熱伝達の境界条件（OF-2.1、OF-2.3）

藤井　15/01/30

熱伝達の境界条件（OF-2.1、OF-2.3）

目次

1. はじめに

現在、OpenFOAM で laplacianFoam を使って、熱伝導の計算ができる状態にあるが、これは熱伝導のみで熱伝達の計算ができない状態にある。商用の構造解析ソフトで熱伝導を計算しようとした場合、熱伝達の計算ができる。この為、OpenFOAM 側でも熱伝達の計算ができる様に、この境界条件を作成してみる。

2. 熱伝達の境界条件（fixedAlphaTemp）の作成

2-1. 考え方

熱伝達は、固体表面とその固体周りの流体側との熱移動を計算する。その流体側に熱移動する熱量は、下式で表される。

$$\boldsymbol{q} = \alpha\left(\boldsymbol{T} - \boldsymbol{T}_{ref}\right)$$

q:熱流束（W/m2）
α:熱伝達率（W/m2.K）
T:固体表面温度（K）
Tref:流体側の温度（K）

また、固体側の熱伝導は、下式で表される。

$$\boldsymbol{q} = -\lambda \cdot \nabla \boldsymbol{T}$$

q:熱流速（W/m2）
λ:熱伝導率（W/m.K）
T:固体温度（K）

OpenFOAM は、温度を変数として、温度の方程式を解いている為、熱伝達を計算する為には、固体表面から温度差に応じた熱量を加減算死求める事になる。この計算をする為には、固体表面に固体の温度勾配（▽T）を設定する事で求める事ができる。以上の理由により、上式を変形して、下式で計算する事になる。

$$\boldsymbol{q} = -\lambda \cdot \nabla \boldsymbol{T} = \alpha\left(\boldsymbol{T} - \boldsymbol{T}_{ref}\right)$$

$$\nabla \boldsymbol{T} = \frac{\alpha}{\lambda}\left(\boldsymbol{T}_{ref} - \boldsymbol{T}\right)$$

上式で算出した温度勾配を境界条件として設定すれば、計算できる事になる。
この温度勾配を計算する為に、境界条件の入力フォーマットとして、以下の内容で値を入力し、温度勾配を計算できるものを作成してみる。

```
type fixedAlphaTemp;                //patchType
refTemp 300;                        //参照温度（雰囲気温度）
alphaH 250;                         //熱伝達係数
KField_or_KValue KField;            //熱伝導率λを field で読むか値で読むかを決定
KField lambda;                      //field で読む場合の field 名
KValue 30;                          //値で読む場合の値
gradient uniform 0;                 //温度勾配 Field の定義
value uniform 300;                  //温度の初期値
```

2-2. fixedAlphaTemp の作成

OpenFOAM に標準で備わっている境界条件「fixedGradient」を改造して新しい境界条件「fixedAlphaTemp」を作成する。
元々のソースコード「$WM_PROJECT_DIR/finiteVolume/fields/fvPatchFields/basic/fixedGradient」を「$WM_PROJECT_USER_DIR/applications/libraries/myBCs/」にフォルダ毎コピーする。
コピー後、FOAM 端末上から以下を入力して、ファイル名と境界条件名を一括変換する。

```
$ cd $WM_PROJECT_USER_DIR/applications/libraries/myBCs/fixedGradient
$ rename 's/fixedGradientFvPatchField/fixedAlphaTempFvPatchScalarField/g' */*
$ sed -i 's/fixedGradientFvPatchField/fixedAlphaTempFvPatchScalarField/g' */*
```

オリジナルの fixedGradient は、field の型が scalar、vector 等どの型でも使う事ができるが、今回の場合、熱伝達の計算のため、field の型を scalar に固定している。この方法は、修正箇所が増えるが、field 間の演算が楽になる。
修正方法は、基本的に以下で修正し、コンパイルが通る事を確認しておく。

(1) template 文は全て削除
(2) public fvPatchField<Type> → public fixedGradientFvPatchScalarField
    この変更により、全ての「fvPatchField<Type>」が「fixedGradientFvPatchScalarField」に変わることになる。
    ただし、「tmp<fvPatchField<Type> >」部は、「tmp<fvPatchScalarField>」に変える。
(3) TypeName("fixedGradient") → TypeName("fixedAlphaTemp")
    この変更により、境界条件名が「fixedAlphaTemp」に変わる。
(4) Type → scalar に変更
    この変更により、field の型が scalar に固定される。
(5) ***.H ファイルに include を追加
      #include "fixedGradientFvPatchFields.H"
(6) ***.C ファイルに include を追加
      #include "addToRunTimeSelectionTable.H"
      #include "fvPatchFieldMapper.H"
      #include "volFields.H"
(7) ***.H ファイルの最後の行をコメントアウト
      //#ifdef NoRepository
      //#   include "fixedHeatTransferFvPatchScalarField.C"
      //#endif
(8) ***.C ファイルに境界条件を db に登録する手続きを追加

