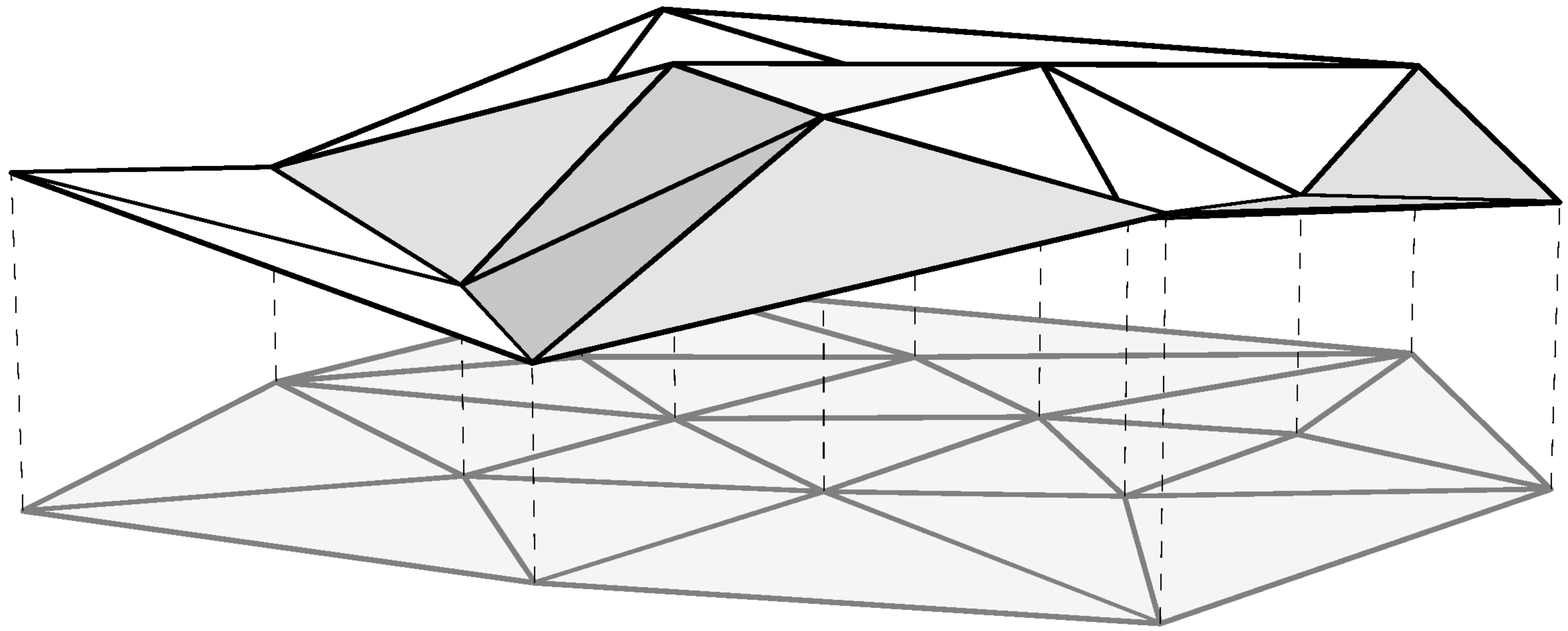
Perfect Laplacians for Polygon Meshes

Phillip Herholz, Jan Eric Kyprianidis, Marc Alexa
TU Berlin

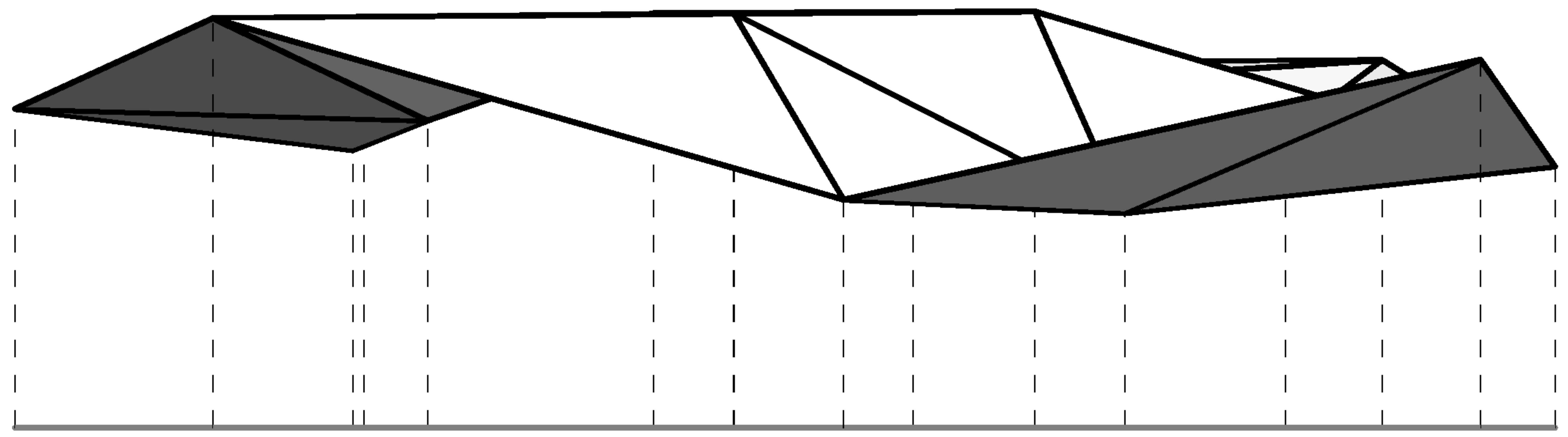
What is a Mesh Laplacian?

- Second order difference operator
- Defined over mesh



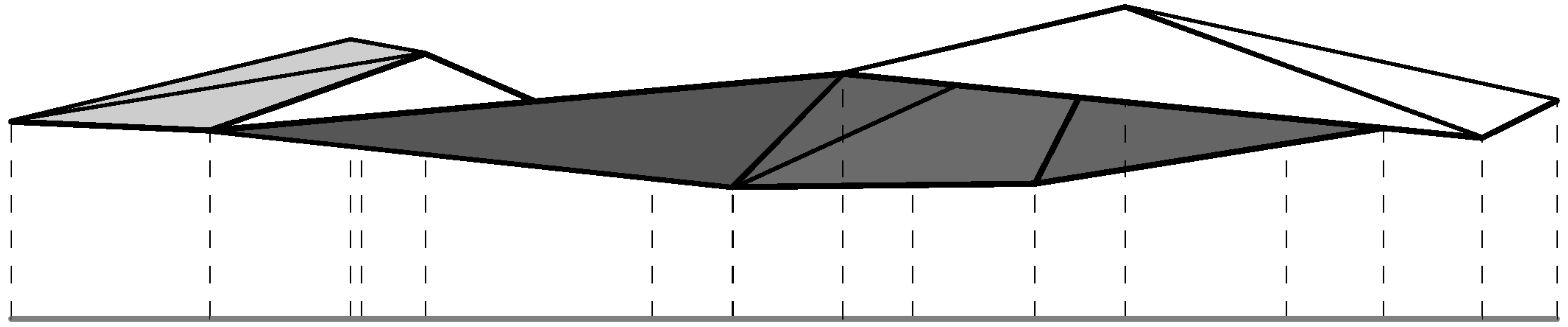
What is a Mesh Laplacian?

- Second order difference operator
- Defined over mesh
- Mapping
    - from values at vertices



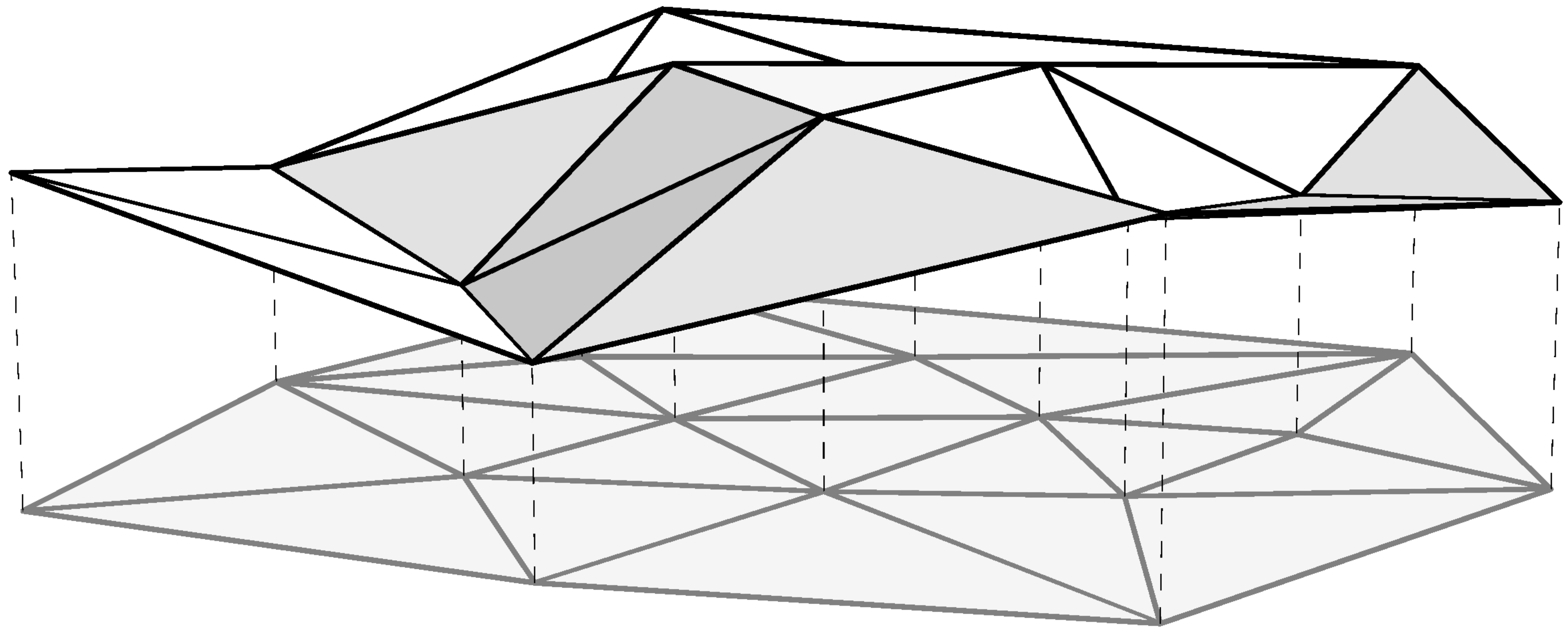
What is a Mesh Laplacian?

- Second order difference operator
- Defined over mesh
- Mapping
    - from values at vertices
    - to values at vertices



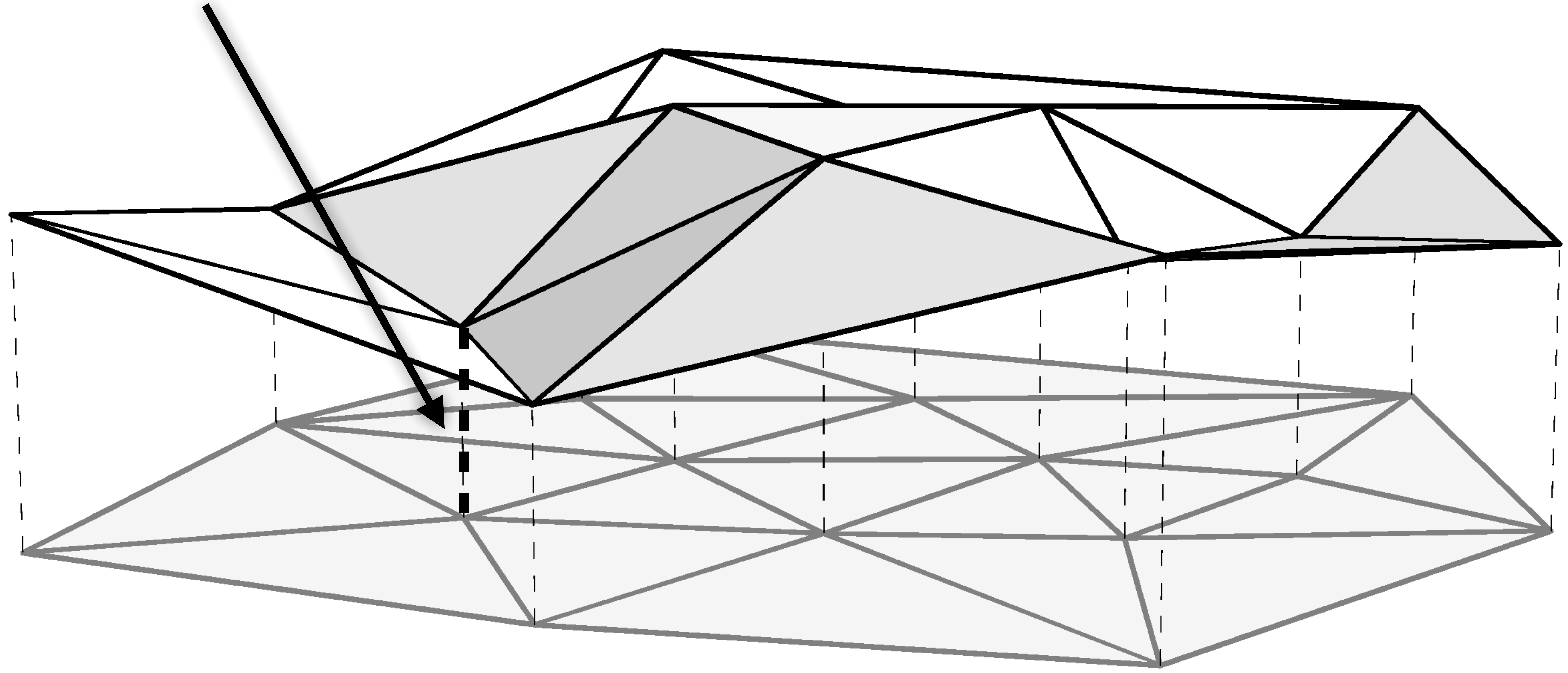
Mesh Laplacian - A common choice

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$



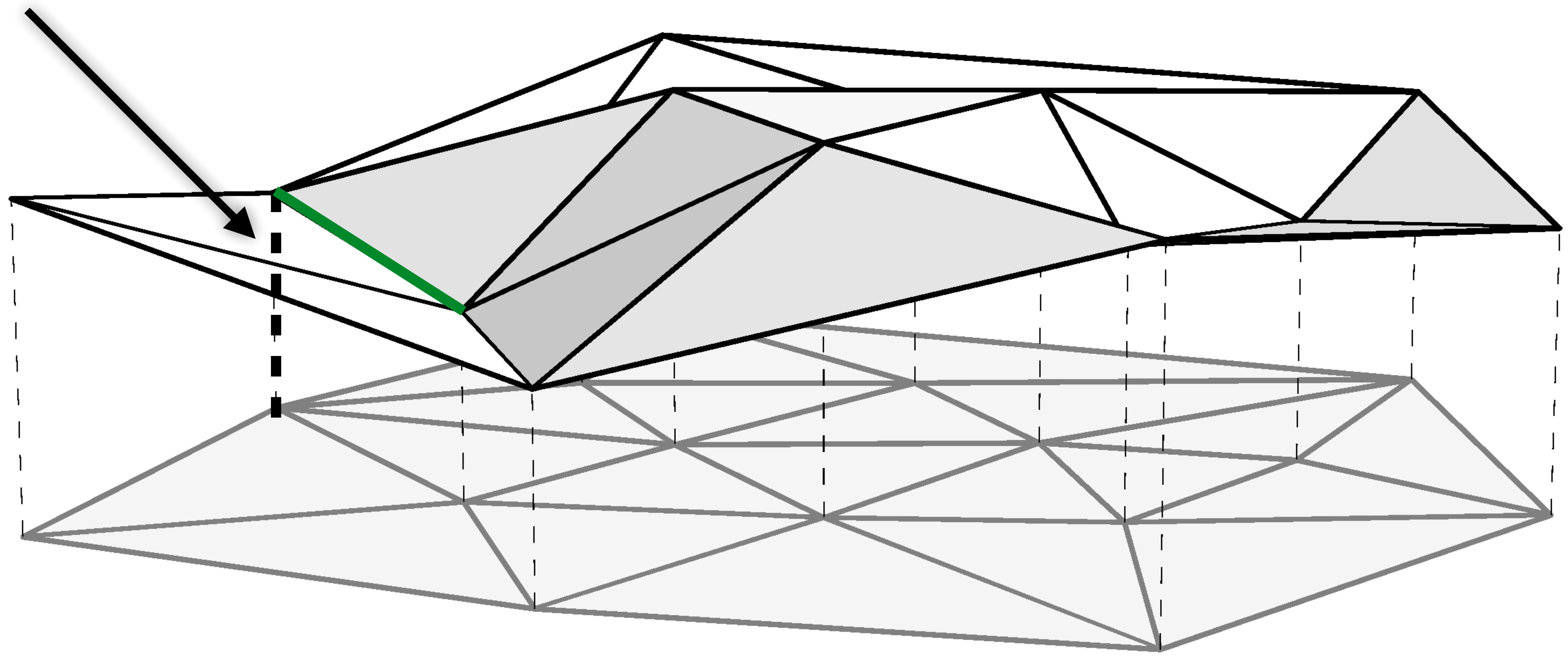
Mesh Laplacian - A common choice

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$



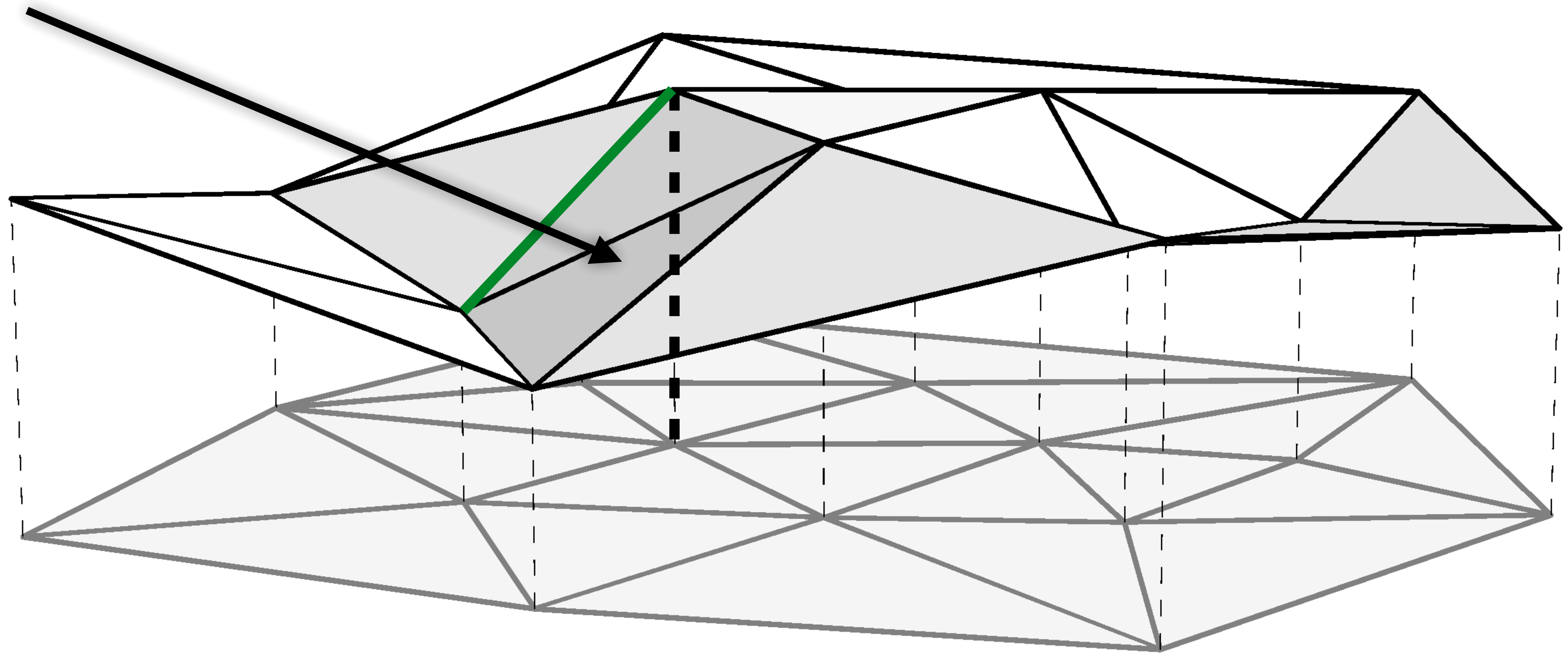
Mesh Laplacian - A common choice

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$



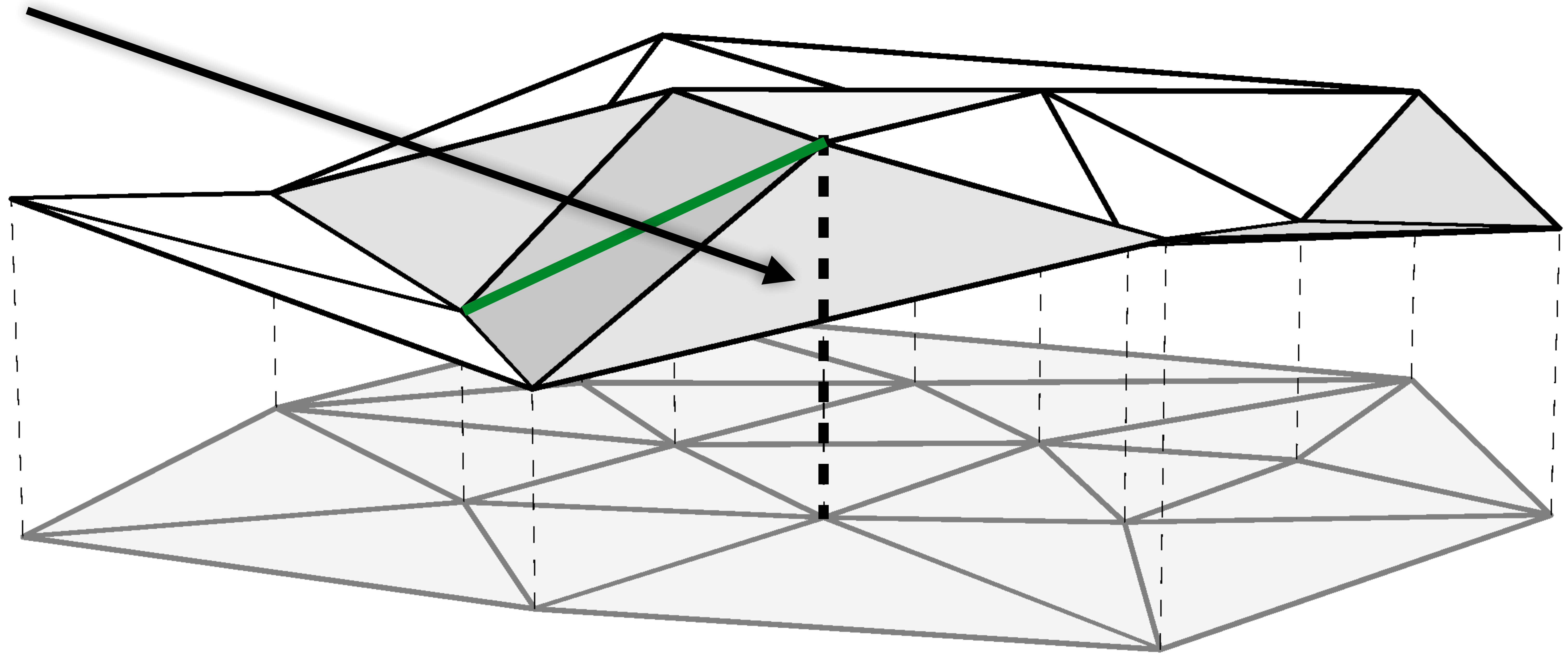
Mesh Laplacian - A common choice

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$



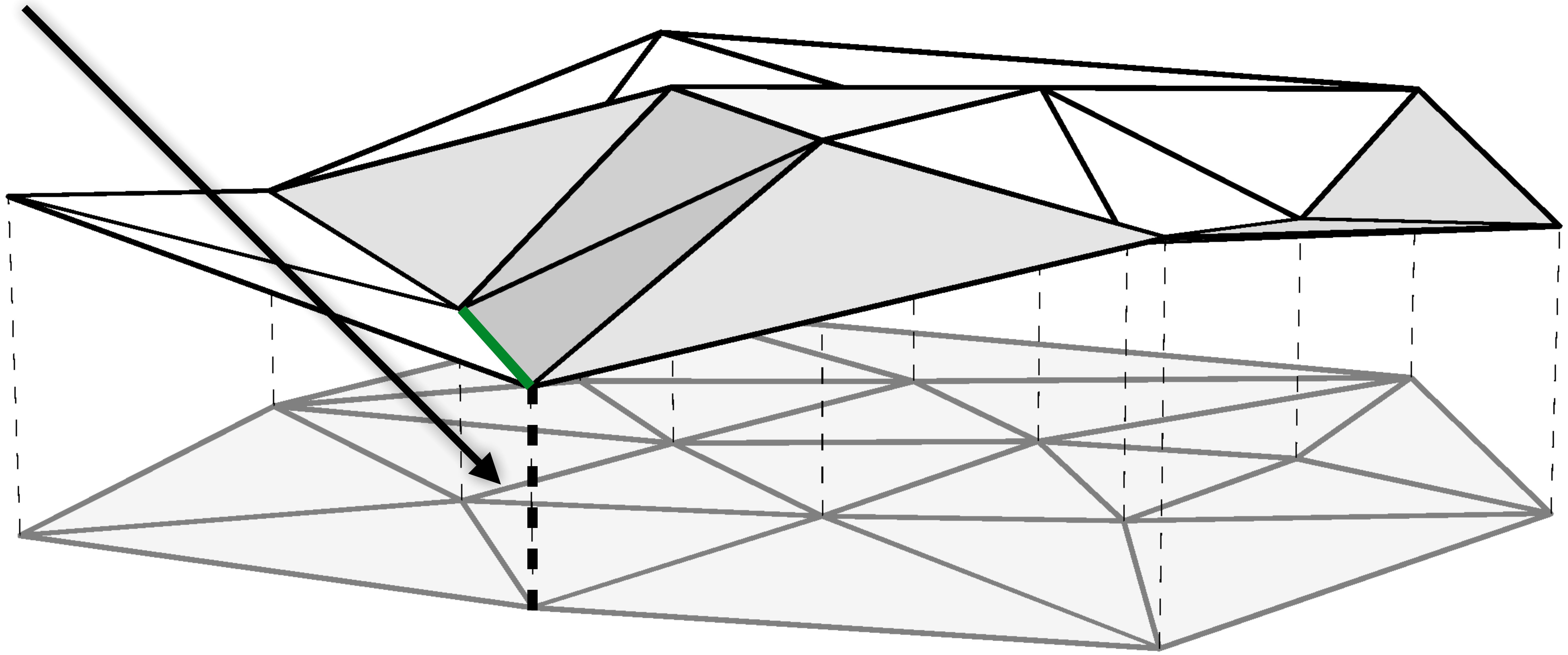
Mesh Laplacian - A common choice

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$



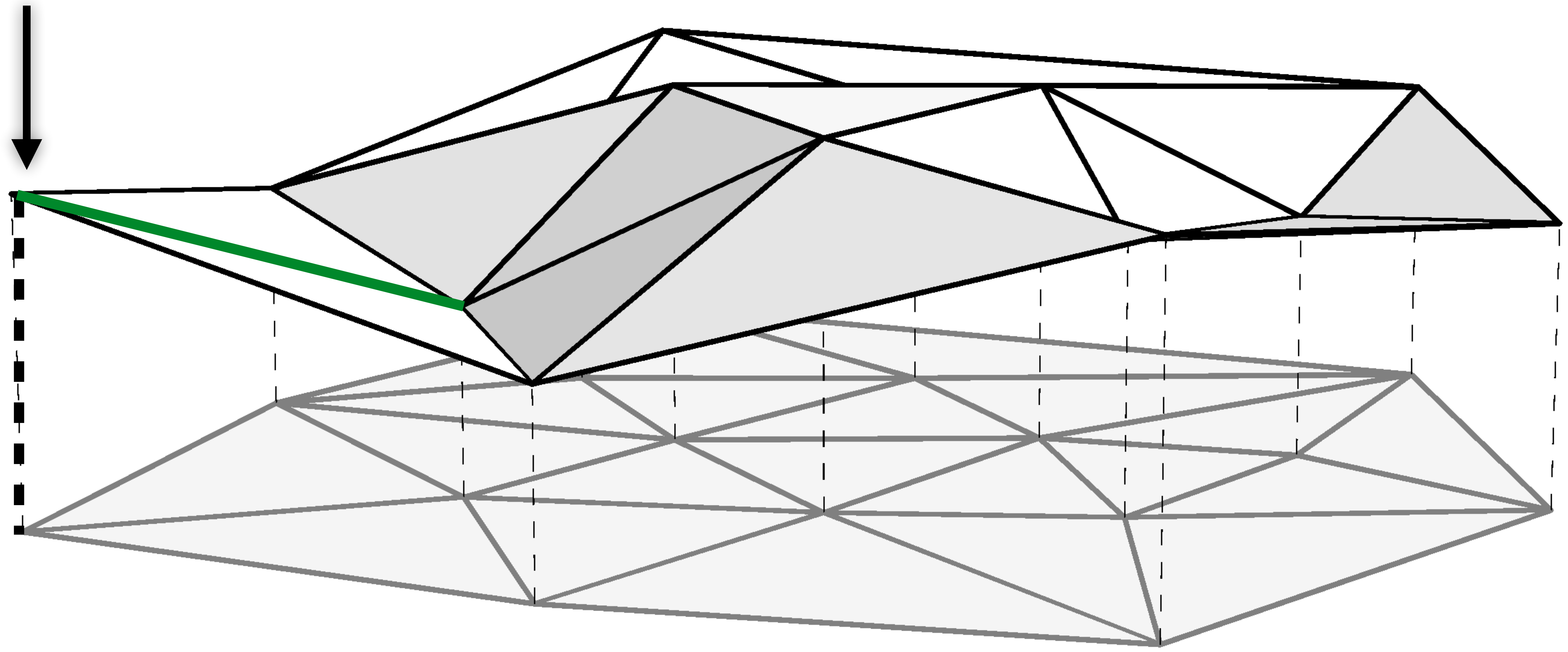
Mesh Laplacian - A common choice

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$



Mesh Laplacian - A common choice

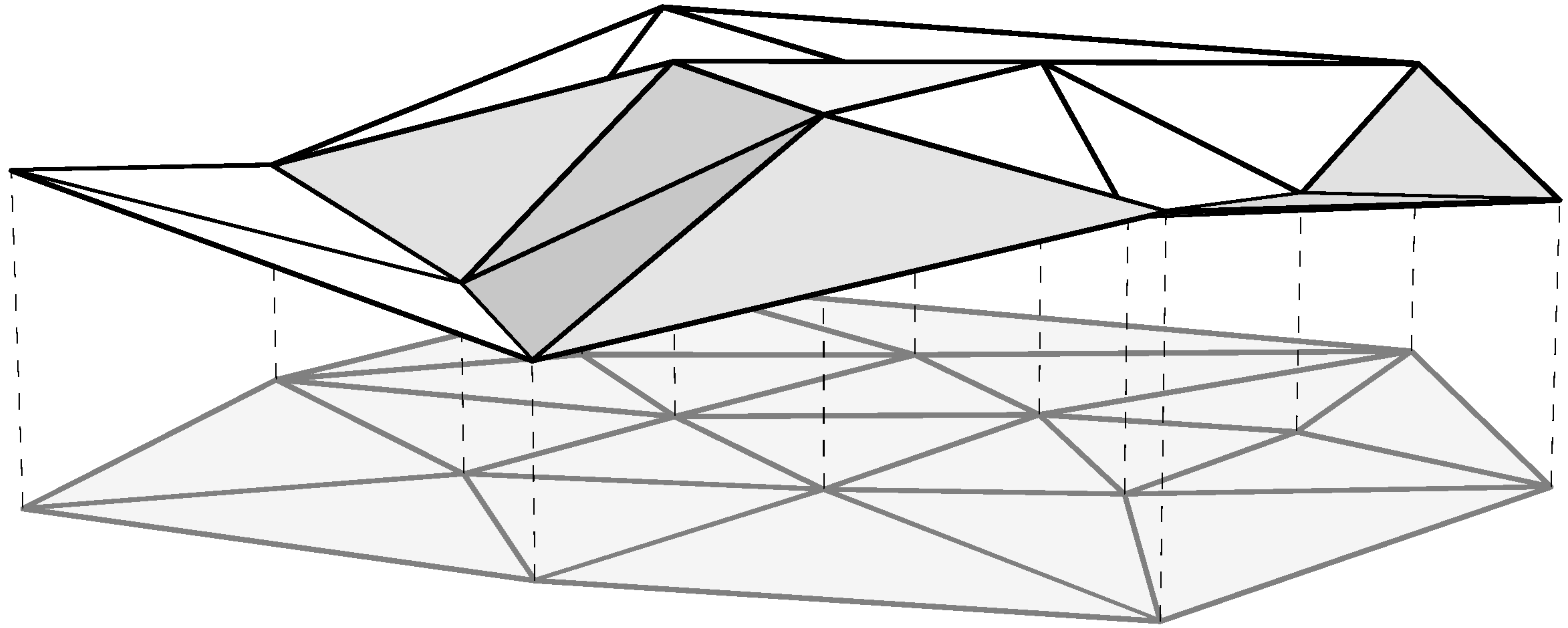
$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$



Why Mesh Laplacian?

- Geometry processing = Applying Laplacian

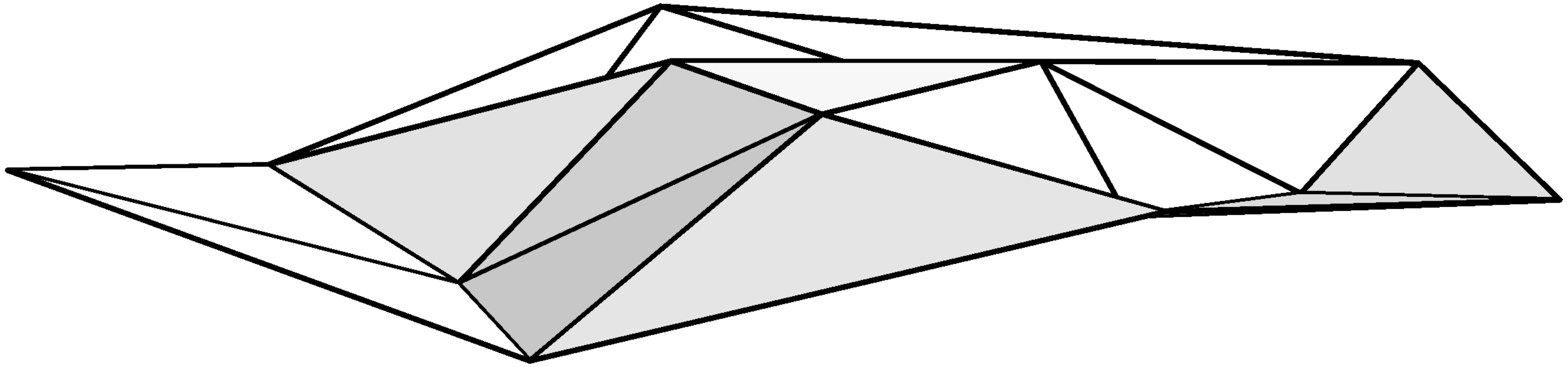
$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$



Why Mesh Laplacian?

- Geometry processing =
  Applying Laplacian to geometry

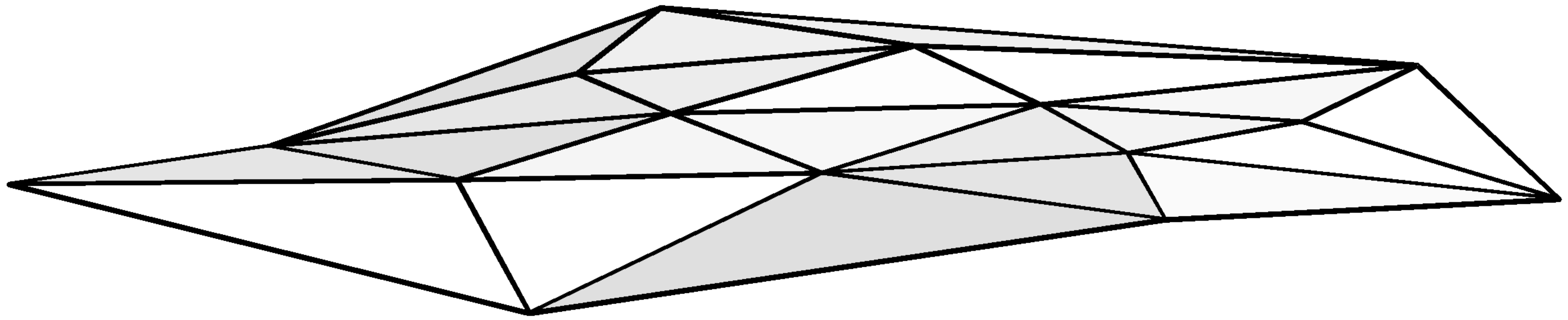
$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$



Why Mesh Laplacian?

- Geometry processing = Applying Laplacian to geometry
  - Smoothing / fairing $\mathbf{V}' = \mathbf{V} + \lambda \mathbf{L}\mathbf{V}$

$$(\mathbf{L}\mathbf{V})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

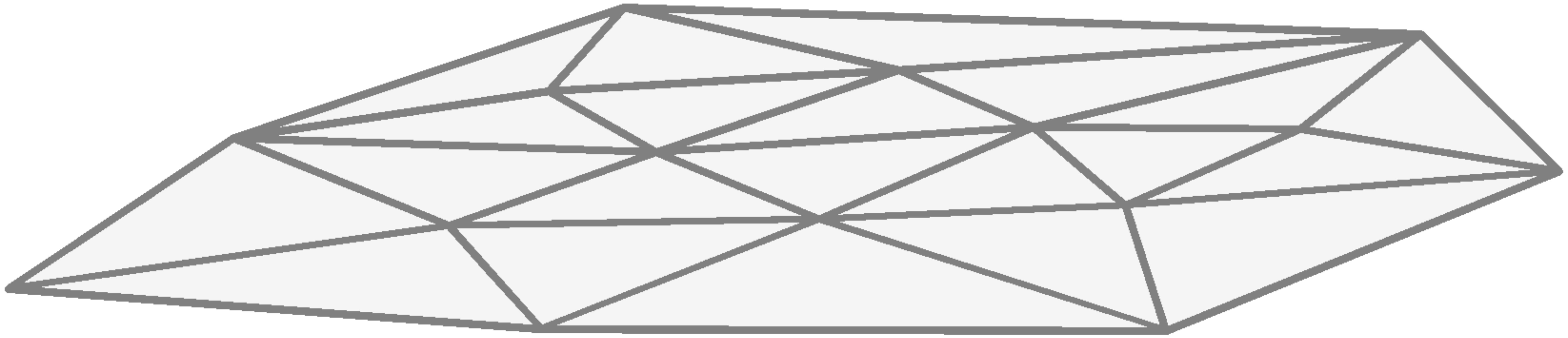


Why Mesh Laplacian?

- Geometry processing =
  Applying Laplacian to geometry

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

  - Smoothing / fairing $\mathbf{V}' = \mathbf{V} + \lambda \mathbf{LV}$
  - Parameterization $\mathbf{LV}' = \mathbf{0}$



Why Mesh Laplacian?

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

- Geometry processing =
  Applying Laplacian to geometry
  - Smoothing / fairing $\mathbf{V}' = \mathbf{V} + \lambda\mathbf{LV}$
  - Parameterization $\mathbf{LV}' = \mathbf{0}$
  - Deformation $\mathbf{LV}' \approx \mathbf{LV}$

Why Mesh Laplacian?

- Geometry processing = Applying Laplacian to geometry

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

  - Smoothing / fairing $\mathbf{V}' = \mathbf{V} + \lambda \mathbf{LV}$
  - Parameterization $\mathbf{LV}' = \mathbf{0}$
  - Deformation $\mathbf{LV}' \approx \mathbf{LV}$
  - Simulation / Animation

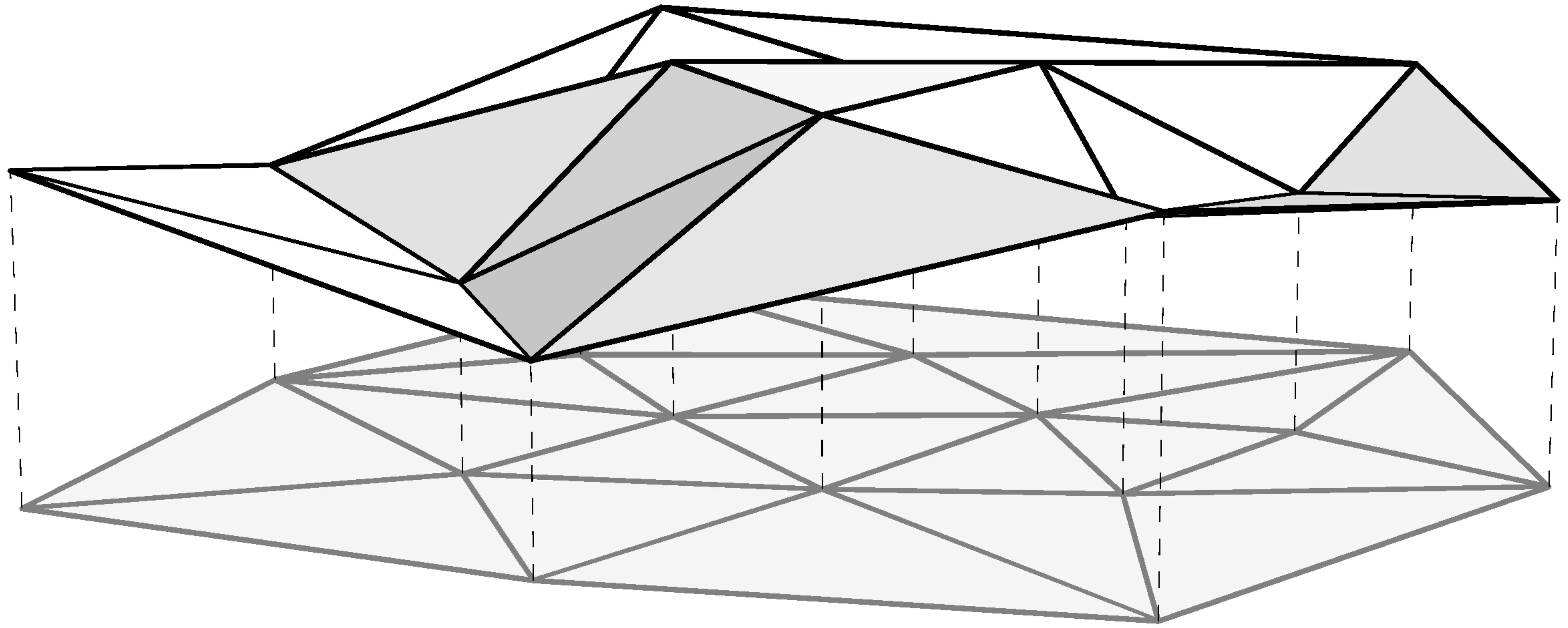
Why Mesh Laplacian?

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

- Geometry processing =
  Applying Laplacian to geometry
  - Smoothing / fairing $\mathbf{V}' = \mathbf{V} + \lambda\mathbf{LV}$
  - Parameterization $\mathbf{LV}' = \mathbf{0}$
  - Deformation $\mathbf{LV}' \approx \mathbf{LV}$
  - Simulation / Animation
- Much more

Mesh Laplacian - Properties

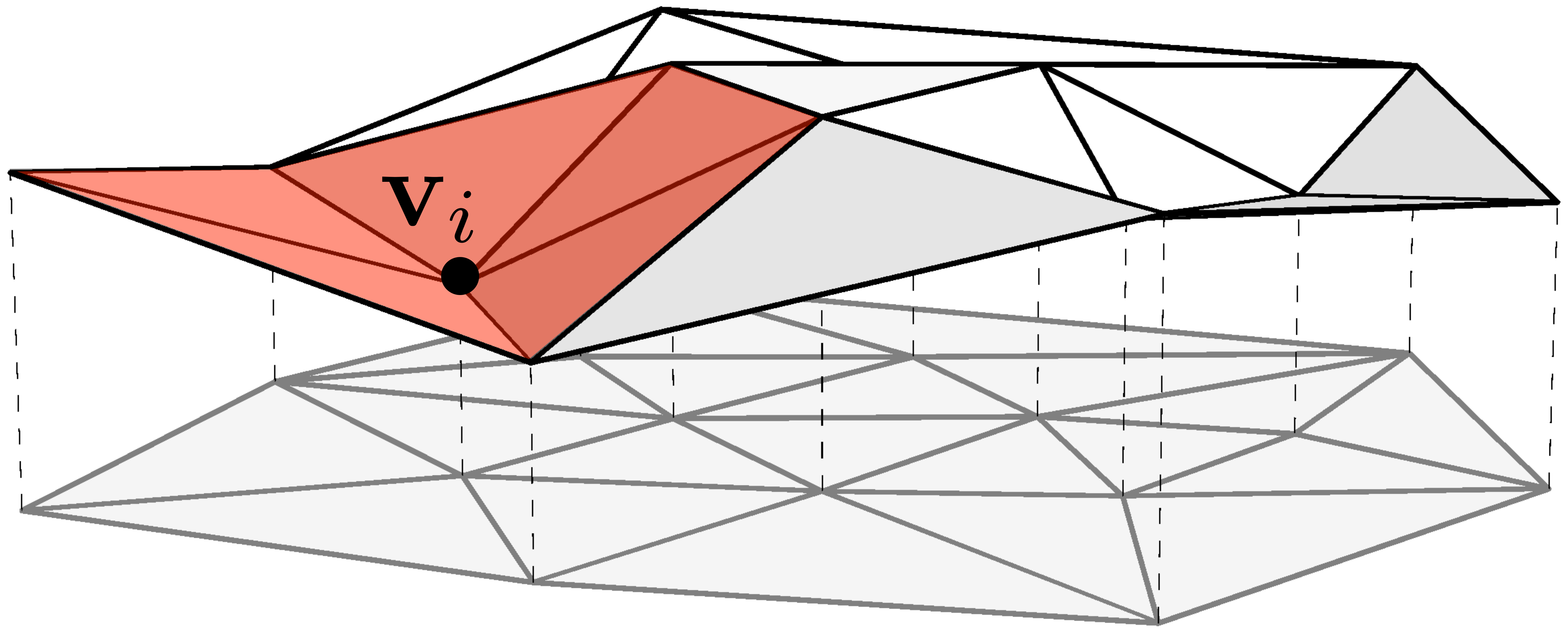
- Locality
  - Smooth Laplacian is local
  - Efficiency



Mesh Laplacian - Properties

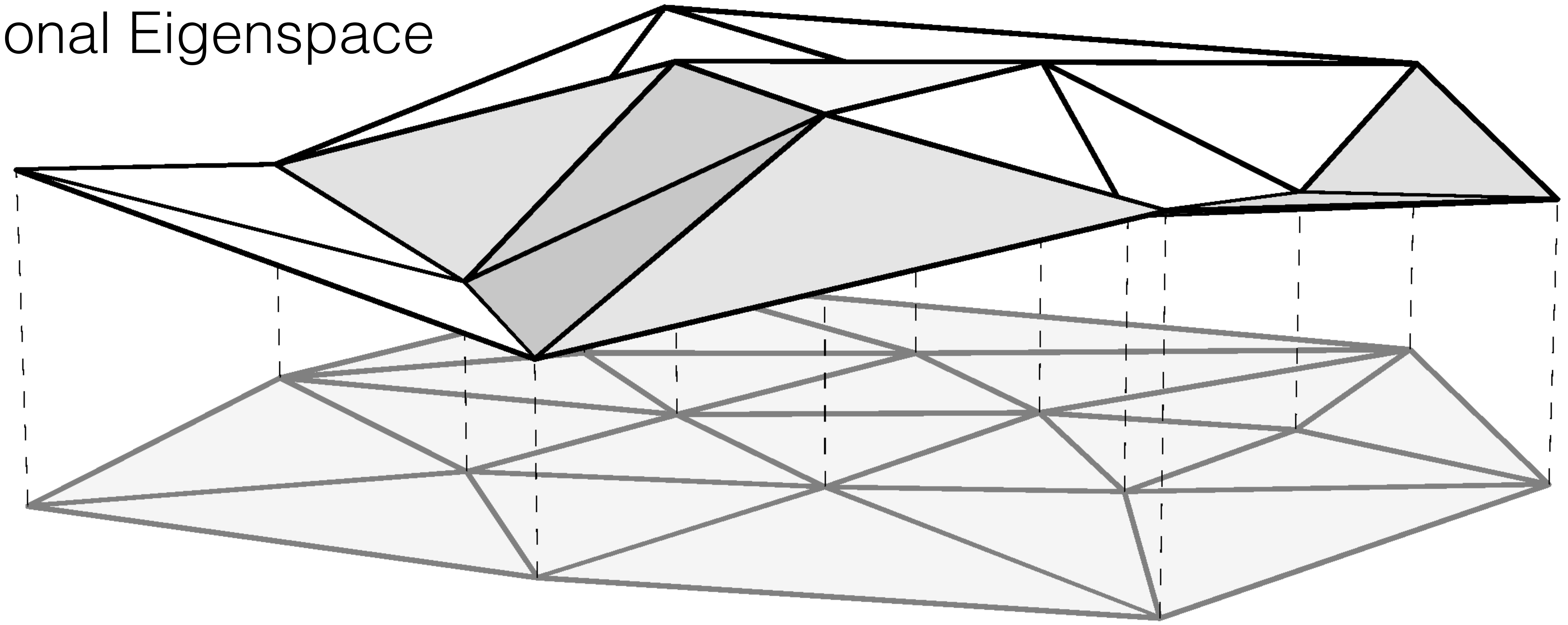
$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$

- Locality
    - Smooth Laplacian is local
    - Efficiency



Mesh Laplacian - Properties

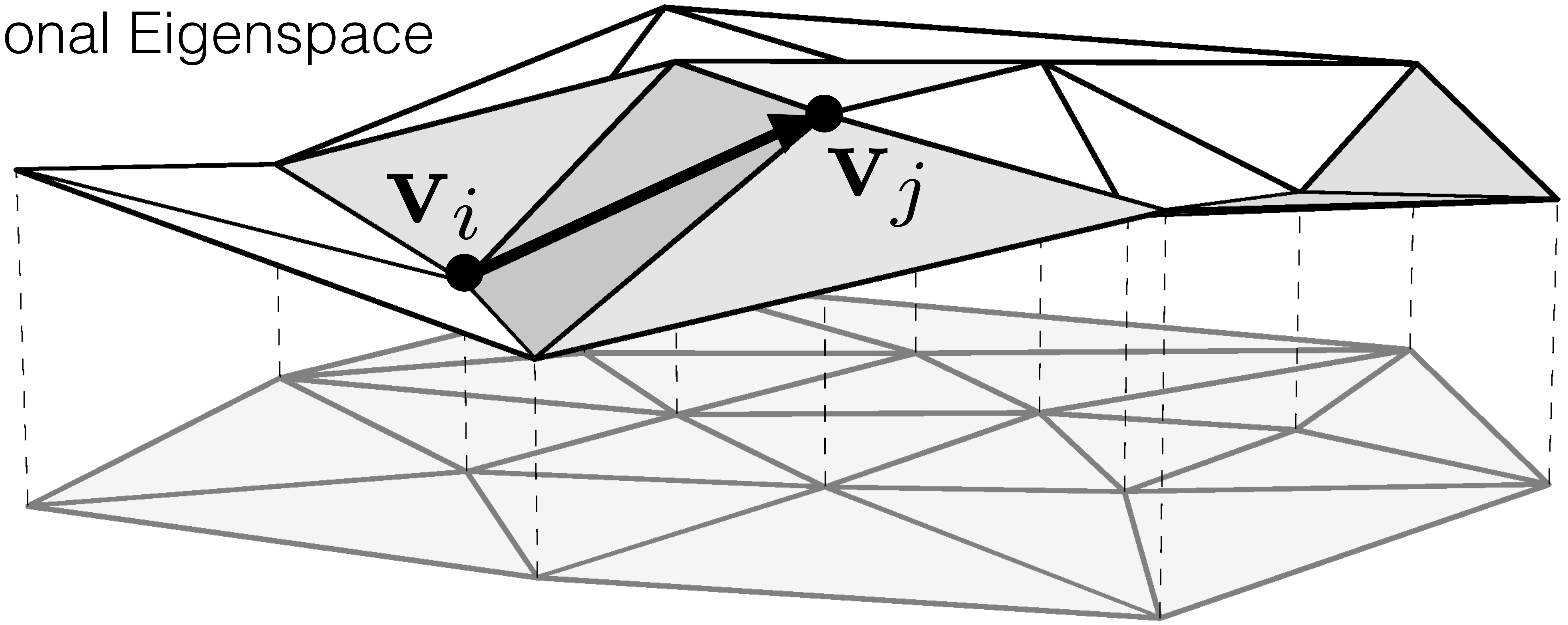
- Symmetry
  - Smooth Laplacian is symmetric
  - Real spectrum, orthogonal Eigenspace



Mesh Laplacian - Properties

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$

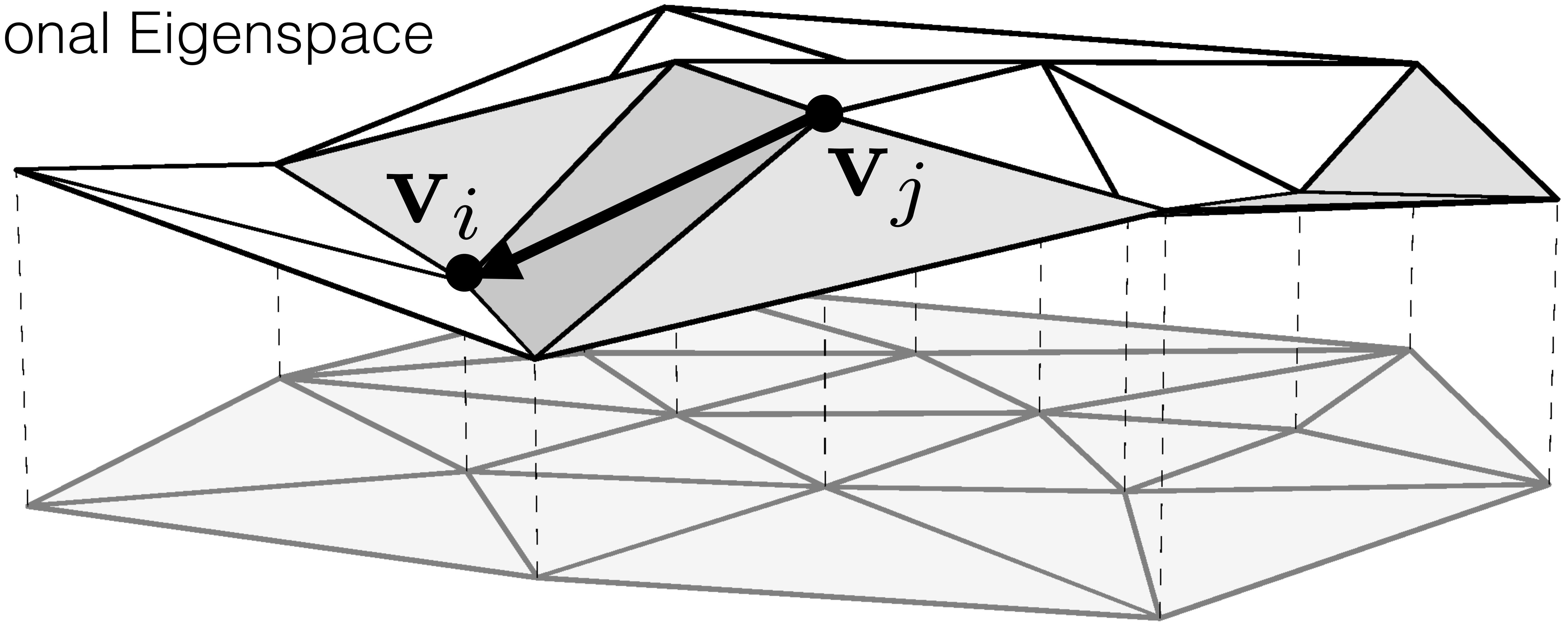
- Symmetry
  - Smooth Laplacian is symmetric
  - Real spectrum, orthogonal Eigenspace



