Energy Technology Center
Luleå University of Technology

CFD with OpenSource software, assignment 3

Tutorial dieselFoam

Author:
Per Carlsson

Peer reviewed by:
NaiXian Lu
Håkan Nilsson

February 17, 2009

Chapter 1

Tutorial dieselFoam

1.1 Introduction

This tutorial describes how to pre-process, run and post-process a case involving compressible reacting flow with Lagrangian evaporating particles in a three-dimensional domain. It also describes how to copy the solver, copy an evaporation model and how to add a second material to the discrete particles.

The geometry consists of a block filled with air, with a 0.01x0.01 meter base and a length of 0.1 meter (figure 1.1). An injector is centrally placed on the top boundary where n-Heptane (C_7H_{16}) is injected. When the discrete droplets enter the domain they evaporate and combustion takes place in the gas phase. There are several gas phase reaction schemes supplied with the case ranging from a reaction scheme with 5 species and one reaction up to a reaction scheme involving ∼300 reactions and 56 species.

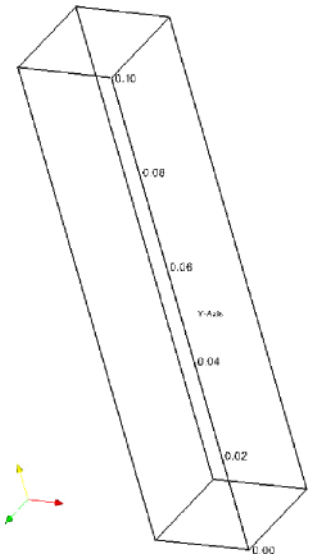


Figure 1.1: Geometry of the dieseFoam tutorial case.

1.2 Pre-processing

This section covers the necessary setup needed to get the dieselFoam case running with chemistry, the tutorial also covers a brief introduction to reacting flows in numerical simulations.

1.2.1 Getting started

Copy the dieselFoam tutorial to the run directory.

```
cp -r  $FOAM_TUTORIALS/dieselFoam/aachenBomb $FOAM_RUN
cd $FOAM_RUN/aachenBomb
```

The file structure of the dieselFoam case is similar to other OpenFOAM tutorials where the case directory has a `/0`, `/constant` and `/system` directory. The dieselFoam case also has a `/chemkin` directory where the gas phase reaction schemes are specified. As usual in OpenFOAM tutorials; the solver-, write- and time-control can be found in the `/system` directory and the mesh setup in `/constant/polyMesh`.

1.2.2 Boundary and initial conditions

Since there are neither outlets nor inlets, apart from the injector, the boundary conditions for the dieselFoam tutorial are very simple. All walls are modeled as adiabatic. The boundary conditions for the injector can be found in `/constant/injectorProperties` file see example below.

```
        injectorType        unitInjector;

        unitInjectorProps
        {
            position        (0 0.0995 0);
            direction       (0 -1 0);
            diameter        0.00019;
            Cd              0.9;
            mass            6e-06;
            temperature     320;
            nParcels        5000;

            X
            (
                1.0
            );
    massFlowRateProfile
            (
                (0 0.1272)
                (4.16667e-05 6.1634)
                (8.33333e-05 9.4778)
                ...
            );
```

In the `/constant/injectorProperties` file it can be seen that the injector is located 0.5 mm from the top of the domain and injects in the negative y direction. Furthermore, the injector nozzle diameter, the nozzle discharge coefficient, mass and temperature of the parcels can be found here as well as the total number of injected parcels. The X is the mass fraction of a specific specie which will be described further in section 1.5. The `massFlowRateProfile` specifies how the mass flow rate should vary over time. From time $t_0 \rightarrow t_1$ the mass flow rate is $\dot{m}_0$. This done in order to simulate opening and closing of the injector. The left column of the `massFlowRateProfile` is t_i and the right, $\dot{m}_i$.

It is possible to define different kinds of injectors, however, in this tutorial only the `unitInjector` will be used. In the `/constant/sprayProperties` file the user can specify what will happen to the droplets as they enter the domain, see table 1.1.

