CFD WITH OPENSOURCE SOFTWARE, ASSIGNMENT 3

OpenFOAM Tutorial

Implementation of Gasoline Properties in OpenFOAM Library

by

CHEN HUANG

Division of Combustion
Department of Applied Mechanics
Chalmers University of Technology
Dec. 2009

Contents

1. Introduction 3
2. Description of thermophysical Functions and liquids library 3
3. Implementation of gasoline properties 4
3.1 Create new thermophysical functions for the gasoline properties 5
3.2 Create a new liquid class gasoline 9
4. A test of implementation through a case study 13
4.1 Modify the reitzDiwakar breakup model 13
4.2 Setup a case for gasoline hollow cone spray in a constant volume 15
4.3 Run the gasoline hollow cone spray case 20
Reference 21

1. Introduction

The real gasoline properties, such as surface tension, dynamic viscosity, heat of vaporization and so on, are important for the spray formation in the gasoline direct injection application. This tutorial focuses on how to implement gasoline properties in the OpenFOAM liquids library. First, a description of OpenFOAM liquids library is given. Second, the detailed implementation of gasoline properties is described. Finally, a case of gasoline hollow cone spray in a constant volume is applied to test the implementation.

2. Description of thermophysical Functions and liquids library

In order to explain how the liquid properties are calculated and the connections among a specific liquid class and thermophysical function classes, a simplified collaboration diagram one liquid, iso-octane (IC_8H_{18}) is shown in Figure 1. Note only NSRDSfunc0 and IC_8H_{18} are shown in Figure 1 in order to make the figure easy to read.

ThermophysicalFunction is an abstract class, which has a serious of sub-classes like, NSRDSfunc0, NSRDSfunc1, NSRDSfunc2, NSRDSfunc4, NSRDSfunc5, NSRDSfunc6, NSRDSfunc7, and APIfunctions. The abstract class and its sub-classes are linked with blue arrows in Fig. 1. The scalar class is used by NSRDSfunc0 class because it returns a scalar through a member function with a serious of scalar variables such as, a_, b_, c_, d_, e_, and f_. In this way, the NSRDSfunc0 and scalar is connected with a pink dashed arrow with a_, b_, c_, d_, e_, and f_ above the arrow. Similarly, NSRDSfunc1, NSRDSfunc2, NSRDSfunc4, NSRDSfunc5, NSRDSfunc6, NSRDSfunc7, and APIfunctions is connected with scalar class separately.

Liquid is an abstract class, which has a serious of sub-classes like, IC_8H_{18}, C_7H_8, C_7H_{16}, and so on. The liquid class is linked with scalar class because the liquid class returns a serious scalar values through its member functions, such as W(), Tc(), Pc(), Vc(), and so on. IC_8H_{18} is a sub-class which is inherited from liquid class. IC_8H_{18} class uses NSRDSfunc0 class, and K_, h_ and cp_ are feedbacks to IC_8H_{18} class through a member function in NSRDSfunc0 class. The value of K_, h_, and cp_ are different because the member function in NSRDSfunc0 class is initialized differently. The relationship between IC_8H_{18} class and the rest of the thermophysical functions, such as NSRDSfunc1, NSRDSfunc2, NSRDSfunc4, NSRDSfunc5, NSRDSfunc6, NSRDSfunc7, and APIfunctions, can be explained in a similar way.

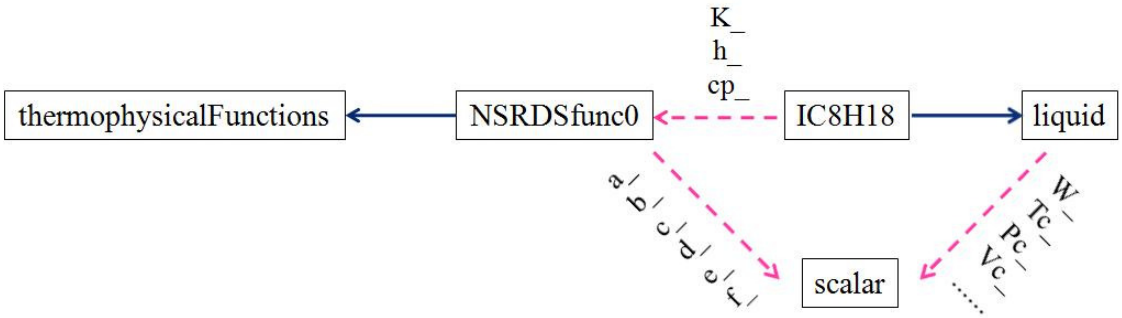

Figure 1 Simplified collaboration diagram for IC_8H_{18}

3. Implementation of gasoline properties

Note: This tutorial is based on OpenFOAM -1.6-x. The linux commands are in italic form, and the modifications are in underlined form.