Mesh Laplacian - Properties

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$

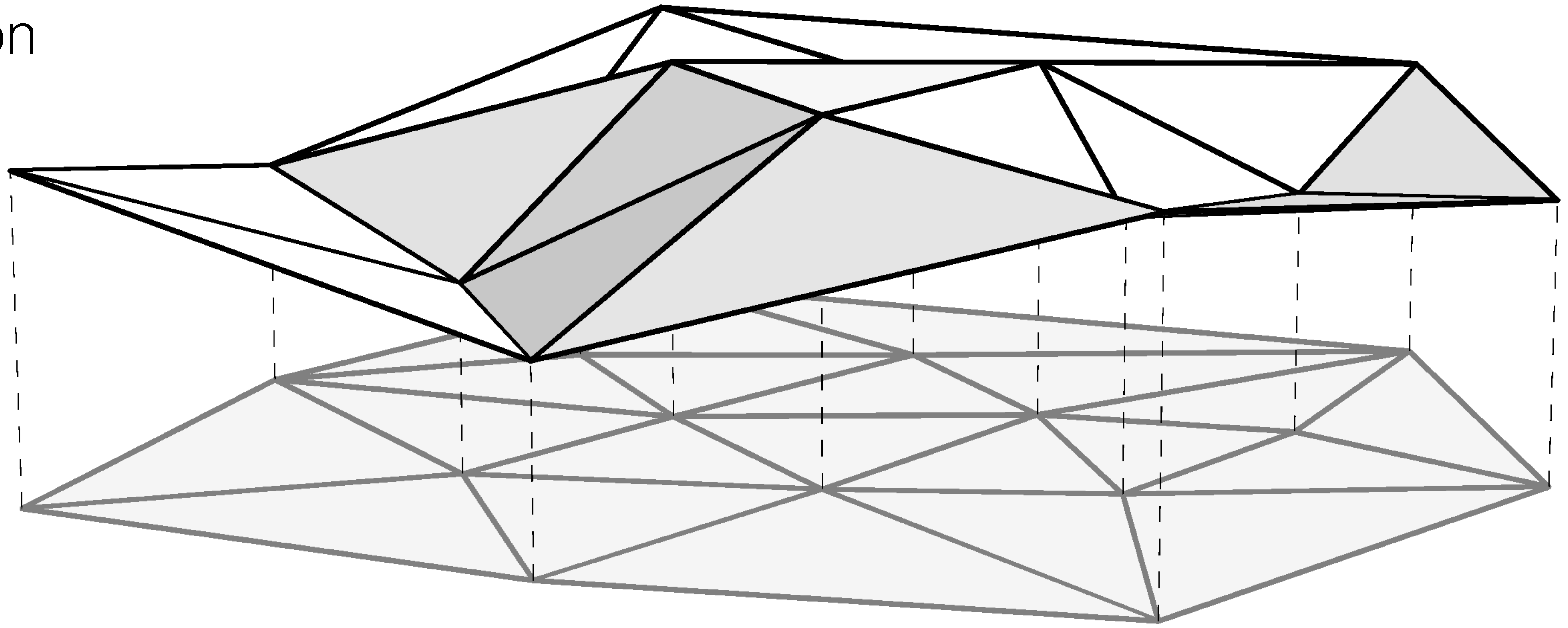
- Symmetry $\omega_{ij} = \omega_{ji}$
    - Smooth Laplacian is symmetric
    - Real spectrum, orthogonal Eigenspace



Mesh Laplacian - Properties

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$

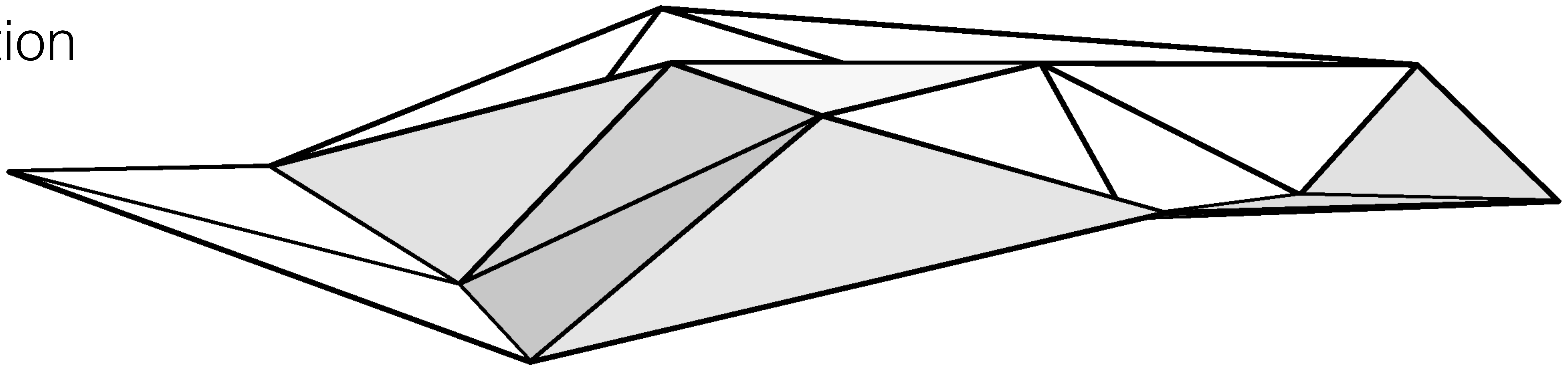
- Constants in kernel
  - Laplacian is differential operator
  - Invariance to translation



Mesh Laplacian - Properties

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

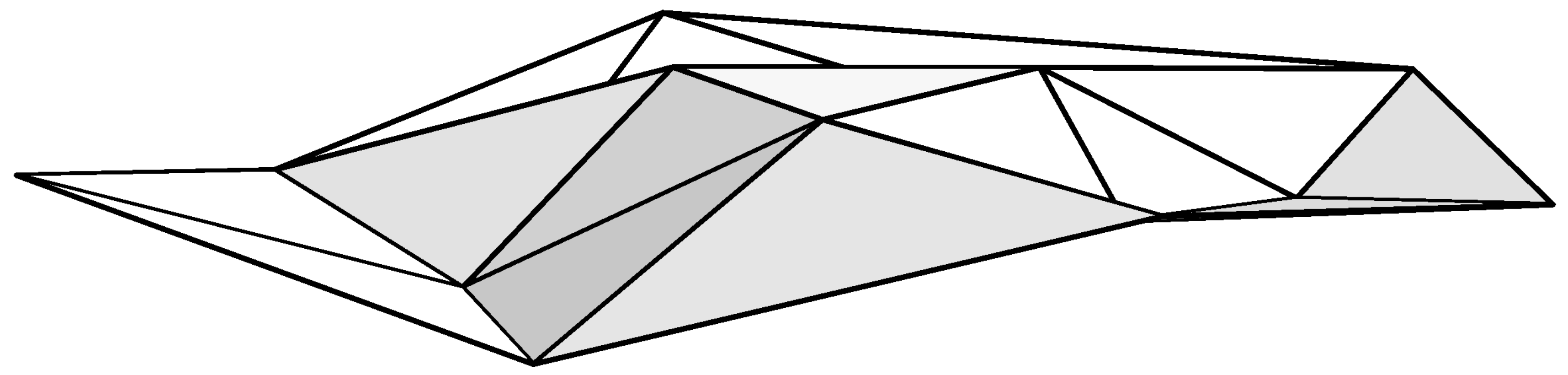
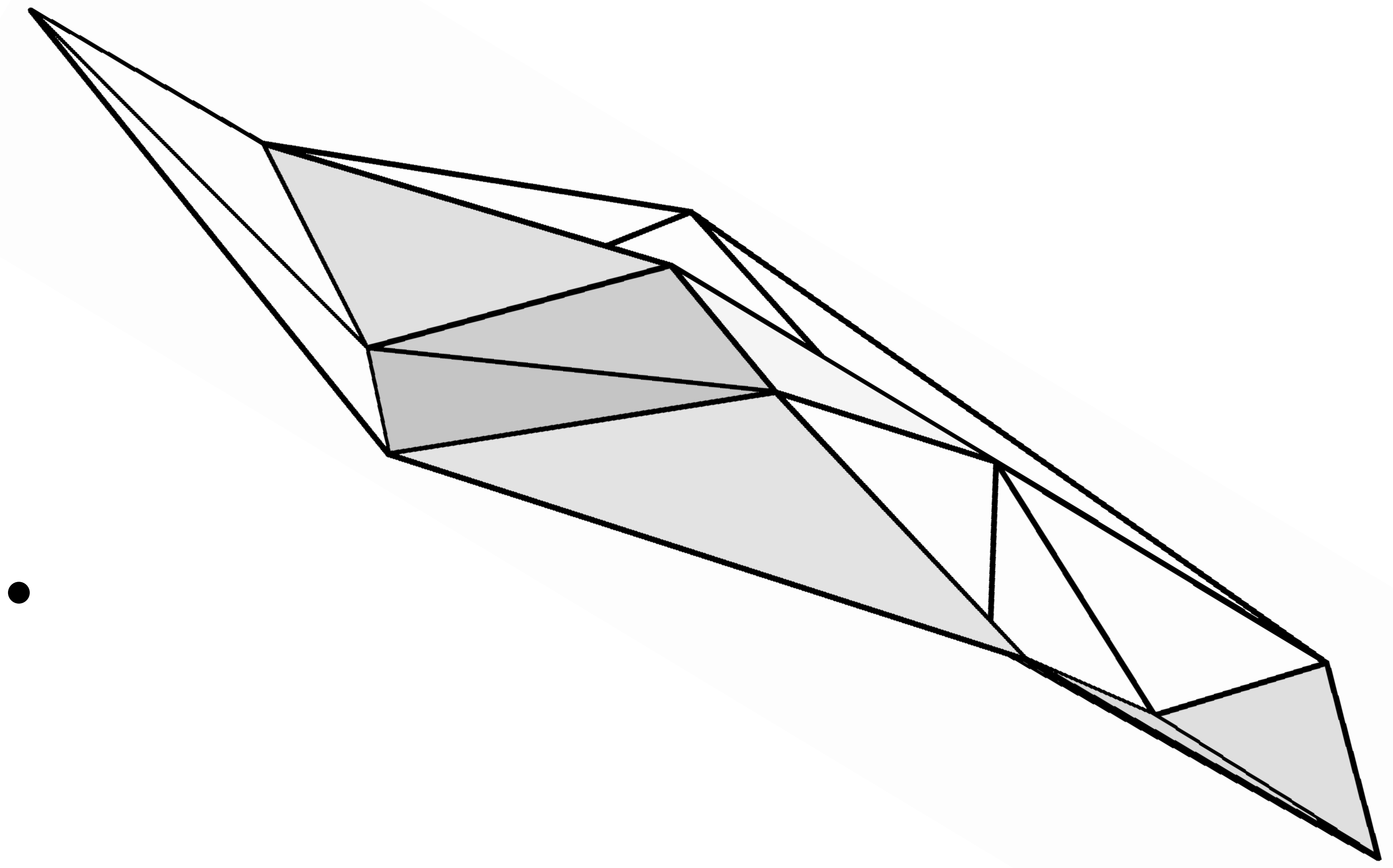
- Constants in kernel
  - Laplacian is differential operator
  - Invariance to translation



Mesh Laplacian - Properties

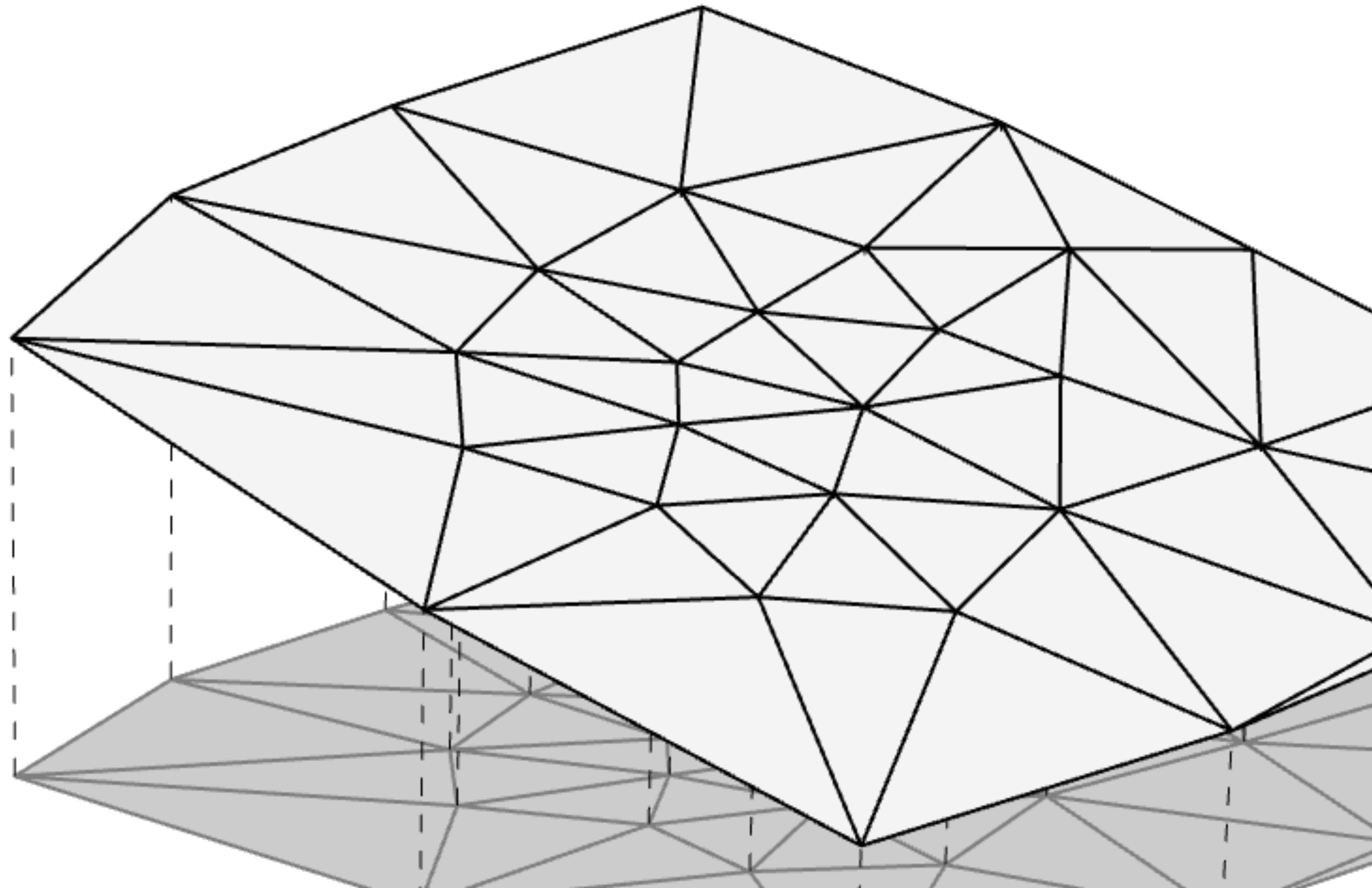
$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

- Constants in kernel
  - Laplacian is differential operator
  - Invariance to translation
  - Affinely independent

$\mathbf{A} \cdot \mathbf{L} \cdot$  $= \mathbf{L} \cdot$ 

Mesh Laplacian - Properties

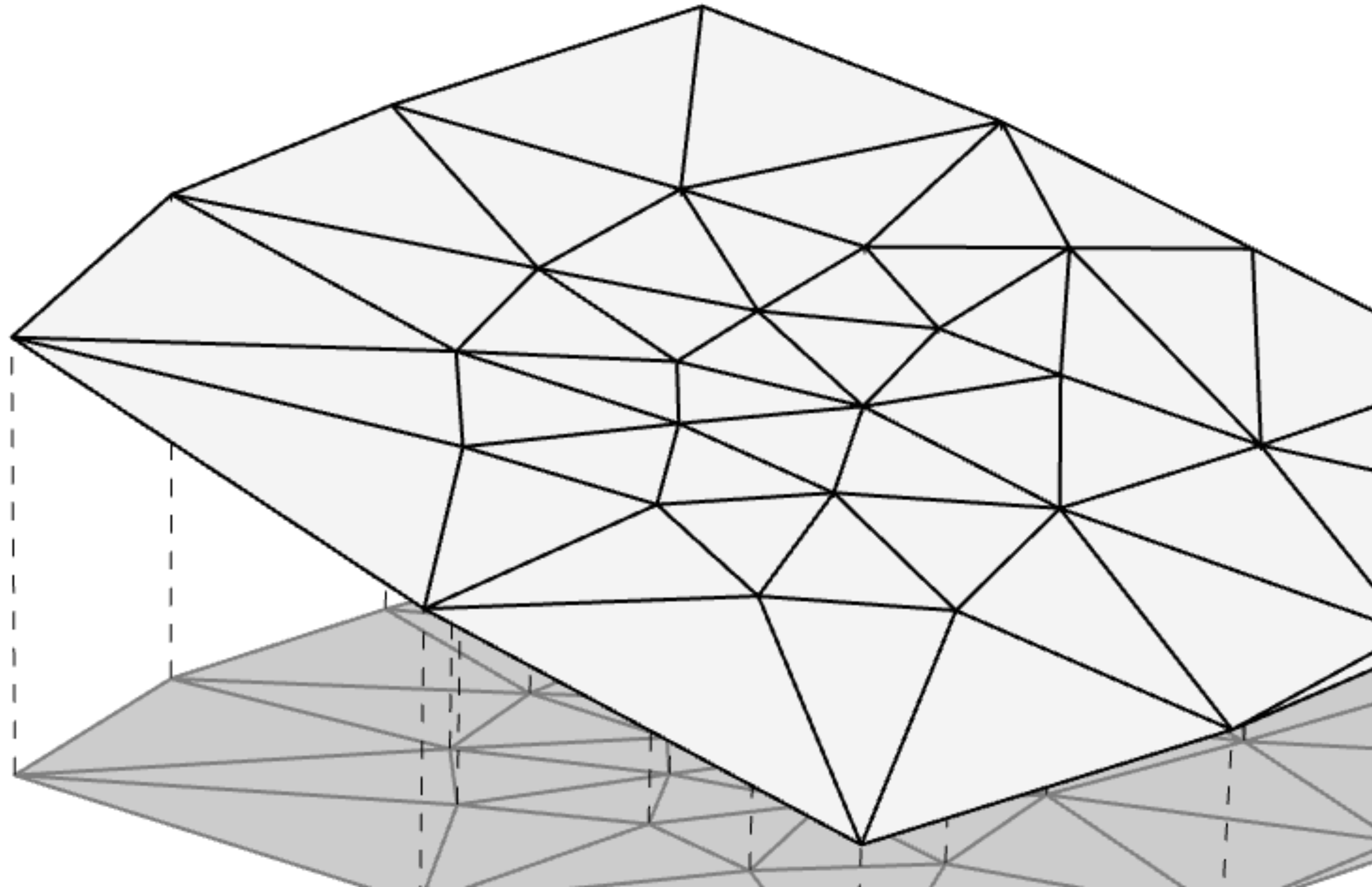
- Linear precision
  - Second order differences vanish on linear functions



Mesh Laplacian - Linear precision

- $\mathbf{L}(c_0\mathbf{1} + c_1\mathbf{x} + c_2\mathbf{y}) = \mathbf{0}$
- $(\mathbf{1}, \mathbf{x}, \mathbf{y})$ span linear functions

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$

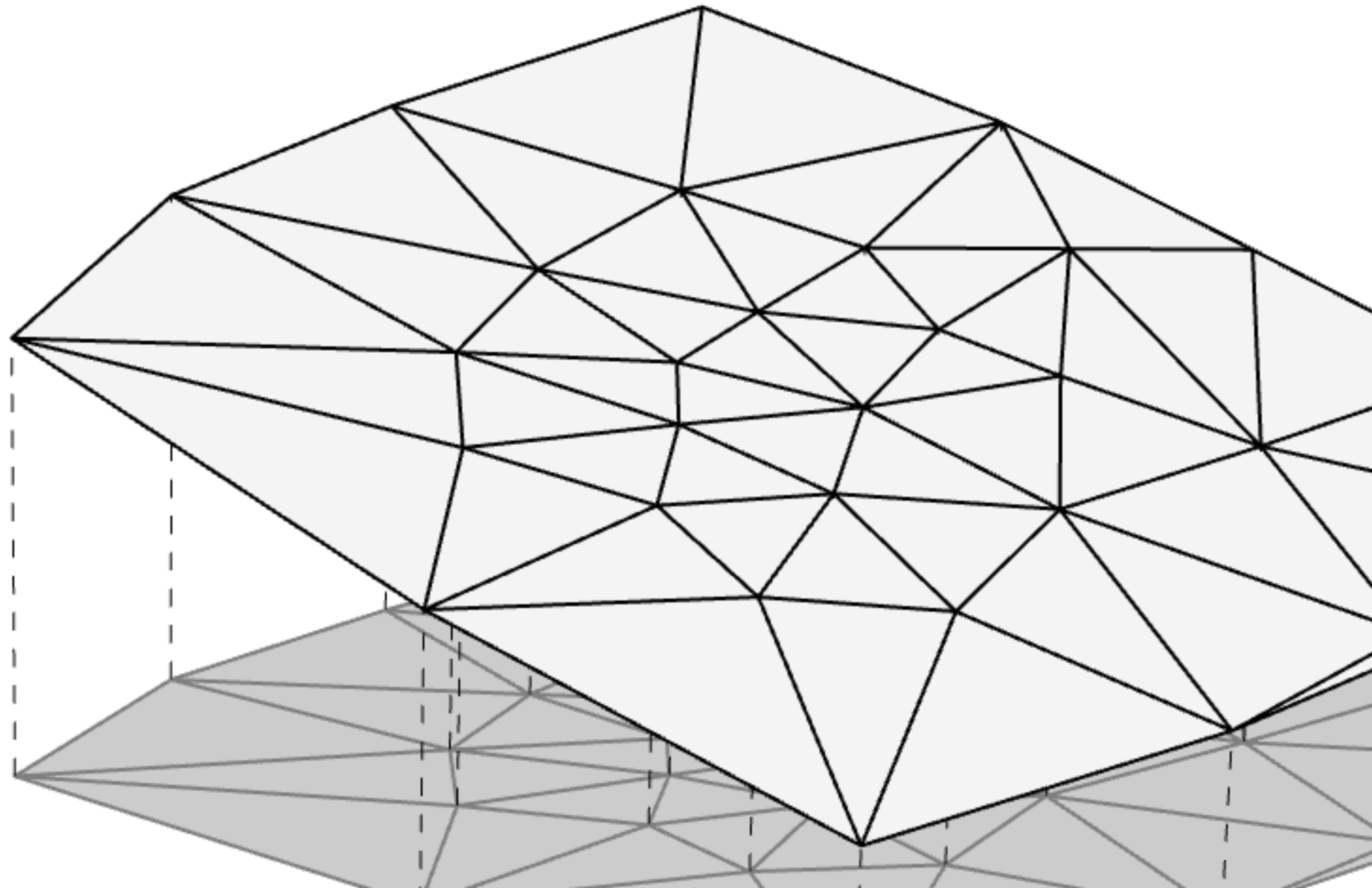


Mesh Laplacian - Linear precision

- $\mathbf{L}(c_0\mathbf{1} + c_1\mathbf{x} + c_2\mathbf{y}) = \mathbf{0}$
- $(\mathbf{1}, \mathbf{x}, \mathbf{y})$ span linear functions
- Take vertex coordinates

$$(\mathbf{Lu})_i = \sum_{(i,j)\in E} \omega_{ij}(u_j - u_i)$$

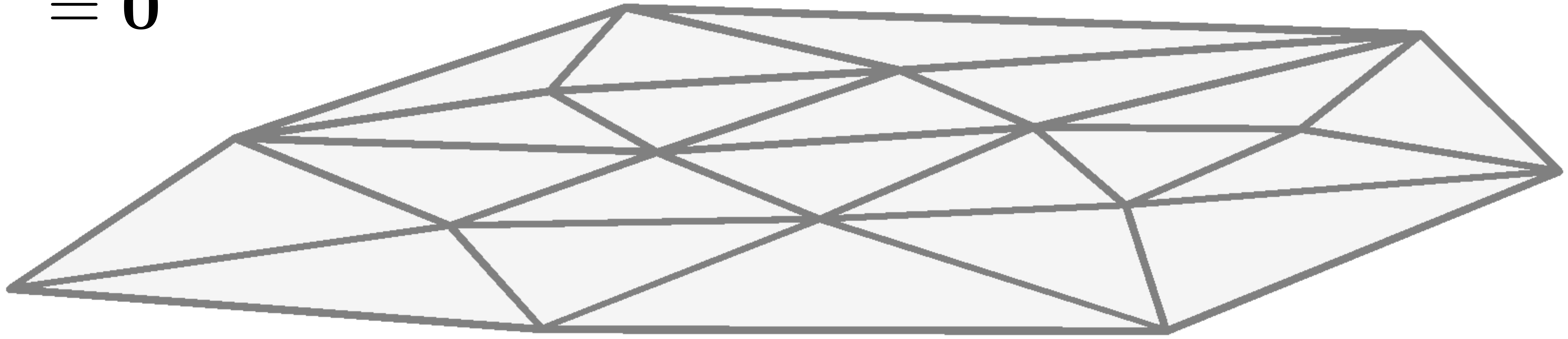
$$(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} x_0, y_0 \\ x_1, y_1 \\ \vdots \end{pmatrix} = \begin{pmatrix} \mathbf{v}_0^\mathsf{T} \\ \mathbf{v}_1^\mathsf{T} \\ \vdots \end{pmatrix} = \mathbf{V}$$



Mesh Laplacian - Properties

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

- Linear precision
  - Second order differences vanish on linear functions
  - Identity for parameterizing flat meshes with $\mathbf{LV}' = \mathbf{0}$

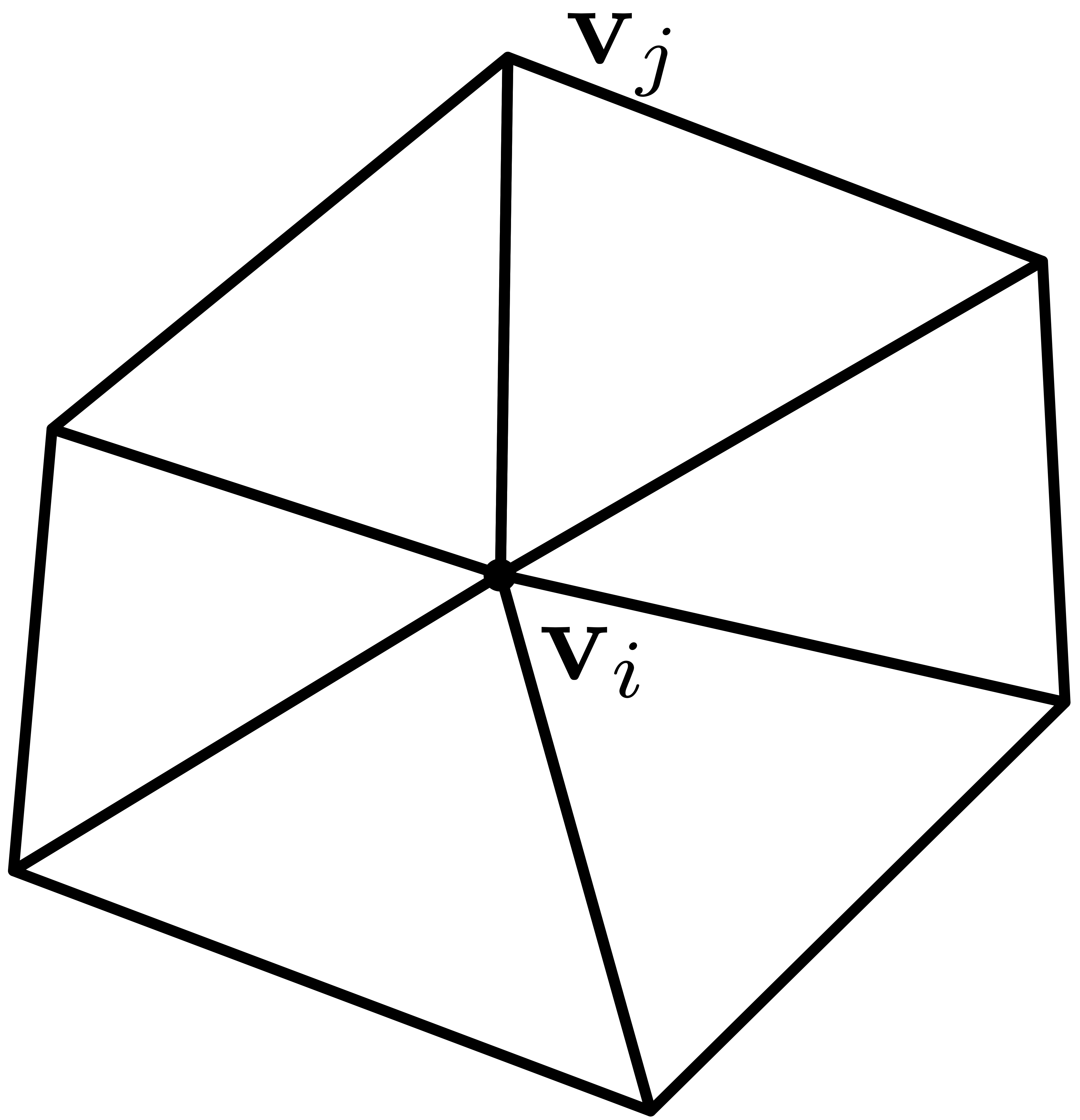


Mesh Laplacian - Linear precision

$$\mathbf{L}\mathbf{V} = \mathbf{0}$$

$$(\mathbf{L}\mathbf{V})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

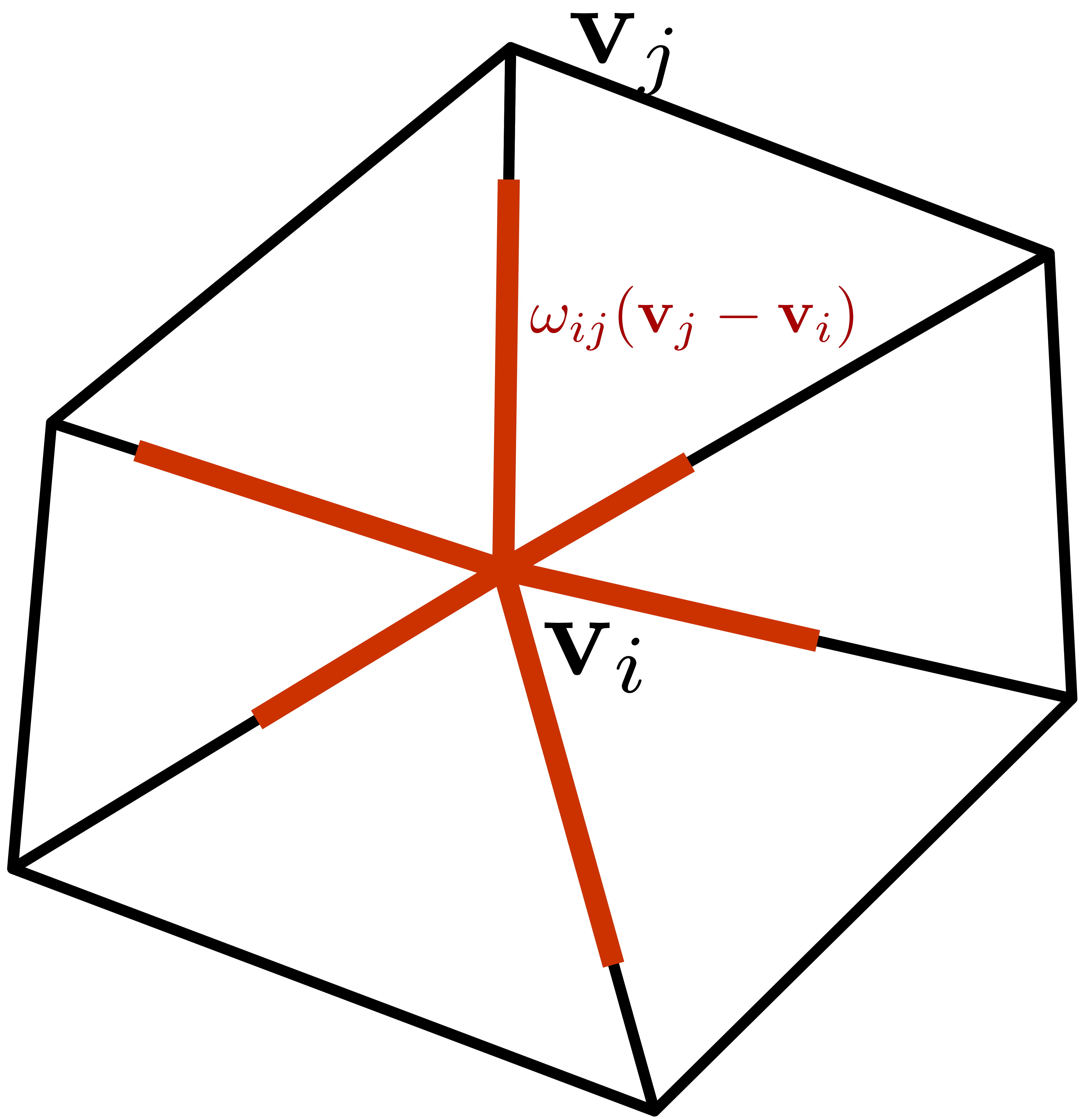
- Tangential component vanishes!



Mesh Laplacian - Linear precision

$$\mathbf{LV} = \mathbf{0}$$

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

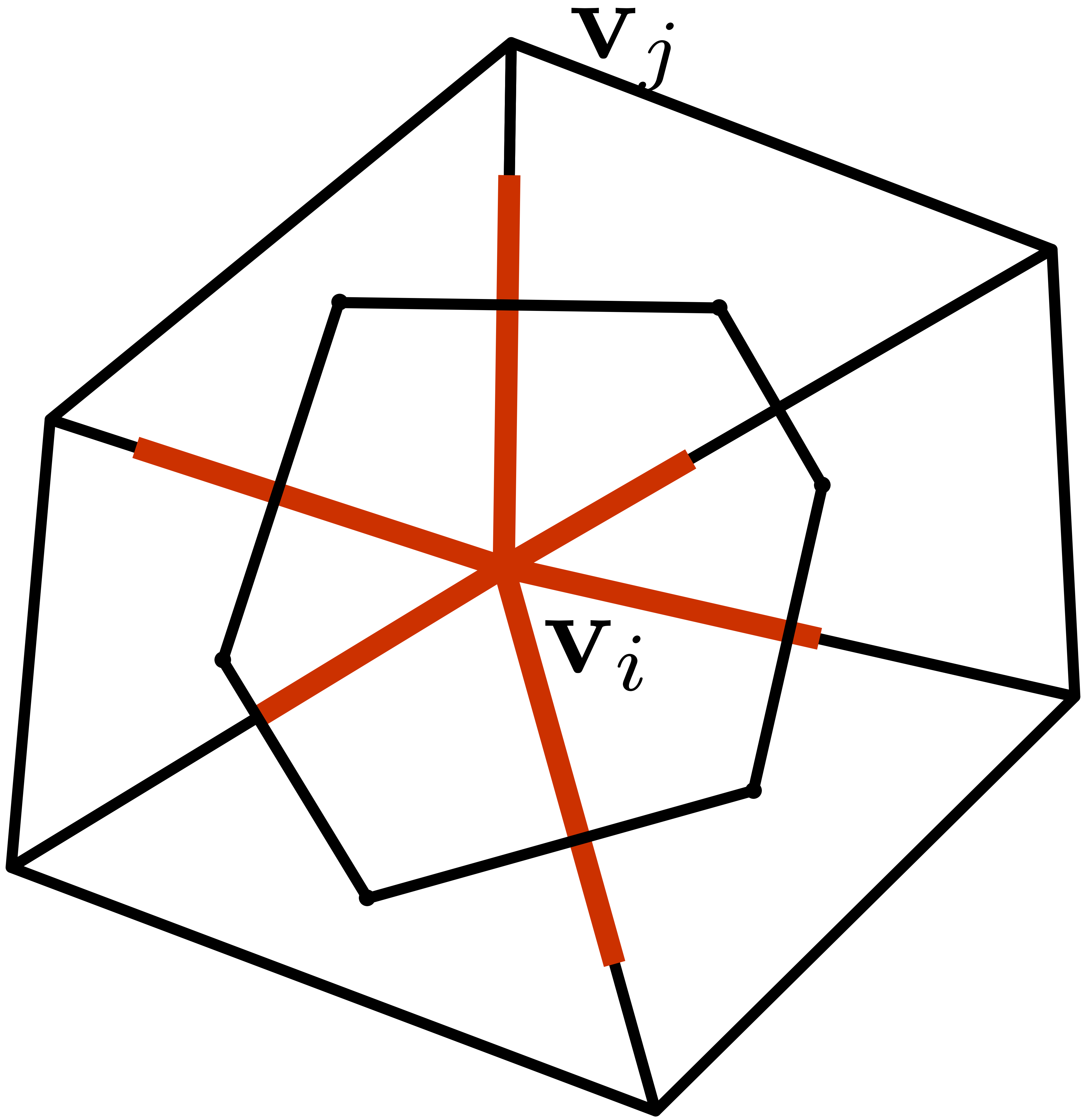


Mesh Laplacian - Linear precision

$$\mathbf{LV} = \mathbf{0}$$

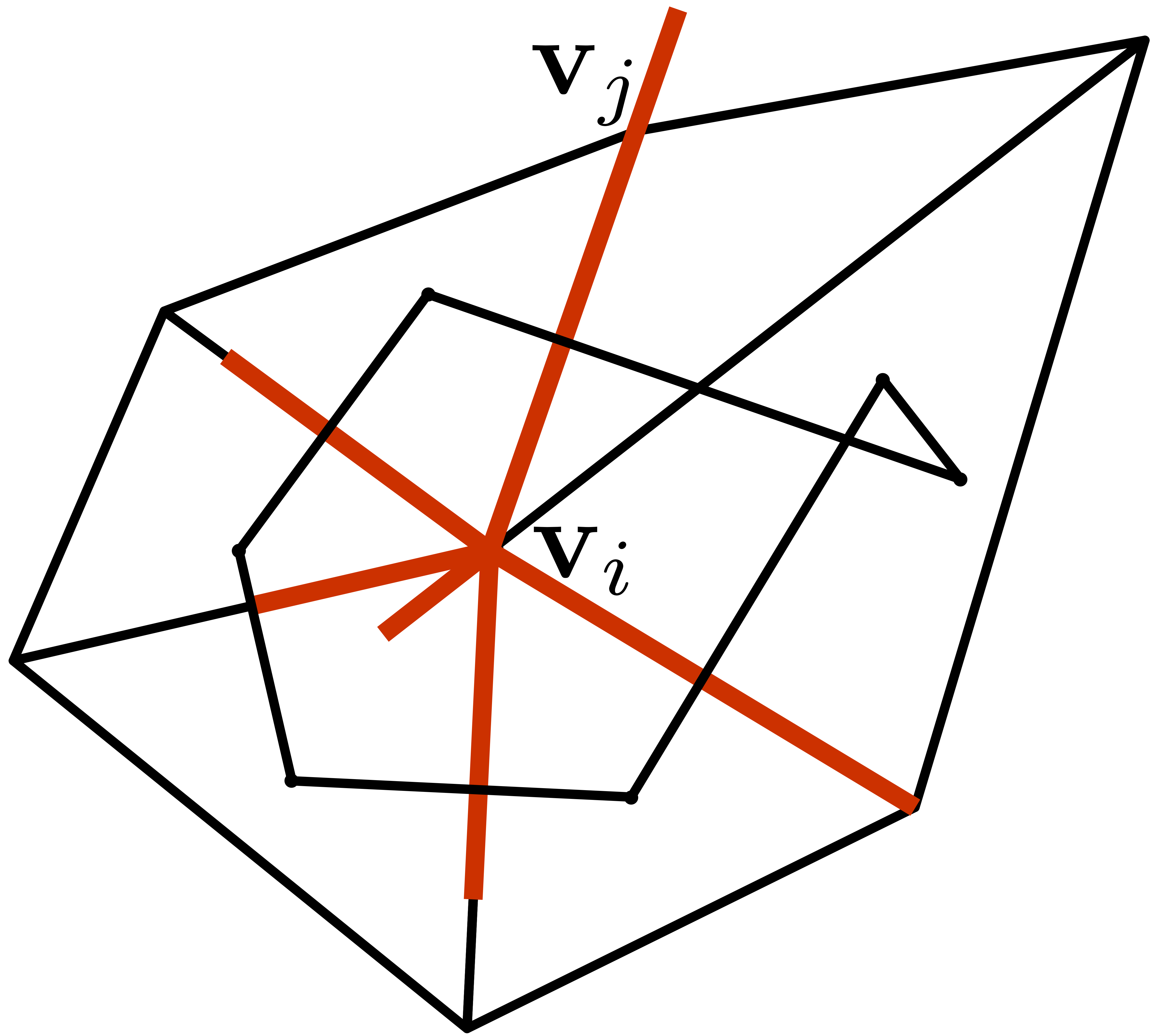
$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

- Orthogonal dual cells close!



Mesh Laplacian - Non-negativity

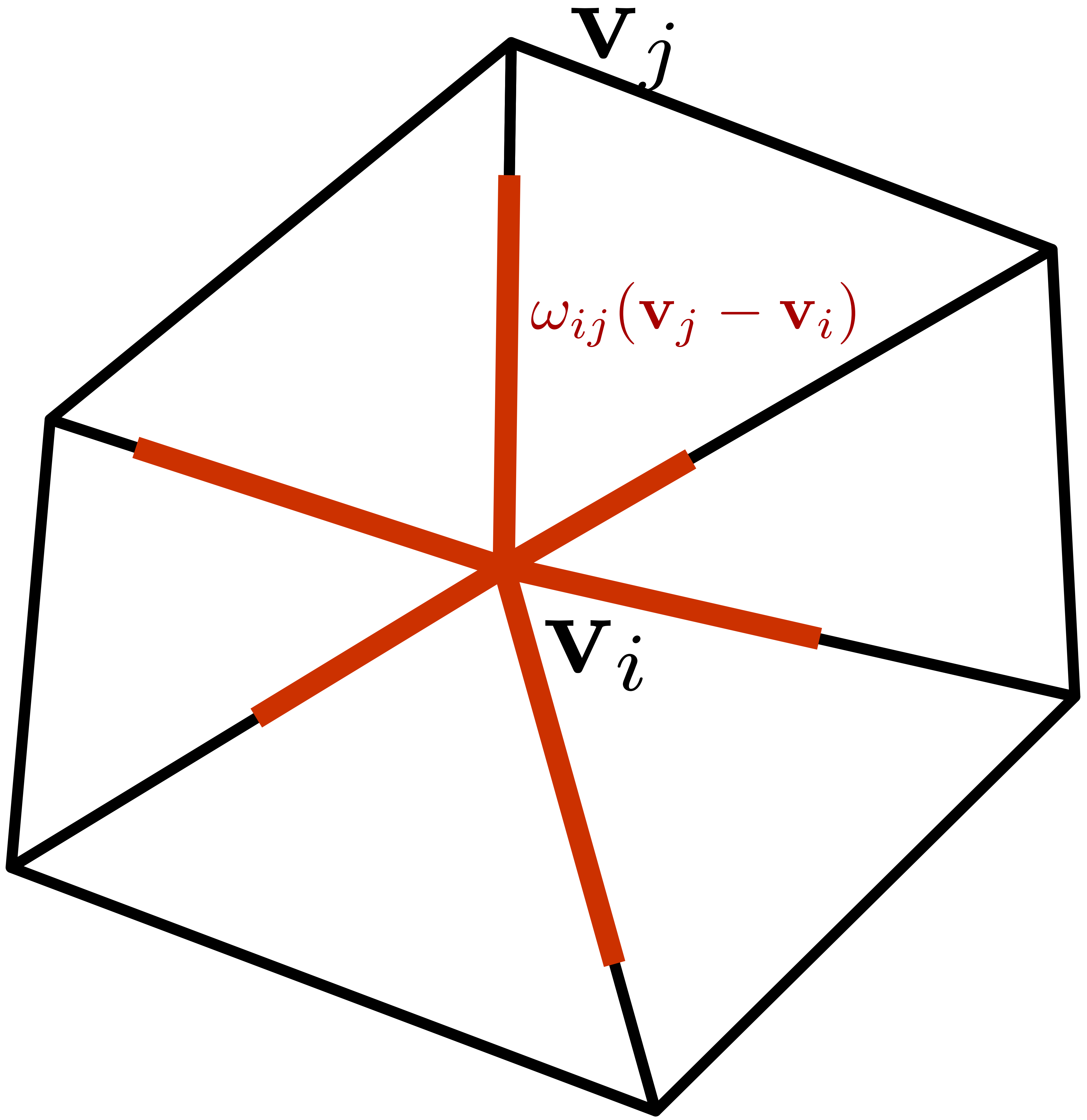
- Orthogonal dual cells close!
- Negative coefficients
    - orthogonal dual not embedded
    - No maximum principle!



Mesh Laplacian - Perfect

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

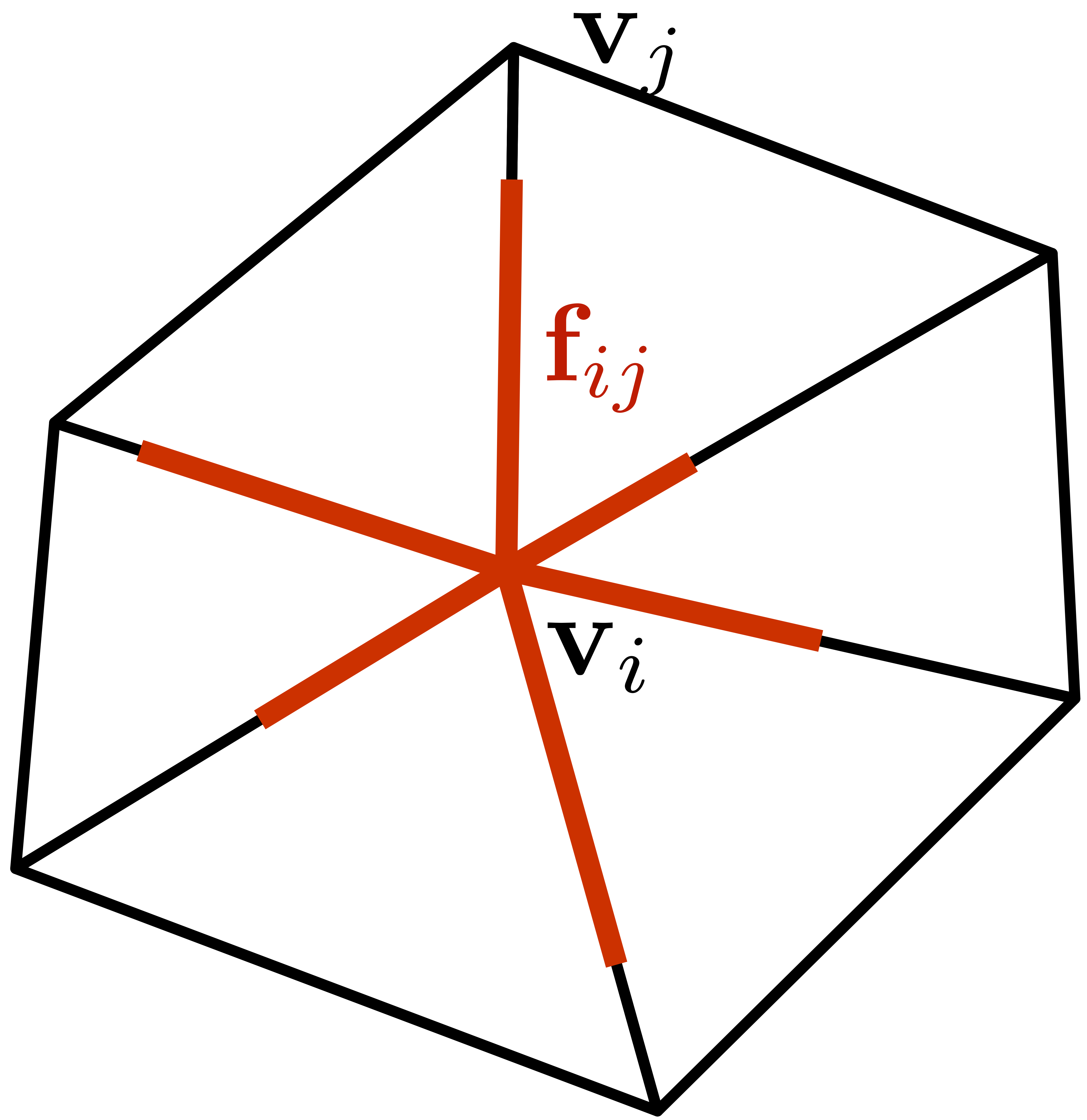
- Local, affine independence: by construction
- Symmetry: $\omega_{ij} = \omega_{ji}$
- Non-negative: $\omega_{ij} \geq 0$
- Linear precision: $\mathbf{LV} = \mathbf{0}$



Mesh Laplacian - Force network

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

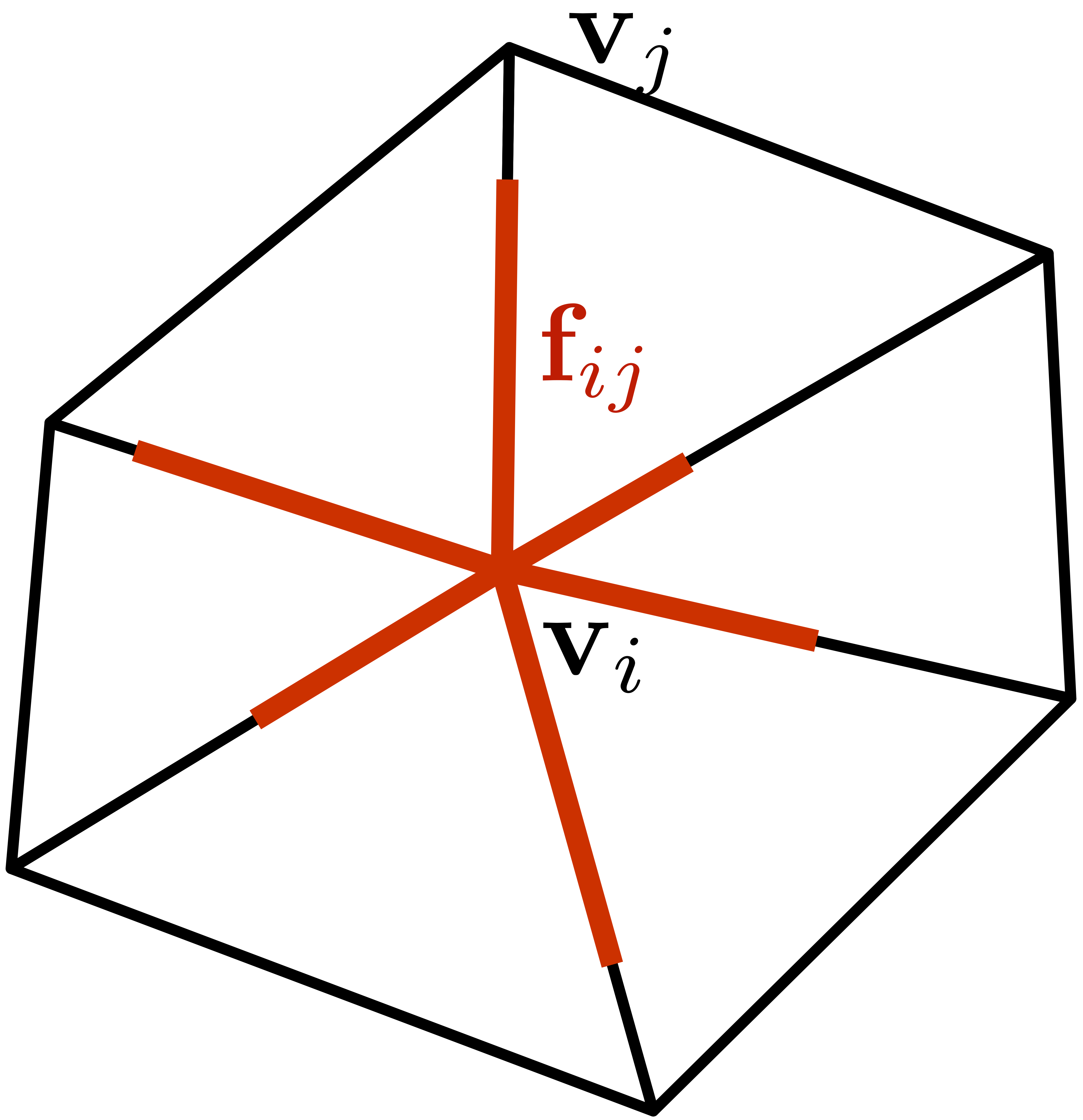
- Vertices connected by springs
- Hooks law: $\mathbf{f}_{ij} = \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$



Mesh Laplacian - Force network

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

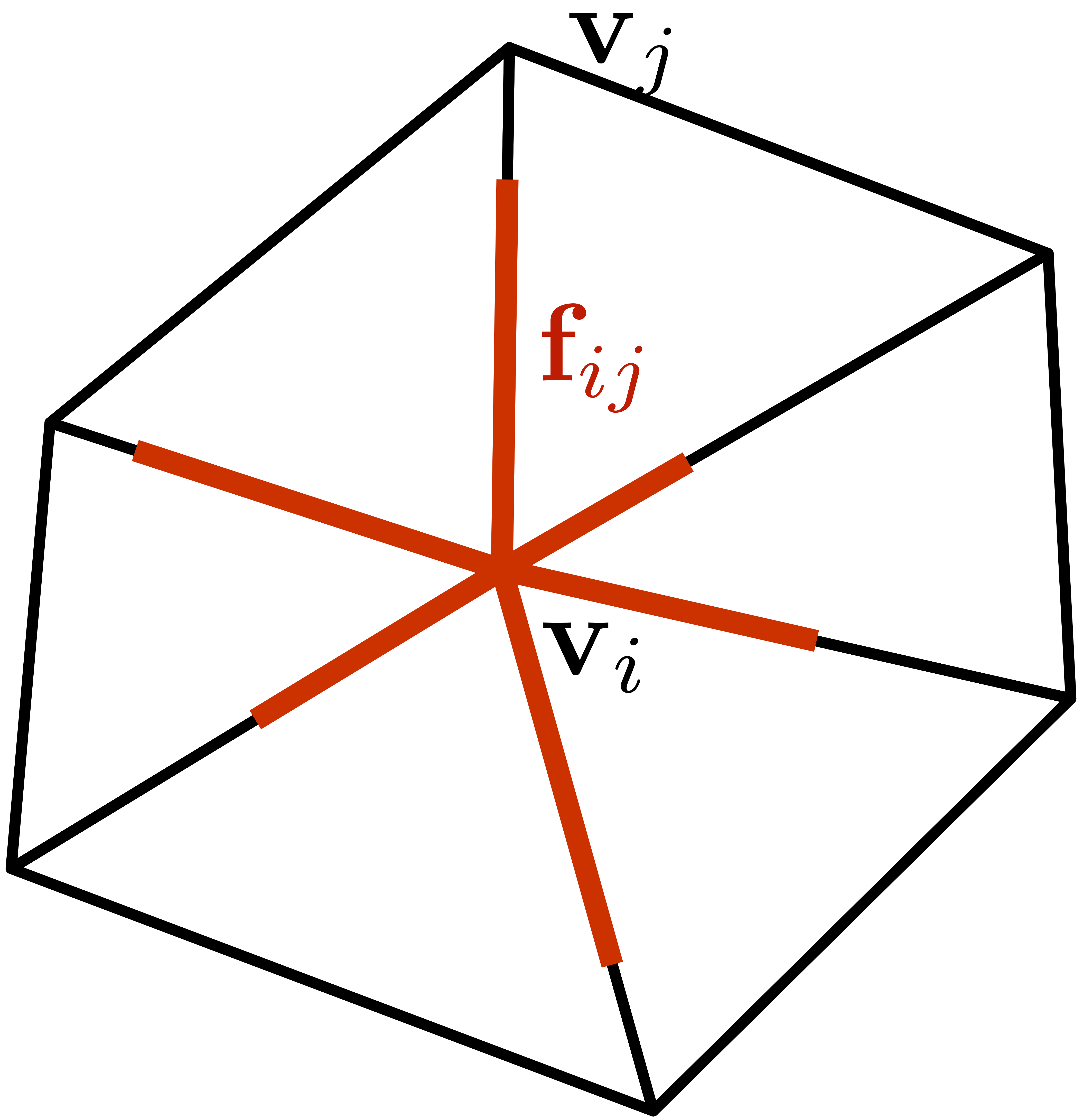
- Vertices connected by springs
- Hooks law: $\mathbf{f}_{ij} = \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$
- ω_{ij} is the spring constant
    - $\omega_{ij} > 0$ spring is pulling



