April 2018

Online International Meeting for Users of OpenFOAM
Published on YouTube

# On the implementation of point-based and cell-based mesh smoothing approaches in OpenFOAM

Philip Cardiff




School of Mechanical and Materials Engineering
University College Dublin

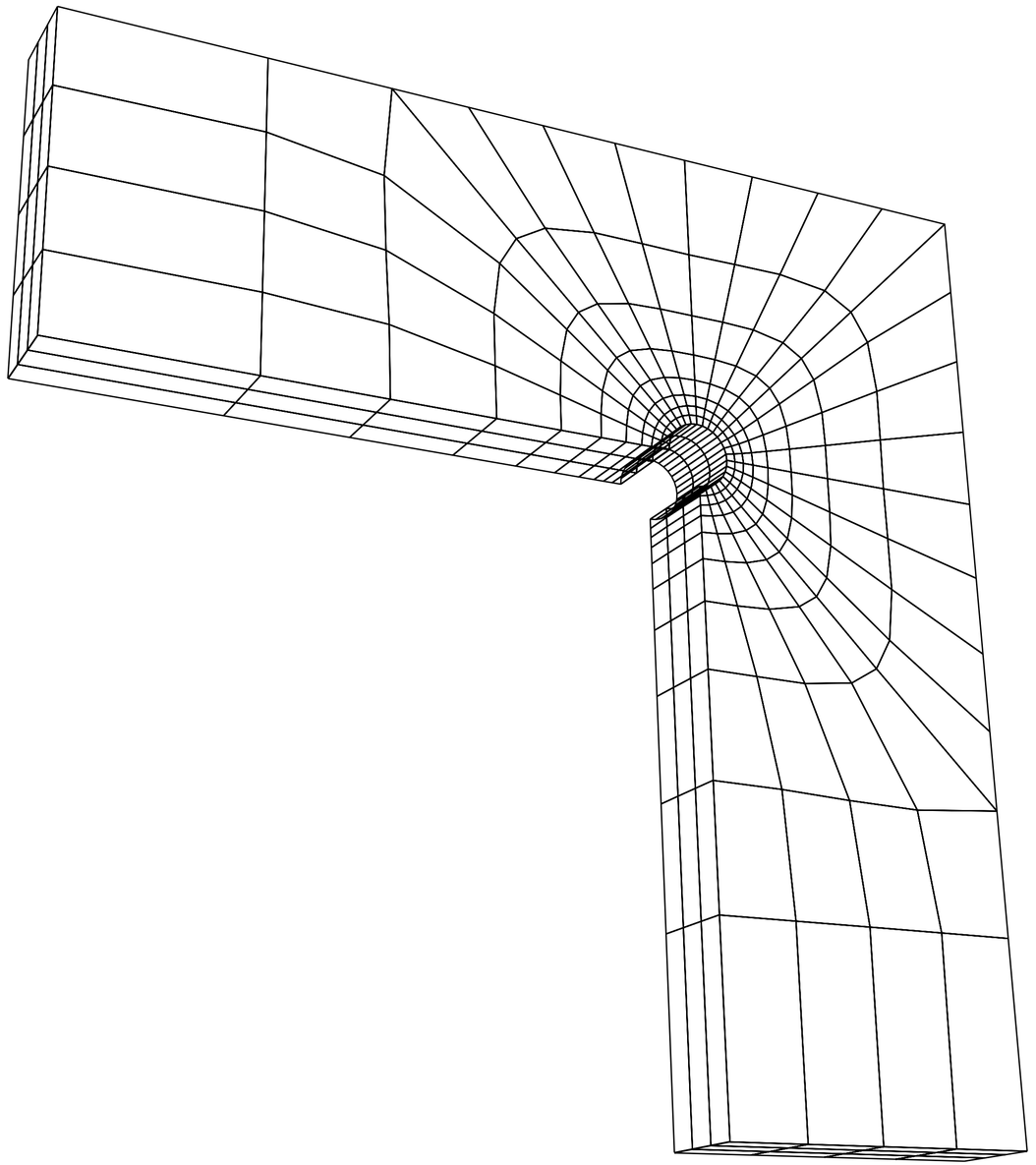
# Background & Motivation

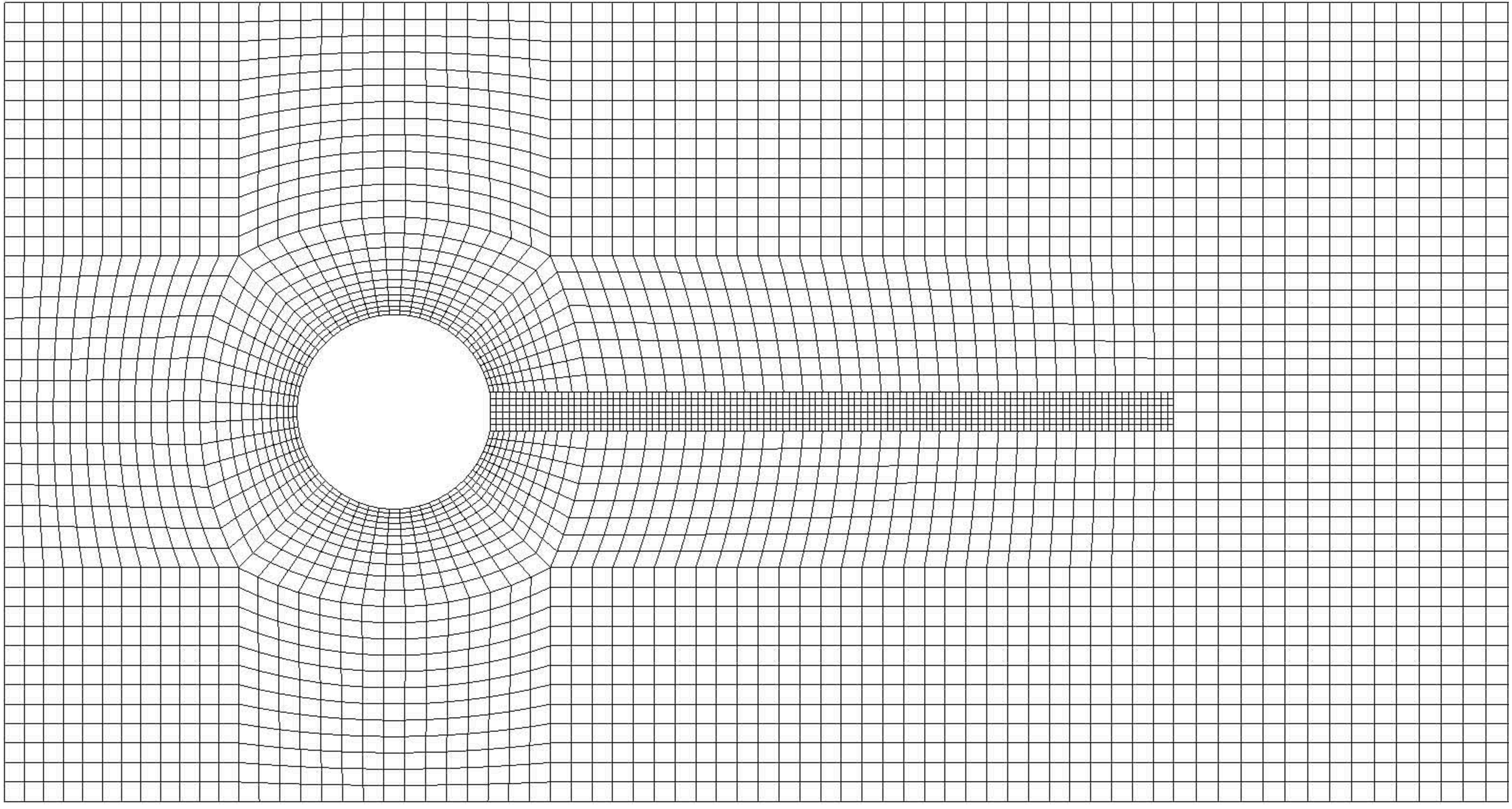
# Background

Mesh quality affects solution quality

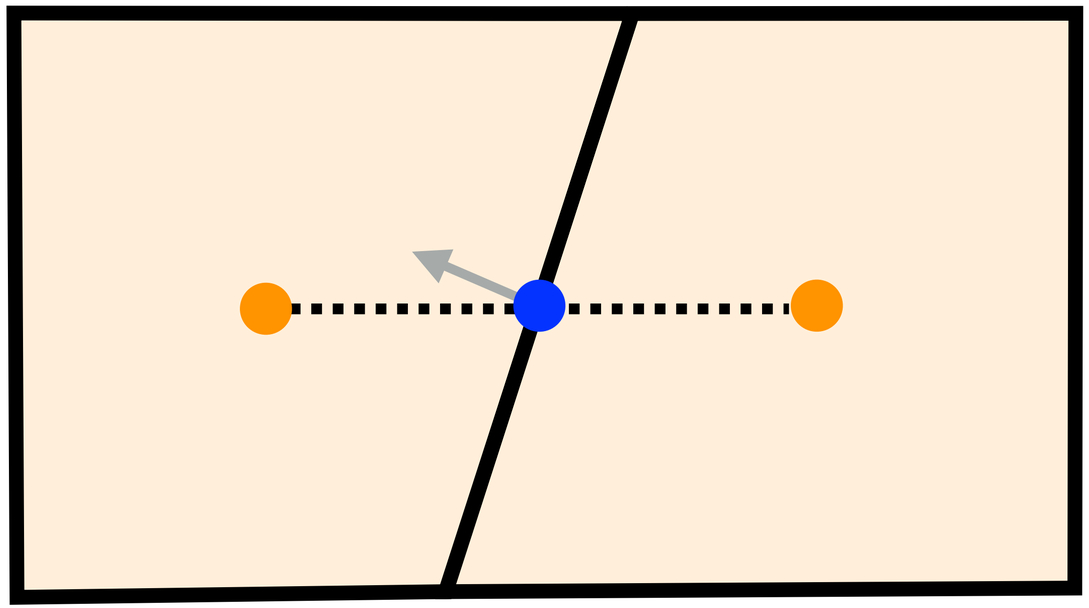
Why smooth?

- improve initial mesh
- maintain mesh quality in moving mesh simulations e.g. Arbitrary Lagrangian-Eulerian

