OpenFOAM®非圧縮性流体解析演習シリーズ

第8回

buoyantBoussinesqSimpleFoam ソルバーや乱流モデルのカスタマイズ

今野 雅　（オープンCAE学会、東京大学）

OpenCAE

目次

OpenCAE

ソルバーのカスタマイズ

OpenCAE

ソルバーのカスタマイズの鉄則1

1. 単純なソルバーと、少しだけ機能が増えているソルバーのソースの差分を見て、カスタマイズの手法を勉強する

2. 自分がカスタマイズして盛り込みたい機能にできるだけ近い動作をするソースを探して、そのソースを参考にしてカスタマイズするのが鉄則

OpenCAE

ソルバーのカスタマイズの鉄則2

1. 単純なソルバーと、少しだけ機能が増えているソルバーのソースの差分を見て、カスタマイズの手法を勉強する

- 今回の対象ソルバー：

  heatTransfer/buoyantBoussinesqSimpleFoam

- より単純なソルバー：

  incompressible/simpleFoam

OpenCAE

ソルバーのカスタマイズの鉄則3

2. 自分がカスタマイズして盛り込みたい機能にできるだけ近い動作をするソースを探して、そのソースを参考にしてカスタマイズするのが鉄則

- 今回盛り込みたい機能：絶対湿度Xの輸送の解析(定常)

$$\nabla \cdot (\mathbf{U}X) = \nabla \cdot \left(\left(DX + \frac{\nu_t}{Sc_t} \right) \nabla X \right)$$

- 近い動作をするソース：T温度の輸送を解くTEqn.H

$$\nabla \cdot (\mathbf{U}T) = \nabla \cdot \left(\left(\frac{\nu}{Pr} + \frac{\nu_t}{Pr_t} \right) \nabla T \right)$$

OpenCAE

ソルバーのファイル構成の違い

```
sol↵ ソルバーのソースの場所に行く
cd heatTransfer/buoyantBoussinesqSimpleFoam/↵
ls↵
ls ../../incompressible/simpleFoam/↵
```

simpleFoam.C	←本体→	buoyantBoussinesqSimpleFoam.C
simpleFoam.dep		buoyantBoussinesqSimpleFoam.dep
Make		*Make*
	追加→	TEqn.H
UEqn.H		UEqn.H
convergenceCheck.H		convergenceCheck.H
createFields.H		createFields.H
initConvergenceCheck.H		initConvergenceCheck.H
pEqn.H		pEqn.H
	追加→	readTransportProperties.H

OpenCAE

TEqn.H(前回の復習)

```
kappat = turbulence->nut()/Prt; 乱流温度拡散率
kappat.correctBoundaryConditions(); 温度拡散率の境界値を更新
volScalarField kappaEff("kappaEff", turbulence->nu()/Pr +
kappat); 実効乱流温度拡散率
fvScalarMatrix TEqn 温度輸送方程式の定義
(
    fvm::div(phi, T)                移流項
  - fvm::Sp(fvc::div(phi), T)       移流項(数値安定用)
  - fvm::laplacian(kappaEff, T)     拡散項
);
TEqn.relax(); 温度輸送方程式の緩和(SIMPLE法)
eqnResidual = TEqn.solve().initialResidual(); 温度輸送方程式を解く
maxResidual = max(eqnResidual, maxResidual); 方程式の最大残差
ブシネスク近似による浮力項用の実効密度(比)
rhok = 1.0 - beta*(T - TRef);
```

$$\kappa_{Eff} = \frac{\nu}{Pr} + \frac{\nu_t}{Prt}$$

$$TEqn$$
$$= \nabla \cdot (\mathbf{U}T)$$
$$- (\nabla \cdot \mathbf{U})T$$
$$- \nabla \cdot (\kappa_{Eff} \nabla T)$$