Mesh Laplacian - Force network

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

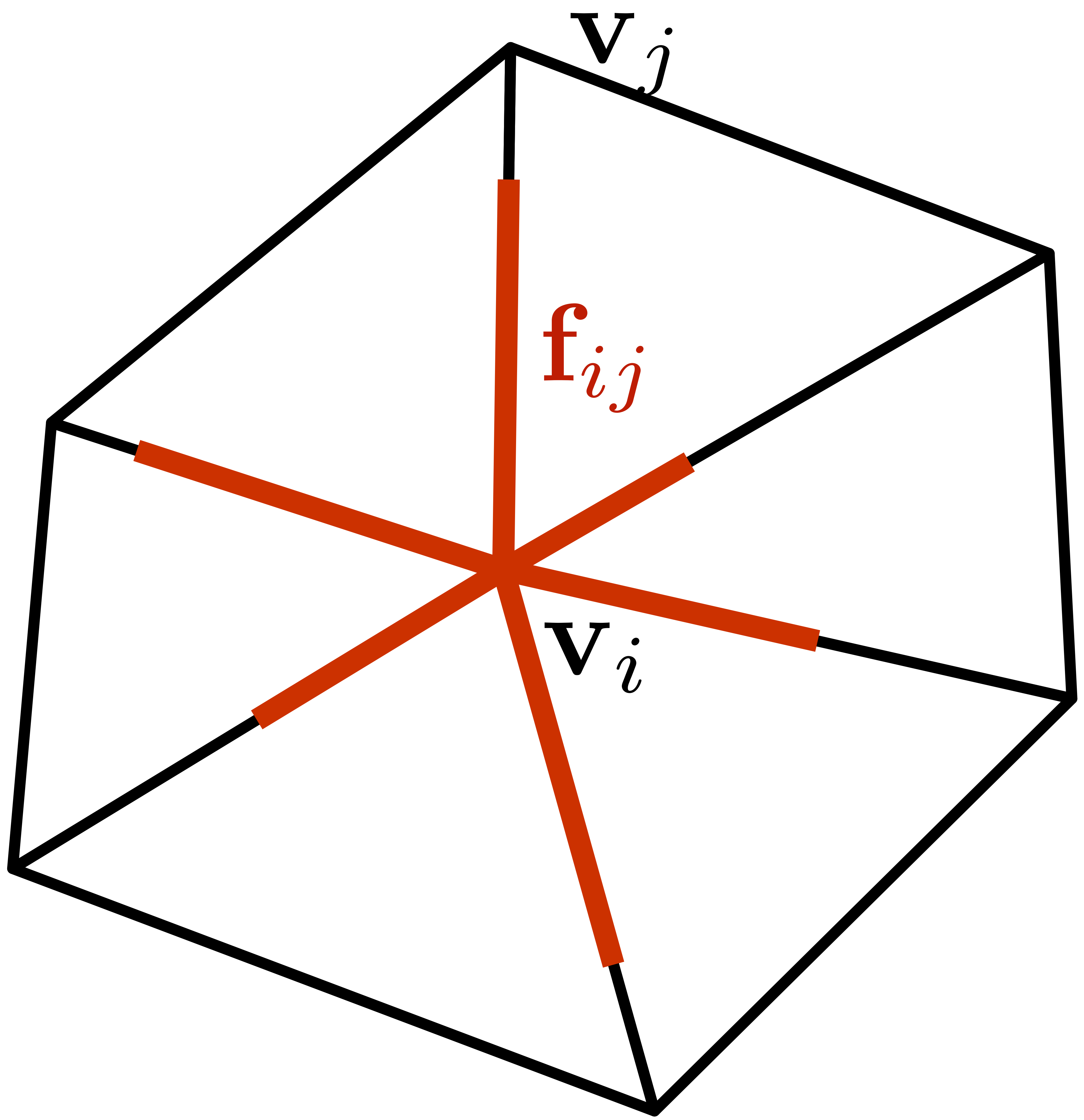
- Vertices connected by springs
- Hooks law: $\mathbf{f}_{ij} = \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$
- Linear precision: forces sum to zero
    - Force network is in equilibrium



Mesh Laplacian - Force network

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

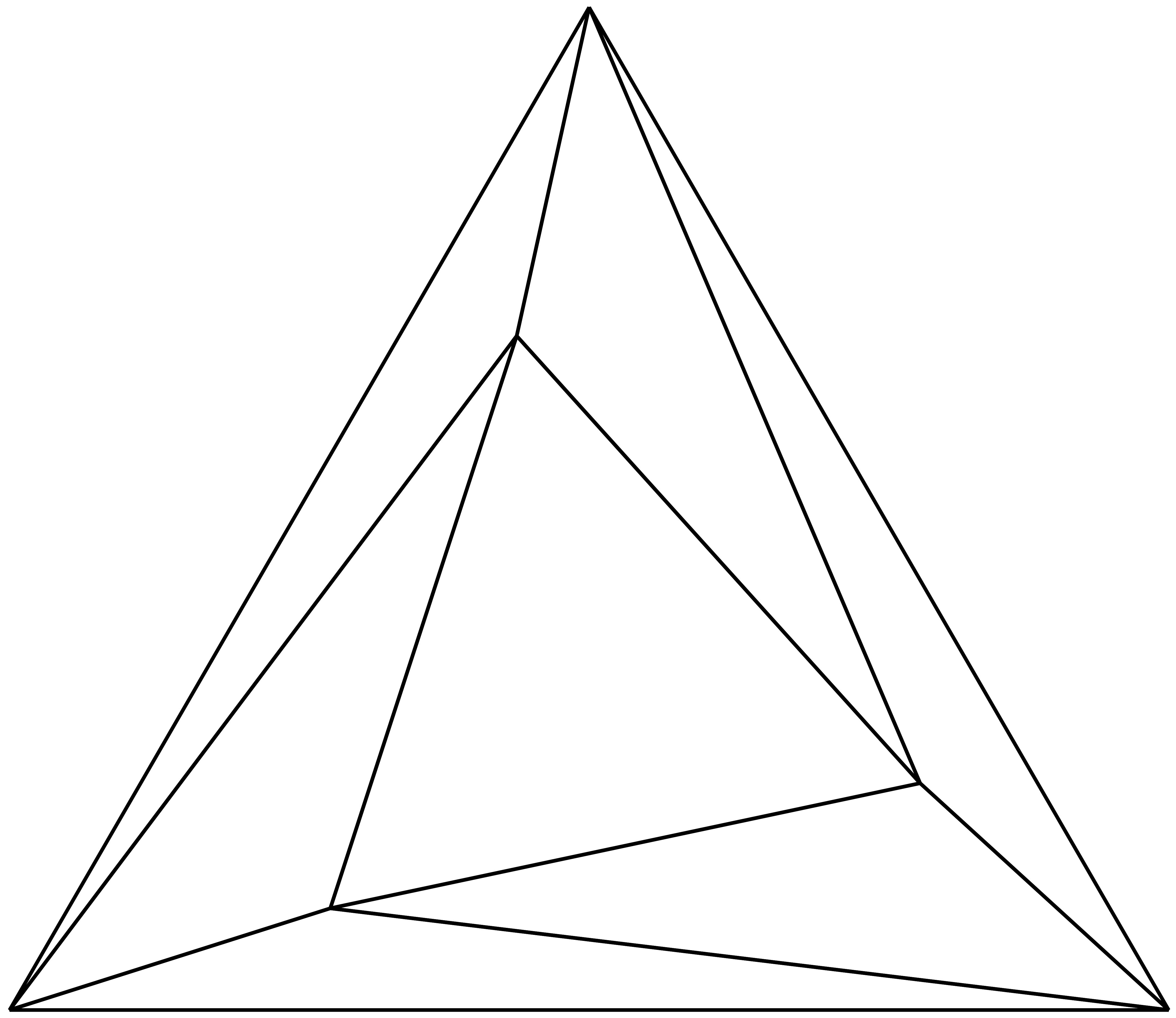
- Perfect Laplacian =
  - Non-negative spring constants
  - Vertices are in equilibrium



Mesh Laplacian - Force network

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

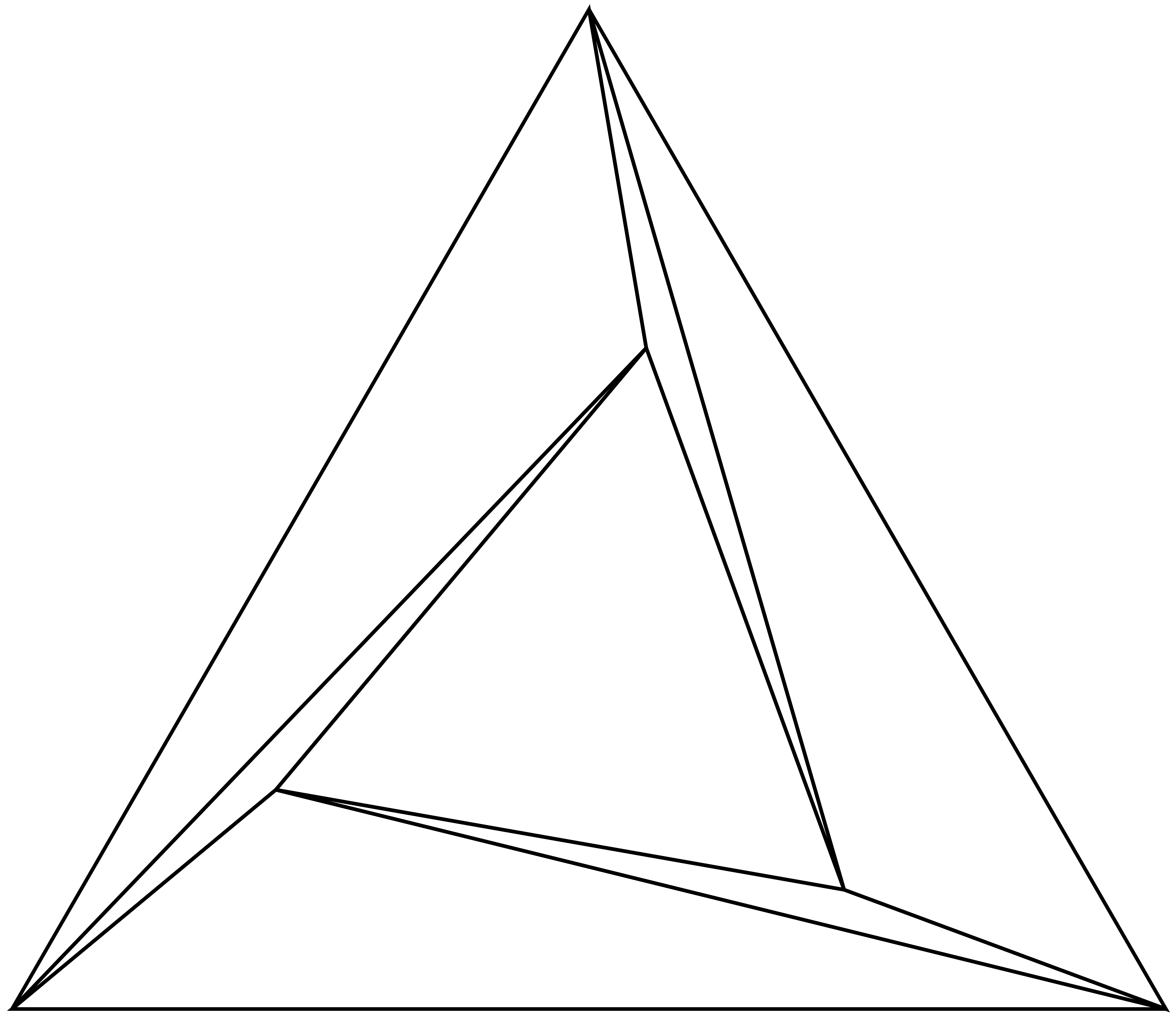
- Equilibrium: $\mathbf{V} \overset{?}{\to} \{\omega_{ij}\}$
- Mapping not unique
  - Glickenstein Laplace / Weighted Delaunay



Mesh Laplacian - Force network

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

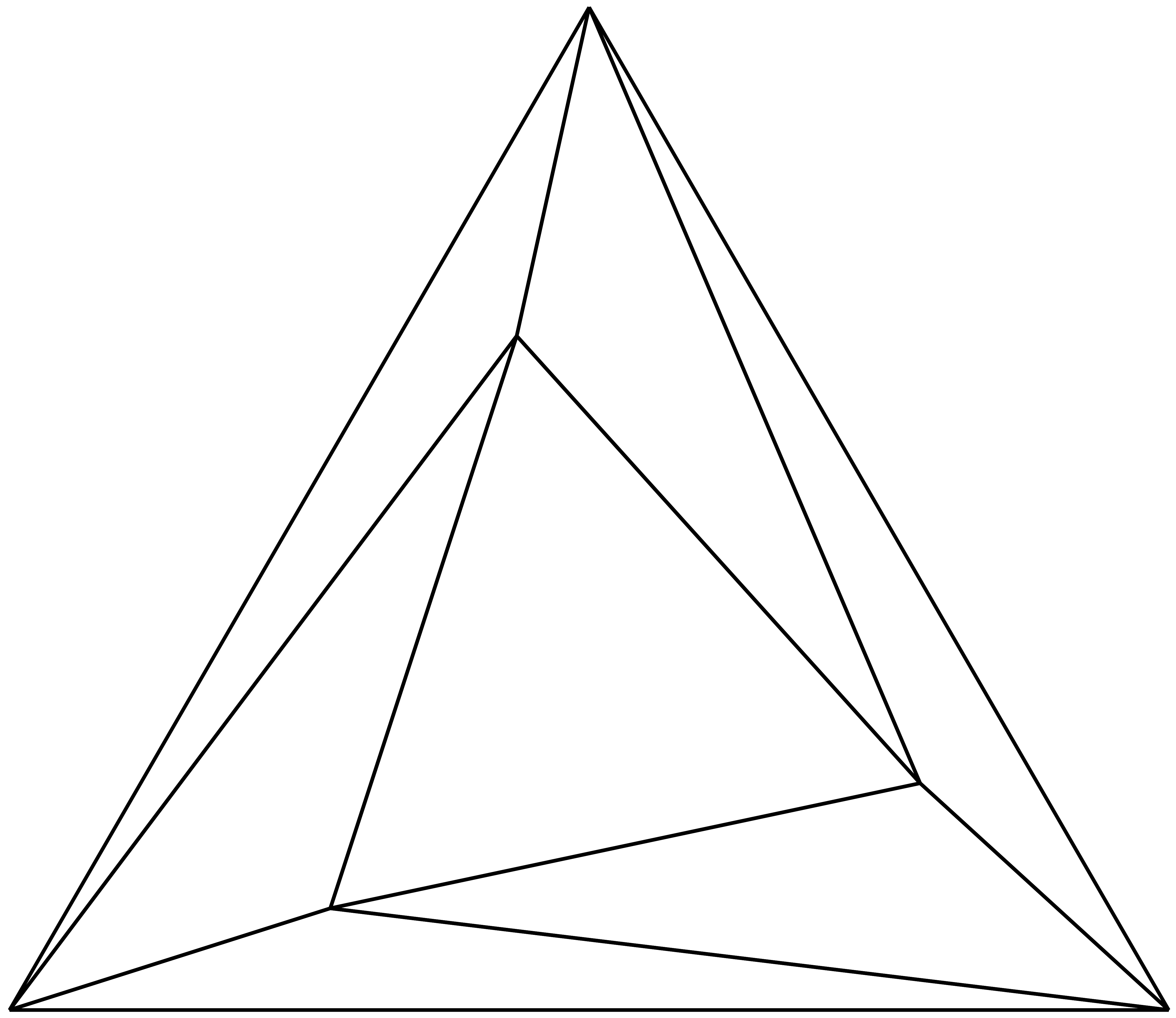
- Equilibrium: $\mathbf{V} \overset{?}{\to} \{\omega_{ij} \geq 0\}$
- Mapping may not exist
    - No free lunch theorem



Mesh Laplacian - Force network

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

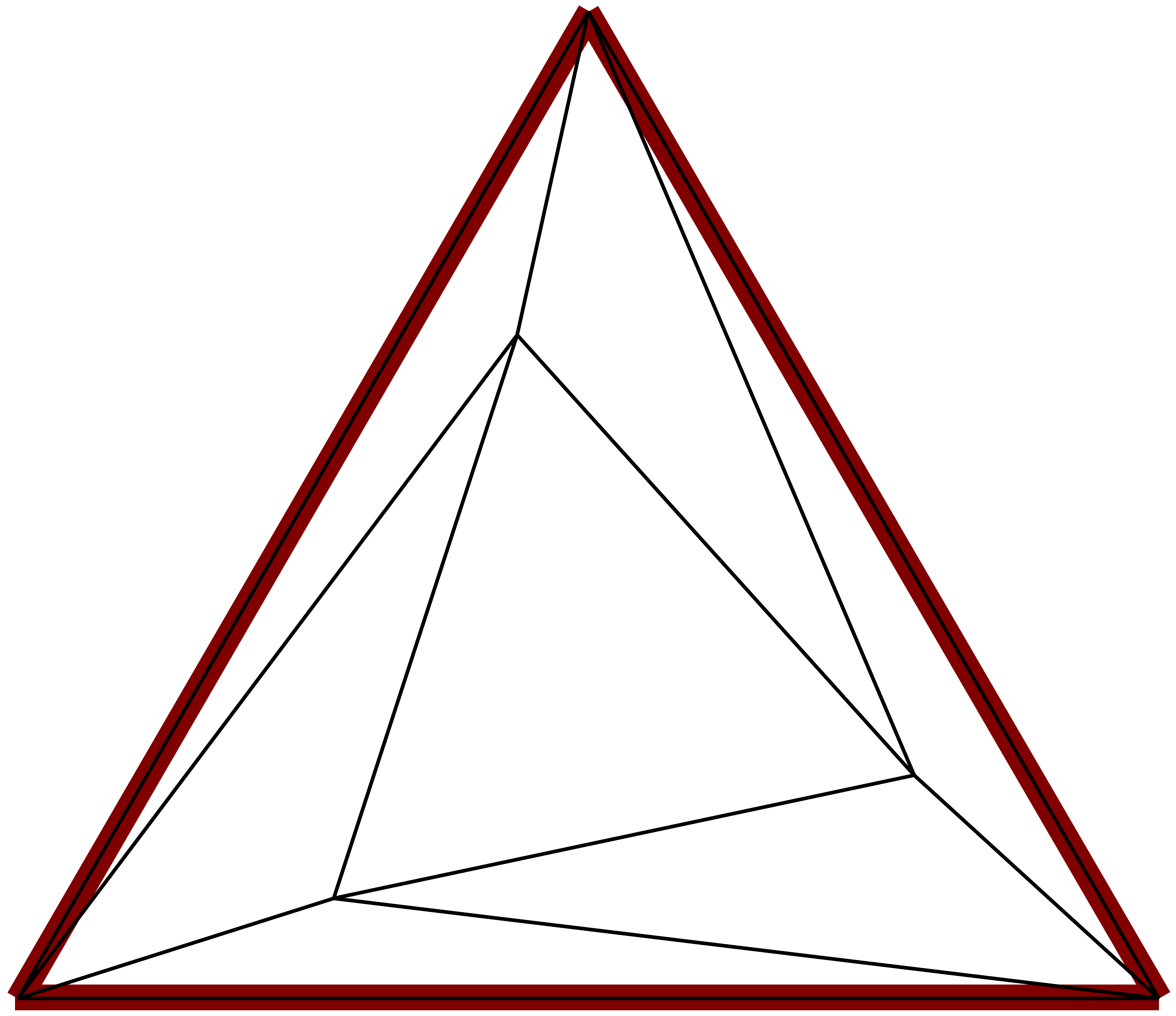
- Equilibrium: $\{\omega_{ij}\} \rightarrow \mathbf{V}_\Omega$



Mesh Laplacian - Force network

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

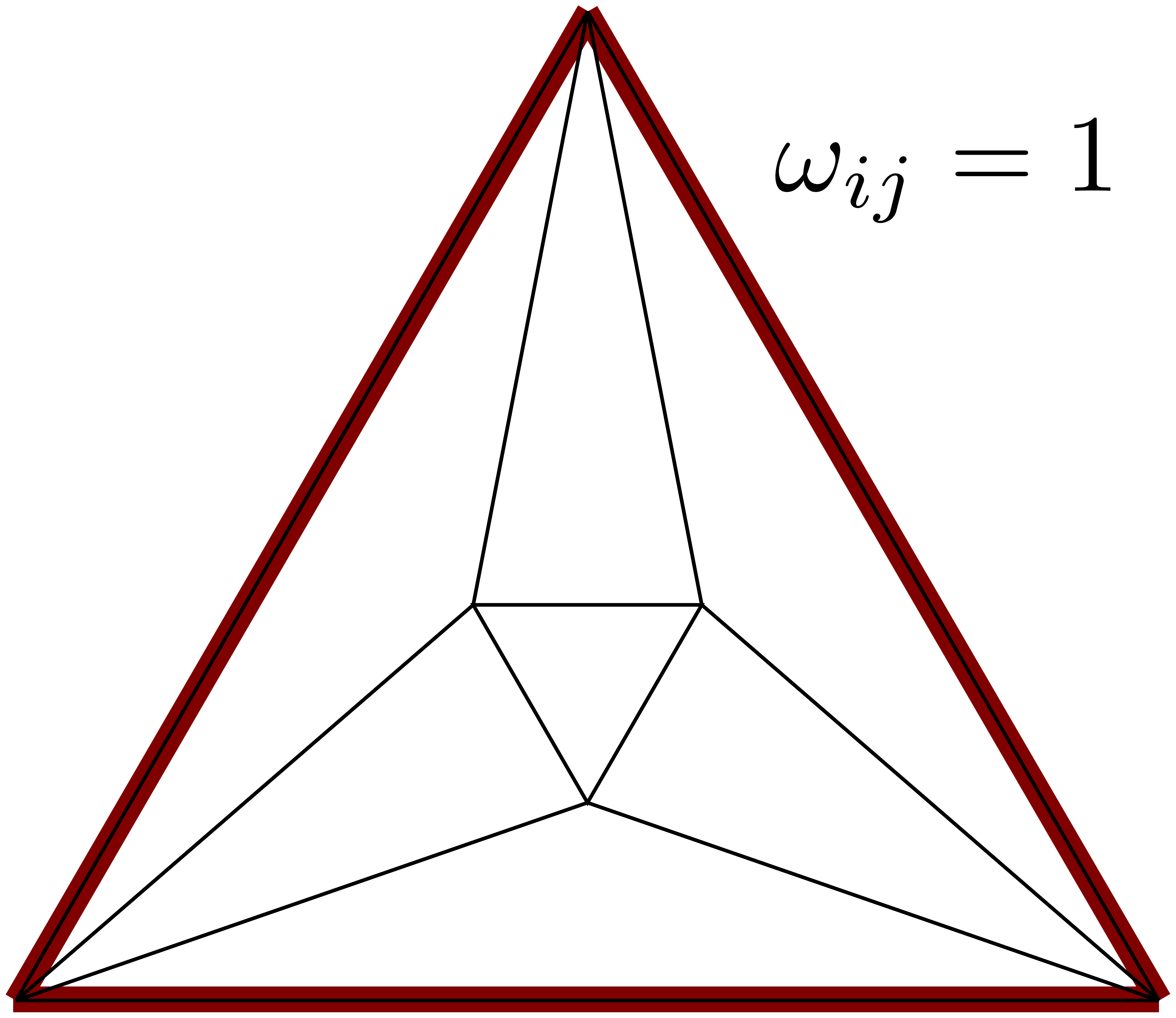
- Equilibrium: $\{\omega_{ij}\} \rightarrow \mathbf{V}_\Omega$
  - Fix boundary



Mesh Laplacian - Force network

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

- Equilibrium: $\{\omega_{ij}\} \rightarrow \mathbf{V}_\Omega$
  - Fix boundary
  - Solve $\mathbf{LV} = \mathbf{0}$
  - Unique

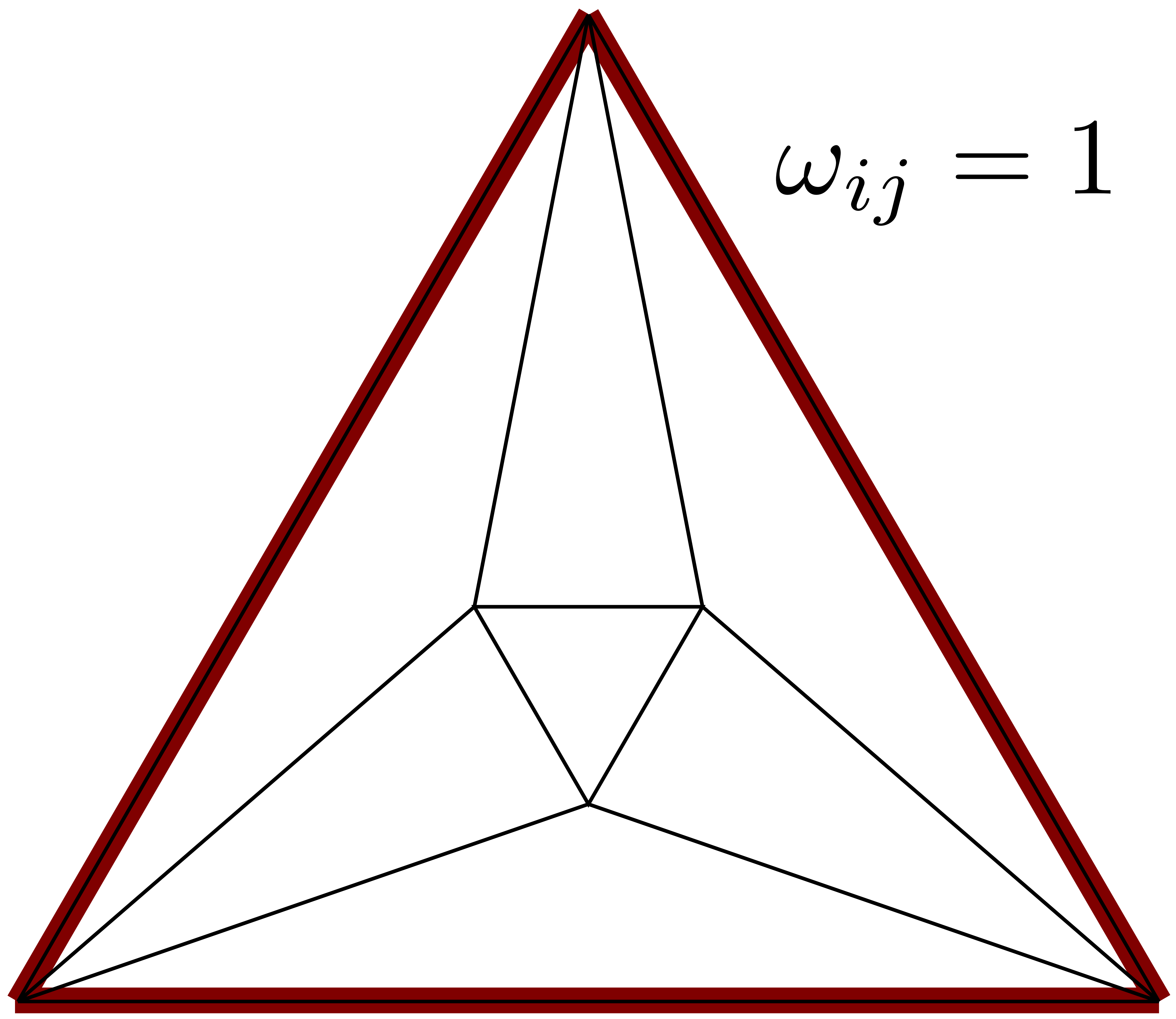


$\omega_{ij} = 1$

Mesh Laplacian - Force network

$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

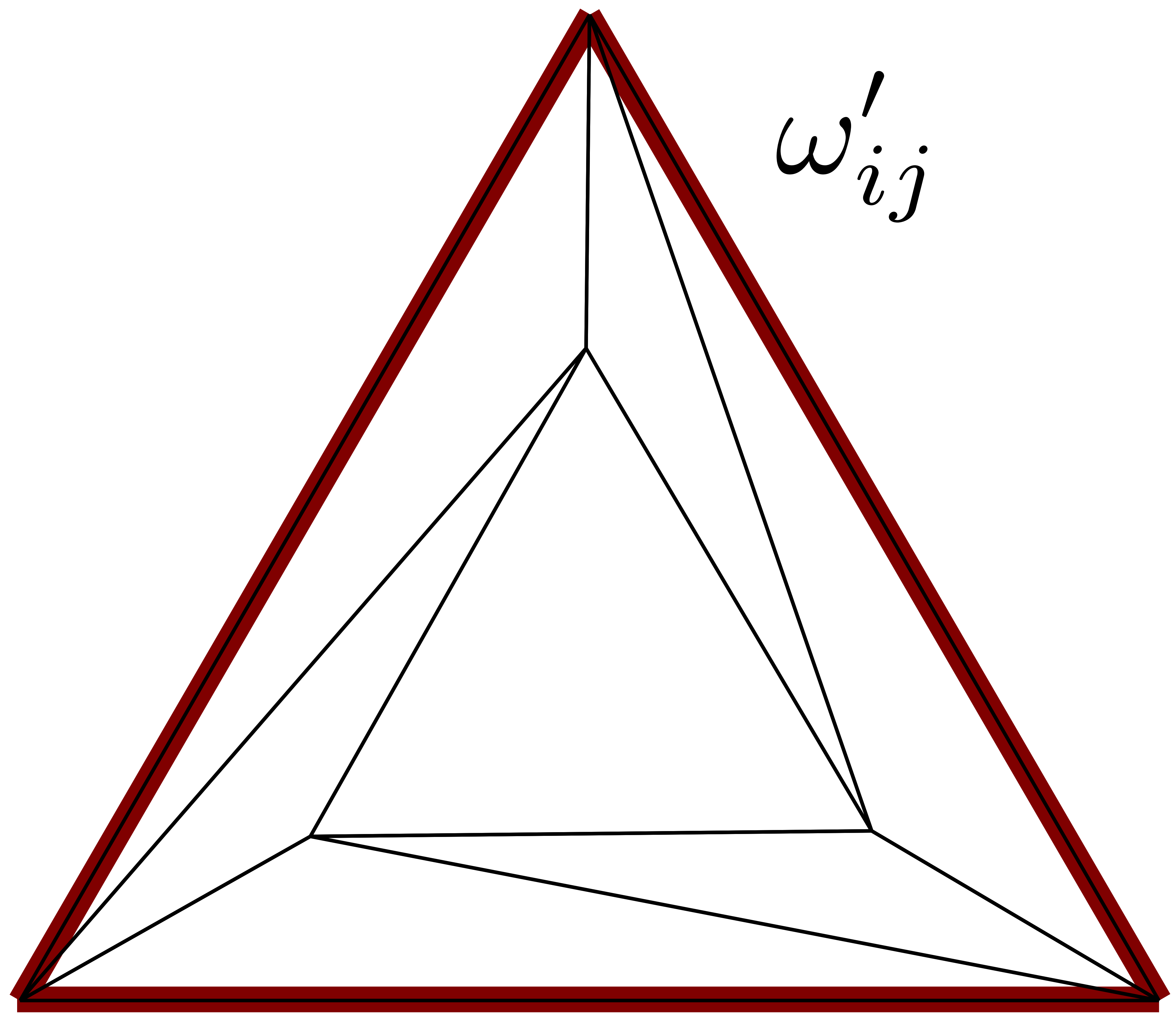
- Equilibrium: $\{\omega_{ij} \geq 0\} \to \mathbf{V}_\Omega$
  - Fix boundary
  - Solve $\mathbf{LV} = \mathbf{0}$
  - Unique embedding (Tutte)



Mesh Laplacian - Force network

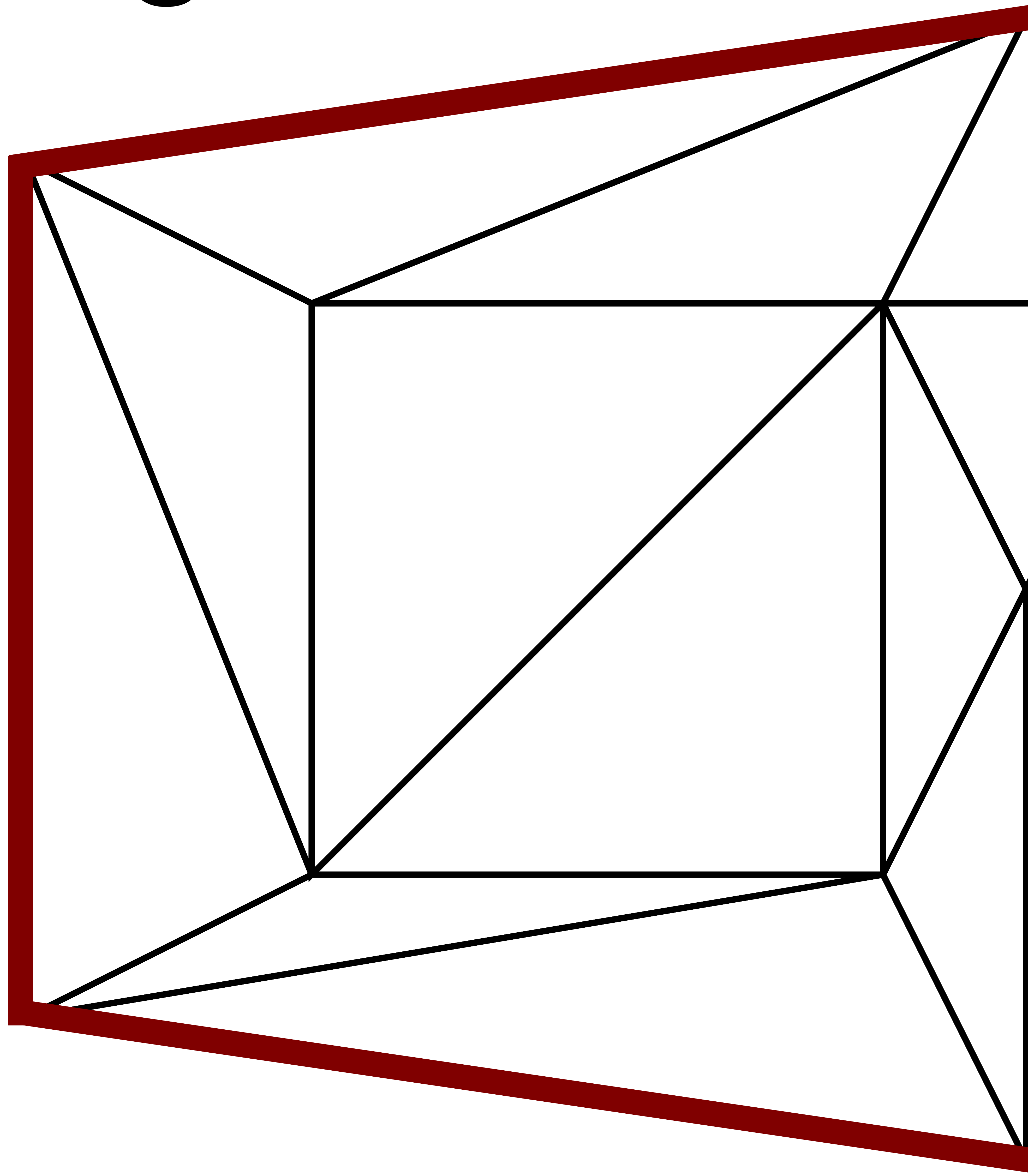
$$(\mathbf{LV})_i = \sum_{(i,j)\in E} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)$$

- Equilibrium: $\{\omega_{ij} \geq 0\} \to \mathbf{V}_\Omega$
  - Fix boundary
  - Solve $\mathbf{LV} = \mathbf{0}$
  - Unique embedding (Tutte)



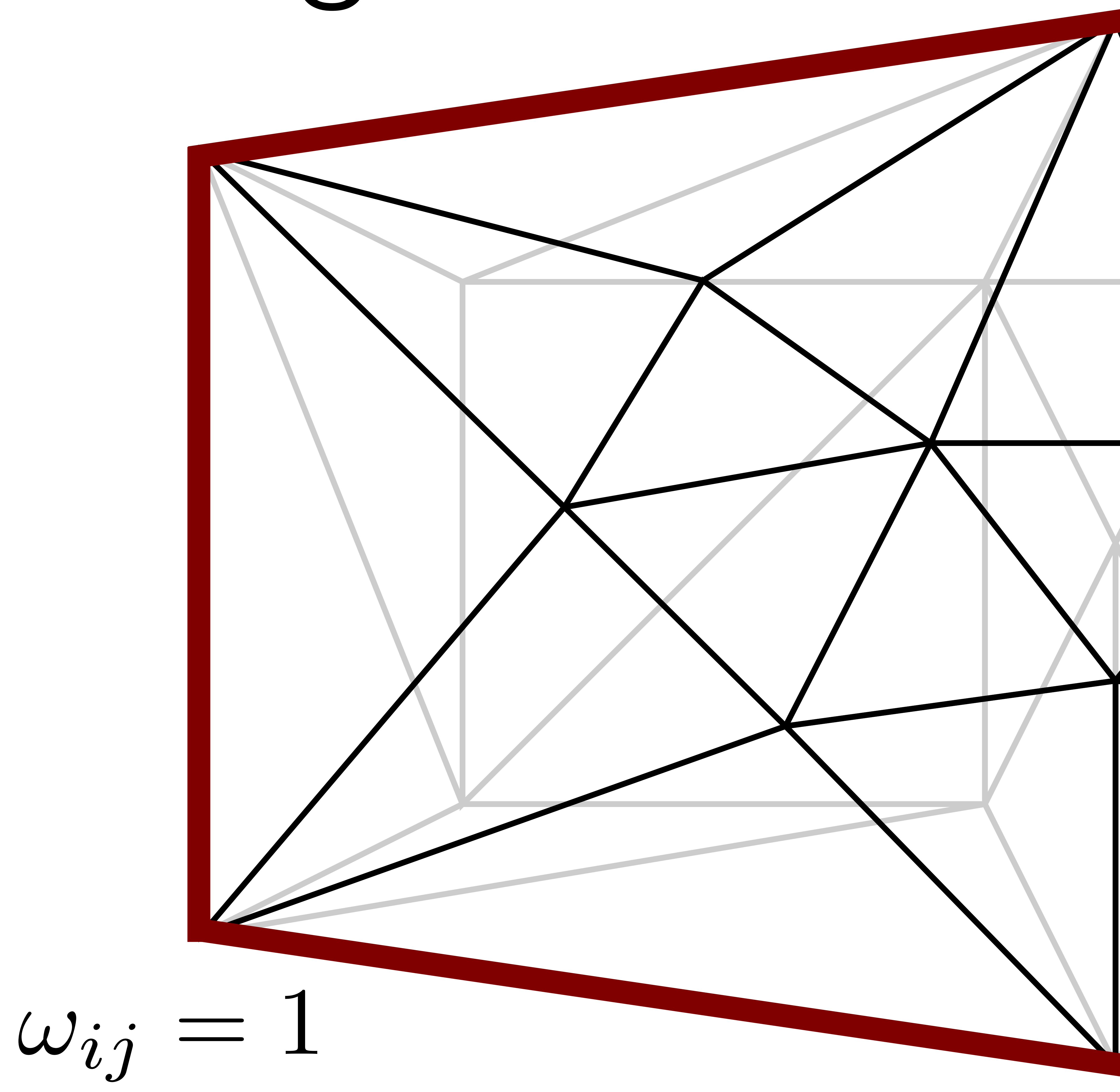
Mesh Laplacian - Algorithm

- Adjust ω_{ij} until $\mathbf{V}_\Omega = \mathbf{V}$



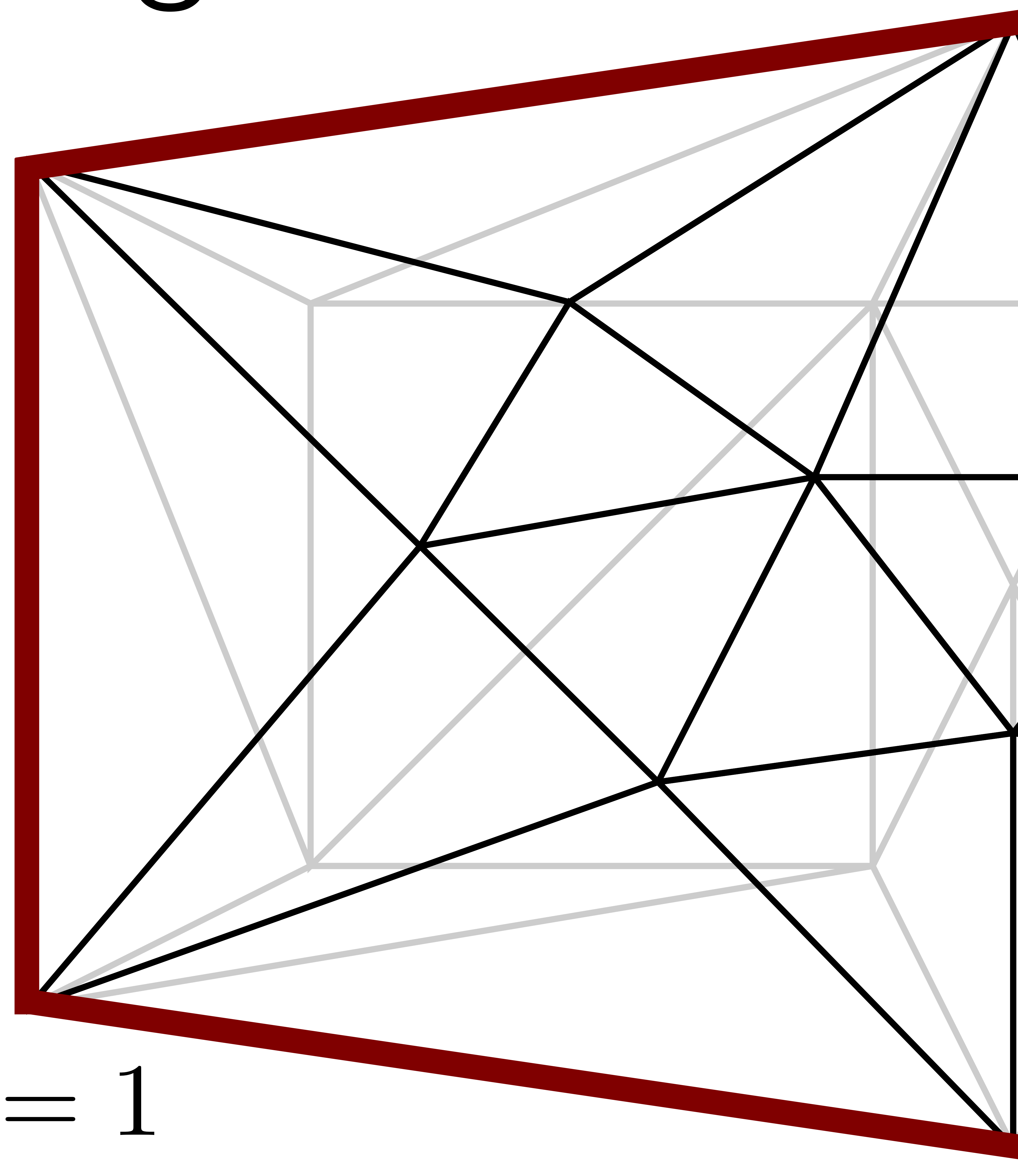
Mesh Laplacian - Algorithm

- Adjust ω_{ij} until $\mathbf{V}_\Omega = \mathbf{V}$
- Set $\omega_{ij} > 0$
  - Compute equilibrium: $\mathbf{L}\mathbf{V}_\Omega = \mathbf{0}$



Mesh Laplacian - Algorithm

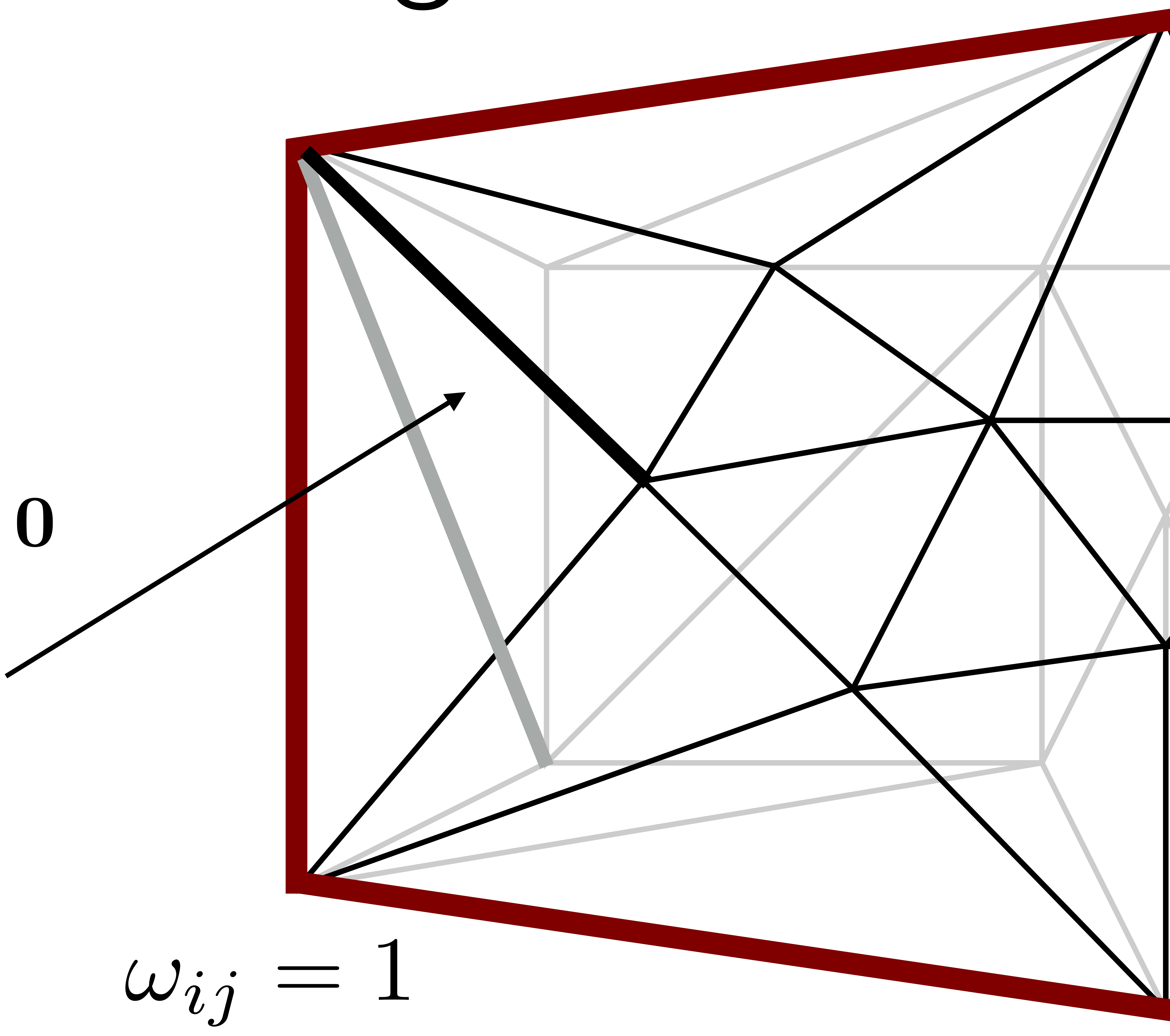
- Adjust ω_{ij} until $\mathbf{V}_\Omega = \mathbf{V}$
- Set $\omega_{ij} > 0$
  - Compute equilibrium: $\mathbf{L}\mathbf{V}_\Omega = \mathbf{0}$
  - Laplacian is intrinsic:
    Just check edge lengths!



