OpenFOAM®非圧縮性流体解析演習シリーズ

第7回
buoyantBoussinesqSimpleFoam
ソルバーを題材としたOpenFOAMの
ソースの読み方演習

今野 雅　（オープンCAE学会、東京大学）

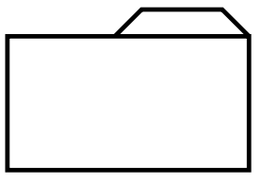
OpenCAE

アプリケーションを読む環境

1. more, find, grepなどのUNIXコマンドやエディタを用いて、手動でソースを読む進む (手軽にソースを見る用、ある意味上級者用)
   - 利点： 新たに他のツールをインストールする必要がない
   - 欠点：一般に効率が悪い
2. 1とOpenFOAM C++ Documentationを併用 (中級者用)
   - 利点：1.の利点+クラスの構造を調べる効率が良い
   - 欠点：ツールを行き来する必要がある
3. Eclipseなどの 統合開発環境(IDE)を用いる (本格的開発用)
   - 利点： IDE内で編集、クラス検索、ビルド等の開発が可能
   - 欠点：新たにIDEのツールをインストールする必要がある

OpenCAE

アプリケーションのソース構成



newApp

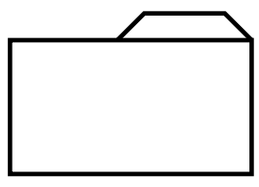
アプリ・ソースのディレクトリ
アプリ名と同じ名前

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

アプリケーションのソース構成

```
sol↵
cd heatTransfer/↵
cd buoyantBoussinesqSimpleFoam/↵
ls ↵
```

```
Make :wmake用ディレクトリ
buoyantBoussinesqSimpleFoam.C :本体ソース(=アプリ名.C)
buoyantBoussinesqSimpleFoam.dep :依存ファイルリスト(生成物)
TEqn.H :本体ソース等に組み込まれるファイル(以下同様)
UEqn.H
convergenceCheck.H
createFields.H
initConvergenceCheck.H
pEqn.H
readTransportProperties.H
```

アプリケーションのソース構成

```
cd Make↵
ls↵
```

files :コンパイル用ファイル

linux64GccDPOpt :wmakeの出力ディレクトリ(プラットフォーム毎)

options :インクルード・パスやリンクするライブラリ

```
more files↵
```

buoyantBoussinesqSimpleFoam.C :コンパイルするファイル

```
EXE = $(FOAM_APPBIN)/buoyantBoussinesqSimpleFoam
```
:実行ファイル名=アプリ名。 $(FOAM_APPBIN)はシステムのアプリ置場

アプリケーションのソース構成

```
more options⏎
```

```
EXE_INC = \ :明示的に指定する必要があるヘッダー・ファイルの検索場所
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/turbulenceModels \
    -I$(LIB_SRC)/turbulenceModels/incompressible/RAS/lnInclude \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel

EXE_LIBS = \ :リンクするライブラリ($FOAM_LIBBIN などに置かれる)
    -lfiniteVolume \ :ファイル名はlibfiniteVolume.so (Macでは.dylib)
    -lmeshTools \ :ファイル名はlibmeshTools.so (以下同様)
    -lincompressibleRASModels \
    -lincompressibleTransportModels
```

OpenCAE

アプリケーションの本体ソース

```
cd ..
more buoyantBoussinesqSimpleFoam.C
```

```
Description
    Steady-state solver for buoyant, turbulent flow of incompressible
fluids

    Uses the Boussinesq approximation:
    \f[
        rho_{eff} = 1 - beta(T - T_{ref})
    \f]

    where:
        \f$ rho_{eff} \f$ = the effective (driving) density
        beta = thermal expansion coefficient [1/K]
        T = temperature [K]
        \f$ T_{ref} \f$ = reference temperature [K]

    Valid when:
    \f[
        rho_{eff} << 1
    \f]
```

Description:
アプリの説明

OpenCAE

ソルバーの本体ソース(導入部)

アプリで使用するクラスのヘッダ・ファイル

クラス宣言部

```
#include "fvCFD.H"
#include "singlePhaseTransportModel.H"
#include "RASModel.H"
```

```
int main(int argc, char *argv[])
{
```

main関数から開始

本体に組み込まれるソース・ファイル

```
    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    #include "readGravitationalAcceleration.H"
    #include "createFields.H"
    #include "initContinuityErrs.H"
```

ヘッダーファイルの検索場所

1. $WM_PROJECT_DIR/src/OpenFOAM/lnInclude
($WM_PROJECT_DIR/src=$FOAM_SRC ←短いのでコマンドで使用)

