OpenFOM의 사소한... 문제들

길재흥

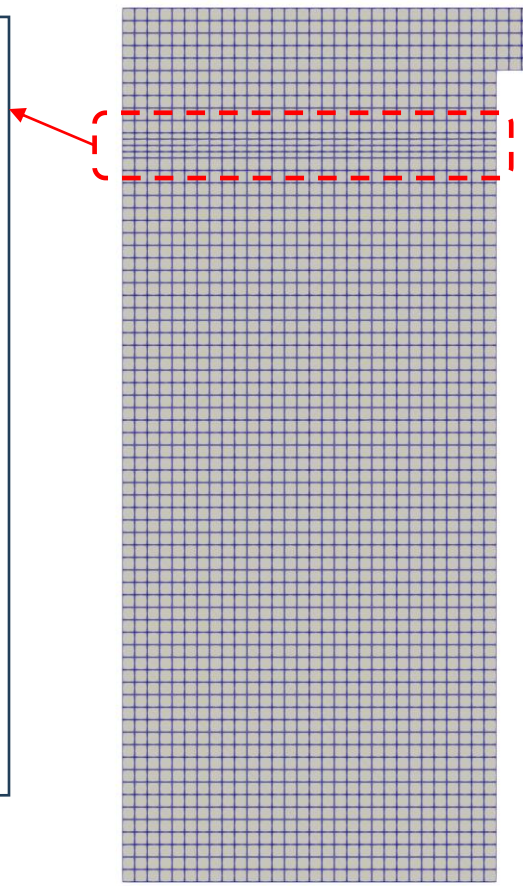
넥스트폼

Porous Media

간단한 문제

fvOptions

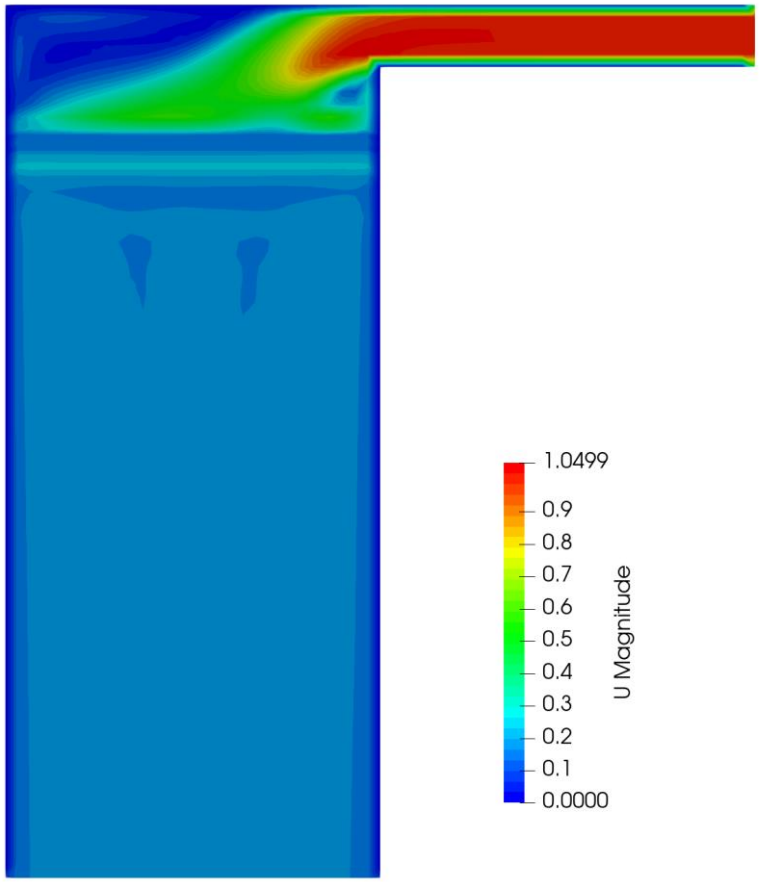
```
porous
{
    type    explicitPorositySource;

    explicitPorositySourceCoeffs
    {
        selectionMode   cellZone;
        cellZone        porous;
        type            powerLaw;

        C0      5000;
        C1      1.9;

        coordinateSystem
        {
            origin  (0 0 0);
            rotation none;
        }
    }
}
```

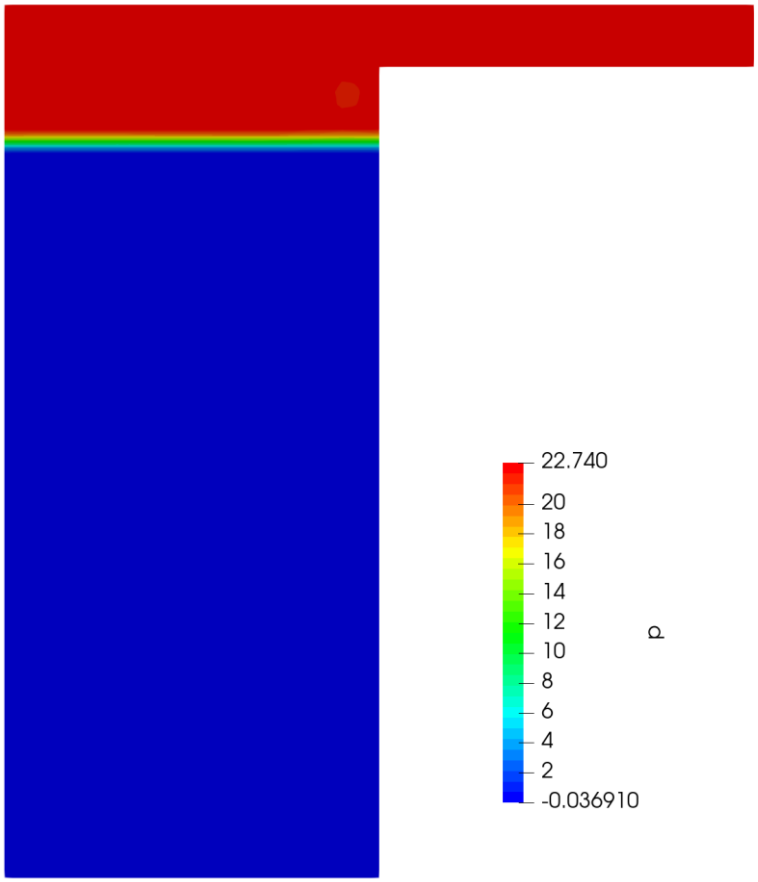


$$\vec{S} = -C_0 \left|\vec{U}\right|^{(C_1-1)} \vec{U}$$

power-law
porous media model

속도가 이상하다...


1.0499
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0000
U Magnitude


22.740
20
18
16
14
12
10
8
6
4
2
-0.036910
p

나만 그런가...?

Home News Forums Wiki Links Jobs Events Tools Feeds About Search

Home > Forums > Software User Forums > OpenFOAM > **OpenFOAM Running, Solving & CFD**

porousSimpleFoam: oscillating velocity in the porous zone

Welcome, ivywalker.
You last visited: Today at 02:21
Private Messages: Unread 0, Total 0.

User Panel | Blogs | FAQ | Community | New Posts | Updated Threads | Search | Quick Links | Log Out

12 Likes

Post Reply

Page 1 of 2 **1** 2 >

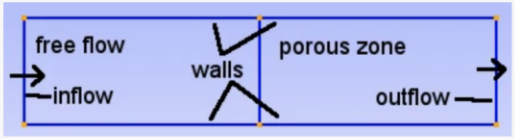
LinkBack | Thread Tools | Search this Thread | Rate Thread | Display Modes

December 4, 2010, 04:49 — **porousSimpleFoam: oscillating velocity in the porous zone** — #1

Se9a
New Member

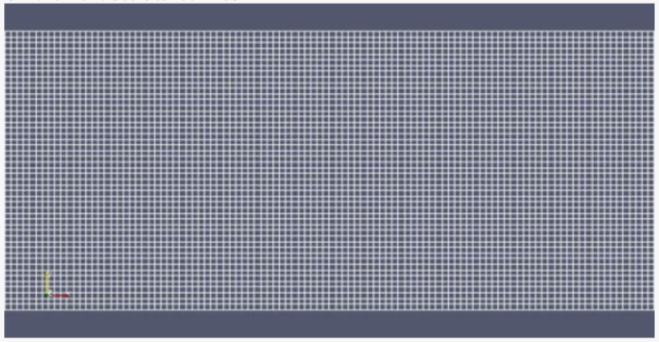
Sergei D.
Join Date: Mar 2009
Posts: 4
Rep Power: 0

Hello!
I try to use a porousSimpleFoam solver to simulate a flow in a 2D channel from two equal parts, a free flow part and a porous part (wall's type set in the 'slip' for simplicity):



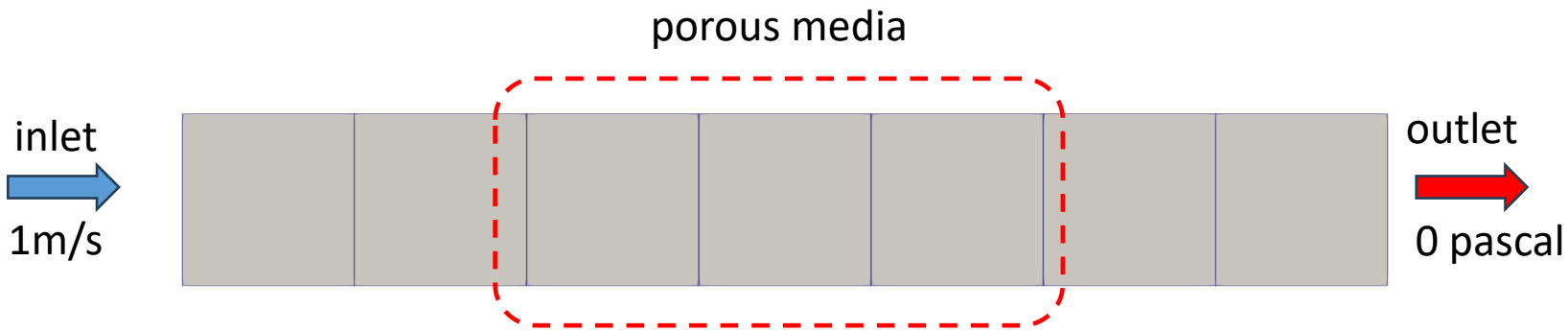
Base case is a angleDictImplicit.

My results are a good p, but strange oscillating U.
On the next sctrutured mesh:



극초간단 문제

- 1 m^3 정육면체 Cell 들로 이루어진 1차원 덕트
  - 비압축성 & 비점성 유체



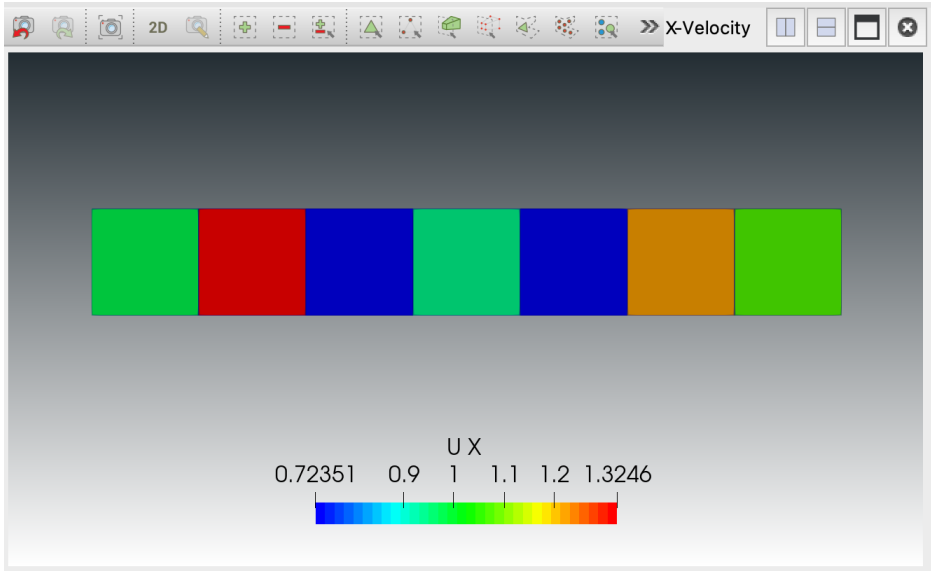
$$\vec{S} = -C_0 |\vec{U}|^{(C_1 - 1)} \vec{U}$$

