

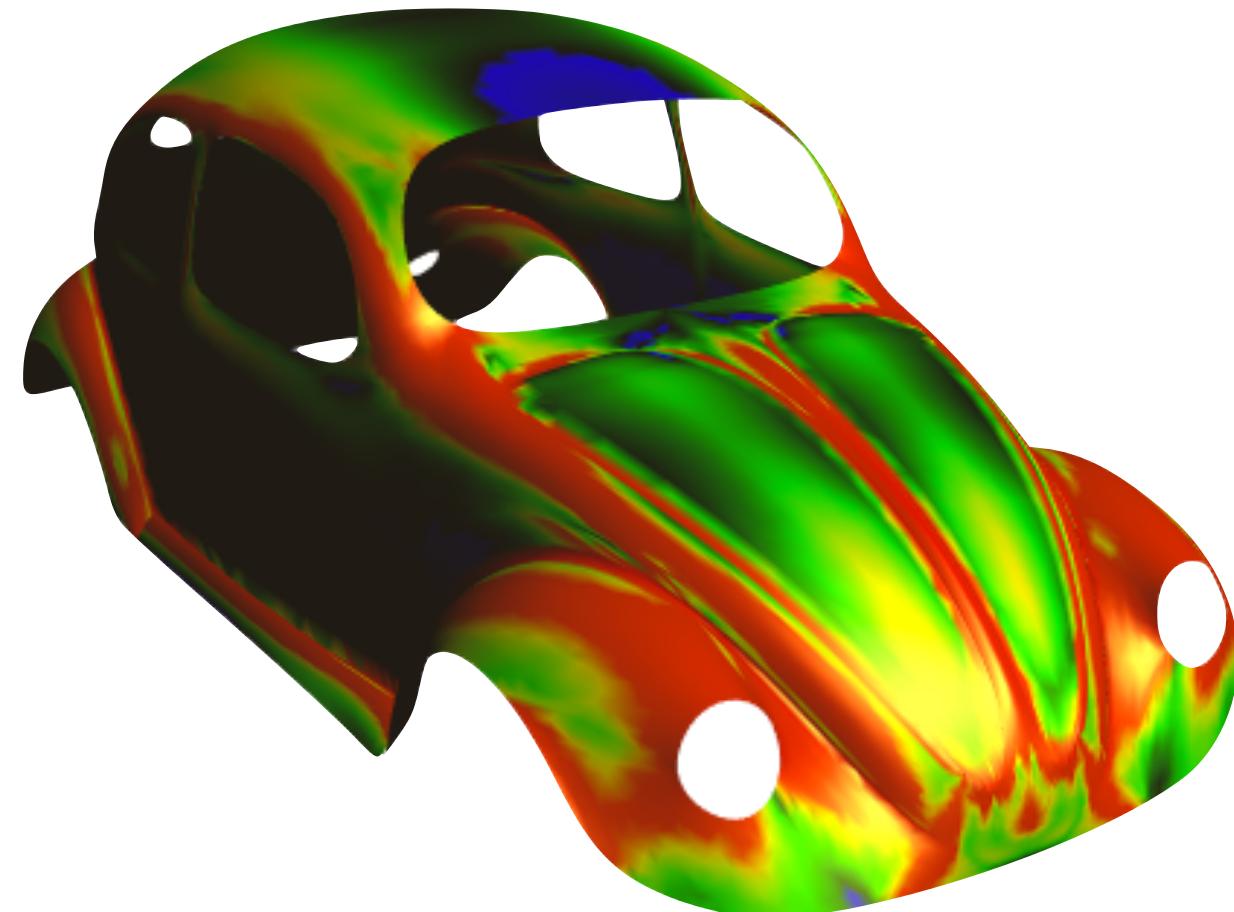
Cotangent Laplacian Derivation

Acknowledgements: Olga Diamanti, Julian Panetta
CSC 486B/586B - Geometric Modeling - Teseo Schneider

Cotan Laplacian Motivation

- Discrete Laplacian enables many useful operations on triangle meshes:

$$H_i = \frac{1}{2} \left\| [\mathbf{L}\mathbf{v}]_i \right\|$$



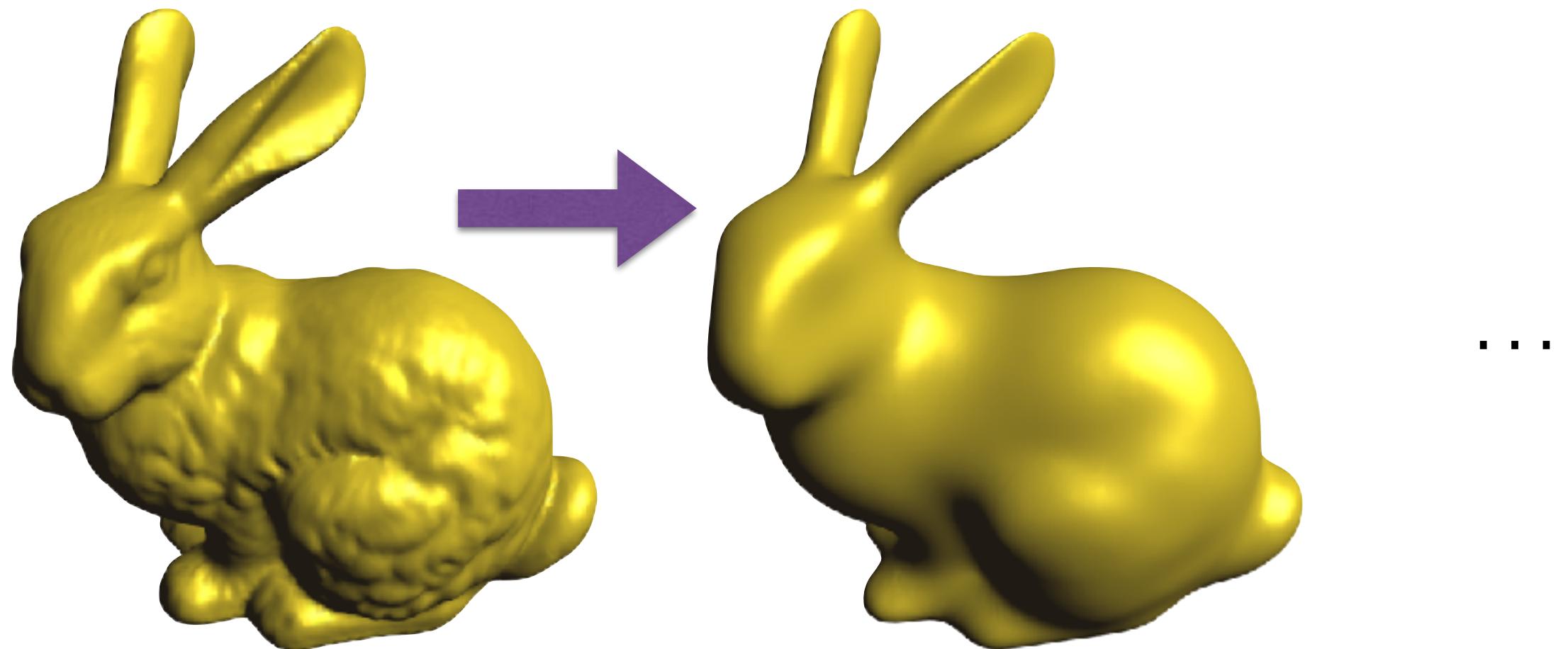
Mean Curvature

$$\mathbf{n} = -\frac{[\mathbf{L}\mathbf{v}]_i}{2H_i}$$



Normal Computation

$$(I - \lambda dt \mathbf{L})\mathbf{v}^{n+1} = \mathbf{v}^n$$

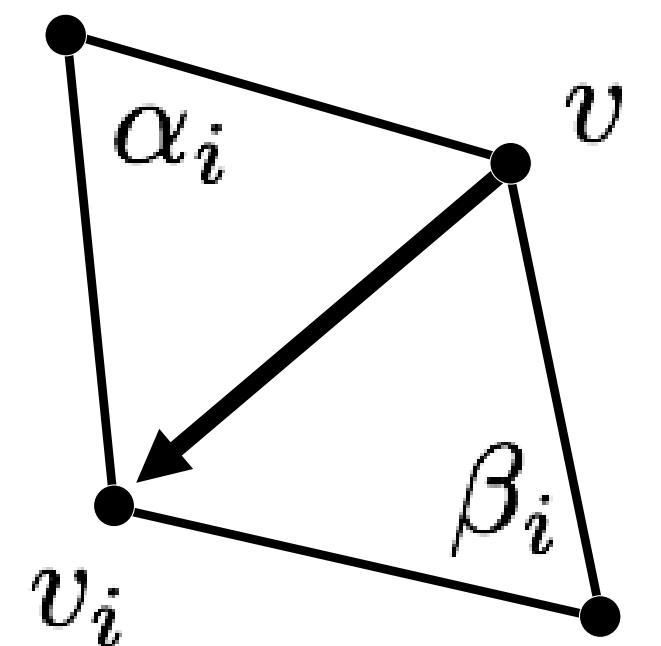


Smoothing

- Cotangent weights give less mesh-dependent results, ensuring proper convergence under mesh refinement

Where We're Headed

$$\Delta_M^{\text{cotan}} f(\mathbf{v}) = \frac{1}{2A(\mathbf{v})} \sum_{\mathbf{v}_i \in N(\mathbf{v})} (\cot(\alpha_i) + \cot(\beta_i)) \left(f(\mathbf{v}_i) - f(\mathbf{v}) \right)$$



- “Cotan Laplacian” refers to $\frac{1}{2}(\cot(\alpha_i) + \cot(\beta_i))$ weight on each edge contribution.
- We’ll consider two different ways to derive this formula:
 - “Traditional graphics approach:” Finite Volume-style
 - Finite Element Approach (more general)

