Implementation of soot model in aachenBomb tutorial

-Vignesh Pandian Muthuramalingam
-PhD student, Chalmers university of Technology

Dec-9-2015, CFD with OpenSource software

Overview

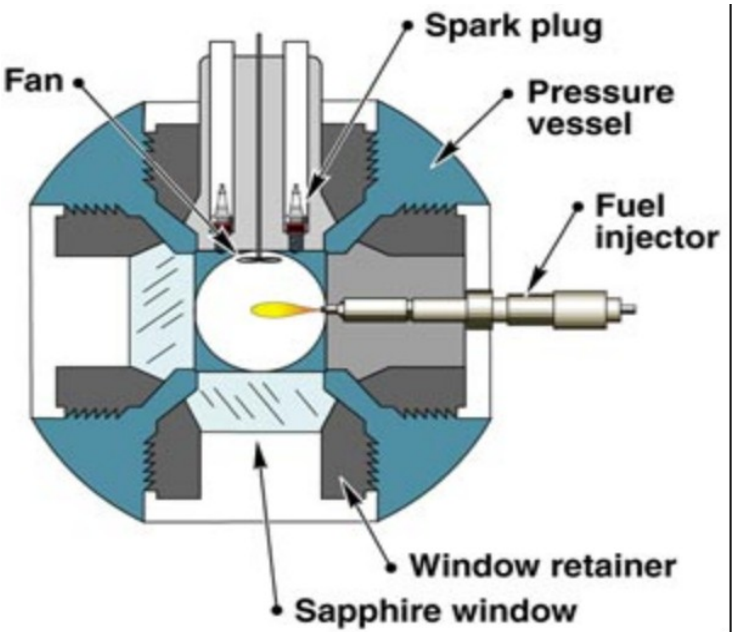
- Learning Objectives
- Introduction
- Computational domain and boundary conditions
- Input files for the aachenBomb case
- Introduction to soot modelling
- Modification of existing soot model
- Implementation
- Post processing and a look into sampleDict

Learning Objectives

- Introduction to combustion solvers in openFOAM 3.0.x and insight into the aachenbomb case of the sprayFoam solver
- learn how to set up the aachenbomb case and use the sprayFoam solver
- learn how to implement a soot model into the aachenBomb tutorial
- learn some basic post processing techniques in openFOAM paraView

Introduction to aachenBomb tutorial

- Computational model to simulate combustion stroke conditions in Internal Combustion Engines
- Computational domain identical to constant volume combustion chamber setup in ECN Sandia
- Boundary conditions in accordance with Engine Combustion Network

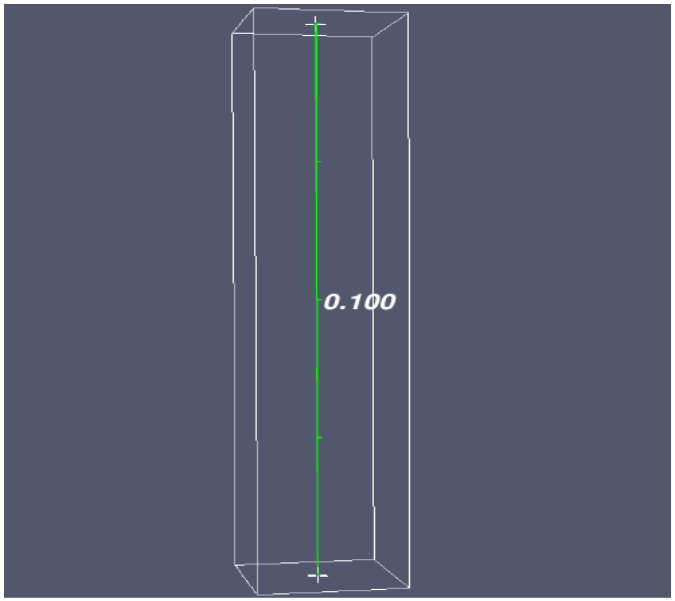


Combustion solvers in OpenFOAM 3.0.x

Solver	Functionality
sprayFoam	constant volume combustion solver for compressible flows involving spray parcels
sprayEngineFoam	moving piston engine solver for compressible flows involving spray parcels
coldEngineFoam	solver for cold flow(without combustion) for internal combustion engines
engineFoam	Solver for combustion in internal combustion engines