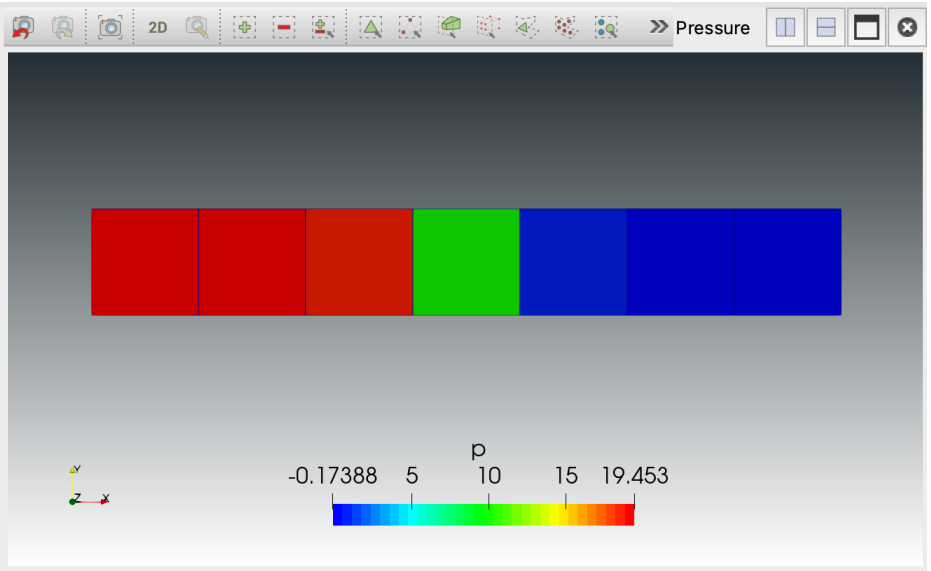
$$C_0 = 10,$$

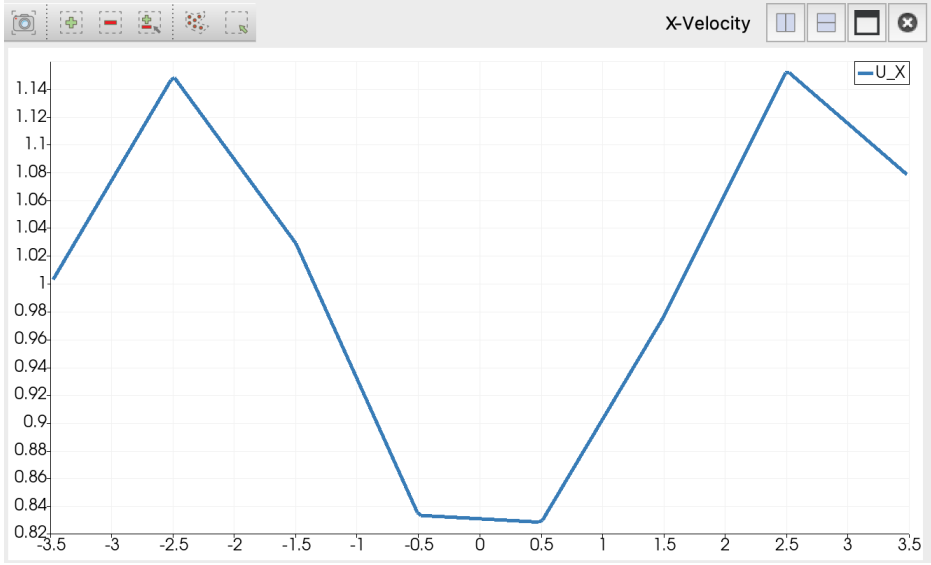
$$C_1 = 2$$

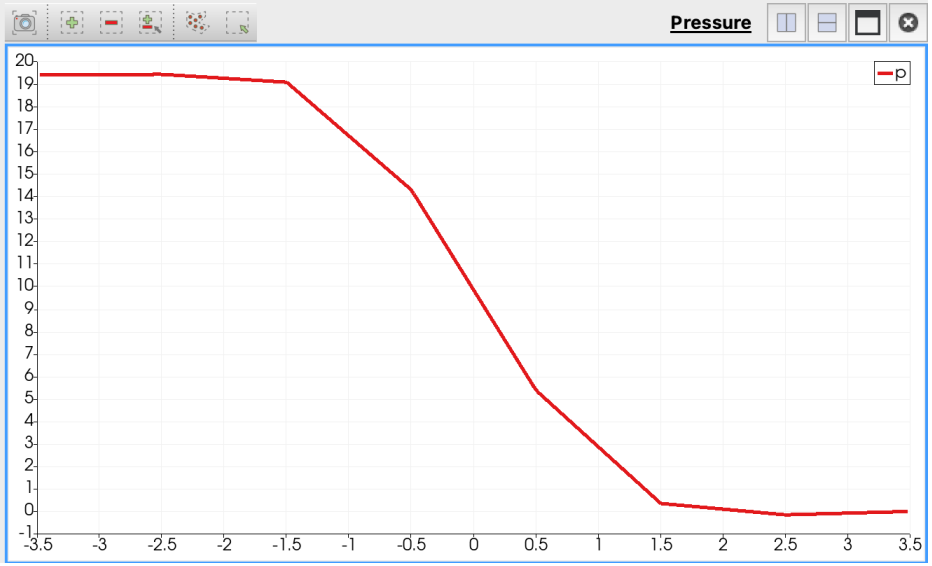
안풀어봐도 답을 알수 있다...

이걸 못 맞추네....

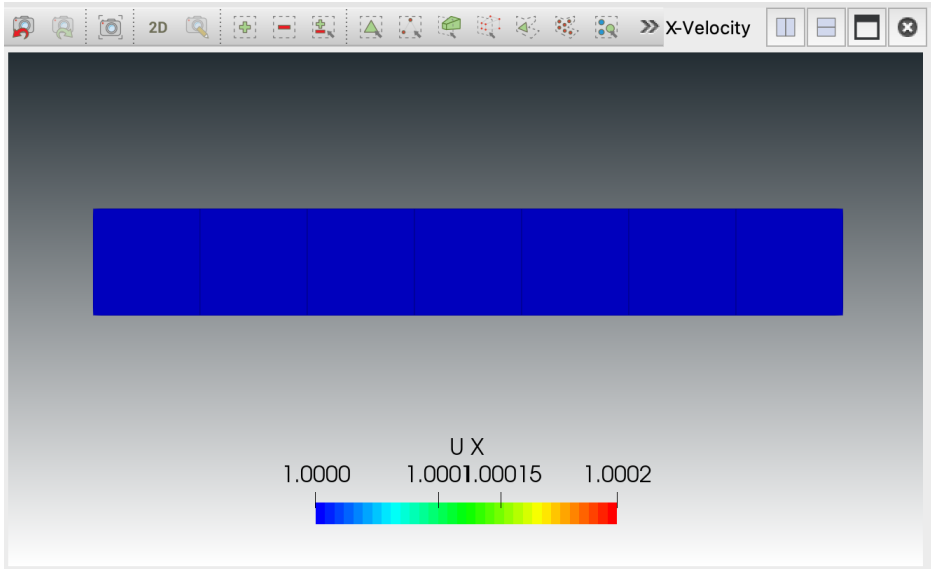

X-Velocity
U X
0.72351 0.9 1 1.1 1.2 1.3246

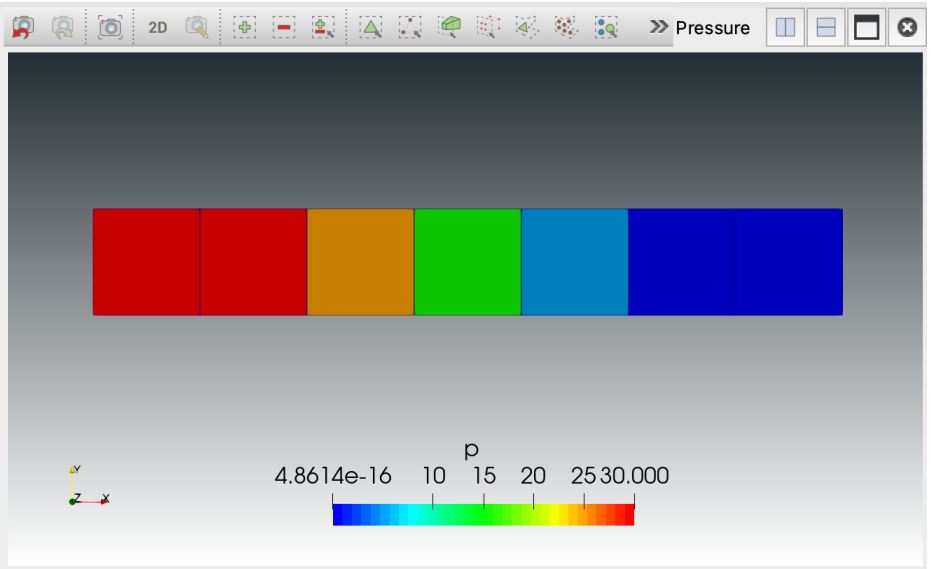

Pressure
p
-0.17388 5 10 15 19.453

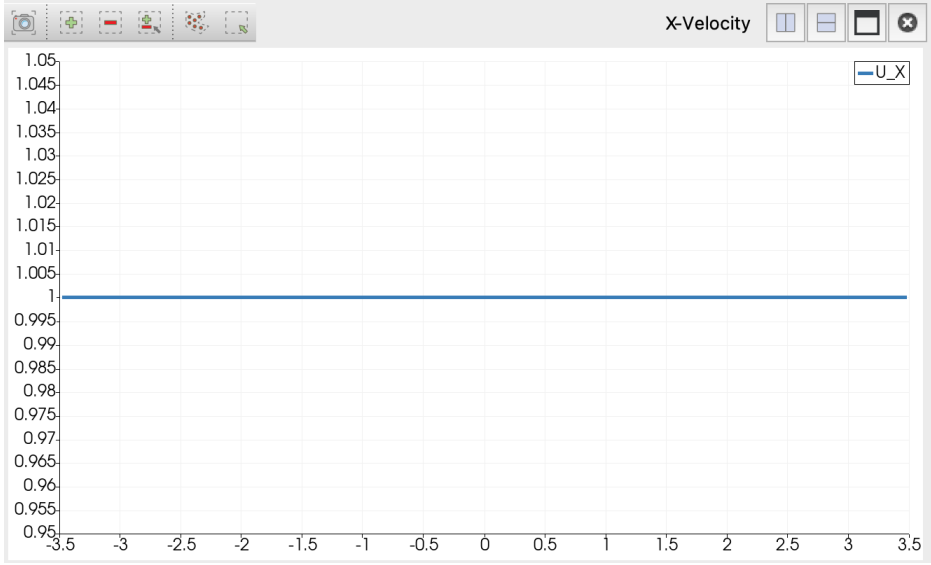

X-Velocity
U_X

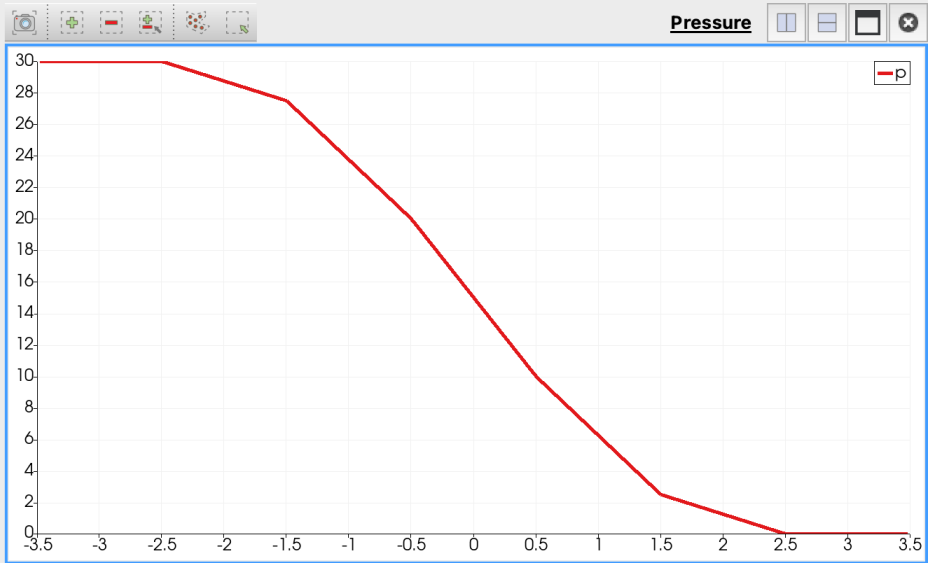

Pressure
p

정답

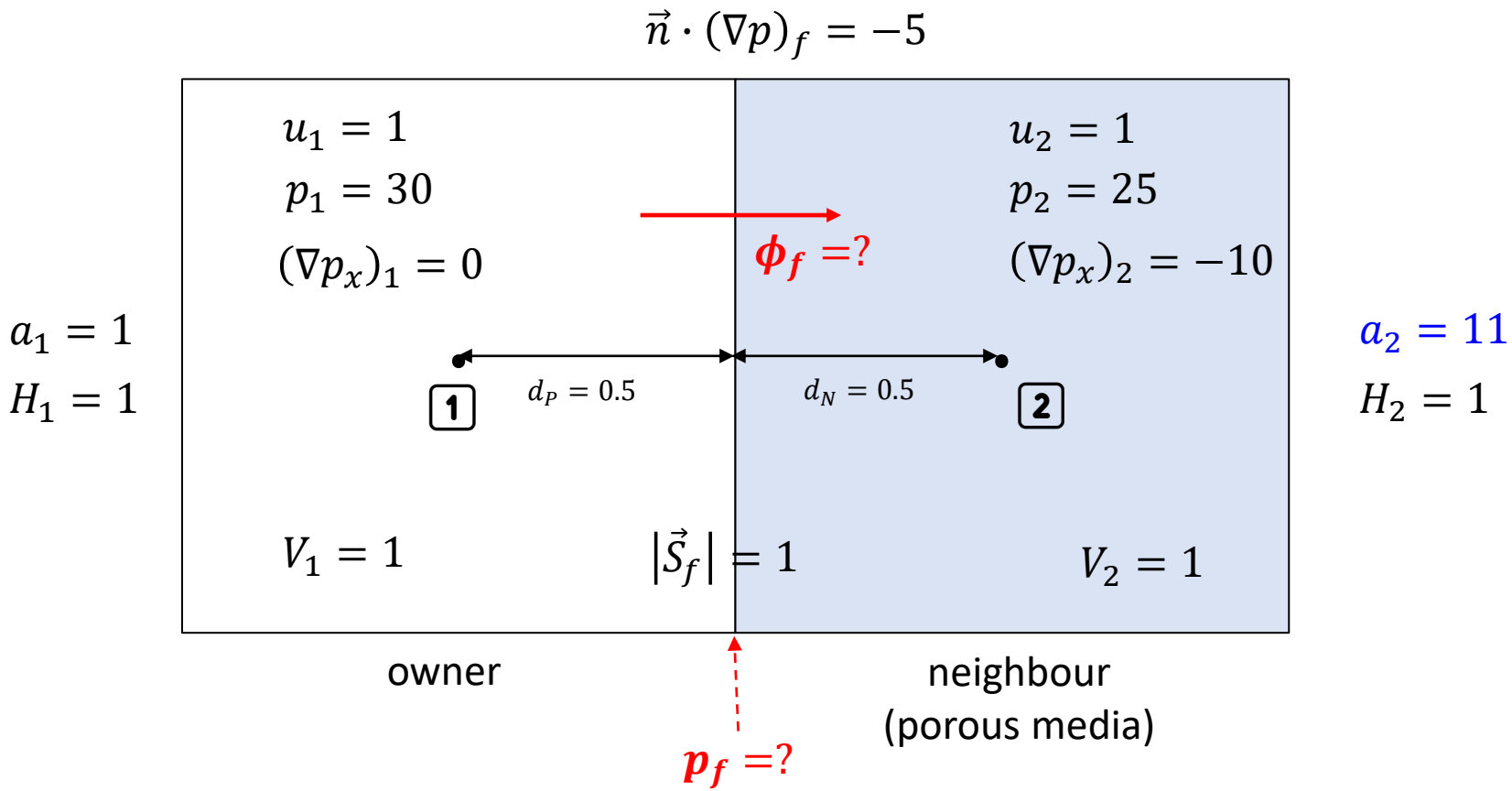

X-Velocity
U X
1.0000
1.0001
1.00015
1.0002


Pressure
p
4.8614e-16
10
15
20
25
30.000


X-Velocity
U_X
1.05
1.045
1.04
1.035
1.03
1.025
1.02
1.015
1.01
1.005
1
0.995
0.99
0.985
0.98
0.975
0.97
0.965
0.96
0.955
0.95
-3.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3
3.5


Pressure
p
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
0
-3.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3
3.5

