20190824版

メイリオ → 源真ゴシック Regular
源真ゴシックP Regular
源真ゴシックP Medium

OpenFOAMソースコードの読み方入門
熱伝導方程式の解法から有限体積法の実装について考える
後編：laplacianFoamでの実装

2019年8月24日 オープンCAE勉強会@富山

富山県立大学　中川慎二

注意

- OpenFOAMユーザーガイド，プログラマーズガイド，OpenFOAM Wiki，CFD Online，その他多くの情報を参考にしています。開発者，情報発信者の皆様に深い謝意を表します。
- この講習内容は，講師の個人的な経験（主に，卒研生等とのコードリーディング）から得た知識を共有するものです。この内容の正確性を保証することはできません。この情報を使用したことによって問題が生じた場合，その責任は負いかねますので，予めご了承ください。

概要

- OpenFOAM の利用者を対象とし，OpenFOAMのソースコードの読み方の基本を学びます。
- OpenFOAMの中で最も基本的なソルバの1つ”laplacianFoam”（熱伝導方程式を解くソルバ）を例にとります。
- 実際にOpenFOAMのソースコードを見ながら，OpenFOAMのソースコードの特徴，ソースコード解読初心者が躓きやすい点などについても，解説します。

概要

- 偏微分方程式 → 有限体積法で離散化 → セル間距離や物性値などで定まる係数列（a_E, a_W, a_P など）から行列式 → 解
- OpenFOAMでは，この部分がソルバ・レベルでは，巧妙に隠蔽されている。（知らなくても使える）
- OpenFOAMで作られる行列式は，どんなもの？どうやって作られる？Schemeを実行時に選べるのはどういう仕組みか？ を調べる

前編では…

- 1次元熱伝導方程式を有限体積法によって離散化し，行列を作成する．
- この行列を手作業で解き，温度分布が求められることを確認する．
- この過程で必要な式変形などを確認する．

後編では…

- 熱伝導方程式を解くソルバ “laplacianFoam” のソースコードを見ながら，上記手作業で出てきた式が，どのように実装されている（コーディングされている）かを読み解く．
- 与えた偏微分方程式から行列が作られる過程を確認する．
- しかし，行列の解法には踏み込まない．

前編のおさらいと 後編のねらい

- 偏微分方程式 → 有限体積法で離散化 → セル間距離や物性値などで定まる係数列（a_E, a_W, a_P など）から行列式 → 解
- OpenFOAMでは，この部分がソルバ・レベルでは，巧妙に隠蔽されている。（知らなくても使える）
- OpenFOAMで作られる行列式は，どうなっているのか？どうやって作られるのか？ Schemeを実行時に選べるのはどういう仕組みか？ を調べる

OpenFOAMとC++

OpenFOAMのソースコード

- C++ 言語
- オブジェクト指向プログラミング
  - カプセル化 (振る舞いの隠蔽とデータ隠蔽)
  - インヘリタンス (継承) -- クラスベースの言語
  - ポリモーフィズム (多態性、多相性) -- 型付きの言語
    - オーバーロード（多重定義） 同じ名前で引数の異なる関数
    - オーバーライド　親クラスの関数を子クラスで上書き
- ジェネリックプログラミング
  - データ型に依存しないコード。Templateを利用。
- OpenFOAM とは、 C++言語でCFDを実行するための工具一式(toolkit)である。

C++ クラスとは

- 部品（オブジェクト）の設計図
- 様々な値（状態，属性）と，それを操作するための機能（function,メソッド，関数）を含む
- クラスは、値と関数の集まりである
- この設計図（クラス）に基づいて，プログラム実行時に，部品（オブジェクト）が作られる
- オブジェクト生成時にはコンストラクタが働く

一般コード例

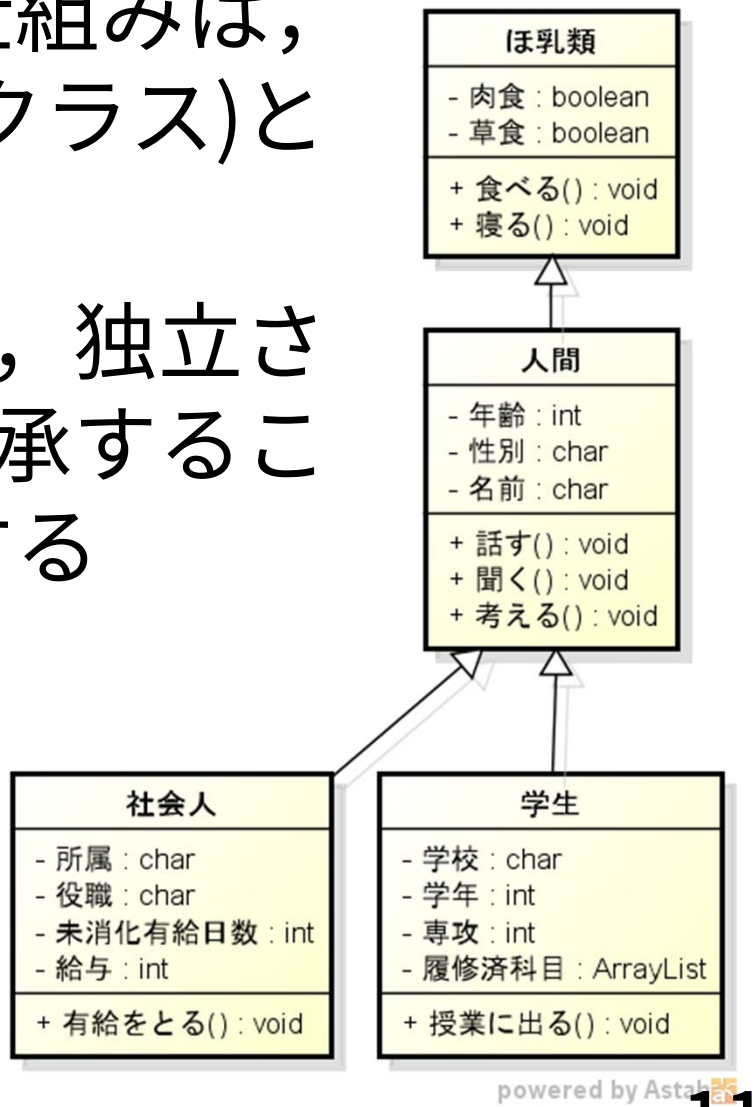
```
int    n(7);
int    i = 10;
型名   変数名（初期値）
```

OpenFOAMコード例

```
dimensionedScalar    DT( x, y);
クラス名             オブジェクト名（初期値）
```

C++ クラスの継承

- 複数のクラスに共通する仕組みは，抽象化した独立クラス(親クラス)とする
- 複数のクラス(子クラス)が，独立させたクラス(親クラス)を継承することで，共通の機能を実現する



