オープンCAE勉強会@富山（第45回）
2016年7月23日
2016年7月27日改訂

rhoCentralFoamを用いた衝撃波シミュレーション

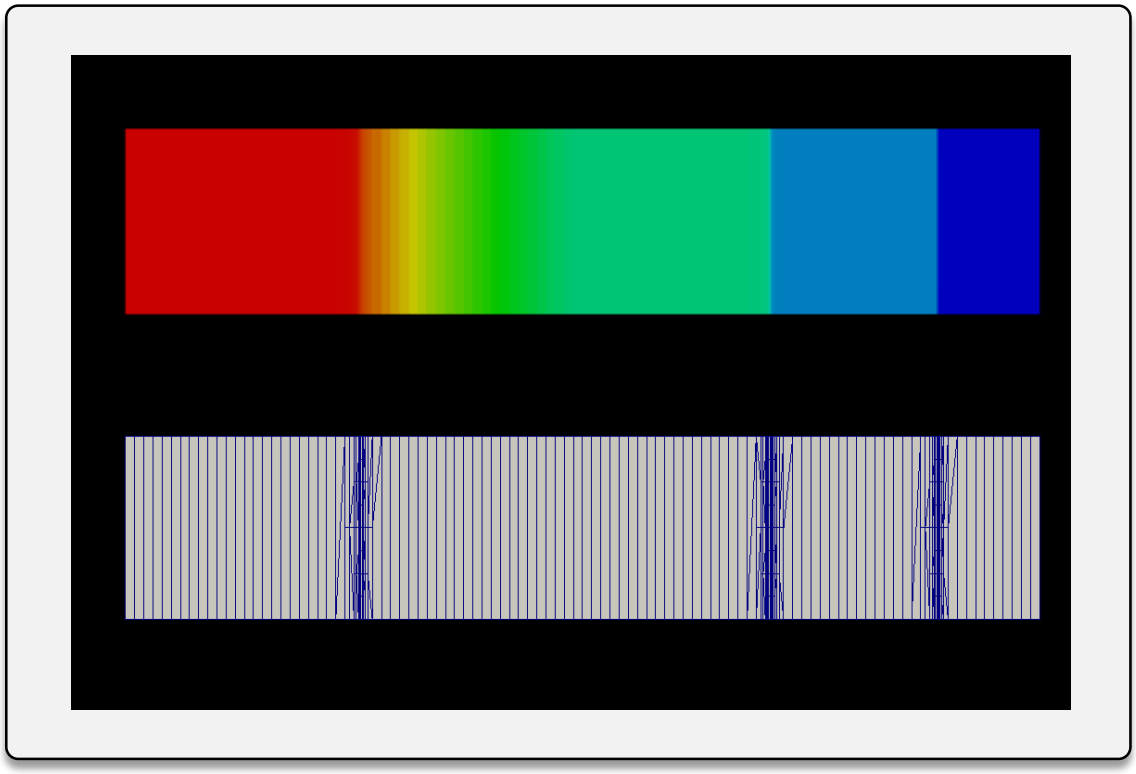
- dynamicRefineFvMeshの利用 その２ -

中山 勝之 （オープンCAE勉強会＠富山）

前回の発表内容(第44回オープンCAE勉強会@富山)

dynamicRefineFvMeshをrhoCentralDyMFoamで使用してみた

基準フィールド値のフィールドを設定するために、ソースファイルの変更を行い、
改造ソルバーを作成した



今回の目標

dynamicRefineFvMeshを
rhoCentralDyMFoamソルバーのソースコード改造以外の方法で使用する

dynamicRefineFvMeshを利用するには

- rhoCentralDyMFoamは既に用意されているためdynamicRefineFvMeshは利用可能
- 使用するソルバーに対して細分化させるための基準フィールド(volScalarField)を指定する必要あり
- 衝撃波の位置でメッシュを細分化させるには、デフォルトで出力されない密度、圧力変化のフィールド値が必要

