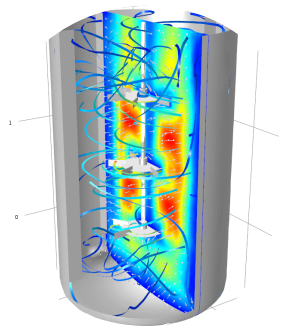
CutFEM 有限元介绍

Luke Chern

March 16, 2025

西安工程大学（机电工程学院）



Overview

1. Part I: 一般有限元
2. Part II: CutFEM 有限元
3. Part III: 线弹性结构有限元
4. Part IV: 杂项
5. Part V: 项目规划

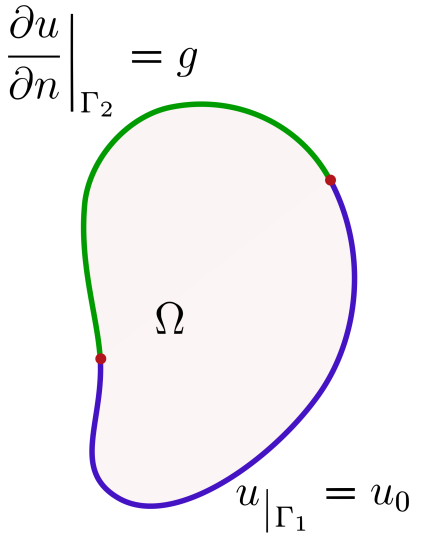
Part I: 一般有限元

一般有限元

Let Ω be a domain in R^d , d $= 2$ or 3, with a piecewise smooth boundary $\partial\Omega$ consisting of two disjoint parts

$$\partial\Omega = \Gamma_D \cup \Gamma_N$$

where Γ_D and Γ_N are the Dirichlet and Neumann parts of the boundary, respectively.



一般有限元

Let us consider first the Poisson equation in Ω with Dirichlet boundary conditions on $\Gamma_D \subset \partial\Omega$ and Neumann boundary conditions on $\Gamma_N \subset \partial\Omega/\Gamma_D$.

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= u_0 \quad \text{on } \Gamma_D \\ \mathbf{n} \cdot \nabla u &= g_N \quad \text{in } \Gamma_N \end{aligned}$$

一般有限元

$$\mathcal{H}^1_D := \left\{u \in \mathcal{H}^1(\Omega) \mid u = u_0 \text{ on } \Gamma_D\right\}$$
$$\mathcal{H}^1_N := \left\{u \in \mathcal{H}^1(\Omega) \mid u = g_N \text{ on } \Gamma_N\right\}$$

The associated weak formulation is the following:
find $u \in \mathcal{H}^1_D$ such that

$$\int_\Omega \nabla u \cdot \nabla v \mathrm{d}\Omega = \int_\Omega v f \mathrm{d}\Omega + \int_{\Gamma_N} v g_N \mathrm{d}s$$

for all $v \in \mathcal{H}^1_N$.

Example 1: 二维泊松方程

考虑一个定义在单位圆区域上的特殊问题，在单位元内部

$$u = \frac{1 - x^2 - y^2}{2}$$

在边界 $\partial\Omega$ 上 $u = 0$。

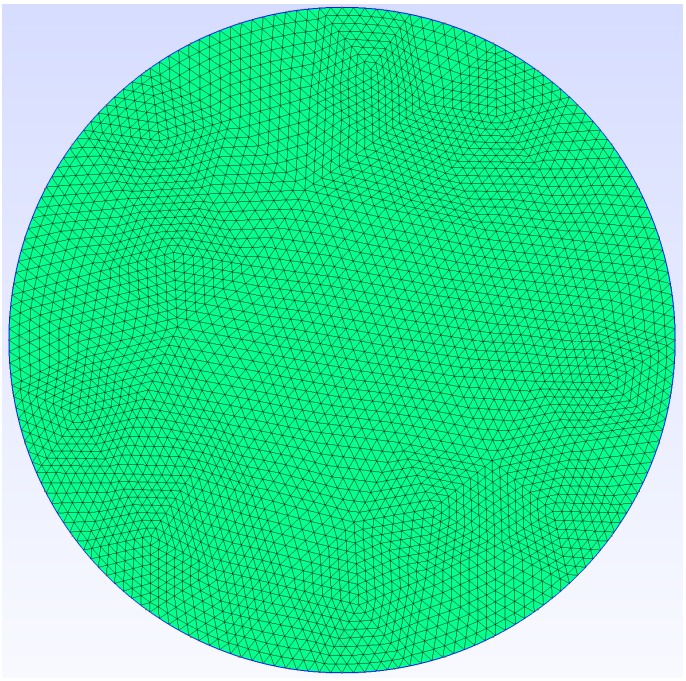


Figure 1: 网格

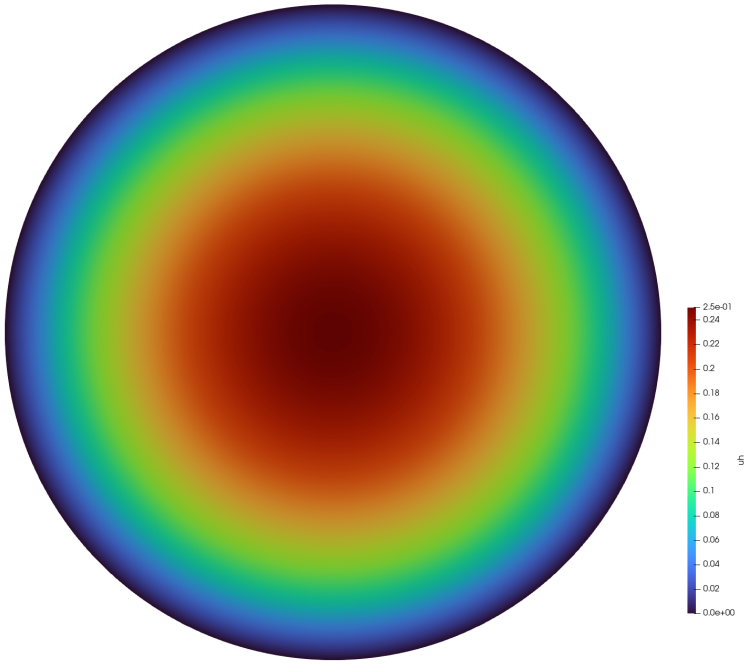


Figure 2: 分析结果

Example 1: 二维泊松方程

```julia
using GridapGmsh
# 读取网格
model = GmshDiscreteModel("./src/circle.msh")

Ω = Triangulation(model)
dΩ = Measure(Ω, 2)

# 有限元空间
reffe = ReferenceFE(lagrangian, Float64, 1)
V = TestFESpace(model, reffe, dirichlet_tags="fixed")

g(x) = 0.0
U = TrialFESpace(V, g)
```


Listing 1: 二维泊松方程

Example 1: 二维泊松方程

```
14 # 定义弱形式
15 f(x) = 1.0
16 a(u,v) = ∫( ∇(v)⋅∇(u) )dΩ
17 l(v) = ∫( v*f )dΩ
18
19 # 求解线性方程组
20 op = AffineFEOperator(a, l, U, V)
21 uh = solve(op)
22
23 # 输出结果
24 writevtk(Ω, "results", cellfields=["uh"=>uh])
```

Listing 2: 二维泊松方程（续）

The Nitsche Method

We introduce a method for treating general boundary conditions in the finite element method generalizing an approach, due to Nitsche, for approximating Dirichlet boundary conditions.
Find $u_h \in V_h$ such that

$$\mathcal{A}_h(u_h, v) = l_h(v).$$

for all $v \in V_h$.

The Nitsche Method

$$\mathcal{A}_h(u_h, v) = \int_\Omega \nabla u \cdot \nabla v \mathrm{d}\Omega + \int_{\Gamma_D} (\lambda uv - v(\mathbf{n} \cdot \nabla u) - u(\mathbf{n} \cdot \nabla v)) \mathrm{d}s$$

$$l_h(v) = \int_\Omega v f \mathrm{d}\Omega + \int_{\Gamma_D} \left(\lambda v g_D - \frac{\partial v}{\partial \mathbf{n}} g_D \right) \mathrm{d}s$$

Example 2: Nitsche 法求解二维泊松方程

```
# 读取网格
Ω = Triangulation(model)
dΩ = Measure(Ω, 2)

Γd = BoundaryTriangulation(model,tags="fixed")
dΓd = Measure(Γd, 2)
n_Γd = get_normal_vector(Γd)

# 有限元空间
reffe = ReferenceFE(lagrangian, Float64, 1)
V = TestFESpace(model, reffe, conformity=:H1)

U = TrialFESpace(V)
```


Listing 3: Nitsche 法求解二维泊松方程

Example 2: Nitsche 法求解二维泊松方程

```
14 γd = 10.0
15 h = 1.0 / 400
16
17 ud(x) = 0.0
18 f(x) = 1.0
19
20 a(u,v) =∫( ∇(v)⋅∇(u) )dΩ +
21 ∫( (γd/h)*v*u  - v*(n_Γd⋅∇(u)) - (n_Γd⋅∇(v))*u )dΓd
22
23 l(v) =  ∫( v*f )dΩ  +
24 ∫( (γd/h)*v*ud - (n_Γd⋅∇(v))*ud ) * dΓd
```

Listing 4: Nitsche 法求解二维泊松方程（续）

Example 2: Nitsche 法求解

分析结果对比：

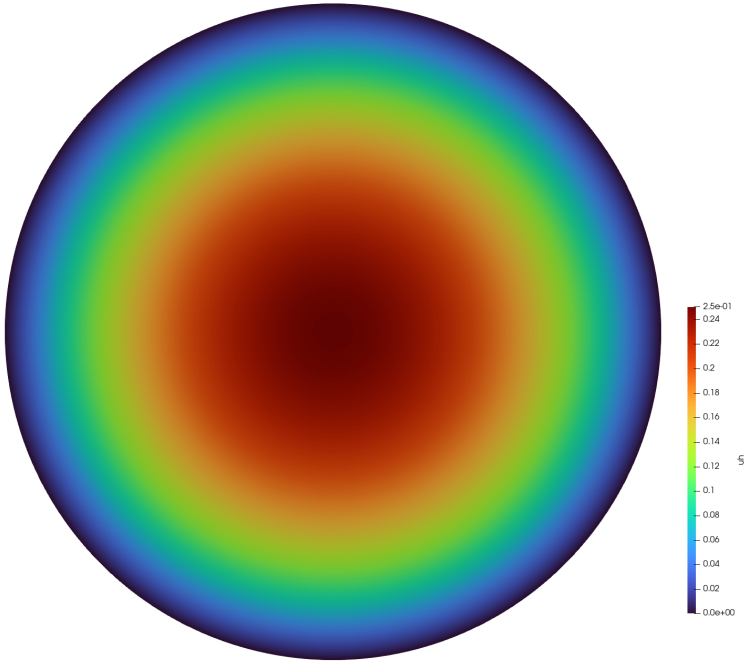


Figure 3: 一般方法

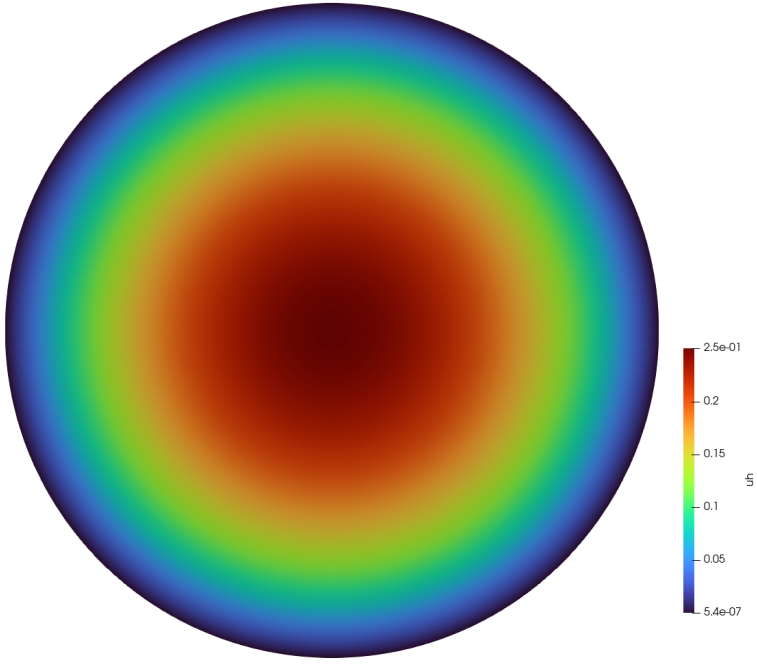


Figure 4: The Nitsche Method

Part II: CutFEM 有限元

CutFEM 有限元

Like in any other *embedded boundary method*, we build the computational mesh by introducing an **artificial domain** Ω_{art} such that it has a simple geometry that is easy to mesh using Cartesian grids and it includes the **physical domain** $\Omega \subset \Omega_{\text{art}}$.

