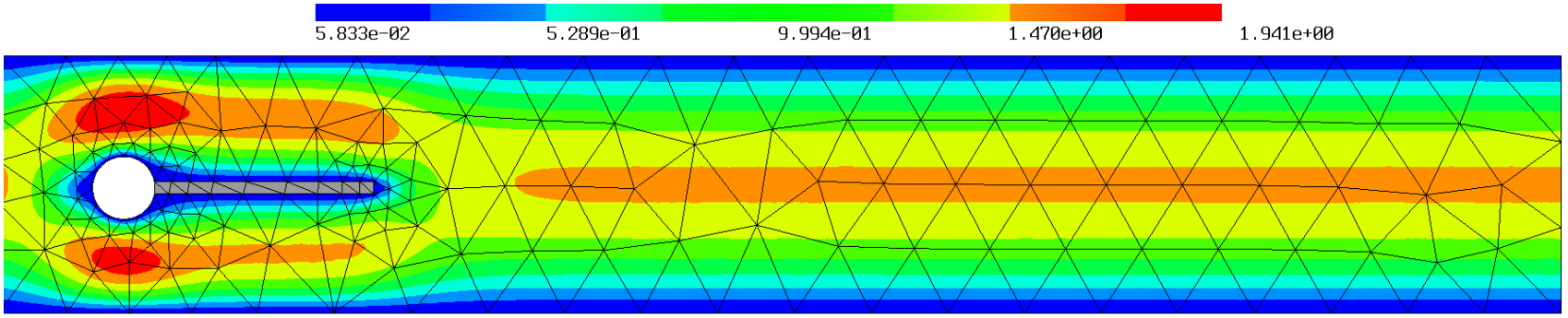
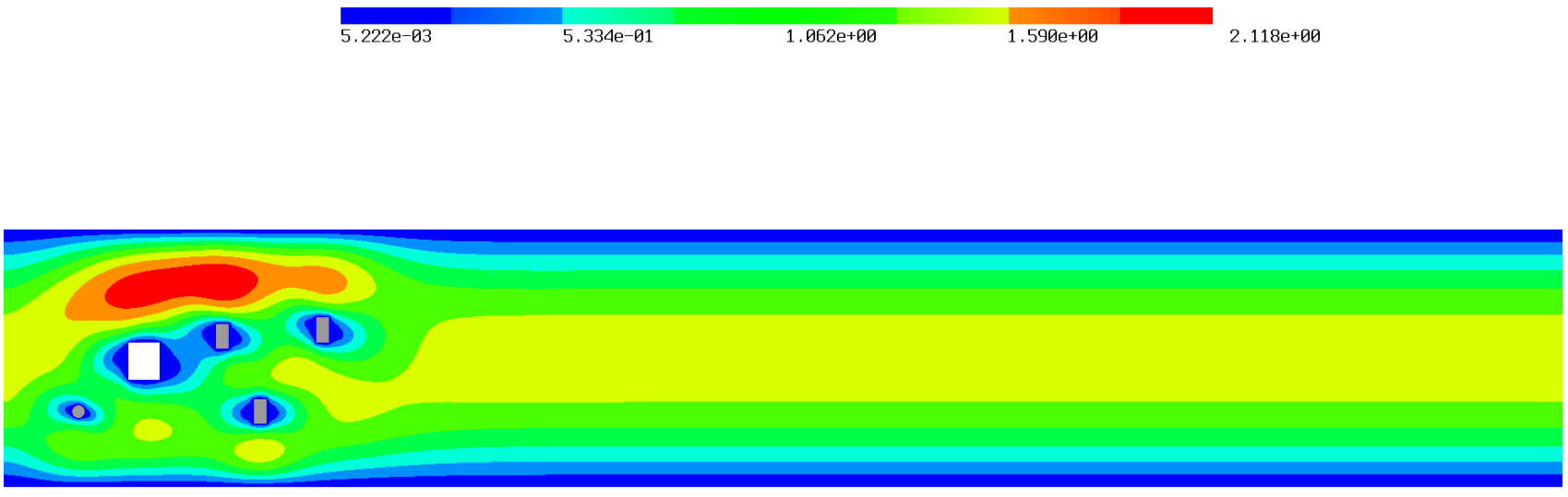
Moving meshes

Michael Neunteufel, Joachim Schöberl

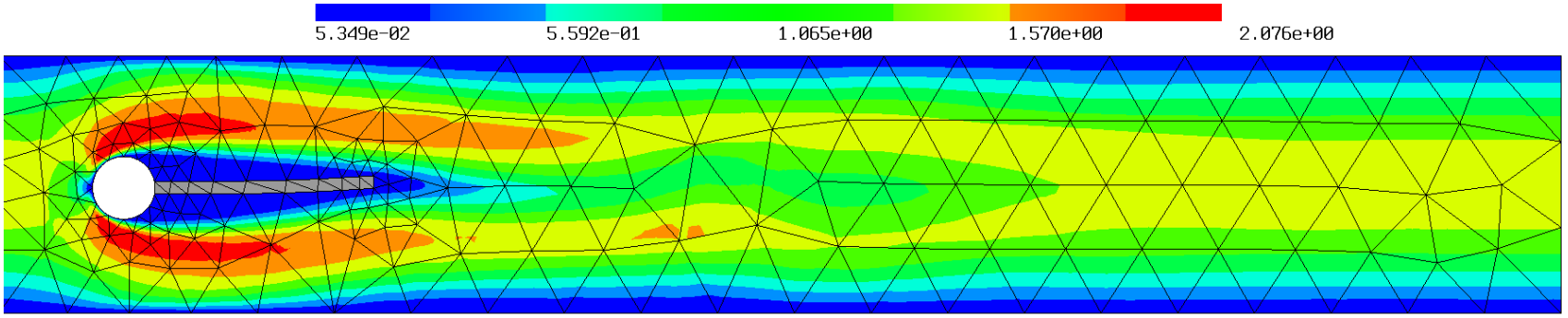
3rd NGSolve Usermeeting, Vienna, July 1-3, 2019

