interPhaseChangeFoamの調査
(探検隊入隊準備)

~コラボに向けて~

宮部 正洋

山本さんの発表資料（第25，26回勉強会@関西）を参考にさせて頂きました．

ありがとうございます．

春日さんのHPを参考にさせて頂きました．

ありがとうございます．

コードの見方については冨原さんにアドバイス頂いております．

ありがとうございます．

interFoam

2012年11月11日

はじめに

interFoam のコードを見てみましょう。

使用バージョン

OpenFOAM 2.1.1

OF-2.1.1からOF-2.2.x で少々記述が変わっています．

Volume Of Fluid (VOF) 法

interFoam は Volume Of Fluid (VOF) 法による二相流ソルバーです。VOF 法では、相の区別をそれぞれの相の体積分率 (0~1) で扱います。
二相だと、ひとつめの相の体積分率を $\alpha 1$ とすると、ふたつめの相の体積分率は $1 - \alpha 1$ で計算できます。

相の体積分率の挙動は、輸送方程式で表します。

```
Dα1/Dt = 0
```

運動方程式は単相の式と同じものをふつうに解きますが、密度と粘性をふたつの流体のそれから求めます。

```
ρ = α1 ρ1 + (1 - α1) ρ2
μ = α1 μ1 + (1 - α1) μ2
```

運動方程式には表面張力を加えます。

```
fs = σ κ n （表面張力）
n = ∇α1/|∇α1| （界面の法線方向ベクトル）
κ = ∇・n （界面の曲率）
```

Interface Compression

interPhaseChangeFoam.C

OF-2.2.x を使用

interFoam.C とほとんど同じ

(違い：#include “alphaCourant No.H” がないだけ)

Level Set を組み込むには
少なくとも②,③を理解する
必要がありそう.
(山本さんの資料より)