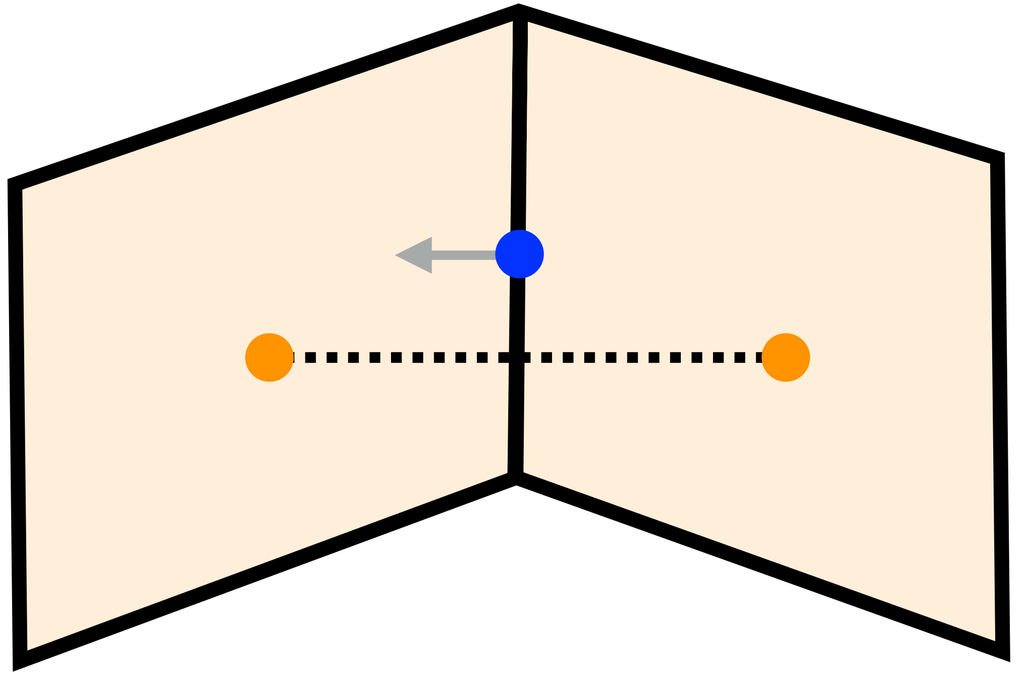




# Important mesh metrics



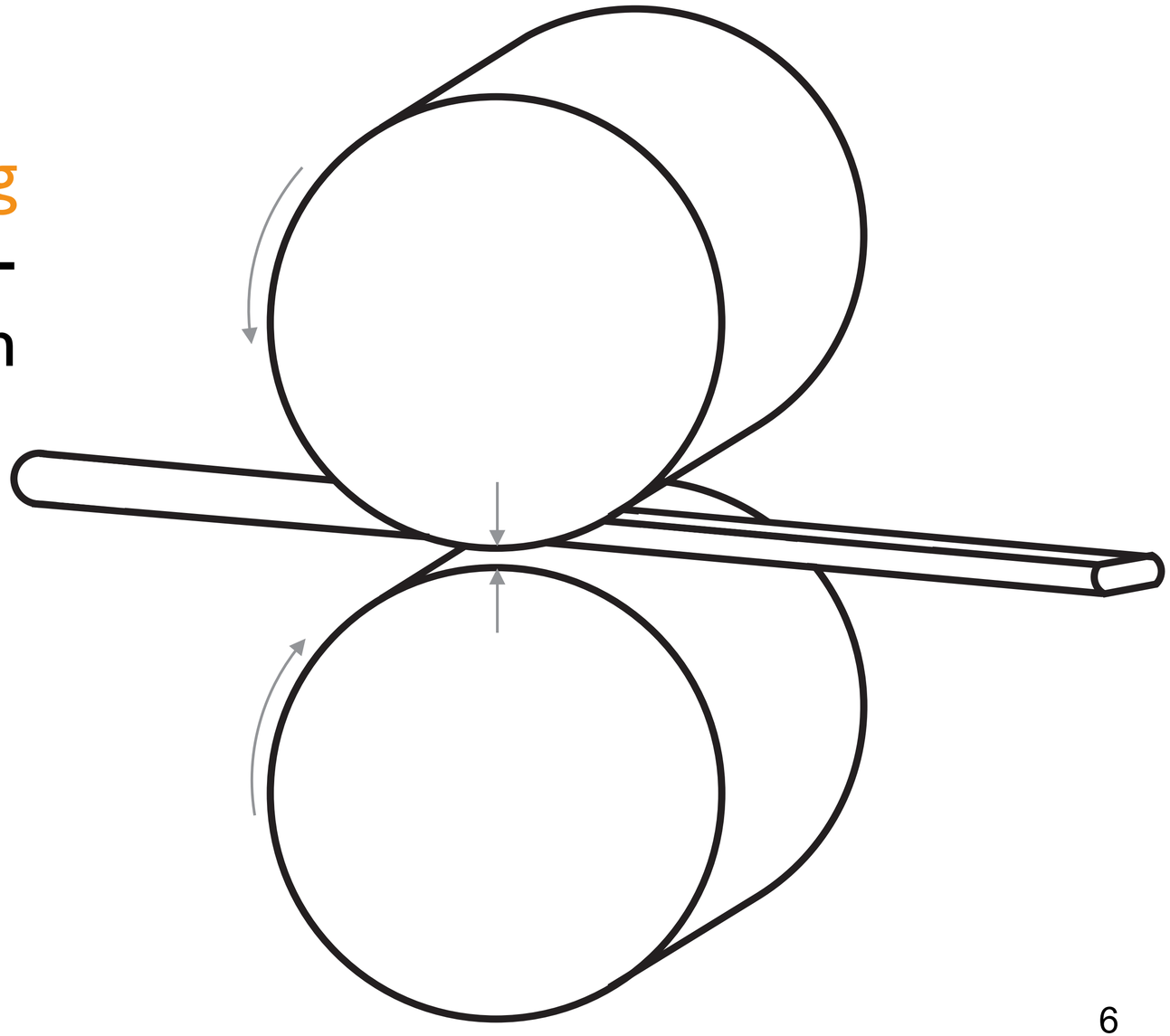
Non-orthogonal face



skewed face

# Goal of this project

Develop a robust mesh smoothing procedure suitable for ALE elasto-plastic metal forming simulations in OpenFOAM

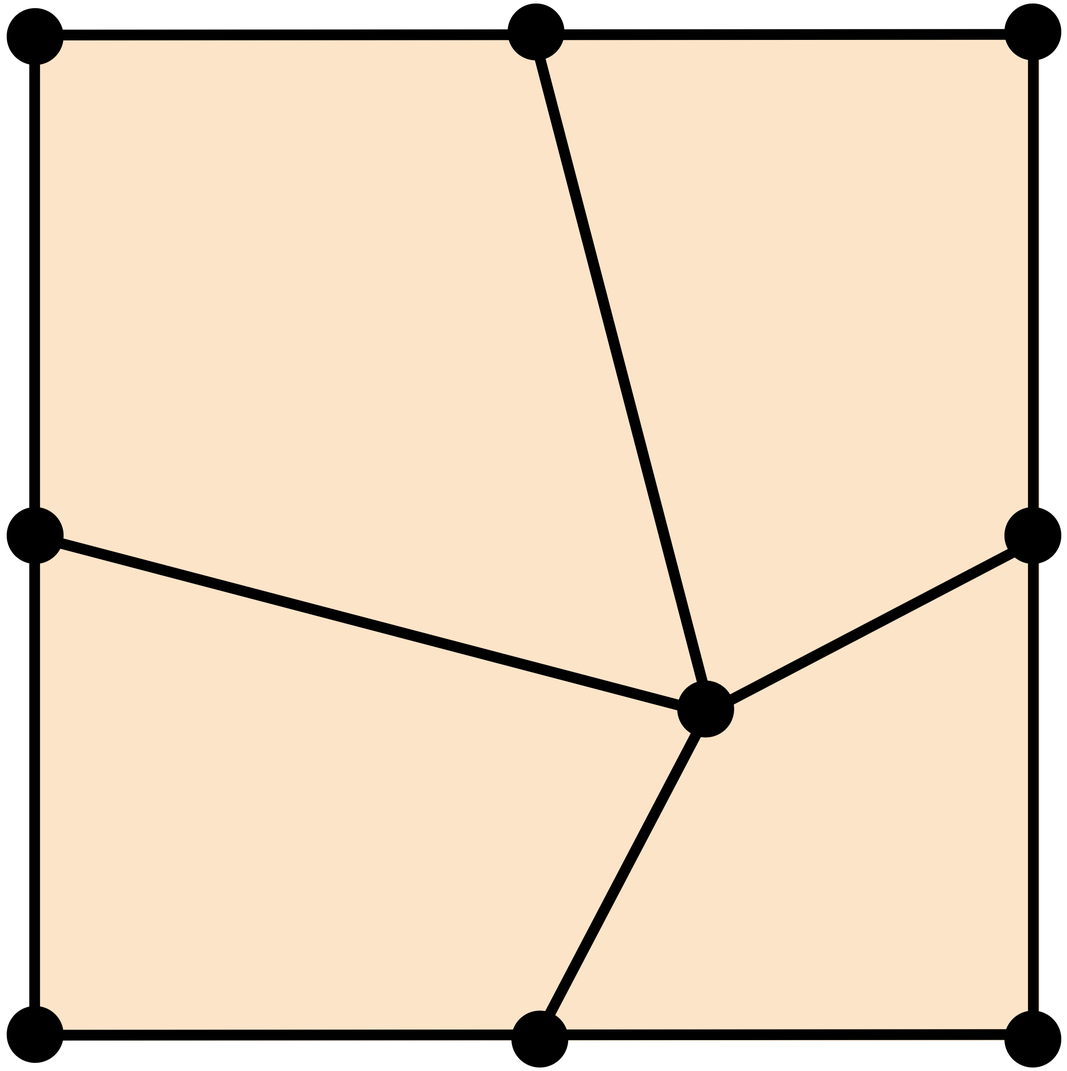


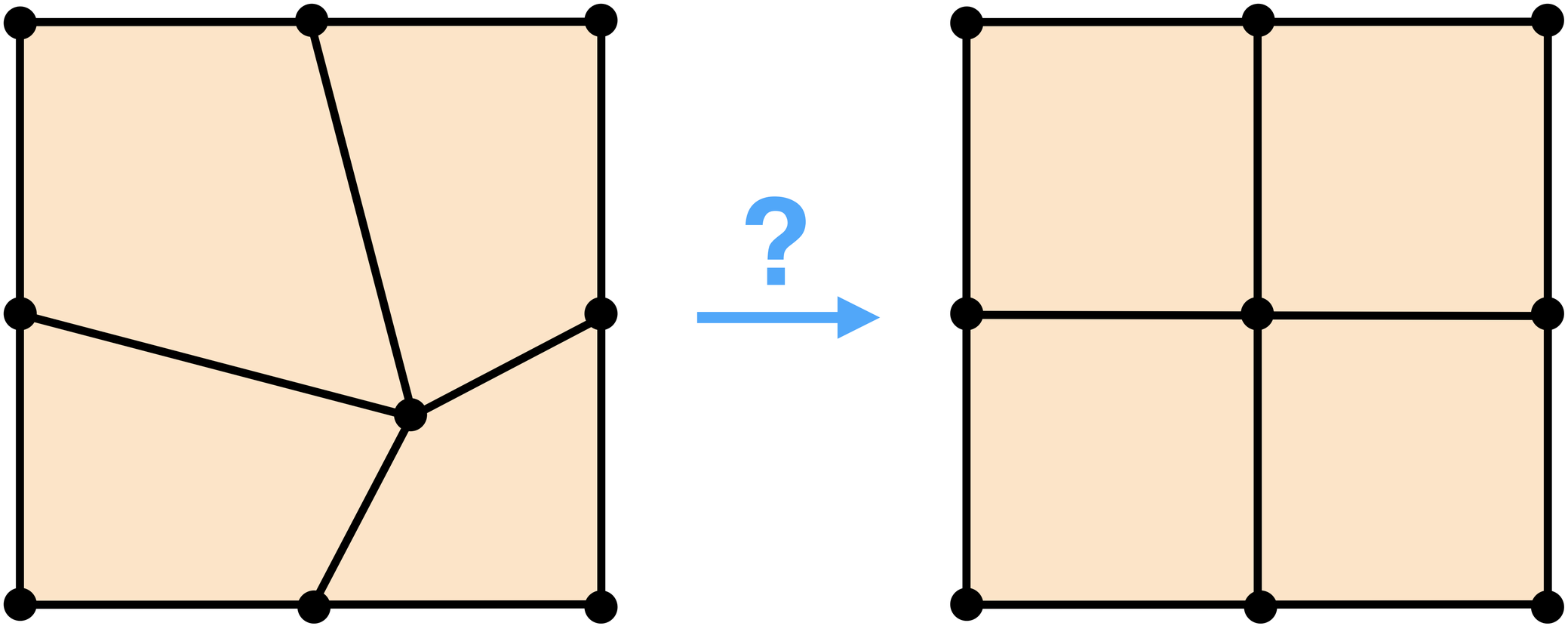
BEKAERT
better together

- Bekaert are global experts in the transformation and coating of high-strength steel wire
- Founded in 1880
- Headquartered in Belgium
- 30,000 employees worldwide

