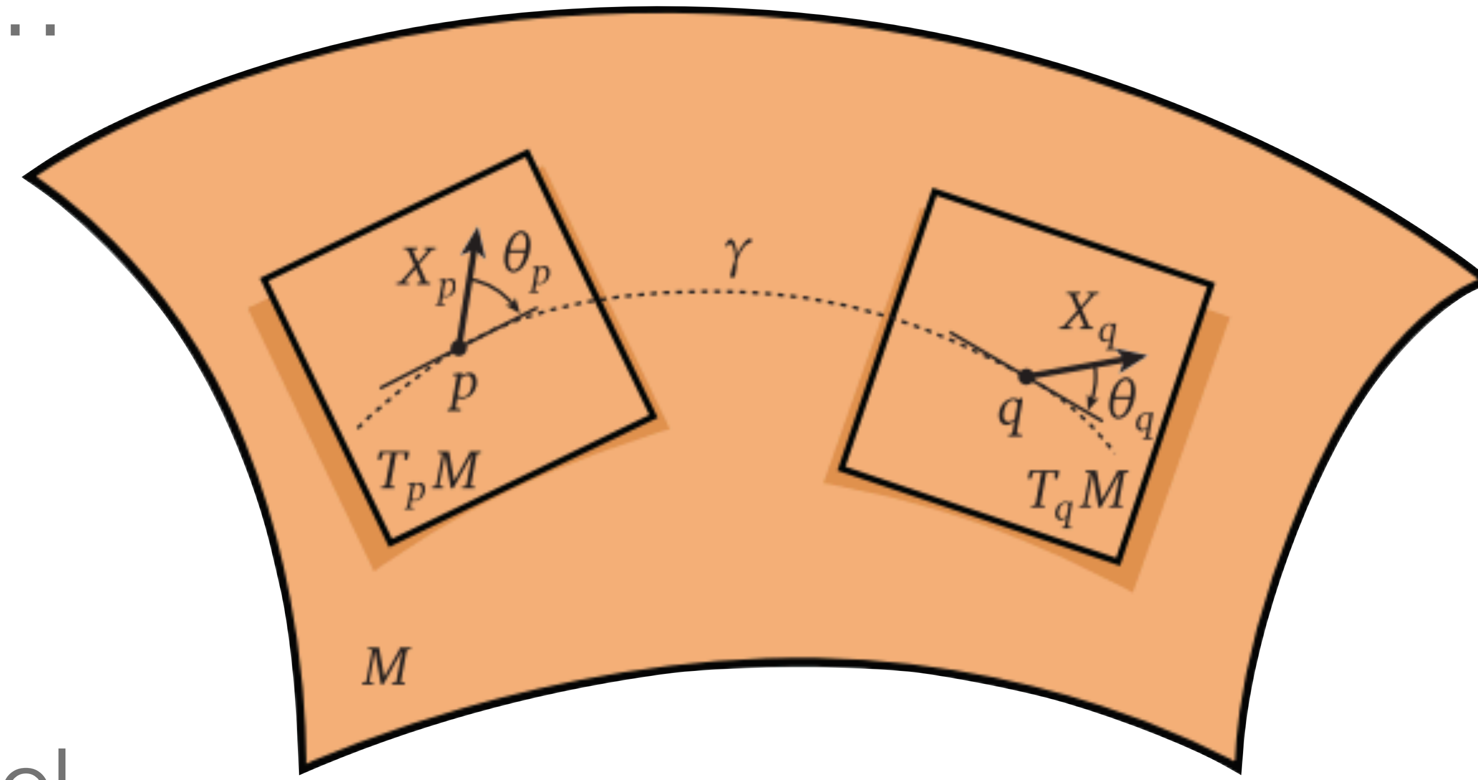
- **Levi-Civita Connection:** $\nabla_{\mathbf{u}} g(\mathbf{v}, \mathbf{w}) = g(\nabla_{\mathbf{u}} \mathbf{v}, \mathbf{w}) + g(\mathbf{v}, \nabla_{\mathbf{u}} \mathbf{w})$

- i.e. scalar product between pairs of vectors does not change when parallel-transported

Riemannian Metric and Levi-Civita Connection

- **Rienmannian metric** g = scalar product $\langle \cdot , \cdot \rangle_p$ on T_pM

- From g, we can measure the **distance between pair of points** on M, define geodesics, compute angles between vectors, between curves, define intrinsic quantities (e.g. **Gaussian curvature**), define **VF differential operators**…



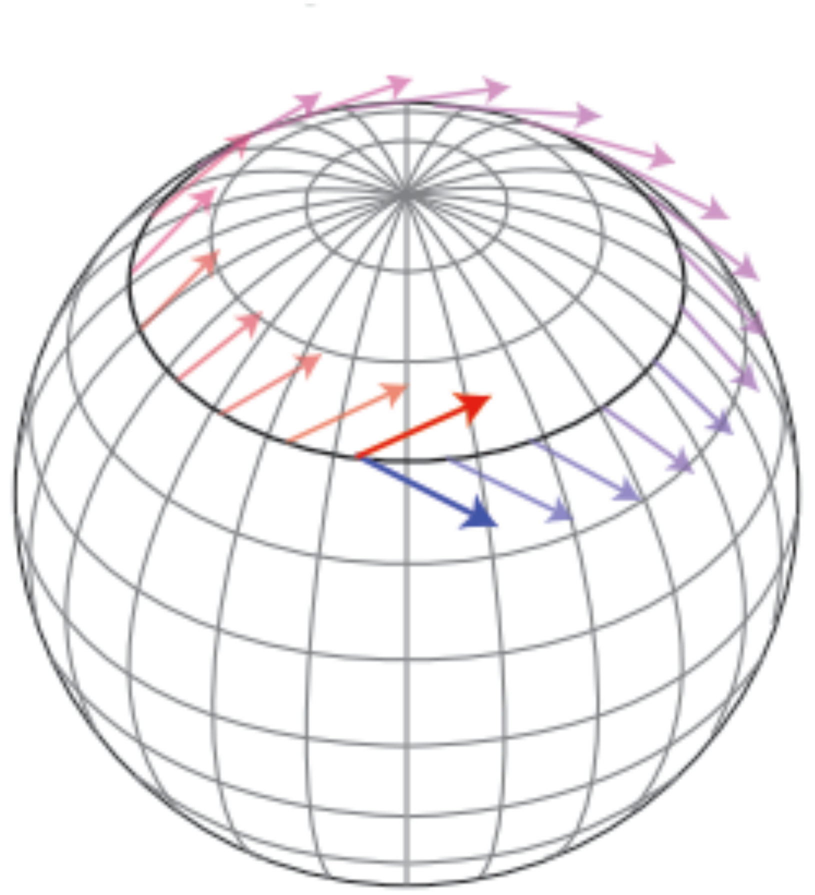
- **Levi-Civita Connection:** $\nabla_\mathbf{u} g(\mathbf{v}, \mathbf{w}) = g(\nabla_\mathbf{u} \mathbf{v}, \mathbf{w}) + g(\mathbf{v}, \nabla_\mathbf{u} \mathbf{w})$

- i.e. scalar product between pairs of vectors does not change when parallel-transported

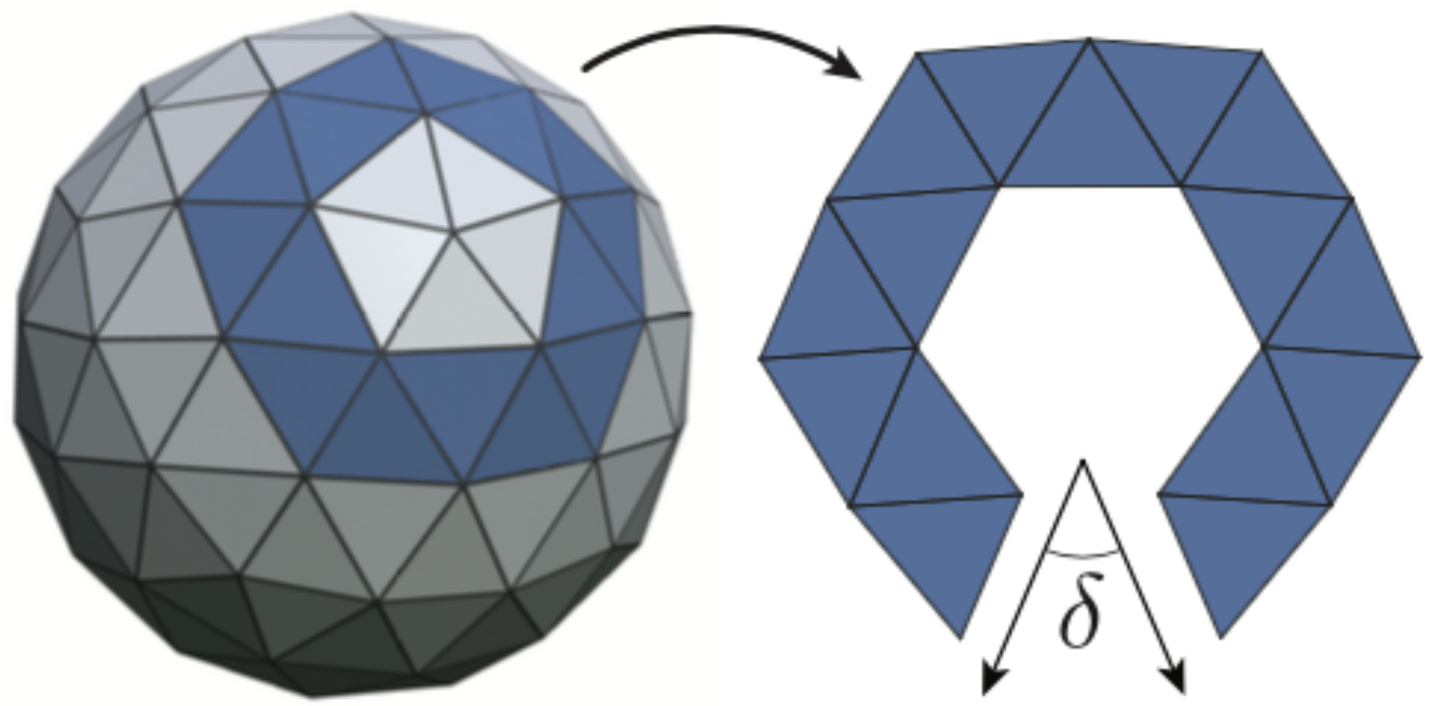
- linked to geodesic curvature of a curve on M

Holonomy

- **Holomony angle:** angle defect when parallel transporting a vector along a closed curve

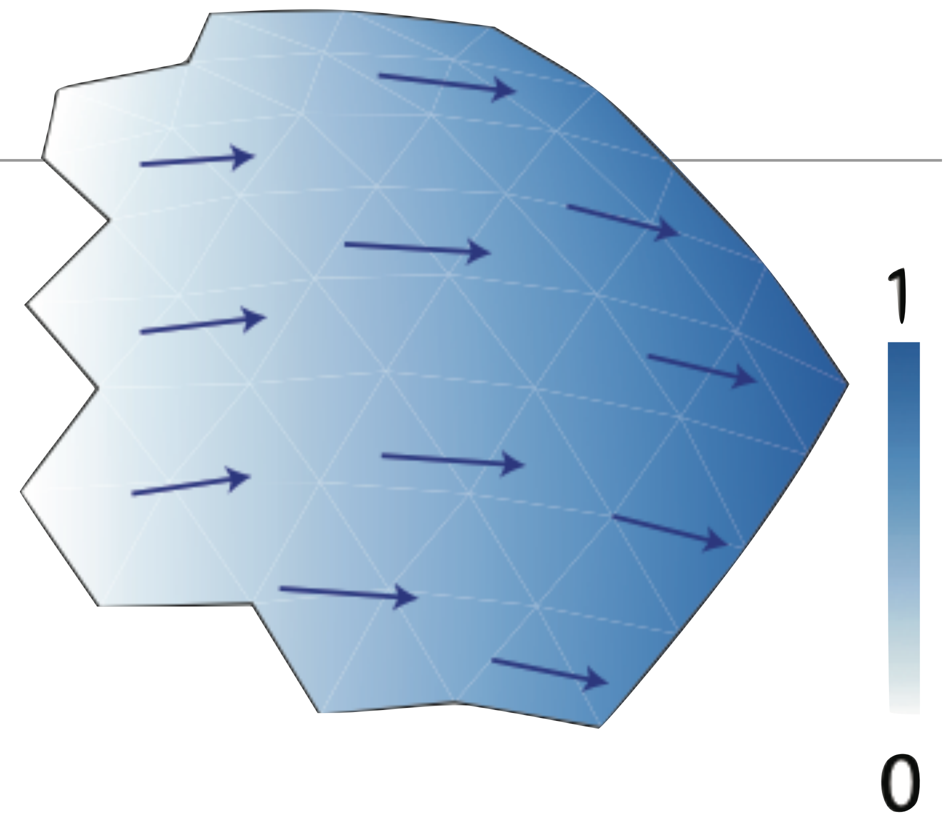


- **Using Levi-Citiva connection:** holomony $\equiv$ integral of the Gaussian curvature on the patch (mod 2π)



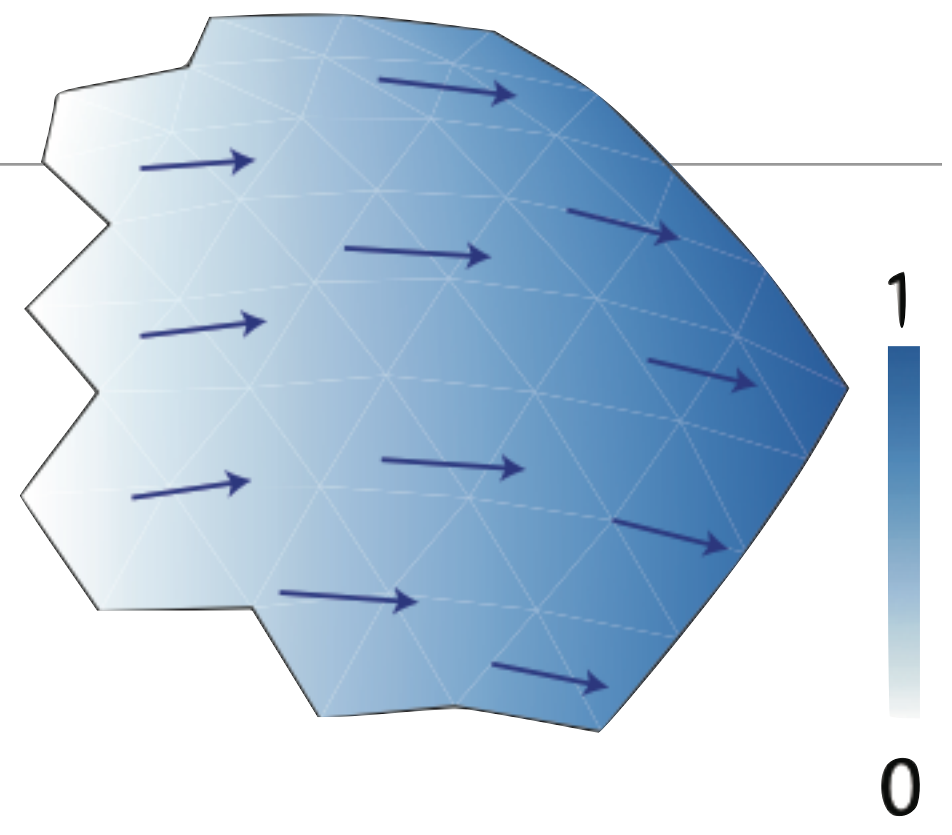
Gradient, divergence and curl

Gradient, divergence and curl



- **Gradient of a function** f: VF « $\text{grad}\, f$ » such that $\langle \text{grad}\, f, \mathbf{v} \rangle = \nabla_{\mathbf{v}} f\,,\ \forall \mathbf{v} \in TM$

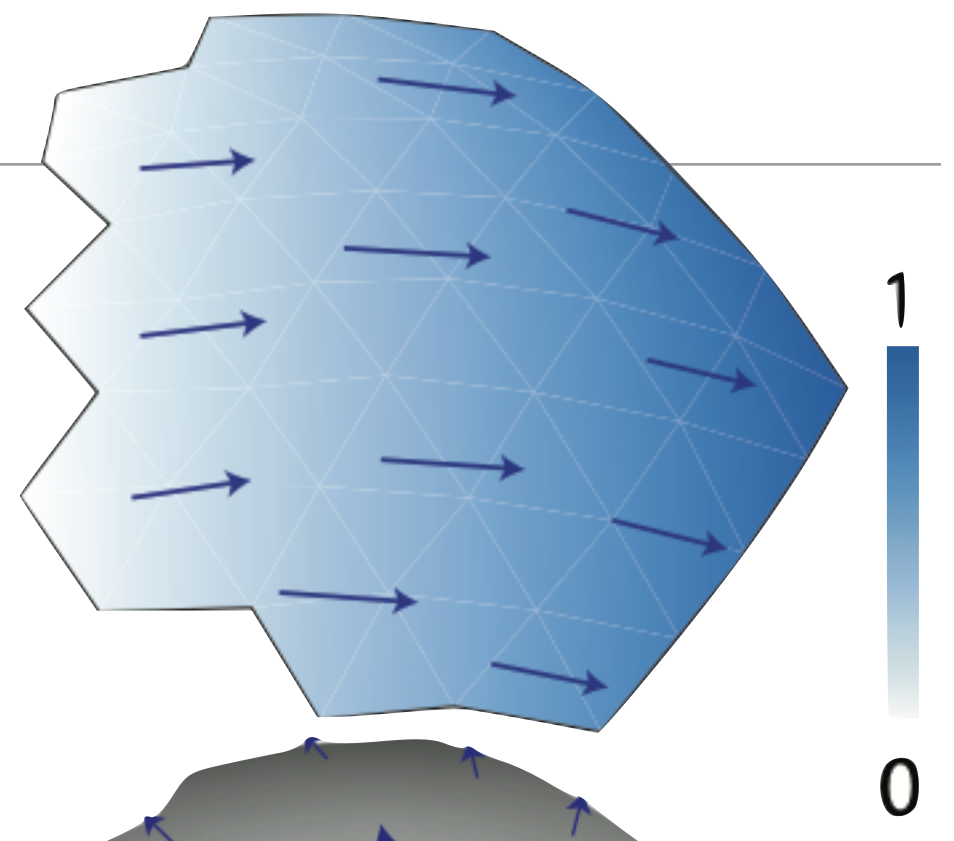
Gradient, divergence and curl



- **Gradient of a function** f: VF « $\mathrm{grad}\, f$ » such that $\langle \mathrm{grad}\, f, \mathbf{v} \rangle = \nabla_{\mathbf{v}} f\,,\ \forall \mathbf{v} \in TM$

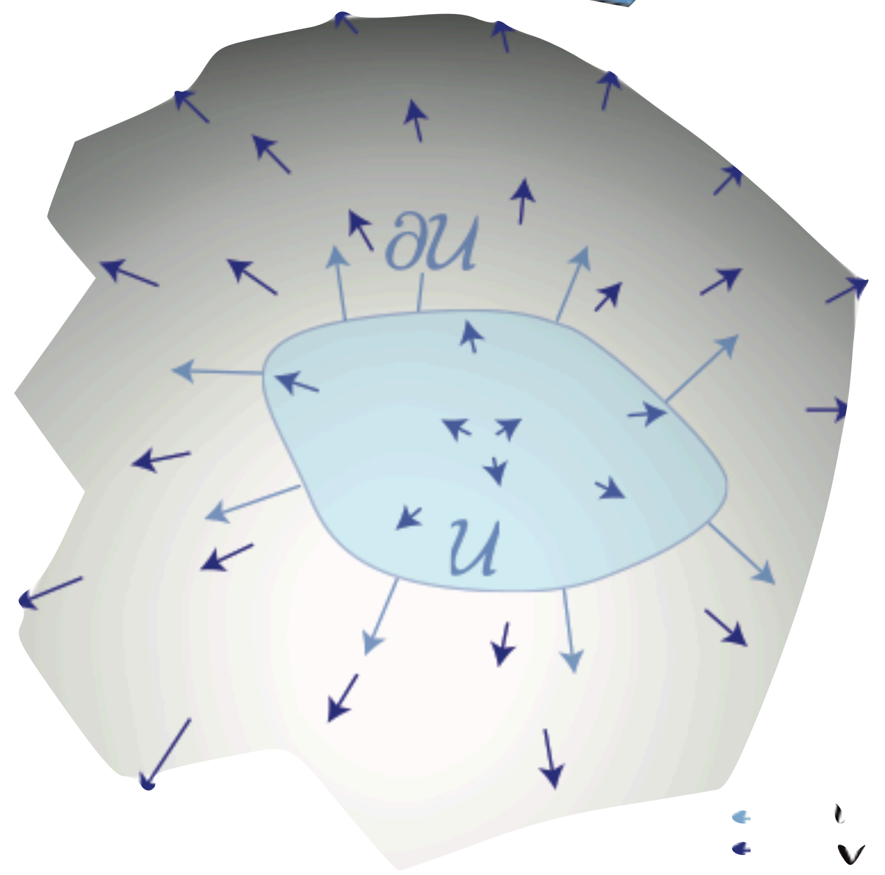
Gradient, divergence and curl

- **Gradient of a function** f: VF « $\mathrm{grad}\ f$ » such that
  $\langle \mathrm{grad}\ f, \mathbf{v} \rangle = \nabla_{\mathbf{v}} f\,,\ \forall \mathbf{v} \in TM$