The gasoline properties, such as vapour pressure (pv), heat of vapourization (hl), liquid heat capacity (cp), liquid enthalpy (h), ideal gas heat capacity (cpg), liquid viscosity (mu), liquid thermo conductivity (K), surface tension (sigma), molecular weight (W), and critical temperature (Tc), are obtained from KIVA fuel library in a form of table. The gasoline density (rho) is taken from reference [1] in a form of table. The second virial coefficient (B) of gasoline is the same as n-octane (C_8H_{18}). The vapour diffusivity (D) is the same as n-heptane (C_7H_{16}). The gasoline surrogate is composed of iso-octane (IC_8H_{18}), toluene (C_7H_8) and n-heptane (C_7H_{16}) in a volume proportion of 55%:35%:10%. The rest of gasoline properties, including vapour viscocity (mug), vapour thermo conductivity (Kg), critical pressure (Pc), critical volume (Vc), critical compressibility factor (Zc), triple point temperature (Tt), triple point pressure (Pt), normal boiling temperature (Tb), dipole moment (dipm), Pitzer's ascentric factor (omega), and solubility parameter (delta), are approximated using mixing rules [2] of the above mentioned ternary mixture, see equation 1.

$$Q_m = \sum_i y_i Q_i \qquad (1)$$

where, Q_m is a mixture parameter.

y_i is a liquid or vapor mole fraction.

Q_i is a property of a pure liquid or vapor.