前回は

作業 1　createFields.Hに基準フィールド値の宣言を記述

作業 2　rhoCentralDyMFoam.Cに基準フィールド値の計算式を記述

という方法をとり、ソースコードを改造することで目的を達成した

ソースコード改造なしにdynamicRefineFvMeshを利用するには…

Function Objects 機能を使用することで、ソースコードの改造なしに
新たなフィールドを作成することが可能

function object Type **coded** (コードを実行する)
を用いて基準フィールド値を作成

その他に
calcFvcGrad (フィールド勾配を計算)
calcMag (フィールドの絶対値を計算)
を利用してみた

参考URL

http://openfoam.org/release/2-0-0/run-time-control-code-compilation/
http://cfd.direct/openfoam/user-guide/v3-function-objects/
http://www.geocities.jp/penguinitis2002/study/OpenFOAM/function_objects.html

OpenFOAMの環境について

OS : Ubuntu MATE 16.04 LTS (64bit)

OpenFOAM Version : 3.0.x, v1606+

ケースファイルの作成

- $FOAM_TUTORIALS/compressible/rhoCentralFoam/shockTubeをベースに作成

ケースファイルをコピーする

```
cp -rf $FOAM_TUTORIALS/compressible/rhoCentralFoam/shockTube $WM_PROJECT_USER_DIR/ ⏎
```