2. ローカルディレクトリ/lnIncludeやローカルディレクトリ自身

(ここではローカルディレクトリ=*buoyantBoussinesqSimpleFoam*)

3. プラットフォーム依存のディレクトリ

4. Make/optionsファイル中のEXE_INC変数で明示的に指定されているディレクトリ (ここでは以下)

```
EXE_INC = \
  -I$(LIB_SRC)/finiteVolume/lnInclude \
  -I$(LIB_SRC)/turbulenceModels \
  -I$(LIB_SRC)/turbulenceModels/incompressible/RAS/lnInclude \
  -I$(LIB_SRC)/transportModels \
  -I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel
```

($LIB_SRC=$WM_PROJECT_DIR/src
=$FOAM_SRC)

ヘッダーファイルの検索方法

1. ローカルディレクトリ以下

   → 範囲が狭いのでls等ですぐわかる(もちろんfind .しても良い)

2. それ以外は以下のように、大抵 $FOAM_SRC 以下にある

   - $FOAM_SRC/OpenFOAM/lnInclude
   - Make/optionsの$EXE_INCのリスト → $FOAM_SRC 以下

   → $FOAM_SRC 以下は範囲が広いのfindで探す

3. ある特定のクラスの宣言ヘッダファイルを探す場合

   → ”クラス名.H”を$FOAM_SRC からfind

OpenCAE

ヘッダーファイルの検索方法

```
find $FOAM_SRC -name fvCFD.H↵
```

```
$FOAM_SRC/finiteVolume/cfdTools/general/include/fvCFD.H
$FOAM_SRC/finiteVolume/lnInclude/fvCFD.H
```

2つのファイルが存在する

前のコマンド出力をxargsを用いてls -lの引数に渡して詳しく調べる

xargsの前の|は縦棒(日本語キーボードならSHIFT+¥)

```
find $FOAM_SRC -name fvCFD.H | xargs ls -l↵
```

```
$FOAM_SRC/finiteVolume/cfdTools/general/include/fvCFD.H
$FOAM_SRC/finiteVolume/lnInclude/fvCFD.H
                        -> ../cfdTools/general/include/fvCFD.H
```

実は下のファイルは上のファイルへのシンボリックリンクで中身は同じ

OpenCAE

ヘッダーファイルの検索方法

```
$FOAM_SRC/finiteVolume/cfdTools/general/include/fvCFD.H
$FOAM_SRC/finiteVolume/lnInclude/fvCFD.H
                        -> ../cfdTools/general/include/fvCFD.H
実は下のファイルは上のファイルへのシンボリックリンクで中身は同じ
```

実体と別にリンクを集めたディレクトリがある理由

- EXE_INCではリンクがあるディレクトリのみをパスに記述
  - cfdTools/general/include等のサブディレクトリを羅列不要にしている
  - turbulenceModels/incompressible/RAS/lnIncludeなども同様

```
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/turbulenceModels \
    -I$(LIB_SRC)/turbulenceModels/incompressible/RAS/lnInclude \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel
```

OpenCAE

ヘッダーファイルの検索方法

ヘッダーを検索する場合、通常EXE_INCに記述されている
シンボリックリンクのみを探せば良い

-type l(エル)オプションでシンボリックリンクのみ検索

```
find $FOAM_SRC -type l -name fvCFD.H
```

$FOAM_SRC/finiteVolume/lnInclude/fvCFD.H 1ファイルのみ表示

1ファイルのみの出力なのでxargsでmoreに渡せばそのソースが見れる

```
find $FOAM_SRC -type l -name fvCFD.H | xargs more
```

OpenCAE

fvCFD.Hの中身

```
#ifndef fvCFD_H 重複取り込み防止(fvCFD_Hが未定義の場合に続行)
#define fvCFD_H 一度取込むとfvCFD_Hが定義される

#include "parRun.H" 他のヘッダー群を取り込む(以下同様)
#include "Time.H"
#include "fvMesh.H"
#include "fvc.H"
(中略)
#ifndef namespaceFoam 名前空間の重複定義防止(最初と同様)
#define namespaceFoam
    using namespace Foam; namespace(名前空間)を定義
#endif
#endif
```

ソルバー解読に必要なクラスの定義のみ調べ、全てのヘッダーの中身を逐次細かく見ることは今回はしない

OpenCAE

組み込まれているソース読む

```
    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    #include "readGravitationalAcceleration.H"
    #include "createFields.H"
    #include "initContinuityErrs.H"
```

```
find $FOAM_SRC -type l -name setRootCase.H | xargs more⏎
```

```
    Foam::argList args(argc, argv);
    if (!args.checkRootCase())
    {
        Foam::FatalError.exit();
    }
```

