Cite as: Phanindra.P.Thummala: reactingTwoPhaseEulerFoam description. In Proceedings of CFD with OpenSource Software, 2016, Edited by Nilsson. H.,
http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2016

CFD with OpenSource software

A course at Chalmers University of Technology
Taught by Håkan Nilsson

Description of reactingTwoPhaseEulerFoam solver with a focus on mass transfer modeling terms

Developed for OpenFOAM-4.x

Author:
Phanindra Prasad Thummala
Anadolu University, Turkey
phanindrapt@anadolu.edu.tr

Peer reviewed by:
Fynn Jerome Ashmoneit
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 1, 2017

Learning outcomes

The reader will learn:

How to use it:

- how to use reactingTwoPhaseEulerFoam solver.

The theory of it:

- the theory of mass transfer and correlations used in mass transfer

How it is implemented:

- A detail description of the reactingTwoPhaseEulerFoam Source Code
- A tutorial, bubbleColumnEvaporatingReacting Tutorial, to apply this code

How to modify it:

- implement new mass transfer model in the reactingTwoPhaseEulerFoam code

Prerequisites

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

- Fundementals of Heat and mass Tranfer , Book by Frank.P Incropera
- Dimensional groups like Sherwood number,Reynolds,Number,Prandtl Number, Schmidt Number and Lewis Number
- Run standard document tutorials like damBreak tutorial
- It is strongly recommonded to gain a brief insight into the physics of two phase reactive flows from the following journal (if accessable):

  D. Darmana et.al.,2007. Detailed modelling of hydrodynamics, mass transfer and chemical reactions in a bubble column using a discrete bubble model: Chemisorption of CO 2 into NaOH solution, numerical and experimental study.Chemical Engineering Science 62,2556 to 2575

Overview

The report is arranged in five chapters:

- Introduction
- Various classes involved in the reactingTwoPhaseEulerFoam solver and their significance
- bubbleColumnEvaporatingReacting Tutorial
- Implementing Higbie mass transfer model
- Study questions

Chapter 1

Introduction

Mass transfer in multiphase (gas-liquid-solid) systems is one of the most critical processes occurring in chemical, petrochemical, and environmental engineering applications. Generally, it entails transport of species among phases through diffusion (physical) and/or chemical reactions in a special unit operation (reactor), allowing such a process to take place. Chemical reactions, often used to speed up the mass transfer rate, occur whenever species of different chemical potentials are brought into contact.In multiphase systems, the species mass transfer rate is controlled not only by the system pressure and temperature, but also by the conductance of mass transfer, concentration gradients, reaction kinetics, activation energy, etc. In some cases, either the conductance of mass transfer or the reaction kinetics could control the overall mass transfer rate; and the slowest one will be the rate limiting or controlling step. For instance, CO2 transfer from the polluted (gas) to the aqueous solution (liquid Mono Ethyl Amine) in absorption reactors could be the overall rate limiting step. In any case, understanding mass transfer behavior in multiphase systems requires precise knowledge of all aspects affecting the overall mass transfer rate.

In order to understand the the basic mass tranfer process let us define the follwing acronyms :
Rate of Mass transfer = RMT
mass transfer coefficient= MTC
interfacial area = ai
concentration gradient = dY
volume fraction of gas = α_g
sauter diameter = Ds
diffusivity = Df

Then the, **Rate of mass transfer** can be written as:

$$\text{RMT} = \text{MTC} * \text{ai} * \text{dY}$$

i.e., the rate of mass transfer is directly proportional to concentration gradient (dY) and interfacial area (ai) and all other terms which can influence these two aspects.Generally, the terms which influence the ai,dY are diffusion , convection and thermodynamic state of the fluid at the instant of observation.Since it is difficult to measure the indvidual effect of these terms on RMT, the net effect was measured using experiments and was defined as MTC. Excellent research was carried out by many researchers to find the value of this MTC in terms of measurable quantities like velocity, temperature and other fluid properties. Also, the results of their work was presented in terms of dimensional groups which are useful in further investigations like scale up.

The most useful dimesnional groups in mass transfer calculations are Sherwood number, Reynolds number , Schmidt number and Prandtl number, Lewis number.In our current study we will use one of the most familiar correlation of mass transfer given by Frossling (Details are presented in chapter 2).

Majority of the correlations are based upon the experiments where, only lumped parameters

can be measured.Measuring the quantities like interfacial area, dY across interface independently by experiment procedure is very difficult. Even if possible, the expiremintal setup will be expensive for an industrial scale reactor. In order to over come this limitation researchers started using numerical approaches for modeling these reactors and study the reactors performance at extreme conditions.

