CFD WITH OPENSOURCE SOFTWARE, ASSIGNMENT 3

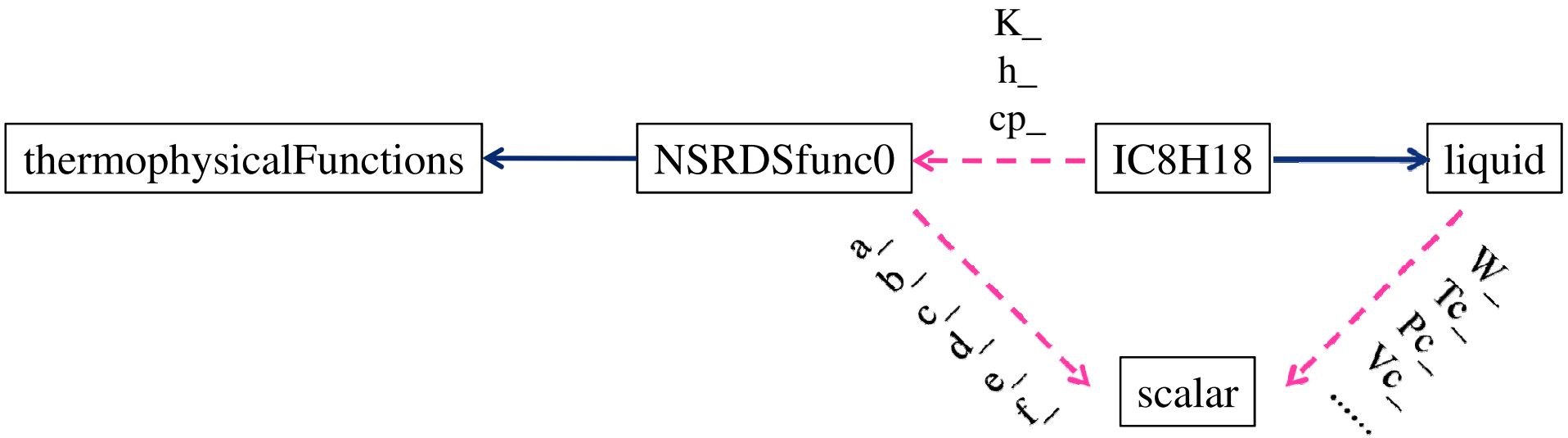
Implementation of Gasoline Properties in OpenFOAM Library

Chen Huang

Division of Combustion
Department of Applied Mechanics
Chalmers University of Technology

Dec. 2009

Description of thermophysical Functions and liquids library



- The abstract class and its sub-classes are linked with blue arrows
- A pink dashed arrow is used if a class is contained or used by another class. The arrow is labeled with the variable(s) through which the pointed class or struct is accessible.

Gasoline Properties

Properties from KIVA fuel library (table):

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), density (rho).

Properties from thermophysical functions:

second virial coefficient (B) (C_8H_{18}), vapour diffusivity (D) (C_7H_{16}).

Properties from mixing rule:

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).

Create new thermophysical functions (1)

- Make a directory for thermophysical functions in user working directory.
- Go to OpenFOAM NSRDSfunctions directory, and copy the NSRDfunc.
- Rename them to the corresponding gasoline properties.
  - *rename NSRDSfunc5 NSRDSfuncgRho **
- We only change NSRDSfunc5 to NSRDSfuncgRho as an example,
  - *vi NSRDSfuncgRho.H*

……

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

Create new thermophysical functions (2)

```
//- 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;
.....}
```

Create new thermophysical functions (3)

```
// 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_;
    }
……
```

Create new thermophysical functions (4)

•Make/files

NSRDSfuncgRho/NSRDSfuncgRho.C

NSRDSfuncgPv/NSRDSfuncgPv.C

……

NSRDSfuncgSigma/NSRDSfuncgSigma.C

LIB = $(FOAM_USER_LIBBIN)/libmyThermophysicalFunctions

•Make/options

```
EXE_INC = \
    -I$(LIB_SRC)/thermophysicalModels/thermophysicalFunctions/lnInclude

LIB_LIBS = \
    -lthermophysicalFunctions
```