$\omega_{ij} = 1$

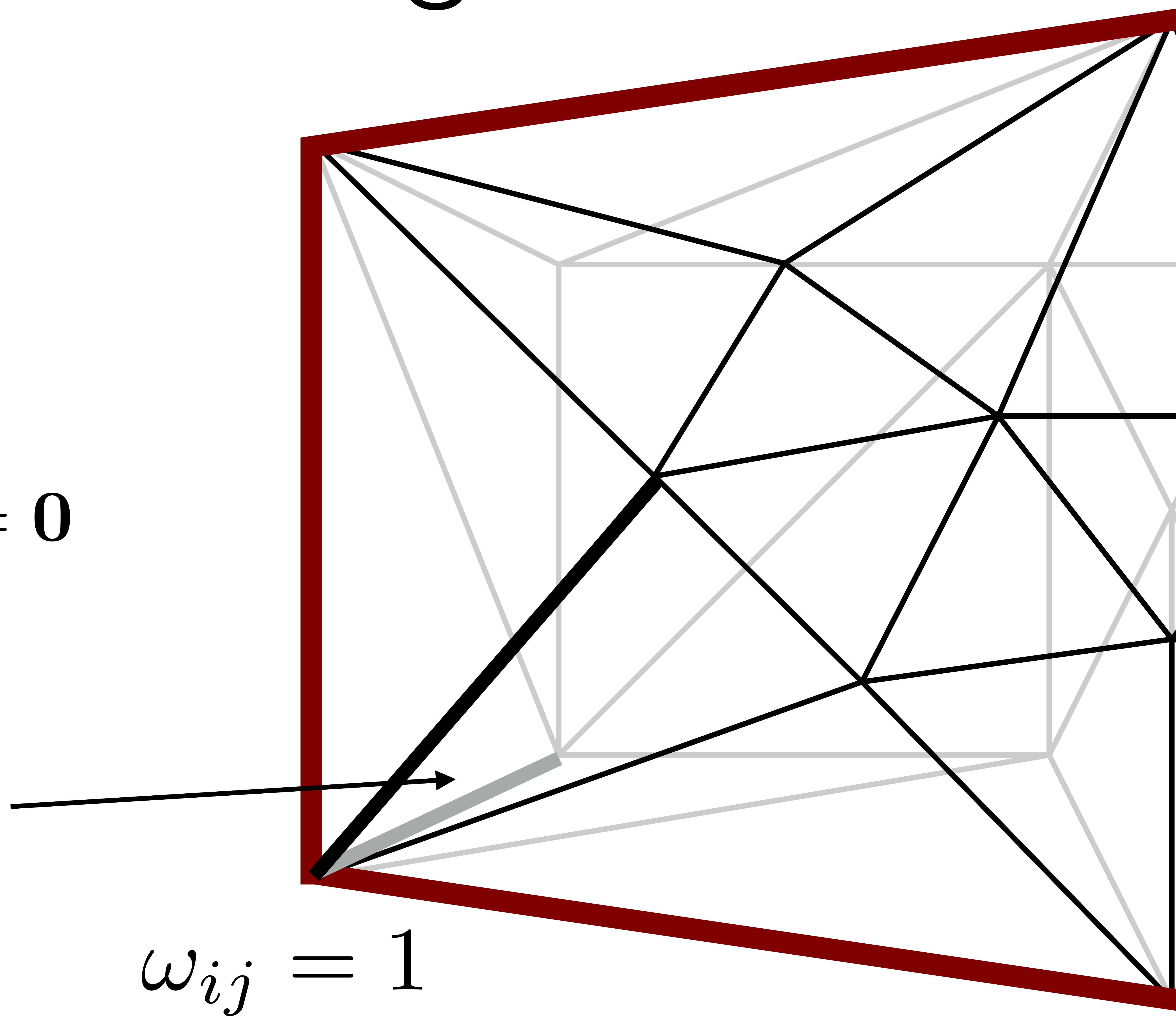
Mesh Laplacian - Algorithm

- Adjust ω_{ij} until $\mathbf{V}_\Omega = \mathbf{V}$
- Set $\omega_{ij} > 0$
  - Compute equilibrium: $\mathbf{L}\mathbf{V}_\Omega = \mathbf{0}$
  - Edge too short: loosen spring



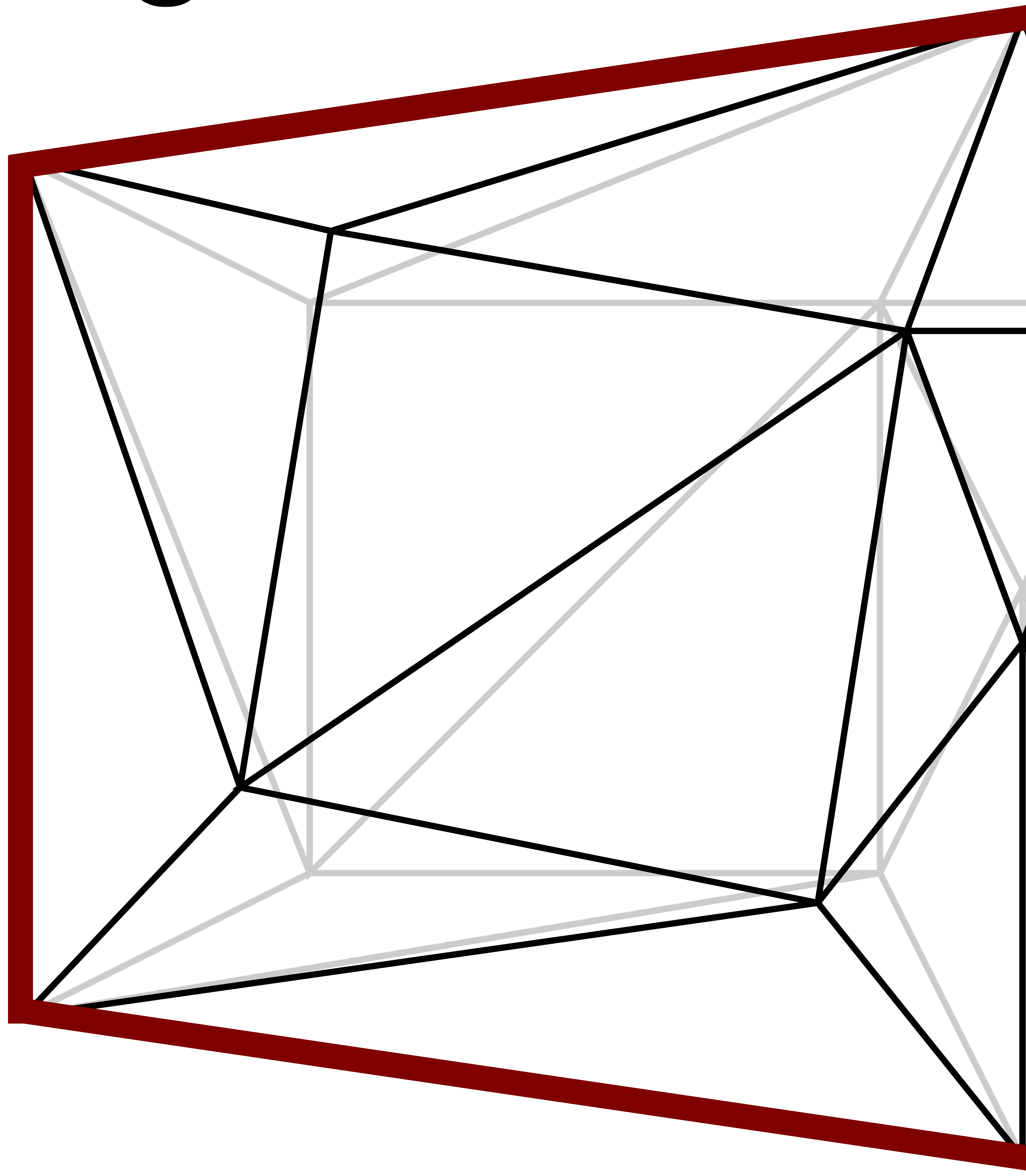
Mesh Laplacian - Algorithm

- Adjust ω_{ij} until $\mathbf{V}_\Omega = \mathbf{V}$
- Set $\omega_{ij} > 0$
  - Compute equilibrium: $\mathbf{L}\mathbf{V}_\Omega = \mathbf{0}$
  - Edge too short: loosen spring
  - Edge too long: tighten spring



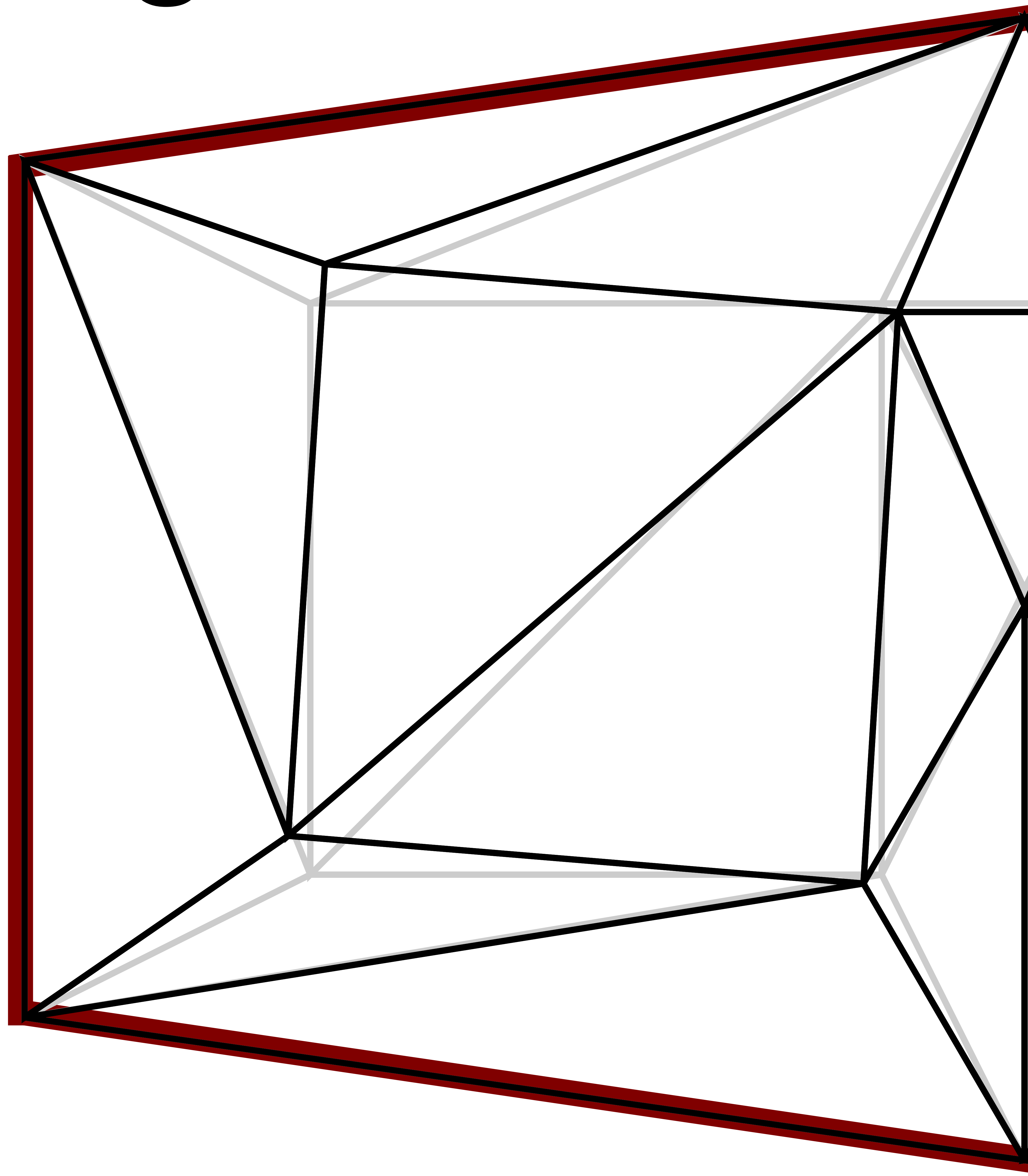
Mesh Laplacian - Algorithm

- Adjust ω_{ij} until $\mathbf{V}_\Omega = \mathbf{V}$
- Set $\omega_{ij} > 0$
  - Compute equilibrium: $\mathbf{L}\mathbf{V}_\Omega = \mathbf{0}$
  - Edge too short: loosen spring
  - Edge too long: tighten spring



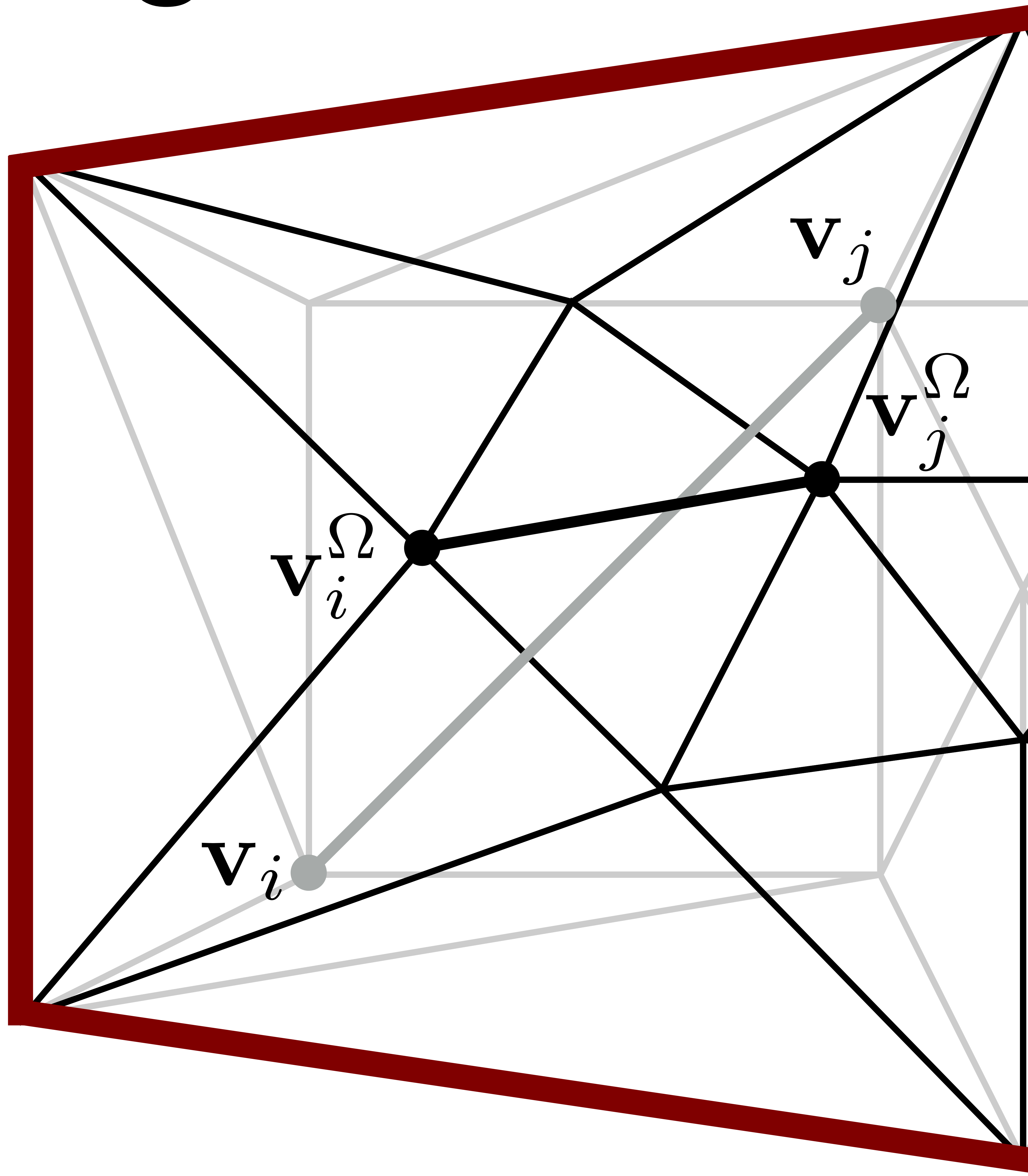
Mesh Laplacian - Algorithm

- Adjust ω_{ij} until $\mathbf{V}_\Omega = \mathbf{V}$
- Set $\omega_{ij} > 0$
  - Compute equilibrium: $\mathbf{L}\mathbf{V}_\Omega = \mathbf{0}$
  - Adjust springs
- Until convergence



Mesh Laplacian - Algorithm

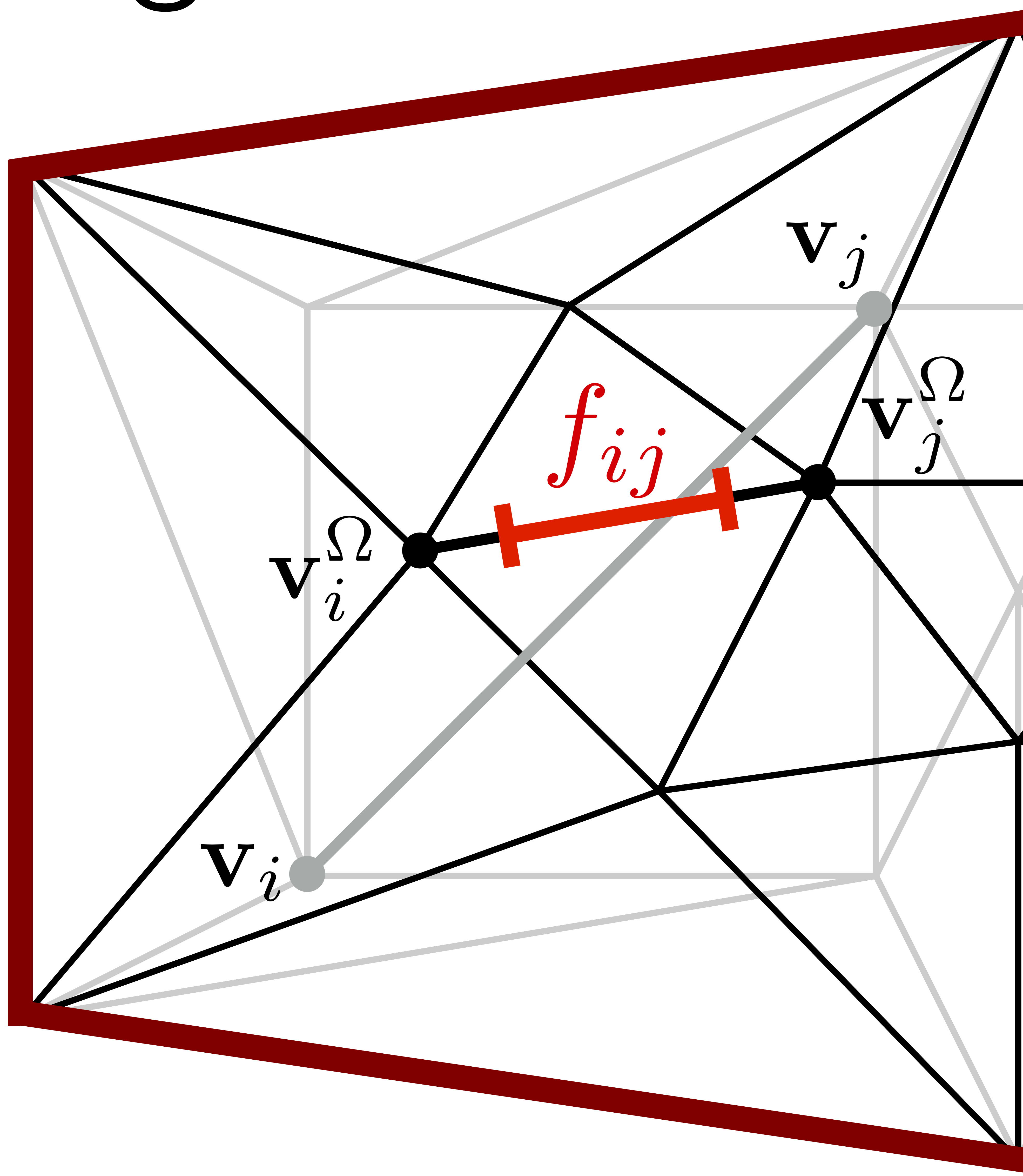
- Adjusting spring constant ω_{ij}



Mesh Laplacian - Algorithm

- Adjusting spring constant ω_{ij}
  - (Scalar) force on current edge:

$$f_{ij} = \omega_{ij} \| \mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega} \|$$

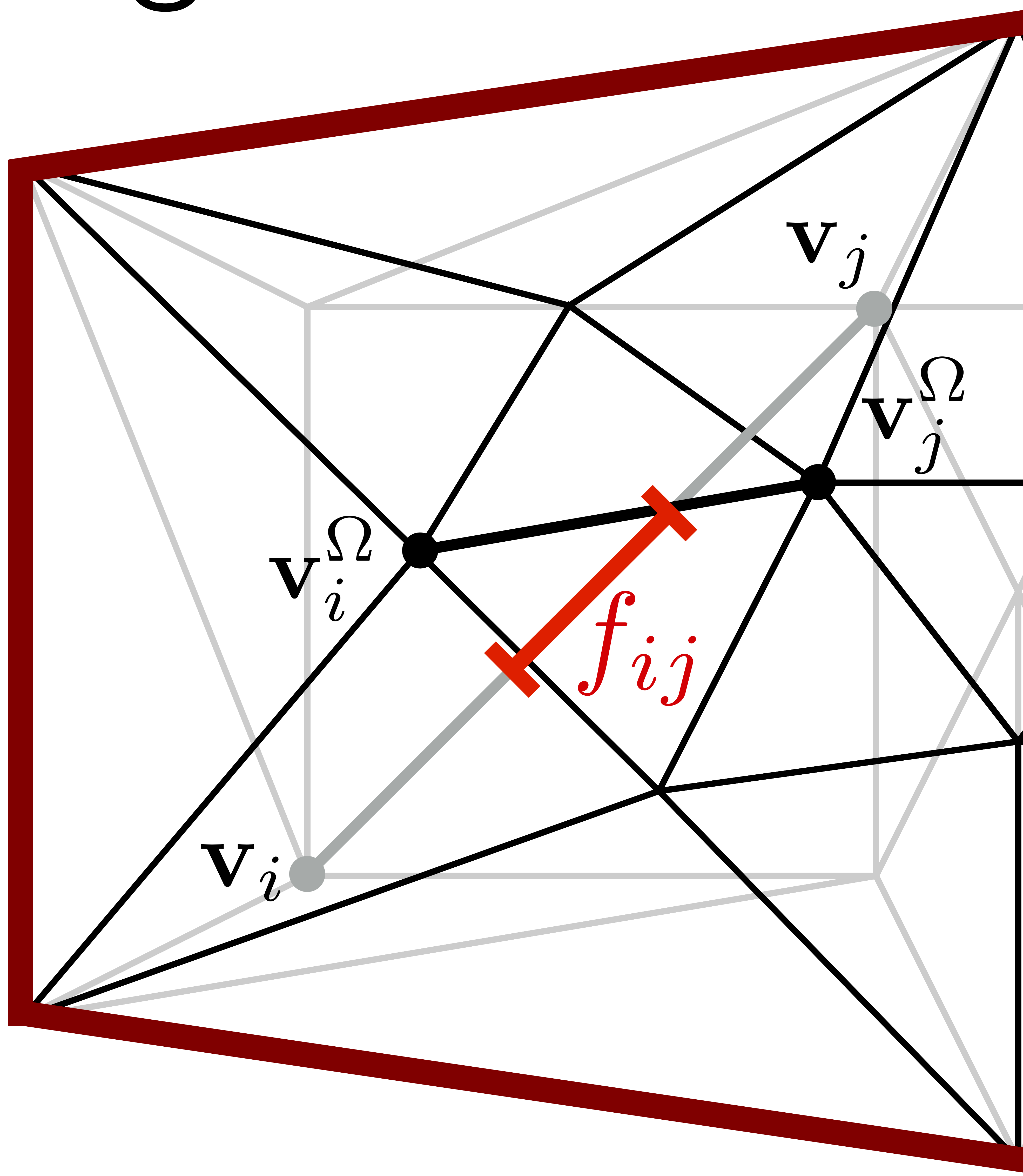


Mesh Laplacian - Algorithm

- Adjusting spring constant ω_{ij}
  - (Scalar) force on current edge:

$$f_{ij} = \omega_{ij} \|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\|$$

  - Assume force is constant



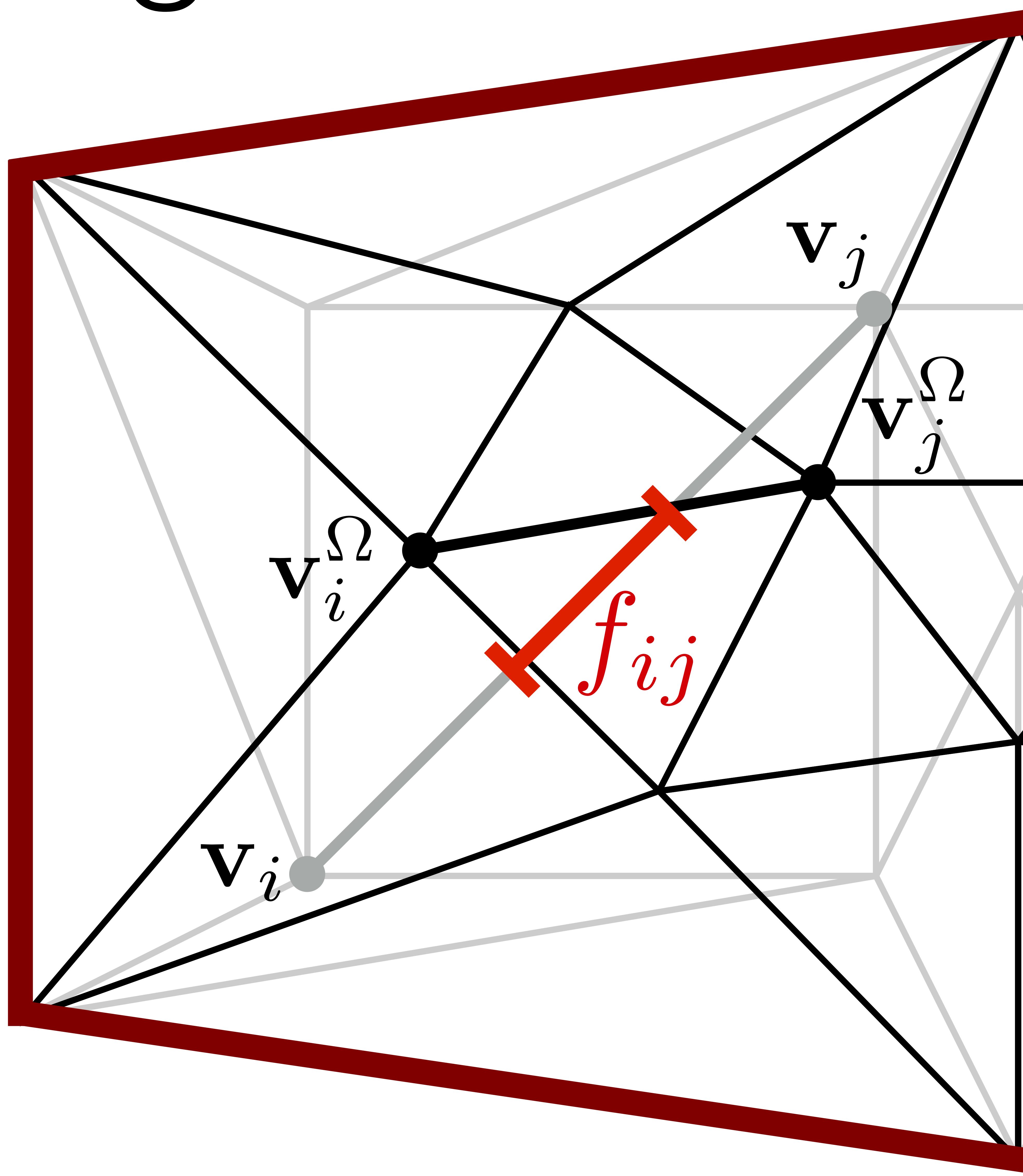
Mesh Laplacian - Algorithm

- Adjusting spring constant ω_{ij}
  - (Scalar) force on current edge:

$$f_{ij} = \omega_{ij} \| \mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega} \|$$

  - Spring constant for desired edge length:

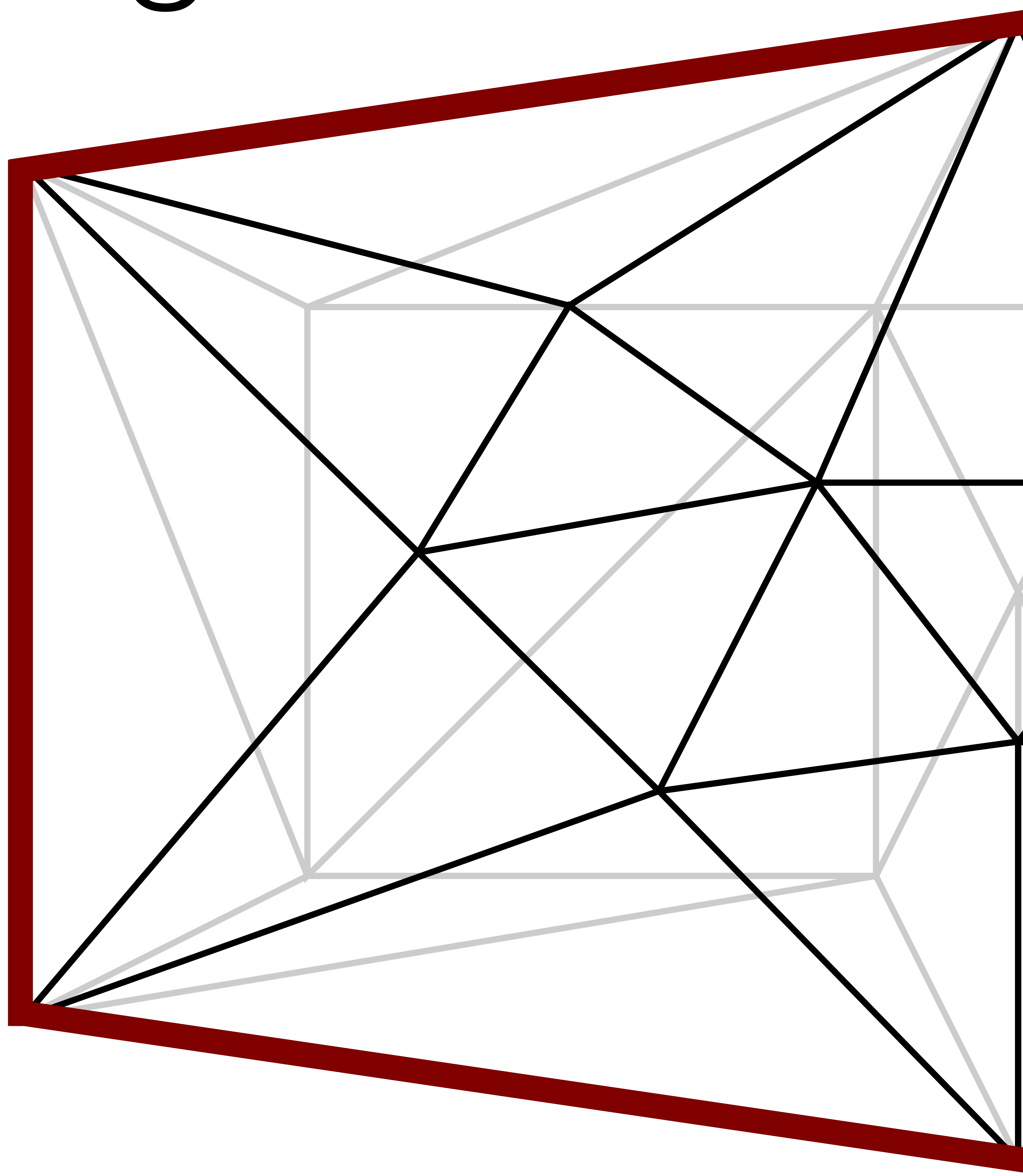
$$\omega'_{ij} = \frac{f_{ij}}{\| \mathbf{v}_j - \mathbf{v}_i \|}$$



Mesh Laplacian - Algorithm

- Adjusting spring constants
- Update rule

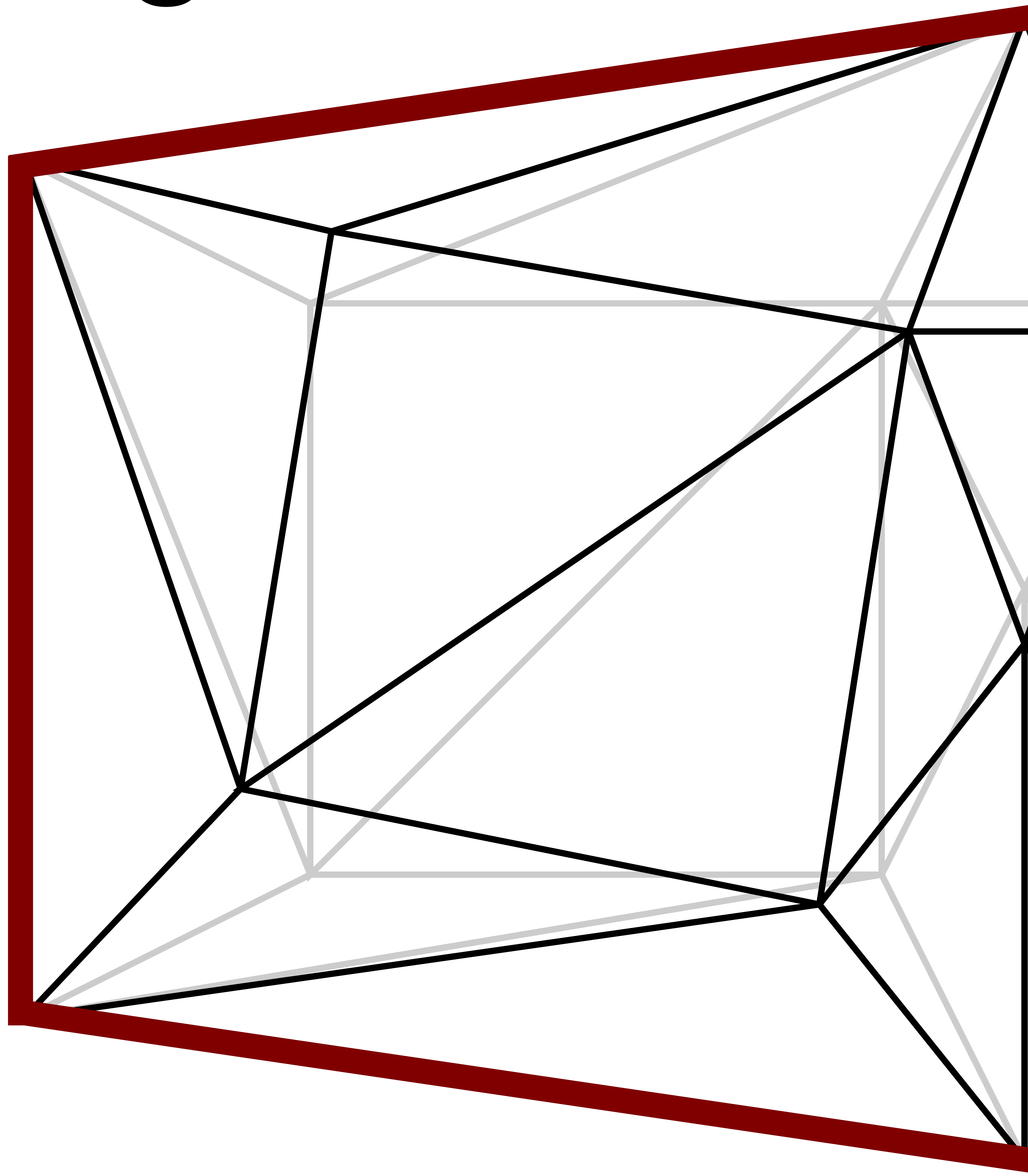
$$\omega'_{ij} = \omega_{ij} \frac{\|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



Mesh Laplacian - Algorithm

- Adjusting spring constants
- Update rule

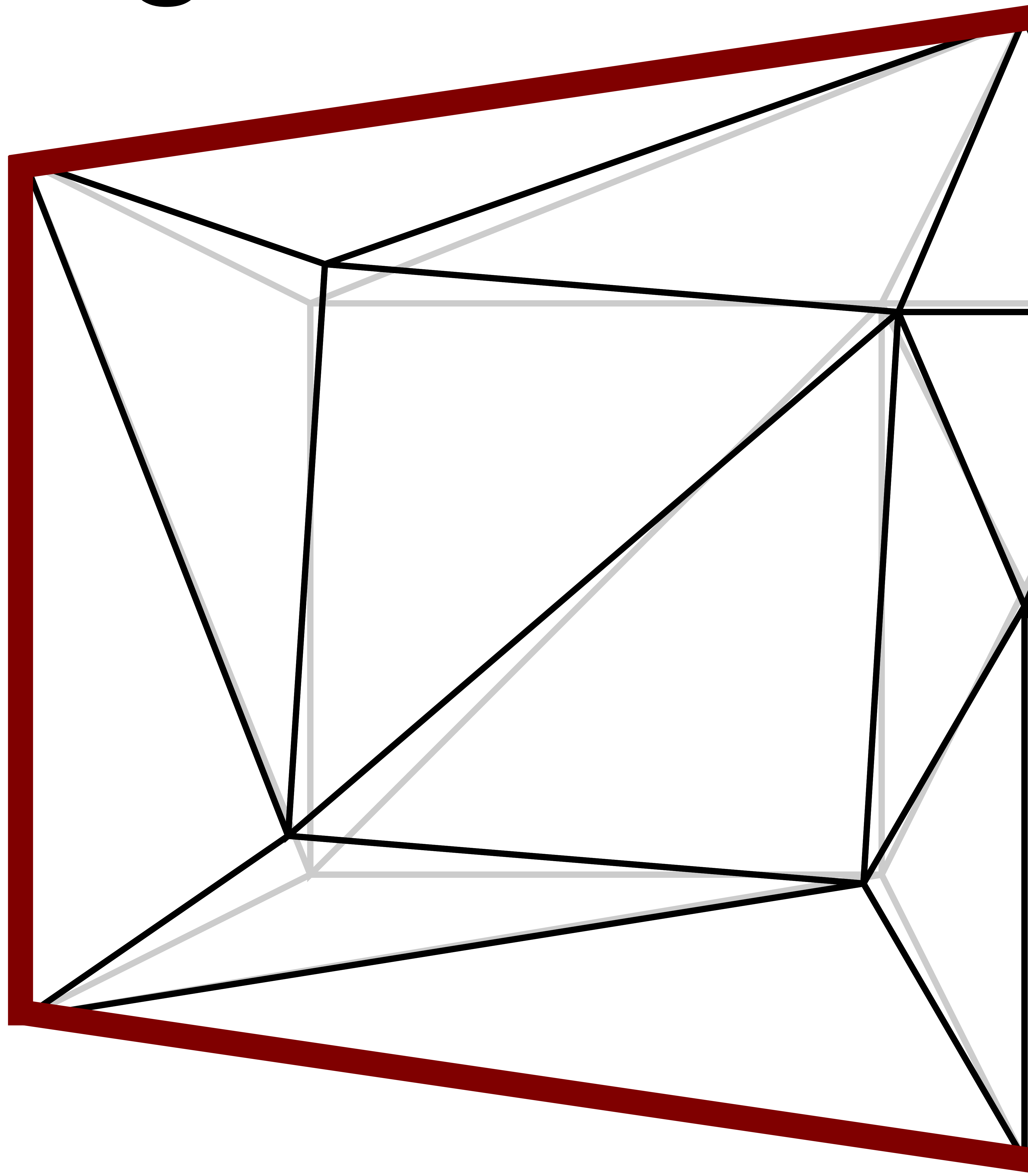
$$\omega'_{ij} = \omega_{ij} \frac{\|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



Mesh Laplacian - Algorithm

- Adjusting spring constants
- Update rule

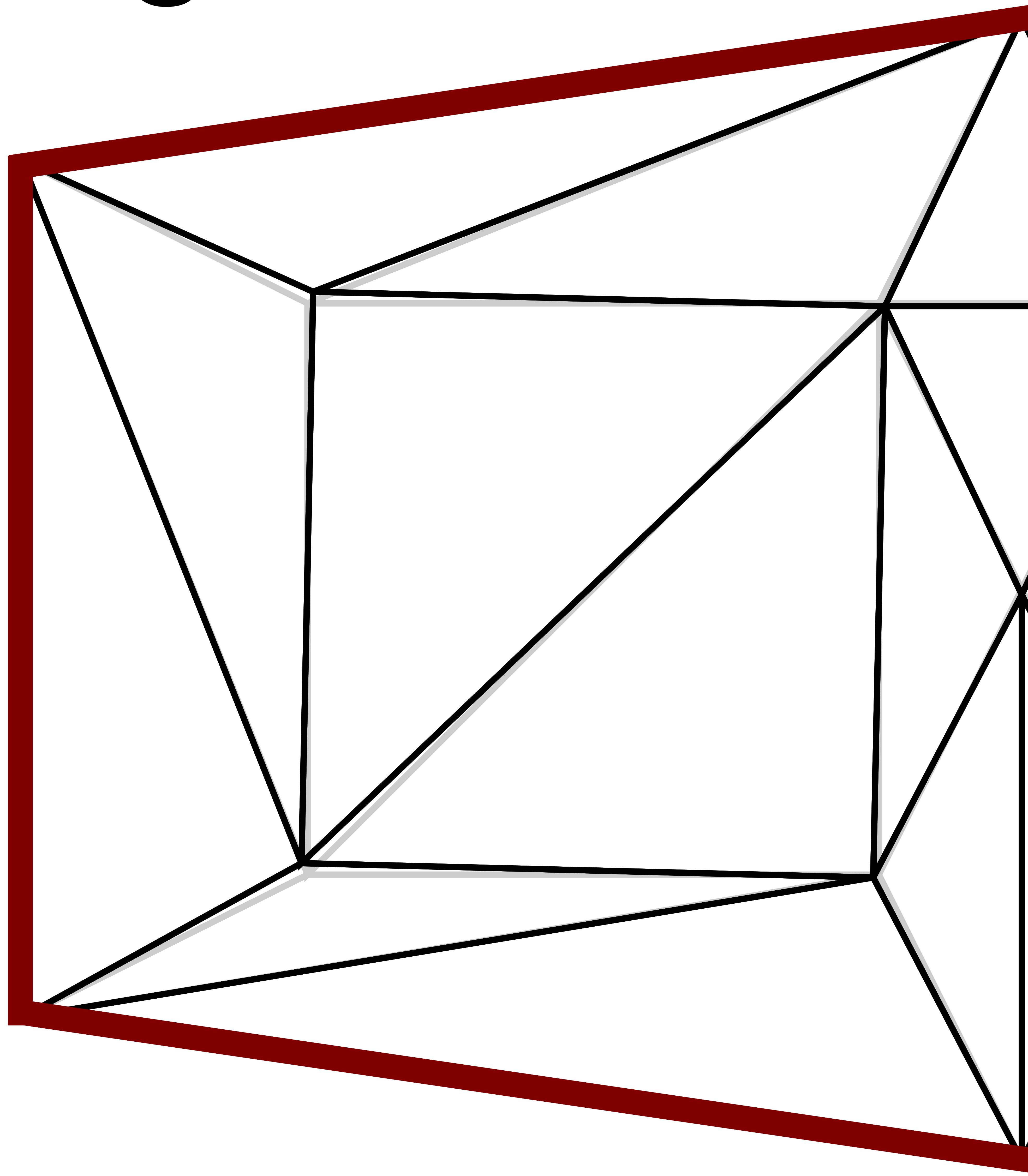
$$\omega'_{ij} = \omega_{ij} \frac{\|\mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega}\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



Mesh Laplacian - Algorithm

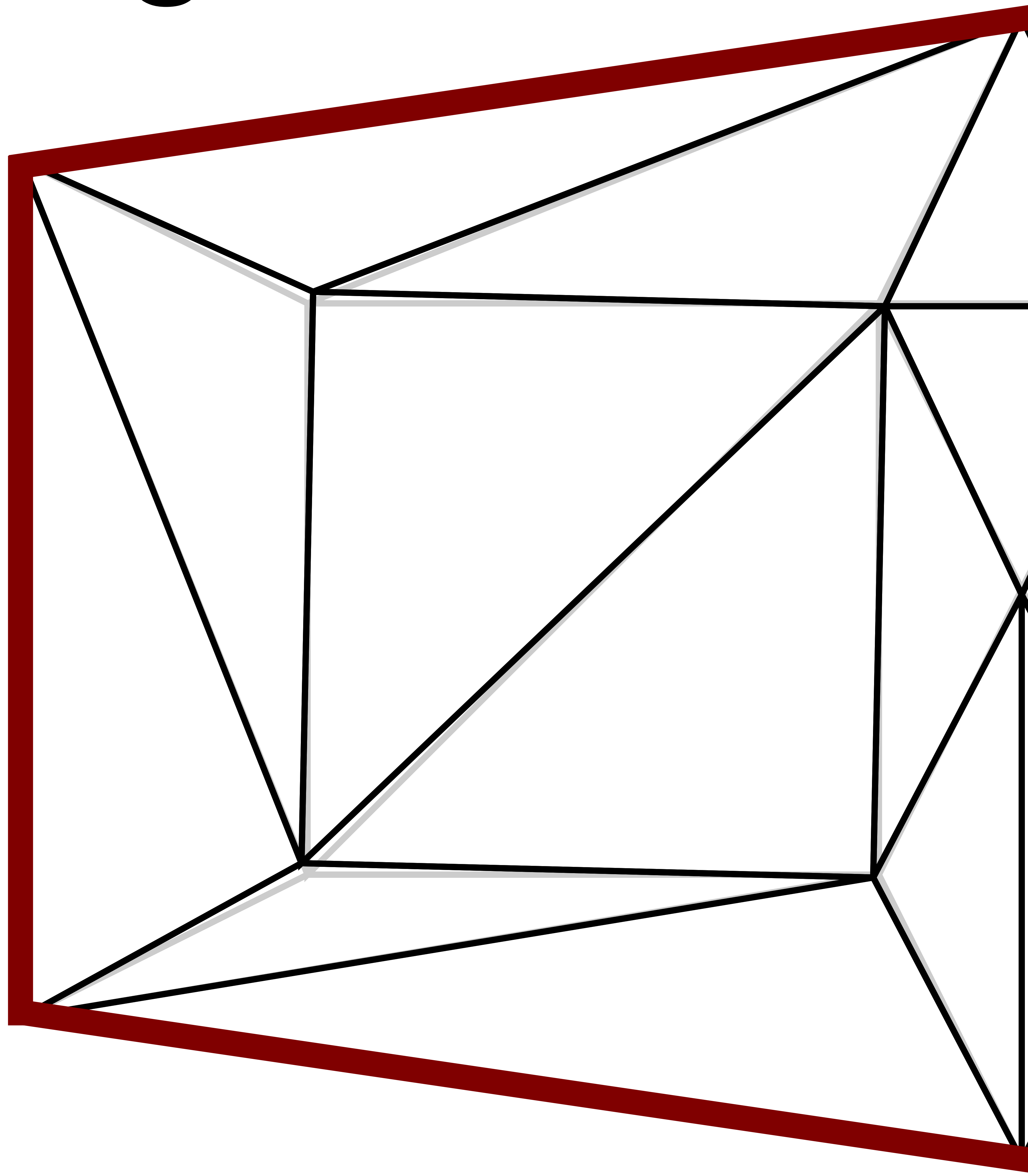
- Adjusting spring constants
- Update rule

$$\omega'_{ij} = \omega_{ij} \frac{\|\mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega}\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



Mesh Laplacian - Algorithm

- Adjusting spring constants
- Important detail: $\mathbf{L}\mathbf{V}_\Omega = 0 \Rightarrow \mu\mathbf{L}\mathbf{V}_\Omega = 0$

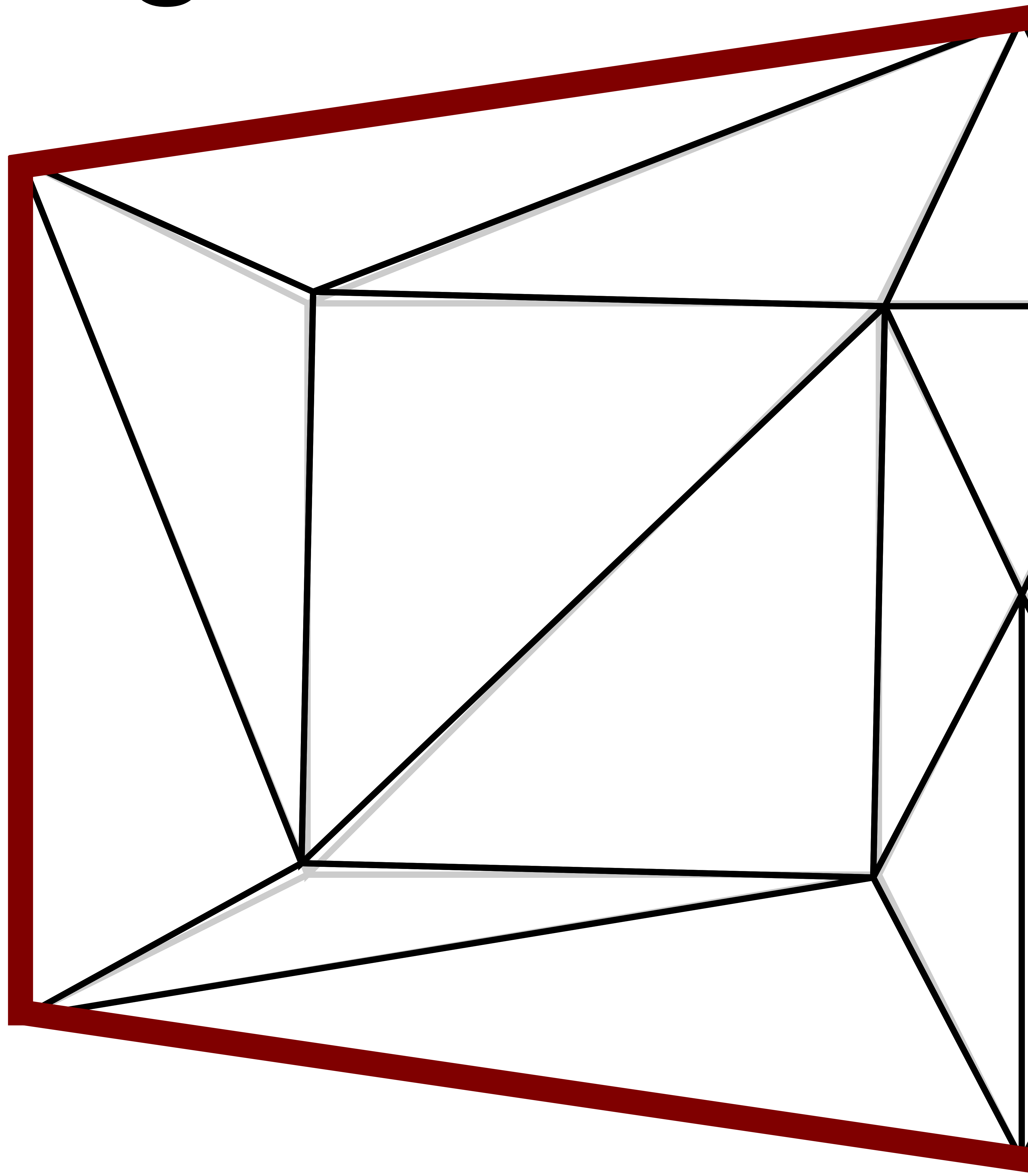


Mesh Laplacian - Algorithm

- Adjusting spring constants
- Important detail: $\mathbf{L}\mathbf{V}_\Omega = 0 \Rightarrow \mu\mathbf{L}\mathbf{V}_\Omega = 0$
- Better update rule:

$$\omega'_{ij} = \mu\ \omega_{ij} \frac{\|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$

- Choose $\mu > 0$

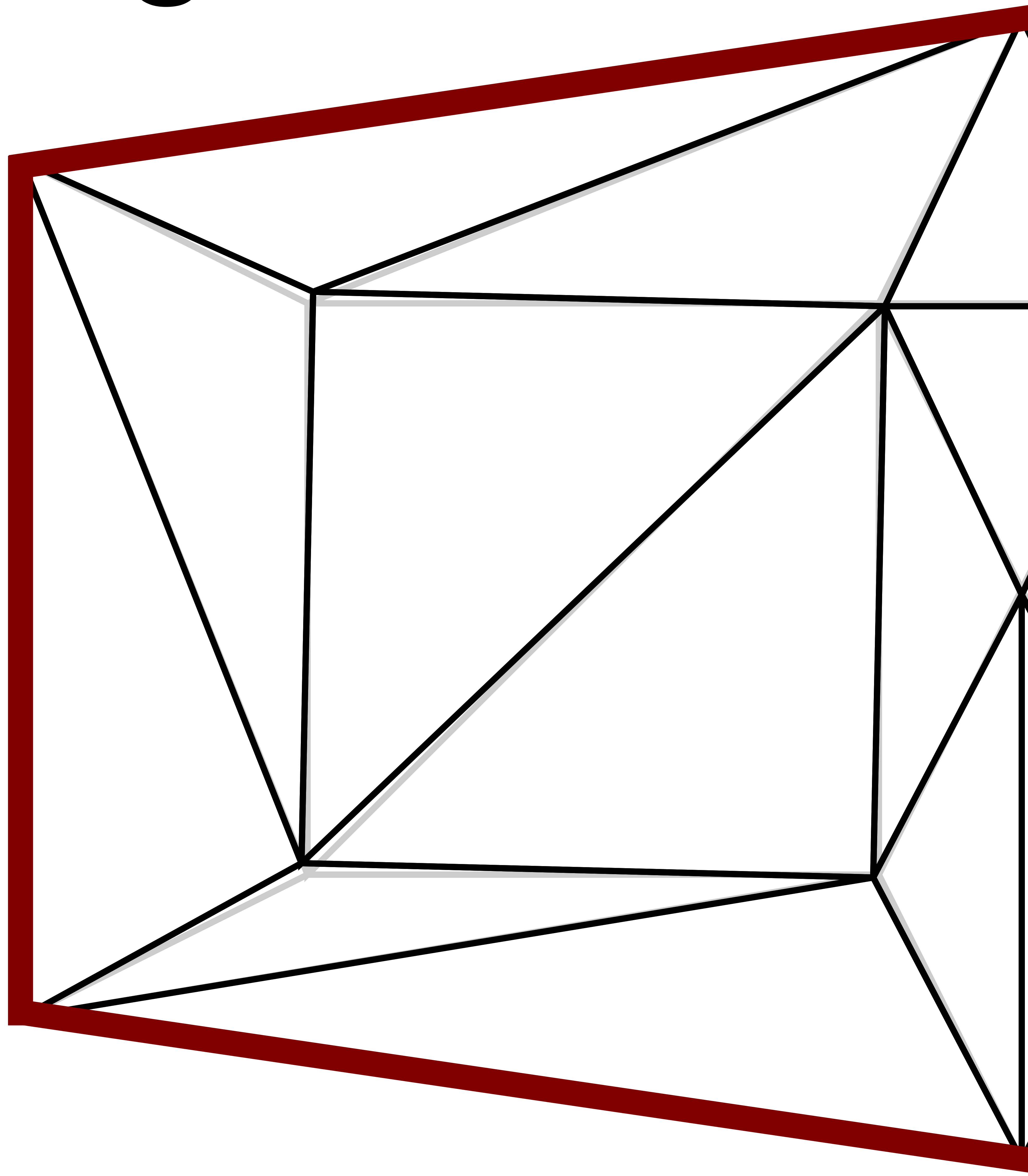


Mesh Laplacian - Algorithm

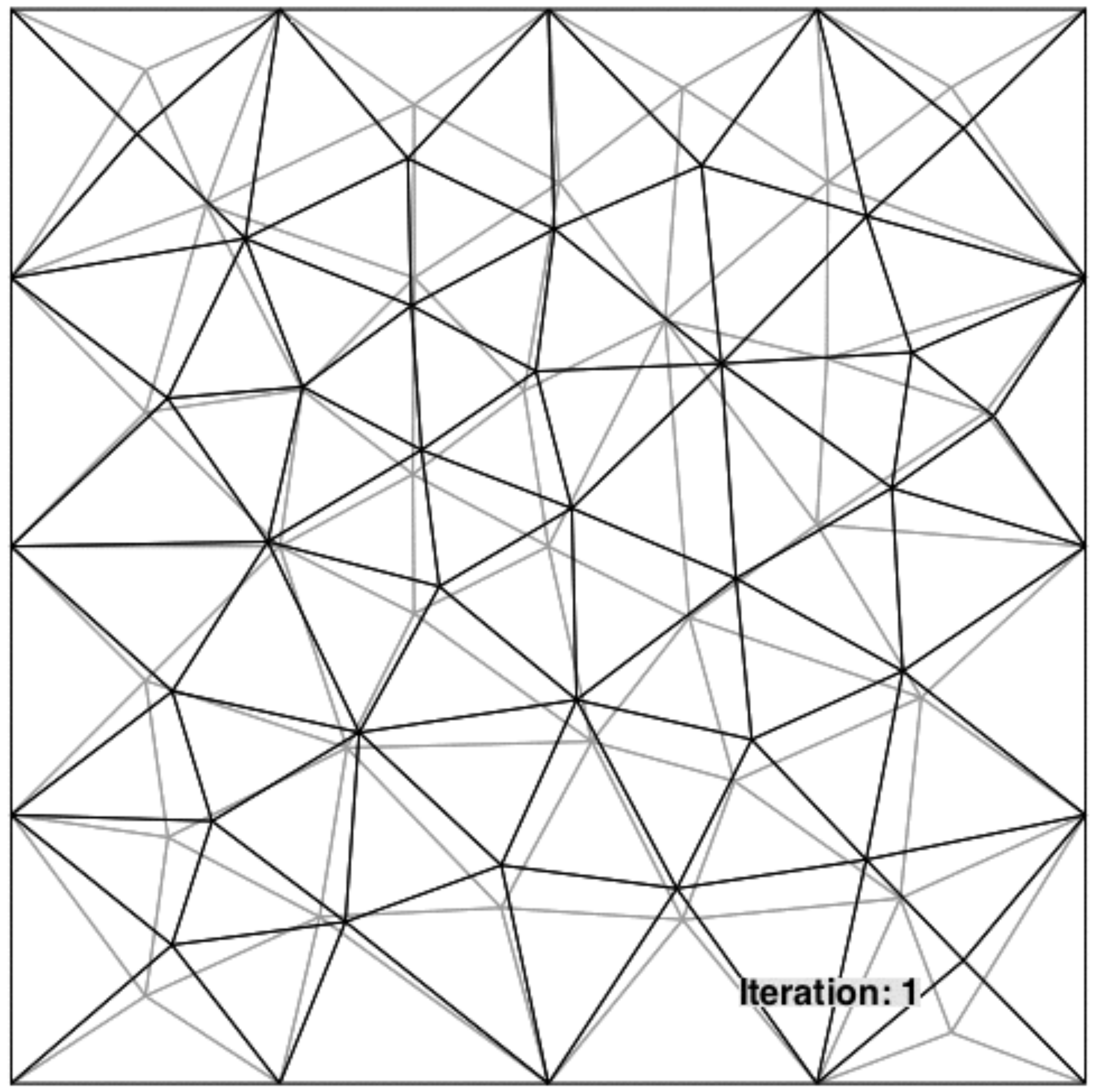
- Adjusting spring constants
- Important detail: $\mathbf{L}\mathbf{V}_\Omega = 0 \Rightarrow \mu\mathbf{L}\mathbf{V}_\Omega = 0$
- Better update rule:

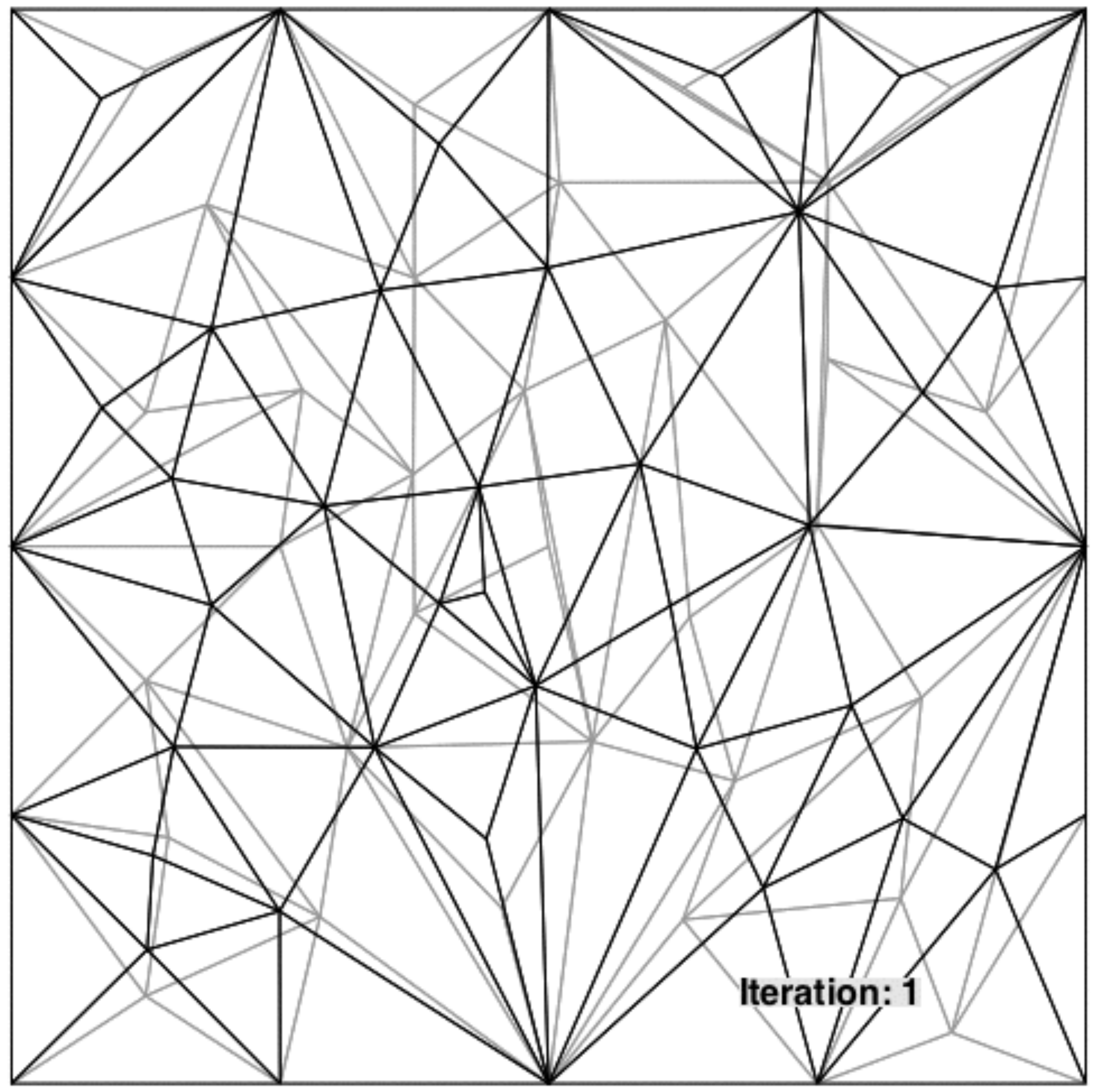
$$\omega'_{ij} = \mu\ \omega_{ij} \frac{\|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$

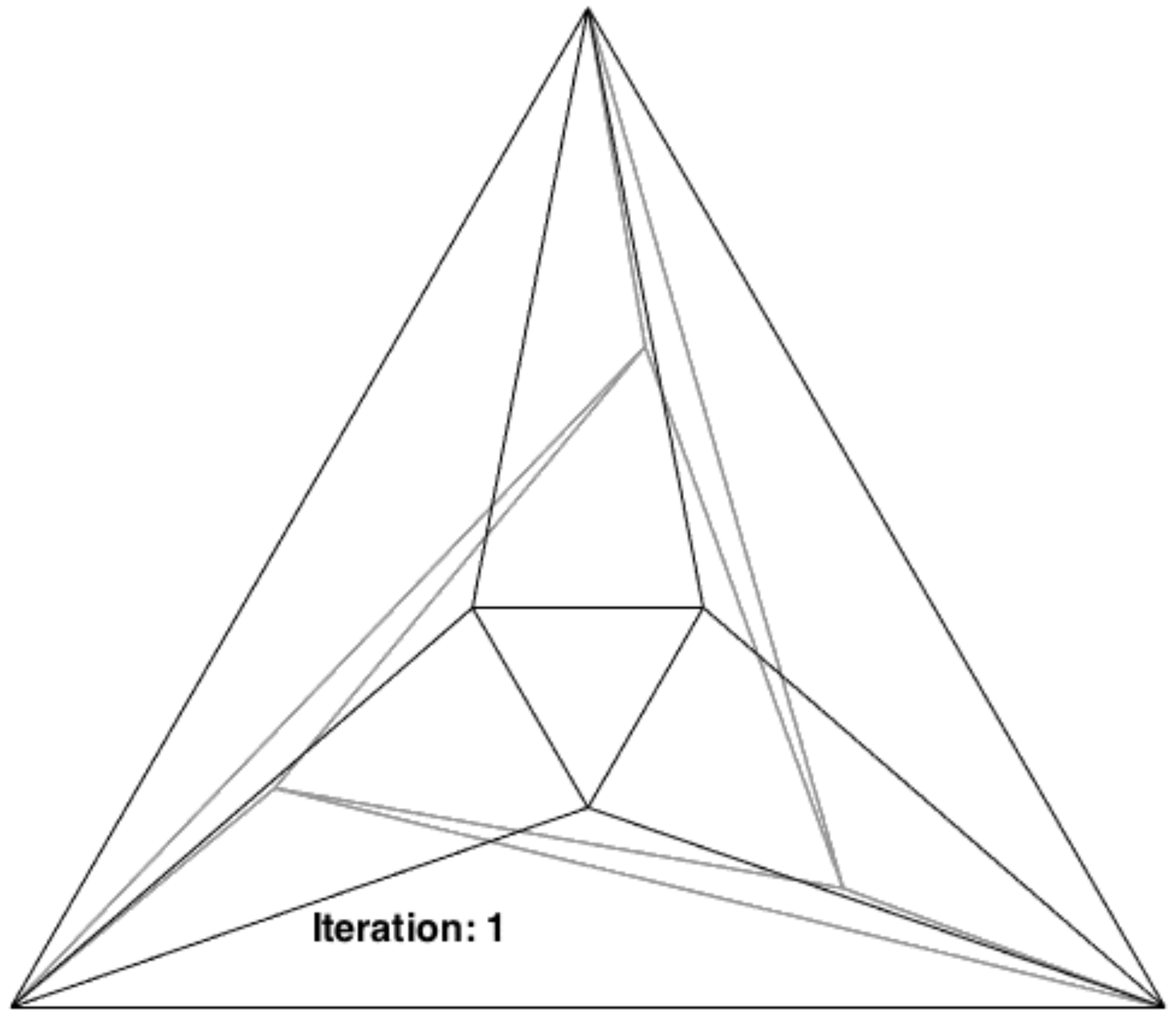
- Set $\mu > 0$ s.t. $\sum_{(i,j)\in E} \omega'_{ij} = 1$



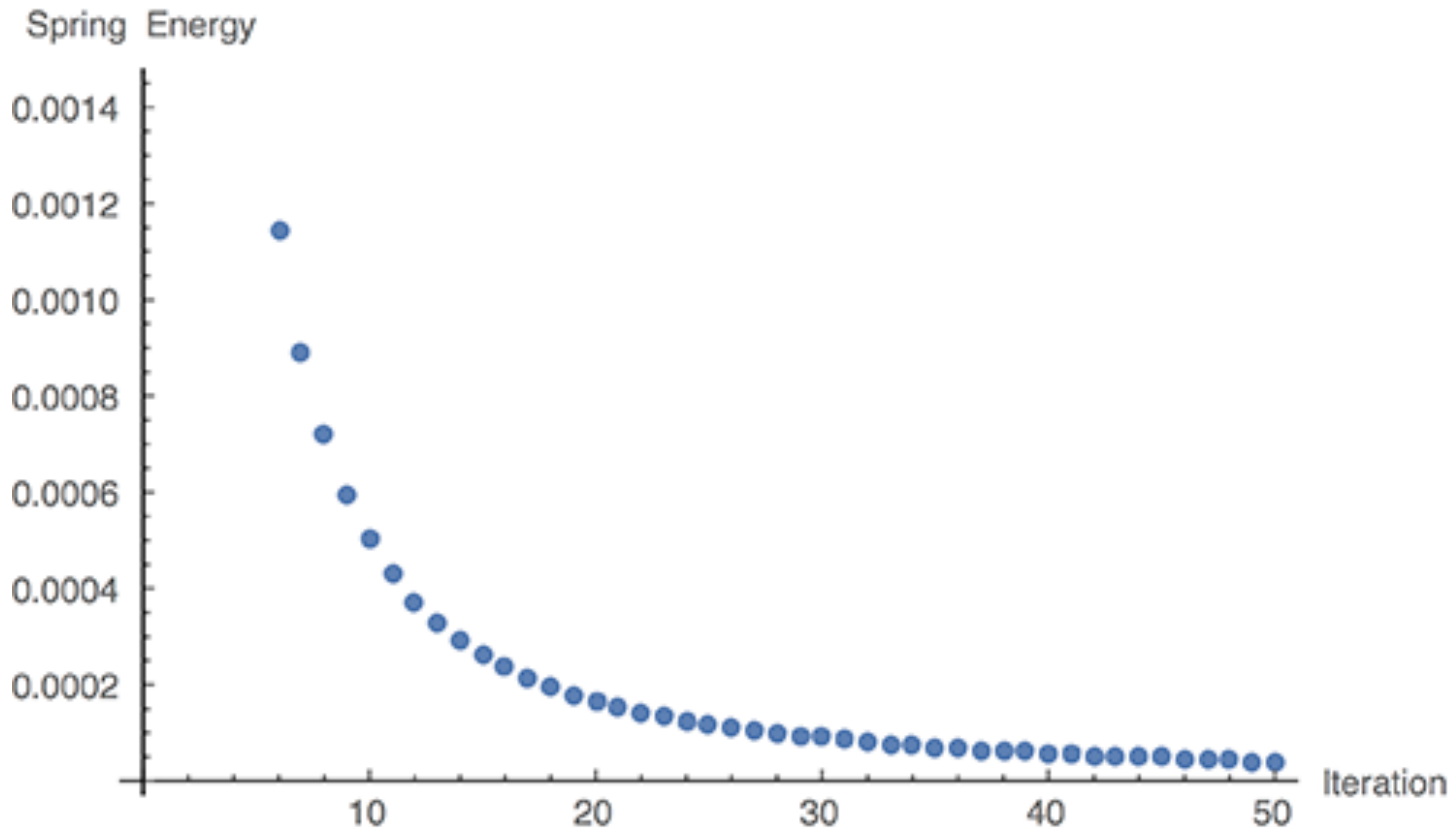
Properties of algorithm: convergence?





