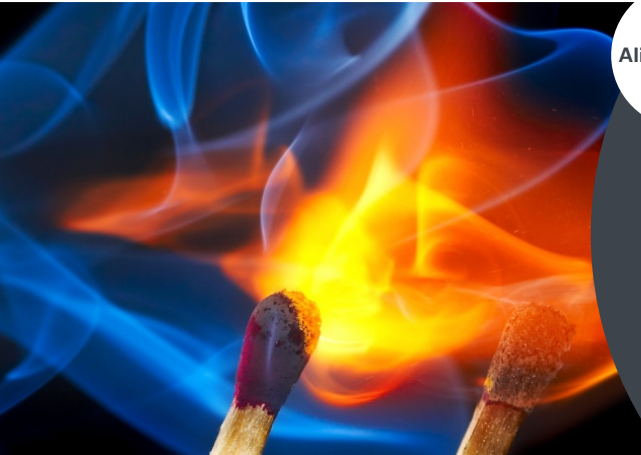
Universität Stuttgart



Ali Shamooni

fcci2022-workshop:

Pre-/post-processing with Cantera and OpenSMOKE++

Outline

Outline

- Pre-processing with OpenSMOKE++ was covered by Dr. Cuoci in ICS2019. You can find the content here
- Pre-processing with Cantera
  - Conversion of chemkin files
  - 0D calculations
    - Equilibrium calculations
    - Creating mixture with different equivalence ratios (Φ)
    - Stoichiometric mixture fraction calculations
    - Adiabatic flame temperature calculations
  - 1D calculations
    - Premixed flat flame simulations
    - counter-flow diffusion flame
- Post-processing with OpenSMOKE++
  - Building a solver to solve mixture fraction equation
  - Evaluation of species formation rates
  - Evaluation of individual reaction rates

Prerequisites

Notes

- x This course is not C++ course!
- x This course is not python course!
- x This course is not Linux course!
- x This course is not OpenFOAM course!
- x This code is not combustion course!

Prerequisites

- Basic knowledge of Linux shell commands
- Basic knowledge of python language
- Basic knowledge of C++ language
- Basic knowledge of OpenFOAM, case setup and coding
- Advanced (MSc.) knowledge of theory of combustion

Starting Point

Installation dependencies

Notes

All steps are tested on Linux: Ubuntu 18.04

Notes

All files are available on Github

OpenSMOKE and OpenFOAM

- Gcc 8.4.0 (see installationScripts/compile-README-gcc8.4.0.txt)
- boost 1.70.0 (see installationScripts/compileBoost.sh)
- OpenFoam (esi) version 2106 (see installationScripts/compileOpenFOAM.sh)
- External required libraries including OpenSMOKE++ are in **externalLibs** folder

Cantera

- Anaconda3 (see installationScripts/conda_cantera_python.md)
- Cantera (see installationScripts/conda_cantera_python.md)

Define repo paths

```
echo "export fcci2022=${HOME}/projects/fcci2022" >> ${HOME}/.bashrc
echo "export fcci2022_postostools=${HOME}/projects/\
fcci2022/postProcessing/openSMOKE/tools" >> ${HOME}/.bashrc
echo "export fcci2022_postostuts=${HOME}/projects/\
fcci2022/postProcessing/openSMOKE/tutorials" >> ${HOME}/.bashrc
echo "export fcci2022_postoscases=${HOME}/projects/\
fcci2022/postProcessing/openSMOKE/cases" >> ${HOME}/.bashrc
```

Pre-processing with Cantera

3

Pre-processing with Cantera

Let's continue in Jupyter lab! (preProcessing/cantera/**preProcessingCantera.ipynb**)

```
# go to the repo
cd $fcci2022/preProcessing/cantera
# launch jupyter! and open preProcessingCantera.ipynb
jupyter lab
```

Post-processing with OpenSMOKE++

4

Build a solver to create database

Notes

All steps are in individual **tutorials** folder

```
# set your openfoam version
ofv2106
# copy original reactingFoam solver into your local directory
# (here $fcci2022_postostuts)
cp -r $FOAM_APP/solvers/combustion/reactingFoam/ $fcci2022_postostuts
cd $fcci2022_postostuts

mv reactingFoam/ ZReactingFoam # rename the folder
cd ZReactingFoam/ # go into the folder
mv reactingFoam.C ZReactingFoam.C # rename the main solver file
# change all the occurunce of reactingFoam to ZReactingFoam
find ./* -type f -print0 | xargs -0 sed -i's #reactingFoam#ZReactingFoam#g'
sed -i s/FOAM_APPBIN/FOAM_USER_APPBIN/g Make/files
```

Let's continue in atom!

