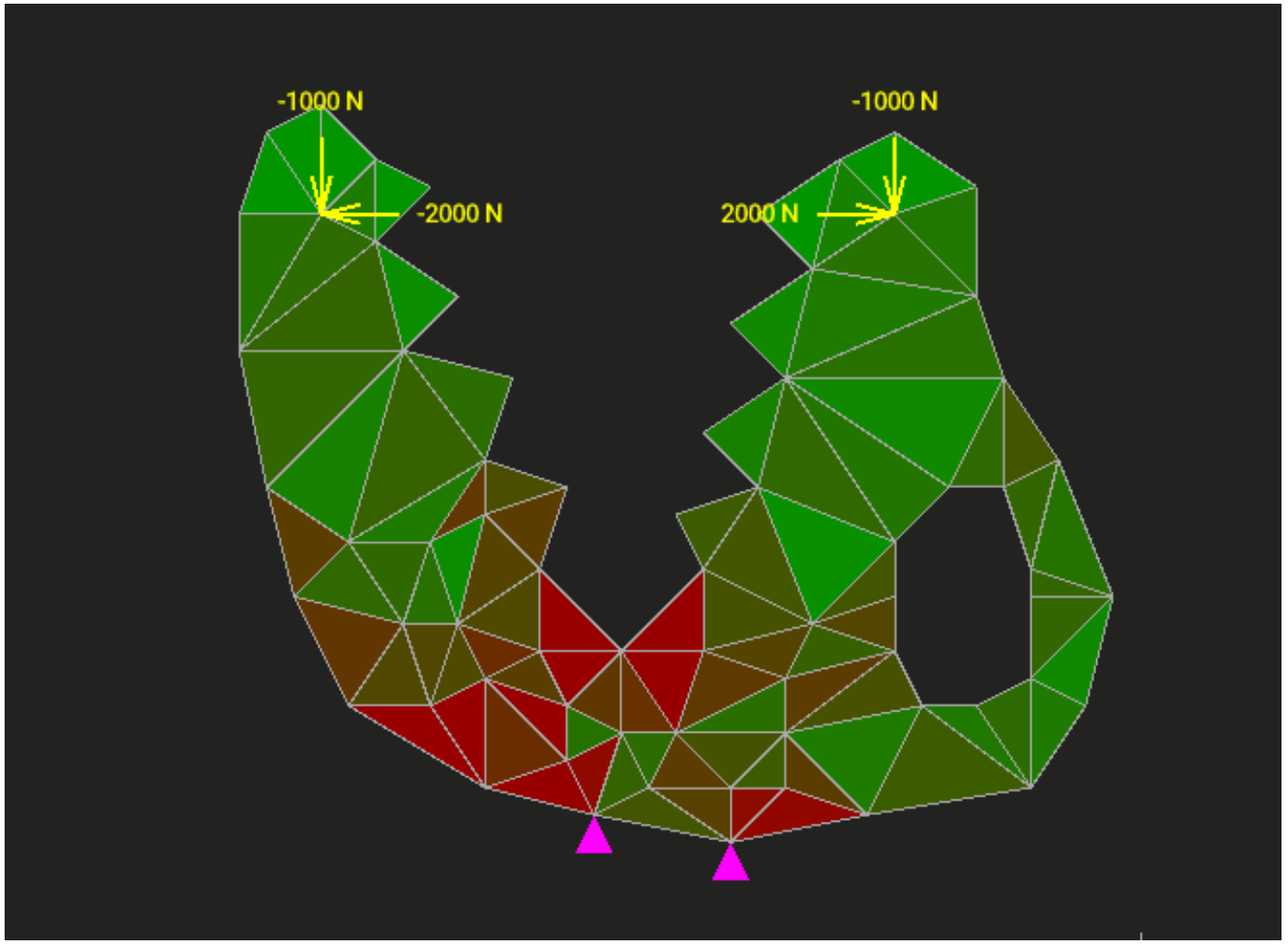


Finite Element Method:

A Homemade 2D Solver

Vasiliy Tereshkov

https://github.com/vtereshkov/fem

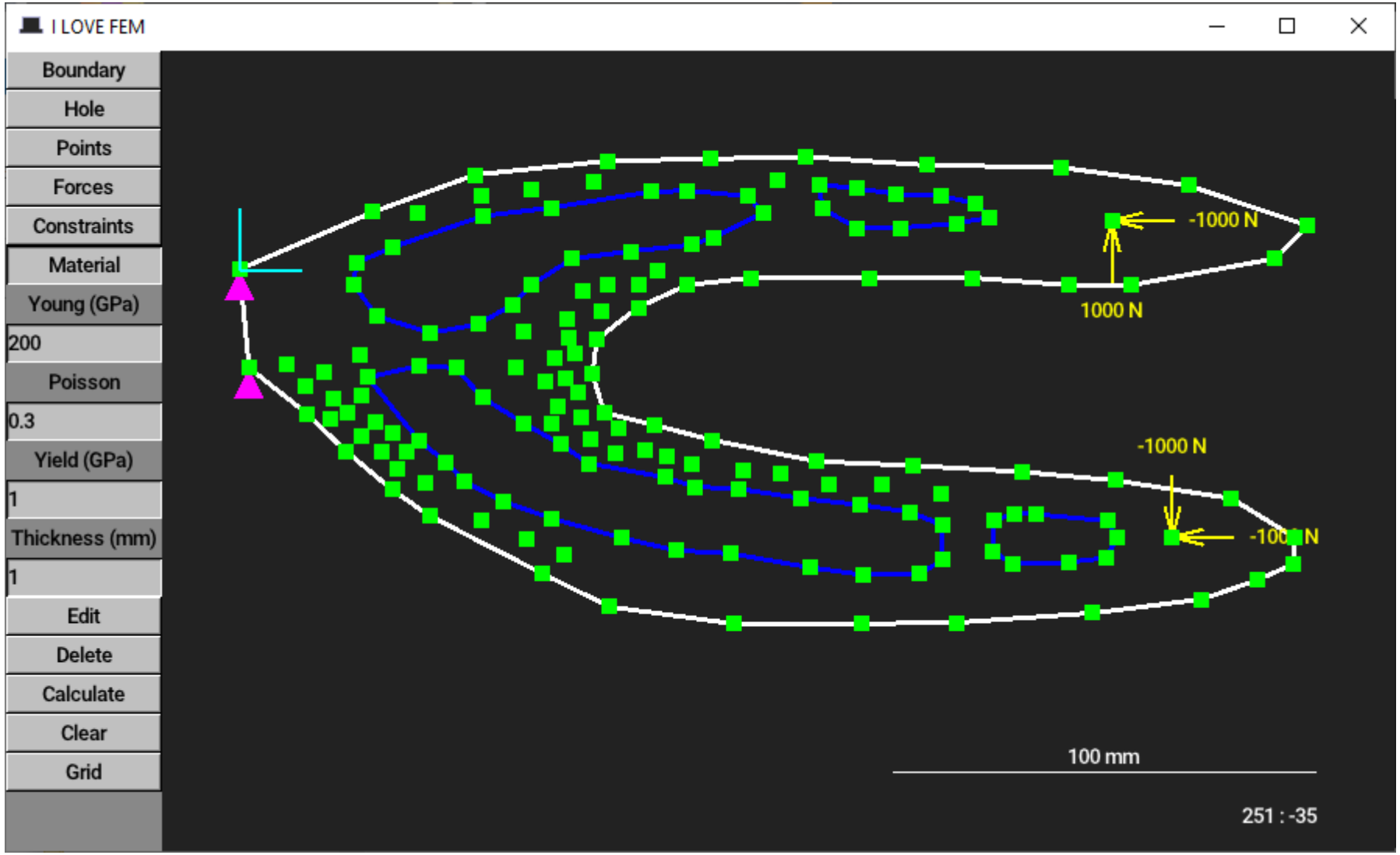
Topics

- Editor
- Mesh generation
- FEM solver
- Visualization
- Self-check
- Implementation

Editor: Requirements

- **2D plate boundary**: one non-self-intersecting polygon
- **Holes**: only polygons, no circles
- **Extra points** for manual mesh refinement
- **Constraints**: each fixing a point in both X and Y
- **External loads**: only forces, no torques
- **Material**: Young modulus, Poisson ratio, max stress

Editor: GUI


I LOVE FEM
Boundary
Hole
Points
Forces
Constraints
Material
Young (GPa)
200
Poisson
0.3
Yield (GPa)
1
Thickness (mm)
1
Edit
Delete
Calculate
Clear
Grid
-1000 N
1000 N
-1000 N
100 mm
251 : -35

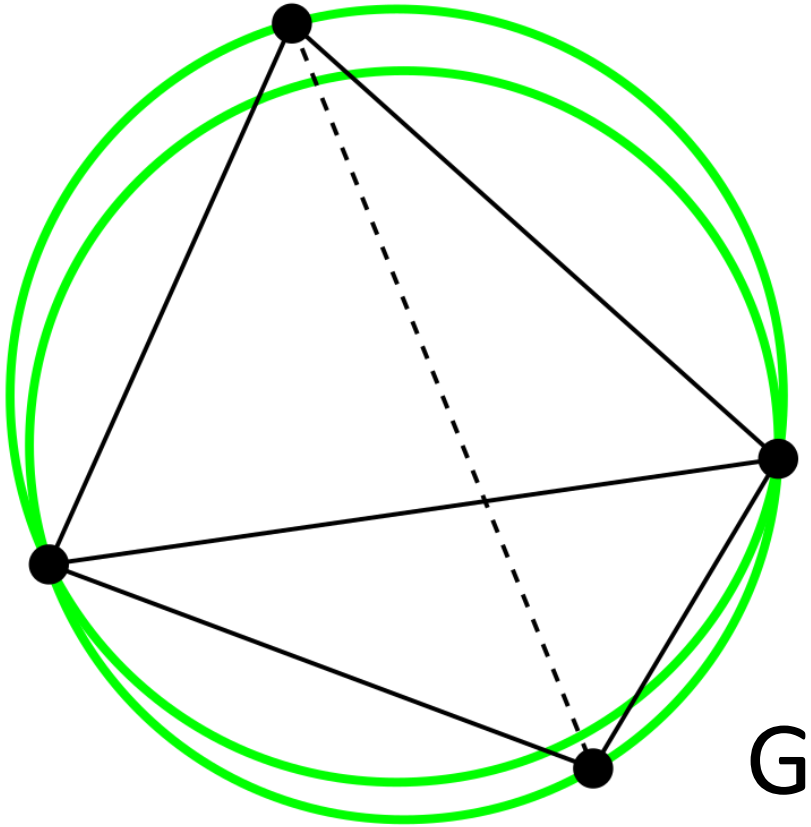
Editor: Clockwise or counter-clockwise?

- **Boundary**: counter-clockwise
- **Holes**: clockwise

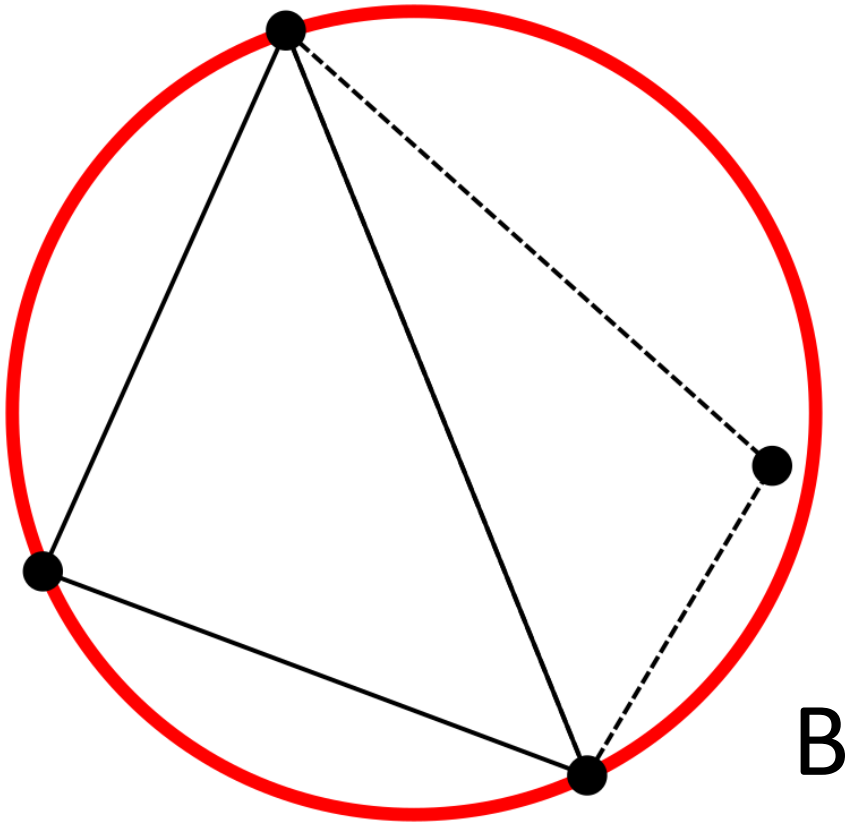
```
fn (ed: ^Editor) isClockwise(poly: Poly): bool {
    sum := 0.0
    for i, pt1 := 0, ed.pts[poly.vert[len(poly.vert) - 1]]; i < len(poly.vert); i++ {
        pt2 := ed.pts[poly.vert[i]]
        sum += (pt2.x - pt1.x) * (pt2.y + pt1.y)
        pt1 = pt2
    }
    return sum > 0.0
}
```

Mesh generation: Delaunay triangulation

How to avoid extremely long and thin triangles?



