interPhaseChangeFoam求解器解析

海兮吾槊
仰望星空3763

另外版本请参阅东岳流体。（已发布）

- 分割 -

正文开始：~

interPhaseChangeFoam是用来求解两种有质量传递的不可压流体多相流求解器，界面处理方式是利用VOF方法进行捕捉。其控制方程主要由四部分组成，分别为：连续方程，动量方程，相方程，质量传递模型组成，其中质量传递模型取Schnerr_Sauer空化模型进行分析，具体形式如下：

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U_l) = \frac{\dot{m}}{\rho_l}$$

$$\nabla \cdot U = \dot{m}\left[\frac{1}{\rho_l} - \frac{1}{\rho_v}\right]$$

$$\frac{\partial(\rho U)}{\partial t} + \nabla \cdot (\rho U U) = -\nabla p_{rgh} + \nabla \cdot \left[\mu_m \left(\nabla U + \nabla U^T\right)\right] - g \cdot h \nabla \rho + \sigma \kappa \nabla \alpha_l$$

$$\dot{m} = \begin{cases} \frac{3\rho_l \rho_v}{\rho}\alpha(1-\alpha)\frac{1}{R_b}\sqrt{\frac{2(p_v - p)}{3\rho_l}}\,(p_v > p) \\ \frac{3\rho_l \rho_v}{\rho}\alpha(1-\alpha)\frac{1}{R_b}\sqrt{\frac{2(p - p_v)}{3\rho_l}}\,(p_v < p) \end{cases}$$

其中，$R_b = \sqrt[\frac{1}{3}]{\frac{1-\alpha_l}{\alpha_l}\frac{3}{4\pi}\frac{1}{n}}$ 为平均泡半径，n为液相中的空泡数目， p_v 为对应温度的饱和蒸汽压力 。对于其他诸如表面张力 $\sigma\kappa\nabla\alpha_l$ ， p_{rgh} 等请参考 interFoam求解器。

对数值求解过程有个整体的认识，介绍流程如下：首先，对相方程进行求解，分为显式和隐式，更新 α_l 及其通量和混合密度 ρ；其次，由动量方程给出速度预测值；最后，求解泊松方程进行压力值更新与速度修正，并更新通量。较之interFoam求解器而言，该求解器最大的不同在于空化模型的加入和处理，所以本文主要解析如下几点：

1. 空化模型；（SchnerrSauer.C/H, phaseChangeTwoPhaseMixture.C/H）

2. 相方程的求解；(alphaControls.H, alphaEqn.H, alphaEqnSubCycle.H)

3. 泊松方程的求解。(pEqn.H)

Section 1: 空化模型

给出在SchnerrSauer.H头文件中的基本参数，主要有四个，在计算设置时，需要用户在transportProperties内指定。统计信息如表1。

参数	n_	dNuc_	Cc_	Cv_
物理意义	来流成核数	成核直径	冷凝率系数	蒸发率系数
一般取值	1.6e+13	2.0e-06	1.0	1.0

表 1 SchnerrSauer.H 中的基础参数说明

在此基础上，延伸出重要的二级中间参数有三个，它们定义在SchnerrSauer.C的成员函数里，说明如表2。

参数关键字	alphaNuc()	rRb	pCoeff
物理意义	蒸汽泡体积分数	气泡半径倒数	公共系数
数学表达式	$V_{nuc} = \frac{n\pi d^3}{6}$ $\gamma = \frac{V_{nuc}}{1 + V_{nuc}}$	$\sqrt[3]{\frac{4n\pi}{3}\alpha_l \frac{1}{1+\gamma-\alpha_l}}$	$\frac{3\rho_l\rho_v}{\rho} * r\mathrm{Rb} * \sqrt{\frac{2}{3\rho_l}} * \frac{1}{\sqrt{\lvert p_v - p\rvert + 0.01p_v}}$