①動粘性係数を計算する

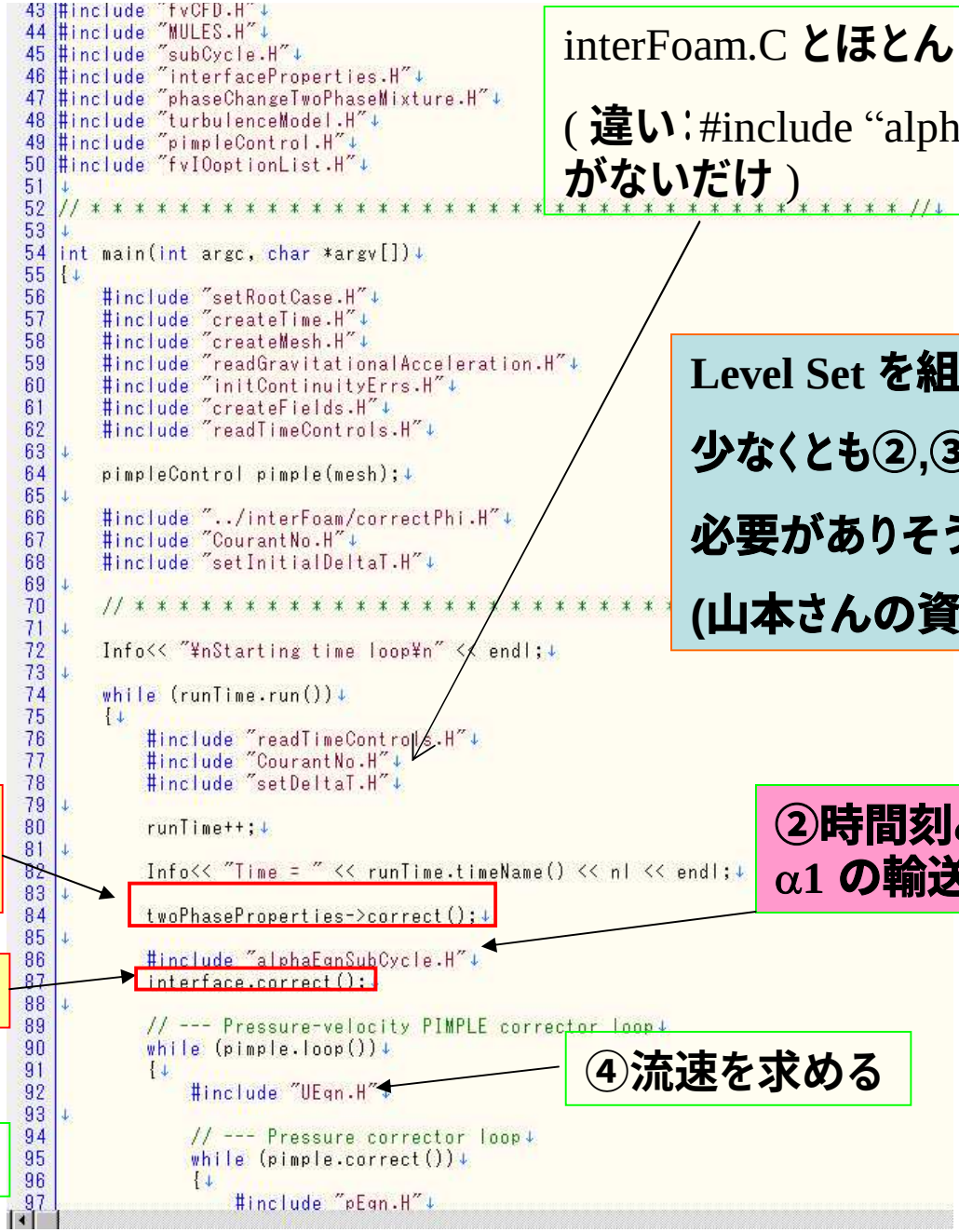
②時間刻みを小さくして α1 の輸送方程式を解く

難っ。。。

③表面張力を計算する

④流速を求める

⑤圧力を求める



```
#include "fvCFD.H"
#include "MULES.H"
#include "subCycle.H"
#include "interfaceProperties.H"
#include "phaseChangeTwoPhaseMixture.H"
#include "turbulenceModel.H"
#include "pimpleControl.H"
#include "fvIOoptionList.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    #include "readGravitationalAcceleration.H"
    #include "initContinuityErrs.H"
    #include "createFields.H"
    #include "readTimeControls.H"

    pimpleControl pimple(mesh);

    #include "../interFoam/correctPhi.H"
    #include "CourantNo.H"
    #include "setInitialDeltaT.H"

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

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

        runTime++;

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

        twoPhaseProperties->correct();

        #include "alphaEqnSubCycle.H"
        interface.correct();

        // --- Pressure-velocity PIMPLE corrector loop
        while (pimple.loop())
        {
            #include "UEqn.H"

            // --- Pressure corrector loop
            while (pimple.correct())
            {
                #include "pEqn.H"
```

① twoPhaseProperties -> correct();

twoPhaseProperties はcreateFields.H で以下のように定義されている.

```
Info<< “Creating phaseChangeTwoPhaseMixture¥n” <<endl;
autoPtr<phaseChangeTwoPhaseMixture>twoPhaseProperties =
        phaseChangeTwoPhaseMixture::New(U, phi);
```

#include “phaseChangeTwoPhaseMixture.H”中で,

#include “incompressibleTwoPhaseMixture.H” とある.

$FOAM_SRC/transportModels/incompressible/incompressibleTwoPhaseMixture の incompressibleTwoPhaseMixture.H に correct() の定義がある.

```
//- Correct the laminar viscosity
viutual void correct()
{
        calcNu();
}
```

Nuは ν で, 動粘性係数を計算していると推察される

① twoPhaseProperties -> correct();