Good



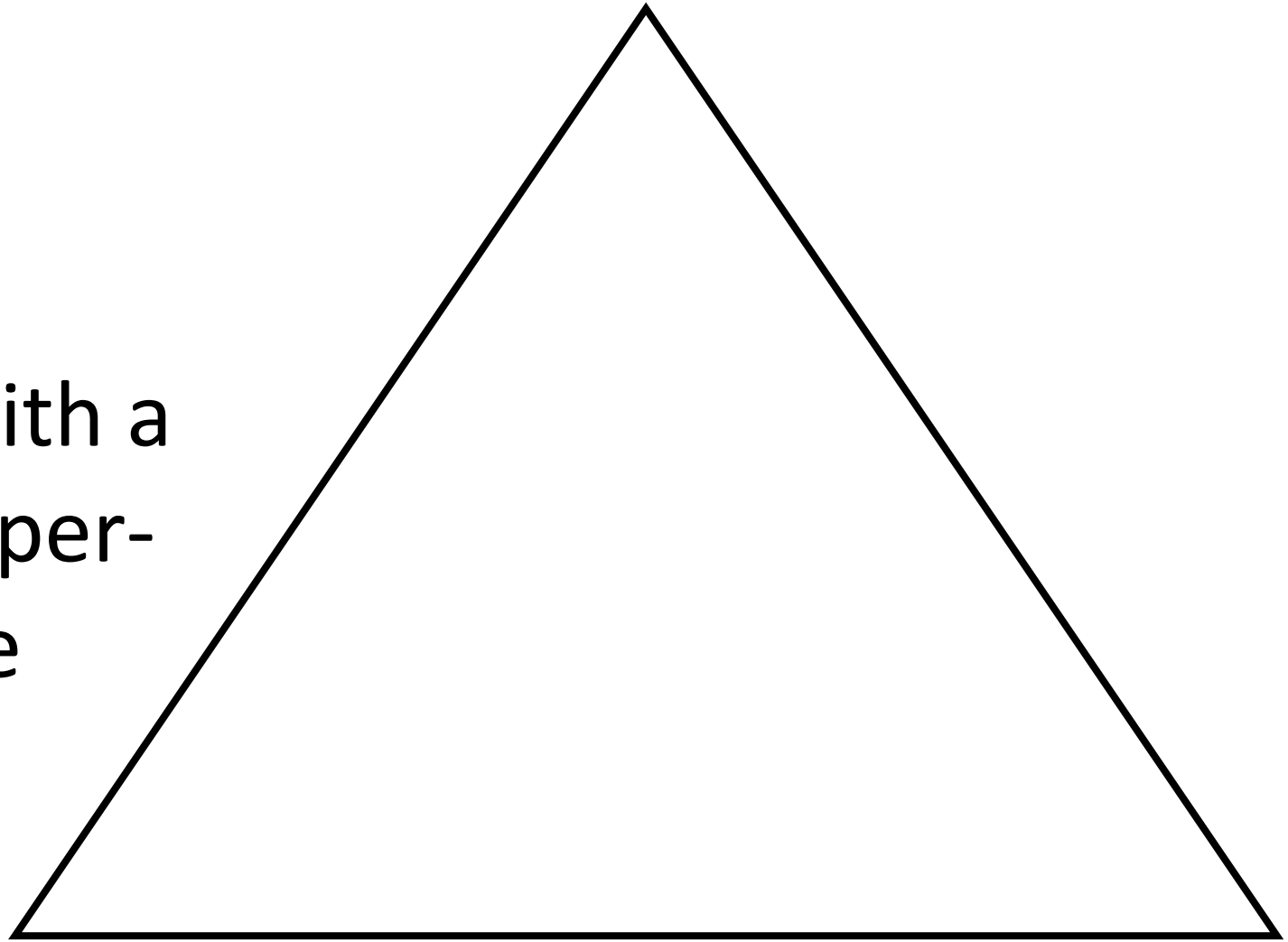
Point inside

Bad

Source: https://en.wikipedia.org/wiki/Delaunay_triangulation

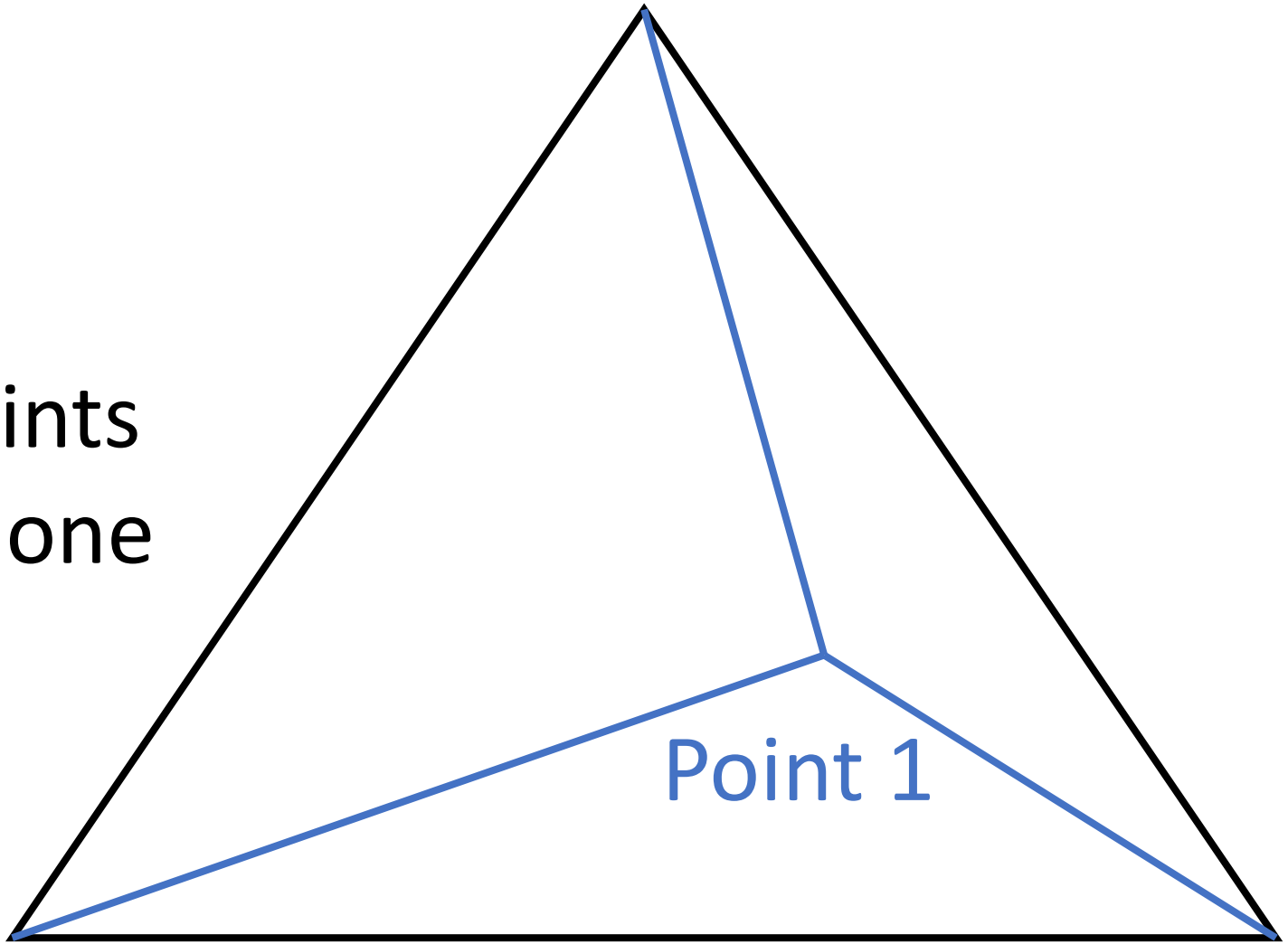
Mesh generation: Bowyer-Watson algorithm

Start with a fake super-triangle



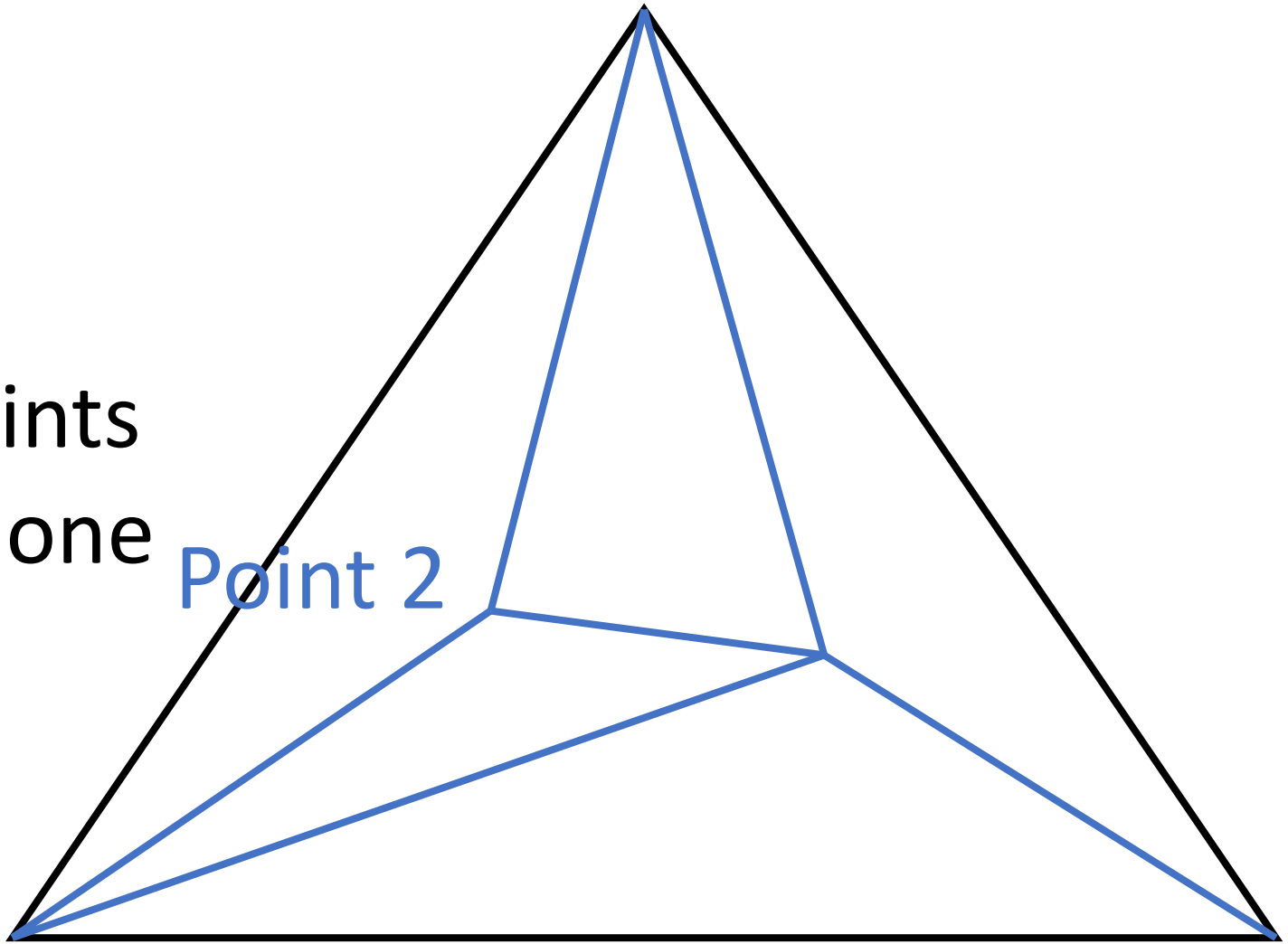
Mesh generation: Bowyer-Watson algorithm

Add points
one by one



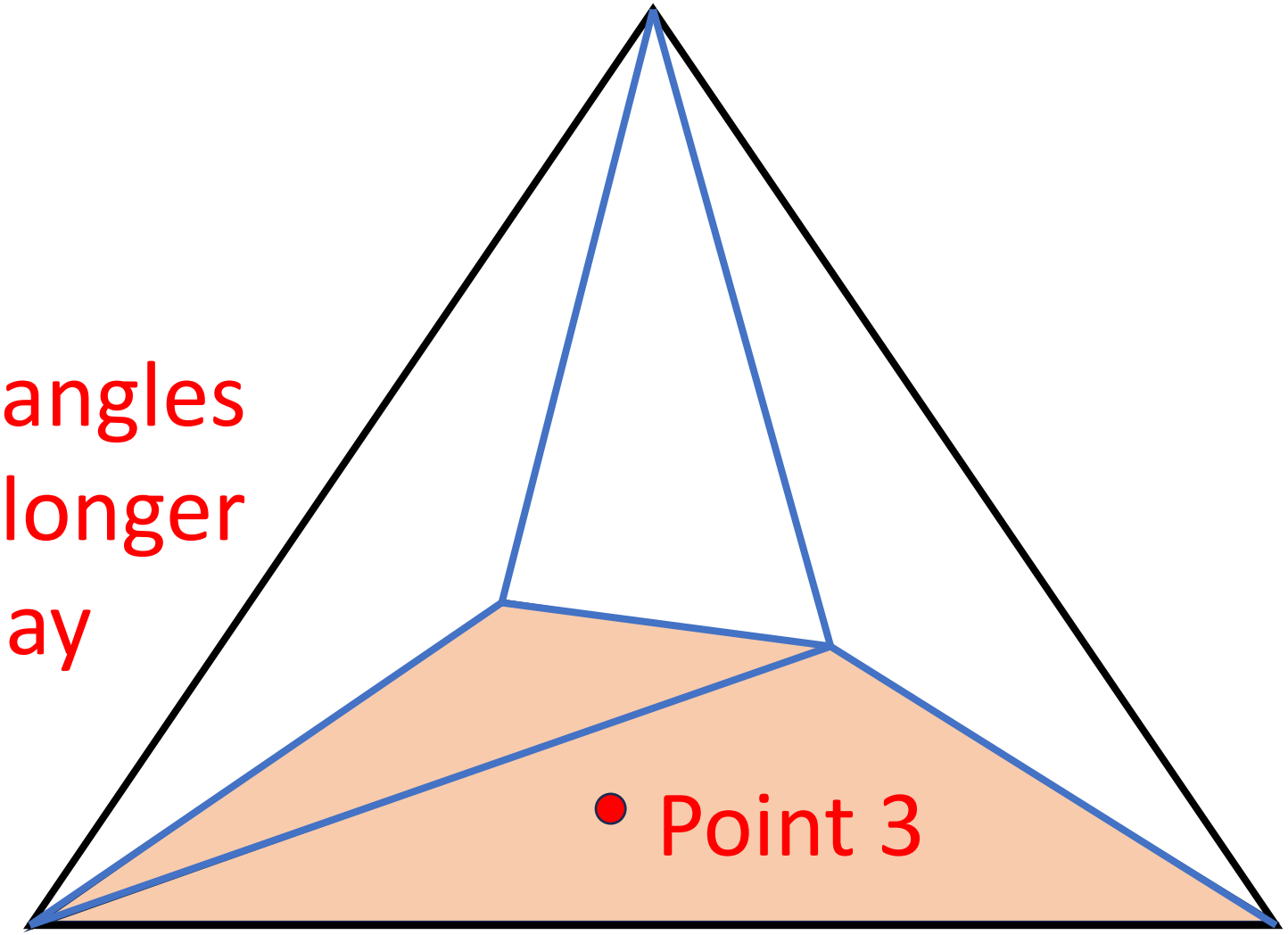
Mesh generation: Bowyer-Watson algorithm