Goal: Mesh Independence

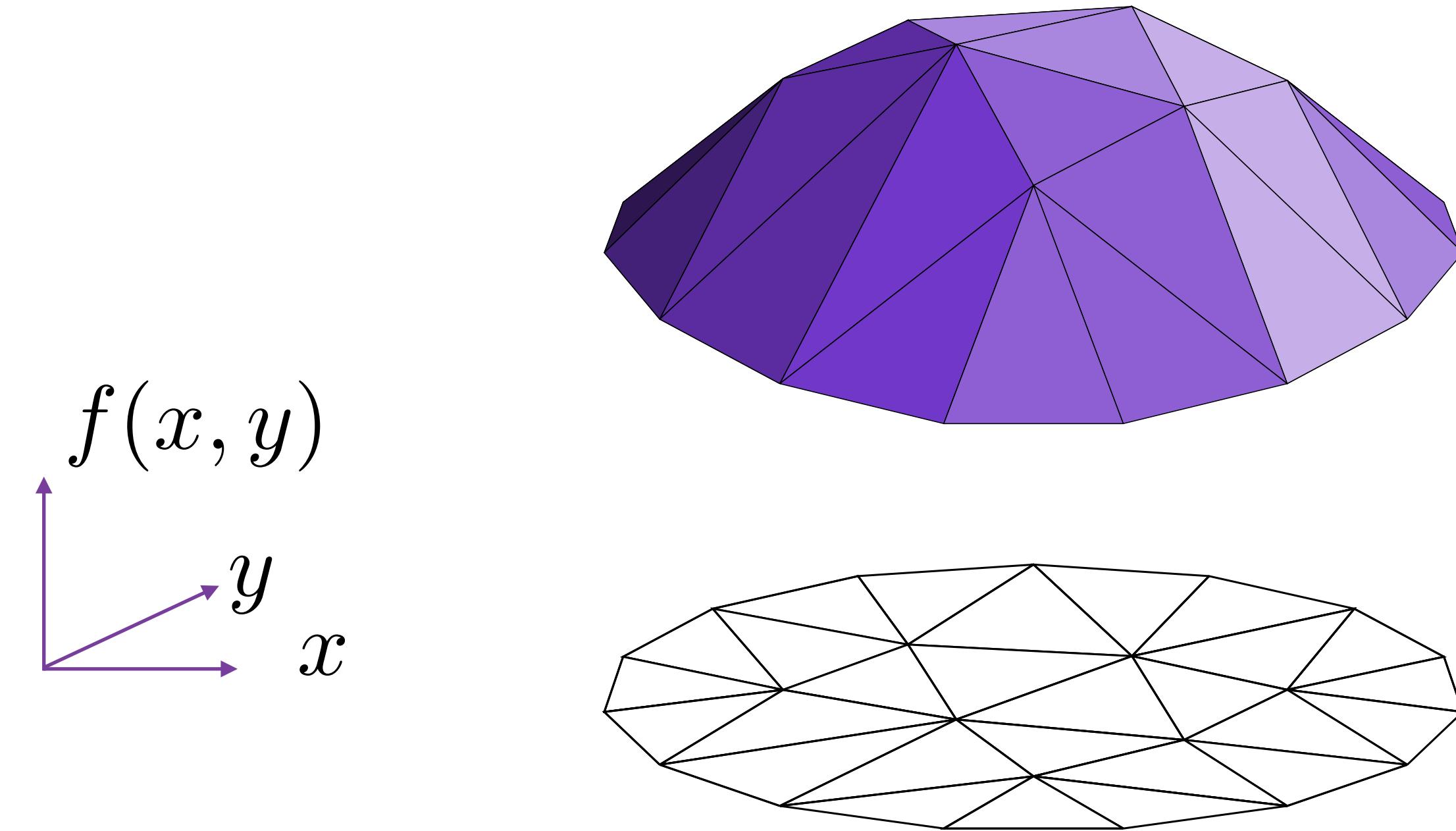
- We want discrete Laplacian to **converge to the Laplace Beltrami** operator of smooth surface S as mesh M is refined to better approximate S .

$$\Delta_M^{\text{cotan}} \mathbf{F} \longrightarrow \Delta_S f \quad \mathbf{F} = \begin{bmatrix} f(\mathbf{v}_1) \\ f(\mathbf{v}_2) \\ f(\mathbf{v}_3) \\ \vdots \\ f(\mathbf{v}_N) \end{bmatrix}$$

- \mathbf{F} is a vector of discrete values (per vertex),
 $f(x)$ is a smooth function.
- **Idea:** reinterpret \mathbf{F} as a **function interpolating the smooth "f,"** and directly compute its Laplacian. Converge as $\mathbf{F} \rightarrow f$

Piecewise-Linear Interpolation

- Since \mathbf{F} holds per-vertex values, it naturally corresponds to a **linear function on each triangle**



- (Corner values uniquely determine linear function over triangle)

“Hat Function” Basis

- Piecewise linear functions are expressed most simply using the **“hat functions”** (linear Lagrange polynomials)

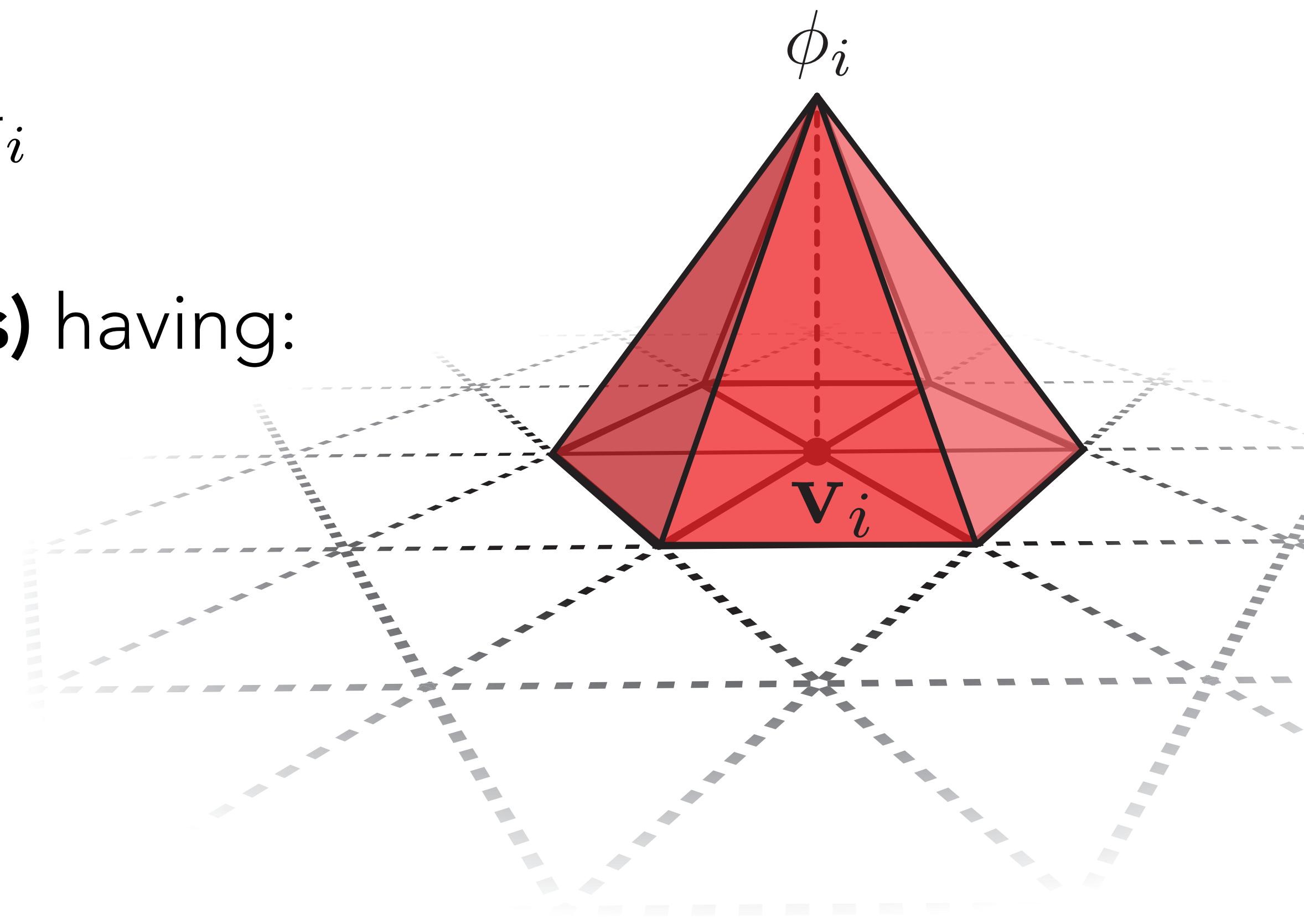
- One basis function $\phi_i(\mathbf{x})$ per vertex, \mathbf{v}_i