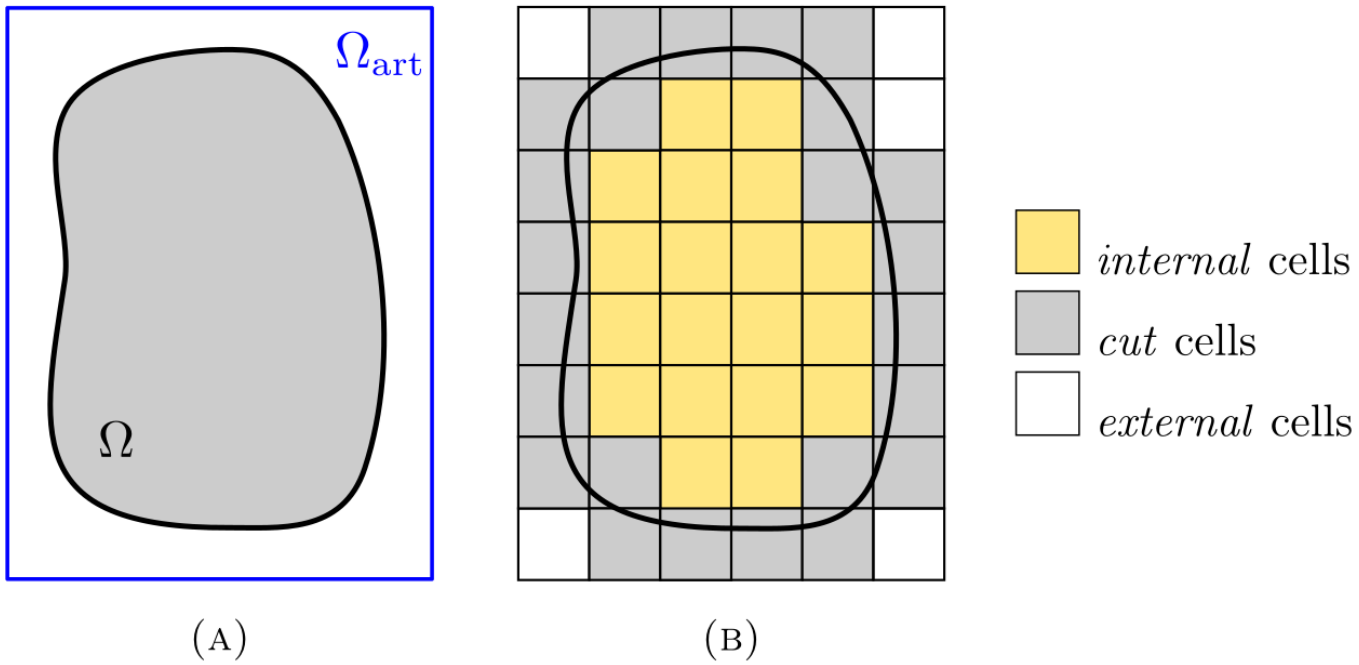
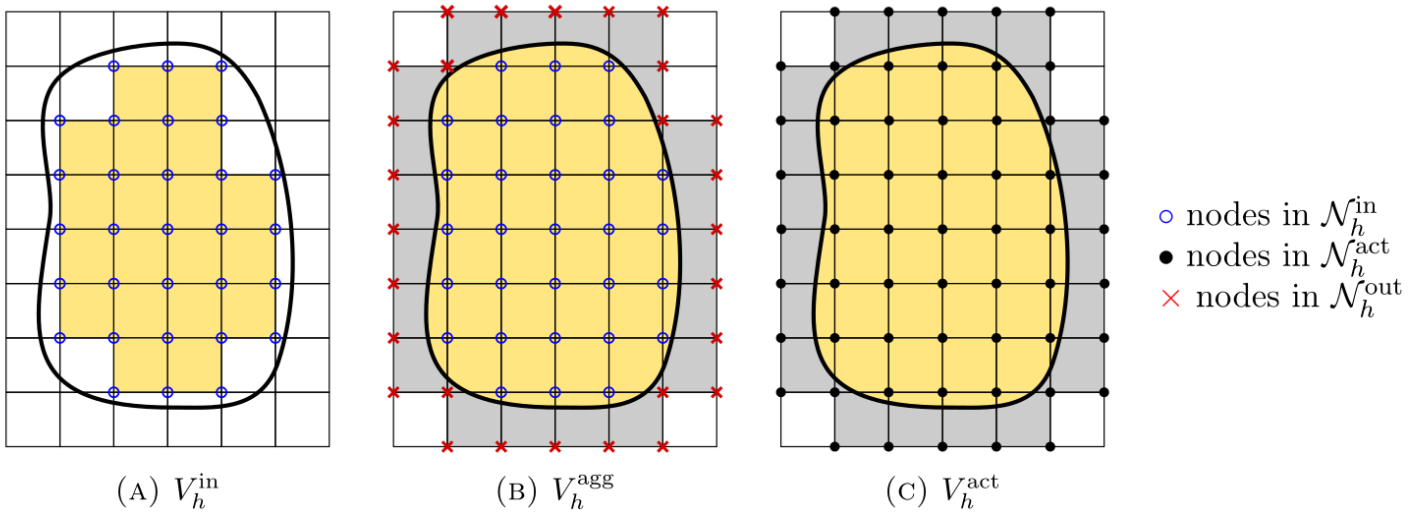


Figure 5: Embedded boundary setup.

CutFEM 有限元

Let us construct a partition of Ω_{art} into cells, represented by T^h_{art} , with characteristic cell size h. Cells in T^h_{art} can be classified as follows: a cell $K \in T^h_{\text{art}}$ such that $K \subset \Omega$ is an **internal cell**; if $K \cap \Omega = \emptyset$, K is an **external cell**; otherwise, K is a **cut cell** (see Fig. 5). Furthermore, we define the set of **active cells** as $T^{\text{act}}_h = T^{\text{in}}_h \cup T^{\text{cut}}_h$ and its union Ω_{act}.



(A) V^{in}_h (B) V^{agg}_h (C) V^{act}_h

Figure 6: Finite Element spaces.

CutFEM 有限元

We define the FE-wise operators:

$$\mathcal{A}_K(u,v) \doteq \int_{K\cap\Omega} \nabla u \cdot \nabla v \mathrm{d}V$$

$$+ \int_{\partial K\cap\Gamma_D} \left(h^{-1}\lambda uv - v(\mathbf{n}\cdot\nabla u) - u(\mathbf{n}\cdot\nabla v)\right) \mathrm{d}S$$

$$+ \int_{\partial K\cap\Gamma_{\text{ghost}}} h^{-1}\mu[\![\partial_n u]\!][\![\partial_n v]\!] \mathrm{d}S$$

$$l_K(u,v) \doteq \int_{K\cap\Omega} vf \mathrm{d}V + \int_{\partial K\cap\Gamma_D} \left(h^{-1}\lambda v g_D - (\mathbf{n}\cdot\nabla v) g_D\right) \mathrm{d}S$$

CutFEM 有限元

For Neumann boundary conditions, the consequence is that the stiffness matrix can become arbitrarily ill-conditioned as the cut-size approaches zero.
For a Dirichlet condition, the situation is even worse. For any finite choice of Nitsche constant, λ_D, the bilinear form $a_h(\cdot, \cdot)$ loses coercivity as the size of a cell cut approaches zero.
This makes the above weak formulation essentially useless because we typically can not control how the cells intersect Γ. One way to avoid this problem is to add a so-called ghost penalty term, s_h, to the weak formulation.

CutFEM 有限元

Given a face $F \in \mathcal{F}_h^{\text{ghost}}$ and the two cells K and K' sharing this face, we define the jump operator

$$[\![\partial_n u]\!] \doteq \mathbf{n}_K \cdot \nabla u|_K + \mathbf{n}_{K'} \cdot \nabla u|_{K'},$$

h_F is some average of h_K and $h_{K'}$.
We define the GP stabilisation term:

$$s_h(u,v) = \sum_{F \in \mathcal{F}_h^{\text{ghost}}} (\lambda_G h_F [\![\partial_n u]\!], [\![\partial_n v]\!])_F \,.$$

Example 3: CutFEM 求解

We assume that Ω is described by a level set function $\psi : \mathbb{R}^{\text{dim}} \to \mathbb{R}$ such that

$$\Omega = \{x \in \mathbb{R}^{\text{dim}} : \psi(x) < 0\},$$
$$\Gamma = \{x \in \mathbb{R}^{\text{dim}} : \psi(x) = 0\}.$$

For simplicity, we choose Ω to be a unit disk, so that $\psi(x) = \|x\| - 1$. As can be seen from the figure below, the level set function is negative for points in Ω, zero on the boundary, and positive everywhere else.

Example 3: CutFEM 求解

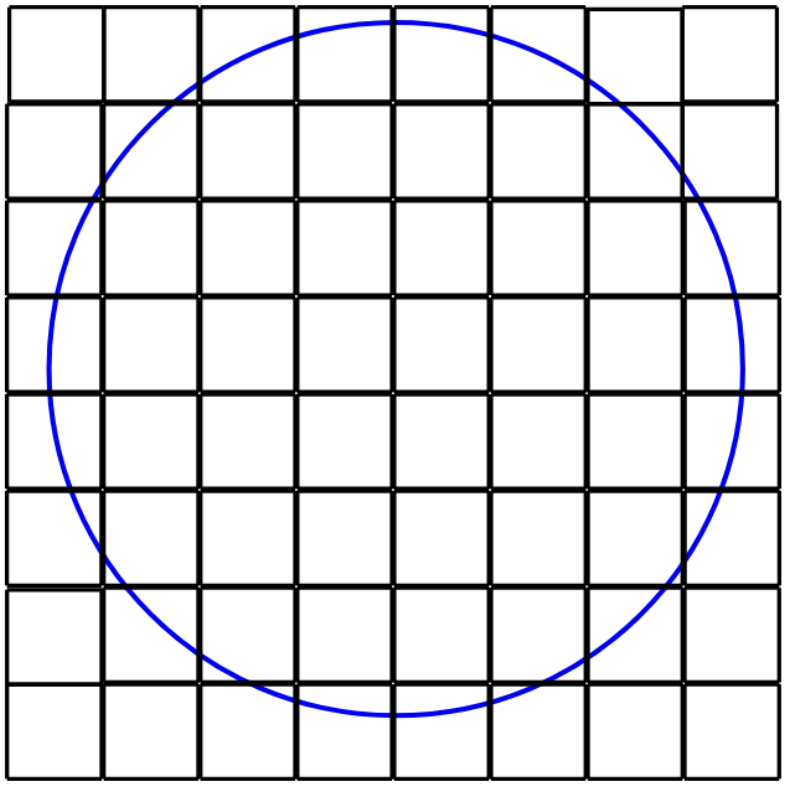


Figure 7: 背景网格 $\mathcal{T}^h$

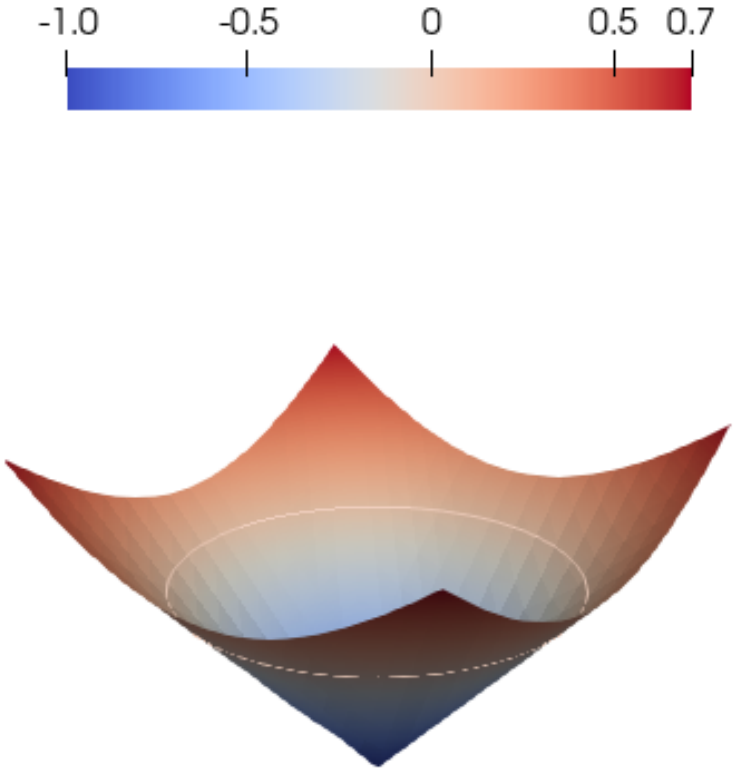


Figure 8: 水平集函数 $\psi(x)$

Example 3: CutFEM 求解

```
# 1. Build background mesh
nn = 40
partition = (nn,nn)
pmin = 1.2*Point(-1,-1)
pmax = 1.2*Point(1,1)
bgmodel = CartesianDiscreteModel(pmin,pmax,partition)

# 2. Build CSG geometry
R = 1.0
geo = disk(R, name="csg")

# 3. Cut the background model
cutgeo = cut(bgmodel,geo)
```

Example 3: CutFEM 求解

To solve this problem, we want to distribute degrees of freedom over the smallest submesh, $\mathcal{T}_\Omega^h$, that completely covers the domain:

$$\mathcal{T}_\Omega^h = \{T \in \mathcal{T}^h : T \cap \Omega \neq \emptyset\}.$$

The finite element space where we want to find our numerical solution, uh, is now

$$V_\Omega^h = \{v \in C(\mathcal{N}_\Omega^h) : v \in Q_p(T),\, T \in \mathcal{T}_\Omega^h\},$$

where $\mathcal{N}_\Omega^h = \bigcup_{T \in \mathcal{T}_\Omega^h} \overline{T}$.