以下、同様にcreateTime.Hなどを読んでいく。ただし、
createFields.Hはカレントディレクトリにある

OpenCAE

createFields.Hを見る

```
more createFields.H
```

Info文はログに出力されるコメントなので、ソース解読に大変重要

```
Info<< "Reading thermophysical properties\n" << endl;

 Info<< "Reading field T\n" << endl;
 volScalarField T   T(温度場)のクラスはvolScalarField
    ( T生成(Construct)時の引数はIOobjectクラスとmesh。meshのクラス名はここではわからないので、別に調べる必要がある
        IOobject
        (
            "T",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );
```

OpenCAE

meshのクラス名を調べる

meshを含む行を、ソルバーのソースがあるディレクトリから検索

```
grep mesh *↵
```

meshの定義らしき行は見つからない→別の場所のインクルードファイルなので、インクルードしている本体ソースを眺める

```
more buoyantBoussinesqSimpleFoam.C↵
```

```
int main(int argc, char *argv[])
{
    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H" ←これらしい(上から順に探しても良い)
    #include "readGravitationalAcceleration.H"
    #include "createFields.H"
    #include "initContinuityErrs.H"
```

OpenCAE

meshのクラス名を調べる

createMesh.Hをfindで探してmoreでソースを見る

```
find $FOAM_SRC -type l -name createMesh.H | xargs more⏎
```

```
    Foam::Info
        << "Create mesh for time = "
        << runTime.timeName() << Foam::nl << Foam::endl;

    Foam::fvMesh mesh  クラスはfvMesh
    (
        Foam::IOobject
        (
            Foam::fvMesh::defaultRegion,
            runTime.timeName(),
            runTime,
            Foam::IOobject::MUST_READ
        )
    );
```

OpenCAE

fvMeshをDoxygenで検索

Doxygenとはソースから自動的にクラスの依存関係やコメントを調べてのドキュメンテーションを作成するツール
以下のコンテンツはこれをWebで見れるようにしているもの

URL: www.openfoam.com/doc/cpp

OpenFOAM OpenFOAM C++ Documentation OpenCFD

Home | Features | Download | Documentation | Support | Training | Resources | News

Main Page | Related Pages | Namespaces | Classes | Files | Directories

fvMesh

fvMesh Foam
fvMesh
fvMesh.C
fvMesh.H
fvMesh_ Foam::fvMe
fvMeshAdder Foam
fvMeshAdder.C
fvMeshAdder.H
fvMeshAdderTemplates.C
fvMeshDistribute Foam
fvMeshDistribute Foam::fvMeshDistribute

検索フィールドにfvMeshを入れて赤丸をクリック

About OpenFOAM

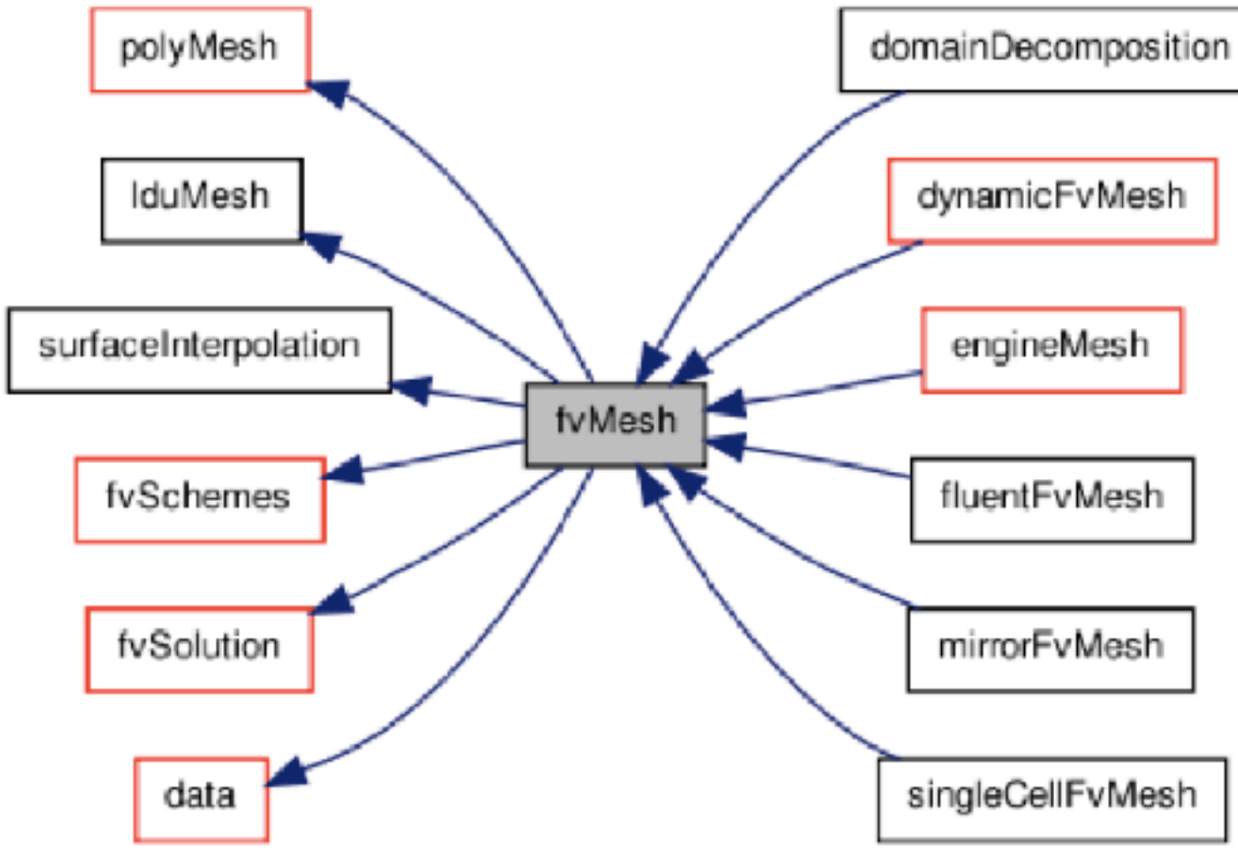
OpenFOAM is a free, open source CFD software packag
areas of engineering and science, from both commercial
anything from complex fluid flows involving chemical reac

