User Guide for OpenFOAM-12

DTLreactingFoam: An efficient CFD tool for laminar reacting flow simulations using detailed chemistry and transport with time-correlated thermophysical properties

Please cite our article if you are using this package: D. N. Nguyen, J. H. Lee, C. S. Yoo, *DTLreactingFoam*: An efficient CFD tool for laminar reacting flow simulations using detailed chemistry and transport with time-correlated thermophysical properties, Computer Physics Communications (2025)(submitted).

Contents

1 General descriptions **3**

1.1 thermophysicalModels library 3

1.2 DTLreactingFoam modular solver 5

1.3 preprocessing utilities 5

1.3.1 DTMchemkinToFoam 5

1.3.2 FTMchemkinToFoam 5

2 Running simulations with DTLreactingFoam **7**

2.1 Simulations for general reacting flows using DTM 7

2.1.1 Prepare input files 7

2.1.2 Setup and run simulation 10

2.2 Simulations for general reacting flows using FTM 14

2.2.1 Prepare input files 14

2.2.2 Setup and run simulation 18

1. General descriptions

The framework is fully native OpenFOAM(OF) and comprises newly developed libraries, modular solvers, and utilities as follows:

- libraries:
  - **thermophysicalModels**
- Modular solvers:
  - **DTLreactingFoam**
- utilities:
  - **DTMchemkinToFoam**
  - **FTMchemkinToFoam**

1.1. thermophysicalModels library

The class diagram of a target library plays a crucial role for guiding code structure and modularity in OF code development (i.e., OOP). It outlines essential features such as inheritance hierarchies and class interfaces. Figure 1 depicts the class diagram of the `thermophysicalModels` library in the OF-12, highlighting key classes associated with reacting flow modeling. The numbered boxes in the diagram indicate the sample classes that can be one of the classes within the box during runtime. All classes can be organized in four conceptual *blocks*; the *ThermoType block* and *MixtureType block* representing various thermophysical models and mixture models; the *ChemistryType Block*, encompassing chemistry models; and the *BasicType Block*, which includes base classes for system types (e.g., *rho*-based or *psi*-based). While there are structural differences between the `thermophysicalModels` libraries in OF-12 and OF-6 (as presented in [1]), the core implementation philosophy of thermophysical property calculation remains consistent.

The updated `thermophysicalModels` library and its interfacing with relevant solvers and utilities are illustrated in Figure 2. The class diagram highlights how the modular