Example 3: CutFEM 求解

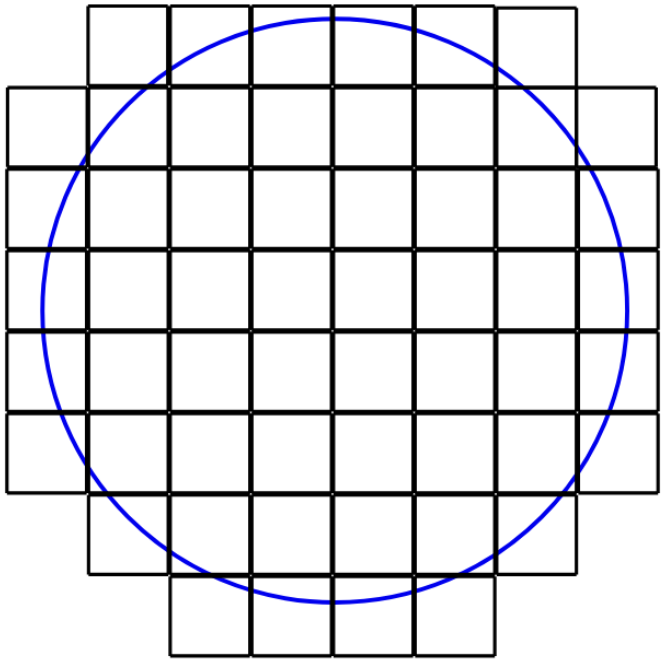
To define the ghost penalty, let $\mathcal{T}_\Gamma^h$ be the set of intersected cells:

$$\mathcal{T}_\Gamma^h = \{T \in \mathcal{T}_\Omega^h : T \cap \Gamma \neq \emptyset\},$$

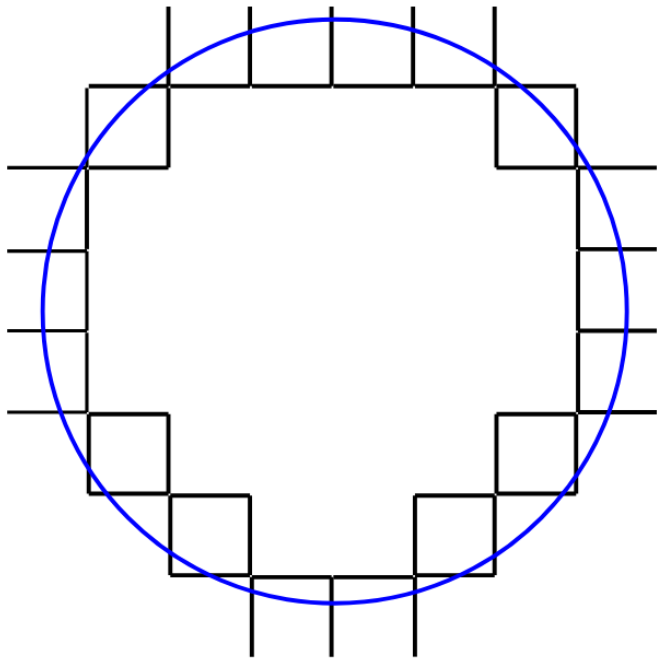
and let $\mathcal{F}_h$ denote the interior faces of the intersected cells in the active mesh:

$$\mathcal{F}_h = \{F = \overline{T}_+ \cap \overline{T}_- : T_+ \in \mathcal{T}_\Gamma^h, T_- \in \mathcal{T}_\Omega^h\}.$$

Example 3: CutFEM 求解



$\mathcal{T}_\Omega^h$



$\mathcal{F}_h$

Example 3: CutFEM 求解

```julia
# 生成计算域
Ω = Triangulation(cutgeo,PHYSICAL,"csg")
Ω_act = Triangulation(cutgeo,ACTIVE,"csg")
```

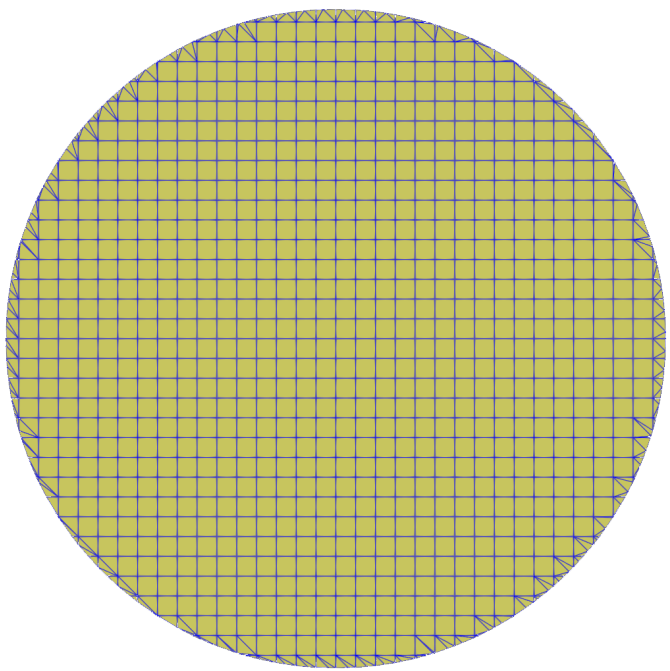


Figure 9: 物理域 Ω

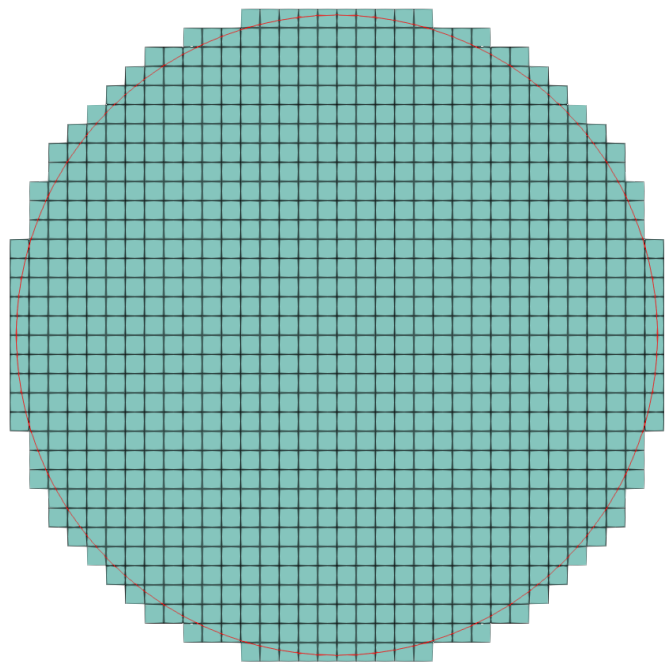


Figure 10: 求解域 Ω_{act}

Example 3: CutFEM 求解

```
1 Γd = EmbeddedBoundary(cutgeo,"csg")
2 Γg = GhostSkeleton(cutgeo,"csg")
```

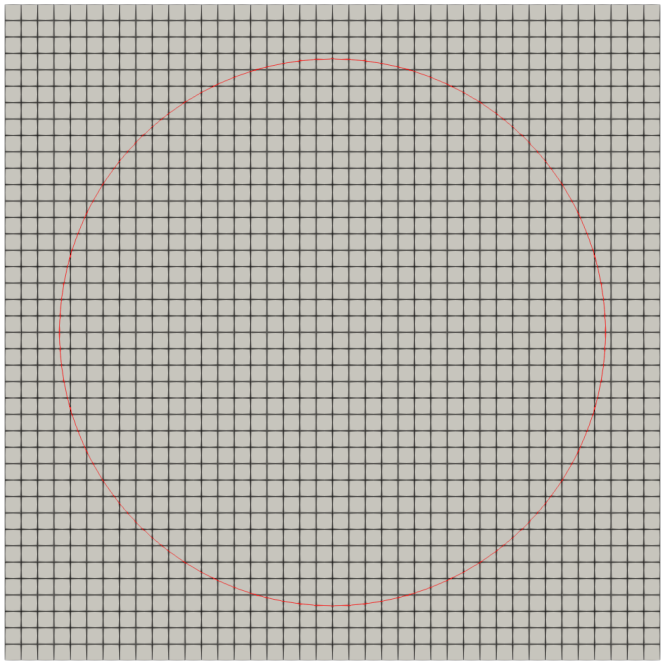


Figure 11: 背景网格 Ω_{bg}

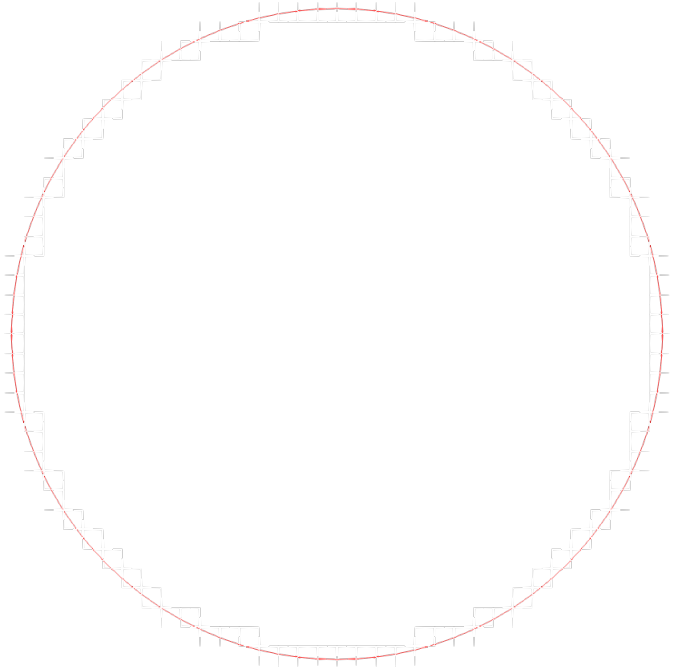


Figure 12: 几何

Example 3: CutFEM 求解

The ghost penalty acts on these faces and reads

$$g_h(u_h, v_h) = \gamma_A \sum_{F \in \mathcal{F}_h} g_F(u_h, v_h),$$

where g_F is the face-wise ghost penalty:

$$g_F(u_h, v_h) = \gamma_A \sum_{k=1}^{p} \left(\frac{h_F^{2k-1}}{k!^2} [\partial_n^k u_h], [\partial_n^k v_h] \right)_F$$

We shall use a continuous space of Q1-elements, so the ghost penalty is reduced to

$$g_h(u_h, v_h) = \gamma_A \sum_{F \in \mathcal{F}_h} (h_F [\partial_n u_h], [\partial_n v_h])_F$$

Example 3: CutFEM 求解

Find $u_h \in V_\Omega^h$ such that

$$a_h(u_h, v_h) = L_h(v_h), \quad \forall v_h \in V_\Omega^h,$$

where

$$a_h(u_h, v_h) = (\nabla u_h, \nabla v_h)_\Omega - (\partial_n u_h, v_h)_\Gamma - (u_h, \partial_n v_h)_\Gamma + \left(\frac{\gamma_D}{h} u_h, v_h\right)_\Gamma$$

$$L_h(v_h) = (f, v)_\Omega + \left(u_D, \frac{\gamma_D}{h} v_h - \partial_n v_h\right)_\Gamma.$$

Example 3: CutFEM 求解

This leads to the stabilized cut finite element method, which reads: Find $u_h \in V_\Omega^h$ such that

$$A_h(u_h, v_h) = L_h(v_h), \quad \forall v_h \in V_\Omega^h,$$

where

$$A_h(u_h, v_h) = a_h(u_h, v_h) + g_h(u_h, v_h).$$

```
# 方程弱形式
a(u,v) = ∫( ∇(v)•∇(u) ) dΩ +
∫( (γd/h)*v*u  - v*(n_Γd•∇(u)) - (n_Γd•∇(v))*u ) dΓd +
∫( (γg*h)*jump(n_Γg•∇(v))*jump(n_Γg•∇(u)) ) dΓg

l(v) = ∫( v*f ) dΩ +
∫( (γd/h)*v*g - (n_Γd•∇(v))*g ) dΓd
```

Example 4: 一个更复杂的例子

```
R = 0.5
geo1 = cylinder(R,v=VectorValue(1,0,0))
geo2 = cylinder(R,v=VectorValue(0,1,0))
geo3 = cylinder(R,v=VectorValue(0,0,1))
geo4 = union(union(geo1,geo2),geo3,name="source")

geo5 = sphere(1)
geo6 = cube(L=1.5)
geo7 = intersect(geo6,geo5)
geo8 = setdiff(geo7,geo4,name="csg")
```