$FOAM_SRC/transportModels/incompressible/incompressibleTwoPhaseMixture
の中の incompressibleTwoPhaseeMixture.C **を見る.**

```
//- Calculate and return the laminar viscosity
void From::incompressibleTwoPhaseMixture::calcNu()
{
        nuModel1_->correct();
        nuModel2_->correct();
・・・・・・・・・・・・・・・・・・・・・・
        // Average kinematic viscosity calculated from dynamic viscosity
        nu_ = mu()/(limitedAlpha1*rho1_+ (scalar(1)-limitedAlpha1)*rho2_);
}
```

動粘性係数(kinematic viscosity) **を計算している**

② alphaEqnSubCycle.H

alphaEqnSubCycle.H **の中で**alphaEqn.H**が呼ばれる．以下左欄に示す．**

```
{
    word alphaScheme("div(phi,alpha)");
    word alpharScheme("div(phirb,alpha)");

    surfaceScalarField phir("phir", phic*interface.nHatf());

    for (int aCorr=0; aCorr<nAlphaCorr; aCorr++)
    {
        surfaceScalarField phiAlpha
        (
            fvc::flux
            (
                phi,
                alpha1,
                alphaScheme
            )
          + fvc::flux
            (
                -fvc::flux(-phir, alpha2, alpharScheme),
                alpha1,
                alpharScheme
            )
        );

        Pair<tmp<volScalarField> > vDotAlphal =
            twoPhaseProperties->vDotAlphal();
        const volScalarField& vDotcAlphal = vDotAlphal[0]();
        const volScalarField& vDotvAlphal = vDotAlphal[1]();

        //MULES::explicitSolve
        //(
        //    geometricOneField(),
        //    alpha1,
        //    phi,
        //    phiAlpha,
        //    (vDotvAlphal - vDotcAlphal)(),
        //    (divU*alpha1 + vDotcAlphal)(),
        //    1,
        //    0
        //);

}
```

ここに定義あり．

alphaEqnSubCycle.H

```
    surfaceScalarField phic(mag(phi/mesh.magSf()));
    phic = min(interface.cAlpha()*phic, max(phic));
```

$\phi\alpha=\phi\alpha+\phi_r(1-\alpha)\alpha$

$\phi\alpha$

$\phi_r(1-\alpha)\alpha$

山本さんの解説より

$\phi_c = \min(C_\alpha\phi_c, \max(\phi_c))$

$\phi_c=\phi/S_f$

$\phi_r=\phi_c n_f$

phaseChangeTwoPhaseMixture.C **に定義あり．**

MULES**の準備？**

vDotAlphal[0]()**は体積的凝縮率***(1-alphal) =vDotcAlphal

vDotAlpha[1]()**は体積的蒸発率***alphal=vDotvAlphal

②’ alphaEqn.H

alphaEqnSubCycle.H **の中で**alphaEqn.H**が呼ばれる．以下に示す（続き）．**

interPhaseChangeFoamの場合

```
42          MULES::implicitSolve
43          (
44              geometricOneField(),
45              alpha1,
46              phi,
47              phiAlpha,
48              (divU + vDotvAlpha1 - vDotcAlpha1)(),
49              vDotcAlpha1,
50              1,
51              0
52          );
53
54          alpha2 = 1.0 - alpha1;
55          rhoPhi = phiAlpha*(rho1 - rho2) + phi*rho2;
56      }
57
58      Info<< "Liquid phase volume fraction = "
59          << alpha1.weightedAverage(mesh.V()).value()
60          << "  Min(alpha1) = " << min(alpha1).value()
61          << "  Max(alpha1) = " << max(alpha1).value()
62          << endl;
63  }
```

