NEXTfoam
open source CFD consulting

초공동 수중체의 제어판 전개 해석 방법

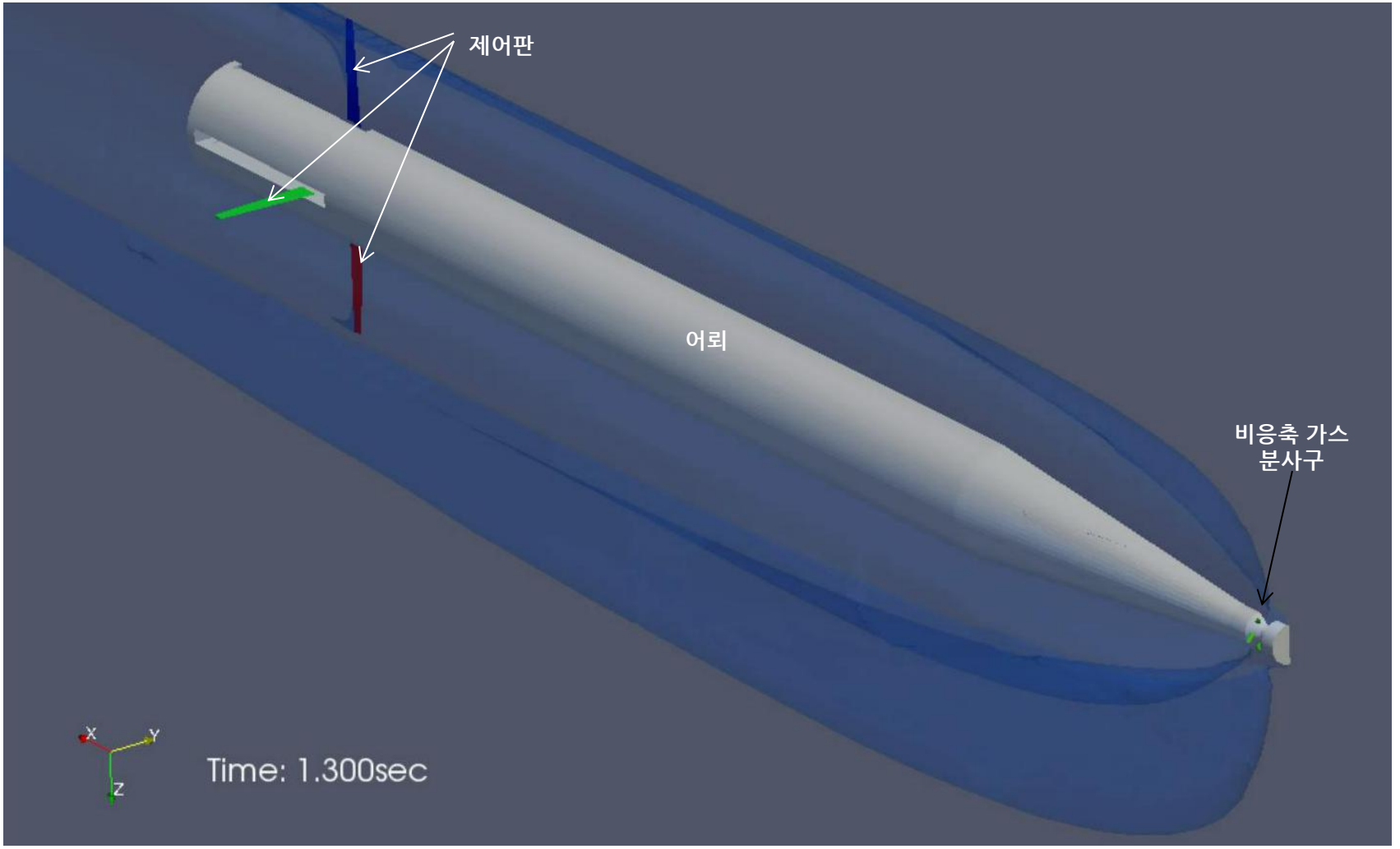
2017.09.22.