Example 4: 一个更复杂的例子

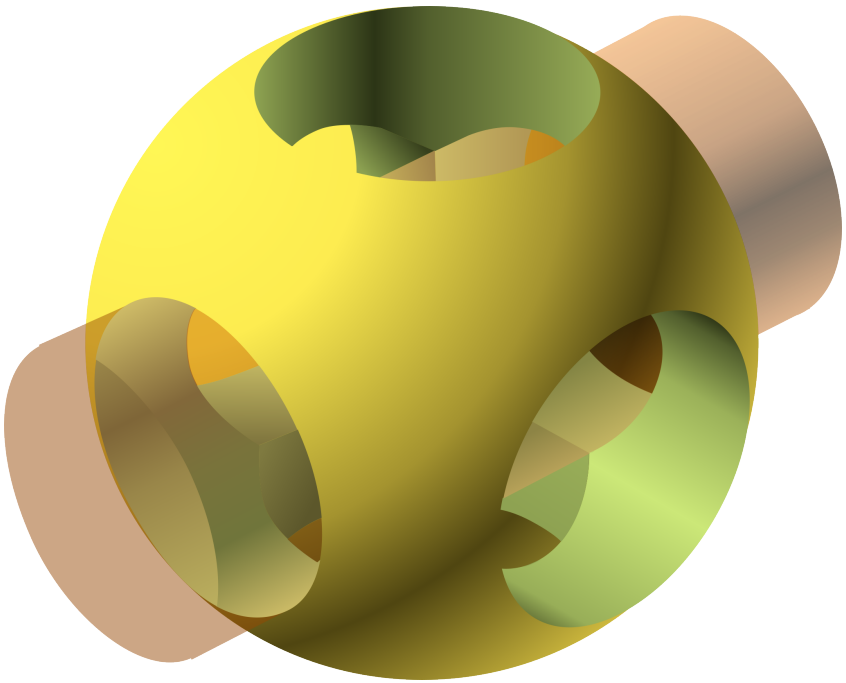


Figure 13: 几何定义

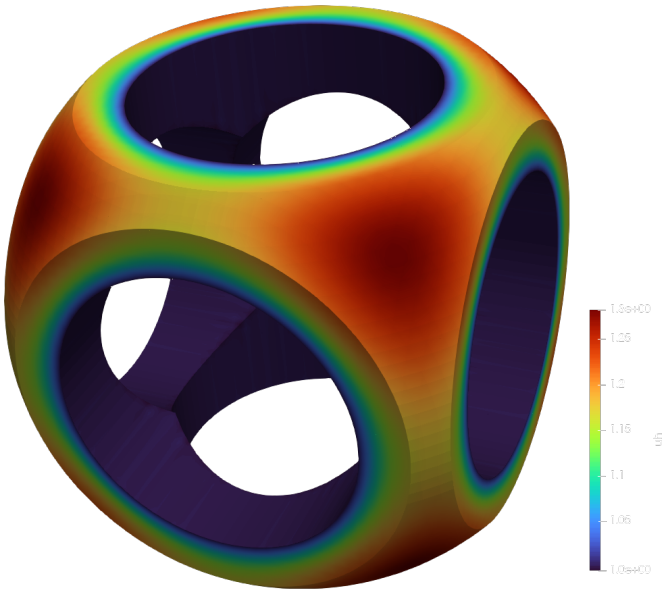


Figure 14: 分析结果

Example 4: 一个更复杂的例子

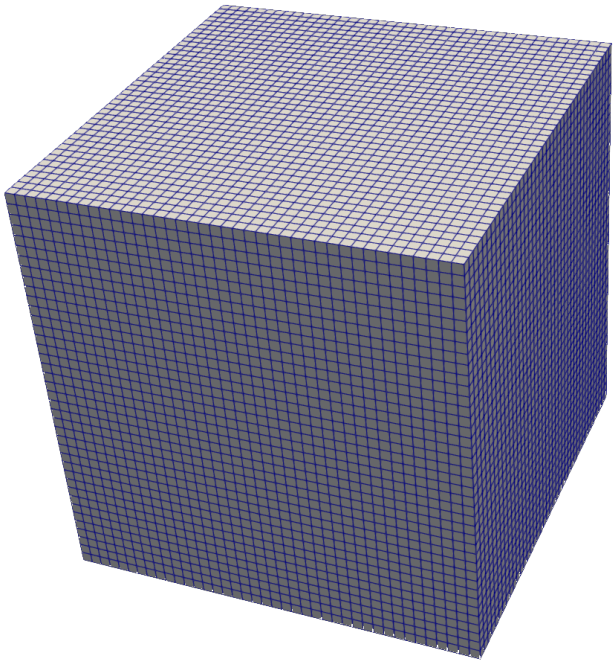


Figure 15: 背景网格

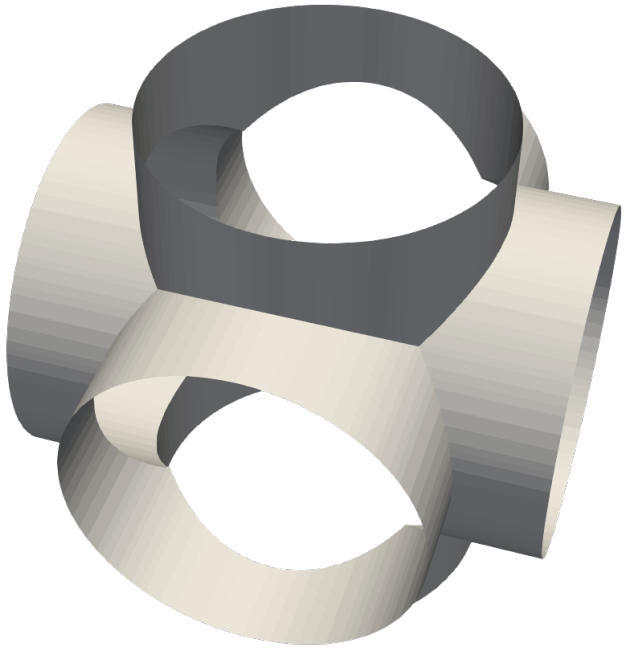


Figure 16: 边界条件

Part III: 线弹性结构有限元

线弹性有限元

Let Ω be a domain in $\mathbb{R}^d$, $d = 2$ or 3, with boundary $\partial\Omega = \Gamma_D \cup \Gamma_N$, $\Gamma_D \cap \Gamma_N = \emptyset$, and exterior unit normal $\mathbf{n}$. We consider the following problems:

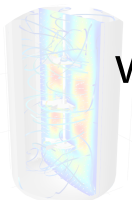
Find the displacement $\mathbf{u} : \Omega \to \mathbb{R}^d$ such that

$$\begin{aligned} -\nabla \cdot \sigma(\mathbf{u}) &= \mathbf{f} && \text{in } \Omega \\ \sigma(\mathbf{u}) \cdot \mathbf{n} &= \mathbf{T} && \text{on } \Gamma_N \\ \mathbf{u} &= \mathbf{g} && \text{on } \Gamma_D \end{aligned}$$

where the strain and stress tensors are defined by

$$\varepsilon(\mathbf{u}) \doteq \frac{1}{2}\left(\nabla \cdot \mathbf{u} + (\nabla \cdot \mathbf{u})^{\mathrm{T}}\right)$$

$$\sigma(\mathbf{u}) \doteq 2\mu\varepsilon(\mathbf{u}) + \lambda \mathrm{tr}(\varepsilon(\mathbf{u}))$$



with Lamé parameters λ and μ.

Let $\mathbf{V}_g = \left\{\mathbf{u} \in \mathbf{H}^1(\Omega) : \mathbf{u} = \mathbf{g} \text{ on } \Gamma_D\right\}$, and define the bilinear form

$$a(\mathbf{u}, \mathbf{v}) = 2\mu \left(\varepsilon(\mathbf{u}), \varepsilon(\mathbf{v})\right)_\Omega + \lambda \left(\text{tr}(\varepsilon(\mathbf{u})), \text{tr}(\varepsilon(\mathbf{v}))\right)_\Omega$$

Find such that

$$a(\mathbf{u}, \mathbf{v}) = l(\mathbf{v}), \quad \forall \mathbf{v} \in \mathbf{V}_0$$

where the linear form on right hand side is defined by

$$l(\mathbf{v}) = (\mathbf{f}, \mathbf{v})_\Omega + (\mathbf{T}, \mathbf{v})_{\Gamma_N}$$

Example 5: 悬臂梁

```julia
using Gridap
using GridapGmsh

# 读取网格文件
model = GmshDiscreteModel("./src/beam.msh")
```

Listing 5: 悬臂梁

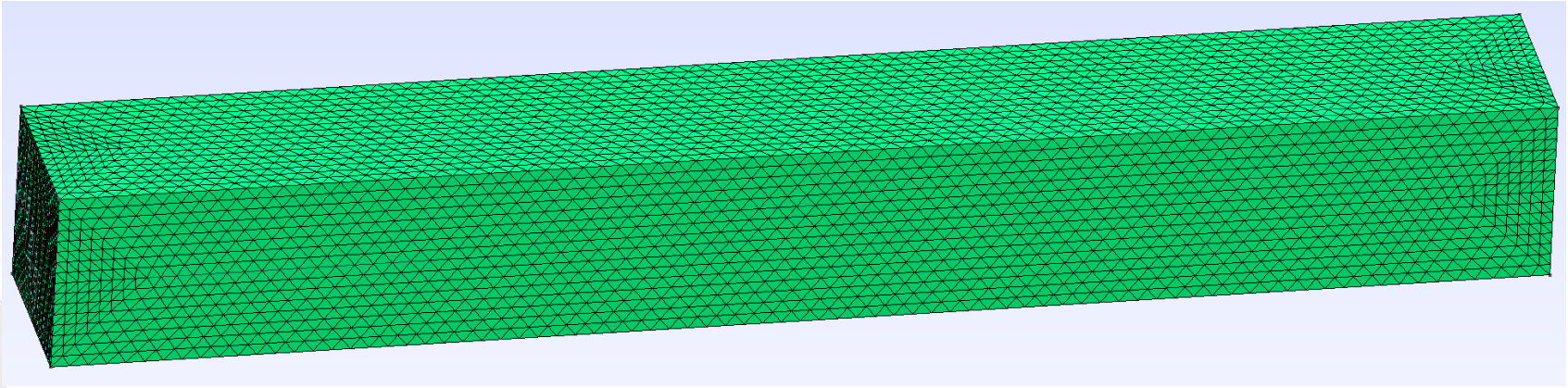


Figure 17: 网格划分

Example 5: 悬臂梁

```
# 参考单元
order = 1
reffe = ReferenceFE(lagrangian,VectorValue{3,Float64},order)

V = TestFESpace(model,reffe; conformity=:H1,
dirichlet_tags=["fixed"],
dirichlet_masks=[(true,true,true)])

g(x) = VectorValue(0.0,0.0,0.0)
U = TrialFESpace(V, g)

degree = 2*order
Ω = Triangulation(model)
dΩ = Measure(Ω,degree)
```


Listing 6: 悬臂梁

Example 5: 悬臂梁

```
20 # 材料参数及本构关系
21 const E = 210.0
22 const ν = 0.3
23 const λ = (E*ν)/((1+ν)*(1-2*ν))
24 const μ = E/(2*(1+ν))
25
26 σ(ε) = λ*tr(ε)*one(ε) + 2*μ*ε
```

Listing 7: 悬臂梁

Example 5: 悬臂梁

```
27 # 弱形式
28 a(u,v) = ∫( ε(v) ⊙ (σ $\circ$ ε(u)) )dΩ
29
30 f(x) = VectorValue(0.0,-9.8,0.0)
31 l(v) = ∫(v ⋅ f)dΩ
32
33 # 求解及输出结果
34 op = AffineFEOperator(a,l,U,V)
35 uh = solve(op)
```

Listing 8: 悬臂梁

Example 5: 悬臂梁

Figure 18: 一般方法

Define the stabilized Nitsche form

$$\mathcal{A}_h(\mathbf{v},\mathbf{w}) = a_h(\mathbf{v},\mathbf{w}) - (\sigma(\mathbf{v})\cdot\mathbf{n},\mathbf{w})_{\Gamma_D} - (\mathbf{v},\sigma(\mathbf{w})\cdot\mathbf{n})_{\Gamma_D} + \beta h^{-1} b_h(\mathbf{v},\mathbf{w})$$

where $\beta > 0$ is a parameter and

$$b_h(\mathbf{v},\mathbf{w}) = 2\mu\,(\mathbf{v},\mathbf{w})_{\Gamma_D} + \lambda\,(\mathbf{v}\cdot\mathbf{n},\mathbf{w}\cdot\mathbf{n})_{\Gamma_D}$$

Find $\mathbf{u}_h \in \mathbf{V}_h$ such that

$$\mathcal{A}_h(\mathbf{u}_h, \mathbf{v}) = \mathcal{L}_h(\mathbf{v}), \quad \forall \mathbf{v}_h \in \mathbf{V}_h$$

where the right hand side is given by

$$\mathcal{L}_h(\mathbf{v}) = (\mathbf{f}, \mathbf{v})_\Omega + (\mathbf{T}, \mathbf{v})_{\Gamma_N} - (\mathbf{g}, \sigma(\mathbf{v}) \cdot \mathbf{n})_{\Gamma_D} + \beta h^{-1} b_h (\mathbf{g}, \mathbf{v})_{\Gamma_D} .$$