MULESって？

H., Wellerさんが開発したらしい．

MULES+OpenFOAMで検索すると

下記論文参照となっていることが多い．

しかし，何やっているかよく分からん。。。

Henrik Rusche, Ph.D thesis (2002),

Computational fluid dynamics of dispersed two-phase flows at high phase fractions

interFoamの場合

```
5       surfaceScalarField phic(mag(phi/mesh.magSf()));
6       phic = min(interface.cAlpha()*phic, max(phic));
7       surfaceScalarField phir(phic*interface.nHatf());
8
9       for (int aCorr=0; aCorr<nAlphaCorr; aCorr++)
10      {
11          surfaceScalarField phiAlpha
12          (
13              fvc::flux
14              (
15                  phi,
16                  alpha1,
17                  alphaScheme
18              )
19            + fvc::flux
20              (
21                  -fvc::flux(-phir, alpha2, alpharScheme),
22                  alpha1,
23                  alpharScheme
24              )
25          );
26
27          MULES::explicitSolve(alpha1, phi, phiAlpha, 1, 0);
28
29          alpha2 = 1.0 - alpha1;
30          rhoPhi = phiAlpha*(rho1 - rho2) + phi*rho2;
31      }
32
33      Info<< "Phase-1 volume fraction = "
34          << alpha1.weightedAverage(mesh.Vsc()).value()
35          << "  Min(alpha1) = " << min(alpha1).value()
36          << "  Max(alpha1) = " << max(alpha1).value()
37          << endl;
```

Multidimensional Universal Limiter with Explicit Solution

②’’ alphaEqn.H とMULES.H

$FOAM_SRC/finiteVolume/fvMatrices/solvers/MULES/MULES.H を見る.

interPhaseChangeFoamの場合

```
42          MULES::implicitSolve
43          (
44              geometricOneField(),
45              alpha1,
46              phi,
47              phiAlpha,
48              (divU + vDotvAlpha - vDotcAlpha)(),
49              vDotcAlpha,
50              1,
51              0
52          );
53
54          alpha2 = 1.0 - alpha1;
55          rhoPhi = phiAlpha*(rho1 - rho2) + phi*rho2;
56      }
57
58      Info<< "Liquid phase volume fraction = "
59          << alpha1.weightedAverage(mesh.V()).value()
60          << "  Min(alpha1) = " << min(alpha1).value()
61          << "  Max(alpha1) = " << max(alpha1).value()
62          << endl;
63  }
```

MULES.H

```
92  template<class RhoType, class SpType, class SuType>
93  void implicitSolve
94  (
95      const RhoType& rho,
96      volScalarField& gamma,
97      const surfaceScalarField& phi,
98      surfaceScalarField& phiCorr,
99      const SpType& Sp,
100     const SuType& Su,
101     const scalar psiMax,
102     const scalar psiMin
103 );
104
105 void implicitSolve
106 (
107     volScalarField& gamma,
108     const surfaceScalarField& phi,
109     surfaceScalarField& phiCorr,
110     const scalar psiMax,
111     const scalar psiMin
112 );
```

(divU+体積的蒸発量－体積的凝縮量)

は陰的に，体積的凝縮量は陽的に解く？

Sp, Su は？

②’’’ MULES の Sp と Su

Equation discretization in OpenFOAM

- Converts the PDEs into a set of linear algebraic equations, **Ax=b**, where **x** and **b** are volFields (geometricFields). **A** is an fvMatrix, which is created by a discretization of a geometricField and inherits the algebra of its corresponding field, and it supports many of the standard algebraic matrix operations
- The fvm (Finite Volume Method) and fvc (Finite Volume Calculus) classes contain static functions for the differential operators, and discretize any geometricField. fvm returns an fvMatrix, and fvc returns a geometricField.

Examples:

Term description	Implicit/explicit	Mathematical expression	fvm::/fvc:: functions
Laplacian	Implicit/Explicit	$\nabla \cdot \Gamma \nabla \phi$	laplacian(Gamma,phi)
Time derivative	Implicit/Explicit	$\partial \phi / \partial t$	ddt(phi)
		$\partial \rho \phi / \partial t$	ddt(rho, phi)
Convection	Implicit/Explicit	$\nabla \cdot (\psi)$	div(psi, scheme)
		$\nabla \cdot (\psi \phi)$	div(psi, phi, word)
			div(psi, phi)
Source	Implicit	$\rho \phi$	Sp(rho, phi)
	Implicit/Explicit		SuSp(rho, phi)

ϕ: vol<type>Field, ρ: scalar, volScalarField, ψ: surfaceScalarField

②’’’ MULES の Sp と Su

HowTo Adding a new transport equation

Suppose you want to solve and additional scalar transport equation for the scalar ψ by adding it to an existing solver. The equation has the form

$$\frac{\partial}{\partial t}(\rho\psi) + \nabla \cdot \phi\psi - \nabla \cdot \Gamma\nabla\psi = S_\psi$$

where ρ and Γ are defined as `dimensionedScalar` and are retrieved from a dictionary using the `lookup` member function. S_ψ is the source term.