OpenFOAM Korea Users' Community Conference(제6회, 대전)

이상돈(넥스트폼 기술연구소)

NEXTfoam
open source CFD consulting

해석 대상



해석 방법 - solver set up

- 해석 모델
  - interPhaseChangeDyMFoam 기반(OpenFOAM 2.4.0)
  - Multiphase Model

    -. VOF(non-condensible gas 포함)

    -. Cavitation Model : Kunz Model

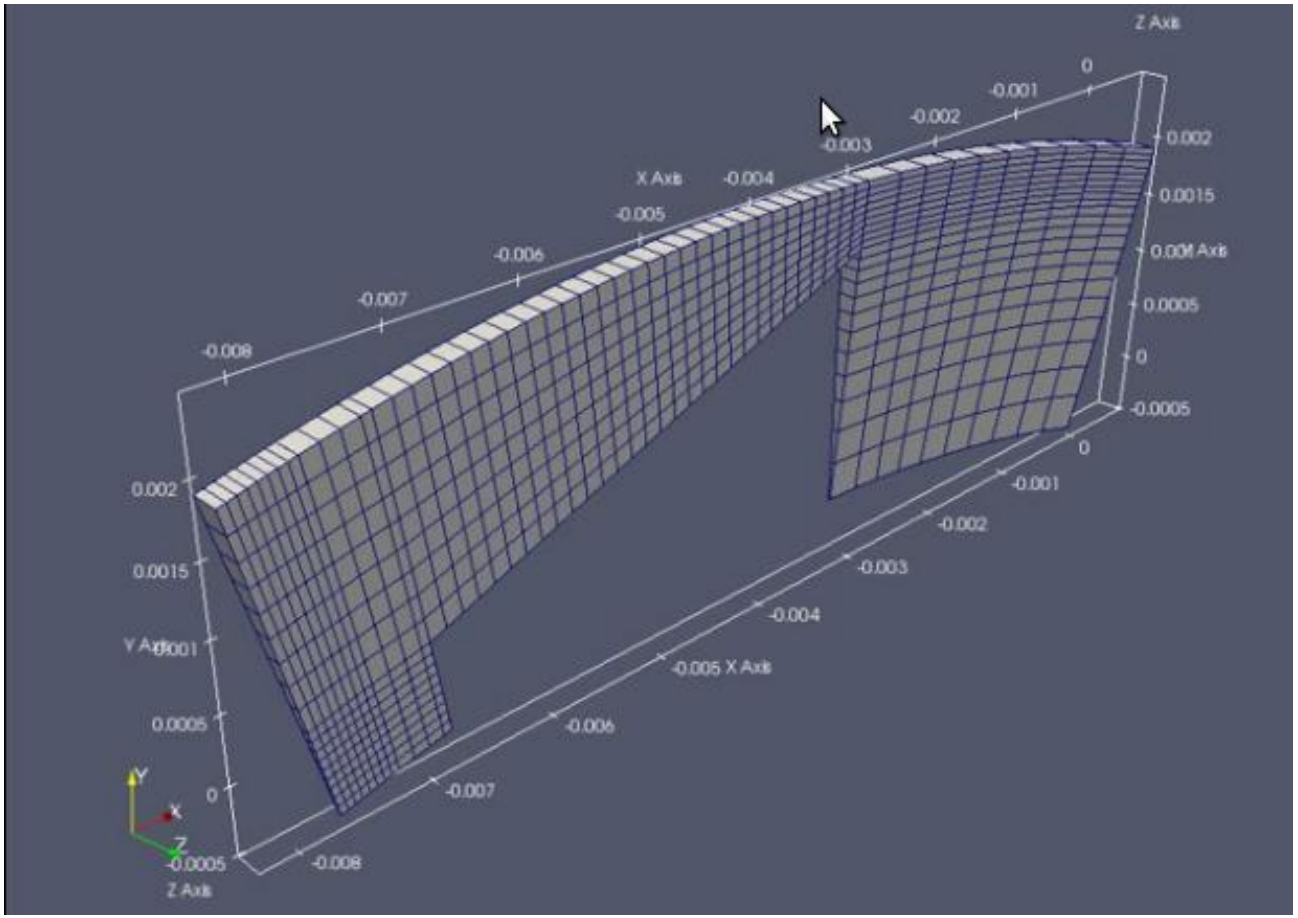
- Phase transport equation

$$\frac{\partial \alpha_l}{\partial t} + \underline{u} \cdot \nabla \alpha_l = \left(\dot{m}^+ + \dot{m}^-\right)\frac{1}{\rho_l} \qquad \frac{\partial \alpha_{ng}}{\partial t} + \underline{u} \cdot \nabla \alpha_{ng} = 0$$

$$\dot{m}^+ = \frac{C_{prod}\rho_l(\alpha_l - \alpha_{ng})^2(1 - \alpha_l - \alpha_{ng})}{t_\infty} \qquad \dot{m}^- = \frac{C_{dest}\rho_l(\alpha_l - \alpha_{ng})MIN[0, p - p_v]}{(1/2\rho_l U_\infty^2)t_\infty}$$

