OpenFOAM
マクロを利用した設定の簡易化

2014年2月8日

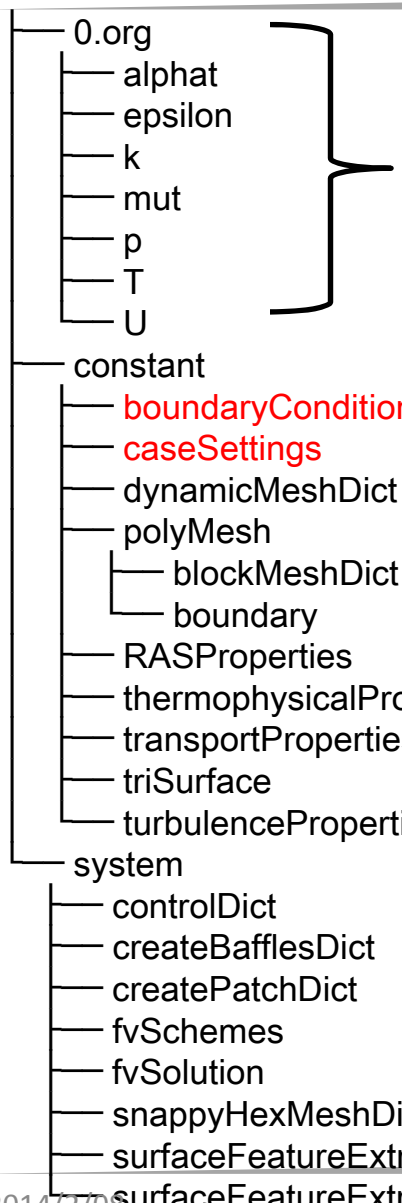
富山県立大学　中川慎二

- OpenFOAM 2.2 以降
- 設定ファイル内でのマクロ機能が強化された
- スコープを制御可能

- 有効に活用することで、情報を集約して、設定が容易となる

- compressible/rhoPimpleDyMFoam/annularThermalMixer 例題に使用例がある

annularThermalMixer例題 ファイル構成



初期条件や境界条件が、変数ごとに分散して記述されている

↓

caseSetting と boundaryConditions ファイルに記述し，そこから読み込むようにしてある

p

```
#include "${FOAM_CASE}/constant/caseSettings"

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 1e5;

boundaryField
{
    inlet       { $:inlet.p; }
    outlet      { $:outlet.p; }
    staticWalls { $:wall.p; }
    movingWalls { $staticWalls; }

    #include "${WM_PROJECT_DIR}/etc/caseDicts/setConstraintTypes"
}
```

caseSetting ファイルの内容がここに読み込まれる。
さらにその中から boundaryConditions ファイルの内容が読み込まれる。

boundaryConditionsに記述されている。
（コロン付→トップレベル）

$ マクロの目印
: (コロン) このディクショナリのトップレベル
. (ドット) アクセスエレメント

1行上のstaticWallsと同じ（コロンなし→同一スコープ内の情報）

個別の値指定が不要な一般的な境界条件：後述

T

```
#include "${FOAM_CASE}/constant/caseSettings"

dimensions      [0 0 0 1 0 0 0];

internalField   uniform $:outerInlet.T;

boundaryField
{
    innerInlet
    {
        type            fixedValue;
        value           uniform $:innerInlet.T;
    }

    outerInlet
    {
        type            fixedValue;
        value           uniform $:outerInlet.T;
    }

    outlet      { $:outlet.T; }
    staticWalls { $:wall.T; }
    movingWalls { $staticWalls; }

    #include "${WM_PROJECT_DIR}/etc/caseDicts/setConstraintTypes"
}
```

constant/boundary

```
12
(
    innerInlet
    {
        type            patch;
        inGroups        1(inlet);
        nFaces          544;
        startFace       244948;
    }
    outerInlet
    {
        type            patch;
        inGroups        1(inlet);
        nFaces          1404;
        startFace       245492;
    }
    innerOutlet
    {
        type            patch;
        inGroups        1(outlet);
        nFaces          544;
        startFace       246896;
    }
    outerOutlet
    {
        type            patch;
        inGroups        1(outlet);
        nFaces          1404;
        startFace       247440;
    }
    rotorBlades
    {
        type            wall;
        inGroups        1(movingWalls);
        nFaces          540;
        startFace       248844;
    }
    rotorBlades_slave
    {
        type            wall;
        inGroups        1(movingWalls);
        nFaces          540;
        startFace       249384;
    }
    shaft
    {
        type            wall;
        inGroups        1(movingWalls);
        nFaces          1052;
        startFace       249924;
    }
    statorBlades
    {
        type            wall;
        inGroups        1(staticWalls);
        nFaces          2128;
        startFace       250976;
    }
    statorBlades_slave
    {
        type            wall;
        inGroups        1(staticWalls);
        nFaces          2128;
        startFace       253104;
    }
    wall
    {
        type            wall;
        inGroups        1(staticWalls);
        nFaces          6165;
        startFace       255232;
    }
    AMI1
    {
        type            cyclicAMI;
        inGroups        1(cyclicAMI);
        nFaces          10944;
        startFace       261397;
        matchTolerance  0.0001;
        transform       noOrdering;
        neighbourPatch  AMI2;
    }
    AMI2
    {
        type            cyclicAMI;
        inGroups        1(cyclicAMI);
        nFaces          10944;
        startFace       272341;
        matchTolerance  0.0001;
        transform       noOrdering;
        neighbourPatch  AMI1;
    }
)
```