- **Divergence:** $\mathrm{div}\,\mathbf{v}(p) = \sum_{i=1}^{d} \langle \nabla_{\mathbf{e}_i} \mathbf{v}(p), \mathbf{e}_i \rangle$

  ($\{\mathbf{e}_i\}$ an orthogonal basis of T_pM)



Gradient, divergence and curl

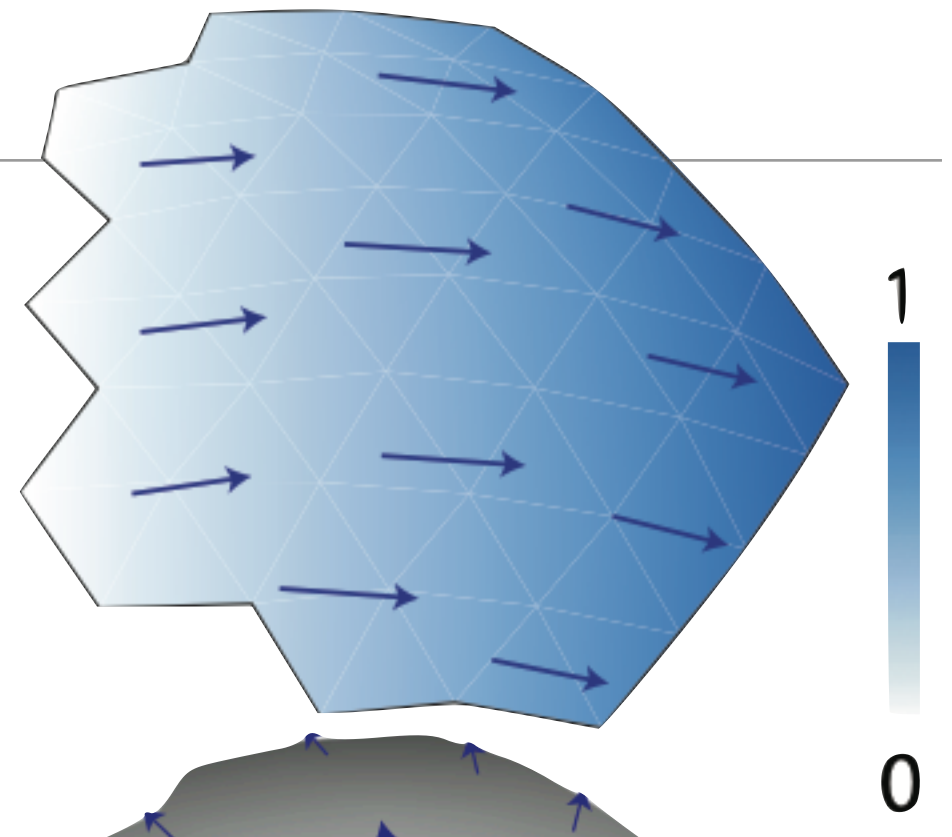
- **Gradient of a function** f: VF « $\mathrm{grad}\, f$ » such that
  $\langle \mathrm{grad}\, f, \mathbf{v} \rangle = \nabla_{\mathbf{v}} f\,,\ \forall \mathbf{v} \in TM$

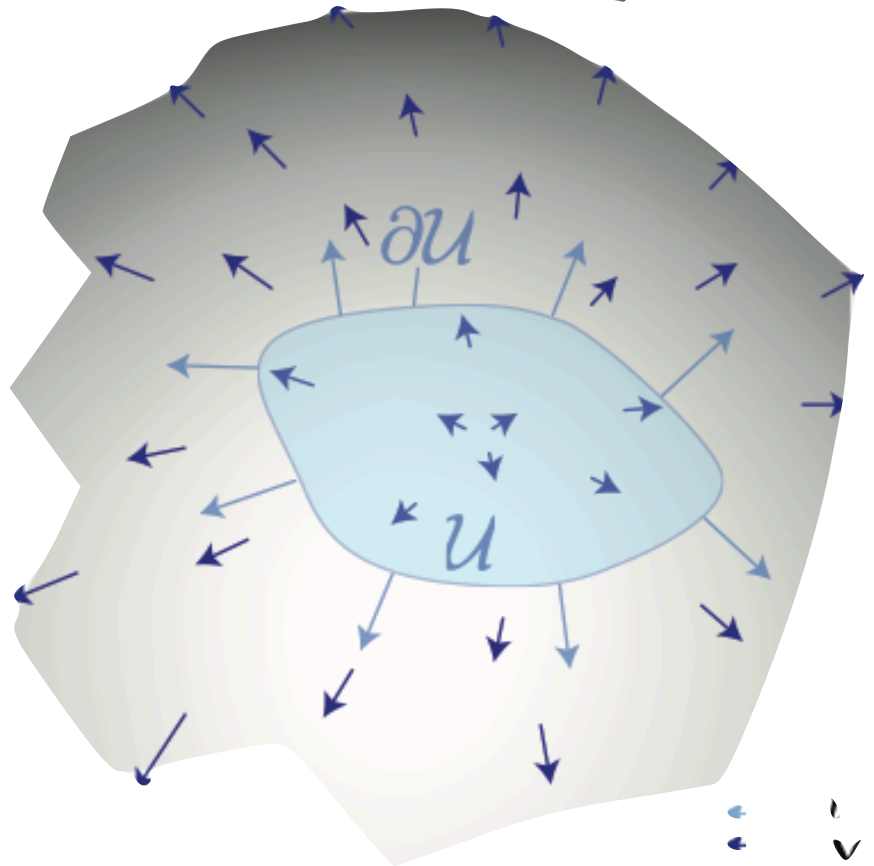
- **Divergence:** $\mathrm{div}\, \mathbf{v}(p) = \sum_{i=1}^{d} \langle \nabla_{\mathbf{e}_i} \mathbf{v}(p), \mathbf{e}_i \rangle$

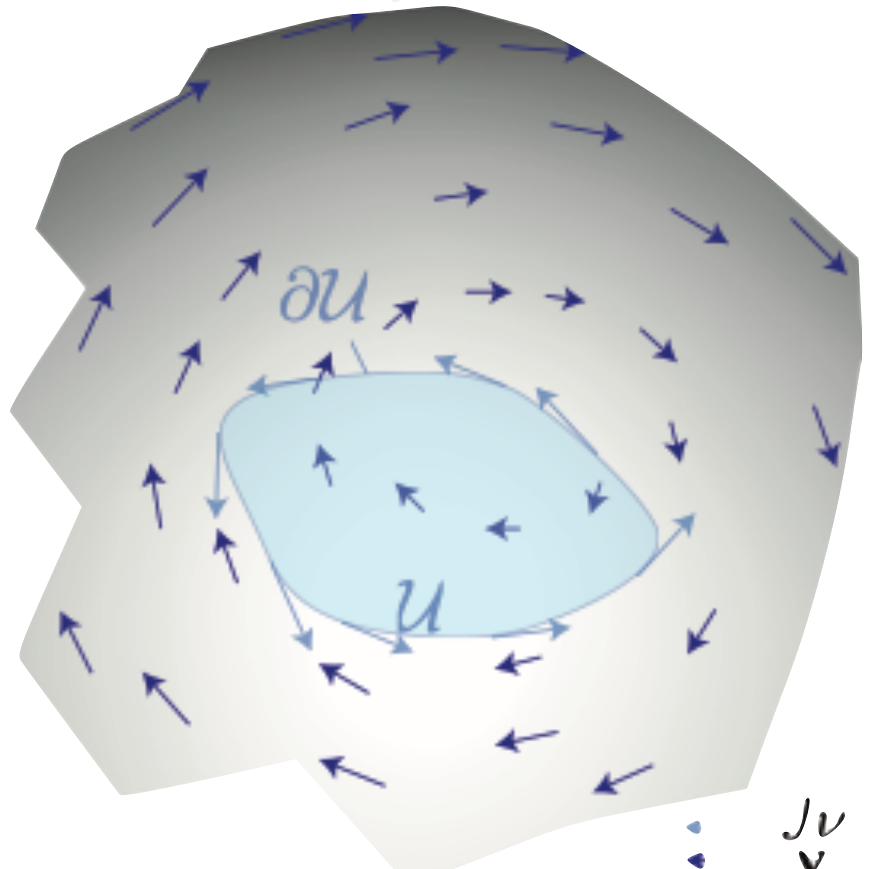
  ($\{\mathbf{e}_i\}$ an orthogonal basis of T_pM)

- **Curl:** $\mathrm{curl}\, \mathbf{v} = -\,\mathrm{div}\, \mathrm{J}\, \mathbf{v}$

  $\mathrm{J} : \mathbf{v} \longmapsto \mathbf{N} \times \mathbf{v}$

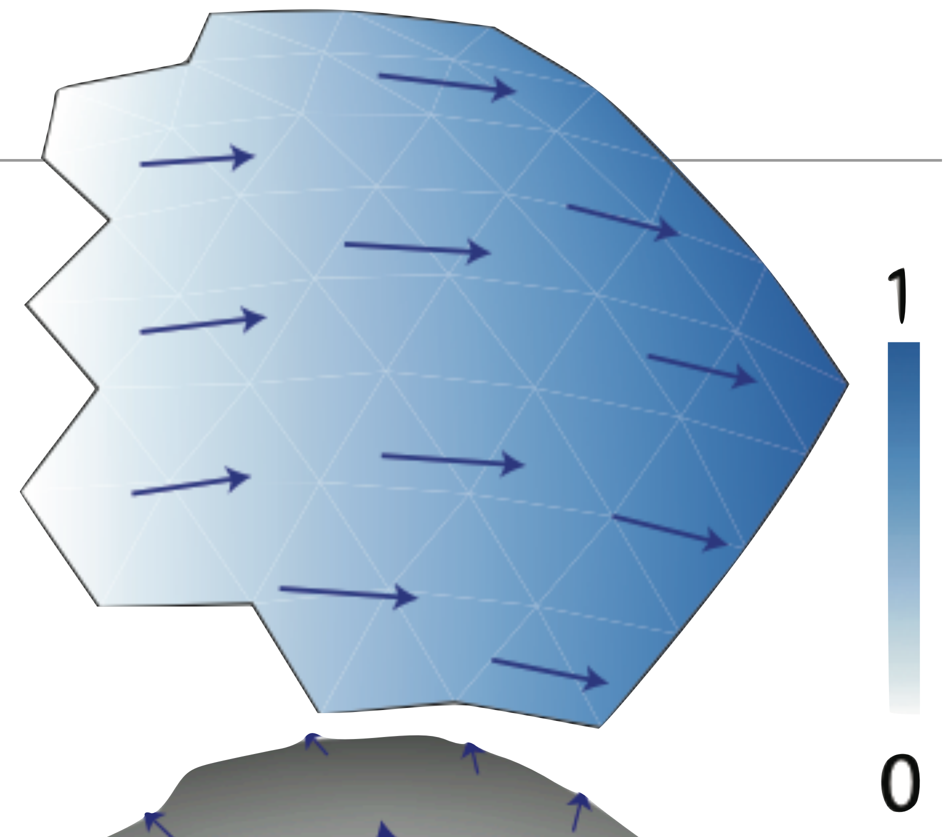






Gradient, divergence and curl

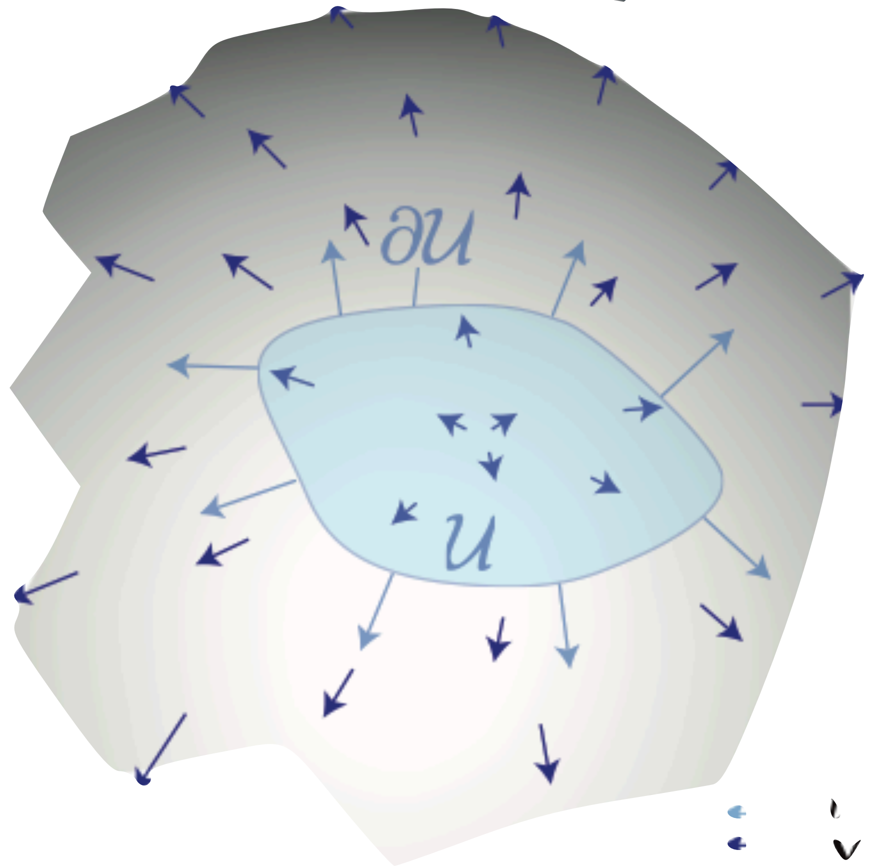
- **Gradient of a function** f: VF « $\mathrm{grad}\, f$ » such that
  $\langle \mathrm{grad}\, f, \mathbf{v} \rangle = \nabla_{\mathbf{v}} f\,,\ \forall \mathbf{v} \in TM$



- **Divergence:** $\mathrm{div}\, \mathbf{v}(p) = \sum_{i=1}^{d} \langle \nabla_{\mathbf{e}_i} \mathbf{v}(p), \mathbf{e}_i \rangle$

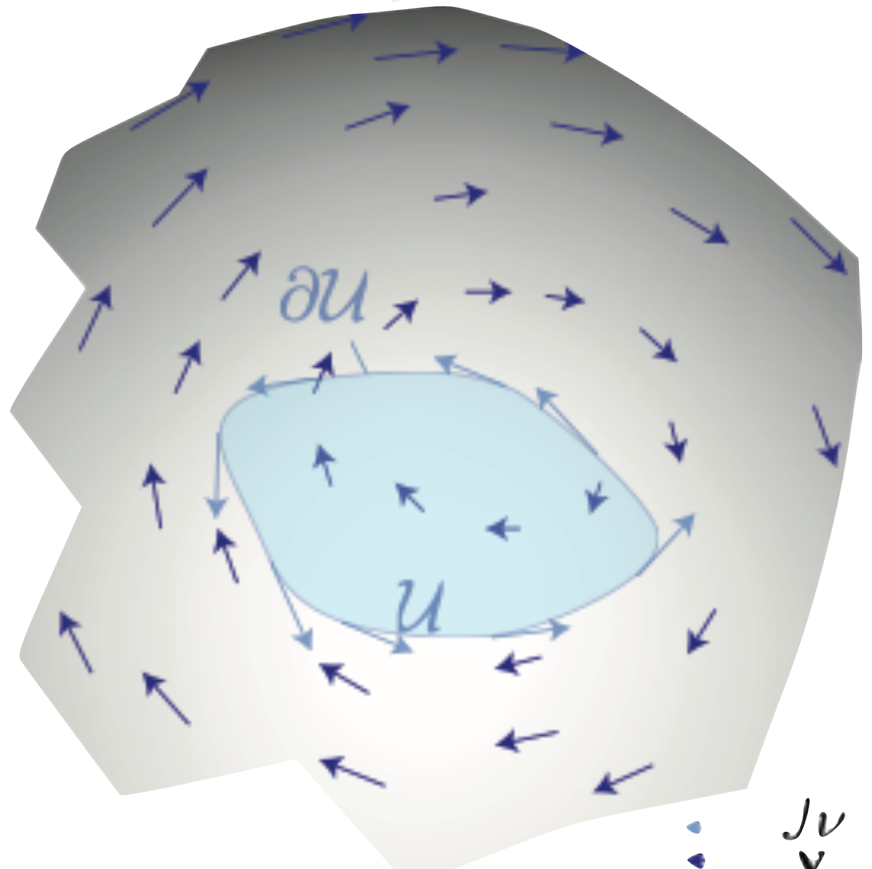
  Sinks and sources

  $(\{\mathbf{e}_i\}$ an orthogonal basis of $T_pM)$

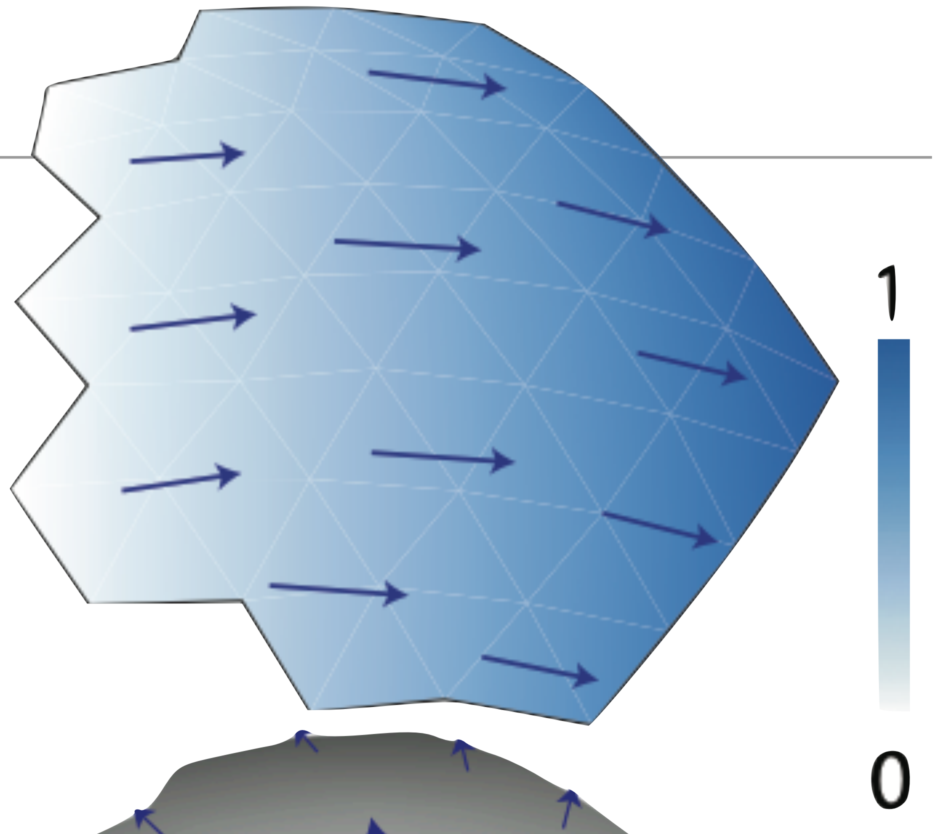


- **Curl:** $\mathrm{curl}\, \mathbf{v} = -\,\mathrm{div}\, \mathrm{J}\, \mathbf{v}$

  $\mathrm{J} : \mathbf{v} \longmapsto \mathbf{N} \times \mathbf{v}$



Gradient, divergence and curl

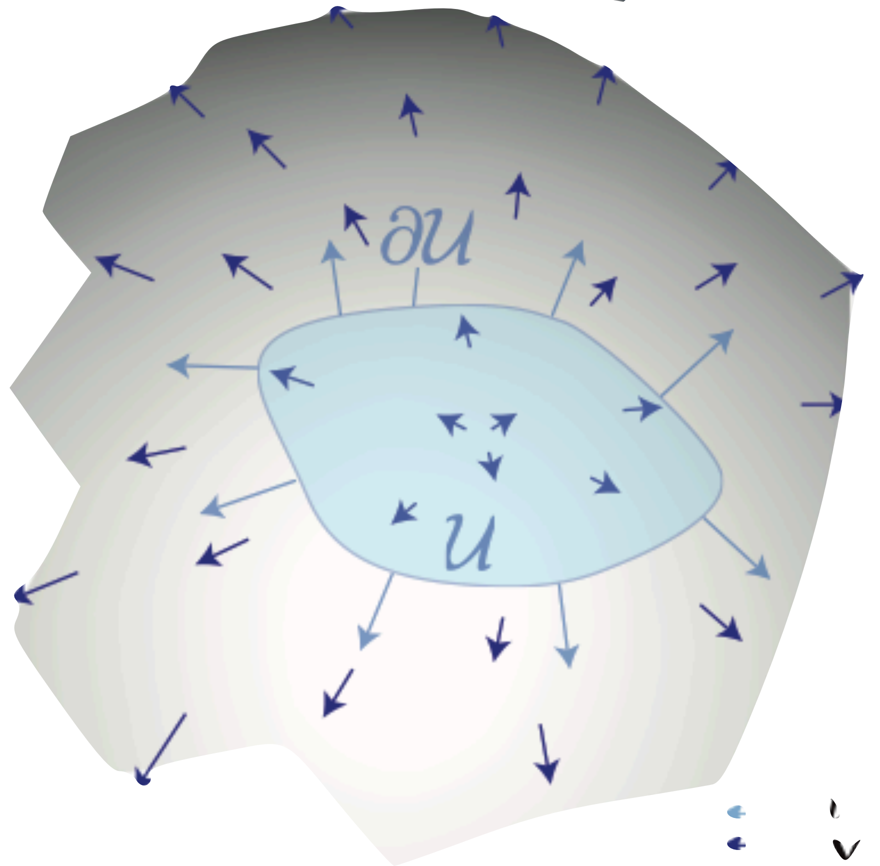


- **Gradient of a function** f: VF « $\mathrm{grad}\, f$ » such that
  $\langle \mathrm{grad}\, f, \mathbf{v} \rangle = \nabla_{\mathbf{v}} f\,,\ \forall \mathbf{v} \in TM$