作業 １　controlDictディクショナリにfunctionObjectの記述を追加

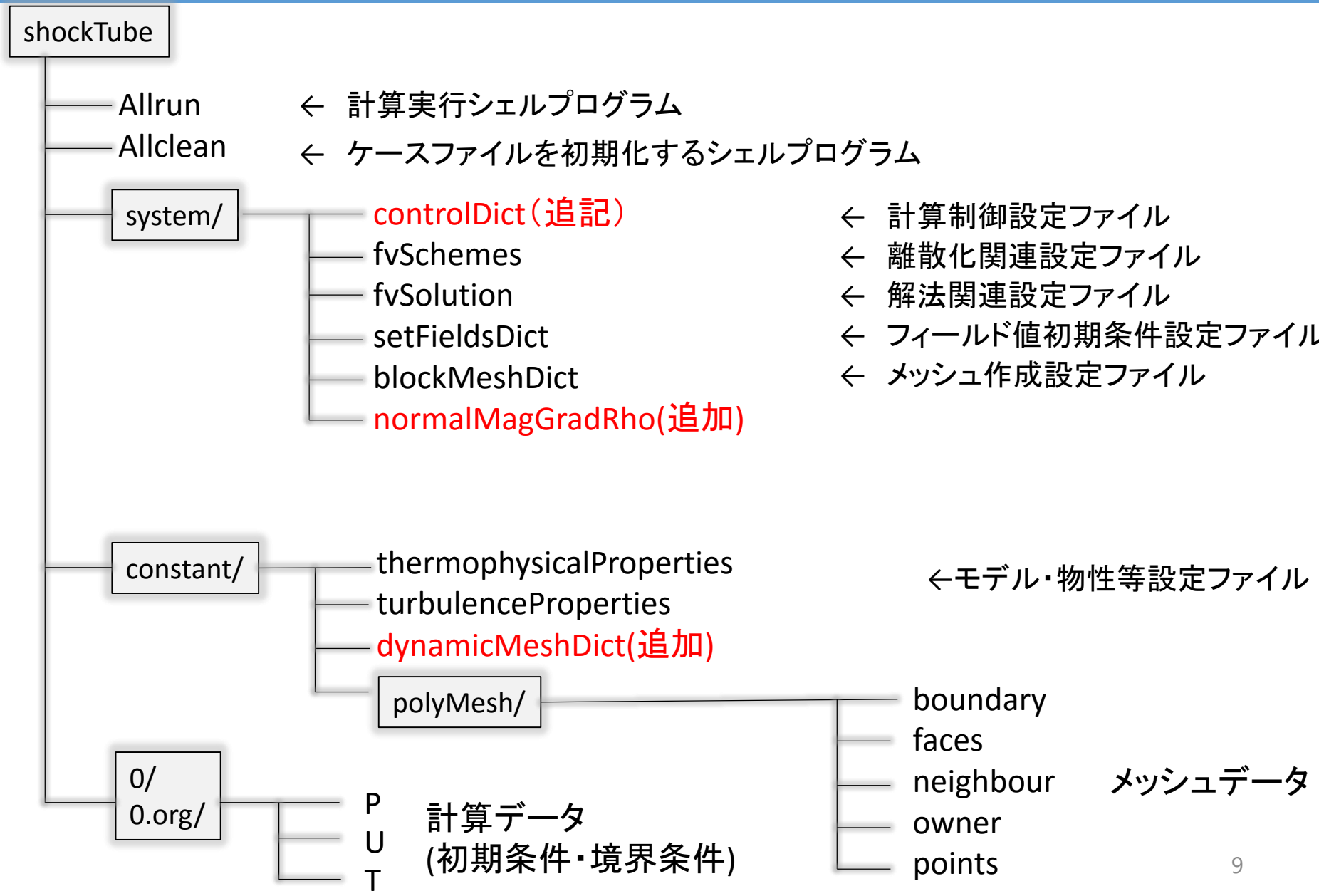
作業 ２　dynamicMeshDictディクショナリを作成

作業 ３　境界面emptyの境界タイプを変更
(blockMeshDict, 0.org/p,U,Tの書き換え)

作業 1

controlDictディクショナリにfunctionObjectの記述を追加

ファイル構成 (ケースファイル)



controlDictディクショナリにfunctionObjectの記述を追加

system/controlDict

```
.....................................
functions
{
// Calculate and store mag(grad(rho)) for refinement
   #include “normalMagGradRho"
}
.....................................
```

system/normalMagGradRho

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       dictionary;
   location    "system";
   object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
 gradRho
   {
      functionObjectLibs ("libFVFunctionObjects.so");
      type calcFvcGrad;
      fieldName rho;
      resultName gradRho;
      outputControl outputTime;
      outputInterval 1;
   }

   magGradRho
   {
      functionObjectLibs ("libFVFunctionObjects.so");
      type calcMag;
      fieldName gradRho;
      resultName magGradRho;
      outputControl outputTime;
      outputInterval 1;
   }
```

ファイルnormalMagGradRhoを読み込む仕組みでcontrolDictに記述する

```
normalMagGradRho
  {
      functionObjectLibs ("libutilityFunctionObjects.so");
     type coded;
      redirectType indicatorField;
     code
     #{
        const volScalarField& magGradRho0 =
                        mesh().lookupObject<volScalarField>("magGradRho");
        static autoPtr<volScalarField> pField;
        if(!pField.valid())
        {
           Info << "Creating normalMagGradRho" << nl;
           pField.set
           (
                 new volScalarField
                 (
                    IOobject
                    (
                       "normalMagGradRho",
                        mesh().time().timeName(),
                        magGradRho0.mesh(),
                        IOobject::NO_READ,
                        IOobject::AUTO_WRITE
                    ),
                    magGradRho0/max(magGradRho0)
                 )
           );
        }
        volScalarField &normalMagGradRho = pField();
        normalMagGradRho.checkIn();
        normalMagGradRho = magGradRho0/max(magGradRho0);
     #};
  }
```

ファイルnormalMagGradRhoの解説（1）

方針

1. volVectorField $\nabla\rho_{iCell}$を作成 : functionObject calcFvcGradを使用
2. volScalarField|$\nabla\rho_{iCell}$|を作成 : functionObject calcMag を使用
3. volScalarField “normalMagGradRho”を作成 : functionObject coded を使用