Example 6: Nitsche 法求解悬臂梁

```
Ω = Triangulation(model)
dΩ = Measure(Ω, 2)

Γd = BoundaryTriangulation(model, tags="fixed")
dΓd = Measure(Γd, 2)
n_Γd = get_normal_vector(Γd)

# 参考单元
order = 1
reffe = ReferenceFE(lagrangian, VectorValue{3,Float64}, order)

V = TestFESpace(model, reffe; conformity=:H1)
U = TrialFESpace(V)
```


Listing 9: 悬臂梁

Example 6: Nitsche 法求解悬臂梁

```
β = 100
h = 1.0 / 400

b(v, w) = 2 * μ * ∫(v ⋅ w)dΓd + λ * ∫((v ⋅ n_Γd) * (w ⋅ n_Γd))dΓd
# 弱形式
a(u, v) = ∫(ε(v) ⊙ (σ ∘ ε(u)))dΩ - ∫(v ⋅ (σ ∘ ε(u)) ⋅ n_Γd)dΓd -
∫(u ⋅ (σ ∘ ε(v)) ⋅ n_Γd)dΓd + β / h * b(u, v)

f(x) = VectorValue(0.0, -9.8, 0.0)
g(x) = VectorValue(0.0, 0.0, 0.0)
l(v) = ∫(v ⋅ f)dΩ - ∫(g ⋅ (σ ∘ ε(v)) ⋅ n_Γd)dΓd + β / h * b(g, v)

# 求解及输出结果
op = AffineFEOperator(a, l, U, V)
uh = solve(op)
```

Example 6: Nitsche 法求解悬臂梁

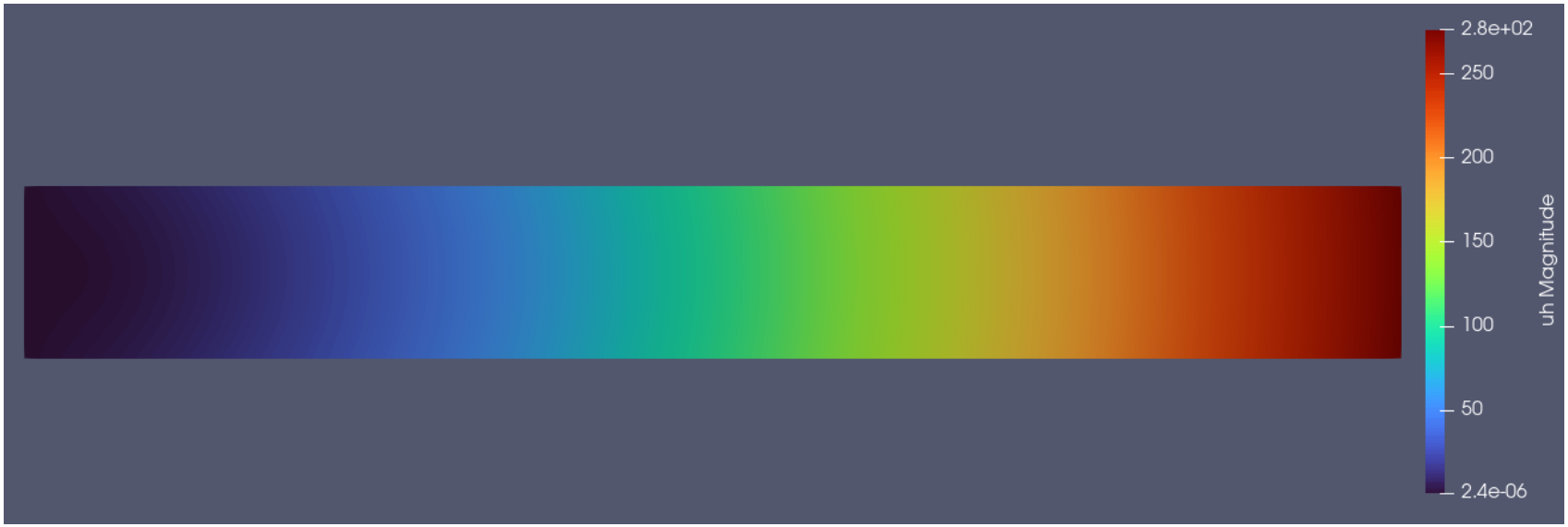


Figure 19: Nitsche 法

Let $\mathcal{F}(\partial\Omega)$ in be the set of interior faces that belongs to an element K such that $K \cap \partial\Omega \neq \emptyset$, and define the stabilization form

$$j_h(v,w) = \sum_{F \in \mathcal{F}_h(\partial\Omega)} \sum_{l=1}^{p} h^{2l+1} \left(\left[D^l_{n_F} v \right], \left[D^l_{n_F} w \right] \right)_F$$

where $D^l_{\mathbf{n}_F}$ is the l:th partial derivative in the direction of the normal $\mathbf{n}_F$ to the face $F \in \mathcal{F}(\partial\Omega)$ and $[\cdot]$ denotes the jump in a discontinuous function at F.

Example 7: 悬臂梁

Listing 10: Gmsh 生成网格

```
SetFactory("OpenCASCADE");
Rectangle(1) = {0, 0, 0, 8, 1, 0};

x = 0.0; y = 0.5; r = 0.3;
For i In {1:7}
    x += 1.0;
    Circle(4+i) = {x, y, 0, r, 0, 2*Pi};
    Curve Loop(1+i) = {4+i};
    Plane Surface(1+i) = {1+i};
EndFor
```

Example 7: 悬臂梁

Listing 11: Gmsh 生成网格（续）

```
11 BooleanDifference(9) = { Surface{1}; Delete; }{ Surface{2}; Delete;
       };
12
13 For i In {1:6}
14     BooleanDifference(9+i) = { Surface{8+i}; Delete; }{ Surface{2+i}
       ; Delete; };
15 EndFor
16
17 Physical Surface("domain", 16) = {15};
18 Physical Curve("fixed", 17) = {2};
19 Physical Curve("force", 18) = {3};
```

Example 7: 悬臂梁

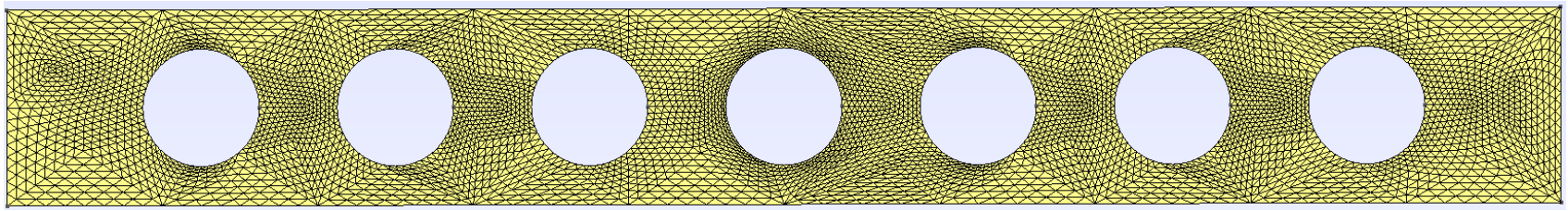


Figure 20: 网格划分

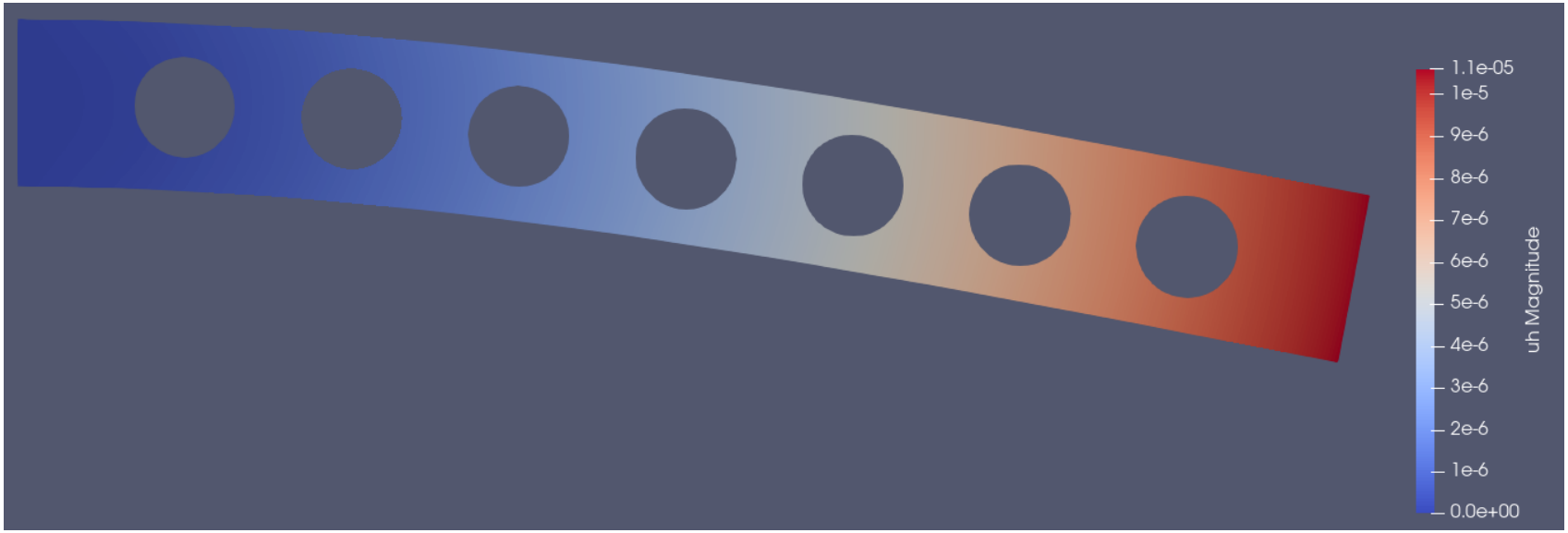


Figure 21: 分析结果

Example 8: CutFEM 求解悬臂梁

```
bgmodel = GmshDiscreteModel("./src/beam2ds.msh")

R = 0.3; x = 0.0; y = 0.5
csg = disk(R,x0=Point(x+1,y))

for dx in range(2,7)
    local geo = disk(R,x0=Point(x+dx,y))

    global csg = union(csg, geo, name="csg")
end

cutgeo = cut(bgmodel, !(csg, name="csg"))

```


Listing 12: 定义几何

Example 8: CutFEM 求解悬臂梁

```
# 1. 物理网格
Ω = Triangulation(cutgeo,PHYSICAL,"csg")
dΩ = Measure(Ω, 2)

# 2. Active网格
Ω_act = Triangulation(cutgeo,ACTIVE,"csg")

# 3. Dirichlet边界条件
Γd = BoundaryTriangulation(bgmodel, tags="fixed")
dΓd = Measure(Γd, 2)

# 4. Neumann边界条件
Γf = BoundaryTriangulation(bgmodel, tags="force")
dΓf = Measure(Γf, 2)
```


Listing 13: 定义网格

Example 8: CutFEM 求解悬臂梁

```
15 # 5. 嵌入边界
16 Γ = EmbeddedBoundary(cutgeo,"csg")
17 dΓ = Measure(Γ, 2)
18
19 # 6. Ghost边界
20 Γg = GhostSkeleton(cutgeo,"csg")
21 dΓg = Measure(Γg, 2)
22 n_Γg = get_normal_vector(Γg)
```

Listing 14: 定义网格

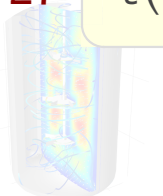
Example 8: CutFEM 求解悬臂梁

```
15 order = 1
16 reffe = ReferenceFE(lagrangian,VectorValue{2,Float64},order)
17 V = TestFESpace(Ω_act, reffe, conformity=:H1, dirichlet_tags=["
   fixed"])
18
19 # Dirichlet values
20 u0 = VectorValue(0,0)
21 U = TrialFESpace(V,[u0])
```

Listing 15: 有限元空间

Example 8: CutFEM 求解悬臂梁

```
15 # 物理参数
16 const E = 210.0; const ν = 0.3
17 const λ = (E*ν)/((1+ν)*(1-2*ν))
18 const μ = E/(2*(1+ν))
19
20 σ(ε) = λ*tr(ε)*one(ε) + 2*μ*ε
21
22 # 定义弱形式
23 γg = 10.0; h = 1.0/400
24 a(u,v) = ∫( ε(v) ⊙ (σ∘ε(u)) )dΩ + ∫( (γg*h)*jump(∇(v)⋅n_Γg)⋅jump(
   ∇(u)⋅n_Γg) ) * dΓg
25
26 T(x) = VectorValue(0.0,-1e3);
27 l(v) = ∫(v ⋅ T)dΓf
```