안풀어봐도 아는 답


$\vec{n} \cdot (\nabla p)_f = -5$
$u_1 = 1$
$p_1 = 30$
$(\nabla p_x)_1 = 0$
$u_2 = 1$
$p_2 = 25$
$(\nabla p_x)_2 = -10$
$\phi_f = ?$
$a_1 = 1$
$H_1 = 1$
$a_2 = 11$
$H_2 = 1$
1
$d_P = 0.5$
$d_N = 0.5$
2
$V_1 = 1$
$|\vec{S}_f| = 1$
$V_2 = 1$
owner
neighbour
(porous media)
$p_f = ?$

p_f는 왜 필요한가

- Cell centroid의 pressure gradient를 구하기 위해서
  - Gauss gradient scheme

$$(\nabla p)_P = \frac{1}{V_P} \sum_f p_f \vec{S}_f$$

OpenFOAM 에서 p_f를 계산하는 방법

- Linear interpolation

fvSchemes

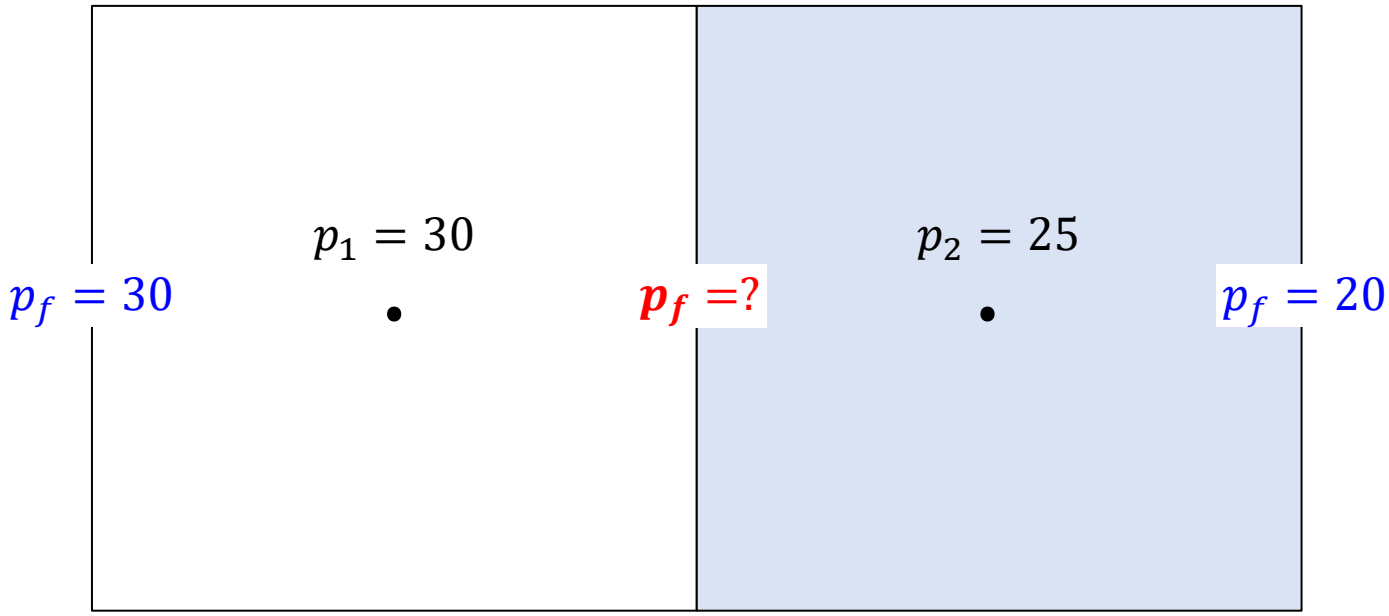
```
⋮
gradSchemes
{
    default             Gauss linear;
    limited             cellLimited Gauss linear 1.0;
}
⋮
```

- 또는

$FOAM_SRC/finiteVolume/interpolation/surfaceInterpolation/schemes

biLinearFit	FitData	localMax	quadraticLinearPureUpwindFit
cellCoBlended	fixedBlended	localMin	quadraticLinearUpwindFit
CentredFitScheme	harmonic	LUST	quadraticUpwindFit
clippedLinear	limiterBlended	midPoint	reverseLinear
CoBlended	linear	outletStabilised	skewCorrected
cubic	linearFit	pointLinear	UpwindFitScheme
cubicUpwindFit	linearPureUpwindFit	PureUpwindFitScheme	weighted
deferredCorrection	linearUpwind	quadraticFit	weightedFlux
downwind	localBlended	quadraticLinearFit	

Linear interpolation으로 p_f 를 구하면



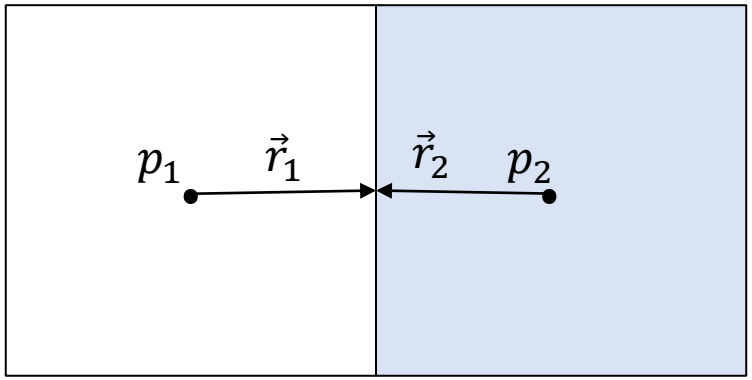
$$p_f = 27.5$$

$$(\nabla p_x)_1 = \frac{1}{V_1}\sum_f p_f S_f = -2.5 \quad \times$$

$$(\nabla p_x)_2 = \frac{1}{V_2}\sum_f p_f S_f = -7.5 \quad \times$$

불연속적인 압력구배를 구현하려면

- Porous Media의 바깥쪽 경계는 압력에 대해서 벽면처럼 작용해야 한다.
  - zeroGradient 경계조건?
- Internal Face이므로 경계조건으로 처리할수 없음
  - 특별한 Interpolation 방법을 사용해야 함



$$p_f = \frac{1}{2}(p_1 + p_2) + \frac{1}{2}\{\vec{r}_1 \cdot (\nabla p^*)_1 + \vec{r}_2 \cdot (\nabla p^*)_2\}$$

OpenFOAM 에서 ϕ_f를 계산하는 방법

- 일단 linear interpolation

pEqn.H

```
volScalarField rAU(1.0/UEqn.A());
surfaceScalarField phiHbyA("phiHbyA", fvc::flux(HbyA));
⋮
tmp<volScalarField> rAtU(rAU);
⋮
// Non-orthogonal pressure corrector loop
while (simple.correctNonOrthogonal())
{
    fvScalarMatrix pEqn
    (
        fvm::laplacian(rAtU(), p) == fvc::div(phiHbyA)
    );
⋮
    if (simple.finalNonOrthogonalIter())
    {
        phi = phiHbyA - pEqn.flux();
    }
}
⋮
```

$$\phi_f = \left\{ \vec{S}_f \cdot \overline{\left(\frac{\vec{H}}{a}\right)}_f - \overline{\left(\frac{V}{a}\right)}_f |\vec{S}_f| (\vec{n} \cdot (\nabla p)_f) \right\}$$

linear interpolate

Linear interpolation으로 ϕ_f 를 구하면

- Linear interpolation

$$\phi_f = \frac{d_2}{d_1+d_2}\left(\frac{H_1}{a_1}\right) + \frac{d_1}{d_1+d_2}\left(\frac{H_2}{a_2}\right) - \left\{\frac{d_2}{d_1+d_2}\left(\frac{V_1}{a_1}\right) + \frac{d_1}{d_1+d_2}\left(\frac{V_2}{a_2}\right)\right\}\{\vec{n}\cdot(\nabla p)_f\}$$

$$= \frac{1}{2}\left(\frac{H_1}{a_1} + \frac{H_2}{a_2}\right) - \frac{1}{2}\left(\frac{V_1}{a_1} + \frac{V_2}{a_2}\right)\{\vec{n}\cdot(\nabla p)_f\}$$

$$= 3.27272727\ldots \quad \times$$

생각해 볼수 있는 것들

- Interpolation weighting factor를 정의하는 근거
  - Cell center와 face center의 거리(d_1, d_2) ?
  - Cell volume(V_1, V_2) ?
  - 압력구배의 불연속($(\nabla p)_1, (\nabla p)_2$) ?
    - porous media 모델의 momentum source는 그 자체로 압력구배를 표현하고 있다.

$$\nabla p = -\boldsymbol{C_0}\left|\vec{\boldsymbol{U}}\right|^{(\boldsymbol{C_1}-\mathbf{1})}\vec{U}$$

- a_1, a_2는 운동량보존방정식에서 선형화한 momentum source의 계수를 포함하고 있다.

$$a_1 = \sum_f \left(w_f \phi_f\right), \qquad a_2 = \sum_f \left(w_f \phi_f\right) + \boldsymbol{C_0}\left|\vec{\boldsymbol{U}}\right|^{(\boldsymbol{C_1}-\mathbf{1})}$$

Weighting factor를 역으로 구해보면…

$$\phi_f = w\left(\frac{H_1}{a_1}\right) + (1-w)\left(\frac{H_2}{a_2}\right) - \left\{w\left(\frac{V_1}{a_1}\right) + (1-w)\left(\frac{V_2}{a_2}\right)\right\}\{\vec{n}\cdot(\nabla p)_f\}$$

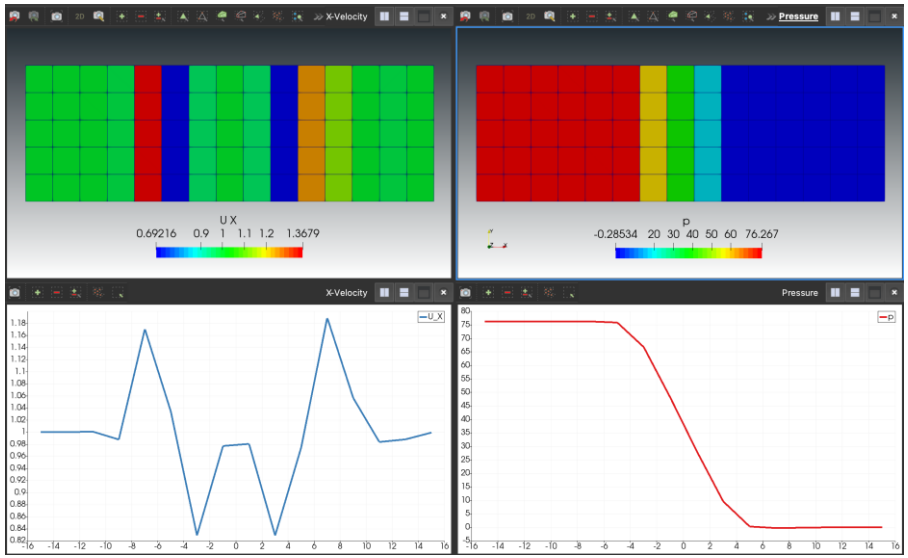
$$w = \frac{\phi_f - \left[\frac{H_2}{a_2} - \left(\frac{V_2}{a_2}\right)\{\vec{n}\cdot(\nabla p)_f\}\right]}{\frac{H_1}{a_1} - \frac{H_2}{a_2} - \left(\frac{V_1}{a_1} - \frac{V_2}{a_2}\right)\{\vec{n}\cdot(\nabla p)_f\}}$$

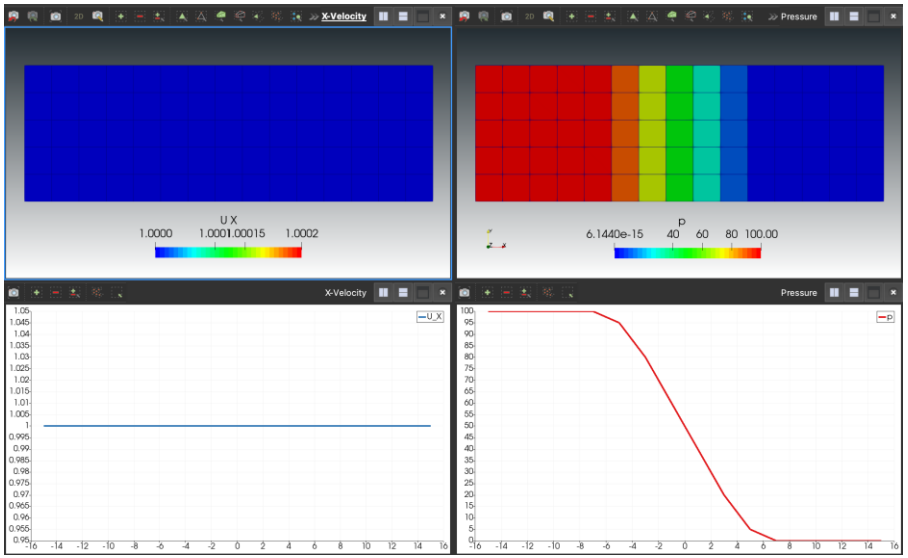
porous media iterface를 압력에 대한 경계면처럼 생각하면…

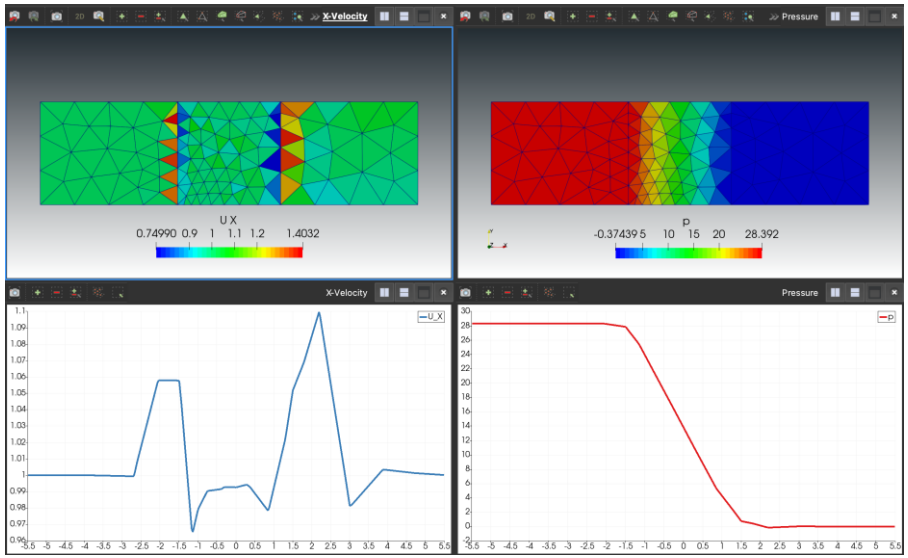
$$\vec{n}\cdot(\nabla p)_f = \frac{d_2}{d_1 + d_2}(\nabla p)_2$$