Model	General meaning
subCycles	Minimum number of Lagrangian sub cycles
atomizationModel	How atomization is treated
includeOscillation	Droplet deformation; will effect droplet drag coefficient
breakupModel	If secondary break up is used
injectorModel	Which injector model to use
collisionModel	Particle - particle interaction
evaporationModel	Which evaporation model to use
heatTransferModel	Particle heat transfer model
dispersionModel	If turbulent dispersion is used or not
dragModel	Particle drag model
wallModel	What happens to particles hitting the walls

Table 1.1: Spray sub-models for the dieselFoam tutorial

The initial conditions are found in the `/0` directory and are summarized in table 1.2. Not all initial conditions are specified here since they are not necessary to get the case running. Note that the initial mass fractions for N_2 and O_2 corresponds to air and that the initial condition for *spray* is empty since it is specified in the `/constant/injectorProperties`-file.

Variable	Initial conditions
ϵ	internalField uniform 90.0, walls zeroGradient
k	internalField uniform 1.0, walls zeroGradient
N_2	internalField uniform 0.766, walls zeroGradient
O_2	internalField uniform 0.233, walls zeroGradient
p	internalField uniform 5e+06, walls zeroGradient
spray	empty
T	internalField uniform 800, walls zeroGradient
U	internalField uniform (0 0 0), walls uniform (0 0 0)

Table 1.2: Initial conditions for the dieselFoam tutorial

1.2.3 Physical properties

In the `/constant` directory the properties files for chemistry, environment, spray, combustion, injector, RAS and thermophysical. The spray and injector properties are described in section 1.2.2 and the RAS properties are thoroughly described in the OpenFOAM user guide and are therefore not described here. The properties files are summarized in table 1.3.

Properties file	General content
chemistryProperties	Chemical reactions are included if `chemistry` is switched `on` Specification and settings for the discretization scheme used to solve the chemistry ODEs
environmentalProperties	Gravity
combustionProperties	Ignition point on or off, timing and duration of ignition point
thermophysicalProperties	Specify the mixture type and which gas phase reaction scheme to use as well as thermodynamic database

Table 1.3: Properties files and general content for the dieselFoam tutorial

A certain mixture type may be more or less suited for a combustion problem and depends on if the flame is non-, partly or full-premixed. In the `thermophysicalProperties` file it is possible to specify the mixture types, several are available[1]. Parts of the `thermophysicalProperties` file is listed below, notice that the location of the `CHEMKINThermoFile` has been changed from `"~OpenFOAM/thermoData/therm.dat"` to `"$FOAM_CASE/chemkin/therm.dat"`.

```
thermoType hMixtureThermo<reactingMixture>;
CHEMKINFile         "$FOAM_CASE/chemkin/chem.inp";
CHEMKINThermoFile   "$FOAM_CASE/chemkin/therm.dat";
```

In this tutorial we will use the predefined `reactingMixture` together with the thermophysical model `hMixtureThermo` which calculates enthalpy for combustion mixture. The choice of mixture and thermo physical model depends both on the physics of the flame and which variables that are needed for the combustion model. The `thermophysicalProperties` file also contains information on where the gas phase reactions are defined as well which thermo dynamic data base to use. The gas phase reactions are specified in the `"$FOAM_CASE/chemkin/chem.inp"` file and the thermo dynamic data base in the `"$FOAM_CASE/chemkin/therm.dat"` file. The `therm.dat` and `chem.inp`-file will be described further in section 1.2.4 as well as the combustion model.

1.2.4 Chemistry

When the droplets enter the domain they start to evaporate. The $C_7H_{16}(g)$ [2] then reacts with oxygen forming CO_2 and H_2O. However, this reaction can be called a global reaction and is not what would happen if $C_7H_{16}(g)$ would burn with air in a real combustor or burner. As an example, think about hydrogen burning with pure oxygen, see reaction 1.1.

$$H_2 + \frac{1}{2}O_2 \Leftrightarrow H_2O \tag{1.1}$$

However, in order for the hydrogen to react with oxygen, the bond between the atoms first have to be broken and a more complex reaction scheme is required, see reaction 1.2 to 1.6.

$$H_2 \Leftrightarrow 2H \tag{1.2}$$

$$O_2 \Leftrightarrow 2O \tag{1.3}$$

[1] http://www.opencfd.co.uk/openfoam/doc/thermophysicalModels.html

[2] Notation (g) meaning that the specie is in gas phase. Similar notation is (s) for solid and (l) for liquid. The notation is used here to emphasize that no heterogeneous reactions are taking place.