Build a solver to create database

```
# duplicate createFields.H --> createFields_Z.H
# add Z field (must be read)
# delete the rest
# add createFields_Z.H at the bottom of createFields.H
# duplicate YEqn.H --> ZEqn.H
# add Z equation to be solved
# delete the rest!
# add mixture fraction equation in the main function (ZReactingFoam.C)
# compile your new solver
ofv2106 # activate your OpenFOAM version
cd $fcci2022_postostuts/ZReactingFoam
. mybashrc # source extra paths
wclean # first clean!
wmake # then compile!
```

Build a solver to create database

Files that were modified/added:

- createFields_Z.H
- createFields.H
- ZEqn.H
- ZReactingFoam.C
- Make/files

Build a solver to create database

Case setup modifications:

- you need schemes for divergence of Z — > div(phi,Z) Gauss limitedLinear01 1;
- you need a solver for Z — > |epsilon|Z ...
- you need Z in 0 folder —> let's use duplicate Ydefault but we need proper BC (see preProcessing)
  - Z_pilot = 0.04207
  - Z_jet = 0.15551
  - Z_air = 0.0
- you may also want to change the setFields if you want to be precise!
- rememebr to add Z into the 0.org folder
- !!!! MAKE SURE YOU CHANGE THE Allrun FILE (see controlDict) !!!!

Build OS++ kinetics format

Notes

The code is in **kinetics** folder

```
# set your openfoam version
ofv2106
# go to source folder
cd $fcci2022/kinetics\
/openSMOKEppCHEMKINPreProcessor
# set your env.
. mybashrc
# fitst clean
wclean
# then make!
wmake

# go to kinetics folder
cd ../gri30
# run preprocessor
openSMOKEppCHEMKINPreProcessor --input input.dic
```

input.dic file:

```
#Dictionary CHEMKIN_PreProcessor
#{
#^^I@Thermodynamics chemkin/thermo30.dat;
#^^I//@Transport^^Ih2_v1a_tran.txt;
#^^I@Kinetics ^^Ichemkin/grimech30.dat;
#^^I@Output^^I^^IopenSMOKE-gri30;
#}
```

Build a code to generate species source terms

Math behind the code!
Consider a chemical kinetics mechanism with N_s species and N_r reactions. In the short format it is written like:

$$\sum_{k=1}^{N_s} \nu'_{k,j} \Phi_k \rightleftharpoons \sum_{k=1}^{N_s} \nu''_{k,j} \Phi_k \quad j = 1, ..., N_r, \tag{1}$$

where $\nu'_{k,j}$ and $\nu''_{k,j}$ are the molar stoichiometric coefficients of species k in forward and backward reactions, receptively and Φ_k the species k symbol. The stoichiometric coefficients should satisfy Eq. 2 relations to enforce the mass conservation:

$$\sum_{k=1}^{N_s} \nu_{k,j} W_k = 0 \quad j = 1, ..., N_r, \tag{2}$$

with W_k the molecular weight of species k and:

$$\nu_{k,j} = \nu''_{k,j} - \nu'_{k,j} \quad j = 1, ..., N_r. \tag{3}$$

Build a code to generate species source terms

The law of mass action states that the reaction rate is proportional to the product of concentration of reactants. For reversible reactions like what is introduced in Eq. 1, the **net** reaction rate of reactions j ($\dot{r}_j$) can be computed as:

$$\dot{r}_j \equiv k_{f,j} \prod_{k=1}^{N_s} \left(\frac{\rho Y_k}{W_k} \right)^{\nu'_{k,j}} - k_{b,j} \prod_{k=1}^{N_s} \left(\frac{\rho Y_k}{W_k} \right)^{\nu''_{k,j}}, \tag{4}$$

where $k_{f,j}$ and $k_{b,j}$ are forward and backward Arrhenius rates of reactions j, respectively. These rates depend on the temperature and are computed from the Arrhenius law and using equilibrium constants:

$$k_{f,j} \equiv A_{f,j} T^{\beta_j} \exp\left(-\frac{E_{f,j}}{RT} \right), \tag{5a}$$

$$k_{b,j} \equiv \frac{k_{r,j}}{\left(\frac{p_a}{RT} \right)^{\sum_{k=1}^{N_s} \nu_{k,j}} \exp\left(\frac{\Delta \hat{S}^0_j}{R} - \frac{\Delta \hat{H}^0_j}{RT} \right)}, \tag{5b}$$

where $E_{f,j}$ is the forward activation energies of reactions j, $p_a = 1$ bar and $\Delta \hat{H}^0_j$ and $\Delta \hat{S}^0_j$ refer to the enthalpy and the entropy changes occurring when passing from reactants to products.
Finally, thenet rate of the production of species,$\dot{\omega}_k$, is computed from:

$$\dot{\omega}_k \equiv \sum_{j=1}^{N_r} \left(\nu_{k,j} \dot{r}_j \right) \quad j = 1, ..., N_s \tag{6}$$