OpenCAE

readTransportProperties.H

```
    singlePhaseTransportModel laminarTransport(U, phi);

    // Thermal expansion coefficient [1/K] 熱膨張係数
    dimensionedScalar beta(laminarTransport.lookup("beta"));

    // Reference temperature [K] 参照温度(浮力計算用)
    dimensionedScalar TRef(laminarTransport.lookup("TRef"));

    // Laminar Prandtl number (層流)プラントル数(流体の物性値)
    dimensionedScalar Pr(laminarTransport.lookup("Pr"));

    // Turbulent Prandtl number 乱流プラントル数(モデル係数)
    dimensionedScalar Prt(laminarTransport.lookup("Prt"));
```

OpenCAE

本体ファイルの違い 1

diff ../../incompressible/simpleFoam/simpleFoam.C buoyantBoussinesqSimpleFoam.C↵　diffコマンドで差分が見れる

先頭のコメントの違いは省略

42a59 File1での行番号+a(追加)+File2での行番号

```
>     #include "readGravitationalAcceleration.H"
```

57c74 File1での行番号+c(変更)+File2での行番号

```
<         p.storePrevIter();
---
>         p_rgh.storePrevIter();
```

行頭が<のはFile1の行

---は区切り

行頭が>のはFile2の行

61a79

```
>             #include "TEqn.H"
```

OpenCAE

本体ファイルの違い２

```
diff -u ../../incompressible/simpleFoam/simpleFoam.C
buoyantBoussinesqSimpleFoam.C⏎
```

unified形式で差分を表示

先頭は省略。行頭が+なのは追加された行、-なのは削除された行

```
+     #include "readGravitationalAcceleration.H"
      #include "createFields.H"
      #include "initContinuityErrs.H"
```

←追加行(重力加速度を読む)

```
@@ -54,11 +71,12 @@
          #include "readSIMPLEControls.H"
          #include "initConvergenceCheck.H"

-         p.storePrevIter();
+         p_rgh.storePrevIter();

          // Pressure-velocity SIMPLE corrector
          {
              #include "UEqn.H"
+             #include "TEqn.H"
              #include "pEqn.H"
          }
```

熱流体では圧力pの代りにp_rgh(p-rho*g*h)を解く

TEqn.H(温度Tの輸送を解く)が追加

湿度の輸送も同様に追加すれば良い!

OpenCAE

本体ファイルの構造(復習)と変更方針

```
    Info<< "\nStarting time loop\n" << endl;

    while (runTime.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        #include "readSIMPLEControls.H"
        #include "initConvergenceCheck.H"

        p.storePrevIter();

        // Pressure-velocity SIMPLE corrector
        {
            #include "UEqn.H"
            #include "TEqn.H"
            #include "pEqn.H"
        }

        turbulence->correct();

        runTime.write();

        #include "convergenceCheck.H"
    }
```

時間ループ(定常解法なので、厳密には反復ループ)

時刻(反復数)を出力

SIMPLE法の設定を読む

収束判定の反復毎初期化

緩和計算用に圧力の前回の値を保存

SIMPLE法を用いて、
速度場・圧力場を解く
さらに、温度場も解く

ここに湿度用XEqn.Hを加える

乱流量を解く

解を条件に応じて出力する

収束判定

OpenCAE

カスタマイズの方針と手順

1. カスタム・ソルバー名は

buoyantBoussinesqHumiditySimpleFoam とする

2. 本体ソースのSIMPLE corrector部で絶対湿度Xの輸送を解くソースXEqn.Hをインクルードするようにする

3. 絶対湿度場Xを読むコードをcreateFields.Hに加える(定石)

4. 絶対湿度の輸送方程式に必要な物性値やモデル係数を読み込むコードをreadTransportProperties.Hに加える (定石)

OpenCAE

カスタムソルバーの置き場所作成

標準ソルバーのソースは$FOAM_SOLVERS
(=OpenFOAMのシステム/applications/solvers)
に置かれているが、カスタム・ソルバーはユーザー用ディレクトリ
($WM_PROJECT_USER_DIR)の下に置く。
この下ならどこでも良いが
$WM_PROJECT_USER_DIR/applications/solvers がお勧め

run⏎ runで通常$WM_PROJECT_USER_DIR/runに行くので
cd ..⏎ さらにcd ..で上に行けば$WM_PROJECT_USER_DIRに行く
mkdir -p applications/solvers⏎ -pオプションでapplications
の親ディレクトリが無くても問題なくサブディレクトリsolversが作れる