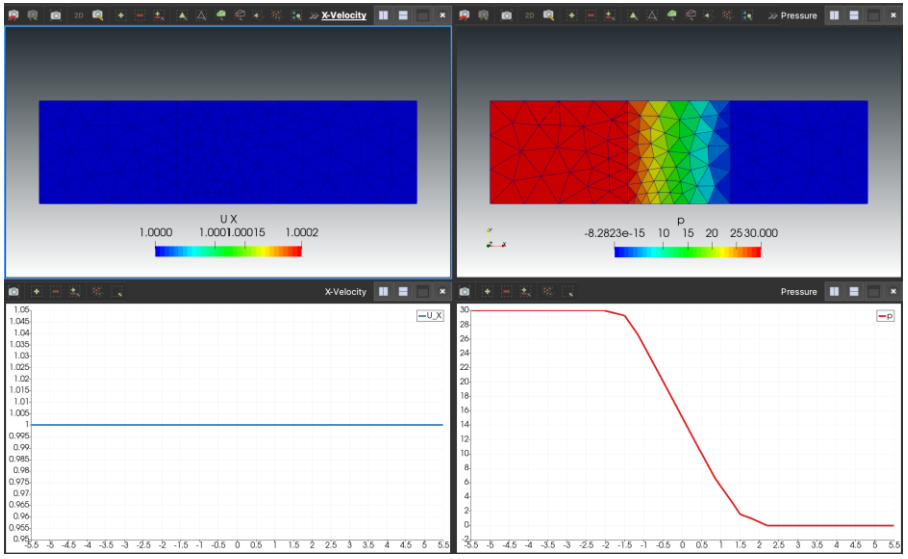
$$\boldsymbol{w = \frac{a_1 d_1}{a_1 d_1 + a_2 d_2}}$$

