The BetaSigmaSlurryFoam solver

Qi Yang[1)] and Gianandrea Vittorio Messa[2)]

[1)] Department of Hydraulic Engineering, Tsinghua University, Beijing 100084, China; kimyyeung@mail.tsinghua.edu.cn
[2)] Department of Civil and Environmental Engineering, Politecnico di Milano, Milano 20133, Italy; gianandreavittorio.messa@polimi.it

Introduction

betaSigmaSlurryFoam is an Euler-Euler two-phase model for simulating slurry flows in which the particle transport is governed by turbulent dispersion, which typically takes place when the particle size is sufficiently small and the conveying velocity is sufficiently high. It cannot be applied to laminar flow as well as when particle-particle interactions play a role.

betaSigmaSlurryFoam arises as an OpenFOAM implementation of the β-σ two-fluid model, which was developed by the authors with important contributions by colleagues from CHAM (UK) and Czech Technical University in Prague. The β-σ two-fluid model was developed and implemented into the PHOENICS code, and it has the subject of extensive analyses reported in several research articles, e.g. [1-3].

Hereafer some detailed instructures to run the betaSigmaSlurryFoam are given. Given that "betaSigmaSlurryFoam" solver has different type of turbulence model, drag model, alphaEquation, etc., the built-in library of openFOAM should be modified. Specifically, three libraries are changed, based on the defaults ones "compressibleTwoPhaseSystem", "compressibleEulerianInterfacialModels", and "phaseCompressibleTurbulenceModels". Note that this step also need to be followed in any library compiling if the other modified library are to be included. For example, "mycompressibleTwoPhaseSystem" and "mycompressibleEulerianInterfacialModels" should be added in Make/options of "myphaseCompressibleTurbulenceModels".

Detail procedures

Copy from the twoPhaseEulerFoam folder and make the following changes:

change twoPhaseEulerianFoam.C into betaSigmaSlurryFoam.C

Make->files
change twoPhaseEulerianFoam into betaSigmaSlurryFoam

Make->options
Add line 26-29

```
  -L$(FOAM_USER_LIBBIN) \
   -lmycompressibleTwoPhaseSystem \
   -lmycompressibleEulerianInterfacialModels \
   -lmyphaseCompressibleTurbulenceModels
```

pUf->UEqns.H
Add line 26-78,

```
//define the dimensional scalar and constants
const dimensionedScalar myE("myE", dimensionSet(0,0,0,0,0,0,0),scalar(9.8) ); //roughness parameter for smooth pipe
const dimensionedScalar myKappa("myKappa", dimensionSet(0,0,0,0,0,0,0),scalar(0.41) ); //von Karman constant
const dimensionedScalar myZero("myZero", dimensionSet(0,0,0,0,0,0,0),scalar(0) );
const dimensionedScalar myTwo("myTwo", dimensionSet(-1,5,0,0,0,0,0),scalar(1) );
const dimensionedScalar myThree("myThree", dimensionSet(0,-1,1,0,0,0,0),scalar(1) );
const dimensionedScalar myFour("myFour", dimensionSet(-1,0,0,0,0,0,0),scalar(1) );
const dimensionedScalar vSmall("vSmall", dimensionSet(0,2,-1,0,0,0,0),scalar(1e-10) );

//extract the velocity field in the near-wall cell Uw
```

```
volVectorField Uw=U1*myZero; //velocity vector in the near-wall cell
volVectorField newFaceNormal=U1*myThree*myZero; //Normal vector of near-wall cell surface
volScalarField newArea = rho1*myTwo*myZero; //area of near-wall cell surface
volScalarField inverseVolume = rho1*myFour*myZero; // the reciprocal of near-wall cell volume

   forAll(mesh.cells(),cellI)
   {
     inverseVolume[cellI] = pow(mesh.V()[cellI],-1);
   }

const fvPatchList& patches = mesh.boundary();
labelList wallList;
wallList.clear();

   forAll(patches, patchi)
   {
      const fvPatch& curPatch = patches[patchi];
      if (isType<wallFvPatch>(curPatch))
      {
         forAll(curPatch, facei)
         {
            label faceCelli = curPatch.faceCells()[facei];
            Uw[faceCelli] = U1[faceCelli];
            newArea[faceCelli] = mesh.magSf().boundaryField()[patchi][facei];
            newFaceNormal[faceCelli] = mesh.Sf().boundaryField()[patchi][facei]/mesh.magSf().boundaryField()[patchi][facei];
         }
      }
   }


//obtain the velocity component in near-wall cell parallel to the wall Utau
volVectorField Utau=Uw-Unormal;
volVectorField Unormal=(newFaceNormal & Uw)*newFaceNormal;

//calculate the solid viscosity nus and friction parameter St for solid phase
volScalarField nus=1/max(alpha1+scalar(0.001))*(rho2/rho1)*(exp(2.5/mybeta*(pow(alpha2,-mybeta)-1))-alpha2)*phase2.turbulence().nu();
volScalarField Res=max(mag(Utau)*myY/max(nus,vSmall),1e-6);
volScalarField st = 1/max(Res,1e-8);

  for (int i=0; i<6; i++)
   {
      st = pow(myKappa,2)/pow(log(myE*(Res*sqrt(st))),2);
   }
```