- **Unique linear function (on triangles) having:**

$$\phi_i(\mathbf{v}_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

- Interpolation of \mathbf{F} on mesh M is just:

$$f_M(\mathbf{x}) = \sum_i f(\mathbf{v}_i) \phi_i(\mathbf{x})$$



Gradients

- Recall, the Laplacian is the divergence of the gradient:

$$\Delta f = \nabla \cdot (\nabla f)$$

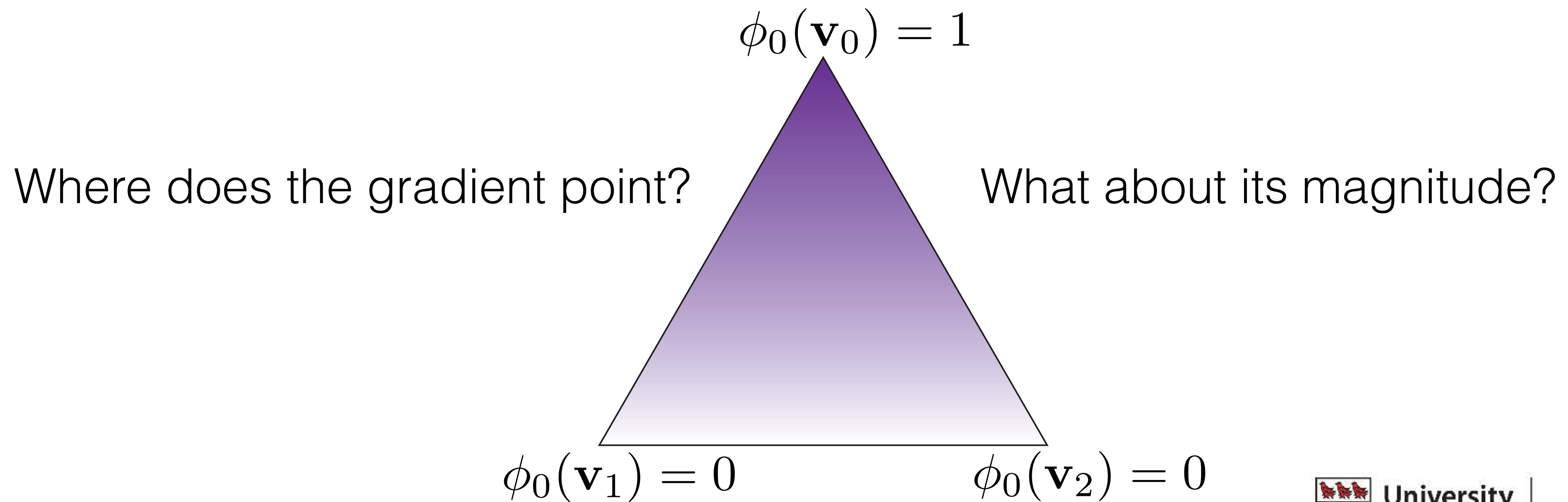
- So we first need to compute our interpolant's **gradient**:

$$\nabla f_M(\mathbf{x}) = \nabla \left(\sum_i f(\mathbf{v}_i) \phi_i(\mathbf{x}) \right) = \sum_i f(\mathbf{v}_i) \nabla \phi_i(\mathbf{x})$$

- In other words, the gradient is just a linear combination of the **basis function gradients** $\nabla \phi_i(\mathbf{x})$ weighted by the per-vertex f values.

Hat Function Gradients

- Basis functions are linear over each triangle, so these gradients are **constant on each triangle**
- We can determine gradient with simple geometric arguments:



Gradient Derivation

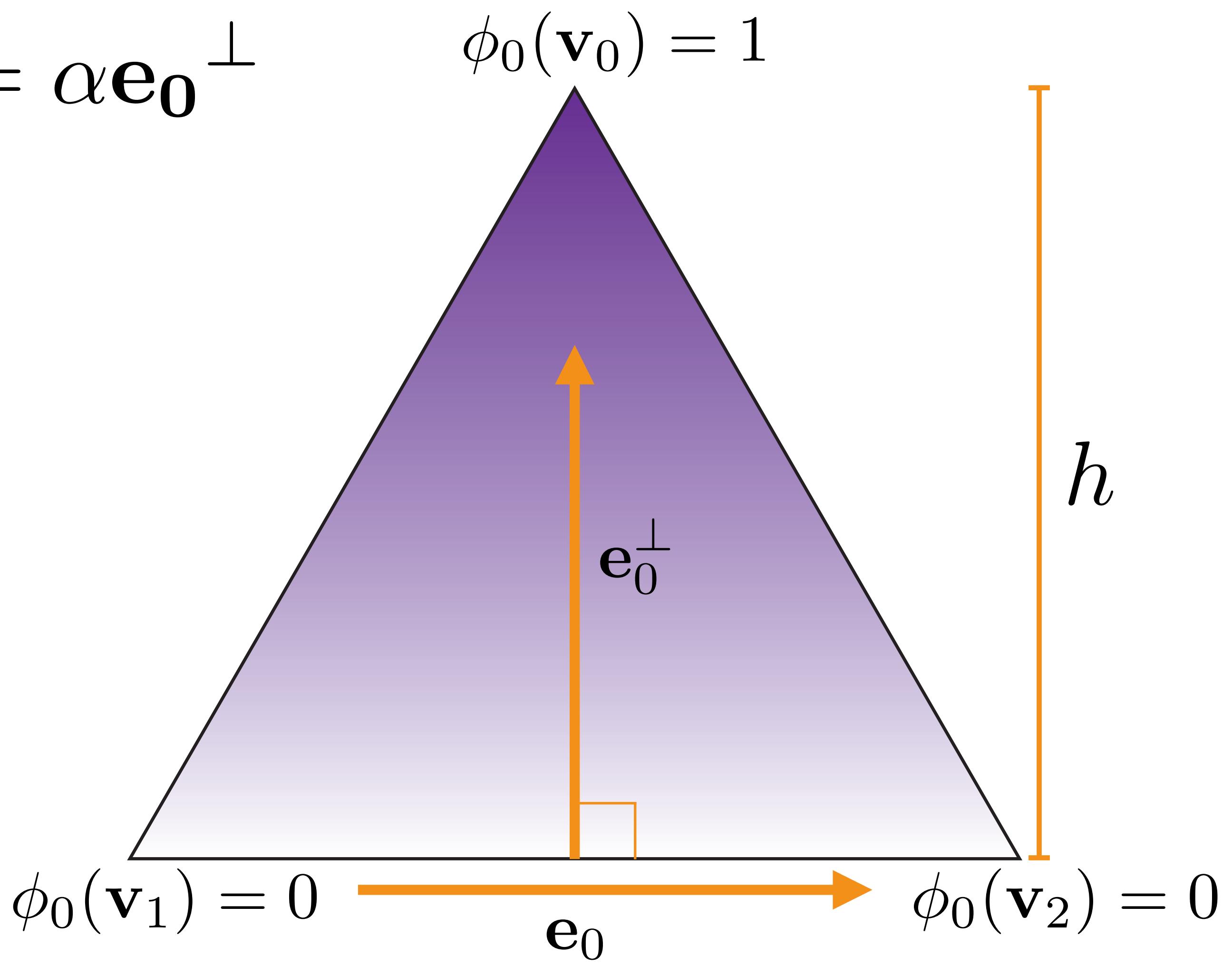
$$\nabla \phi_0 \cdot \mathbf{e}_0 = 0 \implies \nabla \phi_0 = \alpha \mathbf{e}_0^\perp$$

$$\nabla \phi_0 \cdot \left(h \frac{\mathbf{e}_0^\perp}{\|\mathbf{e}_0^\perp\|} \right) = 1$$

$$\implies \nabla \phi_0 = \frac{1}{h} \frac{\mathbf{e}_0^\perp}{\|\mathbf{e}_0^\perp\|}$$

$$\implies \nabla \phi_0 = \frac{\mathbf{e}_0^\perp}{2A}$$

(area is $1/2 * \text{base} * \text{height}$)