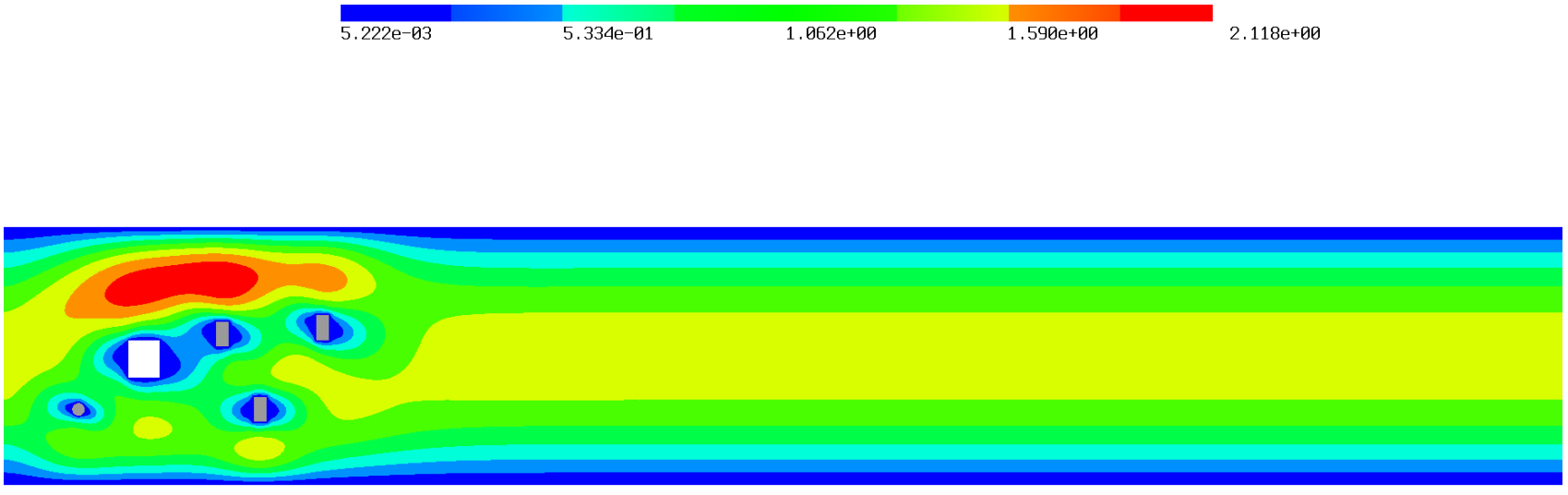
Motivation


5.833e-02
5.289e-01
9.994e-01
1.470e+00
1.941e+00


5.222e-03
5.334e-01
1.062e+00
1.590e+00
2.118e+00

Motivation

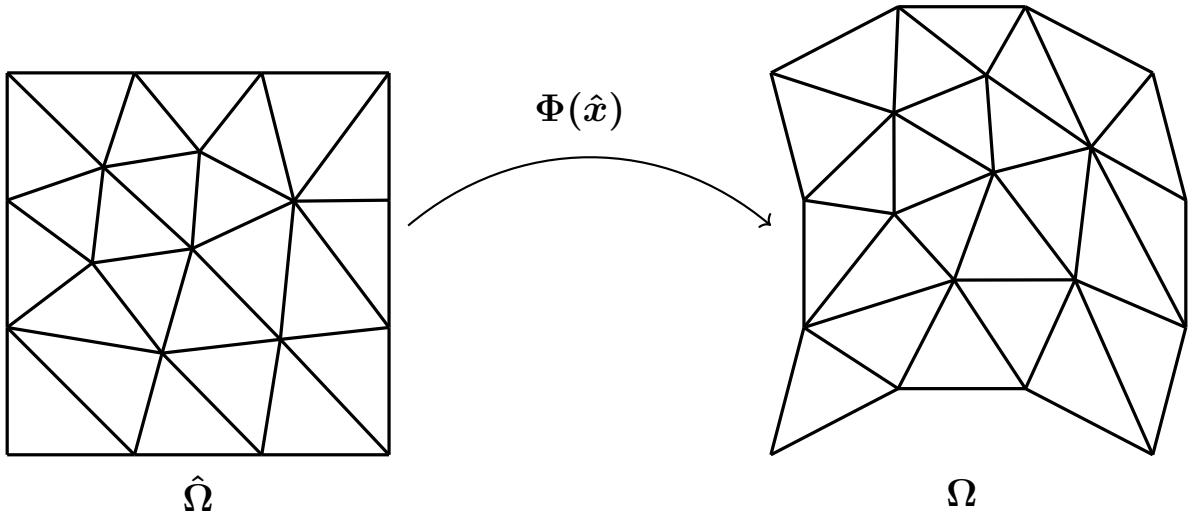

5.349e-02
5.592e-01
1.065e+00
1.570e+00
2.076e+00


5.222e-03
5.334e-01
1.062e+00
1.590e+00
2.118e+00

Contents

Arbitrary Lagrangian Eulerian

ALE spatial transformation

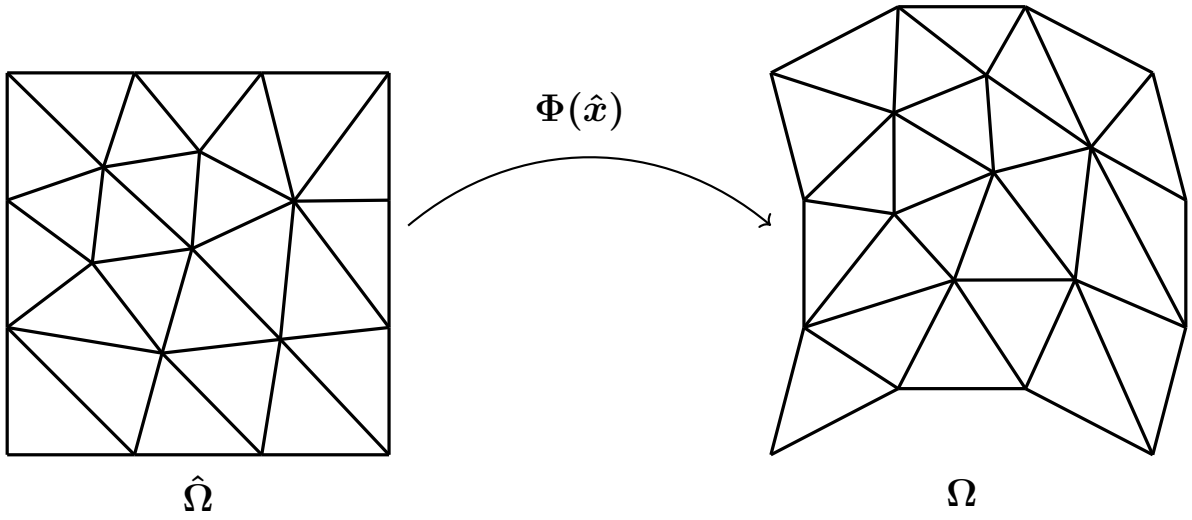


$$-\Delta_x u = f \qquad \text{in } \Omega$$

$$u = u_D \qquad \text{on } \Gamma$$

$$\int_\Omega \nabla_x u \nabla_x v \, dx = \int_\Omega f v \, dx$$

ALE spatial transformation



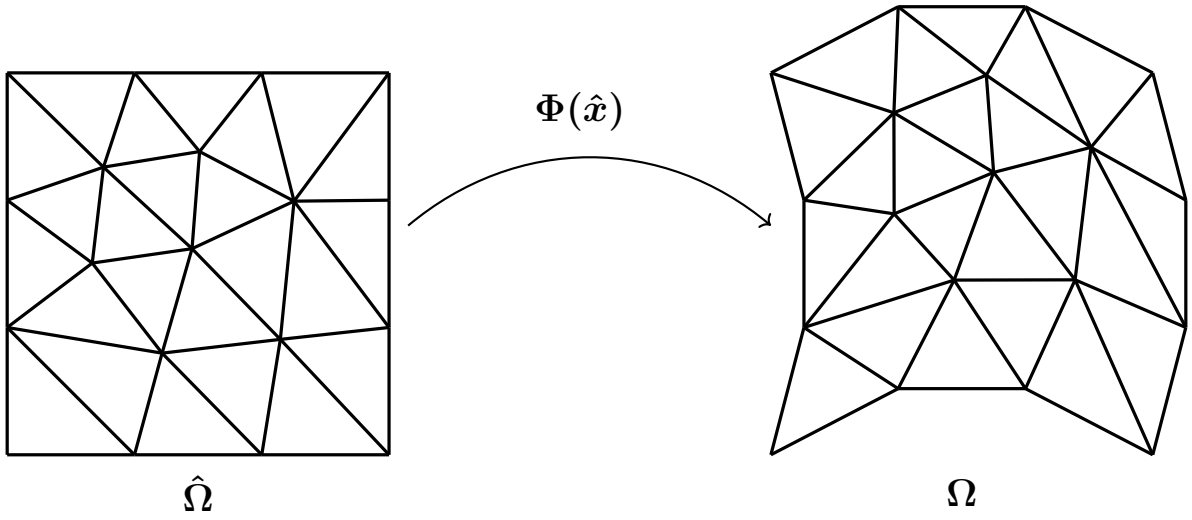
$$-\Delta_x u = f \qquad \text{in } \Omega$$
$$u = u_D \qquad \text{on } \Gamma$$

$$u \circ \Phi = \hat{u}$$

$$\underbrace{\nabla_{\hat{x}} \Phi^T}_{=\boldsymbol{F}^T} (\nabla_x u) \circ \Phi = \nabla_{\hat{x}} \hat{u}$$

$$\int_\Omega \nabla_x u \nabla_x v \, dx = \int_\Omega f v \, dx$$

ALE spatial transformation



$$-\Delta_x u = f \qquad \text{in } \Omega$$

$$u = u_D \qquad \text{on } \Gamma$$

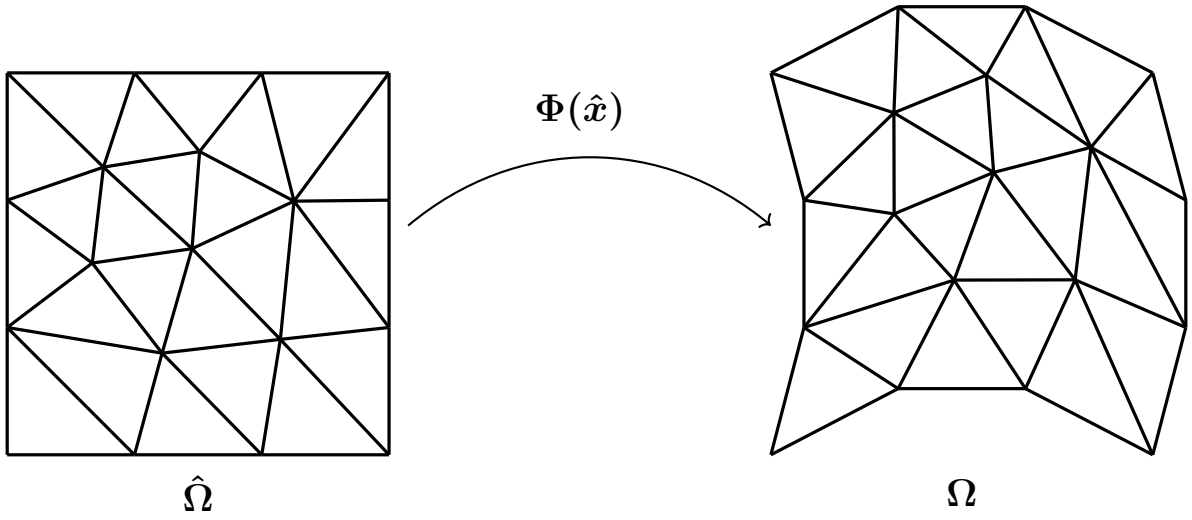
$$u \circ \Phi = \hat{u}$$

$$\underbrace{\nabla_{\hat{x}} \Phi^T}_{=\boldsymbol{F}^T} (\nabla_x u) \circ \Phi = \nabla_{\hat{x}} \hat{u}$$

$$\int_{\hat{\Omega}} J \nabla_x u \circ \Phi \nabla_x v \circ \Phi \, d\hat{x} = \int_{\hat{\Omega}} J f \circ \Phi v \circ \Phi \, d\hat{x}$$

$$J = \det(\boldsymbol{F})$$

ALE spatial transformation



$$-\Delta_x u = f \qquad \text{in } \Omega$$

$$u = u_D \qquad \text{on } \Gamma$$

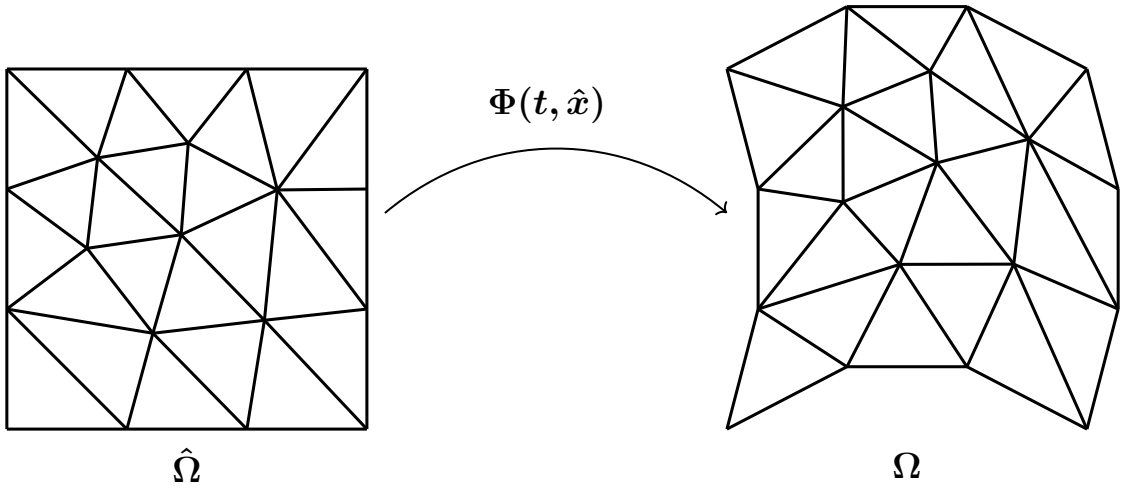
$$u \circ \Phi = \hat{u}$$

$$\underbrace{\nabla_{\hat{x}} \Phi^T}_{=\boldsymbol{F}^T} (\nabla_x u) \circ \Phi = \nabla_{\hat{x}} \hat{u}$$

$$J = \det(\boldsymbol{F})$$

$$\int_{\hat{\Omega}} J \boldsymbol{F}^{-T} \nabla_{\hat{x}} \hat{u} \boldsymbol{F}^{-T} \nabla_{\hat{x}} \hat{v} \, d\hat{x} = \int_{\hat{\Omega}} J \hat{f} \hat{v} \, d\hat{x}$$