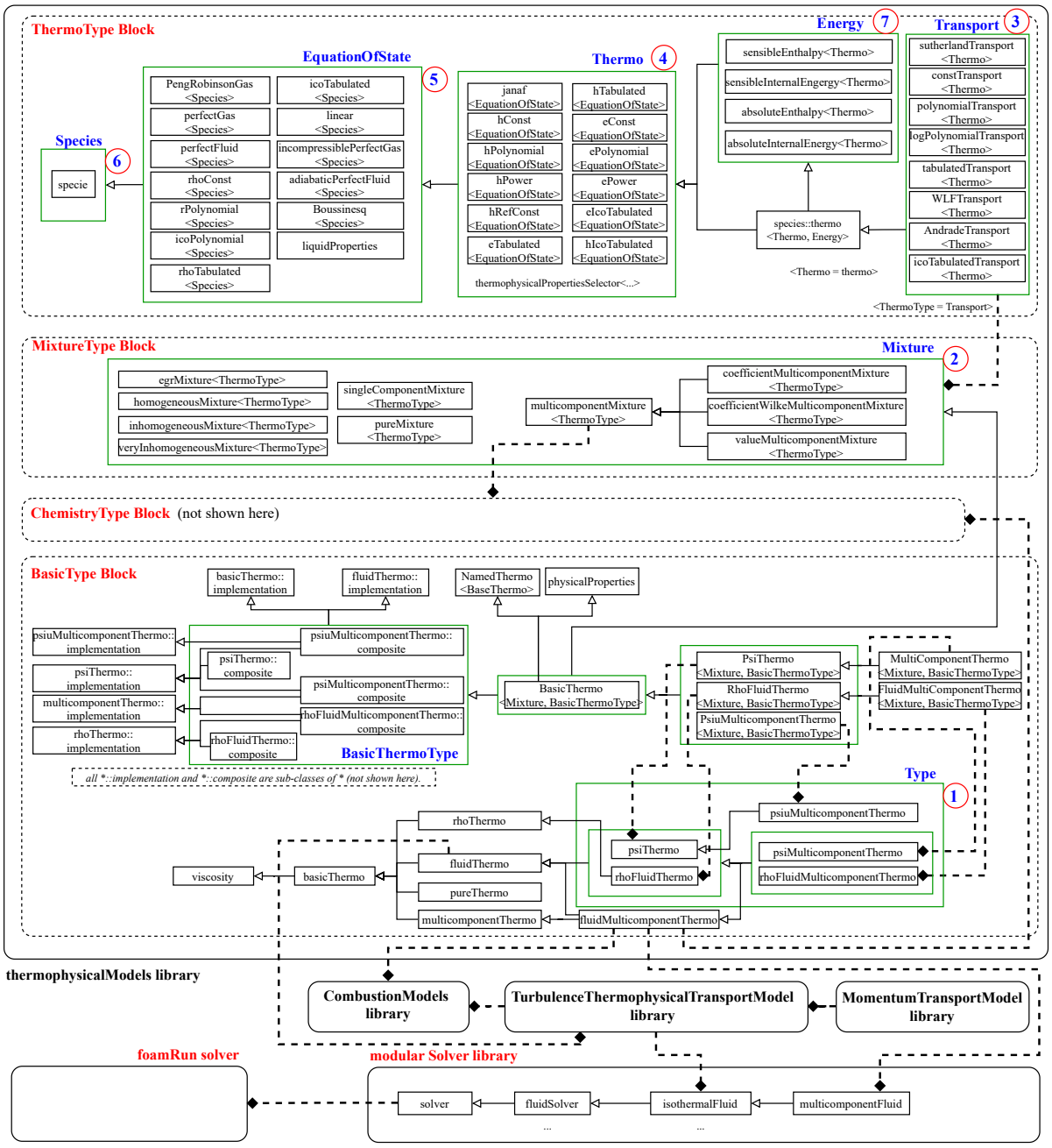

Figure 1: The class diagram of the `thermophysicalModels` library in the OF-12. The boxes with marked numbers denote sample classes. The arrow-line denotes the inheritance relationship in which the direction of arrow is from a subclass to its base class. A dashed line denotes a class-class or class-solver interface in which one class is used in another or in a solver.

solver interacts with the thermophysical models. In the diagram, green boxes marked with numbers denote sample classes that typify groups of models, following the approach presented by [1]. Yellow boxes represent newly developed classes (i.e., DTM, FTM) added to the original `thermophysicalModels` library. The DTM and FTM are implemented following the methodology proposed by Nguyen et al. [1] to address limitations in OF's capability to support complex mixing rules in custom thermophysical models. The coTHERM method is subsequently integrated into these models via the `PsiThermo` class.

1.2. DTLreactingFoam modular solver

In OF-10 and earlier versions, solvers were compiled as stand-alone applications. Starting from OF-11, this structure has been replaced by the `foamRun` driver solver, which employs a run-time selection mechanism to dynamically load the specified modular solver (e.g., `multicomponentFluid` which is same as `reactingFoam` solver).

In this work, a new modular solver named `DTLreactingFoam` is developed based on the original `multicomponentFluid` modular solver, with governing equations presented in the main text of the paper. It supports both transient and steady-state laminar reacting flow simulations incorporating DTM/FTM with the coTHERM method to reduce computational time while preserving accuracy.

1.3. preprocessing utilities

1.3.1. DTMchemkinToFoam

Based on the original OF `chemkinToFoam` utility, a new `DTMchemkinToFoam` pre-processing utility is created to accommodate the newly developed DTM. It automates the conversion of input files from CHEMKIN to OF format.