Change line 88-89 and line 105-106,
- fvm::laplacian(alpha1*rho1*(nus+phase2.turbulence().nut()), U1) //viscous force term with modified solid viscosity
- fvm::div(fvc::snGrad(alpha1)*mesh.magSf()*fvc::interpolate(phase2.turbulence().nut())*fvc::interpolate(rho1)/mysigma, U1) // phase-diffusion term

- fvm::laplacian(alpha2*rho2*phase2.turbulence().nuEff(), U2)
- fvm::div(fvc::snGrad(alpha2)*mesh.magSf()*fvc::interpolate(phase2.turbulence().nut())*fvc::interpolate(rho2)/mysigma, U2) // phase-diffusion term

interfacialModels->dragModels->dragModel
(1) Make\files: Add line 64, LIB = $(FOAM_USER_LIBBIN)/libmycompressibleEulerianInterfacialModels
(2) Make\options: add line 12-17,

LIB_LIBS = \
   -L$(FOAM_USER_LIBBIN) \
   -lmycompressibleTwoPhaseSystem \
   -lcompressibleTransportModels \
   -lfluidThermophysicalModels \
   -lspecie
(3) Create a file "myVarsInter.H"
(4) Add line 109, #include "myVarsInter.H"
(5) Change line 111-116, 0.75 *CdRe()*swarmCorrection_->Cs()*pair_.continuous().rho()*(exp(2.5/mybeta*(pow(pair_.continuous(),-mybeta)-1))*pair_.continuous().nu())/sqr(pair_.dispersed().d());

twoPhaseSystem->twoPhaseSystem.C
(1) Make\files: add line 25, LIB = $(FOAM_USER_LIBBIN)/libmycompressibleTwoPhaseSystem
(2) Make\options: add line 14-21,
LIB_LIBS = \
   -lfiniteVolume \
   -lincompressibleTransportModels \
   -lcompressibleTransportModels \
   -lfluidThermophysicalModels \
   -lspecie \
   -lturbulenceModels \
   -lcompressibleTurbulenceModels
(3) Create a file "myVars.H"
(4) Add line 353, #include "myVars.H"
(5) Add line 510, - fvm::laplacian(phase2_.turbulence().nut()/mysigma, alpha1)

phaseCompressibleTurbulenceModels->diffusionkEpsilon (copy from kEpsilon folder)
(1) Make\files: add line 1, myphaseCompressibleTurbulenceModels.C; add line 40, LIB = $(FOAM_USER_LIBBIN)/libmyphaseCompressibleTurbulenceModels
(2) Make\options: add line 23-25,
   -L$(FOAM_USER_LIBBIN) \
   -lmycompressibleTwoPhaseSystem \
   -lmycompressibleEulerianInterfacialModels
(3) Create file myphaseCompressibleTurbulenceModels.C
(4) diffusionkEpsilon.H -> add line 114, dimensionedScalar mysigmake;
(5) diffusionkEpsilon.C -> add line 28, #include "diffusionkEpsilon.H"
(6) diffusionkEpsilon.C -> add line 164-172,
```
   mysigmake_  //tunning parameter sigma
   (
      dimensioned<scalar>::lookupOrAddToDict
      (
         "mysigmake",
         this->coeffDict_,
         1
      )
   ),
```
(7) diffusionkEpsilon.C -> add line 270, - fvm::div(fvc::snGrad(alpha)*this->mesh().magSf()*fvc::interpolate(rho*nut)/mysigmake_, epsilon_)
(8) diffusionkEpsilon.C -> add line 292, -fvm::div(fvc::snGrad(alpha)*this->mesh().magSf()*fvc::interpolate(rho*nut)/mysigmake_, k_)

References

[1]. Messa, G.V. and Matoušek, V. Analysis and discussion of two-fluid modelling of pipe flow of fully suspended slurry. Powder Technol. 2020, 360,747-768.

[2]. Messa, G.V.; Malin, M.; Matoušek, V. Parametric study of the β-σ two-fluid model for simulating fully suspended slurry flow: effect of flow conditions. Meccanica 2021, 56, 1047-1077.
[3]. Messa, G.V.; Yang, Q.; Rasteiro, M.G.; Faia, P.; Matoušek, V.; Silva, R.; Garcia, F. Computational fluid dynamic modelling of fully-suspended slurry flows in horizontal pipes with different solids concentrations. KONA Powder Part. J. 2023, 40, 219-235.