해석 방법 - solver set up

- dynamicMesh
  - layering 기법 적용
  - foam-extend-3.2 version의 multiTopoBodyFvMesh library 활용



해석 방법 - solver set up

수정전(topoBody.C)

```
Foam::topoBody::topoBody
(
    const word& name,
    const polyMesh& mesh,
    const dictionary& dict
)
:
    name_(name),
    mesh_(mesh),
    movingCellsName_(dict.lookup("movingCells")),
    layerFacesNames_(dict.lookup("layerFaces")),
    minThickness_(readScalar(dict.lookup("minThickness"))),
    maxThickness_(readScalar(dict.lookup("maxThickness"))),
    SBMFPtr_(solidBodyMotionFunction::New(dict, mesh.time())),
    invertMotionMask_
    (
        dict.lookupOrDefault<bool>("invertMotionMask", false)
    ),
    movingPointsMaskPtr_(NULL)
{
+--  4 줄: Info<< "Moving body " << name << ":" << nl-----------------------------------------
        << endl;
}


// * * * * * * * * * * * * * * * * * * Destructor  * * * * * * * * * * * * * * * * * //

Foam::topoBody::~topoBody()
{
    clearPointMask();
}


// * * * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //

Foam::tmp<Foam::vectorField> Foam::topoBody::pointMotion() const
{
    // Rotational speed needs to be converted from rpm
    scalarField mpm = movingPointsMask();

    if (invertMotionMask_)
    {
        Info << "Inverting motion mask" << endl;
        mpm = 1 - mpm;
    }

    return mpm*transform(SBMFPtr_().velocity(), mesh_.allPoints())*
        mesh_.time().deltaT().value();
}
```

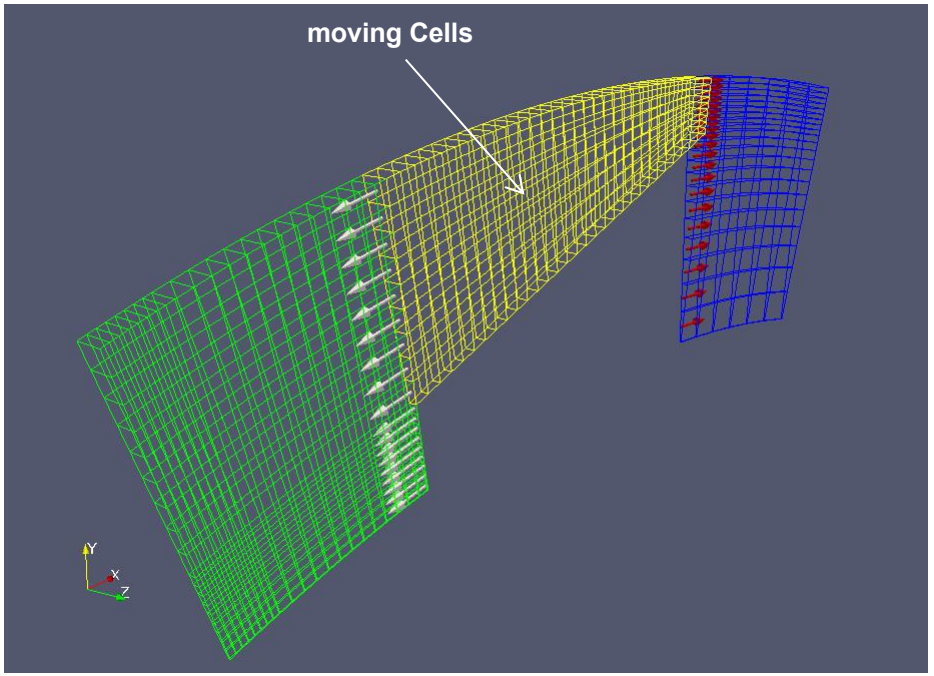
수정후(topoBodyLayer.C)

```
Foam::topoBodyLayer::topoBodyLayer
(
    const word& name,
    const polyMesh& mesh,
    const dictionary& dict
)
:
    name_(name),
    mesh_(mesh),
    movingCellsName_(dict.lookup("movingCells")),
    layerFacesNames_(dict.lookup("layerFaces")),
    minThickness_(readScalar(dict.lookup("minThickness"))),
    maxThickness_(readScalar(dict.lookup("maxThickness"))),
    SBMFPtr_(SBMotionFunction::New(dict, mesh.time())),
    invertMotionMask_
    (
        dict.lookupOrDefault<bool>("invertMotionMask", false)
    ),
    movingPointsMaskPtr_(NULL)
{
+--  4 줄: Info<< "Moving body " << name << ":" << nl-----------------------------------------
        << endl;
}


// * * * * * * * * * * * * * * * * * * Destructor  * * * * * * * * * * * * * * * * * //

Foam::topoBodyLayer::~topoBodyLayer()
{
    clearPointMask();
}


// * * * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //

Foam::tmp<Foam::vectorField> Foam::topoBodyLayer::pointMotion(pointField& newPoints) const
{
    // Rotational speed needs to be converted from rpm
    scalarField mpm = movingPointsMask();

    if (invertMotionMask_)
    {
        Info << "Inverting motion mask" << endl;
        mpm = 1 - mpm;
    }

////    return mpm*transform(SBMFPtr_().velocity(), mesh_.points())*
////        mesh_.time().deltaT().value();
    return mpm*(transform(SBMFPtr_().newPosition(), newPoints) - newPoints);
}
```