Add points
one by one



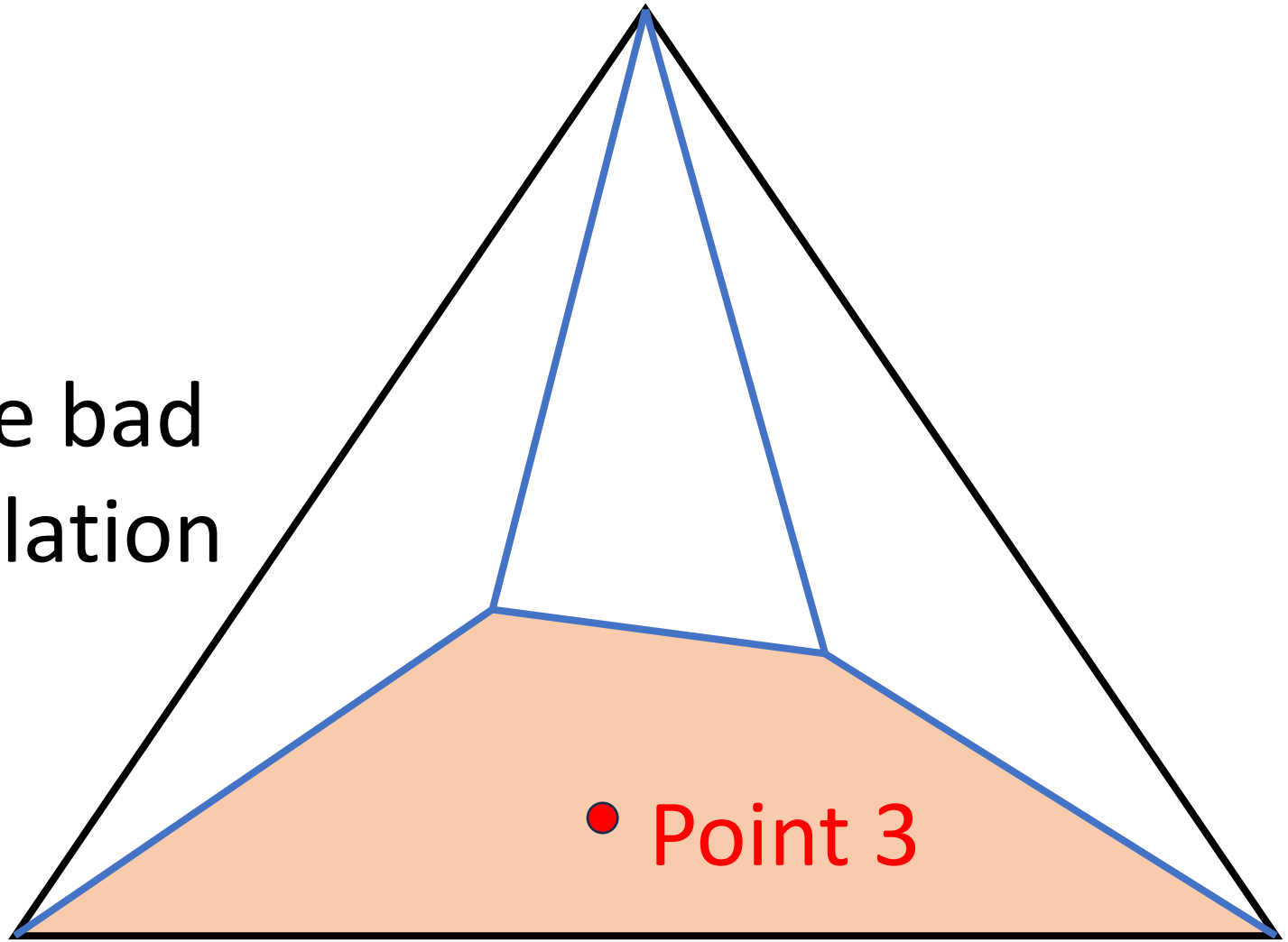
Mesh generation: Bowyer-Watson algorithm

Two triangles are no longer Delaunay



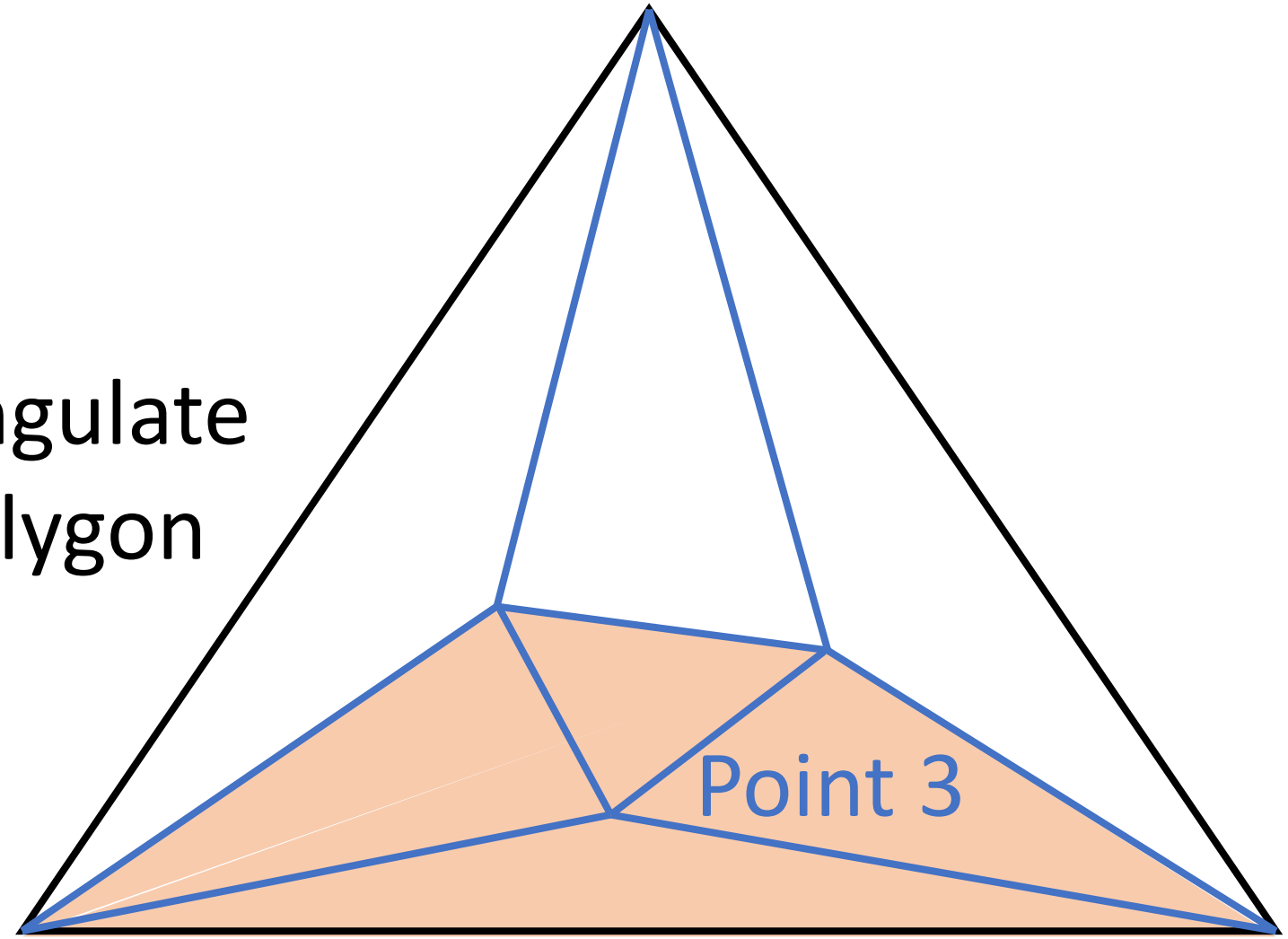
Mesh generation: Bowyer-Watson algorithm

Remove bad triangulation



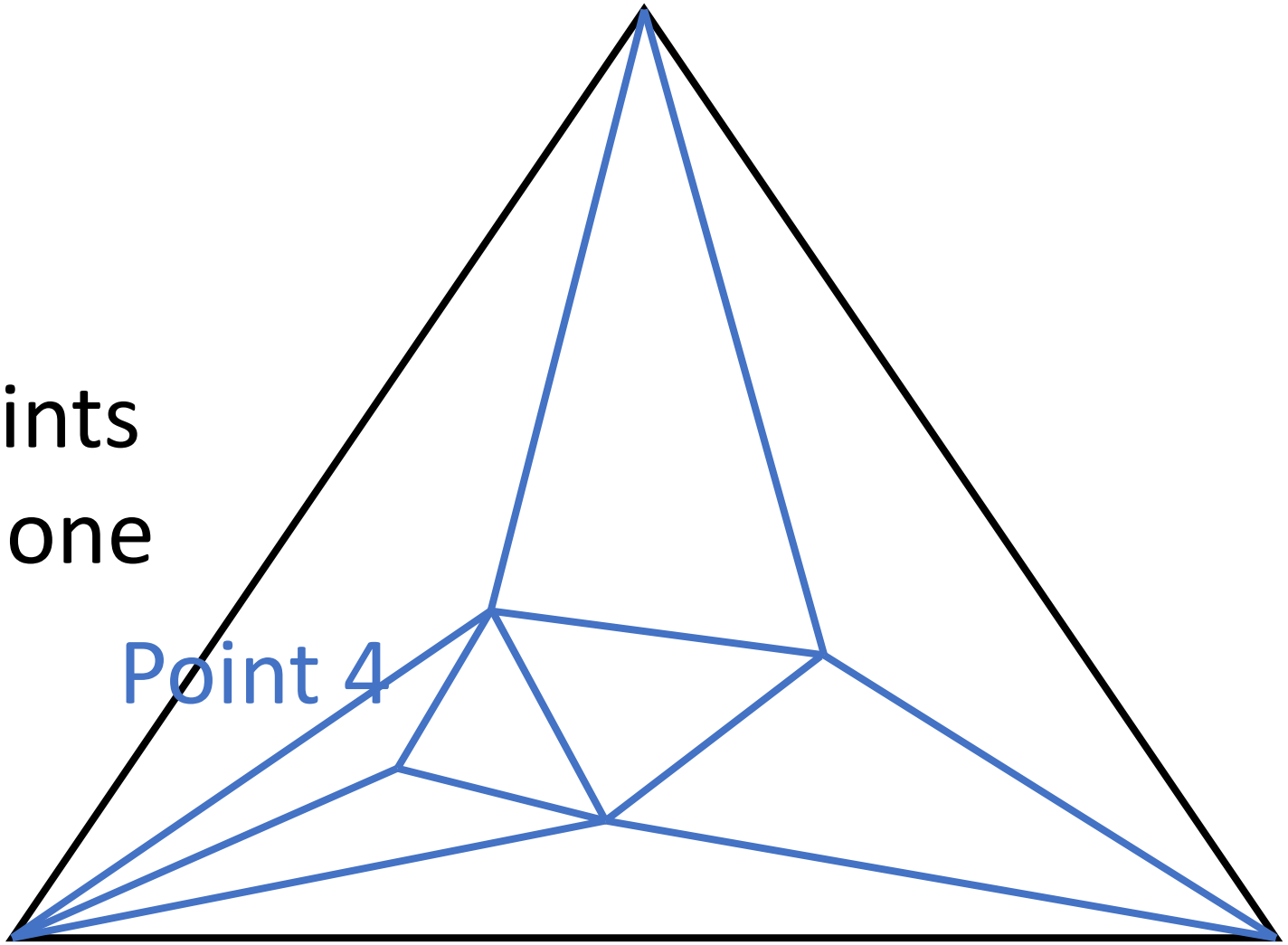
Mesh generation: Bowyer-Watson algorithm

Retriangulate
bad polygon



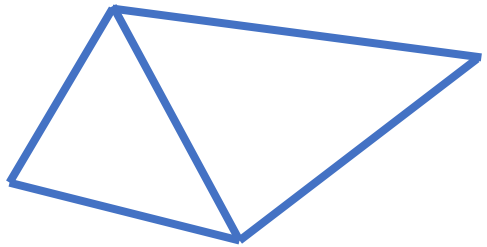
Mesh generation: Bowyer-Watson algorithm

