LUND UNIVERSITY

CFD with OpenSource Software
Assignment 3

A modified version of the reactingFoam tutorial for LES

Author
Piero Iudiciani

Peer reviewers
Chen Huang
Håkan Nilsson

January 2010

Contents

1 Introduction

If one wants to simulate turbulent reacting flows in OpenFOAM, one of the possibilities is to use the the RANS solver *reactingFoam*. However time dependent solutions and resolution of smaller turbulent scales are important in order to capture the flame dynamics. Therefore the main purpose is to transform reactingFoam into a LES solver. Files from a LES solver (XiFoam) will be used and changed. All the passages will de described in details. The new solver *reactingFoamLES* will then be applied on a testcase already available for reactingFoam. reactingFoam solves transport equations for the species involved in the combustion. In the tutorial the files which set the reaction mechanism will be described, and it will be shown how to select a different one. This tutorial works on OpenFoam version 1.5.x, thus we have to source it:

```
source $FOAM_INST_DIR/OpenFOAM-1.5.x/etc/bashrc
```

2 Creating a new solver

We copy the source of the available solver *reactingFoam* from the $FOAM_APP folder:

```
cd $FOAM_USER_APPBIN
cp -r $FOAM_APP/solvers/combustion/reactingFoam reactingFoamLES
cd reactingFoamLES
```

Then we copy some files from the course homepage using an internet browser or *wget*:

```
wget http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2009/ \
PieroIudiciani/tutFiles.tar.gz
tar xzf tutFiles.tar.gz
rm tutFiles.tar.gz
```

First we rename the source file `reactingFoam.C` into `reactingFoamLES.C` and we edit it:

```
mv reactingFoam.C reactingFoamLES.C
```

In particular we change line 35 from

```
#include "compressible/RASModel/RASModel.H"
```

to

```
#include "compressible/LESModel/LESModel.H"
```

and we add the following at line 40: (all the LES solvers, i.e. *coodles, XiFoam* have this line in the source file)

```
#define divDevRhoReff divDevRhoBeff
```

In `createFields.H` we substitute RAS with LES in lines 61-63. We also comment lines 72-74. According to a low Mach number approximation, in fact, in the following we will neglect variations in time of the pressure in the discretization of the equations:

```
Info << "Creating turbulence model." << nl;
autoPtr<compressible::RASModel> turbulence
(
    compressible::RASModel::New
    (
        rho,
        U,
        phi,
        thermo()
    )
);

Info<< "Creating field DpDt" << endl;
volScalarField DpDt =
   fvc::DDt(surfaceScalarField("phiU", phi/fvc::interpolate(rho)), p);}
```

becomes:

```
Info << "Creating turbulence model" << nl;
autoPtr<compressible::LESModel> turbulence
(
    compressible::LESModel::New
    (
        rho,
        U,
        phi,
        thermo()
    )
```

```
//Info<< "Creating field DpDt\n" << endl;
//volScalarField DpDt =
//    fvc::DDt(surfaceScalarField("phiU", phi/fvc::interpolate(rho)), p);
```

We now need files in which the discretization of the Navier-Stokes equations is performed. We can copy the pressure, momentum and energy equations from the solver *XiFoam*:

```
cp  $FOAM_APP/solvers/combustion/XiFoam/[phU]Eqn.H .
```

and slightly modify them. Basically we comment out the terms containing the variation in time of the pressure. In the pressure equation *p.Eqn.H* we comment out the last line (line68):

```
//DpDt = fvc::DDt(surfaceScalarField("phiU", phi/fvc::interpolate(rho)), p);
```

Similarly in the energy equation *hEqn.H* we comment out line 8 and move one term to the right hand side:

```
{
    solve
    (
        fvm::ddt(rho, h)
      + mvConvection->fvmDiv(phi, h)
      - fvm::laplacian(turbulence->alphaEff(), h)
     ==
        DpDt
    );

    thermo->correct();
}
```

---

becomes:

```
{
    solve
    (
        fvm::ddt(rho, h)
      + mvConvection->fvmDiv(phi, h)
//      - fvm::laplacian(turbulence->alphaEff(), h)
```

```
     ==
//        DpDt
        fvm::laplacian(turbulence->alphaEff(), h)
    );

    thermo->correct();
}
```

In the momentum equation we can neglect the gravitational forces and thus we delete lines 6-7:

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

    if (momentumPredictor)
    {
        solve(UEqn == -fvc::grad(p));
    }
```

becomes:

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



    if (momentumPredictor)
```

```
    {
        solve(UEqn == -fvc::grad(p));
    }
```

These equation files are also found in the tutorial files:

```
 ls tutFiles/*Eqn.H .
```

Then we change the file `Make/files` so that:

```
reactingFoamLES.C

EXE = $(FOAM_USER_APPBIN)/reactingFoamLES
```

In the file `Make/options` again we change the RAS libraries with the LES ones. We can add the following at line 5:
`-I$(LIB_SRC)/turbulenceModels/LES/LESdeltas/lnInclude\`
and delete line 2: `-I../XiFoam\`

```
EXE_INC = \
    -I../XiFoam \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/turbulenceModels/RAS \
    -I$(LIB_SRC)/thermophysicalModels/specie/lnInclude \

[...]

EXE_LIBS = \
    -lcompressibleRASModels \
    -lcombustionThermophysicalModels \
```

---

becomes:

```
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/turbulenceModels/LES \
    -I$(LIB_SRC)/turbulenceModels/LES/LESdeltas/lnInclude \
```

```
[...]

EXE_LIBS = \
    -lcompressibleLESModels \
    -lcombustionThermophysicalModels \
```

We are now ready to compile:

```
wclean
rm -r Make/linuxGccDP*
wmake
```

3 Setting up a case

Once we have the solver, we can set up a case. A tutorial file for *reactingFoam* is found in the OpenFOAM wiki. Either we download from the webpage
`http://openfoamwiki.net/index.php/Tut_reactingFoam_firstTutorial`
or we get it by typing directly in the shell:

```
wget http://openfoamwiki.net/images/b/b6/ReactingFoamCase.tar.gz
```

Once downloaded we extract the files in the folder ReactingFoamCase:

```
run
mkdir ReactingFoamCase
mv ReactingFoamCase.tar.gz ReactingFoamCase
cd ReactingFoamCase
tar xzf ReactingFoamCase.tar.gz
rm ReactingFoamCase.tar.gz
```

3.1 The *chemkin* folder

We can see that we have a folder called *chemkin*. This folder contains the informations for the chemical reaction mechanism.

```
ls chemkin/
chem.inp  chem.inp.1  chem.inp_15  chem.inp.full  therm.dat
```

chem.inp file

When solving flows with reactions, a transport equation for each of the species involved is also solved. The files **inp** contain the information about the species and the reaction mechanism. A reaction mechanism is a list of reactions that occur during a combustion process. Each reaction is characterized by a reaction rate k_i which is the "speed" at which the reaction occurs and is characterized by the Arrhenius equation:

$$k_i = A_i T^{\beta_i} exp\left(-\frac{E_i}{RT}\right) \tag{1}$$

where E_i is the energy of activation, A_i and β_i are experimental parameters, T is the temperature. A detailed mechanism containing all the species is composed by hundreds of species and reactions. It is therefore not practically solvable and usually reduced mechanism with very few reactions are used. The simplest mechanism possible is composed by only one global reaction. This is the case of the *chem.inp* file which contains a simple one-step reaction mechanism for heptane (C7H16):

```
ELEMENTS
 H   O    C N
END
SPECIE
C7H16 O2 N2 CO2 H2O
END
REACTIONS
 C7H16 + 11O2            => 7CO2 + 8H2O        5.00E+8  0.0   15780.0! 1
        FORD    / C7H16 0.25 /
        FORD    / O2 1.5 /
END
```

In this mechanism the chemical elements involved are hydrogen (H), oxygen (O), carbon (C) and nitrogen (N). The species that partecipate are heptane (C7H16), oxygen molecule (O2), nitrogen molecule (N2), carbon dioxide (C02), water, (H2O). In this case there is only one reaction in which heptane is the fuel, oxygen is the oxidizer, C02 and water are the products. Nitrogen is the inert species. The three numbers before the question mark carry the information about the reaction rate and represent respectively the parameters A_i, β_i an dE_i in 1. An example of a more complex mechanism can be found in chem.inp.full

therm.dat file

The file therm.dat instead contains a database of coefficients for several species. Such coefficients are needed to compute thermodinamical variables such as specific heat c_p/R, enthalpy H^0/RT, enthropy S^0/R, according to the following equations:

$$c_p/R = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4 \tag{2}$$

$$H^0/RT = a_1 + \frac{a_2}{2}T + \frac{a_3}{3}T^2 + \frac{a_4}{4}T^3 + \frac{a_5}{5}T^4 + \frac{a_6}{T} \tag{3}$$

$$S^0/R = a_1 lnT + \frac{a_2}{2}T + \frac{a_3}{2}T^2 + \frac{a_4}{3}T^3 + \frac{a_5}{4}T^4 + a_7 \tag{4}$$