constant/boundary

```
rotorBlades
{
    type          wall;
    inGroups       1(movingWalls);
    nFaces        540;
    startFace      248844;
}
rotorBlades_slave
{
    type          wall;
    inGroups       1(movingWalls);
    nFaces        540;
    startFace      249384;
}
shaft
{
    type          wall;
    inGroups       1(movingWalls);
    nFaces        1052;
    startFace      249924;
}
```

caseSettings

```
innerInlet
{
    U           (0 0 0.2);
    epsilon       5.70e-5;
    k           2.40e-5;
    T           233;
}

outerInlet
{
    U           (0 0 0.1);
    epsilon       3.98e-5;
    k           6.00e-6;
    T           293;
}

meshMotionProperties
{
    omega 25; // rad/s
}

#include "${FOAM_CASE}/constant/boundaryConditions"
```

boundaryConditions

```
calculated
{
    alphat
    {
        type        calculated;
        value       uniform 0;
    }
    mut
    {
        type        calculated;
        value       uniform 0;
    }
}

inlet
{
    p
    {
        type        zeroGradient;
    }
}

outlet
{
    p
    {
        type        totalPressure;
        value       uniform 1e5;
        p0          uniform 1e5;
        U           U;
        phi         phi;
        rho         rho;
        psi         none;
        gamma       1.4;
    }
    U
    {
        type        pressureInletOutletVelocity;
        value       uniform (0 0 0);
    }
    T
    {
        type        inletOutlet;
        inletValue  uniform $:outerInlet.T;
        value       $inletValue;
    }
    k
    {
        type        inletOutlet;
        inletValue  uniform $:innerInlet.k;
        value       $inletValue;
    }
    epsilon
    {
        type        inletOutlet;
        inletValue  uniform $:innerInlet.epsilon;
        value       $inletValue;
    }
}

wall
{
    p
    {
        type        zeroGradient;
    }
    U
    {
        type        fixedValue;
        value       uniform (0 0 0);
    }
    T
    {
        type        zeroGradient;
    }
    k
    {
        type        compressible::kqRWallFunction;
        value       uniform $:innerInlet.k;
    }
    epsilon
    {
        type        compressible::epsilonWallFunction;
        value       uniform $:innerInlet.epsilon;
    }
    mut
    {
        type        mutkWallFunction;
        value       uniform 0;
    }
    alphat
    {
        type        compressible::alphatWallFunction;
        Prt         0.85;
        value       uniform 0;
    }
}

movingWall
{
    U
    {
        type        movingWallVelocity;
        value       uniform (0 0 0);
    }
}
```

${WM_PROJECT_DIR}/etc/caseDicts/setConstraintTypes

```
cyclic
{
    type  cyclic;
}

cyclicAMI
{
    type  cyclicAMI;
}

cyclicSlip
{
    type  cyclicSlip;
}

empty
{
    type  empty;
}

nonuniformTransformCyclic
{
    type  nonuniformTransformCyclic;
}

processor
{
    type  processor;
}

processorCyclic
{
    type  processorCyclic;
}

symmetryPlane
{
    type  symmetryPlane;
}

wedge
{
    type  wedge;
}
```

setConstraintTypes

- 一般的な境界条件
- 個別の値指定が不要なもの

- Typeと同じ名前にすればよい

応用：cavity例題で使うとどうなる？

Cavity例題用 boudaryConditions

```
movingWall
{
    U
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }
    p
    {
        type            zeroGradient;
    }
}

fixedWalls
{
    U
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    p
    {
        type            zeroGradient;
    }
}

frontAndBack
{
    U
    {
        type            empty;
    }

    p
    {
        type            empty;
    }
}
```

U ファイルと p ファイルの一部

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
#include "${FOAM_CASE}/constant/boundaryConditions"
boundaryField
{
    movingWall { $:movingWall.U }
    fixedWalls { $:fixedWalls.U }
    frontAndBack { $:frontAndBack.U }
}
```

frontAndBackパッチのパッチ名をemptyとし、setConstraintTypes をinclude しても良い。
その方が良い。

```
boundaryField
{
    movingWall { $:movingWall.p }
    fixedWalls { $:fixedWalls.p }
    frontAndBack { $:frontAndBack.p }
}
```