Cite as: Xu, S.J.: The implementation of a stochastic reactor (StoR) combustion model. In Proceedings of CFD with OpenSource Software, 2018, Edited by Nilsson H., http://dx.doi.org/10.17196/OS_CFD#YEAR_2018

CFD WITH OPENSOURCE SOFTWARE

A COURSE AT CHALMERS UNIVERSITY OF TECHNOLOGY
TAUGHT BY HÅKAN NILSSON

The implementation of a stochastic reactor (StoR) combustion model

Developed for OpenFOAM-v1806

Author:
Shijie XU
Lund University
shijie.xu@energy.lth.se
shijiexu2010@gmail.com

Peer reviewed by:
William HAY
Mohammad ARABNEJAD
Håkan NILSSON

Disclaimer: This is a student project work, done as part of a course where OpenFOAM and some other OpenSource software are introduced to the students. Any reader should be aware that it might not be free of errors. Still, it might be useful for someone who would like learn some details similar to the ones presented in the report and in the accompanying files. The material has gone through a review process. The role of the reviewer is to go through the tutorial and make sure that it works, that it is possible to follow, and to some extent correct the writing. The reviewer has no responsibility for the contents.

January 15, 2019

Learning outcomes

The reader will learn:

How to use it:

- How to use a Lagrangian-Euler solver 'sprayFoam' to simulate the spray and combustion process. How to set your own spray parameters, including injector configuration, chemical mechanism and combustion model.
- How to use our group new-build stochastic reactor (StoR) combustion model to simulate turbulence combustion case under liquid-gas phase.
- How to use the chemistry coordinate mapping approach (CCM)[1] developed by our group to speedup your chemical reaction calculation.

The theory of it:

- A general description of the stochastic reactor (StoR) combustion model based on Probability Distribution Function (PDF) method. How it can take turbulence combustion interaction into account and improve the accuracy.
- The theory of the chemistry coordinate mapping (CCM) chemistry speedup approach. How it can improve computing efficiency in finite-rate chemistry based simulation.
- The idea of the combination of those two approaches will be described in detail.

How it is implemented:

- A detailed description of the implementation of StoR combustion model.
- How to implement the case setup for 'sprayFoam' to use both StoR combustion model and CCM chemistry speedup approach.

How to modify it:

- The combustion model will be modified from an existing eddy dissipation concept (EDC) combustion model.

Prerequisites

The reader is expected to know the following in order to get maximum benefit out of this report:

- Basic theory of turbulence combustion;
- Run standard document tutorials like sprayFoam tutorial or reactingFoam
- It is strongly recommonded to gain a brief insight into the the chemistry coordinate mapping approach (CCM) from the following journals, if you think they are useful, please cite them:

  Jangi, Mehdi, and Xue-Song Bai. “Multidimensional chemistry coordinate mapping approach for combustion modelling with finite-rate chemistry." Combustion Theory and Modelling 16.6 (2012): 1109-1132.
  Jangi, Mehdi, Rixin Yu, and Xue-Song Bai. “A multi-zone chemistry mapping approach for direct numerical simulation of auto-ignition and flame propagation in a constant volume enclosure." Combustion Theory and Modelling 16.2 (2012): 221-249.

Contents

Chapter 1

Introduction

1.1 Background

Turbulence combustion involves the complex interactions between fluid dynamics, thermodynamics and chemical processes[2]. Fuel spray and combustion has been widely used in energy conversion apparatus, such as vehicle engine and gas turbine, which is a typical turbulence combustion including a wide range of temporal and spatial scales. Consequently, turbulence and combustion interaction (TCI) need to be considered.

In OpenFOAMv1806, a number of combustion models are implemented to take turbulence and combustion interaction (TCI) into account, such as partial stired reactor(PaSR), eddy dissipation concept (EDC) and so on. It is worth mentioning that well stired reactor(WSR) also exists, which is so-called laminar combustion model in OpenFOAM.

1.2 Motivation

Recently, our group proposed a stochastic reactor (StoR) to take turbulence and combustion interaction into account. By coupling with chemistry speedup method, it can improve the accuracy of chemical reaction source terms calculation without increasing the computational cost.

The purpose of this project is to implement a stochastic reactor (StoR) combustion model based on presumed Probability Distribution Function (PDF) method. The chemistry coordinate mapping approach (CCM) is utilized to improve the computational efficiency. This project can be divided into two parts: the first part deals with the implementation of StoR combustion model; and the second part concerns with the creation of a case that will use both StoR combustion model and CCM chemistry speedup approach.

The models shown in this project is only a demo program, the PDF is presumed and stochastic reactor only consisting of three temperature states. In the future, in order to reach higher accuracy and lower computational cost, I will develop two solvers to optimize the combination of those two models for both gas phase and liquid-gas phase combustion separately. We named it as 'multi-zones stochastic reactor (MStoR)', which can achieve deep integration. If interested, feel free to contact me or our group (Division of Fluid Mechanics, Lund University) for detailed information and cooperation.

Chapter 2

Theory

This chapter is designed to introduce the theory of the combustion model and chemistry speedup approach. In the following paragraphs, those parts are presented in order: the basic theory of Probability density function, a brief introduction to chemistry coordinate mapping approach, and the combination of PDF and CCM.

2.1 Probability density function

Probability density function (PDF) methods is one of the most powerful and promissing apprpaches [3] to deal with the turbulence and combustion interaction (TCI) problem.

In numerical simulation, given a small reaction cell with homogenous temperature and species molar fraction inside, the state of the cell and its reaction rate can be uniquely identified by two composition variables, the local temperature and species mass fraction $\mathbf{\Phi} = \mathbf{\Phi}(T, \mathbf{Y})$.

Let ϕ denotes the value of a composition variable, for example the mass fraction of components or temperature, at a particular location and time $(\mathbf{X}, t)$ in a turbulent reactive flow. The probability density function(PDF) of the random variable ϕ is denoted as $f_\phi(\psi)$ [4]. ϕ can be decomposed into its mean and the fluctuation, which are denoted by $\overline{\phi}$ and ϕ' respectively. The expectation of the random variable ϕ is denoted by $\mathbf{E}(\phi)$ or $\overline{\phi}$, and the variance of variable ϕ is defined as the expectation of the square of the fluctuation, denoted as $\mathbf{D}(\phi)$ or $\overline{\phi'^2}$. They are defined by,

$$\phi = \overline{\phi} + \phi'. \tag{2.1}$$

$$\mathbf{E}(\phi) = \overline{\phi} \equiv \int_{-\infty}^{\infty} \psi f_\phi(\psi) d\psi. \tag{2.2}$$

$$\mathbf{D}(\phi) = \overline{\phi'^2} \equiv \mathbf{E}([\phi - \overline{\phi}]^2) = \int_{-\infty}^{\infty} (\psi - \overline{\phi})^2 f_\phi(\psi) d\psi. \tag{2.3}$$

Given a function Q, if Q is a function of ϕ, then the expectation, or the ensemble average, $\overline{Q}$ can be exactly calculated by Eq.2.4[4]. For linear function, $\overline{\mathbf{Q}(\phi)}$ is equal to $\mathbf{Q}(\bar{\phi})$, but for non-linear function, the analytical solution for the integration in Eq. 2.4 is not easy to find. An alternative method is to discrete the PDF function into finite segments.

$$\overline{\mathbf{Q}(\phi)} = \int_{-\infty}^{\infty} \mathbf{Q}(\psi) f_\phi(\psi) d\psi. \tag{2.4}$$

$$\begin{aligned} \int_{-\infty}^{\infty} \mathbf{Q}(\psi) f_\phi(\psi) d\psi &= \sum_{i=1}^{N-1} \int_{\phi_i}^{\phi_{i+1}} \mathbf{Q}(\psi) f_\phi(\psi) d\psi, \phi_1 = -\infty, \phi_N = \infty. \\ &= \sum_{i=1}^{N-1} \mathbf{Q}(\phi_i) \int_{\phi_i}^{\phi_{i+1}} f_\phi(\psi) d\psi, \psi_i \in [\phi_i, \phi_{i+1}]. \end{aligned} \tag{2.5}$$

Let us denote the PDF definite integration within segment $[\phi_i, \phi_{i+1}]$ in Eq.2.5 as α_i. Then $\overline{\mathbf{Q}(\phi)}$, the averaged value of $\mathbf{Q}$, can be approximately evaluated by a set of discrete magnitudes of $\mathbf{Q}(\phi_i)$ and corresponding coefficients α_i.

$$\overline{\mathbf{Q}(\phi)} = \sum_{i=1}^{N} \alpha_i \mathbf{Q}(\phi_i), \alpha_i \equiv \int_{\psi_i}^{\psi_{i+1}} f_\phi(\psi) d\psi. \tag{2.6}$$

2.2 The chemistry coordinate mapping approach

As mentioned above, in reaction flow simulation with finite rate chemistry method, the reaction rate $\dot{\omega}$ is a typical non-linear function of temperature T and species mass fraction $\mathbf{Y}$. The solution of reacting flow governing equations is computational expensive. In order to reduce the computational cost, various of approaches have been developed for efficient coupling with flow simulation and speeding up the chemistry calculation.