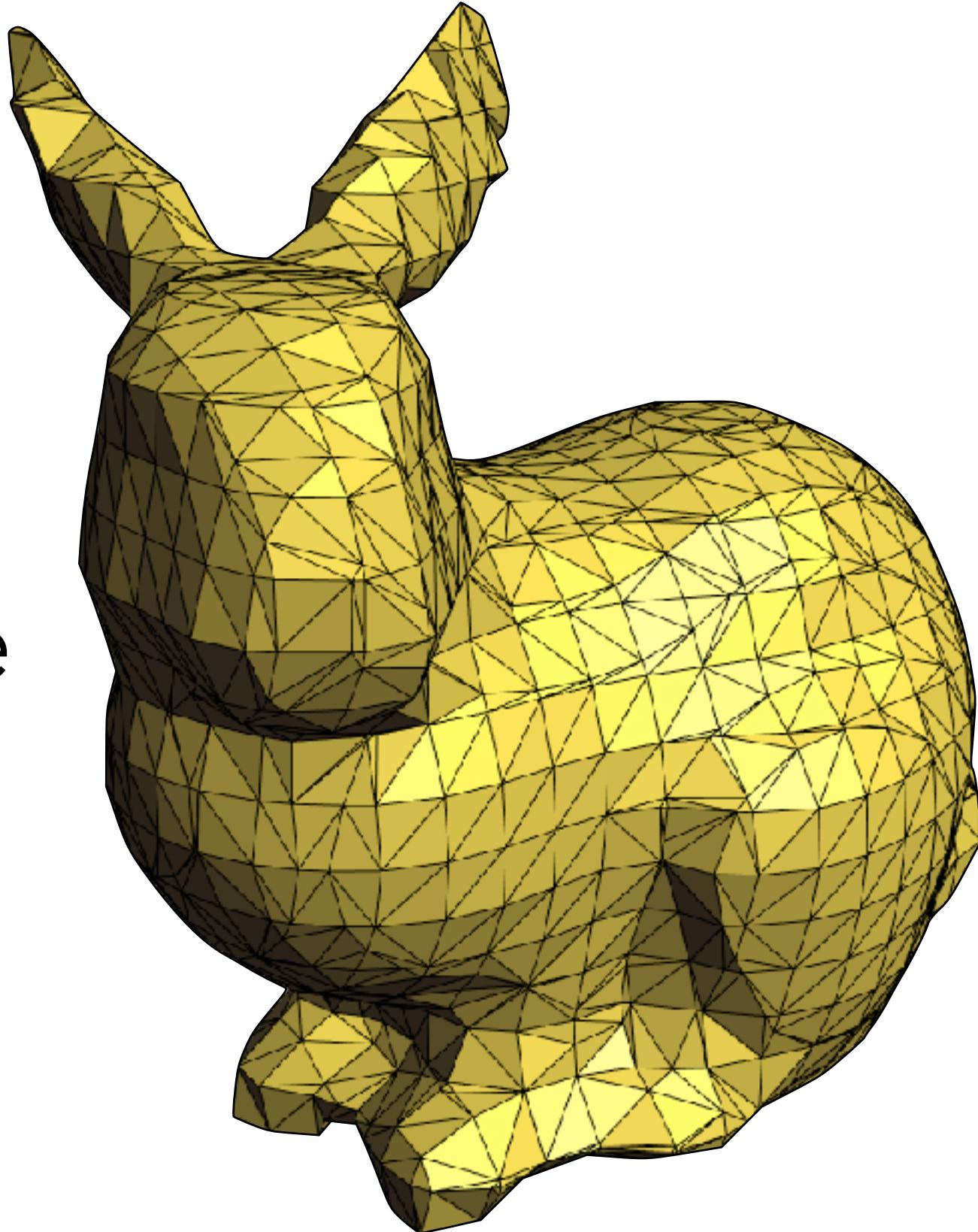


Final Gradient Expression

$$\nabla \phi_i|_T = \frac{\mathbf{e}_i^\perp}{2A_T}$$

- Note, this is the gradient **evaluated on one triangle T incident vertex i**. (Different gradient on others).
- Works on curved triangle meshes too! Just need triangle area and inward-pointing perpendicular edge vectors:
$$\mathbf{e}_i^\perp = \mathbf{n} \times \mathbf{e}_i$$
- Full gradient is then the linear combination

$$\nabla f_M(\mathbf{x}) = \sum_i f(\mathbf{v}_i) \nabla \phi_i(\mathbf{x})$$

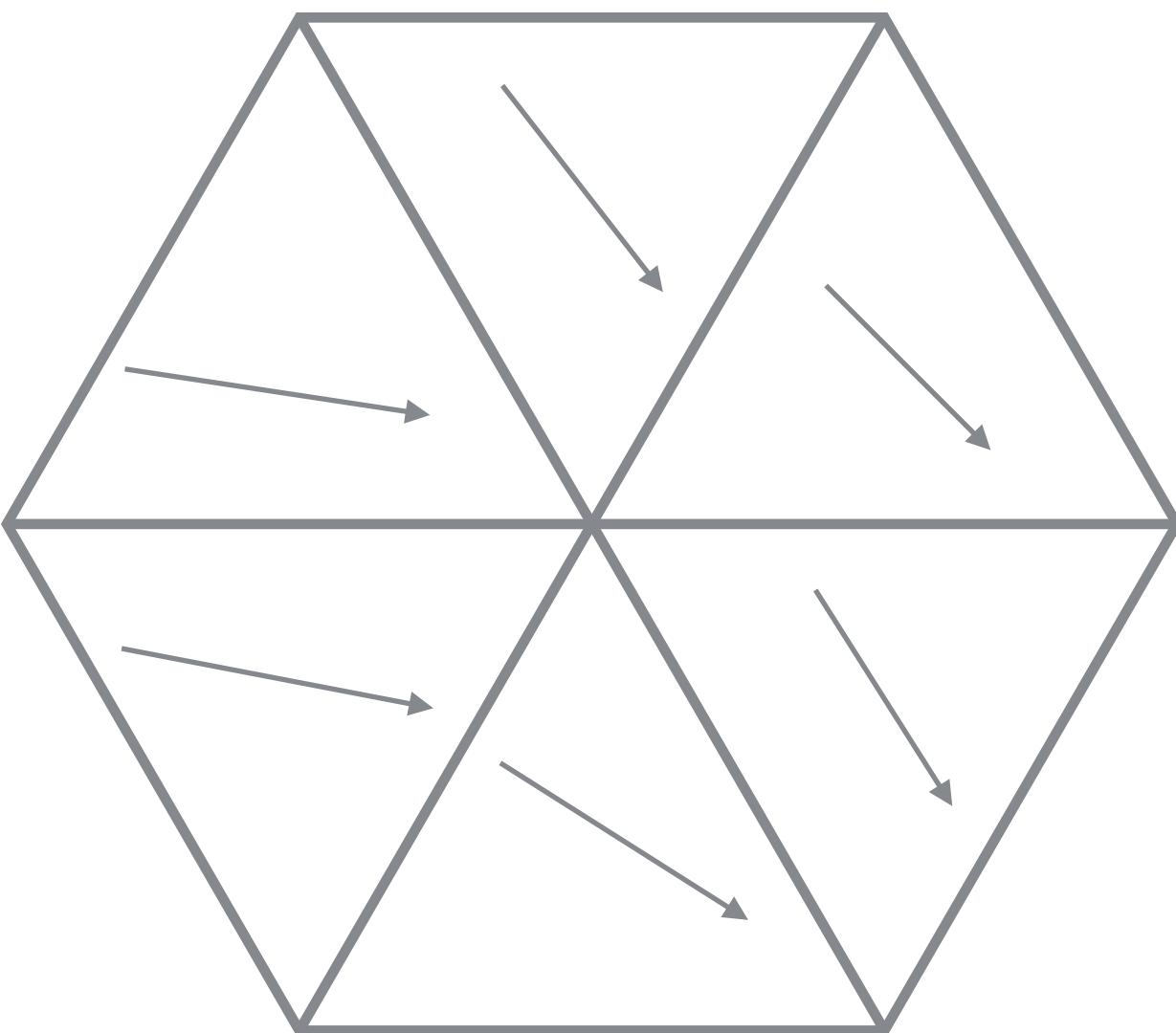


Now the Laplacian!

- Now we want to compute

$$\Delta f_M = \nabla \cdot \nabla f_M(\mathbf{x}) = \nabla \cdot \left(\sum_i f(\mathbf{v}_i) \nabla \phi_i(\mathbf{x}) \right)$$

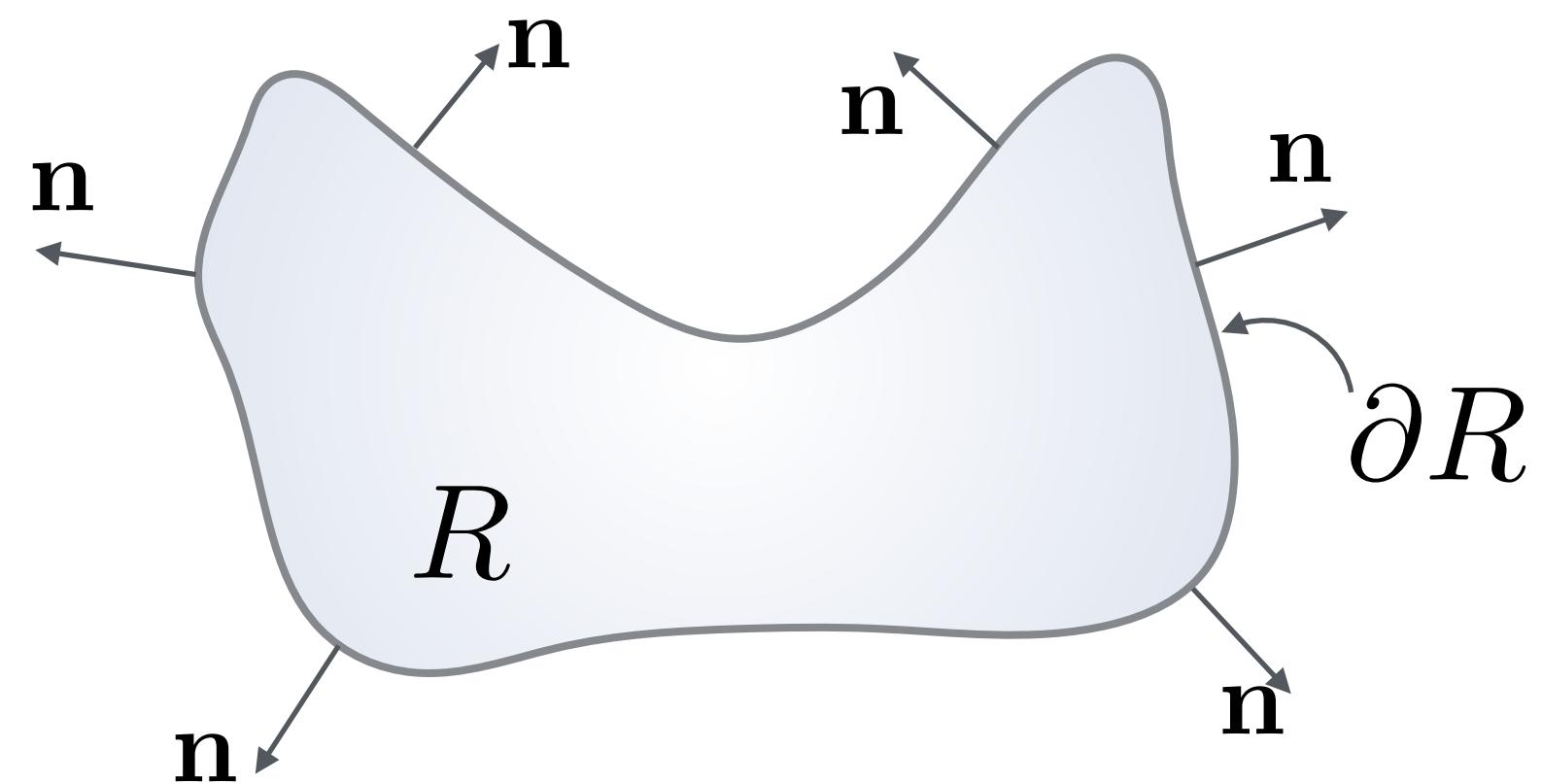
- But the gradient is piecewise constant!
What does divergence even mean?



