POLITECNICO
MILANO 1863

CFD Assignment report

Master of science in aeronautical engineering

Computational Techniques for
Thermochemical Propulsion
051176

Antonio Pucciarelli
974675

Academic year: **2021/2022**

Abstract

This report explains the procedural steps and the results of CFD simulations made with OpenFOAM-9.

The simulations are about the flow analysis in a combustor chamber.

The problem domain consists of a combustion chamber with a splitter. In the combustion chamber there is the combustion of a hydrocarbon - `C7H16` or `CH4` -, that is injected through an injector, with air - `O2 + N2` -.

The combustor is discretized with 2 meshes: a 2D mesh[1] and a 3D mesh[2].

All the different steps made to reach the final 3D simulation are treated first separately on the 2D mesh. These steps are:

- **Mesh generation**
- **Incompressible flow problem**
- **Compressible flow problem**
- **Lagrangian particle tracking and wall film modeling problems**
- **Reactive flows problem**

The simulations are solved with:

- **Processor** Intel i7-1051U
- **RAM** 16 GByte
- **System** Ubuntu 20.04.3 LTS

[1] Hydrocarbon used for the reactive case: `CH4`.
[2] Hydrocarbon used for the reactive case: `C7H16`.

Contents

List of Figures

List of Codes

1 Mesh analysis

The mesh allows to discretize all the space occupied by the fluid. Every flow property is computed at the center of each control volume because of the finite volume method.

The mesh generation is one of the first step for achieving the convergence to the solution. The reason behind this relies on the fact that the mesh influences the numerics and so the simulation result. The main mesh properties to keep in mind are: **non orthogonality** and **skewness**. These properties are described in A.

1.1 3D mesh

The geometry (given in `.stl` format) is converted into 3D mesh[3] with OpenFOAM `snappyHexMesh` functions. The meshing procedure followed in OpenFOAM are:

- **Surface features generation with `surfaceFeatures`** This step allows extracting important features from the geometry e.g. important edges and/or important areas in the mesh where it is needed more accuracy (the more the control volumes the more the accuracy, not considering numerical discretization). `surfaceFeaturesDict` sets the commands to follow for the features generation, the main ones are the splitter edges.
- **Mesh generation with `snappyHexMesh`**[4] This step has 3 substeps:
  - `castellatedMesh` This substep allows the removing of blocks that are inside the geometry. The result is a raw block mesh that surrounds the body
  - `snap` This substep allows the projection of the raw mesh onto the body surface.
  - `addLayers` This last substep allows the inflation of the control volumes close to the surface with smaller control volumes.

Coding the meshing steps, it is important to keep in mind the CPU power available - maximum number of control volumes to have good results in appreciable

[3] The 2D mesh is already given as `blockMesh` file and its properties are:

- **Skewness**: max 0.84302.
- **Non orthogonality**: max: 39.635 average: 16.5967.

[4] The 2D mesh and the 3D mesh follow 2 different tracks: the 2D mesh is based on already existing mesh points (using `blockMesh`); the 3D mesh is based on the computation of the control volumes' points through an optimization algorithm (using `snappyHexMesh`).

time - and physical phenomenon to study - shock waves, vortex shedding and flames need more accurate meshes than an airfoil studied with potential flow theory -.

With respect to these points, since the simulation deals with a hydrocarbon combustion, it is needed to have a good mesh close to the injector position - in order to track better the particles motion -, close to the splitter - where there is a discrete complexity in the geometry and there is the impingement of hydrocarbon particles with the wall film layer - and after the splitter (bluff body) where it is present vortex shedding phenomena.

With respect to the numerics and topology, it is set a smooth layer transition among the splitter, the combustion region and the combustor case (the layer passes from a level 4 close to the splitter to level 3 close to the case).

At the end of the meshing process the 3D mesh is checked through `checkMesh` and its properties are:

- **Skewness**: max: 1.9082.
- **Non orthogonality**: max: 64.8094 average: 7.20918.