해석 방법 - solver set up

- interface Zone 설정

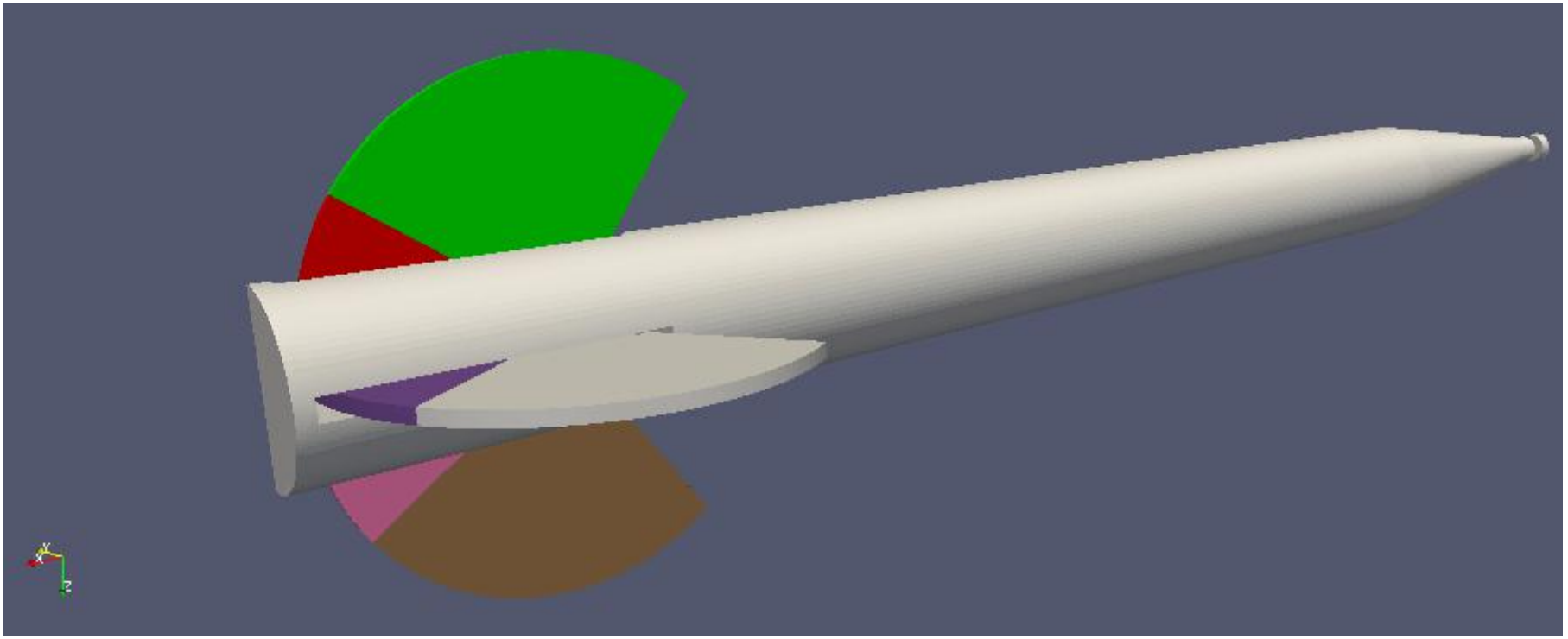
  -. setSet utility 활용(batch script 작성)




```
dynamicFvMesh      multiTopoBodyLayerFvMesh;

multiTopoBodyLayerFvMeshCoeffs
{
    bodies
    (
        body
        {
            movingCells     moving;
            layerFaces ( rightZone leftZone );

            SBMotionFunction   circularMotion;

            circularMotionCoeffs
            {
                origin   (-0.00375 -0.015 0);
                  axis    (0 0 1 );
                omega   constant -0.01;
            }

            minThickness       1e-4;
            maxThickness       2e-4;
        }
    );
}
```