```
gradRho
  {
    functionObjectLibs ("libFVFunctionObjects.so");
    type calcFvcGrad;
    fieldName rho;
    resultName gradRho;
    outputControl outputTime;
    outputInterval 1;
  }
magGradRho
  {
    functionObjectLibs ("libFVFunctionObjects.so");
    type calcMag;
    fieldName gradRho;
    resultName magGradRho;
    outputControl outputTime;
    outputInterval 1;
  }
```

libFVFunctionObjects.so
calcFvcGrad (フィールド勾配を計算)
calcMag (フィールドの絶対値を計算)
を使用する際に必要

fieldName : 対象とするフィールド名
resultName: 作成するフィールド名

outputControl : 出力方法の設定
adjustableTime
clockTime
cpuTime
outputTime
runTime
timeStep から選択

outputInterval :出力間隔

ファイルnormalMagGradRhoの解説（2）

```
normalMagGradRho
{
    functionObjectLibs ("libutilityFunctionObjects.so");
    type coded;
    redirectType indicatorField;
    code
    #{
        const volScalarField& magGradRho0 =
                    mesh().lookupObject<volScalarField>("magGradRho");
        static autoPtr<volScalarField> pField;
        if(!pField.valid())
        {
            Info << "Creating normalMagGradRho" << nl;
            pField.set
            (
                new volScalarField
                (
                    IOobject
                    (
                        "normalMagGradRho",
                        mesh().time().timeName(),
                        magGradRho0.mesh(),
                        IOobject::NO_READ,
                        IOobject::AUTO_WRITE
                    ),
                    magGradRho0/max(magGradRho0)
                )
            );
        }
        volScalarField &normalMagGradRho = pField();
        normalMagGradRho.checkIn();
        normalMagGradRho = magGradRho0/max(magGradRho0);
    #};
}
```

libutilityFunctionObjects.so

codedを使用する際に必要

redirectType : 任意の名前を設定

記述方法

```
code
#{
    コードの記述
#};
```

時刻ディレクトリに出力させる
フィールドnormalMagGradRho を作成

正規化された密度勾配を計算

$$\text{normalMagGradRho} = \frac{|\nabla \rho_{iCell}|}{\max(|\nabla \rho_{iCell}|)}$$

作業 2

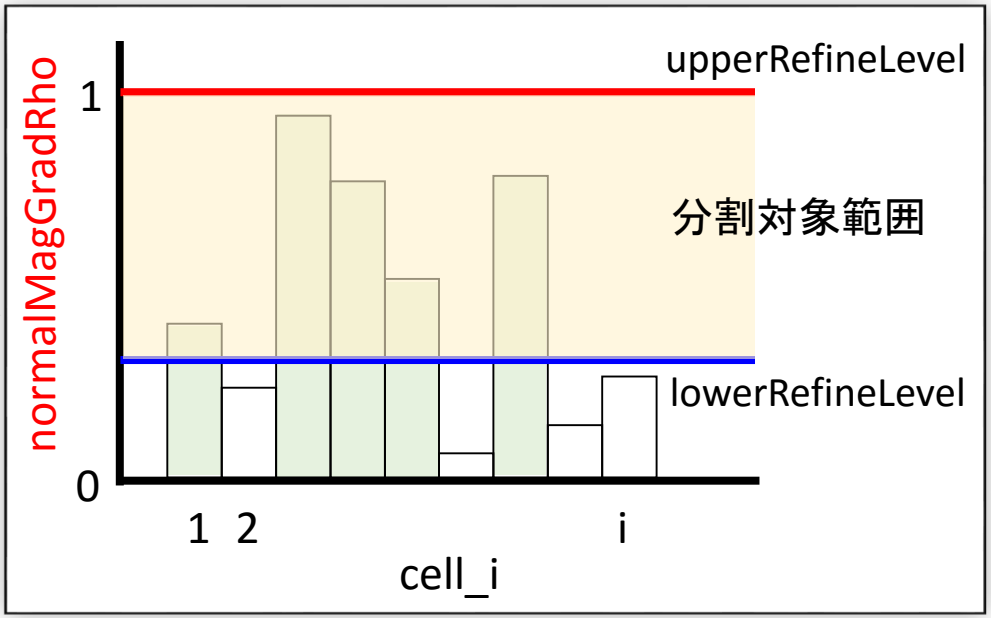
dynamicMeshDictディクショナリを作成

dynamicMeshDictの追加（1）

constant/dynamicMeshDict

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
dynamicFvMesh   dynamicRefineFvMesh;

dynamicRefineFvMeshCoeffs
{
    // How often to refine
    refineInterval  2; //1では計算時にエラーがでた
    // Field to be refinement on
    field           normalMagGradRho;
    // Refine field inbetween lower..upper
    lowerRefineLevel 0.2;
    upperRefineLevel 1.0;
    // If value < unrefineLevel unrefine
    unrefineLevel   10;
    // Have slower than 2:1 refinement
    nBufferLayers   2;
    // Refine cells only up to maxRefinement levels
    maxRefinement   3;
    // Stop refinement if maxCells reached
    maxCells        200000;
    // Flux field and corresponding velocity field. Fluxes on changed
    // faces get recalculated by interpolating the velocity. Use 'none'
    // on surfaceScalarFields that do not need to be reinterpolated.
    correctFluxes
    (
        (phi none)
        (neg none)
        (pos none)
);
    // Write the refinement level as a volScalarField
    dumpLevel       true;
}
```