Build a code to generate species source terms

$$\dot{\omega}_k \equiv \sum_{j=1}^{N_r} \left(\nu_{k,j}\dot{r}_j\right) \quad j = 1, ..., N_s$$
$$\nu_{k,j} = \nu''_{k,j} - \nu'_{k,j} \quad j = 1, ..., N_r.$$

$$\dot{\omega}_k^+ \equiv \sum_{j=1}^{N_r} \left(\nu''_{k,j}\dot{r}_j^f - \nu'_{k,j}\dot{r}_j^b\right) \quad | \quad \dot{r}_j^f \equiv k_{f,j} \prod_{k=1}^{N_s} \left(\frac{\rho Y_k}{W_k}\right)^{\nu'_{k,j}}, \quad k = 1, ..., N_s, \quad j = 1, ..., N_r \tag{7}$$

$$\dot{\omega}_k^- \equiv \sum_{j=1}^{N_r} \left(\nu'_{k,j}\dot{r}_j^b - \nu''_{k,j}\dot{r}_j^f\right) \quad | \quad \dot{r}_j^b \equiv k_{b,j} \prod_{k=1}^{N_s} \left(\frac{\rho Y_k}{W_k}\right)^{\nu''_{k,j}}, \quad k = 1, ..., N_s, \quad j = 1, ..., N_r \tag{8}$$

$$\dot{\omega}_k \equiv \dot{\omega}_k^+ - \dot{\omega}_k^- \quad k = 1, ..., N_s \tag{9}$$

Build a code to generate species source terms

Notes

All steps are in **tutorials** folder

```
# set your openfoam version
ofv2106
# copy the native reactingFoam solver into the local tools directory
cp -r $FOAM_APP/solvers/combustion/reactingFoam/ $fcci2022_postostuts
# go to the destination
cd $fcci2022_postostuts
# rename the folder
mv reactingFoam/ osppt_omegaDot # rename the folder
cd osppt_omegaDot
rm -rf rhoReactingBuoyantFoam rhoReactingFoam # remove other solvers
mv reactingFoam.C osppt_omegaDot.C # rename the main solver
# change all the occurunce of reactingFoam to osppt_omegaDot
find ./* -type f -print0 | xargs -0 sed -i 's/reactingFoam/osppt_omegaDot/g'
# force OF to put your new solver in your home
sed -i s/FOAM_APPBIN/FOAM_USER_APPBIN/g Make/files
```

Let's continue in atom!

Build a code to generate species source terms

```
# open the folder with atom
cd ..
atom osppt_omegaDot
# comment out (or delete) line 48..118
# copy the contents of $fcci2022_postostools/osppt_omegaDot/osppt_omegaDot.C
# into the local directory
cp $fcci2022_postostools/osppt_omegaDot/osppt_omegaDot.C \
 $fcci2022_postostutsosppt_omegaDot/
# Lets have a look at osppt_omegaDot/osppt_omegaDot.C
# copy $fcci2022_postostools/osppt_omegaDot/readOptions.H into the local directory
cp $fcci2022_postostools/osppt_omegaDot/readOptions.H \
 $fcci2022_postostutsosppt_omegaDot/
# It is currently empty and will be used for other tools
# copy $fcci2022_postostools/osppt_omegaDot/getAndWriteRR.H into the local directory
cp $fcci2022_postostools/osppt_omegaDot/getAndWriteRR.H \
 $fcci2022_postostutsosppt_omegaDot/
# Lets have a look at osppt_omegaDot/getAndWriteRR.H
```