例

- 親クラス＝人間
- 子クラス＝学生，社会人

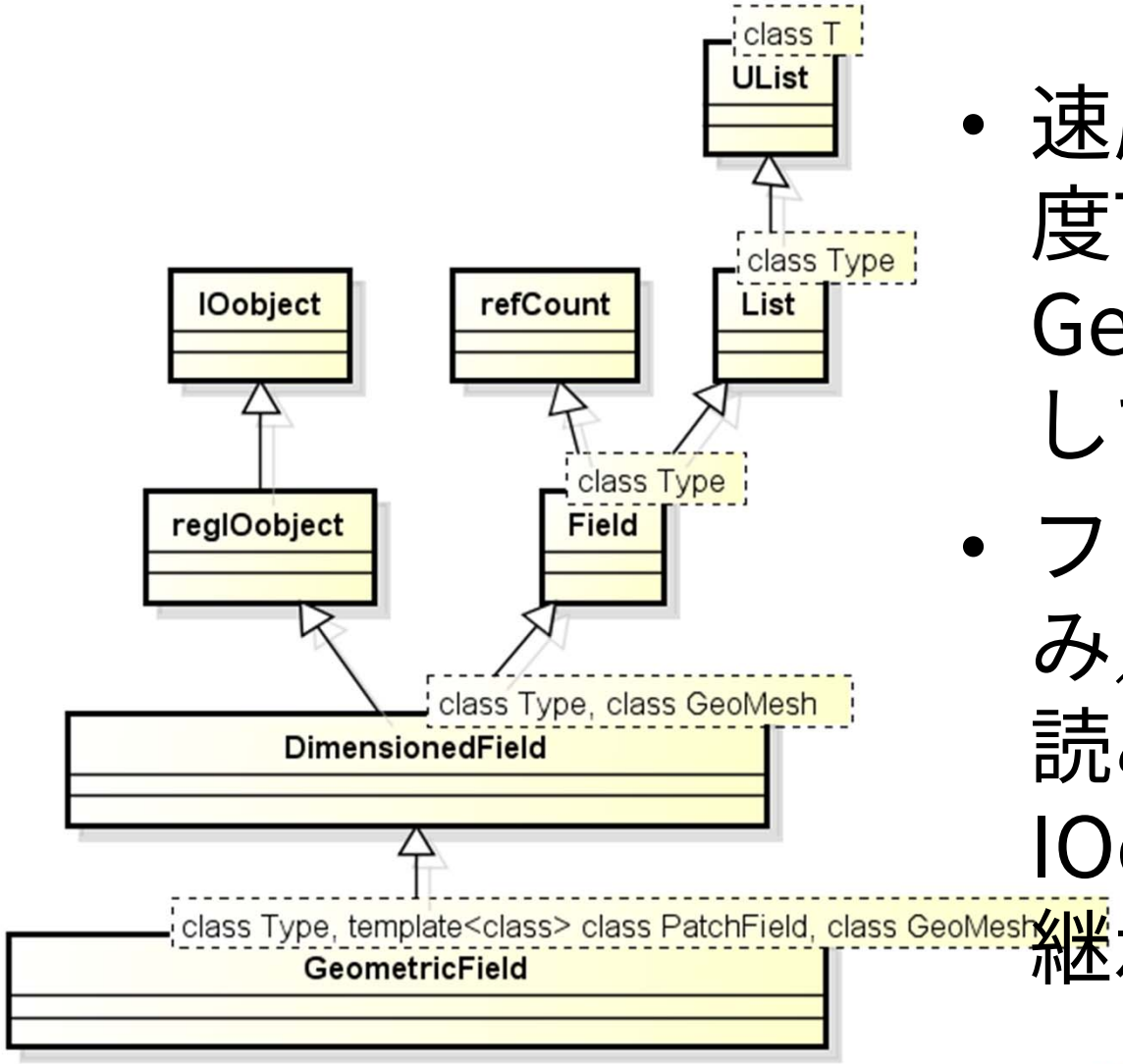
よく使う クラス

	非フィールド値 場所によらず一定	フィールド値 場所によって値が変わる	
		セル面での値	セル体積（中心）での値
スカラー	dimensionedScalar dimensioned<scalar>	surfaceScalarField GeometricField<scalar, fvsPatchField, surfaceMesh>	volScalarField GeometricField<scalar, fvPatchField, volMesh>
ベクトル	dimensionedVector dimensioned<vector>	surfaceVectorField GeometricField<vector, fvsPatchField, surfaceMesh>	volVectorField GeometricField<vector, fvPatchField, volMesh>

上の行は，typedefで定義された別名
下の行が本来の定義

templateクラスで多様なタイプに対応

GeometricFieldクラスの継承関係



- 速度U，圧力p，温度Tなどのデータは，GeometricField型として保存されている。
- ファイルへの書き込み／ファイルからの読み込みなどは，IOobjectクラスから継承している。

ソルバ

- 特定の問題を解くために，OpenFOAMのコードを組み合わせたプログラム。
- ソルバのソースコードは，シンプル。難しい作業は，部品（ライブラリ，クラス）に任せる。
- まとまった作業は別ファイルに記述し，includeすることで，読みやすさを保つ。
- 様々な部品（オブジェクト）が，協調しながら，目的を果たす。
- 部品どうしは，適切な独立性を持っている。
- ソルバのソースコードは，ソルバ名.C である。

ライブラリ

- 特定の機能を実現するのに必要な部品を集めたもの。
- 関連する複数のクラスから，1つのライブラリを作成する。
- 例えば，incompressibleTurbulenceModelsライブラリの中に，kEpsilonクラスや，kOmegaSSTクラス etc. が含まれている。
- srcディレクトリ内で，Makeディレクトリが存在するディレクトリから，ライブラリが作られる。

チュートリアル

- ソルバ，ライブラリ，クラスなどの機能・使い方を説明するために用意された例題。
- チュートリアルは，説明書の一部と考えられる。
- チュートリアルを実行しながら，理解を深めることが大切。
- ソースコード改造時には，その改造内容に応じたチュートリアルを作成する。

laplacianFoam

シミュレーションの流れ

- 偏微分方程式

$$\frac{\partial T}{\partial t} = \mathrm{div}(\Gamma\, \mathrm{grad} T) + S_T$$

有限体積法

- 離散方程式

$$a_P T_P = a_E T_E + a_W T_W + S_u$$

係数列の計算

- 行列

$$[A][T]=[b]$$

行列の解法

- 解

プログラムの階層構造

- Level 0, Top：
  - 偏微分方程式　ソルバー
- Level 1:
  - 離散化・有限体積法　finiteVolume
- Level 2:
  - 行列 fvMatrix, lduMatrix
- Level 3:
  - 基礎的部品　OpenFOAM
  - 単位，時間，リスト，メモリ，OS関連，並列化

laplacianFoam

Application　laplacianFoam

Description　Solves a simple Laplace equation, e.g. for thermal diffusion in a solid.

非定常　拡散方程式

$$\frac{\partial T}{\partial t} - \mathrm{div}(\Gamma\, \mathrm{grad} T) = 0$$

$$\int_V \frac{\partial T}{\partial t} dV - \int_V \mathrm{div}(\Gamma\, \mathrm{grad} T) dV$$

$$= \int_V \frac{\partial T}{\partial t} dV - \int_S (\Gamma\, \mathrm{grad} T) \cdot \boldsymbol{n} dS = 0$$

$$[A][T]=[b]$$

laplacianFoamで使うソースコード

- ソルバディレクトリ
  $FOAM_APP/solver/basic/laplacianFoam
- srcディレクトリ（ソルバディレクトリ/Make/optionsで指定）
  - $WM_PROJECT_DIR/src/finiteVolume
  - $WM_PROJECT_DIR/src/meshTools
- 全ソルバ共通 src
  - $WM_PROJECT_DIR/src/OpenFOAM
  - $WM_PROJECT_DIR/src/OSspecific/POSIX

Linuxの基礎：$付きの文字列は，変数を意味する。その中身を知りたい時は，端末にて次のコマンドを実行する。(echoコマンド)
echo $FOAM_APP

Level 0：メイン部分の概説

```
#include "fvCFD.H “    // OpenFOAMの基盤的機能を有効にする。
#include "fvOptions.H “    // 任意のソース項を扱う fvOptions ヘッダーファイル読み込み
#include "simpleControl.H “    // simple法の機能を使うため。
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
int main(int argc, char *argv[])
{
// OpenFOAMの基盤的な機能を有効にする。
    #include "addCheckCaseOptions.H “
    #include "setRootCaseLists.H"
    #include "createTime.H"
    #include "createMesh.H"
// simple法クラスから，simple法をコントロールするためのオブジェクトsimpleを作成
    simpleControl simple(mesh);
// 変数 温度場T，拡散係数DT，設定ディクショナリ transportProperties の作成，ファイル読込。
    #include "createFields.H"
    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
    Info<< "￥nCalculating temperature distribution￥n" << endl;
// シンプル法のオブジェクトを使って，繰り返し回数をコントロールする
    while (simple.loop(runTime))
```

C++の基礎：#include “ファイル名” の部分には，指定したファイルの中身がそのまま貼付けられる。そのファイルの存在する場所は，コンパイル用設定で指定している。

Level 0：メイン部分の概説

```
while (simple.loop(runTime))  // シンプル法オブジェクトで，繰返回数をコントロール
{
    Info<< "Time = " << runTime.timeName() << nl << endl;

    while (simple.correctNonOrthogonal())   // 非直行性補正が有効な場合のみ実行。
    {           // 非定常拡散方程式から線形代数式に相当する行列を作成する。
        fvScalarMatrix TEqn
        (    // 非定常項と拡散項をimplicit(陰的)に解く。fvm
            fvm::ddt(T) - fvm::laplacian(DT, T)
            ==
            fvOptions(T)
        );

        fvOptions.constrain(TEqn);      // 行列から解を求める。
        TEqn.solve();
        fvOptions.correct(T);
    }
    #include "write.H"// 結果の出力
    runTime.printExecutionTime(Info);
}
```

$$\frac{\partial T}{\partial t} - \nabla^2(\Gamma T) = 0$$

OpenFOAMでの偏微分方程式の項の離散化
P-39　Table 3.2 (OpenFOAM Programmer's Guide Version v1812, 7th December 2018, OpenCFD limited.)