$$\text{normalMagGradRho} = \frac{|\nabla \rho_{icell}|}{\max(|\nabla \rho_{icell}|)}$$



// If value < unrefineLevel unrefine
unrefineLevel 10;
細分化されたセルが粗雑化の対象となるセルの
フィールド値の基準値を入力
例えば
フィールド値 normalMagGradRhoの最大値は1であるので、
unrefineLevel 10に設定すると、全セルが粗雑化の対象となる
また
unrefineLevel 0に設定すると、一度細分化されたセルは粗雑化
の対象から外れる

dynamicMeshDictの追加（2）

constant/dynamicMeshDict

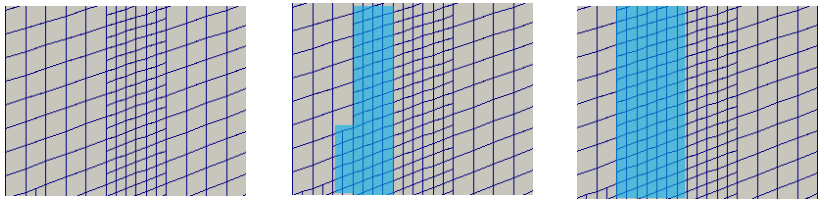
```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
dynamicFvMesh   dynamicRefineFvMesh;

dynamicRefineFvMeshCoeffs
{
    // How often to refine
    refineInterval  2; //1では計算時にエラーがでた
    // Field to be refinement on
    field           normalMagGradRho;
    // Refine field inbetween lower..upper
    lowerRefineLevel 0.2;
    upperRefineLevel 1.0;
    // If value < unrefineLevel unrefine
    unrefineLevel   10;
    // Have slower than 2:1 refinement
    nBufferLayers   2;
    // Refine cells only up to maxRefinement levels
    maxRefinement   3;
    // Stop refinement if maxCells reached
    maxCells        200000;
    // Flux field and corresponding velocity field. Fluxes on changed
    // faces get recalculated by interpolating the velocity. Use 'none'
    // on surfaceScalarFields that do not need to be reinterpolated.
    correctFluxes
    (
        (phi none)
        (neg none)
        (pos none)
    );
    // Write the refinement level as a volScalarField
    dumpLevel       true;
}
```

細分化する計算ステップ間隔
1 を設定すると毎ステップ実行
非定常現象の場合小さな値に設定すると良い

細分化セルの上限の制御は
最大分割レベル(maxRefinement)の設定
最大セル数(maxCells)の設定で行う
nBufferLayersは細分化セルのバッファーを設定

ex.