OpenCAE

元のソルバーのコピー

```
cd applications/solvers⏎

cp -r $FOAM_SOLVERS/heatTransfer/
buoyantBoussinesqSimpleFoam/ .⏎
```

環境変数$FOAM_SOLVERSもTABキーで補完でき、その後に/を打ってから

TABで、実際の内容であるディレクトリ名に展開されるので、その後の

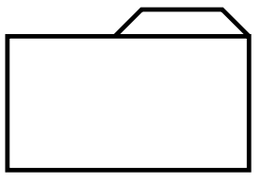
heatTransfer以下もTABで補間できる。

```
mv buoyantBoussinesqSimpleFoam
buoyantBoussinesqHumiditySimpleFoam⏎ 名前の変更もmvで行う

cd buoyantBoussinesqHumiditySimpleFoam/⏎

wclean⏎ コンパイル・オブジェクトや依存ファイルリスト等を消す
```

ソルバーのソース構成(復習)と変更



newApp

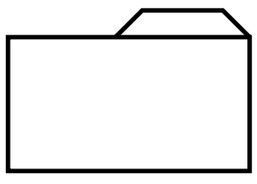
アプリ・ソースのディレクトリ
アプリ名と同じ名前

newApp.C

アプリ本体ソース。ソルバー名.C

otherHeader.H

本体に組み込まれるソース。*.H



Make

wmake(マルチプラットホーム対応
ビルドスクリプト)用ディレクトリ

files

ファイル・リストと実行ファイル名

options

インクルード・パスとライブラリ

OpenCAE

ファイル名の変更

本体ファイル名をソルバー名に合せて変更

```
mv buoyantBoussinesqSimpleFoam.C
buoyantBoussinesqHumiditySimpleFoam.C⏎
```

```
cd Make/⏎
```

```
gedit files⏎
```
ファイル・リストと実行ファイル名を変更

filesを以下のように変更(赤字を追加)：

```
buoyantBoussinesqHumiditySimpleFoam.C

EXE = $(FOAM_APPBIN)/buoyantBoussinesqHumiditySimpleFoam
```

```
cd ../⏎
```

```
wmake⏎
```
ソルバーを改造する前に、設定が間違っていないか、wmakeでコンパイルして確かめる(原則として変更は少しずつ行うほうが安全!)

OpenCAE

本体ファイルの修正

```
gedit buoyantBoussinesqHumiditySimpleFoam.C↵
```

```
        // Pressure-velocity SIMPLE corrector
        {
            #include "UEqn.H"
            #include "TEqn.H"
            #include "XEqn.H"
            #include "pEqn.H"
        }
```

ここにXEqn.Hをインクルードする行を加える

OpenCAE

XEqn.Hの作成

```
cp TEqn.H XEqn.H↵ 参考にするTEqn.Hをコピーして XEqn.Hに
gedit XEqn.H↵
```

~~kappat = turbulence->nut()/Prt;~~ 簡便のため乱流分の場をわざわざ作らない

~~kappat.correctBoundaryConditions();~~ よってこれも不要

volScalarField DXEff("DXEff", ~~a~~DX + turbulence->nut()/Sct); 実効分

fvScalarMatrix XEqn 湿度輸送方程式の定義

```
(
    fvm::div(phi, X)                移流項
  - fvm::Sp(fvc::div(phi), X)       移流項(数値安定用)
  - fvm::laplacian(DXEff, X) 拡散項
);
```