Create a new liquid class gasoline (1)

- *mkdir –p $WM_PROJECT_USER_DIR/src/thermophysicalModels/liquids /*
- *cd $WM_PROJECT_USER_DIR/thermophysicalModels/thermophysicalFunctions/ liquids*
- *cp –r $FOAM_SRC/ thermophysicalModels/thermophysicalFunctions/ liquids/IC8H18 .*
- 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*

Create a new liquid class gasoline (2)

- Modify file gasoline.H,

```
……
#include "NSRDSfunc14.H"
#include "APIdiffCoefFunc.H"
// include the gasoline property functions
#include "NSRDSfuncgRho.H"
#include "NSRDSfuncgPv.H"
……
#include "NSRDSfuncgSigma.H“
……
class gasoline
:   public liquid
{   // Private data
      NSRDSfuncgRho rho_;
      NSRDSfuncgPv pv_;
……
```

```
//- Construct from components
      gasoline
      (   const liquid& l,
          const NSRDSfuncgRho& density,
          const NSRDSfuncgPv& vapourPressure,
……
          const NSRDSfunc4& secondVirialCoeff,
          const NSRDSfuncgMu& dynamicViscosity,
          const NSRDSfunc2& vapourDynamicViscosity,
          const NSRDSfuncgK& thermalConductivity,
          const NSRDSfunc2& vapourThermalConductivity,
          const NSRDSfuncgSigma& surfaceTension,
          const APIdiffCoefFunc& vapourDiffussivity
      );
……
```

Create a new liquid class gasoline (3)

- Modify gasoline.C,

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_(),

……

// 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),

……

{}

Create a new liquid class gasoline (4)

- Modify gasoline.C,

……

Foam::gasoline::gasoline

(

const liquid& l,

const NSRDSfuncgRho& density,

……

const NSRDSfunc4& secondVirialCoeff,

……)

liquid(l),

rho_(density),

……

B_(secondVirialCoeff),

……

{}

……

Create a new liquid class gasoline (5)

- Make/files

gasoline.C

LIB = $(FOAM_USER_LIBBIN)/libmyLiquids

- 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
```

Modify the reitzDiwakar breakup model (1)

- 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;
……
```

Modify the reitzDiwakar breakup model (2)

- Make/files

myReitzDiwakar.C

LIB = $(FOAM_USER_LIBBIN)/libmyReitzDiwakar

- Make/options

EXE_INC = \

-I$(LIB_SRC)/finiteVolume/lnInclude \

……

-I$(WM_PROJECT_USER_DIR)/src/thermophysicalModels/liquids/lnInclude \

……

-I$(WM_PROJECT_USER_DIR)/src/thermophysicalModels/thermophysicalFunctions/NSRDSfunctions/lnInclude \

……

LIB_LIBS = \

……

- -L$(WM_PROJECT_USER_DIR)/lib/$(WM_OPTIONS) \
- -lmyLiquids

A case for gasoline hollow cone spray in a constant volume (1)

- Based on aachenBomb
- chemkin/chem.inp

```
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
```

- chemkin/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
```

A case for gasoline hollow cone spray in a constant volume (2)

- constant/sprayProperties,

breakupModel myReitzDiwakar;

……

myReitzDiwakarCoeffs

{

Cbag 6;

……

- constant/thermophysicalProperties,

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

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

inertSpecie N2;

liquidComponents (C8H15);

liquidProperties

{

C8H15 gasoline defaultCoeffs;

}

- System/ controlDict,

……

libs ("libmyLiquids.so");

libs ("libmyReitzDiwakar.so");

- Run and check the log file

We got

T = 320.326

sigma = 0.0163615

A case for gasoline hollow cone spray in a constant volume (3)