3.1 Create new thermophysical functions for the gasoline properties

a. Make a directory for thermophysical functions in user working directory.
   *mkdir –p *
   $WM_PROJECT_USER_DIR/src/thermophysicalModels/thermophysicalFunctions/NSRDSfunctions/
b. Go to OpenFOAM NSRDSfunctions directory, and copy the NSRDfunc.
   cd $FOAM_SRC/thermophysicalModels/thermophysicalFunctions/NSRDSfunctions/
   *cp –r NSRDSfunc0 NSRDSfunc1 NSRDSfunc5 NSRDSfunc6 NSRDSfunc7 *
   $WM_PROJECT_USER_DIR/
   src/thermophysicalModels/thermophysicalFunctions/NSRDSfunctions/.
c. Rename them to the corresponding gasoline properties. We only change NSRDSfunc5 to NSRDSfuncgRho as an example.
   mv NSRDSfunc5 NSRDSfuncgRho
   cd NSRDSfuncgRho
   *rename NSRDSfunc5 NSRDSfuncgRho **
   sed –i s/"" NSRDSfunc5"/"" NSRDSfuncgRho"/g NSRDSfuncgRho.C
   sed –i s/"" NSRDSfunc5"/"" NSRDSfuncgRho"/g NSRDSfuncgRho.H
   rm NSRDSfuncgRho.dep
   Similarly, we can change NSRDSfunc1 to NSRDSfuncgPv, NSRDSfunc6 to NSRDSfuncgHl, NSRDSfunc0 to NSRDSfuncgCp, NSRDSfunc0 to NSRDSfuncgH, NSRDSfunc7 to NSRDSfuncgCpg, NSRDSfunc1 to NSRDSfuncgMu, NSRDSfunc0 to NSRDSfuncgK, and NSRDSfunc6 to NSRDSfuncgSigma.
d. Modify the new NSRDSfunc. We only modify the NSRDSfuncgRho function as an example.
   vi NSRDSfuncgRho.H
   insert or modify the underlined contents as following,

```
……
    // NSRDS function 105 coefficients
    scalar a_, b_, c_, d_;
    // create an scalar array with 56 members
    scalar rho[56];

public:

  //- Runtime type information
  TypeName("NSRDSfuncgRho");
……

    //- Construct from Istream
    NSRDSfuncgRho(Istream& is)
    :
```

```
    a_(readScalar(is)),
    b_(readScalar(is)),
    c_(readScalar(is)),
    d_(readScalar(is))
  {}

  /*- Construct from null & initialize the density (kg/m^3) from a table.
Density, rho, ranges from 0 K to 550 K (critical temperature for gasoline) with an interval of 10 K */
  NSRDSfuncgRho() {
    rho[0] = 9.53673e+02;
    rho[1] = 9.48499e+02;
    rho[2] = 9.43127e+02;
    rho[3] = 9.37560e+02;
    rho[4] = 9.31801e+02;
    rho[5] = 9.25855e+02;
    rho[6] = 9.19726e+02;
    rho[7] = 9.13417e+02;
    rho[8] = 9.06934e+02;
    rho[9] = 9.00280e+02;
    rho[10] = 8.93461e+02;
    rho[11] = 8.86479e+02;
    rho[12] = 8.79341e+02;
    rho[13] = 8.72050e+02;
    rho[14] = 8.64612e+02;
    rho[15] = 8.57031e+02;
    rho[16] = 8.49313e+02;
    rho[17] = 8.41461e+02;
    rho[18] = 8.33482e+02;
    rho[19] = 8.25380e+02;
    rho[20] = 8.17160e+02;
    rho[21] = 8.08827e+02;
    rho[22] = 8.00387e+02;
    rho[23] = 7.91844e+02;
    rho[24] = 7.83204e+02;
    rho[25] = 7.74473e+02;
    rho[26] = 7.65654e+02;
    rho[27] = 7.56754e+02;
    rho[28] = 7.47777e+02;
    rho[29] = 7.38729e+02;
    rho[30] = 7.29615e+02;
    rho[31] = 7.20441e+02;
```

```
        rho[32] = 7.11210e+02;
        rho[33] = 7.01929e+02;
        rho[34] = 6.92603e+02;
        rho[35] = 6.83236e+02;
        rho[36] = 6.73834e+02;
        rho[37] = 6.64401e+02;
        rho[38] = 6.54943e+02;
        rho[39] = 6.45464e+02;
        rho[40] = 6.35970e+02;
        rho[41] = 6.26464e+02;
        rho[42] = 6.16953e+02;
        rho[43] = 6.07439e+02;
        rho[44] = 5.97928e+02;
        rho[45] = 5.88425e+02;
        rho[46] = 5.78934e+02;
        rho[47] = 5.69458e+02;
        rho[48] = 5.60003e+02;
        rho[49] = 5.50573e+02;
        rho[50] = 5.41171e+02;
        rho[51] = 5.31802e+02;
        rho[52] = 5.22470e+02;
        rho[53] = 5.13177e+02;
        rho[54] = 5.03929e+02;
        rho[55] = 4.94729e+02;
          }

   // Member Functions

      //- Evaluate the function and return the result
      scalar f(scalar, scalar T) const
      {
      /* instead of returning NSRDSfunc5, we make interpolation in the density table
rho[56]. */
            scalar rho_ = 0.0;
            for(int i=0; i<55; i++){
                  if(T>=10*i && T<10*(i+1))
                  rho_ = rho[i]+(T-10*i)*(rho[i+1]-rho[i])/10;
                  }
            return rho_;
      }
……
```

Similarly, we can modify the rest functions, including NSRDSfuncgPv, NSRDSfuncgHl, NSRDSfuncgCp, NSRDSfuncgH, NSRDSfuncgCpg, NSRDSfuncgMu, NSRDSfuncgK, and NSRDSfuncgSigma. The data for the above properties can be found in the corresponding appendix files.

e. Create Make/files and Make/options, and compile the user thermophysical functions.

cd \
$WM_PROJECT_USER_DIR/thermophysicalModels/thermophysicalFunctions/NSRDSfunctions/
mkdir –r Make
touch Make/files
the contents of Make/files
NSRDSfuncgRho/NSRDSfuncgRho.C
NSRDSfuncgPv/NSRDSfuncgPv.C
NSRDSfuncgHl/NSRDSfuncgHl.C
NSRDSfuncgCp/NSRDSfuncgCp.C
NSRDSfuncgH/NSRDSfuncgH.C
NSRDSfuncgCpg/NSRDSfuncgCpg.C
NSRDSfuncgMu/NSRDSfuncgMu.C
NSRDSfuncgK/NSRDSfuncgK.C
NSRDSfuncgSigma/NSRDSfuncgSigma.C