Fig. 2.1 shows the concept of chemistry fractional step in reaction flow simulation with finite rate chemistry method. The first and last lines represent for the discrete cells in physical phase at specific position $\mathbf{x}_i$ at time t_n and $t_n + \Delta t$, the second line denotes the chemistry fractional step, in which, each of the cells are regarded as an independent combustor and the chemical equations are solved independently. In practice, the composition variable, for example the mass fraction of components or temperature at different space are not completely independent. As we can see in Fig. 2.1, the variable in different cells at time t_n have the same value, therefore there is space for solver to improve the computational efficiency.

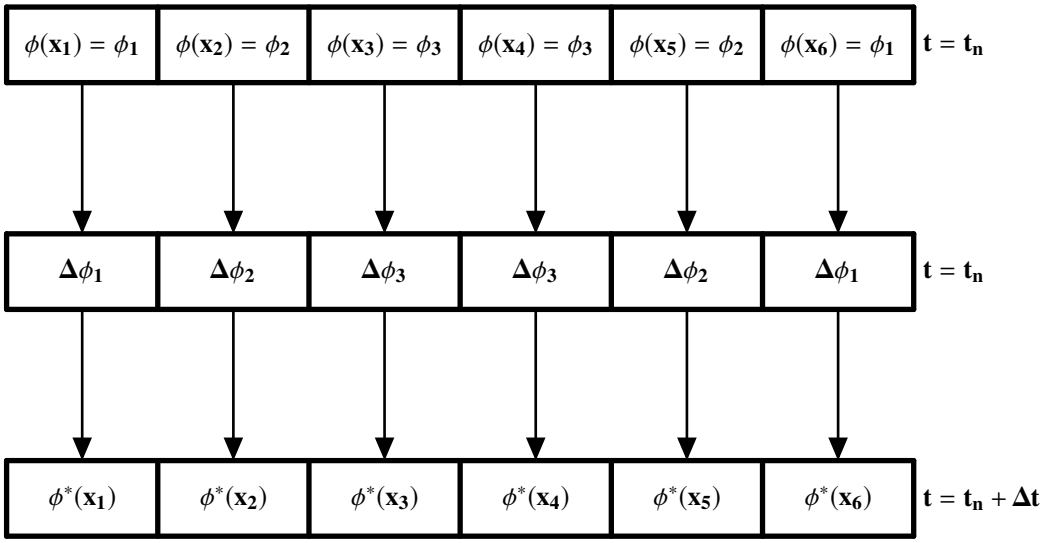

Fig. 2.1. The concept of finite rate chemistry method without chemistry speedup.

The chemistry coordinate mapping (CCM) approach is a promising speedup method in reactive flow simulation. In CCM method, a multidimensional thermodynamic space is adopted for solving chemistry in phase space instead of physical space. This method provides an efficient and high precision mapping procedure between the physical space and the chemistry phase space, each of the chemistry phase space corresponds to one or more cells in physical space, resulting in a speedup of the numerical integration [1]. Fig.2.2 illustrates the idea of CCM method. As shown in the second line, the composition variables in physical space at time step t_n are mapped into a thermodynamic space according to its value. Then, chemistry solver is applied for source term from the second line to the third line, only based on the mapped thermodynamic space. In the last step, the results are distributed back to physical space according to its original corresponding relationship.

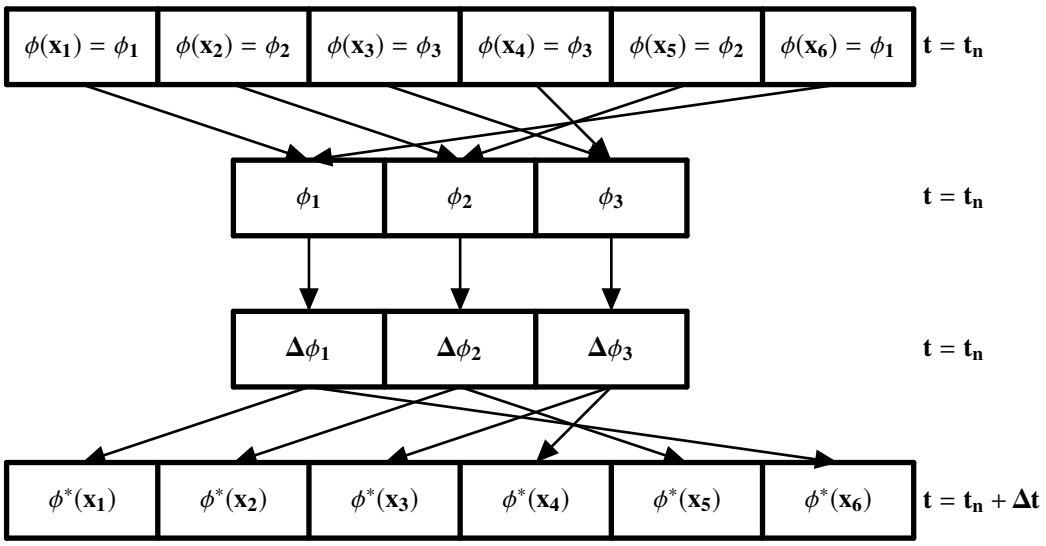

Fig. 2.2. The concept of the chemistry coordinate mapping approach.

2.3 The combination of PDF and CCM

From PDF method viewpoint, any cell in physical space can be expressed by many specific composition variable states $\mathbf{\Phi} = \mathbf{\Phi}(T, \mathbf{Y})$, each state has a specific composition and its corresponding possibilities, as shown in Eq.2.6. Furthermore, it is worth noting that the mapping process of CCM approach provides a set of feasible composition variable states in thermodynamic phase space. Therefore, it is possible to adopt the chemical results data in CCM phase space to reconstruct the discrete PDF without increasing computational burden.

Fig.2.3 shows the idea of the combination of PDF discretization and chemistry coordinate mapping approach. In this sketch, from line three to line four, the composition variables state $\mathbf{\Phi}^*$after the chemistry fractional step are updated by all of the chemical results in phase space at time t_n with a set of coefficients $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$. Then, the problem is simplified and focused on how to find this set of coefficients $\boldsymbol{\alpha}$.

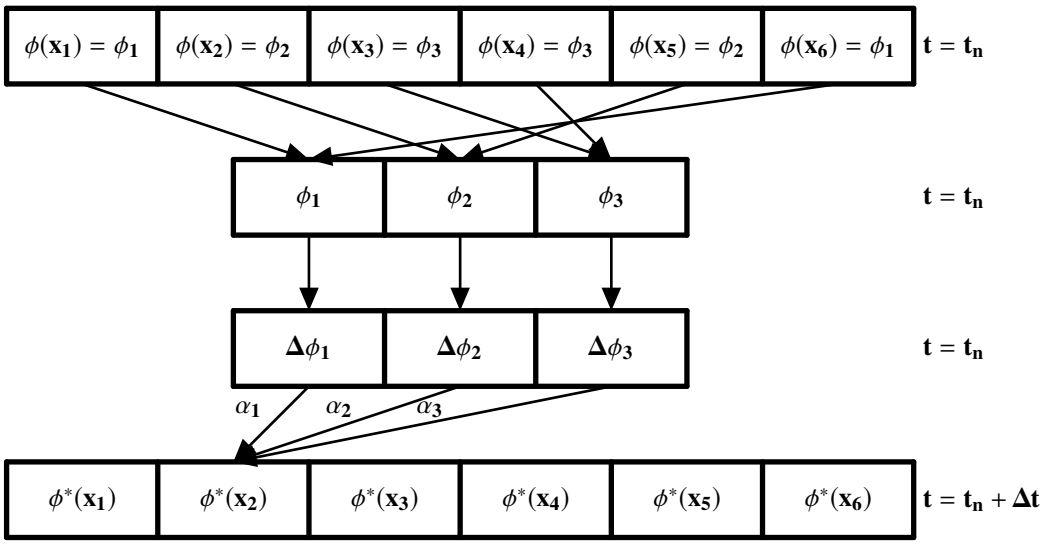

Fig. 2.3. The idea of the combination of PDF discretization and coordinate mapping approach.