ALE time transformation

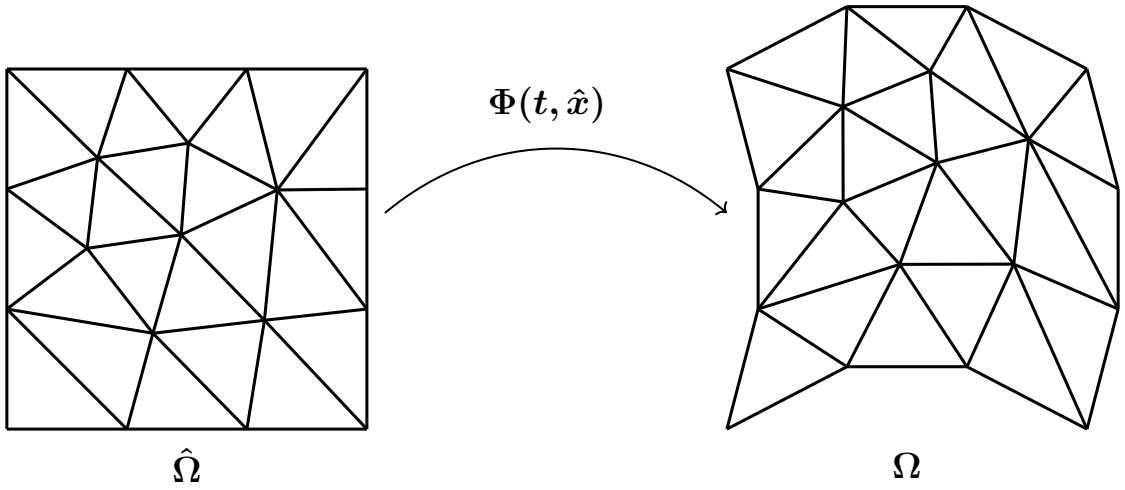


$$\frac{\partial u}{\partial t} - \Delta_x u = f \qquad \text{in } \Omega$$

$$u = u_D \qquad \text{on } \Gamma$$

$$\int_\Omega \frac{\partial u}{\partial t} v \, dx$$

ALE time transformation



$$\frac{\partial u}{\partial t} - \Delta_x u = f \qquad \text{in } \Omega$$

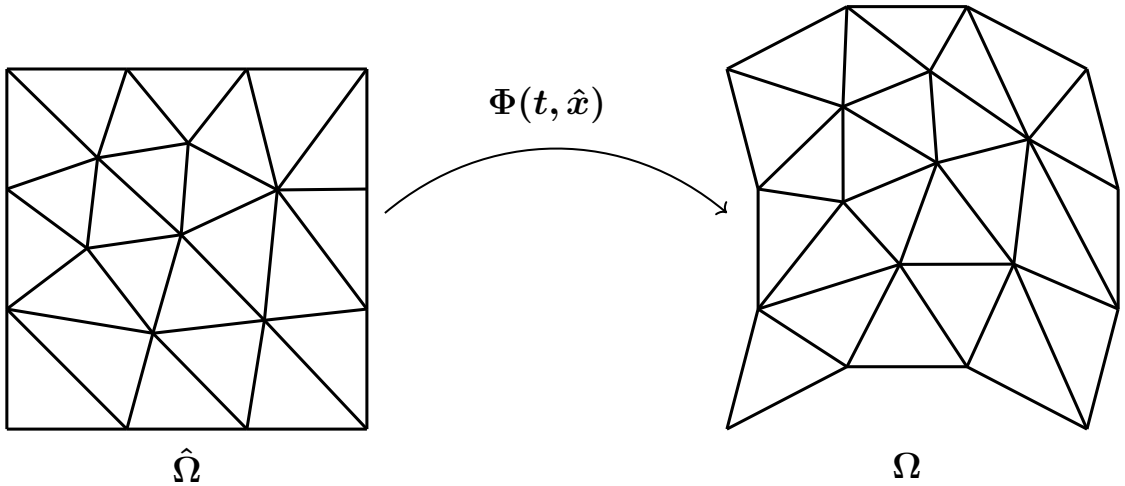
$$u = u_D \qquad \text{on } \Gamma$$

$$\int_\Omega \frac{\partial u}{\partial t} v \, dx$$

$$u(t, \Phi(t, \hat{x})) = \hat{u}(t, \hat{x})$$

$$\frac{\partial u}{\partial t} + \nabla_x u \frac{\partial \Phi}{\partial t} = \frac{\partial \hat{u}}{\partial t}$$

ALE time transformation



$$\frac{\partial u}{\partial t} - \Delta_x u = f \qquad \text{in } \Omega$$

$$u = u_D \qquad \text{on } \Gamma$$

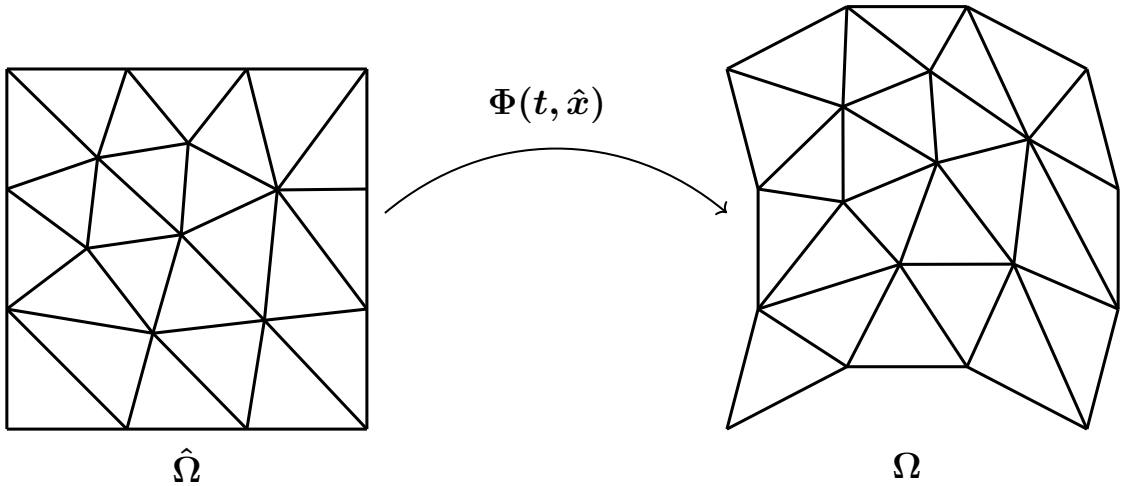
$$\int_\Omega \frac{\partial u}{\partial t} v \, dx$$

$$u(t, \Phi(t, \hat{x})) = \hat{u}(t, \hat{x})$$

$$\frac{\partial u}{\partial t} + \nabla_x u \frac{\partial \Phi}{\partial t} = \frac{\partial \hat{u}}{\partial t}$$

$$\frac{\partial u}{\partial t} + \boldsymbol{F}^{-T} \nabla_{\hat{x}} \hat{u} \frac{\partial \Phi}{\partial t} = \frac{\partial \hat{u}}{\partial t}$$

ALE time transformation



$$\frac{\partial u}{\partial t} - \Delta_x u = f \qquad \text{in } \Omega$$

$$u = u_D \qquad \text{on } \Gamma$$

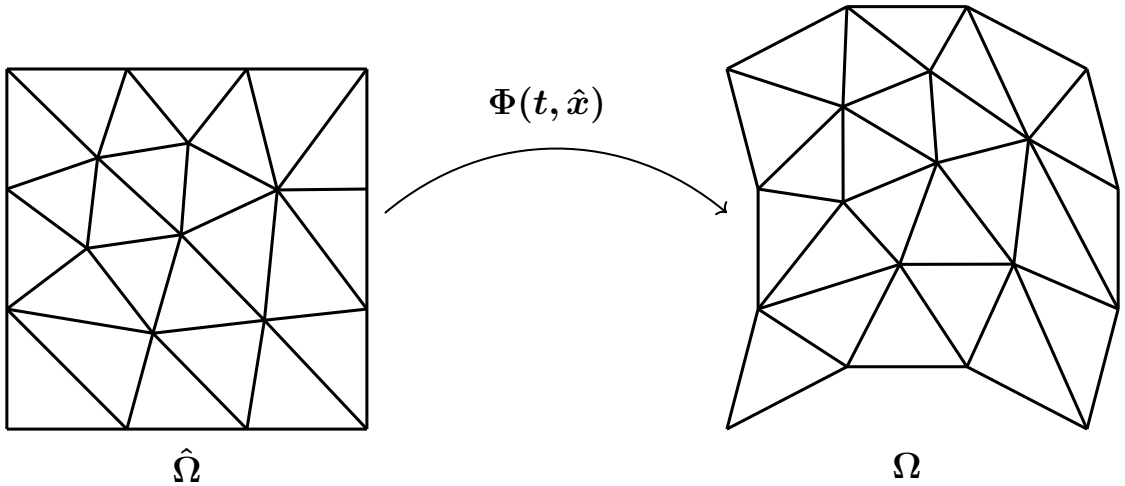
$$\int_{\hat{\Omega}} J \frac{\partial u}{\partial t} \circ \Phi v \circ \Phi \, d\hat{x}$$

$$u(t, \Phi(t, \hat{x})) = \hat{u}(t, \hat{x})$$

$$\frac{\partial u}{\partial t} + \nabla_x u \frac{\partial \Phi}{\partial t} = \frac{\partial \hat{u}}{\partial t}$$

$$\frac{\partial u}{\partial t} + \boldsymbol{F}^{-T} \nabla_{\hat{x}} \hat{u} \frac{\partial \Phi}{\partial t} = \frac{\partial \hat{u}}{\partial t}$$

ALE time transformation



$$\frac{\partial u}{\partial t} - \Delta_x u = f \qquad \text{in } \Omega$$

$$u = u_D \qquad \text{on } \Gamma$$

$$u(t, \Phi(t, \hat{x})) = \hat{u}(t, \hat{x})$$

$$\frac{\partial u}{\partial t} + \nabla_x u \frac{\partial \Phi}{\partial t} = \frac{\partial \hat{u}}{\partial t}$$

$$\frac{\partial u}{\partial t} + \boldsymbol{F}^{-T} \nabla_{\hat{x}} \hat{u} \frac{\partial \Phi}{\partial t} = \frac{\partial \hat{u}}{\partial t}$$

$$\int_{\hat{\Omega}} J\left(\frac{\partial \hat{u}}{\partial t} - \boldsymbol{F}^{-T} \nabla_{\hat{x}} \hat{u} \frac{\partial \Phi}{\partial t}\right) \hat{v} \, d\hat{x}$$

ALE transformations

$$\frac{\partial u}{\partial t} - \Delta_x u = f \qquad \text{in } \Omega$$

$$u = u_D \qquad \text{on } \Gamma$$

$$\int_{\hat{\Omega}} J\left(\frac{\partial \hat{u}}{\partial t}\hat{v} - \boldsymbol{F}^{-T}\nabla_{\hat{x}}\hat{u}\frac{\partial \Phi}{\partial t}\hat{v} + \boldsymbol{F}^{-T}\nabla_{\hat{x}}\hat{u}\boldsymbol{F}^{-T}\nabla_{\hat{x}}\hat{v}\right) d\hat{x} = \int_{\hat{\Omega}} J\hat{f}\hat{v}\, d\hat{x}$$

ALE transformations

$$\frac{\partial u}{\partial t} - \Delta_x u = f \qquad \text{in } \Omega$$

$$u = u_D \qquad \text{on } \Gamma$$

$$\int_{\hat{\Omega}} J\left(\frac{\partial \hat{u}}{\partial t}\hat{v} - \boldsymbol{F}^{-T}\nabla_{\hat{x}}\hat{u}\frac{\partial \Phi}{\partial t}\hat{v} + \boldsymbol{F}^{-T}\nabla_{\hat{x}}\hat{u}\boldsymbol{F}^{-T}\nabla_{\hat{x}}\hat{v}\right) d\hat{x} = \int_{\hat{\Omega}} J\hat{f}\hat{v}\, d\hat{x}$$