表 2 二级中间变量及 SchnerrSauer.C 中的数学表达式

最后给出质量传递源项，该项分别在求解相方程和泊松方程中加入。具体分别是：

SchnerrSauer.C中的mDotAlphal()和mDotP()的质量传递源项；

phaseChangeTwoPhaseMixture.C中的vDotAlphal()和vDotP()体积传递源项。

这四项返回值均为Pair<tmp<volScalarField>>，即为一个数组，代表了净空化传质 率和净冷凝传质率，它们具体的形式在相方程和泊松方程的求解中会继续分析，下面给出具体形式：

位置	相方程源项	
关键字	mDotAlphal()	vDotAlphal()
数学表达式	冷凝：$Cc\_ * \alpha_l * \mathrm{pCoeff} * \max(p - p_v, 0.0)$ 空化：$Cv\_ * (1+\gamma-\alpha_l) * \mathrm{pCoeff} * \min(p - p_v, 0.0)$	冷凝：$\left[\frac{1}{\rho_l} - \alpha_l\left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right)\right] * \mathrm{mDotAlphal}()[0]$ 空化：$\left[\frac{1}{\rho_l} - \alpha_l\left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right)\right] * \mathrm{mDotAlphal}()[1]$
位置	泊松方程源项	
关键字	mDotP()	vDotP()
数学表达式	冷凝：$Cc\_ * (1-\alpha_l) * \alpha_l * \mathrm{pCoeff} * \mathrm{pos}(p - p_v)$ 空化：$-Cv\_ * (1+\gamma-\alpha_l) * \alpha_l * \mathrm{pCoeff} * \mathrm{neg}(p - p_v)$	冷凝：$\left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right) * \mathrm{mDotP}()[0]$ 空化：$\left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right) * \mathrm{mDotP}()[1]$

表 3 四项净传质率

这里要注意，在相方程和泊松方程的求解中，对传质源项采用的都是隐式处理。

Section 2: 相方程

对相方程的求解需对原方程进行变换。在Weller所提出的MULES方法中，为了使interface面更佳清晰(sharp)，处理时加入了人工压缩项 $\nabla \cdot \left(U_c \alpha_l \alpha_v\right)$，其中 U_c 保证界面压缩，∇ 保证守恒性，$\alpha_l \alpha_v$ 保证有界性。具体形式如Formulation 1:

$$\frac{\partial \alpha_l}{\partial t}+\nabla \cdot\left(\alpha_l \mathrm{U}\right)+\nabla \cdot\left(\alpha_l U_l\right)-\nabla \cdot\left(\alpha_l U\right)=\frac{\dot{m}}{\rho_l}$$

$$\frac{\partial \alpha_l}{\partial t}+\nabla \cdot\left(\alpha_l \mathrm{U}\right)+\nabla \cdot\left(\alpha_l U_l\right)-\nabla \cdot\left(\alpha_l\left(\alpha_v U_v+\alpha_l U_l\right)\right)=\frac{\dot{m}}{\rho_l}$$

$$\frac{\partial \alpha_l}{\partial t}+\nabla \cdot\left(\alpha_l U\right)+\nabla \cdot\left(\alpha_l \alpha_v U_c\right)=\frac{\dot{m}}{\rho_l}$$

这里强调一点，为什么说 $\nabla \cdot \left(U_c \alpha_l \alpha_v\right)$ 是人工压缩项呢？因为在VOF假设中，各相是共享速度和压力的。也就是说，基于这个假设，$U_c = 0$。所以 U_c 是人为设定的，其形式为：

$$U_c=c_\alpha|U| \frac{\nabla \alpha}{|\nabla \alpha|}$$

from alphaEqnSubCycle.H，phic代表的是 $c_\alpha |U|$。

```
surfaceScalarField phic(interface.cAlpha()*mag(phi/mesh.magSf()));
```

from alphaEqn.H，phir代表的是 U_c，interface.nHatf()即为 $\dfrac{\nabla \alpha}{|\nabla \alpha|}$。