Video from: https://www.youtube.com/watch?v=uaJVATgIgvk&feature=youtu.be

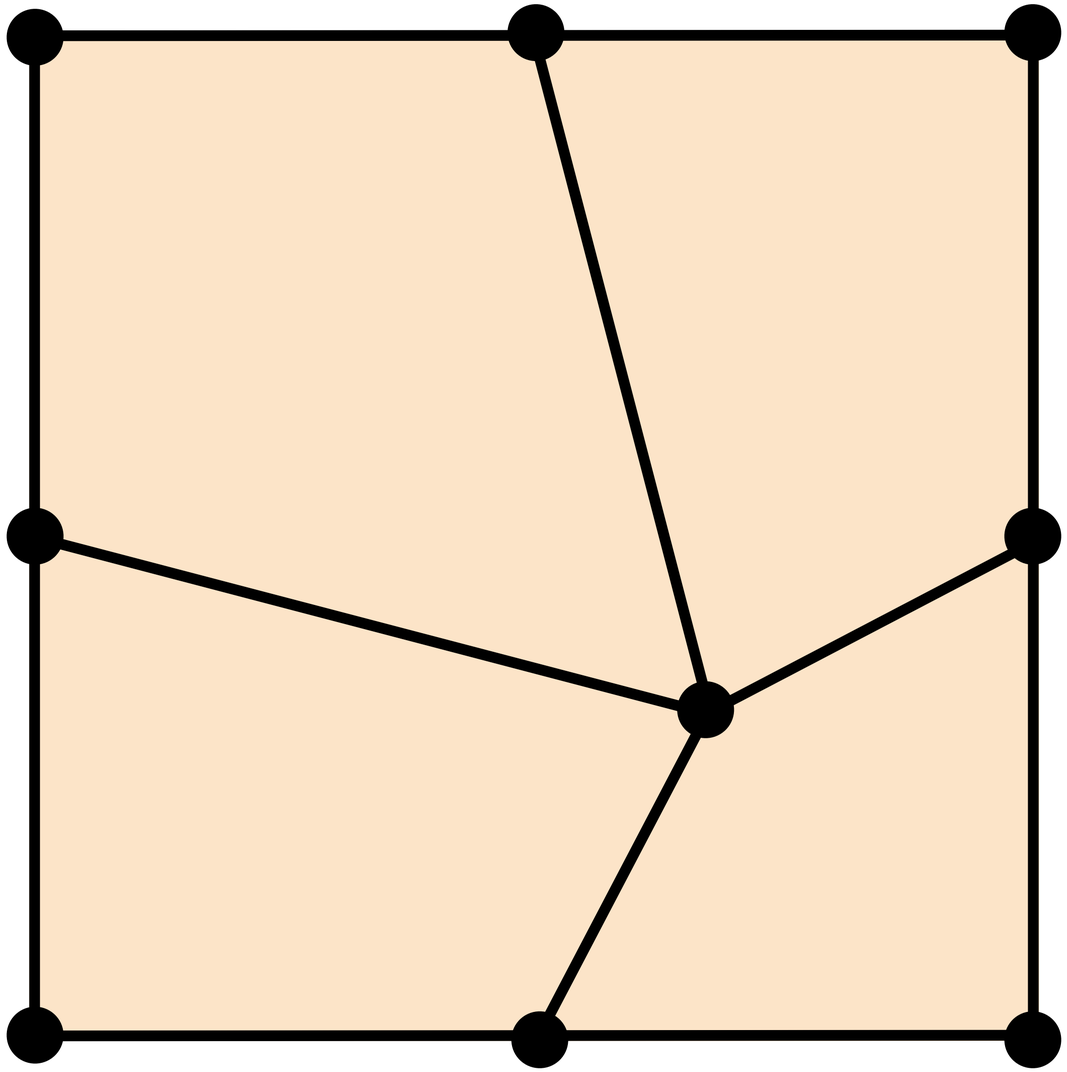
# the challenge





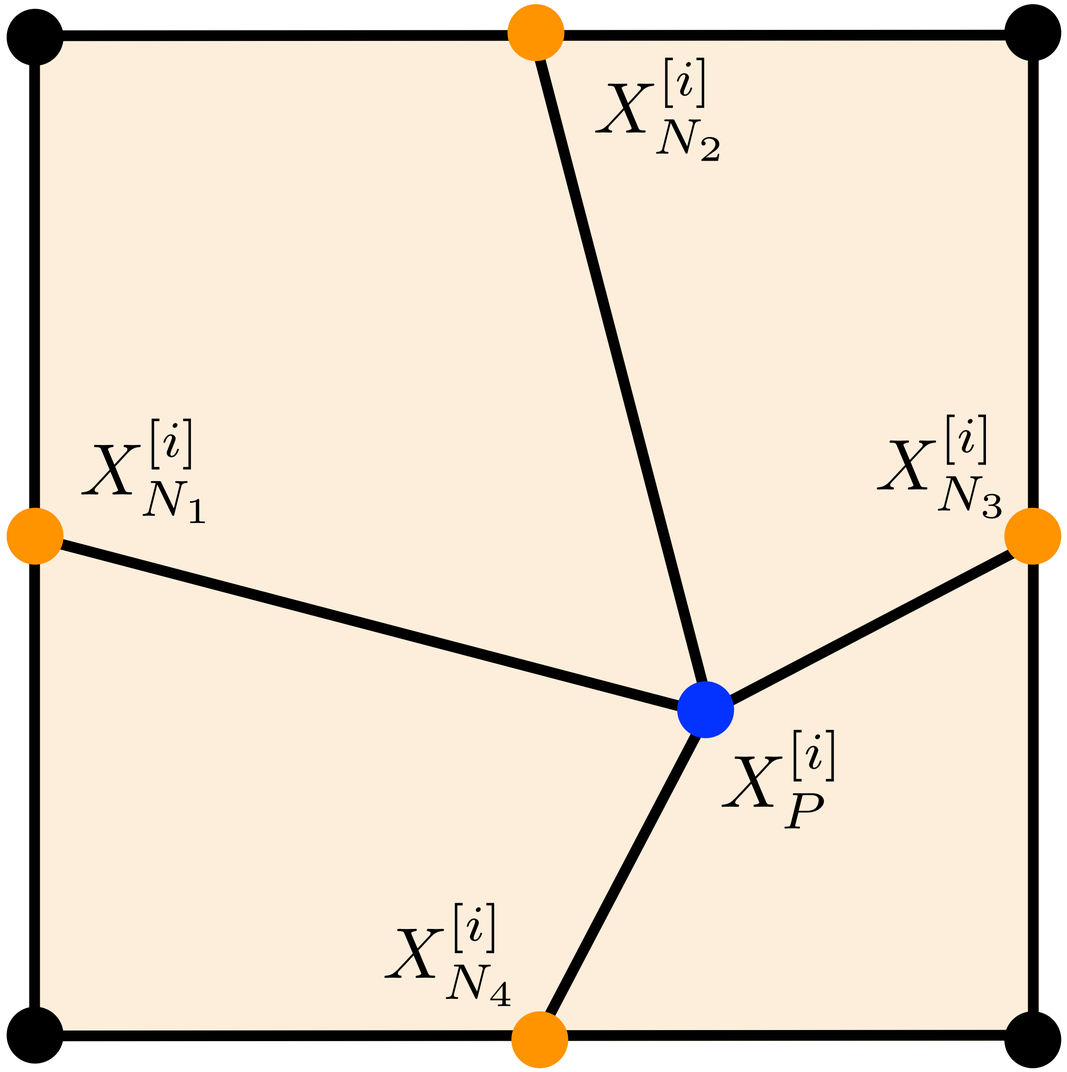

How can we move the mesh points to improve the mesh quality, without changing the connectivity?

# point-based Laplacian smoothing



## Approach 1

point-based Laplacian




## Approach 1

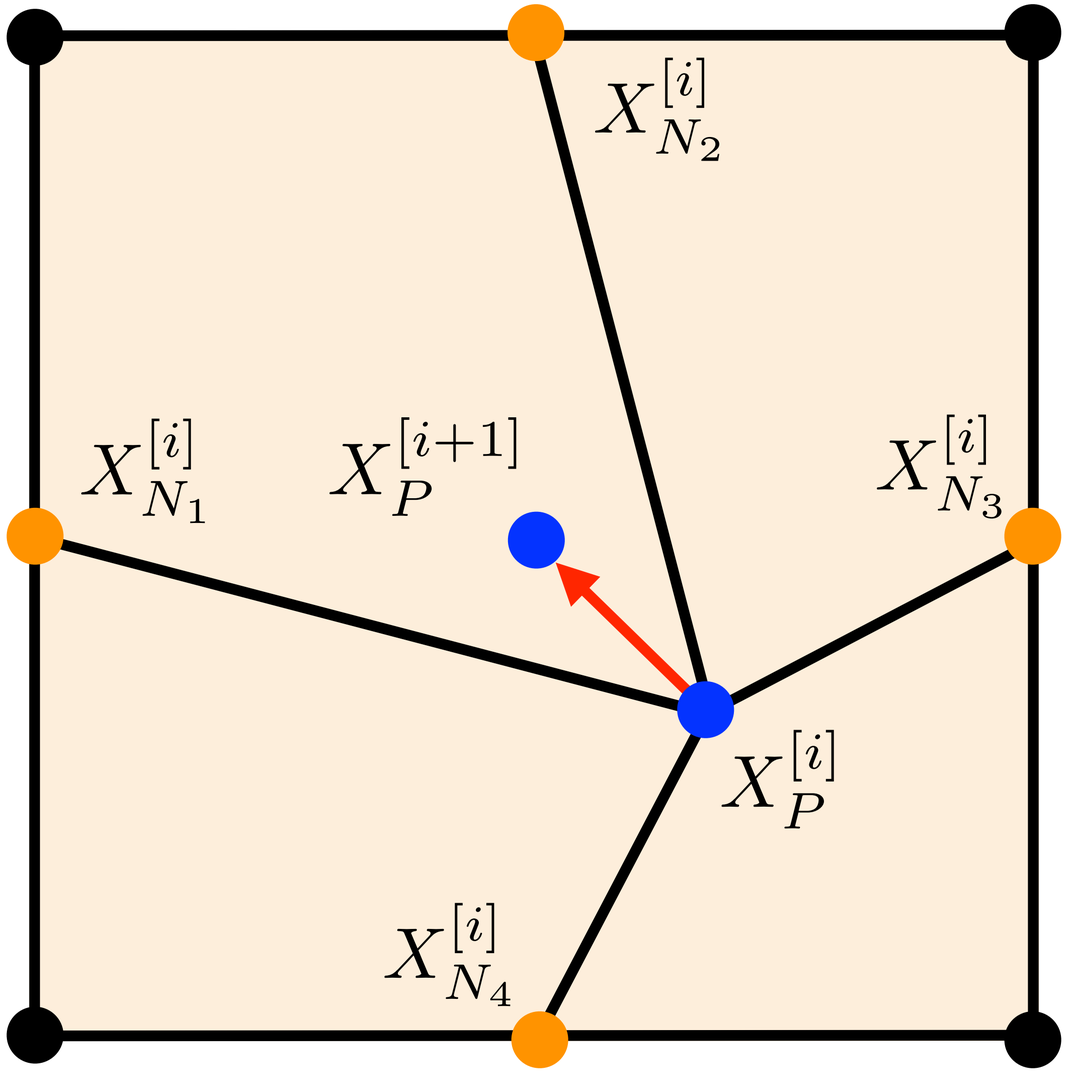
point-based Laplacian

$$X_P^{[i+1]} = \sum_{j=1}^{n} w_j X_{N_j}^{[i]}$$

Where we can calculate the weights as an arithmetic average:

$$w_j = \frac{1}{n}$$

number of neighbour points




## Approach 1

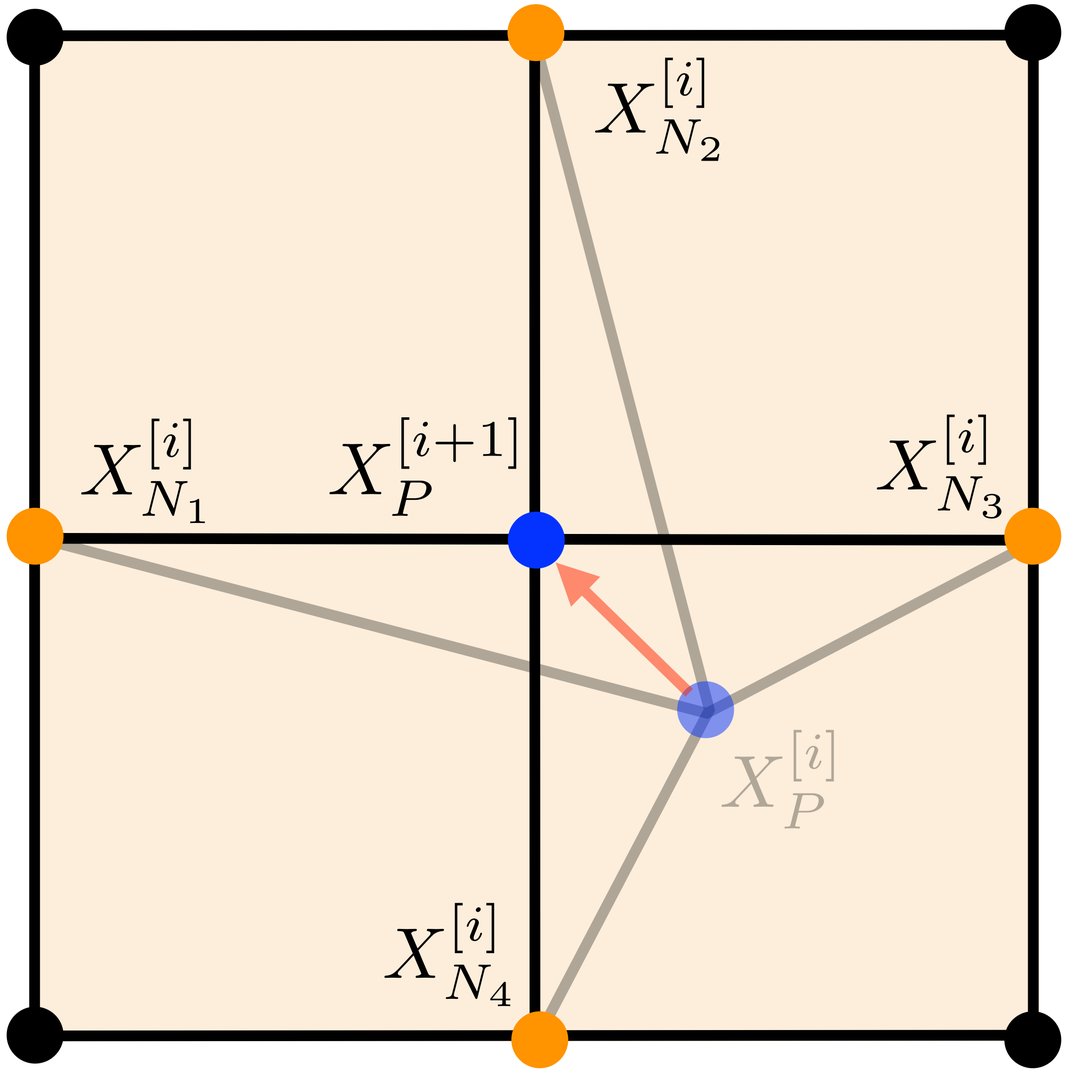
point-based Laplacian

$$X_P^{[i+1]} = \sum_{j=1}^{n} w_j X_{N_j}^{[i]}$$

Where we can calculate the weights as an arithmetic average:

$$w_j = \frac{1}{n}$$

number of neighbour points




## Approach 1

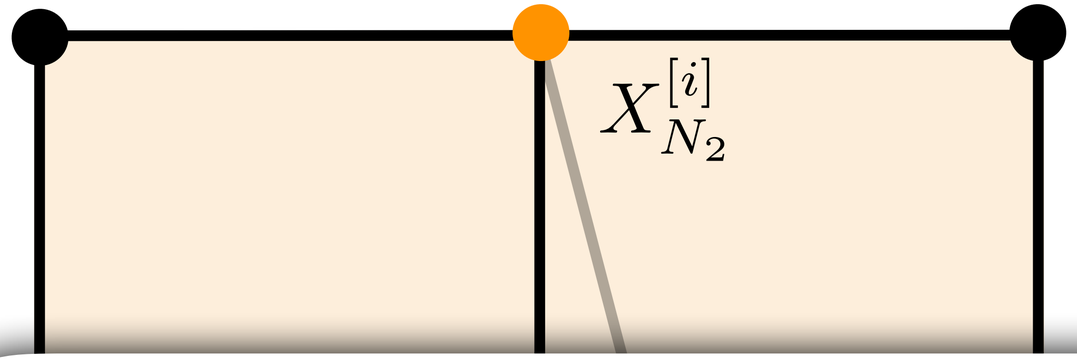
point-based Laplacian

$$X_P^{[i+1]} = \sum_{j=1}^{n} w_j X_{N_j}^{[i]}$$

Where we can calculate the weights as an arithmetic average:

$$w_j = \frac{1}{n}$$

number of neighbour points




## Approach 1
point-based Laplacian

$n$

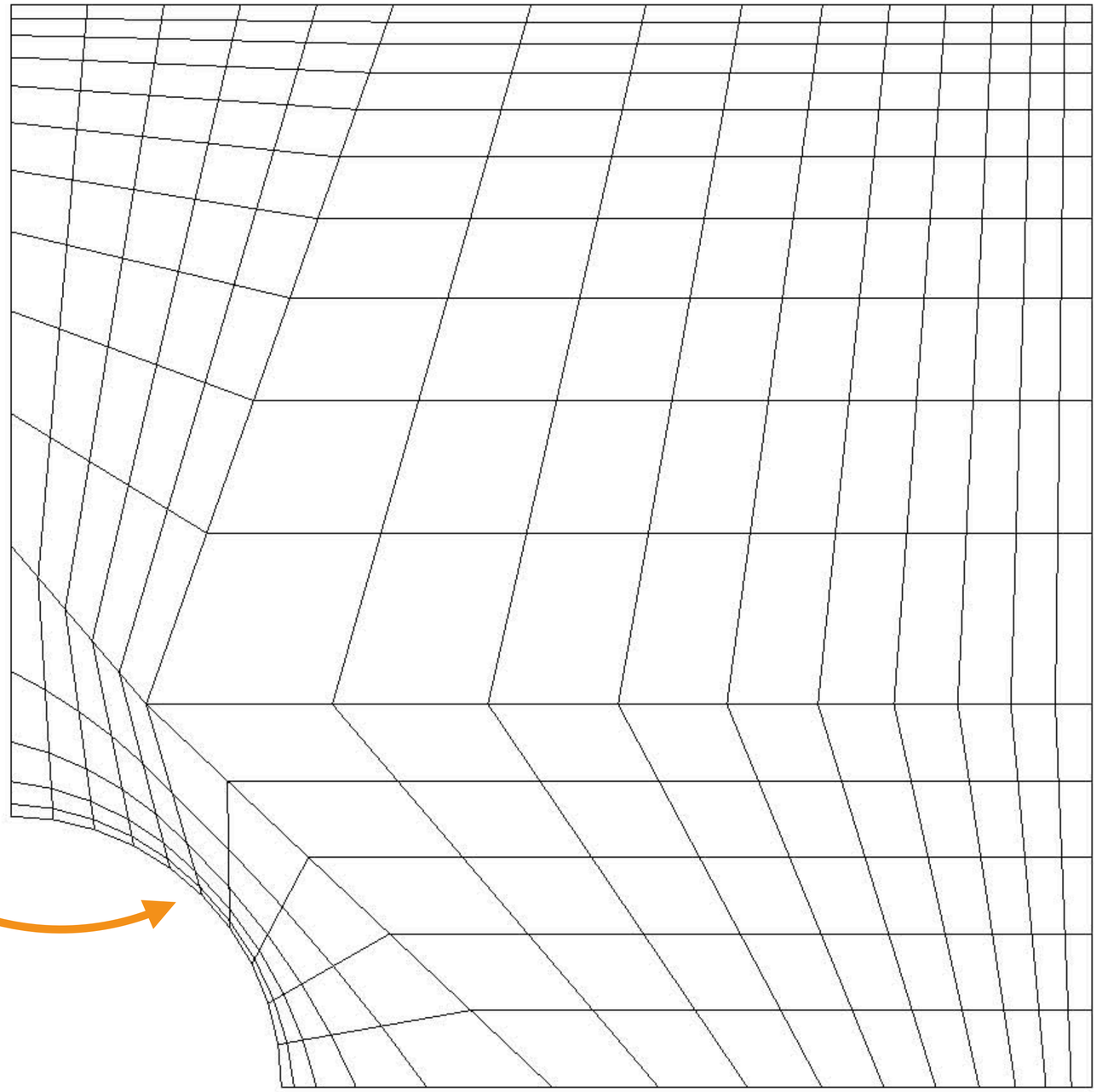
**algorithm 1:**

```
for all internal points in the mesh
    newPoint = vector::zero
    for all neighbour points of the current point
        neighbourPointWeight = 1.0;
        newPoint += neighbourPointWeight*neighbourPoint;
        sumWeights += neighbourPointWeight;
    end
    newPoint =/ sumWeights;
end
```

Note: we have to be careful with boundary points, and also in parallel

neighbour points

Point-based Laplacian



Laplacian tends to shrink cells in concave regions

Iteration: 0

point-based Laplacian smoothing with a Taubin expansion step

Approach 2: to eliminate the shrinkage effect produce by the standard Laplacian smoothing, a Taubin expansion step can be taken after each smoothing step:

$$X_P^{[i+1]} = X_P^{[i]} + \lambda \left( \sum_{j=1}^{n} w_j X_{N_j}^{[i]} - X_P^{[i]} \right)$$

Laplacian step

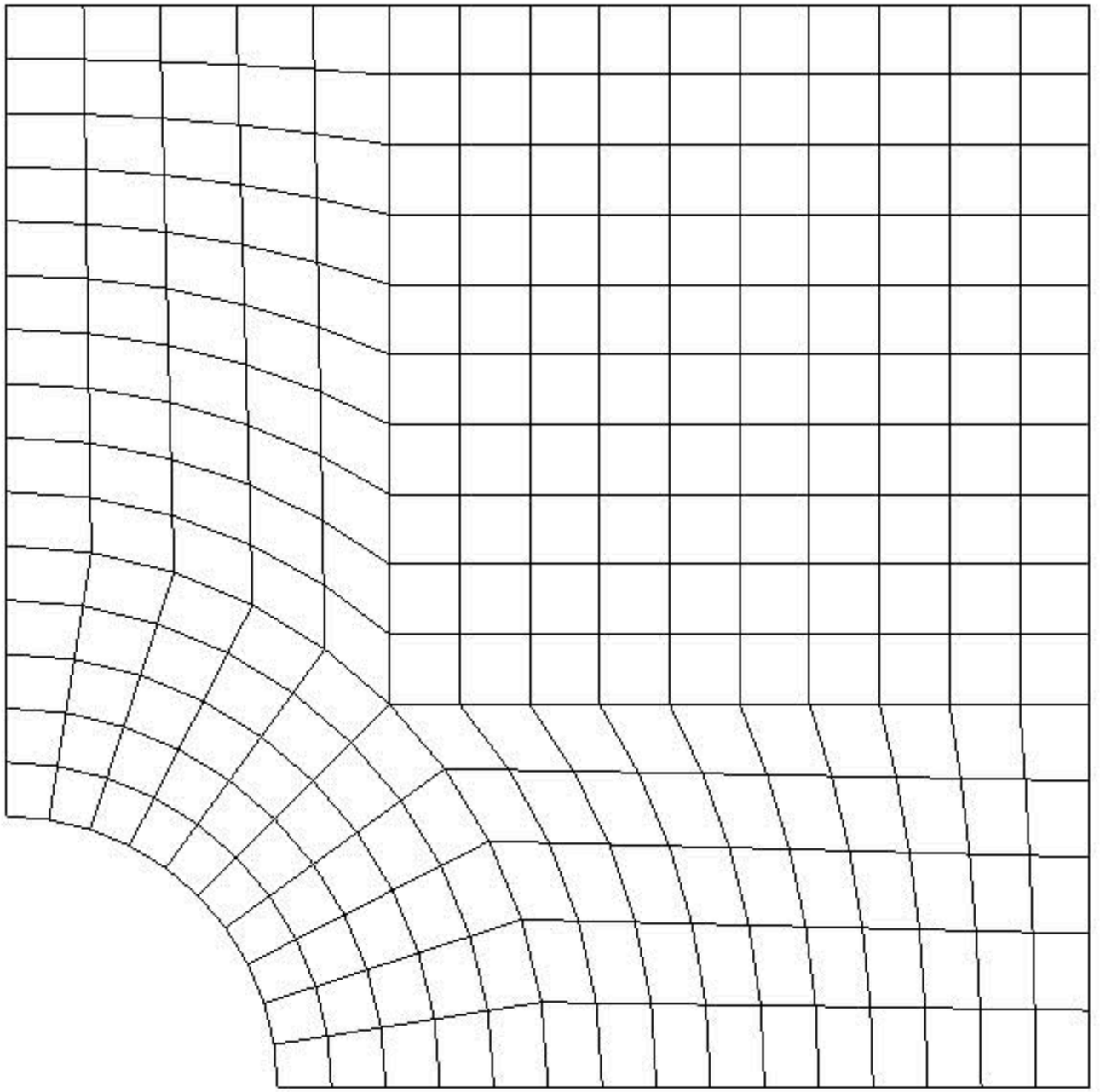
$$X_P^{[i+2]} = X_P^{[i+1]} + \mu \left( \sum_{j=1}^{n} w_j X_{N_j}^{[i+1]} - X_P^{[i+1]} \right)$$

Taubin step

where $0 < \lambda < -\mu$

G. Taubin, Curve and surface smoothing without shrinkage, 1995, IEEE

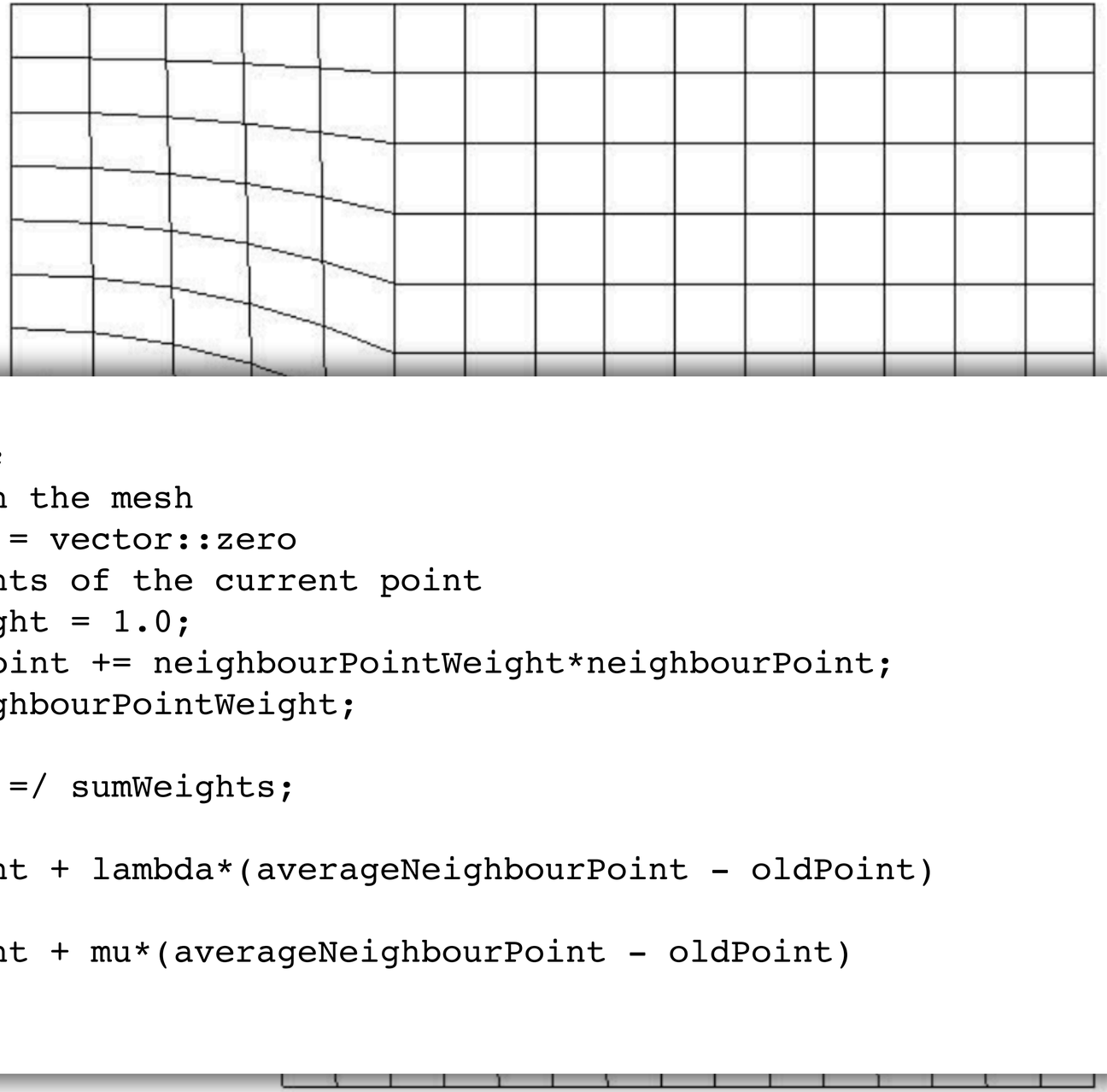
Point-based Laplacian with arithmetic weights and a Taubin expansion step