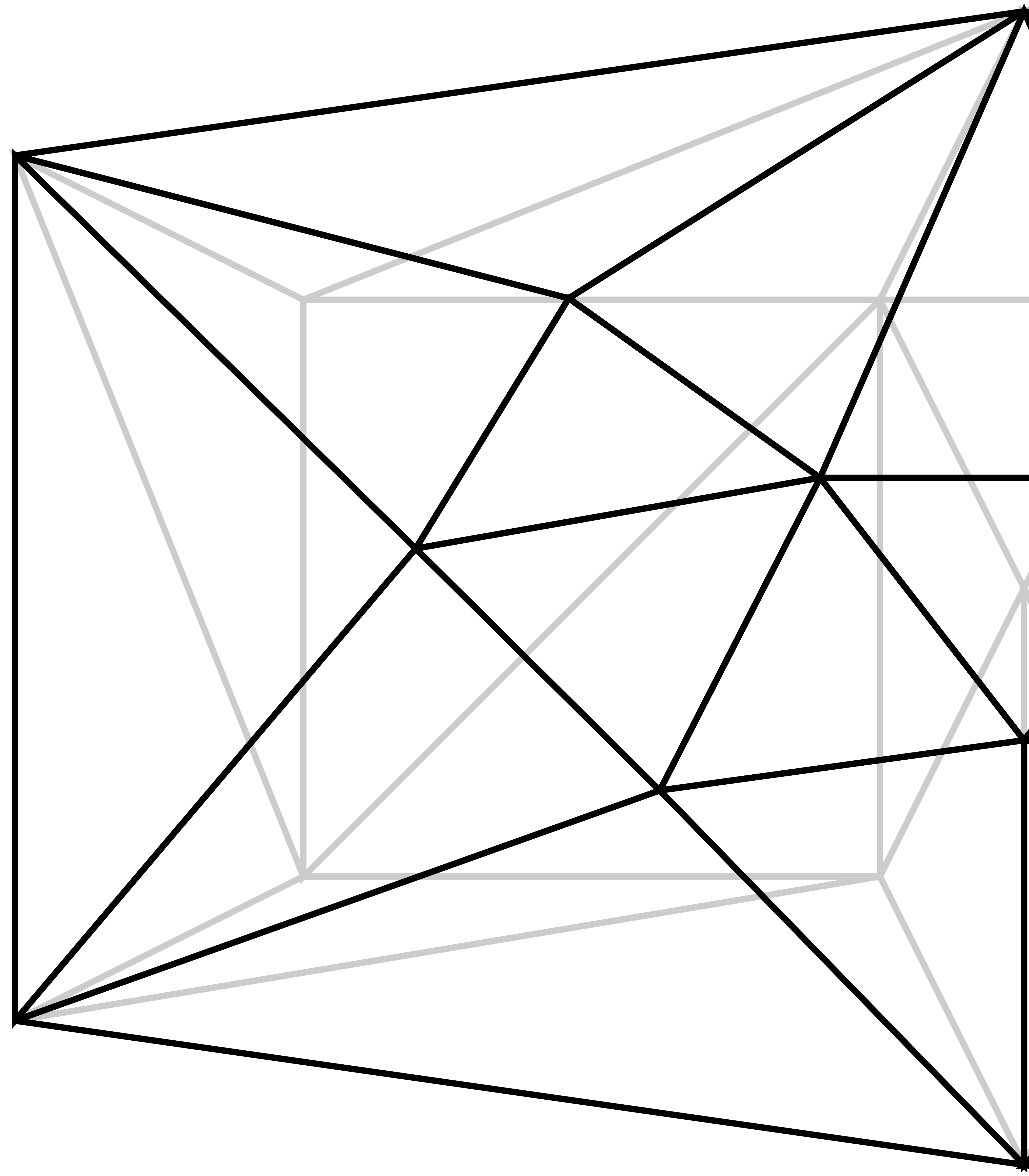


Properties of algorithm: convergence?



Properties of algorithm: non-negativity!

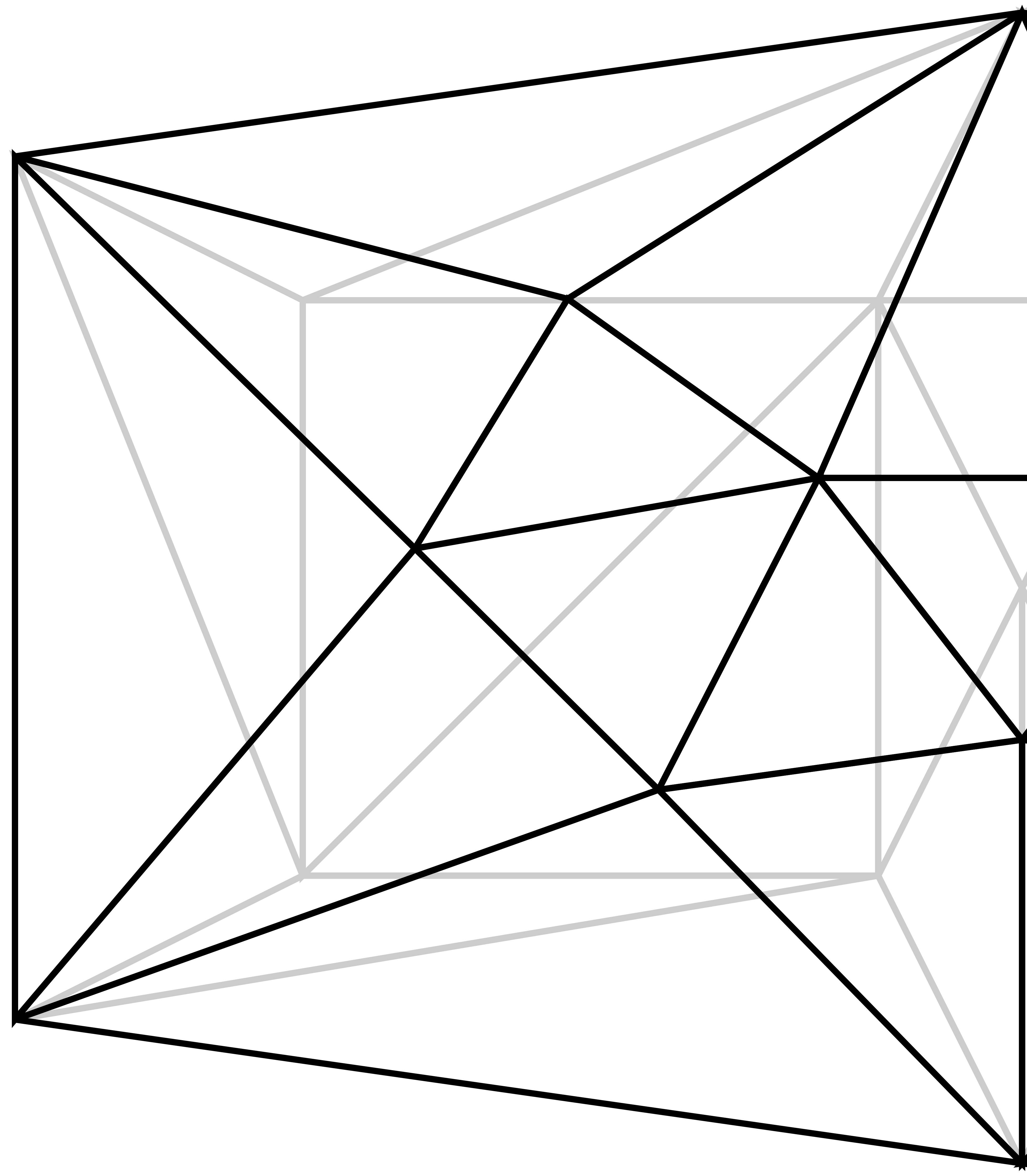
$$\omega'_{ij} \quad = \mu \omega_{ij} \frac{\|\mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega}\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



Properties of algorithm: non-negativity!

$$\omega'_{ij} = \mu \omega_{ij} \frac{\|\mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega}\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$

$$\omega_{ij} > 0$$

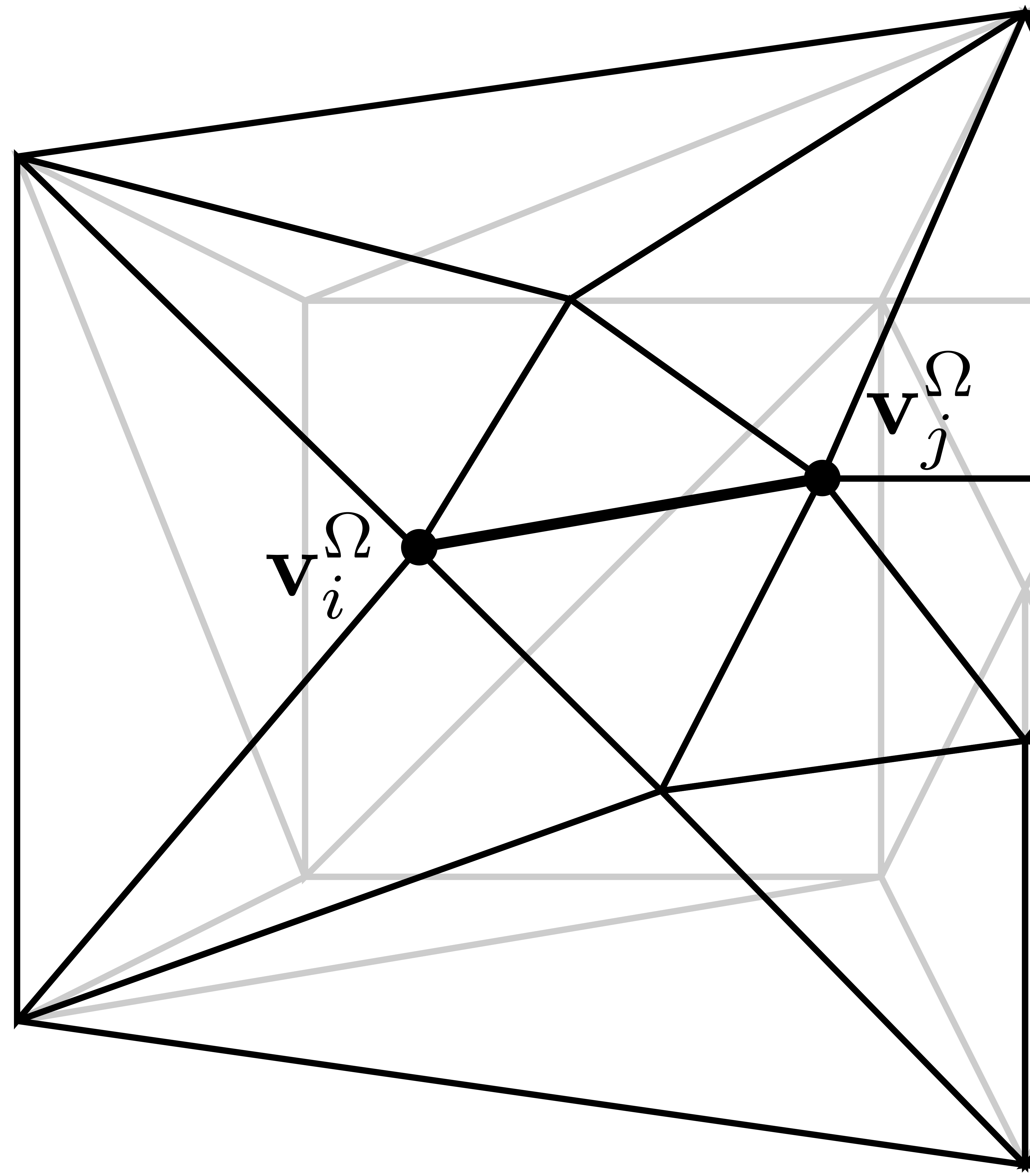


Properties of algorithm: non-negativity!

Tutte embedding

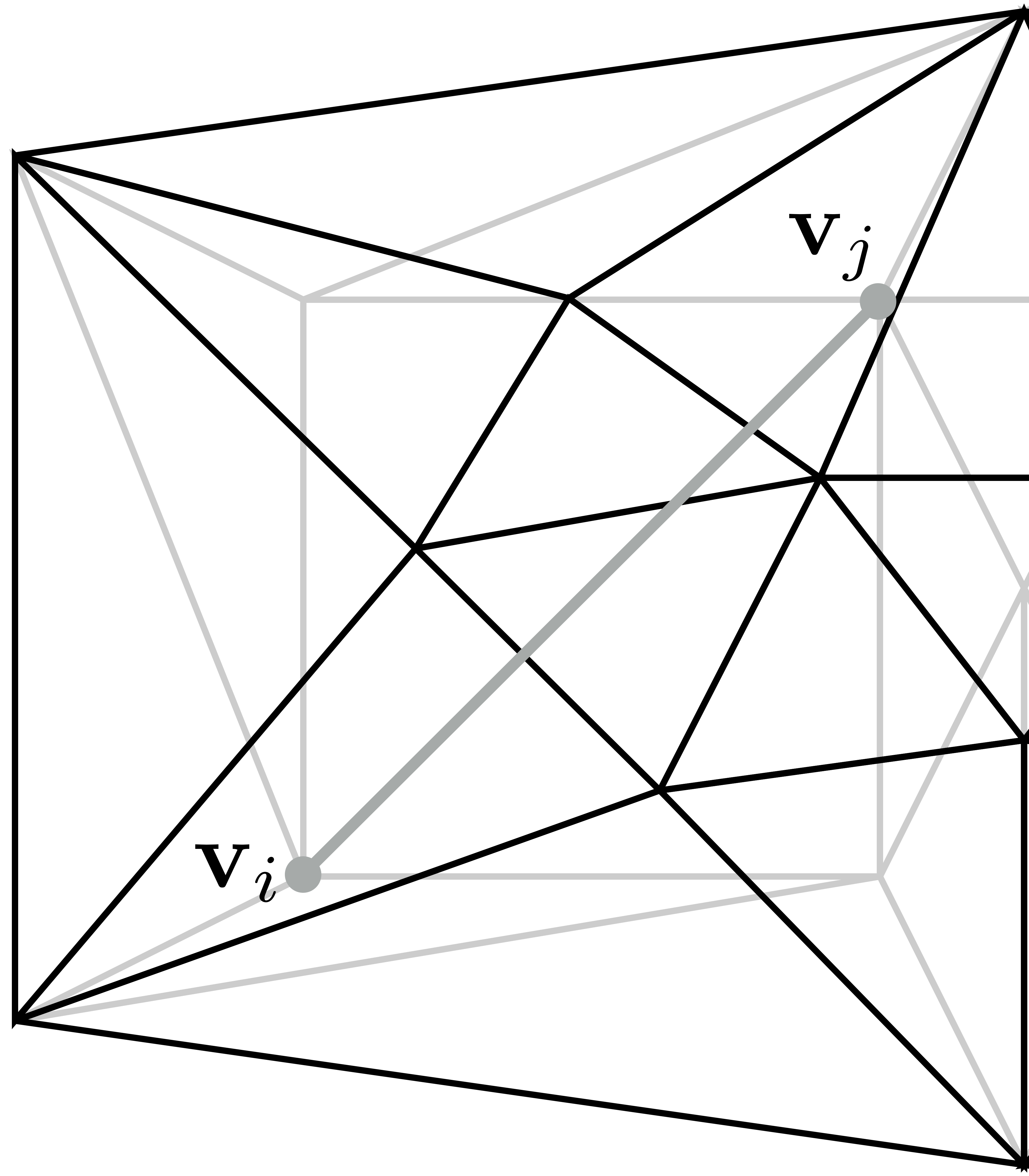
$$\|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\| > 0$$

$$\omega'_{ij} = \mu\omega_{ij}\frac{\|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



Properties of algorithm: non-negativity!

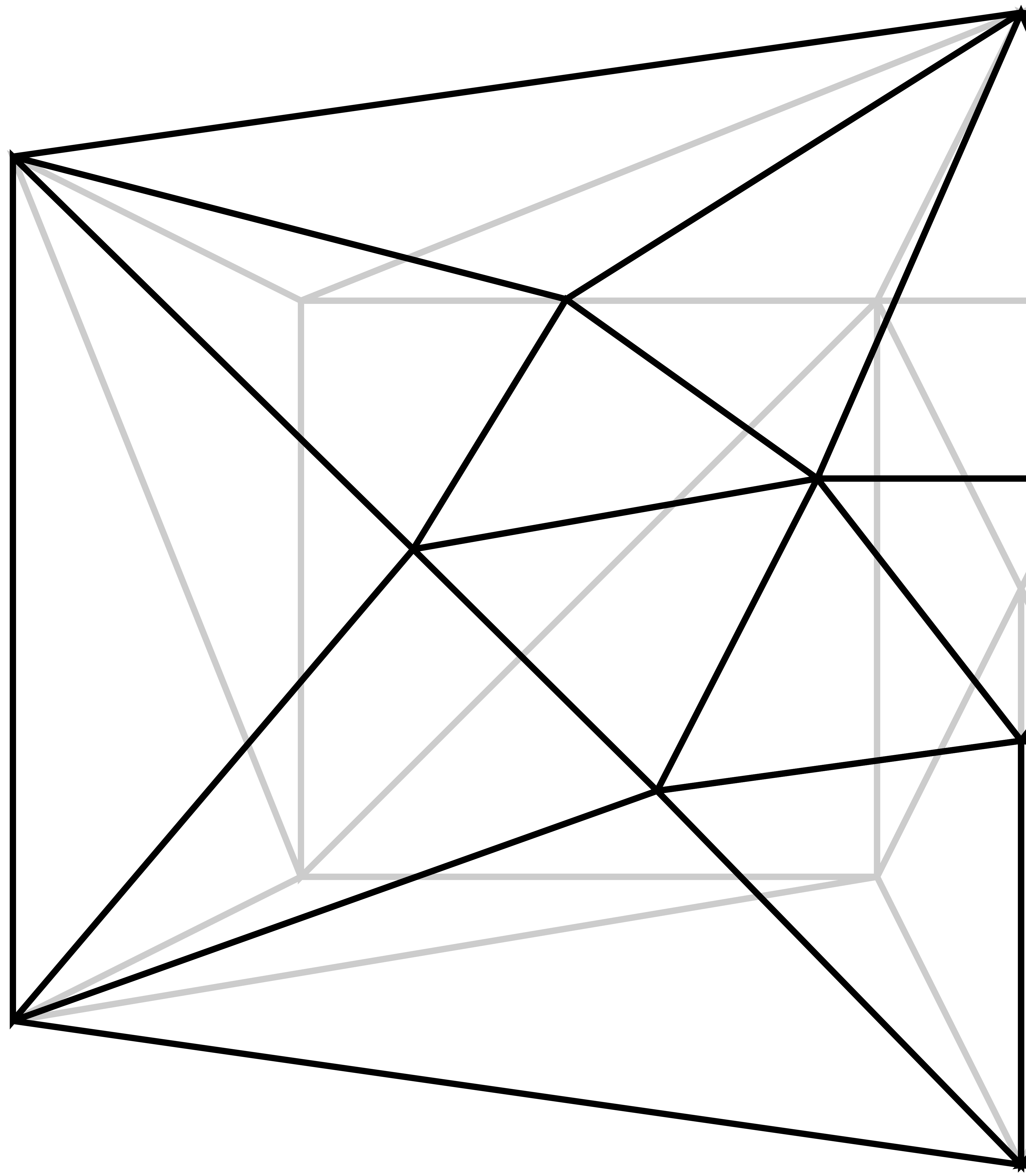
$$\omega'_{ij} = \mu\omega_{ij}\frac{\|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\|}{\|\mathbf{v}_j - \mathbf{v}_i\|} > 0$$



Properties of algorithm: non-negativity!

$$\omega'_{ij} = \mu \omega_{ij} \frac{\|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$

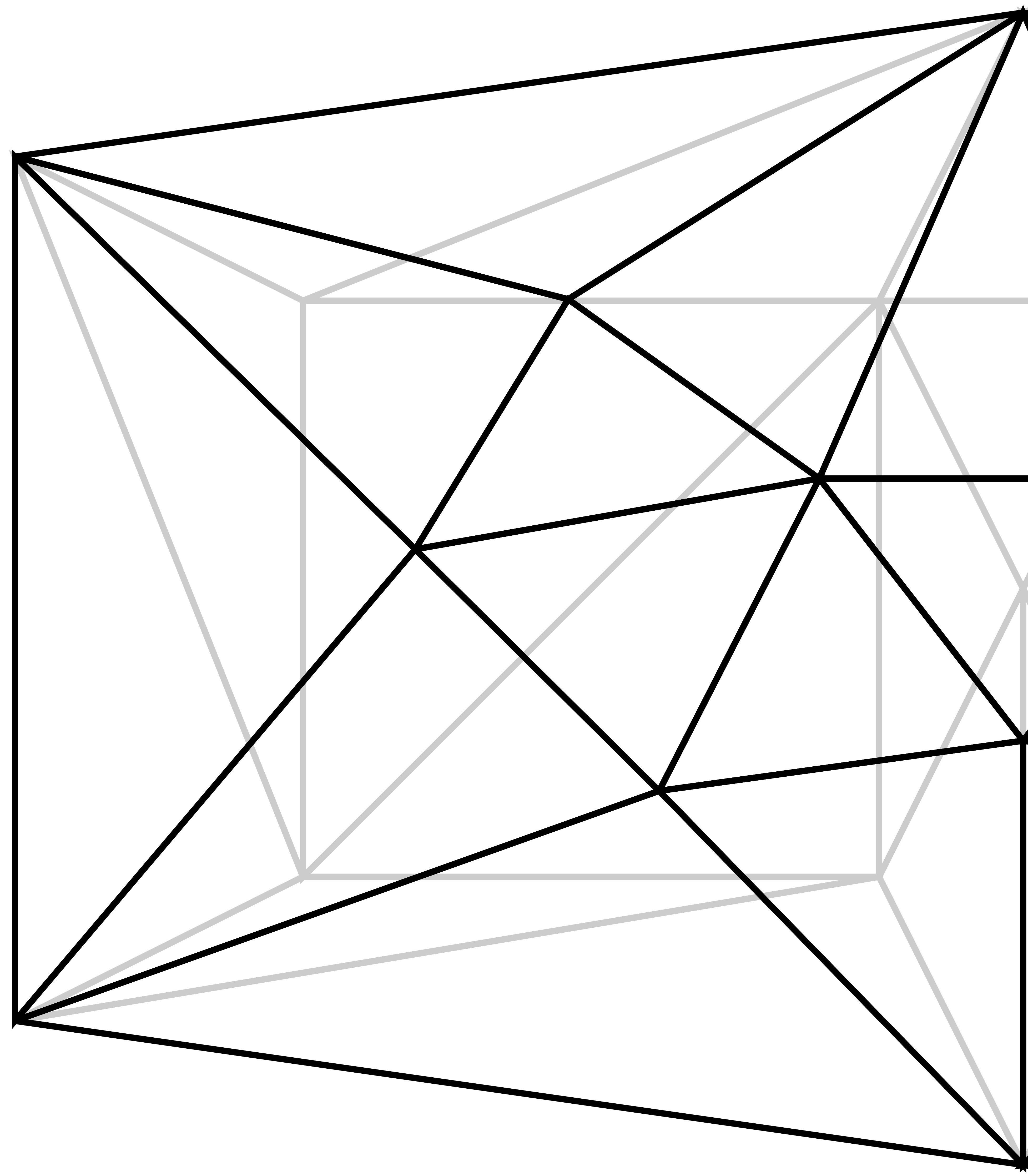
$$\mu > 0$$



Properties of algorithm: non-negativity!

$$\omega'_{ij} > 0 \Leftarrow$$

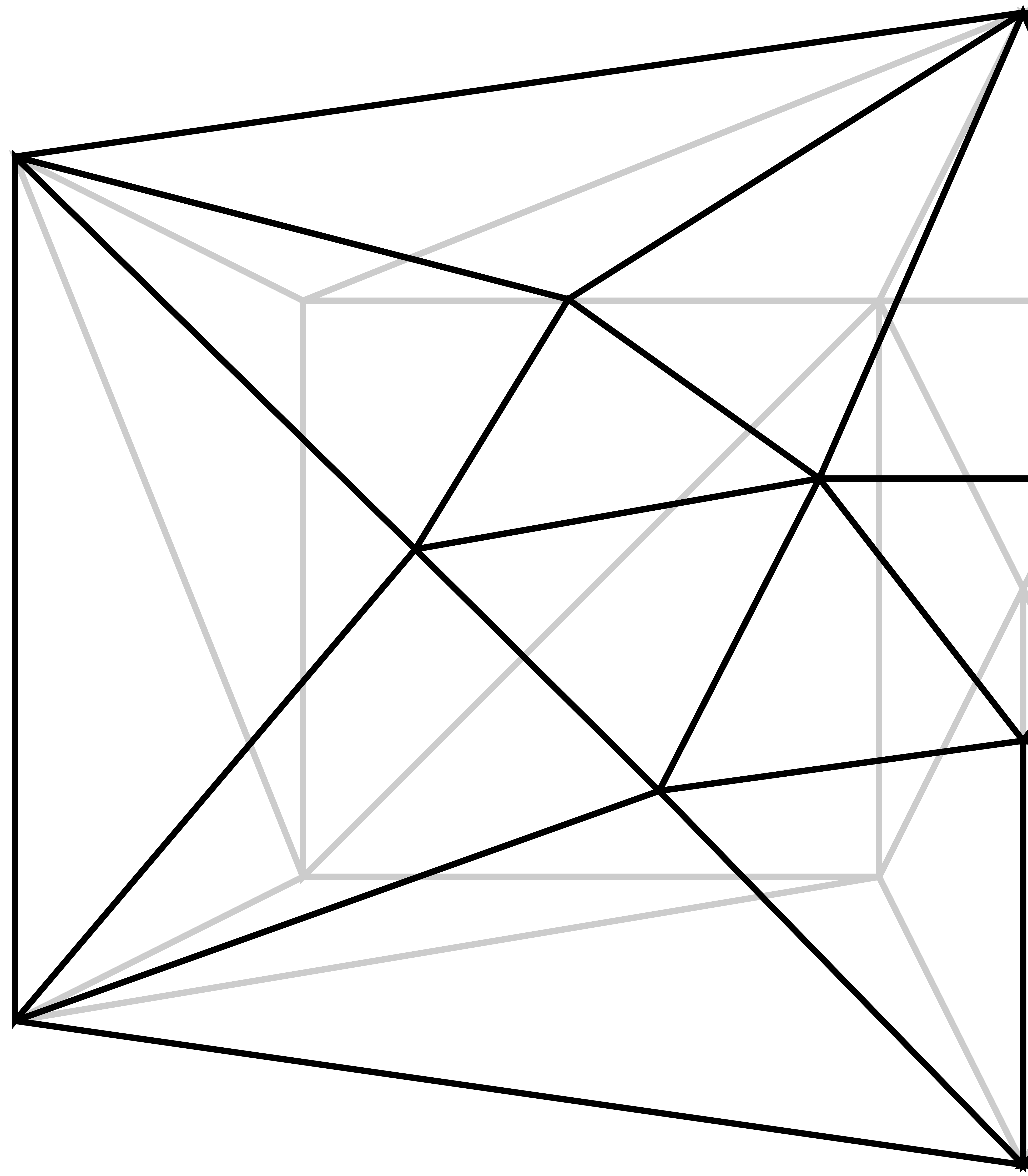
$$= \mu\omega_{ij}\frac{\|\mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega}\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



Properties of algorithm: non-negativity!

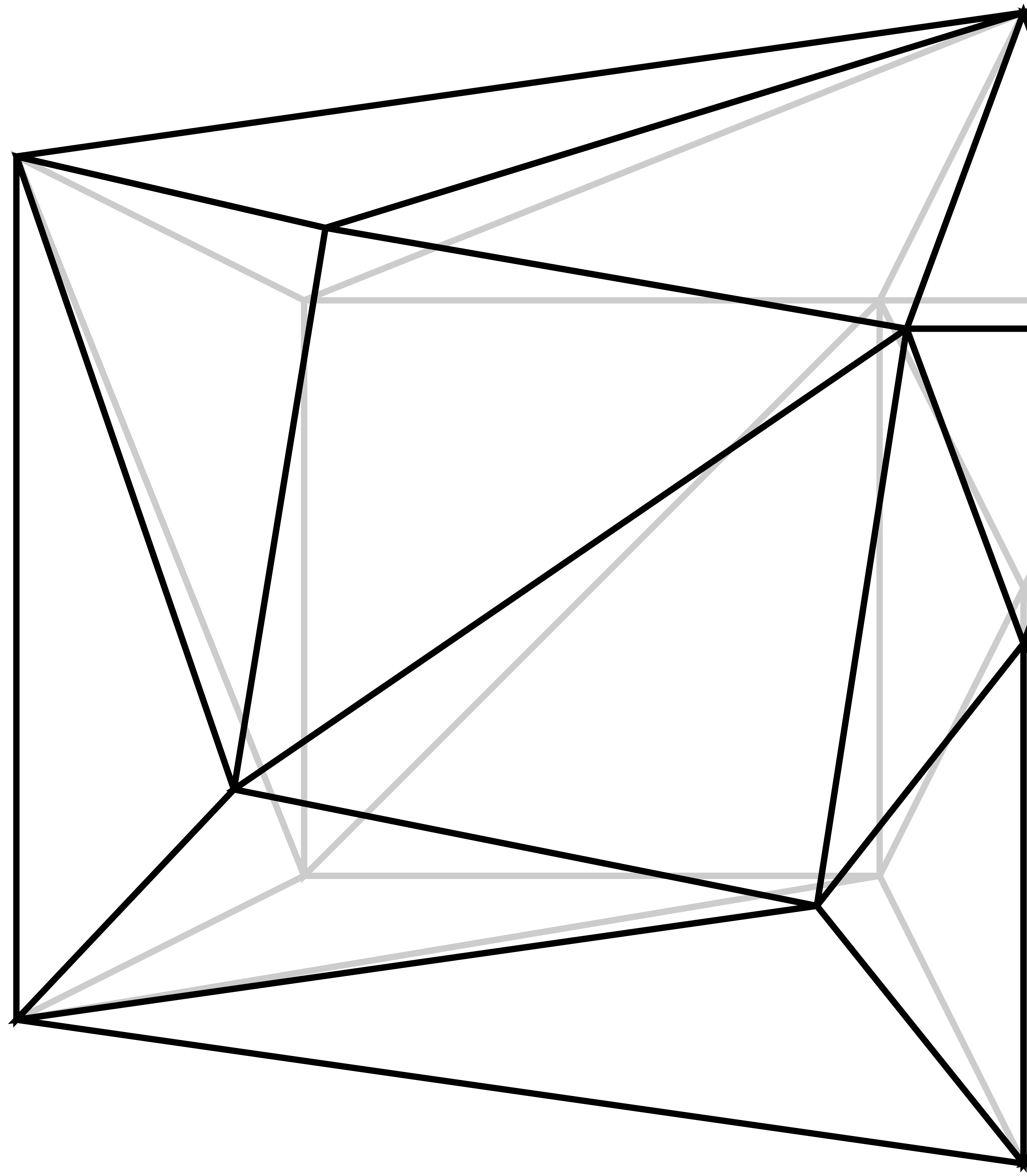
$$\omega'_{ij} > 0$$

$$= \mu\omega_{ij}\frac{\|\mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega}\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



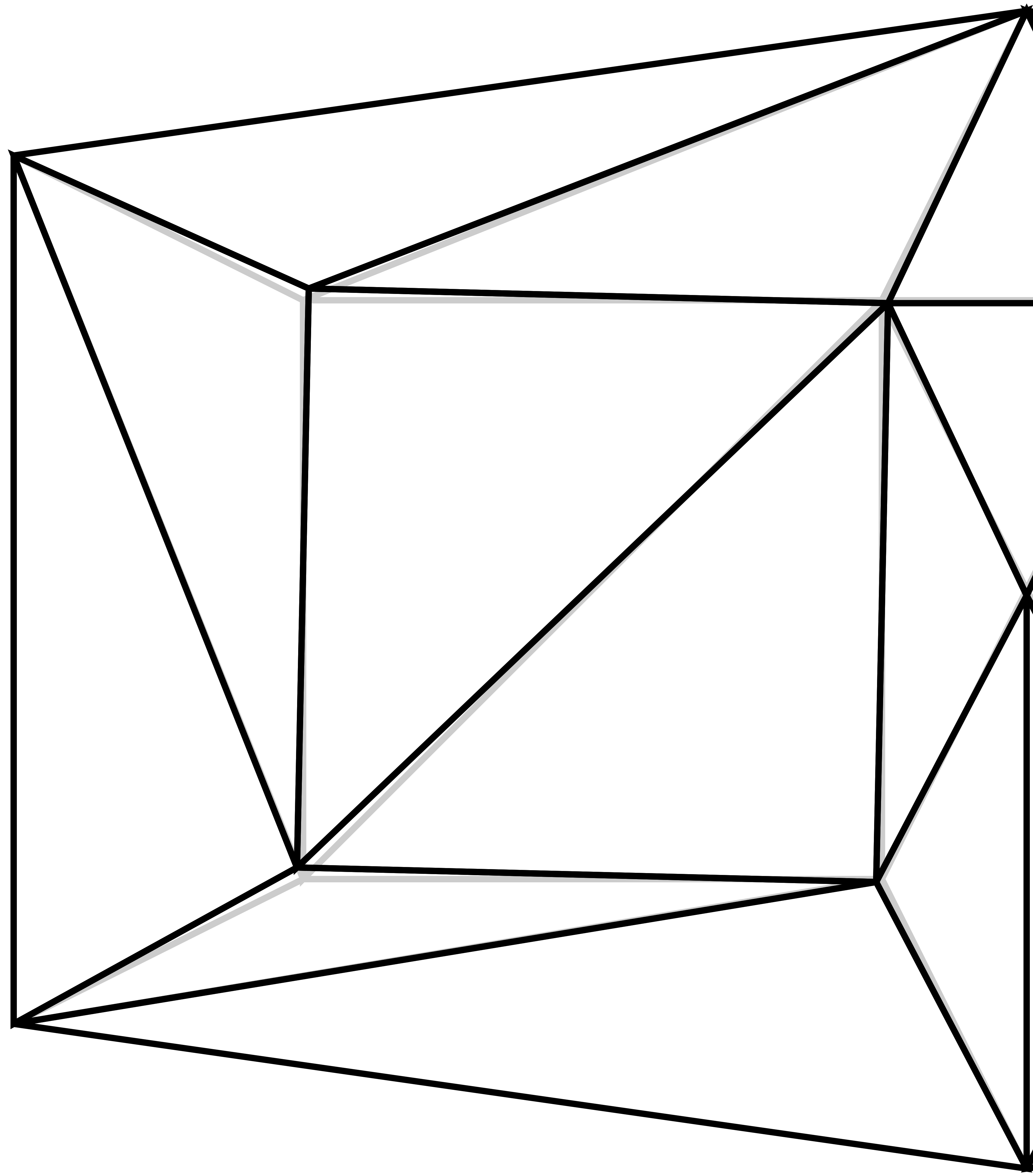
Properties of algorithm: non-negativity!

$$\omega'_{ij} > 0 = \mu\omega_{ij}\frac{\|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



Properties of algorithm: non-negativity!

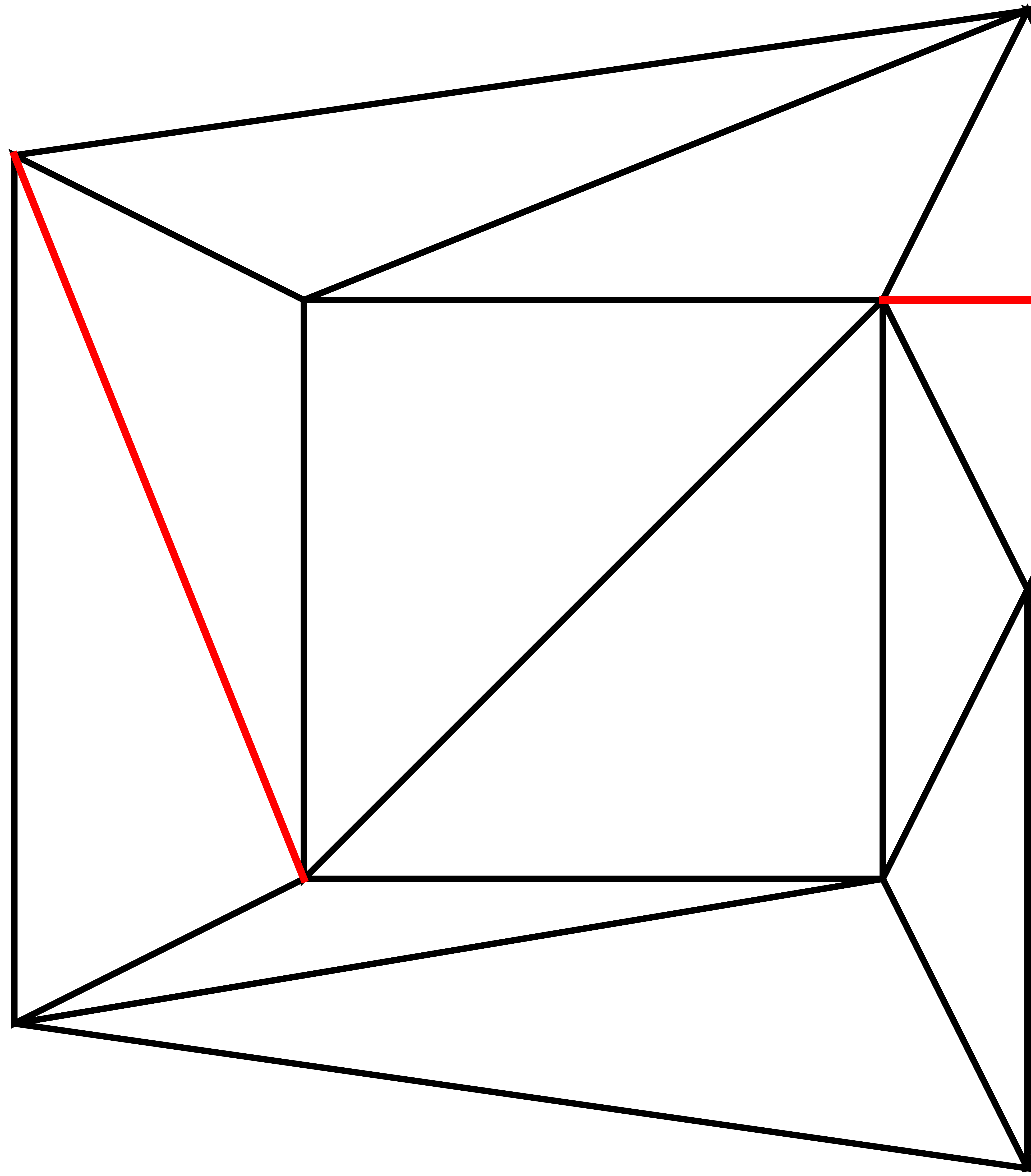
$$\omega'_{ij} > 0 = \mu\omega_{ij}\frac{\|\mathbf{v}_j^\Omega - \mathbf{v}_i^\Omega\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



Properties of algorithm: non-negativity!

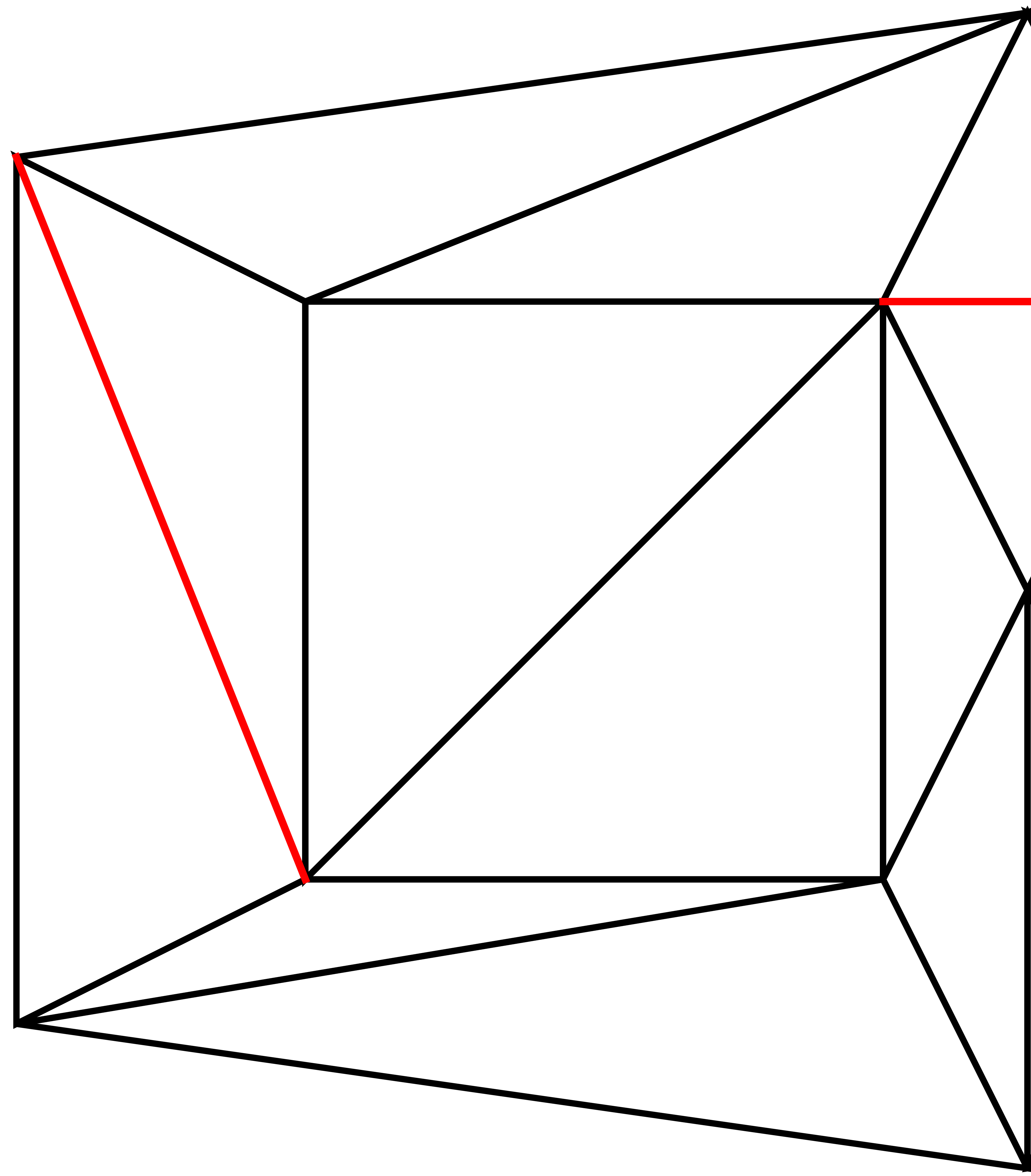
$$\omega_{ij}^{\infty} > 0$$

$$\omega_{ij}^{\infty} = 0$$



Properties of algorithm: steady state

$$\omega_{ij} = \mu \; \omega_{ij} \frac{\|\mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega}\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$