해석 방법 - mesh set up

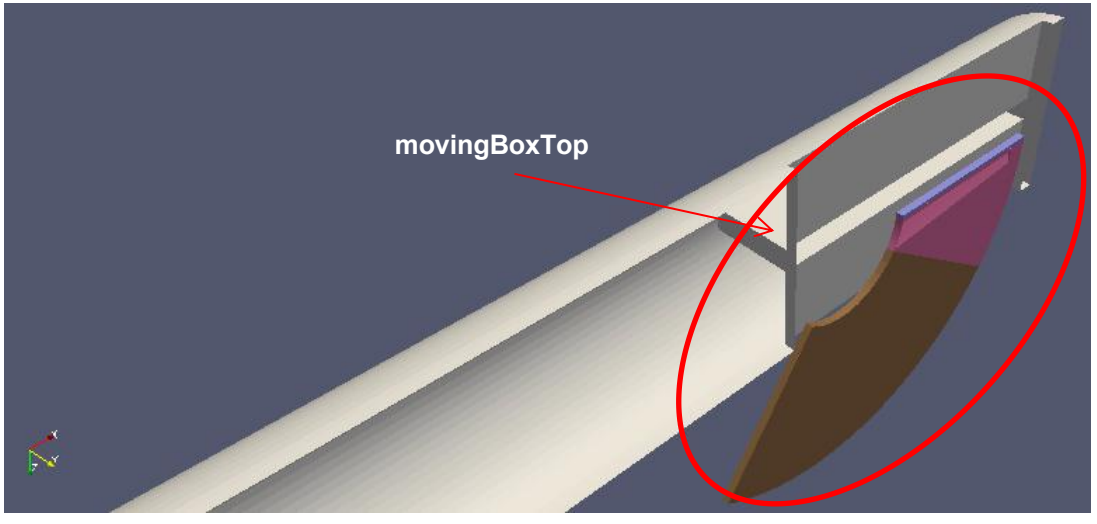
- 형상 모델링 및 제어판 격자 생성 : SALOME 활용
  - export to FOAM : salomeToOpenFOAM(python utility)
- AMI interface 설정
  - 제어 판별로 layering 적용 영역 구분



해석 방법 - mesh set up

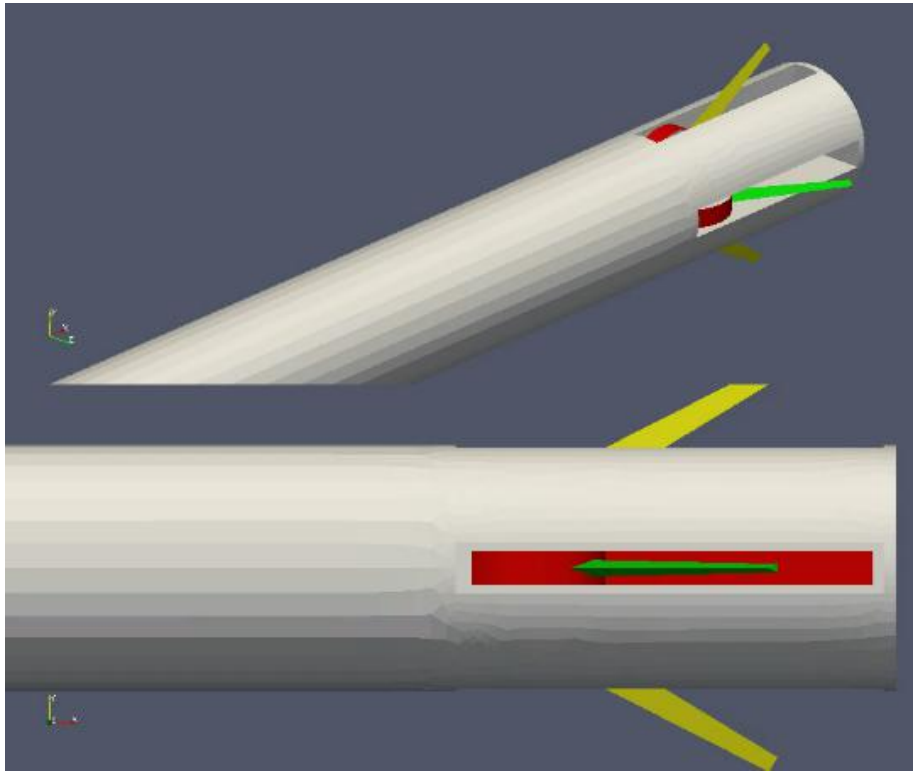
- Parallel setting
  - layering 적용 영역은 동일 core로 지정