1.3.2. FTMchemkinToFoam

Similarly, dedicated `FTMchemkinToFoam` pre-processing utility is created to convert input files from CHEMKIN to OF format for the newly developed FTM.

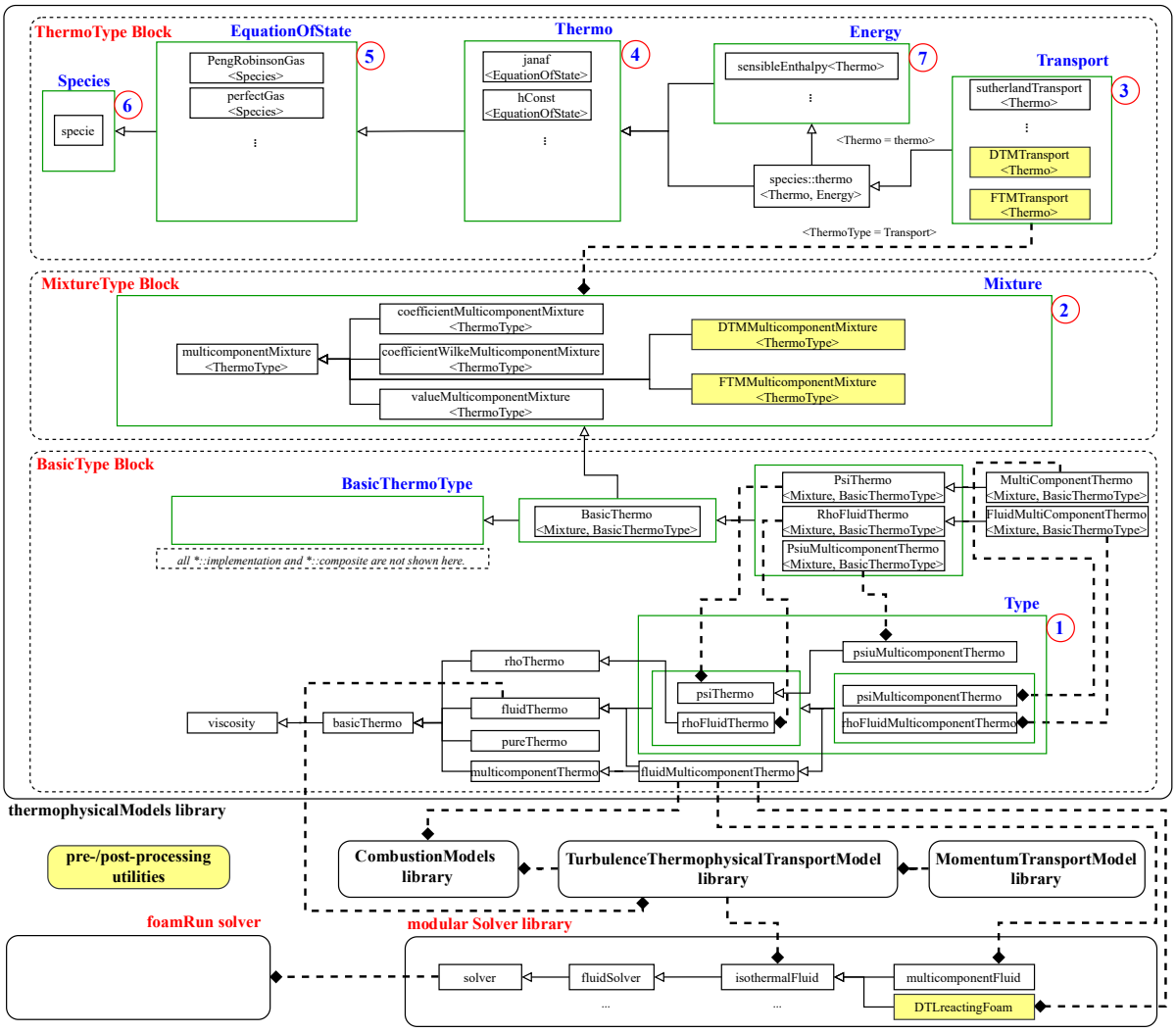

Figure 2: The class diagram of the updated `thermophysicalModels` library in the `DTLreactingFoam`-12 framework. The boxes with marked numbers denote sample classes. The arrow-line denotes the inheritance relationship in which the direction of arrow is from a subclass to its base class. A dashed line denotes a class-class or class-solver interface in which one class is used in another or in a solver.