LIB = $(FOAM_USER_LIBBIN)/libmyThermophysicalFunctions

touch Make/options
the contents of Make/options
EXE_INC = \
    -I$(LIB_SRC)/thermophysicalModels/thermophysicalFunctions/lnInclude

LIB_LIBS = \
    -lthermophysicalFunctions

Compile the library

wmake libso

A new thermophysical function library, libmyThermophysicalFunctions, will appear in the directory $FOAM_USER_LIBBIN.

3.2 Create a new liquid class gasoline

a. Make a directory for gasoline liquid in user working directory.
   mkdir –p $WM_PROJECT_USER_DIR/src/thermophysicalModels/liquids /
b. Go to user liquids directory, and copy the IC8H18/ directory.
   cd $WM_PROJECT_USER_DIR/thermophysicalModels/thermophysicalFunctions/ liquids
   cp –r $FOAM_SRC/ thermophysicalModels/thermophysicalFunctions/ liquids/IC8H18 .
c. rename IC8H18 to gasoline.
   mv IC8H18 gasoline
   cd gasoline
   *rename IC8H18 gasoline **
   sed –i s/"IC8H18"/"gasoline"/g gasoline.C
   sed –i s/"IC8H18"/"gasoline"/g gasoline.H
   sed –i s/"IC8H18"/"gasoline"/g gasolineI.H
   rm gasoline.dep
d. modify the gasoline class.
   The file gasoline.H is modified as follows,

```
……
#include "NSRDSfunc5.H"
#include "NSRDSfunc6.H"
#include "NSRDSfunc7.H"
#include "NSRDSfunc14.H"
#include "APIdiffCoefFunc.H"
// include the gasoline property functions
#include "NSRDSfuncgRho.H"
#include "NSRDSfuncgPv.H"
#include "NSRDSfuncgHl.H"
#include "NSRDSfuncgCp.H"
#include "NSRDSfuncgH.H"
#include "NSRDSfuncgCpg.H"
#include "NSRDSfuncgMu.H"
#include "NSRDSfuncgK.H"
#include "NSRDSfuncgSigma.H"

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

namespace Foam
{

/*---------------------------------------------------------------------------*\
                    Class gasoline Declaration
\*---------------------------------------------------------------------------*/
```

```
class gasoline
:
    public liquid
{
    // Private data

        NSRDSfuncgRho rho_;
        NSRDSfuncgPv pv_;
        NSRDSfuncgHl hl_;
        NSRDSfuncgCp cp_;
        NSRDSfuncgH h_;
        NSRDSfuncgCpg cpg_;
        NSRDSfunc4 B_;
        NSRDSfuncgMu mu_;
        NSRDSfunc2 mug_;
        NSRDSfuncgK K_;
        NSRDSfunc2 Kg_;
        NSRDSfuncgSigma sigma_;
        APIdiffCoefFunc D_;

public:

    //- Runtime type information
    TypeName("gasoline");


    // Constructors

        //- Construct null
        gasoline();

        //- Construct from components
        gasoline
        (
            const liquid& l,
            const NSRDSfuncgRho& density,
            const NSRDSfuncgPv& vapourPressure,
            const NSRDSfuncgHl& heatOfVapourisation,
            const NSRDSfuncgCp& heatCapacity,
            const NSRDSfuncgH& enthalpy,
            const NSRDSfuncgCpg& idealGasHeatCapacity,
```

```
        const NSRDSfunc4& secondVirialCoeff,
        const NSRDSfuncgMu& dynamicViscosity,
        const NSRDSfunc2& vapourDynamicViscosity,
        const NSRDSfuncgK& thermalConductivity,
        const NSRDSfunc2& vapourThermalConductivity,
        const NSRDSfuncgSigma& surfaceTension,
        const APIdiffCoefFunc& vapourDiffussivity
    );

    //- Construct from Istream
    gasoline(Istream& is);
……
```