コンパイルする為の Make フォルダ内の files と options ファイルの内容は以下で設定して、コンパイルする。

----------files の内容----------

```
./fixedAlphaTempFvPatchScalarField.C                    //コンパイルの対象
LIB = $(FOAM_USER_LIBBIN)/libFixedAlphaTemp             //ライブラリ名
-----------------------------------------------------------
```

```
---------------optionsの内容-----------------------------
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/triSurface/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude
LIB_LIBS = \
    -lOpenFOAM \
    -ltriSurface \
    -lmeshTools
-----------------------------------------------------------
```

コンパイルが通る事を確認した上で、fixedAplphaTemp.H、fixedAlphaTemp.C ファイルに以下の修正を加える。（ここでは、代表的な部分のみ記載。詳細は、ソースコードを参照。）

```
-----------------------fixedAlphaTemp.H-----------------------------------
              :
        // Evaluation functions

            //追加---------------------------------
            //- Update the coefficients associated whith the patch field
            virtual void updateCoeffs();
            //-------------------------------------

            //- Return gradient at boundary
            //virtual tmp<Field<Type> > snGrad() const
            virtual tmp<Field<scalar> > snGrad() const
            {
                return gradient_;
            }
              :
-------------------------------------------------------------------------------
```

```
------------------fixedAlphaTemp.C（処理部を追加）-----------------------------
              :
//追加-------------------------------------
void Foam::fixedAlphaTempFvPatchScalarField::updateCoeffs()
{
    if (this->updated())
    {
        return;
    }
    //変数定義
    Field<scalar> flowHeat;
    //Field<scalar> intValue = this->patchInternalField();
    const fvPatchField<scalar>& patchValue = *this;
    if (KName_or_KValue_ == "KField")
    {
        //  Tの境界条件に lambda field の値を読んで設定する為、
        //  solver側の createFieldで fieldを定義する順番は、先に lambda、その後に Tを定義する
        //  lambdaと Tの定義する順番を間違えると「region0内に lambdaFieldが無い」のエラーが
```

```
        //  solver 実行時に発生する。
        const fvPatchField<scalar>& lamp =
            patch().lookupPatchField<volScalarField, scalar>(KName_);
        //flowHeat = (refTemp_ - intValue) * alphaH_ / lamp;
        flowHeat = (refTemp_ - patchValue) * alphaH_ / lamp;
    }
    else
    {
        //熱伝導率を値で取得する場合
        //flowHeat = (refTemp_ - intValue) * alphaH_ / KValue_;
        flowHeat = (refTemp_ - patchValue) * alphaH_ / KValue_;
    }
   gradient_ = flowHeat;
   fixedGradientFvPatchScalarField::updateCoeffs();
}
//---------------------------------------
//template<class Type>
void Foam::fixedAlphaTempFvPatchScalarField::evaluate(const Pstream::commsTypes)
{
    if (!this->updated())
    {
        this->updateCoeffs();
    }
              :
-------------------------------------------------------------------------------------------
```

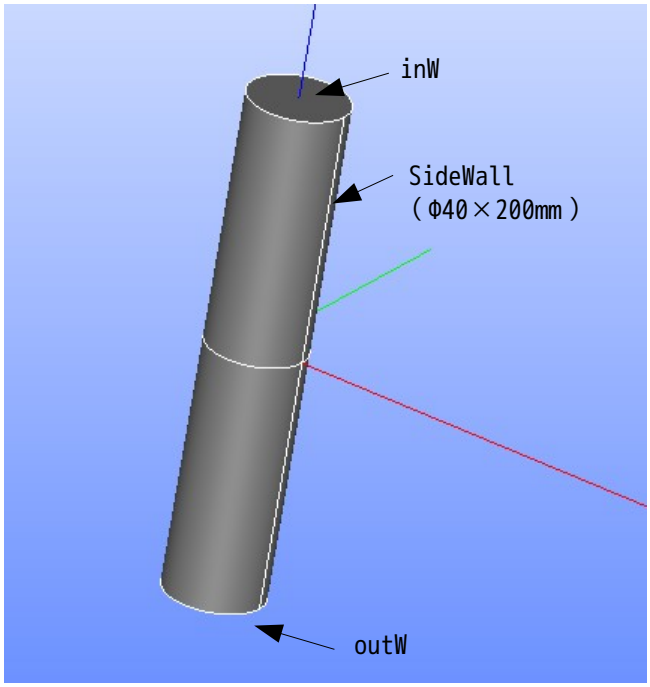
3.　作動確認

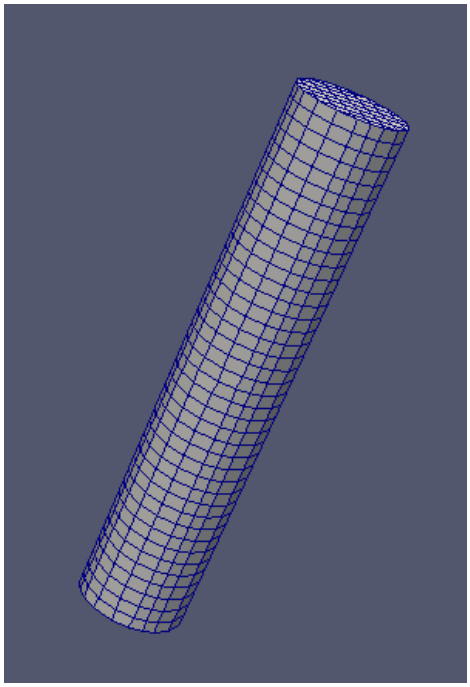
境界条件「fixedAlphaTenp」を作成する事ができたので、ここで作動確認してみる。