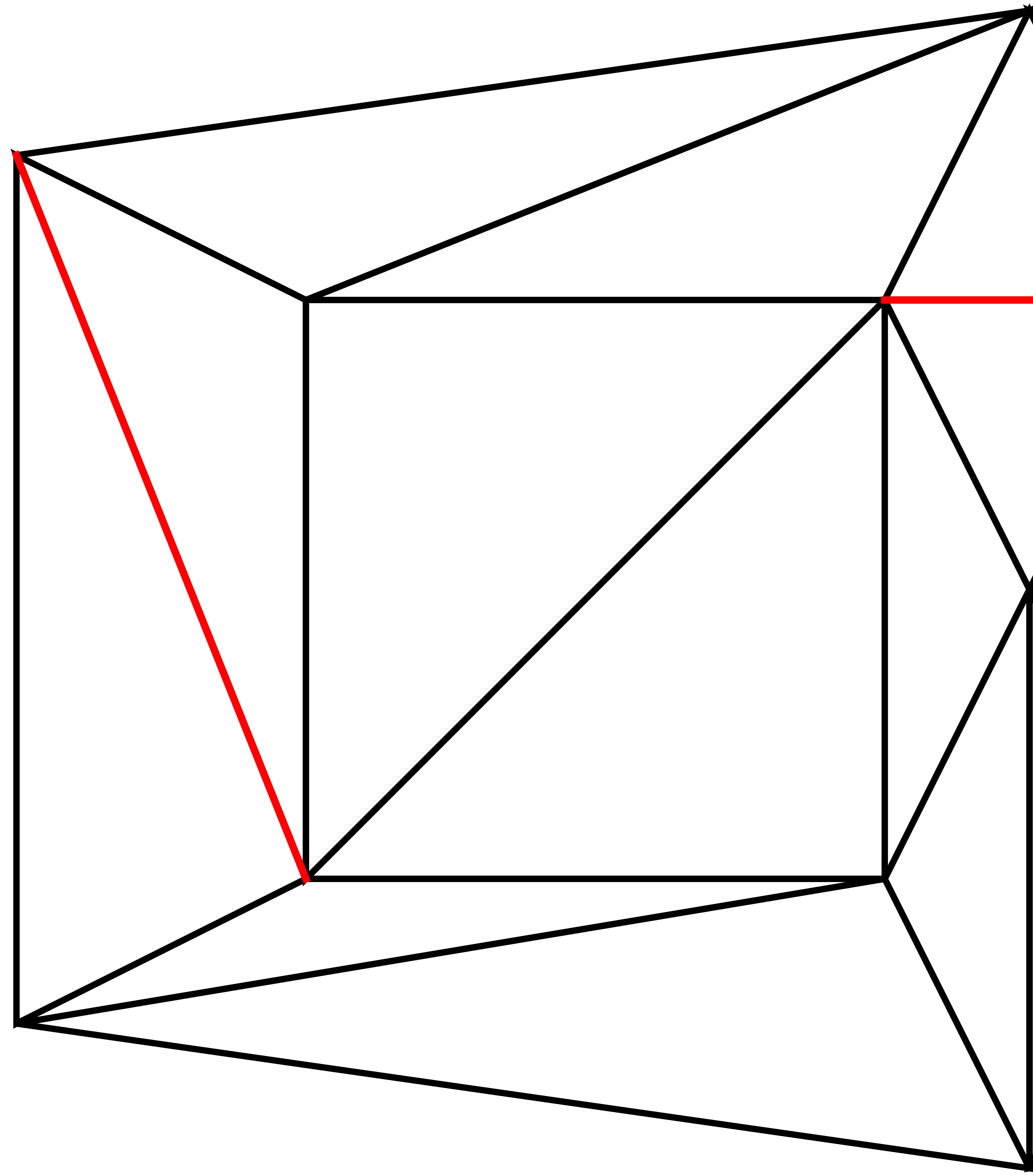
Properties of algorithm: steady state

$$\omega_{ij} = \mu \; \omega_{ij} \frac{\|\mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega}\|}{\|\mathbf{v}_j - \mathbf{v}_i\|}$$

$$\omega_{ij} > 0$$

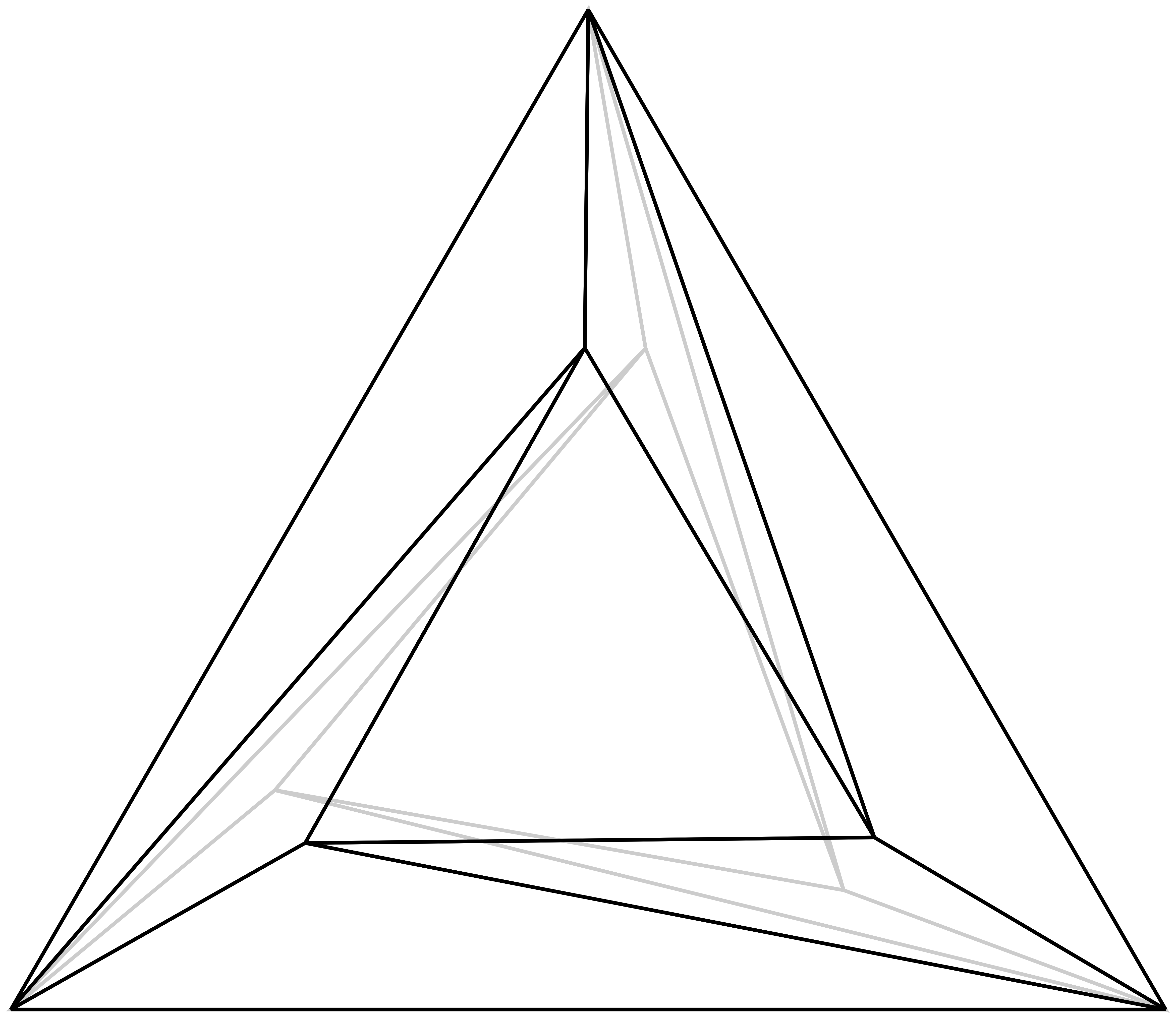
$$\|\mathbf{v}_j^{\Omega} - \mathbf{v}_i^{\Omega}\| = \mu^{-1} \|\mathbf{v}_j - \mathbf{v}_i\|$$

constant factor for all edges



Properties of algorithm: steady state

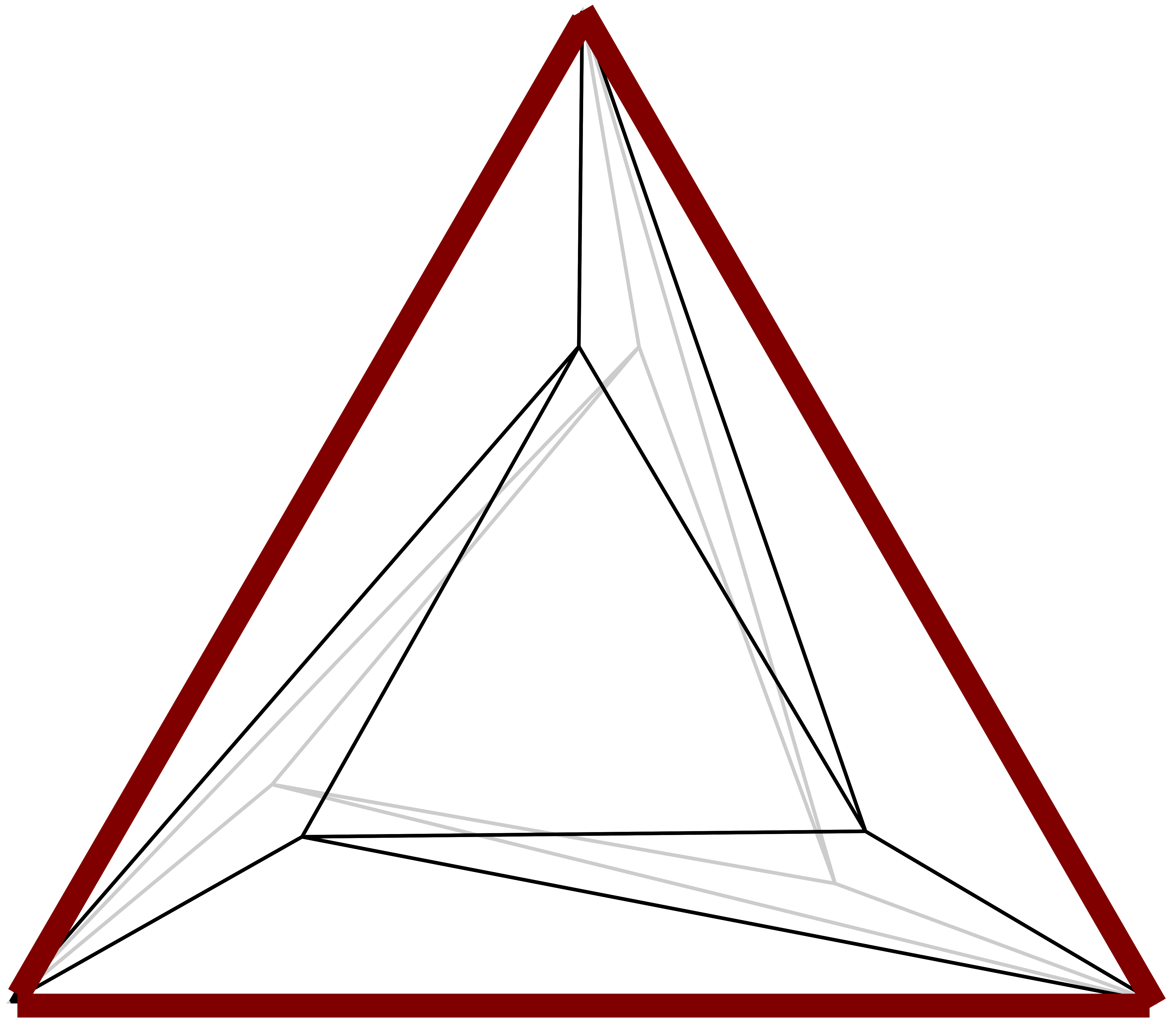
- Three types of edges



Properties of algorithm: steady state

- Three types of edges

1. Boundary

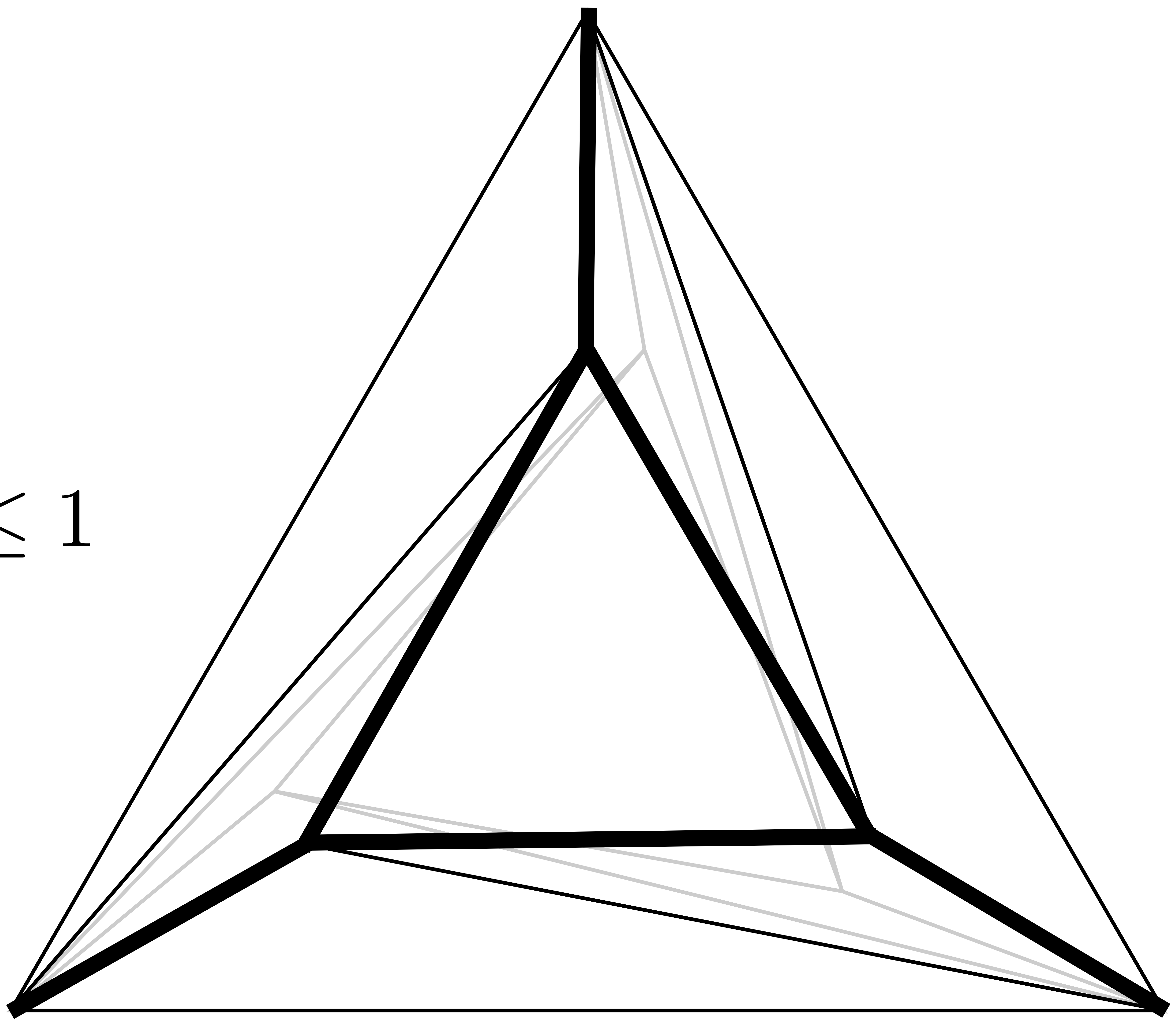


Properties of algorithm: steady state

- Three types of edges

1. Boundary

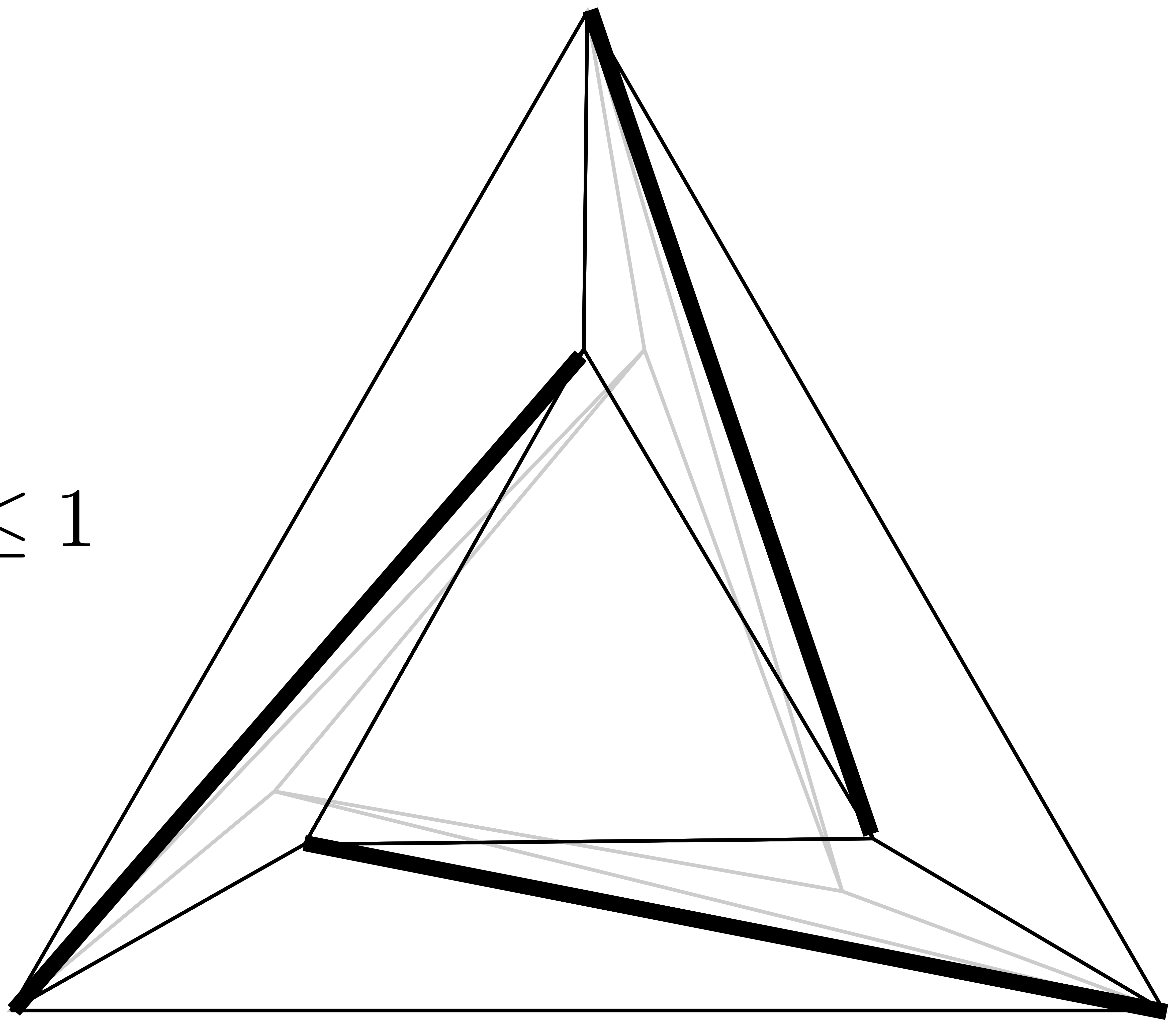
2. Scaled by a constant factor $\mu^{-1} \leq 1$
   Positive coefficient $\omega_{ij} > 0$



Properties of algorithm: steady state

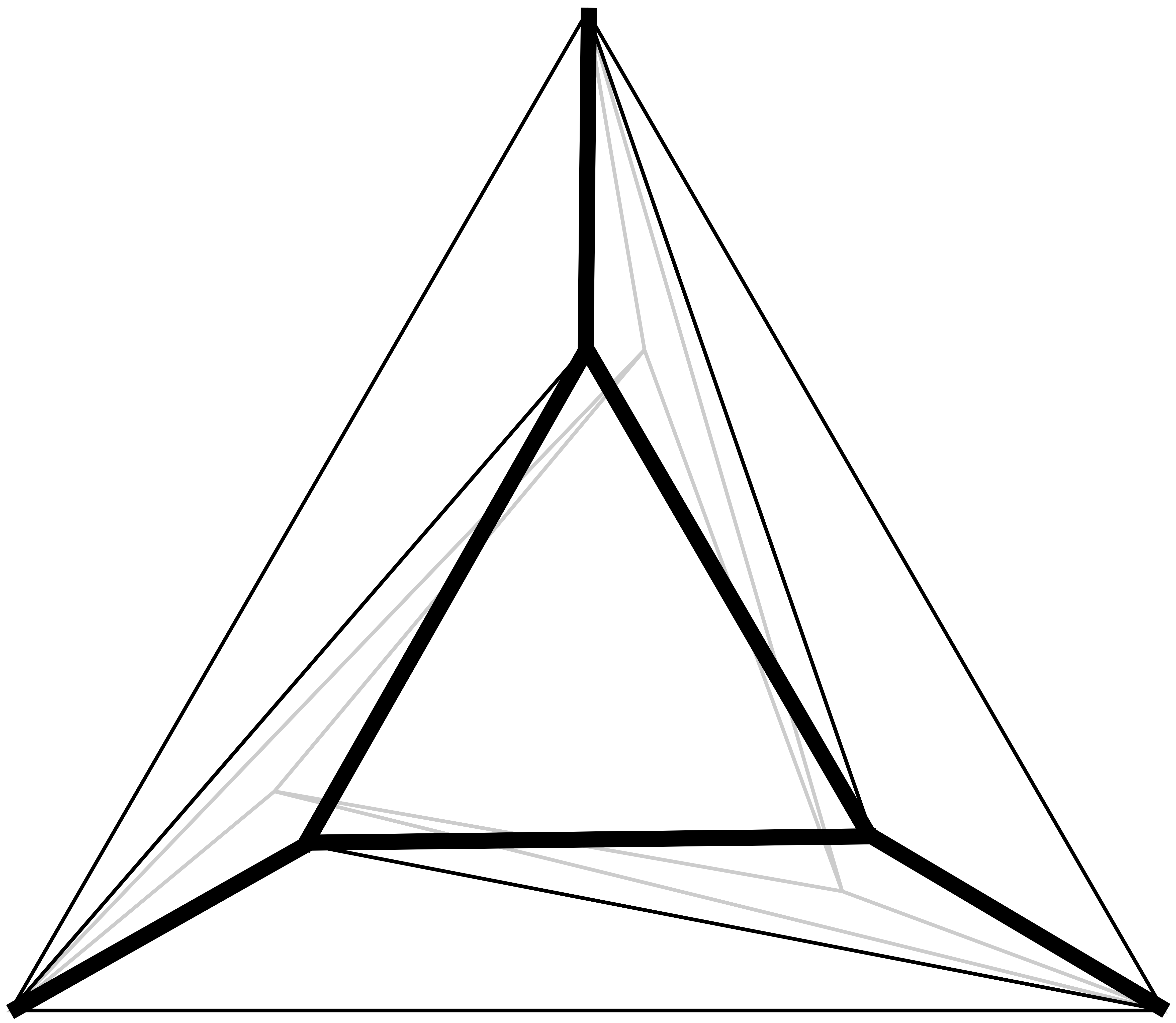
- Three types of edges

1. Boundary
2. Scaled by a constant factor $\mu^{-1} \leq 1$
   Positive coefficient $\omega_{ij} > 0$
3. Too short, varying factor $\leq \mu^{-1}$
   Zero coefficient $\omega_{ij} = 0$



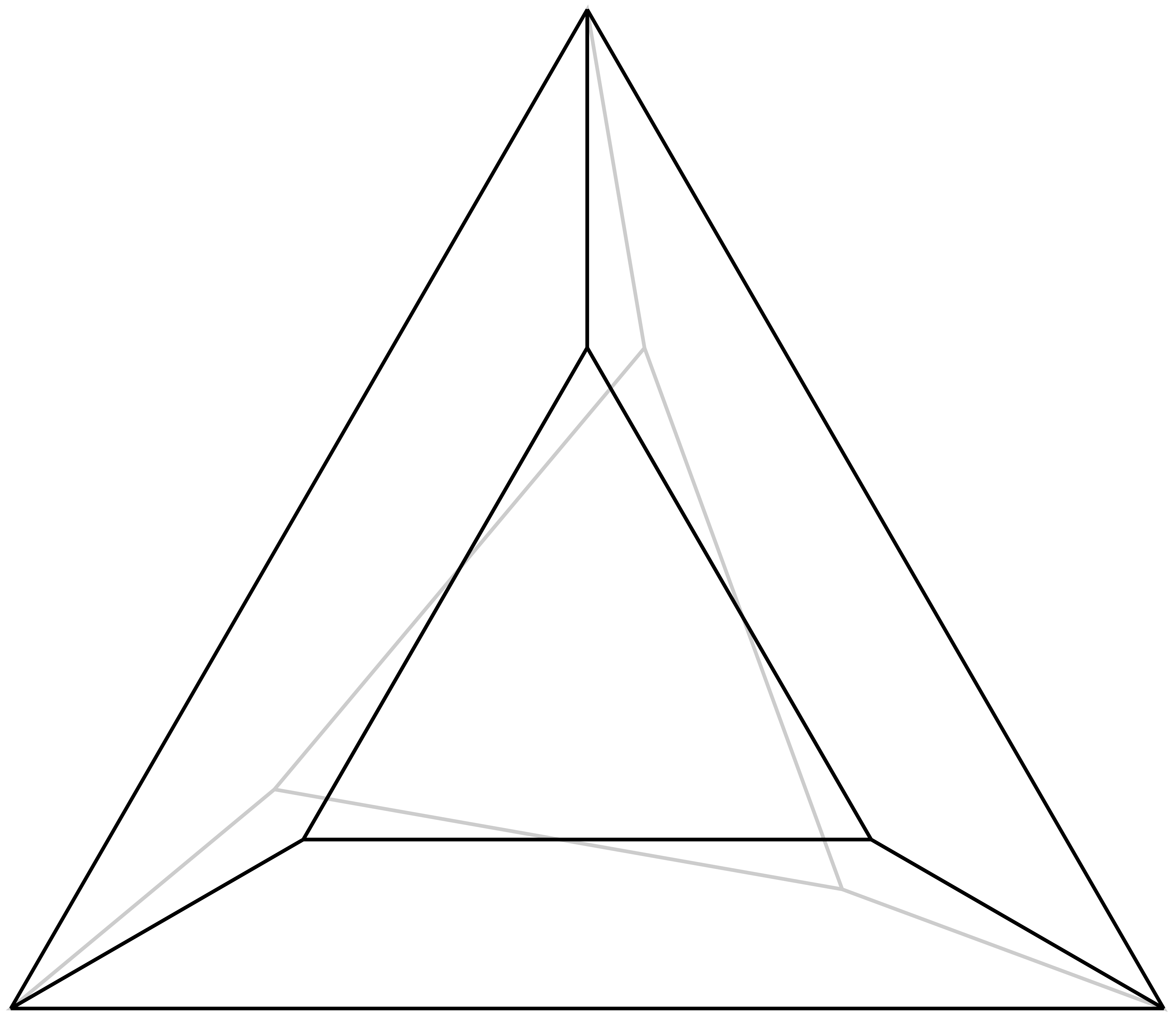
Properties of algorithm: Laplacian

- Selects the subset of edges that
  - can be embedded with positive coefficients $\omega_{ij} > 0$
  - so that edge lengths are preserved up to global scale



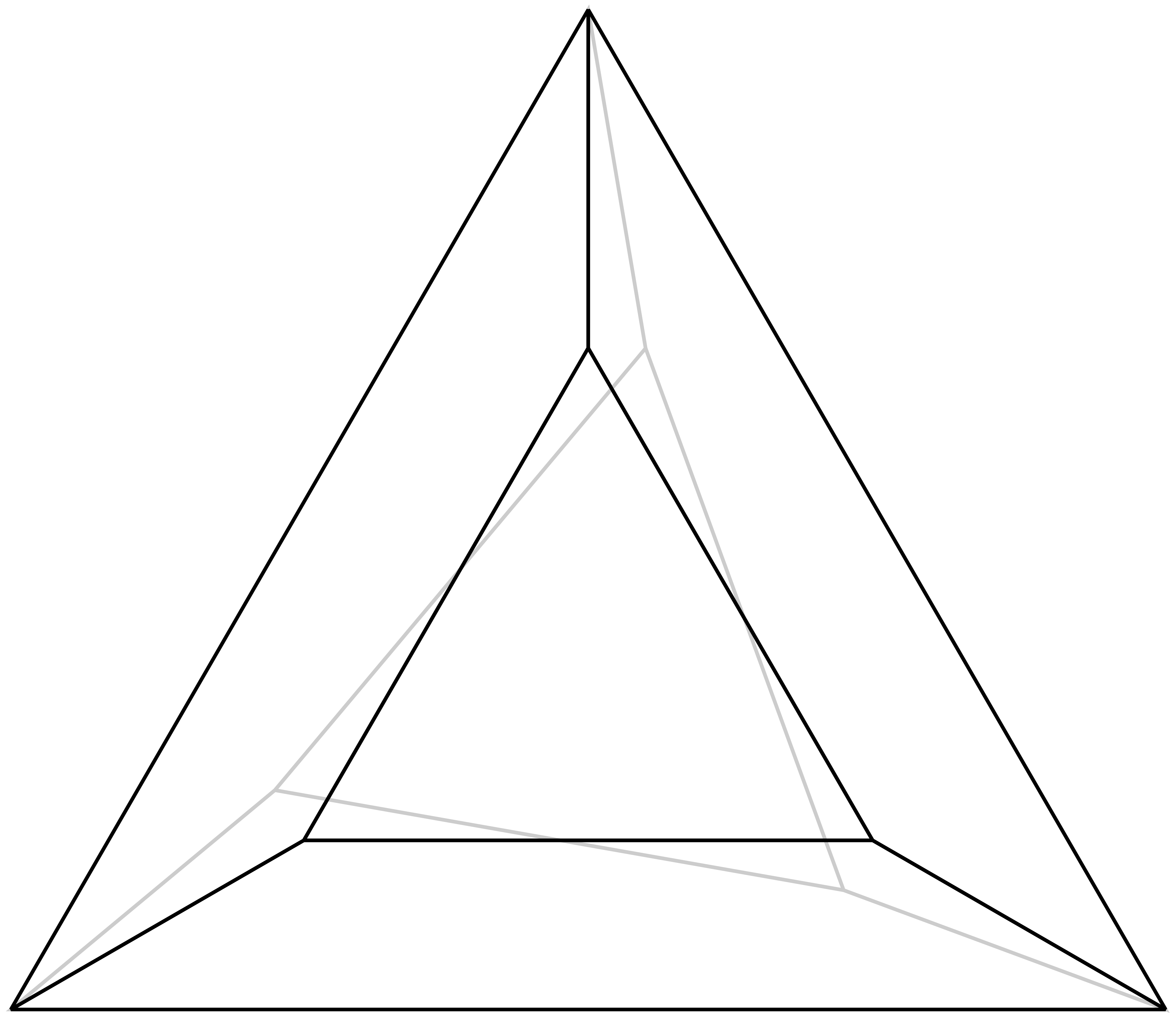
Properties of algorithm: polygonal!

- Works for any (planar) three-connected graph

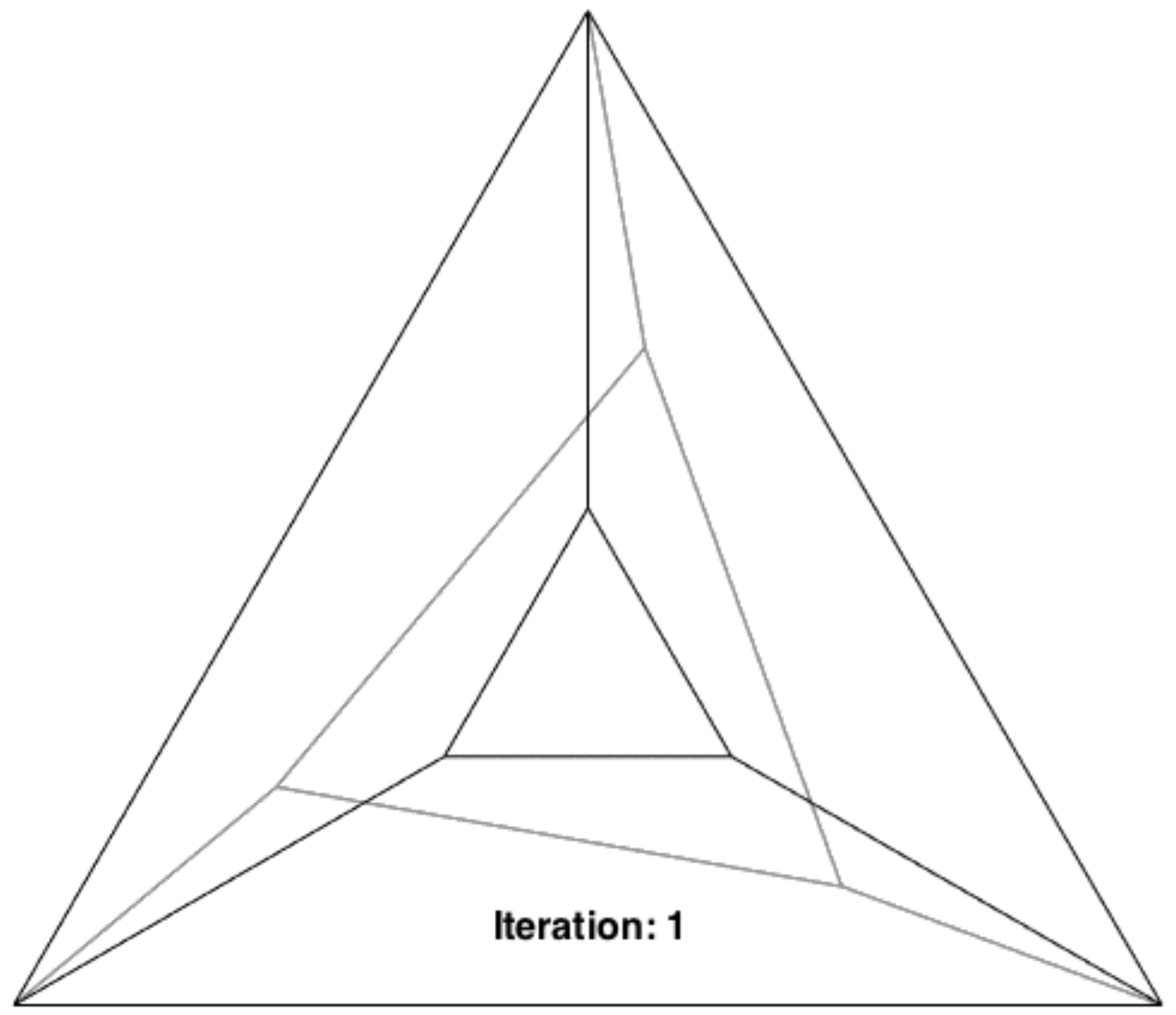


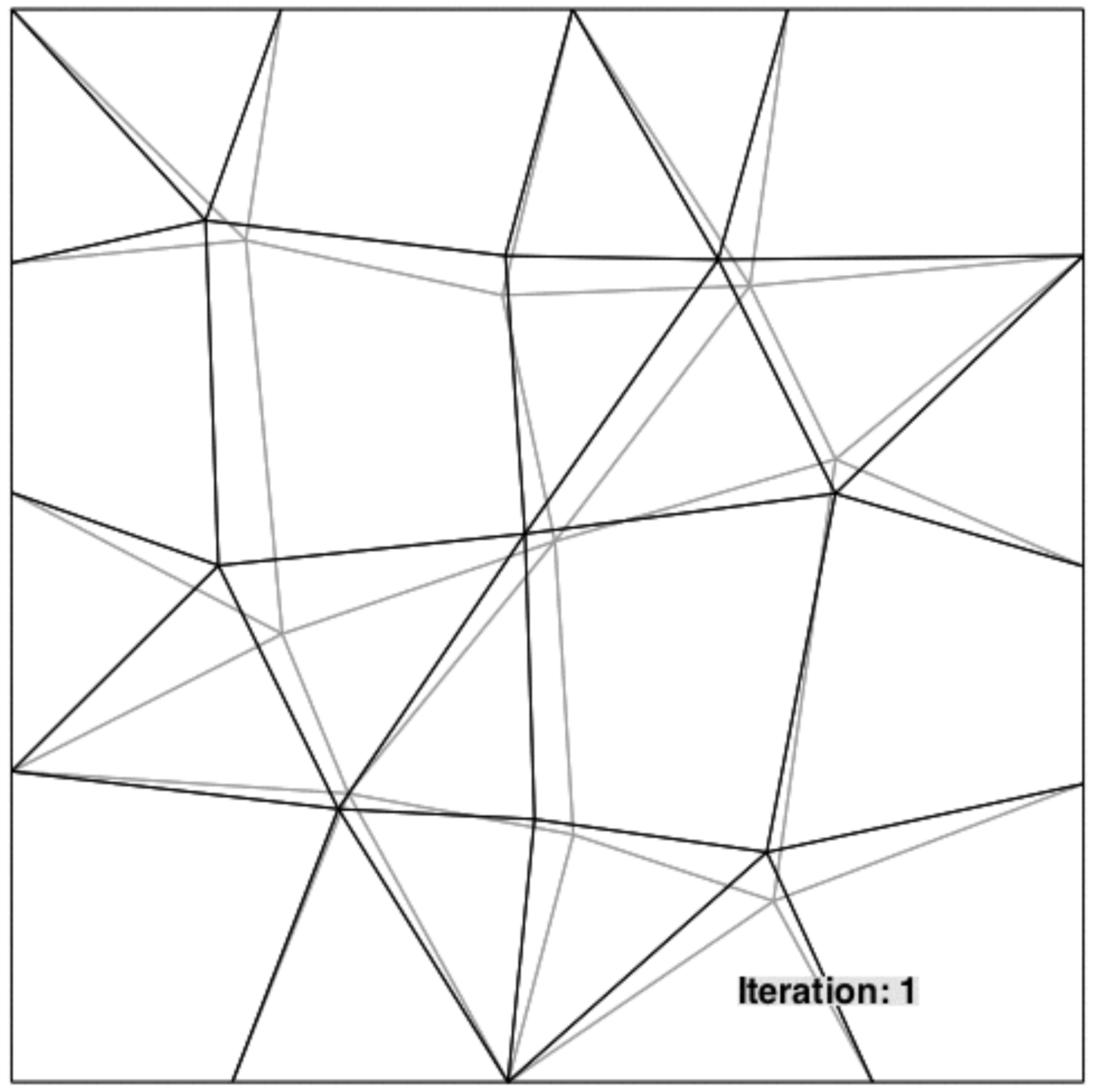
Properties of algorithm: polygonal!

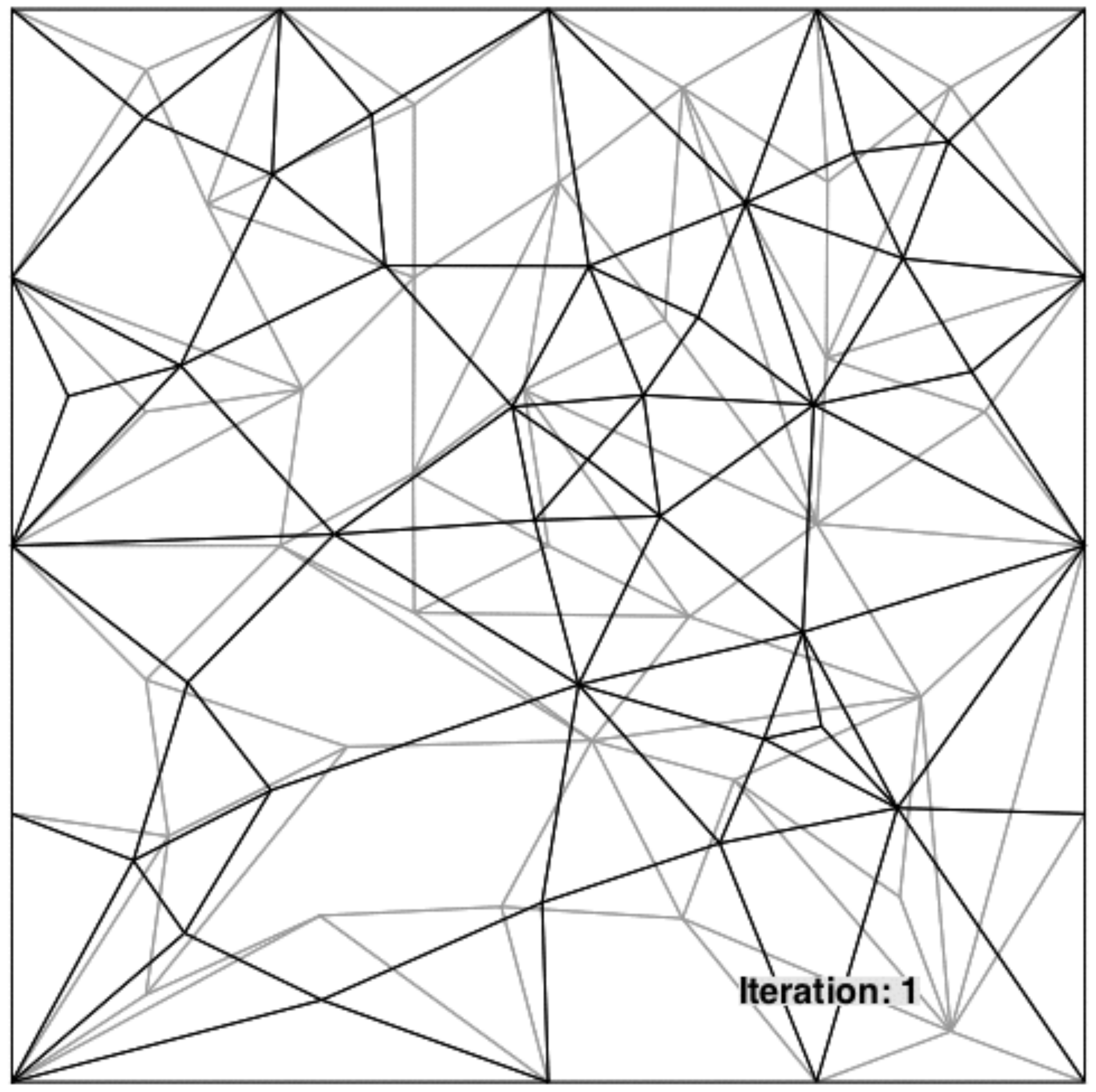
- No free lunch theorem for polygon meshes
  - Same as triangles: “regular subdivisions” (= power diagrams)



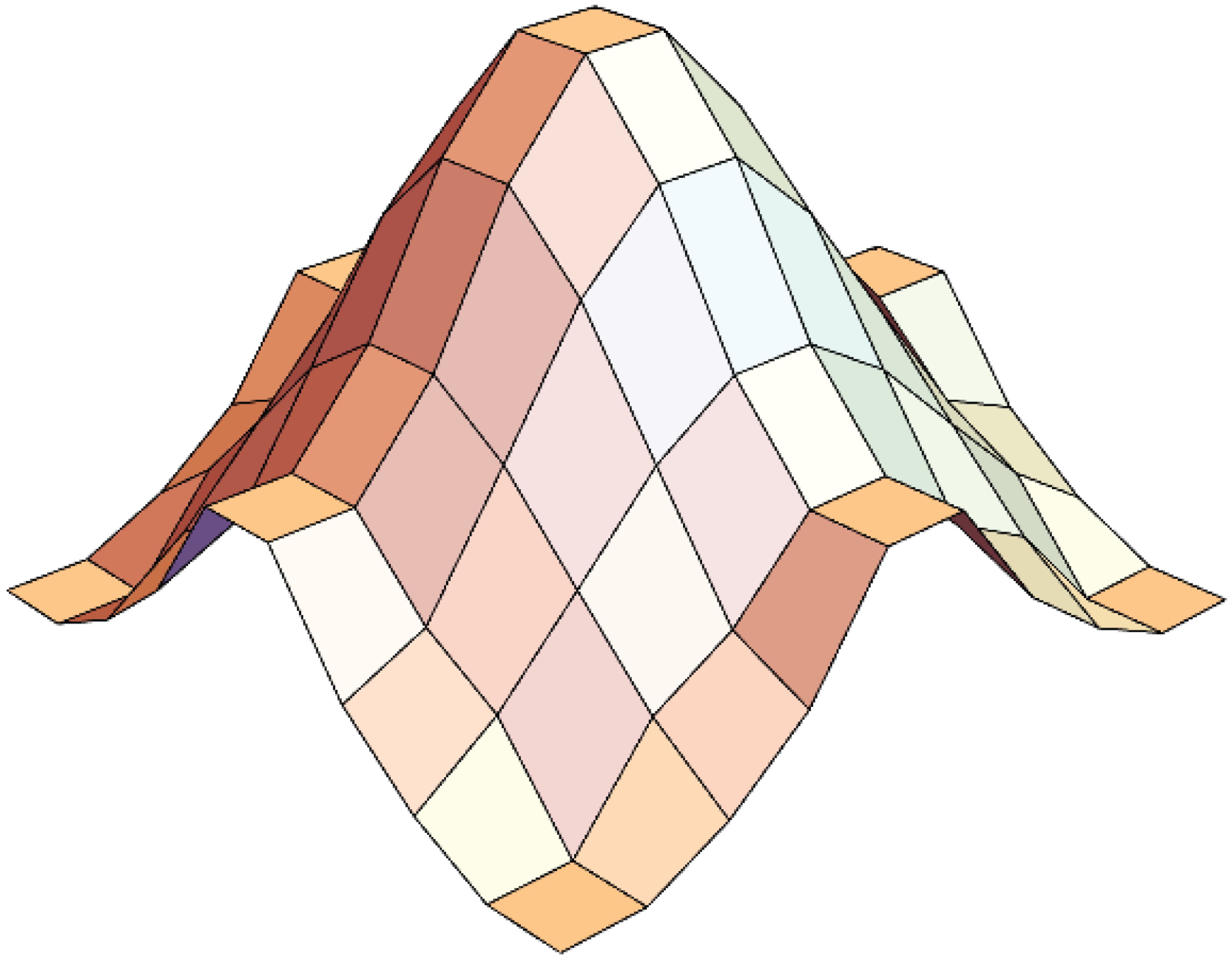
Properties of algorithm: polygonal!