The file gasoline.C is modified as follows,

```
// * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * //

Foam::gasoline::gasoline()
:
/* the gasoline liquid properties are approximated based on gasoline surrotates in mole fraction except for mole weight and critical temperature */
  liquid(113.228,  548.00,  3.12419e+6,  0.411,  0.265,  171.80,  4.05773e-2,  376.30,  0.0, 0.2941, 1.5669e+4),
  rho_(),
  pv_(),
  hl_(),
  cp_(),
  // NN: enthalpy, h_, is not used in the sprayModel.
  // For consistency, the enthalpy is derived from hlat and hl.
  // It is, however, convenient to have it available.
  h_(),
  cpg_(),
  // B is the same as n-octane C8H18;
  B_(0.00239777293379205,  -2.81394717721109,  -585042.589139551,  -1.11265768486663e+18, 1.40968738783693e+20),
  mu_(),
  //Mug is approximated using mixing rule in equation 1.
  mug_(7.77735e-08, 8.30817e-01, 4.83952e+01, 0.0),
  K_(),
  //Kg is approximated using mixing rule in equation 1.
  Kg_(-6.98476e-03, 1.20363e+00, -3.00945e+02, -2.40050e+05),
  sigma_(),
  D_(147.18, 20.1, 114.231, 28) // NN: Same as nHeptane
```

```
{}


Foam::gasoline::gasoline
(
    const liquid& l,
    const NSRDSfuncgRho& density,
    const NSRDSfuncgPv& vapourPressure,
    const NSRDSfuncgHl& heatOfVapourisation,
    const NSRDSfuncgCp& heatCapacity,
    const NSRDSfuncgH& enthalpy,
    const NSRDSfuncgCpg& idealGasHeatCapacity,
    const NSRDSfunc4& secondVirialCoeff,
    const NSRDSfuncgMu& dynamicViscosity,
    const NSRDSfunc2& vapourDynamicViscosity,
    const NSRDSfuncgK& thermalConductivity,
    const NSRDSfunc2& vapourThermalConductivity,
    const NSRDSfuncgSigma& surfaceTension,
    const APIdiffCoefFunc& vapourDiffussivity
)
:
    liquid(l),
    rho_(density),
    pv_(vapourPressure),
    hl_(heatOfVapourisation),
    cp_(heatCapacity),
    h_(enthalpy),
    cpg_(idealGasHeatCapacity),
    B_(secondVirialCoeff),
    mu_(dynamicViscosity),
    mug_(vapourDynamicViscosity),
    K_(thermalConductivity),
    Kg_(vapourThermalConductivity),
    sigma_(surfaceTension),
    D_(vapourDiffussivity)
{}
……
```

e. Create Make/files and Make/options, and compile new liquid class gasoline.

cd $WM_PROJECT_USER_DIR/thermophysicalModels/liquids/gasoline/

mkdir –r Make

touch Make/files

the contents of Make/files

gasoline.C

LIB = $(FOAM_USER_LIBBIN)/libmyLiquids

touch Make/options
the contents of Make/options
EXE_INC = \
-I$(LIB_SRC)/thermophysicalModels/liquids/lnInclude \
-I$(LIB_SRC)/thermophysicalModels/thermophysicalFunctions/lnInclude \
-I$(WM_PROJECT_USER_DIR)/src/thermophysicalModels/thermophysicalFunctions/NSRDSfunctions/lnInclude \

LIB_LIBS = \
-lliquids \
-lthermophysicalFunctions \
-L$(WM_PROJECT_USER_DIR)/lib/$(WM_OPTIONS) \
-lmyThermophysicalFunctions

Compile the library,

wmake libso

A new liquid library, libmyLiquids, will appear in the directory $FOAM_USER_LIBBIN.

4. A test of implementation through a case study

A gasoline liquid spray into a constant volume will be used to test our implementation of gasoline properties. Before we set up the case, the reitzDiwakar breakup source code which is related to surface tension, is changed to test the implementation.

