reactingFoam Solver to Calculate Radiative Heat Transfer

By:

Sajjad Haider

Assignment for the Course 'CFD with OpenSource software', 2011

Chalmers University of Technology

In some combustion devices, involving large combustion chambers, the radiative heat transfer is the major mode of heat transfer compared to conduction and convection e.g. large furnaces etc. OpenFOAM® provides different solvers for modelling combustion. However, amongst the standard combustion solvers, only fireFOAM includes calculation of radiative heat transfer. The purpose of this tutorial is to provide a step by step procedure to develop a user-modified reactingFoam solver that calculates radiative heat transfer. The approach here will be to modify an existing standard solver in OpenFOAM®. This may help the readers, considering any code changes, to be able to use this tutorial possibly for future versions of OpenFOAM®. However, it is important to mention that this tutorial is based on OpenFOAM® version 2.0.1.

Some standard solvers for particle-tracking flows (Lagrangian Solvers) i.e. reactingParcelFoam, reactingParcelFilmFoam, porousExplicitSourceReactingParcelFoam and LTSReactingParcelFoam, are based on reacting flow and also include calculation of radiative heat transfer. However, this tutorial will use one of the aforementioned solvers to develop a user-modified reactingFoam solver that can be used for non-particulate reacting flows. The solver used to be modified is reactingParcelFoam.

Copy the original solver and rename it

```
cd $WM_PROJECT_USER_DIR
cp -r $FOAM_APP/solvers/lagrangian/reactingParcelFoam reactingFoamRadiation
```

Rename the file

```
cd reactingFoamRadiation
mv reactingParcelFoam.C reactingFoamRadiation.C
```

Now open the reactingFoamRadiation.C file in any text editor and comment/remove following lines:

```
#include "basicReactingCloud.H"

#include "createClouds.H"

#include "SLGThermo.H"

parcels.evolve();

rho = thermo.rho();
```

Include following (see reactingFoam.C file in the $FOAM_APP/solvers/combustion/reactingFoam)

```
#include "multivariateScheme.H"

runTime.write();
```

Remove the header files named rhoEqn.H and creatClouds.H

```
rm rhoEqn.H
rm createClouds.H
```

Open the header file hsEqn.H and modify

```
fvScalarMatrix hEqn
    (
        fvm::ddt(rho, hs)
      + mvConvection->fvmDiv(phi, hs)
      - fvm::laplacian(turbulence->alphaEff(), hs)
     ==
        DpDt
     +  parcels.Sh(hs)
     +  radiation->Shs(thermo)
     +  chemistrySh
    );

    hEqn.relax();

    hEqn.solve();
```

to

```
fvScalarMatrix hsEqn
    (
        fvm::ddt(rho, hs)
      + mvConvection->fvmDiv(phi, hs)
      - fvm::laplacian(turbulence->alphaEff(), hs)
     ==
        DpDt
     +  radiation->Shs(thermo)
     +  chemistrySh
    );

    hsEqn.relax();

    hsEqn.solve();
```

Open headerfile Ueqn.H and modify

```
fvVectorMatrix UEqn
    (
        fvm::ddt(rho, U)
      + fvm::div(phi, U)
      + turbulence->divDevRhoReff(U)
     ==
        rho.dimensionedInternalField()*g
      + parcels.SU(U)
    );
```

to

```
fvVectorMatrix UEqn
    (
    fvm::ddt(rho, U)
  + fvm::div(phi, U)
  + turbulence->divDevRhoReff(U)
 ==
    rho*g
);
```

Open headerfile Yeqn.H and modify

```
solve
            (
                fvm::ddt(rho, Yi)
              + mvConvection->fvmDiv(phi, Yi)
              - fvm::laplacian(turbulence->muEff(), Yi)
              ==
               parcels.SYi(i, Yi)
              + kappa*chemistry.RR(i)().dimensionedInternalField(),
                mesh.solver("Yi")
            );
```

to

```
solve
            (
                fvm::ddt(rho, Yi)
              + mvConvection->fvmDiv(phi, Yi)
              - fvm::laplacian(turbulence->muEff(), Yi)
             ==
                kappa*chemistry.RR(i),
                mesh.solver("Yi")
            );
```

Open headerfile peqn.H and modify

```
fvScalarMatrix pEqn
        (
            fvm::ddt(psi, p)
          + fvm::div(phid, p)
          - fvm::laplacian(rho*rAU, p)
         ==
            parcels.Srho()
        );
```

to

```
fvScalarMatrix pEqn
        (
            fvm::ddt(psi, p)
          + fvm::div(phid, p)
          - fvm::laplacian(rho*rAU, p)
        );
```

and

```
fvScalarMatrix pEqn
      (
         fvm::ddt(psi, p)
       + fvc::div(phi)
       - fvm::laplacian(rho*rAU, p)
      ==
         parcels.Srho()
      );
```

to

```
fvScalarMatrix pEqn
      (
         fvm::ddt(psi, p)
       + fvc::div(phi)
       - fvm::laplacian(rho*rAU, p)
      );
```

Open headerfile createFields.H and remove line

```
SLGThermo slgThermo(mesh, thermo);
```

Then go to Make folder and open 'files' and modify as follows

reactingParcelFoam.C   to be changed to reactingFoamRadiation.C

EXE = $(FOAM_APPBIN)/reactingParcelFoam to be changed to EXE = $(FOAM_USER_APPBIN)/reactingFoamRadiation

Then open 'Options' and remove following lines in EXE_INC = \

```
-I$(LIB_SRC)/lagrangian/basic/lnInclude\
-I$(LIB_SRC)/lagrangian/intermediate/lnInclude \
-I$(LIB_SRC)/lagrangian/distributionModels/lnInclude\
-I$(LIB_SRC)/thermophysicalModels/properties/liquidProperties/lnInclude\
-I$(LIB_SRC)/thermophysicalModels/properties/liquidMixtureProperties/lnInclude \
-I$(LIB_SRC)/thermophysicalModels/properties/solidProperties/lnInclude \
-I$(LIB_SRC)/thermophysicalModels/properties/solidMixtureProperties/lnInclude \
-I$(LIB_SRC)/thermophysicalModels/SLGThermo/lnInclude \
-I$(LIB_SRC)/regionModels/regionModel/lnInclude \
-I$(LIB_SRC)/regionModels/surfaceFilmModels/lnInclude \
-I$(LIB_SRC)/sampling/lnInclude \
```

And in EXE_LIBS = \

```
-llagrangian \
-llagrangianIntermediate \
-lregionModels \
-lSLGThermo \
-lliquidProperties \
-lliquidMixtureProperties \
-lsolidProperties \
-lsolidMixtureProperties \
-lsurfaceFilmModels \
-lsampling\
```

Save file.

Go back to reactingFoamRadiation folder and run 'wmake' command to compile the new solver.

References:

Vdovin Alexey, Radiation heat transfer in OpenFOAM®, Final assignment CFD with OpenSource software, 2009, Chalmers University of Technology.

Lecture Slides, CFD with OpenSource software, 2011, Chalmers University of Technology.