2차원 덕트 해석





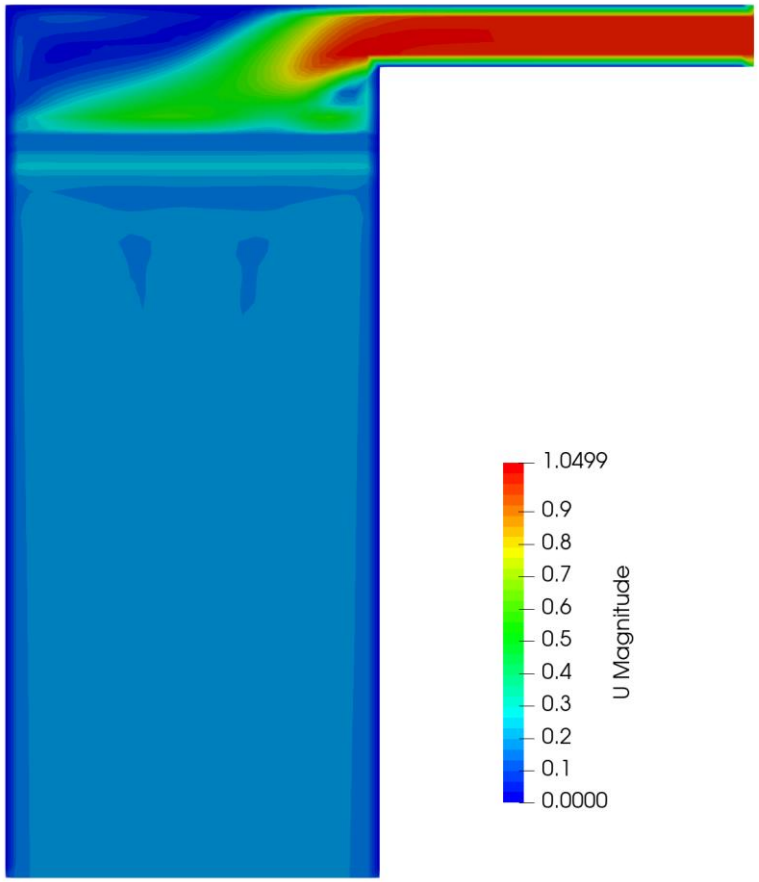




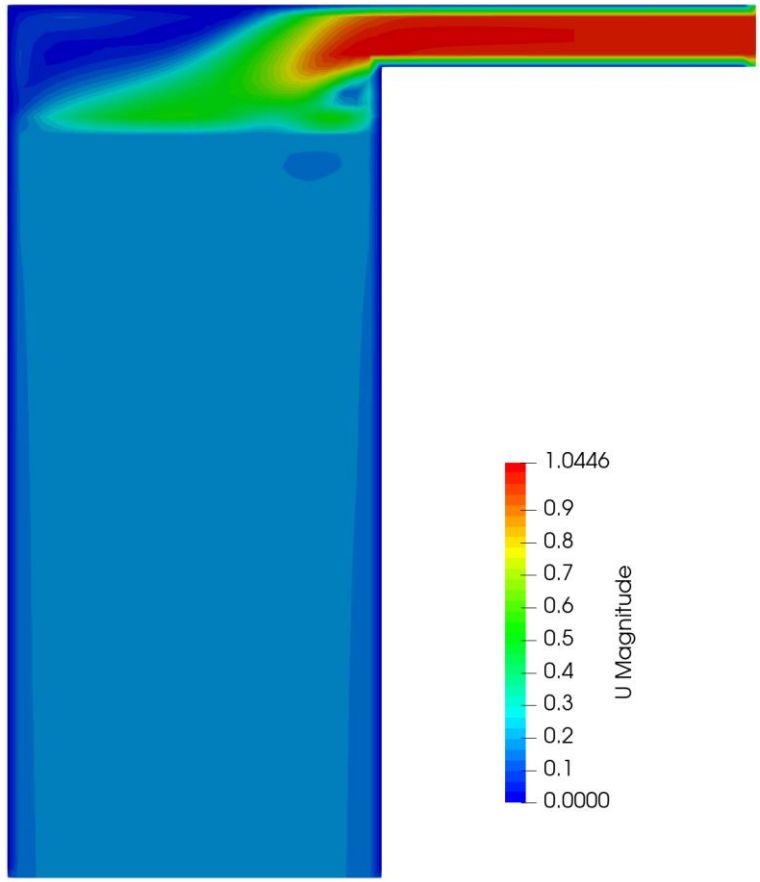
simpleFoam

simpleNFoam

이상했던 문제

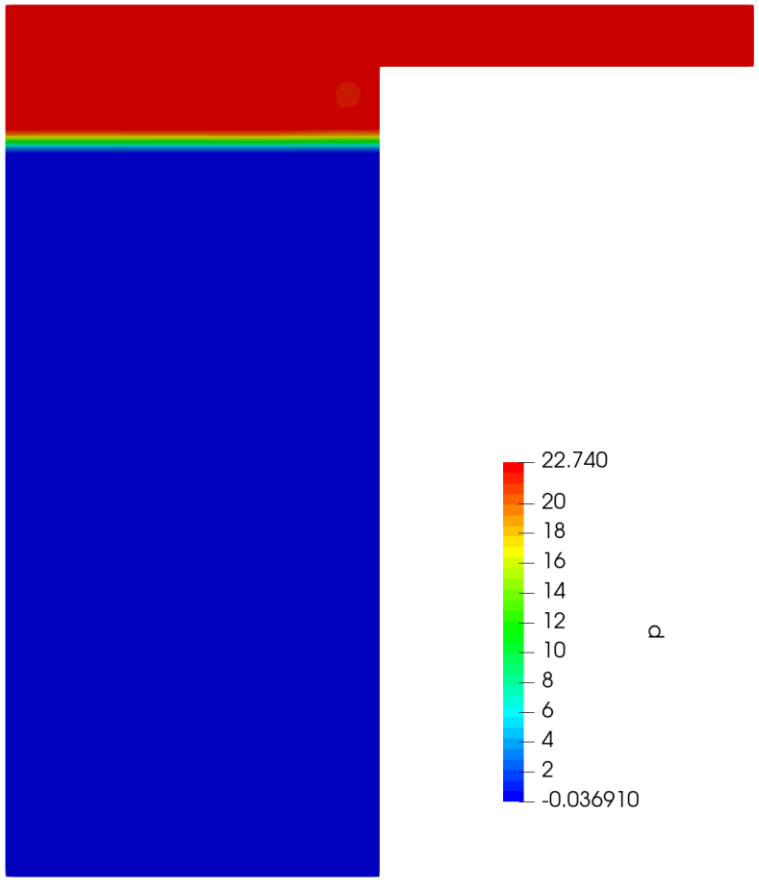


simpleFoam

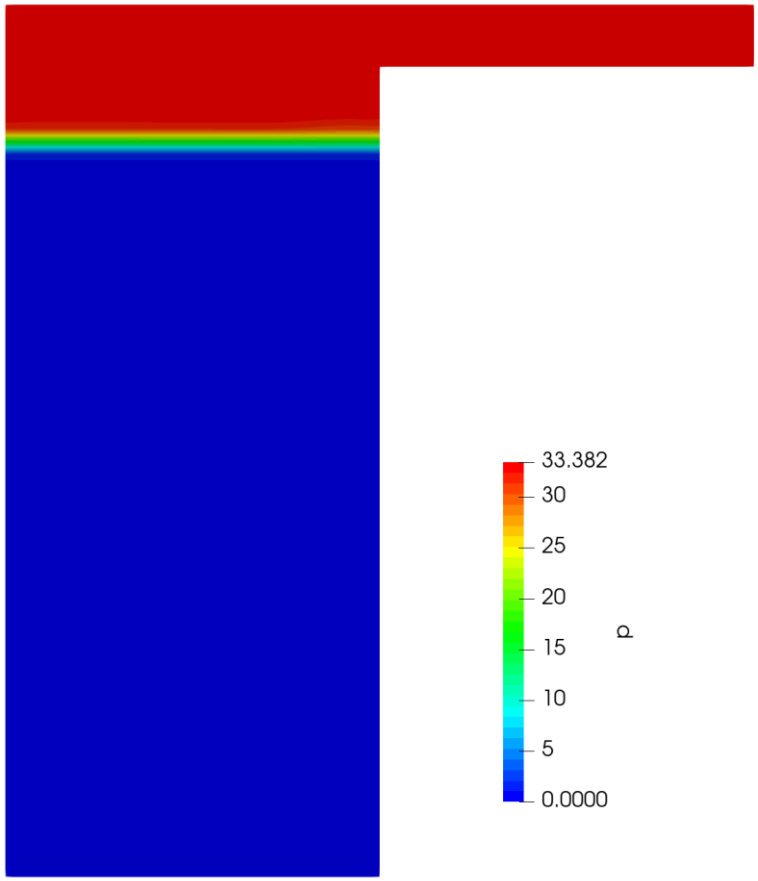


simpleNFoam

이상했던 문제



simpleFoam

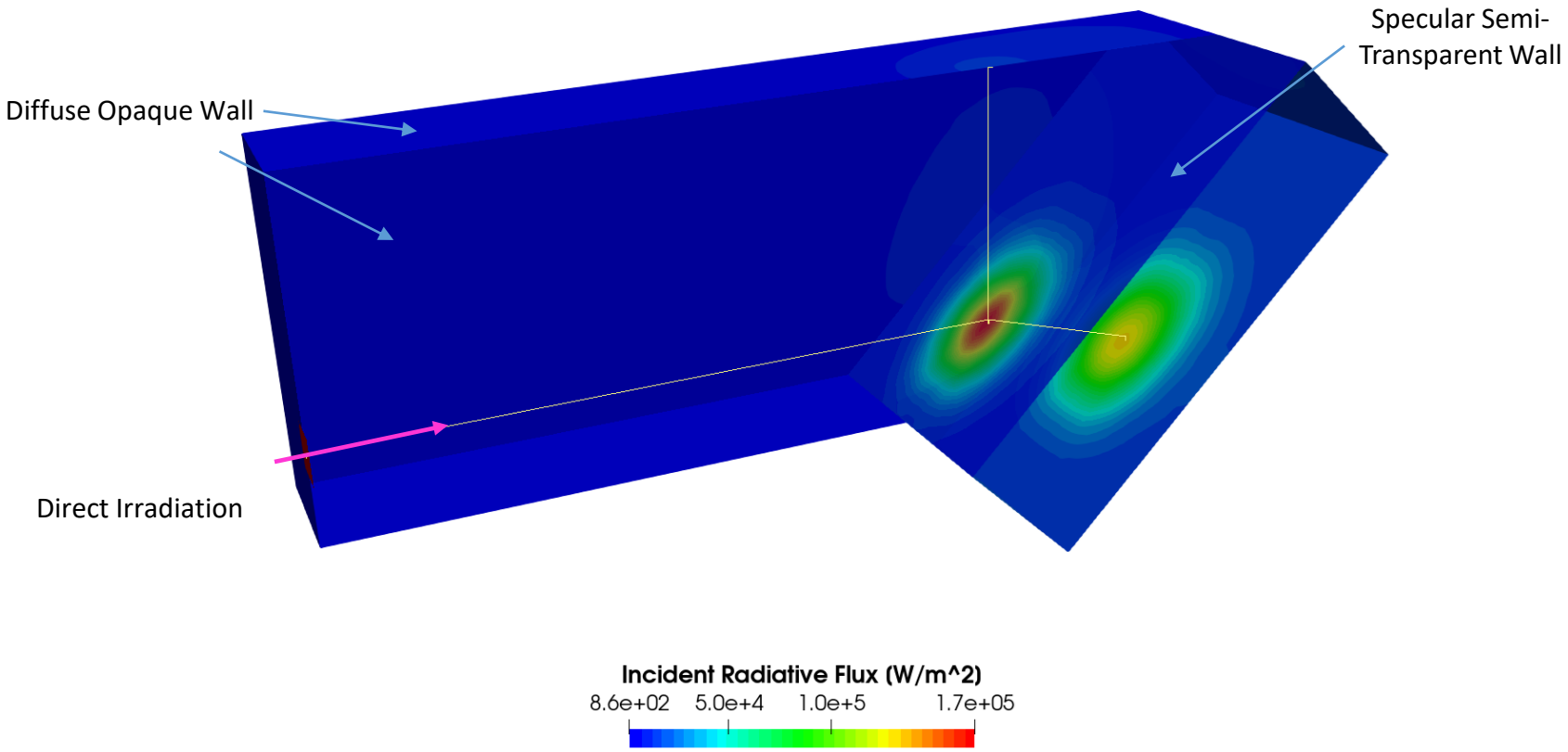


simpleNFoam

Radiative Heat Flux가 있는 경계면의 온도 경계조건

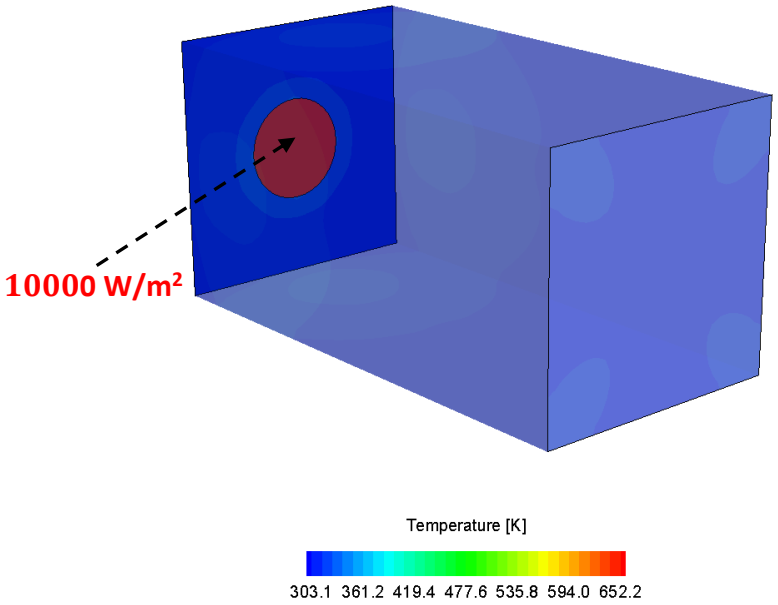
fvDOM 모델에 투과 및 굴절 구현

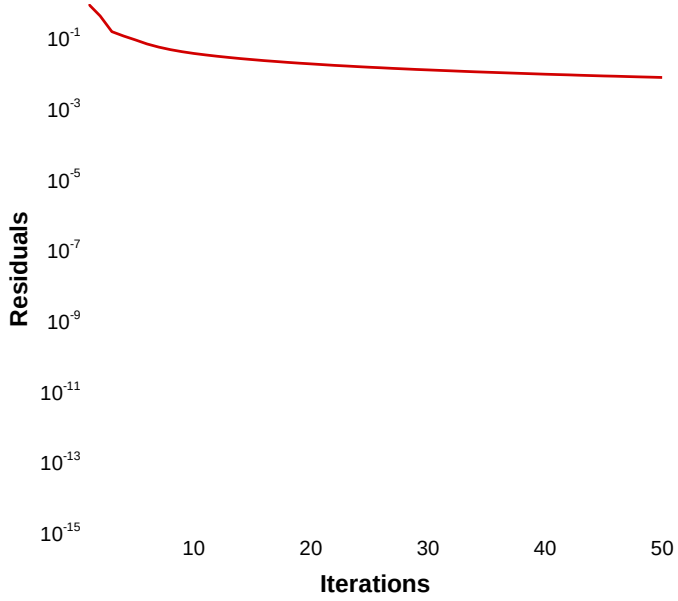
- 45° 기울어진 Glass의 경사면에 Radiation Flux 조사
  - Fresnel's Relation에 따른 투과 및 반사가 일어남을 확인
  - Snell's Law로 계산된 굴절각과 정확히 일치하며 굴절됨을 확인



간단한 복사열전달 문제

- 10000 W/m^{2}의 열을 발산하는 열원을 둘러싼 Box
- 열원을 제외한 모든 외벽은 300 K의 외부와 대류열전달(h = 10 W/m^{2}K)





외벽의 온도경계조건

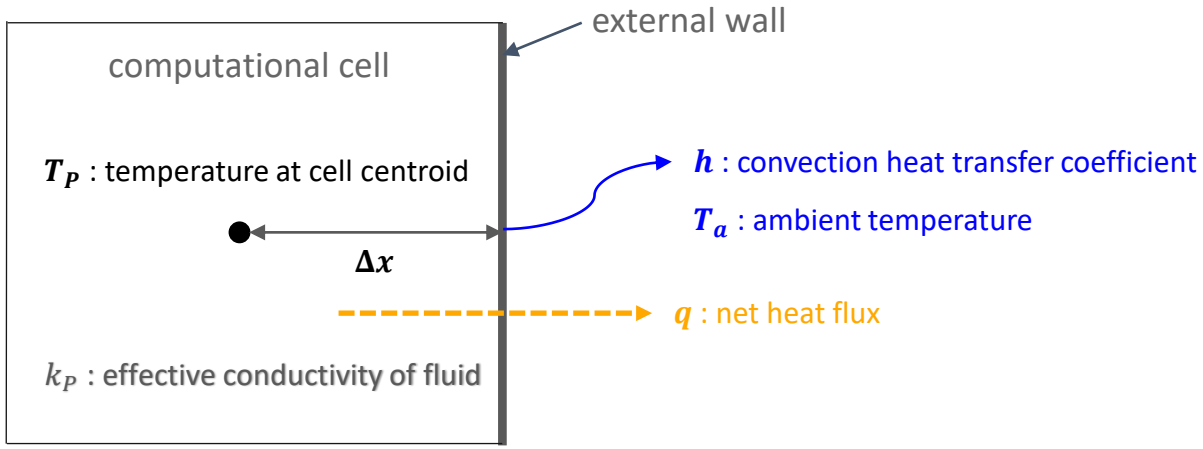
- under-relax를 하지 않으면 발산…

T

```
⋮
boundaryField
{
    wall
    {
        type            externalWallHeatFluxTemperature;
        kappaMethod     fluidThermo;
        mode            coefficient;
        h               uniform 10;
        Ta              uniform 300;
        value           uniform 300;
        qrRelaxation    1e-6;
        qr              qr;
    }
⋮
```

에너지방정식에 대한 벽면 경계조건

- Example: External Wall에 대한 대류열전달 경계조건
  - **Radiative Heat Flux가 없는 경우**



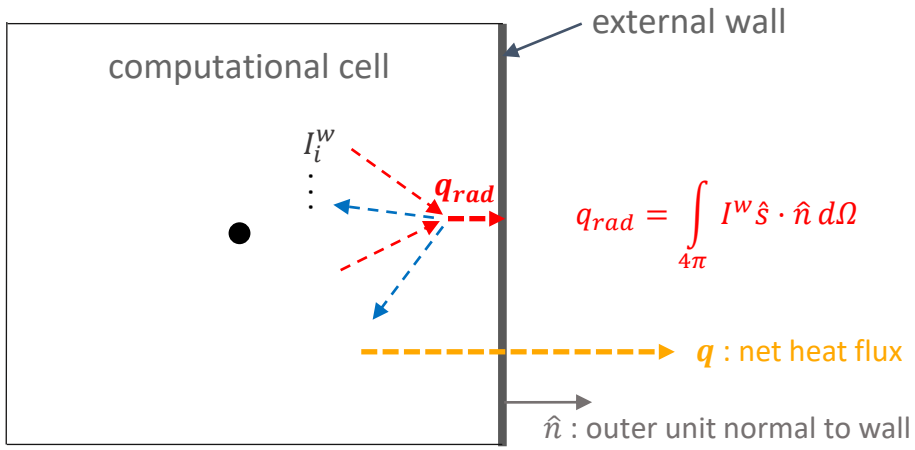
T_b : temperature at wall boundary(경계조건)

$$q = k_P \frac{T_P - T_b}{\Delta x} = h(T_b - T_a) \text{ : energy balance on a boundary}$$

$$T_b = fT_a + (1-f)T_P\,, \qquad f = \frac{h}{h + \frac{k_P}{\Delta x}}$$

에너지방정식에 대한 벽면 경계조건

- Example: External Wall에 대한 대류열전달 경계조건
  - **Radiative Heat Flux가 있는 경우**
    - **Energy balance에 radiative heat flux가 포함되어야 한다**

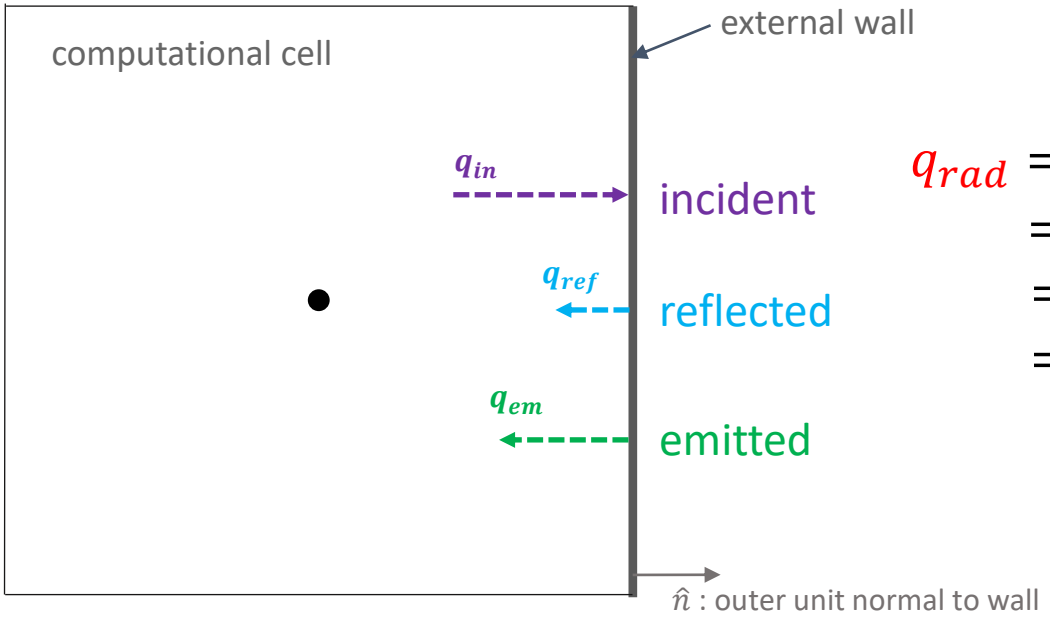


$$q = k_P \frac{T_P - T_b}{\Delta x} + q_{rad} = h(T_b - T_a) \text{ : energy balance on a boundary}$$

$$T_b = f\frac{hT_a + q_{rad}}{h} + (1-f)T_P\,, \qquad f = \frac{h}{h + \frac{k_P}{\Delta x}}$$

Radiative Heat Flux의 계산

- 이론적으로 모든 Radiative Intensity의 벽면에 수직인 성분을 전체 입체각(Solid Angle)에 대하여 적분한 값
  - Grey radiation에 대하여 벽면에서의 Radiative Heat Flux(q_{rad})는 다음과 같이 성분을 나누어 생각할 수 있다



$$q_{rad} = q_{in} - q_{ref} - q_{em}$$
$$= q_{in} - (1-\varepsilon)q_{in} - q_{em}$$
$$= \varepsilon q_{in} - q_{em}$$
$$= \varepsilon q_{in} - n^2 \varepsilon \sigma T_b^4$$

ε : $emissivity$

q_{rad} 를 에너지방정식의 경계조건에 반영

- 기존 OpenFOAM Library의 방법
  - Radiative Intensity Equation들을 계산한 후 q_{rad}를 다음 식을 이용해서 계산

$$q_{rad} = \varepsilon q_{in} - n^2 \varepsilon \sigma (T_b^*)^4$$

$$q_{in} = \int_{4\pi} I^I \hat{s} \cdot \hat{n} \, d\Omega \,, \qquad \hat{s} \cdot \hat{n} \geq 0$$

T_b^* 는 에너지방정식의 이전 iteration에서 계산된 경계면의 온도

  - 위와 같이 계산된 q_{rad} 의 값을 에너지방정식의 경계조건식에 그대로 대입

$$T_b = f \frac{hT_a + q_r}{h} + (1 - f) T_P \,, \qquad f = \frac{h}{h + \frac{k_P}{\Delta x}}$$

기존 방식의 문제점

- 앞의 식에서 에너지방정식의 경계조건식에 나타나는 q_{rad} 를 계산식의 형태로 다시 쓰면 다음과 같다

$$T_b = f\frac{hT_a + \varepsilon q_{in} - n^2\varepsilon\sigma(T_b^*)^4}{h} + (1-f)T_P\,, \qquad f = \frac{h}{h + \frac{k_P}{\Delta x}}$$

  - 이 경우 T_b의 변화는 매 iteration 마다 T_b^* 의 네제곱에 비례하는 flux의 영향을 받기때문에 매우 불안정하며, 열원과 주변의 온도차가 큰 경우 q_{rad}의 값에 under-relaxation을 적용해야 한다

  - 열원과 주변 온도차가 수백K에 이르면 과도한 under-relaxation factor를 적용해야만 계산이 가능하며 이 경우 수렴성이 너무 낮아져서 사실상 사용할수 없는 경계조건이 된다

문제 해결 방안

- 선형화(Linearization)
  - 경계면 온도의 네제곱(T_b^4)으로 표현된 q_{rad}에 대한 식을 T_b에 대해 선형(1차식)이 되도록 수정
    - T_b의 이전 iteration 값(T_b^*)에 대한 의존성의 크기를 줄이고 에너지 방정식을 이산화한 Matrix의 대각지배성(diagonal dominance) 을 강화하여 계산 안정성 향상
  - 일반적인 함수 $f(x)$의 선형화 방법

$$f(x) = f^* + \left(\frac{\partial f}{\partial x}\right)^* (x - x^*)$$

x^* 는 이전 iteration에서 계산된 값

$$f^* = f(x^*)$$

$$\left(\frac{\partial f}{\partial x}\right)^* = \frac{\partial f(x^*)}{\partial x}$$

Radiative Heat Flux 선형화

- q_{rad}를 경계면 온도 T_b에 대한 식으로 표현

$$q_{rad}(T_b) = \varepsilon q_{in} - n^2 \varepsilon \sigma T_b^4$$

$$\left(\frac{\partial q_r}{\partial T_b}\right)^* = -4n^2 \varepsilon \sigma (T_b^*)^3$$

  - 위의 식으로부터 선형화된 Radiative Heat Flux(q_{rad}^L)를 표현하면 다음과 같다

$$q_{rad}^L = \varepsilon q_{in} - n^2 \varepsilon \sigma (T_b^*)^4 - 4n^2 \varepsilon \sigma (T_b^*)^3 (T_b - T_b^*)$$
$$= \varepsilon q_{in} + 3n^2 \varepsilon \sigma (T_b^*)^4 - 4n^2 \varepsilon \sigma (T_b^*)^3 T_b$$