Given any of the composition variable ϕ with expectation $\overline{\phi}$ and variance σ, the expectation can be approximated as the averaged value within the specific cell in physical space, while the variance of this

composition variable is not available in CFD system, and should to be modeled. Focusing on an arbitrary cell in physical space, the composition variable states are rearranged in order of magnitudes $\phi_1 < \phi_2 < ... < \phi_N$. Therefore, the expectation of any function Q of composition variable state $\overline{\mathbf{Q}(\phi)}$ can be piecewise fitted by a linear interpolation $\mathbf{L}(\phi)$ based on a set of discrete magnitudes $\mathbf{Q}(\phi_i)$. After a serieous of derivation (too long to put into the report, if interesting, Email me.), the set of coefficients $\boldsymbol{\alpha}$ solved.

$$\alpha_i = \begin{cases} 0, & \phi_i \notin [\overline{\phi} - 3\sigma, \overline{\phi} + 3\sigma] \\ \frac{\mathbf{E}_\phi(\phi_1 < \phi < \phi_2) - \phi_2\left[F_\phi(\phi_2) - F_\phi(\phi_1)\right]}{\phi_1 - \phi_2}, & \phi_i = \overline{\phi} - 3\sigma \\ \dfrac{\mathbf{E}_\phi(\phi_i < \phi < \phi_{i+1}) - \phi_{i+1}\left[F_\phi(\phi_{i+1}) - F_\phi(\phi_i)\right]}{\phi_i - \phi_{i+1}} & \\ \quad - \dfrac{\mathbf{E}_\phi(\phi_{i-1} < \phi < \phi_i) - \phi_{i-1}\left[F_\phi(\phi_i) - F_\phi(\phi_{i-1})\right]}{\phi_{i-1} - \phi_i}, & \phi_i \in (\overline{\phi} - 3\sigma, \overline{\phi} + 3\sigma) \\ -\frac{\mathbf{E}_\phi(\phi_{N-1} < \phi < \phi_N) - \phi_{N-1}\left[F_\phi(\phi_N) - F_\phi(\phi_{N-1})\right]}{\phi_{N-1} - \phi_N}, & \phi_i = \overline{\phi} + 3\sigma \end{cases} \tag{2.7}$$

Chapter 3

Implementation

Here, we will investigate the structure of combustion model in OpenFOAM, and try to modify an existing EDC combustion model to our stochastic reactor model.

3.1 The structure of combustion model library

Firstly, we need to locate the combustion model classes directory and files which need to be modified. The source code of library 'libcombustionModels.so' can be found by typing the following commands:

```
OFv1806
cd $WM_PROJECT_DIR/src/combustionModels
```

The following steps are similar with the modification of 'kOmegaSSTF' for Turbulence model classes in the tutorials of the course exercise. If you have installed 'tree' software, let us follow the instruction to look into those classes structure by using 'tree' command.

```
tree $WM_PROJECT_DIR/src/combustionModels
```

Fig. 3.1 shows the tree structure of combustion models in OpenFOAM, where we can find the 'laminar', or well stirred reactor (WSR) model. In this model, the composition variables $(T, \mathbf{Y})$ in each computational fluid dynamics (CFD) cell is regarded as homogeneous. The eddy dissipation concept (EDC) model is listed as well. In EDC, the mixture within CFD cell is divided into two states $\mathbf{\Phi} = \mathbf{\Phi}(T, \mathbf{Y})$, the volume of two states and the state composition variables $(T, \mathbf{Y})$ such as temperature and species mass fraction, are modeled by local turbulence parameters.

As we can see in Fig. 3.2, the stochastic reactor can consist of various states. In theory, the estimation of chemical reaction term will be more accurate if abundant stochastic states are used. From this viewpoint, the concept of our stochastic reactor (StoR) model is similar to eddy dissipation concept (EDC), so we will copy and revise the files which is related to 'EDC'.

Therefore, we copy the whole directory into user project folder $WM_PROJECT_USER_DIR without changing anything.

```
foam
cp -r --parents src/combustionModels $WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/src/combustionModels
```

Revise the the final 'libcombustionModels.so' location from $FOAM_LIBBIN to $FOAM_USER_LIBBIN in Make/files, and check whether the changes has been made by using 'tail' command. Compile it after changing.

```
sed -i s/FOAM_LIBBIN/FOAM_USER_LIBBIN/g Make/files
tail Make/files
wmake
```

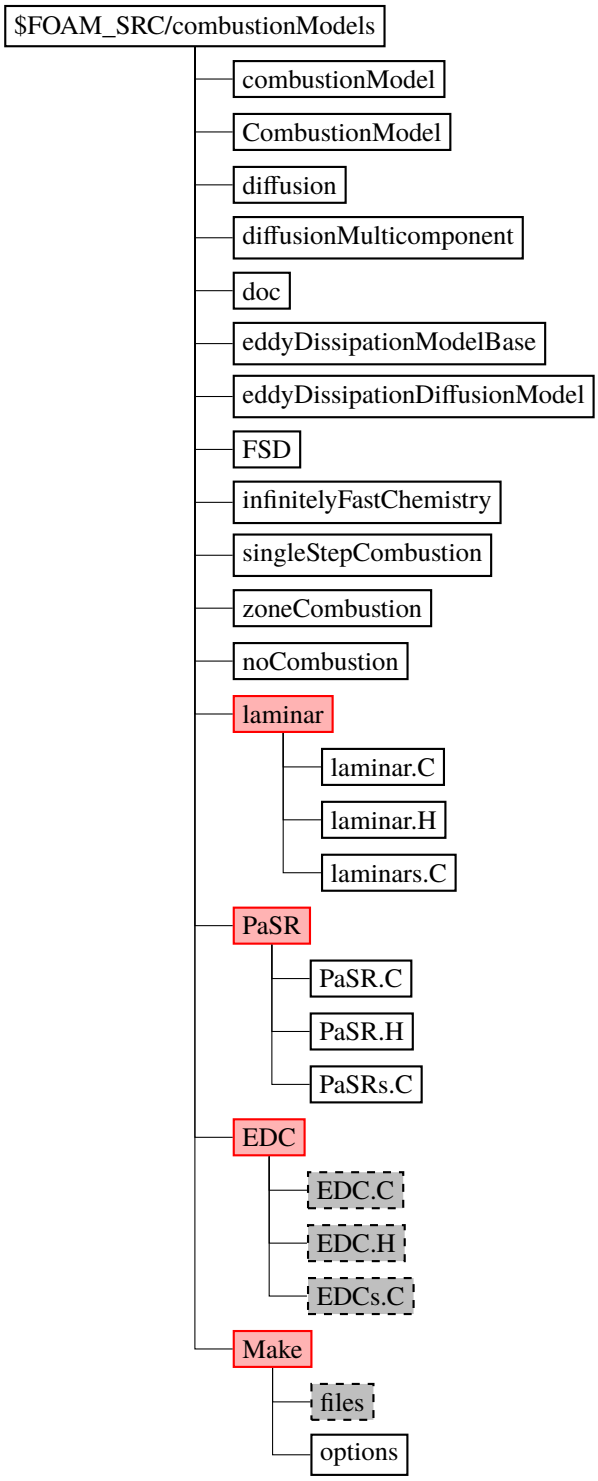

Fig. 3.1. The structure of combustion models in OpenFOAMv1806.

It takes about 2 minutes to finish the compilation. When the compilation is finished, we can check the location of new compiled library by using 'which' and 'grep' command.

```
ldd `which sprayFoam` | grep libcombustionModels.so
```

If you see the following information in the output, it means that new compiled '.so' file can be found in $FOAM_USER_LIBBIN folder and it can be used by existing solvers. The new compiled '.so' file will be used instead of the ones in the main installation, since the environment variable LD_LIBRARY_PATH points at $FOAM_USER_LIBBIN directory before pointing at the original directory ($FOAM_LIBBIN).

```
/*********************    OUTPUT INFORMATION    *********************/
libcombustionModels.so => $FOAM_USER_LIBBIN/libcombustionModels.so ...
```

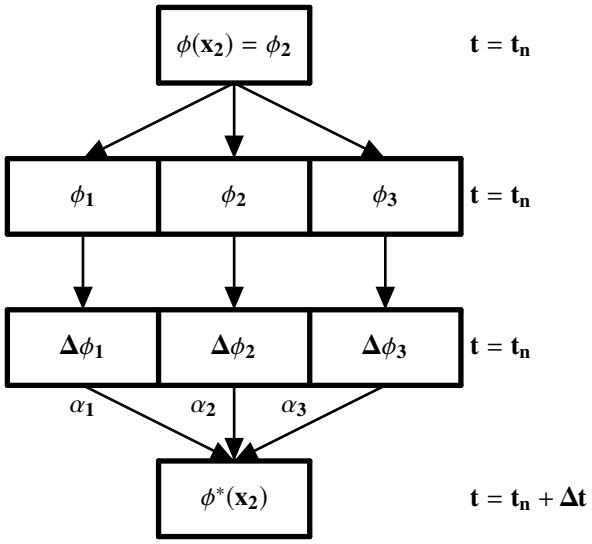

Fig. 3.2. The concept of stochastic reactor.