$$H + O \Leftrightarrow OH \tag{1.4}$$

$$OH + H_2 \Leftrightarrow H_2O + H \tag{1.5}$$

$$OH + H \Leftrightarrow H_2O \tag{1.6}$$

So, instead of having two reactions (backward and forward) with three species we have ten reactions (backward and forward) with six species (the scheme described above is *ad hoc* and is just used to describe the difficulties describing chemistry in numerical simulations). The transport equations for these species have to be solved as well as the ODEs for the reactions. In this tutorial the gas phase reactions are specified in the `/chemkin/chem.inp`-file, see below.

```
REACTIONS
 C7H16 + 11O2            => 7CO2 + 8H2O        5.00E+8  0.0   15780.0! 1
        FORD    / C7H16 0.25 /
        FORD    / O2 1.5 /
END
```

The entries behind the reaction in the `/chemkin/chem.inp`-file are Arrhenius coefficient that are used to calculate the chemical reaction rate. `FORD` is the forward reaction order. The chemical reaction rate will be calculated according to equations 1.7 and 1.8

$$k_f = AT^b \cdot exp\left(\frac{-E_a}{RT}\right) \tag{1.7}$$

Where k_f is the forward reaction coefficient, A pre exponential factor, b temperature exponent, E_a activation energy, R ideal gas coefficient and T temperature.

$$\dot{\omega_i} = \frac{\mathrm{d}[product]}{\mathrm{d}t} = -k_f * [fuel]^c [oxidizer]^d \tag{1.8}$$

Where $\dot{\omega_i}$ is the chemical reaction rate, t time, c and d the forward reaction order and, $[C]$ is concentration of specie C. In simplified terms the `/chemkin/chem.inp`-file can thus be written as:

```
REACTIONS
 fuel + oxidizer            => product        A  b  Ea
        FORD    / fuel c /
        FORD    / oxidizer d /
END
```

Due to the numerical cost only the simplest scheme (chem.inp) will be used in this tutorial but the user is encouraged to look in the chem.inp.full file to see how a complex but still reduced reaction scheme might look like.

With out going into great detail regarding thermodynamics in combustion processes[3], it is possible to realize that when a fuel and an oxidizer react, they will produce heat. The amount of heat released from the flame as well as the flame temperature can be predicted using thermodynamics. The thermodynamic data base is located in the `/chemkin/therm.dat`-file. An example from the `/chemkin/therm.dat`-file is listed below.

```
C7H16             P10/95 C   7H  16    0    0G   200.000  5000.000  1391.000    1
 2.22148969e+01 3.47675750e-02-1.18407129e-05 1.83298478e-09-1.06130266e-13    2
-3.42760081e+04-9.23040196e+01-1.26836187e+00 8.54355820e-02-5.25346786e-05    3
 1.62945721e-08-2.02394925e-12-2.56586565e+04 3.53732912e+01                   4
```

[3]Combustion 4th edition, Chapter 4, J.Warnatz et al. Springer 2006

- The first row contains species name, date (not used in the code), atomic symbols and formula, phase of species (S, L, or G for gas), low temperature, high temperature and common temperature (if needed).
- The second row contains coefficients $a_1 - a_5$ in equation 1.9 for upper temperature interval.
- The third row contains coefficients a_6, a_7 for upper temperature interval, and a_1, a_2, and a_3 for lower.
- The fourth row contains coefficients a_4, a_5, a_6, a_7 for lower temperature interval.

From these constants, (NASA) polynomials for specific heat C_p, enthalpy H and entropy S can be calculated.

$$\frac{C_p}{R} = a_1 + a_2 \cdot T + a_3 \cdot T^2 + a_4 \cdot T^3 + a_5 \cdot T^4 \tag{1.9}$$

$$\frac{H}{RT} = a_1 + \frac{a_2 T}{2} + \frac{a_3 T^2}{3} + \frac{a_4 T^3}{4} + \frac{a_5 T^4}{5} + \frac{a_6}{T} \tag{1.10}$$

$$\frac{S}{R} = a_1 \ln T + a_2 T + \frac{a_3 T^2}{2} + \frac{a_4 T^3}{3} + \frac{a_5 T^4}{4} + a_7 \tag{1.11}$$

The specific heat C_p, enthalpy H and entropy S are then used in the code to solve the conservation equations.