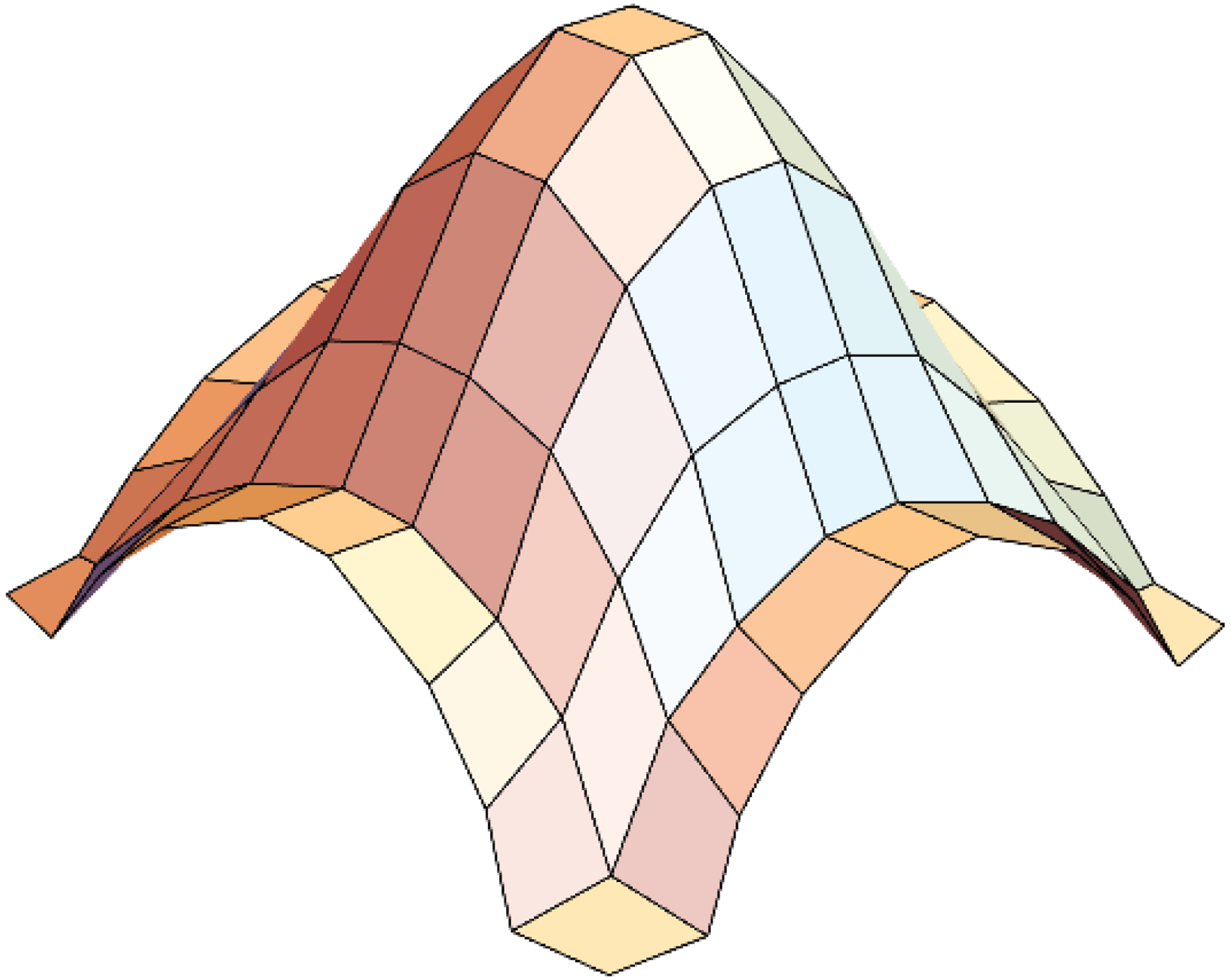




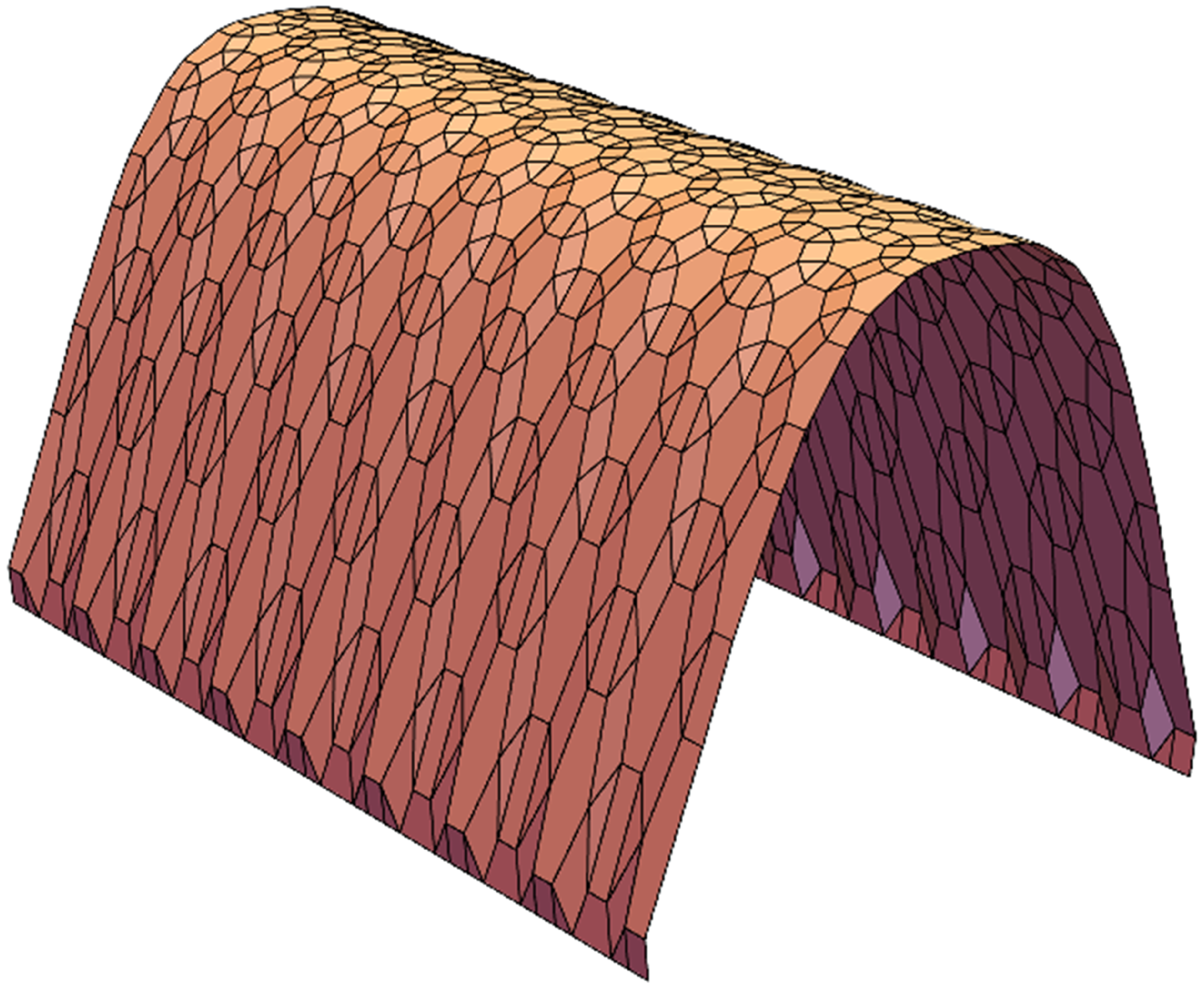


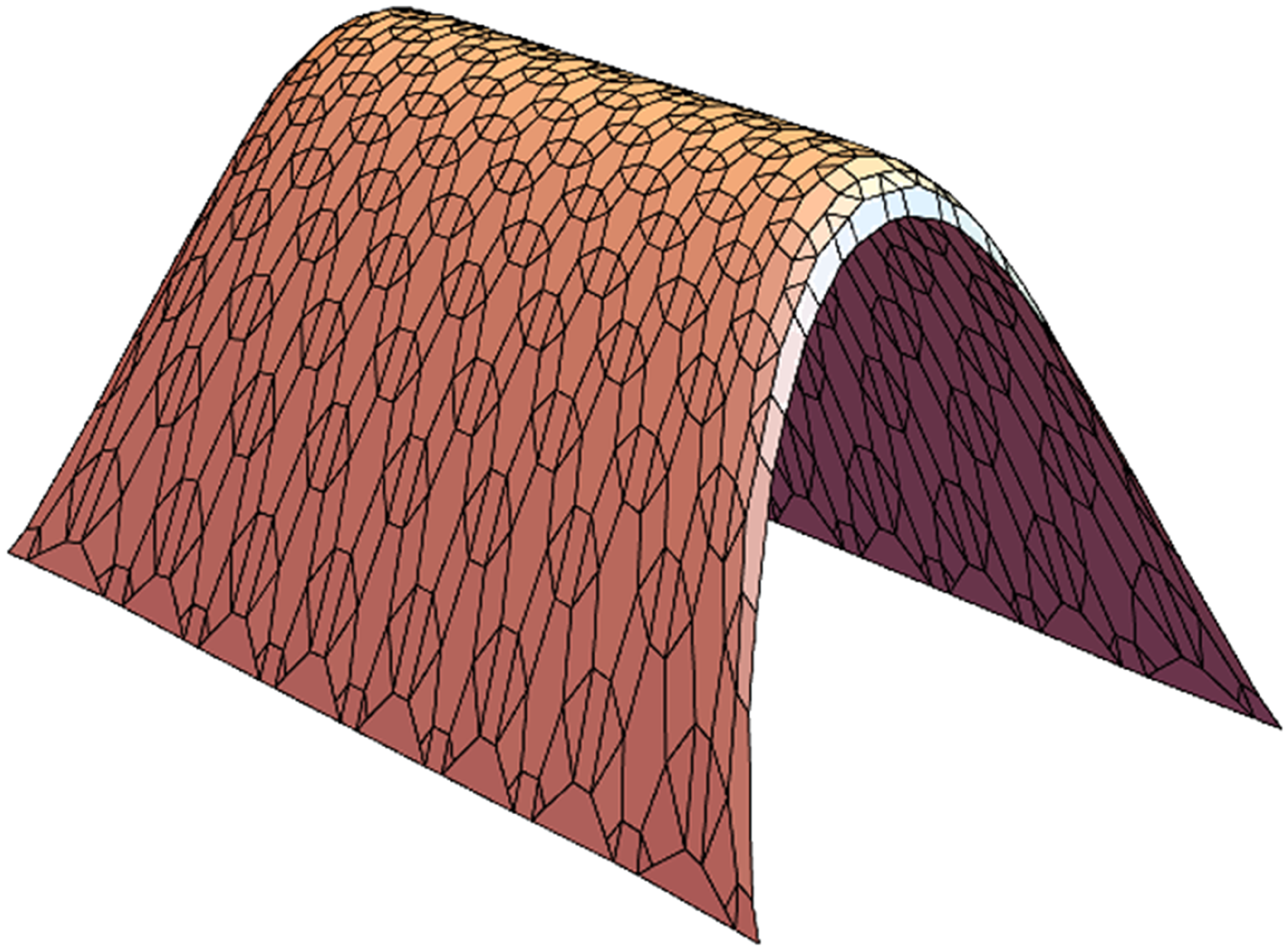
Application: self-supporting surface



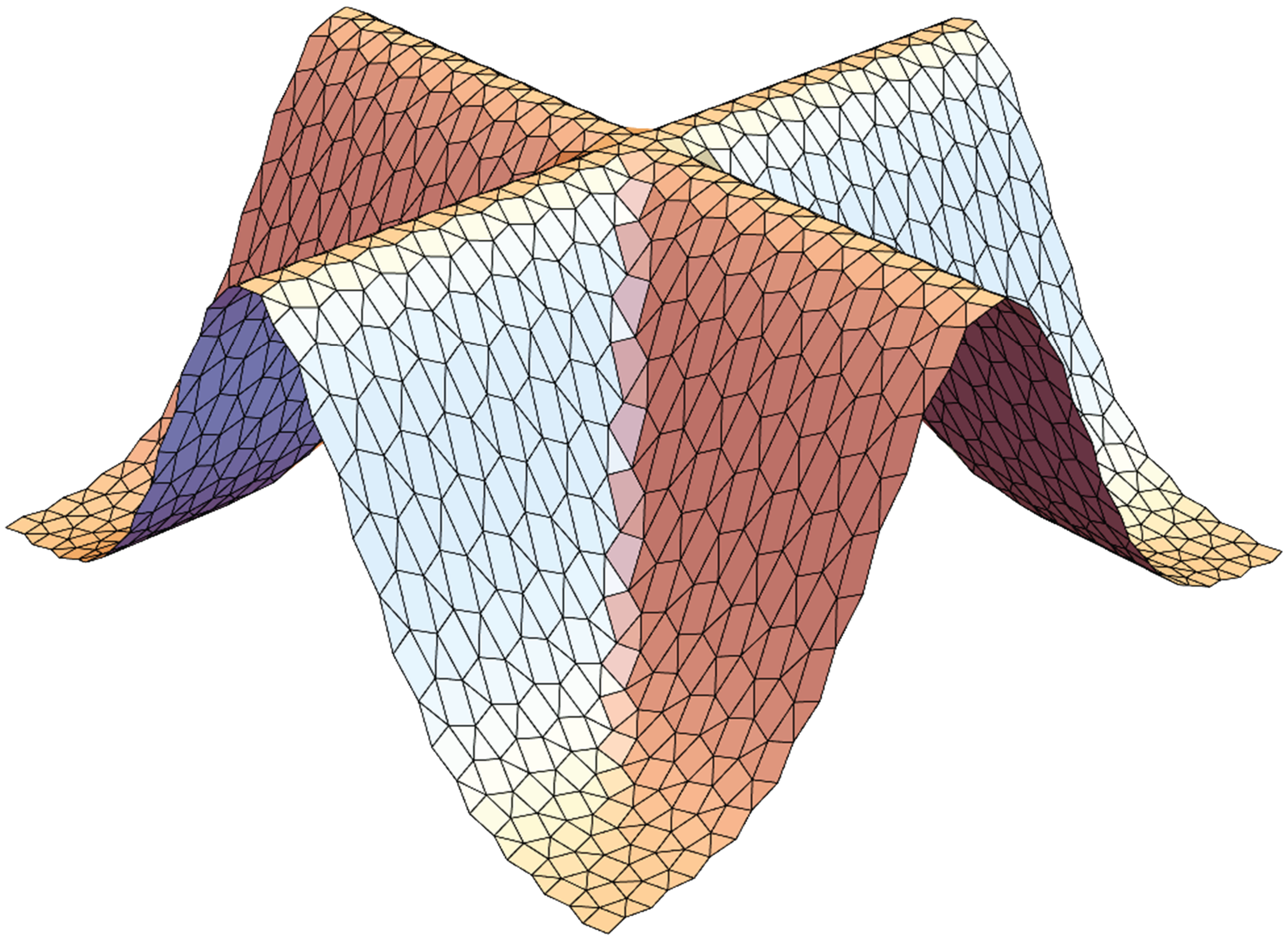


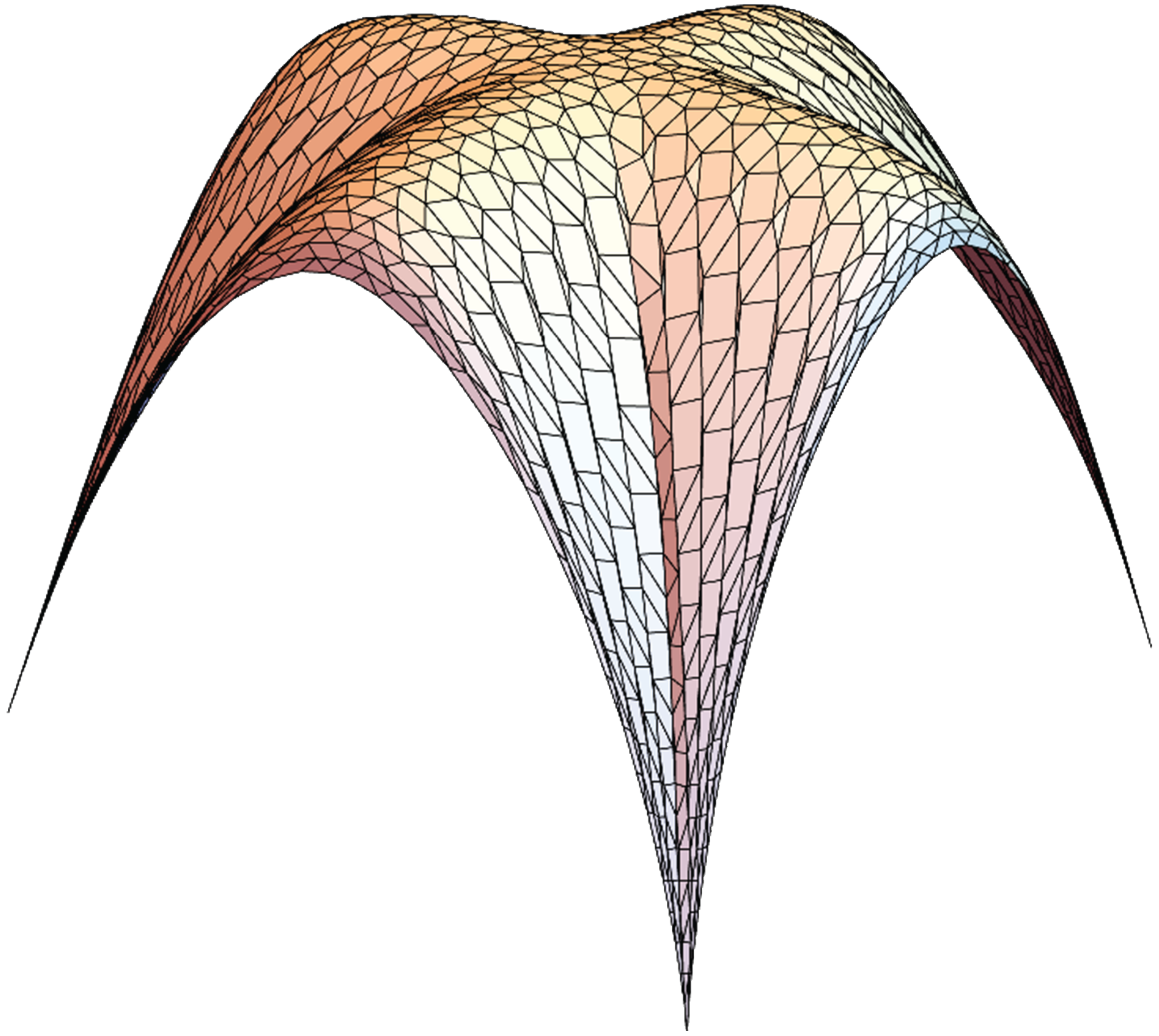
Application: self-supporting surface





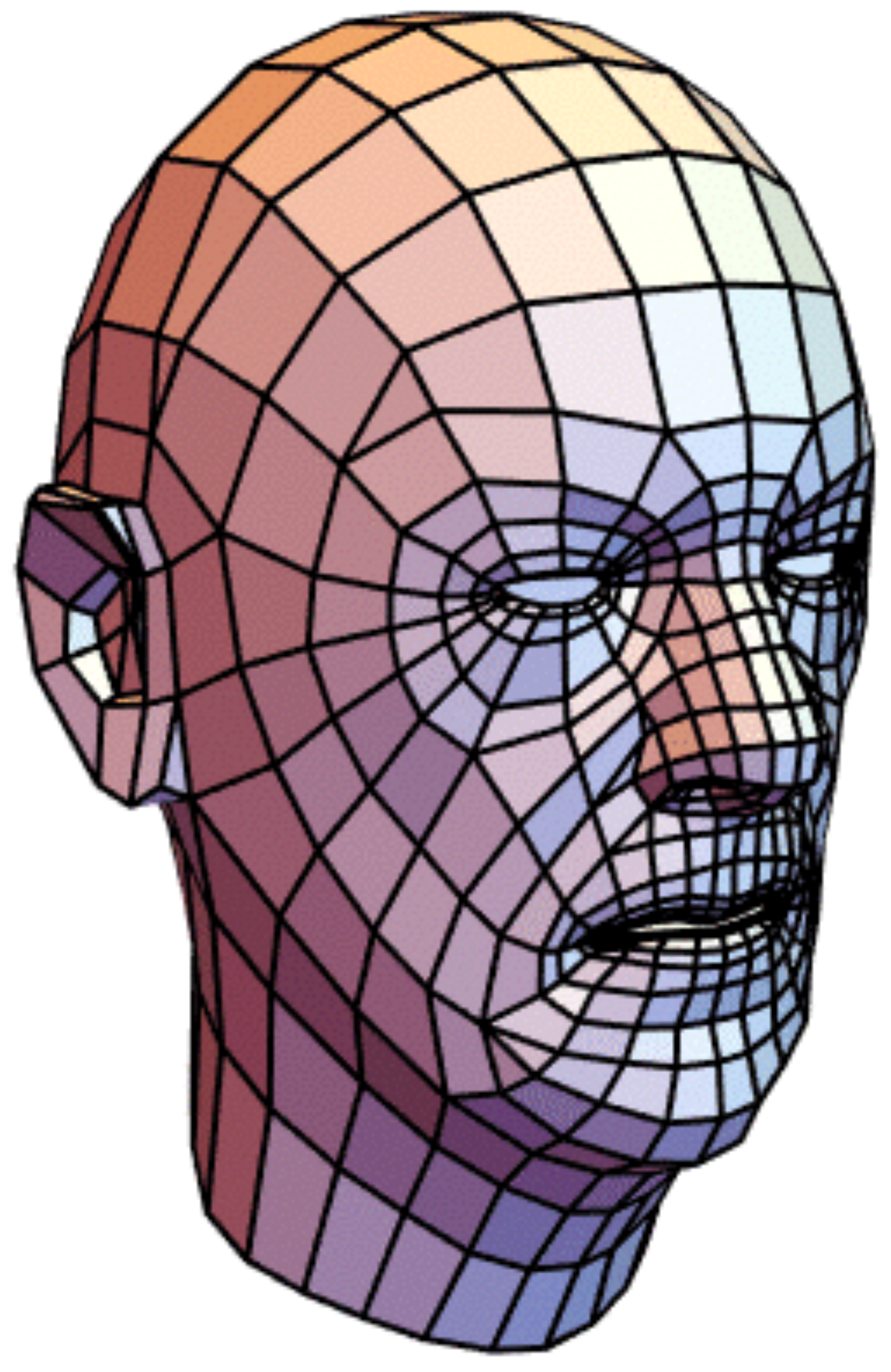
Application: self-supporting surface





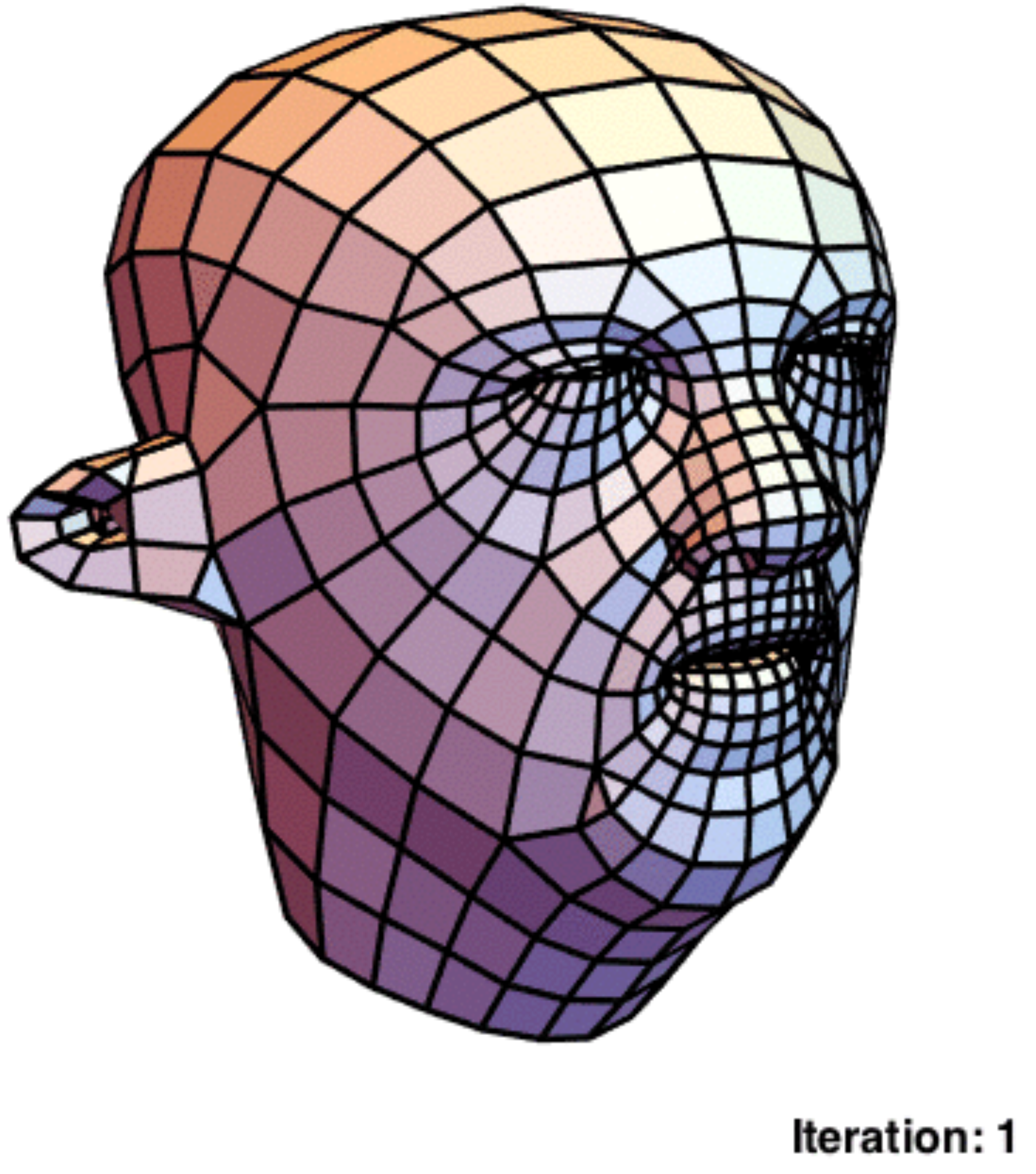
3D Mesh Laplacian

- Recall $\mathbf{L}\mathbf{V}_\Omega = \mathbf{0}$
  - All vertices flat
- Instead $(\mathbf{L}\mathbf{V}_\Omega)_i = H_i \mathbf{n}_i$
  - Take area gradient for $H_i \mathbf{n}_i$



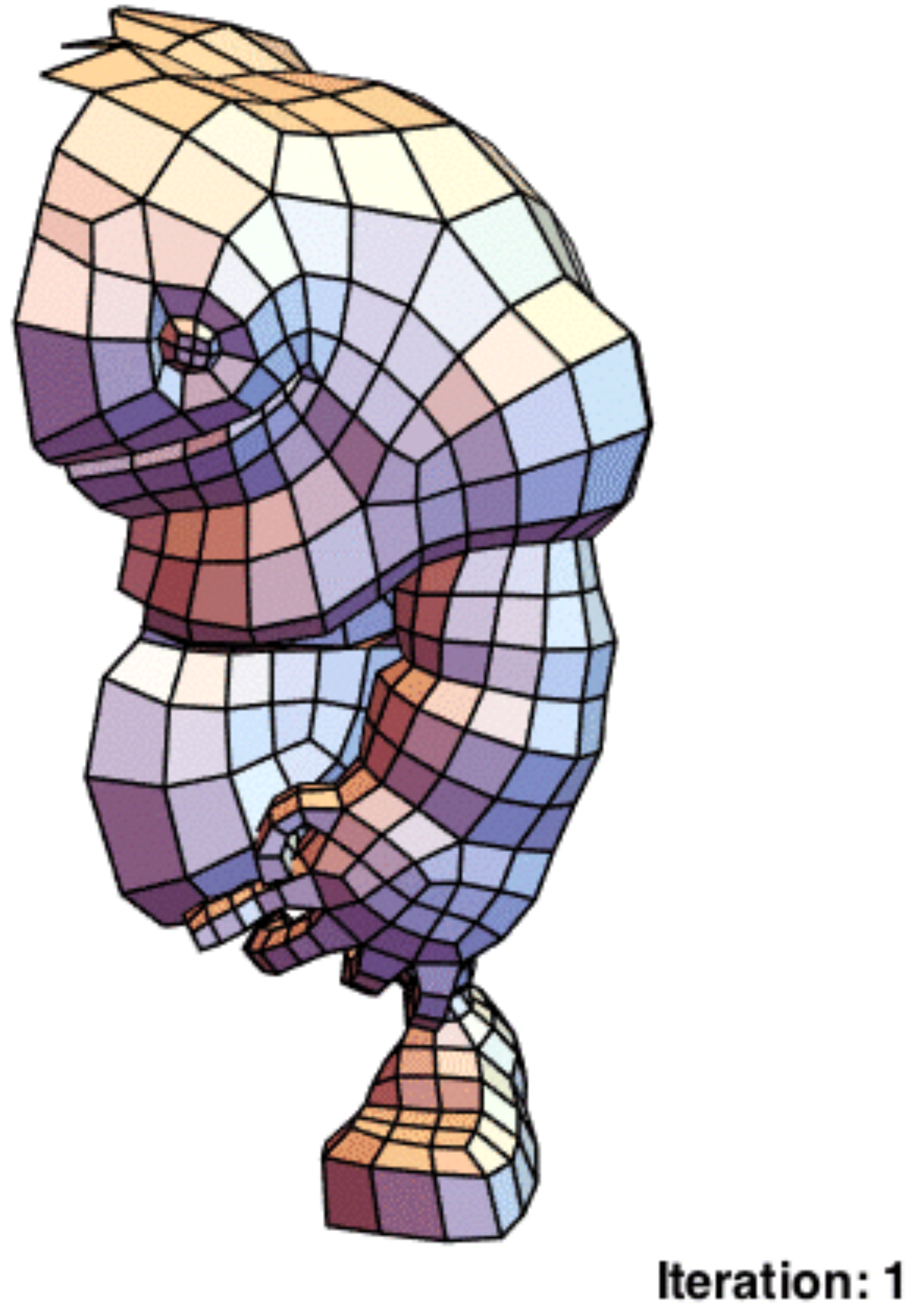
3D Mesh Laplacian

- Recall $\mathbf{L}\mathbf{V}_\Omega = \mathbf{0}$
  - All vertices flat
- Instead $(\mathbf{L}\mathbf{V}_\Omega)_i = H_i \mathbf{n}_i$
  - Take area gradient for $H_i \mathbf{n}_i$



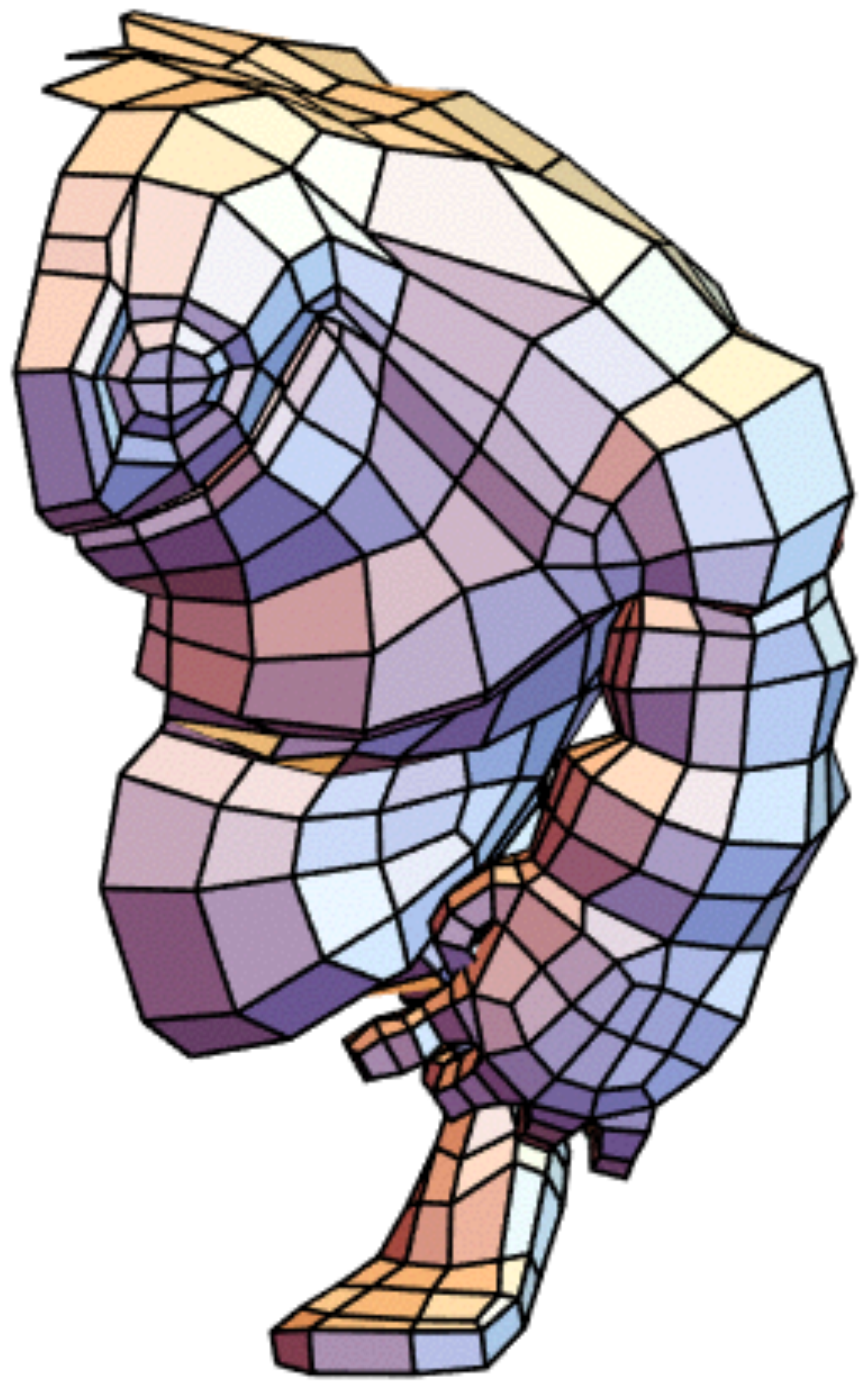
3D Mesh Laplacian

- Recall $\mathbf{L}\mathbf{V}_\Omega = \mathbf{0}$
    - All vertices flat
- Instead $(\mathbf{L}\mathbf{V}_\Omega)_i = H_i\mathbf{n}_i$
    - Take area gradient for $H_i\mathbf{n}_i$
- Other variants are possible



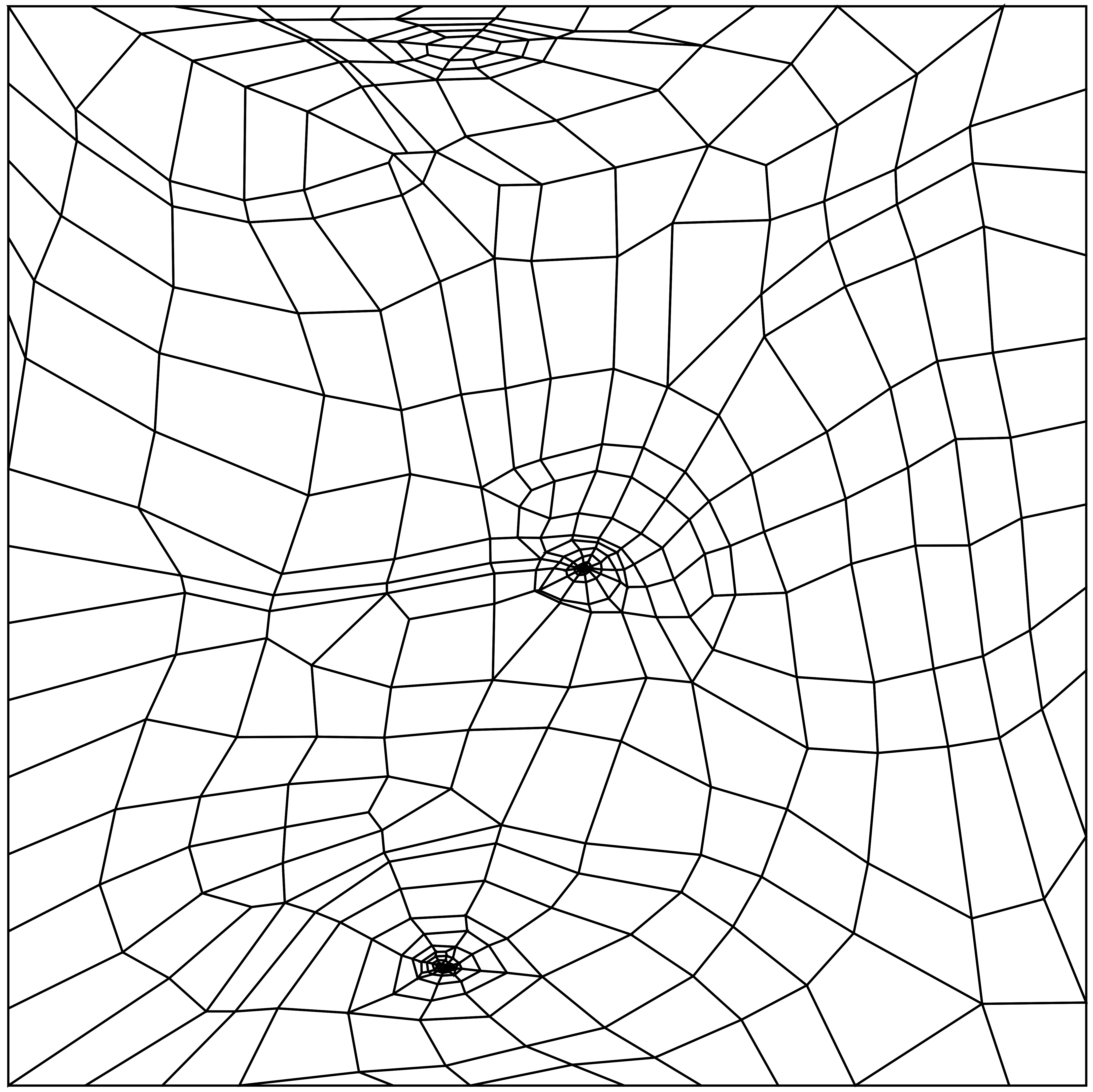
Mesh parameterization

- Fix boundary in the plane
- Solve $\mathbf{LV} = \mathbf{0}$
- $\omega_{ij} \geq 0$ guarantees no flipped faces



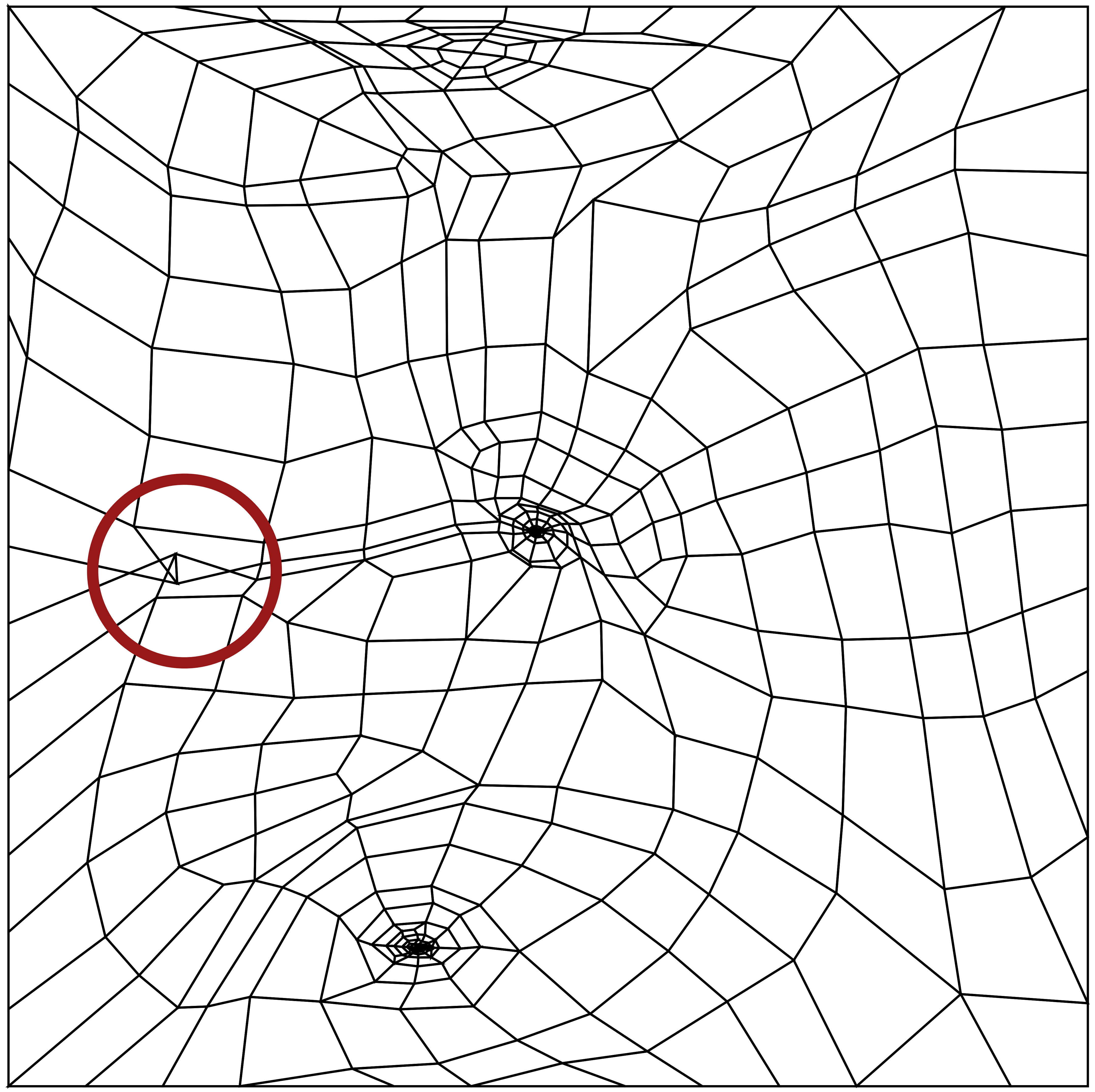
Mesh parameterization

- Fix boundary in the plane
- Solve $\mathbf{LV} = \mathbf{0}$
- $\omega_{ij} \geq 0$ guarantees no flipped faces



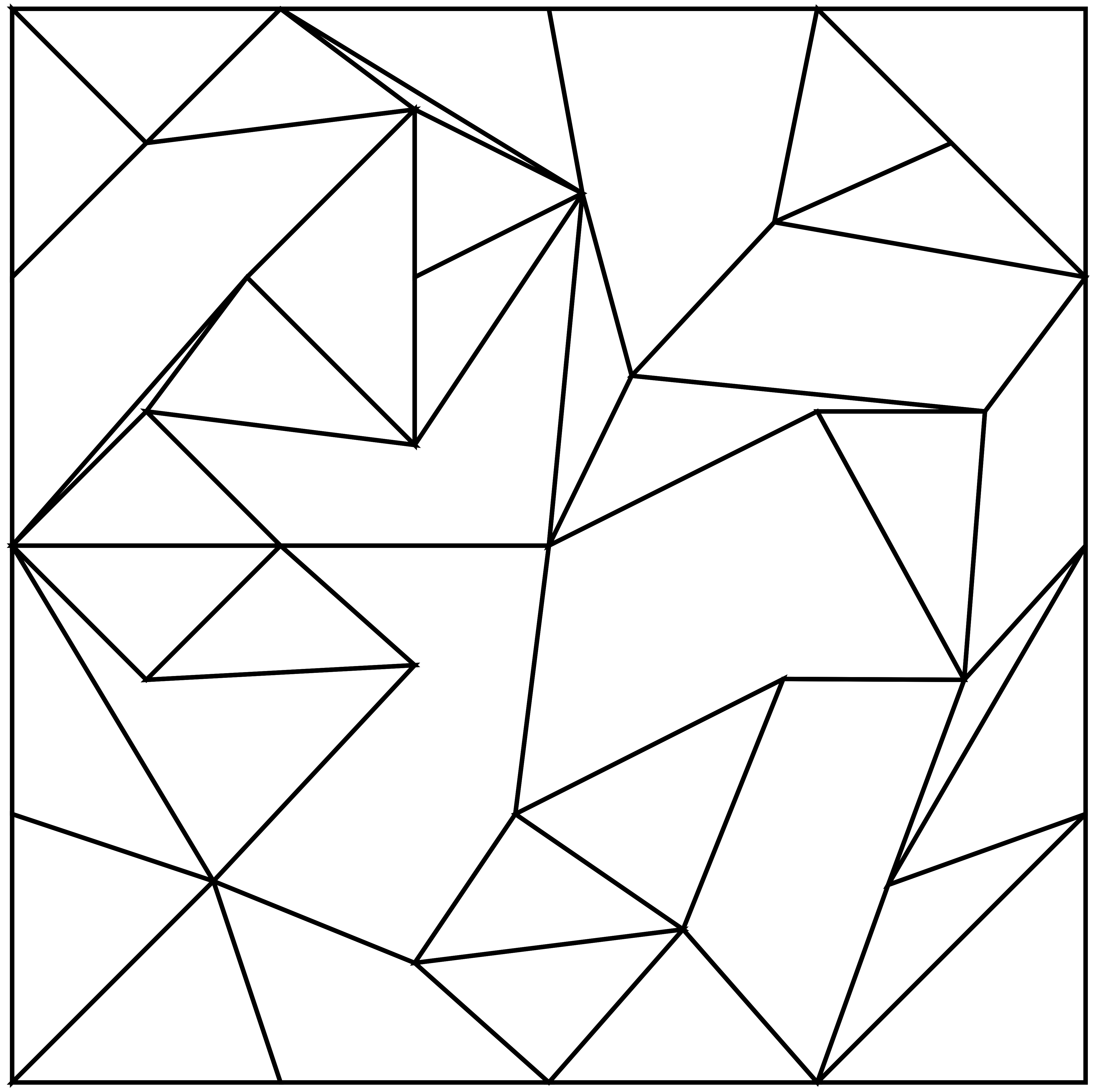
Mesh parameterization

- Fix boundary in the plane
- Solve $\mathbf{LV} = \mathbf{0}$
- $\omega_{ij} < 0$ flips may occur
  - Wardetzky & co-worker