行列

$$a_P T_P = a_E T_E + a_W T_W + S_u$$

$$[A][T]=[b]$$

$$\begin{bmatrix} a_{p1} & -a_{E1} & 0 & 0 \\ -a_{W2} & a_{P2} & -a_{E2} & 0 \\ 0 & -a_{W3} & a_{P3} & -a_{E3} \\ 0 & 0 & -a_{W4} & a_{P4} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} S_{u1} \\ S_{u2} \\ S_{u3} \\ S_{u4} \end{bmatrix}$$

- 偏微分方程式の各項から，これら行列に入る係数が出てくる。
- 項毎に行列を作り，まとめることで，最終的に解きたい行列式を作成する。

行列

$$\begin{bmatrix} a_{p1} & -a_{E1} & 0 & 0 \\ -a_{W2} & a_{P2} & -a_{E2} & 0 \\ 0 & -a_{W3} & a_{P3} & -a_{E3} \\ 0 & 0 & -a_{W4} & a_{P4} \end{bmatrix} =$$

$$\begin{bmatrix} S_{P1} & 0 & 0 & 0 \\ 0 & S_{P2} & 0 & 0 \\ 0 & 0 & S_{P3} & 0 \\ 0 & 0 & 0 & S_{P4} \end{bmatrix} - \begin{bmatrix} -a_{E1} & a_{E1} & 0 & 0 \\ a_{W2} & -a_{W2} - a_{E2} & a_{E2} & 0 \\ 0 & a_{W3} & -a_{W3} - a_{E3} & a_{E3} \\ 0 & 0 & a_{W4} & -a_{W4} \end{bmatrix}$$

非定常項

拡散項

$$\begin{bmatrix} S_{BC0} + S_{u1} \\ S_{u2} \\ S_{u3} \\ S_{BC5} + S_{u4} \end{bmatrix} = \begin{bmatrix} S_{u1\_ddt} \\ S_{u2\_ddt} \\ S_{u3\_ddt} \\ S_{u4\_ddt} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} S_{BC0} \\ 0 \\ 0 \\ S_{BC5} \end{bmatrix}$$

非定常項

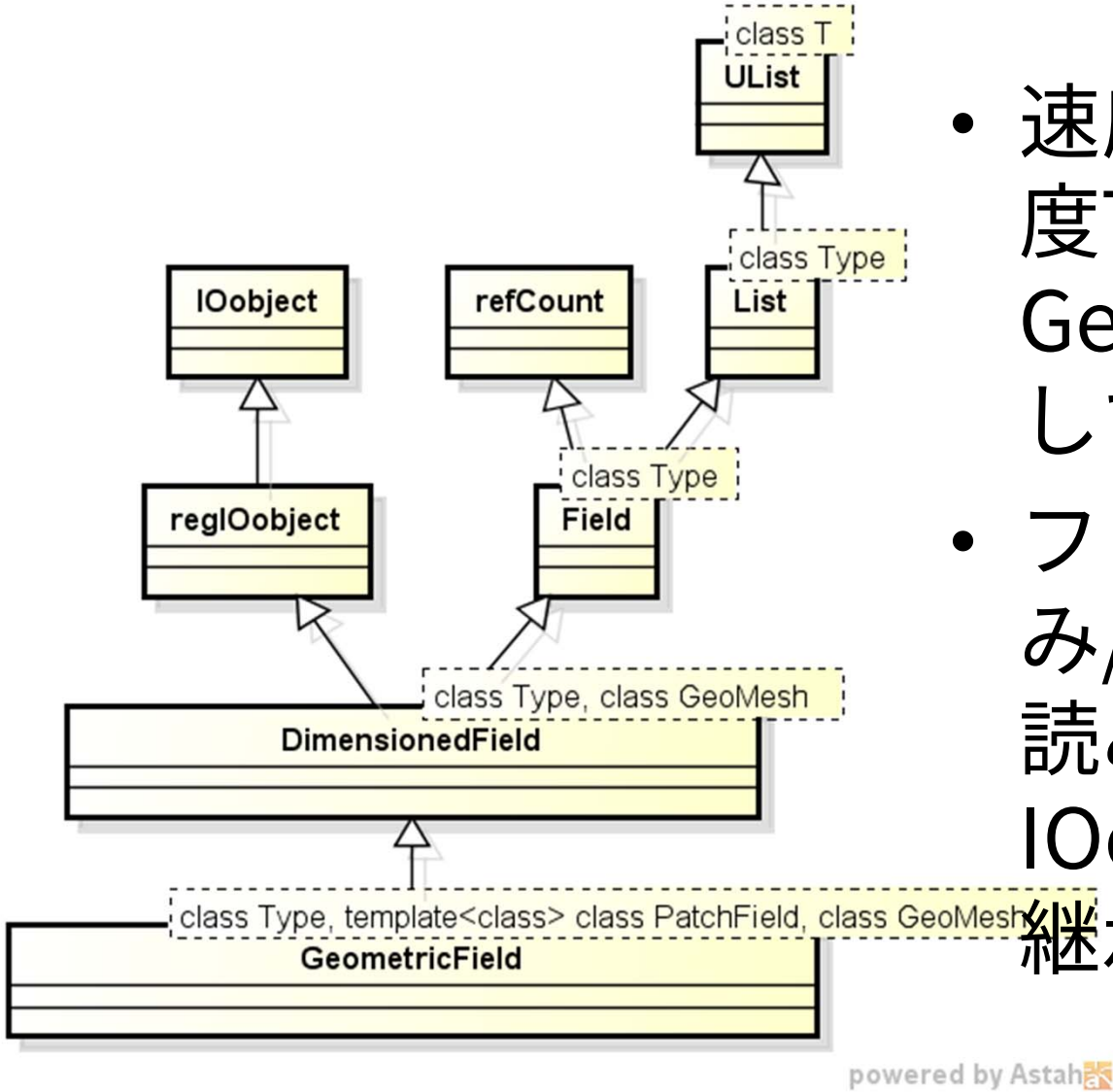
拡散項

境界条件

typedef

- 既存の型に　新しい名前(別名)を付ける
- コードが見やすくなる
- テンプレートなどを含んだ長い名前の型・クラスも，短く理解しやすい名前を付けることができる

GeometricFieldクラスの継承関係



- 速度U，圧力p，温度Tなどのデータは，GeometricField型として保存されている。
- ファイルへの書き込み/ファイルからの読み込みなどは，IOobjectクラスから継承している。

TとDTの定義：createFields.H

volScalarField T(IOobject, mesh);
Tという名前で，volScalarFieldクラスのオブジェクトを作成する。引数を2つ渡してコンストラクタ指定。

```
volScalarField T
(
    IOobject
    (
        "T",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

https://openfoam.com/documentation/guides/latest/api/classFoam_1_1dimensioned.html#a79486ccfa5d57a1e06b6e5386bef857f

dimensionedScalar DT(…);
DT という名前の dimensionedScalarクラスのオブジェクトを作成する。コンストラクタの引数は word, dimensionSet, dictionary。

```
dimensionedScalar DT
(
    "DT",
    dimViscosity,
    transportProperties
);
```

transportPropertiesディクショナリのDTという項目の値を読み込む。ディクショナリに次元が書かれている場合には，相違がないかチェックする。

transportPropertiesの定義：createFields.H

```
IOdictionary transportProperties
(
    IOobject
    (
        "transportProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
```

IOdictionary transportProperties(IOobject);
設定ファイル（ディクショナリ）クラスのオブジェクトとして，transportPropertiesを宣言する。
1つのIOobjectを引数とするコンストラクタ指定。

ディクショナリ名

格納場所

OpenFOAM-v1812/src/OpenFOAM/db/IOobjects/IOdictionary/

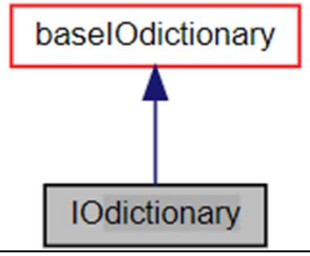
IOdictionaryクラス

IOdictionary Description

IOdictionary is derived from dictionary and IOobject to give the dictionary automatic IO functionality via the objectRegistry. To facilitate IO, IOdictionary is provided with a constructor from IOobject and with readData/writeData functions.

IOdictionary.H

```
53 class IOdictionary
54 :
55  public baseIOdictionary
```




```
class baseIOdictionary
:
    public regIOobject,
    public dictionary
```

```
62 //- Construct given an IOobject
63  IOdictionary(const IOobject&);
```

IOdictionary.C

```
33 Foam::IOdictionary::IOdictionary(const IOobject& io)
34 :
35    baseIOdictionary(io)
36 {
37     readHeaderOk(IOstream::ASCII, typeName);
38
39     // For if MUST_READ_IF_MODIFIED
40     addWatch();
41 }
```

クラスとコンストラクタ

```
class baseIOdictionary
:
    public regIOobject,
    public dictionary
```

```
Foam::baseIOdictionary::baseIOdictionary(const IOobject& io)
:
    regIOobject(io)
{
    dictionary::name() = IOobject::objectPath();
}
```