maxRefinement 1
nBufferLayers 0　　4　　8



correctFluxes
再補完するフィールドを設定する
noneを指定することで再補完しない

dumpLevel
細分化レベルを出力
時刻ディレクトリにファイルcellLevelを作成

作業 3

境界面emptyの境界タイプを変更

境界タイプの変更（1）

赤字の部分を変更

system/controlDict

```
.....

application    rhoCentralDyMFoam

.....
```

境界条件はすべての境界面でslipを選択

0.org/p

```
.....
boundaryField
{
   sides
   {
      type         slip;
   }

   empty
   {
      type         slip;
   }
}
.....
```

0.org/U

```
.....
boundaryField
{
   sides
   {
      type         slip;
   }

   empty
   {
      type         slip;
   }
}
.....
```

0.org/T

```
.....
boundaryField
{
   sides
   {
      type         slip;
   }

   empty
   {
      type         slip;
   }
}
.....
```

境界タイプの変更（2）

system/blockMeshDict

```
.....

boundary
(
    sides
    {
        type patch;
        faces
        (
            (1 2 6 5)
            (0 4 7 3)
        );
    }
    empty
    {
        type patch;
        faces
        (
            (0 1 5 4)
            (5 6 7 4)
            (3 7 6 2)
            (0 3 2 1)
        );
    }
);
.....
```

emptyの境界タイプをpatchに変更

ケースファイルの実行

```
cd $WM_PROJECT_USER_DIR/shockTube ⏎
./Allrun ⏎
```

Allrun

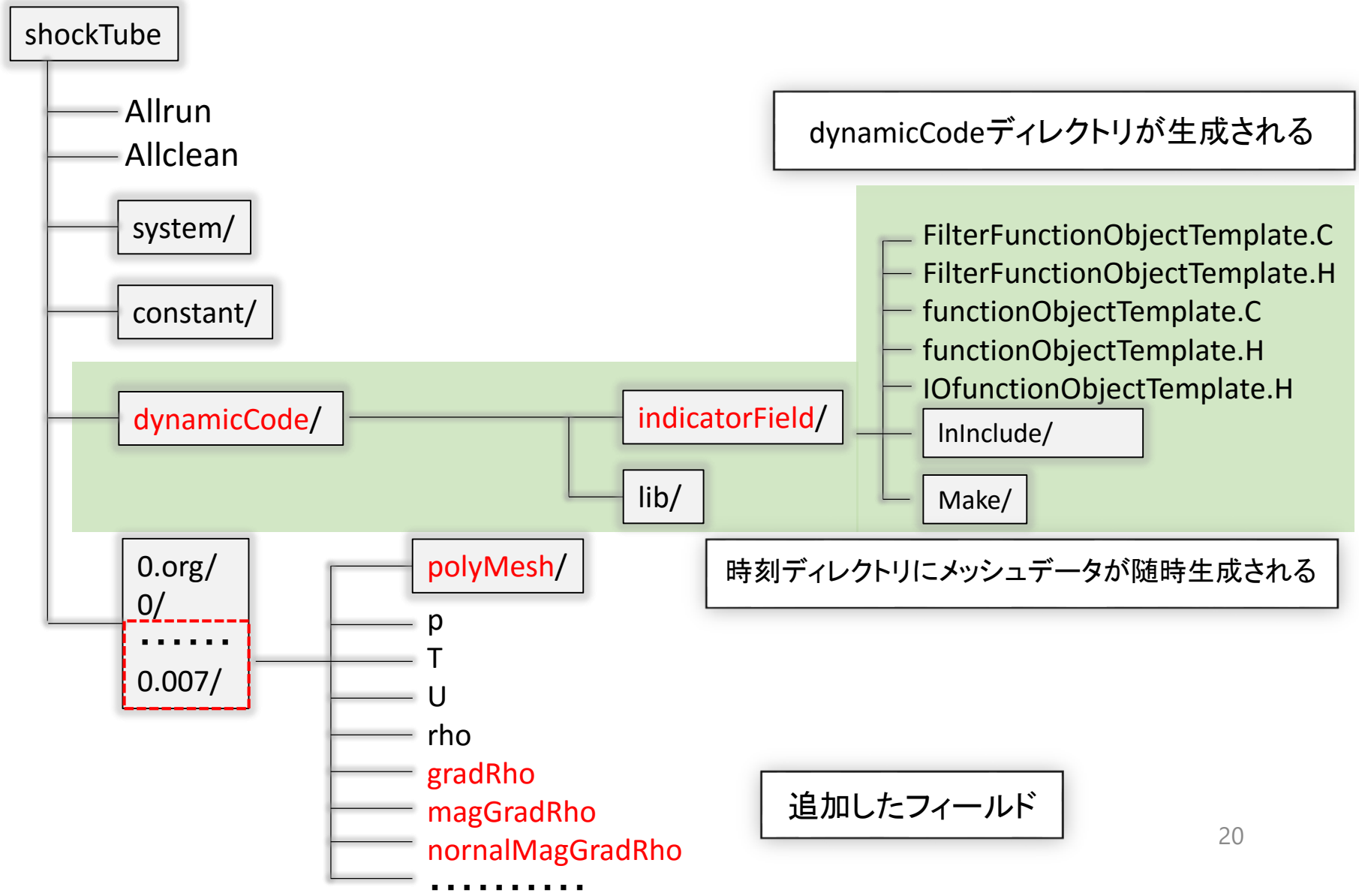
```
#!/bin/sh
cd ${0%/*} || exit 1    # Run from this directory

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

runApplication blockMesh
runApplication setFields
runApplication `getApplication`
```