Too Many Derivatives!

- **Problem:** function is piecewise linear (only C⁰ continuous) and we're trying to take second derivatives
- But we can make sense of things by using the Divergence Theorem over some region "R":

$$\int_R \nabla \cdot (\nabla \phi) d\mathbf{x} = \int_{\partial R} \mathbf{n} \cdot \nabla \phi ds$$

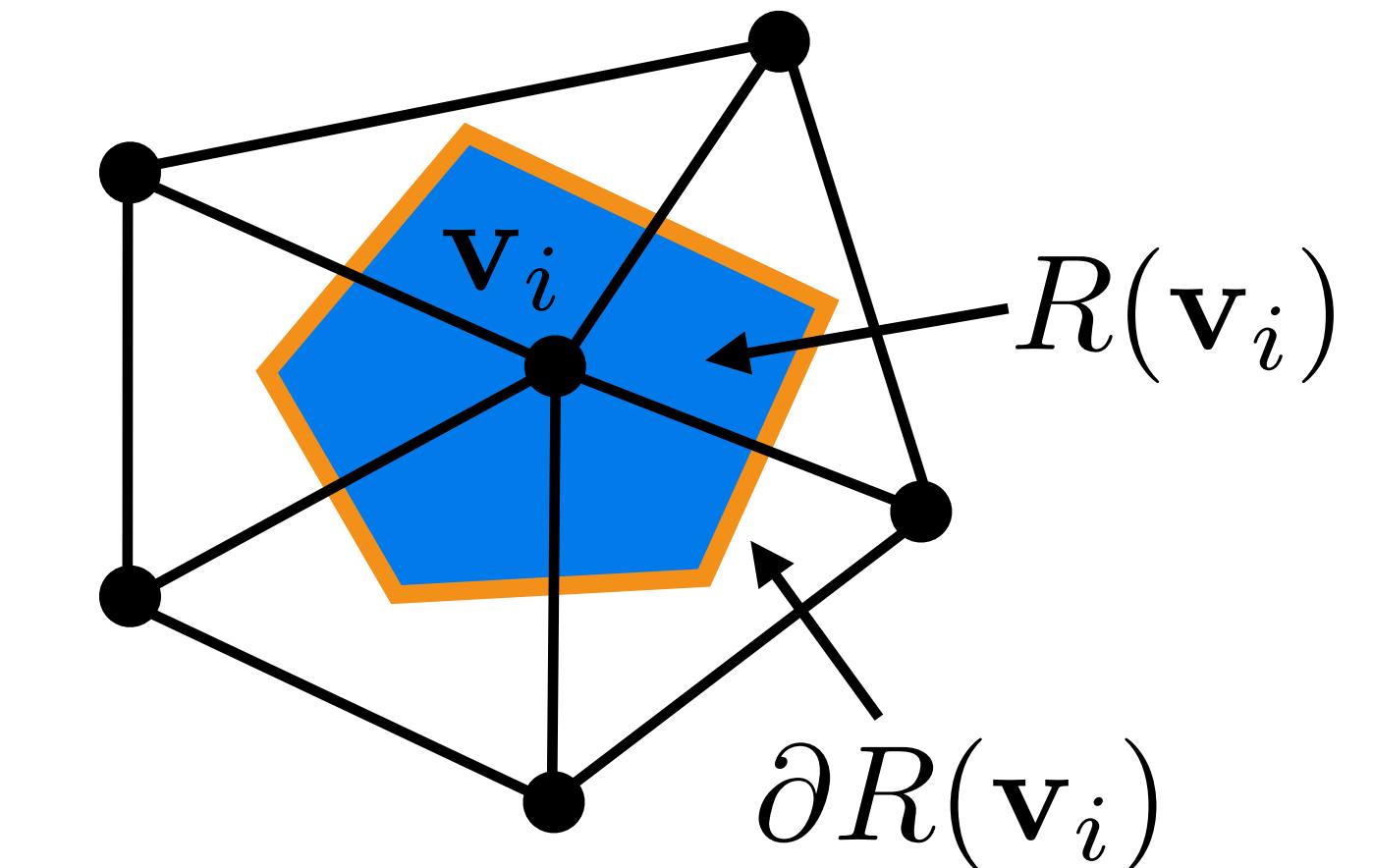


- (This allows us to avoid explicitly computing divergence; just need dot products with the curve normals on R 's boundary.)

Approach 1: (Traditional in Graphics)

- Idea: define Laplacian on a vertex to be the **average over some region surrounding the vertex.**

$$\begin{aligned}[L\mathbf{F}]_i &:= \frac{1}{|R(\mathbf{v}_i)|} \int_{R(\mathbf{v}_i)} \nabla \cdot \nabla f_M(\mathbf{x}) d\mathbf{x} \\ &= \frac{1}{|R(\mathbf{v}_i)|} \int_{\partial R(\mathbf{v}_i)} \mathbf{n} \cdot \nabla f_M(\mathbf{x}) ds\end{aligned}$$

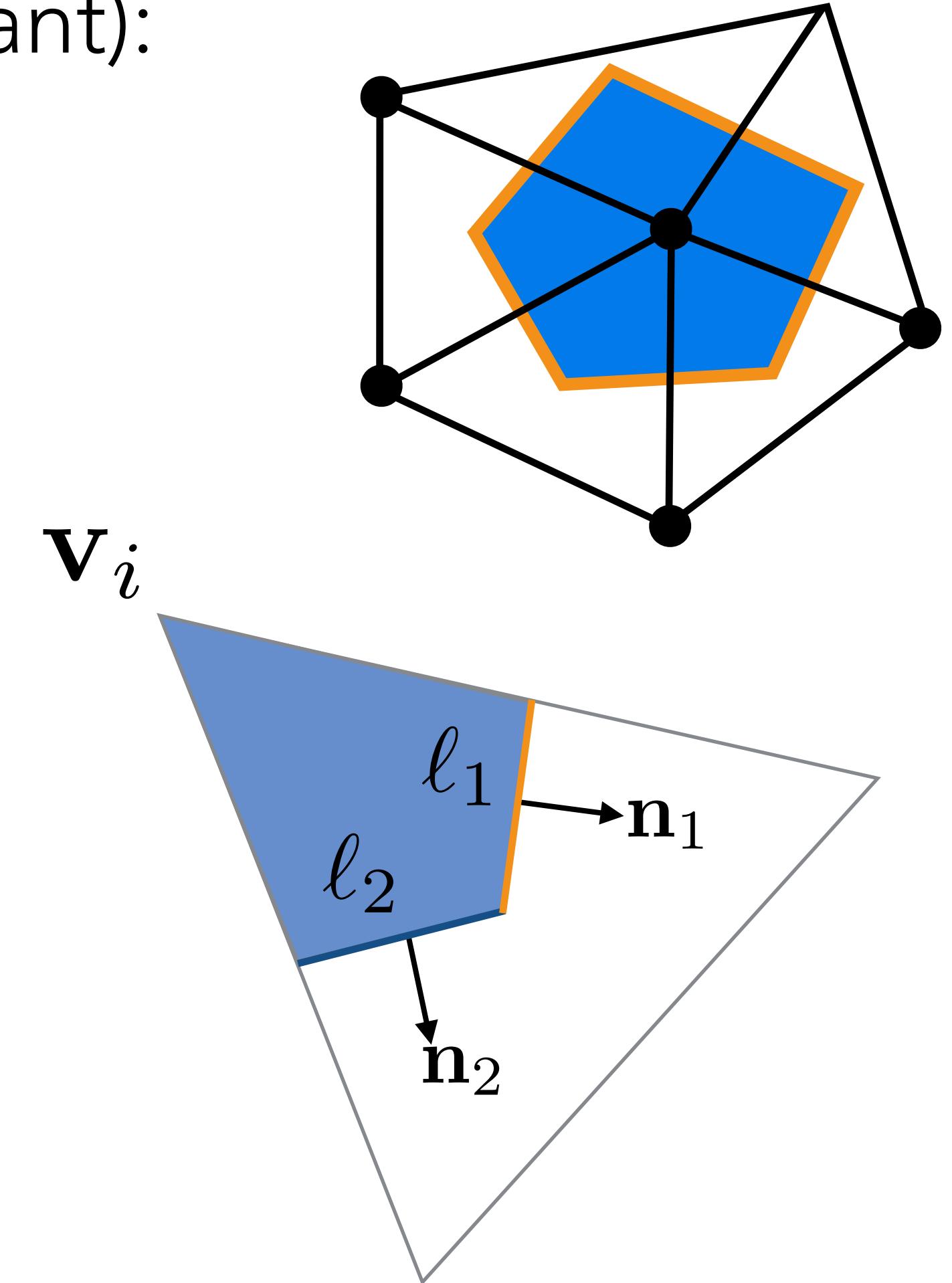


- To get cotan Laplacian, you need to average over the **vertices' Voronoi regions** (region for which v is the closest vtx).
- This region seems somewhat arbitrary; approach 2 won't need this choice.