Now we need to find out the files which are related to the 'EDC', copy them and revise according to the new model. We can do this by searching for keywords 'EDC' as below:

```
grep -r EDC
```

You will get the following results, where some of the repeat files are hidden and eddyDissipationModelBase.C and eddyDissipationModelBase.H can be ignored because the key words are 'CEDC'. Thus EDC/EDC.C, EDC/EDC.H, EDC/EDCs.C and Make/files need to be modified. Those files are marked in the directory tree with grey dotted line in Fig. 3.1.

```
/*********************    OUTPUT INFORMATION    *********************/
EDC/EDC.C:#include "EDC.H"
...
EDC/EDC.H:      Foam::combustionModels::EDC
...
EDC/EDCs.C:#include "EDC.H"
...
eddyDissipationModelBase/eddyDissipationModelBase.C:      CEDC_(...
...
eddyDissipationModelBase/eddyDissipationModelBase.H:      scalar CEDC_
...
Make/files:EDC/EDCs.C
```

Copy the EDC folder and rename it as StoR. Rename the files in StoR directory.

```
cp -r EDC StoR
mv StoR/EDC.C StoR/StoR.C
mv StoR/EDC.H StoR/StoR.H
mv StoR/EDCs.C StoR/StoRs.C
```

Then, we need to replace the keywords 'EDC' with 'StoR'

```
sed -Ei 's/EDC/StoR/g' StoR/StoR*.*
```

By typing above command, the copy and revision of EDC/EDC.C, EDC/EDC.H and EDC/EDCs.C files are finished. Then we need to add a new line

```
StoR/StoRs.C
```

into Make/files, to let the compiler knows that a new added model 'StoR' needs to be compiled.

```
vi Make/files
```

```
/*********************    REPLACE FILES: Make/files    *********************/
combustionModel/combustionModel.C
CombustionModel/CombustionModel/CombustionModels.C

diffusion/diffusions.C
infinitelyFastChemistry/infinitelyFastChemistrys.C

PaSR/PaSRs.C
EDC/EDCs.C
StoR/StoRs.C
eddyDissipationDiffusionModel/eddyDissipationDiffusionModels.C

laminar/laminars.C


FSD/reactionRateFlameAreaModels/consumptionSpeed/consumptionSpeed.C
FSD/reactionRateFlameAreaModels/reactionRateFlameArea/reactionRateFlameArea.C
FSD/reactionRateFlameAreaModels/reactionRateFlameArea/reactionRateFlameAreaNew.C
FSD/reactionRateFlameAreaModels/relaxation/relaxation.C
FSD/FSDs.C


diffusionMulticomponent/diffusionMulticomponents.C


zoneCombustion/zoneCombustions.C

noCombustion/noCombustions.C

LIB = $(FOAM_USER_LIBBIN)/libcombustionModels
```

Update the lnInclude links and compile to get a new 'libcombustionModels.so' library with 'StoR' model.

```
wmakeLnInclude -u .
wmake
```

So far, a new combustion model named 'StoR' has been created, which includes the identical operation with 'EDC' model but different name. Next, we will create a case to run 'sprayFoam' solver with 'StoR' model to test it.

3.2 The creation of a case with StoR model

Please check the current path by typing

```
pwd
```

It should be still in '$WM_PROJECT_USER_DIR/src/combustionModels', since we will come back to revise combustion model, you can choose jump to case path or stay here to run the case in distance. It is important that we need to jump back to '$WM_PROJECT_USER_DIR/src/combustionModels' directory after finish this section. The command of this section is independent of current path.

Copy a spray combustion tutorial 'aachenBomb' to '$FOAM_RUN' directory, renamed it as 'aachenBombStoR'. Replace the constant/combustionProperties file from 'SandiaD_LTS' case, which employs a 'EDC' combustion model.

```
cp -r $FOAM_TUTORIALS/lagrangian/sprayFoam/aachenBomb \
$FOAM_RUN/aachenBombStoR
cp -r $FOAM_TUTORIALS/combustion/reactingFoam/RAS/SandiaD_LTS\
/constant/combustionProperties $FOAM_RUN/aachenBombStoR/constant
```

Replace the keyword 'EDC' with 'StoR' in constant/combustionProperties file.

```
sed -Ei 's/EDC/StoR/g' $FOAM_RUN/aachenBombStoR/constant\
/combustionProperties
```

If you check the revised combustionProperties files, it should be like that:

```
/**********   REPLACE FILES: constant/combustionProperties   **********/
/*---------------------------------*- C++ -*----------------------------*\
| =========                 |                                            |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox      |
|  \\    /   O peration     | Version:  v1806                            |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                 |
|    \\/     M anipulation  |                                            |
\*-------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      combustionProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

combustionModel  StoR;

active  true;

StoRCoeffs
{
    version v2005;
}

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

Create mesh and run the case to see the information in the output whether the StoR model works.

```
blockMesh -case $FOAM_RUN/aachenBombStoR  > & log
sprayFoam -case $FOAM_RUN/aachenBombStoR  > & log
```

If the information in the output is like the following outputs and no error printed, it means that StoR model has been added into combustion model.

```
/*---------------------   OUTPUT INFORMATION   ---------------------*/
...
Creating combustion model

Selecting combustion model StoR
Selecting chemistry solver
{
    solver          ode;
    method          standard;
}
...
```

3.3 The modification of StoR model

As mentioned before, there are only four files that are related with StoR model, where Make/files has been revised. Only three files need to be modified, let us start with StoR/StoRs.C file. Open it by using vi command.

```
vi StoR/StoRs.C
```

This is a template file, a Static Member Data Enum for EDC combustion model version is defined (line 13-26), we do not need it any more, comments it as follows. Reserve the headers included and namespace Foam dictionary.

```
/*---------------------      StoRs.C      ---------------------*/

...

#include "makeCombustionTypes.H"

#include "psiReactionThermo.H"
#include "rhoReactionThermo.H"
#include "StoR.H"

// * * * * * * * * * * Static Member Data  * * * * * * * * * * * * * //
/*
const Foam::Enum
<
    Foam::combustionModels::StoRversions
>
Foam::combustionModels::StoRversionNames
{
    { StoRversions::v1981, "v1981" },
    { StoRversions::v1996, "v1996" },
    { StoRversions::v2005, "v2005" },
    { StoRversions::v2016, "v2016" },
};

const Foam::combustionModels::StoRversions
```

```
26 Foam::combustionModels::StoRdefaultVersion(StoRversions::v2005);
27 */
28
29 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
30
31 namespace Foam
32 {
33
34 makeCombustionTypes(StoR, psiReactionThermo);
35 makeCombustionTypes(StoR, rhoReactionThermo);
36
37 }
38
39 // ************************************************************************* //
```

Then open StoR/StoR.H file to modify the declaration of StoR class and its members. In this file, I only modify the private data and functions.

```
vi StoR/StoR.H
```

```
1  /*********************      StoR.H      *********************/
2
3  ...
4
5  #ifndef StoR_H
6  #define StoR_H
7
8  #include "../laminar/laminar.H"
9  #include "Enum.H"
10 #include "laplaceFilter.H"//Add a header for the calculation of variance
11 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
12
13 namespace Foam
14 {
15 namespace combustionModels
16 {
17
18 /*---------------------------------------------------------------------------*\
19                          Class StoR Declaration
20 \*---------------------------------------------------------------------------*/
21
22 template<class Type>
23 class StoR
24 :
25     public laminar<Type>
26 {
27     // Private data
28
29     //- The selected model version
30 //         StoRversions version_;
31
32         //- The followings are for StoR combustion model
33         //- The parameter for presumed PDF
34         word PDF_Name_;
35         scalar spanZoneForPDF_;
```

```
36          scalar truncationForPDF_;
37          scalarField P_i_;//The posibility for Zone i in spanZoneForPDF;
38          scalarField T_i_;//The normalized temperature value for Zone i;
39
40      //- Filter parameter
41          laplaceFilter spaceFilter_;
42          scalar deviationSimilarCoeff_;//similarity model for deviation;
43
44          //- volScalarField
45          volScalarField Tsgs_;
46          volScalarField Qdot_;
47          PtrList<fvScalarMatrix> R_;
48
49      // Private Member Functions
50
51          //- Disallow copy construct
52          StoR(const StoR&);
53
54          //- Disallow default bitwise assignment
55          void operator=(const StoR&);
56
57
58                  //- The followings are for StoR combustion model
59      //- The functions for presumed PDF
60          tmp<scalar> PDF_Norm(scalar, scalar, scalar);
61          tmp<scalar> PDF_Density(scalar, word);
62          tmp<scalar> PDF_Accumulate(scalar, word);
63          tmp<scalar> PDF_Expection(scalar, word);
64          tmp<scalar> PDF_Expection(scalar, scalar, word);
65          tmp<scalar> PDF_Density(scalar, scalar, scalar, word);
66          tmp<scalar> PDF_Accumulate(scalar, scalar, scalar, word);
67          tmp<scalar> PDF_Expection(scalar, scalar, scalar, word);
68          tmp<scalar> PDF_Expection(scalar,scalar,scalar,scalar,word);
69          tmp<scalarField> Alpha_Calculation(scalar,scalar,scalar,word);
70          tmp<scalarField> NormT_Calculation(scalar,scalar,scalar,word);
71
72
73  public:
74
75      //- Runtime type information
76      TypeName("StoR");
77
78
79      // Constructors
80
81          //- Construct from components
82          StoR
83          (
84              const word& modelType,
85              const fvMesh& mesh,
86              const word& combustionProperties,
87              const word& phaseName
88          );
89
```