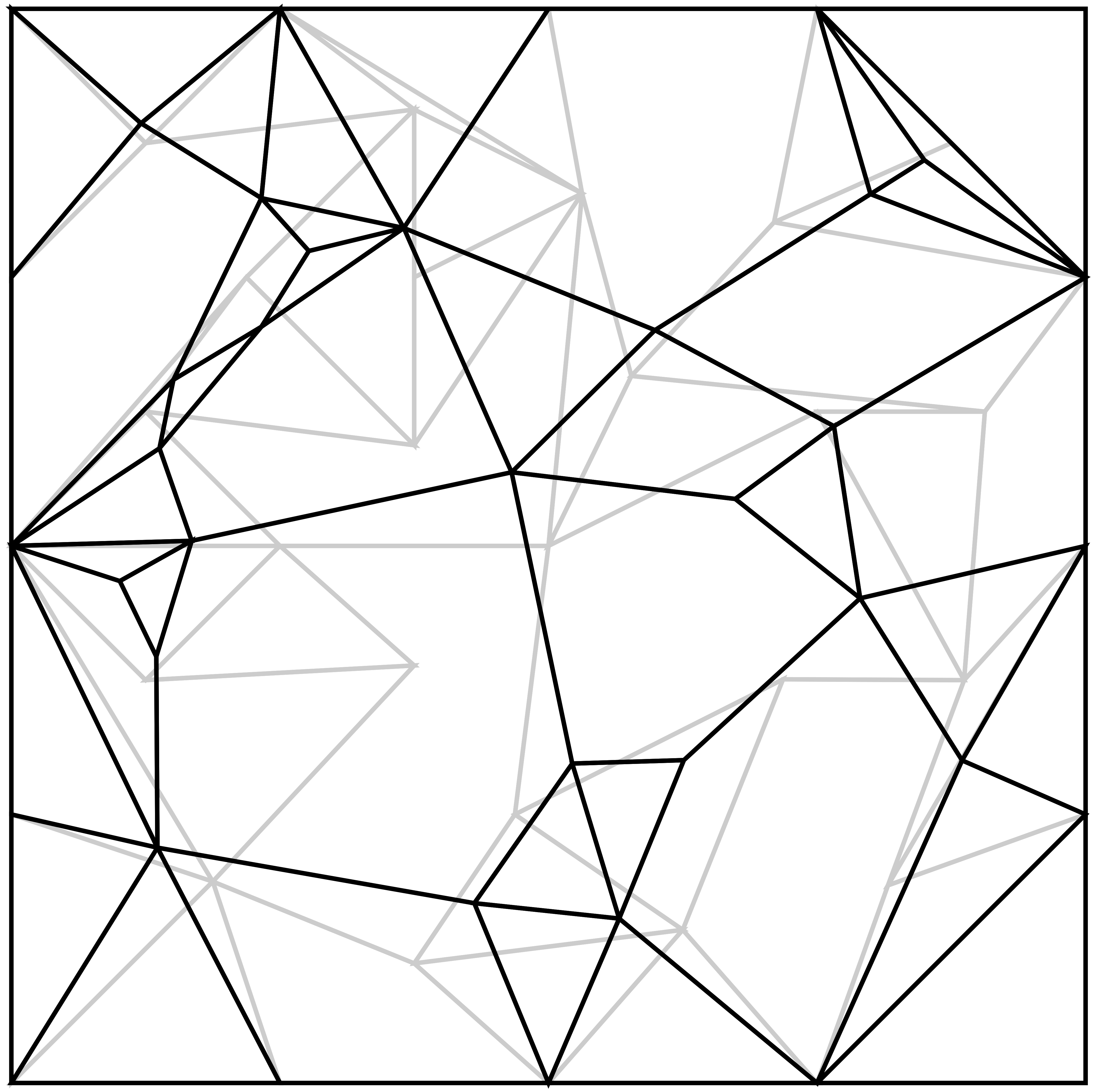
Non-zero coefficients on polygons

- Non-zero coefficients only on edges
    - No convex faces



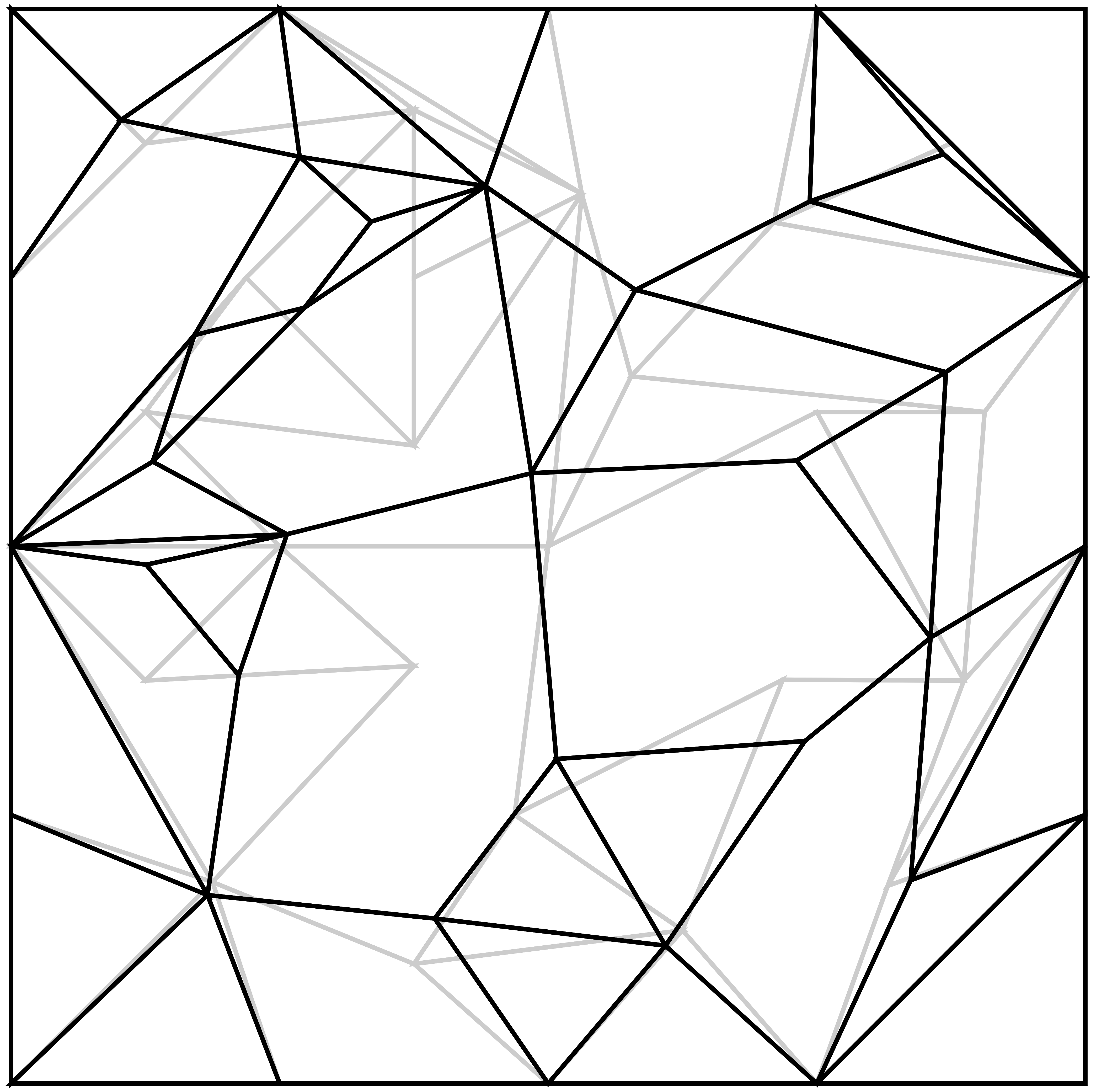
Non-zero coefficients on polygons

- Non-zero coefficients only on edges
  - No convex faces



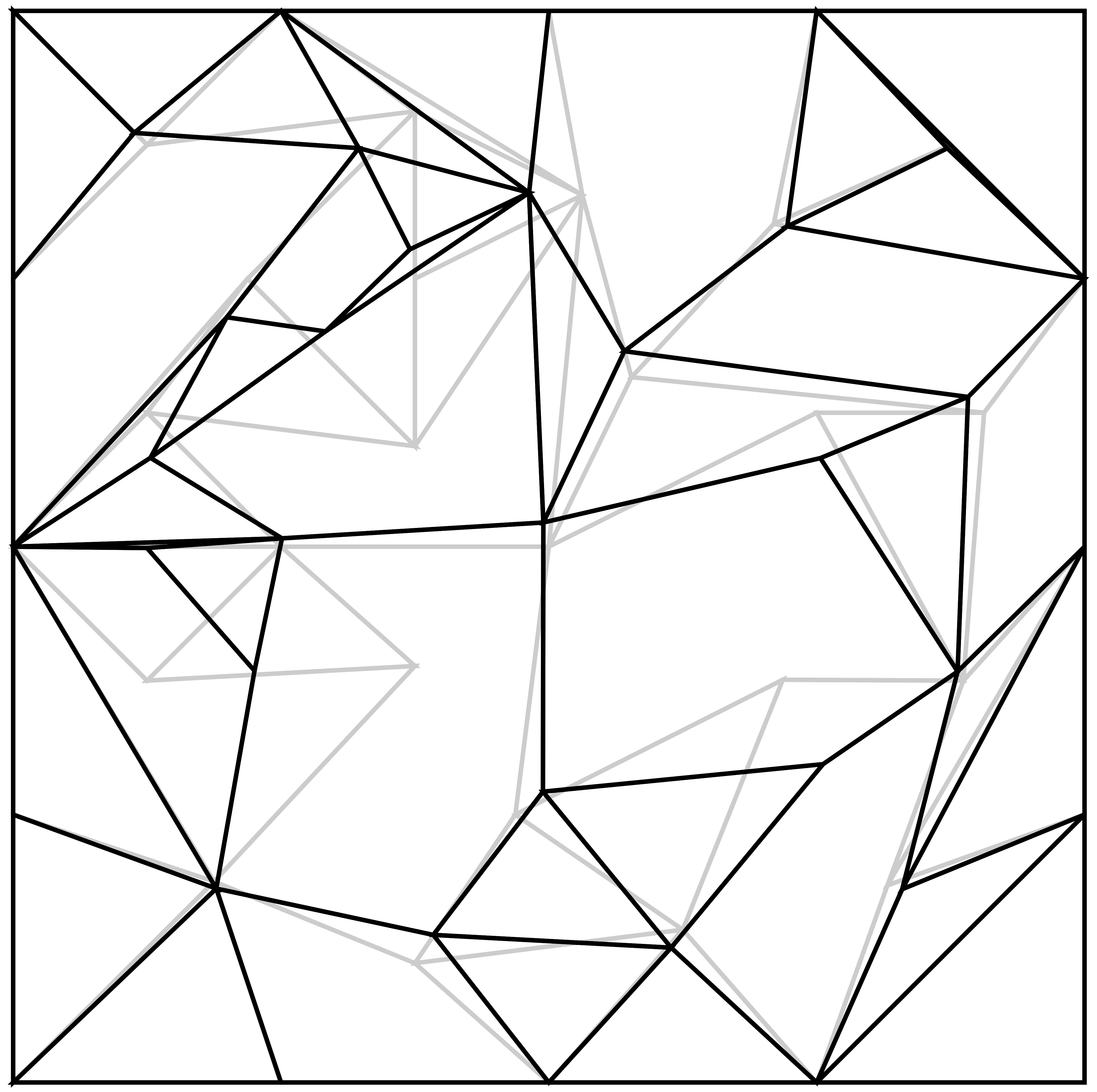
Non-zero coefficients on polygons

- Non-zero coefficients only on edges
    - No convex faces



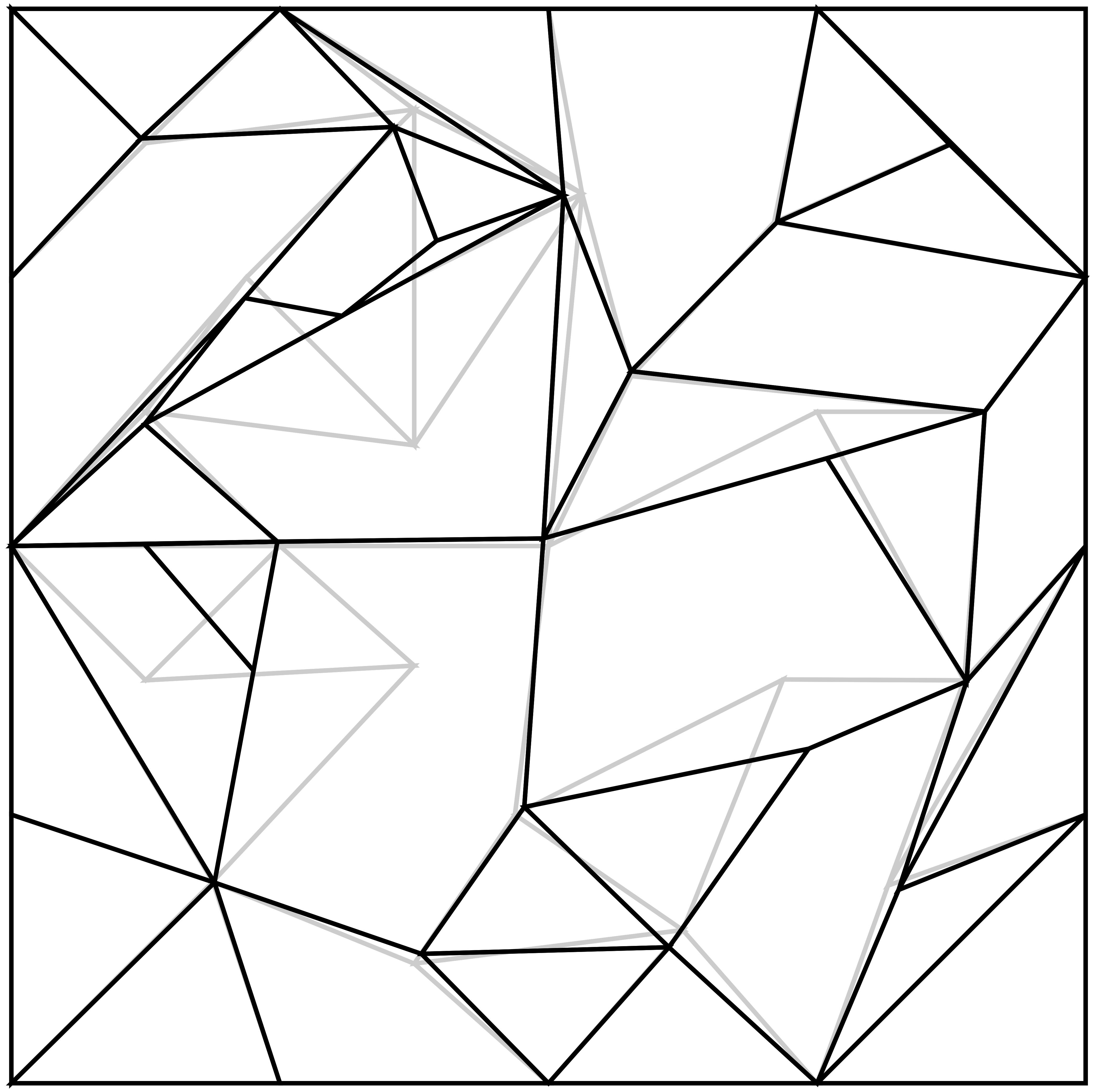
Non-zero coefficients on polygons

- Non-zero coefficients only on edges
    - No convex faces



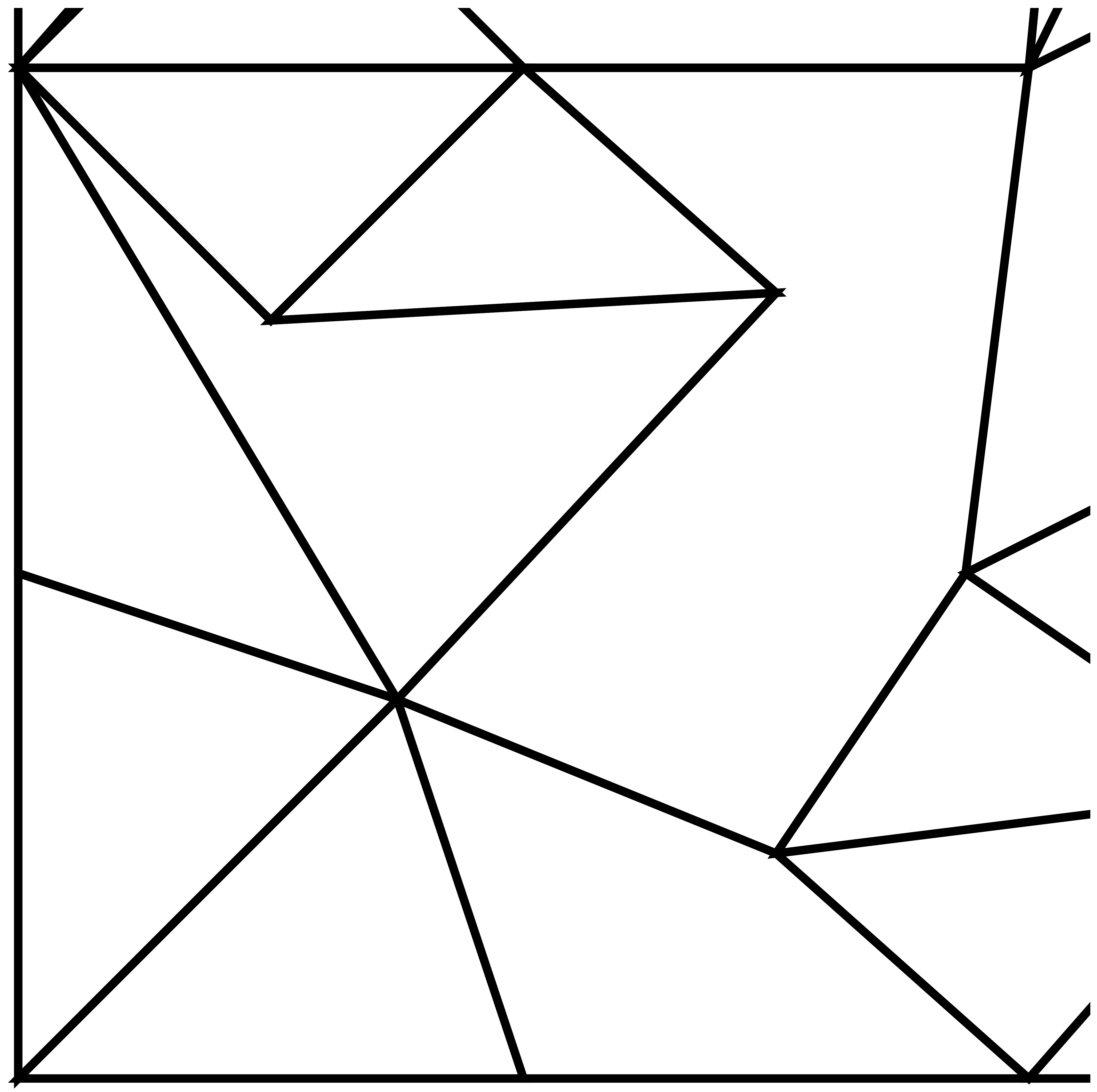
Non-zero coefficients on polygons

- Non-zero coefficients only on edges
    - No convex faces



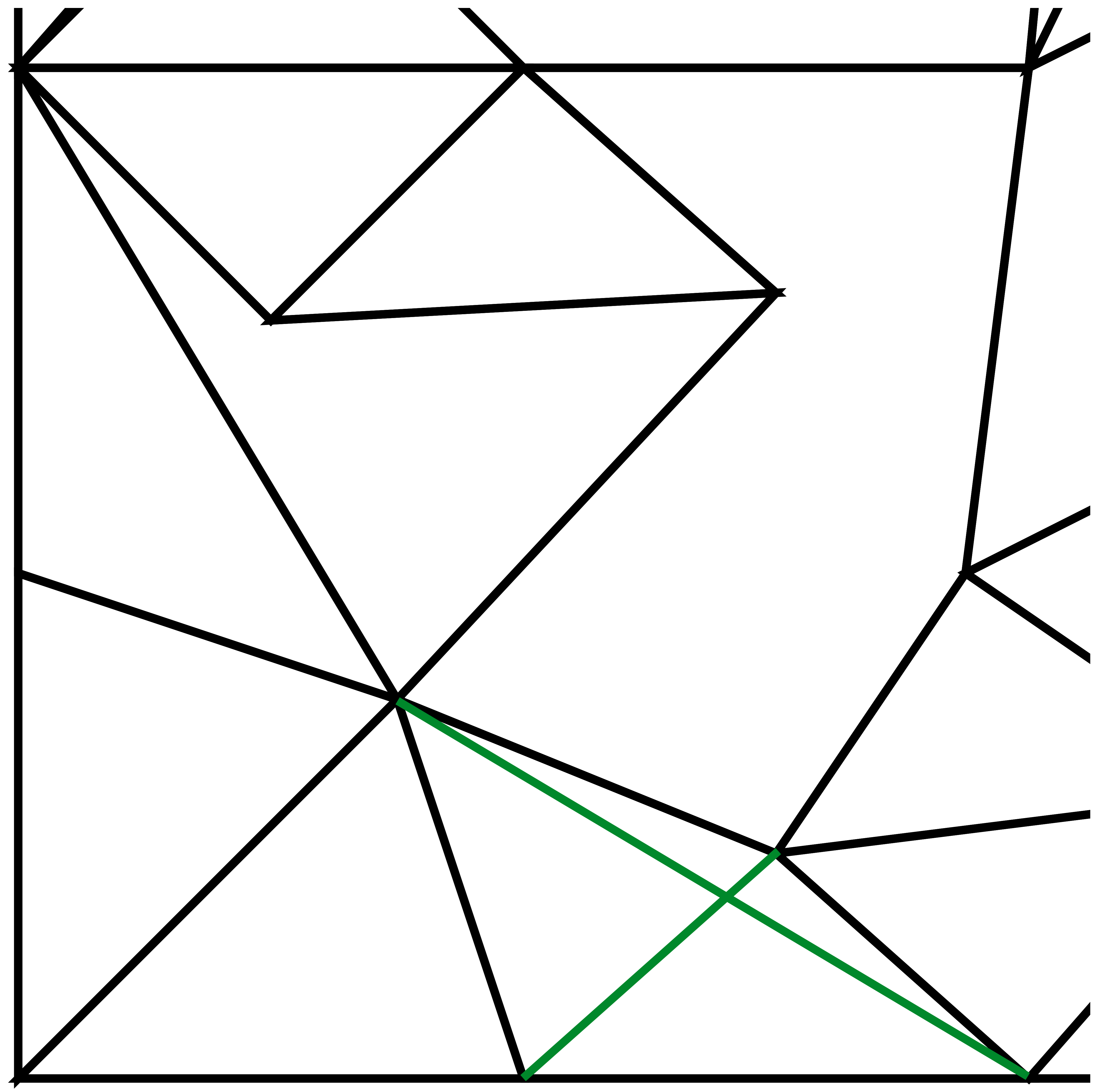
Non-zero coefficients on polygons

- Wardetzky & co-worker:
  all diagonals



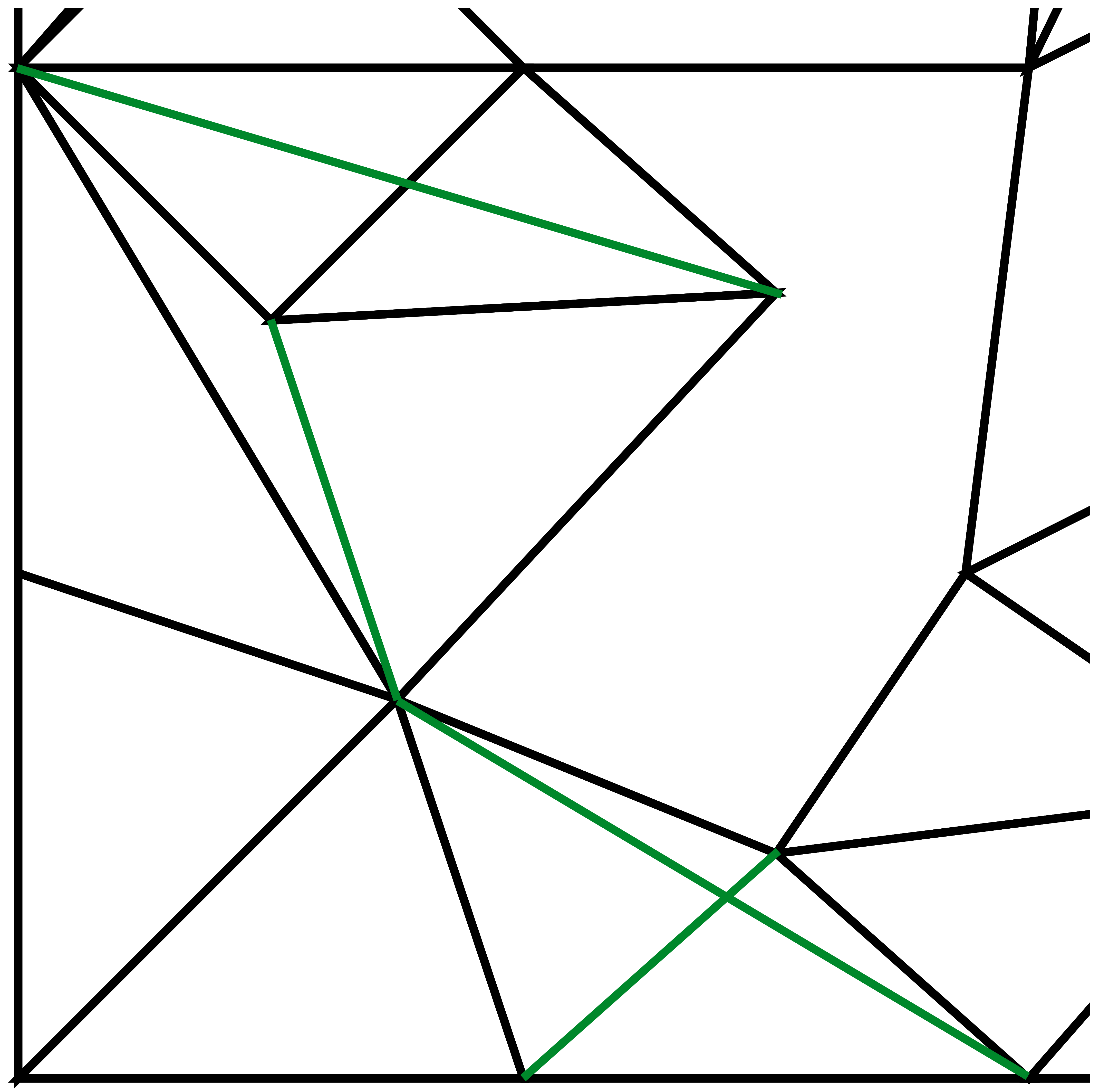
Non-zero coefficients on polygons

- Wardetzky & co-worker:
  all diagonals



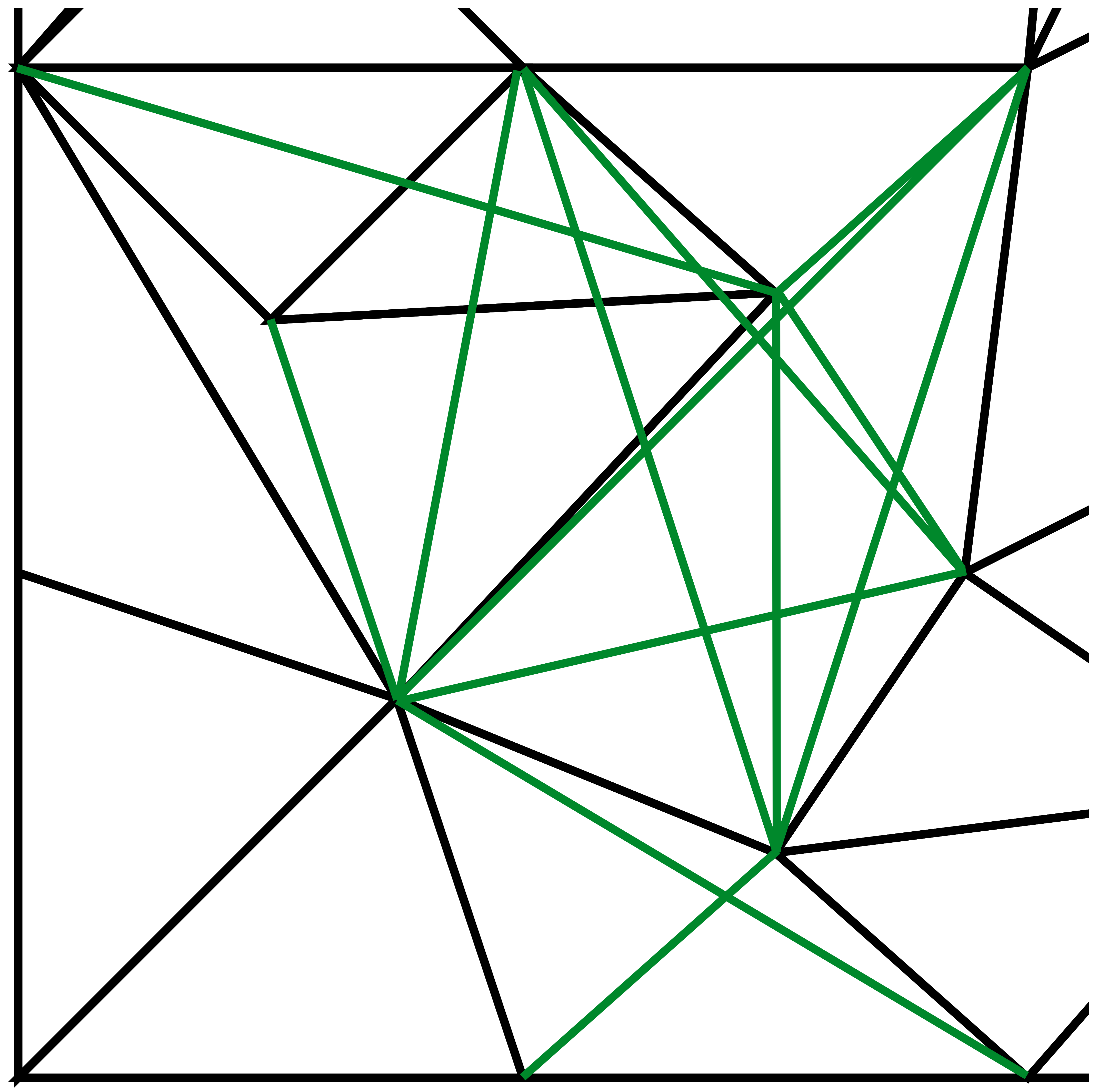
Non-zero coefficients on polygons

- Wardetzky & co-worker:
  all diagonals



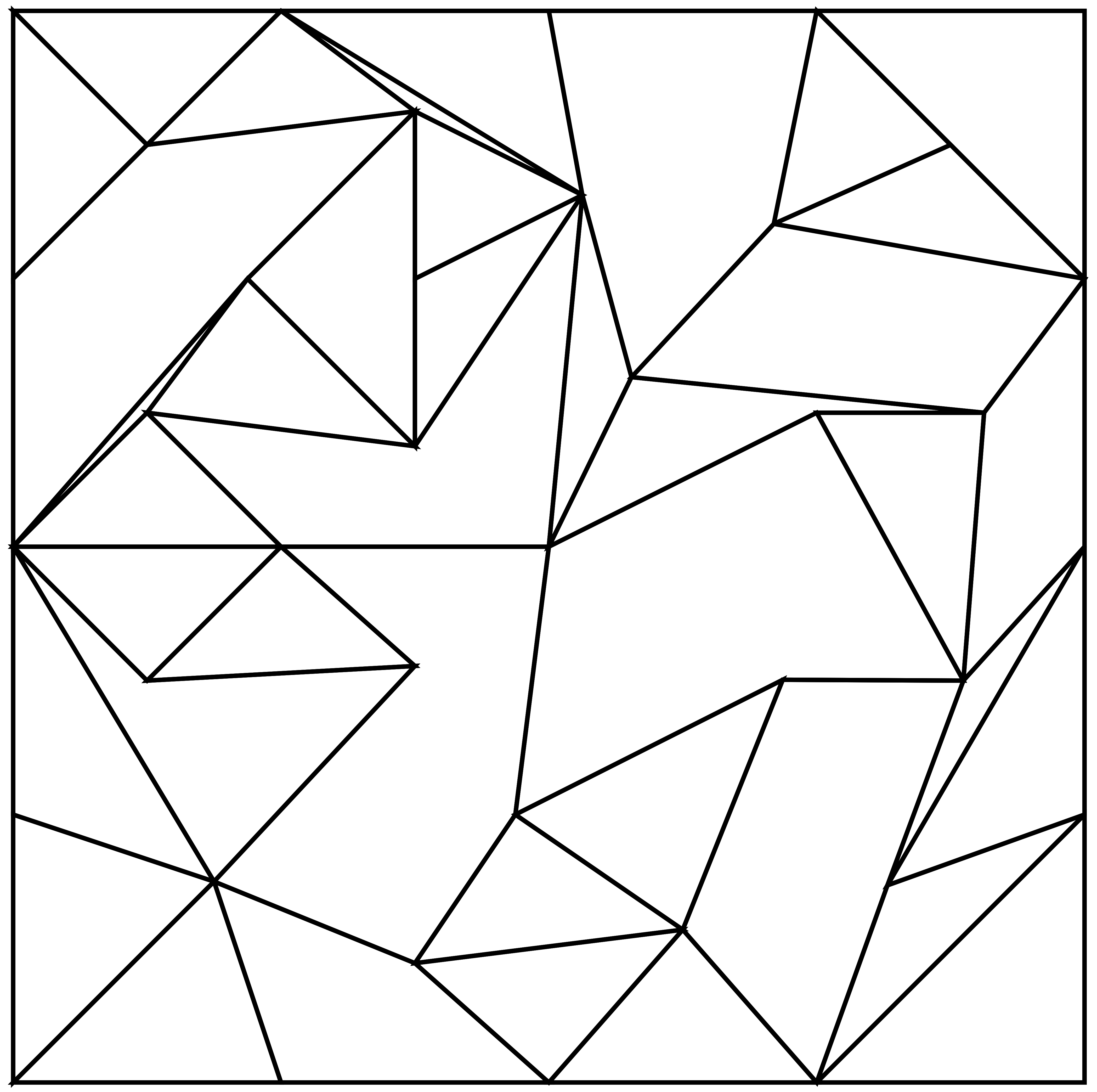
Non-zero coefficients on polygons

- Wardetzky & co-worker: all diagonals



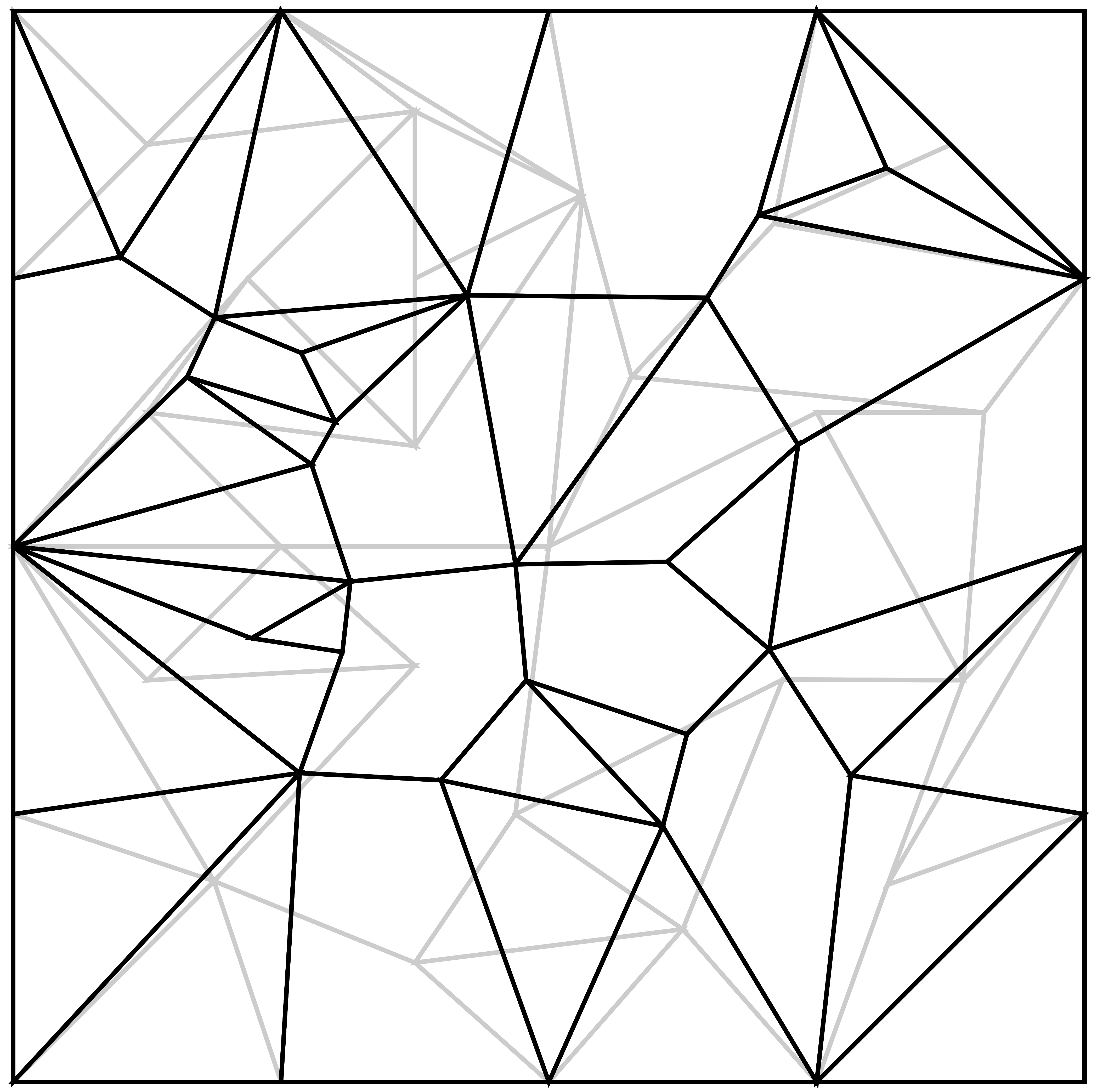
Non-zero coefficients on polygons

- Wardetzky & co-worker:
  all diagonals
- Considering all diagonals
  - Non-convex faces possible
  - No guarantee for embedding



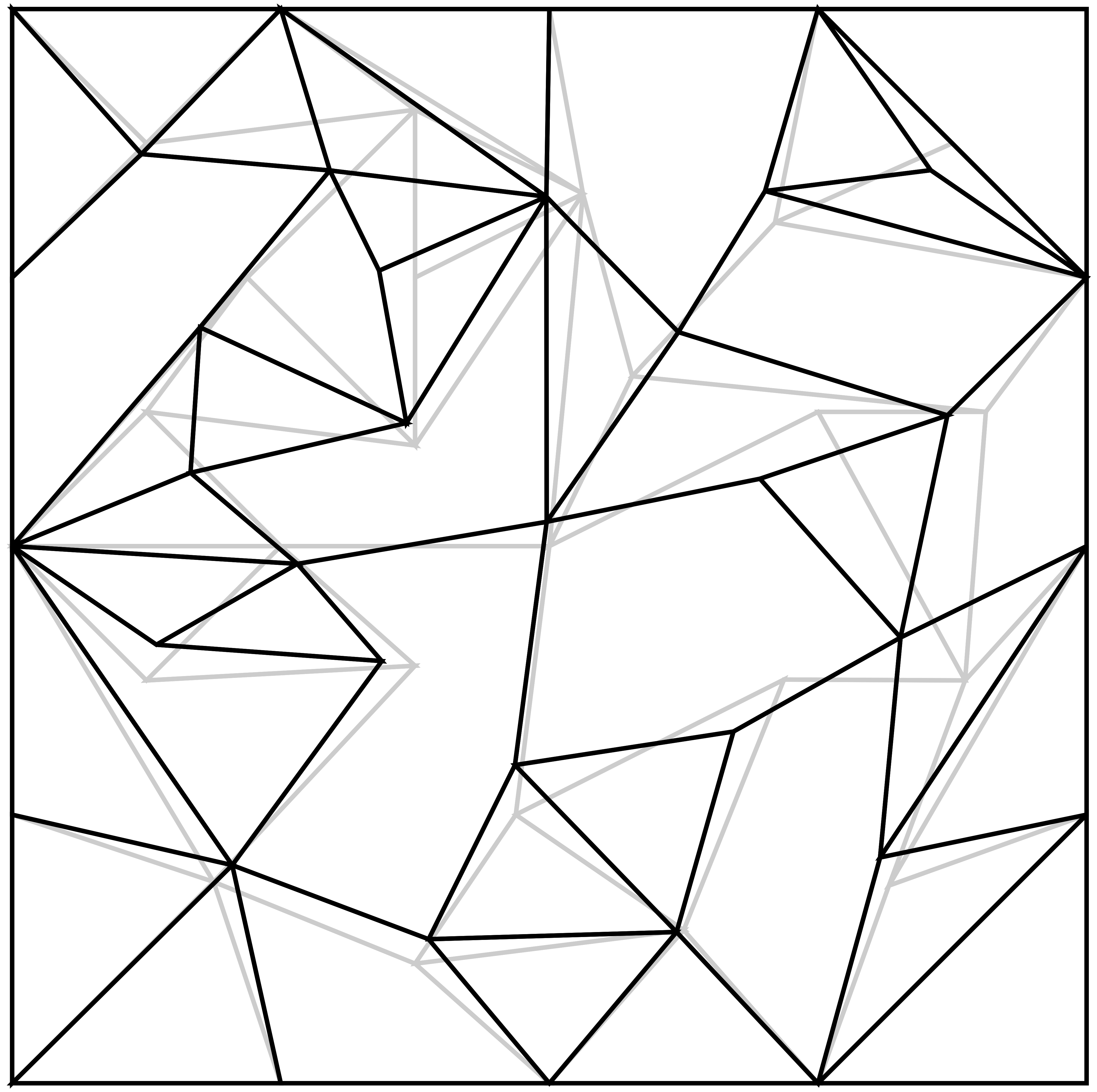
Non-zero coefficients on polygons

- Wardetzky & co-worker:
  all diagonals
- Considering all diagonals
  - Non-convex faces possible
  - No guarantee for embedding



Non-zero coefficients on polygons

- Wardetzky & co-worker: all diagonals
- Considering all diagonals
  - Non-convex faces possible
  - No guarantee for embedding



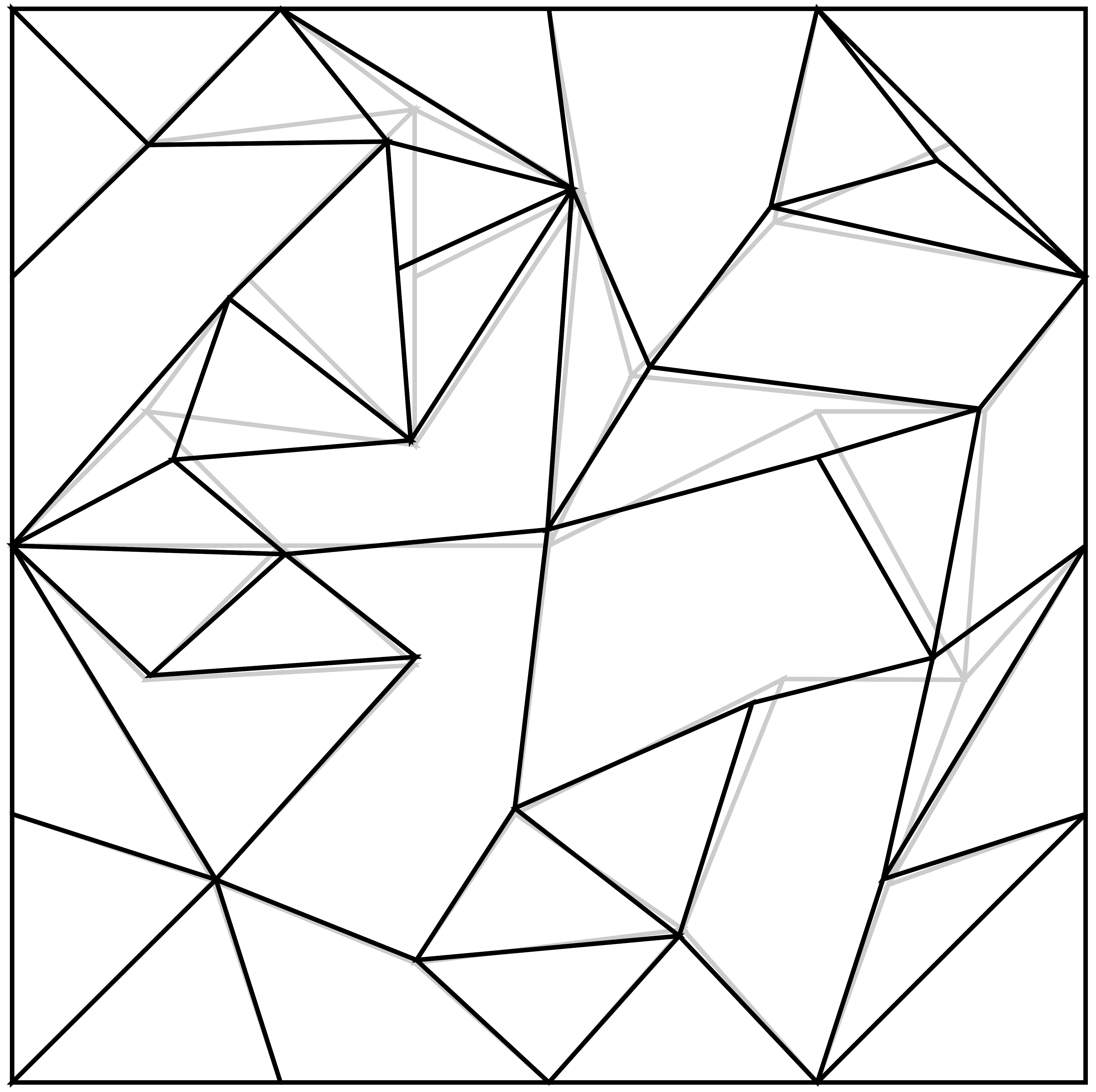
Non-zero coefficients on polygons

- Wardetzky & co-worker:
  all diagonals

- Considering all diagonals

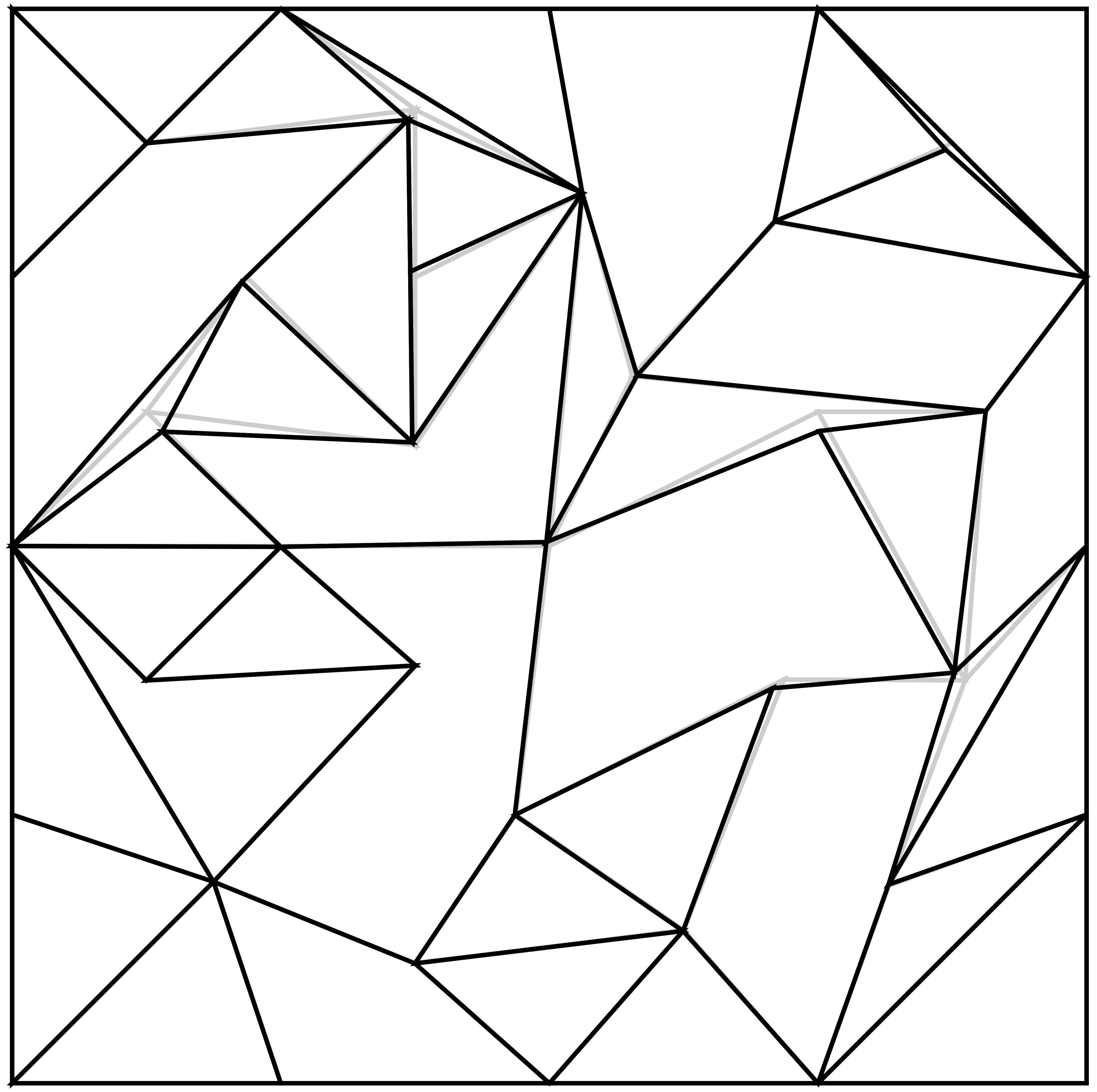
  - Non-convex faces possible

  - No guarantee for embedding



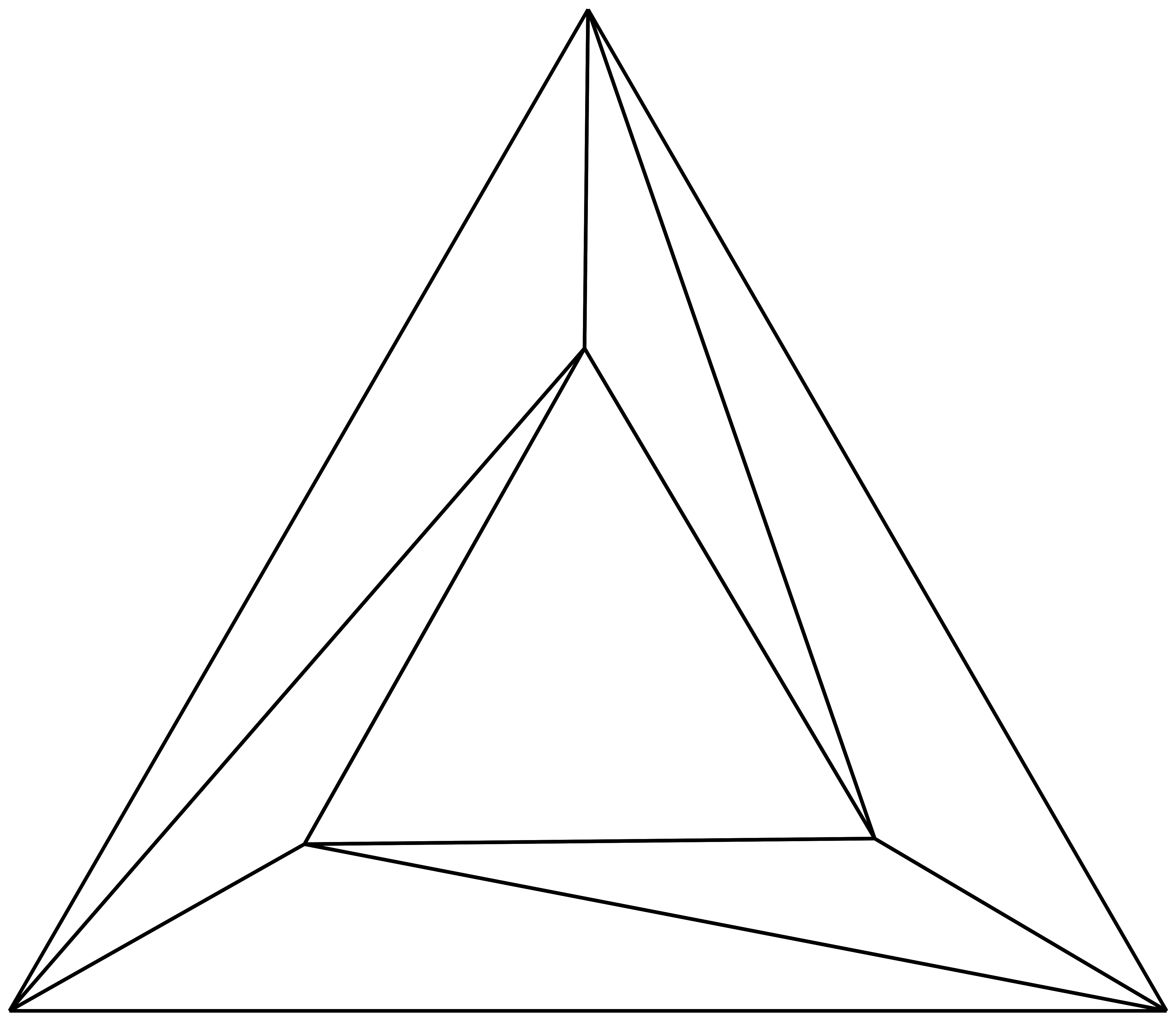
Non-zero coefficients on polygons

- Wardetzky & co-worker:
  all diagonals
- Considering all diagonals
  - Non-convex faces possible
  - No guarantee for embedding



Perfect Laplacians for Polygon Meshes

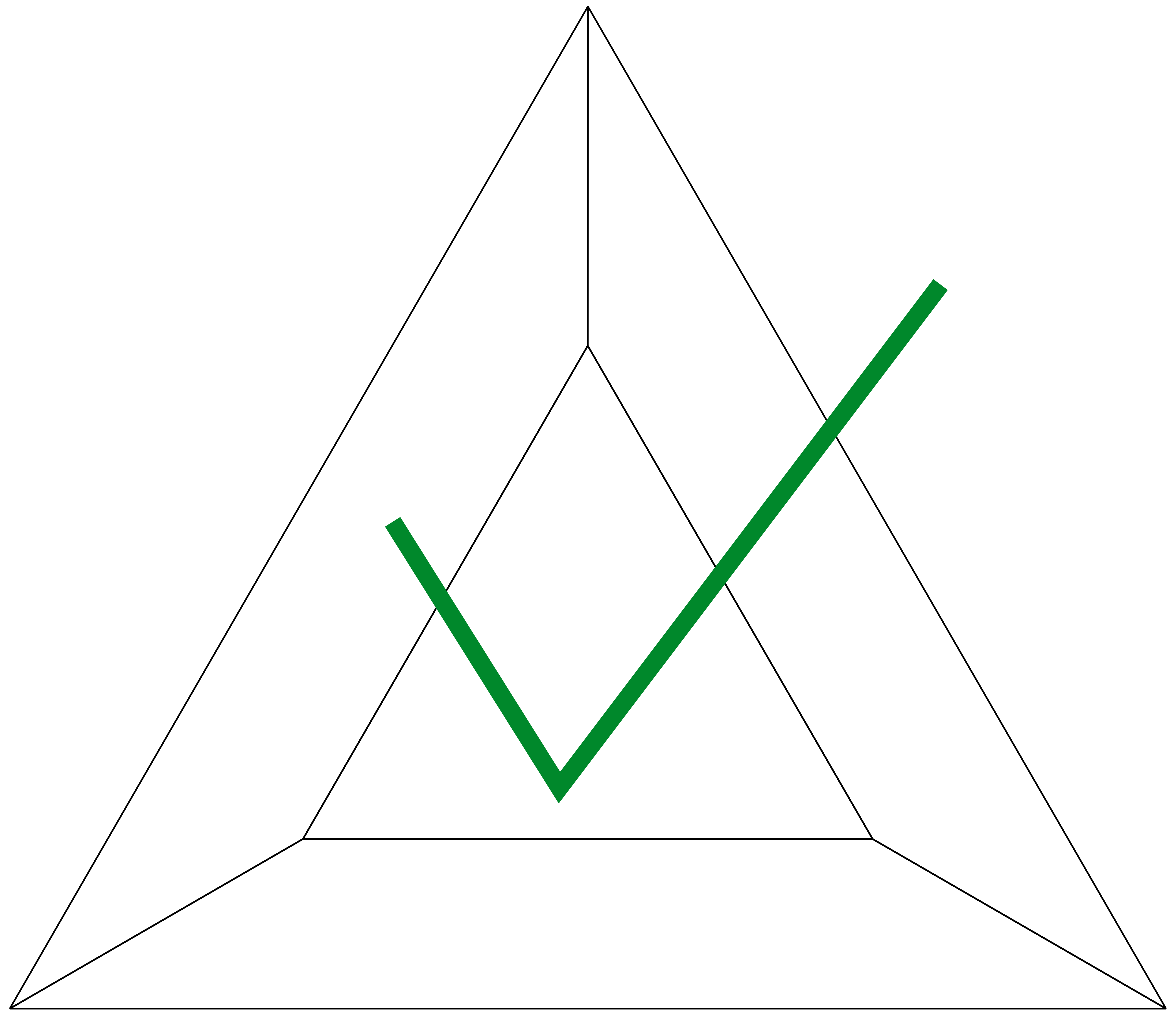
- Without diagonals
  - Weakly regular subdivision



Perfect Laplacian?

Perfect Laplacians for Polygon Meshes

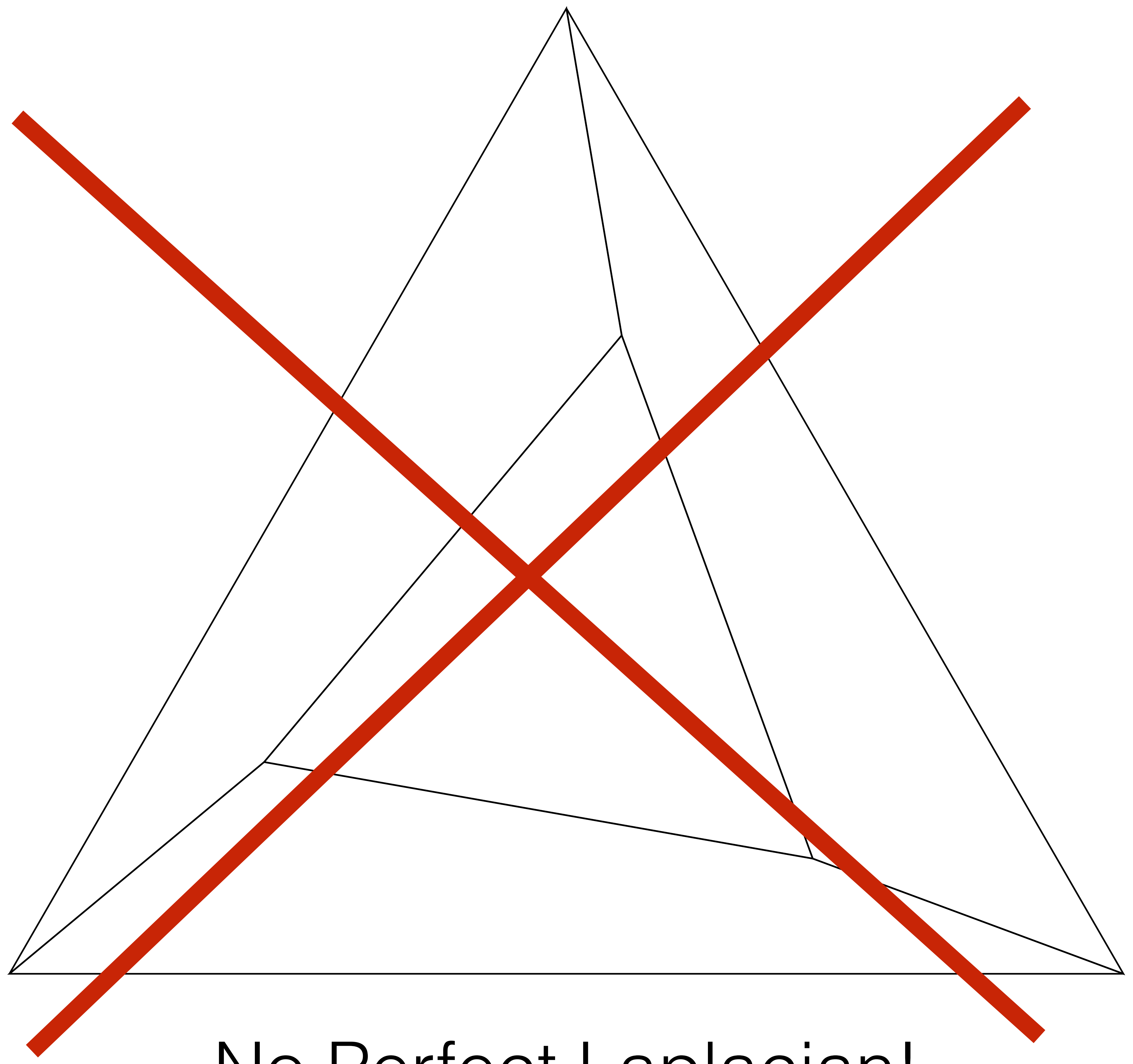
- Without diagonals
  - Weakly regular subdivision
    - A subset of the mesh is a regular subdivision



Perfect Laplacian!

Perfect Laplacians for Polygon Meshes

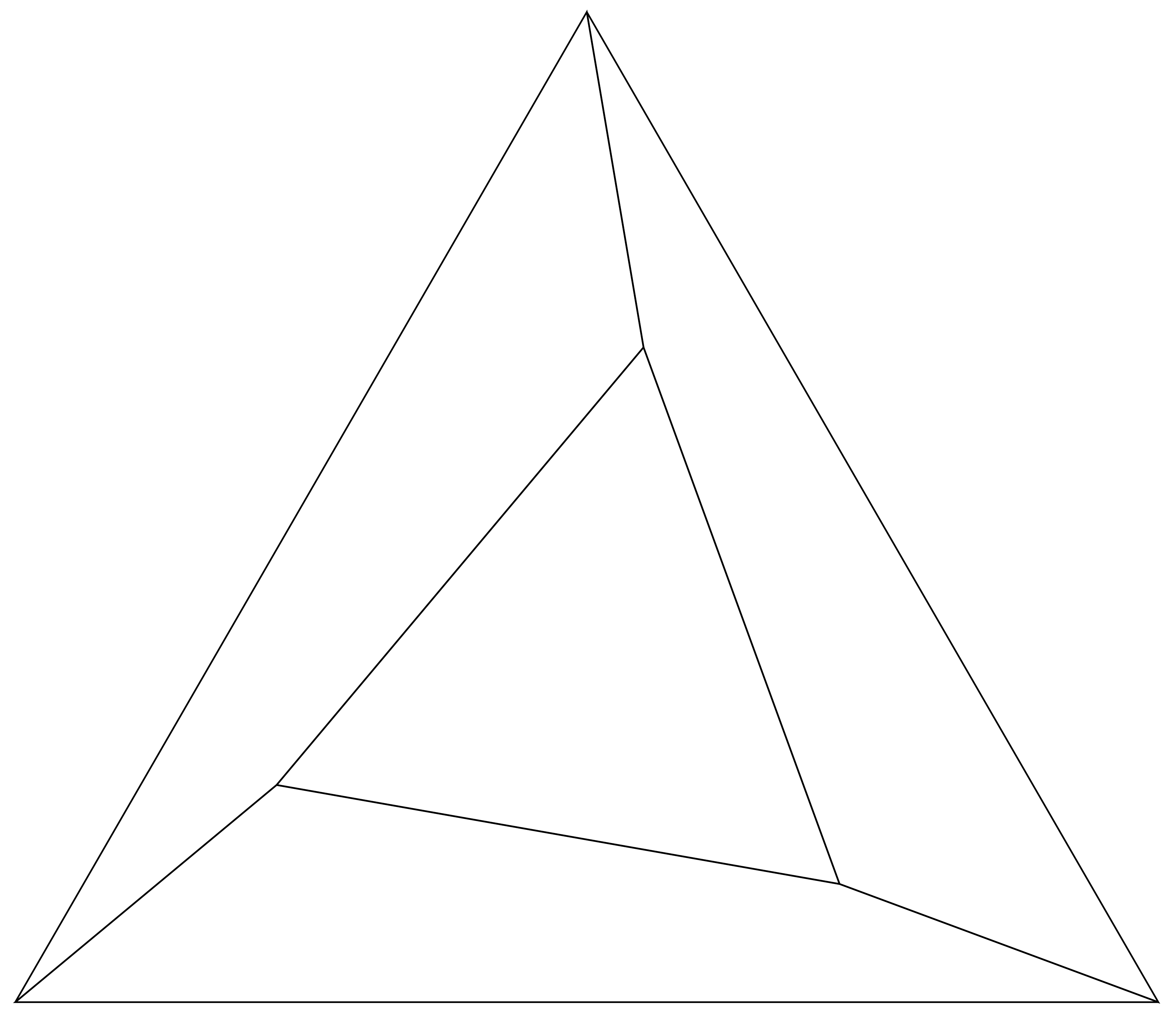
- Without diagonals
    - Weakly regular subdivision
        - A subset of the mesh is a regular subdivision



No Perfect Laplacian!

Perfect Laplacians for Polygon Meshes

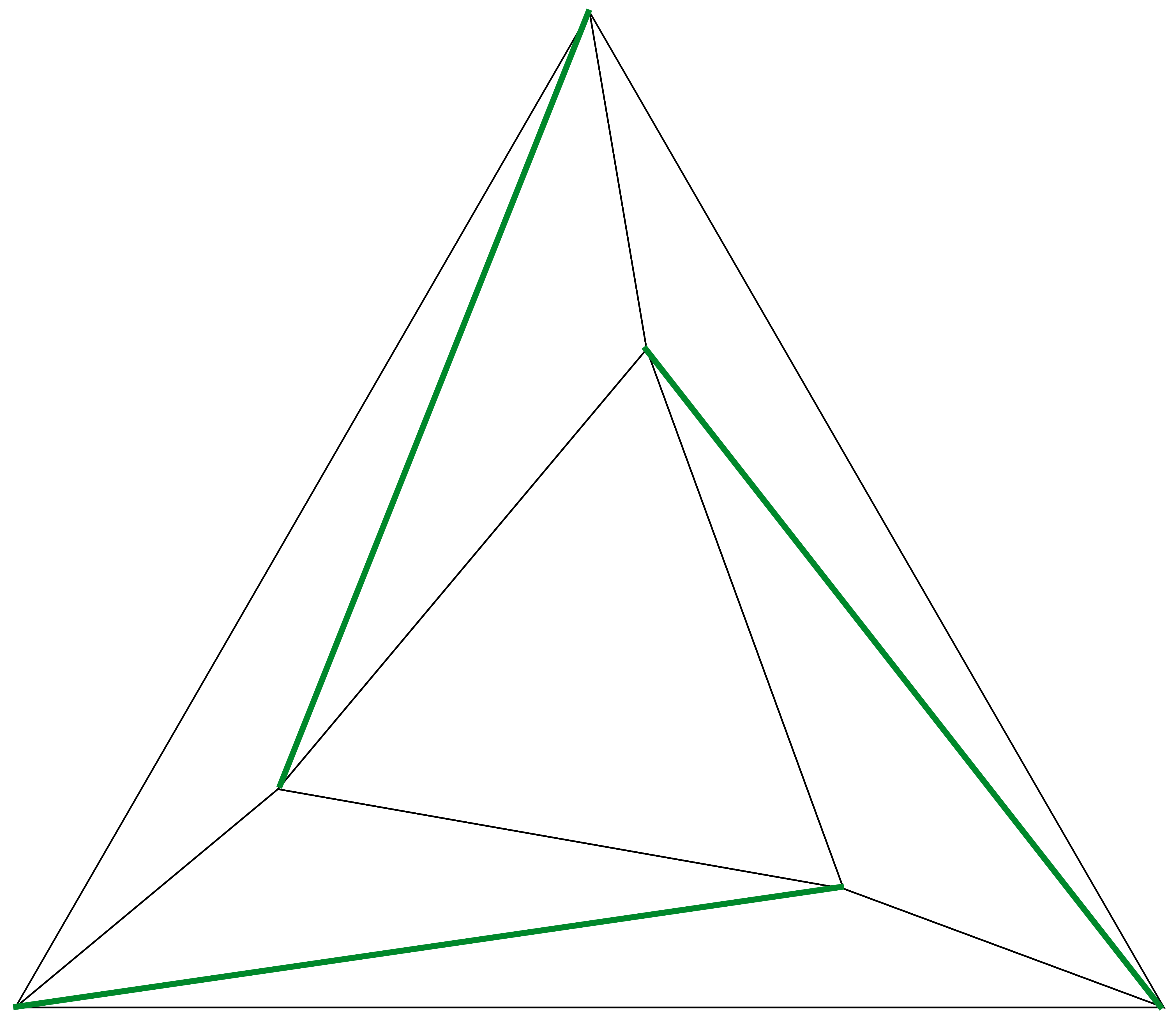
- With diagonals
    - Can be refined into a regular subdivision



Perfect Laplacian?

Perfect Laplacians for Polygon Meshes

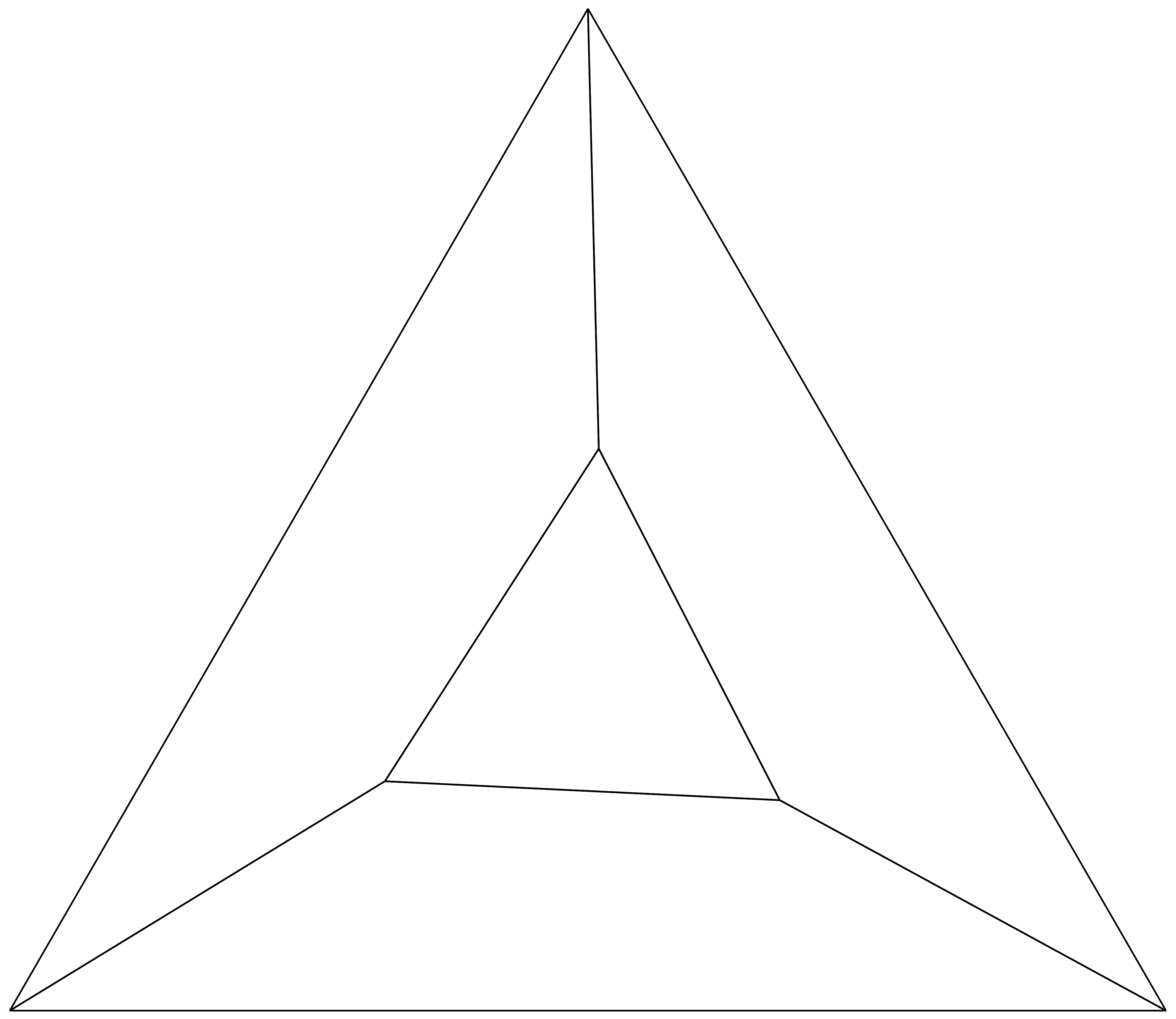
- With diagonals
  - Can be refined into a regular subdivision



Perfect Laplacian!

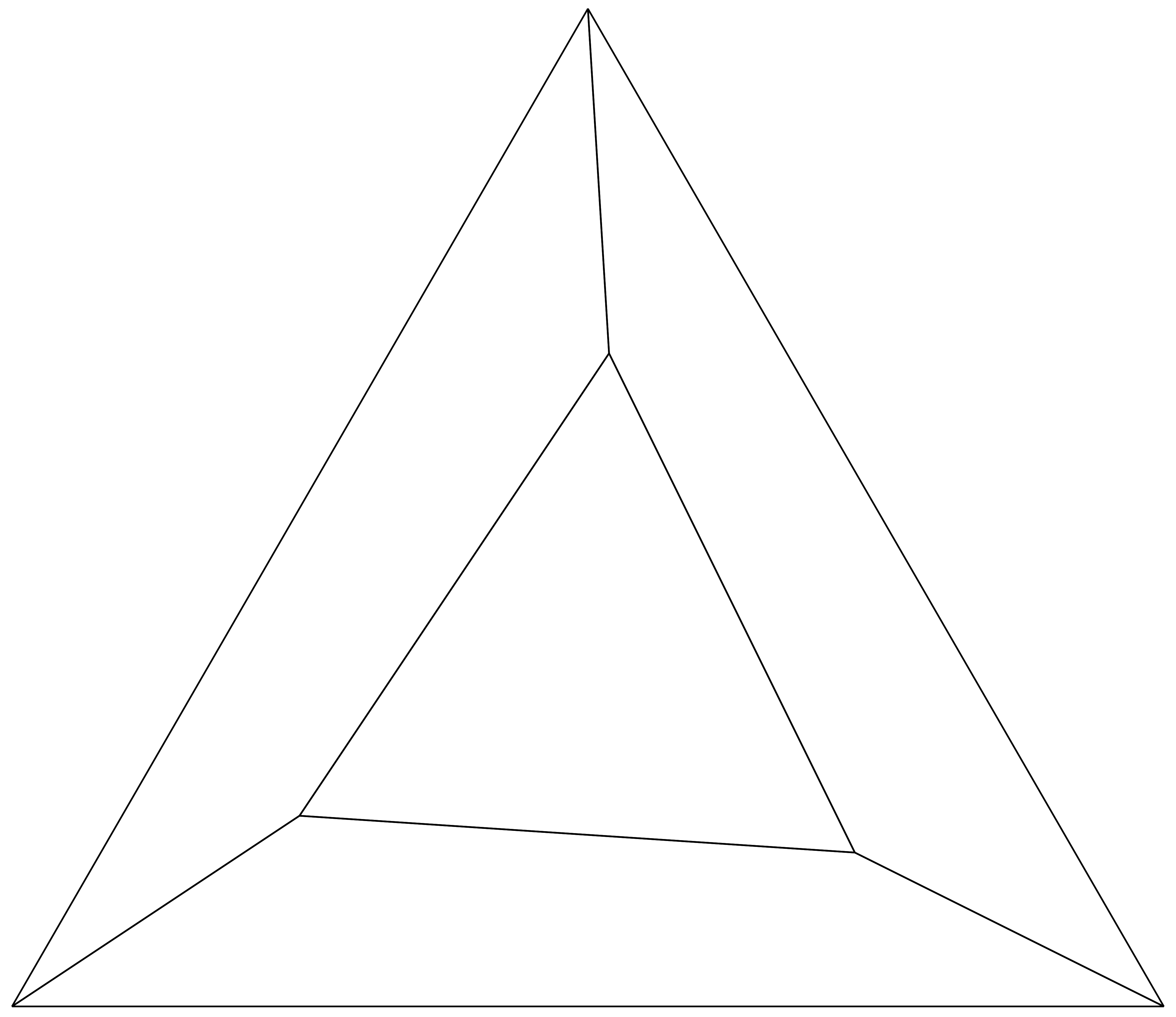
Perfect Laplacians for Polygon Meshes

- With diagonals
  - Can be refined into a regular subdivision



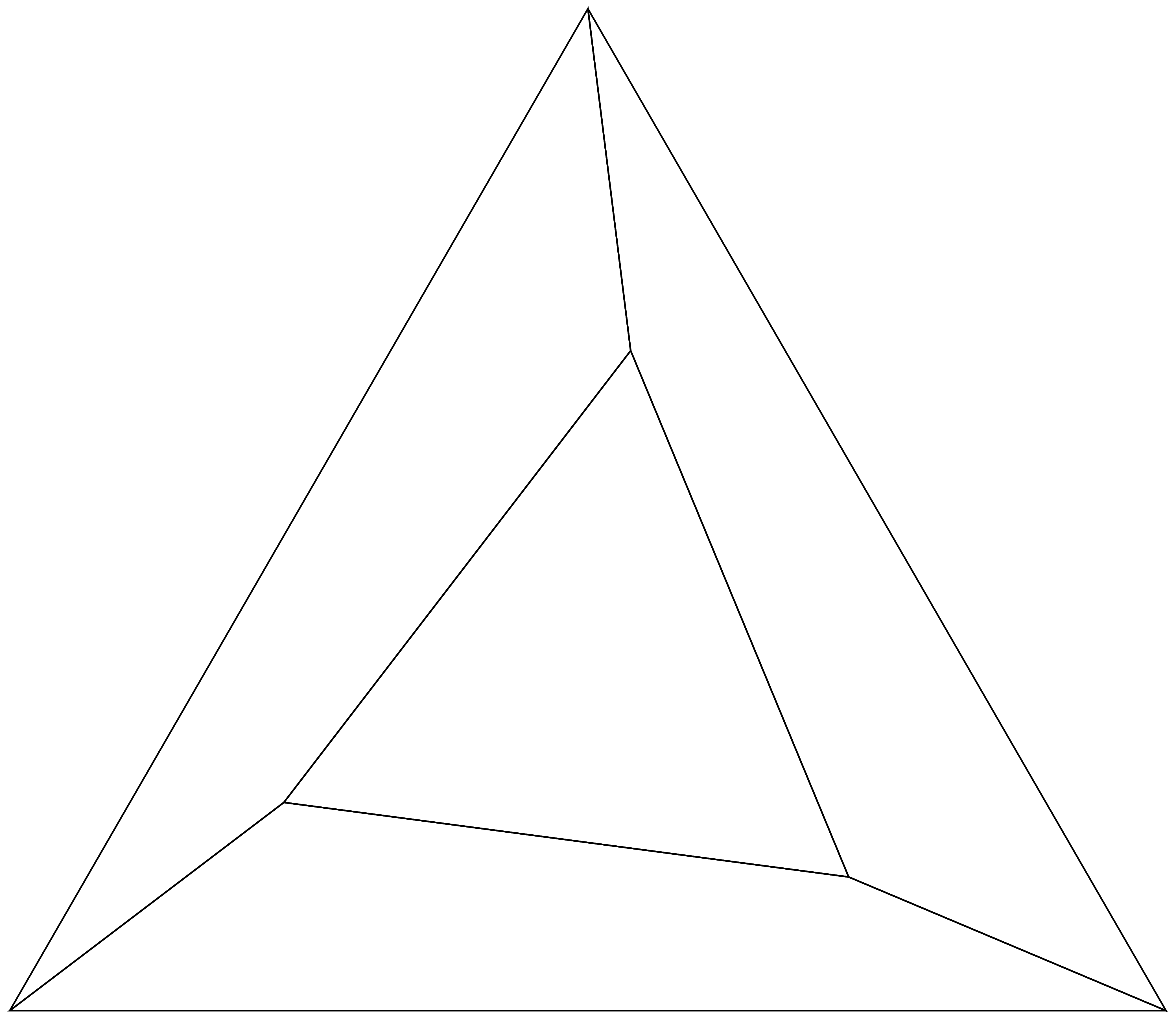
Perfect Laplacians for Polygon Meshes

- With diagonals
  - Can be refined into a regular subdivision



Perfect Laplacians for Polygon Meshes

- With diagonals
  - Can be refined into a regular subdivision
  - There are reasons not to use diagonals



Take home message: Compute perfect Laplacians

- Given a polygon mesh
- Provides perfect Laplacian if possible
- Otherwise compromises on linear precision
    - Finds a subset that admits perfect Laplacian
        - Other edges receive zero weight
        - May use diagonals in faces
- Open: constraints for meshes in 3d