- $\frac{\partial \Phi}{\partial t}$ mesh velocity

ALE transformations

$$\frac{\partial u}{\partial t} - \Delta_x u = f \qquad \text{in } \Omega$$

$$u = u_D \qquad \text{on } \Gamma$$

$$\int_{\hat{\Omega}} J\left(\frac{\partial \hat{u}}{\partial t}\hat{v} - \boldsymbol{F}^{-T}\nabla_{\hat{x}}\hat{u}\frac{\partial \Phi}{\partial t}\hat{v} + \boldsymbol{F}^{-T}\nabla_{\hat{x}}\hat{u}\boldsymbol{F}^{-T}\nabla_{\hat{x}}\hat{v}\right) d\hat{x} = \int_{\hat{\Omega}} J\hat{f}\hat{v}\, d\hat{x}$$

- $\frac{\partial \Phi}{\partial t}$ mesh velocity
- $\mathrm{div}_x \underline{u} = \mathrm{tr}(\nabla_{\hat{x}}\underline{\hat{u}}\boldsymbol{F}^{-1})$

ALE implementation method 1

- Given mesh deformation Φ
- Assemble every time step

```
gfset     = GridFunction(fesset)
gfsetold = GridFunction(fesset)

F = Grad(gfset) + Id(2)
J = Det(F)

diff = J*InnerProduct(Inv(F).trans*grad(u), Inv(F).
    trans*grad(v))*dx
mass = J*u*v*dx
conv = -J*InnerProduct(Inv(F).trans*grad(u), 1/tau*(
    gfset-gfsetold))*v*dx
```

ALE implementation method 2

- SetDeformation function/argument

$$\int_{\hat{\Omega}} \left(\frac{\partial \hat{u}}{\partial t}\hat{v} - \nabla_{\hat{x}}\hat{u}\frac{\partial \Phi}{\partial t}\hat{v} + \nabla_{\hat{x}}\hat{u}\nabla_{\hat{x}}\hat{v}\right) d\hat{x} = \int_{\hat{\Omega}} \hat{f}\hat{v}\, d\hat{x}$$

```
1  diff = InnerProduct(grad(u), grad(v))*dx
2  mass = u*v*dx
3  conv = -InnerProduct(grad(u), 1/tau*(gfset-gfsetold))*v
       *dx
4
5  mesh.SetDeformation(gfset)
6  a.Assemble()
7  m.Assemble()
8  b.Apply(gfu.vec, r)
9  f.Assemble()
10 mesh.UnsetDeformation()
```

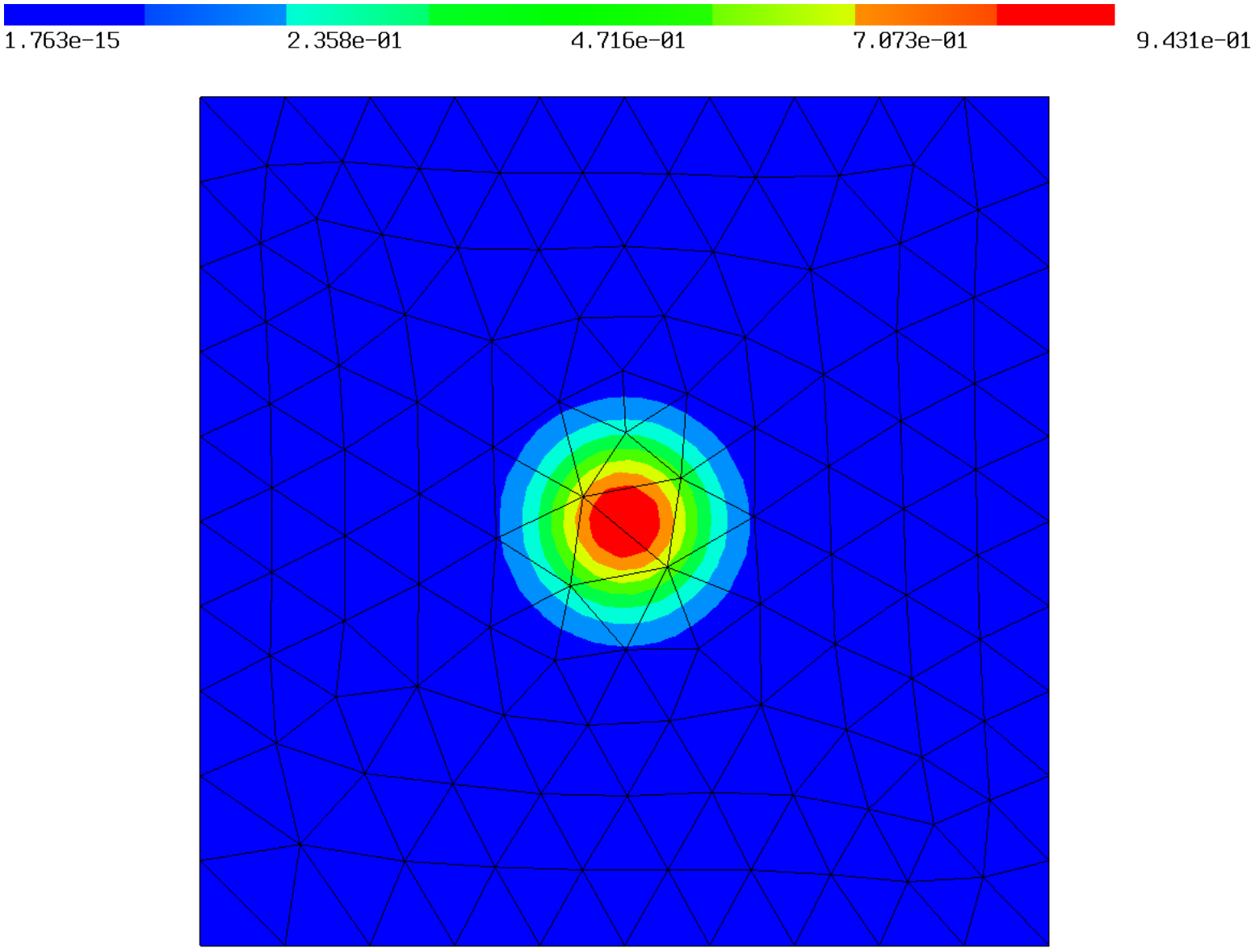
ALE implementation method 2

- SetDeformation function/argument

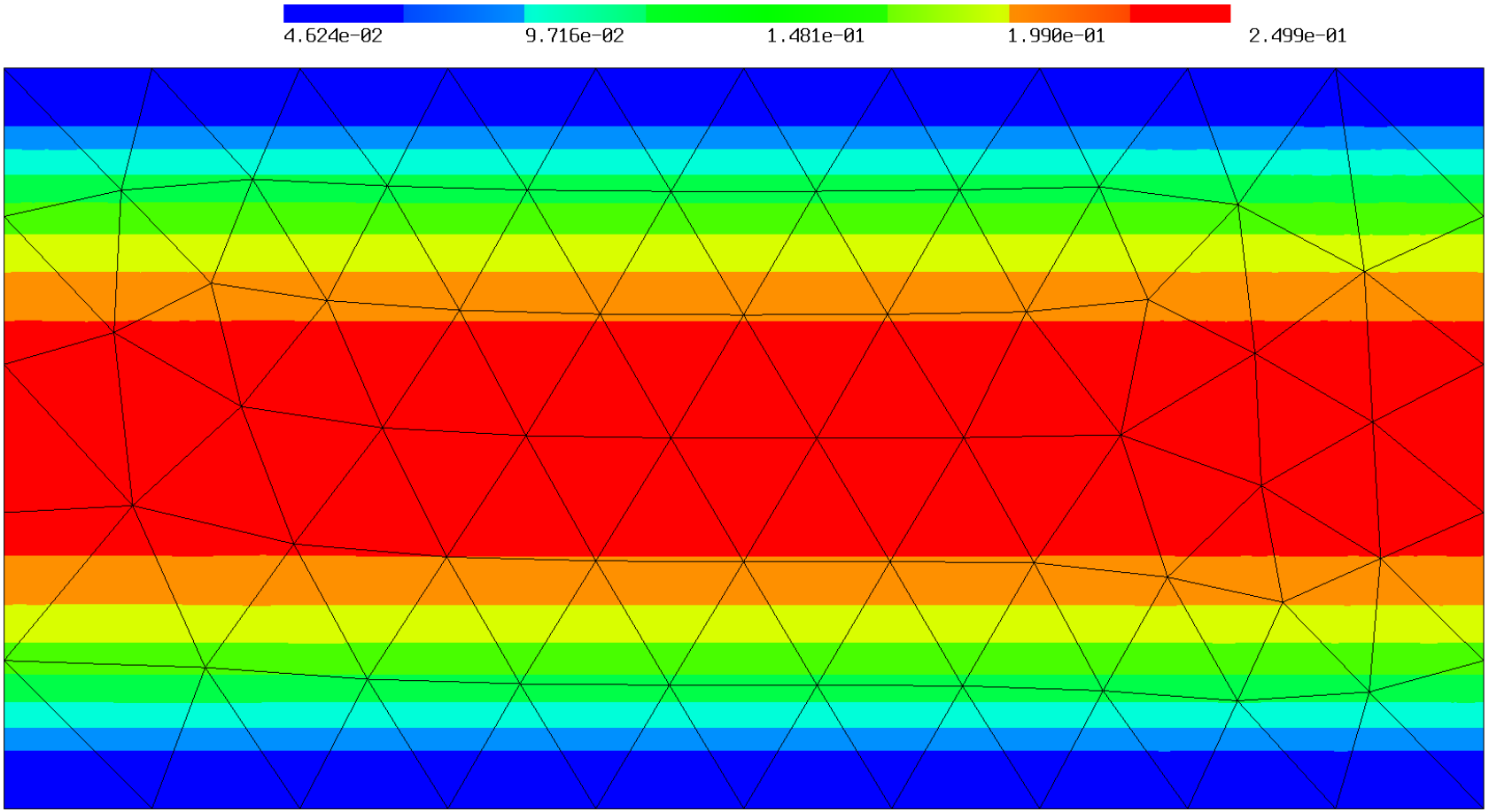
$$\int_{\hat{\Omega}} \left(\frac{\partial \hat{u}}{\partial t}\hat{v} - \nabla_{\hat{x}}\hat{u}\frac{\partial \Phi}{\partial t}\hat{v} + \nabla_{\hat{x}}\hat{u}\nabla_{\hat{x}}\hat{v}\right) d\hat{x} = \int_{\hat{\Omega}} \hat{f}\hat{v}\, d\hat{x}$$

```
1 diff = InnerProduct(grad(u), grad(v))*dx(deformation=
      gfset)
2 mass = u*v*dx(deformation=gfset)
3 conv = -InnerProduct(grad(u), 1/tau*(gfset-gfsetold))*v
      *dx(deformation=gfset)
4
5 a.Assemble()
6 m.Assemble()
7 b.Apply(gfu.vec, r)
8 f.Assemble()
```