CFD modeling is one such numerical approach which plays a vital role in understanding the impact of various hydrodynamic parameters on overall efficiency of the multiphase systems.In order to use this technique in general we need to have commercial software which are again expensive. But thanks to many engineers across globe who initiated open source softwares which are avaialble at free of cost. One of the most famous CFD open source software is OpenFoam and it has many capabilities of CFD interest (user is encouraged to visit the OpenFoam website for further info).

reactingTwoPhaseEulerFoam one of the CFD modeling capabilities of OpenFoam, which helps in understanding the specie mass transfer between two phases and in this project we try to understand its source code in detail from the perspective of mass transfer modeling. The bubbleColumnEvaporatingReacting tutorial in the default tutorials of the openfoam tutorial list will be used for the purpose of demonstration of the code.Also, we will try to code and implement Higbie mass transfer model , which isn't available by default.

Chapter 2

Various classes involved in the reactingTwoPhaseEulerFoam solver and their significance

The source code of the solver is located at:

`$FOAM_APP/solvers/multiphase/reactingEulerFoam/reactingTwoPhaseEulerFoam/reactingTwoPhaseEulerFoam.C`

The classes involved in the code and their significance:

`#include"fvCFD.H`:- This file brings in the most fundamental tools for performing any finite volume calculation.Â It includes declaration of namespace Foam and controls for time.

`#include"twoPhaseSystem.H"`:- Class which solves the volume fraction equations for two phases

`#include"phaseCompressibleTurbulenceModel.H"`:- its a type definition for PhaseCompressibleTubulence template abstract class. PhaseCompressibleTurbulence is an abstract base class for multiphase compressible turbulence models

`#include"fixedFluxPressureFvPatchScalarField.H"`:- its a class used to calculate the pressure field near boundary conditions based on velocity specifications

`#include"pimpleControl.H"`:- this class refers to the PIMPLE alogorithm that is used for pressure-velocity coupling. The controls of this algorithm are specified by user in casedirectory/system/fvsolution file

`#include"localEulerDdtScheme.H"`:- class is used to use the local time step (LTS) approach for transient calculations. It is mentioned in casedirectory/system/fvschemes under ddtschemes option.

`#include"fvcSmooth.H"`:- class is used to smooth spread and sweep the values in order to improve the stability.

`#include"setRootCase.H"`:- it executes checkrootcase function which is used to verify whether the given case is setup is done properly or not. If any file is missing it will return Fatalerror and stops the execution of the program

`#include"createTime.H"`:- helps in reading the time control values from the controldict file in casedirectory/system folder

`#include"createMesh.H"`:- it helps in reading the mesh and adds a time stamp to it

`#include"createTimeControls.H"`:- initiates the functions which are useful in adjusting the times step during the runtime.

`#include "createRDeltaT.H"`:-calculates the timestep for each mesh cell if LTS scheme is enabled (if localEuler is selected as ddt scheme)

`#include"createFields.H"`:- creates the matrix for all variables U,p from initial conditions.Also initializes other dependent variables like phi (mass flux) and others important for calculations. It also creates an object `fluid` of two phase system class. Using this object we can easily call functions like

`correctTurbulence()`, `correctKinematics()` of twophasesystem class.Also the phases are defined as fluid.phase1,fluid.phase2. The details can be found in `createFields.H` file located in:

```
$FOAM_APP/solvers/multiphase/reactingEulerFoam/
reactingTwoPhaseEulerFoam/createFields.H
```

`#include "pUf/createDDtU.H"`:- it is createDDTU.H in pUf folder. It calculates the time derivate terms of phase1 and phase2 equations

`#include "readTimeControls.H"`:- reads the controls for time step given by setDeltat settimgs like adjustTimeStep,maxCo and maxDeltat

An important observation from developers documentation regarding face moemntum condition: Facemomentum formulation provides C-grid like pressure-flux staggering on an unstructured mesh which is hugely beneficial for Euler-Euler multiphase equations as it allows for all forces to be treated in a consistent manner on the cell-faces which provides better balance, stability and accuracy. However, to achieve face-force consistency the momentum transport terms must be interpolated to the faces reducing accuracy of this part of the system but this is offset by the increase in accuracy of the force-balance. Currently it is not clear if this face-based momentum equation formulation is preferable for all Euler-Euler simulations so I have included it on a switch to allow evaluation and comparison with the previous cell-based formulation.

`The facemomentum formulation can be activated by explicitly specifying in PIMPLE controls in casedirectory/system/fvsolution file as follows:`

```
PIMPLE
{
    nOuterCorrectors 3;
    nCorrectors      1;
    nNonOrthogonalCorrectors 0;
    faceMomentum     yes;
}
```

In general the default collocated approach is used by the code involving two or more phases.

`#include "YEqns.H":-` this is the header file responsible for species transport including mass transfer of species.