To implement the equation, just add the line:

```
dimensionedScalarField psi;
```

to the `createFields.H` file of the solver.

Then, solve the equation by adding the following lines in the point of the solver where you want the equation to be solved.

```
solve
(
    fvm::ddt(rho, psi)
  + fvm::div(phi, psi)
  - fvm::laplacian(gamma, psi)
  ==
    S_psi
);
```

In this example, the source term is treated explicitly. To manage it implicitly, OpenFOAM provides the `Sp` and the `SuSp` functions. Suppose we can write it as $S_\psi = K\psi$. The code lines to solve the same equation with an implicit treatment of the source term are:

```
solve
(
    fvm::ddt(rho, psi)
  + fvm::div(phi, psi)
  - fvm::laplacian(gamma, psi)
  ==
    fvm::Sp(kappa, psi)
);
```

`SuSp(kappa, psi)` can be used to discretise the source term implicitly or explicitly according to the sign of `kappa`.

②’’ alphaEqn.H とMULES.H

alphaEqnSubCycle.H **の中で**alphaEqn.H**が呼ばれる．以下に示す（続き）．**

interFoamの場合

```
60 template<class RhoType, class SpType, class SuType>
61 void explicitSolve
62 (
63     const RhoType& rho,
64     volScalarField& psi,
65     const surfaceScalarField& phiPsi,
66     const SpType& Sp,
67     const SuType& Su
68 );
69
70 template<class RhoType, class SpType, class SuType>
71 void explicitSolve
72 (
73     const RhoType& rho,
74     volScalarField& psi,
75     const surfaceScalarField& phiBD,
76     surfaceScalarField& phiPsi,
77     const SpType& Sp,
78     const SuType& Su,
79     const scalar psiMax,
80     const scalar psiMin
81 );
82
83 void explicitSolve
84 (
85     volScalarField& psi,
86     const surfaceScalarField& phiBD,
87     surfaceScalarField& phiPsi,
88     const scalar psiMax,
89     const scalar psiMin
90 );
```

これに対応してそう

```
5     surfaceScalarField phic(mag(phi/mesh.magSf()));
6     phic = min(interface.cAlpha()*phic, max(phic));
7     surfaceScalarField phir(phic*interface.nHatf());
8
9     for (int aCorr=0; aCorr<nAlphaCorr; aCorr++)
10    {
11        surfaceScalarField phiAlpha
12        (
13            fvc::flux
14            (
15                phi,
16                alpha1,
17                alphaScheme
18            )
19          + fvc::flux
20            (
21                -fvc::flux(-phir, alpha2, alpharScheme),
22                alpha1,
23                alpharScheme
24            )
25        );
26
27        MULES::explicitSolve(alpha1, phi, phiAlpha, 1, 0);
28
29        alpha2 = 1.0 - alpha1;
30        rhoPhi = phiAlpha*(rho1 - rho2) + phi*rho2;
31    }
32
33    Info<< "Phase-1 volume fraction = "
34        << alpha1.weightedAverage(mesh.Vsc()).value()
35        << "  Min(alpha1) = " << min(alpha1).value()
36        << "  Max(alpha1) = " << max(alpha1).value()
37        << endl;
```

山本さんが解説されるとのことですので，勉強させて頂きます．

③ interface.correct()

③ interface.correct();

interface はcreateFields.H で以下のように定義されている.

```
// Construct interface from alpha1 distribution
interfaceProperties interface(alpha1, U, twoPhaseProperties());
```