The max skewness value is acceptable and also the non orthogonality values are good. The 3D mesh is composed by hexahedron (geometry that works best in OpenFOAM), polyhhedra and prisms.

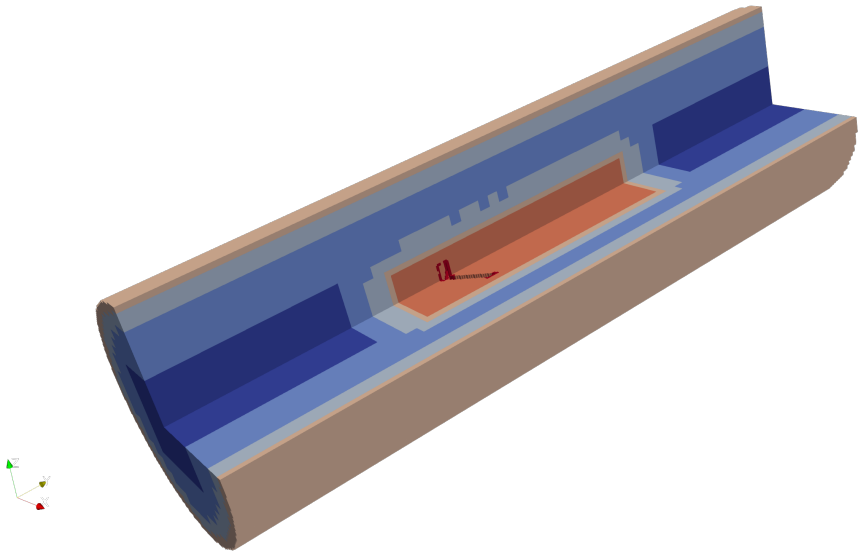

Figure 1: 3D mesh. Cell layers where identified with different colors. Due to the available computational power, it has been decided to use a lower number of cells in order to reach an acceptable solution in the shortest time possible.

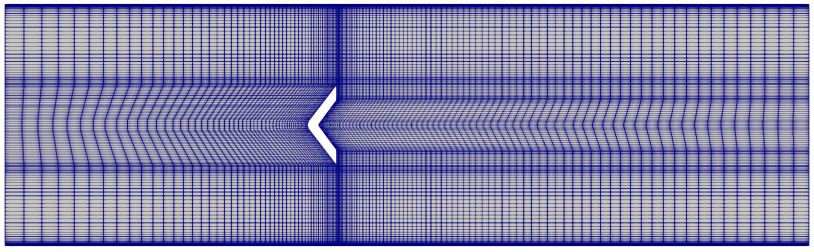

Figure 2: 2D mesh.

2 Incompressible flow - `Lab02`

All the properties relative to the incompressible flow solvers - `SIMPLE`, `PISO` and `PIMPLE` - are described in detail in B.

2.1 Problem setup

2.1.1 Boundary conditions

The boundary conditions to be set are relative to the $p - \boldsymbol{u}$ and the turbulence model. Since the problem is incompressible, p is expressed as $\frac{p}{\rho}$ in order to facilitate the computation[5]. The internal field for $p - \boldsymbol{u}$ is 0.

Turbulence model Since the `RAS` model uses the $\kappa - \varepsilon$ tubulence model [3, Ch. 10.4], set in `constant` with `momentumTransport`, it is needed to set up $\kappa - \varepsilon - \nu_t$ boundary conditions.

ε For ε (turbulent dissipation rate $\frac{1}{2}\ \nu\ <\ s'_{ij}s'_{ij}\ >$[6]), the inlets have the same turbulence setup `turbulentMixingLengthDissipationRateInlet` and the `mixingLength` is fixed[7]. The walls are modeled with a wall function model: `epsilonWallFunction`. The outlet is set to `zeroGradient` (it means that ε does not change **in space** at the outlet).

κ The κ (turbulent kinetic energy $\sum_i < u'_i u'_i >$) follows the same philosophy of ε - turbulent model at the inlet with `turbulentIntensityKineticEnergyInlet`, `kqrWallFunction` at the wall and `zeroGradient` at the outlet -.