コマンドrunApplication
変数　getApplication
を使用するためにRunFunctionsの設定を読み込む

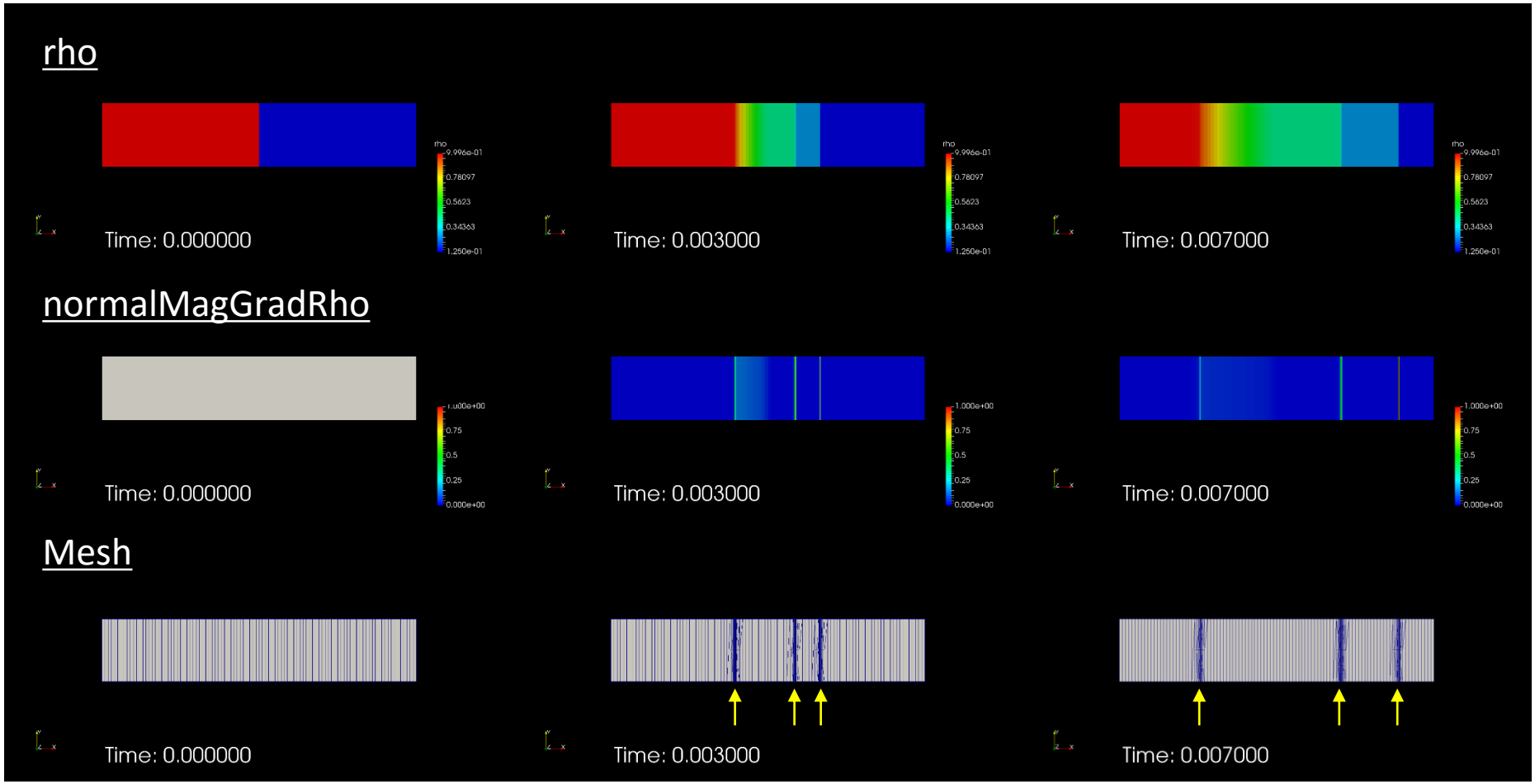
runApplication command ⏎は
command > log.command 2>&1 ⏎
を意味する
log.commandが既に存在する場合はコマンドが実行されないので、予め削除する