```
90
91      //- Destructor
92      virtual ~StoR();
93
94
95      // Member Functions
96
97          //- Correct combustion rate
98          virtual void correct();
99
100         //- Fuel consumption rate matrix.
101         virtual tmp<fvScalarMatrix> R(volScalarField& Y) const;
102
103         //- Heat release rate [kg/m/s3]
104         virtual tmp<volScalarField> Qdot() const;
105
106         //- Update properties from given dictionary
107         virtual bool read();
108
109 };
110
111
112 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
113
114 } // End namespace combustionModels
115 } // End namespace Foam
116
117 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

As we can see, the header #include "laplaceFilter.H" has been added for the calculation of temperature variance, the member variable spaceFilter_ is a laplaceFilter which depends on this header. The theory of the variance calculation will be introduced in Eq.4.1, which is calculated from the temperature distribution. Apart from that, we create a volScalarField Tsgs_ to store the variance of local temperature. The other private data are PDF_Name_ and spanZoneForPDF_, which are the categories of presumed PDF and the amount of stochastic states for each cells. The truncationForPDF_ is used to bound the temperature fluctuations of stochastic states. Since there is a limitation of temperature in physic, but some of the PDF is unbounded, the default value of truncationForPDF_ is unity, means that the temperature of stochastic states are limited in the region of $[\overline{T} - 3\sigma, \overline{T} + 3\sigma]$, where σ is the variance of the local temperature.

scalarField P_i_ and T_i_ are the most important value which is used to specific the possibility and normalized temperature value for all the stochastic states. For example, we choose states $\overline{T} - \sigma$, $\overline{T}$ and $\overline{T} + \sigma$ as three temperature states, then it is possibility could be calculated according to Eq.2.7 and stored in P_i_.

In addition, two virables are created for the storage of heat release rate and species consumption for each of the cells and species. There are Qdot_ and R_ respectively. Those two variables are the essential outputs for the source term of species transportation and energy equation. They can be accessed by calling function Qdot() and R(). Those two functions will be rewritten in StoR/StoR.C file.

Apart from that, you can found a series of private functions, such as PDF_Norm(scalar, scalar, scalar), PDF_Density(scalar, word), Alpha_Calculation(scalar,scalar,scalar,word) and so on. Those are the stochastic posibility P_i_ calculation functions, it is the implementation of Eq.2.7.

After modifying the StoR/StoR.H file, save it and open the last file StoR/StoR.C to edit by typing

```
vi StoR/StoR.C
```

```
1 /*********************      StoR.C      *********************/
2
3 ...
```

```
4
5  #include "StoR.H"
6  #include "fixedValueFvPatchFields.H"
7  // * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * //
8
9  template<class Type>
10 Foam::combustionModels::StoR<Type>::StoR
11 (
12     const word& modelType,
13     const fvMesh& mesh,
14     const word& combustionProperties,
15     const word& phaseName
16 )
17 :
18 laminar<Type>(modelType, mesh, combustionProperties, phaseName),
19 PDF_Name_(this->coeffs().template lookupOrDefault<word>
20 ("temperaturePDF", "NormalDistribution")),
21 spanZoneForPDF_(3),
22 truncationForPDF_(2),
23 truncationForPDF_(this->coeffs().lookupOrDefault("truncationForPDF",1)),
24 //The posibility for Zone i in spanZoneForPDF;
25 P_i_( spanZoneForPDF_ ),
26 //The normalized temperature value for Zone i;
27 T_i_( spanZoneForPDF_ ),
28 //coeff_=12, means that use near 6 points to calculate the filterd data
29 spaceFilter_( mesh, 12 ),
30 deviationSimilarCoeff_(this->coeffs().
31 lookupOrDefault("deviationSimilarCoeff", 1)),
32 Tsgs_
33 (
34         IOobject
35         (
36                 "Tsgs",
37                 mesh.time().timeName(),
38                 mesh,
39                 IOobject::NO_READ,
40                 IOobject::AUTO_WRITE
41         ),
42         mesh,
43         dimensionedScalar("Tsgs", dimTemperature, 0.0 ),
44         fixedValueFvPatchField<scalar>::typeName
45 ),
46 Qdot_
47 (
48         IOobject
49         (
50                 "Qdot",
51                 this->mesh().time().timeName(),
52                 this->mesh(),
53                 IOobject::NO_READ,
54                 IOobject::NO_WRITE,
55                 false
56         ),
57         this->mesh(),
```

```
58          dimensionedScalar("Qdot", dimEnergy/dimVolume/dimTime, 0.0)
59 ),
60 R_(this->thermo().composition().Y().size())
61 {
62          forAll(R_, i)
63      {
64          R_.set
65          (
66              i,
67              new fvScalarMatrix
68              (
69                                      this->thermo().composition().Y()[i],
70                                      dimMass/dimTime
71              )
72          );
73      }
74
75          forAll(R_, speciesI) R_[speciesI] = 0*R_[speciesI];
76
77      truncationForPDF_[0] = -1;
78      truncationForPDF_[1] = 1; //Default Number
79
80 //     P_i_[0] = 0.24; P_i_[1] = 0.52; P_i_[2] = 0.24;//Default Number
81 //     T_i_[0] = -1;   T_i_[1] = 0;    T_i_[2] = 1;    //Default Number
82 P_i_ = this->Alpha_Calculation( spanZoneForPDF_, truncationForPDF_[0],
83 truncationForPDF_[truncationForPDF_.size()-1], PDF_Name_);
84 T_i_ = this->NormT_Calculation( spanZoneForPDF_, truncationForPDF_[0],
85 truncationForPDF_[truncationForPDF_.size()-1], PDF_Name_);
86
87          Info<< "StoR Combustion Model constructed: presumed PDF name is"
88          << PDF_Name_ << " and Truncations are " << truncationForPDF_
89          << endl
90                  << " StoR posibility coeff array are " << P_i_
91                  << " normalized variances are  " << T_i_ << endl;
92
93 }
94
95
96 // * * * * * * * * * * * * * * * * Destructor  * * * * * * * * * * * * * * * //
97
98 template<class Type>
99 Foam::combustionModels::StoR<Type>::~StoR()
100 {}
101
102
103 // * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * //
104
105 template<class Type>
106 void Foam::combustionModels::StoR<Type>::correct()
107 {
108     forAll(R_, speciesI) R_[speciesI] = 0*R_[speciesI];
109     Qdot_ *= 0;
110     Tsgs_ *= 0;
111
```

```
112     if (this->active())
113     {
114     volScalarField& T = this->thermo().T();
115     const volScalarField T_org = this->thermo().T();
116     volScalarField T_Filter = 0*T_org;
117     T_Filter = spaceFilter_(T_org);
118     Tsgs_ = sqrt( deviationSimilarCoeff_
119     * mag(spaceFilter_((T_Filter - T_org)*(T_Filter - T_org))));
120     Tsgs_.correctBoundaryConditions();
121
122
123     scalar deltaT = 0.;
124     deltaT = this->mesh().time().deltaTValue();
125
126     for( label i=0; i< spanZoneForPDF_; i++)
127     {
128         //Update the stochastic temperature
129         T = T_i_[i]*Tsgs_ + T_org;
130         //Run the reaction calculation, get reaction rate
131         this->chemistryPtr_->solve(deltaT);
132         //Get the weighted reaction rate for a grid
133         Qdot_.ref() += P_i_[i] * this->chemistryPtr_->Qdot();
134         //Get the weighted reaction source term for each species
135         forAll(R_, speciesI)
136           R_[speciesI] += P_i_[i] * this->chemistryPtr_->RR(speciesI);
137     }
138
139     T = T_org;
140     }
141
142 }
143
144 template<class Type>
145 Foam::tmp<Foam::fvScalarMatrix>
146 Foam::combustionModels::StoR<Type>::R(volScalarField& Y) const
147 {
148     tmp<fvScalarMatrix> tSu(new fvScalarMatrix(Y, dimMass/dimTime));
149
150     fvScalarMatrix& Su = tSu.ref();
151
152     if (this->active())
153     {
154         const label specieI =
155             this->thermo().composition().species()[Y.member()];
156         Su += R_[specieI];
157     }
158
159     return tSu;
160 }
161
162
163 template<class Type>
164 Foam::tmp<Foam::volScalarField>
165 Foam::combustionModels::StoR<Type>::Qdot() const
```

```
166 {
167     return Qdot_;
168 }
169
170
171 template<class Type>
172 bool Foam::combustionModels::StoR<Type>::read()
173 {
174     if (Type::read())
175     {
176         PDF_Name_ = this->coeffs().template lookupOrDefault<word>
177         ("temperaturePDF", "NormalDistribution");
178         truncationForPDF_ = this->coeffs().lookupOrDefault
179         ("truncationForPDF", 1);
180         deviationSimilarCoeff_ = this->coeffs().lookupOrDefault
181         ("deviationSimilarCoeff", 1);
182         spanZoneForPDF_ = 3;
183
184         P_i_ = Alpha_Calculation( spanZoneForPDF_, truncationForPDF_[0],
185         truncationForPDF_[truncationForPDF_.size()-1], PDF_Name_);
186         T_i_ = NormT_Calculation( spanZoneForPDF_, truncationForPDF_[0],
187         truncationForPDF_[truncationForPDF_.size()-1], PDF_Name_);
188
189         Info<< "StoR Combustion Model constructed: presumed PDF name is"
190         << PDF_Name_ << " and Truncations are " << truncationForPDF_
191         << endl
192                 << " StoR posibility coeff array are " << P_i_
193                 << " normalized variances are  " << T_i_ << endl;
194
195         return true;
196     }
197     else
198     {
199         return false;
200     }
201 }
202
203 ...
```