Listing 16: 定义弱形式

Example 8: CutFEM 求解悬臂梁

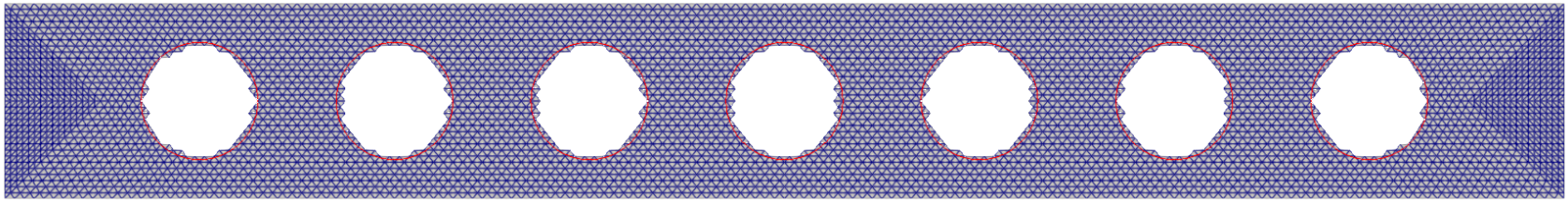


Figure 22: 网格划分

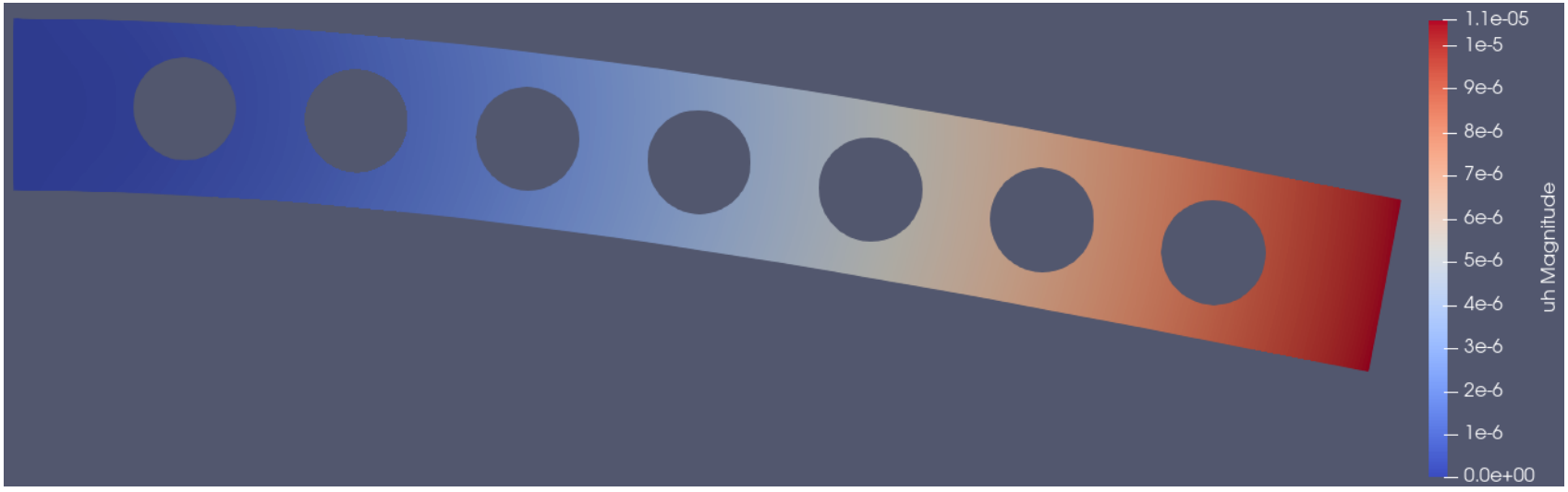


Figure 23: 分析结果

Part IV: 杂项

Interface 问题

We have the following interface conditions on Γ_{12}

$$[\mathbf{u}] = 0 \quad \text{on } \Gamma_{12}$$
$$[\sigma(\mathbf{u}) \cdot \mathbf{n}] = 0 \quad \text{on } \Gamma_{12}$$

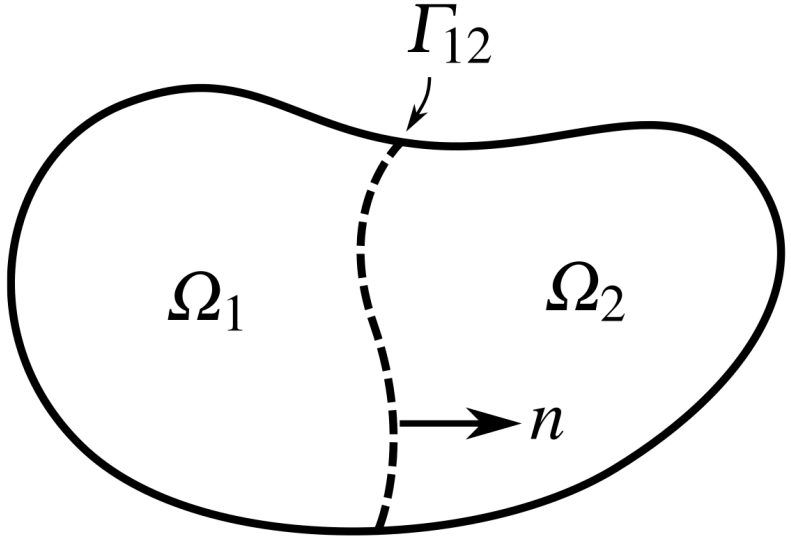
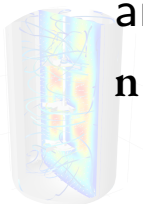


Figure 24: Illustration of compound body consisting of two subdomains Ω_1 and Ω_2 joined at the interface $\Gamma_{12} = \partial\Omega_1 \cap \partial\Omega_2$ with prescribed normal $\mathbf{n} = \mathbf{n}_{\partial\Omega_1}$



Interface 问题

With integration by parts, we have the following interface term

$$I_{12} = -\left(\sigma \cdot \mathrm{n}_{\partial\Omega_1}, \mathbf{v}\right)_{\partial\Omega_1 \cap \Gamma_{12}} - \left(\sigma \cdot \mathrm{n}_{\partial\Omega_2}, \mathbf{v}\right)_{\partial\Omega_2 \cap \Gamma_{12}}$$

From this term and the interface conditions, we derive Nitsche's method for the interface condition via the calculation

$$\begin{aligned}
I_{12} &= -\left(\sigma(\mathbf{u}) \cdot \mathbf{n}, \mathbf{v}\right)_{\partial\Omega_1 \cap \Gamma_{12}} + \left(\sigma(\mathbf{u}) \cdot \mathbf{n}, \mathbf{v}\right)_{\partial\Omega_2 \cap \Gamma_{12}} \\
&= -\left(\sigma(\mathbf{u}) \cdot \mathbf{n} - [\sigma(\mathbf{u}) \cdot \mathbf{n}]/2, \mathbf{v}\right)_{\partial\Omega_1 \cap \Gamma_{12}} + \left(\sigma(\mathbf{u}) \cdot \mathbf{n} - [\sigma(\mathbf{u}) \cdot \mathbf{n}]/2, \mathbf{v}\right)_{\partial\Omega} \\
&= -\left(\sigma(\mathbf{u}) \cdot \mathbf{n} - [\sigma(\mathbf{u}) \cdot \mathbf{n}]/2, [\mathbf{v}]\right)_{\Gamma_{12}} \\
&= -\left(\langle \sigma(\mathbf{u}) \cdot \mathbf{n} \rangle, [\mathbf{v}]\right)_{\Gamma_{12}} - \left([\mathbf{u}], \langle \sigma(\mathbf{v}) \cdot \mathbf{n} \rangle\right)_{\Gamma_{12}} + \gamma_D h^{-1} \left([\mathbf{u}], [\mathbf{v}]\right)_{\Gamma_{12}}
\end{aligned}$$

Example 8: CutFEM 求接口问题

```
# 1. Build background model
L = VectorValue(3,2,3)
partition = (L[1]*n,L[2]*n,L[3]*n)
pmin = Point(0.,0.,0.)
pmax = pmin + L
bgmodel = CartesianDiscreteModel(pmin,pmax,partition)
Ω_bg = Triangulation(bgmodel)

# Identify Dirichlet boundaries
labeling = get_face_labeling(bgmodel)
add_tag_from_tags!(labeling,"support0",[1,3,5,7,13,15,17,19,25])
add_tag_from_tags!(labeling,"support1",[2,4,6,8,14,16,18,20,26])
```

Listing 17: 定义背景网格

Example 8: CutFEM 求接口问题

```
13 # 2. Define geometry
14 R = 0.4
15 δ = 0.3 + R
16 pmid = 0.5*(pmax+pmin)
17
18 geo1 = cylinder(R,x0=Point(0.0,pmid[2],pmin[3]+δ))
19 geo2 = cylinder(R,x0=Point(0.0,pmid[2],pmax[3]-δ))
20
21 geo3 = union(geo1,geo2,name="steel")
22 geo4 = !(geo3,name="concrete")
23
24 # Cut the background model
25 cutgeo = cut(bgmodel,union(geo3,geo4))
```


Listing 18: 定义几何

Example 8: CutFEM 求接口问题

```
26 # 3. Setup interpolation mesh
27 Ω1_act = Triangulation(cutgeo,ACTIVE,"steel")
28 Ω2_act = Triangulation(cutgeo,ACTIVE,"concrete")
29
30 # Setup integration meshes
31 Ω1 = Triangulation(cutgeo,PHYSICAL,"steel")
32 Ω2 = Triangulation(cutgeo,PHYSICAL,"concrete")
33
34 Γ = EmbeddedBoundary(cutgeo,"steel","concrete")
35 # Setup normal vectors
36 n_Γ = get_normal_vector(Γ)
```

Listing 19: 积分区域

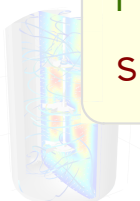
Example 8: CutFEM 求接口问题

```
37 # 4. Define material
38 # Material parameters 1
39 E1 = 4.0
40 ν1 = 0.2
41 λ1, μ1 = lame_parameters(E1,ν1)
42 σ1(ε) = λ1*tr(ε)*one(ε) + 2*μ1*ε
43
44 # Material parameters 2
45 E2 = 1.0
46 ν2 = 0.2
47 λ2, μ2 = lame_parameters(E2,ν2)
48 σ2(ε) = λ2*tr(ε)*one(ε) + 2*μ2*ε
```

Listing 20: 定义材料参数

Example 8: CutFEM 求接口问题

```
49 # 5. Setup Lebesgue measures
50 order = 1
51 degree = 2*order
52 dΩ1 = Measure(Ω1,degree)
53 dΩ2 = Measure(Ω2,degree)
54 dΓ = Measure(Γ,degree)
55
56 # 6. Setup FESpace
57 V1 = TestFESpace(Ω1_act,
58 ReferenceFE(lagrangian,VectorValue{3,Float64},order), conformity=
   :H1, dirichlet_tags=["support0","support1"])
59
60 V2 = TestFESpace(Ω2_act, ReferenceFE(lagrangian,VectorValue{3,
   Float64},order), conformity=:H1, dirichlet_tags=["support0","
   support1"])
```



Listing 21: 定义有限元空间

Example 8: CutFEM 求接口问题

```
61 # Dirichlet values
62 u0 = VectorValue(0,0,0)
63 u1 = VectorValue(0.0,0.0,-0.001)
64
65 U1 = TrialFESpace(V1,[u0,u1])
66 U2 = TrialFESpace(V2,[u0,u1])
67
68 V = MultiFieldFESpace([V1,V2])
69 U = MultiFieldFESpace([U1,U2])
```

Listing 22: 定义有限元空间（续）

Example 8: CutFEM 求接口问题

```
70 # Setup stabilization parameters
71 meas_K1 = get_cell_measure(Ω1, Ω_bg)
72 meas_K2 = get_cell_measure(Ω2, Ω_bg)
73 meas_KΓ = get_cell_measure(Γ, Ω_bg)
74
75 γ_hat = 2
76 κ1 = CellField( (E2*meas_K1) ./ (E2*meas_K1 .+ E1*meas_K2), Ω_bg)
77 κ2 = CellField( (E1*meas_K2) ./ (E2*meas_K1 .+ E1*meas_K2), Ω_bg)
78 β  = CellField( (γ_hat*meas_KΓ) ./ ( meas_K1/E1 .+ meas_K2/E2 ),
   Ω_bg)
79
80 # Jump and mean operators for this formulation
81 jump_u(u1,u2) = u1 - u2
82 mean_t(u1,u2) = κ1*(σ1∘ε(u1)) + κ2*(σ2∘ε(u2))
```


Listing 23: 定义 Jump 和 Mean 函数

Example 8: CutFEM 求接口问题

界面 Γ_{12} 处的惩罚项：

$$T_{12} = -\left(\langle\sigma(\mathbf{u})\cdot\mathbf{n}\rangle, [\mathbf{v}]\right)_{\Gamma_{12}} - \left([\mathbf{u}], \langle\sigma(\mathbf{v})\cdot\mathbf{n}\rangle\right)_{\Gamma_{12}} + \gamma_D h^{-1}\left([\mathbf{u}], [\mathbf{v}]\right)_{\Gamma_{12}}$$

```
83  # 6. Weak form
84  a((u1,u2),(v1,v2)) =
85  ∫( ε(v1) ⊙ (σ1∘ε(u1)) ) * dΩ1 + ∫( ε(v2) ⊙ (σ2∘ε(u2)) ) * dΩ2 + ∫
    ( β*jump_u(v1,v2)⋅jump_u(u1,u2)-n_Γ⋅mean_t(u1,u2)⋅jump_u(v1,v2) -
     n_Γ⋅mean_t(v1,v2)⋅jump_u(u1,u2) ) * dΓ
86
87  l((v1,v2)) = 0
```