Numerical examples

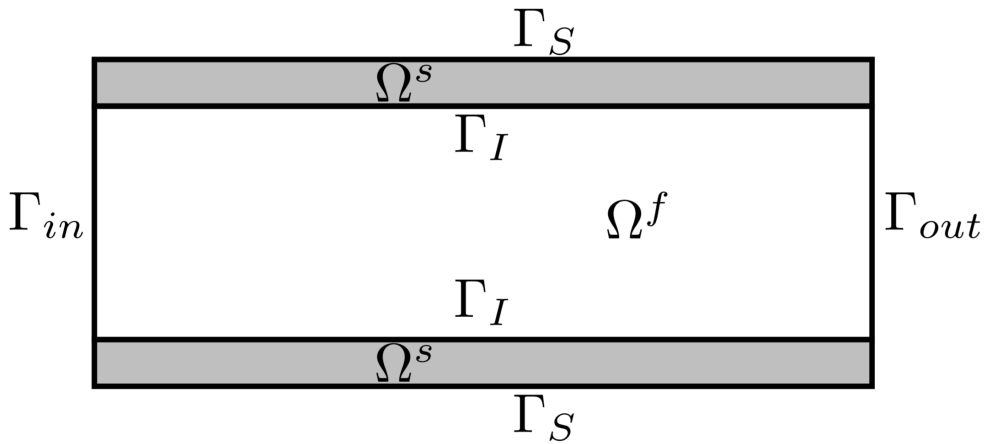


Numerical examples



Deformation extension

Extension problem



- Displacement variable in elastic part of problem
- Extend displacement to fluid domain

Deformation extension, first approach

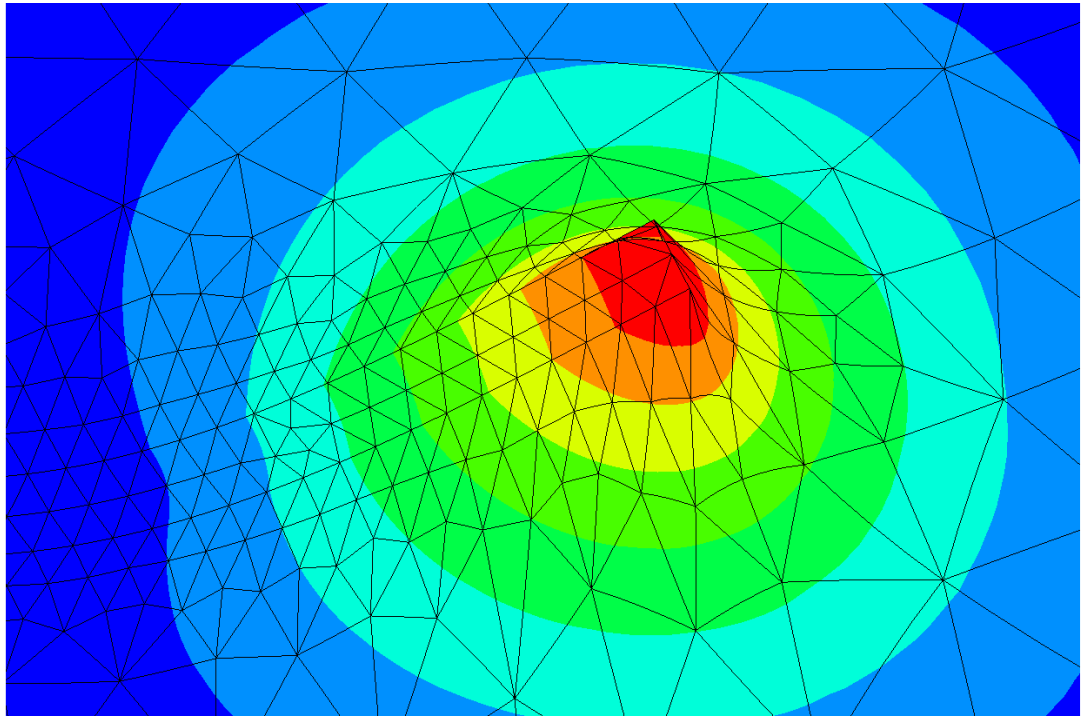
- Solve a simple Poisson problem for a given d^s

$$
\begin{aligned}
-\Delta d^f &= 0, \quad && \text{in } \Omega^f \\
d^f &= 0, && \text{on } \partial\Omega \\
d^f &= d^s, && \text{on } \Gamma_I
\end{aligned}
$$

Deformation extension, first approach

- Solve a simple Poisson problem for a given d^s

$$
\begin{aligned}
-\Delta d^f &= 0, \quad && \text{in } \Omega^f \\
d^f &= 0, && \text{on } \partial\Omega \\
d^f &= d^s, && \text{on } \Gamma_I
\end{aligned}
$$



- Elements get pressed through

Deformation extension, second approach

- Penalize compressed elements by Neo–Hooke material law
- Important for stability

$$\boldsymbol{C} = \boldsymbol{F}^T \boldsymbol{F}, \qquad \boldsymbol{E} = 0.5(\boldsymbol{C} - \boldsymbol{I})$$

$$N(d) = h(\cdot)\mu(\boldsymbol{E} + \frac{\mu}{\lambda} \det(\boldsymbol{C})^{-\frac{\lambda}{2\mu}} - 1)$$

Deformation extension, second approach

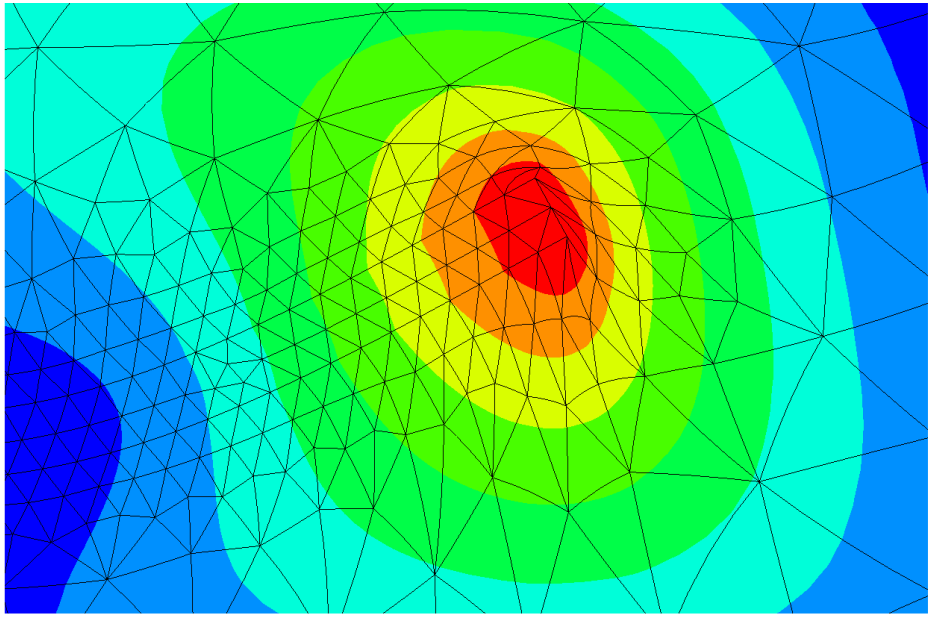
- Penalize compressed elements by Neo–Hooke material law
- Important for stability

$$\boldsymbol{C} = \boldsymbol{F}^T \boldsymbol{F}, \qquad \boldsymbol{E} = 0.5(\boldsymbol{C} - \boldsymbol{I})$$

$$N(d) = h(\cdot)(\boldsymbol{E} + \det(\boldsymbol{C})^{-\frac{1}{2}} - 1)$$

Deformation extension, second approach

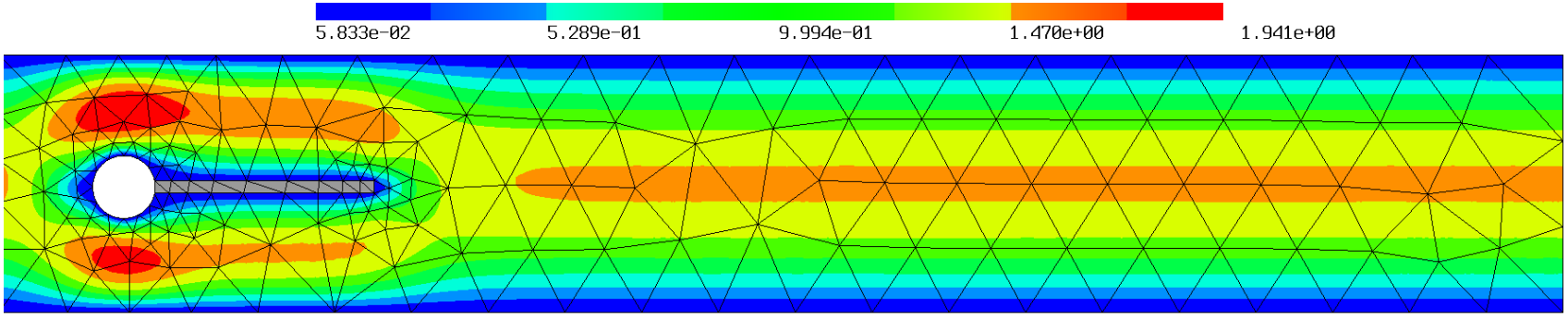
- Penalize compressed elements by Neo–Hooke material law
- Important for stability



$$\boldsymbol{C} = \boldsymbol{F}^T \boldsymbol{F}, \qquad \boldsymbol{E} = 0.5(\boldsymbol{C} - \boldsymbol{I})$$