getApplicationは
system/controlDictに書かれている
applicationの変数（ソルバー名）を取得する

計算実行後のファイル構成 (ケースファイル)



計算結果



前回の結果と変わらないことを確認した

まとめ

dynamicRefineFvMeshをrhoCentralDyMFoamで使用するためにfunctionObjects機能を利用した

参考文献

以下のHPを参考にさせていただきました。ありがとうございます。

OpenFOAM User Guide: 6.2 Function Objects

http://cfd.direct/openfoam/user-guide/v3-function-objects/

PENGUINTIS: function objects

http://www.geocities.jp/penguinitis2002/study/OpenFOAM/function_objects.html

OpenFOAM 2.0.0: Run-time Control

http://openfoam.org/release/2-0-0/run-time-control-code-compilation/

ディスカッションでの内容（1）

Q1 functionObjectを使用する利点は？

ケースファイルの記述のみで済む点
別の言い方ではソースコードを変更しないで済む点

- ソルバーのソースファイルに記述する方法の場合、OpenFOAMのバージョンによってその都度rhoCentralDyMFoamの改造を行う必要があるため（本方法はバージョンの依存率は基本的には低い）
- 他者に利用してもらうとき、ソルバーのソースファイルに記述する方法の場合、ソルバーをコンパイルしてもらう必要があり、そのことで敷居を高く感じる方がいるかもしれないので

ディスカッションでの内容（2）

Q2 dynamicMeshDictの記述で“refineInterval 1”が設定できない理由は？

rhoCentralDyMFoam.Cの一部

```
.....

while (runTime.run())
  {
    #include "readTimeControls.H"
    #include "setDeltaT.H"

    runTime++;

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

    // Do any mesh changes
    mesh.update();

.....
.....

    runTime.write();


  }
.....
```

- functionObjectsの呼び出されるタイミングが、**runTime.write();内**で実行されている

したがって1回目のループでは、まだnormalMagGradRhoのフィールドが作成されていないので判定不能となる

メッシュ細分化のタイミング

計算結果の出力(controlDictで管理)
functionObjects実行のタイミング

$FOAM_SRC/OpenFOAM/db/Time/Time.C内に記述を発見（詳細は次回発表にて）