Listing 24: 定义弱形式

Example 8: CutFEM 求接口问题

```
88  # 7. FE problem
89  op = AffineFEOperator(a,l,U,V)
90  uh1, uh2 = solve(op)
91  uh = (uh1,uh2)
92
93  # 8. Postprocess
94  if outputfile !== nothing
95      writevtk(Ω1,"$(outputfile)_steel",cellfields=["uh"=>uh1,"
    sigma"=>σ1∘ε(uh1)])
96      writevtk(Ω2,"$(outputfile)_concrete",cellfields=["uh"=>uh2,"
    sigma"=>σ2∘ε(uh2)])
97  end
```

Listing 25: 求解及输出分析结果

Example 8: CutFEM 求接口问题

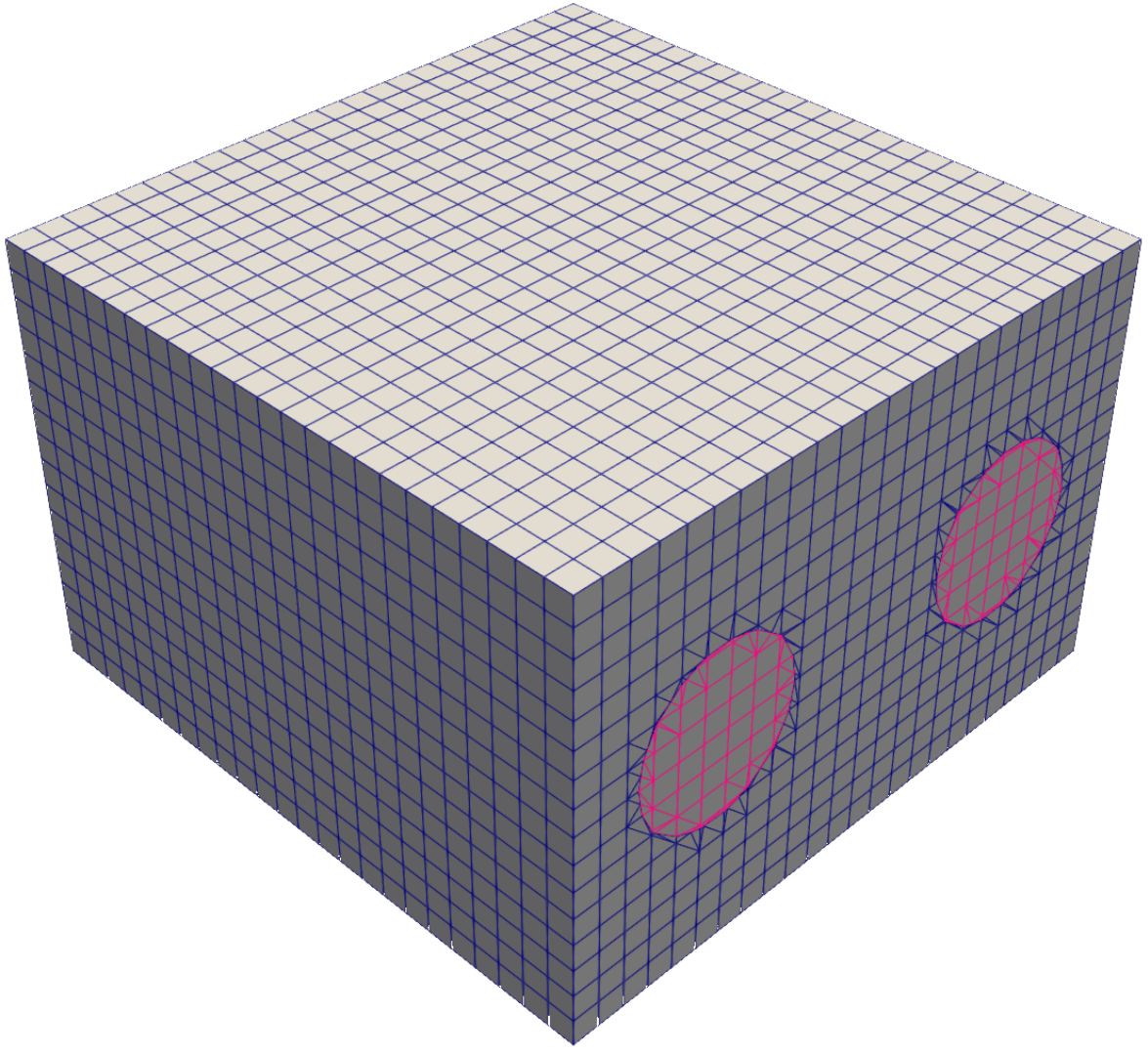


Figure 25: 网格划分

Example 8: CutFEM 求接口问题

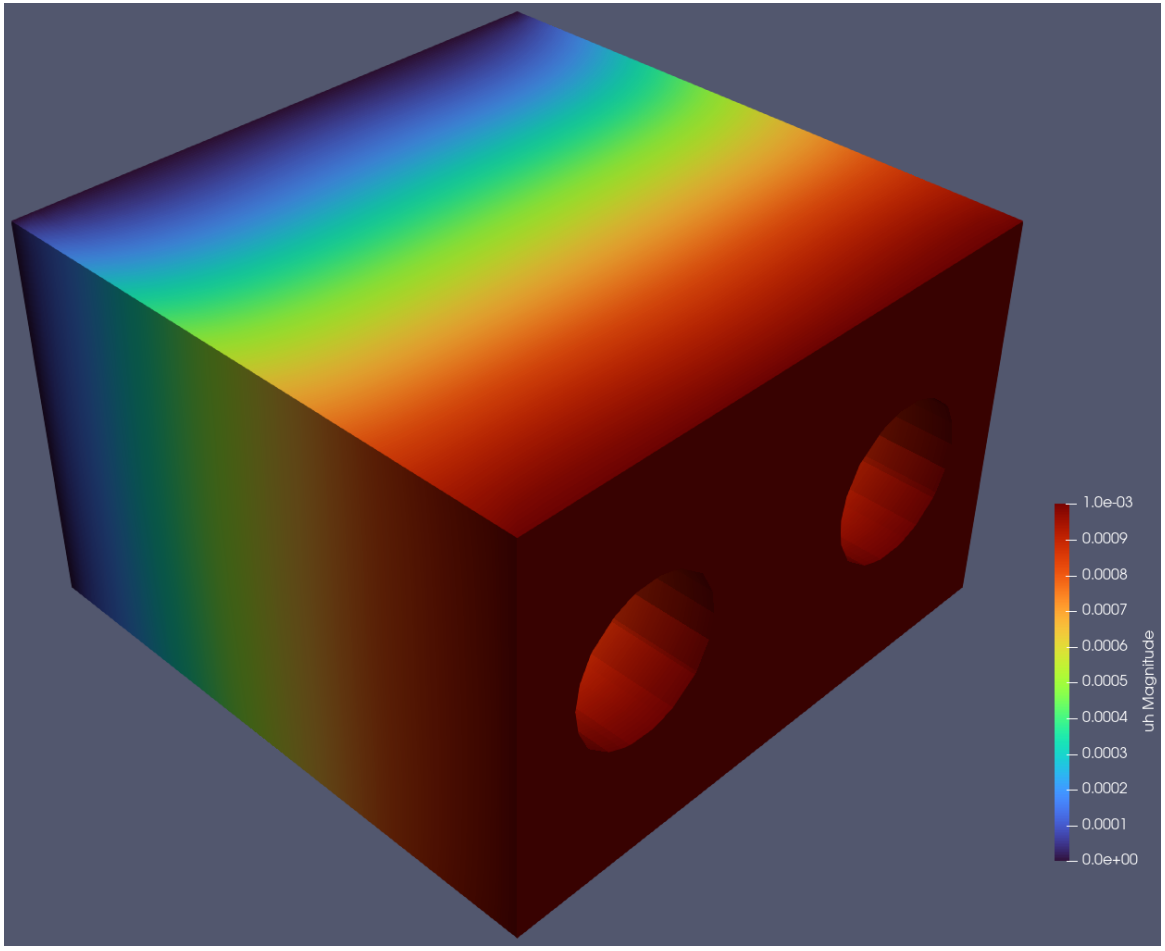


Figure 26: 区域 1 分析结果

Example 8: CutFEM 求接口问题

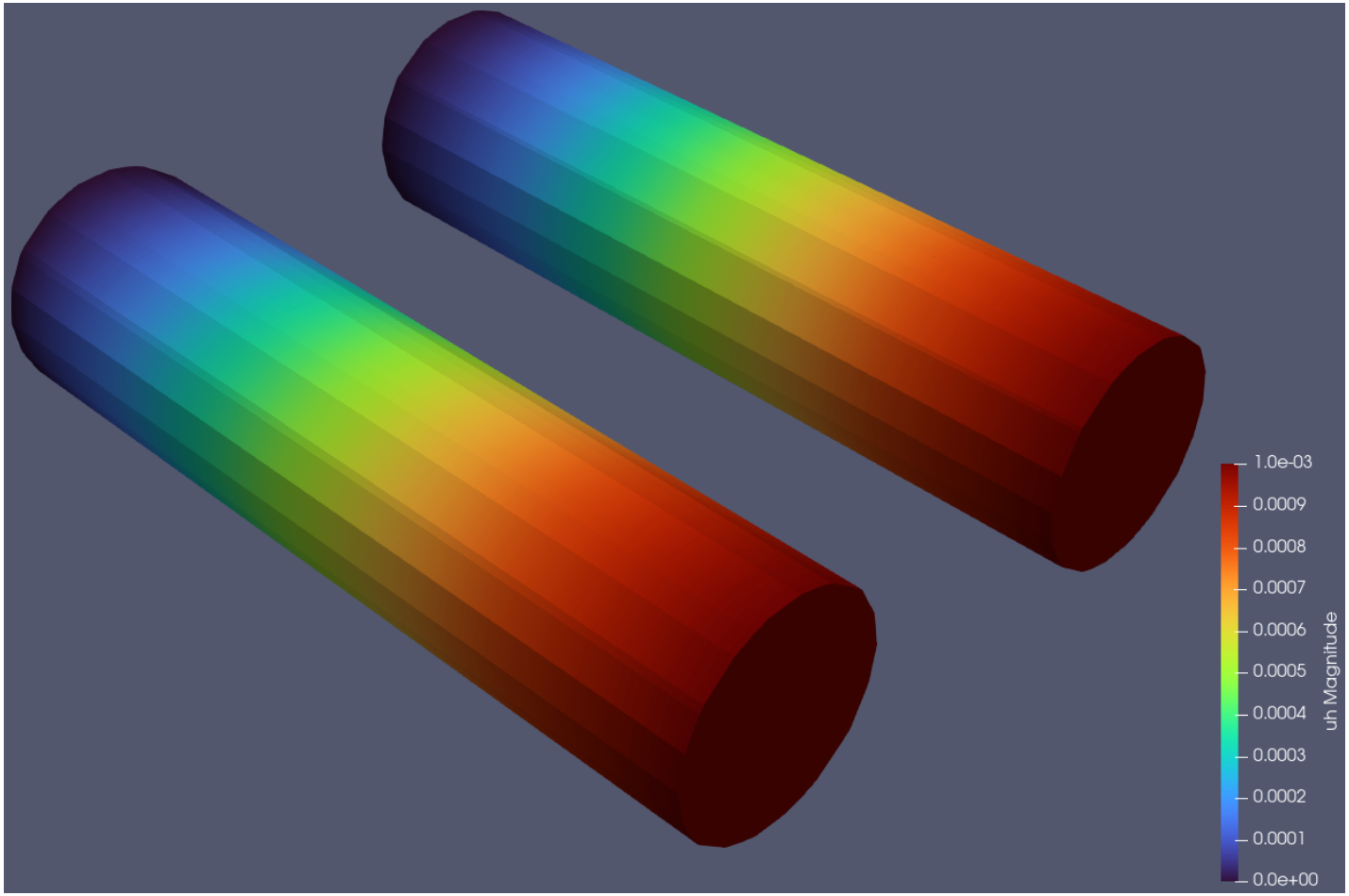


Figure 27: 区域 2 分析结果

Part V: 项目规划

项目规划

系统计划采用 C/S 架构设计，其中：

- 客户端运行在 Windows 系统，采用 C#语言开发 (界面与操作逻辑仿效 PrePoMax)，主要负责有限元分析的前处理和后处理；
- 服务端安装在运行 Linux 操作系统的边缘计算盒子中，主要负责调用有限元求解器。

Figure 28: 软件架构

PrePoMax 前后处理程序

PrePoMax is an open-source pre and post-processor for the Calculix FEM solver based on a modern user interface to speed up the FEM workflow.