```
class regIOobject
:
    public IOobject
//- Construct from IOobject. Optional flag for if IOobject is the
//  top level regIOobject.
regIOobject(const IOobject&, const bool isTime = false);
```

```
Foam::regIOobject::regIOobject(const IOobject& io, const bool isTime)
:
    IOobject(io),
    registered_(false),
    ownedByRegistry_(false),
    watchIndices_(),
    eventNo_                // Do not get event for top level Time
database
```

IOobject(io)

```
class IOobject
```

IOobject(io)　このコンストラクタは？

```
        IOobject(const IOobject& io)
```

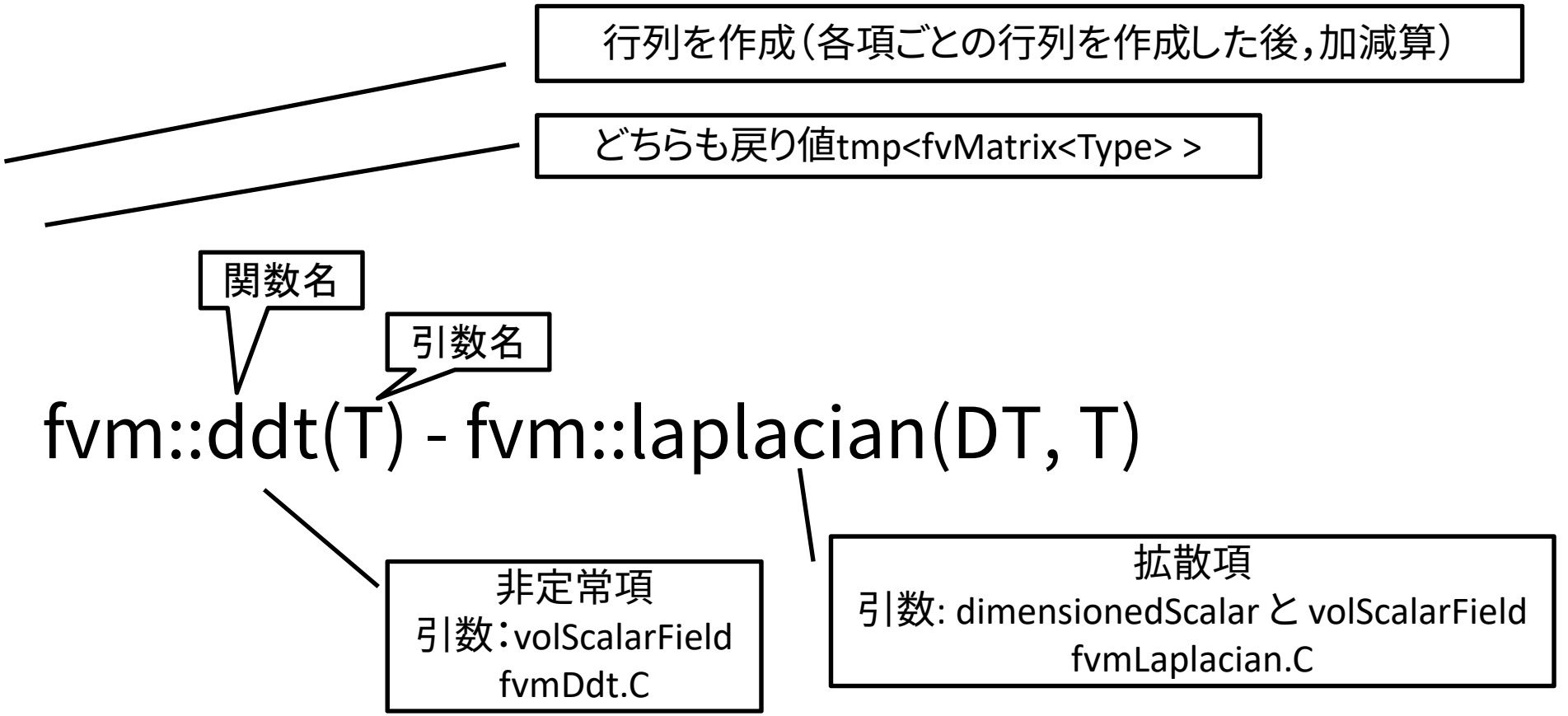
コピーコンストラクタ

定義されていない → デフォルトコンストラクタ

参考：regIOobjectでは コピーコンストラクタ が定義されている。

コピーコンストラクタ
https://ja.cppreference.com/w/cpp/language/copy_constructor

laplacianFoam.C メインループ



volScalarField は別名であり，本体は GeometricField<scalar, fvPatchField, volMesh>

src¥finiteVolume¥finiteVolume¥fvm¥

fvmDdt.C

テンプレートクラスを使う宣言

戻り値の型

関数名

引数

関数の中身

戻り値

scalar

参照渡し：vf は，読み出し側変数の別名となる。同一のものを操作することになる。これを変更すると，もとの変数の内容も変わる。

volScalarField T

T

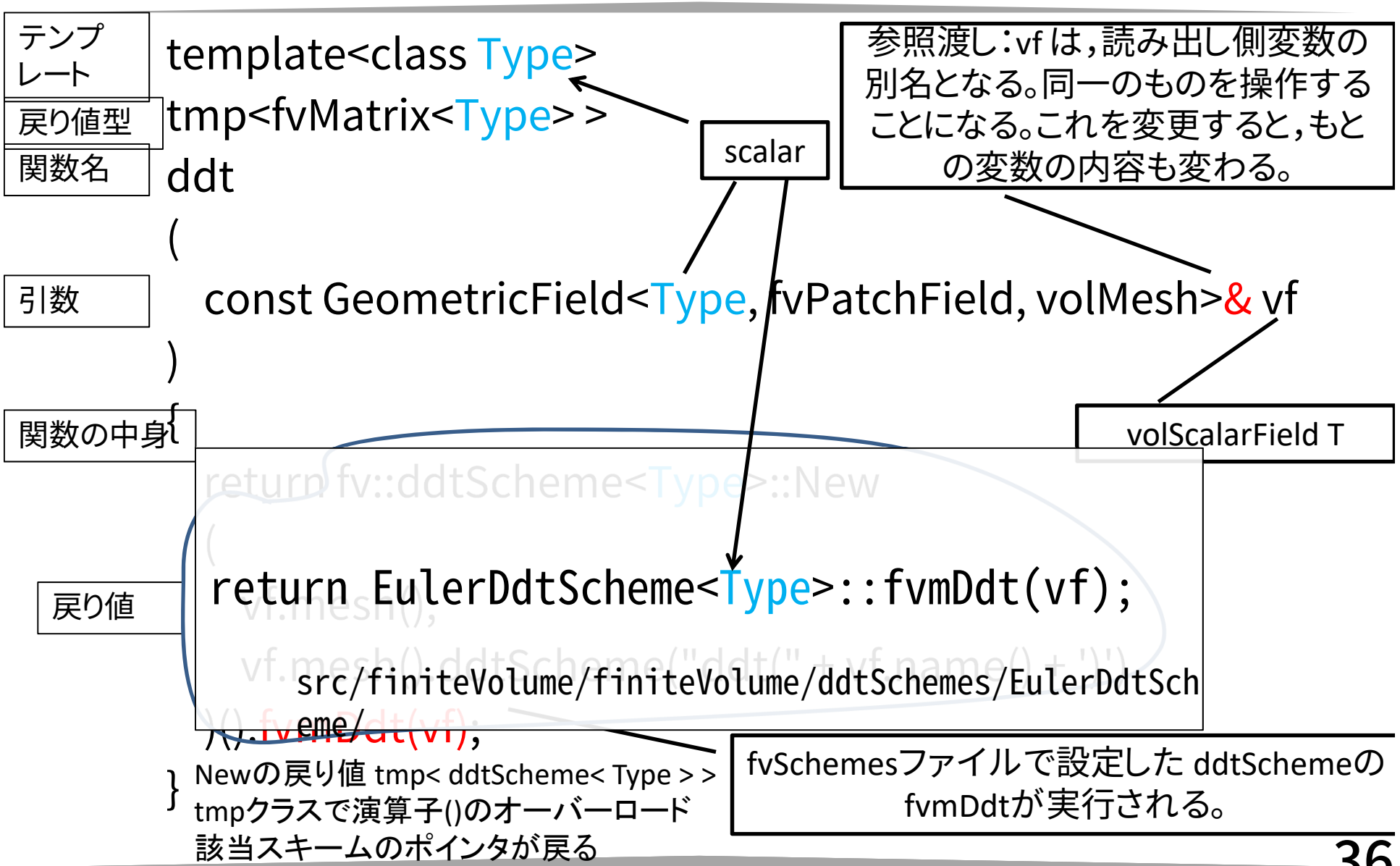
fvSchemesファイルで設定した ddtSchemeの fvmDdtが実行される。