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

在相方程的隐式求解中，加入了速度散度项的影响，其目的和在动量方程中考虑速度散度是一样的，都是为了强调连续方程的作用。不过不同的是，由于质量传递的作用，该模型的速度散度不为0。具体演化过程如Formulation 2:

$$\frac{\partial \alpha_l}{\partial t}+\nabla \cdot\left(\alpha_l U_l\right)=\alpha_l(\nabla \cdot U)-\alpha_l(\nabla \cdot U)+\frac{\dot{m}}{\rho_l}$$

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U_l) = \alpha_l (\nabla \cdot U) - \alpha_l \left(\dot{m} \left[\frac{1}{\rho_l} - \frac{1}{\rho_v} \right] \right) + \frac{\dot{m}}{\rho_l}$$

$$\frac{\partial \alpha_l}{\partial t} + \nabla \cdot (\alpha_l U_l) = \alpha_l (\nabla \cdot U) + \dot{m} \left[\frac{1}{\rho_l} - \alpha_i \left(\frac{1}{\rho_l} - \frac{1}{\rho_v} \right) \right]$$

在alphaControls.H中，有以下控制计算的参数：

nAlphaCorr，代表的是 α_l 通量修正的次数；
nAlphaSubCycles，代表的是将每个时间步长均分成对应个数的子步长，循环后加和
MULESCorr，即Weller所提出的MULES修正，一般选为yes；
icAlpha，各向同性压缩系数，类似于一个松弛因子的作用，一般为0。

以上参数均在fvSolution中的alpha子字典中进行定义。

以下代码演示的是在alphaEqnSubCycle.H中，控制相方程求解的整个大循环，此举的目的在于可使用大的时间步长。

可以看出，求解相方程的次数为nAlphaSubCycles，当该参数大于1时，需要将每一个子时间步的结果进行加和。

```
    volScalarField divU(fvc::div(phi));

     if (nAlphaSubCycles > 1)
     {
         dimensionedScalar totalDeltaT = runTime.deltaT();
         surfaceScalarField rhoPhiSum("rhoPhiSum", rhoPhi);

         for
         (
             subCycle<volScalarField> alphaSubCycle(alpha1, nAlphaSubCycles);
             !(++alphaSubCycle).end();
         )
         {
             #include "alphaEqn.H"
             rhoPhiSum += (runTime.deltaT()/totalDeltaT)*rhoPhi;
         }

         rhoPhi = rhoPhiSum;
     }
     else
     {
         #include "alphaEqn.H"
```

```
    }

    rho == alpha1*rho1 + alpha2*rho2;
```

下面进入相方程的具体求解步骤。在alphaEqn.H中，对Formulation2进行离散，可以看出，在MULESCorr为yes时，第一步是预测 α_l ；相方程时间项和对流项的离散方法已定，也就是说在fvScheme中不可选；另外，在源项的处理上，对 $\dot{m}$ 中的 α_l 进行隐式处理。对应的代码为：

```
   if (MULESCorr)
    {
        fvScalarMatrix alpha1Eqn
        (
            fv::EulerDdtScheme<scalar>(mesh).fvmDdt(alpha1)
          + fv::gaussConvectionScheme<scalar>
            (
                mesh,
                phi,
                upwind<scalar>(mesh, phi)
            ).fvmDiv(phi, alpha1)
          - fvm::Sp(divU, alpha1)
         ==
            fvm::Sp(vDotvmcAlphal, alpha1)
          + vDotcAlphal
        );

        alpha1Eqn.solve();

  //省略了write()函数

        talphaPhi = alpha1Eqn.flux();
    }
```

对于MULES通量修正方法，它是基于FCT(Flux-Corrected Transport, 文章链接) 发展而来。定义的talphaPhiCorr为相方程的Formulation1左侧的后两项；MULESCorr为yes时，通过Formulation1与2相减修正相通量（MULES算法分析**在此**）。如果MULESCorr为false则相方程的求解采用显式方法。

