Installation

1. Compile Cantera-2.4.0

2. Replace adjustPhi

Replace `adjustPhi.C` and `adjustPhi.H` in the folder

`$FOAM_SRC/finiteVolume/cfdTools/general/adjustPhi`

To compile, go all the way up to `$FOAM_SRC`, type `wmake libso finiteVolume` or `./Allwmake`. Replacing these files will not affect on the other applications.

3. Install dynCompSansThermoSF

Copy *dynCompSansThermoSF* folder below *$FOAM_SRC/TurbulenceModels*, type *dynCompSansThermoSF/Allwmake*.
It is recommended to add a line *dynCompSansThermoSF/Allwmake $** in the *turbulenceModels/Allwmake* file.

4. Install Lapack & Blas

Download and unpack the `lapack-3.1.0.tgz` file (or something similar). In the directory, you should see a `Makefile` and a `make.inc` file, or at least a `make.inc.example` file. If there is no `make.inc` file, copy the `make.inc.example` file to `make.inc`.

Open the `make.inc` file and change the Fortran compiler environment variables to match the compiler on your local machine. Lapack and Blas are compiled using only Fortran, so you will not need to touch any C/C++ compilers. For the Intel Fortran compiler, we recommand:

```
FORTRAN  = ifort
OPTS     = -O3 -xN -ip
DRVOPTS  = \$(OPTS)
NOOPT    =
LOADER   = ifort
LOADOPTS = -O3 -xN -ip
```

Usually **NGA** wants these libraries in the form `libblas.a` and `liblapack.a`. Change the names of the libraries in `make.inc`:

```
BLASLIB      = ../../libblas.a
LAPACKLIB    = liblapack.a
```

Now you are ready to compile. The compilation is done in two steps:

- compiling Blas: `make blaslib`
- compiling Lapack: `make lapacklib`

Create the directory where you want to install the Lapack & Blas libraries. We recommand installing the libraries in `/opt/lapack`. Copy the two libraries (`libblas.a` and `liblapack.a`) to the install directory.

5. Compile the soot model

- Start a new terminal, go the the folder *your_solver_dir/sootCode*,
- load the gnu environment *module swap Prng-cray Prng-gnu*

- clean the lib: *make clean*
- compile soot model: *make*

6. Compile the solver
   - Start a new terminal, go the the folder *your_solver_dir*
   - Modify the corresponding path: *vim Make/options*
   - Copy *./lib/** to *your_OpenFOAM-7_dir/platforms/linux64GccDPInt32Opt/lib/*
   - Load your OpenFOAM environment: *OF7* (it depends on your definition in *bashrc* file)
   - Clean lib: *wclean*
   - Compile solver: *wmake*

How to use the solver

To start a case from scratch from time 0, here are a few useful tips:

1) The folder example_case/0 contains all necessary fields that umDetailedChemFoam needs to read in to initiate the test case simulation. Notice that: for a turbulent case, a few more fields such as muSgs, k will be needed; Zmix is needed here just as a dummy field to construct the object of turbulence model that's required during the solver initialization; the mass fractions of few key species needs to be read-in directly from the initial time folder, whereas the rest of the species will be automatically created by the solver. For more details, please read umDetailedChemFOAM_tut_pkg/src/solver/createDetailedChemistry.H
2) Only the boundary values of T, U, Mom_*, p, and species (Y_*) are important. For other fields, once the boundary type is correct, the boundary values will be updated by the solver.
3) The file example_case/constant/inputParameters contains some important simulation options. Please read the comments in the file to see the meaning of each option.
4) First, develop a cold flow, use the following simulation options as:
   vim system/controlDict
   startTime 0;
   solveYEqn true;
   solvehEqn true;
   CsrcOnOff 0.0;
   A large time step (10^{-4}) can be used for cold flow simulation by controlling the CFL number (maxCo in system/controlDict).
5) After cold flow mixing has been developed. Initiate reaction by chemical equilibrium, use the following options:
   vim constant/inputParameters
   equilibrate true;
   vim system/controlDict
   startTime your_cold_flow_time;
   solveYEqn false;
   solvehEqn false;
   CsrcOnOff 0.0;
   Then resubmit the case to start the simulation but with species and enthalpy transport equation disabled. This step is to allow the low'Mach algorithm to correct the velocity field after the drastic change of density introduced by the patch of temperature field, which is critical for convergence. After the velocity field has been corrected, as the magnitude of max(mag(U)) and max(p)-min(p) drops to

a magnitude that is comparable to the former cold flow simulation. After completing the chemical equilibrium calculation, it is best to save the data.

6) Then decreasing the timestep to a small value (10^{-7}), and re-enable solving the species transport equation and enthalpy equation. Notice that these options can be changed on-the-fly during a simulation, but set reaction source term to exact 0.

```
vim system/controlDict
solveYEqn       true;
solvehEqn       true;
CsrcOnOff       0.0;
```

It is suggested to first enable species transport equation, and freeze the enthalpy equation for a few more time steps.

7) Once the species and enthalpy transport equations are enabled, the max(mag(U)) may increases several times and then decreases to a normal value. Once the max(mag(U)) is normal, you can enable the chemistry ODE integration, gradually increase the reaction source coefficient from 0 to 1.0 at a step size of 0.01~0.1. Last, note that CsrcOnOff has to reach 1.0 eventually in order to have the CFD solution being physical.

```
vim system/controlDict
CsrcOnOff       0.1;
```

8) Sometimes, if the case appears to be very unstable, the user may want to use a more dissipative divergence schemes, as:

```
vim system/fvSchemes
div(((phi+phi_0)|4),U)      Gauss upwind;
div(((phi+phi_0)|4),U_0)    Gauss upwind;
div((phi|2),Yi)             Gauss upwind;
div((phi|2),h)              Gauss upwind;
div((phi|2),h_0)            Gauss upwind;
```