```
template<class Type>
tmp<fvMatrix<Type> >
ddt
(
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    return fv::ddtScheme<Type>::New
    (
        vf.mesh(),
        vf.mesh().ddtScheme("ddt(" + vf.name() + ')')
    ).ref().fvmDdt(vf);
}
```

Newの戻り値 tmp< ddtScheme< Type > >
tmpクラスで演算子()のオーバーロード
該当スキームのポインタが戻る

src¥finiteVolume¥finiteVolume¥fvm¥

fvmDdt.C



コードの流れを追うために

- New関数がでてきたら，Run-Timeセレクション
- 該当するライブラリの中から，実行時（ケース内の設定）に指定したモデルが使用される
- 使用するモデルのコードを見る

Run-Time Selection

- OpenFOAM guide/runTimeSelection mechanism – OpenFOAMWiki

  http://openfoamwiki.net/index.php/OpenFOAM_guide/runTimeSelection_mechanism

- Run-time type selection in OpenFOAM - the type name system – sourceflux

  http://www.sourceflux.de/blog/runtime-type-selection-openfoam/

ddtScheme

- Euler, localEuler, CrankNicholson, backward, steadyState など（UserGuide 4.4.6）
- src/finiteVolume/finiteVolume/ddtSchemes にソースコードがある
- 今回は，最もシンプルな Euler (implicit)を考える。
  - クラス名　EulerDdtScheme

例題の設定
tutorials/basic/laplacianFoam/flange/system/fvSchemes

EulerDdtSchemeクラス

- src/finiteVolume/finiteVolume/ddtSchemes/EulerDdtScheme/
- クラス名 EulerDdtScheme
- タイプ名 Euler (EulerDdtScheme.Hで定義)

```
template<class Type>
class EulerDdtScheme
:
    public ddtScheme<Type>
{
//line 74    //- Runtime type information
    TypeName("Euler");
```

行列

$$\begin{bmatrix} a_{p1} & -a_{E1} & 0 & 0 \\ -a_{W2} & a_{P2} & -a_{E2} & 0 \\ 0 & -a_{W3} & a_{P3} & -a_{E3} \\ 0 & 0 & -a_{W4} & a_{P4} \end{bmatrix} =$$

ソースコード
fvm::ddt(T) - fvm::laplacian(DT, T)

$$\begin{bmatrix} S_{P1} & 0 & 0 & 0 \\ 0 & S_{P2} & 0 & 0 \\ 0 & 0 & S_{P3} & 0 \\ 0 & 0 & 0 & S_{P4} \end{bmatrix} - \begin{bmatrix} -a_{E1} & a_{E1} & 0 & 0 \\ a_{W2} & -a_{W2} - a_{E2} & a_{E2} & 0 \\ 0 & a_{W3} & -a_{W3} - a_{E3} & a_{E3} \\ 0 & 0 & a_{W4} & -a_{W4} \end{bmatrix}$$

これから設定
非定常項

未設定
拡散項

$$\begin{bmatrix} S_{BC0} + S_{u1} \\ S_{u2} \\ S_{u3} \\ S_{BC5} + S_{u4} \end{bmatrix} = \begin{bmatrix} S_{u1\_ddt} \\ S_{u2\_ddt} \\ S_{u3\_ddt} \\ S_{u4\_ddt} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} S_{BC0} \\ 0 \\ 0 \\ S_{BC5} \end{bmatrix}$$

今回は考えない：境界
条件に応じた値が入る

これから設定
非定常項

今回は考えない：拡散項
非直交性補正では値が入る

Euler implicit

ソースコード
fvm::ddt(T)

$$\int_V \frac{\partial T}{\partial t} dV = \frac{(T_P V_P)^{new} - (T_P V_P)^{old}}{\Delta t}$$

$$= \frac{(V_P)^{new}}{\Delta t} T_P - \frac{(T_P V_P)^{old}}{\Delta t}$$

T_Pの係数に追加

生成項に追加
（現時刻での温度とは無関係）

src/finiteVolume/finiteVolume/ddtSchemes/EulerDdtScheme/

EulerDdtScheme.C

line 332

```
template<class Type>
tmp<fvMatrix<Type> >
EulerDdtScheme<Type>::fvmDdt
(
    const GeometricField<Type, fvPatchField,
volMesh>& vf
)
{
    tmp<fvMatrix<Type> > tfvm
    (
        new fvMatrix<Type>
        (
            vf,
            vf.dimensions()*dimVol/dimTime
        )
    );

    fvMatrix<Type>& fvm = tfvm.ref();

    scalar rDeltaT =
1.0/mesh().time().deltaTValue();

    fvm.diag() = rDeltaT*mesh().Vsc();

    if (mesh().moving())
    {
        fvm.source() = rDeltaT*vf.oldTime().
primitiveField()*mesh().Vsc0();
    }
    else
    {
        fvm.source() = rDeltaT*vf.oldTime().
primitiveField()*mesh().Vsc();
    }