Adding the Taubin helps seems to make the method less robust on initially bad meshes

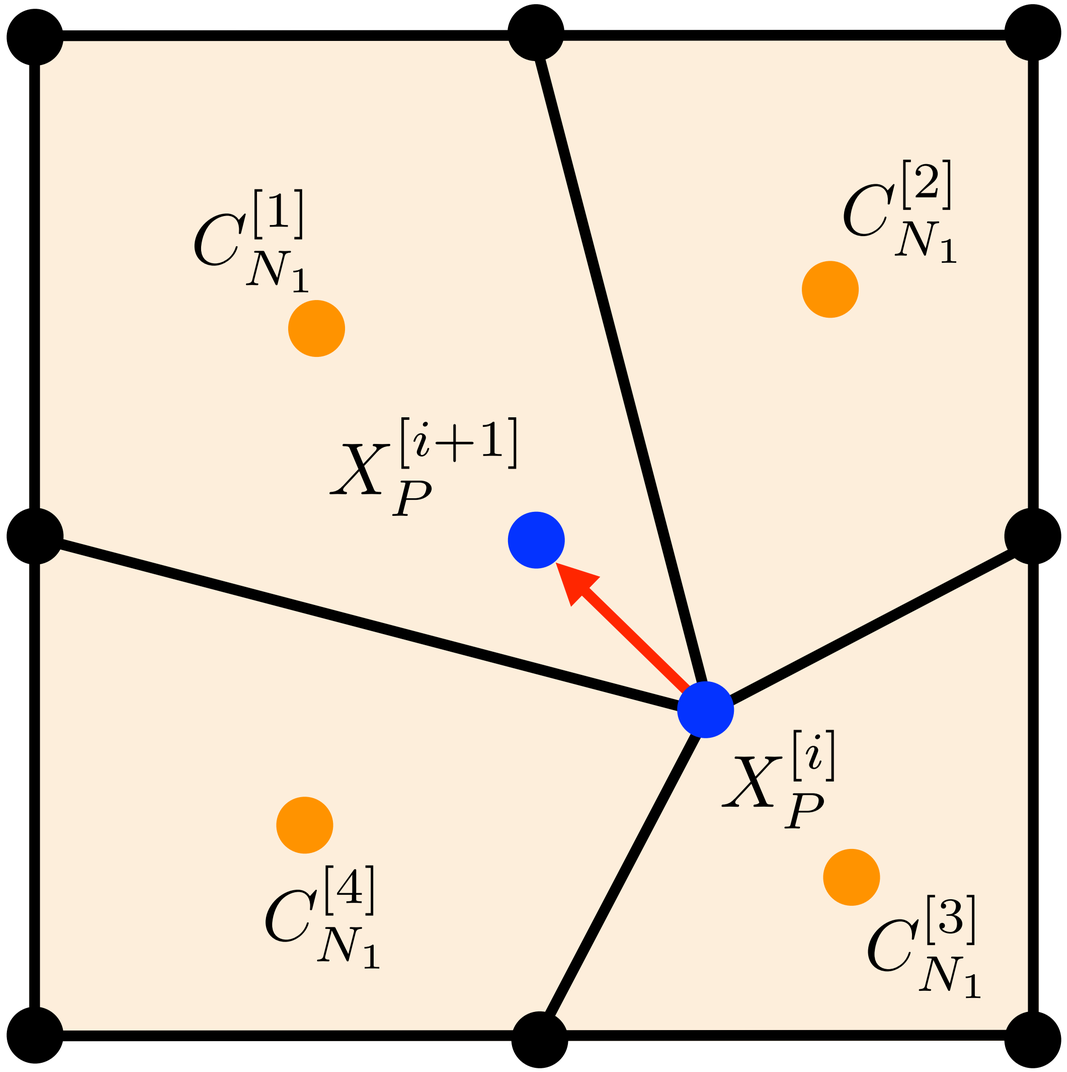
Iteration: 0

# Point-based Laplacian with arithmetic weights and a Taubin expansion step



```
algorithm 2:
lambda = 0.33; mu = -0.34;
for all internal points in the mesh
    averageNeighbourPoint = vector::zero
    for all neighbour points of the current point
        neighbourPointWeight = 1.0;
        averageNeighbourPoint += neighbourPointWeight*neighbourPoint;
        sumWeights += neighbourPointWeight;
    end
    averageNeighbourPoint =/ sumWeights;
    if laplacianStep
        newPoint = oldPoint + lambda*(averageNeighbourPoint - oldPoint)
    else // taubin step
        newPoint = oldPoint + mu*(averageNeighbourPoint - oldPoint)
    end
end
```

point-based Laplacian smoothing with cell-volume weights




## Approach 3

point-based Laplacian with cell-volume weights

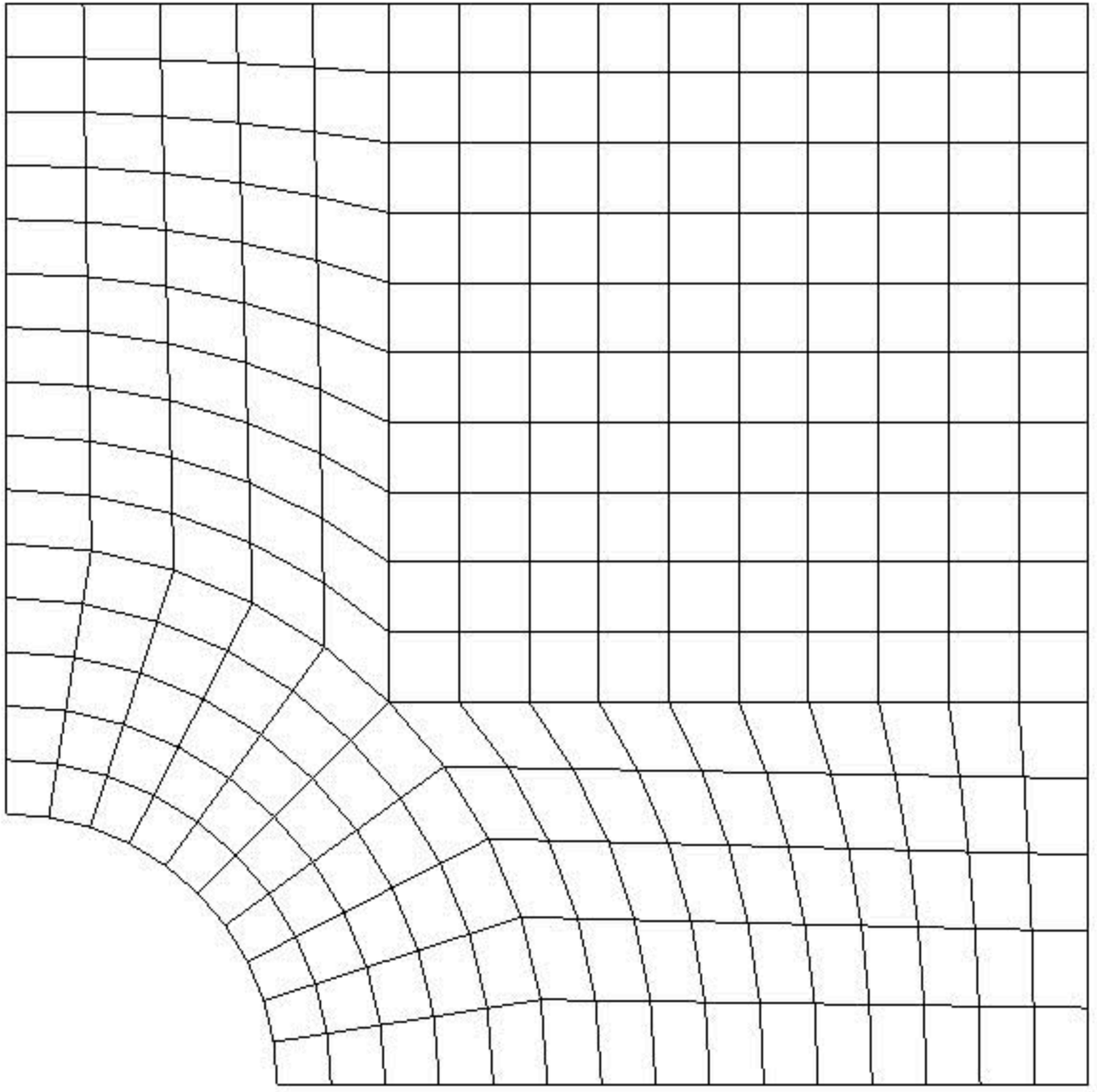
$$X_P^{[i+1]} = \sum_{j=1}^{\text{ncells}} w_j C_{N_j}^{[i]}$$

cell centre

We can use the cell volumes to calculate the weights:

$$w_j = \frac{V_{N_j}^{[i]}}{\sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]}}$$

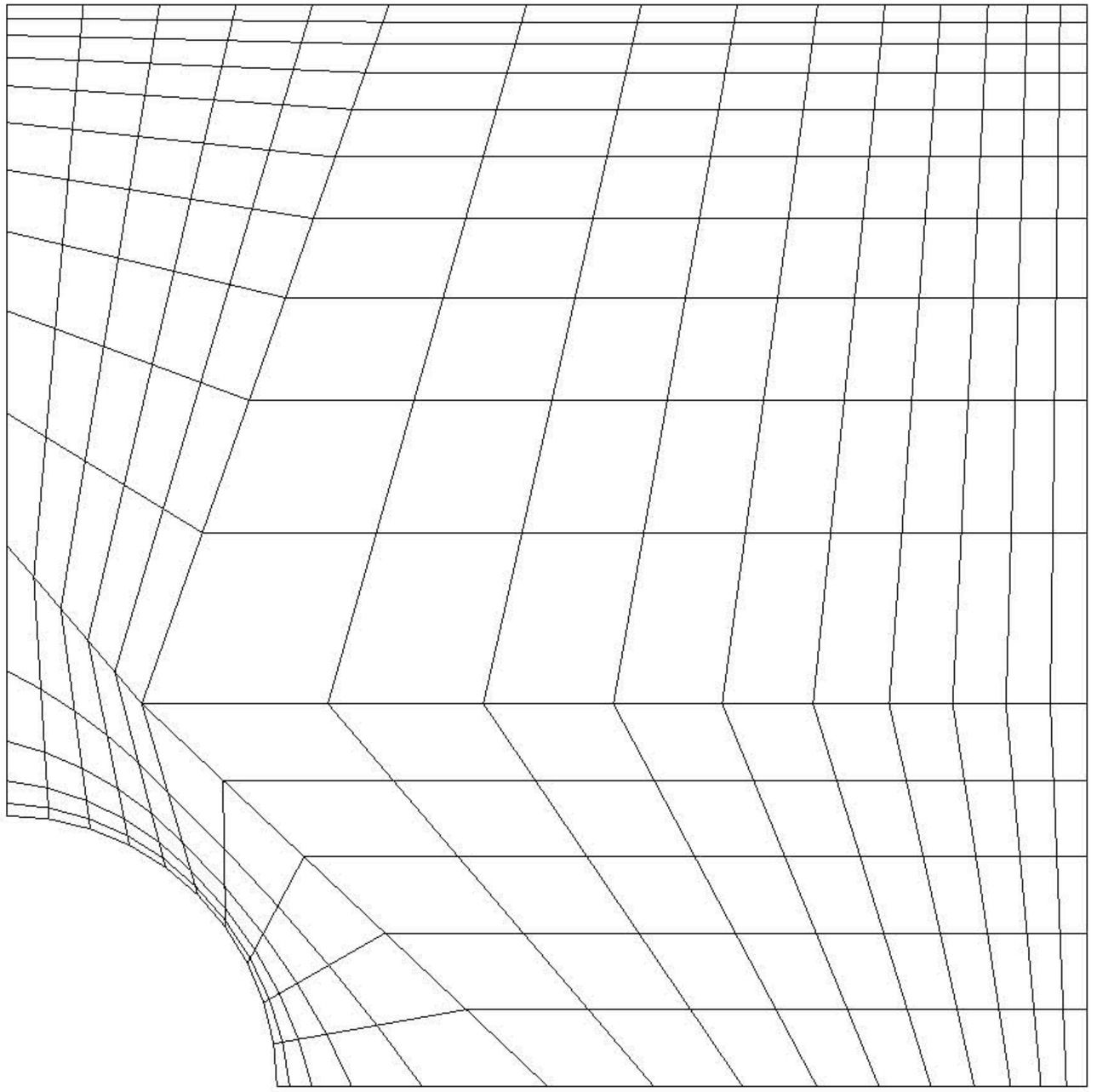
Point-based Laplacian with cell-volume weights