에너지방정식의 경계조건에 반영

- 선형화된 Radiative Heat Flux 를 반영한 온도 경계조건
  - 앞에서 표현한 선형화된 q_{rad}^{L} 을 이용해 Energy Balance를 표현

$$q = k_P \frac{T_P - T_b}{\Delta x} + q_{rad}^{L} = h(T_b - T_a)$$

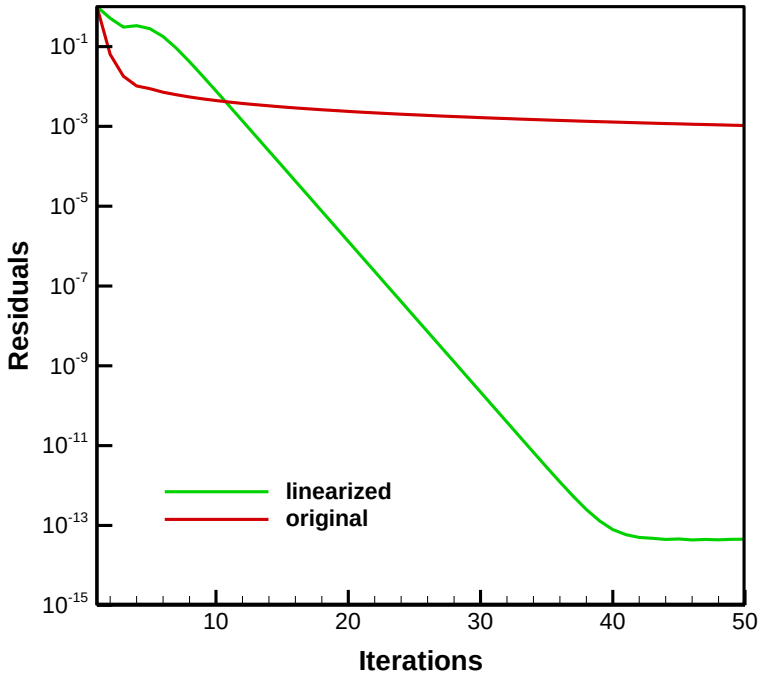
$$k_P \frac{T_P - T_b}{\Delta x} + \varepsilon q_{in} + 3n^2 \varepsilon \sigma (T_b^*)^4 - 4n^2 \varepsilon \sigma (T_b^*)^3 T_b = h(T_b - T_a)$$

  - 위의 식을 정리하면 Radiative Heat Flux를 반영한 External Wall의 대류열전달 경계조건은 다음과 같이 쓸수 있다

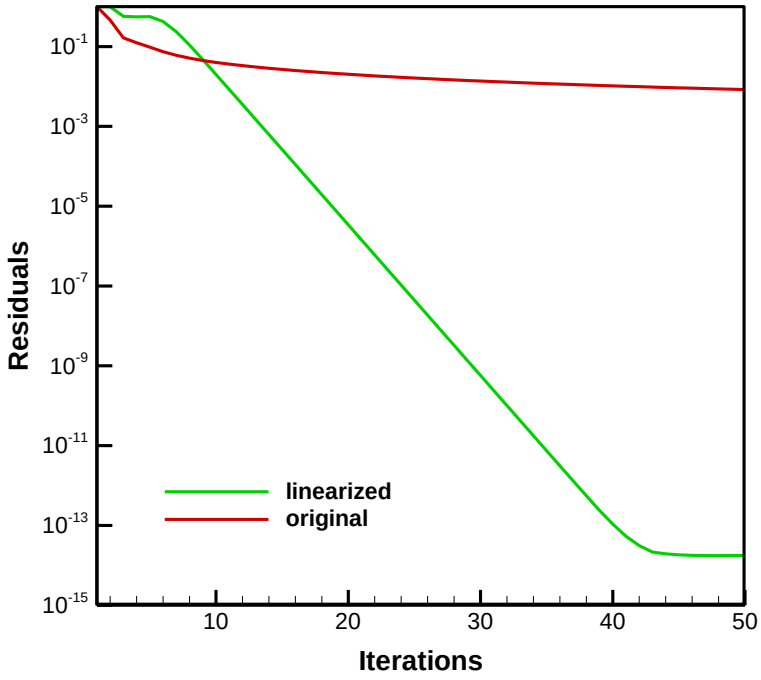
$$T_b = f \frac{hT_a + \varepsilon q_{in} + 3n^2 \varepsilon \sigma (T_b^*)^4}{h + 4n^2 \varepsilon \sigma (T_b^*)^3} + (1 - f)T_P \,, \qquad f = \frac{h + 4n^2 \varepsilon \sigma (T_b^*)^3}{h + \frac{k_P}{\Delta x} + 4n^2 \varepsilon \sigma (T_b^*)^3}$$