The species transport equation `YEqns.H` in detail:

The header file of `#includes "YEqns.H"` is located at:

```
$FOAM_APP/solvers/multiphase/reactingEulerFoam/reactingTwoPhaseEulerFoam/YEqns.H
```

The header file connects to a pointer where the mass transfer rates table will be defined for the given species. : `autoPtr<phaseSystem::massTransferTable>` The definition of this pointer can be found in the `InterfaceCompositionPhaseChangePhaseSystem.C` file located at:

```
$FOAM_APP/solvers/multiphase/reactingEulerFoam/phaseSystems/PhaseSystems/
InterfaceCompositionPhaseChangePhaseSystem/
InterfaceCompositionPhaseChangePhaseSystem.C
```

Using the code in this file the amount of mass transfer of the species/phase as a whole is calculated. The term referring to the mass transfer rate is given by **dmdt**. The definition of mass transfer involved in other phase systems like HeatandMassTransferphasesystem and Thermal-PhaseChangephasesystem is one and same like this.

The default available mass transfer models are : Frossling and Speherical. Their definition can be found at:

```
$FOAM_APP/solvers/multiphase/reactingEulerFoam/
interfacialCompositionModels/massTransferModels
```

The frossling equation is given by: `Sh= 2`+$0.552 * Re^{1/2} * Sc^{1/3}$

Even though we mention the Sc for every individual phase in phaseprperties file in casedirectory/constant folder, the Sc (Schmidt number) in OpenFoam is calculated from Prandtl Number(Pr) and Lewis number(Le) as: `Sc= Le`∗Pr

The Le is read from the phase properties file in case directory/constant folder and the Pr is calculated from phasePair properties read calculated for pair of inetracting phases located at:

`$FOAM_APP/solvers/multiphase/reactingEulerFoam/phaseSystems/phasePair/phasePair.C`

The way the mass transfer coefficient is derived from Sherwood number is bit tricky in OpenFOAM and is explained here:

Using the acronyms defined in Chapter1 , for a gas liquid two phase flow, the interfacial area can be defined as:

ai = $6 * \alpha_g$/Ds

Substituting this definition of ai in the definition of RMT, we get
RMT = MTC∗6∗α_g∗dY/Ds

The Sherwood number by definition is:
Sh= MTC∗Ds/Df
⇒ MTC=Sh∗Df/Ds

Substituting the MTC definition in RMT equations gives:
RMT = Sh ∗ Df ∗ 6 ∗ α_g ∗ dY/Ds^2

Since the diffusivity and concentration gradient are calculated as part of solution, all the remaining terms are combined and calculated in the MTC term andwe have the return argument of MTC, K() function in the Frossling model as:

6 ∗ α_g∗Sh/Ds^2

in the code it is written as:

```
Foam::massTransferModels::Frossling::K() const
{
volScalarField Sh(scalar(2) + 0.552*sqrt(pair_.Re())*cbrt(Le_*pair_.Pr()));
return 6.0*pair_.dispersed()*Sh/sqr(pair_.dispersed().d());
}

where:
pair_.dispersed()= volume fraction of dispersed phase
pair_.dispersed().d() = Sauter diameter of the dispersed phase

Similiarly,the Spherical mass transfer coefficient is based upon the
assumption of laminar flow where Sh=10

And hence have return statement as:

Foam::massTransferModels::spehrical::K() const
{
return 60.0*pair_.dispersed()/sqr(pair_.dispersed().d());
}
```

Once the mass transfer terms are calculated as described above, the solution proceeds to source these **dmdt** term in the place ***massTransfer[Y1[i].name()]** of the equation and proceed for solving the equation of the code:

```
Y1iEqn == *massTransfer[Y1[i].name()]+ fvOptions(alpha1, rho1, Y1[i])
```

The left hand side of this equation is a species fraction equation which is createddepending upon the phase model that has been selected. For example, if the phase model of gas is **reactingPhaseModel** then on reading the command the OpenFoam creates the equations for all individual species as described in:

```
$FOAM_APP/solvers/multiphase/reactingEulerFoam/phaseSystems/phaseModel/
MultiComponentPhaseModel/MultiComponentPhaseModel.C

template<class BasePhaseModel>
Foam::tmp<Foam::fvScalarMatrix>
Foam::MultiComponentPhaseModel<BasePhaseModel>::YiEqn
(
    volScalarField& Yi
)
{
    if
    (
        (inertIndex_ != -1)
     && (
            Yi.name()
         == IOobject::groupName
            (
                this->thermo_->composition().species()[inertIndex_],
                this->name()
            )
        )
    )
    {
        return tmp<fvScalarMatrix>();
    }

    const volScalarField& alpha = *this;
    const surfaceScalarField& alphaRhoPhi = this->alphaRhoPhi();
    const volScalarField& rho = this->rho();

    return
    (
        fvm::ddt(alpha, rho, Yi)
      + fvm::div(alphaRhoPhi, Yi, "div(" + alphaRhoPhi.name() + ",Yi)")
      - fvm::Sp(this->continuityError(), Yi)

      - fvm::laplacian
        (
            fvc::interpolate(alpha)
           *fvc::interpolate(this->turbulence().nut()*rho/Sc_),
            Yi
```

```
        )
     ==
        this->R(Yi)

      + fvc::ddt(residualAlpha_*rho, Yi)
      - fvm::ddt(residualAlpha_*rho, Yi)
    );
}
```