    return tfvm;
}
```

行列の対角成分に（セル体積／タイムステップ）を入れる。Vcell/Δt

$$\frac{(V_P)^{new}}{\Delta t} T_P$$

$$\frac{(T_P V_P)^{old}}{\Delta t}$$

行列の生成項に（旧時刻での値×セル体積／タイムステップ）を入れる。
Told*Vcell/Δt

行列

$$\begin{bmatrix} a_{p1} & -a_{E1} & 0 & 0 \\ -a_{W2} & a_{P2} & -a_{E2} & 0 \\ 0 & -a_{W3} & a_{P3} & -a_{E3} \\ 0 & 0 & -a_{W4} & a_{P4} \end{bmatrix} =$$

ソースコード
fvm::ddt(T) - fvm::laplacian(DT, T)

$$\begin{bmatrix} S_{P1} & 0 & 0 & 0 \\ 0 & S_{P2} & 0 & 0 \\ 0 & 0 & S_{P3} & 0 \\ 0 & 0 & 0 & S_{P4} \end{bmatrix} - \begin{bmatrix} -a_{E1} & a_{E1} & 0 & 0 \\ a_{W2} & -a_{W2} - a_{E2} & a_{E2} & 0 \\ 0 & a_{W3} & -a_{W3} - a_{E3} & a_{E3} \\ 0 & 0 & a_{W4} & -a_{W4} \end{bmatrix}$$

設定済み
非定常項

これから設定
拡散項

$$\begin{bmatrix} S_{BC0} + S_{u1} \\ S_{u2} \\ S_{u3} \\ S_{BC5} + S_{u4} \end{bmatrix} = \begin{bmatrix} S_{u1\_ddt} \\ S_{u2\_ddt} \\ S_{u3\_ddt} \\ S_{u4\_ddt} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} S_{BC0} \\ 0 \\ 0 \\ S_{BC5} \end{bmatrix}$$

今回は考えない：境界条件に応じた値が入る

設定済み
非定常項

今回は考えない：拡散項
非直交性補正では値が入る

Laplacian 拡散項

https://openfoam.com/documentation/user-guide/fvSchemes.php#x23-900006.2.4

- schemeの選択肢は，Gauss だけ（UserGuide 6.2.4）　ただし，下記補間方法の選択肢がある
  - Gauss <interpolationScheme> <snGradScheme>
- 拡散係数の補間(interpolation) 方法の選択肢は，linear, upwind, limitedLinear, vanLeer, MUSCL など（UserGuide 6.2.1, Table 6.3）
  - 隣接するセル中心での値から，セル面での値を補間する
- 面垂直方向勾配(snGrad)の補間方法の選択肢は，corrected, uncorrected, limited ψ, bounded, fourth など（UserGuide 6.2.2, Table 6.4）
  - 隣接するセル中心での勾配の値から，セル面での値を補間する
- src/finiteVolume/finiteVolume/laplacianSchemes にソースコードがある
- 今回は，最もシンプルな場合を考える。補間の詳細については触れない。

拡散項

ベクトルの内積

$$\int_S (\Gamma \,\mathrm{grad}T)\cdot \boldsymbol{n}dS = \sum_f \left[(\Gamma \,\mathrm{grad}T)\cdot \boldsymbol{S_f}\right] = (\Gamma \,\mathrm{grad}T)_e \cdot \boldsymbol{S_{f\_e}} + (\Gamma\, \mathrm{S}\, \mathrm{grad}T)_w \cdot \boldsymbol{S_{f\_w}}$$

$$= \left(\Gamma \frac{dT}{dx}\right)_e S_{f\_e} - \left(\Gamma \frac{dT}{dx}\right)_w S_{f\_w} = \Gamma_e \left(\frac{dT}{dx}\right)_e S_{f\_e} - \Gamma_w \left(\frac{dT}{dx}\right)_w S_{f\_w}$$

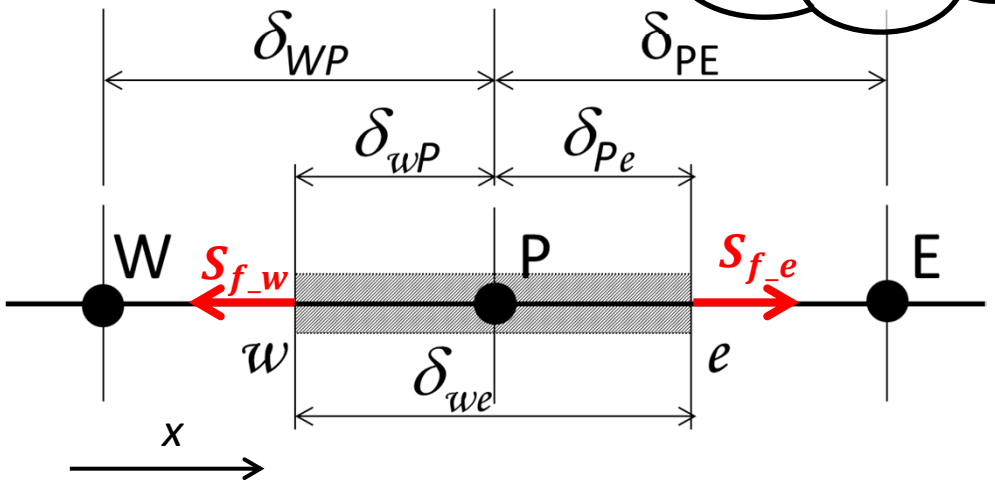
スカラー値の積

$\boldsymbol{S_{f\_w}}$ が逆向きなので，負

前編同様の1次元・直交を想定

snGradScheme

interpolationScheme



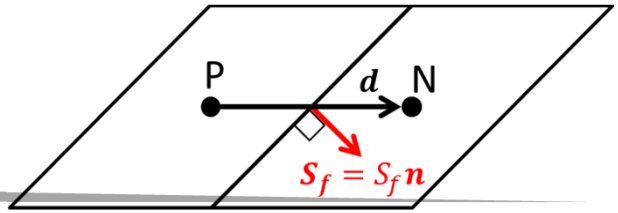
$\boldsymbol{n}$ control volume界面での単位面積ベクトル
大きさ：1，方向：面に垂直

$\sum_f$ control volumeの全ての界面での和

$\boldsymbol{S_f} = S_f \boldsymbol{n}$

control volume界面での面積ベクトル
大きさ：面積S_f，方向：面に垂直

拡散項



$$\int_S (\Gamma \,\mathrm{grad}T)\cdot \boldsymbol{n} dS = \sum_f \Gamma_f \left[(\mathrm{grad}T)_f \cdot \boldsymbol{n}\right] S_f$$

$$= \sum_f \left[\Gamma_f (\mathrm{grad}T)_f \cdot (S_f \boldsymbol{n})\right]$$

$$= \sum_f \left[\Gamma_f (\mathrm{grad}T)_f \cdot \boldsymbol{S_f}\right]$$

面f におけるΓの値(scalar) .面を挟むセル中心の値から補間する.interpolationScheme

面f の面積

面f における$(\mathrm{grad}T)$の値(vector)と単位面ベクトルとの内積＝面f に垂直な成分
snGradScheme

$$\left(\frac{dT}{dx}\right)_f = \frac{T_N - T_O}{\delta_f} = \frac{T_N - T_O}{|C_N - C_O|}$$

T_N temperature of neighbor cell
T_O temperature of owner cell
C_N position vector: center of neighbor cell
C_O position vector: center of owner cell

↑基準:直交の場合. 非直交の場合には, 補正項を追加.

$\sum_f$ control volumeの全ての界面での和

$\boldsymbol{n}$ control volume界面での単位面積ベクトル
大きさ:1, 方向:面に垂直

$$\boldsymbol{n} = \frac{\boldsymbol{S_f}}{S_f}$$

$\boldsymbol{S_f} = S_f \boldsymbol{n}$
control volume界面での面積ベクトル
大きさ:面積S_f , 方向:面に垂直

laplacian( ) 多数のオーバーロード

引数の数と種類によって多数あり

ソースコード　fvm::laplacian(DT, T)

Functions

template<class Type >
tmp< fvMatrix< Type > > **laplacian** (const GeometricField< Type, fvPatchField, volMesh > &vf, const word &nar

template<class Type >
tmp< fvMatrix< Type > > **laplacian** (const GeometricField< Type, fvPatchField, volMesh > &vf)

template<class Type >
tmp< fvMatrix< Type > > **laplacian** (const zero &, const GeometricField< Type, fvPatchField, volMesh > &vf, co

template<class Type >
tmp< fvMatrix< Type > > **laplacian** (const zero &, const GeometricField< Type, fvPatchField, volMesh > &vf)

template<class Type >
tmp< fvMatrix< Type > > **laplacian** (const one &, const GeometricField< Type, fvPatchField, volMesh > &vf, cor

template<class Type >
tmp< fvMatrix< Type > > **laplacian** (const one &, const GeometricField< Type, fvPatchField, volMesh > &vf)

template<class Type , class GType >
tmp< fvMatrix< Type > > **laplacian** (const dimensioned< GType > &**gamma**, const GeometricField< Type, fvPat
> &vf, const word &name)

Foam::fvm Namespace Reference の一部

https://openfoam.com/documentation/guides/latest/api/fvmLaplacian_8H.html

引数の 数と種類が合わない?

- 関数を検索するとき，「引数の数と種類が一致するものがない！」と困惑することがある。
- 引数のデフォルト値を，関数宣言時（*.H）に与える場合，使用時に省略可能であることに注意。
- デフォルト値を記入するのは宣言時だけであり，定義時（*.C）には不要であることに，さらに注意。

src/OpenFOAM/fields/GeometricFields/GeometricField/

GeometricField.H

```
296  GeometricField
297  (
298  const IOobject& io,
299  const Mesh& mesh,
300  const dimensionSet& ds,
301  const word& patchFieldType = PatchField<Type>::calculatedType()
302  );
```

GeometricField.C

```
192 Foam::GeometricField<Type, PatchField, GeoMesh>::GeometricField
193 (
194  const IOobject& io,
195  const Mesh& mesh,
196  const dimensionSet& ds,
197  const word& patchFieldType
198 )
```

laplacian(DT, T)からlaplacianSchemeまで

line 179　スカラ値（dimensiondScalar）DTとスカラー場（volScalarField）Tとが引数；スカラ場Gammaを作成して，すべてにDTの値を入れる．→次のlaplacianを呼び出す

```
tmp<fvMatrix<Type> > laplacian
(   const dimensioned<GType>& gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
```

拡散係数が一様な場合

line303　DTをスカラー場（surfaceScalarField）にしたgammaとスカラー場（volScalarField）Tとが引数；laplacianの名前を付ける→次のlaplacianを呼び出す

```
tmp<fvMatrix<Type> >
laplacian
(   const GeometricField<GType, fvsPatchField, surfaceMesh>& gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
```

拡散係数が場所によって変わる場合

line 271　スカラー場（surfaceScalarField）gammaとスカラー場（volScalarField）Tと名前laplacian(DT,T)が引数；laplacianSchemeを実行する→ディクショナリで設定したスキーム

```
tmp<fvMatrix<Type> >
laplacian
(
    const GeometricField<GType, fvPatchField, volMesh>& gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf,
    const word& name
)
```

上記いずれの場合もここに到達

/src/finiteVolume/finiteVolume/fvm/fvmLaplacian.C

src¥finiteVolume¥finiteVolume¥fvm¥

fvmLaplacian.C

```
template<class Type, class GType>
tmp<fvMatrix<Type> >
laplacian
(
    const dimensioned<GType>& gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    const GeometricField<GType, fvsPatchField, surfaceMesh> Gamma
    (
        IOobject
        (
            gamma.name(),
            vf.instance(),
            vf.mesh(),
            IOobject::NO_READ
        ),
        vf.mesh(),
        gamma
    );

    return fvm::laplacian(Gamma, vf);
}
```

line 179

引数を（ demensionedScalar, GeometricField ）とするlaplacian

引数を（ GeometricField ，GeometricField ）とするlaplacianを実行。

src¥finiteVolume¥finiteVolume¥fvm¥

fvmLaplacian.C