$FOAM_SRC/transportModels/interfaceProperties/interfaceProperties.H を見る

③ interface.correct()

$FOAM_SRC/transportModels/interfaceProperties/interfaceProperties.H を見る

```
51 /*---------------------------------------------------------------------------*\
52                            Class interfaceProperties Declaration
53 \*---------------------------------------------------------------------------*/
54
55 class interfaceProperties
56 {
57     // Private data
58
59         //- Keep a reference to the transportProperties dictionary
60         const dictionary& transportPropertiesDict_;
61
62         //- Compression coefficient
63         scalar cAlpha_;
64
65         //- Surface tension
66         dimensionedScalar sigma_;
67
68         //- Stabilisation for normalisation of the interface normal
69         const dimensionedScalar deltaN_;
70
71         const volScalarField& alpha1_;
72         const volVectorField& U_;
73         surfaceScalarField nHatf_;
74         volScalarField K_;
75
76
77     // Private Member Functions
78
79         //- Disallow default bitwise copy construct and assignment
80         interfaceProperties(const interfaceProperties&);
81         void operator=(const interfaceProperties&);
82
83         //- Correction for the boundary condition on the unit normal nHat on
84         //  walls to produce the correct contact dynamic angle
85         //  calculated from the component of U parallel to the wall
86         void correctContactAngle
87         (
88             surfaceVectorField::GeometricBoundaryField& nHat,
89             surfaceVectorField::GeometricBoundaryField& gradAlphaf
90         ) const;
```

Cα:compression**係数**

σ:**表面張力**

δN:**安定化するもの**

③ interface.correct()

$FOAM_SRC/transportModels/interfaceProperties/interfaceProperties.H を見る

```
113     // Member Functions
114
115         scalar cAlpha() const
116         {
117             return cAlpha_;
118         }
119
120         const dimensionedScalar& deltaN() const
121         {
122             return deltaN_;
123         }
124
125         const surfaceScalarField& nHatf() const
126         {
127             return nHatf_;
128         }
129
130         const volScalarField& K() const
131         {
132             return K_;
133         }
134
135         const dimensionedScalar& sigma() const
136         {
137             return sigma_;
138         }
139
140         tmp<volScalarField> sigmaK() const
141         {
142             return sigma_*K_;
143         }
144
145         void correct()
146         {
147             calculateK();
148         }
```

Cα:compression**係数**

δN:**安定化するもの**

n_f

K:**曲率**

σ:**表面張力**

σK

K:**曲率を計算**

interfaceProperties.C**のコンストラクタ**

```
    deltaN_
    (
        "deltaN",
        1e-8/pow(average(alpha1.mesh().V()), 1.0/3.0)
    ),
```

```
    sigma_(dict.lookup("sigma")),
```

③ interface.correct()

$FOAM_SRC/transportModels/interfaceProperties/interfaceProperties.C を見る

```
109 void Foam::interfaceProperties::calculateK()↓
110 {↓
111     const fvMesh& mesh = alpha1_.mesh();↓
112     const surfaceVectorField& Sf = mesh.Sf();↓
113 ↓
114     // Cell gradient of alpha↓
115     const volVectorField gradAlpha(fvc::grad(alpha1_));↓
116 ↓
117     // Interpolated face-gradient of alpha↓
118     surfaceVectorField gradAlphaf(fvc::interpolate(gradAlpha));↓
119 ↓
120     //gradAlphaf -=↓
121     //    (mesh.Sf()/mesh.magSf())↓
122     //   *(fvc::snGrad(alpha1_) - (mesh.Sf() & gradAlphaf)/mesh.magSf());↓
123 ↓
124     // Face unit interface normal↓
125     surfaceVectorField nHatfv(gradAlphaf/(mag(gradAlphaf) + deltaN_));↓
126     correctContactAngle(nHatfv.boundaryField(), gradAlphaf.boundaryField());↓
127 ↓
128     // Face unit interface normal flux↓
129     nHatf_ = nHatfv & Sf;↓
130 ↓
131     // Simple expression for curvature↓
132     K_ = -fvc::div(nHatf_);↓
133 ↓
134     // Complex expression for curvature.↓
135     // Correction is formally zero but numerically non-zero.↓
136     /*↓
137     volVectorField nHat(gradAlpha/(mag(gradAlpha) + deltaN_));↓
138     forAll(nHat.boundaryField(), patchi)↓
139     {↓
140         nHat.boundaryField()[patchi] = nHatfv.boundaryField()[patchi];↓
141     }↓
142 ↓
143     K_ = -fvc::div(nHatf_) + (nHat & fvc::grad(nHatfv) & nHat);↓
144     */↓
145 }↓
```