The term R(Yi) is the source due to reaction term. If the gas is reacting (as we decided the gas as reacting gas) then the term R(Yi) is derived from the code:

```
$FOAM_APP/solvers/multiphase/reactingEulerFoam/phaseSystems/
phaseModel/ReactingPhaseModel/ReactingEulerFoam.C

template<class BasePhaseModel, class ReactionType>
Foam::tmp<Foam::fvScalarMatrix>
Foam::ReactingPhaseModel<BasePhaseModel, ReactionType>::R
(
    volScalarField& Yi
) const
{
    return reaction_->R(Yi);
}
```

where the `"reaction_"` term is a table of species,their reaction,thermo information retireved using `"foamChemistryReader"` key word. The source term R(Yi) is calculated using combustion models available for "rhoCombustionModel" setting through combustion properties. They are called upon when the fluid model (phase model) is "ReactingPhaseModel" as defined in:

`$FOAM_APP/solvers/multiphase/reactingEulerFoam/phaseSystems/
phaseModel/phaseModel/phaseModels.C`

The combustion model intern refers to the Chemistry model setting which will be described in a seperate chemistry file corresponding to the phase. The chemistry models will generate the source terms R(Yi) and feed back to the species code. The details of the various templates available for chemistry models and corresponding solvers can be found at:

`$FOAM_SRC/thermophysicalModels/chemistryModel/chemistryModel/
chemistryModel/chemistryModel.H`

Chapter 3

bubbleColumnEvaporatingReacting Tutorial

3.1 Introduction

This tutorial describes how to pre-process, run and post-process a Water Gas Shift Reaction (WGSR)taking place in a 3D bubble column reactor.The reacting fluids were multicomponent gas (AIR,CO) and pure liquid water.The products formed were CO2 and H2 along with water vapor.The flow of phases were modeled with Eulerian-Eulerian Solver. ALso,Seperate species mass fraction transport equations were solved for all the individual species involved:CO,H2O,AIR,CO2,H2. The tutorial helps in understanding the modeling of multiphase reactive fluids involving reaction, heat and mass transfer between phases.

The geometry consists of a 3D block with a 0.15x0.1 m^2 base and a length of 1m (figure 3.1). The reactor is filled with water till height of 0.5m and assumed to have 0.1% dissolved air. Air enters from the bottom inlet at 400K. The system is initally assumed to be at 400K. The overall reaction involved is:

$$0.93CO + 0.24H2O \rightarrow 0.69CO2 + H2 \tag{3.1}$$

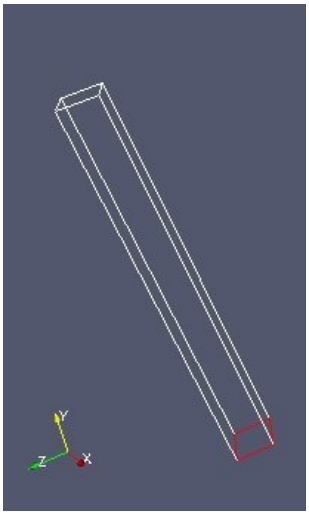

Figure 3.1: Geometry of the bubbleColumnEvaporatingReacting tutorial case.

3.2 Pre-processing

This section covers the necessary setup needed to get the bubbleColumnEvaporatingReacting case run with reaction,heat and mass transfer. The tutorial also covers a brief introduction to modeling of multiphase reacting flows in OpenFoam with detailed emphasis on modeling mass transfer.

3.2.1 Getting started

Copy the bubbleColumnEvaporatingReacting tutorial to the run directory and open the tutorial directory.

```
cp -r  $FOAM_TUTORIALS/multiphase/reactingTwoPhaseEulerFoam/RAS/
bubbleColumnEvaporatingReacting $FOAM_RUN

cd $FOAM_RUN/bubbleColumnEvaporatingReacting
```

The file structure of the bubbleColumnEvaporatingReacting case is similar to other OpenFOAM tutorials where the case directory has a `/0`, `/constant` and `/system` directory. 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`.