The combustion model for this tutorial is a partially stirred reactor concept model developed at Chalmers Gothenburg described by equations 1.12 and 1.13

$$CST_i = \frac{\tau_{chem}}{\tau_{mix} + \tau_{chem}} \dot{\omega}_i \tag{1.12}$$

Where CST_i is the chemical source term, τ_{chem} chemical time $\propto \frac{1}{k_f}$ and τ_{mix} mixing time. The mixing time τ_{mix} is calculated according to

$$\tau_{mix} = C_{mix} \sqrt{\frac{\mu_{eff}}{\rho \epsilon}}. \tag{1.13}$$

Where C_{mix} is a constant specified in the `/constant/combustionProperties`-file, μ_{eff} is the effective viscosity, ρ density and ϵ rate of dissipation of turbulent kinetic energy. The combustion model can be found on lines 81-95 in `$FOAM_SOLVERS/combustion/dieselFoam/dieselFoam.C`

1.3 Running the code

Remove `ft` and `fu` and in the `aachenBomb/0` directory since these are not needed for this setup (keeping them will result in post-processing problems).

```
cd $FOAM_RUN/aachenBomb/0
rm ft fu
```

Turn chemistry on in the `/constant/chemistryProperties` file

```
chemistry               on;
```

and ignition on in the `/constant/combustionProperties` file. This step is not necessary, the mixture will still ignite when the species are properly mixed due to the high temperature.

```
ignite                  on;
```

Mesh the geometry using `blockMesh`, and start the `dieselFoam` solver.

```
cd $FOAM_RUN/aachenBomb
blockMesh
dieselFoam
```

The solution is only 0.01 seconds long, however, due to the fast chemistry a minimum of 4000 time steps are needed to resolve it. Furthermore, there is a total of 5000 parcels ($parcel_{mass} = N * Droplet_{mass}$ where N is the statistical number of drops in the parcel) that enter the domain and the source term from these all have to be calculated.

1.4 Post-processing in ParaView

Since paraFoam can not handle Lagrangian particles use foamToVTK and then ParaView.

```
cd $FOAM_RUN/aachenBomb
foamToVTK
paraview
```

In the /VTK directory open the case file (`aachenBomb\1.vtk`) and also, open the particles in the `/Lagrangian/defaultCloud_2.vtk` file. Create glyphs for the particles, see figure 1.2 for settings.

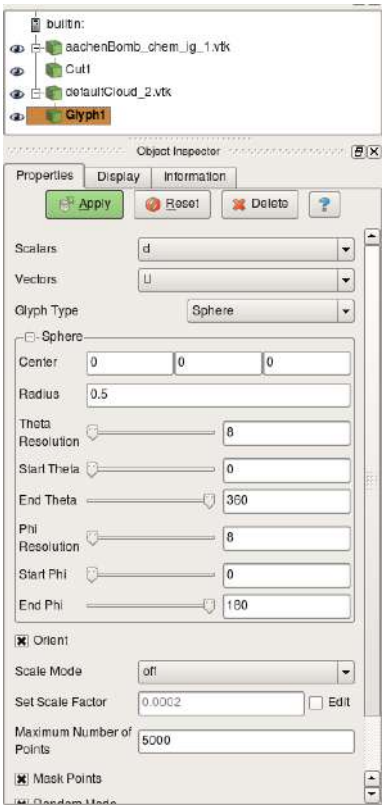


Figure 1.2: Settings for glyphs in ParaView to visualize the Lagrangian particles