ν_t Although ν_t (turbulent viscosity to be added to the molecular viscosity ν described in `constant/transportProperties`) is completelly described by $\kappa - \varepsilon$, it is possible to impose boundary conditions on it. ν_t at the walls is modelled with a wall function model `nutkWallFunction`; the rest of the field is directly computed via $\kappa - \varepsilon$ model.

2.1.2 Schemes

`fvSchemes` All the fields are discretized with a second order interpolation based on Gauss integration. The laplacian field and the surface normal gradient schemes

[5] This brings to set the `p` boundary conditions dictionary `dimensions [0 2 -2 0 0 0 0];`.

[6] $s'_{ij} = \frac{1}{2}\left(\frac{\partial u'_i}{\partial x_j} + \frac{\partial u'_j}{\partial x_i}\right)$, deviatoric part of $\boldsymbol{\nabla}\ \boldsymbol{u}'$

[7] Key parameter for the computation of ν_t.

are set to `corrected` that means the non orthogonality property is corrected via correctors (that allows sticking with a second order accuracy model).

`fvSolution` In the unsteady solver it is described the number of outer correctors, `nOuterCorrectors`, and the number of internal correctors, `nCorrectors`[8]. Turbulence is set as present in all the outer correctors with `turbOnFinalIterOnly no;`, this allows to have a better evaluation of ν_t. It is set the relaxation factors only for `combustorPimpleCFL15` in order to guarantee convergence due to `maxCo 15;`. In `combustorPimpeCFL1` there are not present any relaxation factors, this partially because `maxCo 1;` that allows less changes of the solution over the time steps. The `combustorPimple1` set up is not like `PISO` because there are 100 outer correctors (`PISO` has none).

For the error check it has been used `tolerance` and `relTol` both in the iterative methods, used for the computation of the fields, and also on the outer correctors (that can be seen as a refinement process of the $p^* - \boldsymbol{u}^*$ variables). Solvers are based on `GAMG`[9] and `smoothSolver`[10]. In the `combustorPimpleCFL15`, the `*Final`[11] relative tolerances (`relTol`) are set to 0; this in order to guarantee convergence using an error check based on `tolerance`.

2.2 Post processing

The post processing functions are declared in `system/controlDict`. The main values extracted are the solution's initial residuals, the forces on the splitter, the y^+ in the boundarly layer close to the walls and the U_y velocity at $y = 100mm$.

2.3 Results

2.3.1 `combustorSimple`

It is possible to see that `simpleFoam` does not reach convergence. This is a **numerical error** related to the body geoemtry: it is a bluff body. As result the solution oscillates between 2 *numerical* admissible solutions. The point here is that it is trying to simulate an unsteady phenomena with a steady solver.

[8] `nNonOrthogonalCorrectors 0;` because the mesh has low non orthogonality.

[9] Generalised geometric algebraic multi grid: used for the momentum equation solution (asymmetric matrix).

[10] Solver that uses a smoother: in the cases it is used for the continuity equation - Poisson equation - (symmetric matrix). Of course the smoother model has to be described with `smoothSolver`.

[11] This is a way of treating differently the last outer corrector cycle. The last outer corrector cycle depends on `nOuterCorrectors` or if the outer corrector tolerances are satisfied.

There are not problems with the drag force computation. Since the geometry is symmetric, the drag force over the *fitious* time steps is the same.

2.3.2 `combustorPimpleCFL1`

With `maxCo 1;` constraint, the results represent the vortex shedding after the splitter. Since the non physical initial conditions, the drag force has a large escursion the first time steps but then it settles down to a constant value. This constant value of drag force is made by the low changing pressure field around the splitter, although the vortex shedding in time.

2.3.3 `combustorPimpleCFL15`

With `max Co 15;` constraint, in this case the vortex shedding phenomena starts to show later in the solution. This is related to the filtering action of using a stable - outer corrector - solver with unlinked time-space discretization. As in the other solvers, the first time steps are affected by large escursion in the drag force due to the convergence of the internal field.