OpenCAE

XEqn.relax(); 湿度輸送方程式の緩和(SIMPLE法)

eqnResidual = XEqn.solve().initialResidual(); 湿度輸送方程式を解く

maxResidual = max(eqnResidual, maxResidual); 方程式の最大残差

~~rhok = 1.0 - beta*(T - TRef);~~ これも不要

readTransportProperties.Hへの追加

```
gedit readTransportProperties.H⏎
```

```
    // Laminar Prandtl number (層流)プラントル数(流体の物性値)
    dimensionedScalar Pr(laminarTransport.lookup("Pr"));

    // Turbulent Prandtl number 乱流プラントル数(モデル係数)
    dimensionedScalar Prt(laminarTransport.lookup("Prt"));
```

上の行をコピーして、ファイルの末尾に加え、以下の赤字の所を書き変える

```
    // Vapor diffusion number 湿度拡散数
    dimensionedScalar DX(laminarTransport.lookup("DX"));

    // Turbulent Schmidt number 乱流シュミット数(モデル係数)
    dimensionedScalar Sct(laminarTransport.lookup("Sct"));
```

OpenCAE

createField.Hへの追加

```
gedit createField.H↵
```

Tの場を読む行をコピーして、以下の赤字の部分を書き変える

```
    Info<< "Reading field X\n" << endl;
    volScalarField X
    (
        IOobject
        (
            "X",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );
```

OpenCAE

コンパイル&テスト

wmakeでコンパイルして、成功したら。helpオプションで起動テスト

```
wmake↵
buoyantBoussinesqHumiditySimpleFoam -help↵
```

buoyantBoussinesqSimpleFoamのケースをコピーして書き変える

```
run↵
cd tutorials/heatTransfer↵
cp -r buoyantBoussinesqSimpleFoam/
buoyantBoussinesqHumiditySimpleFoam/↵
cd buoyantBoussinesqHumiditySimpleFoam/↵
cd hotRoom/0/↵
cp T X↵
```

OpenCAE

Xの修正

```
gedit X↵
```

```
dimensions      [0 0 0 0 0 0 0]; [kg/kg(DA)] 1kgの乾き空気に対して湿り空気に水蒸気の重量(重量絶対湿度)。DAはDry Airの略
internalField   uniform 0.009; 9[g/kg(DA)]

boundaryField
{
    floor
    {
        type            fixedValue;
        value           uniform 0.01; 10[g/kg(DA)]
    }
    ceiling
    {
        type            fixedValue;
        value           uniform 0.008;8[g/kg(DA)]
    }
```

transportPropertiesへの追加

```
cd ../constant/↵
gedit transportProperties↵
```

```
// Vapor diffusion number
DX                  DX  [0 2 -1 0 0 0 0] 2.61e-5;

// Turbulent Schmidt number
Sct                 Sct [0 0 0 0 0 0 0] 1.0;
```

OpenCAE

systemファイル修正1

```
cd ../system/↵
gedit controlDict↵
```

```
application     buoyantBoussinesqHumiditySimpleFoam;
```

```
gedit fvSchemes↵
```

```
divSchemes
略
    div(phi,T)      Gauss upwind;
    div(phi,X)      Gauss upwind;
略
laplacianSchemes
略
    laplacian(kappaEff,T) Gauss linear corrected;
    laplacian(DXEff,X) Gauss linear corrected;
```

OpenCAE

systemファイル修正2と実行

```
gedit fvSolution↵
```

```
solvers
略
    "(U|T|X|k|epsilon|R)"
略

relaxationFactors
略
    T                   0.5;
    X                   0.5;
```

実行

```
cd ..↵
./Allclean↵
./Allrun↵
```

OpenCAE

乱流モデルのカスタマイズ (基礎)

OpenCAE

乱流モデルの場所と種類

```
src⏎ ソースの場所
cd turbulenceModels/⏎ 全ての乱流モデル・壁関数等の場所
cd incompressible/⏎ 非圧縮性乱流モデル
cd RAS/⏎ 非圧縮性RAS乱流モデル（LESモデルは別ディレクトリ)
ls⏎
```

```
LRR
略
Make wmake用の設定ディレクトリ
NonlinearKEShih
略
RASModel RASModelクラス
略
kEpsilon 標準k-epsilonモデル
```