Geometry and boundary conditions

- Dimensions : 0.02x0.1x0.02m
- Fuel Injection in the middle directed in negative y



Boundary conditions

Ambient gas temperature	800K
Ambient gas pressure	50bar
Fuel	n-Heptane
Ambient O_2 concentration	23.4%
Ambient N_2 concentration	76.6%
Injection temperature	320K%
ϵ	90 (uniform internal field, walls:epsilonWallFunction)
k	1 (uniform internal field, walls:kqRWallFunction)

Input files for the aachenBomb tutorial

Input file	Functionality
sprayCloudProperties	contains inputs for the lagrangian spray model
thermophysicalProperties	contains inputs for the thermophysical models
radiationProperties	contains inputs for radiation, absorption- emission and soot models
ChemistryProperties	Specifies the chemistry solver and option to switch on/off the chemical reactions
combustionProperties	contains inputs for combustion model and option to switch on/off combustion%
turbulenceProperties	contains inputs for turbulence modeland option to switch on/off turbulence%

Introduction to soot modeling

- Basic combustion chemistry

$$(nuf)C7H16 + (nuOx)O_2 = (nuP1)CO_2 + (nuP2)H_2O + (nuSoot)soot$$

- Soot model based on existing OpenFOAM model
- Mixturefraction soot model
- Developed for fireFoam solver (methane combustion)
- Here it is modified for n-Heptane combustion

- Simple state model
- It is not solved separately
- Based on mass fraction of CO2

$$soot[cellI] = sootMax * (YCO2[cellI]/YCO2_{stoch})$$

- Drawback: Calculates soot at all times whenever CO2 is produced

Modification of soot model

- Soot produced only in fuel rich regions
- Satisfy the following condition:

$$YC7H16[cellI] > YC7H16_{stoch}$$

Implementation of the soot model

- Copying and renaming the soot model

```
cd $FOAM_RUN
cd ..
cp -r –parents $FOAM_SRC/thermophysicalModels/radiation/ .
```

Once the source folder is copied the next step is to rename the mixturefraction soot model.

```
cd thermophysicalModels/radiation/submodels/sootModel/
mv mixtureFractionSoot mymixtureFractionSoot

cd mymixtureFractionSoot
sed -i s/mixtureFractionSoot/mymixtureFractionSoot/g mymixtureFractionSoot.C
sed -i s/mixtureFractionSoot/mymixtureFractionSoot/g mymixtureFractionSoot.H
sed -i s/mixtureFractionSoot/mymixtureFractionSoot/g mixtureFractionSoots.C
```

Modify the code

```
// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * //

template<class ThermoType>
void Foam::radiation::mymixtureFractionSoot<ThermoType>::correct()
{


   const volScalarField& mapField =
      mesh_.lookupObject<volScalarField>(mappingFieldName_);

const volScalarField& mapField1 =
      mesh_.lookupObject<volScalarField>(mappingFieldName1_);

const volScalarField& mapField2 =
      mesh_.lookupObject<volScalarField>(mappingFieldName2_);



forAll (mapField2,i)
{

if( mapField1[i] >0.068) // Ystoch=0.068 for fuel n-Heptane, calculated from single step
combustion of n-Heptane in air (76.6% N2 and 23.4% O2 by weight)

soot_[i] = sootMax_*(mapField[i]/mapFieldMax_);
else
soot_[i] = 0;
}




}

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

Create IO Object to output calculated soot value

```
// * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * //

template<class ThermoType>
Foam::radiation::mymixtureFractionSoot<ThermoType>::mymixtureFractionSoot
(
    const dictionary& dict,
    const fvMesh& mesh,
    const word& modelType
)
:
    sootModel(dict, mesh, modelType),
    soot_
    (
        IOobject
        (
            "soot",
            mesh_.time().timeName(),
            mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_
    ),
    coeffsDict_(dict.subOrEmptyDict(modelType + "Coeffs")),
    nuSoot_(readScalar(coeffsDict_.lookup("nuSoot"))),
    Wsoot_(readScalar(coeffsDict_.lookup("Wsoot"))),
    sootMax_(-1),
    mappingFieldName_
    (
        coeffsDict_.lookupOrDefault<word>("mappingFieldName", "none")