```
template<class Type, class GType>
tmp<fvMatrix<Type> >
laplacian
(
    const GeometricField<GType, fvsPatchField, surfaceMesh>&
gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    return fvm::laplacian
    (
        gamma,
        vf,
        "laplacian(" + gamma.name() + ',' + vf.name() + ')'
    );
}
```

line 238

引数を（ GeometricField ，GeometricField ）とするlaplacian

引数を（GeometricField ，GeometricField，word ）とするlaplacianを再実行。

src¥finiteVolume¥finiteVolume¥fvm ¥

fvmLaplacian.C

```
template<class Type, class GType>
tmp<fvMatrix<Type> >
laplacian
(
    const GeometricField<GType, fvsPatchField, surfaceMesh>&
gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf,
    const word& name
)
{
    return fv::laplacianScheme<Type, GType>::New
    (
        vf.mesh(),
        vf.mesh().laplacianScheme(name)
    )().fvmLaplacian(gamma, vf);
}
```

line 271

RunTimeSelectionの目印

fvSchemesファイルで設定した
laplacianSchemeのfvmLaplacianが実行される。
laplacian(Dt,T) Gauss linear corrected;

src¥finiteVolume¥finiteVolume¥laplacianSchemes¥gaussLaplacianScheme

gaussLaplacianScheme.C

```
template<class Type, class GType>  tmp<fvMatrix<Type> >  gaussLaplacianScheme<Type,
GType>::fvmLaplacian
(
    const GeometricField<GType, fvsPatchField, surfaceMesh>& gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    const fvMesh& mesh = this->mesh();
    const surfaceVectorField Sn(mesh.Sf()/mesh.magSf());
    const surfaceVectorField SfGamma(mesh.Sf() & gamma);
    const GeometricField<scalar, fvsPatchField, surfaceMesh> SfGammaSn ( SfGamma & Sn );
    const surfaceVectorField SfGammaCorr(SfGamma - SfGammaSn*Sn);
    tmp<fvMatrix<Type> > tfvm = fvmLaplacianUncorrected
    (
        SfGammaSn,
        this->tsnGradScheme_().deltaCoeffs(vf),
        vf
    );
// 中略　非直交性の補正など
    return tfvm;
}
```

line 154

SfGamma の面に垂直な成分（大きさ）

fvmLaplacianUncorrected を呼んで tfvm を作成。ここで行列を操作。

src¥finiteVolume¥finiteVolume¥laplacianSchemes¥gaussLaplacianScheme

gaussLaplacianScheme.C

line 44

```
template<class Type, class GType>
tmp<fvMatrix<Type> > gaussLaplacianScheme<Type, GType>::fvmLaplacianUncorrected
(
    const surfaceScalarField& gammaMagSf,
    const surfaceScalarField& deltaCoeffs,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    tmp<fvMatrix<Type> > tfvm
    (
        new fvMatrix<Type>
        (
            vf,
            deltaCoeffs.dimensions()*gammaMagSf.dimensions()*vf.dimensions()
        )
    );
    fvMatrix<Type>& fvm = tfvm.ref();
    fvm.upper() = deltaCoeffs.primitiveField()*gammaMagSf.primitiveField();
    fvm.negSumDiag();
    //境界での処理：省略
    return tfvm;
}
```

行列の上三角に(セル間距離の逆数×ガンマΓ×セル面の面積)を入れる。$\Gamma S/\delta$
対称性から,upperのみに代入。自動的にlowerにも入る。

行列の対角成分に 周辺の係数を加える。(次のスライド)
LduMatrix::negSumDiag

src¥OpenFOAM¥matrices¥LduMatrix¥LduMatrix¥

lduMatrixOperations.C

line 50

```
void Foam::lduMatrix::negSumDiag()
{
    const scalarField& Lower = const_cast<const lduMatrix&>(*this).lower();
    const scalarField& Upper = const_cast<const lduMatrix&>(*this).upper();
    scalarField& Diag = diag();

    const labelUList& l = lduAddr().lowerAddr();
    const labelUList& u = lduAddr().upperAddr();

    for (label face=0; face<l.size(); face++)
    {
        Diag[l[face]] -= Lower[face];
        Diag[u[face]] -= Upper[face];
    }
}
```

Diag　対角成分
Lower　下三角成分
Upper 上三角成分
l　下三角での場所を指す
u　上三角での場所を指す

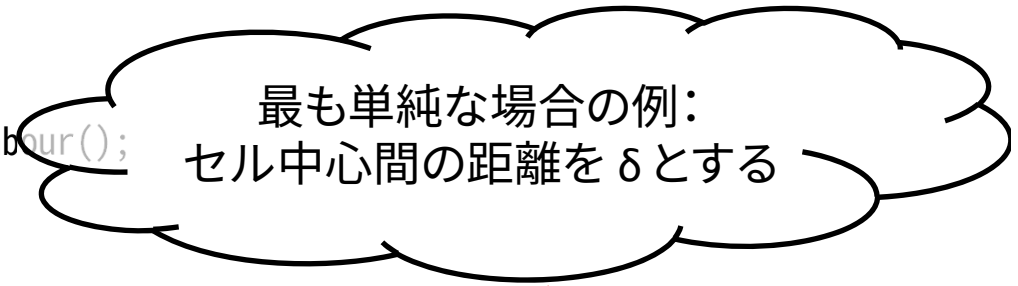
$a_P = -(-a_E) + -(-a_W)$

対角成分と下三角，上三角とでは，符号が逆になることに注意。

src¥finiteVolume¥interpolation¥surfaceInterpolation¥surfaceInterpolation¥

surfaceInterpolation.C

```
void Foam::surfaceInterpolation::makeDeltaCoeffs() const
{
    //前略
    deltaCoeffs_ = new surfaceScalarField
    (
        IOobject
        (
            "deltaCoeffs", mesh_.pointsInstance(), mesh_,
            IOobject::NO_READ, IOobject::NO_WRITE, false // Do not register
        ),
        mesh_, dimless/dimLength
    );
    surfaceScalarField& DeltaCoeffs = *deltaCoeffs_;
    // Set local references to mesh data
    const volVectorField& C = mesh_.C();
    const labelUList& owner = mesh_.owner();
    const labelUList& neighbour = mesh_.neighbour();
    forAll(owner, facei)
    {
        deltaCoeffs[facei] = 1.0/mag(C[neighbour[facei]] - C[owner[facei]]);
    }
    //境界での処理：省略
```



行列

$$\begin{bmatrix} a_{p1} & -a_{E1} & 0 & 0 \\ -a_{W2} & a_{P2} & -a_{E2} & 0 \\ 0 & -a_{W3} & a_{P3} & -a_{E3} \\ 0 & 0 & -a_{W4} & a_{P4} \end{bmatrix} =$$

$$\begin{bmatrix} S_{P1} & 0 & 0 & 0 \\ 0 & S_{P2} & 0 & 0 \\ 0 & 0 & S_{P3} & 0 \\ 0 & 0 & 0 & S_{P4} \end{bmatrix} - \begin{bmatrix} -a_{E1} & a_{E1} & 0 & 0 \\ a_{W2} & -a_{W2} - a_{E2} & a_{E2} & 0 \\ 0 & a_{W3} & -a_{W3} - a_{E3} & a_{E3} \\ 0 & 0 & a_{W4} & -a_{W4} \end{bmatrix}$$

設定済み
非定常項

設定済み
拡散項

$$\begin{bmatrix} S_{BC0} + S_{u1} \\ S_{u2} \\ S_{u3} \\ S_{BC5} + S_{u4} \end{bmatrix} = \begin{bmatrix} S_{u1\_ddt} \\ S_{u2\_ddt} \\ S_{u3\_ddt} \\ S_{u4\_ddt} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} S_{BC0} \\ 0 \\ 0 \\ S_{BC5} \end{bmatrix}$$

設定済み
非定常項

今回は考えない：
＊境界条件の項はsolve()の中で追加される。
＊拡散項の非直交性補正でも値が入る

さらに詳しく・・・

- 今回は，手作業計算と同じように，セル中心間を結ぶベクトルが面と直交する状態を想定して説明した。
- 実際には，これらが直交しない場合もある。
- その時には，面での勾配，面での拡散係数の値の算出時に，非直交性の影響を考慮する必要がある。
- 考慮の方法は，補間スキームに依存する。
- 今回は，境界条件の寄与については考えていない。実際には，境界条件に応じた値が行列に追加される。

ソースコードを読み解くために

- 変数のタイプ(クラス)を意識
  - volScalarField, dimensionedScalar？など
- Slow and steady wins the race
  - 少しずつ,理解を深める
  - 小さな部分の積み重ね
  - 繰り返す,繰り返す,繰り返す
- 基礎を学習
  - 理論とソースの両方を学ぶ

参考資料

- OpenFOAM Programmers Guide, User Guide，ソースコード
- Imperial College 博士論文など
  - Hrvoje Jasak, Henrik Rusche, Franjo Juretic などなど
  - http://powerlab.fsb.hr/ped/kturbo/OpenFOAM/docs/
- PENGUINITIS サイト(OpenFOAMたんけんたい)
  - http://penguinitis.g1.xrea.com/study/OpenFOAM/tankentai/index.html
- 野崎さんSlideShare
  - http://www.slideshare.net/fumiyanozaki96/
  - OpenFOAM 空間の離散化と係数行列の取り扱い
- http://openfoamwiki.net/
- http://www.cfd-online.com

追記：runTimeSelectionの理解に向けて

C++ のマクロ

ロベールのC++教室　第28章
http://www7b.biglobe.ne.jp/~robe/cpphtml/html01/cpp01028.html

マクロは，

- テキストを差し込むもの，
- 引数もとれる。

書式

#define　<マクロ名>　<差し込むテキスト>

注意：改行したい時は、行の終わりに バックスラッシュ(＼ or ¥) を付ける

トークン連結演算子 (##)

https://docs.microsoft.com/ja-jp/cpp/preprocessor/token-pasting-operator-hash-hash?view=vs-2017

- 二重シャープ記号または"トークン連結演算子 (##)、両方のオブジェクトのような関数に似たマクロで使用されて、「マージ」演算子と呼ばれることがあります。この演算子を使用すると、別々のトークンを 1 つのトークンに結合できます。そのため、これをマクロ定義の最初または最後のトークンにすることはできません。

- マクロ定義の仮パラメーターの前または後ろにトークン連結演算子がある場合、仮パラメーターは展開されていない実引数に即座に置き換えられます。引数に対するマクロ展開が置換の前に行われることはありません。

トークン連結演算子 (##)

https://docs.microsoft.com/ja-jp/cpp/preprocessor/token-pasting-operator-hash-hash?view=vs-2017