2. Running simulations with DTLreactingFoam

2.1. Simulations for general reacting flows using DTM

2.1.1. Prepare input files

Before running simulation using the DTM, an input file, say `thermo.DTM`, is needed. It involves essential parameters for transport property calculations and must be in the following format:

```
O2
{
    ...
    transport
    {
        // input parameters for DTM
        linearity       1;
        epsilonOverKb   107.4;
        sigma           3.458;
        dipoleMoment    0;
        alpha           1.6;
        Zrot            3.8;
    }
}
```

The `thermo.DTM` input file can be generated using `DTMchemkinToFoam` utility with the following command:

```
DTMchemkinToFoam <input1> <input2> <input3> <input4> <output1> <output2>
```

where:

- `input1`: chemical mechanism file in CHEMKIN format, e.g., `chem.inp`
- `input2`: data file for thermodynamics in CHEMKIN format, e.g., `therm.dat`
- `input3`: data file for DTM transport in CHEMKIN format, e.g., `trans.dat`
- `input4`: data file for Sutherland transport, e.g., `transportSutherland`

- `output1`: chemical mechanism file in OF format, e.g., `reactions`
- `output2`: data file for thermodynamic and transport properties using DTM, e.g., `thermo.DTM`

The three first input files in CHEMKIN format can be taken directly from the desired chemical mechanism available in the literature. The `input4` file must be in OF format as following:

```
".*"
{
    transport
    {
        As 1.512e-06;
        Ts 120.;
    }
}
```

At `DTLreactingFoam-12/tutorials/Mech/DTMchemkinToFoam/example/` directory, an example is provided to demonstrate the use of the `DTMchemkinToFoam` utility. It is important to note that using `DTMchemkinToFoam` requires only four input files placed in the case directory. In other words, `0`, `constant`, and `system` directories are not needed in this step. To generate the `thermo.DTM` file, first go to the case directory by:

```
cd ~/OpenFOAM/yourDirectory/DTLreactingFoam-12/tutorials/Mech/
   DTMchemkinToFoam/example/
```

Then execute the following command in the terminal:

```
// Orders of files are important and need to be specified correctly
DTMchemkinToFoam chem.inp therm.dat trans.dat transportSutherland
   reactions thermo.DTM
```

After executing that command, two output files named `reactions` and `thermo.DTM` are generated as:

```
// This is inside the 'reactions' file:
reactions
```

```
{
    un-named-reaction-0
    {
        type            reversibleArrhenius;
        reaction        "CH4 + 2O2 = CO2 + 2H2O";
        A               1.2e+11;
        beta            -1;
        Ta              0;
    }
}
Tlow            200;
Thigh           3500;
```

```
// This is inside the 'thermo.DTM' file:
species         5 ( CH4 H2O O2 CO2 N2 );

N2
{
    specie
    {
        molWeight       28.0134;
    }
    thermodynamics
    {
        Tlow            200;
        Thigh           5000;
        Tcommon         1000;
        highCpCoeffs    ( 2.92664 0.0014879768 -5.68476e-07 1.0097038e-10
            -6.753351e-15 -922.7977 5.980528 );
        lowCpCoeffs     ( 3.298677 0.0014082404 -3.963222e-06 5.641515e-09
             -2.444854e-12 -1020.8999 3.950372 );
    }
    transport
    {
        As              1.512e-06;
```

```
        Ts              120;
        linearity       1;
        epsilonOverKb   97.53;
        sigma           3.621;
        dipoleMoment    0;
        alpha           1.76;
        Zrot            4;
    }
    elements
    {
        N               2;
    }
}

CO2
{
    ...
}
...
```

These two files now are ready for use.