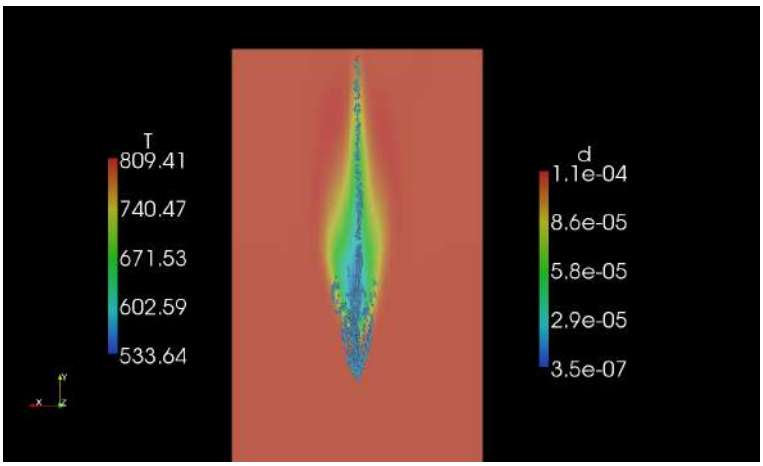


Figure 1.3: Droplets entering the domain, droplets colored by diameter and cut plane by temperature

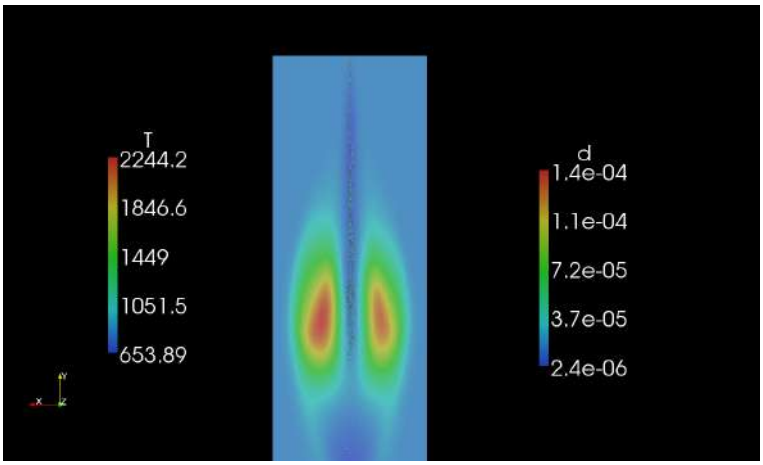


Figure 1.4: Gas phase ignition, droplets colored by diameter and cut plane by temperature

1.5 Adding a second liquid

To add a second liquid to the droplets follow these step-by-step instructions.

Remove `ft` and `fu` in the `aachenBomb/0` directory since these are not needed for this setup.

```
cd $FOAM_RUN/aachenBomb/0
rm ft fu
```

Turn chemistry on in the `/constant/chemistryProperties` file

```
chemistry               on;
```

and ignition on in the `/constant/combustionProperties` file.

```
ignite               on;
```

In `aachenBomb/constant/thermophysicalProperties` add an extra liquid material, here C_6H_{14} is used as an example.

```
liquidComponents
(
    C7H16
    C6H14
);

liquidProperties
{
    C7H16  C7H16 defaultCoeffs;
    C6H14  C6H14 defaultCoeffs;
}
```

In `aachenBomb/chemkin/chem.inp` add the C_6H_{14} in species.

```
ELEMENTS
 H   O    C   N   AR
END
SPECIE
C6H14 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
```

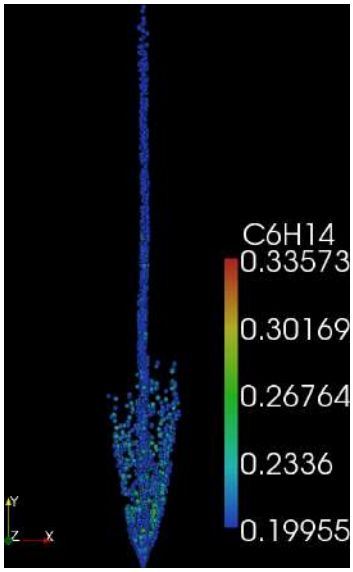
In `aachenBomb/constant/injectorProperties` change the mass fractions for C_7H_{16} and C_6H_{14}.

```
 X
            (
                0.8
                0.2
            );
```