Iteration: 0

Point-based Laplacian with cell-volume weights

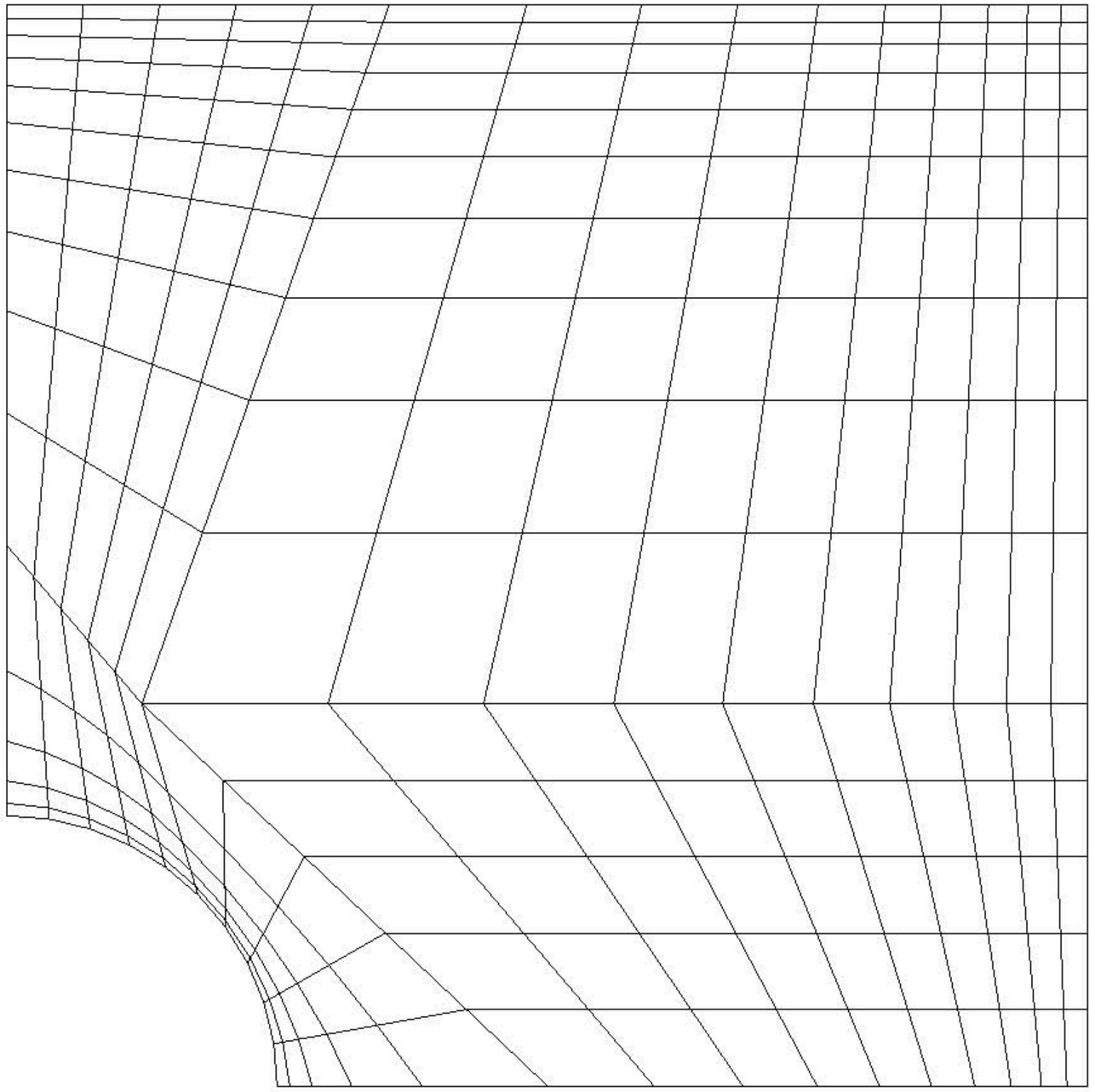
lambda = 1.0



Iteration: 0

Point-based Laplacian with cell-volume weights

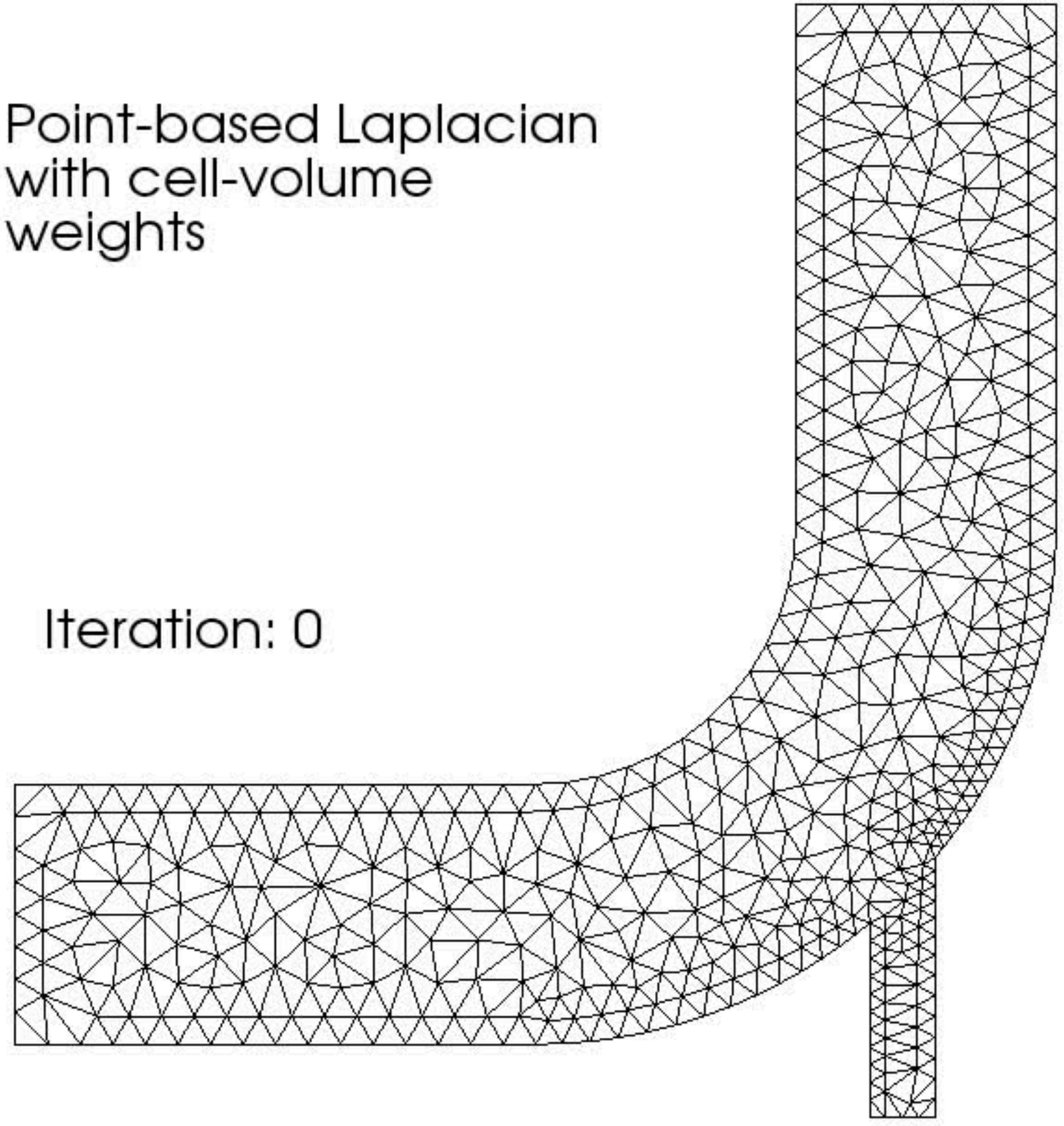
lambda = 0.5



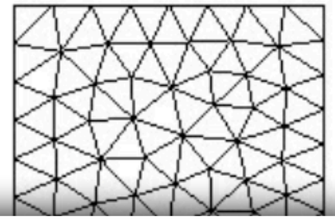
Iteration: 0

Point-based Laplacian
with cell-volume
weights

Iteration: 0



Point-based Laplacian
with cell-volume

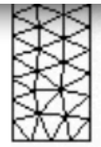


Before

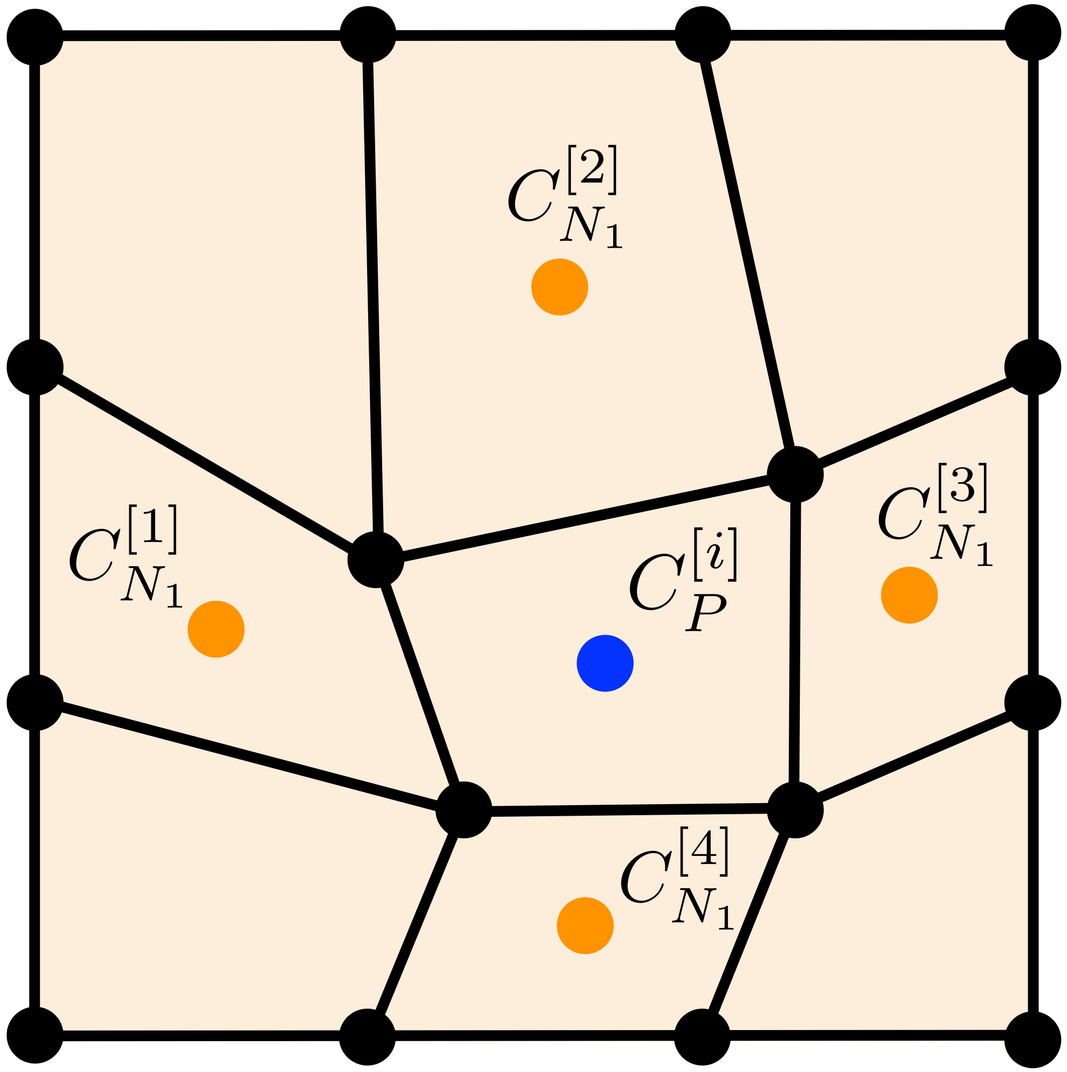
```
    Max aspect ratio = 6.7264 OK.
    Min volume = 0.521792. Max volume = 7.36354.
    Mesh non-orthogonality Max: 36.302 average: 11.1868
    Max skewness = 0.500248 OK.
```