Section 3: 泊松方程

将动量方程离散成以下形式：

$$A_P U_P = \sum A_N U_N + S_P \left(-\left(\nabla p_{rgh}\right) - g \cdot h \nabla \rho + f_{\sigma,P}\right) V$$

$$U_P = \frac{1}{A_P} \sum A_N U_N + \frac{1}{A_P} \left(-\left(\nabla p_{rgh}\right) - g \cdot h \nabla \rho + f_{\sigma,P}\right)$$

设 $H_P^A = \frac{1}{A_P^U} \sum A_N U_N$ ，向面上插值有:

$$U_f = H_f^A + \frac{1}{A_f} \left(-\left(\nabla p_{rgh}\right)_f - (g \cdot h)_f (\nabla \rho)_f + f_{\sigma,f}\right)$$

乘以面积 S_f ，通量为:

$$\phi_f = U_f \cdot S_f = H_f^A \cdot S_f + \frac{1}{A_f} \left(-(g \cdot h)_f (\nabla \rho)_f + f_{\sigma,f}\right) \cdot S_f - \frac{1}{A_f} \left(\nabla p_{rgh}\right)_f \cdot S_f$$

其中，PhiHbyA为 $H_f^A \cdot S_f$ ，Phig为 $\frac{1}{A_f^U} \left(-(g \cdot h)_f (\nabla \rho)_f + f_{\sigma,f}\right) \cdot S_f$ ，代码如下:

```
volScalarField rAU("rAU", 1.0/UEqn.A());
 surfaceScalarField rAUf("rAUf", fvc::interpolate(rAU));
 volVectorField HbyA(constrainHbyA(rAU*UEqn.H(), U, p_rgh));
 surfaceScalarField phiHbyA
 (
     "phiHbyA",
     fvc::flux(HbyA)
   + fvc::interpolate(rho*rAU)*fvc::ddtCorr(U, phi)
 );
 adjustPhi(phiHbyA, U, p_rgh);

 surfaceScalarField phig
 (
     (
         interface.surfaceTensionForce()
       - ghf*fvc::snGrad(rho)
     )*rAUf*mesh.magSf()
 );

 phiHbyA += phig;//注意此处，PhiHbyA与Phig二者相加
```

由连续方程有:

$$\sum U_f \cdot S_f = [\text{ vDotP() [0]} - \text{ vDotP() [1]}]\,(p - p_v)$$

$$\sum PhiHbyA + \sum Phig - \sum \frac{1}{A_f}(\nabla p_{rgh})_f \cdot S_f = [\text{ vDotP() [0]} - \text{ vDotP() [1]}]\,(p_{rgh} - (p_v - \rho gh))$$

对应代码如下:

```
while (pimple.correctNonOrthogonal())
 {
     fvScalarMatrix p_rghEqn
     (
         fvc::div(phiHbyA) - fvm::laplacian(rAUf, p_rgh)
       - (vDotvP - vDotcP)*(pSat - rho*gh) + fvm::Sp(vDotvP - vDotcP, p_rgh)
     );

     p_rghEqn.setReference(pRefCell, pRefValue);

     p_rghEqn.solve(mesh.solver(p_rgh.select(pimple.finalInnerIter())));

     if (pimple.finalNonOrthogonalIter())
     {
         phi = phiHbyA + p_rghEqn.flux();

         U = HbyA + rAU*fvc::reconstruct((phig + p_rghEqn.flux())/rAUf);
         U.correctBoundaryConditions();
         fvOptions.correct(U);
     }
 }
```

可以看出在p_rghEqn方程中加入了质量传递项，与interFoam稍有不同，而其他部分基本一致。

最后，由p_{rgh}求解p代码如下，如果进出口边界都为自然边界，则需要在fvSolution中设定参考压力值。

```
p == p_rgh + rho*gh;

 if (p_rgh.needReference())
 {
     p += dimensionedScalar
     (
         "p",
         p.dimensions(),
         pRefValue - getRefCellValue(p, pRefCell)
```

```
    );
    p_rgh = p - rho*gh;
}
```