Add points
one by one

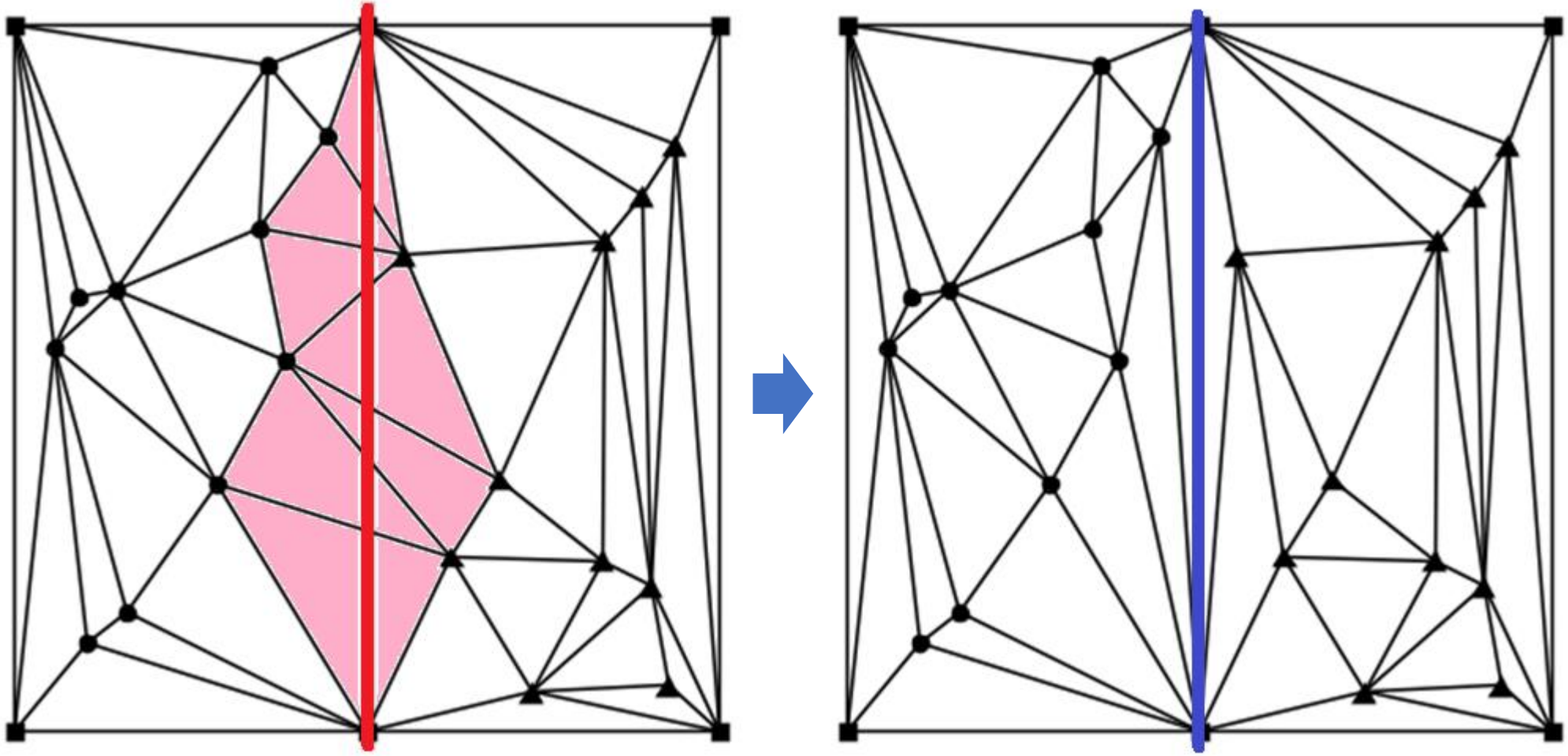


Mesh generation: Bowyer-Watson algorithm

Remove the
super-triangle

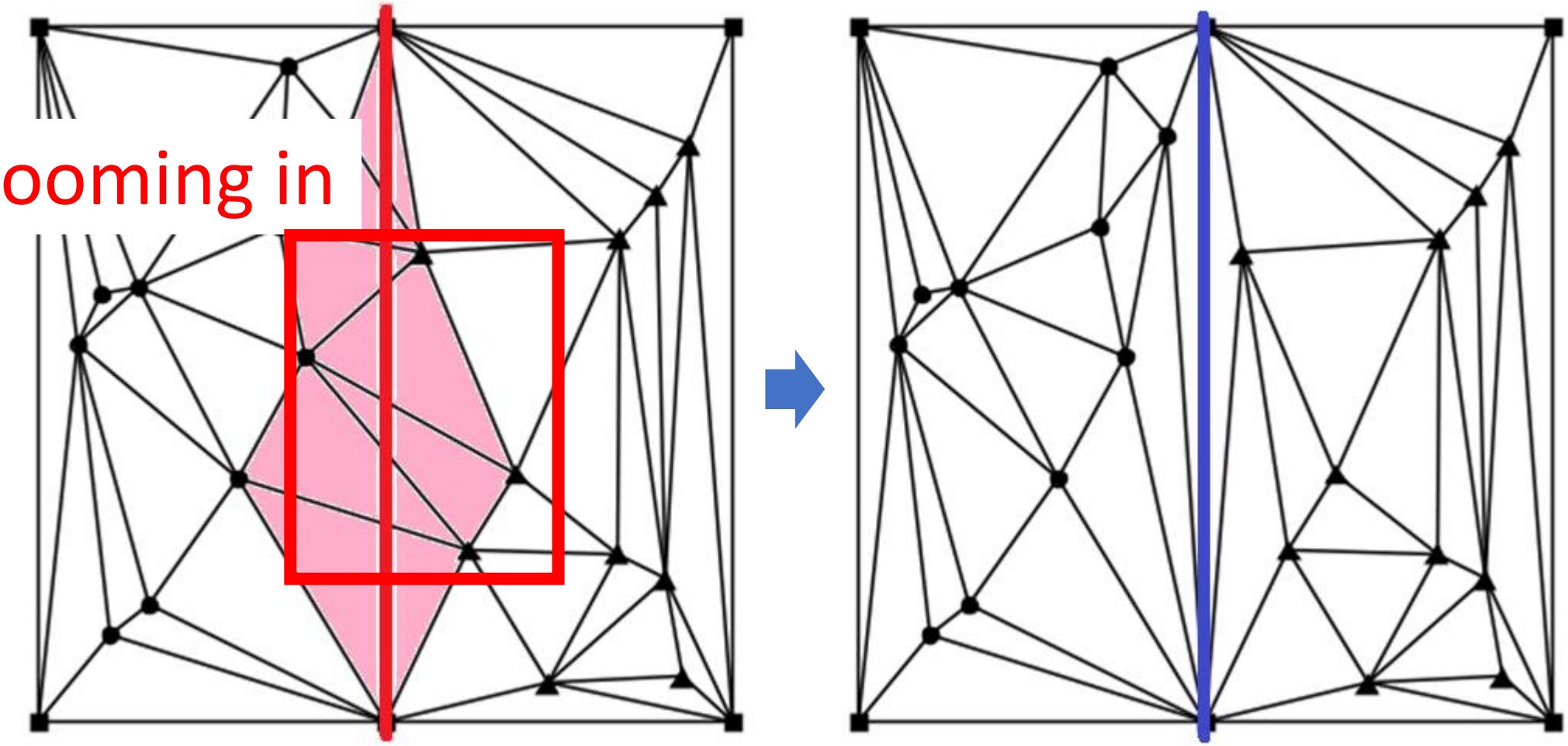


Mesh generation: Missing edges



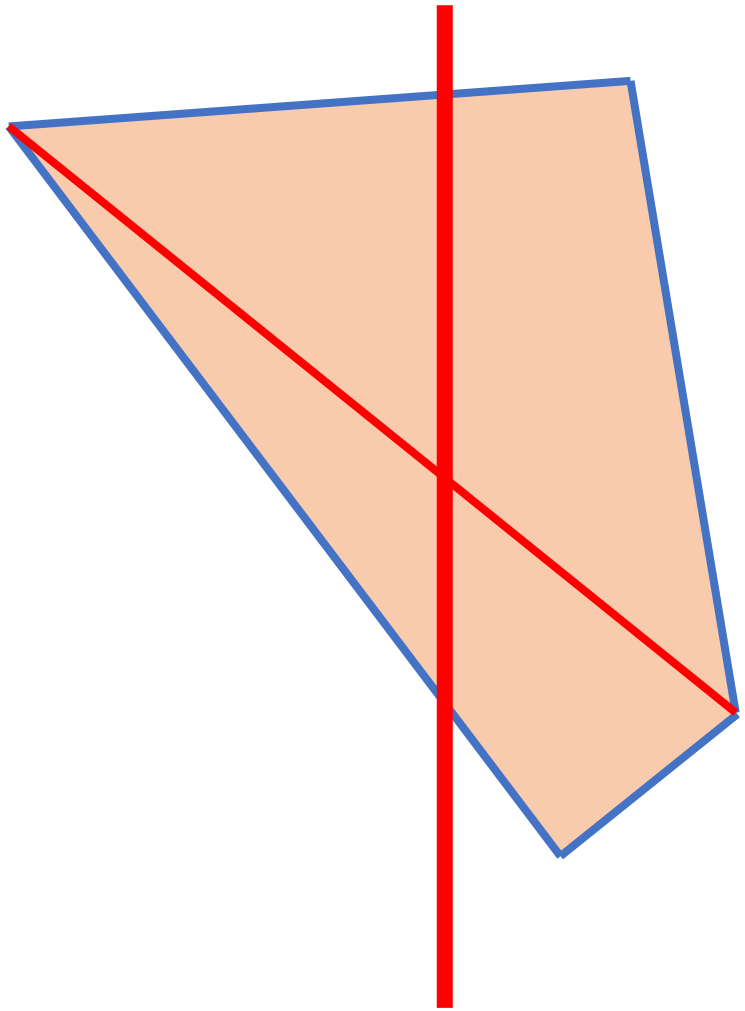
Source: https://gwlucastrig.github.io/TinfourDocs/DelaunayIntroCDT/index.html

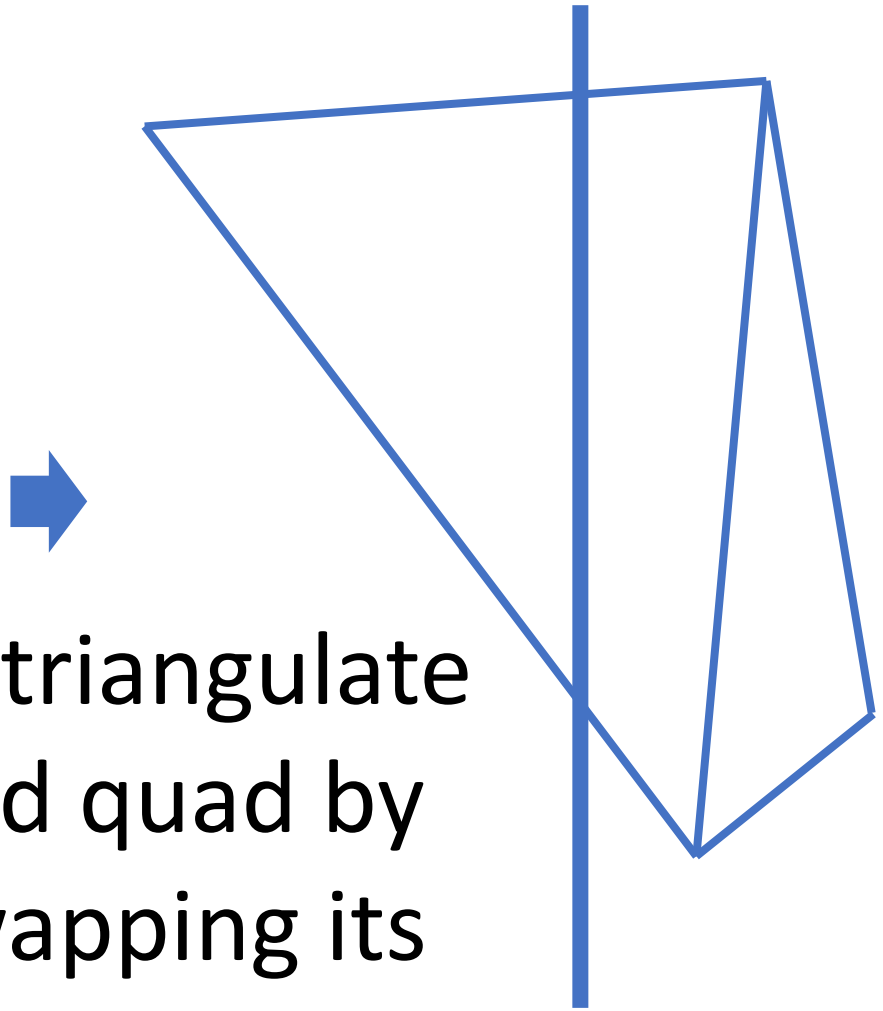
Mesh generation: Missing edges



Source: https://gwlucastrig.github.io/TinfourDocs/DelaunayIntroCDT/index.html

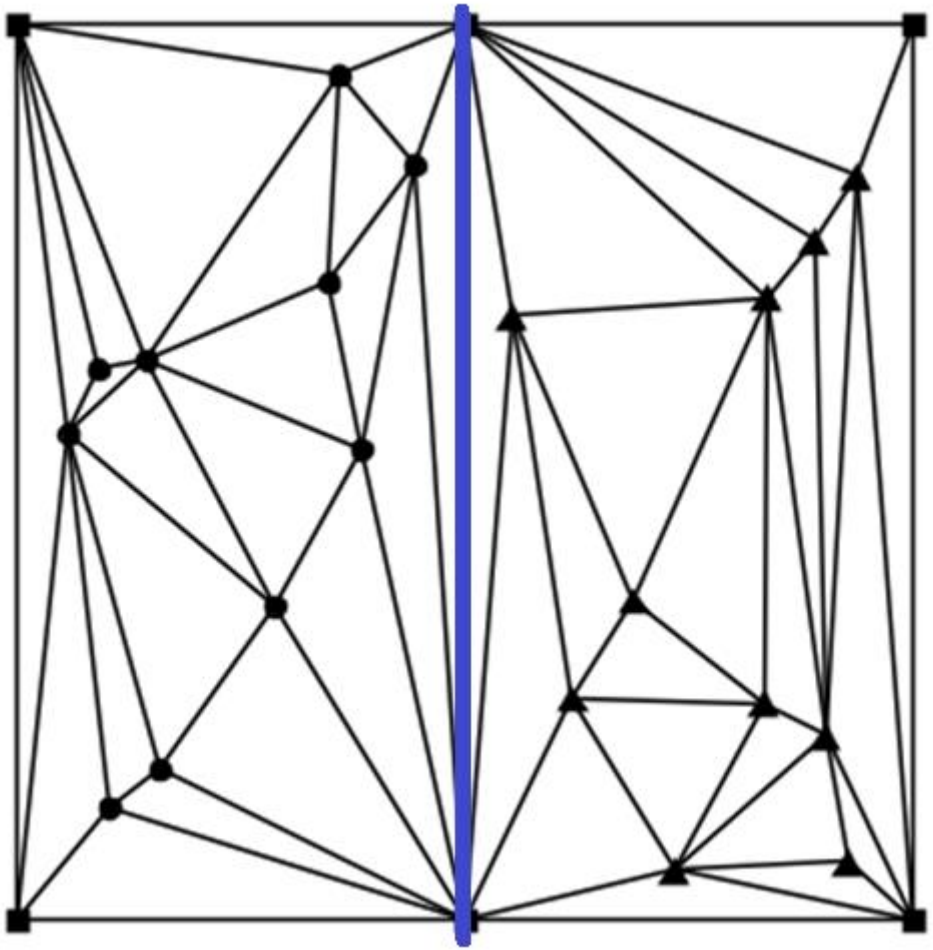
Mesh generation: Missing edges: Sloan algorithm