After

```
    Max aspect ratio = 4.57246 OK.
    Min volume = 0.943911. Max volume = 6.3874.
    Mesh non-orthogonality Max: 33.0798 average: 10.467
    Max skewness = 0.363852 OK.
```



cell-based Laplacian smoothing with cell-volume weights



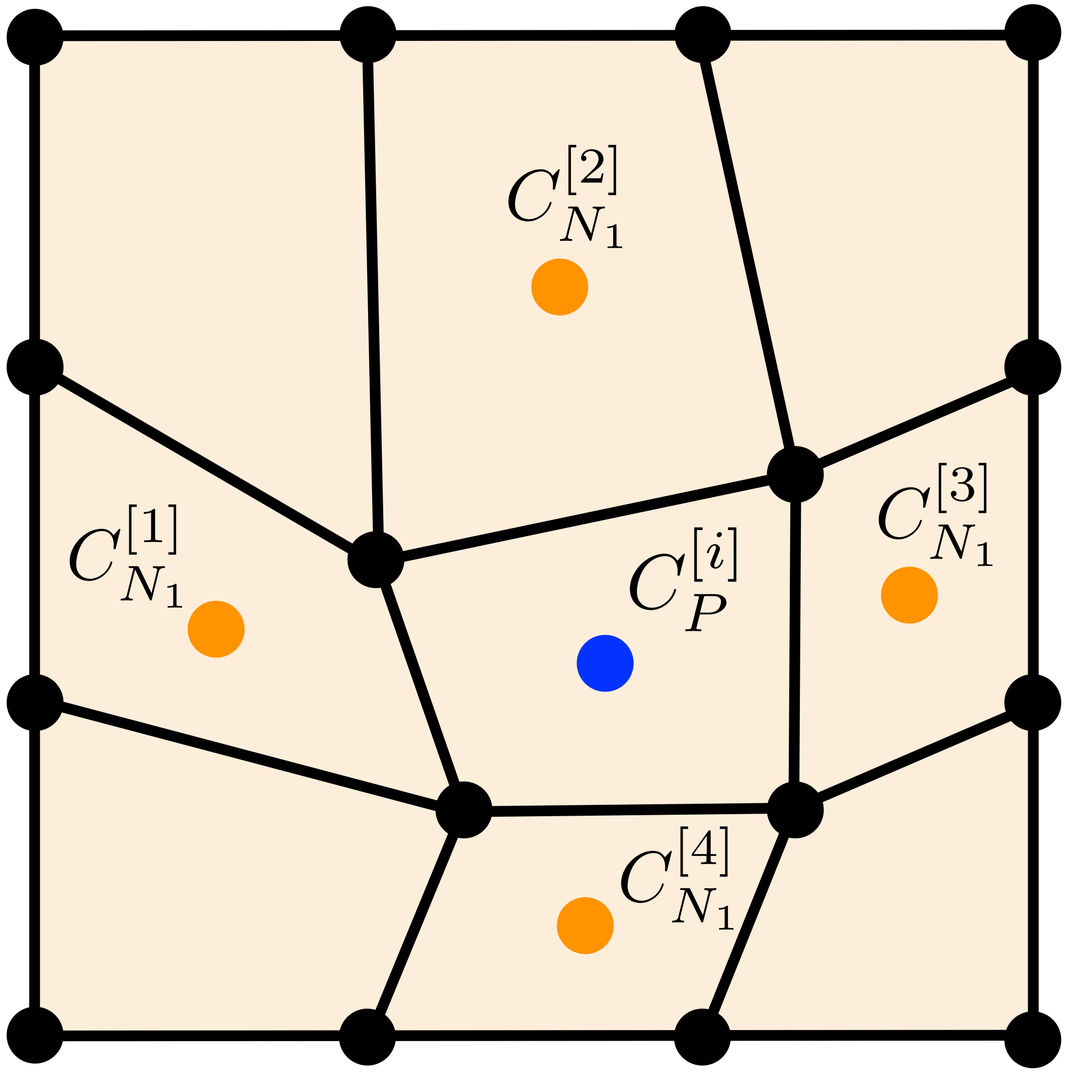

## Approach 3

**cell**-based Laplacian with cell-volume weights

$$C_P^{[i+1]} = \sum_{j=1}^{\text{ncells}} w_j C_{N_j}^{[i]}$$

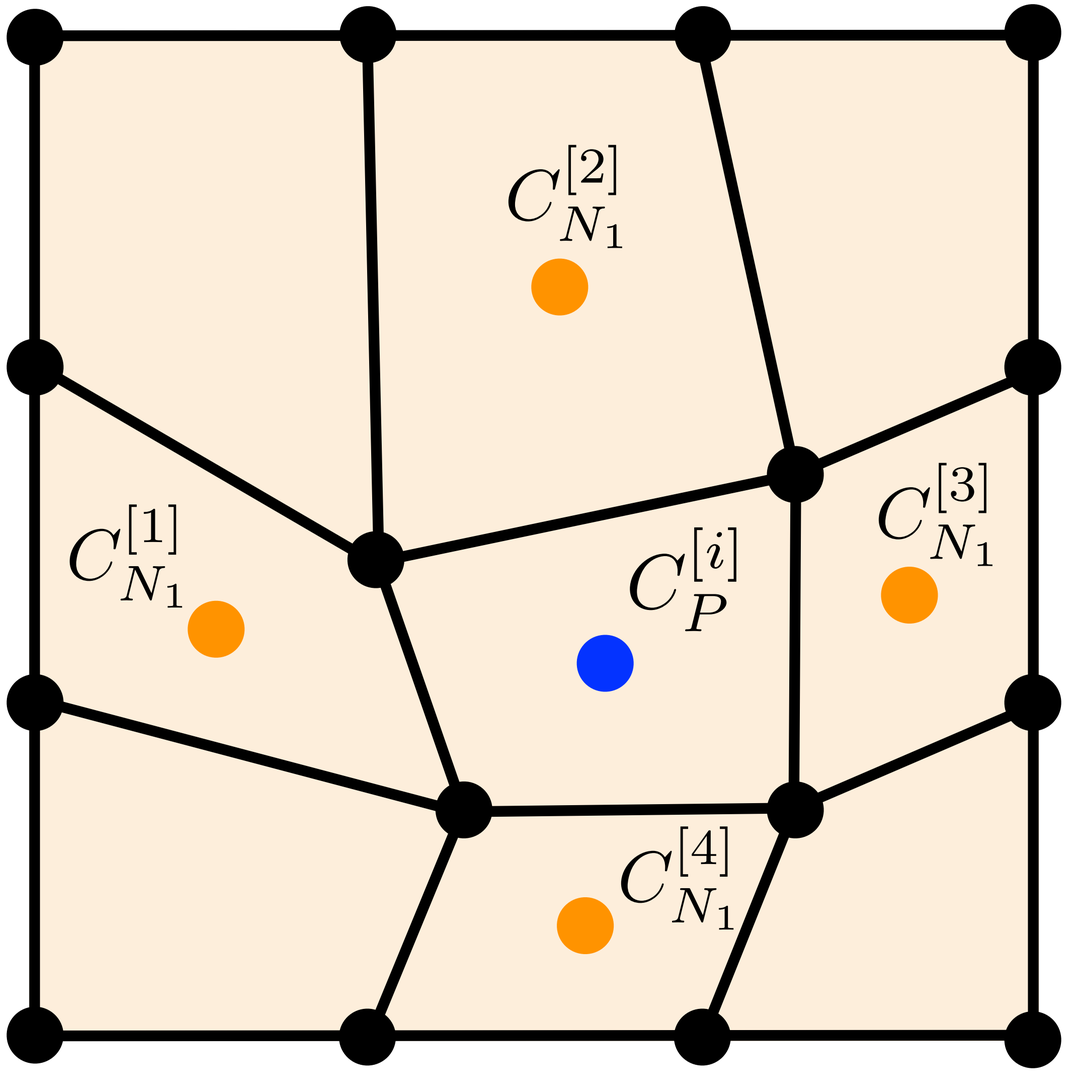
Where the cell-volume weights are calculated as:

$$w_j = \frac{V_{N_j}^{[i]}}{\sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]}}$$




Advantage

It can be implemented in an implicit fashion using the finite volume infrastructure within OpenFOAM.




Disadvantage

However, this approach gives the mesh motion at the cell centres, which must be interpolated to the points

A simple method is to take an arithmetic average.

# Deriving an implicit finite volume approach

The explicit method iterates over the equation to the right until convergence is achieved; note the left-hand side has iterator [i + 1], whereas the right-hand side is [i].

$$C_P^{[i+1]} = \sum_{j=1}^{\text{ncells}} w_j C_{N_j}^{[i]}$$

This equation can be re-written solely in terms of unknown cell-centre positions as:

$$-C_P^{[i+1]} + \sum_{j=1}^{\text{ncells}} w_j C_{N_j}^{[i+1]} = 0$$

$$-C_P^{[i+1]} + \sum_{j=1}^{\text{ncells}} w_j C_{N_j}^{[i+1]} = 0$$

$$-C_P^{[i+1]} + \sum_{j=1}^{\text{ncells}} \left[ \frac{V_{N_j}^{[i]}}{\sum_{k=1}^{\text{ncells}} V_{N_k}^{[i]}} \right] C_{N_j}^{[i+1]} = 0$$

$$-\left[ \sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]} \right] C_P^{[i+1]} + \sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]} C_{N_j}^{[i+1]} = 0$$

Note: this is the definition of a discrete Laplacian operator

$$-\left[\sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]}\right] C_P^{[i+1]} + \sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]} C_{N_j}^{[i+1]} = 0$$

cell centre displacements

$$-\left[\sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]}\right] \left[C_P^{[\text{orig}]} + U_P^{[i+1]}\right] + \sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]} \left[C_{N_j}^{[\text{orig}]} + U_{N_j}^{[i+1]}\right] = 0$$

$$-\left[\sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]}\right] U_P^{[i+1]} + \sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]} U_{N_j}^{[i+1]} = \left[\sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]}\right] C_P^{[\text{orig}]} - \sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]} C_{N_j}^{[\text{orig}]}$$

$$-\left[\sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]}\right] U_P^{[i+1]} + \sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]} U_{N_j}^{[i+1]} = \left[\sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]}\right] C_P^{[\text{orig}]} - \sum_{j=1}^{\text{ncells}} V_{N_j}^{[i]} C_{N_j}^{[\text{orig}]}$$

$$\boldsymbol{\nabla} \cdot (V \boldsymbol{\nabla} \boldsymbol{U}) = -\boldsymbol{\nabla} \cdot (V \boldsymbol{\nabla} \boldsymbol{X})$$

$$\boldsymbol{\nabla} \cdot (V \boldsymbol{\nabla} \boldsymbol{U}) = -\boldsymbol{\nabla} \cdot (V \mathbf{I})$$

$$\boldsymbol{\nabla} \cdot (V \boldsymbol{\nabla} \boldsymbol{U}) = -\boldsymbol{\nabla} V$$

$$\nabla \cdot (V \nabla U) = -\nabla V$$

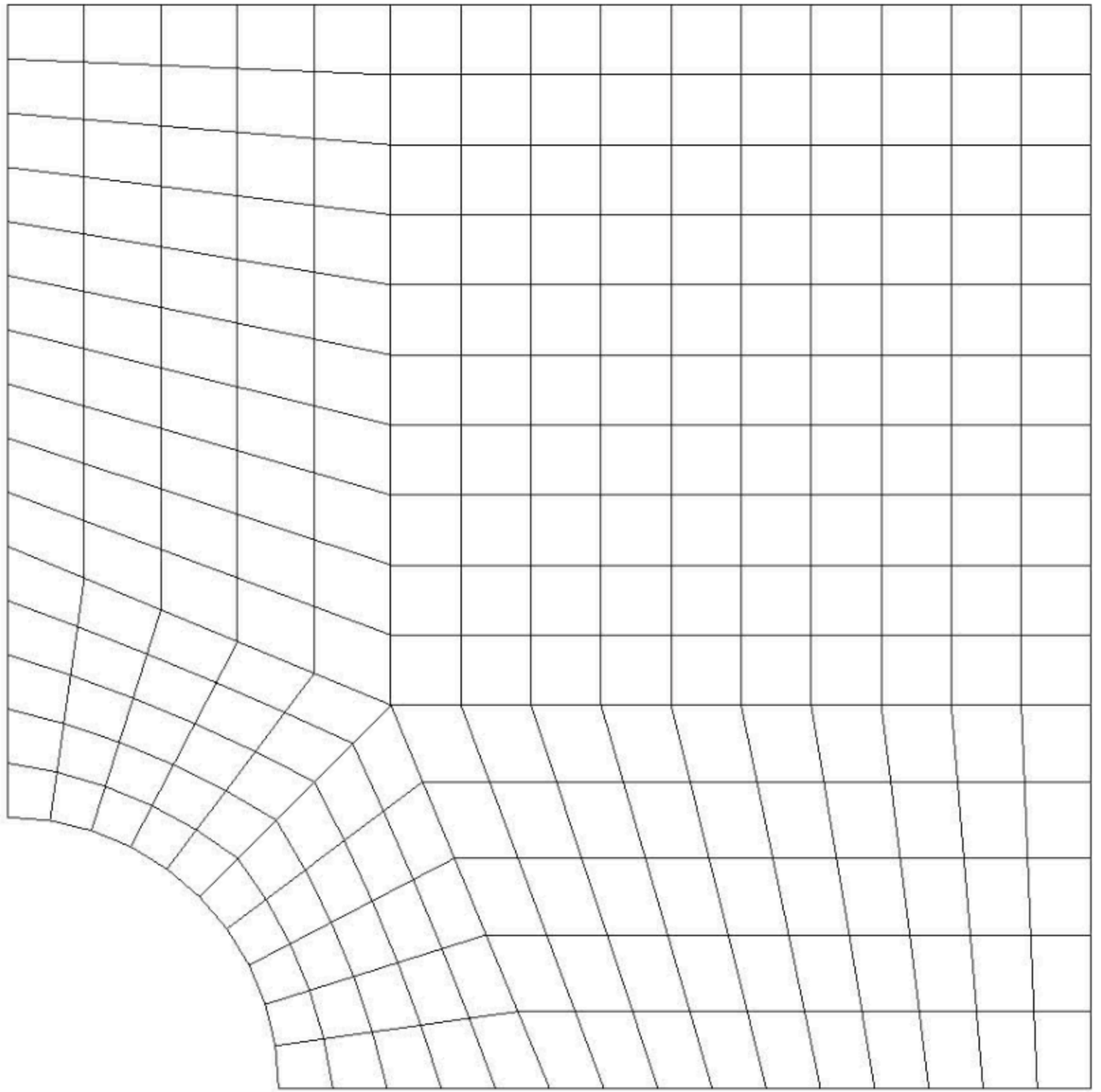
Laplacian of displacement, where the cell-volume field is the diffusivity: cells with smaller volumes will tend to increase in size more

The gradient of the cell-volume field is the driving force: smaller cells will move towards larger cells

The cell volumes are a function of the displacements so some outer iterations may be required