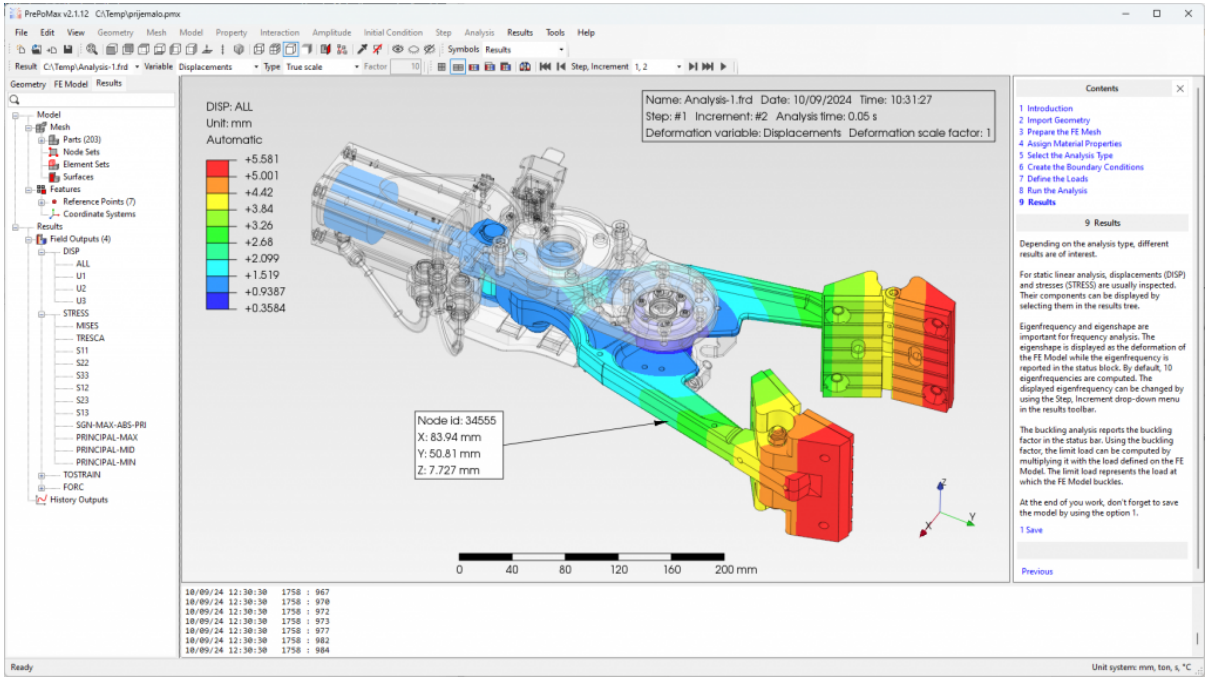


Figure 29: PrePoMax 操作界面

客户端

客户端的开发语言采用 C#，其界面与软件架构借鉴 PrePoMax，主要区别如下：

- 不同于 PrePoMax 采用 ActiViz.NET 作为后处理的显示控件，我们采用 Eyeshot FEM 作为显示控件；几何核心采用 Eyeshot，网格划分采用 Gmsh。

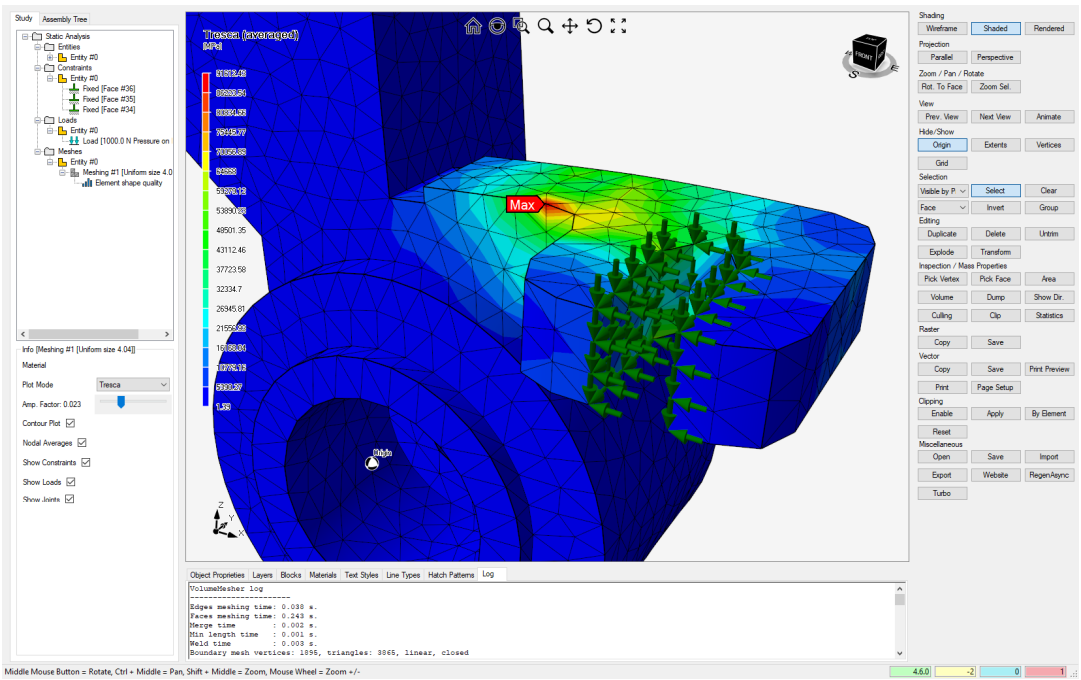


Figure 30: Eyeshot FEM 控件

客户端

客户端主要功能如下：

- **几何操作**：简单的几何建模功能及中间格式 step 文件导入；
- **前处理**：材料定义及管理、边界条件施加、载荷定义等；
- **网格划分**： Physical Group 定义、调用 Gmsh 进行网格划分；
- **切削区域定义**：采用隐式方法定义切削区域；
- **后处理**：分析结果展示、位移/应力/应变提取等。

服务端

服务端运行 Ubuntu 操作系统，主要功能如下：

- **有限元求解器**：支持静力学线弹性分析，内置 CutFEM 有限元算法；
- **简单后处理**：提供一个简单的用户界面（Dashboard），可以提交任务、监控执行情况，以及查看分析结果等。

服务端——有限元求解器

有限元求解器（LinearElasticitySolver，简称 LES) 采用 C++ 开发，其内核封装以下两种支持 CutFEM 的有限元库：

- **Gridap.jl**： Gridap provides a set of tools for the grid-based approximation of partial differential equations (PDEs) written in the Julia programming language.
- **NgSolve**： Netgen/NGSolve is a high performance multiphysics finite element software.

服务端——Dashboard

Trame是由开发 VTK 和 ParaView 的 Kitware 公司最新发布的 Web 框架，该框架采用 Python 语言，可以将原先 pyQt 开发的桌面程序迁移到 Web 端，提供接口丰富的数据可视化功能。

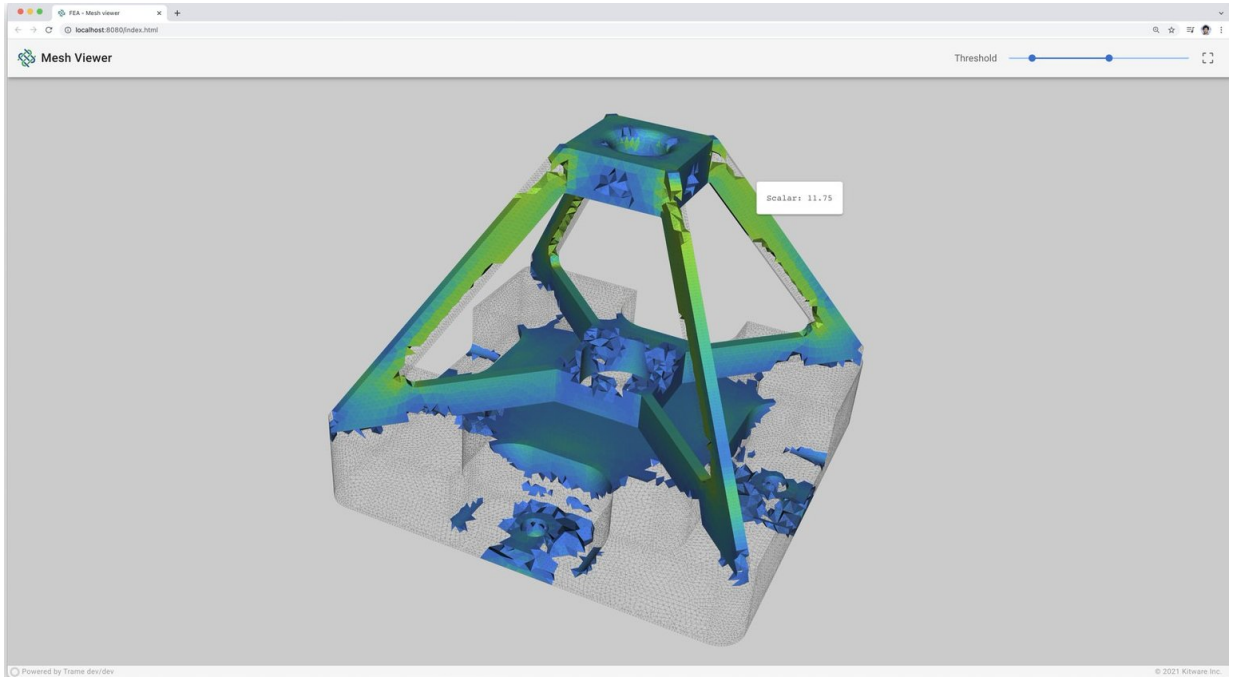


Figure 31: Eyeshot FEM 控件

技术路线图

开发工作的技术路线图：

- **开发客户端**：调研 Eyeshot FEM 控件，设计软件界面，规划操作逻辑，集成 Gmsh 网格划分，进行后继详细的功能开发和调试
- **开发有限元求解器**：一般有限元 -> CutFEM
- 开发 **Dashbord**

目前亟待解决的问题

目前亟待解决的问题：

- 购买 Eyeshot FEM 模块授权
- 敲定边缘计算盒子硬件配置
- 实验平台搭建，对有限元算法进行初步验证
- 协调隐函数的定义方式

水平集方法

水平集是跟踪轮廓和表面运动的一种数字化方法，它不直接对轮廓进行操作，而是将轮廓设置成一个高维函数的零水平集。

In two dimensions, the curve Γ is represented as the zero-level set of φ by

$$\Gamma = \{(x, y) \mid \varphi(x, y) = 0\}$$

and the level-set method manipulates Γ implicitly through the function φ. The dynamic motion of the boundary is governed by the so-called, level set equation:

$$\frac{\partial \phi}{\partial t} = v_n |\nabla \phi|$$

Where, v_n is the normal velocity and $|\nabla \phi|$ is the norm of the gradient of the level set function.

This function φ is assumed to take positive values inside the region delimited by the curve Γ and negative values outside.

$$\begin{aligned} \phi(x) > 0 \quad &: \quad x \in \Omega/\partial\Omega \\ \phi(x) = 0 \quad &: \quad x \in \partial\Omega \\ \phi(x) < 0 \quad &: \quad x \in D/\Omega \end{aligned}$$

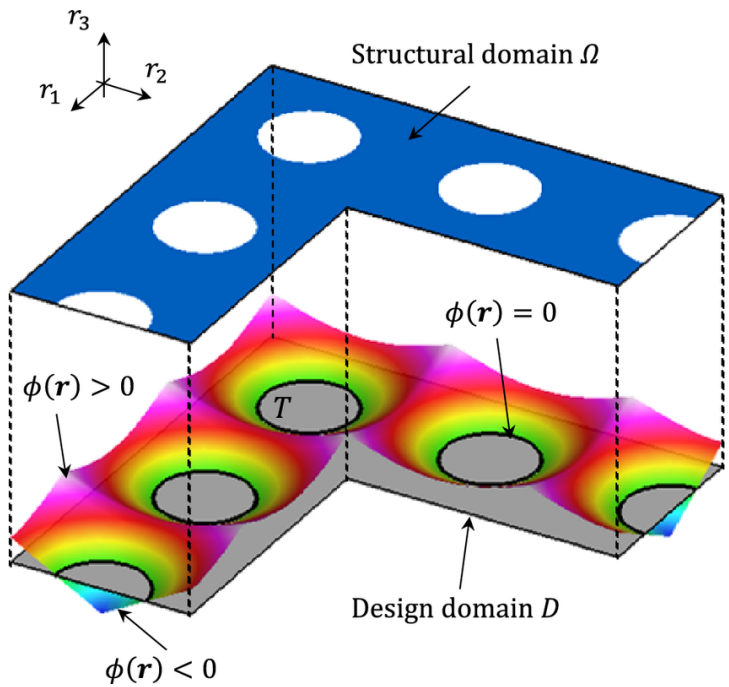


Figure 32: 水平集函数

If geometries are represented as level sets, Boolean operations can be expressed as simple operations on the corresponding LS functions.

Union:

$$\mathcal{M}_A = \mathcal{M}_B \cup \mathcal{M}_C \Leftrightarrow \Phi_A = \min(\Phi_B, \Phi_C)$$

Intersection:

$$\mathcal{M}_A = \mathcal{M}_B \cap \mathcal{M}_C \Leftrightarrow \Phi_A = \max(\Phi_B, \Phi_C)$$

Complement:

$$\mathcal{M}_A = \mathbb{R}^D \setminus \mathcal{M}_B \Leftrightarrow \Phi_A = -\Phi_B$$

Relative complement:

$$\mathcal{M}_A = \mathcal{M}_B \setminus \mathcal{M}_C \Leftrightarrow \Phi_A = \max(\Phi_B, -\Phi_C)$$

结束 Q & A