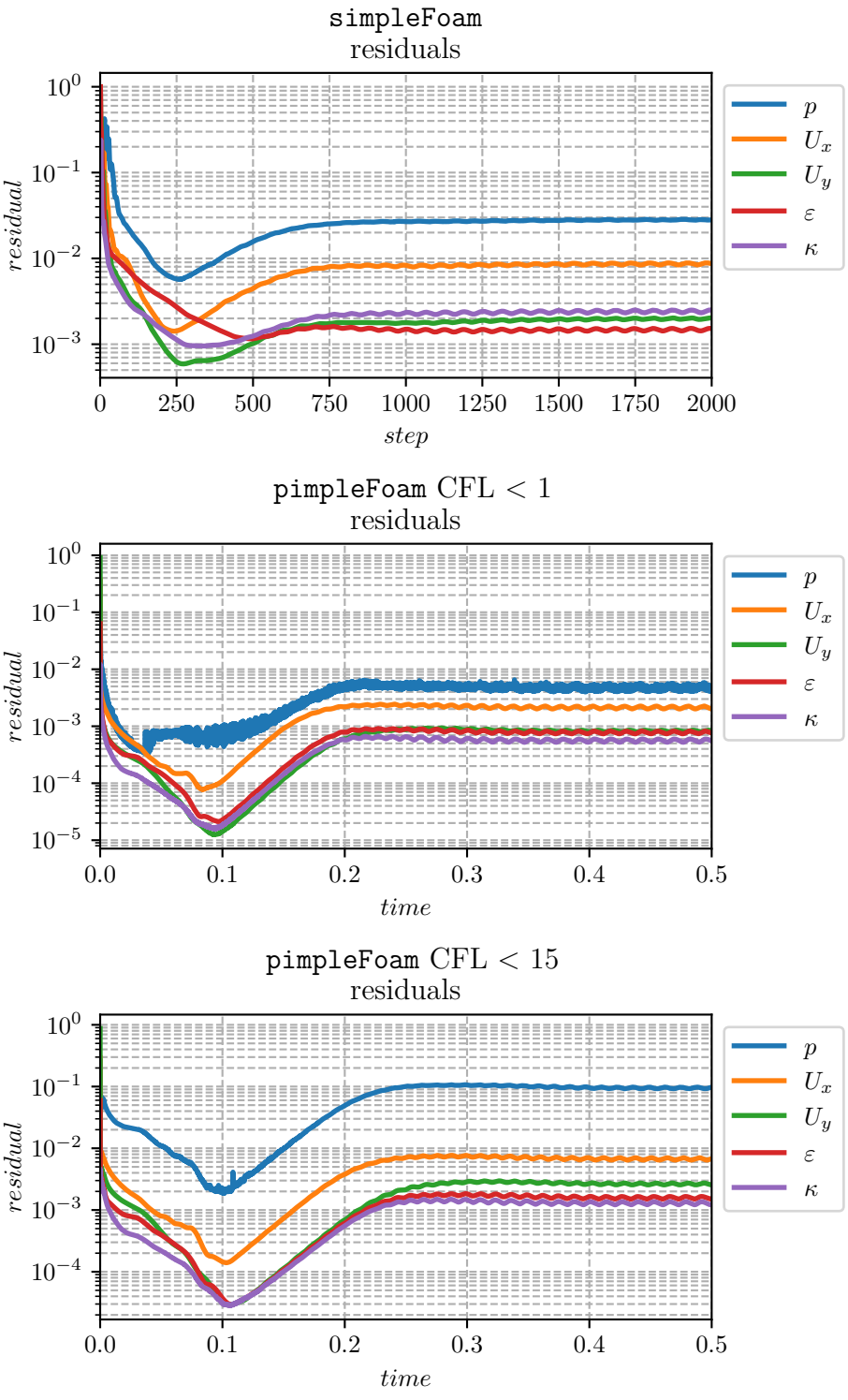

Figure 3: Incompressible cases: residuals.