- **Divergence:** $\mathrm{div}\, \mathbf{v}(p) = \sum_{i=1}^{d} \langle \nabla_{\mathbf{e}_i} \mathbf{v}(p), \mathbf{e}_i \rangle$

  ($\{\mathbf{e}_i\}$ an orthogonal basis of T_pM)

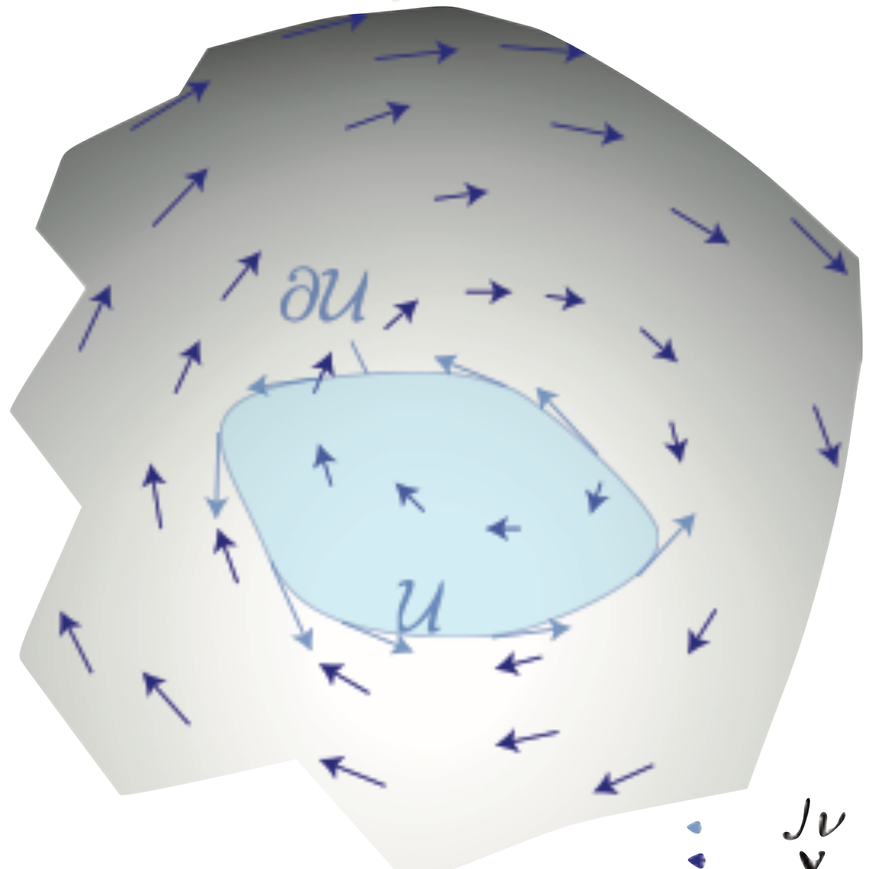
  Sinks and sources



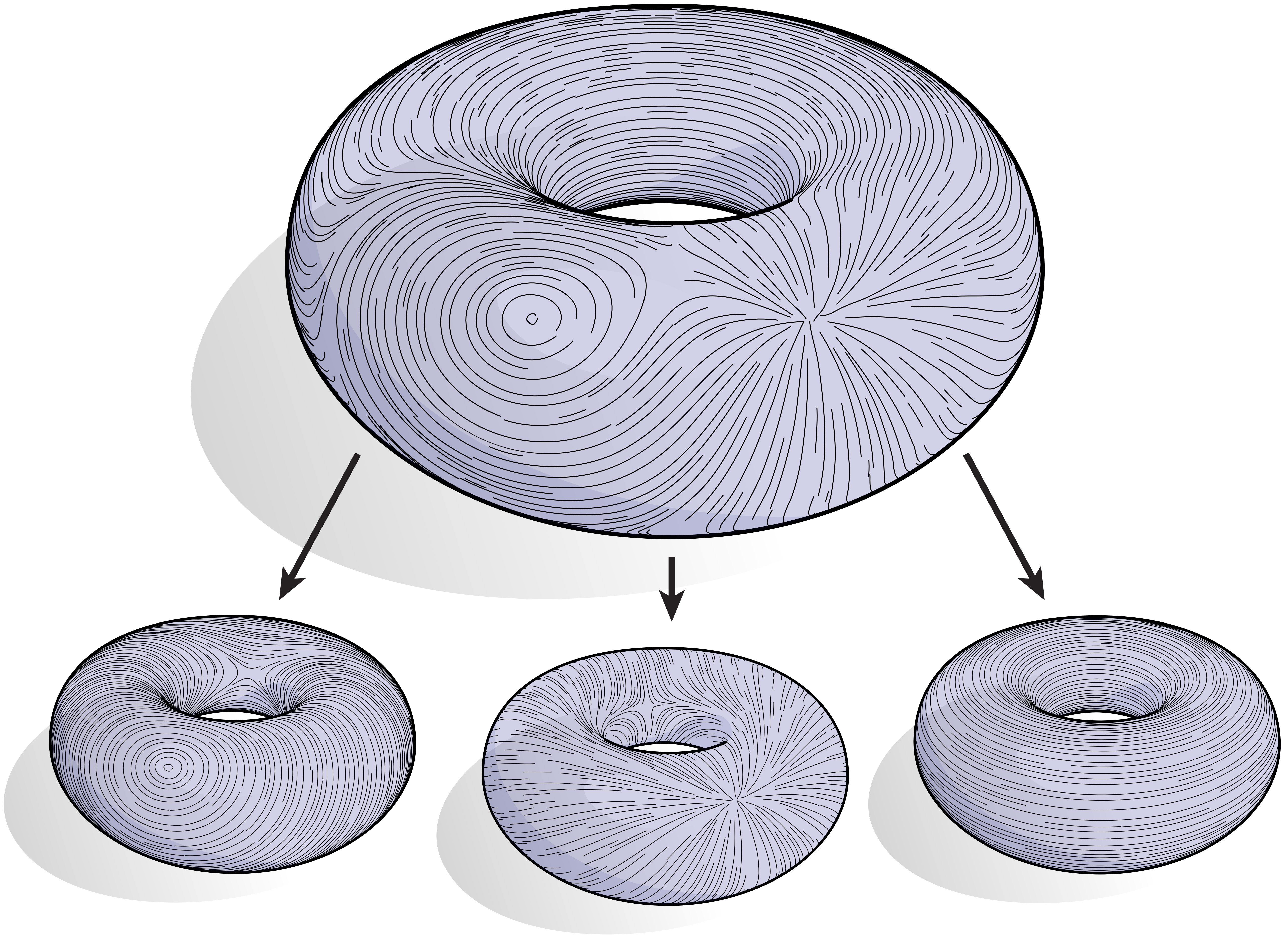
- **Curl:** $\mathrm{curl}\, \mathbf{v} = -\,\mathrm{div}\, \mathrm{J}\, \mathbf{v}$

  $\mathrm{J} : \mathbf{v} \longmapsto \mathbf{N} \times \mathbf{v}$

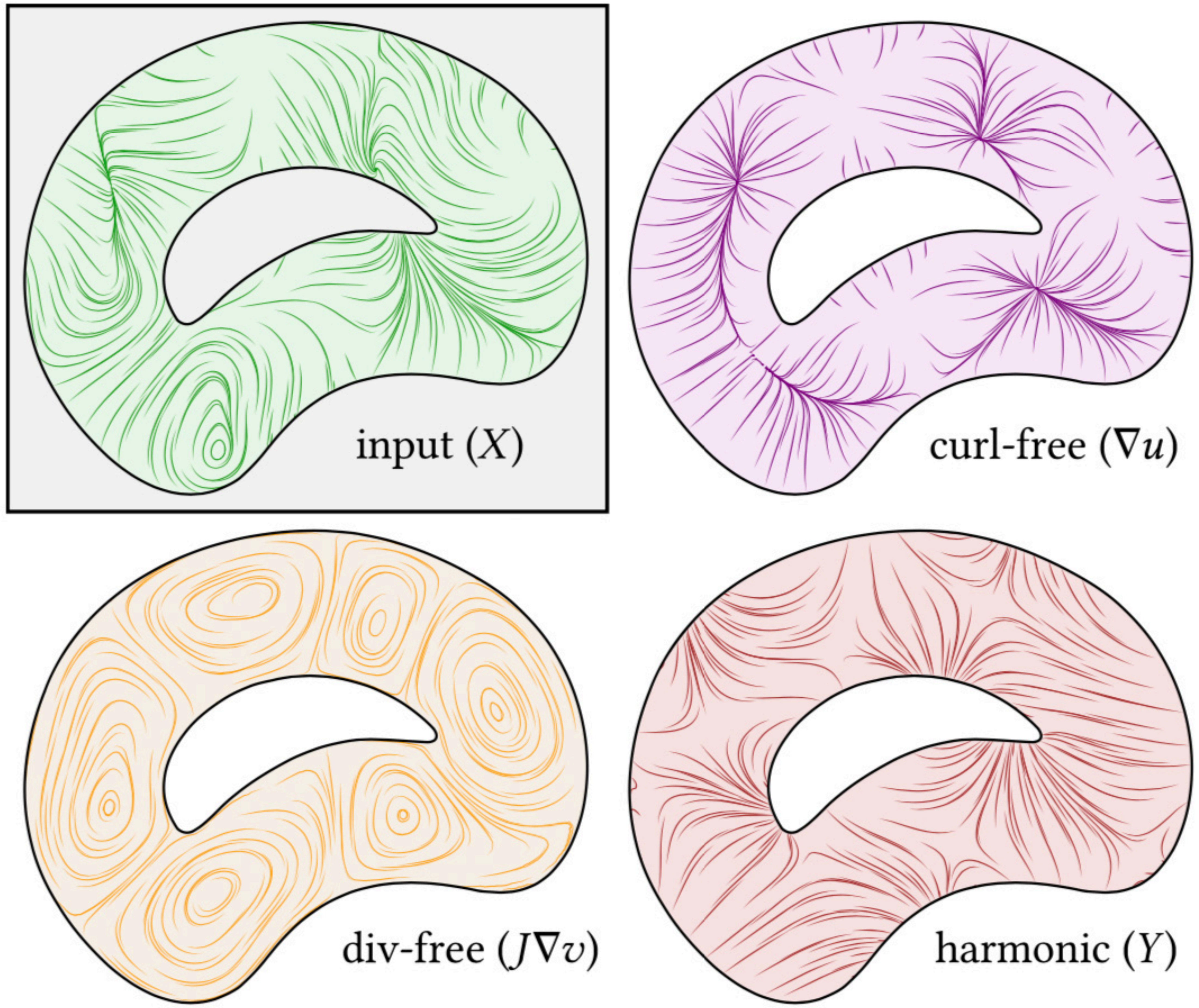
  rotation around a point



Helmholtz-Hodge decomposition

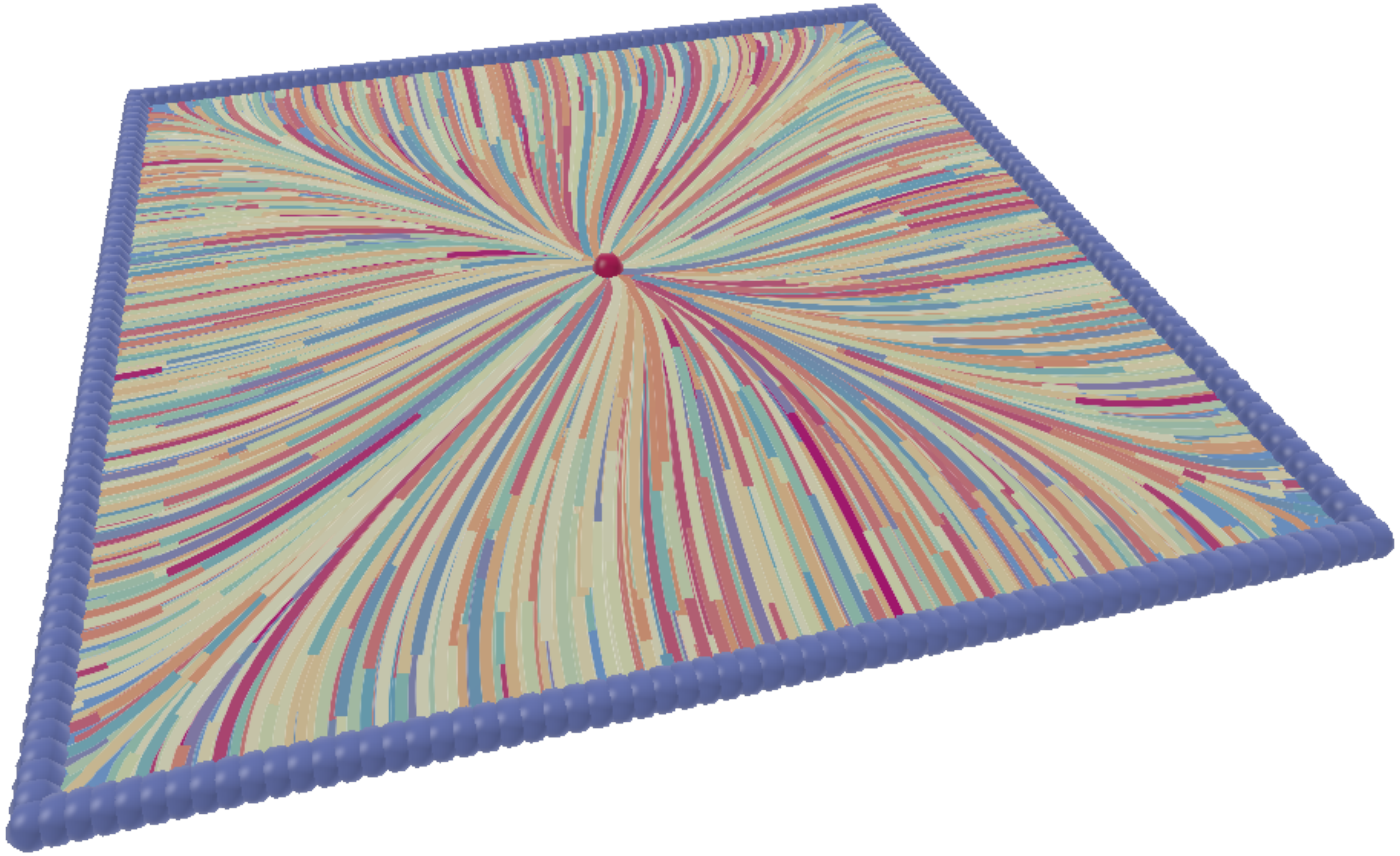


Helmholtz-Hodge decomposition

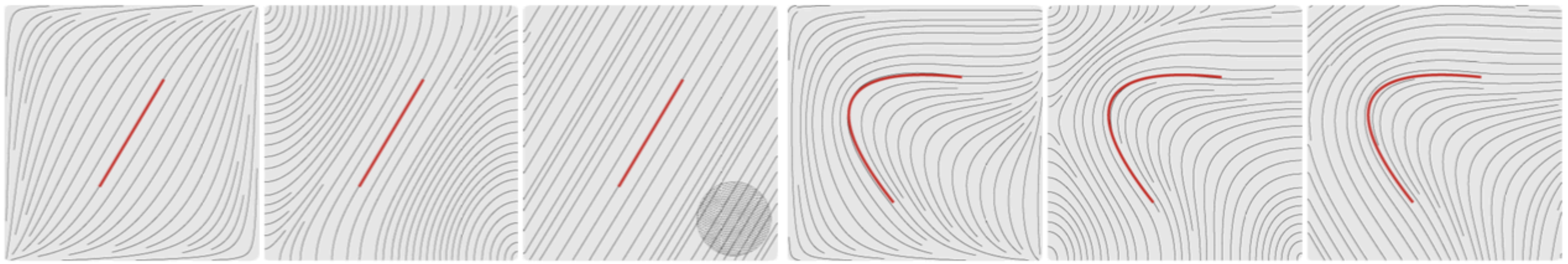


$$\mathcal{X} = \text{Image(grad)} \oplus \text{Image(J grad)} \oplus \mathcal{H}$$