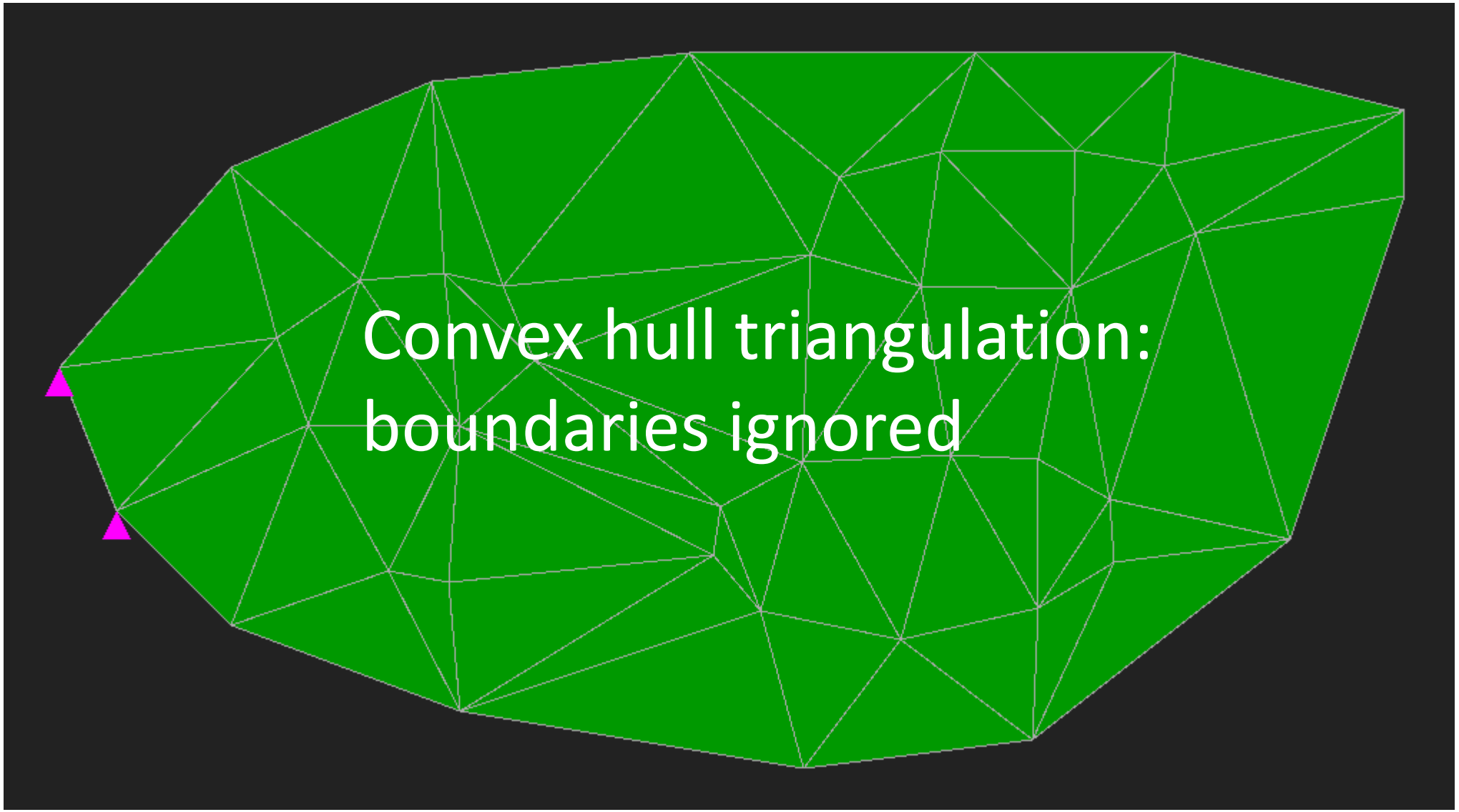
Retriangulate bad quad by swapping its diagonals

Mesh generation: Missing edges

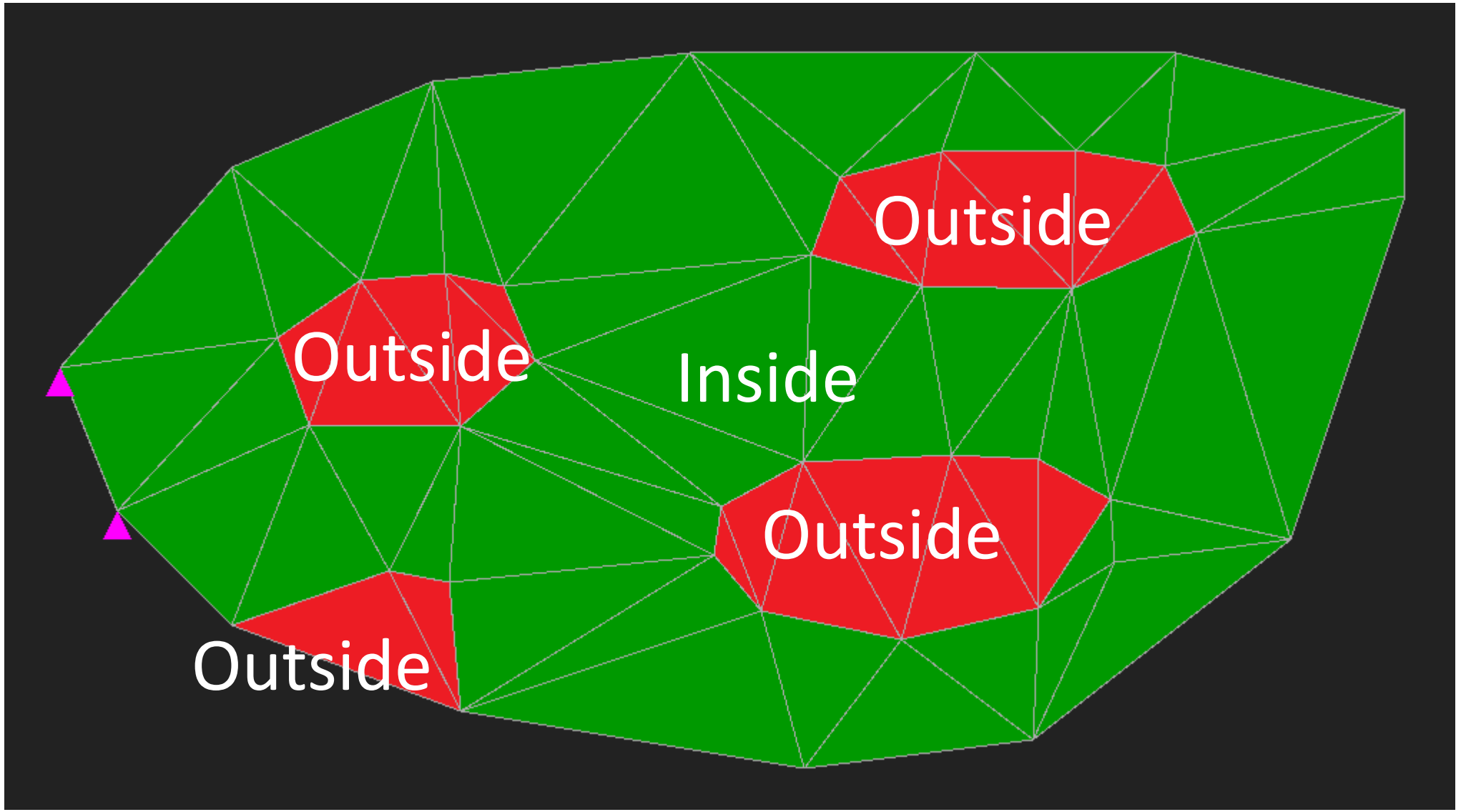


No longer Delaunay, but who cares if we need this edge?

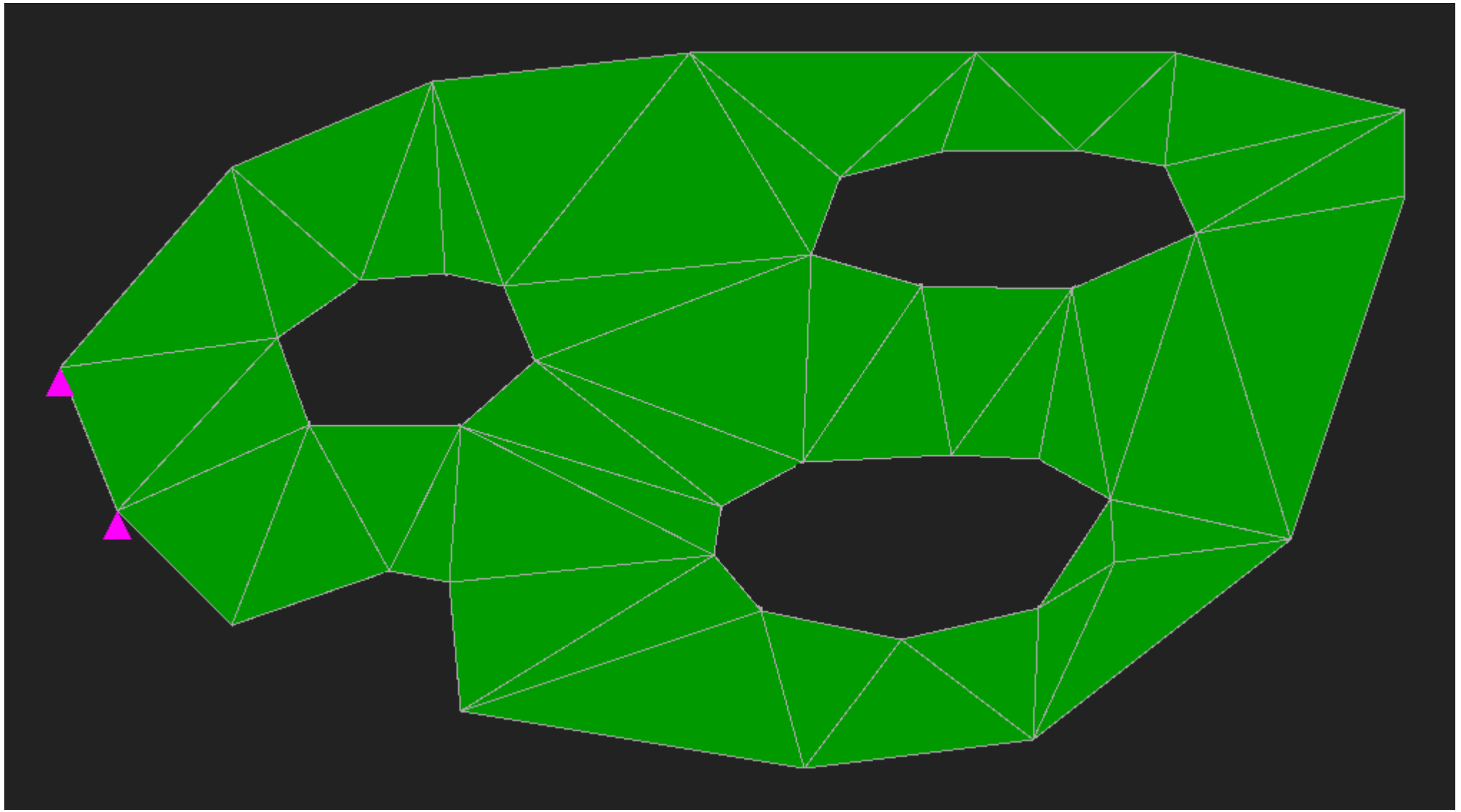
Mesh generation: Removing outside triangles