3.2.2 Boundary and initial conditions

The boundary conditions for the bubbleColumnEvaporatingReacting tutorial are very simple. All walls are modeled as adiabatic.The multicomponent gas enetrs from the bottom of the column at a speed of 0.1m/s. The AIR mass fraction in the inlet gas is 0.9 and that of CO is 0.1. The presence of water initially upto a height of 0.5m is declared by using setFieldsDict (given in system folder) functionionality as given below:

```
defaultFieldValues
(
    volScalarFieldValue alpha.gas 1
    volScalarFieldValue alpha.liquid 0
);

regions
(
    boxToCell
    {
        box (0 0 0) (0.15 0.501 0.1);
        fieldValues
        (
            volScalarFieldValue alpha.gas 0.01
            volScalarFieldValue alpha.liquid 0.99
        );
    }
);
```

Note:-For undesrtanding the usage of setFieldsDict the user is recommended to look into the OpenFoam basic damBreak tutorial

The initial specie mass fraction for all the species over the entire geometry field distribution can be found in the 0 folder.

3.2.3 Physical properties

In the `cd $FOAM_RUN/bubbleColumnEvaporatingReacting/constant` directory,the properties files for chemistry,environmental, combustion, reaction, combustion, RAS and thermophysical. All the properties are thoroughly described in the OpenFOAM user guide and are therefore not described here. The properties files are summarized in table 3.1.

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	models used for combustion are specified. In this tutorial PaSR(Partially Stirred Reactor Combustion Model) is used
thermophysicalProperties	Specify the mixture type, properties and which gas phase reaction scheme to use. Also inert species present (if any) are declared
phaseProperties	models describing the interaction between the phases are specified here

Table 3.1: Properties files and general content for the bubbleColumnEvaporatingReacting tutorial

Some important observations on the the script in phaseProperties file in tutorial folder,`cd $FOAM_RUN/bubbleColumnEvap`
:

```
// Phases that will interact. By default the phases are read in same order given here.
That is phase1= dispersed phase= gas,phase2=continuousphase=liquid.

phases (gas liquid);


//the phase is a reacting phase and creates corresponding equations for all the
//components involved including their reactions and reads their initial value
gas
{
    type            reactingPhaseModel;
    diameterModel   isothermal;     //the change in the diameter of the gas due to reactions
    isothermalCoeffs
    {
        d0              3e-3;      // size of the diameter of the gas bubble.
           // It must be lesser than mesh cell size
        p0              1e5;       //initial pressure used as reference for
                                   //calculating the changes in diameter model
    }
    Sc              0.7;        // Schmidt number used to calculate mass diffusivity internally

    residualAlpha   1e-6;
}

//the liquid phase is a purephase and is initialized with
//all component mass fraction equating to zero

liquid
{
    type            purePhaseModel;
    diameterModel   constant;      //no composition change
```

```
    constantCoeffs
    {
        d               1e-4;
    }

    residualAlpha   1e-6;
}

//models which dictate the sudden jump in the value of gas mass fraction at interface
//due to gas solubility in liquid

interfaceComposition
(
    (gas in liquid)
    {
        type Saturated;      //interface model name
        species ( H2O );     //Species considered to be at Saturation (usually liquid)
        Le 1.0;

//Le=1.0emphasis MTC is approximately equals to HTC. hence deltaT =deltaY
//and deltaT is calculated using the difference between starurated pressure
//and local pressure

saturationPressure
        {
            type ArdenBuck;   //calculates the saturation pressure using ArdenBuck correlation
        }
    }
);



//models used to find the mass transfer coeffieint (MTC) for component transfer
//within the gas phase (from bulk to interface)

massTransfer.gas
(
    (gas in liquid)   //MTC to find the amount of mass transfered from gas to liquid
                      //when gas as dispersed phase
    {
        type spherical;
        Le 1.0;
//Lewis number used find Schmidt number which is used to find Sherwood number
//from which MTC is derived
    }

    (liquid in gas) //MTC to find the amount of mass transfered from liquid to gas
                  //when liquid is the dispersed phase
    {
        type Frossling;
        Le 1.0;
    }
);
```

```
massTransfer.liquid //mass transfer in liquid phase is negligible in this case
(
);
```

Note:- When calculating these interface terms for each phase, Ex:- "massTransfer.gas" we mentioned "(gas in liquid)" and "(liquid in gas)" and I mentioned that one of these terms are used depending upon the dispersed phase and continuous phase derived based on the phase fractions at each cell volume locally.

OpenFoam decides the the despersed phase and continuous phase locally (for each cell) by using "blending method" specifications given in the same phase properties file as below:

```
blending
{
   ........

    massTransfer
    {
        type            linear;
        minFullyContinuousAlpha.gas 1;
        minPartlyContinuousAlpha.gas 0;
        minFullyContinuousAlpha.liquid 1;
        minPartlyContinuousAlpha.liquid 0;
    }
}
```

Here the minFullyContinuousAlpha.gas means the minimum volume fraction of a phase to be considered as continuous phase and by default the other phase will become dispersed phase

Similiarly, the minPartlyContinuousAlpha.gas means the minimum volume fraction of the gas phase in a cell volume , for which it can be treated as partial phase(dispersed phase).