Examples with script `Allrun` to automate converting files for several detailed chemical mechanisms are also available in `DTLreactingFoam-12/tutorials/Mech/DTMchemkinToFoam/` directory.

2.1.2. Setup and run simulation

A test case of 2-D counterflow diffusion flame of CH_4/air, which is available in `tutorials` directory of the original OF, is selected to demonstrate the setting and running simulation using the `DTLreactingFoam` modular solver with the DTM and coTHERM method in this manual (see the `DTLreactingFoam-12/tutorials/counterFlowFlame2D_example/` directory for reference). To create this test case, first, go to the tutorials in your directory. Then copy original `counterFlowFlame2D` case from OpenFOAM into your directory as:

```
cd ~/OpenFOAM/yourDirectory/DTLreactingFoam-12/tutorials/
```

```
cp -rf ~/OpenFOAM/OpenFOAM-12/tutorials/multicomponentFluid/
   counterFlowFlame2D/ .
```

Then go to the `counterFlowFlame2D` directory:

```
cd ~/OpenFOAM/yourDirectory/DTLreactingFoam-12/tutorials/
   counterFlowFlame2D/
```

Run *Allclean* script first to clean the old file as:

```
./Allclean
```

Then, copy two files (e.g., `reactions` and `thermo.DTM`) which are generated using the `DTMchemkinToFoam` utility, as described in Section 2.1.1, into the `constant` directory. In this directory, the `physicalProperties` dictionary file must be modified to be compatible with the `DTLreactingFoam` modular solver. Particularly, in the `physicalProperties` dictionary file, add new essential keywords and modify the `thermoType` entry as follows:

```
usingDetailedTransportModel  true;  //newly added
usingCoTHERM                 true;  //newly added

majorSpeciesForCoTHERM (CH4 O2 CO2 H2O);  //newly added
epsilonT          0.2;   // [K], newly added
epsilonP          100;   // [Pa], newly added
epsilonS          0.001; // [-], in mass fraction, newly added

thermoType
{
    type            hePsiThermo;
    mixture         DTMMulticomponentMixture; //for DTM
    transport       DTMTransport;             //for DTM
    thermo          janaf;
    energy          sensibleEnthalpy;
    equationOfState perfectGas;
    specie          specie;
}
defaultSpecie N2; //this can be changed
```

```
//#include "thermo.compressibleGas"
#include "thermo.DTM"  //for DTM
```

where meaning and options of newly added keywords are explained as below:

- `usingDetailedTransportModel`: It must be explicitly set to be `true` when using DTM. Otherwise the thermophysical properties will be not updated in each time step during simulation. The default value of this keyword is `false`, appropriate for using original STM since our code does not eliminate any functionality of original OF.

- `usingCoTHERM`: It must be explicitly set to be `true` if you want to apply the coTHERM method with DTM for your simulation to reduce the computational time. The default value of this keyword is `false`, appropriate for using the DTM alone.

- `majorSpeciesForCoTHERM`: This is a list of selected species in the chemical mechanism considered as representative major species for the mixture when using the coTHERM method. Note that species selection is case dependent. Typically, it includes dominant species in both unburnt and burnt regions, for example of CH_4/air flame: CH_4, O_2, N_2, CO_2, H_2O, OH, HO_2, CH_2O [2].

- `epsilonT`: This is threshold value (in [K]) to control residual of temperature, corresponding to ε_T in the coTHERM method. For a general reacting flow, we recommend this value is set to be 0.2 K for computational efficiency without sacrificing accuracy. If this value is not specified, the program will automatically use 0.2 K as a default value for ε_T.

- `epsilonP`: This is threshold value (in [Pa]) to control residual of pressure, corresponding to ε_P in the coTHERM method. For a general reacting flow, we recommend this value is set to be 100 Pa for computational efficiency without sacrificing accuracy. If this value is not specified, the program will automatically use 100 Pa as a default value for ε_P.

- `epsilonS`: This is threshold value (dimensionless) to control residual of species mass fraction, corresponding to ε_S in the coTHERM method. For a general reacting flow, we recommend this value is set to be 0.001 for computational efficiency without sacrificing accuracy. If this value is not specified, the program will automatically use 0.001 as a default value for ε_S.