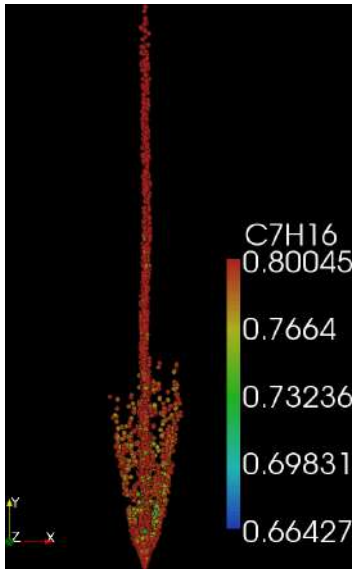
The droplets will now consist of 80 weight percent C_7H_{16} and 20 percent $C6H_{14}$. Mesh the geometry using `blockMesh`, and start the `dieselFoam` solver.

```
cd $FOAM_RUN/aachenBomb
blockMesh
dieselFoam
```

Post in ParaView as described in section 1.4. Notice the difference in evaporation pressure between the two species C_6H_{14} and C_7H_{16}.



(a) C_6H_{14}



(b) C_7H_{16}

Figure 1.5: Droplets colored by mass fractions

1.6 Your own evaporation model

To copy and create your own evaporation model follow these step-by-step instructions.

1.6.1 Copy the dieselFoam solver

Copy the dieselFoam solver to your working directory.

```
mkdir -p $WM_PROJECT_USER_DIR/applications/solvers/combustion
cp -r $FOAM_SOLVERS/combustion/dieselFoam $WM_PROJECT_USER_DIR\
/applications/solvers/combustion/mydieselFoam
```

Rename solver to mydieselFoam.

```
cd $WM_PROJECT_USER_DIR/applications/solvers/combustion/mydieselFoam/Make
```

Change in the `/Make/files` so it reads.

```
dieselFoam.C

EXE = $(FOAM_USER_APPBIN)/mydieselFoam
```

Change in the `/Make/options` so the second line reads.

```
-I$(LIB_SRC)/../applications/solvers/combustion/dieselEngineFoam \
```

1.6.2 Copy the dieselSpray library

Copy the `src/lagrangian/dieselSpray` dictionary to your user dictionary and rename it to `mydieselSpray`.

```
cd $WM_PROJECT_DIR
cp -riuv --parents --backup src/lagrangian/dieselSpray \
$WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/src/lagrangian
mv dieselSpray mydieselSpray
```

Copy the `standardEvaporationModel` dictionary to `my_standardEvaporationModel`.

```
cd $WM_PROJECT_USER_DIR/src/lagrangian/mydieselSpray/spraySubModels/evaporationModel
cp -r standardEvaporationModel my_standardEvaporationModel
```

Change `standardEvaporationModel` to `my_standardEvaporationModel` in the `.C` and `.H` file using `sed` and rename the files to `my_standardEvaporationModel`.

```
cd $WM_PROJECT_USER_DIR/src/lagrangian/mydieselSpray/spraySubModels/evaporationModel
cd  my_standardEvaporationModel
sed s/standardEvaporationModel/my_standardEvaporationModel/g \
standardEvaporationModel.C >my_standardEvaporationModel.C
sed s/standardEvaporationModel/my_standardEvaporationModel/g \
standardEvaporationModel.H >my_standardEvaporationModel.H
rm standardEvaporationModel.C standardEvaporationModel.H
```

Add `my_standardEvaporationModel.C` to the list of evaporation models in `/mydieselSpray/Make/files` line 59.

```
.
.
$(evaporationModels)/saturateEvaporationModel/saturateEvaporationModel.C
$(evaporationModels)/my_standardEvaporationModel/my_standardEvaporationModel.C
```

Also, change the name of the library at the bottom of the `/mydieselSpray/Make/files` file to

```
LIB = $(FOAM_USER_LIBBIN)/libmydieselSpray
```

1.6.3 Adding mydieselSpray library to mydieselFoam solver

Go to your `mydieselFoam` directory

```
cd $WM_PROJECT_USER_DIR/applications/solvers/combustion/mydieselFoam/Make
```

Open the `options` file and change line 6 from

```
-I$(LIB_SRC)/lagrangian/dieselSpray/lnInclude \
```

to:

```
-I$(WM_PROJECT_USER_DIR)/src/lagrangian/mydieselSpray/lnInclude \
```

At the end of the `options` file change so it reads:

```
-lpdf \
-L$(WM_PROJECT_USER_DIR)/lib/$(WM_OPTIONS) \
-lmydieselSpray
```