標準k-εモデルのクラス

```
cd kEpsilon↵
ls↵
more kEpsilon.H↵
```

```
略
Description
    Standard k-epsilon turbulence model for incompressible
flows.
略
class kEpsilon kEpsilonクラス
:
    public RASModel kEpsilonクラスはRASModelクラスを継承している
略
```

標準k-εのコンストラクタ

more kEpsilon.C↵

kEpsilon::kEpsilon コンストラクタ

略

epsilon_は以下のようにコンストラクト時に宣言され、epsilonという名前で読み書きされるのでソルバー本体側で定義する必要がない。kOmegaモデルでは、epsilonの代りにomegaが定義される。k等も同様→乱流モデルの変数が変更可

```
    epsilon_
    (
        IOobject
        (
            "epsilon",
            runTime_.timeName(),
            mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        autoCreateK("epsilon", mesh_)
    ),
```

標準k-εのepsilon方程式

void kEpsilon::correct() 本体ソースのturbulence->correct()で呼ばれる関数

略

```
    // Dissipation equation
    tmp<fvScalarMatrix> epsEqn 輸送方程式(カスタマイズの要点)
    (
        fvm::ddt(epsilon_)
      + fvm::div(phi_, epsilon_)
      - fvm::Sp(fvc::div(phi_), epsilon_) 数値安定化(無くても可)
      - fvm::laplacian(DepsilonEff(), epsilon_)
     ==
        C1_*G*epsilon_/k_
      - fvm::Sp(C2_*epsilon_/k_, epsilon_)
    );

    epsEqn().relax()

    epsEqn().boundaryManipulate(epsilon_.boundaryField());

    solve(epsEqn);
```

標準k-εのコピーと修正

cd ..↵

cp -r kEpsilon kEpsilon2↵

cd kEpsilon2↵

kEpsilon.HはkEpsilon2.Hに名前を変更する必要があり、またファイル内のkEpsilonというクラス名やマクロ名も全てkEpsilon2に変更する必要があるので、sedコマンドで置換する。(s/置換前/置換後/g)

sed s/kEpsilon/kEpsilon2/g kEpsilon.H > kEpsilon2.H↵

more kEpsilon2.H↵ 確認

kEpsilon.Cも同様

sed s/kEpsilon/kEpsilon2/g kEpsilon.C > kEpsilon2.C↵

more kEpsilon2.C↵ 確認

rm kEpsilon.*↵ 確認が終わったら、旧いkEpsilon.*は消去

ソース修正

```
gedit kEpsilon2.C↵
```

```
void kEpsilon2::correct()
略

    // Dissipation equation
    tmp<fvScalarMatrix> epsEqn
    (
        fvm::ddt(epsilon_)
      + fvm::div(phi_, epsilon_)
      - fvm::Sp(fvc::div(phi_), epsilon_)   この行を消す
      - fvm::laplacian(DepsilonEff(), epsilon_)
     ==
        C1_*G*epsilon_/k_
      - fvm::Sp(C2_*epsilon_/k_, epsilon_)
    );
```

OpenCAE

wmakeの設定とコンパイル

```
cd ../Make↵
gedit files↵
```

```
RASModel/RASModel.C

laminar/laminar.C
kEpsilon/kEpsilon.C
kEpsilon2/kEpsilon2.C この行を追加
```

乱流モデル・ライブラリのコンパイル

```
cd ..↵
wmake libso↵
```

OpenCAE

新乱流モデルで計算

```
run↵
cd tutorials/heatTransfer/
buoyantBoussinesqHumiditySimpleFoam/↵
cd constant↵
gedit RASProperties↵
```

```
//RASModel          kEpsilon;
RASModel          kEpsilon2;
```

実行

```
cd ..↵
./Allclean↵
./Allrun↵
```

OpenCAE

以上です

質問をどうぞ!

OpenCAE