The *therm.dat* file looks like this:

```
THERMO ALL
   200.000  1000.000  5000.000
(CH2O)3             70590C   3H   6O   3     G  0200.00   4000.00  1500.00      1
 0.01913678E+03 0.08578044E-01-0.08882060E-05-0.03574819E-08 0.06605143E-12    2
-0.06560876E+06-0.08432507E+03-0.04662286E+02 0.06091547E+00-0.04710536E-03    3
 0.01968843E-06-0.03563271E-10-0.05665404E+06 0.04525265E+03                   4
(CH3)2SICH2         61991H   8C   3SI  1     G  0200.00   2500.00  1500.00      1
 0.01547852E+03 0.01065700E+00-0.01234345E-05-0.01293352E-07 0.02528715E-11    2
-0.06693076E+04-0.05358884E+03 0.02027522E+02 0.04408673E+00-0.03370024E-03    3
 0.01484466E-06-0.02830898E-10 0.03931454E+05 0.01815821E+03                   4
AL                  62987AL  1               G  0200.00   5000.00  0600.00      1
 0.02559589E+02-0.01063224E-02 0.07202828E-06-0.02121105E-09 0.02289429E-13    2
 0.03890214E+06 0.05234522E+02 0.02736825E+02-0.05912374E-02-0.04033938E-05    3
 0.02322343E-07-0.01705599E-10 0.03886795E+06 0.04363880E+02                   4
[...]
```

These files are written and organized according to chemkin software format. Details can be found in[1] and are here resumed. The first line is chemkin syntax necessary at the beginning of the file. The three values in the second line specify three values of temperature and therefore two temperature ranges. The intermediate temperature is generally always 1000K. For each species then four lines are reported. The first species is in this case (CH2O)3 and does not necessarily need to be used in the reaction mechanism. The entries in the first line report respectively the name of the species, its elemental composition, its electronic composition, its phase (G for gas, L for liquid, S for solid), and three temperatures (low, high, intermediate). The fourteen entries in the following 3 lines report the 7 coefficients a_{1-7} in equations 2-4 for the two temperature ranges, (higher range and lower range respectively).

The chemkin directory should be located in *constant*:

```
mv chemkin/ constant/
```

3.2 The *constant* folder

In the folder *constant* the following files are found:

```
ls constant/
chemistryProperties   environmentalProperties   turbulenceProperties
chemkin               polyMesh
combustionProperties  thermophysicalProperties
```

[1] http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2007/AndreasLundstrom/reactingFoam.pdf
http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2008/PerCarlsson/PC_Tutorial_dieselFoam_peered_NL_HN.pdf

thermophysicalProperties file

In the `constant/thermophysicalProperties` file we change the path of the chemical files:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

thermoType hMixtureThermo<reactingMixture>;

CHEMKINFile                 "/cluster/samples/reactingFoam/chemkin/chem.inp";
CHEMKINThermoFile           "/cluster/samples/reactingFoam/chemkin/therm.dat";
```

becomes:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

thermoType hMixtureThermo<reactingMixture>;

CHEMKINFile                 "chemkin/chem.inp";
CHEMKINThermoFile           "chemkin/therm.dat";
```

chemistryProperties file

The settings for the *chemistryProperties* file are as follows:

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

chemistry               on;

turbulentReaction off;
Cmix                Cmix [ 0 0 0 0 0 0 0 ] 1.0 ;

//chemistrySolver         ODE;
//chemistrySolver       EulerImplicit;
chemistrySolver       sequential;

initialChemicalTimeStep     1.0e-8;
```

The entry *chemistry* should be switched on in order to solve for the chemistry equations as well. The entry *turbulentReaction* defines if a model for the effect of the smallest turbulent scales on the flame should be adopted. The model available is the partially stirred reaction developed at Chalmers University. and *Cmix* is the constant to compute the mixing time in such a model:

$$\tau_{mix} = C_{mix}\sqrt{\frac{\mu_{eff}}{\rho\epsilon}} \tag{5}$$

Three different solvers (ODE, EulerImplicit, sequential) are available for the chemisrty computations.

Subgrid turbulence model

When performing Large Eddy Simulation, a model for the subgrid scales needs to be adopted. The model is set in a *LESProperties* file. We can take it from a LES tutorial, for example *coodles*:

```
cp $WM_PROJECT_DIR/tutorials/coodles/pitzDaily/constant/LESProperties constant/
```

The turbulence model is chosen under the entry *LESmodel*. In OpenFOAM several models are available for anisochoric turbulence:

Entry	Model
Smagorinsky	Smagorinsky
oneEqEddy	k-equation eddy-viscosity
dynOneEqEddy	Dynamic k-equation eddy-viscosity
lowReOneEqEddy	Low-Re k-equation eddy-viscosity
DeardorffDiffStress	Deardorff differential stress
SpalartAllmaras	Spalart-Allmaras 1-eqn mixing-length

and we choose the Smagorinsky subgrid model:

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

LESModel            Smagorinsky;

printCoeffs         off;

delta               cubeRootVol;
```

3.3 The *system* folder

In the folder In the `system/controlDict` file we change the name of the application to `reactingLESFoam` and use the following settings:

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application         reactingFoamLES;

startFrom           latestTime;

startTime           0;

stopAt              endTime;

endTime             1;

deltaT              1e-04;

writeControl        adjustableRunTime;

writeInterval       1.0e-3;

purgeWrite          0;

writeFormat         binary;

writePrecision      6;

writeCompression    uncompressed;

timeFormat          general;

timePrecision       6;

adjustTimeStep      yes;

maxCo               0.1;

runTimeModifiable   yes;

// ************************************************************************
```

3.4 The *0/* directory and boundary conditions

We remove the latest directory, and in the *0/* directory we create a file for the turbulent viscosity *muSgs* which is needed by the Smagorinsky model. The right dimensions should be assigned to the turbulent viscosity:

```
rm -r 0.055/
cp 0/k 0/muSgs

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

dimensions          [ 1 -1 -1 0 0 0 0 ];

internalField       uniform 0.0;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value uniform 0.0;
    }
    lowerInlet
    {
        type            fixedValue;
        value uniform 0.0;
    }

    outlet
    {
        type            zeroGradient;
    }

    upperWall
    {
        type            zeroGradient;
    }
```

Make sure that *blockMesh* has been run and then run *reactingFoamLES*

```
reactingFoamLES > log &
```

4 Use another Reaction Mechanism

One might want to simulate different fuels or use a more detailed mechanism. In the tutFiles directory for example the Westbrook and Dryer two-steps mechanism for methane is available:

```
cp $FOAM_USER_APPBIN/reactingFoamLES/tutFiles/WD2steps.inp constant/chemkin/
```

```
ELEMENTS
C H O N
END
SPECIE
CH4 O2 CO H2O CO2 N2
END
REACTIONS
 CH4  +  1.5O2  =>  CO + 2H2O           2.80E+09        0.0     48400.
    FORD /CH4 -0.3/
    FORD /O2 1.3/
 CO2 => CO + 0.5O2                      5.00E+08        0.0     40000.
    FORD /CO2 1.0 /
 CO + H2O + 0.5O2 => CO2 + H2O          3.98E+14        0.0     40000.
    FORD /CO  1.0 /
    FORD /H2O 0.5 /
    FORD /O2  0.25/
END
```

One can see that in this mechanism carbon monoxide (CO) is first formed as intermediate species and then carbon dioxide. Therefore in the 0/ directory we should change the fuel from heptane (C7H16) to methane(CH4) and additionally create files for carbon monoxide, carbon dioxide and water (optionally since any other species is treated by the Ydefault file).

```
cp 0/C7H16 0/CH4
```

Accordingly the *constant/thermophysicalProperties* file should be updated:

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

thermoType hMixtureThermo<reactingMixture>;

//CHEMKINFile                  "chemkin/chem.inp";
CHEMKINFile                  "chemkin/WD2steps.inp";
CHEMKINThermoFile            "chemkin/therm.dat";

inertSpecie                  N2;
```

It is now possible to run with the new mechanism.

```
rm 0.*/
reactingFoamLES > log2 &
```