Approach 1 Continued

- Break into sum over triangles (so gradient is constant):

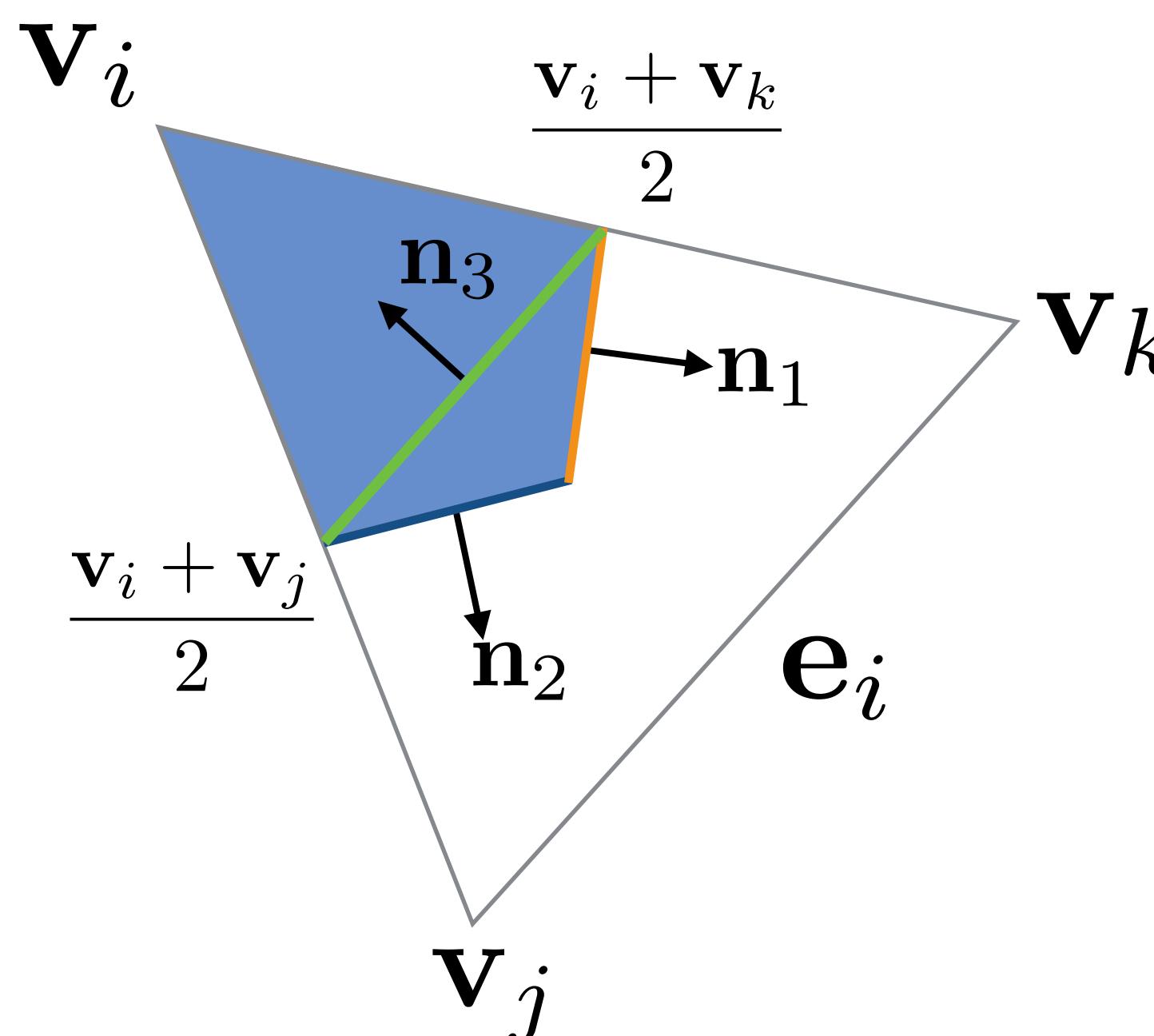
$$\begin{aligned}
 [L\mathbf{F}]_i &= \frac{1}{|R(\mathbf{v}_i)|} \int_{\partial R(\mathbf{v}_i)} \mathbf{n} \cdot \nabla f_M(\mathbf{x}) \, ds \\
 &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \int_{T \cap \partial R(\mathbf{v}_i)} \mathbf{n} \cdot \nabla f_M(\mathbf{x}) \, ds \\
 &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \nabla f_T \cdot \int_{T \cap \partial R(\mathbf{v}_i)} \mathbf{n} \, ds \\
 &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \nabla f_T \cdot (\mathbf{n}_1 \ell_1 + \mathbf{n}_2 \ell_2)
 \end{aligned}$$



Voronoi Diagram Properties

$$[L\mathbf{F}]_i = \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \nabla f_T \cdot (\mathbf{n}_1 \ell_1 + \mathbf{n}_2 \ell_2)$$

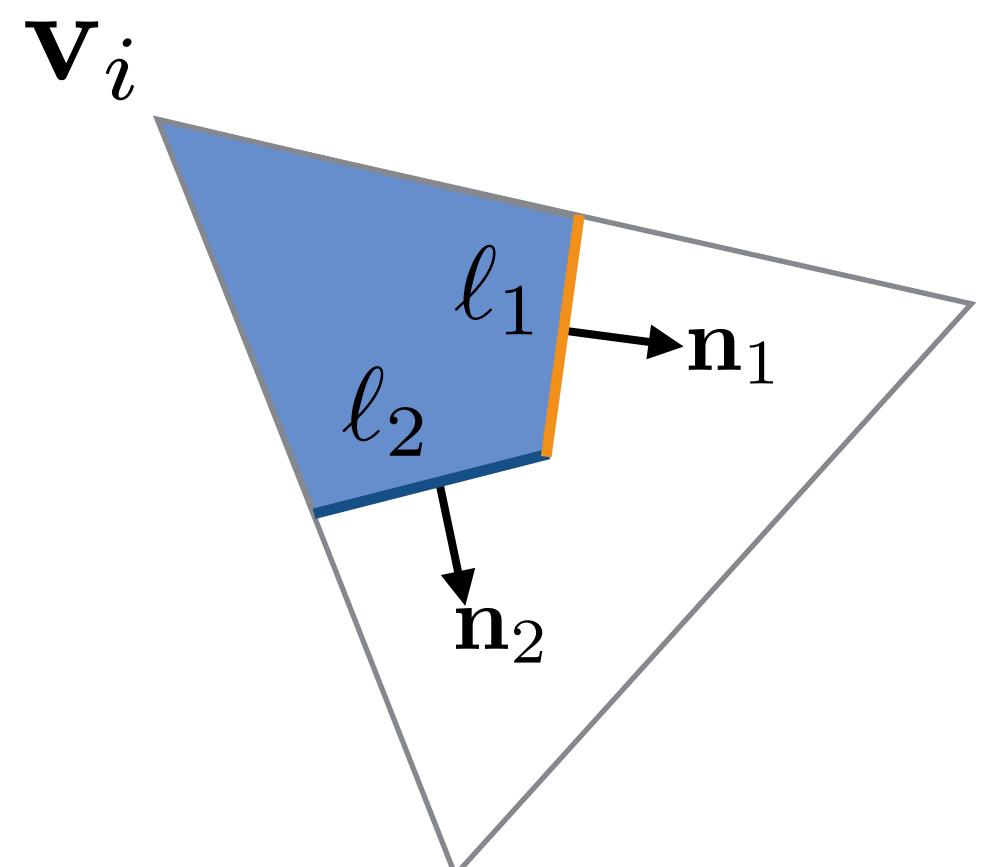
- Voronoi diagram edges **bisect** \mathbf{v}_i 's incident edges



$$\mathbf{n}_1 \ell_1 + \mathbf{n}_2 \ell_2 + \mathbf{n}_3 \ell_3 = 0$$

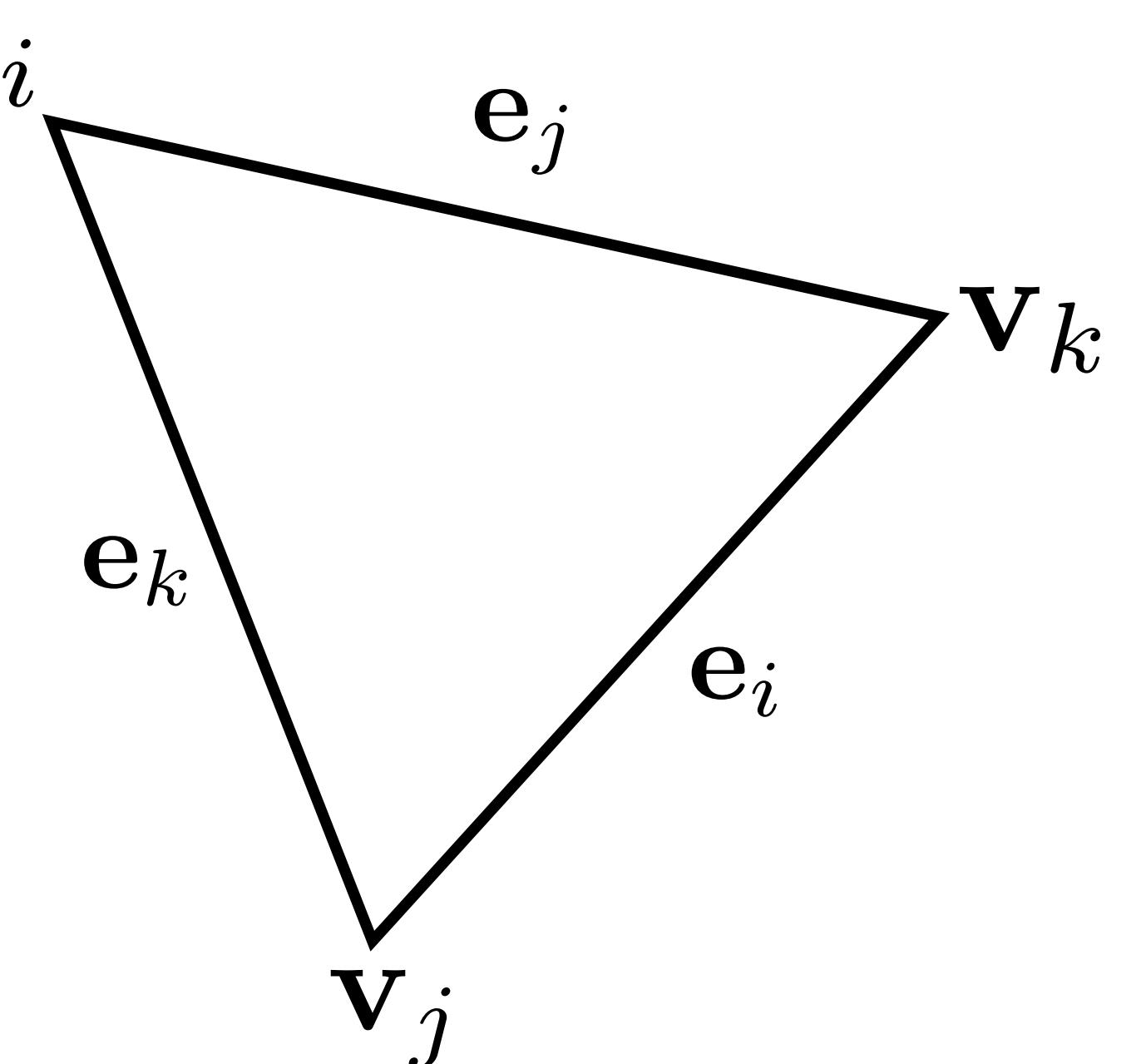
(Tri edge vectors sum to zero; so do their perpendiculars.)