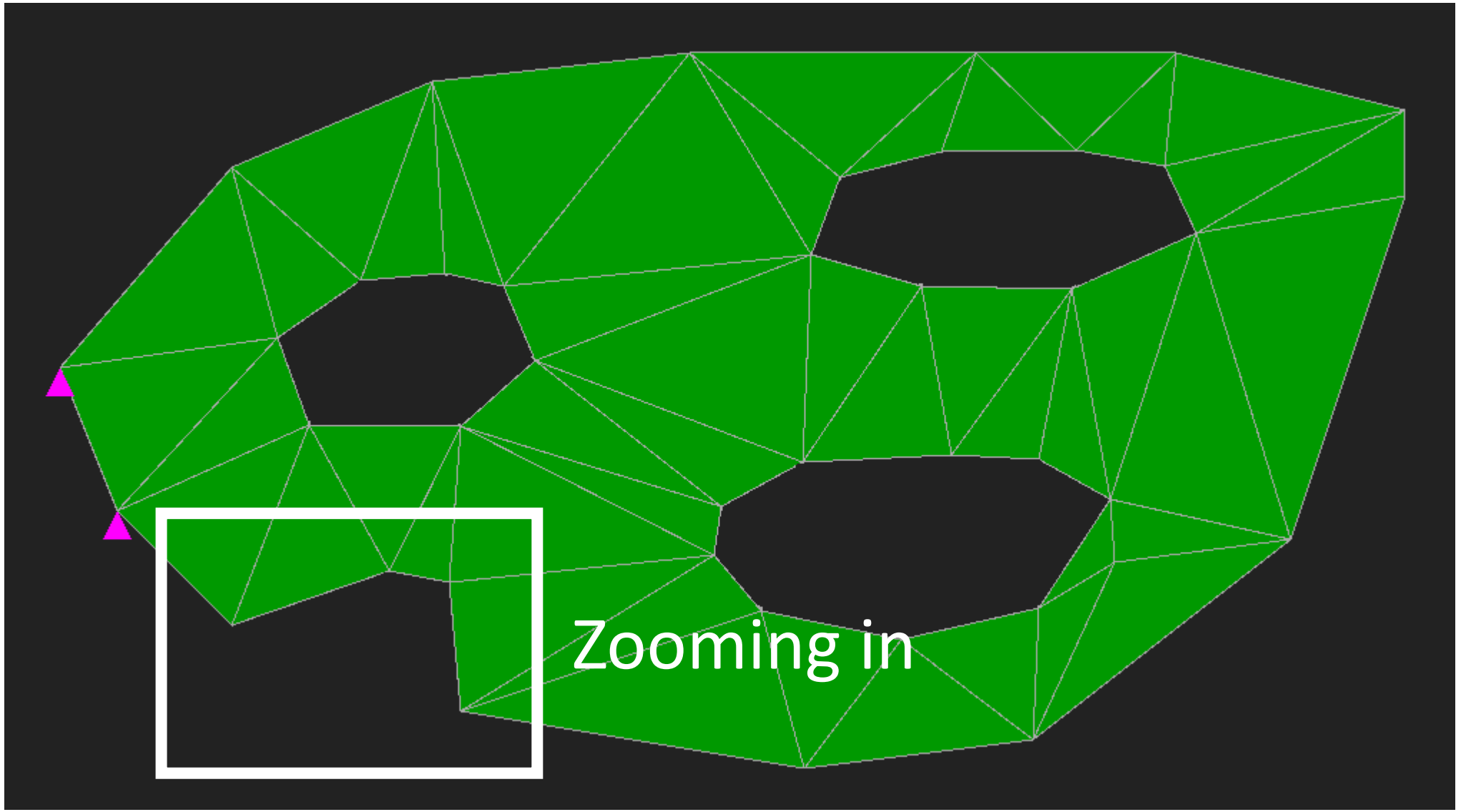
Mesh generation: Removing outside triangles



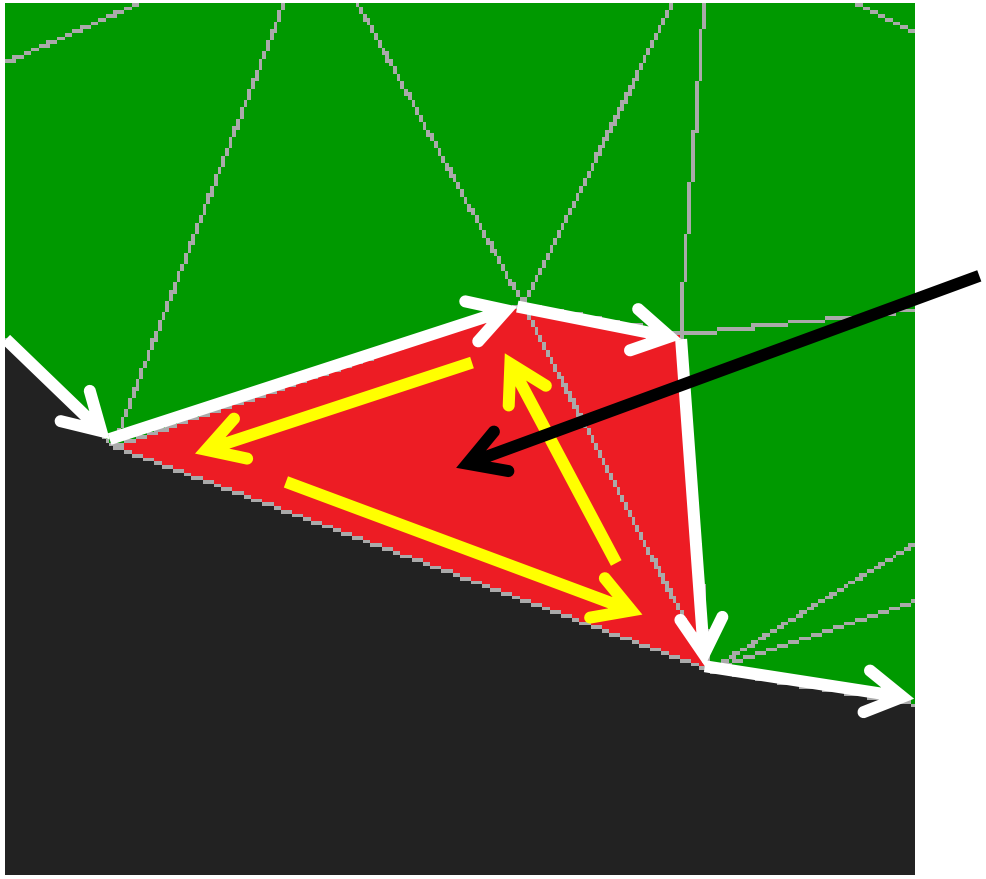
Mesh generation: Removing outside triangles



Mesh generation: Removing outside triangles

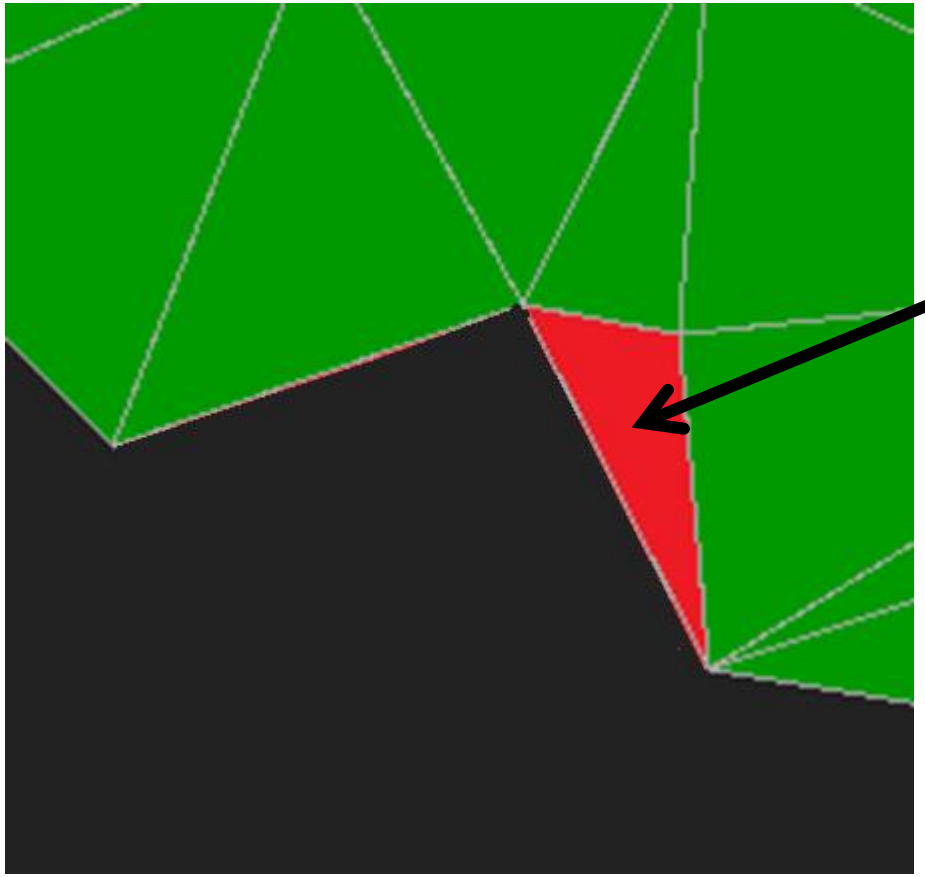


Mesh generation: Removing outside triangles



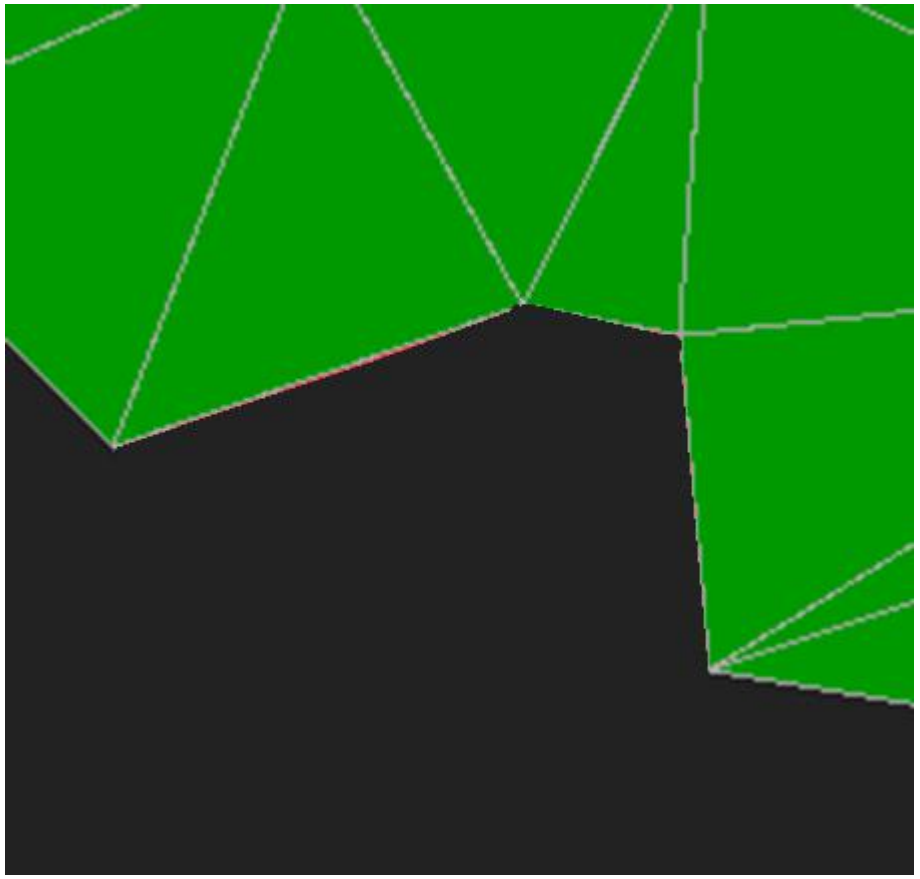
First outside triangle: contains a boundary segment, but in opposite direction

Mesh generation: Removing outside triangles



Next outside triangle(s): all neighbors, except the inside triangles

Mesh generation: Removing outside triangles



Remove recursively!

Mesh generation: Validation

- **Boundary:**
  - No intersections
  - Each point must belong to 0 or 2 edges
- **Mesh:**
  - Each point must belong to a triangle

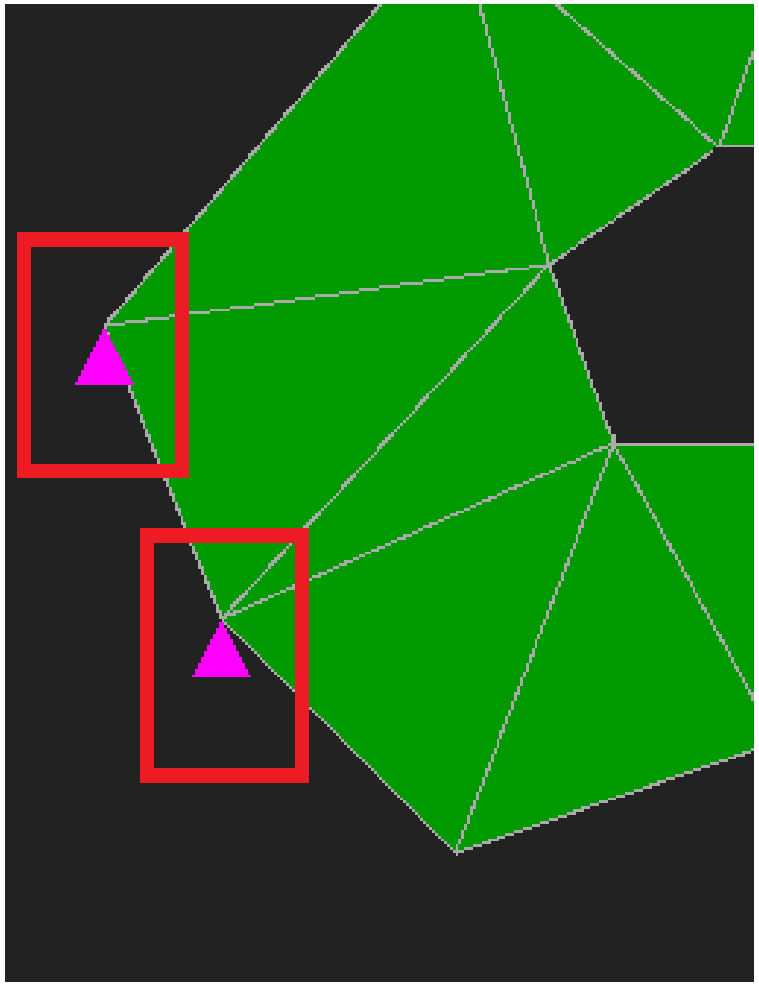
FEM: Discretization

∞ points → *N* points (nodes)