Recompile the combustion model with new revised StoR model

```
wmake
```

Jump to the 'aachenBombStoR' case and run it again,

```
blockMesh -case $FOAM_RUN/aachenBombStoR
sprayFoam -case $FOAM_RUN/aachenBombStoR
more log.sprayFoam
```

Now, you can see the print information for new developed StoR model, it is generated by construction function when a StoR class object is created.

```
1 /*********************    OUTPUT INFORMATION    *********************/
2 ...
3
4 Creating combustion model
```

```
5
6  Selecting combustion model StoR
7
8  ...
9
10 Selecting ODE solver seulex
11     using integrated reaction rate
12 StoR Combustion Model constructed:
13 presumed PDF name is NormalDistribution
14 and Truncations are 2(-1 1)
15 posibility coeff array are 3(0.24 0.52 0.24)
16 normalized variances are  3(-1 0 1)
17 ...
```

Chapter 4

Tutorials

In Chapter 3, we use one tutorial case 'aachenBomb' to generate two cases 'aachenBombStoR' and 'aachenBombCCM' for StoR model and CCM approach separately. We will then dig into the tutorial case and explain the case geometry, injector configuration and chemical mechanism for a typical sprayFoam combustion case. Then, we will create our third case 'aachenBombCCMStoR' to run the 'sprayFoam' solver with both StoR model and CCM approach.

4.1 Case introduction

As described by Per Carlsson [5] in his report 'A dieselFoam tutorial', 'aachenBombCCM' tutorial case is similar with the one used in his OpenFOAM version, difference is that for the OpenFOAM version after OpenFOAM-2.0.x, the solver name was changed from 'dieselFoam' to 'sprayFoam', and the lagrangian library changes substantially. The case 'aachenBomb' in OpenFOAMv1806 is located at

$FOAM_TOTORIALS/lagrangian/sprayFoam/aachenBomb.

The geometry of the 'aachenBomb' is a cuboid, as described in the course report [5]. The injector is located on the top center of the geometry, which is specified in the file constant/sprayCloudProperties:

```
/*---------------------   sprayCloudProperties   ---------------------*/
...
constantProperties
{
    T0                  320;

    // place holders for rho0 and Cp0
    // - reset from liquid properties using T0
    rho0                1000;
    Cp0                 4187;

    constantVolume      false;
}


subModels
{
    particleForces
    {
        sphereDrag;
    }

    injectionModels
```

```
24      {
25          model1
26          {
27              type            coneNozzleInjection;
28              SOI             0;
29              massTotal       6.0e-6;
30              parcelBasisType mass;
31              injectionMethod disc;
32              flowType        flowRateAndDischarge;
33              outerDiameter   1.9e-4;
34              innerDiameter   0;
35              duration        1.25e-3;
36              position        (0 0.0995 0);
37              direction       (0 -1 0);
38              parcelsPerSecond 20000000;
39              flowRateProfile table
40              (
41                  (0              0.1272)
42                  (4.16667e-05    6.1634)
43                  (8.33333e-05    9.4778)
44                  (0.000125       9.5806)
45                  ...
46                  (0.001251       0.0000)
47                  (1000           0.0000)
48              );
49
50              Cd              constant 0.9;
51
52              thetaInner      constant 0.0;
53              thetaOuter      constant 10.0;
54
55              sizeDistribution
56              {
57                  type        RosinRammler;
58
59                  RosinRammlerDistribution
60                  {
61                      minValue        1e-06;
62                      maxValue        0.00015;
63                      d               0.00015;
64                      n               3;
65                  }
66              }
67          }
68      }
69
70      dispersionModel none;
71
72      patchInteractionModel standardWallInteraction;
73
74      heatTransferModel RanzMarshall;
75
76      compositionModel singlePhaseMixture;
77
```

```
78      phaseChangeModel liquidEvaporationBoil;
79
80      surfaceFilmModel none;
81
82      atomizationModel none;
83
84      breakupModel     ReitzDiwakar; // ReitzKHRT;
85
86      stochasticCollisionModel none;
87
88      radiation        off;
89
90      standardWallInteractionCoeffs
91      {
92          type             rebound;
93      }
94
95      RanzMarshallCoeffs
96      {
97          BirdCorrection   true;
98      }
99
100     singlePhaseMixtureCoeffs
101     {
102         phases
103         (
104             liquid
105             {
106                 C7H16                1;
107             }
108         );
109     }
110
111     liquidEvaporationBoilCoeffs
112     {
113         enthalpyTransfer enthalpyDifference;
114
115         activeLiquids     ( C7H16 );
116     }
117
118     ReitzDiwakarCoeffs
119     {
120         solveOscillationEq yes;
121         Cbag             6;
122         Cb               0.785;
123         Cstrip           0.5;
124         Cs               10;
125     }
126 ...
127 }
128
129
130 cloudFunctions
131 {}
```

```
132
133
134 // ************************************************************************* //
```

The constantProperties dictionary is used to define the liquid properties in the injector, such as temperature, density and heat capacity. The dictionary subModels contains the definition of injector parameter, break-up model etc. Table.4.1 shows the 12 models for particles force.

Model
BrownianMotion
SRF
SaffmanMeiLiftForce
TomiyamaLift
distortedSphereDrag
gravity
nonInertialFrame
nonSphereDrag
paramagnetic
pressureGradient
sphereDrag
virtualMass

Table 4.1. particleForces sub-models for the sprayCloudProperties.

Table.4.2 shows the available injector models. They define the shape of injector nozzle, whether it is a cone nozzle or a patch or set manually. The SOI (Line 28) is a abbreviation of 'start of injection', defining the injection time after the start of simulation. massTotal is the total mass of the injection process which has a dimension of kg, the SOI time and massTotal will combine with flowRateProfile table to get the mass flow rate and calculate how much mass of fuel should be injected at each time step. injectionMethod is either point or disc. flowType is a very import parameter and it is used to calculate the partical initial velocity profile. Three choices are available: 'constantVelocity', 'pressureDrivenVelocity' or 'flowRateAndDischarge'. It means that the initial velocity can be specified with a constant value or calculate from the pressure difference between injector and ambient pressure. 'flowRateAndDischarge' means that the velocities of particles are calculated from flow rate profile. It is worth mentioning that those velocity are the magnitude and the direction of it are calculated from a random process according to the injector type and cone angle.

Model
cellZoneInjection
coneInjection
coneNozzleInjection
fieldActivatedInjection
inflationInjection
injectedParticleDistributionInjection
injectedParticleInjection
manualInjection
none
patchFlowRateInjection
patchInjection

Table 4.2. Injection model types in sprayCloudProperties.

The outerDiameter and innerDiameter are used to specificed the size of the injector. Since we chose it as coneNozzleInjection, the shape of the injector is defined as a ring with a inner circle and outer circle and

fuel is injected between the inner ring and outer ring. duration is the valid injection duration. position is the location of injector, and the direction is the main axis for fuel injection. parcelsPerSecond is an esential parameter. Since the total number of particles is too many to captured one by one in reality, we imagine a parcel to contain some particels those have the same size, velocity and thermal properties. This is a kind of simplication, the parcelsPerSecond defines the parcel number, the more parcel it has the less particles in each parcel are. flowRateProfile table consists of a series of vectors with two components, the left one is the time (dimension, second) and the right side is the height of profile. There is no physical meaning for the value of the right side, it will be normalized according to duration time and massTotal to get the mass injection at each time step by interpolation.

The parameter Cd is the charge coefficient and it has a value between 0 and 1, determing the intial velocity magnitude if we chose the flowType as flowRateAndDischarge. thetaInner and thetaOuter are the shape of corresponding spray cone, the cone is regarded as hollow, which has a inner taper angle and outer taper angle.

Model
RosinRammler
binned
exponential
fixedValue
general
massRosinRammler
multiNormal
normal
uniform

Table 4.3. Injection model types in sprayCloudProperties.

sizeDistribution is the dictionary for the specification of initial particle size. Table.4.3 shows the 9 available distributions. If RosinRammler is choosen, the particle size will be distributed randomly according to the RosinRammler size distribution funtion. In some case, we may set it as an uniform distribtuion, it means that the initial particle size is only controlled by breakup and evaporation process.

The following models configuration are the definition of models for particles after injector, including the break-up model, evapoarion model and collision model. Some of it can be found in Per Carlsson's report [5]. As we can see in singlePhaseMixtureCoeffs dictionary, the fuel n-Heptane (C_7H_{16}) is injected.