In `system` directory, the `controlDict` dictionary must be modified to ensure that the `foamRun` solver will load the `DTLreactingFoam` modular solver instead of `multicomponentFluid` as follows:

```
...
//solver            multicomponentFluid; //original OF
solver           DTLreactingFoam;  //new modular solver
...
```

It is also important to note that the `default` keyword of the `divSchemes` entry in the `fvSchemes` dictionary file, placed in the `system` directory, needs to be changed from `none` to a specific type, for example:

```
...
divSchemes
{
    default    none; // original OF
    default    Gauss linear; // change from 'none' to 'Gauss linear'
    ...
}
...
```

All other settings are identical to those used in the `multicomponentFluid` modular solver.

Everything now is ready for use. For running simulation, first run `blockMesh` to generate geometry, then type the `foamRun` solver in the terminal (in the same way with `multicomponentFluid`) as:

```
blockMesh
foamRun
```

2.2. Simulations for general reacting flows using FTM

Simulations for general reacting flows using FTM are the same as using DTM except for input files and keyword setup for models combined with the FTM.

2.2.1. Prepare input files

Before running simulation using the FTM, an input file, say `thermo.FTM`, is needed. It involves essential parameters for transport property calculations and must be in the following format:

```
O2
{
    ...
    transport
    {
        // input parameters for FTM
        muCoeffs        (-19.3405 2.62765 -0.265133 0.0118167);
        kappaCoeffs     (-13.8944 3.07851 -0.297333 0.0127535);
        DijCoeffs       (
                            (-24.3747 3.3315 -0.221273 0.00973849)
                            (-30.453 5.60295 -0.492151 0.0205307)
                            (-9.21034 -6.45e-26 9.26e-27 -4.40e-28)
                            (-27.6164 4.47195 -0.36524 0.0158195)
                            (-25.3927 3.75885 -0.275697 0.0120551)
                            ...
                        );
    }
}
```

The `thermo.FTM` input file can be generated using the `FTMchemkinToFoam` utility.

Unlike the `DTMchemkinToFoam` utility, the `FTMchemkinToFoam` requires a case directory that already works with the `DTLreactingFoam` modular solver using the DTM. In other words, the case directory involves `0`, `constant`, and `system` directories, as the same as the standard OF case. Especially, in the `constant` directory, the `thermo.DTM` file must be provided as an input for fitting procedure.

The `counterFlowFlame2D` case in the Section 2.1.2 can be used to demonstrate the procedure of generating the `thermo.FTM` file. To do so, first go to that case as:

```
cd ~/OpenFOAM/yourDirectory/DTLreactingFoam-12/tutorials/
   counterFlowFlame2D/
```

Then open the `constant/physicalProperties` file and modify as following:

```
// All setting of coTHERM are same as applied for DTM
usingDetailedTransportModel  true;   //newly added
usingCoTHERM                 false;   //newly added
usingPreProcessingFTM        true;   //must be true, default is false
...
thermoType
{
    ...
    mixture          DTMMulticomponentMixture;  //for DTM
    //transport        DTMTransport;            //for DTM
    transport        preprocessingFTMTransport; //for preprocessing only
    ...
}
defaultSpecie N2;
#include "thermo.DTM"  //The thermo file for DTM must be included
```

where `usingPreProcessingFTM` keyword must be added, and must be explicitly set to be `true`. The default value of this keyword is `false`.