OpenCAE

fvMeshクラスの説明

fvMesh Class Reference

Mesh data needed to do the Finite Volume discretisation. More...
Inheritance diagram for fvMesh:



クラスの継承
のダイアグラム

typedef fvMesh Mesh

fvMeshクラスは別名(typedef)はMesh

OpenCAE

volScalarFieldを検索

www.openfoam.com/docs/cpp/

OpenFOAM M C++ DocumeOpenCFD

検索フィールドにvolScalarField
を入れて赤丸をクリック

Home | Features | Downloa

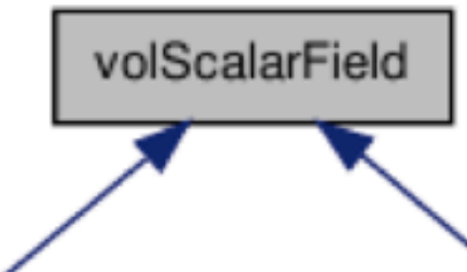
Class List | Class Index | Class Hierarchy | Class Members

volScalarField

volScalarField
volScalarField
Foam::volScalarField()

volScalarField Class Reference

Inheritance diagram for volScalarField:



OpenCAE

volScalarFieldクラス

volScalarFieldはGeometricFieldテンプレートの<scalar, fvPatchField, volMesh>版

typedef GeometricField<scalar, fvPatchField, volMesh> volScalarField

Definition at line 52 of file volFieldsFwd.H.

typedef GeometricField<vector, fvPatchField, volMesh> volVectorField

Definition at line 55 of file volField

赤丸をクリック

```
00051 template<class Type, template<class> class PatchField, class GeoMesh>
00052 class GeometricField;
00053
00054 typedef GeometricField<scalar, fvPatchField, volMesh> volScalarField;
00055 typedef GeometricField<vector, fvPatchField, volMesh> volVectorField;
00056 typedef GeometricField<sphericalTensor, fvPatchField, volMesh>
00057     volSphericalTensorField;
00058 typedef GeometricField<symmTensor, fvPatchField, volMesh> volSymmTensorField;
00059 typedef GeometricField<tensor, fvPatchField, volMesh> volTensorField;
```

赤丸をクリック

OpenCAE

volScalarFieldクラス

Public Member FunctionsのConstructorの中で、引数の数とクラス(IOobjectとMesh)が合っているものを探す

GeometricField (const **IOobject** &, const **Mesh** &, const **dimensionSet** &, const **Field**< Type > &, const **PtrList**< PatchField< Type > > &)
Constructor from components.

GeometricField (const **IOobject** &, const **Mesh** &)
Construct and read given **IOobject**.