Radiative Heat Flux 선형화 효과

- 수렴성 비교



Convergence of Energy Equation

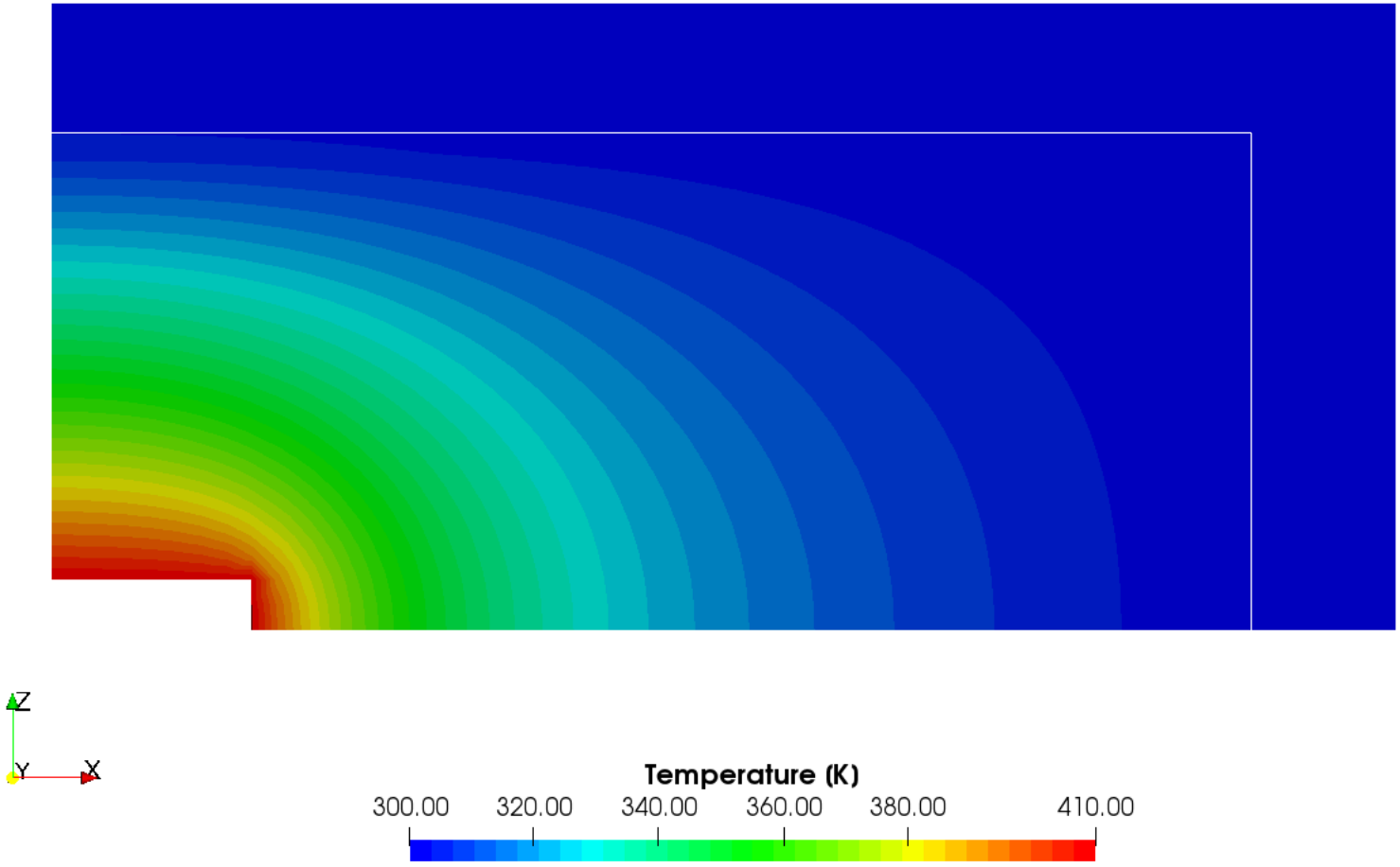


Convergence of Intensity Equation

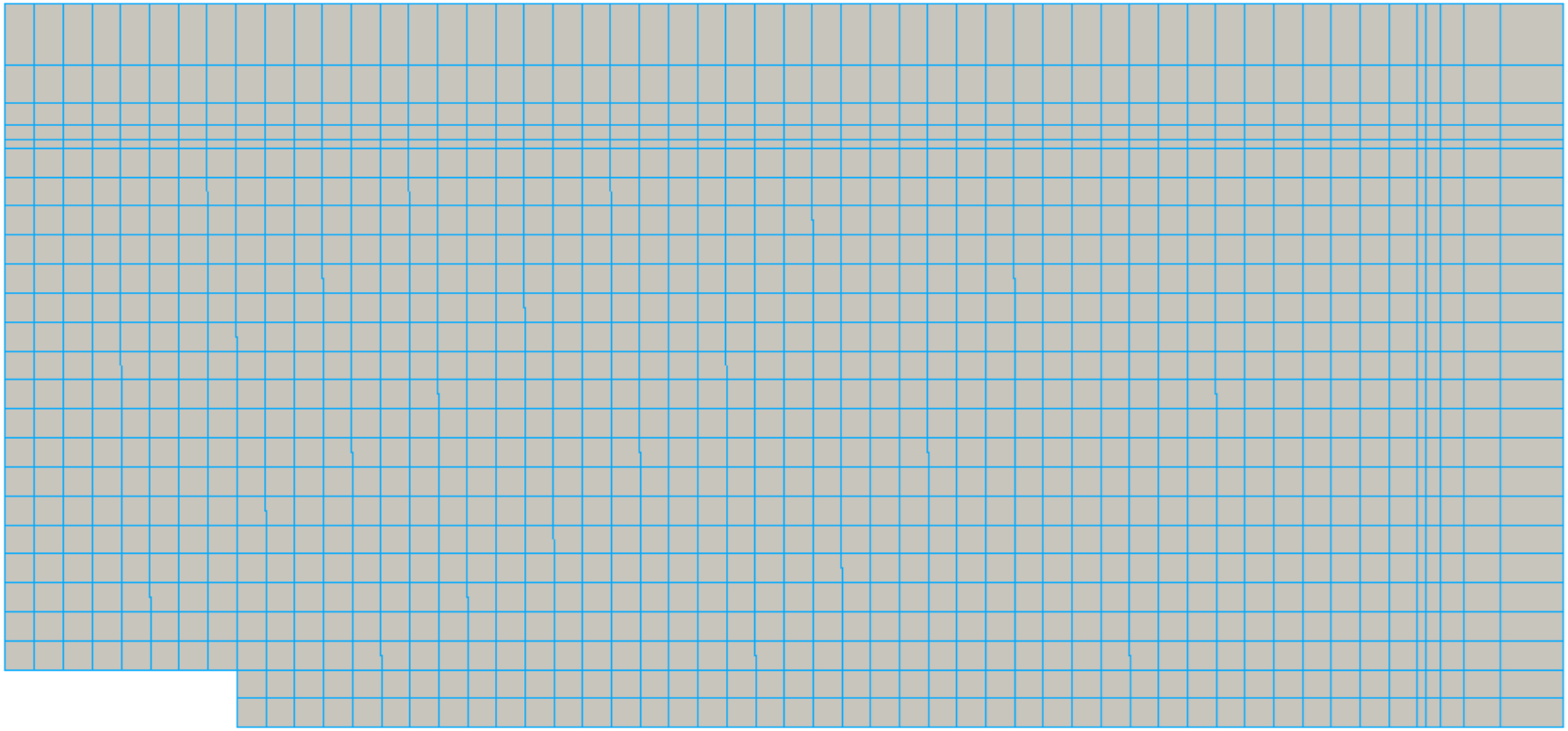
CHT Solver의 에너지 수렴성

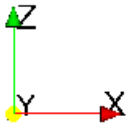
단순 열전도 문제

- 물성이 다른 두개의 Solid

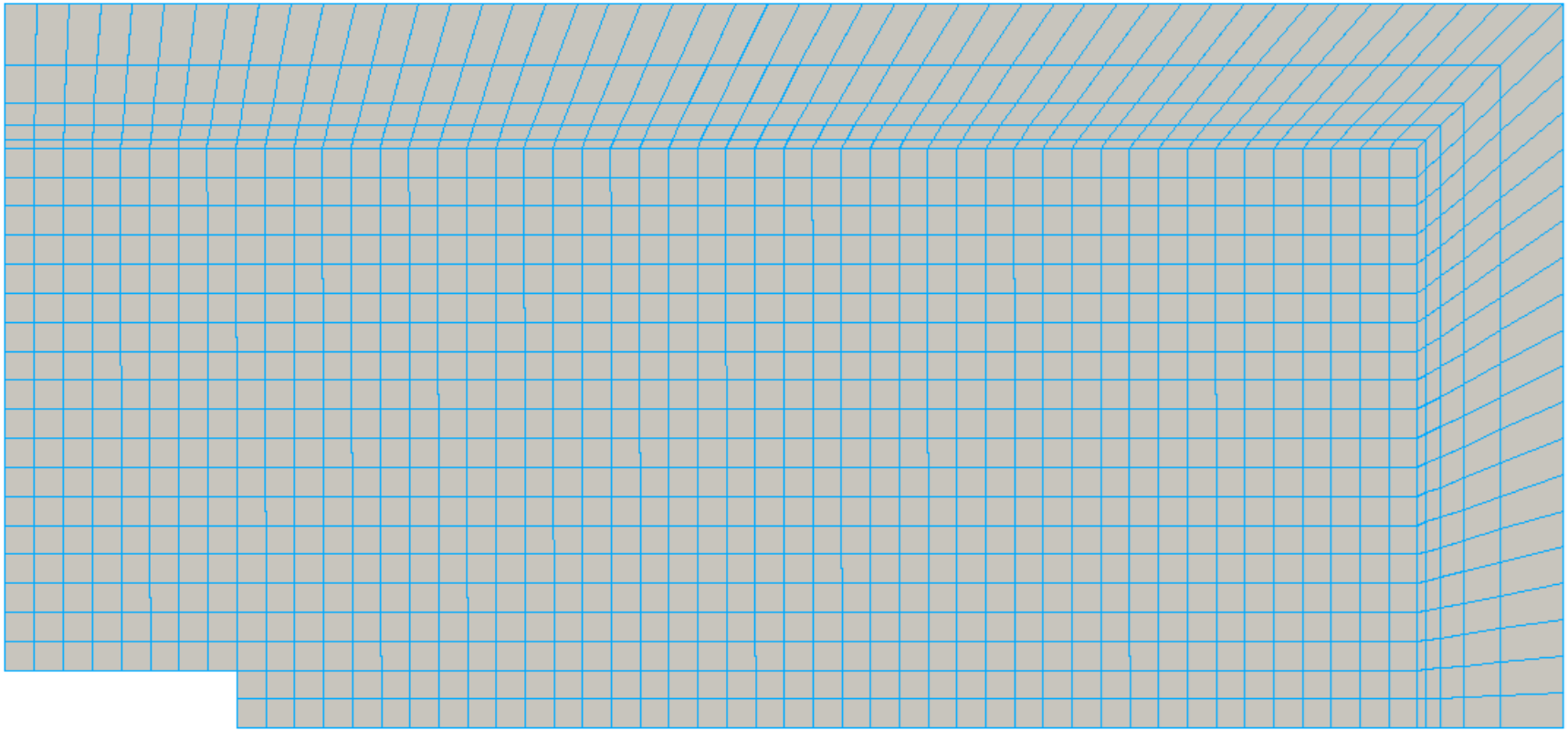


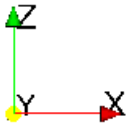
GRID1



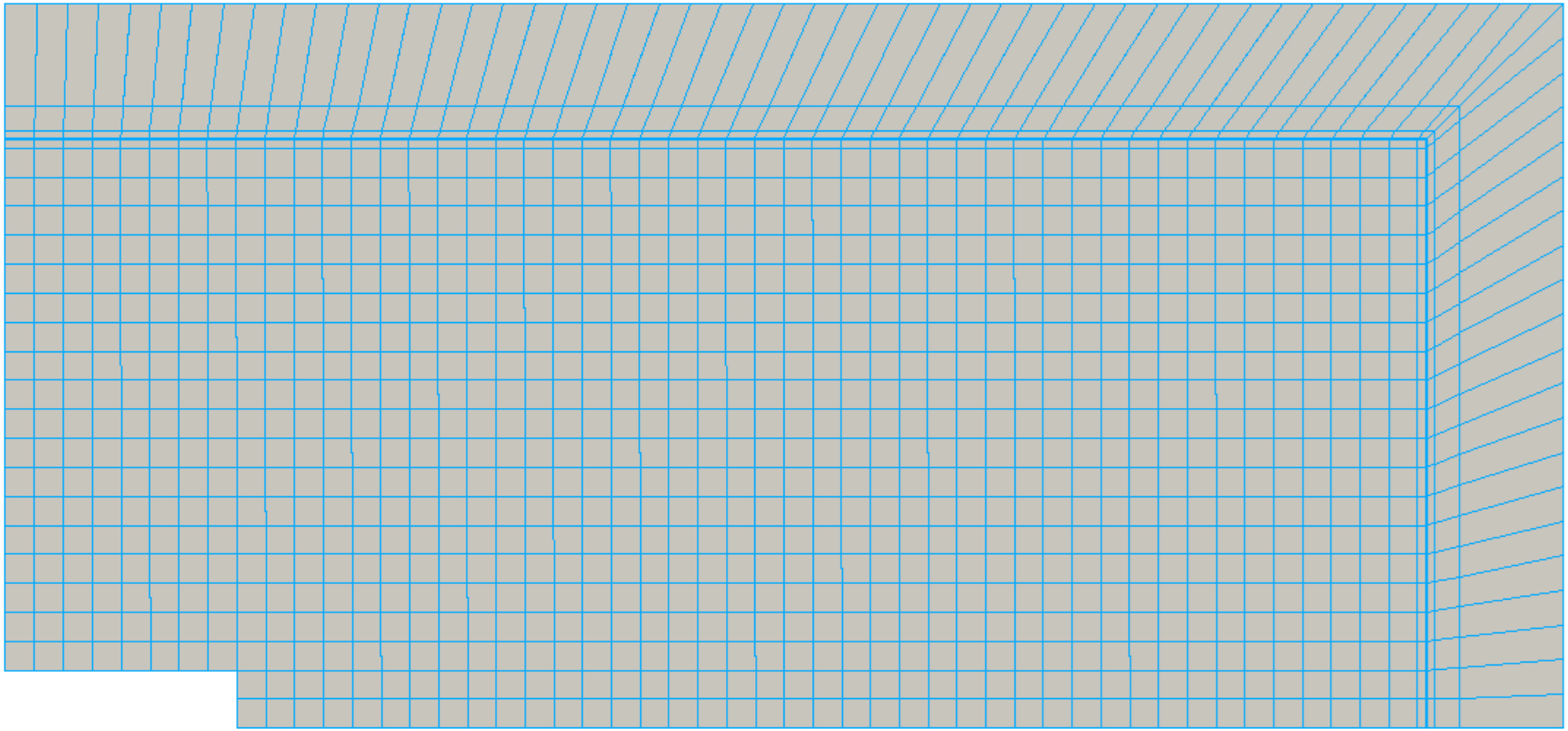


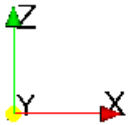
GRID2



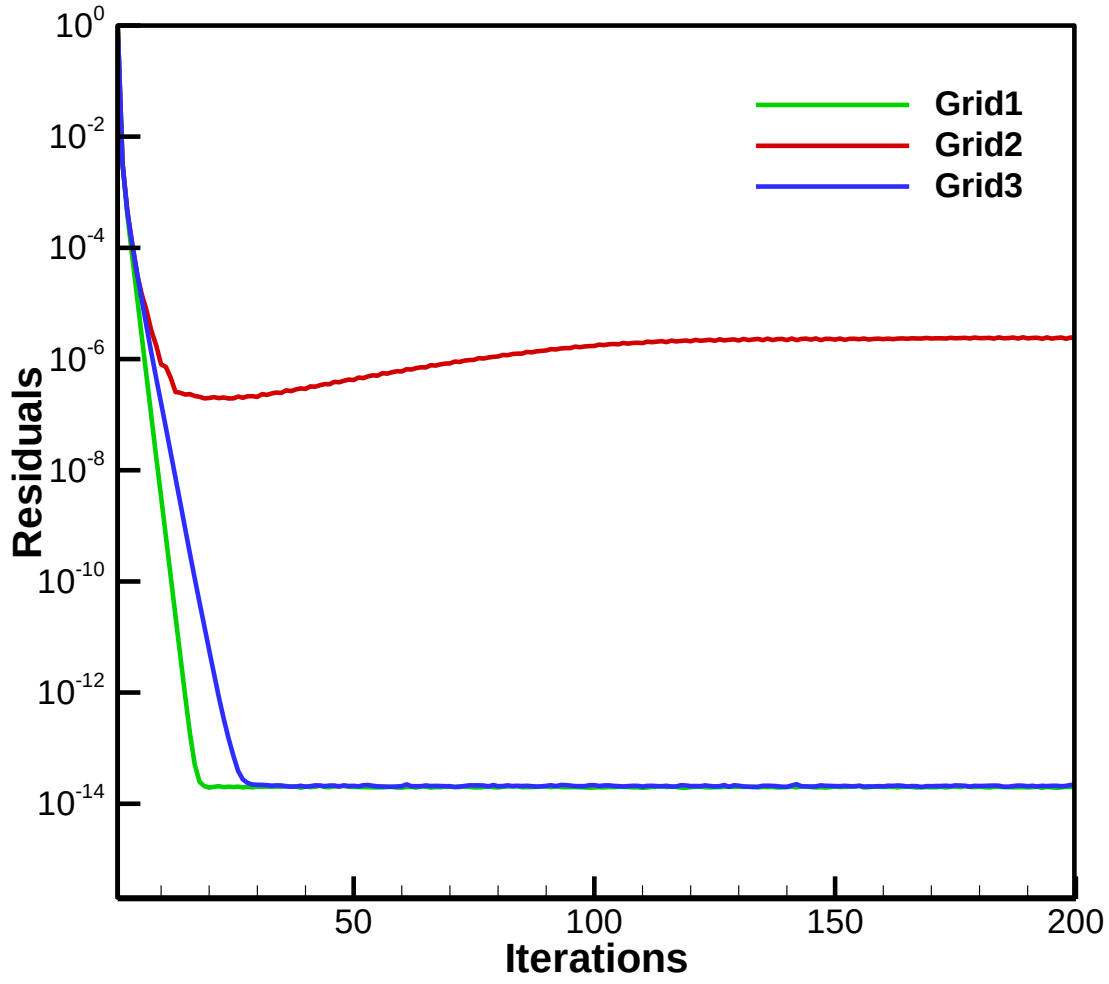


GRID3





격자에 따른 수렴성


Grid1
Grid2
Grid3
Residuals
Iterations
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
50
100
150
200

non-orthogonal correction

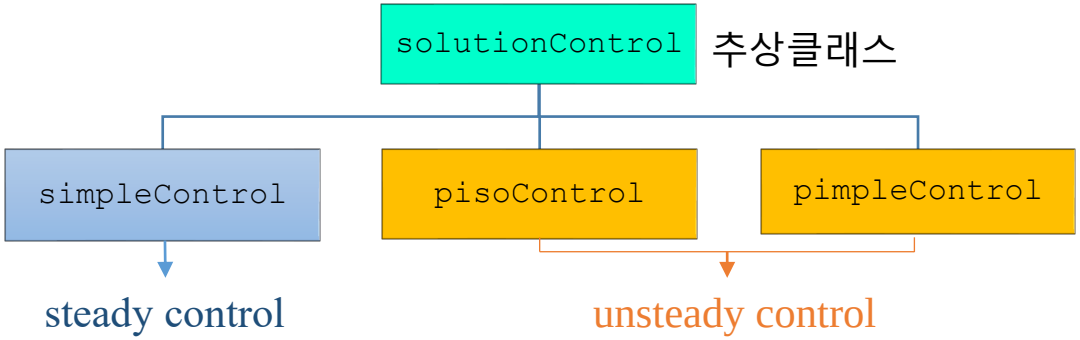
- mappedWall은 region 경계면

- non-orthgonal correction이 적용되는 경계면은?
  - processor boundary
  - cyclic boundary

Multi-region 솔버의 수렴 판정 기능

Multi-region 솔버의 수렴 판정 기능

- OpenFOAM의 loop control libraries
  - 클래스 계층 구조



  - 각 솔버마다 해당 클래스 타입의 객체를 생성하여 loop control
    - `buoyantSimpleFoam` => `simpleControl`
      - check max iterations, non-orthogonal correction loop, convergence
    - `buoyantPimpleFoam` => `pimpleControl`
      - check max inner-iterations, pressure correction loop, non-orthogonal correction loop, convergence of inner-iterations
    - `chtMultiRegionFoam` => ???
      - `simpleControl`이나 `pimpleControl` 클래스는 multi-region을 지원하지 않음
      - 최신 버전 OpenFOAM에서는 솔버의 소스코드에 하드코딩으로 구현되어 있지만 `simpleControl`이나 `pimpleControl`클래스의 control 알고리즘과 다름

Multi-region 솔버의 수렴 판정 기능

- Multi-region loop control libraries 개발
  - 두개의 독립적인 클래스로 개발
    - `solutionControl` 클래스와 상속관계가 아님
  - 클래스
    - **`multiRegionSimpleControl`**
    - **`multiRegionPimpleControl`**
  - 생성자 함수에 각 region에 대한 `fvMesh` 객체의 포인터 리스트를 전달

```
// Constructors

        //- Construct from mesh and the name of control sub-dictionary
        multiRegionPimpleControl
        (
            const Time& runTime,
            const PtrList<fvMesh>& fluid,
            const PtrList<fvMesh>& solid
        );
```

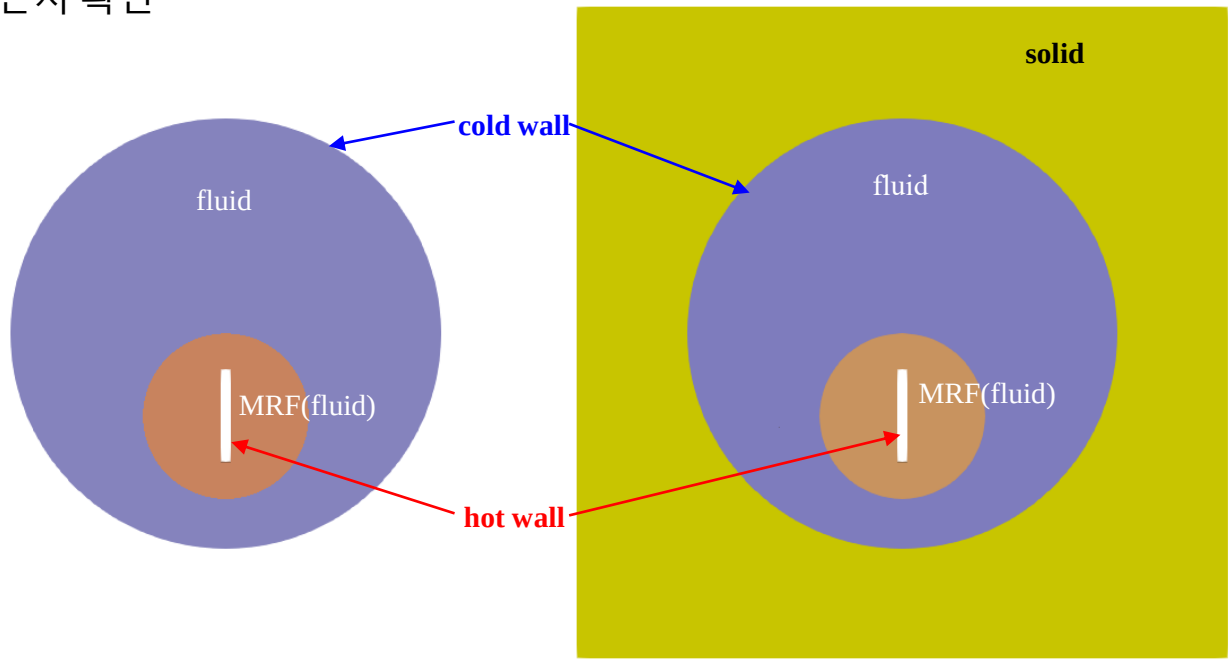
Multi-region 솔버의 수렴 판정 기능

- Multi-region loop control libraries 개발
  - 멤버함수에서는 생성자에서 전달받은 모든 region에 대해서 looping 지속 여부를 판단

```
bool Foam::NEXT::multiRegionPimpleControl::criteriaSatisfied()
{
    bool checkedAndAchieved = true;

    forAll(regions_, regionI) // 모든 region에 대한 looping
    {
        const fvMesh* mesh(regions_[regionI]);
        List<fieldData>& residualControl(residualControls_[regionI]);

        // no checks on first iteration - nothing has been calculated yet
        if ((corr_ == 1) || residualControl.empty() || finalIter())
        {
            return false;
        }

        bool storeIni = this->storeInitialResiduals();

        ...
```

Multi-region 솔버의 수렴 판정 기능

- Multi-region loop control 기능 검증 #1
  - single-region 솔버와 multi-region 솔버로 같은 문제를 해석
    - MRF 영역을 포함한 자연대류 문제
    - Multi-region 솔버에서 solid region을 풀지 않도록 설정하고 single-region 솔버와 같은 거동을 하는지 확인