After injection and evaporation, liquid fuel will transform to gas phase and react with oxygen, the reaction of the gas phase are defined in constant/thermalphysicalProperties.

```
/*---------------------*   thermophysicalProperties   *---------------------*/
...
CHEMKINFile        "$FOAM_CASE/chemkin/chem.inp";
CHEMKINThermoFile  "$FOAM_CASE/chemkin/therm.dat";
CHEMKINTransportFile "$FOAM_CASE/chemkin/transportProperties";

newFormat          yes;

inertSpecie        N2;

liquids
{
    C7H16;
}

solids
{}
// ************************************************************************* //
```

where we can found that the location of CHEMKINFile, CHEMKINThermoFile and CHEMKINTransportFile are specified in line 3-5, the solver will call for the functions in thermal class to read thermophysicalProperties and find the chemkin file location, read the reactions, species and its thermal properties from those three files. It is worth mentioning that in the old version of OpenFOAM, CHEMKINTransportFile is not necessary. The liquids dictionary defines the liquid phase properties which is needed in OpenFOAM liquid library.

Let's have a look at the CHEMKINFile, which is located in "$FOAM_CASE/chemkin/chem.inp", ELEMENTS keyword is the list of total elements in reaction solver system, SPECIE list is the reaction species such as fuel oxygen products and intermidiate products. REACTIONS defines the total reaction in the system. Each reaction consists of reactants products and three coefficients which is corresponding to the Arrhenius theory. Some of the reactions contain the reverse reaction. In this tutorial, a one step reaction is employed.

```
1  /*******************    $FOAM_CASE/chemkin/chem.inp    *******************/
2  ELEMENTS
3   H    O     C    N    AR
4  END
5  SPECIE
6  C7H16  O2  N2  CO2  H2O
7  END
8  REACTIONS
9   C7H16 + 11O2              => 7CO2 + 8H2O          5.00E+8   0.0    15780.0! 1
10          FORD      / C7H16 0.25 /
11          FORD      / O2 1.5 /
12 END
```

The explanation and modifaction of combustionProperties and chemistryProperties files will be introduced in section 4.2

4.2 Case modification

In this section, we will create a spray combustion case to run the sprayFoam with new-build StoR combustion model and CCM approach. Meanwhile, we will generate two batch file to run and clean the case automatically.

Let us source the OpenFOAM, copy and rename the tutorial case we introduce above.

```
foam
cp -r $FOAM_TUTORIALS/lagrangian/sprayFoam/aachenBomb\
$FOAM_RUN/aachenBombCCMStoR
cd $FOAM_RUN/aachenBombCCMStoR
```

Go into the folder and revise the combustionProperties file as follows. In this revised file, combustion model is set as StoR and active is set as true which means that the chemical reaction calculation is on. Sometime we need to turn it off, for example, when we use sprayFoam to simulate the spray and evaporation process without combustion.

```
vi constant/combustionProperties
```

```
1  /***************    constant/combustionProperties    ***************/
2  ...
3  combustionModel   StoR;
4
5  active   true;
6
7  StoRCoeffs
8  {
```

```
    //For Presumed PDF
    temperaturePDF          NormalDistribution;
    //From (-1 \sigma, +1 \sigma),1 stands for 1*standard variance.
    truncationForPDF        1;
    //Similarity model for getting variance of T
    //it is the similarity coefficients, default 1.
    deviationSimilarCoeff   1;
}

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

StoRCoeffs dictionary is used to set the parameters for StoR combustion model, such as PDF distribution and variance coefficient. It will be read in the construction of the combustion model or in read() function. We can keep it empty as well, then the parameters of StoR combustion model will be the default set, with NormalDistribution PDF for temperature.

Specifically, temperaturePDF is the name of presumed PDF, truncationForPDF is the truncation boundary for the PDF function, for example, if we set it as 3, means that the temperature of generated stochastic reactor will be located in the region $[\overline{T} - 3\sigma, \overline{T} + 3\sigma]$, where σ is the variance of the local temperature, $\overline{T}$ is the averaged value within the cell. Parameter deviationSimilarCoeff is a scale coefficient used to calculate the variance of temperature. As shown in Eq. 4.1.

$$\sigma^2 = \widetilde{T''^2} \approx C\left(\widetilde{\widetilde{T}^2} - \widehat{\widetilde{T}}^2\right) \tag{4.1}$$

where C is a scale coefficient which can be determined from dynamical method [6]. $\widetilde{}$ denotes the filtered value with filter width $\widetilde{\Delta}$, accordingly, $\widehat{}$ denotes the filtered value with filter width $\widehat{\Delta}$. In this work, the method [7] is applied, with the assumption that $\widehat{\Delta} = 2\widetilde{\Delta}$, then:

$$\widehat{\widetilde{\phi}}_{ijk} = \frac{1}{12}\left(6\tilde{\phi}_{i,j,k} + \tilde{\phi}_{i-1,j,k} + \tilde{\phi}_{i+1,j,k} + \tilde{\phi}_{i,j-1,k} + \tilde{\phi}_{i,j+1,k} + \tilde{\phi}_{i,j,k-1} + \tilde{\phi}_{i,j,k+1}\right). \tag{4.2}$$

After revising the combustionProperties, lets come to modify the chemistryProperties file which is located in constant/chemistryProperties and it is used to specify the chemistry model. We only need to add a line in chemistryType dictionary to turn CCM on and the other coefficients will be read from constant/CCMProperties files.

```
vi constant/chemistryProperties
```

```
/*****************    constant/chemistryProperties    ***************/

...

chemistryType
{
    chemistrySolver    ode;
    chemistryThermo    psi;
    CCM                on;      //add a line to choose CCM chemistry approach
}

chemistry          on;

initialChemicalTimeStep 1e-07;

odeCoeffs
{
    solver            seulex;
```

```
19      eps             0.05;
20  }
```

Let's have a look at constant/CCMProperties file, which will be loaded at the construction of CCM class object or calling readCCMproperties() function when we turn CCM on in chemistryProperties. In this file, the dimension of CCM is specifed by min:max:SpanZoneJe, min:max:SpanZoneT and min:max:SpanZoneXi. These are three mapping dimension for mass fraction, temperature and scalar dispassion rate. Apart from that, sometimes we need the concentration of some key species as additional CCM dimension. More information about that can be found in [1]. In this case, we use three basic dimension and inert species N_2 as CCM mapping dimension.

```
1  /*****************    constant/CCMProperties    *****************/
2
3  ...
4
5  babaCCM                                  on;
6  ratioOxygenToCarbonElementInFuel         0; //fuel:C7H16;
7  chemicallyFrozenT                        chemicallyFrozenT [0 0 0 1 0] 600.;
8  maxFlammabilityLimit                     maxFlammabilityLimit [0 0 0 0 0] 20;
9  minFlammabilityLimit                     minFlammabilityLimit [0 0 0 0 0] 1e-6;
10
11 ...
12
13 min:max:SpanZoneJe   0    20   0.01;
14 min:max:SpanZoneT    300  3000 5;
15 min:max:SpanZoneXi   0    1    0.025;
16
17 ...
```

In line 5, we can see the babaCCM is a switch for CCM application, chemicallyFrozenT is the temperature limit for the reaction, when the temperature of the cell is lower then chemicallyFrozenT, we think the chemical reaction is negligible, the same with maxFlammabilityLimit and minFlammabilityLimit, they define the mass fraction limitation, when it is lower than minFlammabilityLimit or higher than maxFlammabilityLimit, it is a inert cell, the chemical reaction is negligible. All of those inert cells will be mapped into one CCM cell.

After modifying those combustionProperties, chemistryProperties and CCMProperties, the case setup is done. But for convenience, we create two batch files to run the case automatically.

Create two batch files and edit them by using 'touch' and 'vi' command.

```
vi Allrun
```

```
1  /*****************************    Allrun    *****************************/
2
3  #!/bin/sh
4  cd ${0%/*} || exit 1                        # Run from this directory
5  . $WM_PROJECT_DIR/bin/tools/RunFunctions    # Tutorial run functions
6
7  runApplication blockMesh
8  runApplication $(getApplication)
9
10 #------------------------------------------------------------------------
```

```
vi Allclean
```

```
1  /*****************************    Allclean    *****************************/
2
```

```
#!/bin/sh
cd ${0%/*} || exit 1                            # Run from this directory
. $WM_PROJECT_DIR/bin/tools/CleanFunctions      # Tutorial clean functions

cleanCase

#------------------------------------------------------------------------
```

Allocate execution rights to the batch files by using command 'chmod', check the privilege of Allclean and Allrun files:

```
chmod 755 All*
ll All*
```

You can get the following results, means that batch files is executable now.

```
/*********************    OUTPUT INFORMATION    *********************/
-rwxr-xr-x 1 ... ... 243 nov 24 12:02 Allclean*
-rwxr-xr-x 1 ... ... 289 nov 24 12:02 Allrun*
```

Run the case and wait for completion, see the usage of StoR combustion model CCM chemistry approach, and find out the total execution time.

```
./Allrun
more $FOAM_RUN/aachenBombCCMStoR/log.sprayFoam
tail $FOAM_RUN/aachenBombCCMStoR/log.sprayFoam
```

```
/*********************    OUTPUT INFORMATION    *********************/

...
Creating combustion model

Selecting combustion model StoR<psiChemistryCombustion>
Selecting chemistry type
{
    chemistrySolver ode;
    chemistryThermo psi;
    CCM             on;
}

...