∞ unknown displacements → 2*N* unknown displacements

Interpolate in between

FEM: Constraints



No displacement

↓

No DOF

↓

No equation

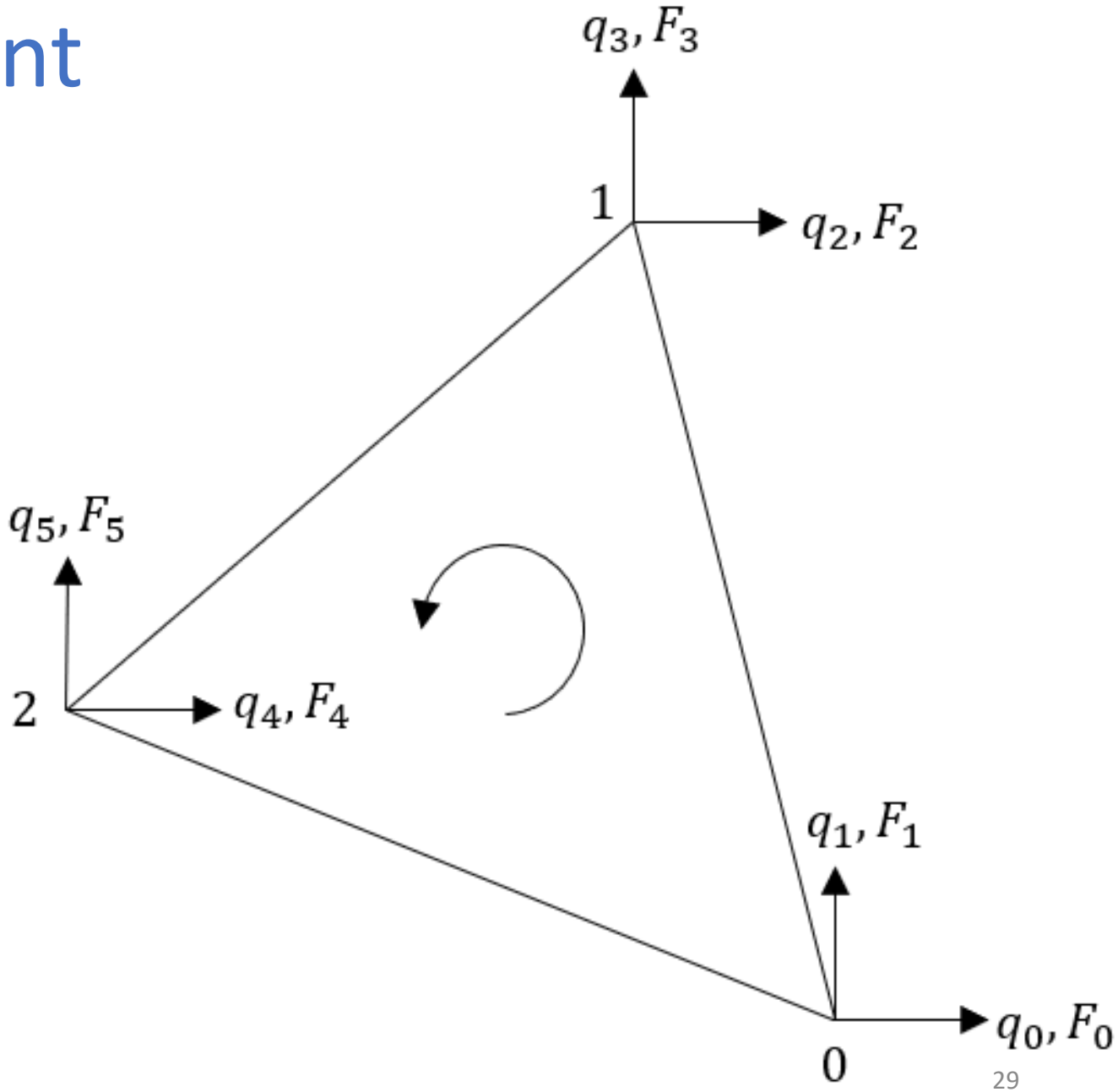
↓

No reaction force

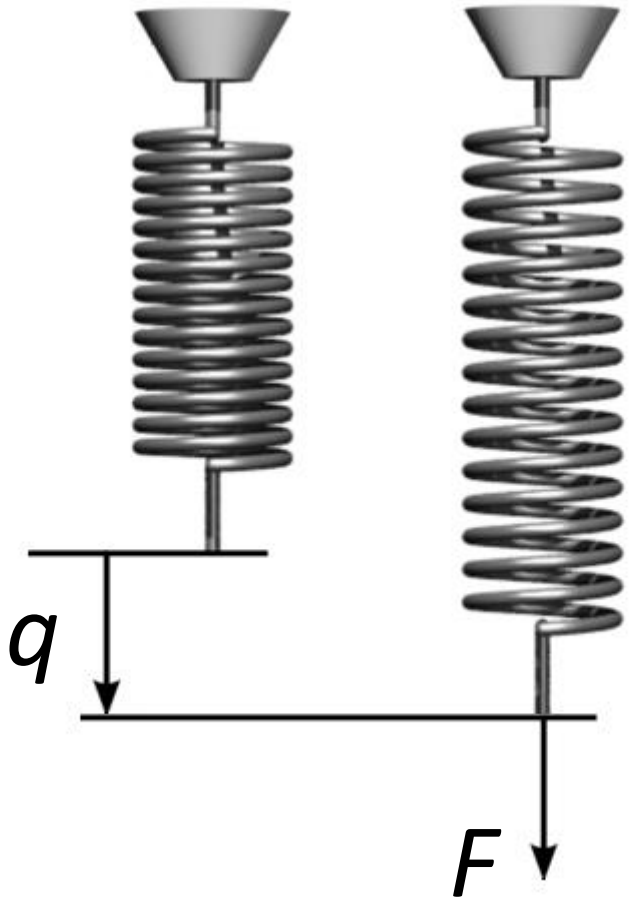
FEM: Triangular element

q = unknown independent displacements (DOFs)

F = external forces (0 if not set)



FEM: Triangular element: “Hooke’s law”



$q \overset{?}{\Longleftrightarrow} F$

1 DOF: $kq = F$

6 DOFs:

$$k \begin{bmatrix} q_0 \\ \vdots \\ q_5 \end{bmatrix} = \begin{bmatrix} F_0 \\ \vdots \\ F_5 \end{bmatrix}$$

Source: https://en.wikipedia.org/wiki/Stiffness

FEM: Triangular element: “Hooke’s law”

In scalar form:

$$k_{00}q_0 + k_{01}q_1 + \cdots + k_{05}q_5 = F_0$$

$$\vdots$$

$$k_{50}q_0 + k_{51}q_1 + \cdots + k_{55}q_5 = F_5$$

FEM: Triangular element: Stiffness matrix

$$k = \begin{bmatrix} k_{00} & \dots & k_{05} \\ \vdots & \vdots & \vdots \\ k_{50} & \dots & k_{55} \end{bmatrix}$$

- 6×6 matrix
- Indexed by element DOFs
- Depends on triangle sides, material and thickness

k_{ij} = force F_i needed to make $q_j = 1$

Explicit formula in `fem::createElement()`

FEM: Triangular element: Stiffness matrix

```
det := b.x * c.y - c.x * b.y
area := 0.5 * fabs(det)

el.geometry = matrix::Matrix{
    {-b.y,    0, -c.y,    0, -a.y,    0},
    {   0,  b.x,    0,  c.x,    0,  a.x},
    { b.x, -b.y,  c.x, -c.y,  a.x, -a.y}
}.mulf(1.0 / det)

el.elasticity = matrix::Matrix{
    {                1, material.poisson,                                0},
    {material.poisson,                 1,                                0},
    {                0,                 0, (1 - material.poisson) / 2}
}.mulf(material.young / (1 - material.poisson * material.poisson))

el.stiffness = el.geometry.transpose().mul(el.elasticity).mul(el.geometry)
    .mulf(material.thickness * area)
```

FEM: Connecting elements

- **Element**: 6 DOFs
- **Whole model**: 2*N* DOFs

Element DOF
(0…5)

Global DOF
(0…2*N*—1)

FEM: Global stiffness matrix

$$K \begin{bmatrix} q_0 \\ \vdots \\ q_{2N-1} \end{bmatrix} = \begin{bmatrix} F_0 \\ \vdots \\ F_{2N-1} \end{bmatrix}$$

- $2N \times 2N$ matrix
- Indexed by global DOFs
- Made of element stiffness matrices

$$K = \begin{bmatrix} K_{00} & \cdots & K_{0,2N-1} \\ \vdots & \vdots & \vdots \\ K_{2N-1,0} & \cdots & K_{2N-1,2N-1} \end{bmatrix}$$

FEM: Global stiffness matrix

Pseudocode:

```
K = 0
for each element
    for each i
        for each j
```

$$K_{mn} += k_{ij}$$

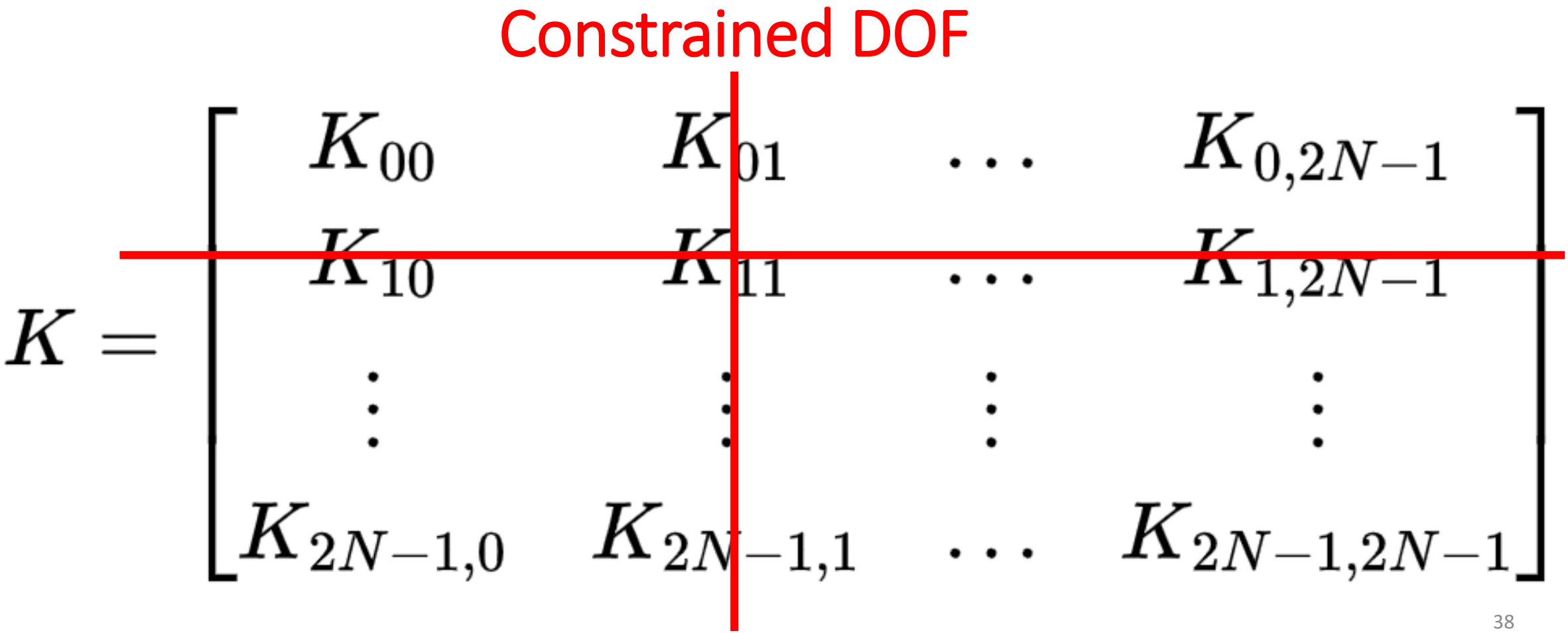
Element DOF		Global DOF
i	⟺	m
j	⟺	n

FEM: Global stiffness matrix

```
fn getGlobalStiffness(els: []Element, numPts: int): matrix::Matrix {
    stiffness := matrix::zeros(2 * numPts, 2 * numPts)
    for _, el in els {
        for row in el.stiffness {
            globalRow := el.convertDofToGlobal(row)
            for col in el.stiffness[0] {
                globalCol := el.convertDofToGlobal(col)
                stiffness[globalRow][globalCol] += el.stiffness[row][col]
            }
        }
    }
    return stiffness
}
```

FEM: Constraints

Constrained DOFs? Just strike them out!