Build a code to generate species source terms

```
# copy $fcci2022_postostools/osppt_omegaDot/createOpenSMOKEFieldsGlobal.H
# into the local directory
cp $fcci2022_postostools/osppt_omegaDot/createOpenSMOKEFieldsGlobal.H\
$fcci2022_postostutsosppt_omegaDot/
# let's have a look at osppt_omegaDot/createOpenSMOKEFieldsGlobal.H
# copy $fcci2022_postostools/osppt_omegaDot/createOpenSMOKEFields_omegaDot.H
# into the local directory
cp $fcci2022_postostools/osppt_omegaDot/createOpenSMOKEFields_omegaDot.H\
  $fcci2022_postostutsosppt_omegaDot/
# let's have a look at osppt_omegaDot/createOpenSMOKEFields_omegaDot.H
# copy $fcci2022_postostools/osppt_omegaDot/openSMOKE_headers.H
# into the local directory
cp $fcci2022_postostools/osppt_omegaDot/openSMOKE_headers.H\
  $fcci2022_postostutsosppt_omegaDot/
# let's have a look at osppt_omegaDot/openSMOKE_headers.H
# add the header (#include "openSMOKE_headers.H") on top of
# your solver permeable (see line 32)
# let's have a look at osppt_omegaDot/osppt_omegaDot.C
```

Build a code to generate species source terms

```
# copy $fcci2022_postostools/osppt_omegaDot/createFields.H
# into the local directory
cp $fcci2022_postostools/osppt_omegaDot/createFields.H\
  $fcci2022_postostutsosppt_omegaDot/
# Lets have a look at osppt_omegaDot/createFields.H

# Now we need to tell the solver where external libraries (and headers are)
# copy $fcci2022_postostools/osppt_omegaDot/Make/options.H
# into the local Make directory
cp $fcci2022_postostools/osppt_omegaDot/Make/options.H\
  $fcci2022_postostutsosppt_omegaDot/Make/
# Lets have a look at osppt_omegaDot/Make/options.H
```

Build a code to generate species source terms

```
# to compile:
ofv2106
cd $fcci2022_postostuts/osppt_omegaDot
. mybashrc
wclean
wmake
```

Build a code to generate individual reaction rates

Notes

All steps are in **tutorials** folder

```
#set your openfoam version
ofv2106
mkdir $fcci2022_postostuts/osppt_rr
# copy contents of osppt_omegaDot into osppt_rr
cp -r $fcci2022_postostools/osppt_omegaDot/* $fcci2022_postostuts/osppt_rr/
# go to the destination
cd $fcci2022_postostuts/osppt_rr
mv osppt_omegaDot.C osppt_rr.C
find ./* -type f -print0 | xargs -0 sed -i 's/osppt_omegaDot/osppt_rr/g'
# open the folder with atom
cd ..
atom osppt_rr
```

Let's continue in atom!

Build a code to generate individual reaction rates

- we want to evaluate individual reaction rates at each cell
- we need to define an object and allocate space for it
- let's have a look at osppt_rr/createOpenSMOKEFields_omegaDot.H
- the object would be similar to omegaDotSpecies object but with different size (i.e. size=NR)
- let's create it in osppt_rr/createOpenSMOKEFields_omegaDot.H
- you can remove the rest of objects in case you want a clean code!
- now we have to compute the reaction rates and fill the object
- let's have a look at osppt_rr/getAndWriteRR.H

Build a code to generate individual reaction rates

```
# to compile:
ofv2106
cd $fcci2022_postostuts/osppt_rr
. mybashrc
wclean
wmake
```

Build a code to generate chemical time scale and non-dimensional numbers

Math behind the code!
In general:

$$\frac{dc_k}{dt} = \dot{\omega}_k \tag{10}$$

where $c_k = \rho \frac{Y_k}{W_k}$ is concentration of species k. In vector form:

$$\frac{d\mathbf{C}}{dt} = \dot{\mathbf{\Omega}} \tag{11}$$

The Jacobian is:

$$\mathbf{J} = \frac{\partial \dot{\mathbf{\Omega}}}{\partial C} \quad or \quad J_{i,j} = \frac{\partial \dot{\Omega}_i}{\partial C_j} \tag{12}$$

We calculate $\mathbf{J}$ numerically by Buzzi method [1], the analytical way can be found in [2].

[1] Buzzi, Manenti, Differential and Differential-Algebraic Systems for the Chemical Engineer, John Wiley & Sons, 2015.
[2] Niemeyer et al., pyJac: Analytical Jacobian generator for chemical kinetics, Computer Physics Communications, Vol. 215, 2017.

Build a code to generate chemical time scale and non-dimensional numbers

In general $\mathbf{J}$ can be decomposed as [1 (sec. 6)]:

$$\mathbf{J} = \mathbf{V}\mathbf{\Lambda}\mathbf{W} \tag{13}$$

where matrix $\mathbf{V}, \mathbf{W}$ are left (row vectors) and right eigenvectors (column vectors), respectively and matrix $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues ($\mathbf{\Lambda} = diag(\lambda_1, ..., \lambda_i)$.
The eigen values determines the time scales of the system:

$$\tau_i = 1/|Re(\lambda_i)| \tag{14}$$

The chemical time scale of a system is the maximum time scale i.e.

$$\tau_c = \tau_{i,max} = 1/|Re(|\lambda_{i,min}|)| \tag{15}$$

Notes

Conservative modes must be excluded!