Fields on patches

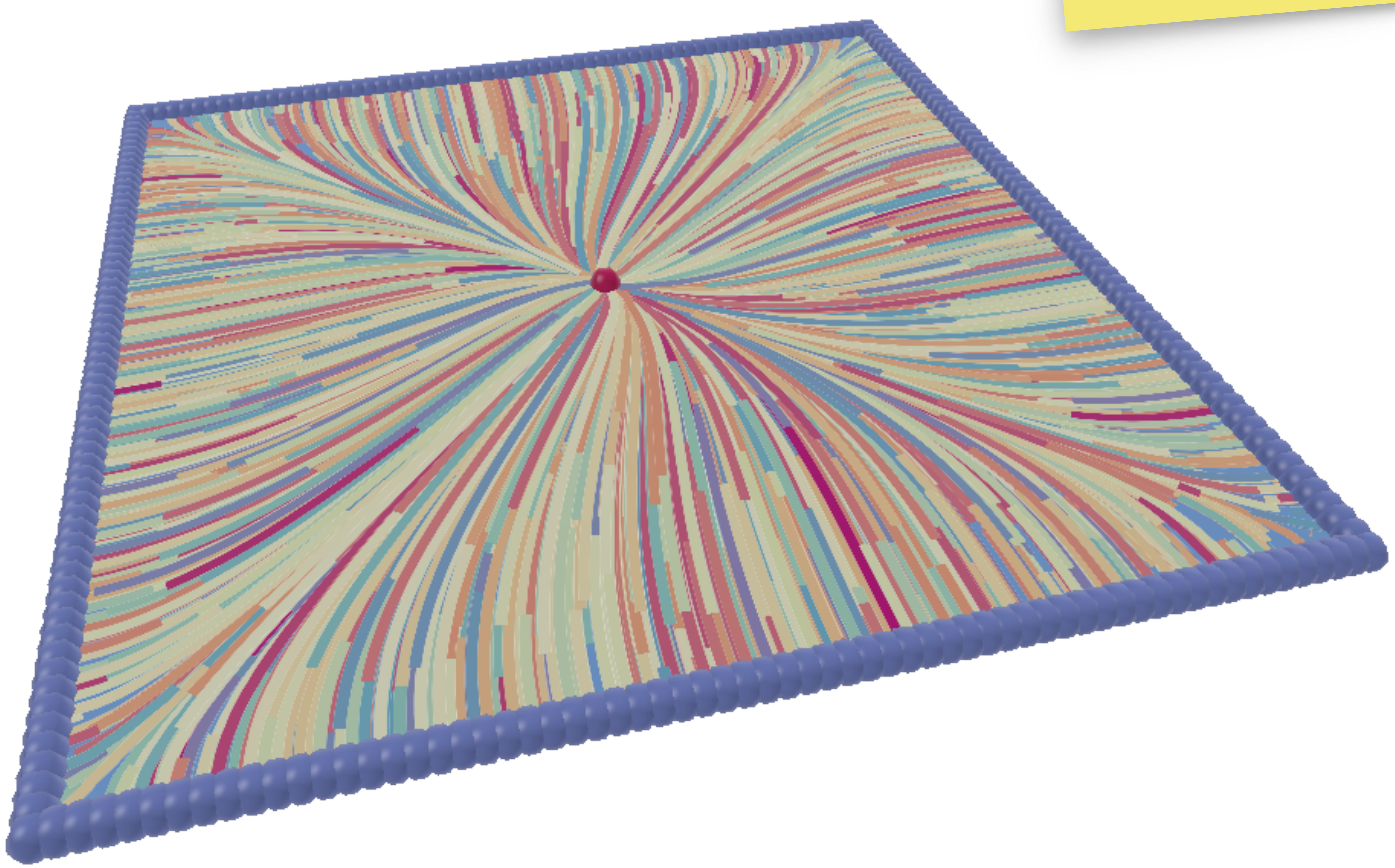


⚠ boundary conditions

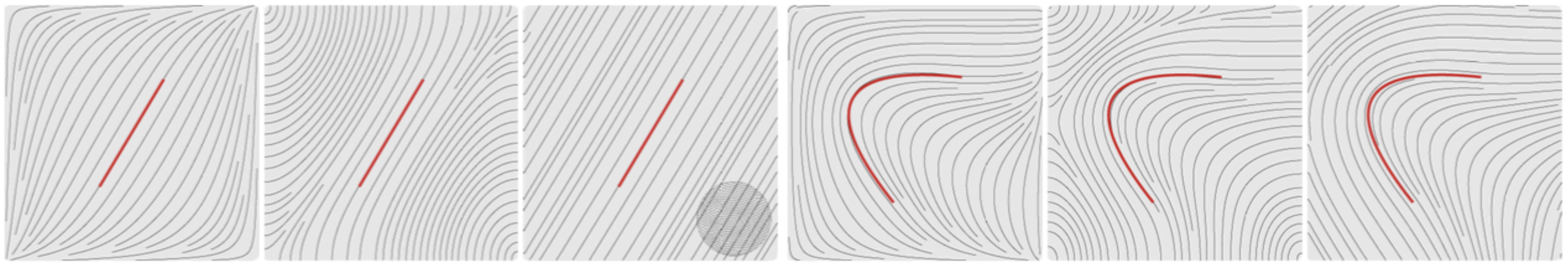


Fields on patches

Bounary points ≅ singularities or hard constraints



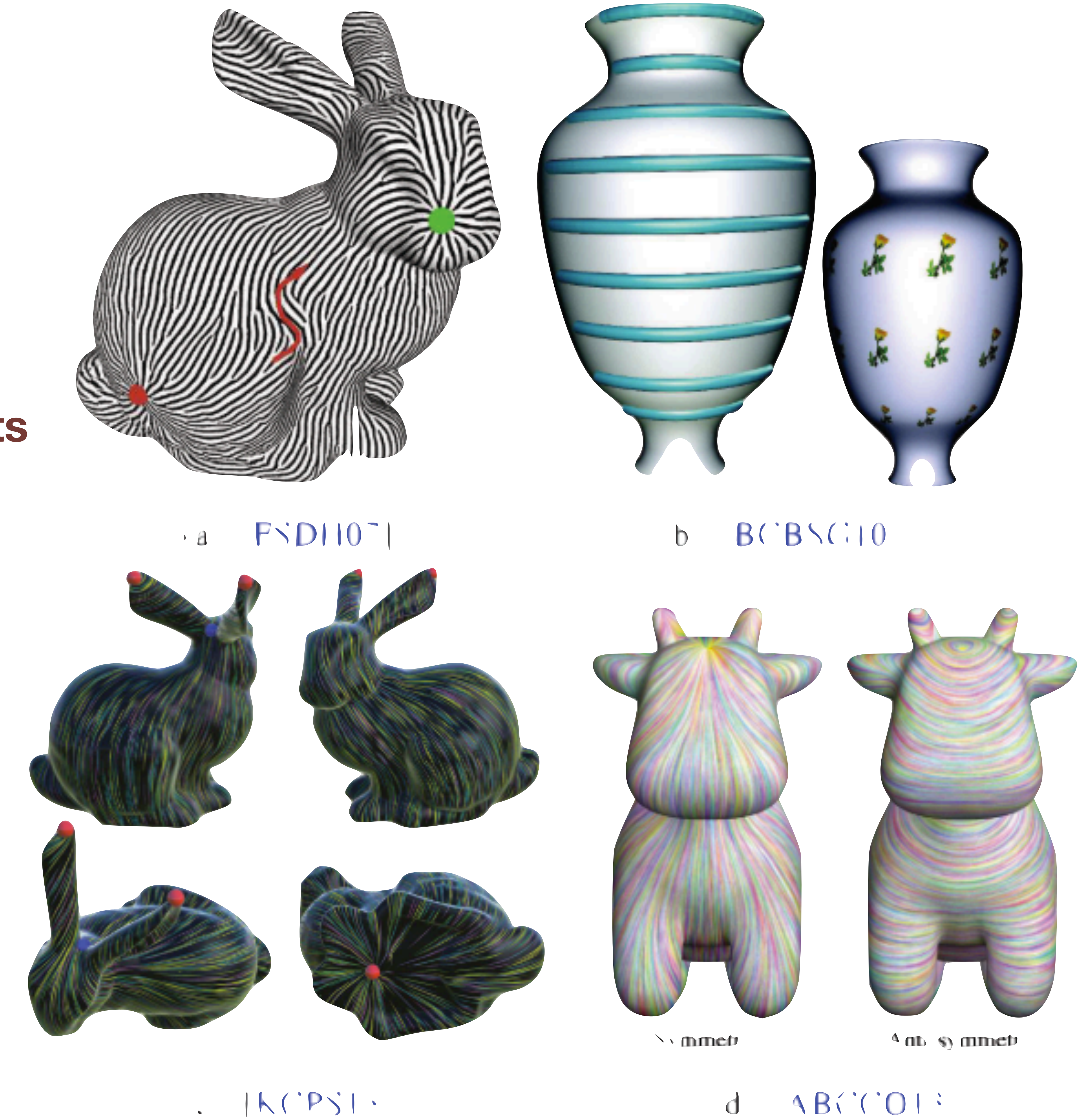
⚠ boundary conditions



Fairness of a vector field

- How to evaluate / control the smoothness of a VF?

Alignment to constraints



Killing energy (self-isometries)

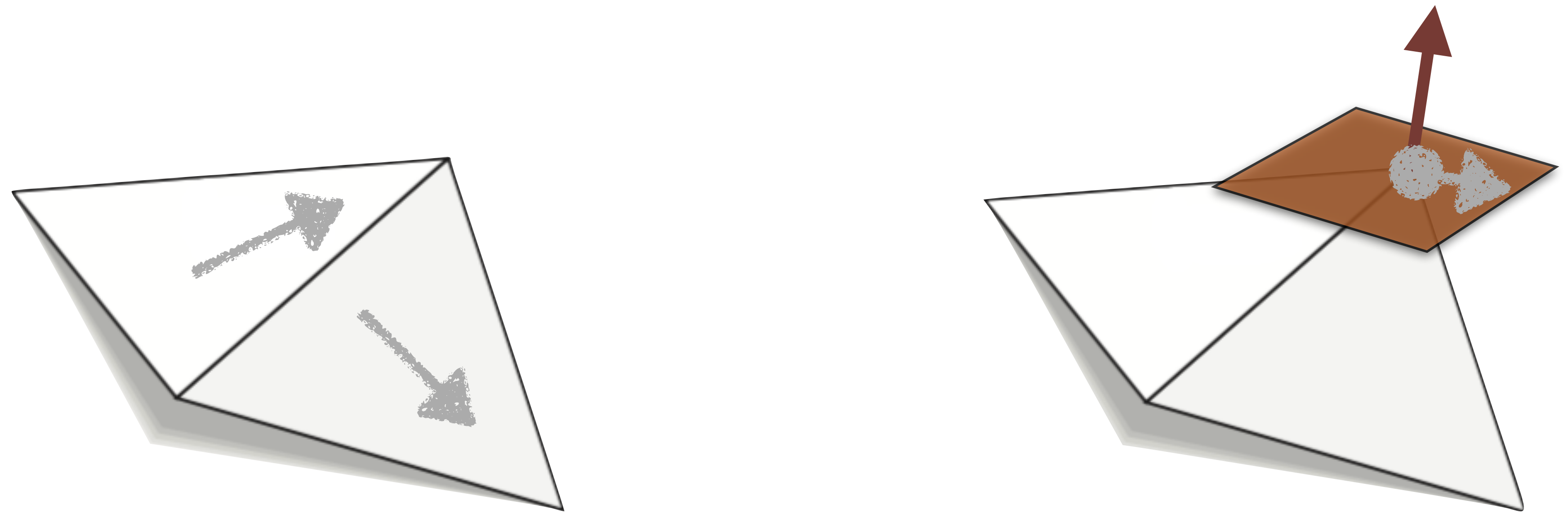
Dirichlet energy

Symmetry/antisymmetry constraints

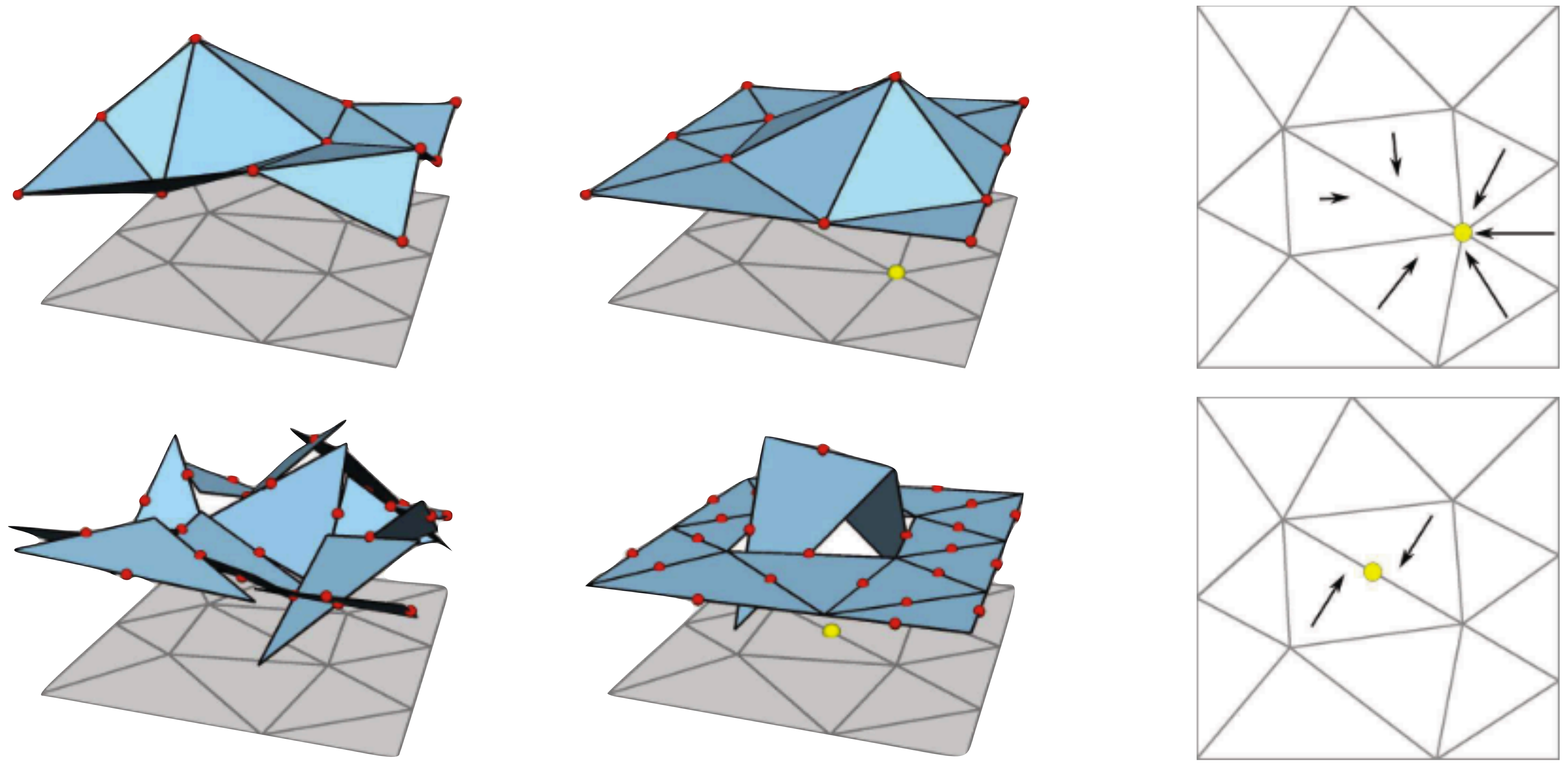
Computing smooth intrinsic n-RoSy fields

Discretization issues

- Piecewise constant VF per face vs linearly interpolated VF



- Closely related to space of scalar functions

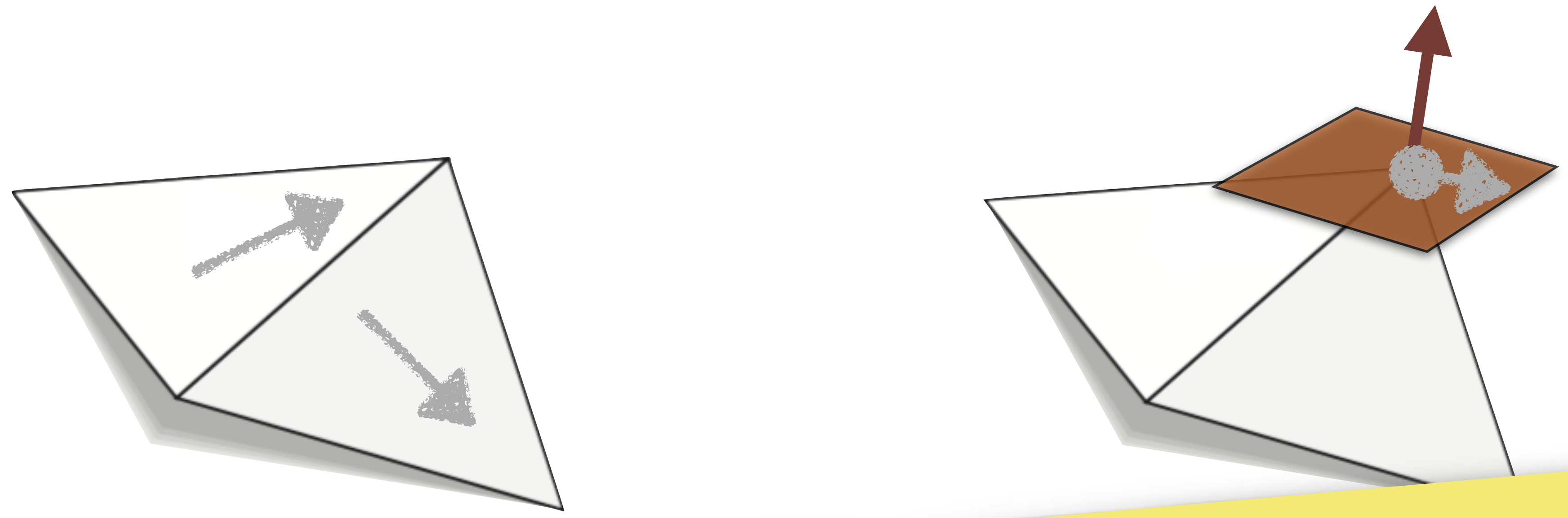


Lagrange elements

Crouzeix-Raviart elements

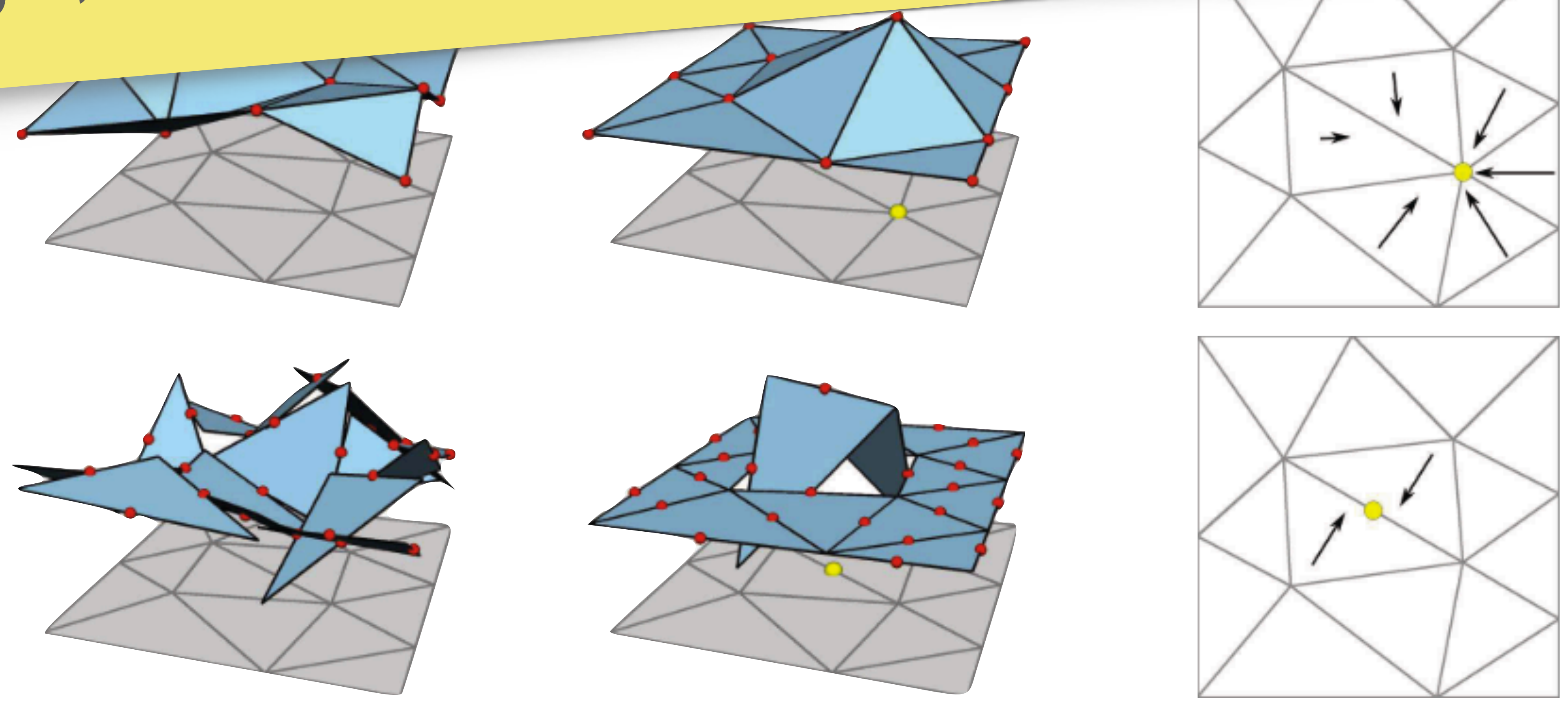
Discretization issues

- Piecewise constant VF per face vs linearly interpolated VF



- Closely related to

Computational efficiency, Discrete Helmotz-Hodge, implicit/explicit singularities…



Lagrange elements

Crouzeix-Raviart elements

How to compute discrete N-RoSy fields ?

TL;DR: unknowns are one angle per face, big linear operator (matrix) to encode the geometry and topology, minimizing energy ⇔ solve linear system or eigendecomposition problem.

Globally Optimal Direction Fields

Felix Knöppel, TU Berlin
Keenan Crane, Caltech
Ulrich Pinkall, TU Berlin
Peter Schröder, Caltech

Eurographics Sympos
Olga Sorkine and Bru
(Guest Editors)

Trivial Connections on Discrete Surfaces

Keenan Crane[1] and Mathieu Desbrun[1] and Peter Schröder[1,2]

Spectral Processing of Tangential Vector Fields

Christopher Brandt, Leonardo Scandolo, Elmar Eisemann and Klaus Hildebrandt

Delft University of Technology, Electrical Engineering, Mathematics and Computer Science, The Netherlands
{C.Brandt, L.Scandolo, E.Eisemann, K.A.Hildebrandt}@tudelft.nl

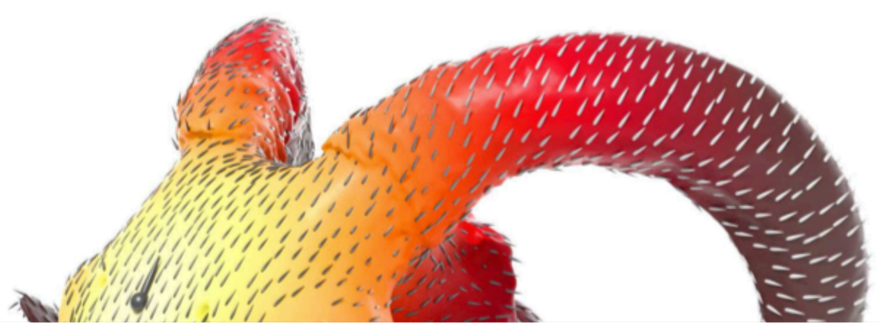
N-Symmetry Direction Field Design

Nicolas Ray
Bruno Vallet
Wan Chiu Li
Bruno Lévy
INRIA - ALICE

The Vector Heat Method

NICHOLAS SHARP, YOUSUF SOLIMAN, and KEENAN CRANE, Carnegie Mellon University

This paper describes a method for efficiently computing parallel transport of tangent vectors on curved surfaces, or more generally, any vector-valued data on a curved manifold. More precisely, it extends a vector field defined over any region to the rest of the domain via parallel transport along shortest geodesics. This basic operation enables fast, robust algorithms for extrapolating level set velocities, inverting the exponential map, computing geometric medians and Karcher/Fréchet means of arbitrary distributions, constructing centroidal Voronoi diagrams, and finding consistently ordered landmarks.