4.1 Modify the reitzDiwakar breakup model

a. Make a directory for new reitzDiwakar breakup model.
   mkdir –p \
   $WM_PROJECT_USER_DIR/src/lagrangian/dieselSpray/spraySubModels/breakupModel/
b. Go to the user breakupModel directory, and copy the original reitzDiwakar directory.
   cd \
   $WM_PROJECT_USER_DIR/src/lagrangian/dieselSpray/spraySubModels/breakupModel/
   cp –r \

$FOAM_SRC/lagrangian/dieselSpray/spraySubModels/breakupModel/reitzDiwakar/ .

c. Rename reitzDiwakar as myReitzDiwakar.

cd reitzDiwakar
*rename reitzDiwakar myReitzDiwakar **
sed –i s/"reitzDiwakar"/"myReitzDiwakar"/g myReitzDiwakar.C
sed –i s/"reitzDiwakar"/"myReitzDiwakar"/g myReitzDiwakar.H
rm myReitzDiwakar.dep

d. Modify myReitzDiwakar.C as follows,

```
……
 // ideal gas law to evaluate density
   scalar rhoAverage = pressure/R/Taverage;
   scalar nuAverage = muAverage/rhoAverage;
   scalar sigma = fuels.sigma(pressure, p.T(), p.X());
 // output the temperature and corresponding surface tension
   Info<< "T = " << p.T() << endl;
   Info<< "sigma = " << sigma << endl;
   // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
   //     The We and Re numbers are to be evaluated using the 1/3 rule.
   // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
……
```

e. Create Make/files and Make/options, and compile myReitzDiwakar breakup model.

cd \
$WM_PROJECT_USER_DIR/src/lagrangian/dieselSpray/spraySubModels/breakupModel/reitzDiwakar/
mkdir –r Make
touch Make/files
the contents of Make/files
myReitzDiwakar.C

LIB = $(FOAM_USER_LIBBIN)/libmyReitzDiwakar

touch Make/options
the contents of Make/options

```
EXE_INC = \
   -I$(LIB_SRC)/finiteVolume/lnInclude \
   -I$(LIB_SRC)/lagrangian/basic/lnInclude \
   -I$(LIB_SRC)/lagrangian/dieselSpray/lnInclude \
   -I$(LIB_SRC)/turbulenceModels \
   -I$(LIB_SRC)/turbulenceModels/compressible/turbulenceModel \
   -I$(LIB_SRC)/turbulenceModels/compressible/RAS/lnInclude \
   -I$(LIB_SRC)/turbulenceModels/LES/LESdeltas/lnInclude \
   -I$(LIB_SRC)/turbulenceModels/compressible/LES/lnInclude \
```

```
    -I$(LIB_SRC)/thermophysicalModels/basic/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/liquids/lnInclude \
    -I$(WM_PROJECT_USER_DIR)/src/thermophysicalModels/liquids/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/liquidMixture/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/thermophysicalFunctions/lnInclude \
    -
I$(WM_PROJECT_USER_DIR)/src/thermophysicalModels/thermophysicalFunctions/NSR
DSfunctions/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/specie/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/reactionThermo/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/pdfs/lnInclude

LIB_LIBS = \
    -lfiniteVolume \
    -llagrangian \
    -ldieselSpray \
    -lcompressibleRASModels \
    -lcompressibleLESModels \
    -lLESdeltas \
    -lliquids \
    -lliquidMixture \
    -lthermophysicalFunctions \
    -lspecie \
    -lpdf \
    -L$(WM_PROJECT_USER_DIR)/lib/$(WM_OPTIONS) \
    -lmyLiquids
```

Compile the library,

wmake libso

A new breakup library, libmyReitzDiwakar, will appear in the directory $FOAM_USER_LIBBIN.

4.2 Setup a case for gasoline hollow cone spray in a constant volume

a. Go to user run directory, and copy the aachenBomb case in user run directory
   run
   cp –r $FOAM_TUTORIALs/combustion/dieselFoam/aachenBomb .

b. Modify the chemkin/ directory. C_8H_{15} is used to represent gasoline fuel when it is in gas phase. Since the combustion is not considered in this case, we don't change the reaction rate constants when we switch a fuel. The chem.inp is modified as follows,

```
ELEMENTS
 H  O   C  N  AR
END
SPECIE
C8H15 O2 N2 CO2 H2O
END
REACTIONS
 C8H15 + 11.75O2          => 8CO2 + 7.5H2O       5.00E+8  0.0   15780.0! 1
          FORD    / C8H15          0.25 /
          FORD    / O2 1.5 /
END
```

Correspondingly, the thermo property of C_8H_{15} is added in the end of therm.dat.

```
C8H15          P 4/85C  8.H 15.   0.   0.G   200.000  5000.000 1396.0      1
 2.15002114e+01    3.36729730e-02-1.16006708e-05    1.82223584e-09-1.06962828e-13
2
-2.59374406e+04-9.22017472e+01-6.34105767e-01    6.77277031e-02    5.91125179e-06
3
-5.53067539e-08 2.69930010e-11-1.77199967e+04 3.02314987e+01              4
```

c. Modify the constant/ directory.