$$\begin{aligned} \Rightarrow \quad \mathbf{n}_1 \ell_1 + \mathbf{n}_2 \ell_2 &= -\mathbf{n}_3 \ell_3 = \left(\frac{\mathbf{v}_k + \mathbf{v}_i}{2} - \frac{\mathbf{v}_j + \mathbf{v}_i}{2} \right)^\perp \\ &= \frac{1}{2} (\mathbf{v}_k - \mathbf{v}_j)^\perp = \frac{\mathbf{e}_i^\perp}{2} \\ \Rightarrow \quad [L\mathbf{F}]_i &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \nabla f_T \cdot \frac{\mathbf{e}_i^\perp}{2} \end{aligned}$$



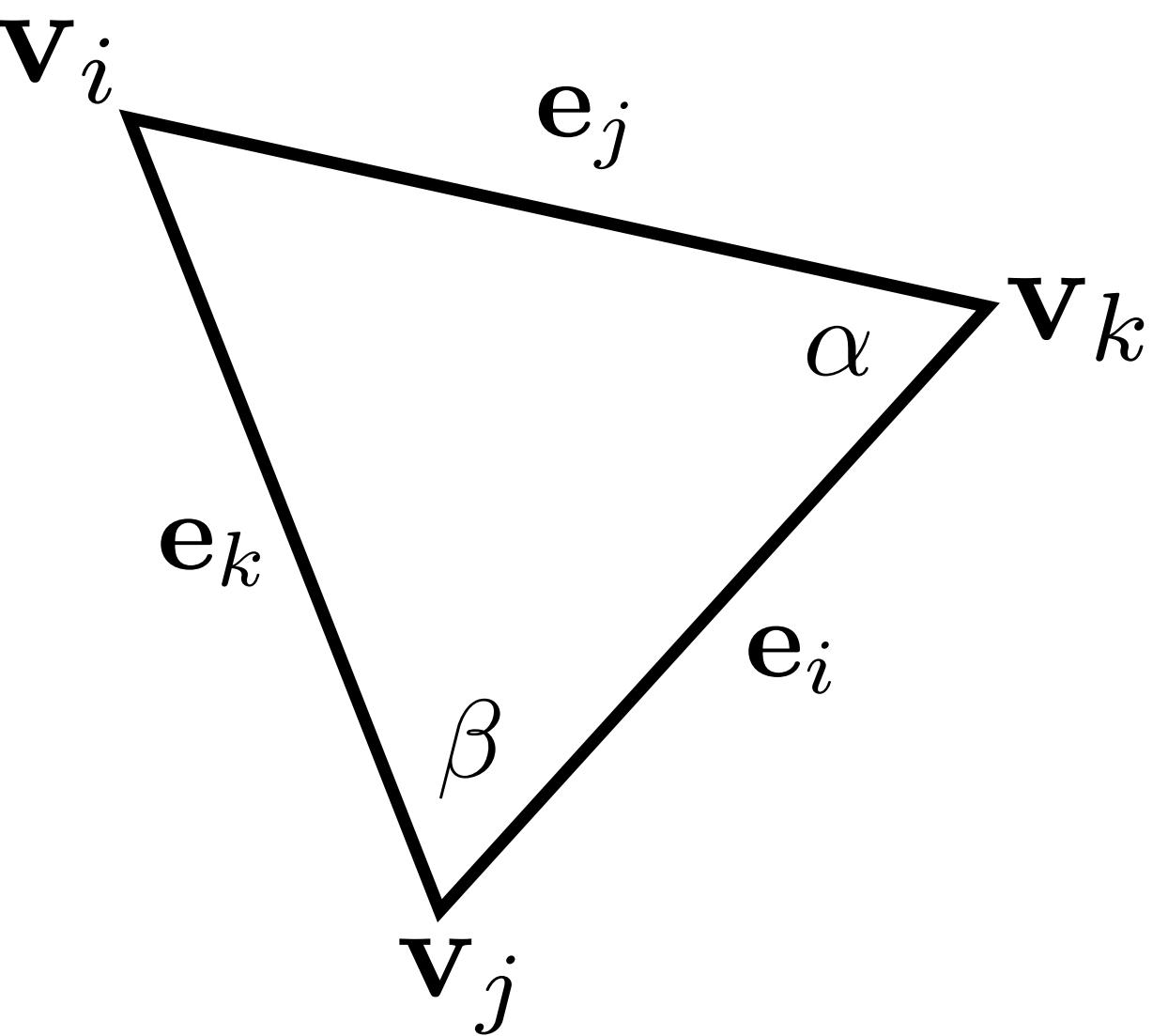
Plug in per-Tri Gradient

$$\begin{aligned}
 [L\mathbf{F}]_i &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \nabla f_T \cdot \frac{\mathbf{e}_i^\perp}{2} \\
 &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} (f_i \nabla \phi_i + f_j \nabla \phi_j + f_k \nabla \phi_k) \cdot \frac{\mathbf{e}_i^\perp}{2} \\
 &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \left[f_i \frac{\mathbf{e}_i^\perp}{2A_T} + f_j \frac{\mathbf{e}_j^\perp}{2A_T} + f_k \frac{\mathbf{e}_k^\perp}{2A_T} \right] \cdot \frac{\mathbf{e}_i^\perp}{2} \\
 &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \left[-f_i \left(\frac{\mathbf{e}_j^\perp}{2A_T} + \frac{\mathbf{e}_k^\perp}{2A_T} \right) + f_j \frac{\mathbf{e}_j^\perp}{2A_T} + f_k \frac{\mathbf{e}_k^\perp}{2A_T} \right] \cdot \frac{\mathbf{e}_i^\perp}{2} \quad \text{(edges sum to zero: } \mathbf{e}_i^\perp + \mathbf{e}_j^\perp + \mathbf{e}_k^\perp = 0\text{)} \\
 &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \left[(f_j - f_i) \frac{\mathbf{e}_j^\perp}{2A_T} + (f_k - f_i) \frac{\mathbf{e}_k^\perp}{2A_T} \right] \cdot \frac{\mathbf{e}_i^\perp}{2}
 \end{aligned}$$



Final Steps

$$\begin{aligned}
 [L\mathbf{F}]_i &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} \left[(f_j - f_i) \frac{\mathbf{e}_j^\perp}{2A_T} + (f_k - f_i) \frac{\mathbf{e}_k^\perp}{2A_T} \right] \cdot \frac{\mathbf{e}_i^\perp}{2} \\
 &= \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} (f_j - f_i) \frac{\mathbf{e}_j^\perp \cdot \mathbf{e}_i^\perp}{4A_T} + (f_k - f_i) \frac{\mathbf{e}_k^\perp \cdot \mathbf{e}_i^\perp}{4A_T}
 \end{aligned}$$



Note:

$$\frac{\mathbf{e}_i^\perp \cdot \mathbf{e}_j^\perp}{4A_T} = \frac{\|\mathbf{e}_i\| \|\mathbf{e}_j\| \cos(\alpha)}{2\|\mathbf{e}_i\| \|\mathbf{e}_j\| \sin(\alpha)} = \frac{1}{2} \cot(\alpha)$$

$$\frac{\mathbf{e}_i^\perp \cdot \mathbf{e}_k^\perp}{4A_T} = \frac{\|\mathbf{e}_i\| \|\mathbf{e}_k\| \cos(\beta)}{2\|\mathbf{e}_i\| \|\mathbf{e}_k\| \sin(\beta)} = \frac{1}{2} \cot(\beta)$$

$$[L\mathbf{F}]_i = \frac{1}{|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} (f_j - f_i) \cot(\alpha) + (f_k - f_i) \cot(\beta)$$

Regrouping Terms

- We have the cotan weights now, but expressed as a sum over triangles:

$$[L\mathbf{F}]_i = \frac{1}{2|R(\mathbf{v}_i)|} \sum_{T \ni \mathbf{v}_i} (f_j - f_i) \cot \alpha + (f_k - f_i) \cot \beta$$

- To get the standard formula, we can regroup these terms as a sum over edges:

$$[L\mathbf{F}]_i = \frac{1}{2|R(\mathbf{v}_i)|} \sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} (\cot(\alpha_{ij}) + \cot(\beta_{ij})) \left(f(\mathbf{v}_j) - f(\mathbf{v}_i) \right)$$

- $|R(\mathbf{v}_i)|$ is the area of the i^{th} vertex's Voronoi region (the averaging area)

Approach 2: Finite Element Method

- Instead of trying to discretize the Laplacian itself (using averaging regions), we **discretize the whole PDE** containing it. For instance:

$$-\Delta u = b \quad \text{in } M$$

- Here “b” is a forcing term (e.g. injecting/sinking heat at each point), and we assume M is a closed surface mesh without a boundary.
- If M had boundary curves, we’d need to specify boundary conditions.