$$n_{fv} = \frac{(\nabla\alpha)_f}{\left|(\nabla\alpha)_f + \delta_N\right|}$$

$$n_f = n_{fv} \cdot S_f$$

$$k = \nabla \cdot \boldsymbol{n}$$

山本さんの資料

OpenFOAMにおける混相流を照らし合わせると何をやっているかよくわかる.

③ interface.correct()

$FOAM_SRC/transportModels/interfaceProperties/interfaceProperties.C を見る

```
109 void Foam::interfaceProperties::calculateK()
110 {
111     const fvMesh& mesh = alpha1_.mesh();
112     const surfaceVectorField& Sf = mesh.Sf();
113 
114     // Cell gradient of alpha
115     const volVectorField gradAlpha(fvc::grad(alpha1_));
116 
117     // Interpolated face-gradient of alpha
118     surfaceVectorField gradAlphaf(fvc::interpolate(gradAlpha));
119 
120     //gradAlphaf -=
121     //    (mesh.Sf()/mesh.magSf())
122     //   *(fvc::snGrad(alpha1_) - (mesh.Sf() & gradAlphaf)/mesh.magSf());
123 
124     // Face unit interface normal
125     surfaceVectorField nHatfv(gradAlphaf/(mag(gradAlphaf) + deltaN_));
126     correctContactAngle(nHatfv.boundaryField(), gradAlphaf.boundaryField());
127 
128     // Face unit interface normal flux
129     nHatf_ = nHatfv & Sf;
130 
131     // Simple expression for curvature
132     K_ = -fvc::div(nHatf_);
133 
134     // Complex expression for curvature.
135     // Correction is formally zero but numerically non-zero.
136     /*
137     volVectorField nHat(gradAlpha/(mag(gradAlpha) + deltaN_));
138     forAll(nHat.boundaryField(), patchi)
139     {
140         nHat.boundaryField()[patchi] = nHatfv.boundaryField()[patchi];
141     }
142 
143     K_ = -fvc::div(nHatf_) + (nHat & fvc::grad(nHatfv) & nHat)
144     */
145 }
```

Level-Set関数 ϕ

$$k = -\nabla \cdot \boldsymbol{n}_f = -\nabla \cdot \left(\frac{(\nabla\phi)_f}{(\nabla\phi)_f + \delta} \right)$$

$$\delta(\phi) = 0 \qquad |\phi| > \varepsilon$$

$$\delta(\phi) = \frac{1}{2\varepsilon}\left(1 + \cos\left(\frac{\pi\phi}{\varepsilon}\right)\right) \qquad |\phi| \le \varepsilon$$

$$H(\phi) = 0 \qquad \phi < -\varepsilon$$

$$H(\phi) = \frac{1}{2}\left(1 + \frac{\phi}{\varepsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\phi}{\varepsilon}\right)\right) \qquad |\phi| \le \varepsilon$$

$$H(\phi) = 1 \qquad \phi > \varepsilon$$

$$\rho = H\rho_l + (1-H)\rho_g$$

$$\mu = H\mu_l + (1-H)\mu_g$$

さて，どこに記述すればよいのやら。。。．

何卒ご教示頂きたくお願い致します．