Two algorithms:

- one focusing on holonomy
- one based on a PDE approach

Eurographics Symposium on Geometry Processing 2010
Olga Sorkine and Bruno Lévy
(Guest Editors)

Volume 29 (2010), Number 5

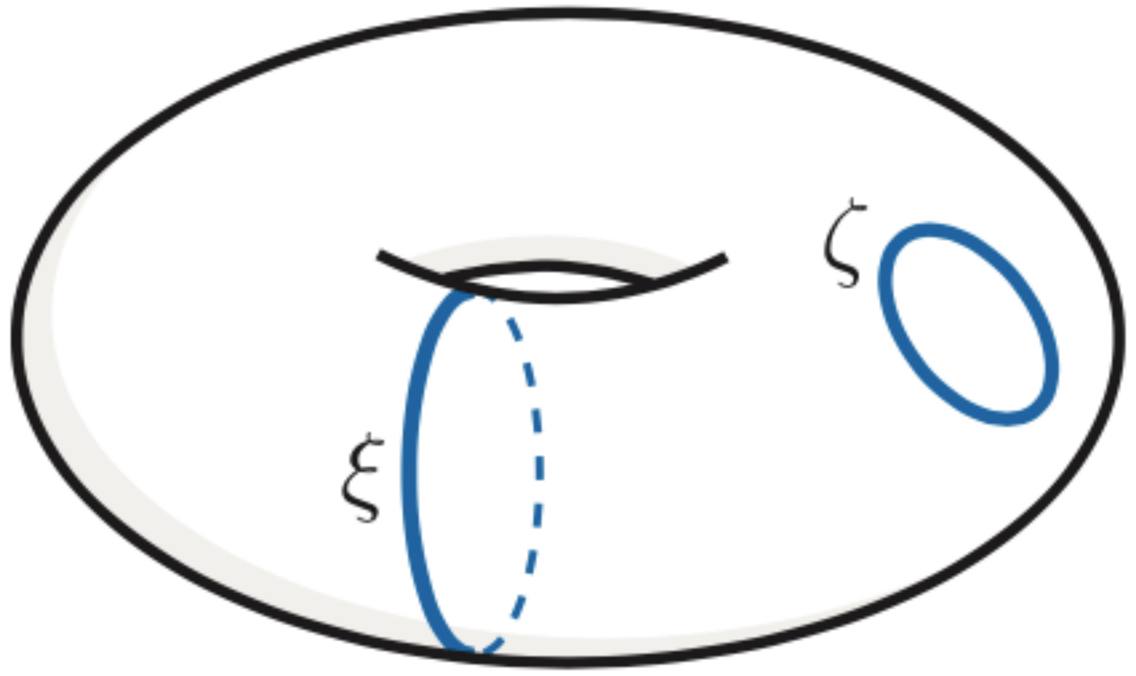
Trivial Connections on Discrete Surfaces

Keenan Crane[1] and Mathieu Desbrun[1] and Peter Schröder[1,2]

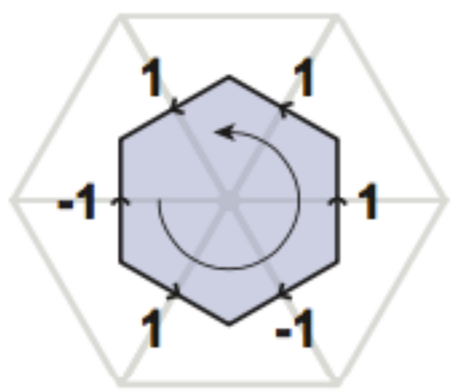
[1]Computing and Mathematical Sciences, Caltech
[2]Institute for Advanced Study, TU München

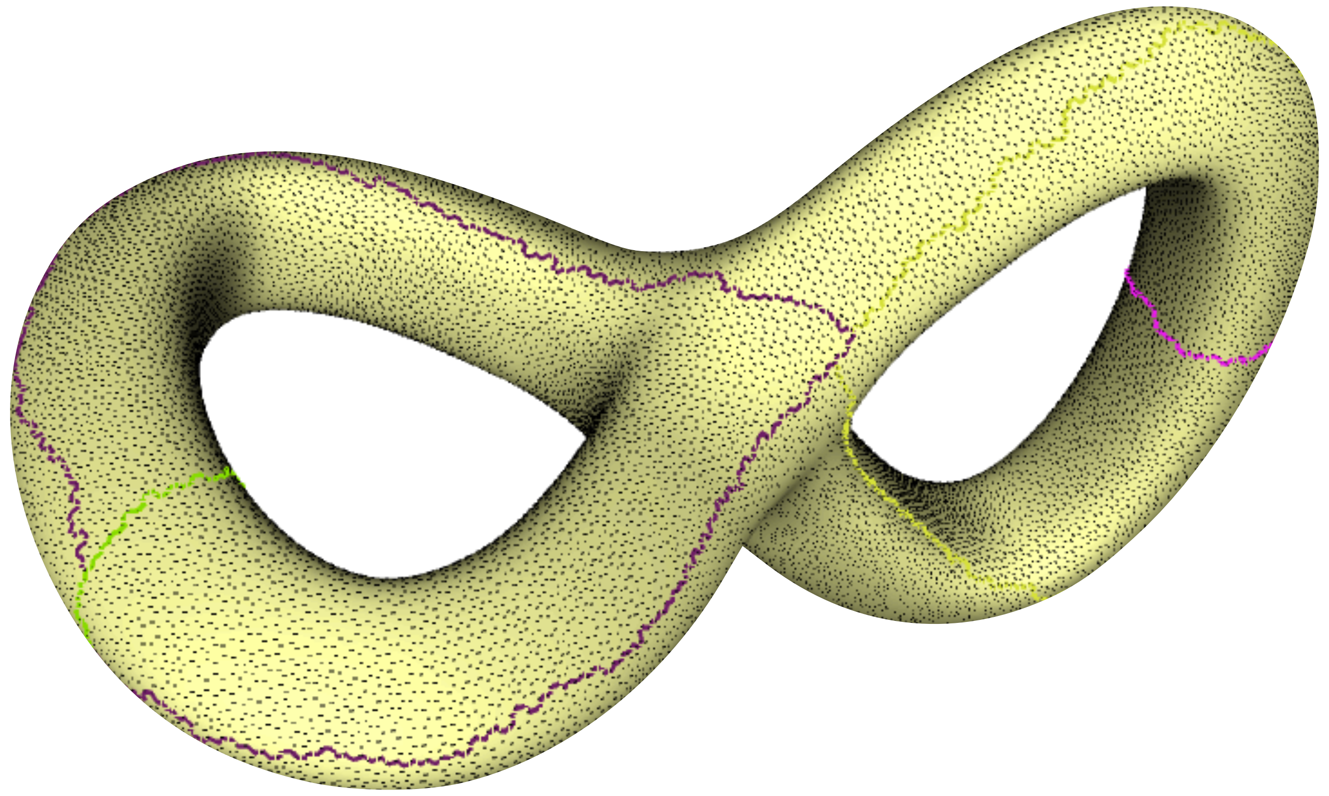
Linear Solve !

Basis for loops on a surface



Cycles: contractibles / noncontractibles





$d_0 \in \mathbb{R}^{|E|\times|V|}$

$H \in \mathbb{R}^{|E|\times 2g}$

$$(d_0)_{ij} = \begin{cases} \pm 1, & \text{dual edge } i \text{ is contained in dual cell } j \\ 0, & \text{otherwise.} \end{cases}$$

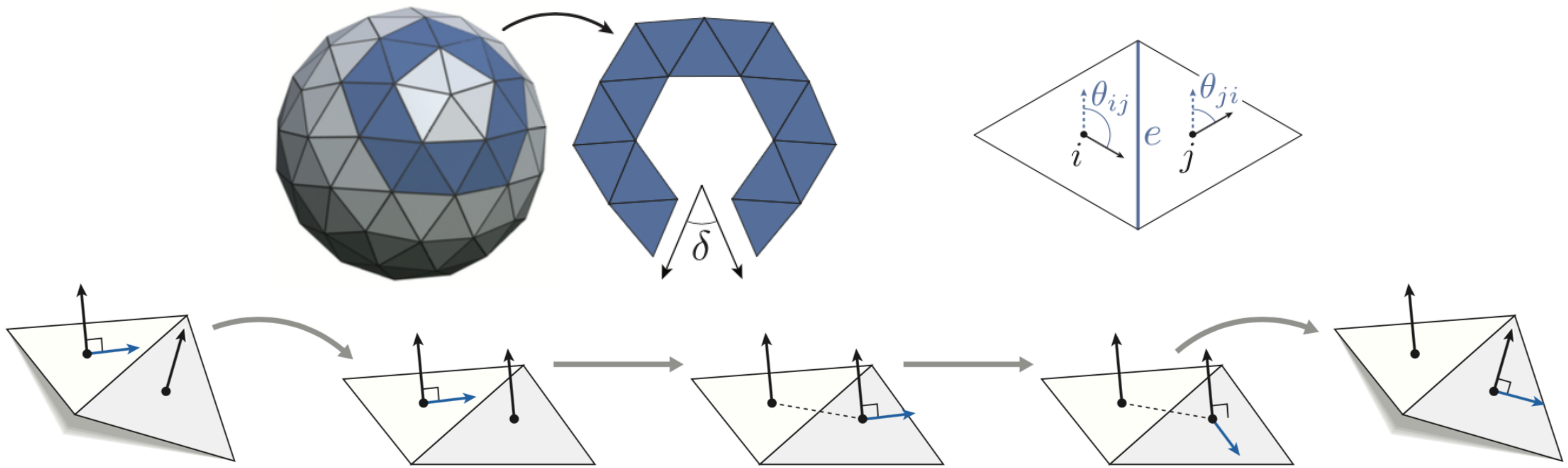
$$H_{ij} = \begin{cases} \pm 1, & \text{if dual edge } i \text{ is in generator } j \\ 0, & \text{otherwise.} \end{cases}$$

$$\boxed{A = \begin{bmatrix} d_0^T \\ H^T \end{bmatrix}.}$$

Holonomy, angular defects and trivial connection

Contractibles : Gaussian curvature $\boxed{K \in \mathbb{R}^{|V|}}$

Non contractibles : holonomie $\boxed{z \in \mathbb{R}^{2g}}$

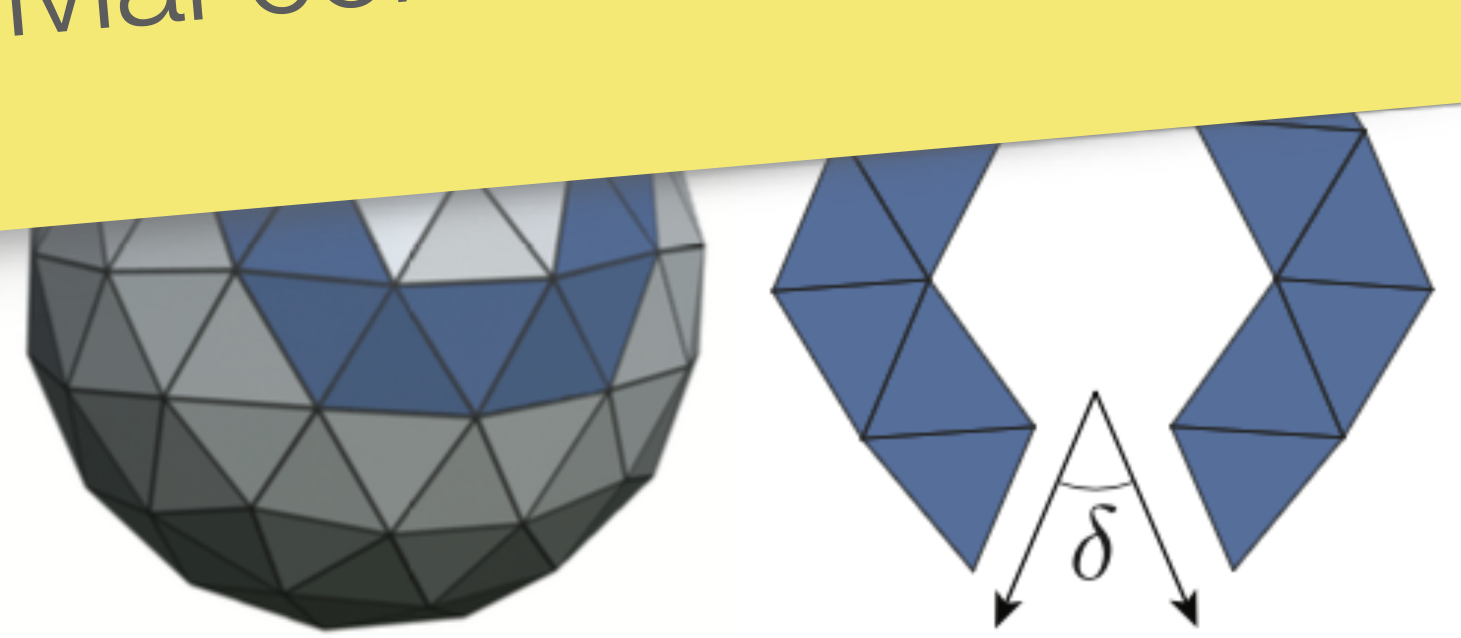


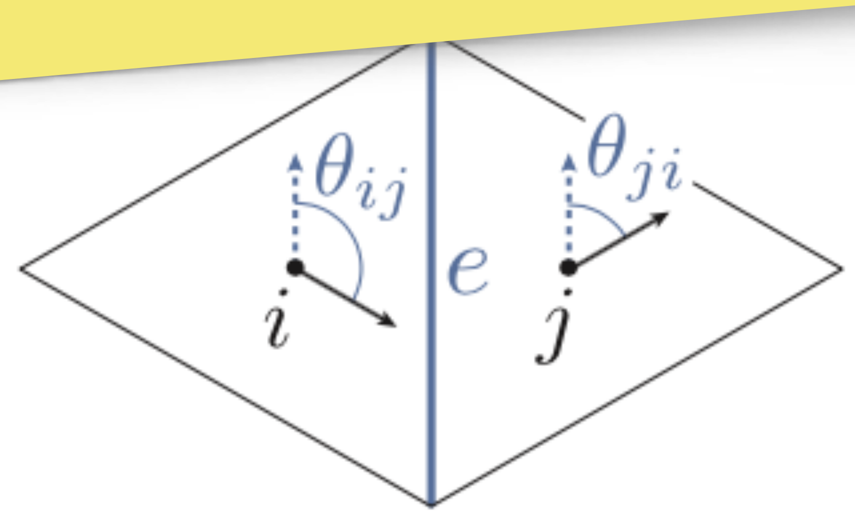
Holonomy, angular defects and trivial connection

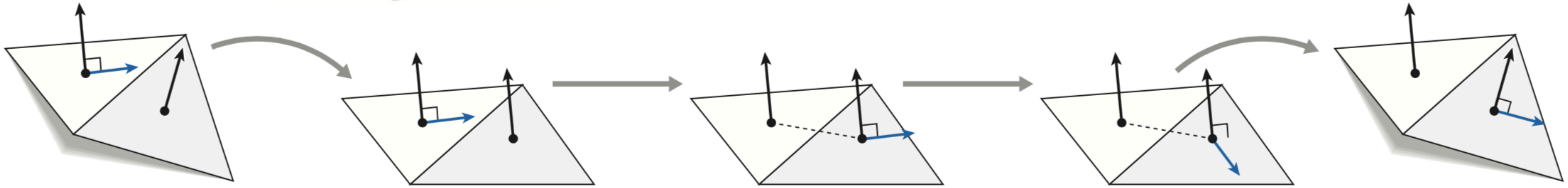
Contractibles : Gaussian curvature $\boxed{K \in \mathbb{R}^{|V|}}$

Non contractibles : holonomie $\boxed{z \in \mathbb{R}^{2g}}$

Trivial connection = angular defect for every cycle is zero







Singularities and indices

Index of a vertex: $k \in \mathbb{Z}^{|V|+2g}$

$\sum_i k_i = \chi$ (Euler)

New angle defects: $\tilde{K}_i = K_i - 2k_i\pi$

$\tilde{z}_i = z_i - 2k_i\pi$

$$\boxed{b = [\,\tilde{K}\ \tilde{z}\,]^T}$$

n-RoSy fields: $k_i = n_i/N,\ n_i \in \mathbb{Z}$

Solve

$$\boxed{\min_x ||x||_2 \quad \text{s.t.} \quad Ax = -b,}$$

A unique solution exists, using a projection onto the kernel:

$$Ax = -b$$

$$x^* = \tilde{x} - d_1^T(d_1 d_1^T)^{-1} d_1 \tilde{x}$$

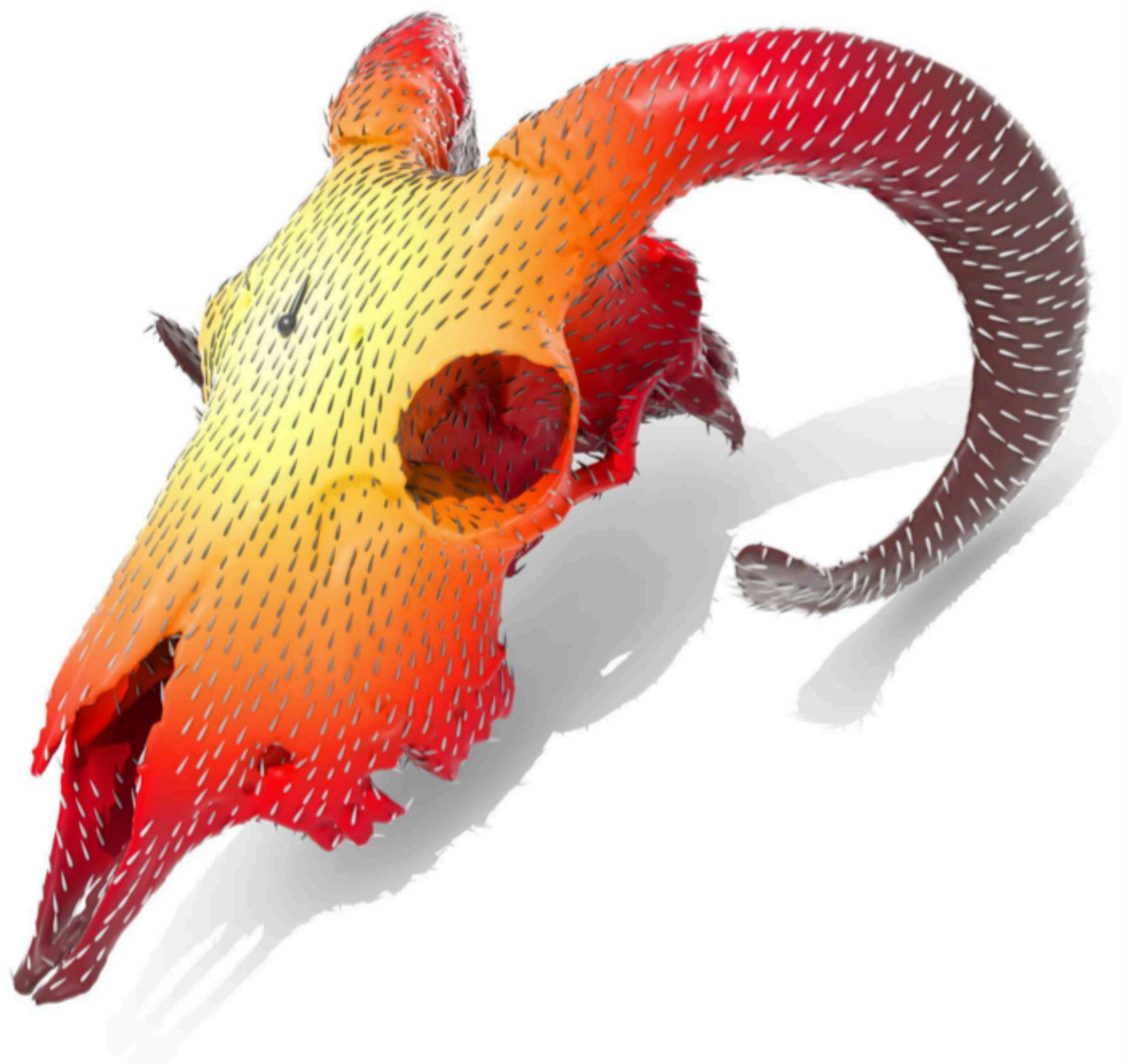
The Vector Heat Method

NICHOLAS SHARP, YOUSUF SOLIMAN, and KEENAN CRANE, Carnegie Mellon University

This paper describes a method for efficiently computing parallel transport of tangent vectors on curved surfaces, or more generally, any vector-valued data on a curved manifold. More precisely, it extends a vector field defined over any region to the rest of the domain via parallel transport along shortest geodesics. This basic operation enables fast, robust algorithms for extrapolating level set velocities, inverting the exponential map, computing geometric medians and Karcher/Fréchet means of arbitrary distributions, constructing centroidal Voronoi diagrams, and finding consistently ordered landmarks. Rather than evaluate parallel transport by explicitly tracing geodesics, we show that it can be computed via a short-time heat flow involving the *connection Laplacian*. As a result, transport can be achieved by solving three prefactored linear systems, each akin to a standard Poisson problem. Moreover, to implement the method we need only a discrete connection Laplacian, which we describe for a variety of geometric data structures (point clouds, polygon meshes, *etc.*). We also study the numerical behavior of our method, showing empirically that it converges under refinement, and augment the construction of intrinsic Delaunay triangulations (iDT) so that they can be used in the context of tangent vector field processing.