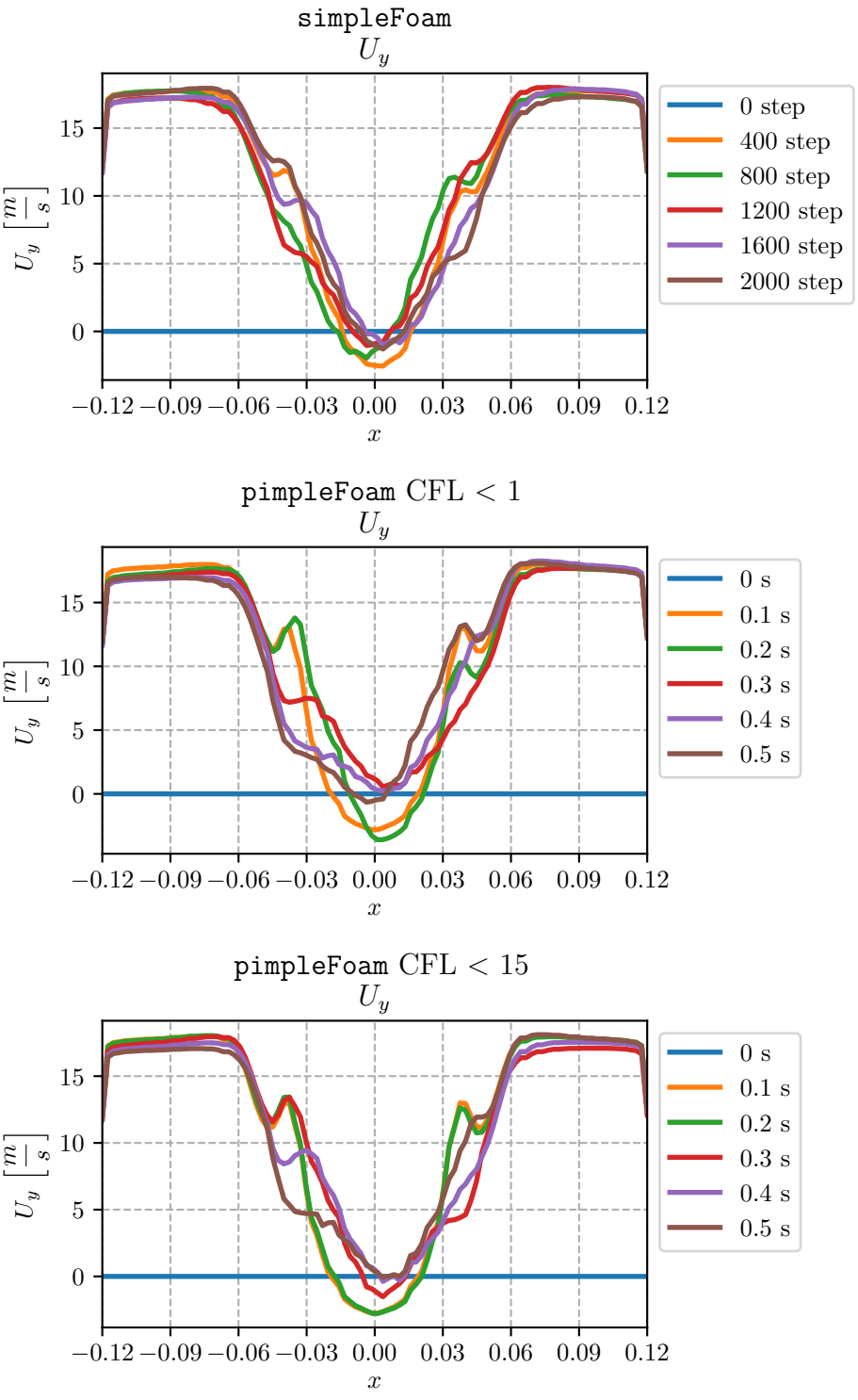

Figure 4: Incompressible cases: U_y.