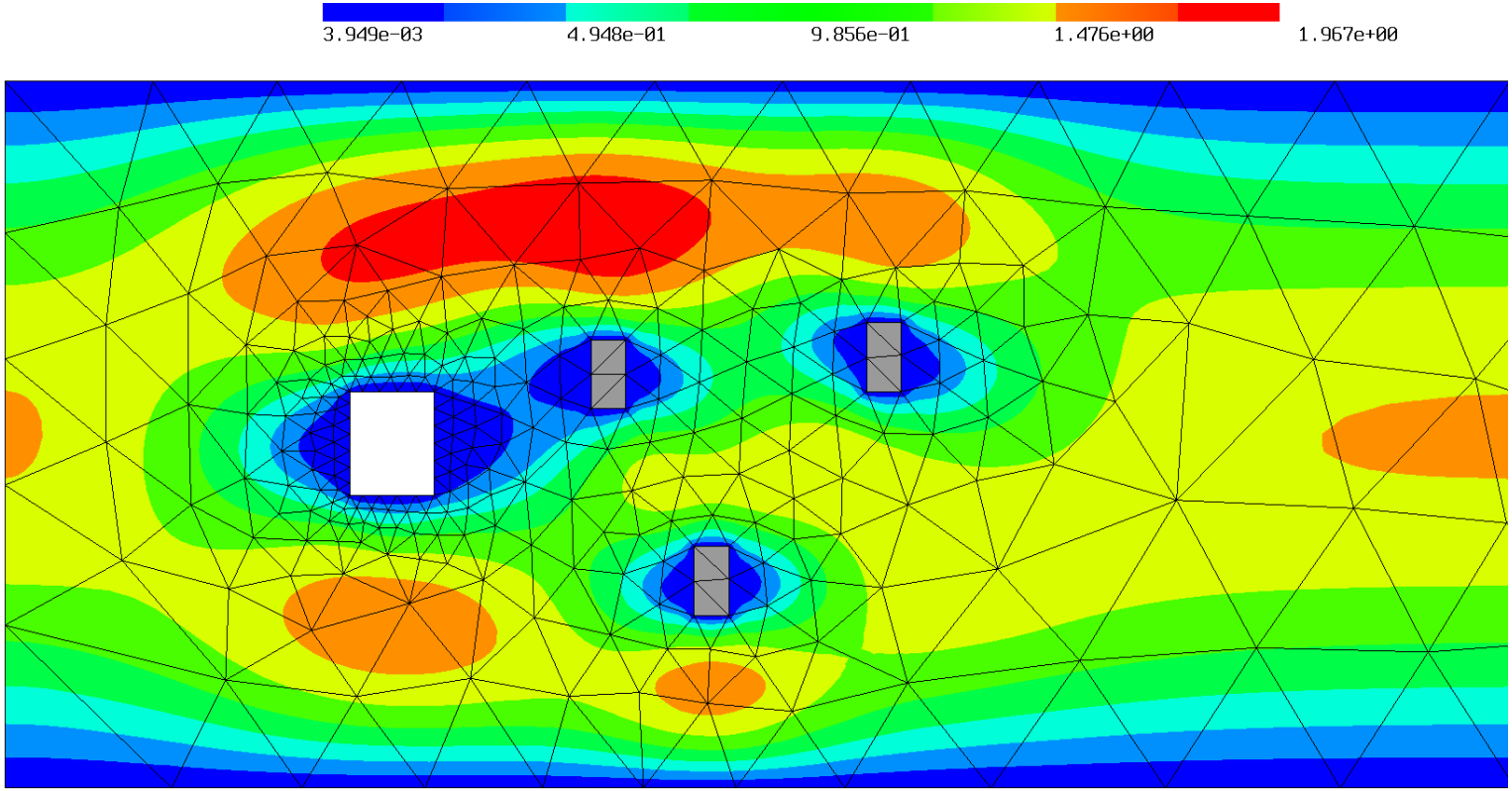
$$N(d) = h(\cdot)(\boldsymbol{E} + \det(\boldsymbol{C})^{-\frac{1}{2}} - 1)$$

Fluid-structure interaction example

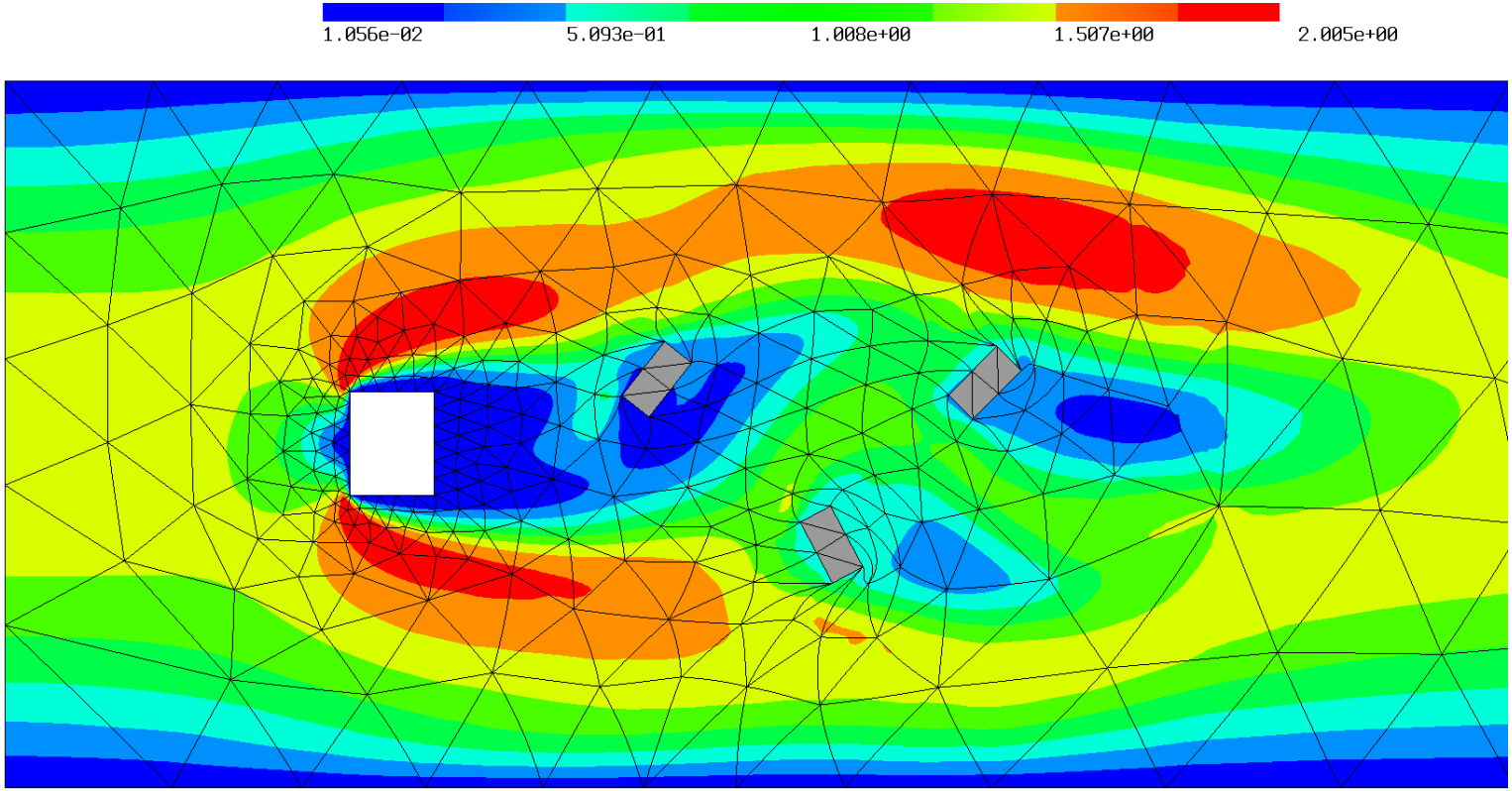


Optimizing mesh

Motivation optimizing mesh

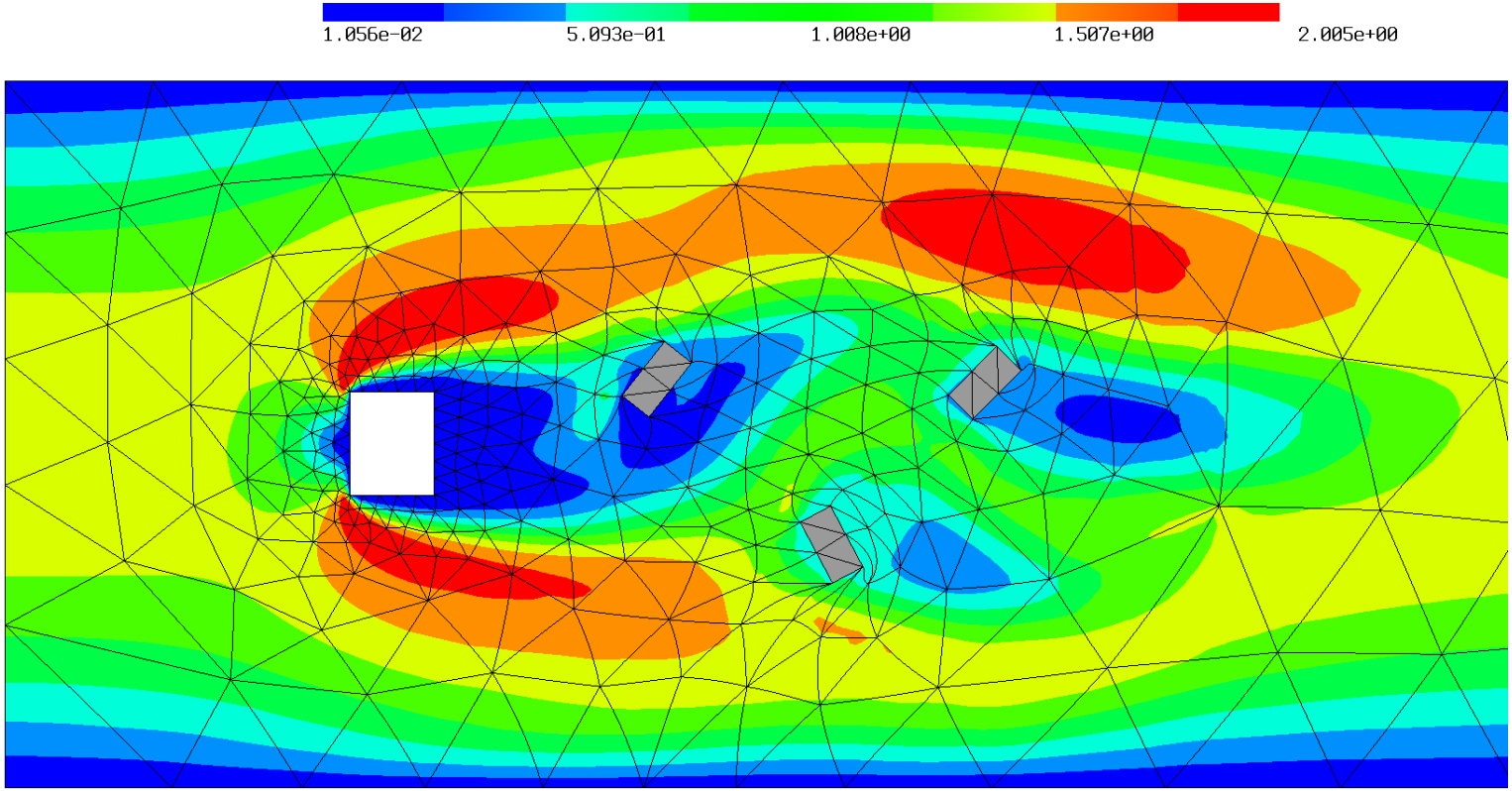


Motivation optimizing mesh



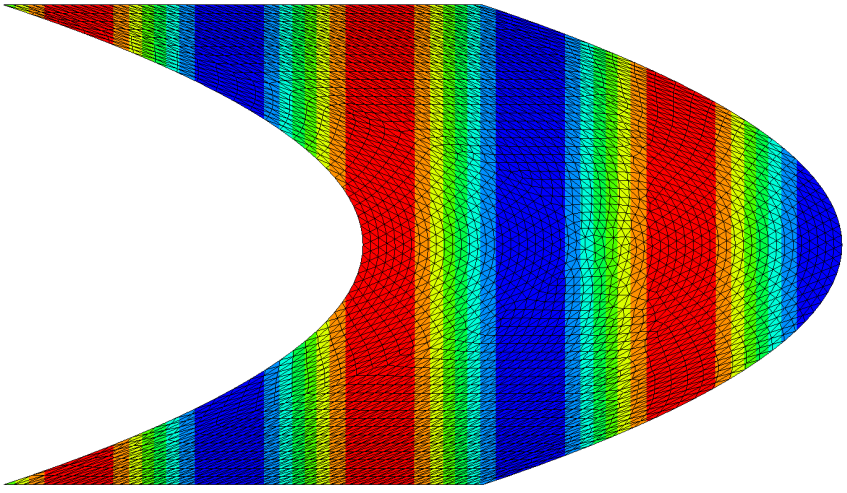
- XFEM (level set function)