CCS: • **Mathematics of computing** → **Discretization**; **Partial differential equations**; • **Computing methodologies** → **Shape analysis**;

Additional Key Words and Phrases: discrete differential geometry, parallel transport, velocity extrapolation, logarithmic map, exponential map, Karcher

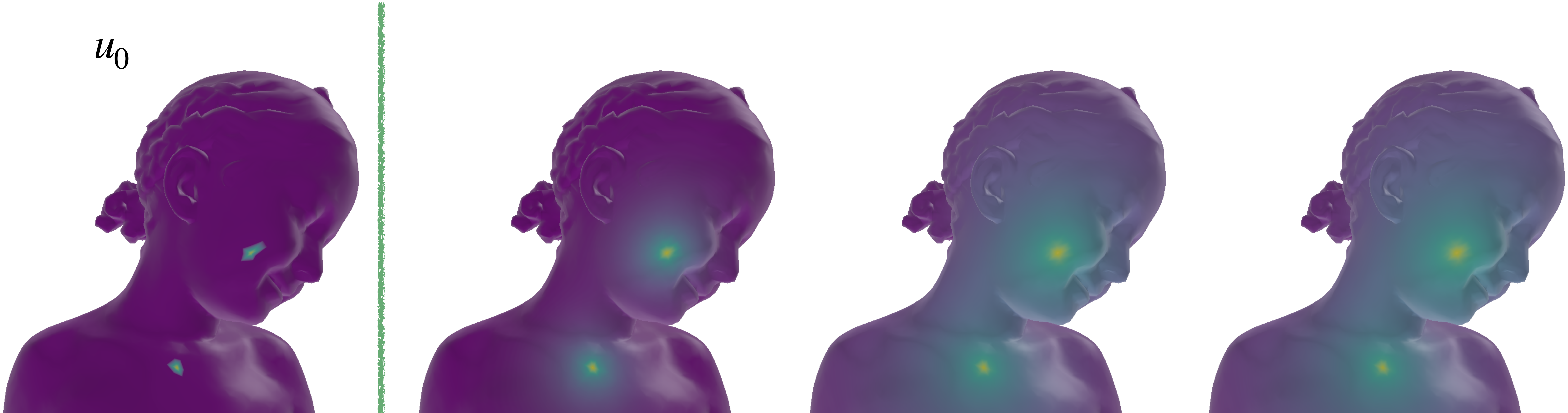


Prelim: Heat diffusion

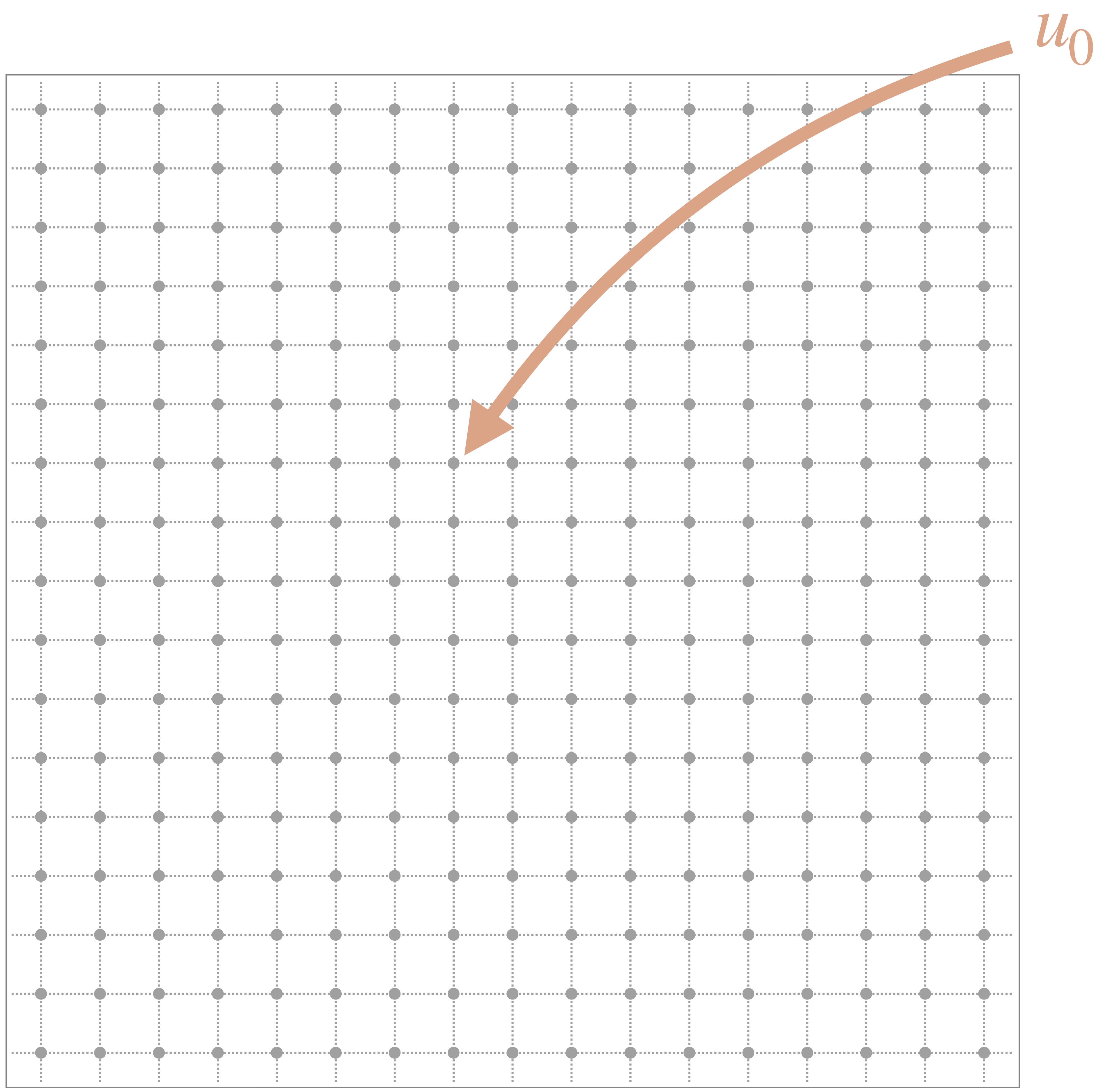
$$u(x,t) : \Omega \times \mathbb{R}^+ \to \mathbb{R}$$

$$\boxed{\frac{\partial u}{\partial t} = \Delta u \quad \text{subject to} \quad u(x,0) = u_0(x)}$$

u_0

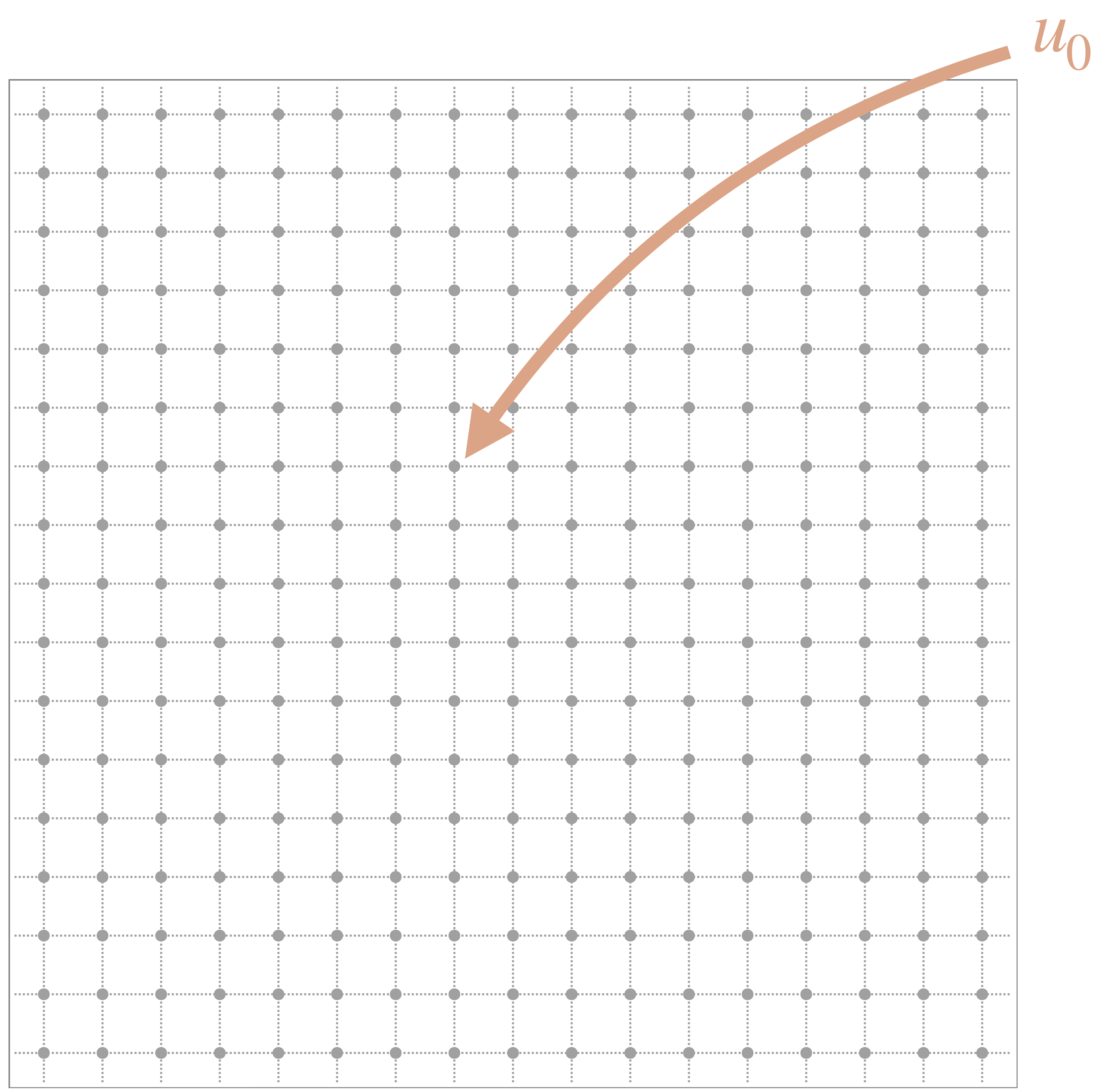


Discrete Laplace Operator in 2D



In $\mathbb{R}^2$, $\quad \Delta f(x, y) = \dfrac{\partial^2 f}{\partial x^2} + \dfrac{\partial^2 f}{\partial y^2}$

Discrete Laplace Operator in 2D

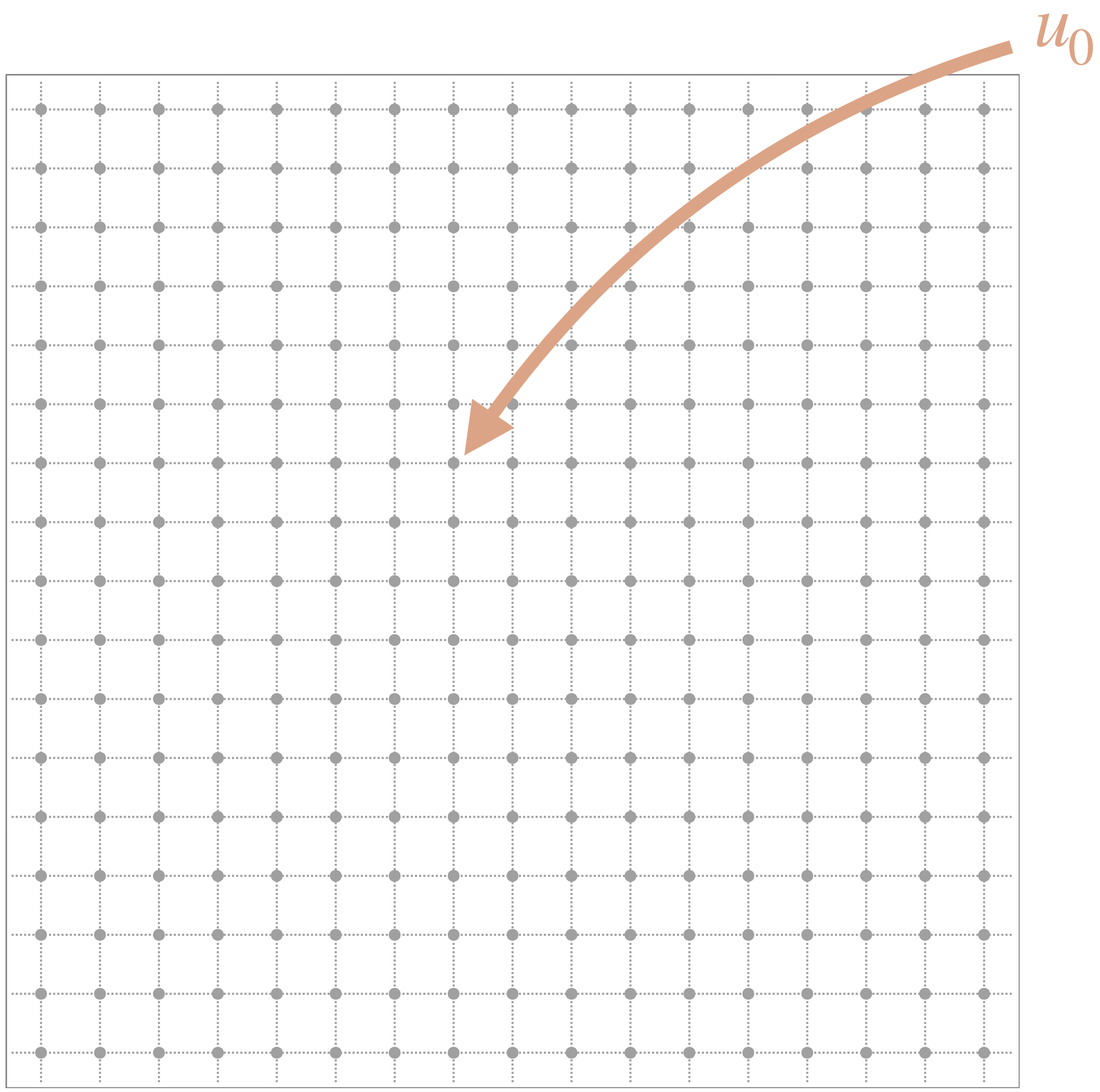


In $\mathbb{R}^2$, $\quad \Delta f(x, y) = \dfrac{\partial^2 f}{\partial x^2} + \dfrac{\partial^2 f}{\partial y^2}$

In $[0,n]^2$, with 1st order finite diff. appRox.

$$\boxed{\begin{aligned} Lf(x, y) = {} & f(x+1,y) + f(x-1,y) \\ & + f(x, y+1) + f(x, y-1) \\ & -4f(x, y) \end{aligned}}$$

Discrete Laplace Operator in 2D



In $\mathbb{R}^2$, $\quad \Delta f(x, y) = \dfrac{\partial^2 f}{\partial x^2} + \dfrac{\partial^2 f}{\partial y^2}$

In $[0,n]^2$, with 1st order finite diff. appRox.

$$\boxed{\begin{aligned} Lf(x, y) = {} & f(x+1,y) + f(x-1,y) \\ & + f(x, y+1) + f(x, y-1) \\ & - 4f(x, y) \end{aligned}}$$

$L \in \mathbb{R}^{|V|\times|V|}$

$L(i, j) = 1$ if i is adjacent to j

$L(i, i) = degree(i)$

Discrete Heat Equation

$$\frac{\partial u}{\partial t} = \Delta u \quad \Rightarrow \quad (id - t\Delta)u_t = u_0 \qquad \text{(backward Euler)}$$

$$\boxed{\text{Solve a (sparse) linear system } \ (Id - tL)u_t = u_0}$$

$$\Omega = 3 \times 3 \text{ image } = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

$$L = \begin{bmatrix} -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & -3 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2 \end{bmatrix}$$

$$u_0 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Rightarrow [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

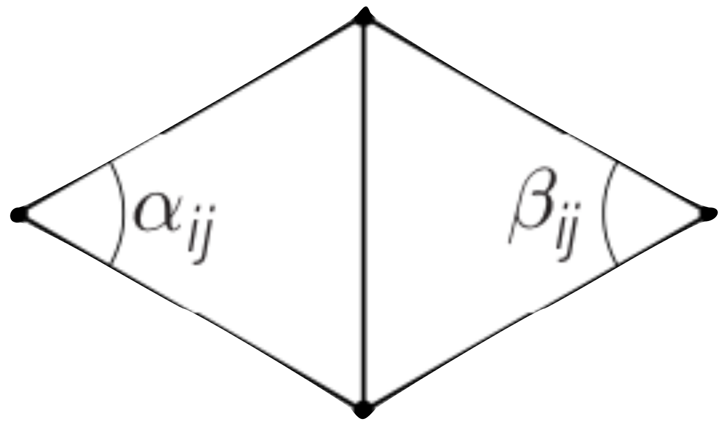
$$u_t = (Id - tL)^{-1} u_0$$

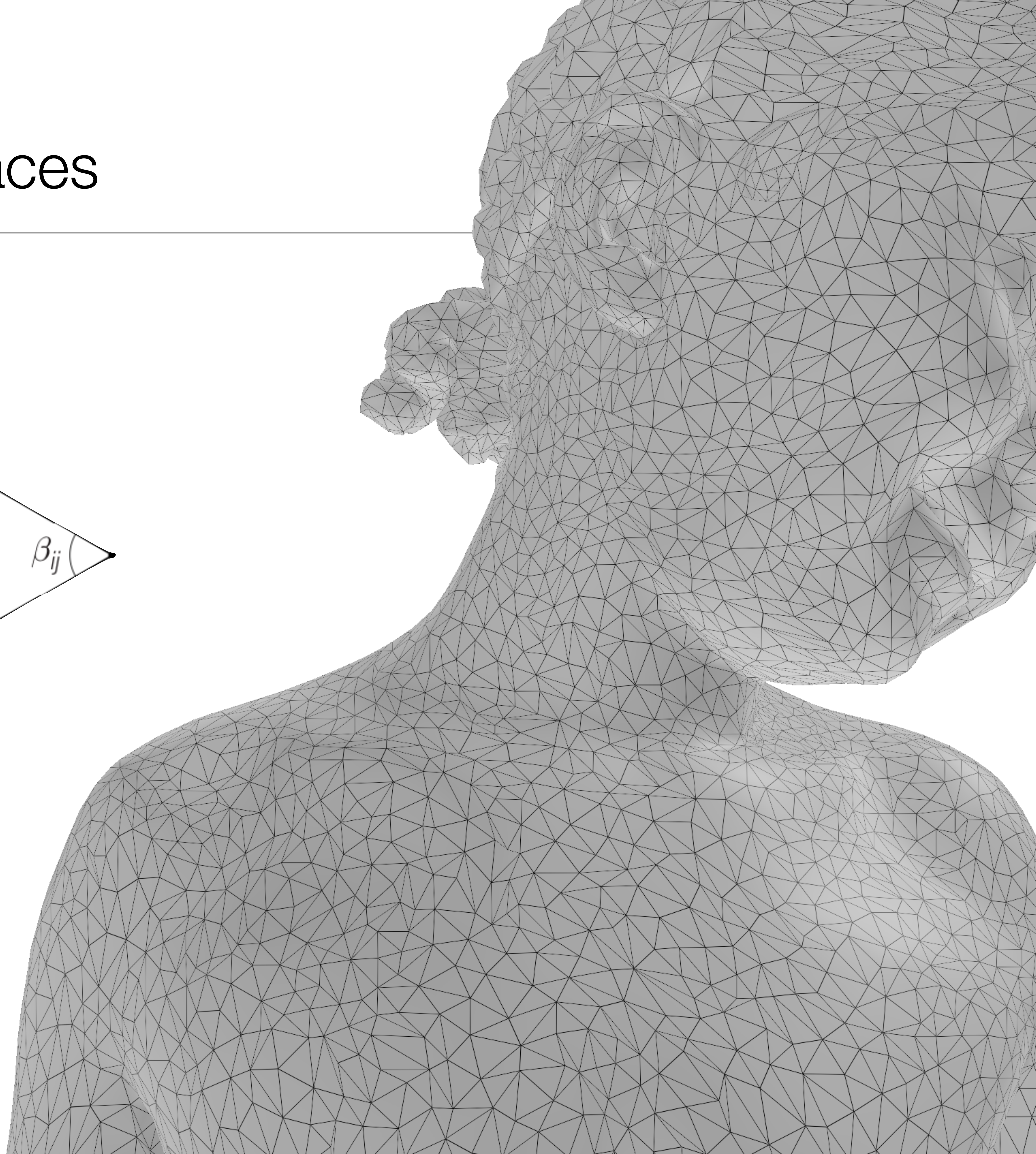
Laplace-Beltrami operator on surfaces

$\tilde{L} \in \mathbb{R}^{|V| \times |V|}$

$$\tilde{L}(i,j) = \frac{1}{2}\left(\cot(\alpha_{ij}) + \cot(\beta_{ij})\right) \text{ if } i \text{ is adjacent to } j$$