Additionally, the `FittingDict` dictionary file is needed for fitting procedure. It is placed in the `constant` directory and has the following format:

```
/*--------------------------------*- C++ -*----------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  12
     \\/     M anipulation  |
\*---------------------------------------------------------------*/
FoamFile
```

```
{
    format      ascii;
    class       dictionary;
    location    "constant";
    object      FittingDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
numberOfPoint      101;  // [points] don't change this value
maxT               3000; // [K] don't change this value
minT               300;  // [K] this may be changed
pressure           101325; // [Pa] this is case dependent

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

where:

- `numberOfPoint`: The number of temperature sampling points used in the polynomial fitting process. The number of 101 points is found to be optimal value for this process. Higher values does not improve fitting accuracy but increase unnecessarily computational cost.
- `maxT`: The upper bound of the fitting temperature range $[T_{\min}, T_{\max}]$ over which transport properties are sampled and fitted. This value should be chosen to cover the expected flame temperature range.
- `minT`: The lower bound of the fitting temperature range $[T_{\min}, T_{\max}]$ for sampling. This value should capture the range of cold reactants or ambient conditions.
- `pressure`: The reference pressure at which all transport properties are evaluated and fitted. This is actually the operating pressure condition of your problem.

Now everything is ready. Then execute the following command in the terminal to run the `FTMchemkinToFoam`:

```
cd ~/OpenFOAM/yourDirectory/DTLreactingFoam-12/tutorials/
   counterFlowFlame2D/
```

```
FTMchemkinToFoam
```

After execution, the `thermo.FTM` file is generated, and placed in the `constant` directory. It is ready for use in simulations with the FTM.

It is important to note that, after running the `FTMchemkinToFoam`, a new file named `TransportCoeffDict` is generated in the `constant` directory. It has the following format:

```
// in side 'TransportCoeffDict' file:
species
(
    CH4
    H2O
    O2
    CO2
    N2
);

CH4
{
    fittingTranCoeff
    {
        muCoeffs    (-22.2457 3.54364 -0.383872 0.0169568);
        kappaCoeffs    (0.812318 -4.54742 0.975145 -0.0536628);
        DijCoeffs    (
                        (-9.21034 -6.44736e-26 9.25847e-27 -4.40439e-28)
                        (-30.6998 5.62052 -0.483426 0.0197159)
                        (-25.3927 3.75885 -0.275697 0.0120551)
                        (-28.4471 4.81747 -0.406845 0.0174929)
                        (-25.0853 3.63653 -0.260141 0.0113939)
                    );
    }
};

H2O
{
```

```
    fittingTranCoeff
    ...
}
...
```

This file is generated for only purpose of checking the fitting transport coefficients of the FTM in case it needs double check. Otherwise, this file can be ignored.

2.2.2. Setup and run simulation

To run simulation with the FTM, new essential keywords for coTHERM are required in the `physicalProperties` dictionary file, as explained in Section 2.1.2. The dictionary file need to be modified to be compatible with the FTM as follows:

```
// All setting of coTHERM are same as applied for DTM
usingDetailedTransportModel  true;  //newly added
usingCoTHERM                 true;  //newly added

majorSpeciesForCoTHERM (CH4 O2 CO2 H2O);  //newly added
epsilonT          0.2;   // [K], newly added
epsilonP          100;   // [Pa], newly added
epsilonS          0.001; // [-], in mass fraction, newly added

thermoType
{
    type            hePsiThermo;
    mixture         FTMMulticomponentMixture; //for FTM
    transport       FTMTransport;  //for FTM
    thermo          janaf;
    energy          sensibleEnthalpy;
    equationOfState perfectGas;
    specie          specie;
}
defaultSpecie N2;
#include "thermo.FTM" //for FTM
```

All other settings are identical to those used in the `DTLreactingFoam` modular solver with DTM model.

Everything now is ready for use. For running simulation, first run `blockMesh` to generate geometry, then type the `foamRun` solver in the terminal (in the same way with `DTLreactingFoam` for DTM) as:

```
blockMesh
foamRun
```

References

[1] D. N. Nguyen, K. S. Jung, J. W. Shim, C. S. Yoo, Real-fluid thermophysicalModels library: An OpenFOAM-based library for reacting flow simulations at high pressure, Comput. Phys. Commun. 273 (2022) 108264.

[2] S. Yang, R. Ranjan, V. Yang, S. Menon, W. Sun, Parallel on-the-fly adaptive kinetics in direct numerical simulation of turbulent premixed flame, Proc. Combust. Inst. 36 (2017) 2025–2032.