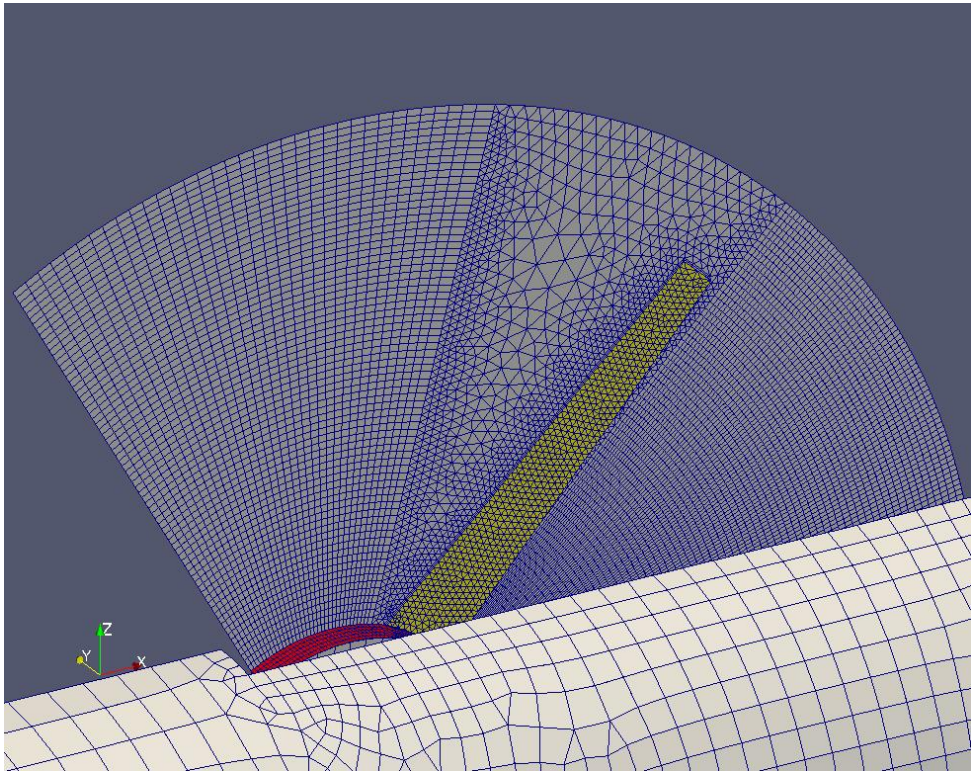
```
method          scotch;

//- Keep owner and neighbour on same processor for faces in zones:
//(makes sense only for cyclic patches)
preserveFaceZones ( leftZoneUW rightZoneUW rightZoneLW leftZoneLW rightZoneTW leftZoneTW);

//- Keep owner and neighbour on same processor for faces in patches:
//  (makes sense only for cyclic patches)
//preservePatches (cyclic_half0 cyclic_half1);
//- Keep all of faceSet on a single processor. This puts all cells
//  connected with a point, edge or face on the same processor.
//  (just having face connected cells might not guarantee a balanced decomposition)
// The processor can be -1 (the decompositionMethod chooses the processor
// for a good load balance) or explicitly provided (upsets balance).
singleProcessorFaceSets ((movingBoxTop -1) (movingBoxUnder -1) (movingBoxLeft -1));
```



해석 결과

제어판 전개 동작





해석 결과

분사 공동 해석 적용