$$\tilde{L}(i,i) = -\sum_j L(i,j)$$



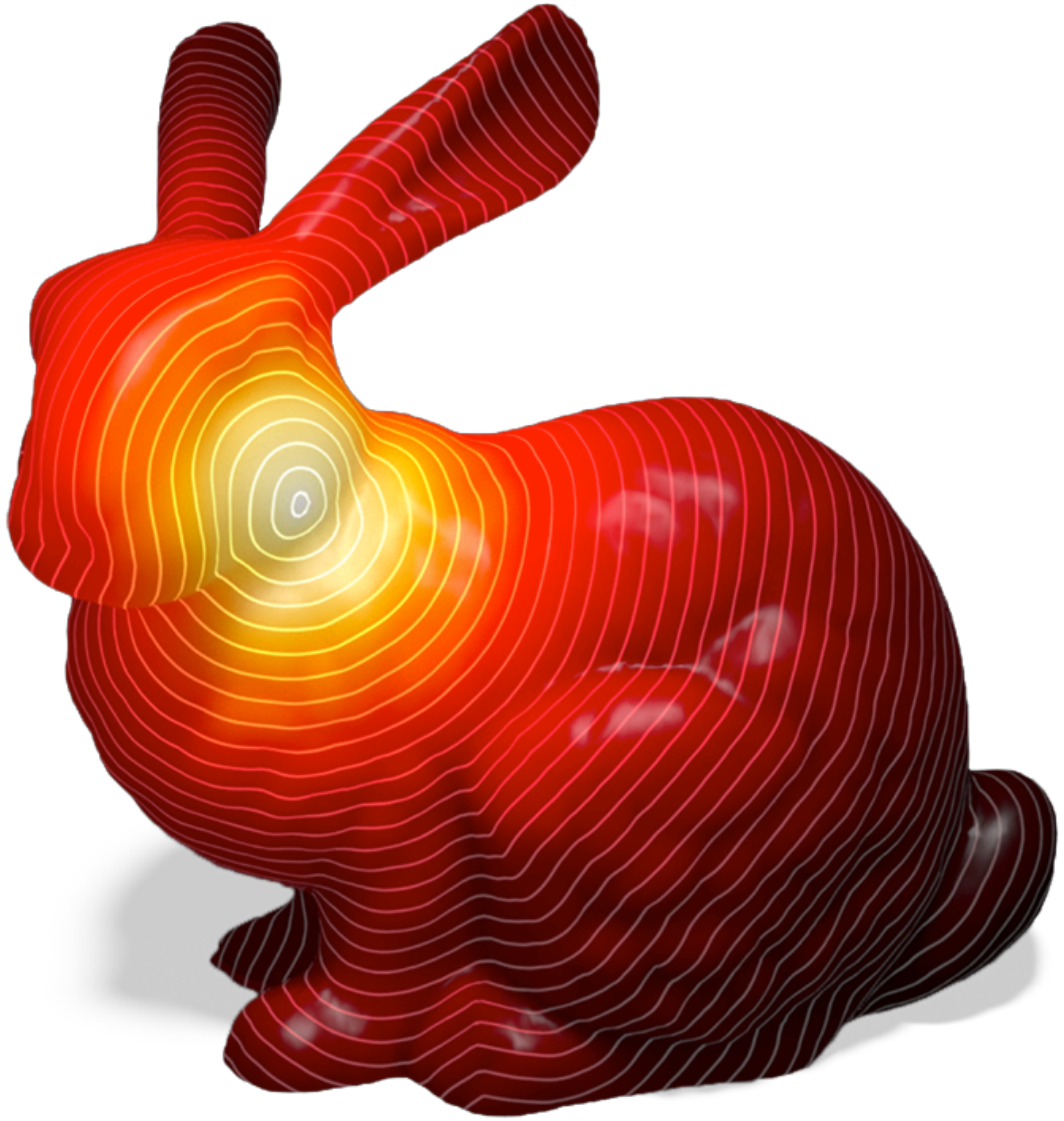


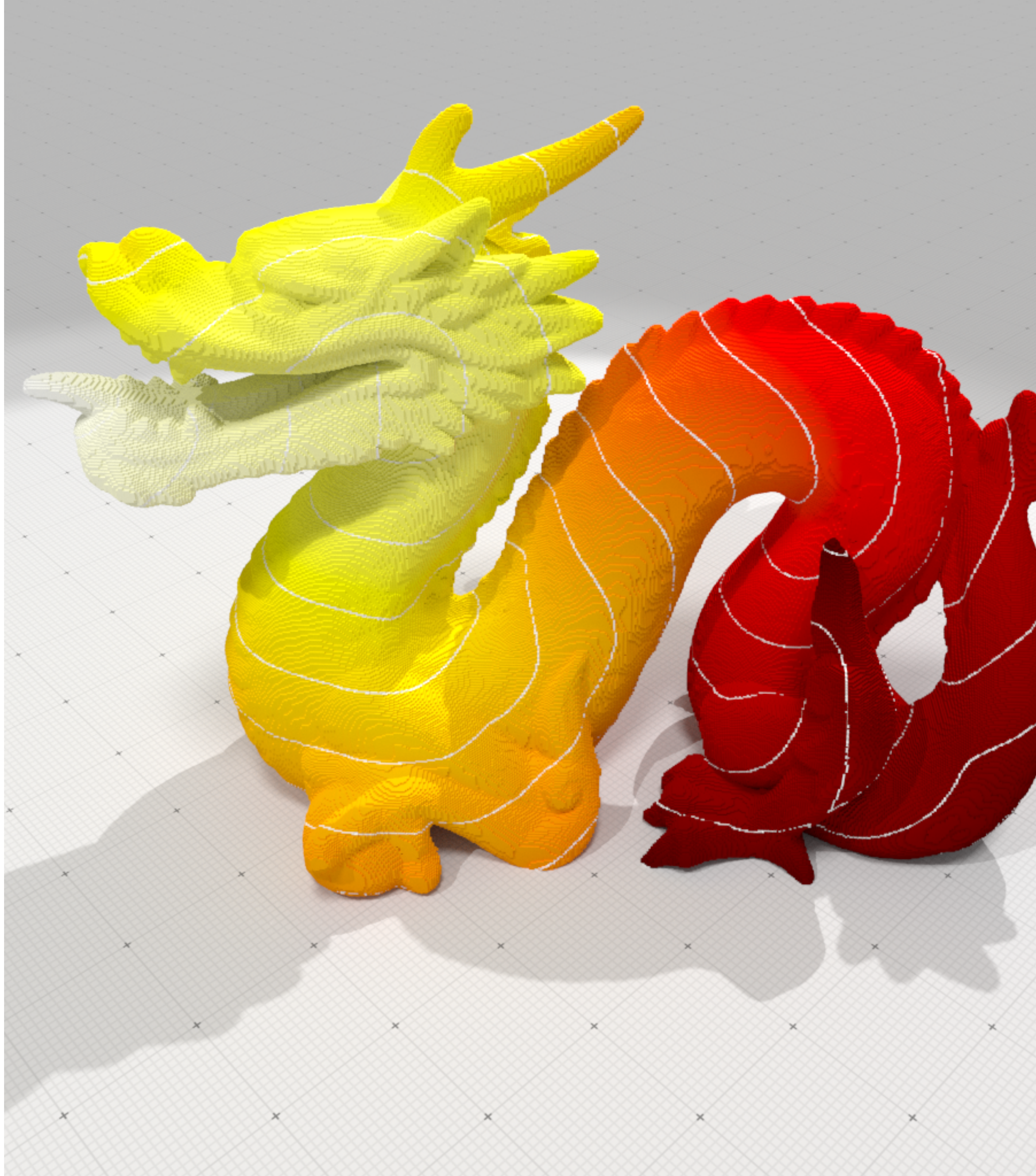
weak form of the laplacian, Heat equation becomes

$$(M + t\tilde{L})u_t = u_0$$

triangle areas

```
if (ImGui::Button("Heat diffusion"))
{
  //Sources
  u0[mesh->vertex(2714)] = 42.0;
  u0[mesh->vertex(1613)] = 17;
  Eigen::SparseMatrix<double> Poisson  = (massMatrix + delta*laplacian);  //same as (Id - tL)

  //We solve are solving the Poisson problem
  SquareSolver<double>  heatSolver(Poisson);
  auto ut = heatSolver.solve(u0.toVector());

  //Display
  utQ.fromVector(ut);
  polyscope::getSurfaceMesh()->addQuantity("u_t", utQ);
  polyscope::getSurfaceMesh()->addQuantity("u_0", u0);
}
```





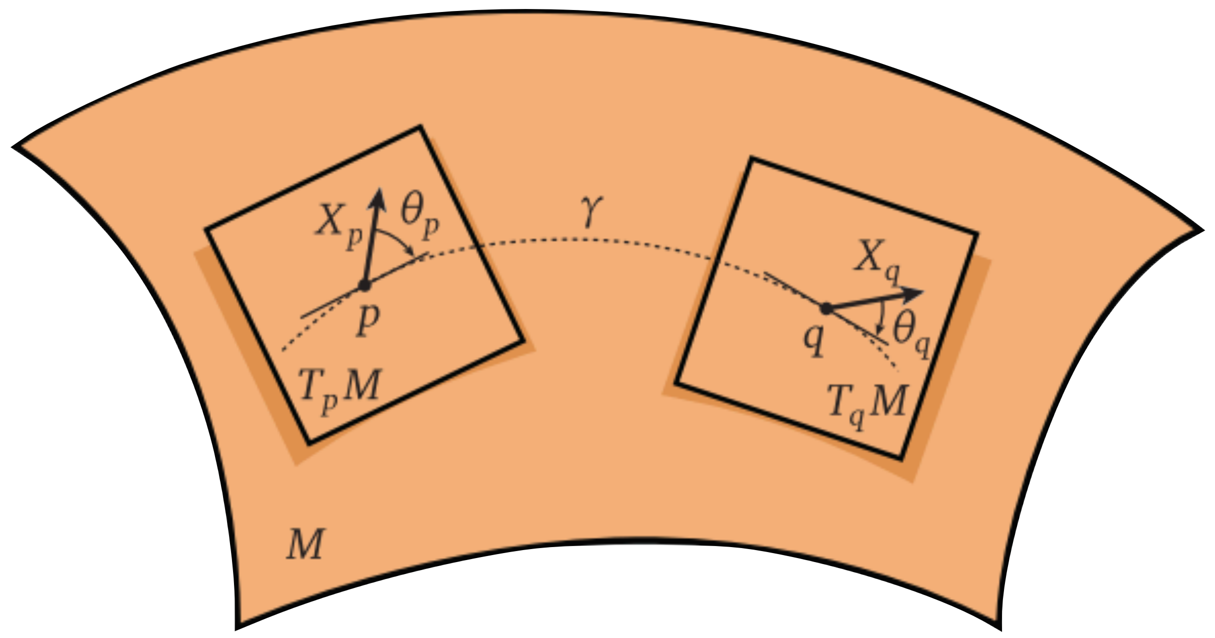
Heat kernel ~ geodesic distances

Connection Laplacian and Vector Heat equation

Δ^{∇} is the connection Laplacian (second derivative of vector fields)

Vector heat equation: $\dfrac{d}{dt}X_t = \Delta^{\nabla} X_t$

$$\Rightarrow (M + tL^{\nabla})Y = Y_0$$



Vector heat kernel behaves like parallel transport along geodesics!

Vector Heat Method [Sharp et al 2019]

ALGORITHM 1: Vector Heat Method

Input: A vector field X supported on a subset $\Omega \subset M$ of the domain M.

Output: A vector field $\overline{X}$ on all of M.

I. Integrate the vector heat flow $\frac{d}{dt} Y_t = \Delta^\nabla Y_t$ for time t, with $Y_0 = X$.

II. Integrate the scalar heat flow $\frac{d}{dt} u_t = \Delta u_t$ for time t, with $u_0 = |X|$.

III. Integrate the scalar heat flow $\frac{d}{dt} \phi_t = \Delta \phi_t$ for time t, with $\phi_0 = \mathbb{1}_\Omega$.

IV. Evaluate the vector field $\overline{X}_t = u_t Y_t / \phi_t |Y_t|$.

Vector Heat Method [Sharp et al 2019]

ALGORITHM 1: Vector Heat Method

Input: A vector field X supported on a subset $\Omega \subset M$ of the domain M.

Output: A vector field $\overline{X}$ on all of M.

I. Integrate the vector heat flow $\frac{d}{dt} Y_t = \Delta^\nabla Y_t$ for time t, with $Y_0 = X$.

II. Integrate the scalar heat flow $\frac{d}{dt} u_t = \Delta u_t$ for time t, with $u_0 = |X|$.

III. Integrate the scalar heat flow $\frac{d}{dt} \phi_t = \Delta \phi_t$ for time t, with $\phi_0 = \mathbb{1}_\Omega$.

IV. Evaluate the vector field $\overline{X}_t = u_t Y_t / \phi_t |Y_t|$.

Heat diffusion on scalars 😍

Vector Heat Method [Sharp et al 2019]

wut?

ALGORITHM 1: Vector Heat Method

Input: A vector field X supported on a subset $\Omega \subset M$ of the domain M.

Output: A vector field $\overline{X}$ on all of M.

I. Integrate the vector heat flow $\frac{d}{dt}Y_t = \Delta^\nabla Y_t$ for time t, with $Y_0 = X$.

II. Integrate the scalar heat flow $\frac{d}{dt}u_t = \Delta u_t$ for time t, with $u_0 = |X|$.

III. Integrate the scalar heat flow $\frac{d}{dt}\phi_t = \Delta \phi_t$ for time t, with $\phi_0 = \mathbb{1}_\Omega$.

IV. Evaluate the vector field $\overline{X}_t = u_t Y_t / \phi_t |Y_t|$.

Heat diffusion on scalars 😍

Vector Heat Method in Euclidean domains

Y_0 : input vectors
u_0 : input vector norms
ϕ_0 : Diracs at vector positions

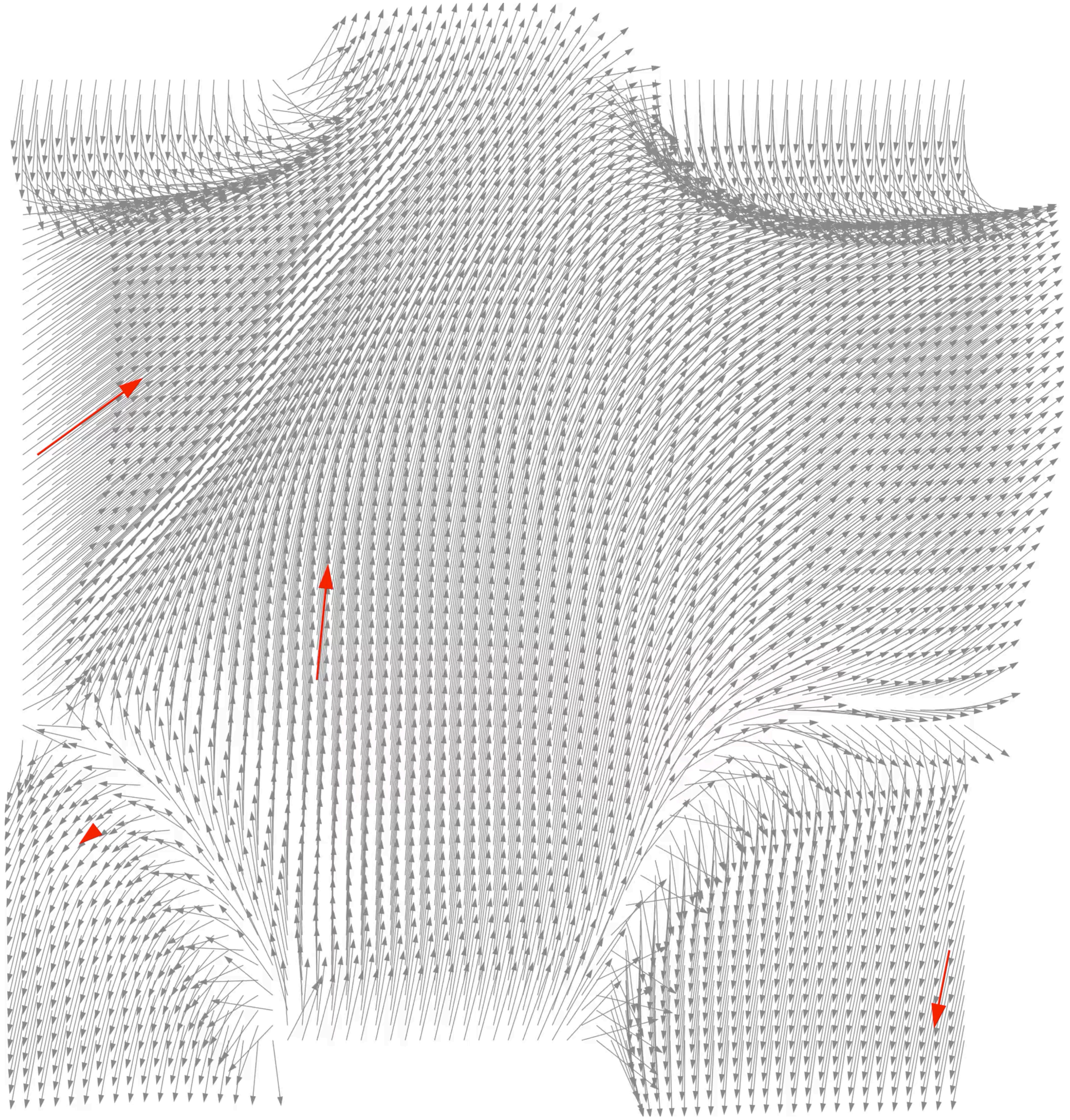
Solve 4 (sparse) Poisson problems

$$(Id - tL)u = u_0 \qquad (Id - tL)\phi = \phi_0$$

$$(Id - tL)Y^x = Y_0^x \qquad (Id - tL)Y^y = Y_0^y$$

$$\Rightarrow \bar{Y} = \frac{u}{\phi} \cdot \frac{Y}{\|Y\|}$$

$(\Delta^\nabla \equiv \Delta \text{ in } \mathbb{R}^d)$



(toroidal boundary conditions 😍)

Vector Heat Method in Euclidean domains

Y_0 : input vectors

u_0 : input vector norms

ϕ_0 : Diracs at vector positions

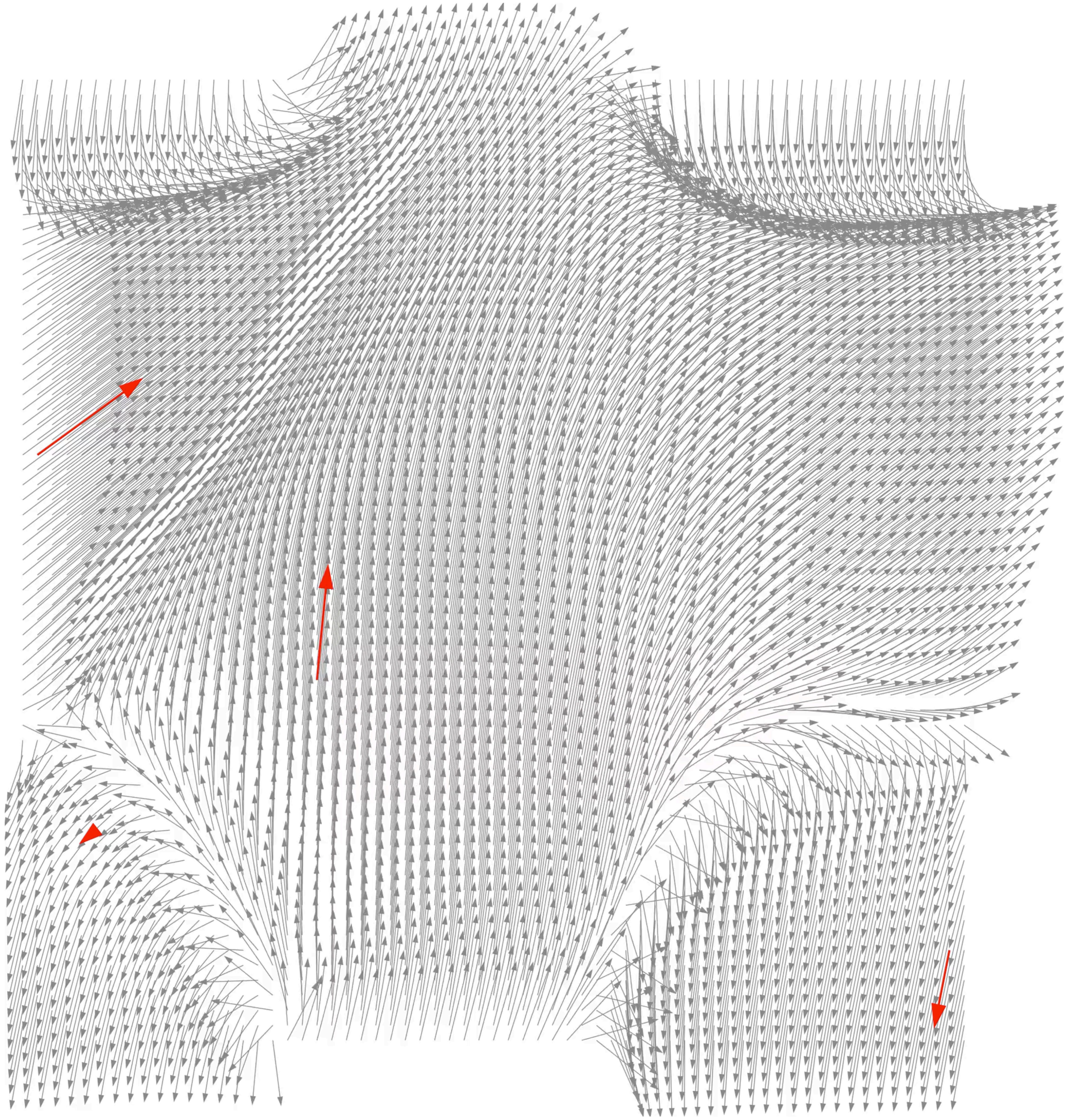
Solve 4 (sparse) Poisson problems

$$(Id - tL)u = u_0 \qquad (Id - tL)\phi = \phi_0$$

$$(Id - tL)Y^x = Y_0^x \qquad (Id - tL)Y^y = Y_0^y$$

$$\Rightarrow \bar{Y} = \frac{u}{\phi} \cdot \frac{Y}{\|Y\|}$$

$(\Delta^\nabla \equiv \Delta \text{ in } \mathbb{R}^d)$

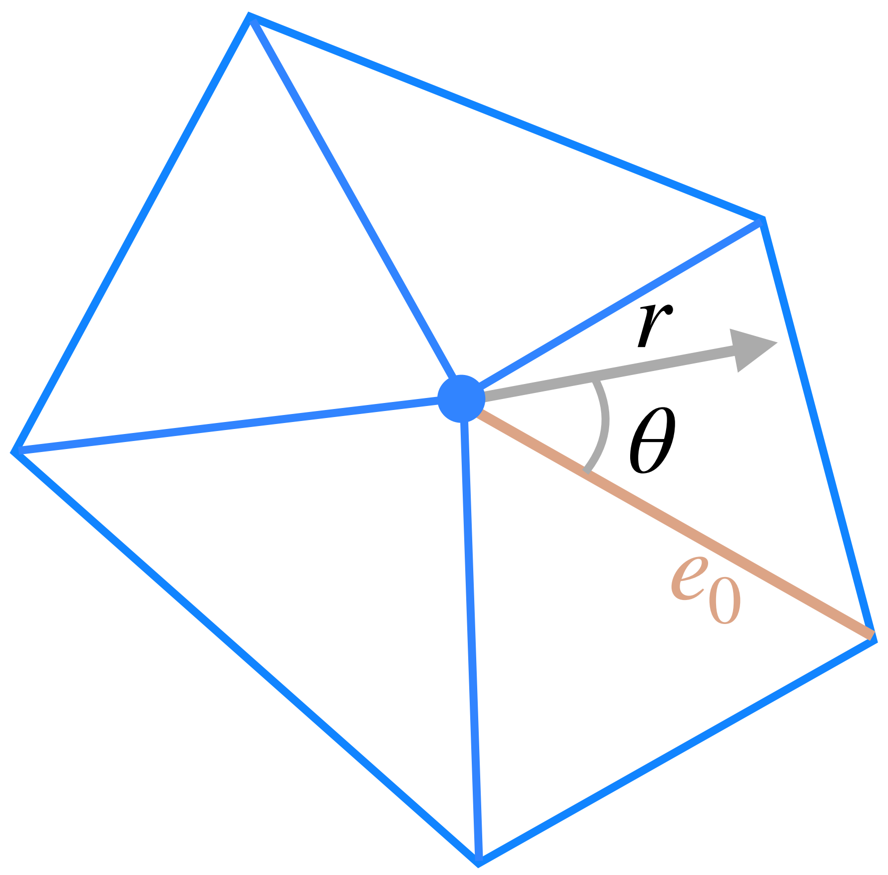


(toroidal boundary conditions 😍)