    ),
```

- Compile the soot model contained in radiation model library

Copying and renaming the sprayFoam solver

- The sprayFoam solver has to include the modified radiation library

```
cp -r –parents applications/solvers/lagrangian/sprayFoam/ .
mv sprayFoam mysprayFoam
```

- Change Make/files and Make/options
- Compile the solver

Running the aachenBomb case

- Add radiationProperties file from

$FOAM_TUTORIALS/combustion/fireFoam/les/smallPoolFire2D/constant/

- This file includes the input for soot model

```
sootModel mymixtureFractionSoot<gasHThermoPhysics>;

mymixtureFractionSootCoeffs
{

   nuSoot          0.055;
   Wsoot           12;
}
```

- Also add soot file in the 0/ folder for
- This is required by the soot model as initial conditions (specified in IO object)
- The case is now ready to be run

Post processing

- Tracking lagrangian particles using glyph

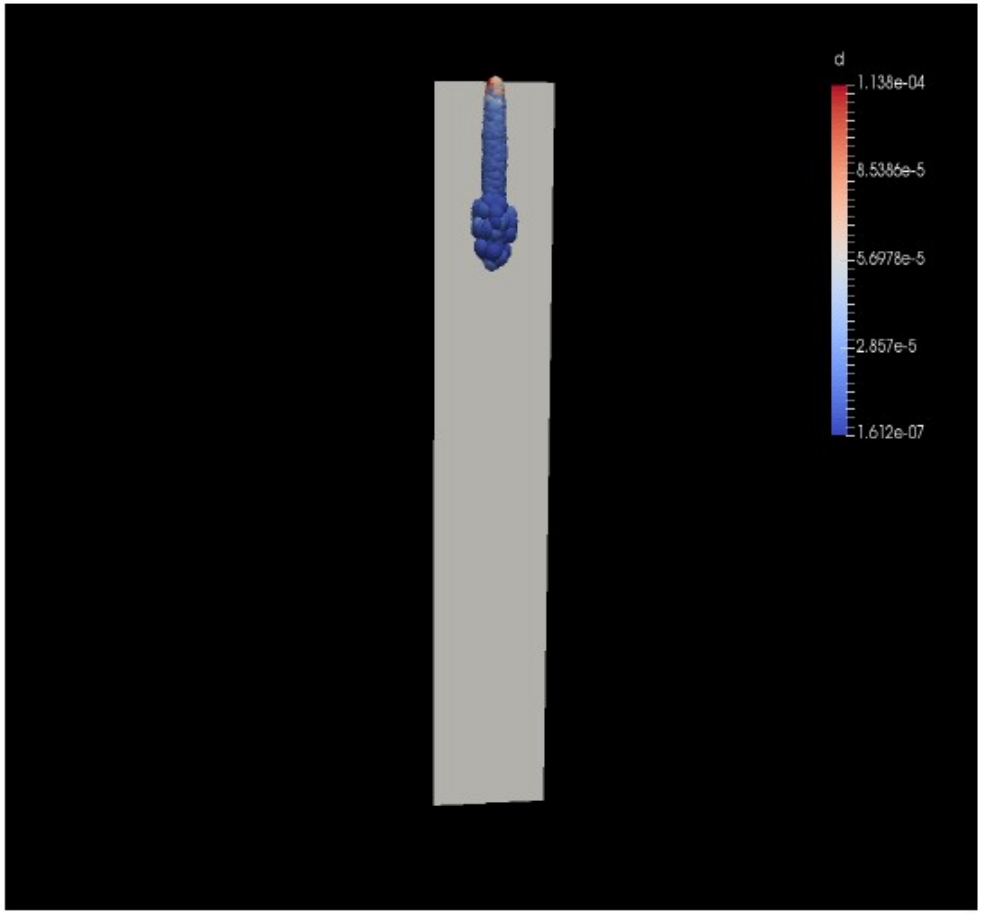
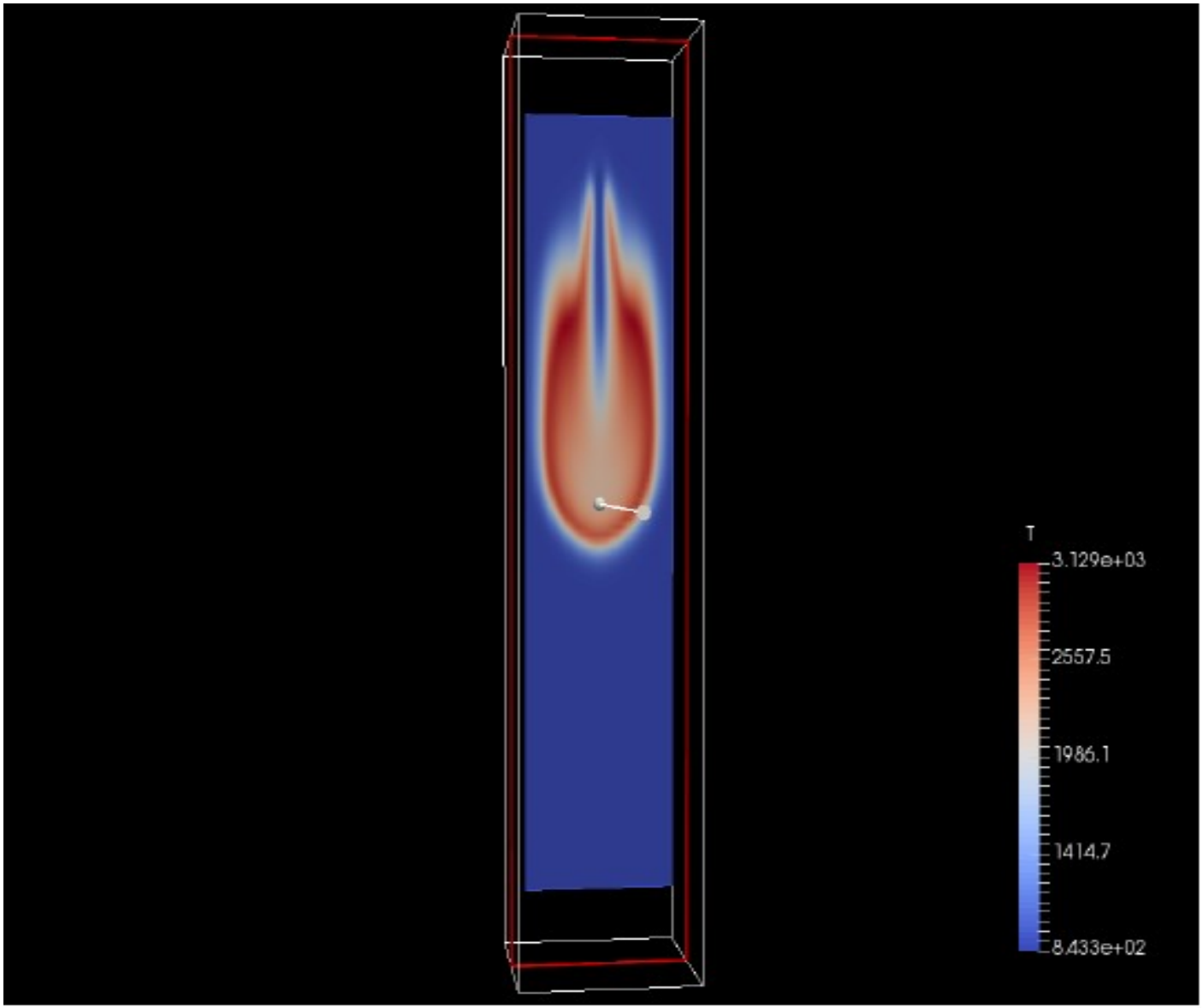


Figure 2.7: glyph of fuel droplets created at t=0.35ms