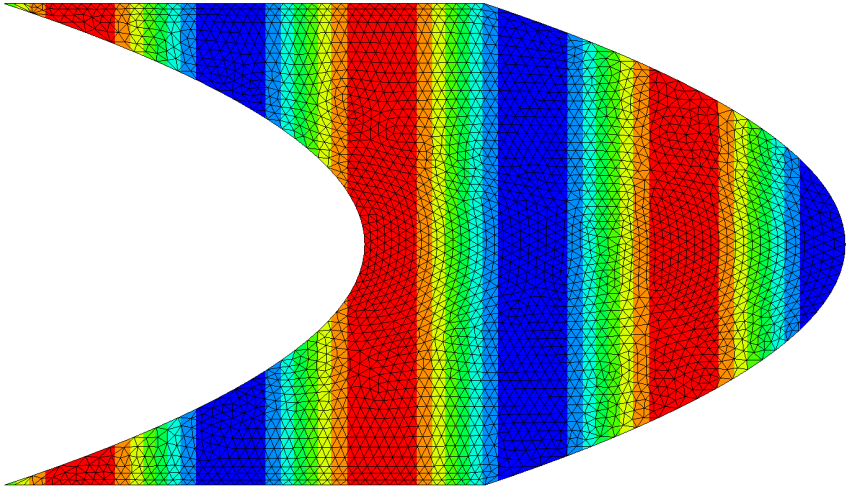
Motivation optimizing mesh



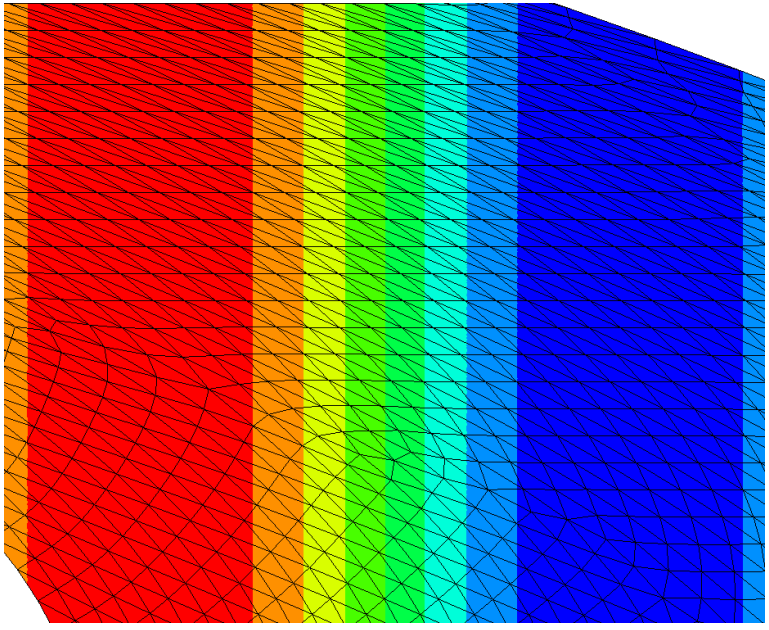
- XFEM (level set function)
- Optimize mesh when quality gets critical