Multi-region 솔버의 수렴 판정 기능

- Multi-region loop control 기능 검증 #1
  - 1초까지 해석후 residual 비교

```
PIMPLE: iteration 5
DILUPBiCGStab:  Solving for Ux, Initial residual = 0.0001018948155195254,
Final residual = 1.931787992241496e-07, No Iterations 1
DILUPBiCGStab:  Solving for Uy, Initial residual = 5.577468656968949e-05,
Final residual = 8.975777426222258e-08, No Iterations 1
GAMGPCG:  Solving for p_rgh, Initial residual = 0.02298659441267836, Final
residual = 0.0008887110289223589, No Iterations 1
GAMGPCG:  Solving for p_rgh, Initial residual = 0.00108468265244804, Final
residual = 7.691586762727594e-05, No Iterations 1
DILUPBiCGStab:  Solving for epsilon, Initial residual = 1.17761647989896e-
05, Final residual = 2.908488217663309e-07, No Iterations 1
DILUPBiCGStab:  Solving for k, Initial residual = 6.806336675700609e-06,
Final residual = 2.074380896992685e-07, No Iterations 1
diagonal:  Solving for rho, Initial residual = 0, Final residual = 0, No
Iterations 0
GAMGPBiCGStab:  Solving for h, Initial residual = 6.143298845904801e-06,
Final residual = 3.478449934155415e-12, No Iterations 1
Min/max T:350 373
time step continuity errors : sum local = 0, global = 0, cumulative = 0
PIMPLE: converged in 5 iterations
ExecutionTime = 95.97 s  ClockTime = 96 s

End
```

buoyantPimpleNFoam

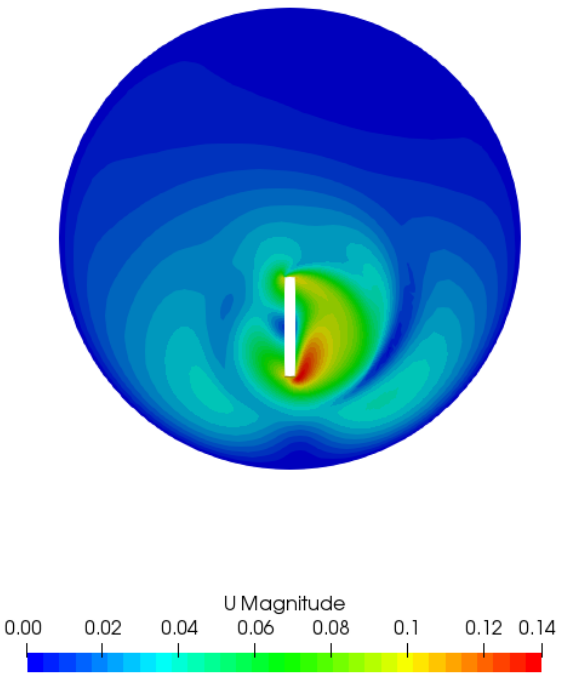
Inner-iteration converged

```
PIMPLE: iteration 5

Solving for fluid region fluid
DILUPBiCGStab:  Solving for Ux, Initial residual = 0.0001018948155195254,
Final residual = 1.931787992241496e-07, No Iterations 1
DILUPBiCGStab:  Solving for Uy, Initial residual = 5.577468656968949e-05,
Final residual = 8.975777426222258e-08, No Iterations 1
GAMGPCG:  Solving for p_rgh, Initial residual = 0.02298659441267836, Final
residual = 0.0008887110289223589, No Iterations 1
GAMGPCG:  Solving for p_rgh, Initial residual = 0.00108468265244804, Final
residual = 7.691586762727594e-05, No Iterations 1
DILUPBiCGStab:  Solving for epsilon, Initial residual = 1.17761647989896e-
05, Final residual = 2.908488217663309e-07, No Iterations 1
DILUPBiCGStab:  Solving for k, Initial residual = 6.806336675700609e-06,
Final residual = 2.074380896992685e-07, No Iterations 1
diagonal:  Solving for rho, Initial residual = 0, Final residual = 0, No
Iterations 0
GAMGPBiCGStab:  Solving for h, Initial residual = 6.143298845904801e-06,
Final residual = 3.478449934155415e-12, No Iterations 1
Min/max T:350 373
time step continuity errors : sum local = 0, global = 0, cumulative = 0
PIMPLE: converged in 5 iterations
ExecutionTime = 99.29000000000001 s  ClockTime = 99 s

End
```

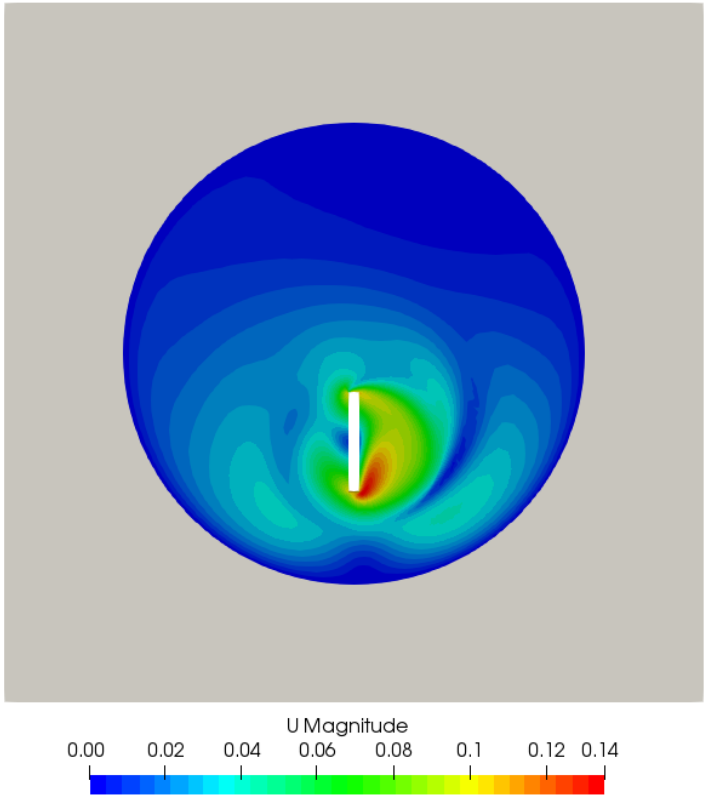
chtMultiRegionPimpleNFoam

Multi-region 솔버의 수렴 판정 기능

- Multi-region loop control 기능 검증 #1
  - 해석 결과 : 1초 후 속도 분포



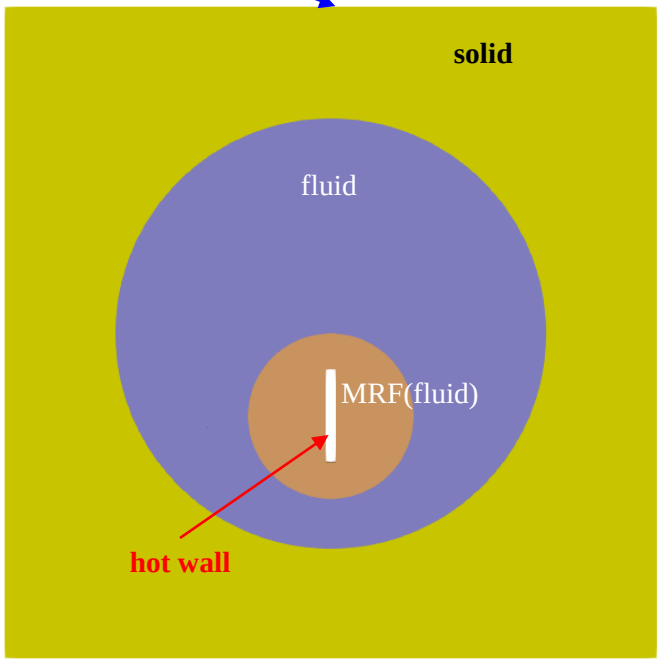
`buoyantPimpleNFoam`



`chtMultiRegionPimpleNFoam`

Multi-region 솔버의 수렴 판정 기능

- Multi-region loop control 기능 검증 #2
  - 검증 #1과 같은 문제를 solid 영역을 포함하여 multi-region 솔버로 해석
    - inner-iteration의 convergence criterion에 따라 **multi-region loop control**이 잘 작동하는지 확인
      - pressure : 1e-3
      - momentum : 1e-3
      - turbulence : 1e-3
      - energy : 1e-6



`chtMultiRegionPimpleNFoam`

Multi-region 솔버의 수렴 판정 기능

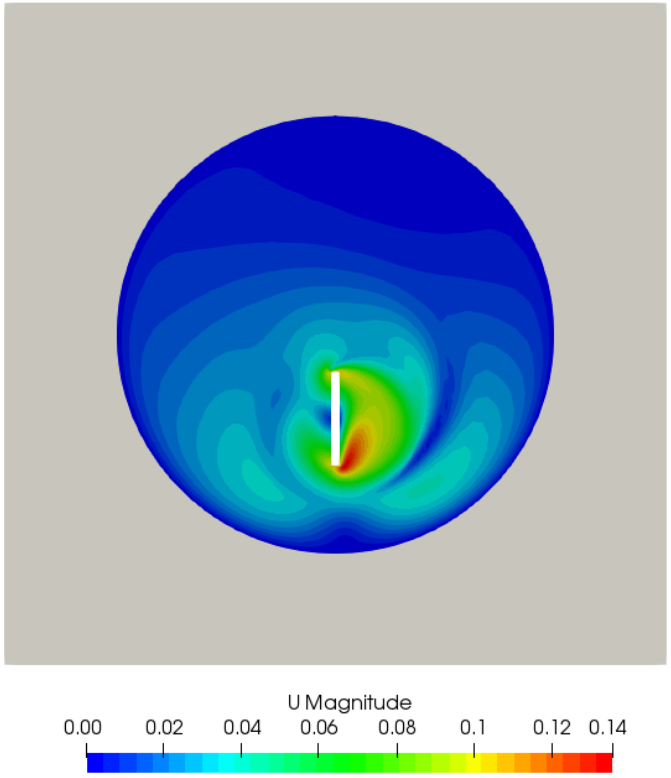
- Multi-region loop control 기능 검증 #2
  - 1초까지 해석후 residual 확인

```
PIMPLE: iteration 8

Solving for fluid region fluid
DILUPBiCGStab:  Solving for Ux, Initial residual = 3.478702854484385e-05, Final residual = 7.09742724106769e-08, No Iterations 1
DILUPBiCGStab:  Solving for Uy, Initial residual = 1.969297275762624e-05, Final residual = 2.781589205683131e-08, No Iterations 1
GAMGPCG:  Solving for p_rgh, Initial residual = 0.01156233594243819, Final residual = 0.0003010856380198718, No Iterations 1
GAMGPCG:  Solving for p_rgh, Initial residual = 0.0004480591768750711, Final residual = 3.495684054473972e-05, No Iterations 1
DILUPBiCGStab:  Solving for epsilon, Initial residual = 2.574343607527129e-06, Final residual = 7.055973106906398e-08, No Iterations 1
DILUPBiCGStab:  Solving for k, Initial residual = 1.320877134103976e-06, Final residual = 4.425784430245165e-08, No Iterations 1
diagonal:  Solving for rho, Initial residual = 0, Final residual = 0, No Iterations 0
GAMGPBiCGStab:  Solving for h, Initial residual = 9.071476042475182e-07, Final residual = 4.887913235950798e-13, No Iterations 1

Solving for solid region solid
GAMGPBiCGStab:  Solving for h, Initial residual = 7.478312571284788e-10, Final residual = 2.71954188417007e-26, No Iterations 1
PIMPLE: iteration 9

Solving for fluid region fluid
DILUPBiCGStab:  Solving for Ux, Initial residual = 2.453929304335645e-05, Final residual = 5.133687330153424e-08, No Iterations 1
DILUPBiCGStab:  Solving for Uy, Initial residual = 1.429085296256882e-05, Final residual = 1.944790763298906e-08, No Iterations 1
GAMGPCG:  Solving for p_rgh, Initial residual
GAMGPCG:  Solving for p_rgh, Initial residual                                                    1
DILUPBiCGStab:  Solving for epsilon, Initial                                                    erations 1
DILUPBiCGStab:  Solving for k, Initial residu                                                   s 1
diagonal:  Solving for rho, Initial residual
GAMGPBiCGStab:  Solving for h, Initial residu                                                   s 1

Solving for solid region solid
GAMGPBiCGStab:  Solving for h, Initial residua                                                  ons 1
Min/max T:313.2765107569926 350.0907069840727
PIMPLE: converged in 9 iterations
```

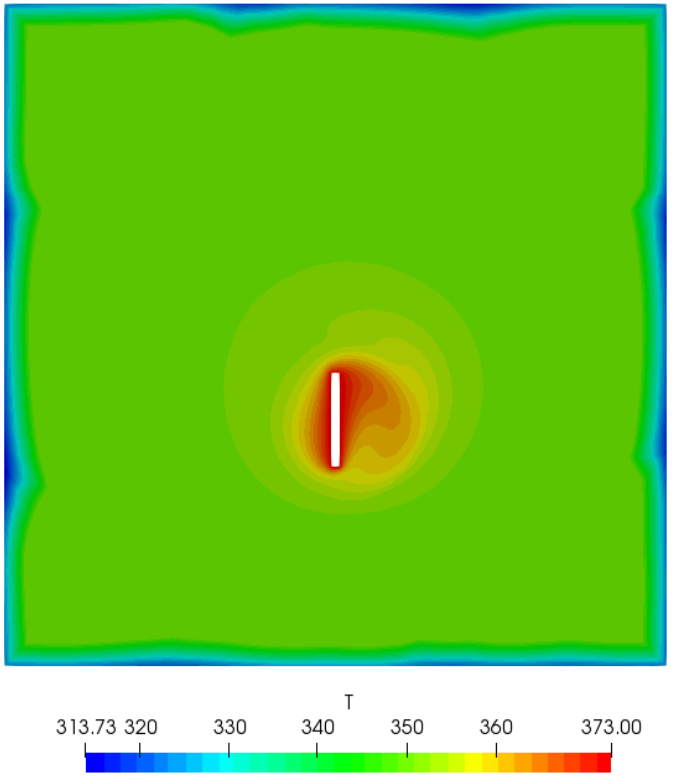
fluid region의 energy equation이 마지막으로 criterion을 만족하면서 8번째 inner-iteration에서 수렴되었지만 time accuracy를 위해 pressure under-relaxation 없이 한번 더 iteration

Multi-region 솔버의 수렴 판정 기능

- Multi-region loop control 기능 검증 #2
  - 해석 결과 : 1초 후 속도 및 온도 분포



속도 분포



온도 분포