```
#define SOME_MACRO(TYPE, arg) ¥
typedef vector<TYPE> vector##arg;
```

- 例えば，上記のマクロを定義し，コード中に

```
SOME_MACRO(int, INT)
```

と書かれていると，そこは下記に置き換わる。

```
typedef vector<int> vectorINT;
```

src/OpenFOAM/db/runTimeSelection/construction/

runTimeSelectionTables.H

Detailed Description

Macros to ease declaration of run-time selection tables.

declareRunTimeSelectionTable is used to create a run-time selection table for a base-class which holds constructor pointers on the table.

declareRunTimeNewSelectionTable is used to create a run-time selection table for a derived-class which holds "New" pointers on the table.

Definition in file runTimeSelectionTables.H.

src/finiteVolume/finiteVolume/ddtSchemes/ddtScheme/

ddtScheme.H

マクロ
src/OpenFOAM/db/runTimeSelection/construction/runTimeSelectionTables.H

```
declareRunTimeSelectionTable
(
    tmp,
    ddtScheme,
    Istream,
    (const fvMesh& mesh, Istream& schemeData),
    (mesh, schemeData)
);
```

src/OpenFOAM/db/runTimeSelection/construction/

runTimeSelectionTables.H

tmp, ddtScheme, Istream, (const fvMesh& mesh, Istream& schemeData), (mesh, schemeData)

```
//- Declare a run-time selection
#define declareRunTimeSelectionTable(autoPtr,baseType,argNames,argList,parList) ¥
                                                                                ¥
    /* Construct from argList function pointer type */                          ¥
    typedef autoPtr<baseType> (*argNames##ConstructorPtr)argList;               ¥
                                                                                ¥
    /* Construct from argList function table type */                            ¥
    typedef HashTable<argNames##ConstructorPtr, word, string::hash>             ¥
        argNames##ConstructorTable;                                             ¥
                                                                                ¥
中略                                                                            ¥
    /* Class to add constructor from argList to table */                        ¥
    template<class baseType##Type>                                              ¥
    class add##argNames##ConstructorToTable                                     ¥
    {                                                                           ¥
    public:                                                                     ¥
```

IstreamConstructorPtr
IstreamConstructorTable

src/finiteVolume/finiteVolume/ddtSchemes/ddtScheme/

ddtScheme.C

line 44

```
// * * * * * Selectors * * * * * * * * * * * * * //
template<class Type>
tmp<ddtScheme<Type>> ddtScheme<Type>::New
(
    const fvMesh& mesh,
    Istream& schemeData
)
{
    if (fv::debug)
    {
        InfoInFunction << "Constructing ddtScheme<Type>"
<< endl;
    }

    if (schemeData.eof())
    {
        FatalIOErrorInFunction
        (
            schemeData
        )   << "Ddt scheme not specified" << endl << endl
            << "Valid ddt schemes are :" << endl
            << IstreamConstructorTablePtr_->sortedToc()
            << exit(FatalIOError);
    }

    const word schemeName(schemeData);

    typename IstreamConstructorTable::iterator cstrIter =
        IstreamConstructorTablePtr_->find(schemeName);

    if (cstrIter == IstreamConstructorTablePtr_->end())
    {
        FatalIOErrorInFunction
        (
            schemeData
        )   << "Unknown ddt scheme " << schemeName << nl
<< nl
            << "Valid ddt schemes are :" << endl
            << IstreamConstructorTablePtr_->sortedToc()
            << exit(FatalIOError);
    }

    return cstrIter()(mesh, schemeData);
}
```

設定ミス時に，よく見かけるメッセージ。
選択可能な候補リストとともに。

src/finiteVolume/finiteVolume/ddtSchemes/ddtScheme/

ddtSchemes.C

ここから，さらに
#include "runTimeSelectionTables.H
src/OpenFOAM/db/runTimeSelection/construction/

```
#include "ddtScheme.H"

// Define the constructor function hash tables

defineTemplateRunTimeSelectionTable(ddtScheme<scalar>, Istream);
defineTemplateRunTimeSelectionTable(ddtScheme<vector>, Istream);
defineTemplateRunTimeSelectionTable(ddtScheme<sphericalTensor>, Istream);
defineTemplateRunTimeSelectionTable(ddtScheme<symmTensor>, Istream);
defineTemplateRunTimeSelectionTable(ddtScheme<tensor>, Istream);
```

src/OpenFOAM/db/runTimeSelection/construction/

runTimeSelectionTables.H

defineTemplateRunTimeSelectionTable(ddtScheme<scalar>, Istream);

```
//- Define run-time selection table for template classes
//  use when baseType doesn't need a template argument (eg, is a typedef)
#define defineTemplateRunTimeSelectionTable(baseType,argNames)                 ¥
                                                                               ¥
    template<>                                                                 ¥
    defineRunTimeSelectionTablePtr(baseType,argNames);                         ¥
    template<>                                                                 ¥
    defineRunTimeSelectionTableConstructor(baseType,argNames);                 ¥
    template<>                                                                 ¥
    defineRunTimeSelectionTableDestructor(baseType,argNames)
————
// Create pointer to hash-table of functions
#define defineRunTimeSelectionTablePtr(baseType,argNames)                      ¥
                                                                               ¥
    /* Define the constructor function table */                                ¥
    baseType::argNames##ConstructorTable*                                      ¥
        baseType::argNames##ConstructorTablePtr_ = nullptr
```

ddtScheme<scalar>

Istream

ddtScheme<scalar>::IstreamConstructorTablePtr_

src/finiteVolume/Make/filesの一部

```
ddtSchemes = finiteVolume/ddtSchemes
$(ddtSchemes)/ddtScheme/ddtSchemes.C
$(ddtSchemes)/steadyStateDdtScheme/steadyStateDdtSchemes.C
$(ddtSchemes)/EulerDdtScheme/EulerDdtSchemes.C
$(ddtSchemes)/CoEulerDdtScheme/CoEulerDdtSchemes.C
```

Eulerスキームは？

EulerDdtSchemes.C

makeFvDdtScheme(EulerDdtScheme)

makeFvDdtSchemeはddtScheme.Hで定義されたマクロ。

ddtスキームのrunTimeSelectionTableに，EulerDdtSchemeが追加される。

EulerDdtSchemeのTypeNameはEulerである。（EulerDdtScheme.Hで定義している。）

他の runTimeSelectionTable

- ライブラリ・機能ごとに，runTimeSelectionTableが宣言されている。
  - declareRunTimeSelectionTable
  - abstract base classが担当することが多い?
- 各クラスをそのtableに追加するためのマクロが存在する
  - addToRunTimeSelectionTable
  - モデル名s.Cで呼び出すことがある
  - 複数のモデルを一括して登録する場合がある(乱流)
- 新クラス作成時には，その仕組みを使うとよい