Weak Form of the Poisson Equation

- We can't expect a piecewise linear function "u" to solve this PDE exactly (why does its Laplacian mean, anyway?)
- But we can ask it to be a "weak solution:"

$$-\int_M \psi \Delta u \, d\mathbf{x} = \int_M \psi b \, d\mathbf{x} \quad \forall \psi \text{ "test functions"}$$

- In other words, we ask for zero residual "along" all test functions.
- Linear FEM: choose piecewise linear functions for both u and ψ
- Physical interpretation: minimizing a potential energy over the space of piecewise linear functions

Expand Functions in Basis

$$-\int_M \psi \Delta u \, d\mathbf{x} = \int_M \psi b \, d\mathbf{x}$$

$$u(\mathbf{x}) := \sum_j u_j \phi_j(\mathbf{x}) \quad \psi(\mathbf{x}) := \sum_i a_i \phi_i(\mathbf{x}) \quad b(\mathbf{x}) := \sum_j b_j \phi_j(\mathbf{x})$$

$$\begin{aligned} -\int_M \left(\sum_i a_i \phi_i(\mathbf{x}) \right) \Delta \left(\sum_j u_j \phi_j(\mathbf{x}) \right) \, d\mathbf{x} &= \int_M \left(\sum_i a_i \phi_i(\mathbf{x}) \right) \left(\sum_j b_j \phi_j(\mathbf{x}) \right) \, d\mathbf{x} \\ \implies -\sum_{i,j} a_i u_j \left(\boxed{\int_M \phi_i \Delta \phi_j \, d\mathbf{x}} \right) &= \sum_{i,j} a_i b_j \left(\int_M \phi_i \phi_j \, d\mathbf{x} \right) \end{aligned}$$

Integration by Parts for

$$\int_M \phi_i \Delta \phi_j \, d\mathbf{x}$$

- To avoid second derivatives, apply **divergence theorem** in the form of an **integration by parts** (move derivative onto ϕ_i)
- Note the “product rule”: $\nabla \cdot (\phi_i \nabla \phi_j) = \phi_i \Delta \phi_j + \nabla \phi_i \cdot \nabla \phi_j \implies \phi_i \Delta \phi_j = \nabla \cdot (\phi_i \nabla \phi_j) - \nabla \phi_i \cdot \nabla \phi_j$

Apply divergence theorem

$$\begin{aligned} - \int_M \phi_i \Delta \phi_j \, d\mathbf{x} &= - \int_M \boxed{\nabla \cdot (\phi_i \nabla \phi_j)} - \nabla \phi_i \cdot \nabla \phi_j \, d\mathbf{x} \\ &= - \int_{\partial M} \mathbf{n} \cdot (\phi_i \nabla \phi_j) \, ds + \int_M \nabla \phi_i \cdot \nabla \phi_j \, d\mathbf{x} \\ &= \int_M \nabla \phi_i \cdot \nabla \phi_j \, d\mathbf{x} \end{aligned}$$

There is no boundary!

Laplacian and Mass Matrices

- Now our weak PDE looks like:

$$\sum_{i,j} a_i u_j \left(\underbrace{\int_M \nabla \phi_i \nabla \phi_j \, d\mathbf{x}}_{L_{ij}} \right) = \sum_{i,j} a_i b_j \left(\underbrace{\int_M \phi_i \phi_j \, d\mathbf{x}}_{\mathcal{M}_{ij}} \right)$$

- Expressed in matrix form:

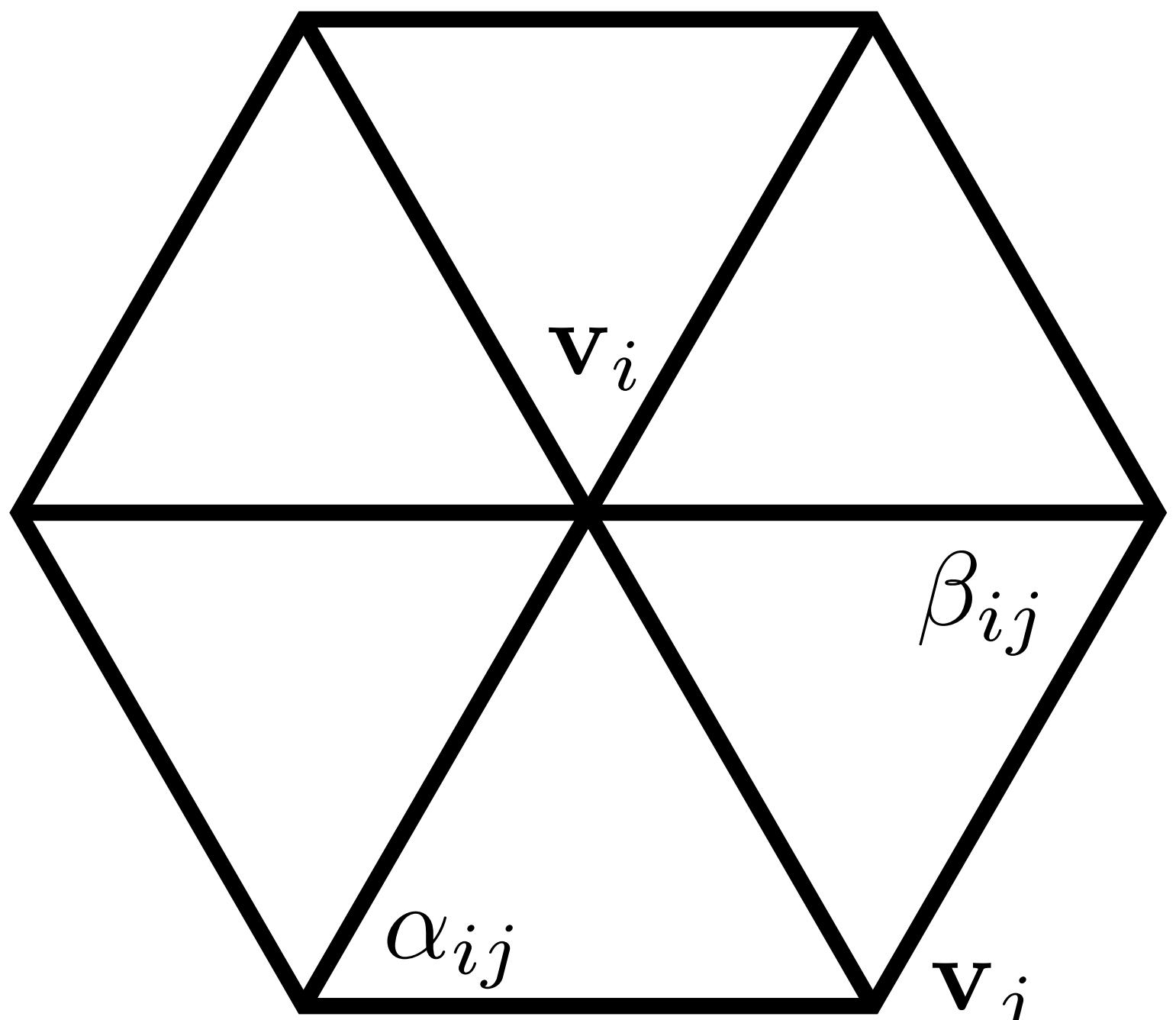
$$\mathbf{a}^T L \mathbf{u} = \mathbf{a}^T \mathcal{M} \mathbf{b} \quad \boxed{\forall \mathbf{a}} \quad \text{Weak form must hold for all test functions!}$$
$$\implies L \mathbf{u} = \mathcal{M} \mathbf{b}$$

- Plugging in gradient expressions:

$$L_{ij} = \int_M \nabla \phi_i \cdot \nabla \phi_j \, d\mathbf{x} = \sum_{T \ni \mathbf{v}_i, \mathbf{v}_j} \frac{\mathbf{e}_{T,i}^\perp}{2A_T} \cdot \frac{\mathbf{e}_{T,j}^\perp}{2A_T} A_T = \sum_{T \ni \mathbf{v}_i, \mathbf{v}_j} \frac{\mathbf{e}_{T,i}^\perp \cdot \mathbf{e}_{T,j}^\perp}{4A_T}$$



Laplacian Matrix Cotangent Weights



$$L_{ij} = \begin{cases} \sum_{T \ni v_i} \frac{\mathbf{e}_{T,i}^\perp \cdot (-\mathbf{e}_{T,j}^\perp - \mathbf{e}_{T,k}^\perp)}{4A_T} = -\frac{1}{2} \sum_{v_k \in N(v_i)} (\cot(\alpha_{ik}) + \cot(\beta_{ik})) & \text{if } i = j \\ \frac{1}{2} (\cot(\alpha_{ij}) + \cot(\beta_{ij})) & \text{if } i \neq j \end{cases}$$

FEM vs “Graphics Approach”

- Get exactly the same cotan weights (up to division by region area)
- FEM doesn't require defining an averaging region
- FEM computes Laplacian **integrated against test function** instead of an **averaged** (point) quantity:

$$L_{\text{FEM}} = M_{\text{diag}} L_{\text{Avg}} \quad M_{\text{diag}} = \text{diag}([R(\mathbf{v}_0), R(\mathbf{v}_1), \dots])$$

- FEM uses **exact mass matrix** to integrate RHS against test functions; graphics approach uses **diagonal “Lumped” approximation** M_{diag} based on region areas (e.g. Voronoi)—still converges, but higher error.
(Lumping effectively treats the RHS function “ b ” as **piecewise constant** over the averaging region, while FEM+exact mass matrix treats it as **piecewise linear**).
- FEM generalizes to higher degree; just use more basis functions!