赤丸をクリック

GeometricField (const **IOobject** & io,
const **Mesh** & mesh
) [inline]

Construct and read given **IOobject**.
Definition at line 354 of file **GeometricField.C**.
References **Foam::endl()**, **Foam::exit()**, **FatalIOErrorIn**, and **mesh**.
Here is the call graph for this function:

赤丸をクリック

OpenCAE

IOobjectクラス

IOobject defines the attributes of an object for which implicit **objectRegistry** management is supported, and provides the infrastructure for performing stream I/O.
An **IOobject** is constructed with an object name, a class name, an instance path, a reference to a **objectRegistry**, and parameters determining its storage status.

Detailed Description

Read options
Define what is do n and explicit reads:

Optionsの説明

Parameters:

MUST_READ	Object must be read from **Istream** on construction. Error if **Istream** does not exist or can't be read. Does not check timestamp or re-read.
MUST_READ_IF_MODIFIED	Object must be read from **Istream** on construction. Error if **Istream** does not exist or can't be read. If object is registered its timestamp will be checked every timestep and possibly re-read.
READ_IF_PRESENT	Read object from **Istream** if **Istream** exists, otherwise don't. Error only if **Istream** exists but can't be read. Does not check timestamp or re-read.
NO_READ	Don't read

Write options
Define what is done

次はobjectRegistryというように深く調べていく

Parameters:

AUTO_WRITE	Object is written automatically when requested to by the **objectRegistry**.
NO_WRITE	No automatic write on destruction but can be written explicitly

createFields.Hを見る

```
    Info<< "Reading field p_rgh\n" << endl;
    volScalarField p_rgh  p_rghはTと同様

(中略)

    Info<< "Reading field U\n" << endl;
    volVectorField U volVectorFieldはvolScalarFieldのvector版
    (
        IOobject
        (
            "U",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

    #include "createPhi.H"
```

OpenCAE

createPhi.Hを見る

```
find $FOAM_SRC -type l -name createPhi.H | xargs more⏎
```

```
Info<< "Reading/calculating face flux field phi\n" << endl;

surfaceScalarField phi surfaceScalarFieldはvolScalarFieldの
surfaceMesh版(格子の界面で定義される場。phiは流束であることに注意)
(
    IOobject
    (
        "phi",
        runTime.timeName(),
        mesh,
        IOobject::READ_IF_PRESENT,
        IOobject::AUTO_WRITE
    ),
    linearInterpolate(U) & mesh.Sf()  初期値を計算して与えている
);
```

Read object from Istream if Istream exists, otherwise don't. Error only if Istream exists but can't be read. Does not check timestamp or re-read.

OpenCAE

const surfaceVectorField & Sf () const
Return cell face area vectors.

createFields.Hを見る

```
    #include "readTransportProperties.H"
```

more readTransportProperties.H⏎　このファイルはローカルにある

```
    singlePhaseTransportModel laminarTransport(U, phi);

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

    // Reference temperature [K]
    dimensionedScalar TRef(laminarTransport.lookup("TRef"));

    // Laminar Prandtl number
    dimensionedScalar Pr(laminarTransport.lookup("Pr"));

    // Turbulent Prandtl number
    dimensionedScalar Prt(laminarTransport.lookup("Prt"));
```

DoxygenでlookupはDoxygenで何か調べてみよう

OpenCAE

createFields.Hを見る

```
    Info<< "Creating turbulence model\n" << endl;
    autoPtr<incompressible::RASModel> turbulence
    (
        incompressible::RASModel::New(U, phi,
laminarTransport)
    );
```

RASModelは次回の演習(カスタマイズ編)で解説予定

```
// Kinematic density for buoyancy force
 volScalarField rhok
 (
     IOobject
     (
         "rhok",
         runTime.timeName(),
         mesh
     ),
     1.0 - beta*(T - TRef)
 );
```

OpenCAE

createFields.Hを見る

```
// kinematic turbulent thermal thermal conductivity m2/s
Info<< "Reading field kappat\n" << endl;
volScalarField kappat
(
    IOobject
    (
        "kappat",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
Info<< "Calculating field g.h\n" << endl;
volScalarField gh("gh", g & mesh.C());    &は内積

surfaceScalarField ghf("ghf", g & mesh.Cf());
```

const volVectorField &	C () const Return cell centres as volVectorField.
const surfaceVectorField &	Cf () const Return face centres as surfaceVectorField.

OpenCAE

ソルバーの本体ソース(コア部)

```
    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

乱流量を解く

解を条件に応じて出力する

収束判定

OpenCAE

TEqn.H(温度輸送方程式)を読む

```
kappat = turbulence->nut()/Prt; 乱流温度拡散率
kappat.correctBoundaryConditions(); 流温度拡散率を境界値を更新
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);
```

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

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

OpenCAE

以上です

質問をどうぞ!

OpenCAE