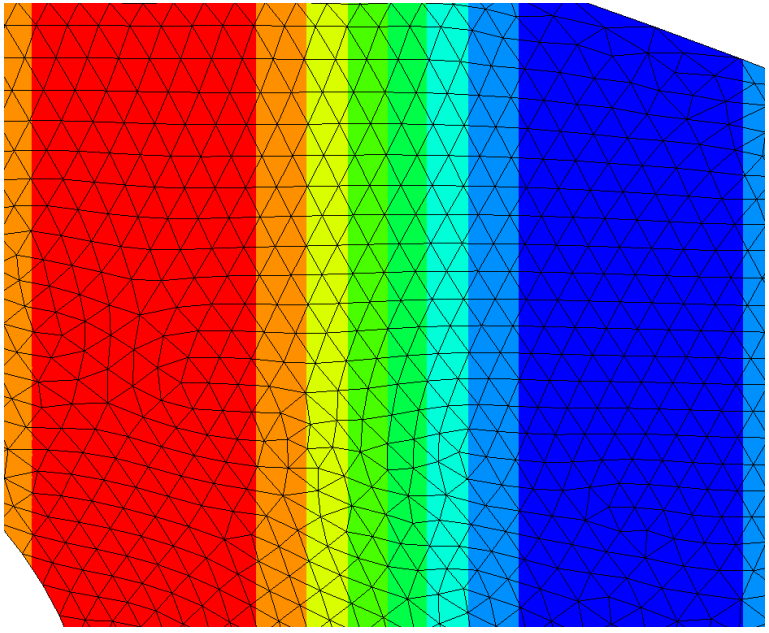
Example



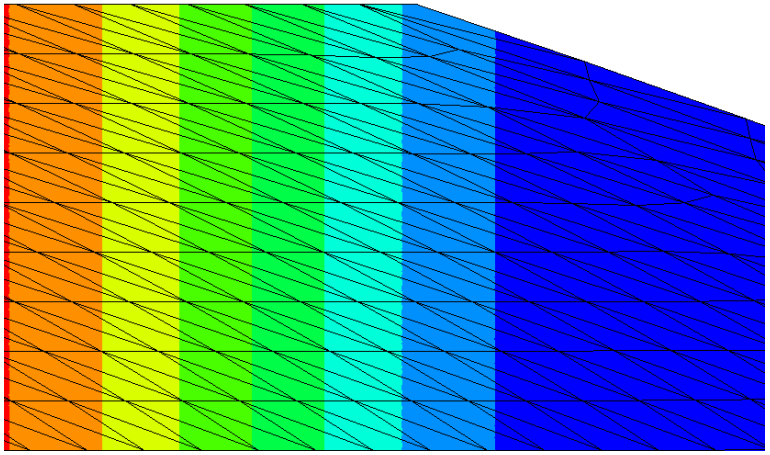


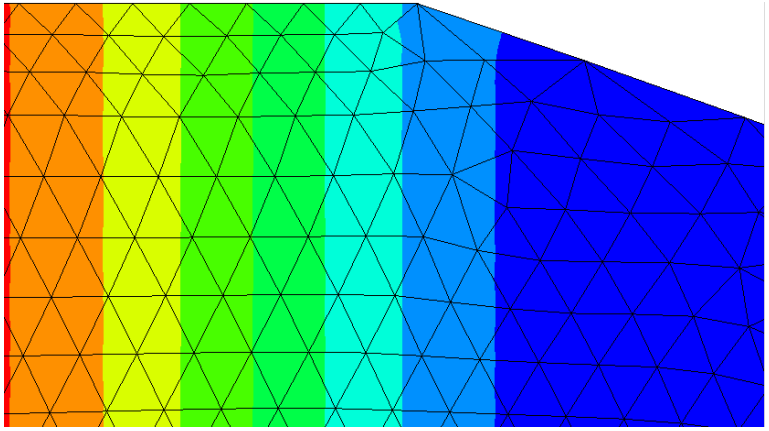
Example



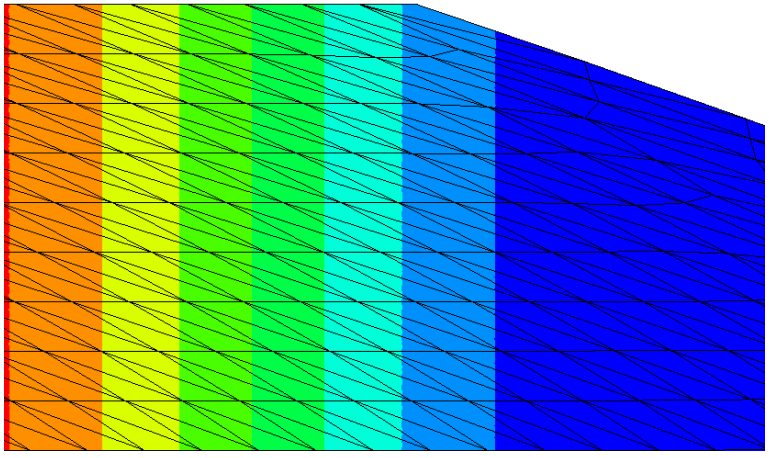


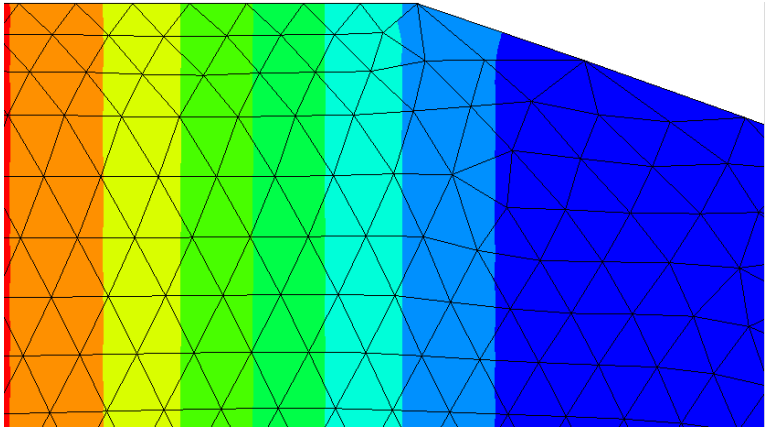
Example





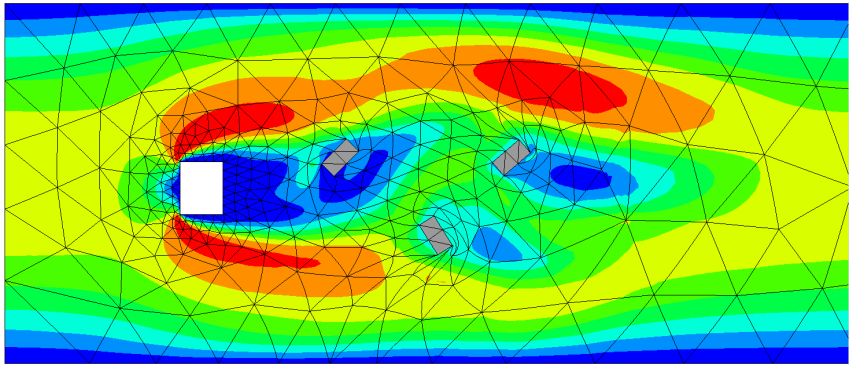
Example



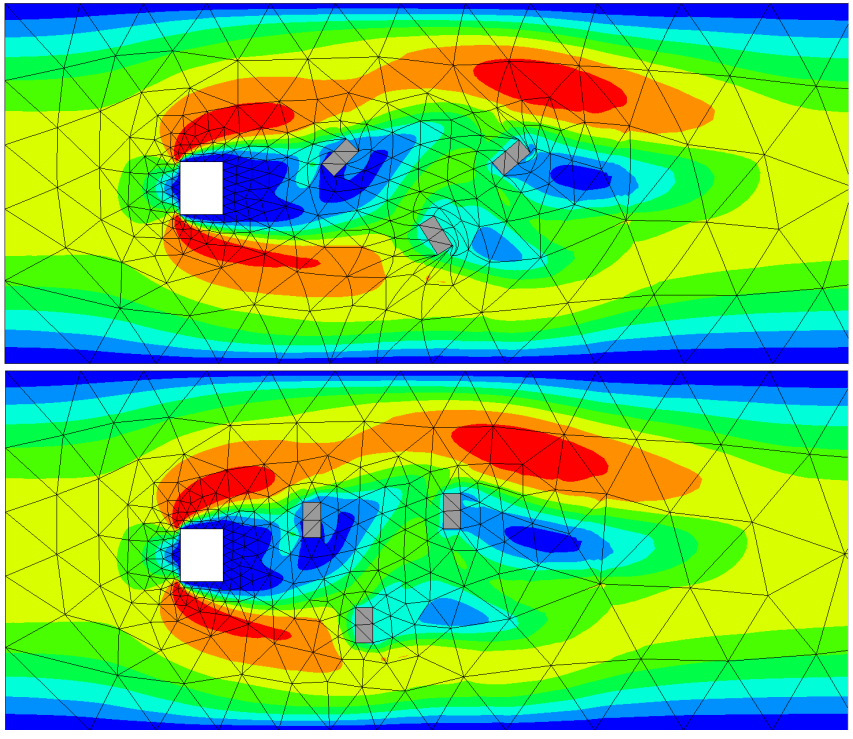


```
1 gfu = GridFunction(H1(mesh, order=3))
2 gfu.Set(sin(10*x))
3
4 mesh2 = Mesh(mesh.ngmesh.Copy())
5 mesh2.ngmesh.OptimizeMesh2d()
6
7 gfu2 = GridFunction(H1(mesh2, order=3))
8 gfu2.Set(gfu)
```

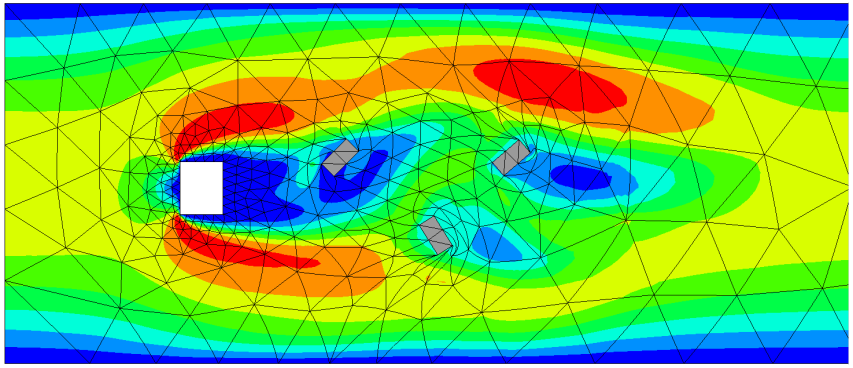
Optimize mesh (ALE)

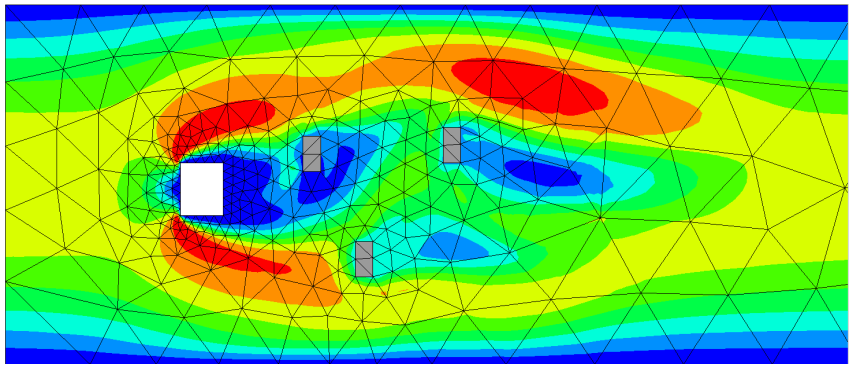


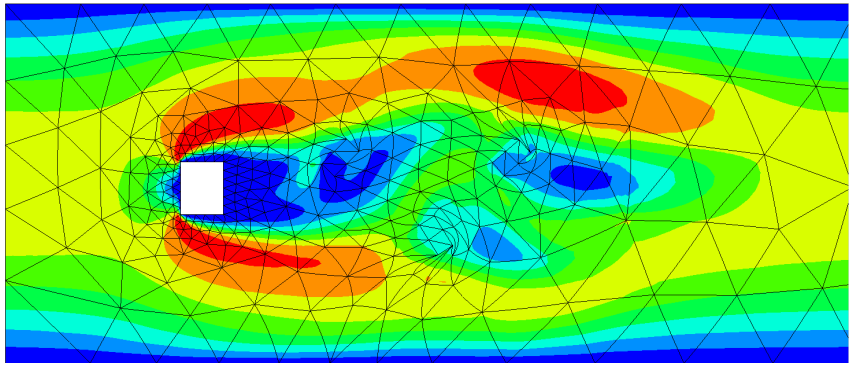
Optimize mesh (ALE)



Optimize mesh (ALE)



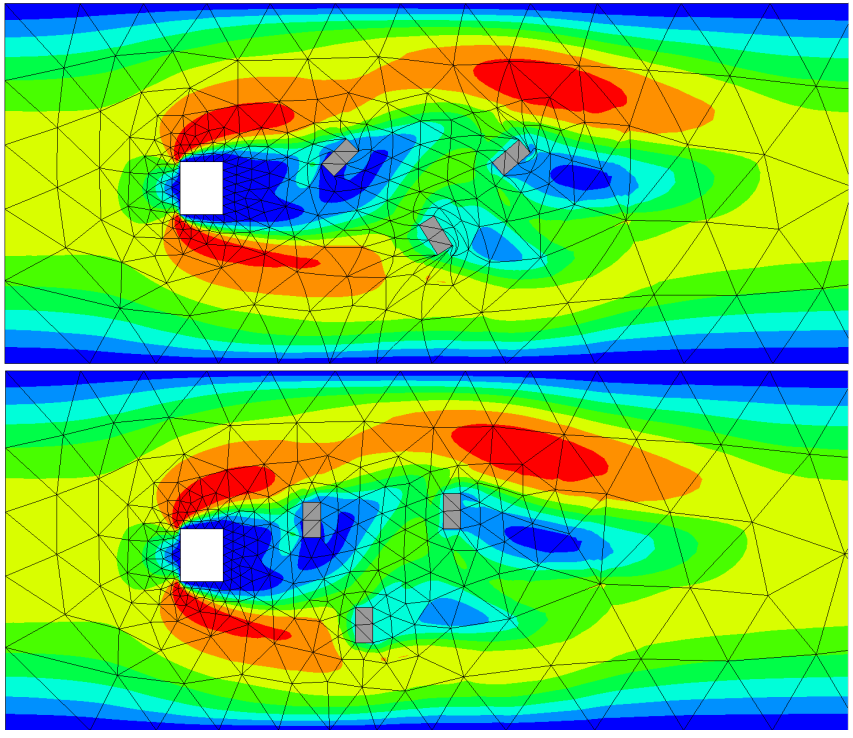


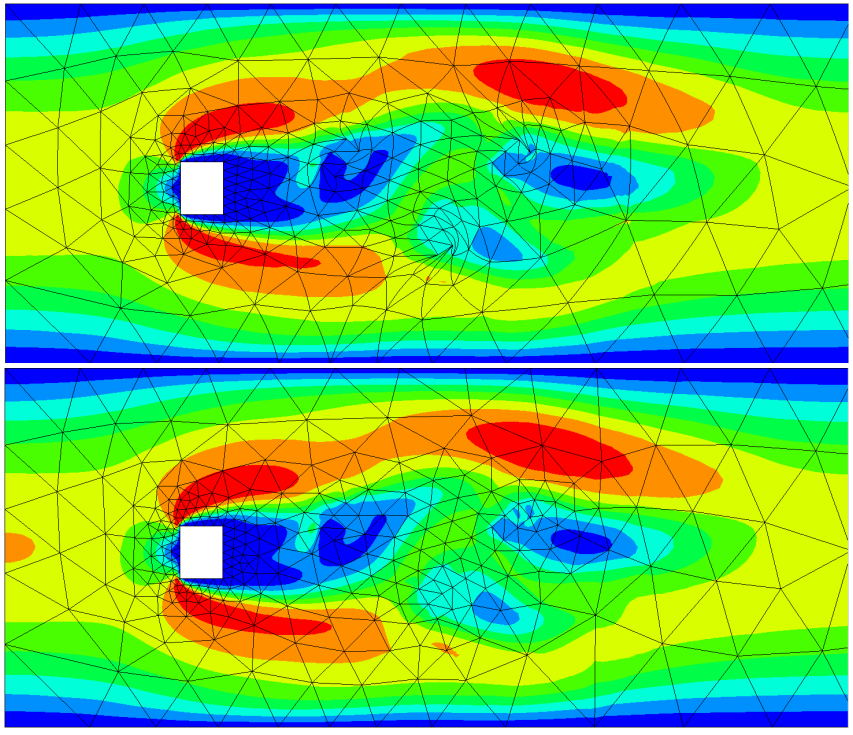


```
tmpmesh = Mesh(mesh.ngmesh.Copy())
for p in tmpmesh.ngmesh.Points():
    p += displacement(p)

gftmp = GridFunction(tmpfes)
gftmp.vec.data = gfu.vec
```

Optimize mesh (ALE)





```
mesh2 = Mesh(tmpmesh.ngmesh.Copy())
ngmesh2.OptimizeMesh2d()

gfu2 = GridFunction(fes2)

gfu2.Set(gftmp)
```

Numerical example