We do eigen value decomposition by ▸ eigen library. See also ▸ here.

[1] Turányi, Tomlin, Analysis of Kinetic Reaction Mechanisms, Springer, 2014.

Build a code to generate chemical time scale and non-dimensional numbers

Kolmogorov time scale:

$$\tau_f = \tau_\eta = \left(\frac{\nu}{\epsilon}\right)^{1/2} \tag{16}$$

Integral time scale:

$$\tau_f = \tau_L = \left(\frac{k}{\epsilon}\right) \tag{17}$$

Damkohler number:

$$Da = \frac{\tau_f}{\tau_c} \quad | \quad Da_\eta = \frac{\tau_\eta}{\tau_c} \quad | \quad Da_L = \frac{\tau_L}{\tau_c} \tag{18}$$

Karlovitz number:

$$Ka = \frac{\tau_c}{\tau_\eta} = \frac{1}{Da_\eta} \tag{19}$$

Build a code to generate chemical time scale and non-dimensional numbers

Notes

All steps are in **tutorials** folder

```
#set your openfoam version
ofv2106
mkdir $fcci2022_postostuts/osppt_daKa
# copy osppt_omegaDot from the tools directory into tutorials
cp -r $fcci2022_postostools/osppt_omegaDot/* $fcci2022_postostuts/osppt_daKa/
# go to the destination
cd $fcci2022_postostuts/osppt_daKa
mv osppt_omegaDot.C osppt_daKa.C
find ./* -type f -print0 | xargs -0 sed -i 's/osppt_omegaDot/osppt_daKa/g'
# open the folder with atom
cd ..
atom osppt_daKa
```

Let's continue in atom!

Build a code to generate chemical time scale and non-dimensional numbers

- we want to evaluate Damkohler (Da) and Karlovitz (Ka) numbers at each cell
- we need to define an object and allocate space for them
- there are different Da definitions
- let's create them in osppt_daKa/createOpenSMOKEFields_daKa.H
- let's create it in osppt_rr/createOpenSMOKEFields_omegaDot.H

```
mv $fcci2022_postostuts/osppt_daKa/createOpenSMOKEFields_omegaDot.H\
 $fcci2022_postostuts/osppt_daKa/createOpenSMOKEFields_daKa.H
```

- the objects are volume scalar fields with the size equals to the mesh size
- we can use xSpecies object
- and create DaEta, DaL, chi, Ka, and tauChem
- remove the rest
- create an object of class "CharacteristicChemicalTimesType"
- you can read user defined options from case files and use them
- see $fcci2022_postostuts/readOptions.H

Build a code to generate chemical time scale and non-dimensional numbers

- in $fcci2022_postostuts/osppt_daKa/osppt_daKa.C
- replace createOpenSMOKEFields_omegaDot.H with createOpenSMOKEFields_daKa.H
- also add the header of the class (see openSMOKE_headers.H)
- now we have to compute the values and fill the objects

```
cp $fcci2022_postostools/osppt_daKa/getAndWriteDaKa.H \
$fcci2022_postostuts/osppt_daKa/

rm $fcci2022_postostuts/osppt_daKa/getAndWriteRR.H
```

- let's have a look at osppt_daKa/getAndWriteDaKa.H
- in $fcci2022_postostuts/osppt_daKa/osppt_daKa.C
- replace getAndWriteRR.H with getAndWriteDaKa.H

Build a code to generate chemical time scale and non-dimensional numbers

- we need to read Z, k and epsilon
- read them in osppt_daKa/createFields.H
- we need limiters by the user (maximum_Ka, minimum_Ka), (maximum_Da, minimum_Da)
- and also (threshold_chemical_time), (threshold_temperature_chemical_time)
- also we need a method for the evaluation of chemical times (characteristic_chemical_times_type)

```
# to compile:
ofv2106
cd $fcci2022_postostuts/osppt_omegaDot
. mybashrc
wclean
wmake
```

Universität Stuttgart

Ali Shamooni
Institut für Technische Verbrennung, Universität Stuttgart
Herdweg 51, 70174 Stuttgart, Germany

E-Mail: ali.shamooni@itv.uni-stuttgart.de
Phone: +49-711-685-65906