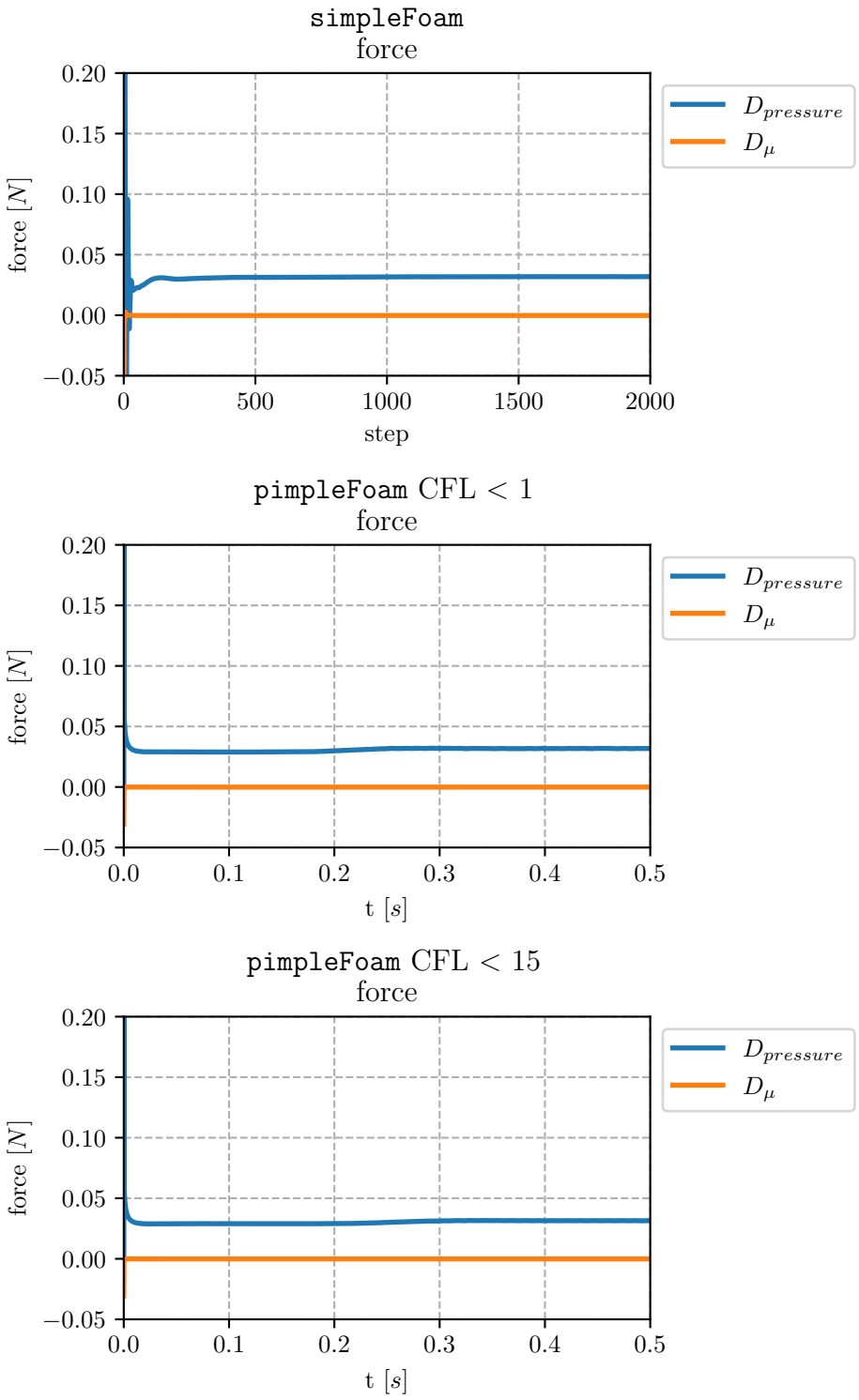

Figure 5: Incompressible cases: forces.