3-1.　モデルの作成

モデルは、単純に一次元で計算できる様に丸棒（Φ40×200mm）とし、モデル全体に初期値を与え、丸棒の片側の端面に熱伝達率を設定し、それ以外の面は、zeroGradient として熱の出入りがない状態で確認する。

熱伝達の境界条件（OF-2.1、OF-2.3）





メッシュは、要素サイズ 5mm でメッシュを切っている。

3-2. solver

使用する solver は、laplacianFoam で計算する。

3-3. 境界条件

境界条件は、以下で設定した。

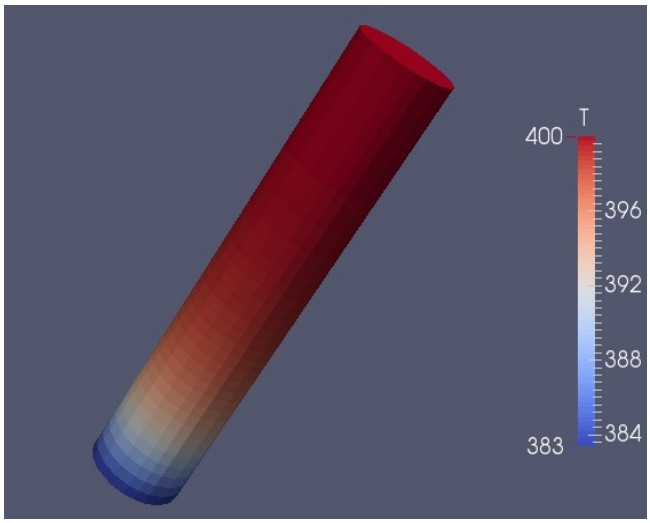
```
patch 名       T
---------------------------------
internalField 400
inW           zeroGradient
outW          fixedAlphaTemp
sideWall      zeroGradient
```

fixedAlphaTemp の内容は以下で設定している。

```
type fixedAlphaTemp;
refTemp 300;                //雰囲気温度
ahphaH 250;                 //熱伝達率
KField_or_KValue KValue;    //KValue データで計算
KField lambda;
KValue 52.8;
gradient uniform 0;
value uniform 300;
```

3-4. 計算結果の確認

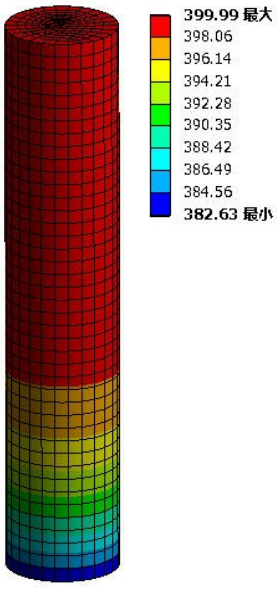
10 秒間隔で 100 秒まで計算させた。100 秒後の結果が下図になる。



100 秒後の outW 端面側の cell の温度が 383K まで下がっている事になる。100 秒後で outW の端面に設定されている温度勾配は、395K/m が設定されていた。

4.　計算結果の検証

同じモデルを同条件で ANSYS で解析してみる。
この結果が、以下になる。熱伝達面の最低温度が、「382.63K」であり、誤差は 1K 以下に収まった。
ANSYS の場合は、メッシュサイズ 5mm で OpenFOAM と同じだが、2 次メッシュで計算しているので、OpenFOAM 側でその半分のメッシュサイズ（2.5mm）でメッシュを切り、結果を確認すると、誤差は 0.4K まで縮まっている。
ANSYS は有限要素法、OpenFOAM は有限体積法で解法の差がある為、誤差は生じてくる。
　　有限要素法：節点の値を求める
　　有限体積法：要素中心の値を要素に流れこむ flux から求める



5. まとめ

今回、ANSYS で行う熱伝達解析を OpenFOAM で置き換える為には、OpenFOAM には無い熱伝達の境界条件を作成する必要があったが、これが作成でき誤差も少ない事が判った為、OpenFOAM に置き換える事ができる。
現在の境界条件は、以下が存在するので、これを組み合わせて熱解析を行うことになる。これらの境界条件を温度 T field に設定する事になる。

熱伝達	: fixedAlphaTemp で熱伝達率を固定
熱流束固定	: fixedGradient で温度勾配を固定（熱流束 q = −▽T/λ で与える）
温度固定	: fixedValue で温度固定
熱絶縁	: zeroGradient で熱的に絶縁する