Slice of Temperature after combustion has started

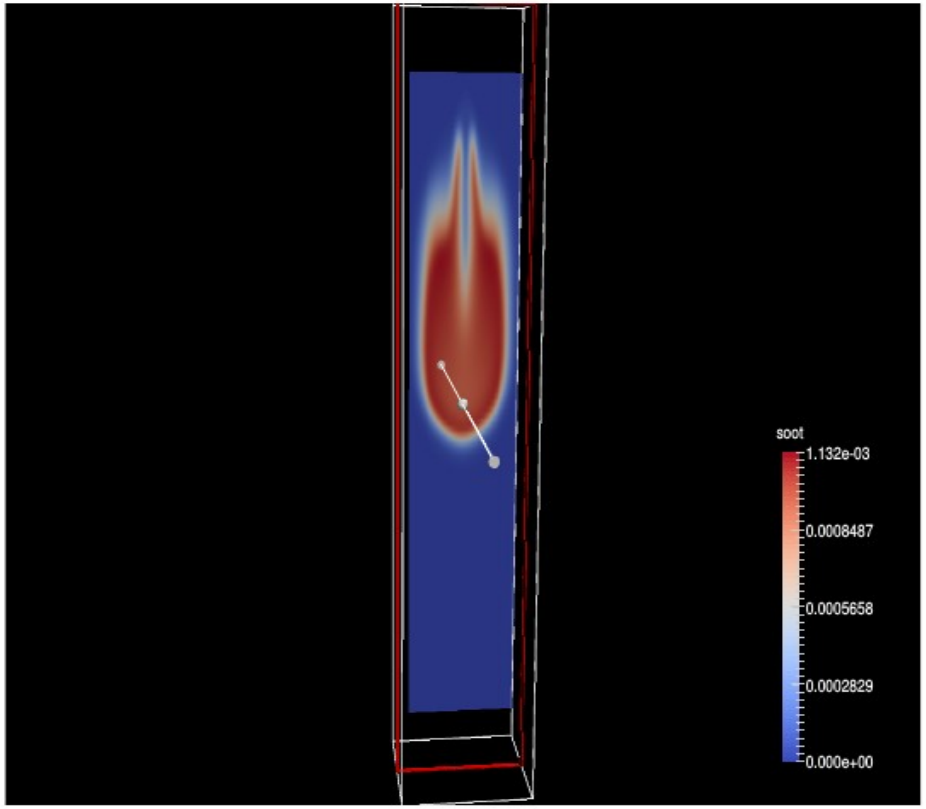


Time, t=1.4ms

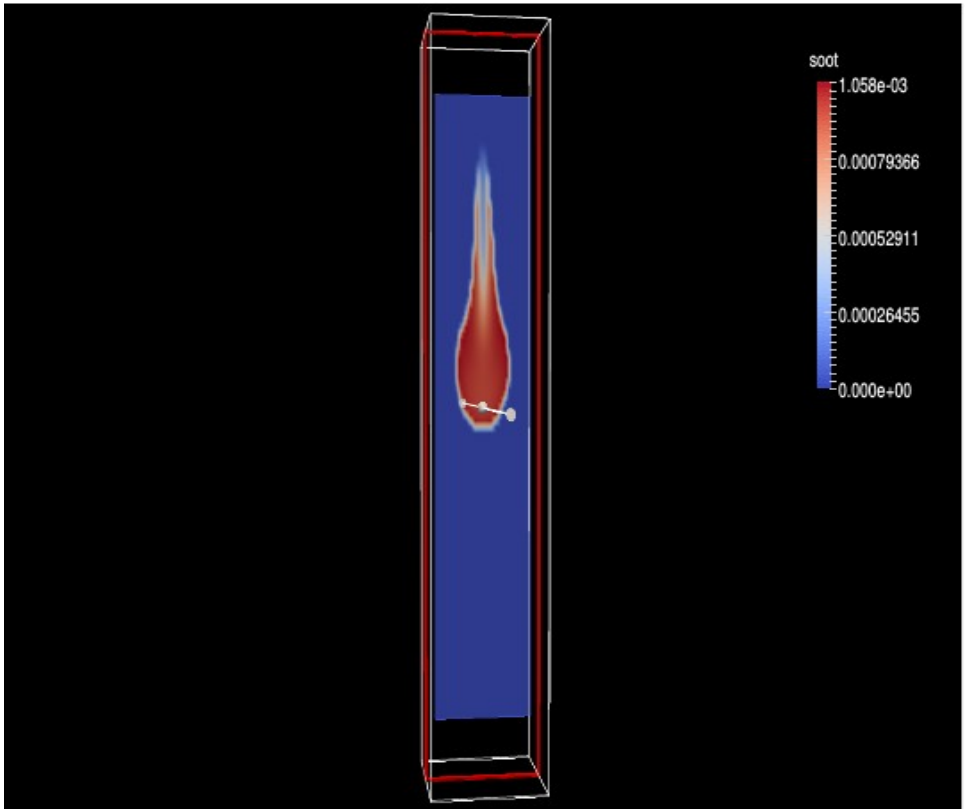
Comparison of soot models

- Slice of soot at t=1.4ms

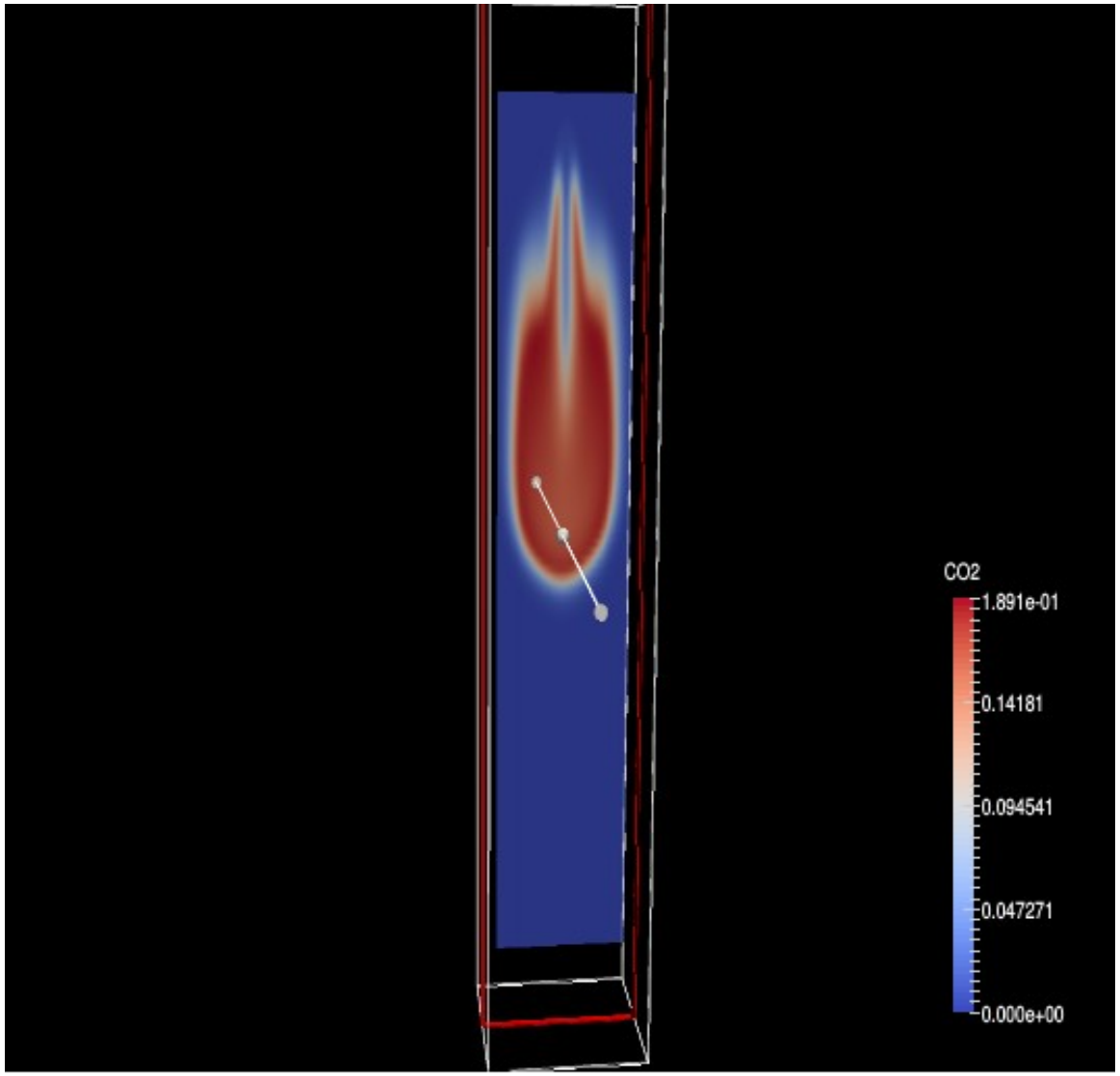
Existing model



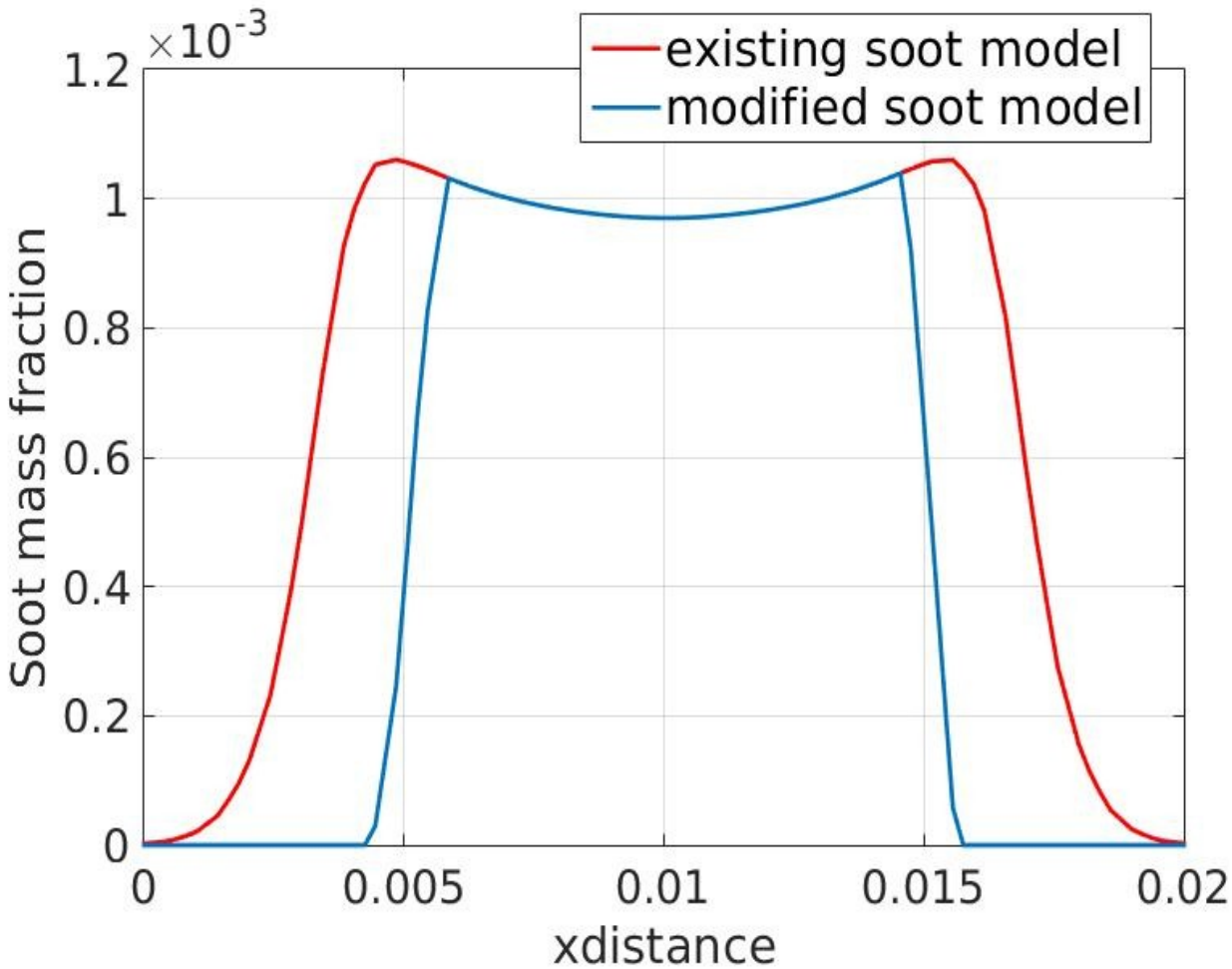
Modified model



Slice of CO2 at t=1.4ms



A look into sampledict



Soot distribution versus x-distance at t=1.4ms, for y=0.05m and z=0.05m

Summary

- Introduction to the aachenBomb tutorial
- Introduction to soot modeling
- Implementation of soot model source code
- Implementation of the sprayFoam solver
- Post processing and a look into sampleDict

Thank you!