Modify the geometry in polyMesh/blockMeshDict file,

```
convertToMeters 0.001;

vertices
(
   (20 -20 0)
   (20 20 0)
   (-20 20 0)
   (-20 -20 0)
   (60.10 -60.10 0)
   (60.10 60.10 0)
   (-60.10 60.10 0)
   (-60.10 -60.10 0)
   (20 -20 -205)
   (20 20 -205)
   (-20 20 -205)
   (-20 -20 -205)
```

```
    (60.10 -60.10 -205)
    (60.10 60.10 -205)
    (-60.10 60.10 -205)
    (-60.10 -60.10 -205)
);

blocks
(
    hex (0 3 2 1 8 11 10 9) (25 25 50) simpleGrading (1 1 2)
    hex (0 1 5 4 8 9 13 12) (25 20 50) simpleGrading (1 2 2)
    hex (1 2 6 5 9 10 14 13) (25 20 50) simpleGrading (1 2 2)
    hex (2 3 7 6 10 11 15 14) (25 20 50) simpleGrading (1 2 2)
    hex (3 0 4 7 11 8 12 15) (25 20 50) simpleGrading (1 2 2)
);

edges
(
    arc 4 5 (85 0 0)
    arc 5 6 (0 85 0)
    arc 6 7 (-85 0 0)
    arc 7 4 (0 -85 0)
    arc 12 13 (85 0 -205)
    arc 13 14 (0 85 -205)
    arc 14 15 (-85 0 -205)
    arc 15 12 (0 -85 -205)
);

patches
(
    wall cylinder
    (
        (4 5 13 12)
        (5 6 14 13)
        (6 7 15 14)
        (7 4 12 15)
    )
    wall cellhead
    (
        (0 3 2 1)
        (0 1 5 4)
        (1 2 6 5)
        (2 3 7 6)
```

```
    (0 4 7 3)
   )

   wall cellbottom
   (
    (8 11 10 9)
    (8 9 13 12)
    (9 10 14 13)
    (10 11 15 14)
    (8 12 15 11)
   )
);

mergePatchPairs
(
);
```

Modify injectorProperties as follows,

```
    injectorType       unitInjector;

     unitInjectorProps
     {
        position       (0 0 -0.002);
        direction      (0 0 -1);
        diameter       0.00046;
        Cd             0.9;
        mass           1.369e-05;
        nParcels       5000;

        X
        (
           1.0
        );

        massFlowRateProfile
        (
           (0 0.1)
           (0.0002 1)
           (0.0004 1)
           (0.0006 0.1)
```

```
        );

        temperatureProfile
        (
          (0.0     243.0)
          (0.0006   243.0)
        );
……
```

Modify sprayProperties as follows,

```
……
breakupModel    myReitzDiwakar;
……
myReitzDiwakarCoeffs
{
   Cbag          6;
……
hollowConeInjectorCoeffs
{
   dropletPDF
   {
      pdfType       RosinRammler;
      RosinRammlerPDF
      {
        minValue      1e-06;
        maxValue      8.00e-5;
        d             ( 3.00e-5 );
        n             ( 3 );
      }

      exponentialPDF
      {
        minValue      0.0001;
        maxValue      0.001;
        lambda        ( 10000 );
      }
   }

   innerConeAngle  ( 80 );
   outerConeAngle  ( 90 );
}
```

……

Modify thermophysicalProperties as follows,

thermoType hPsiMixtureThermo<reactingMixture<gasThermoPhysics>>;

CHEMKINFile "$FOAM_CASE/chemkin/chem.inp";

CHEMKINThermoFile "$FOAM_CASE/chemkin/therm.dat";

inertSpecie N2;

liquidComponents (C8H15);

liquidProperties
{
C8H15 gasoline defaultCoeffs;
}

d. Modify system/ directory
Add the following at the end of controlDict,
libs ("libmyLiquids.so");
libs ("libmyReitzDiwakar.so");

e. Modify 0/ directory

Remove the front and back boundary, and modify wall boundary to cylinder boundary.