CCMChemistryModel: Number of species = 5 and reactions = 1
Reading CHEMKIN thermo data in new file format
 J_H_Ox= 0.0028 J_H_fu= 0.16
 J_C_Ox= 0.0332 J_C_fu= 0.84
 J_O_Ox= 0.2734 J_O_fu= 0
 this Species N2 is added to the CCM phase space with resolution of: 0.001
N2 dimention is added to CCM

...

Selecting ODE solver seulex

...

```

```
rho max/min : 29.3512 8.97779
smoothSolver:  ...
smoothSolver:  ...
ExecutionTime = 25804.4 s  ClockTime = 25805 s

particleDistribution distribtion1 output:
End
```

In order to compare the results and computational efficiency under the usage of combustion model and chemistry speedup approach, we can run a default tutorial case 'aachenBomb' without any modification as a contrast. The command is as follows:

```
cp -r $FOAM_TUTORIALS/lagrangian/sprayFoam/aachenBomb\
 $FOAM_RUN/aachenBomb
cd $FOAM_RUN/aachenBomb
cp $FOAM_RUN/aachenBombCCMStoR/All* $FOAM_RUN/aachenBomb
./Allclean
./Allrun
```

4.3 Results and discussion

Until now, four cases have been tested, the default 'aachenBomb' case with partial stirred reactor combustion model (PaSR), the StoR model case without chemistry speedup approach, CCM with laminar combustion model (WSR), CCM with stochastic reactor combustion model (StoR). Table. 4.4 shows the detailed case information. For each of the cases, we extract the execution time. In this section, the computation time and results will be shown and discussed.

Case Number	Case Name	combustion model	Execution Time(s)
1	aachenBomb	PaSR	38836
2	aachenBombStoR	StoR	
3	aachenBombCCM	WSR	10810
4	aachenBombCCMStoR	StoR	25804

Table 4.4. Initial conditions for the dieselFoam tutorial

Fig.4.1 shows the temperature variance distribution calculated by StoR combustion model at ignition time. The larger variance indicates the strong turbulence interaction. This fluctuation will affect the ignition time. This figure shows that the temperature variance along the liquid particles is not negligible. StoR and PaSR combustion can take turbulence interaction into account, in theory, therefore they can predict ignition time more correct.

Fig.4.2 shows the active CCM zones in computational domain at ignition time. The cell with same zone number share identical CCM zone. As can be seen, all of the blue color share one CCM zone which will be calculate only once. The maximum zone number is 5900, while the total cell is 160000. This will speed up the reaction calculation more than 30 times.

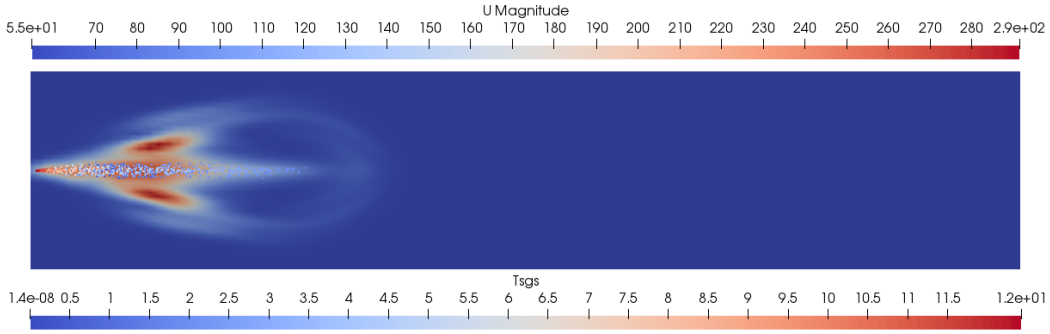

Fig. 4.1. The temperature variance at ignition time.

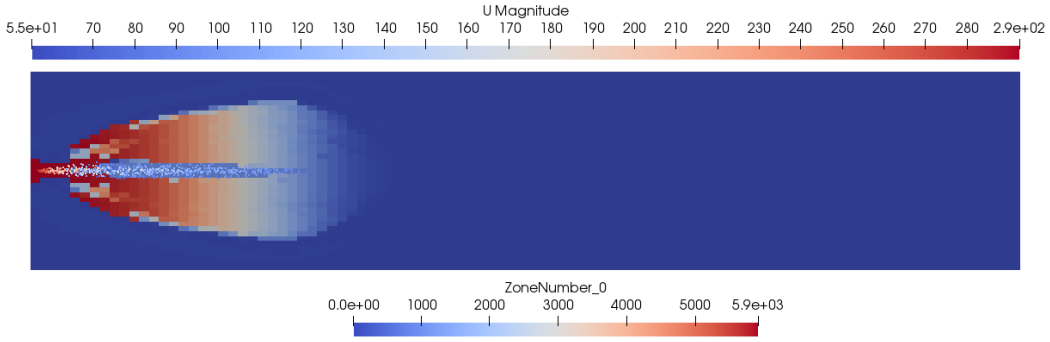

Fig. 4.2. The active zone number in CCM map for the aachenBombCCM case.

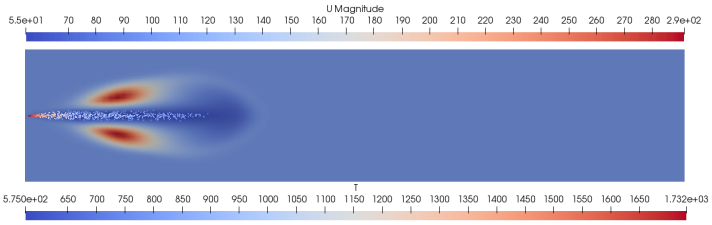

Fig. 4.3. Temperature distribution of the case with StoR combusion model.

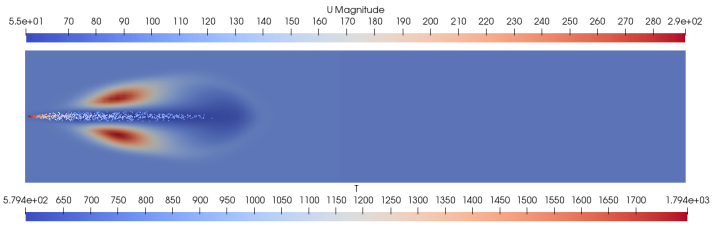

Fig. 4.4. Temperature distribution of the case with PaSR combusion model.

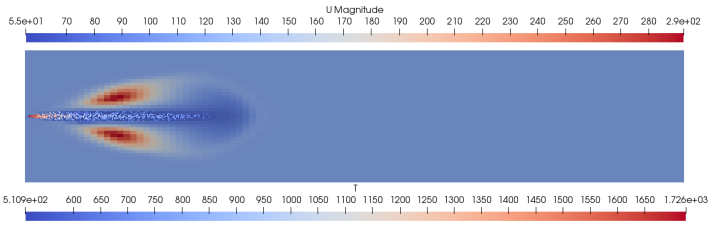

Fig. 4.5. Temperature distribution of the case with WSR combusion model.

Study questions

1. How the turbulence interaction is considered in StoR combustion model?
2. How to set the spray parameters such as injector position and fuel mass flow rate profile?
3. How to modify the EDC model to StoR in this tutorial?
4. How chemistry coordinate mapping (CCM) appraoch can improve computing efficiency in finite-rate chemistry based simulation?
5. How to deeply combine the CCM and StoR, for example, how do we run CCM to map the cells among three StoR states insteat of calculating them individually by CCM for three times.

Bibliography

[1] M. Jangi and X.-S. Bai, "Multidimensional chemistry coordinate mapping approach for combustion modelling with finite-rate chemistry," *Combustion Theory and Modelling*, vol. 16, no. 6, pp. 1109–1132, 2012.

[2] S. Xu, S. Huang, R. Huang, W. Wei, X. Cheng, Y. Ma, and Y. Zhang, "Estimation of turbulence characteristics from piv in a high-pressure fan-stirred constant volume combustion chamber," *Applied Thermal Engineering*, vol. 110, pp. 346–355, 2017.

[3] D. Haworth, "Progress in probability density function methods for turbulent reacting flows," *Progress in Energy and Combustion Science*, vol. 36, no. 2, pp. 168–259, 2010.

[4] S. B. Pope, "Pdf methods for turbulent reactive flows," *Progress in energy and combustion science*, vol. 11, no. 2, pp. 119–192, 1985.

[5] C. P., "A dieselfoam tutorial." *In Proceedings of CFD with OpenSource Software, 2008, Edited by Nilsson. H.*, 2008.

[6] C. D. Pierce and P. Moin, "A dynamic model for subgrid-scale variance and dissipation rate of a conserved scalar," *Physics of Fluids*, vol. 10, no. 12, pp. 3041–3044, 1998.

[7] H. Forkel and J. Janicka, "Large-eddy simulation of a turbulent hydrogen diffusion flame," *Flow, Turbulence and Combustion*, vol. 65, no. 2, pp. 163–175, 2000.