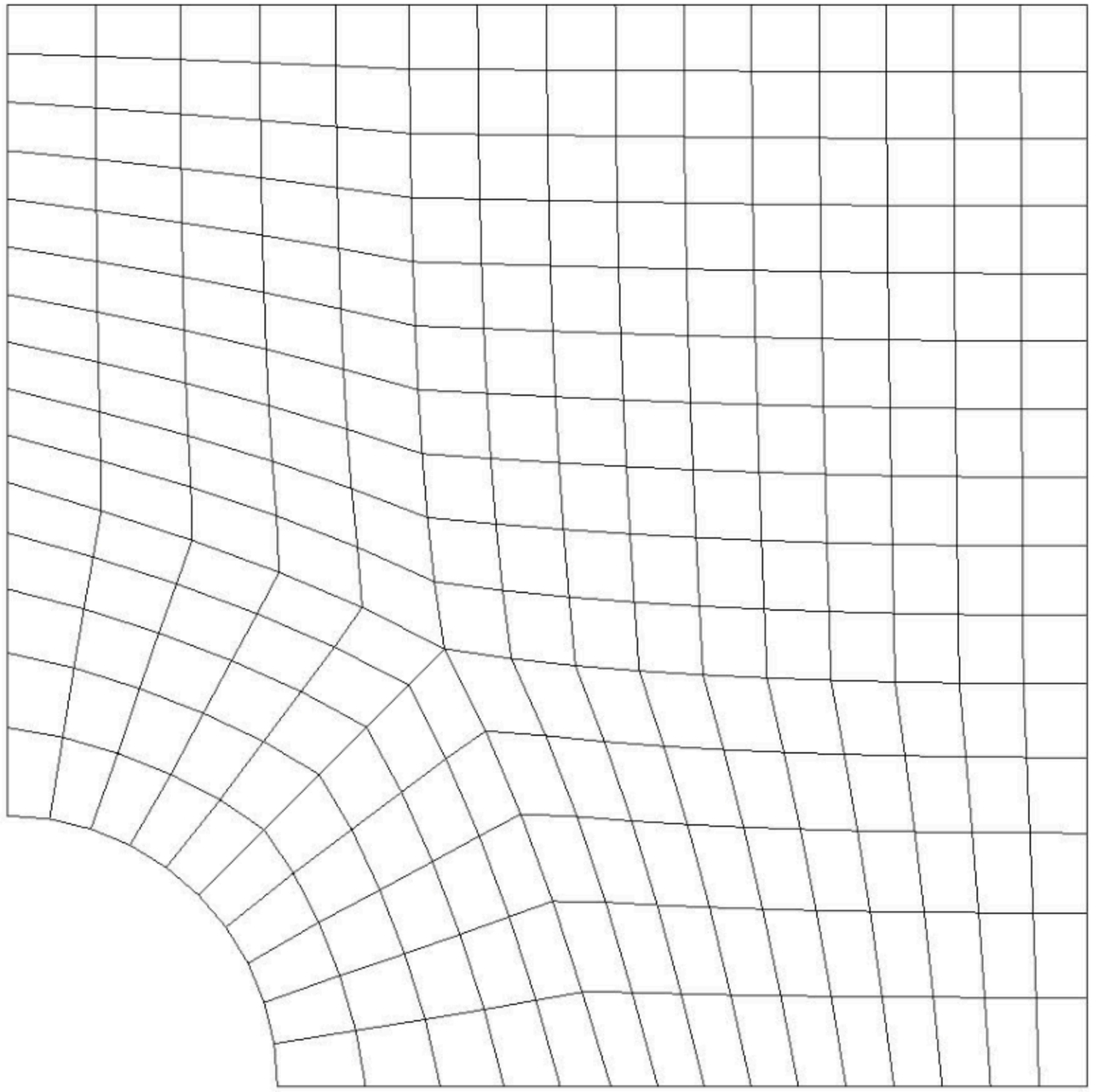
```
solve
(
    fvm::laplacian(cellVolumes, U)
 == -fvc::grad(cellVolumes)
);
```

Implicit
cell-based Laplacian
with cell-volume
weights



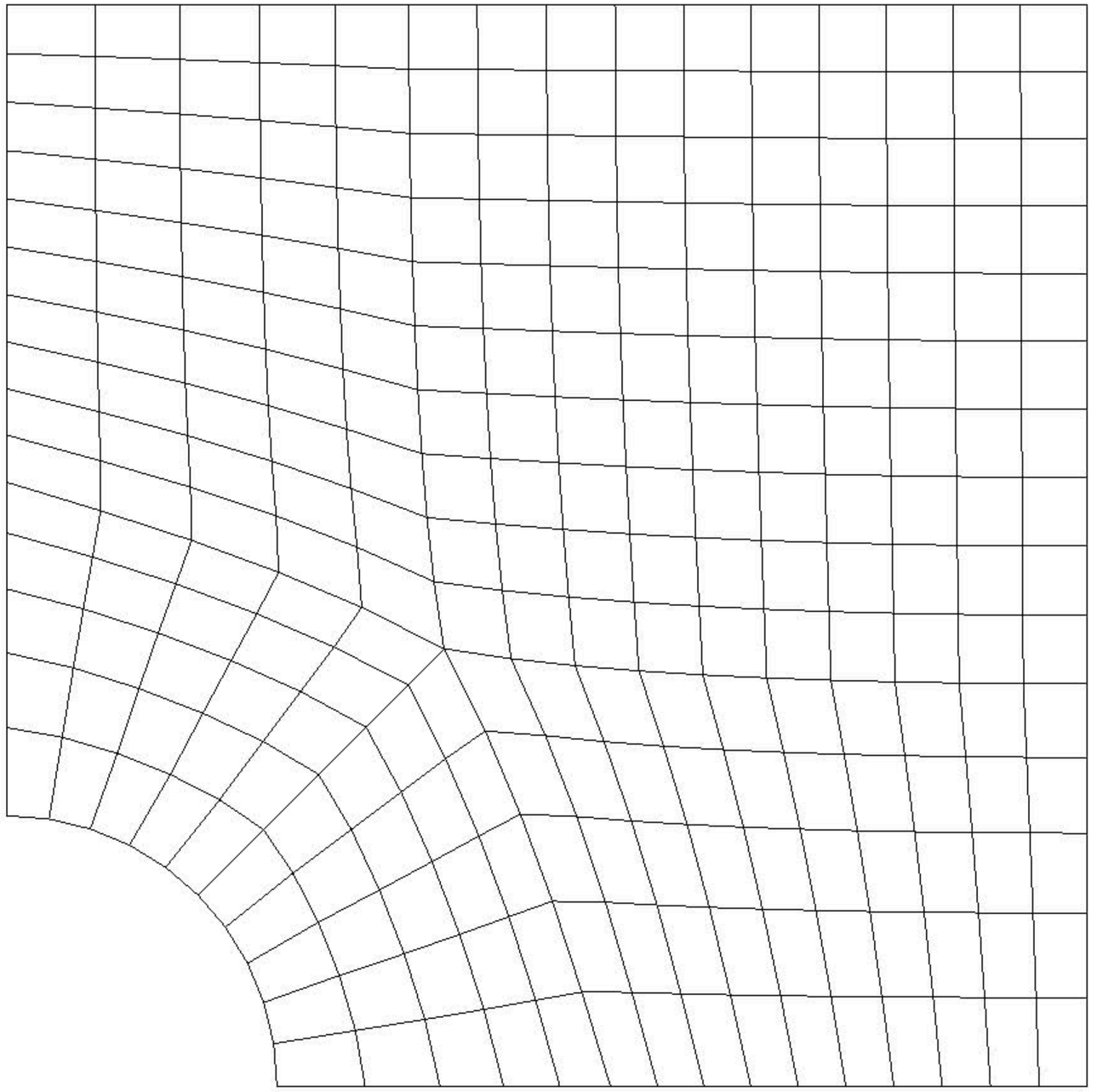
Iteration: 0

Implicit
cell-based Laplacian
with cell-volume
weights



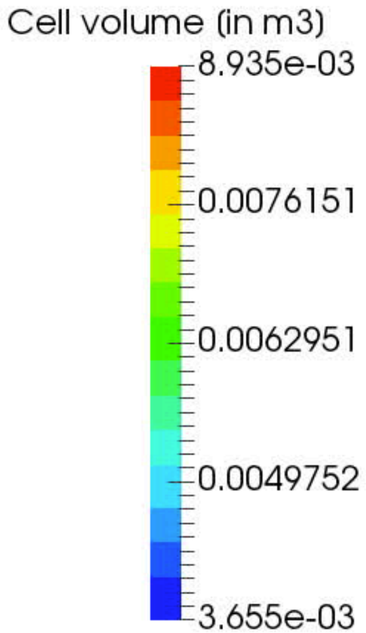
Iteration: 1

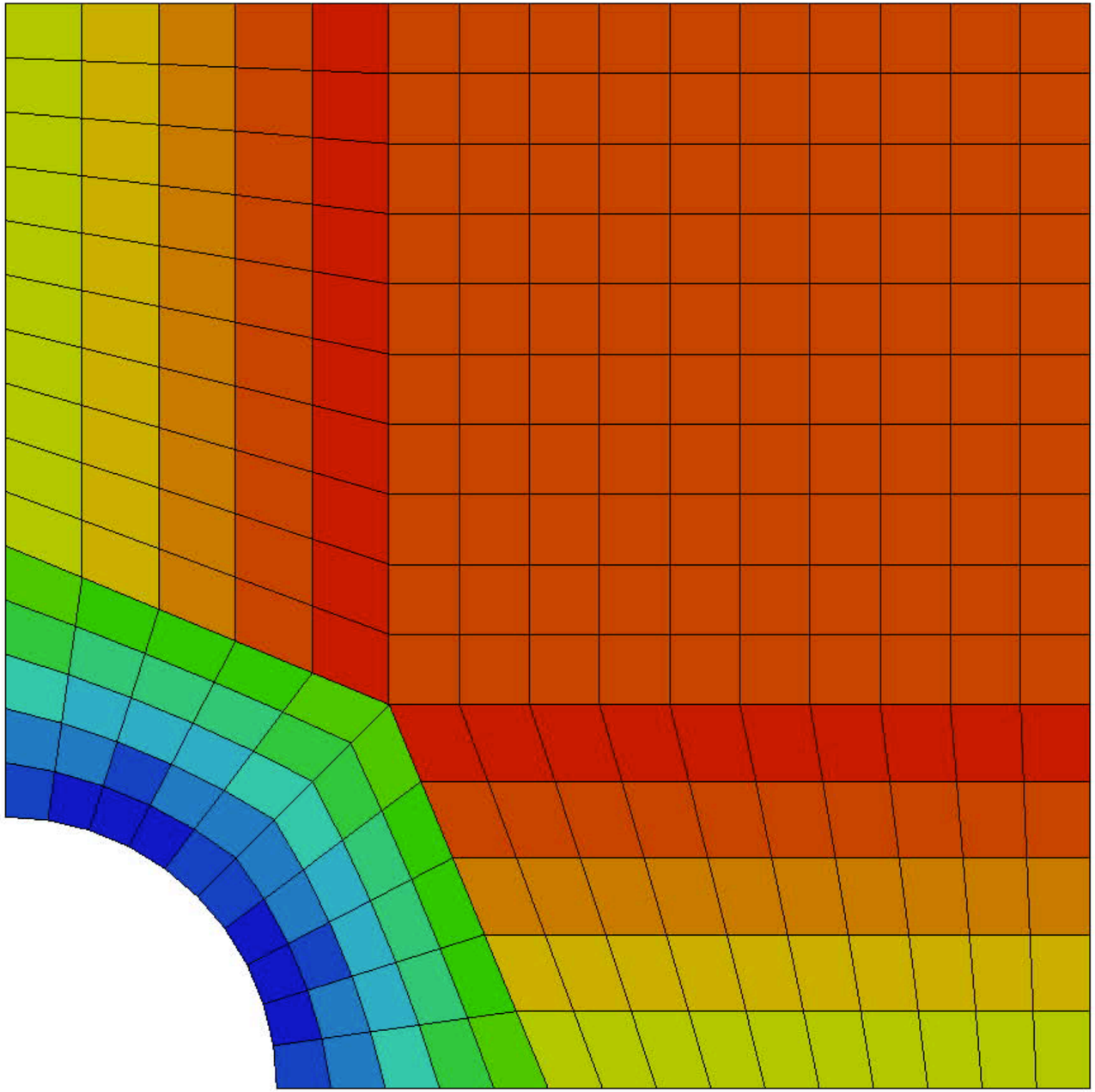
Implicit
cell-based Laplacian
with cell-volume
weights



Iteration: 1

Implicit
cell-based Laplacian
with cell-volume
weights



Iteration: 0

# Summary & Next Steps

# Summary & next steps

- A number of mesh smoothing procedures have been implemented and examined:
  - *explicit* point-based Laplacian
  - *explicit* point-based Laplacian with a Taubin expansion step
  - *explicit* point-based Laplacian with cell-volume weights: most robust
  - *implicit* cell-based Laplacian with cell-volume weights: fastest

Next Steps:

- Examine the smoothing procedures on more complex 2-D and 3-D meshes
- Incorporate the mesh smoothing procedure into an Arbitrary Lagrangian-Eulerian method for elasto-plastic analysis (metal forming)

April 2018

Online International Meeting for Users of OpenFOAM
Published on YouTube

# On the implementation of point-based and cell-based mesh smoothing approaches in OpenFOAM

Philip Cardiff




School of Mechanical and Materials Engineering
University College Dublin