FEM: Constraints

After r constrained DOFs removed:

$$K' \begin{bmatrix} q_0 \\ \vdots \\ q_{2N-r-1} \end{bmatrix} = \begin{bmatrix} F_0 \\ \vdots \\ F_{2N-r-1} \end{bmatrix}$$

- $(2N - r) \times (2N - r)$ matrix
- Non-degenerate: can solve for q

FEM: Displacements

Solve for q

1 DOF: $q = F / k$

($2N - r$) DOFs:

$$\begin{bmatrix} q_0 \\ \vdots \\ q_{2N-r-1} \end{bmatrix} = (K')^{-1} \begin{bmatrix} F_0 \\ \vdots \\ F_{2N-r-1} \end{bmatrix}$$

FEM: Displacements: Gaussian elimination

Don't want to inverse *K*'? Solve *K*'*q* = *F* directly

Example: 2 unconstrained DOFs

$$K_{00}q_0 + K_{01}q_1 = F_0 \quad (1)$$

$$K_{10}q_0 + K_{11}q_1 = F_1 \quad (2)$$

How many (1)s do we subtract from (2) to eliminate $K_{10}q_0$? Answer: $\frac{K_{10}}{K_{00}}$

FEM: Displacements: Gaussian elimination

$$K_{00}q_0 + K_{01}q_1 = F_0$$

$$K_{10}q_0 + K_{11}q_1 = F_1$$

$$K_{00}q_0 + K_{01}q_1 = F_0$$

$$\left(K_{10} - \frac{K_{10}}{K_{00}}K_{00}\right)q_0 + \left(K_{11} - \frac{K_{10}}{K_{00}}K_{01}\right)q_1 = F_1 - \frac{K_{10}}{K_{00}}F_0$$

0

Find q_1, then q_0

FEM: Stresses

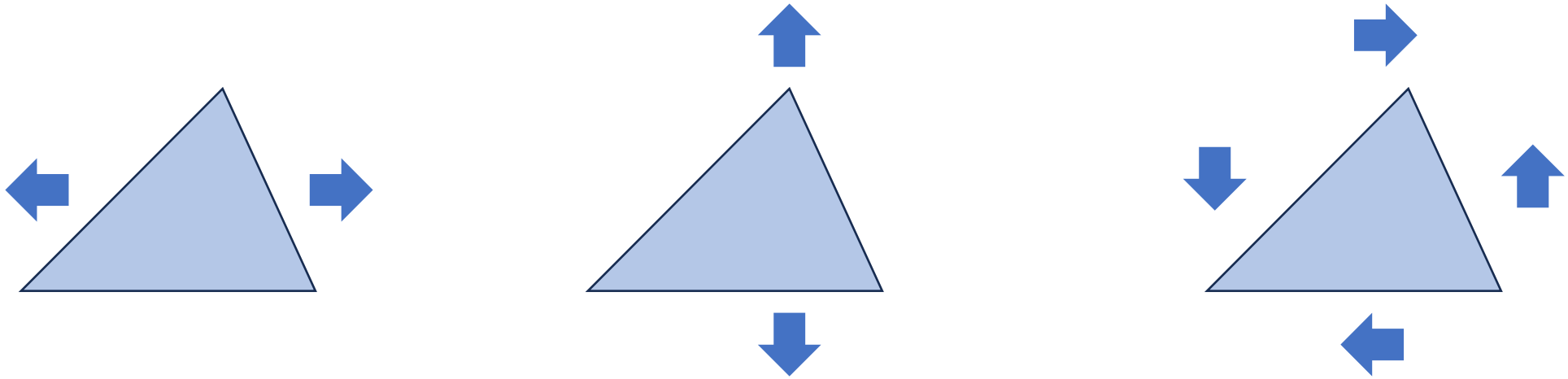
Displacements interpolated linearly

Constant strain over element

Constant stress over element

Constant color in stress visualization

FEM: Stresses

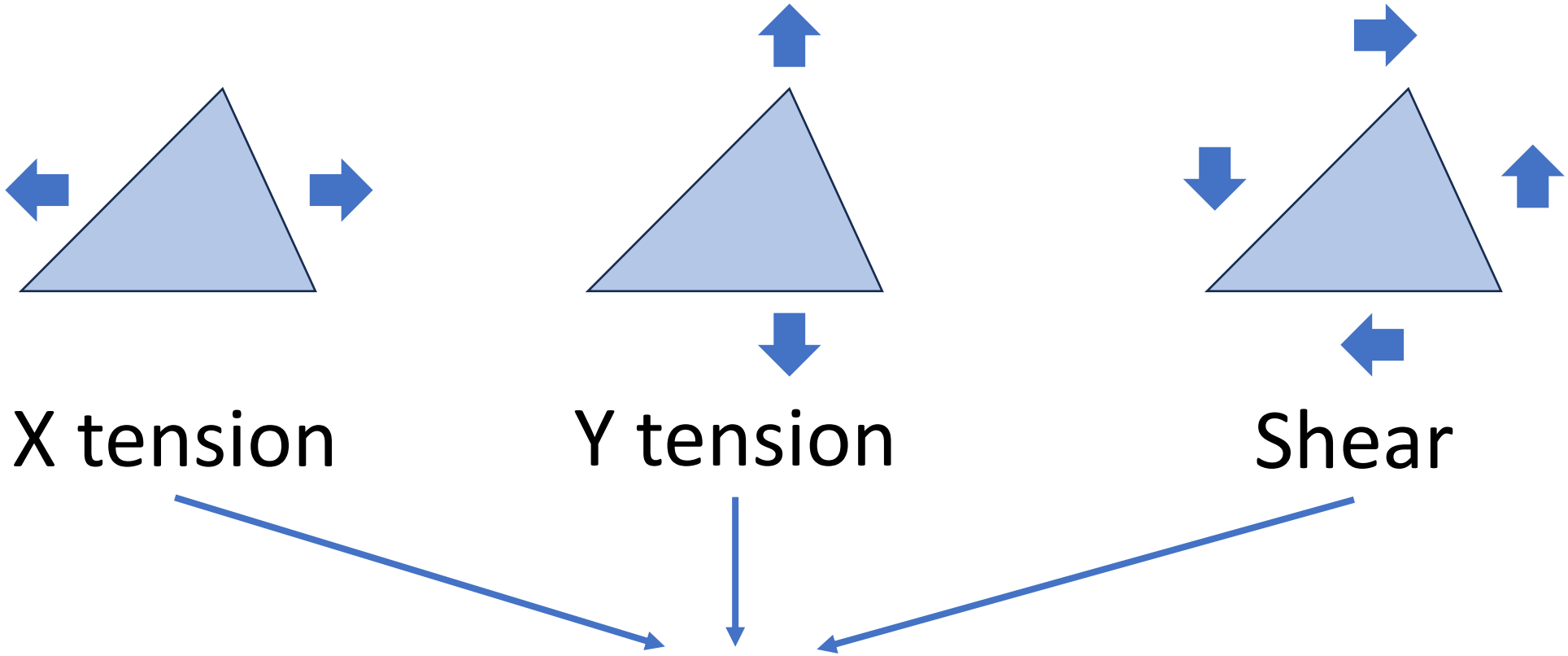


X tension Y tension Shear

3 stresses responsible for 3 “elastic” DOFs

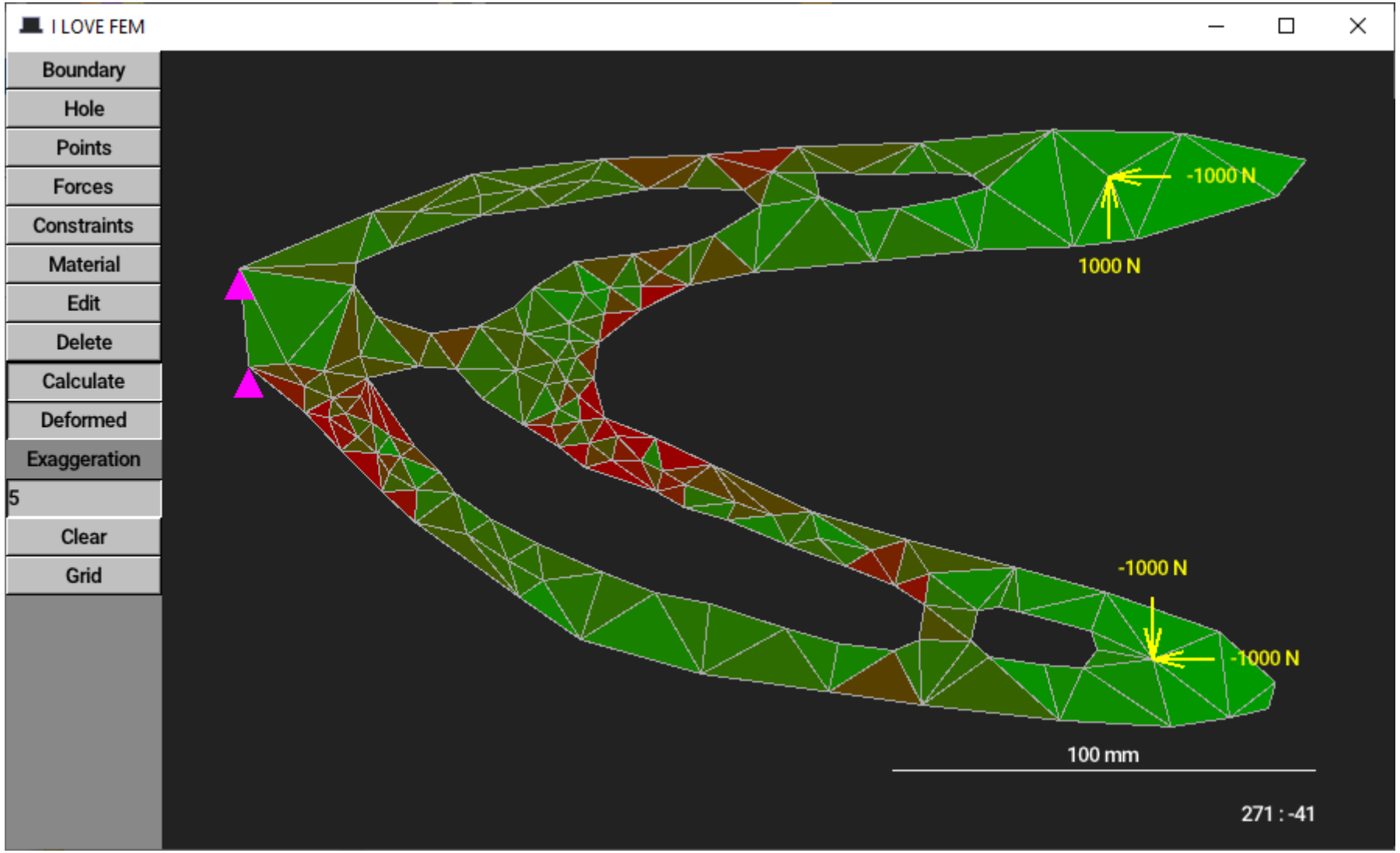
Explicit formula in `fem::getEquivalentStress()`

FEM: Stresses



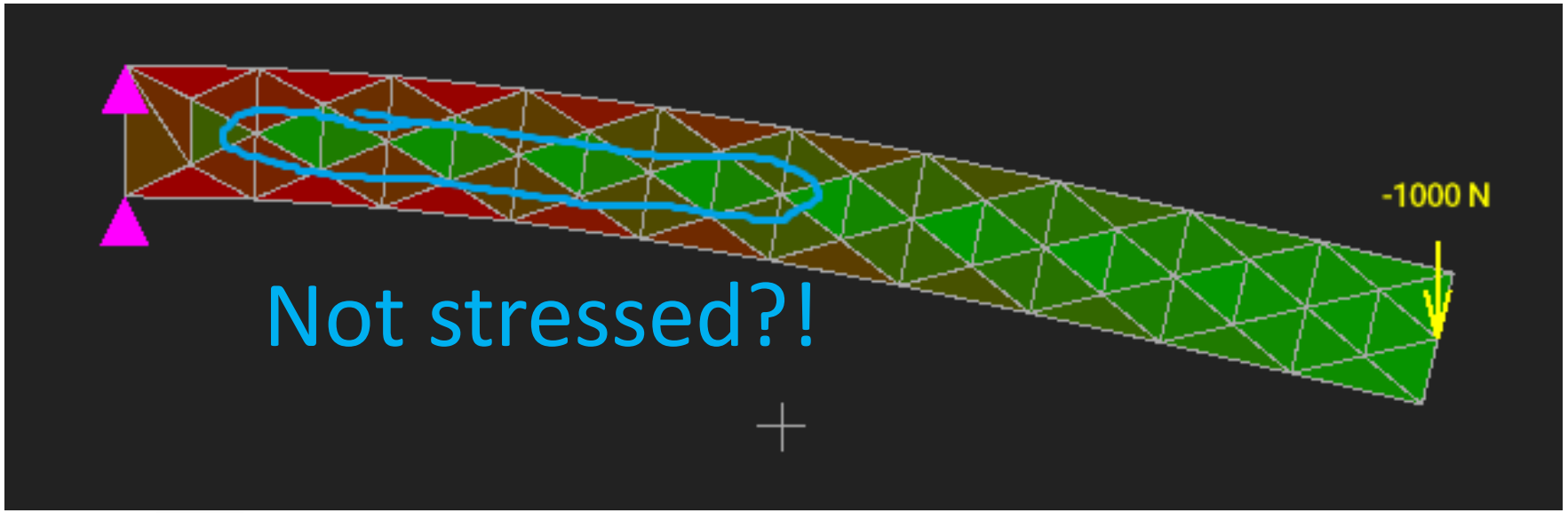
Explicit formula in `fem::getEquivalentStress()`

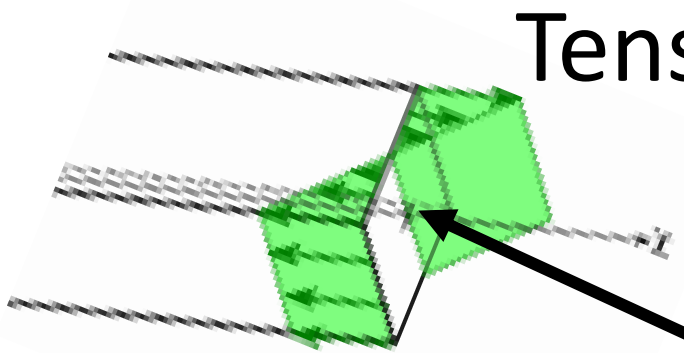
Visualization



Self-check

Beam bending





Implementation

Umka: a statically typed embeddable scripting language

https://github.com/vtereshkov/umka-lang

tophat: a 2D framework for making games in Umka

https://tophat2d.dev/

Questions?