Static
(prefactorized solver)

```
int W=512,H=512;
Eigen::SparseMatrix<double> lap(W*H,W*H);
Eigen::SparseMatrix<double> Id(W*H,W*H);
Id.setIdentity();

Eigen::Vector2d u_0(W*H);
Eigen::Vector2d phi_0(W*H);
Eigen::Vector2d v_x0(W*H);
Eigen::Vector2d v_y0(W*H);

//Single vector source
v_x0[123] = 1.0;
v_y0[123] = 1.0;
u_0[123] = std::sqrt(2.0); // norm
phi_0[123]= 1.0; //diracs

//Lap
for(auto i=0; i < W*H; ++i)
  for(auto j=0; j< H*W; ++j)
  {
    lap.coeffRef(i, j) = -4.0;
    if (adjacent(i,j))
      lap.coeffRef(i,j) = 1.0;
  }

//Solve
SquareSolver<double> solver(Id - t*lap);
auto u_t = solver.solve(u_0);
auto phi_t = solver.solve(phi_0);
auto v_xt = solver.solve(v_x0);
auto v_yt = solver.solve(v_y0);

Eigen::Vector2d final_x(W*H);
Eigen::Vector2d final_y(W*H);

for(auto i=0; i < W*H; ++i)
{
  final_x[i] = v_xt[i] / std::sqrt((v_xt[i]*v_xt[i] + v_yt[i]*v_yt[i])) * u_t[i] / phi_t[i];
  final_y[i] = v_yt[i] / std::sqrt((v_xt[i]*v_xt[i] + v_yt[i]*v_yt[i])) * u_t[i] / phi_t[i];
}
```

Discrete Connection Laplacian



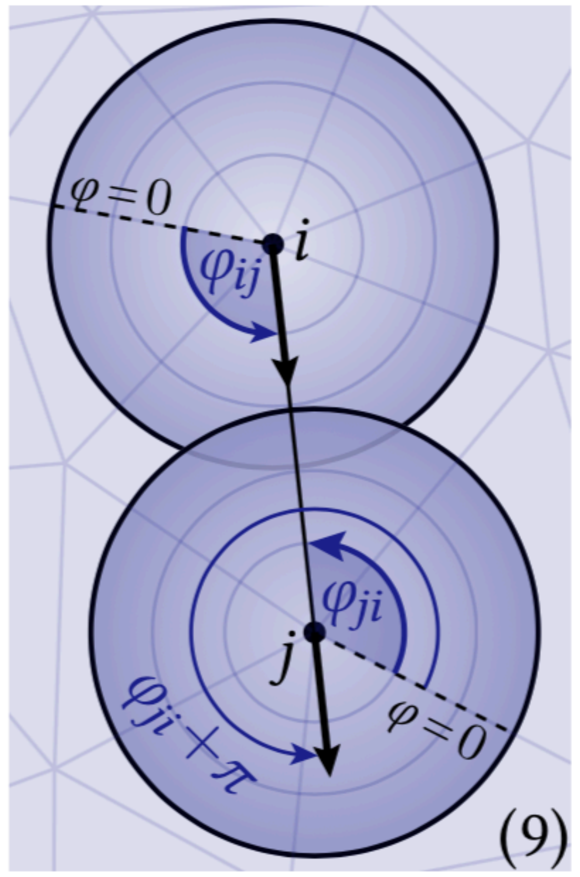
Intrinsic vector per vertex = a complex number $z = re^{i\theta}$

$$L^{\nabla} \in \mathbb{C}^{|V|\times|V|}$$

$$L^{\nabla}(i,j) = \boxed{r_{ij}} \cdot \tilde{L}(i,j) \text{ if } i \text{ is adjacent to } j$$

$$L^{\nabla}(i,i) = \tilde{L}(i,i)$$

$$r_{ij} = e^{\phi_{ji}+\pi-\phi_{i}j}$$



(L^{∇} is sparse!)

```
//Connection Laplacian
connectionLaplacian=Eigen::SparseMatrix<Complex> (laplacian.cast<std::complex<double>>());
for(auto edge: mesh->edges())
{
  auto he = edge.halfedge();
  Complex r_ij = gc.vertexTransportCoefs[ he ];
  auto i = gc.vertexIndices[ he.vertex() ];
  auto j = gc.vertexIndices[ he.twin().vertex() ];

  connectionLaplacian.coeffRef(i, j) = connectionLaplacian.coeffRef(i, j)*inv(r_ij);
  connectionLaplacian.coeffRef(j, i) = connectionLaplacian.coeffRef(j, i)*(r_ij);
}
```

Heat Vector Field

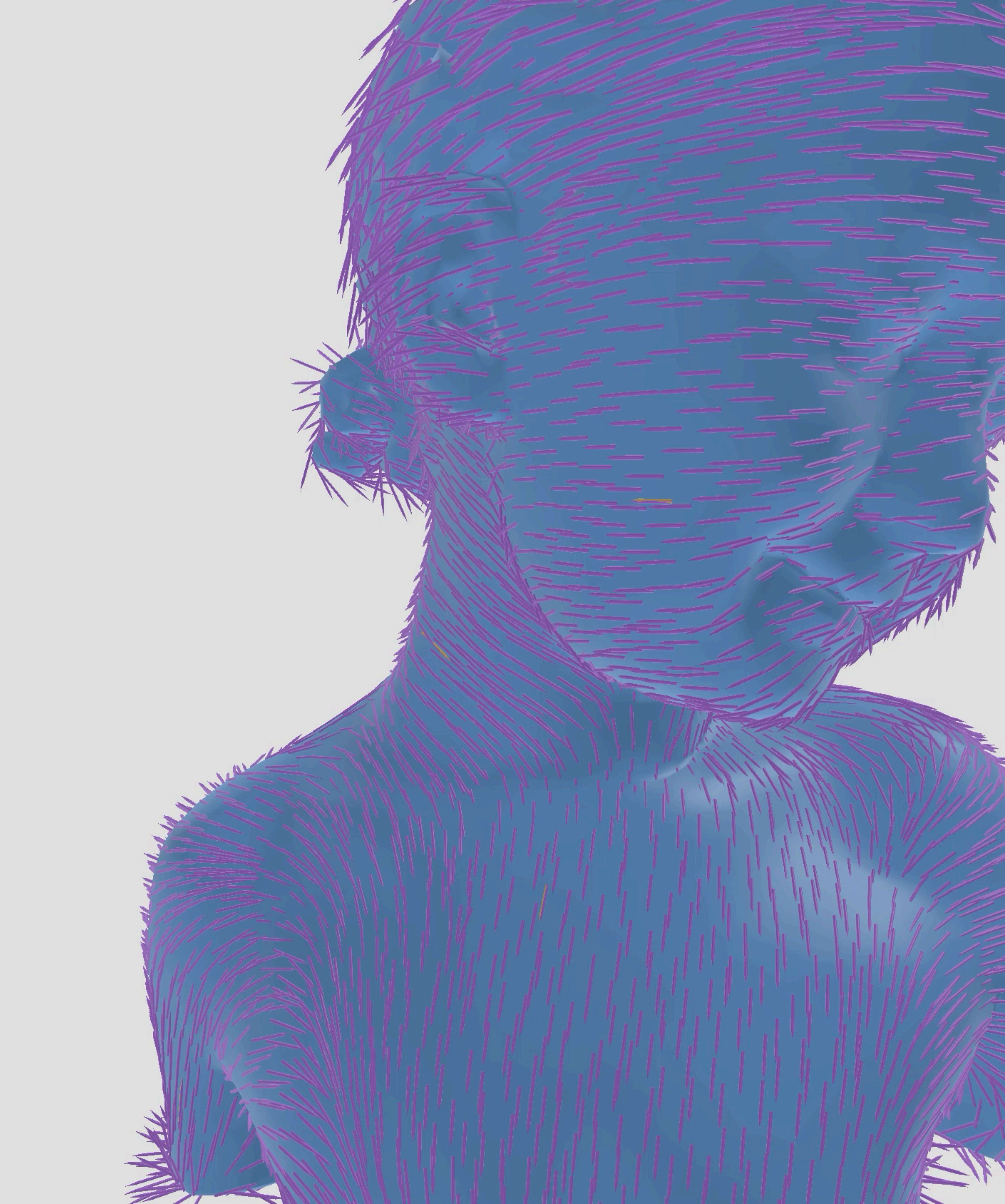
2 linear solves on real numbers + 1 solve on complex numbers

$$(M + t\tilde{L})u = u_0 \qquad (M + t\tilde{L})\phi = \phi_0$$

$$(M + tL^{\nabla})Y = Y_0$$

$$\Rightarrow \bar{Y} = \frac{u}{\phi} \cdot \frac{Y}{||Y||}$$

```
//Two vectors
vf0[mesh->vertex(1613)] = (double)scale* std::exp((double)theta*IM_I);
vf0[mesh->vertex(2714)] = (double)scale* std::exp((double)theta*IM_I);

//Vector norms
u0[mesh->vertex(2714)] = std::abs(vf0[mesh->vertex(2714)]);
u0[mesh->vertex(1613)] = std::abs(vf0[mesh->vertex(1613)]);
//Diracs
phi0[mesh->vertex(2714)] = 1.0;
phi0[mesh->vertex(1613)] = 1.0;

//Solve
auto ut  = solver->solve(u0.toVector());
auto Y   = solverConnection->solve(vf0.toVector());
auto phi = solver->solve(phi0.toVector());

//Final
for(auto v: mesh->vertices())
  YfinalQ2[v] = YQ[v] / std::abs(YQ[v]) * utQ[v] / phiQ[v];
```



Heat Vector Field

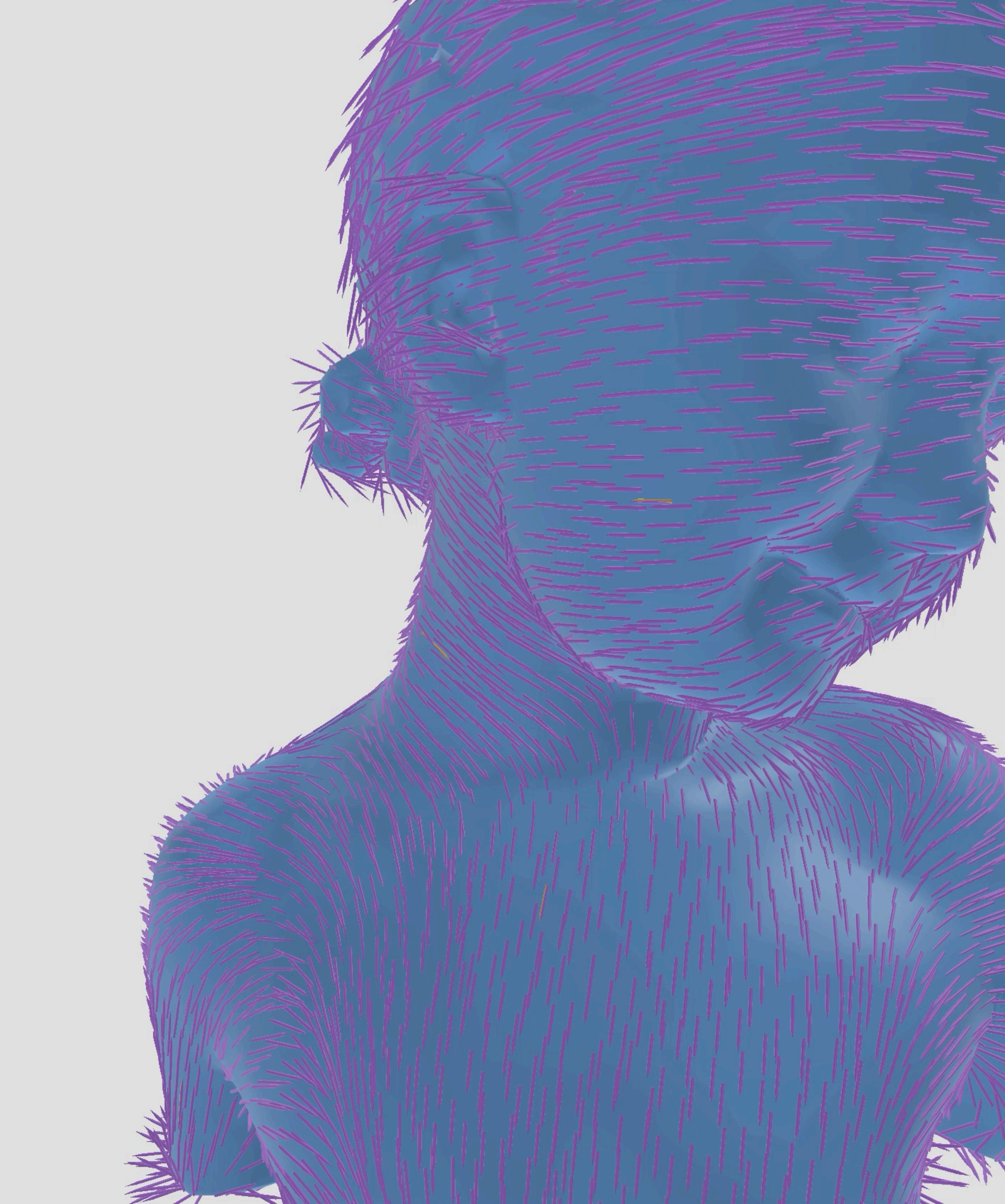
2 linear solves on real numbers + 1 solve on complex numbers

$$(M + t\tilde{L})u = u_0 \qquad (M + t\tilde{L})\phi = \phi_0$$

$$(M + tL^{\nabla})Y = Y_0$$

$$\Rightarrow \bar{Y} = \frac{u}{\phi} \cdot \frac{Y}{\|Y\|}$$

```
//Two vectors
vf0[mesh->vertex(1613)] = (double)scale* std::exp((double)theta*IM_I);
vf0[mesh->vertex(2714)] = (double)scale* std::exp((double)theta*IM_I);

//Vector norms
u0[mesh->vertex(2714)] = std::abs(vf0[mesh->vertex(2714)]);
u0[mesh->vertex(1613)] = std::abs(vf0[mesh->vertex(1613)]);
//Diracs
phi0[mesh->vertex(2714)] = 1.0;
phi0[mesh->vertex(1613)] = 1.0;

//Solve
auto ut  = solver->solve(u0.toVector());
auto Y   = solverConnection->solve(vf0.toVector());
auto phi = solver->solve(phi0.toVector());

//Final
for(auto v: mesh->vertices())
  YfinalQ2[v] = YQ[v] / std::abs(YQ[v]) * utQ[v] / phiQ[v];
```



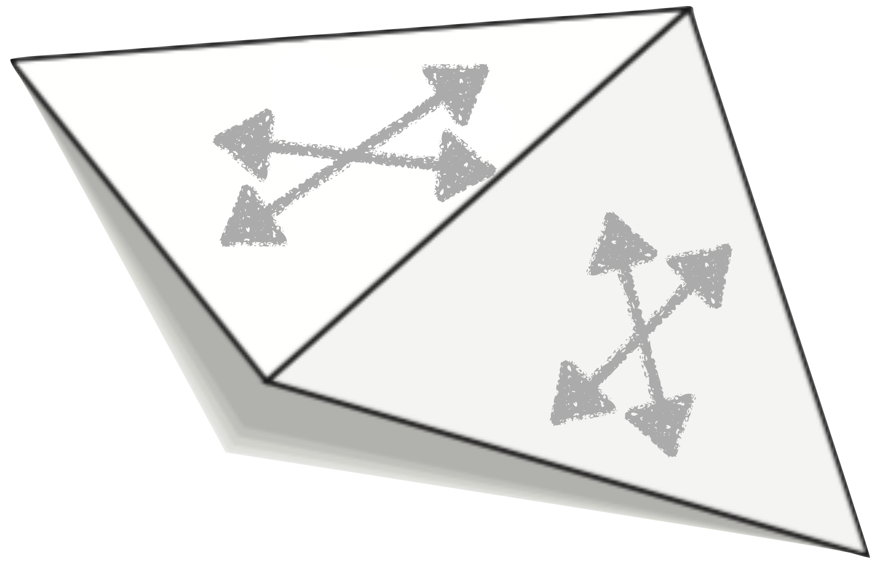
< Demo >

< Demo >

Further readings

- Similarly to Laplace-Beltrami for scalar functions, VF admits a **spectral decomposition** (and eigenvector basis)

- N-polyvector fields



- VF interpretation of optimal transport ($\mathscr{W}_1$)

Refs

- Brandt, C., Scandolo, L., Eisemann, E., & Hildebrandt, K. (2016). Spectral Processing of Tangential Vector Fields. Computer Graphics Forum, 36(6), 338–353. https://doi.org/10.1111/cgf.12942
- Vaxman, A., Campen, M., Diamanti, O., Bommes, D., Hildebrandt, K., Technion, M. B.-C., & Panozzo, D. (2017). Directional field synthesis, design, and processing. ACM SIGGRAPH 2017 Courses on - SIGGRAPH '17, 35(2), 1–30. https://doi.org/10.1145/3084873.3084921
- Sharp, N., Soliman, Y., & Crane, K. (2018). The Vector Heat Method.
- de Goes, F., Desbrun, M., & Tong, Y. (2016). Vector Field Processing on Triangle Meshes. https://doi.org/10.1145/2897826.2927310
- Diamanti, O., Vaxman, A., Panozzo, D., & Sorkine-Hornung, O. (2014). Designing N-polyvector fields with complex polynomials. Eurographics Symposium on Geometry Processing, 33(5), 1–11. https://doi.org/10.1111/cgf.12426
- Crane, K., Desbrun, M., & Schröder, P. (2010). Trivial connections on discrete surfaces. Eurographics Symposium on Geometry Processing, 29(5), 1525–1533. https://doi.org/10.1111/j.1467-8659.2010.01761.x
- Knöppel, F., Crane, K., Pinkall, U., & Schröder, P. (2013). Globally optimal direction fields. ACM Transactions on Graphics, 32(4), 1. https://doi.org/10.1145/2461912.2462005
- Brandt, C., Scandolo, L., Eisemann, E., Hildebrandt, K., & Hilde, K. (2018). Modeling n-Symmetry Vector Fields using Higher-Order Energies. ACM Trans. Graph. Article, 37(18). https://doi.org/10.1145/3177750