The calculations involved in blending method can be found in the file located at:

```
$FOAM_APP/solvers/multiphase/reactingEulerFoam/phaseSystems/BlendedInterfacialModel/
BlendedInterfacialModel.C
```

where in which for this file the fuction weightage values f1 and f2 are determined using blending methods. In our bubbleColumnEvaporatingReacting tutorial, we are using linear blending method.

The calculation of weightage values f1 and f2 for this linear method are described in:

```
$FOAM_APP/solvers/multiphase/reactingEulerFoam/phaseSystems/BlendedInterfacialModel/
blendingMethods/linear/liner.C
```

Bothe files are self descriptive and the user is encouraged to visit them for better understanding of the solver code.

The reaction in gaseous mixture is specified in the thermophysical proerties of the gas as following:

```
thermoType heRhoThermo<reactingMixture>;
foamChemistryFile "$FOAM_CASE/constant/reactions.gas";
foamChemistryThermoFile "$FOAM_CASE/constant/thermo.gas";
```

In this tutorial we will use the predefined `reactingMixture` together with the thermophysical model `heRhoThermo` which calculates enthalpy for reacting mixture. 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 `""$FOAM_CASE/constant/reactions.gas""` file and the thermo dynamic data base in the `"$FOAM_CASE/constant/thermo.gas"` file.

The reaction and the resulting species are declared in the `"reaction.gas"` file as species list: CO,CO2,H2,H2O,AIR. The thermal data for the species in `"thermo.gas"` is also mentioned in the same order. The reaction is a reversible reaction of Arhenius type.

The overall reaction is:

$$0.93CO + 0.24H2O \rightarrow 0.69CO2 + H2 \tag{3.2}$$

The entries behind the reaction in the **reaction.gas** file are Arrhenius coefficient that are used to calculate the chemical reaction rate. The type of the reaction is **reversibleArrheniusReaction** and the chemical reaction rate will be calculated according to the Arhenius equilibrium reaction coeff as given below and and as described in the introduction part.

$$k = AT^b \cdot exp\left(\frac{-E_a}{RT}\right) \tag{3.3}$$

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

With out going into great detail regarding thermodynamics it is possible to realize that the reaction is mildly endothermic and takes place at high temperatures. The thermodynamic properties associated with reaction are mentioned in **thermo**+ file in casedirectory/constant folder.The file structure is like:

- 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 3.4 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.

$$\frac{C_p}{R} = a_1 + a_2 \cdot T + a_3 \cdot T^2 + a_4 \cdot T^3 + a_5 \cdot T^4 \tag{3.4}$$

$$\frac{H}{RT} = a_1 + \frac{a_2 T}{2} + \frac{a_3 T^2}{3} + \frac{a_4 T^3}{4} + \frac{a_5 T^4}{5} + \frac{a_6}{T} \tag{3.5}$$

$$\frac{S}{R} = a_1 \ln T + a_2 T + \frac{a_3 T^2}{2} + \frac{a_4 T^3}{3} + \frac{a_5 T^4}{4} + a_7 \tag{3.6}$$

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 3.7 and 3.8

$$CST_i = \frac{\tau_{chem}}{\tau_{mix} + \tau_{chem}} \dot{\omega}_i \tag{3.7}$$

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

$$\tau_{mix} = C_{mix}\sqrt{\frac{\mu_{eff}}{\rho\epsilon}}. \tag{3.8}$$

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.

3.3 Running the code

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

```
chemistry               on;
```

and combustion `"active true"` in the `/constant/combustionProperties` file.

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

```
cd $FOAM_RUN/bubbleColumnEvaporatingReacting
blockMesh
reactingTwoPhaseEulerFoam
```

The solution is only 0.001 seconds long, however, due to the fast chemistry a minimum of 100 time steps are needed to resolve it.

3.4 Post-processing

The post-processing is quite simple and straightforward. As described in any standard tutorial the post-processing can be initiated by **paraFoam** command and the following results can be seen immidiately. After completion of the run the results are as shown below:

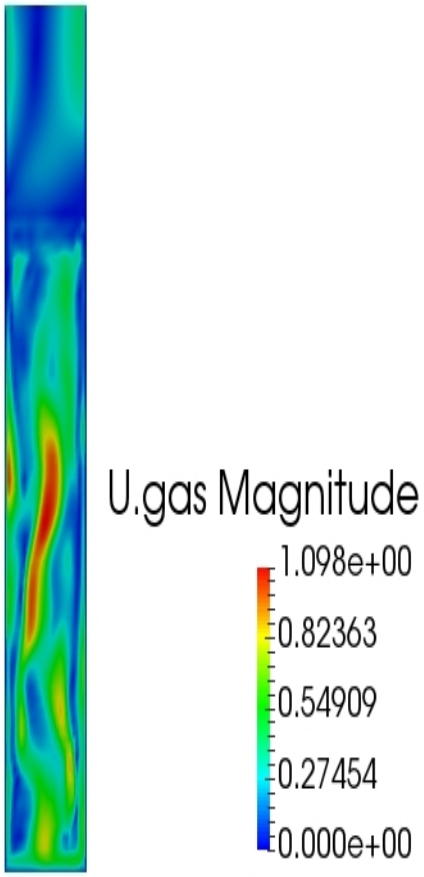

Figure 3.2: Velocity distribution of gas after five time steps

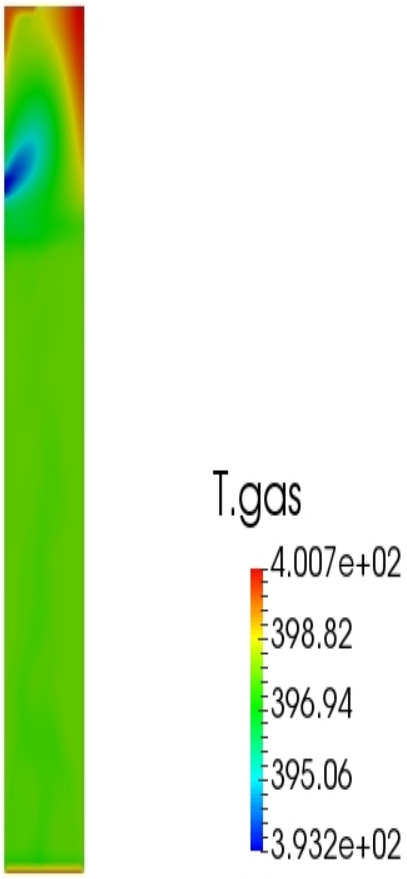

Figure 3.3: Temperature distribution of gas after five time steps

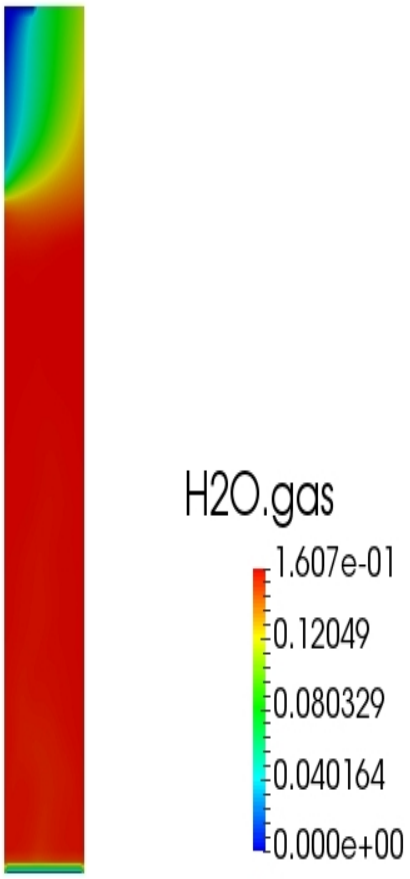

Figure 3.4: H2O gas(water vapor) distribution after five time steps

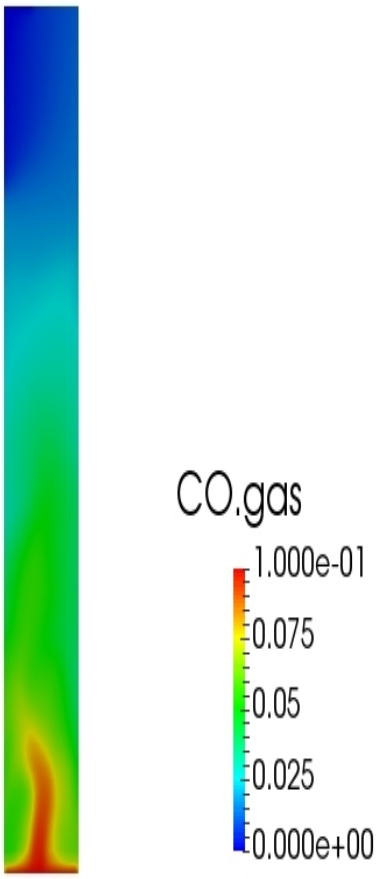

Figure 3.5: CO gas distribution after five time steps

Chapter 4

Implementing Higbie mass transfer model

The Higbie mass transfer equation is given by:

$$Sh = 1.13 * Re^{1/2} * Sc^{1/2}$$

Let us now try to implement and use this in our bubbleColumnEvaporatingReactingTutorial.