1.6.4 Update case files

Update the `sprayProperties` so it includes the coefficients for
your evaporation model in `aachenBomb/constant/sprayProperties`.

```
cd $FOAM_RUN/aachenBomb/constant
```

Edit `sprayProperties` so the evaporation model is set to:

```
evaporationModel my_standardEvaporationModel;
```

Also, add model coefficients for your evaporation model in then `sprayProperties` file

```
my_standardEvaporationModelCoeffs
{
    evaporationScheme explicit;
    preReScFactor   0.6;
    ReExponent      0.5;
    ScExponent      0.333333;
}
```

1.6.5 Customizing the evaporation model

Go back to the `my_standardEvaporationModel` directory.

```
cd $WM_PROJECT_USER_DIR/src/lagrangian/mydieselSpray/spraySubModels/\
evaporationModel/my_standardEvaporationModel
```

Take a closer look in `my_standardEvaporationModel.C`. On line 98 to 104 it reads,

```
scalar my_standardEvaporationModel::Sh
(
    const scalar ReynoldsNumber,
    const scalar SchmidtNumber
) const
{
    return 2.0 + preReScFactor_*pow(ReynoldsNumber,ReExponent_)\
    *pow(SchmidtNumber,ScExponent_);
}
```

That is, the Sherwood number is calculated according to the *Ranz-Marshall* correlation [4], see equation 1.14, observe that the `my_standardEvaporationModelCoeffs` are used here.

$$Sh = 2 + 0.6 Re_r^{0.5} Sc^{0.33333} \tag{1.14}$$

Where Sh is the Sherwood number, Re_r relative Reynold number and Sc Schmidt number.
In this tutorial we will make changes to the evaporation time, located in `my_standardEvaporationModel.C`, on line 142 to 163.

```
scalar Xratio = (Xs - Xf)/max(SMALL, 1.0 - Xs);

if (Xratio > 0.0)
{
    lgExpr = log(1.0 + Xratio);
}

scalar denominator =
6.0 * massDiffusionCoefficient
* Sh(ReynoldsNumber, SchmidtNumber)
* rhoFuelVapor * lgExpr;

if (denominator > SMALL)
{
time = max(VSMALL, liquidDensity * pow(diameter, 2.0)/denominator);
}

return time;
```

[4] Multiphase flows with droplets and particles, Crowe et al. (1998) CRC Press LLC

We want the evaporation time τ_m to be calculated by a D^2-law showed in equations 1.15 and 1.16

$$\lambda = \frac{4Sh\rho_c D_v}{\rho_d}(\omega_{A,s} - \omega_{A,\infty}) \tag{1.15}$$

Where λ is the evaporation constant, ρ_c film density, ρ_d droplet density, D diameter, D_v diffusion coefficient for species A, $\omega_{A,s}$ mass fraction of species A at the droplet surface and $\omega_{A,\infty}$ for the free stream.

$$\tau_m = \frac{D^2}{\lambda} \tag{1.16}$$

Make changes in the `my_standardEvaporationModel.C` file, according to the equations showed. Start by removing `scalar lgExpr = 0.0;` on line 123.
Edit the `my_standardEvaporationModel.C` file and change `scalar Xratio` and `scalar denominator`. Remove the `if (Xratio > 0.0)` statement completely.

```
scalar Xratio = (Xs - Xf);

scalar denominator =
4.0 * massDiffusionCoefficient
* Sh(ReynoldsNumber, SchmidtNumber)
* rhoFuelVapor*Xratio;

if (denominator > SMALL)
{
   time = max(VSMALL, liquidDensity * pow(diameter, 2.0)/denominator);
}

return time;
```

1.6.6 Compile library and solver

Compile the `mydieselSpray` library and your solver.

```
cd $WM_PROJECT_USER_DIR/src/lagrangian/mydieselSpray
wclean
rm -r lnInclude
rm -r Make/linux*
wmake libso

cd $WM_PROJECT_USER_DIR/applications/solvers/combustion/mydieselFoam
wclean
rm -r Make/linux*
wmake
```

1.6.7 Running the case

Start the solver with `mydieselFoam` and post in ParaView as described in section 1.4. When the solver starts check that your evaporation model is being used.

```
cd $FOAM_RUN/aachenBomb
blockMesh
mydieselFoam
```