Add cellhead and cellbottom boundary, and they have the same boundary value as the cylinder boundary.

4.3 Run the gasoline hollow cone spray case

cd $WM_PROJECT_USER_DIR/run/

blockMesh –case aachenBomb

dieselFoam –case aachenBomb >& aachenBomb/log &

Check the log file for the information about surface tension. For example, we got,

T = 320.326

sigma = 0.0163615

which is exactly the same as we have implemented. Figure 2 shows the droplet distribution of gasoline hollow cone spray in a constant volume at different time steps.

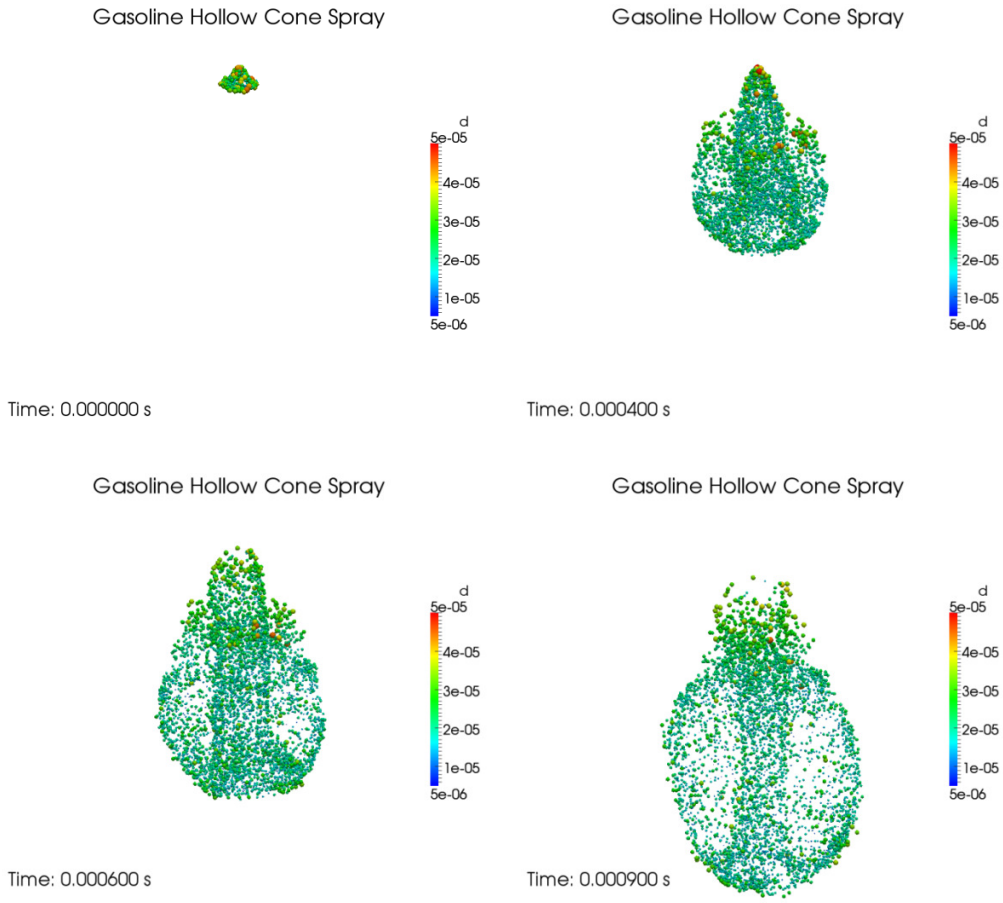

Figure 2 Droplet distribution of gasoline hollow cone spray in a constant volume

Reference

[1] Joseph A. Schetz, Allen E. Fuhs, Handbook of Fluid Dynamics and Fluid Machinery, volume one Fundamentals of Fluid Dynamics. p 156-157.

[2] Robert C. Reid, John M. Prausnitz, Bruce E. Poling, The Properties of Gases & Liquids Fourth Edition. p 75.