4.0.1 Copy the existing code

Run the following commands in terminal in order to copy the existing code:

```
OF4x
foam
cp -r --parents applications/solvers/multiphase/reactingEulerFoam $WM_PROJECT_USER_DIR
```

4.0.2 Clone existing Frossling model file and rename the clone as Higbie

```
cd $WM_PROJECT_USER_DIR/applications/solvers/multiphase/reactingEulerFoam/interfacialCompositionModels
cp -R Frossling Higbie/
cd Higbie/
mv Frossling.C Higbie.C
mv Frossling.H Higbie.H
sed -i s/Frossling/Higbie/g Higbie.C
sed -i s/Frossling/Higbie/g Higbie.H
```

4.0.3 Edit the Higbie.C file

```
cd $WM_PROJECT_USER_DIR/applications/solvers/multiphase/reactingEulerFoam/interfacialCompositionModels
gedit Higbie.C

////Change the member function as shown below///
// * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * * //

Foam::tmp<Foam::volScalarField>
Foam::massTransferModels::Higbie::K() const
{
    volScalarField Sh(1.13*sqrt(pair_.Re())*sqrt(Le_*pair_.Pr()));
```

```
    return 6.0*pair_.dispersed()*Sh/sqr(pair_.dispersed().d());
}

save and quit
```

4.0.4 Compile the Higbie.C file

```
cd $WM_PROJECT_USER_DIR/applications/solvers/multiphase/reactingEulerFoam/interfacialCompositionModels
sed -i s/FOAM_LIBBIN/FOAM_USER_LIBBIN/g Make/files
sed -i '4 i\massTransferModels/Higbie/Higbie.C' Make/files
wmakeLnIncludeAll
wmake
cd $WM_PROJECT_USER_DIR/applications/solvers/multiphase/reactingEulerFoam
wmakeLnIncludeAll
```

4.0.5 Modify reactingTwoPhaseEulerFoam code and create it's library

```
cd $WM_PROJECT_USER_DIR/applications/solvers/multiphase/reactingEulerFoam/reactingTwoPhaseEulerFoam
mv reactingTwoPhaseEulerFoam.C myreactingTwoPhaseEulerFoam.C
sed -i s/FOAM_APPBIN/FOAM_USER_APPBIN/g Make/files
sed -i s/reactingTwoPhaseEulerFoam/myreactingTwoPhaseEulerFoam/g Make/files
cd Make
gedit options
```

After `EXE_LIB` add the line as shown below in order to direct the wmake to look for the presence of any library file before searching in source code installation folders.

```
EXE_LIBS = \
    -L$(FOAM_USER_LIBBIN)\
```

After adding the above line the options file must look like:

```
EXE_INC = \
    -ItwoPhaseSystem/lnInclude \
    -I../phaseSystems/lnInclude \
    -I../interfacialModels/lnInclude \
    -I../interfacialCompositionModels/lnInclude \
    -ItwoPhaseCompressibleTurbulenceModels/lnInclude \
    -I$(LIB_SRC)/thermophysicalModels/basic/lnInclude \
    -I$(LIB_SRC)/transportModels/compressible/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/turbulenceModels/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/compressible/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/phaseCompressible/lnInclude \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude \
    -I$(LIB_SRC)/sampling/lnInclude

EXE_LIBS = \
    -L$(FOAM_USER_LIBBIN)\
    -lreactingPhaseSystem \
    -lreactingTwoPhaseSystem \
    -lreactingEulerianInterfacialModels \
```

```
    -lreactingEulerianInterfacialCompositionModels \
    -ltwoPhaseReactingTurbulenceModels \
    -lfiniteVolume \
    -lfvOptions \
    -lmeshTools \
    -lsampling
```

close the options file and type *wmake*

4.0.6 Using Higbie mass transfer code in tutorial

Open the tutorial directory:

```
cd $FOAM_RUN/bubbleColumnEvaporatingReacting
```

Open the controlDict file in tutorial directory by typing the following command:

```
gedit system/controlDict
```

next to application command type myreactingTwoPhaseEulerFoam as below:

```
application     myreactingTwoPhaseEulerFoam;
```

save and close the controlDict file.
Open the phaseProperties file in tutorial directory by typing the following command:

```
gedit constant/phaseProperties
```

In mass transfer for gas options delete Frossling and write Higbie. After writing it should look as follows:

```
massTransfer.gas
(
    (gas in liquid)
    {
        type spherical;
        Le 1.0;
    }

    (liquid in gas)
    {
        type Higbie;
        Le 1.0;
    }
);
```

4.0.7 Running the tutorial

Run the tutorial by typing *myreactingTwoPhaseEulerFoam* and post-process as described in previous chapter

Chapter 5

Study questions

- What are default mass transfer models available?
- The mass transfer models return the mass transfer coefficient or something else?
- Which equation in reactingTwoPhaseEulerFoam.C represent species transport?
- The Y values in species transport equation represent mole fraction or mass fraction?