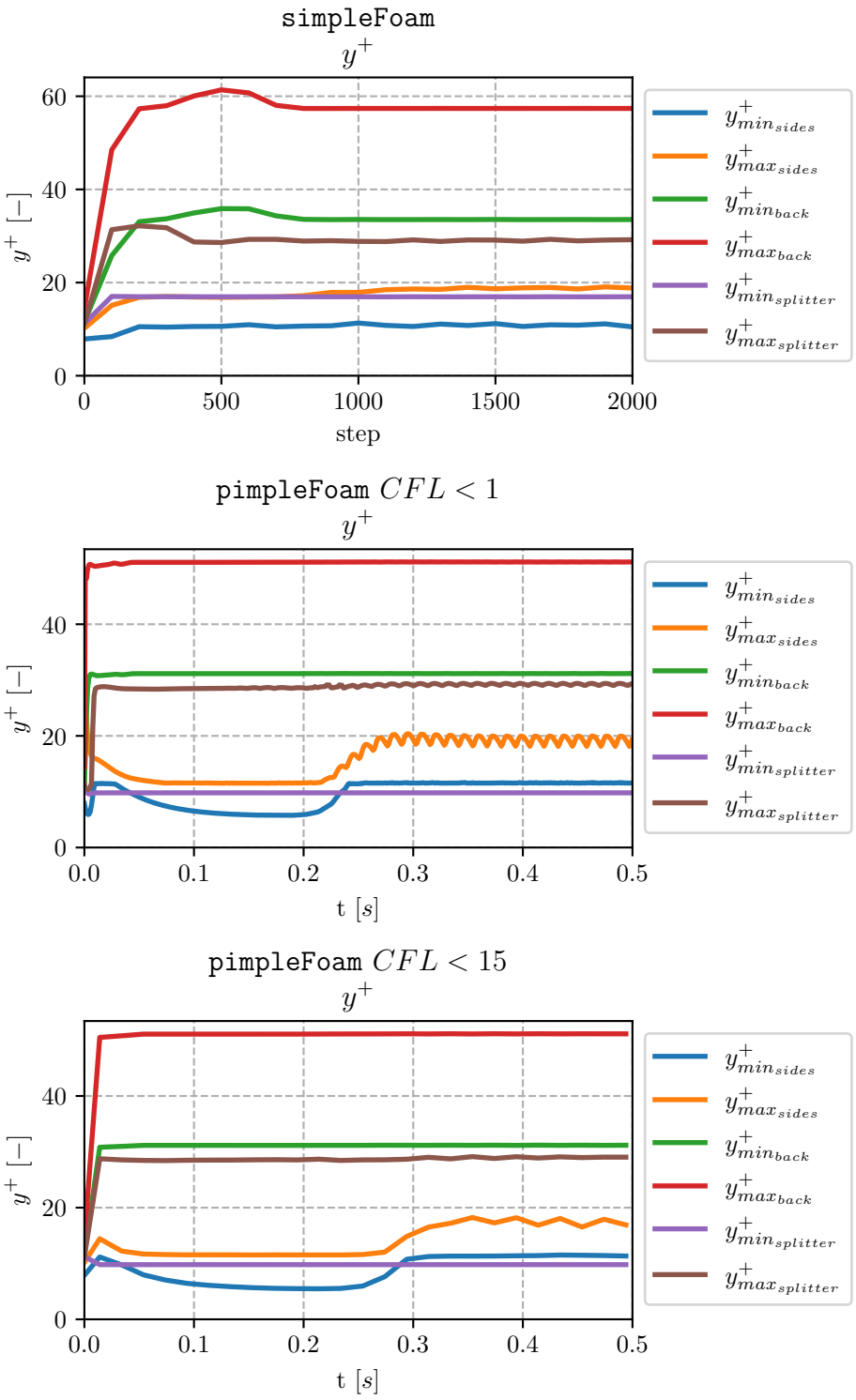

Figure 6: Incompressible cases: y^+.

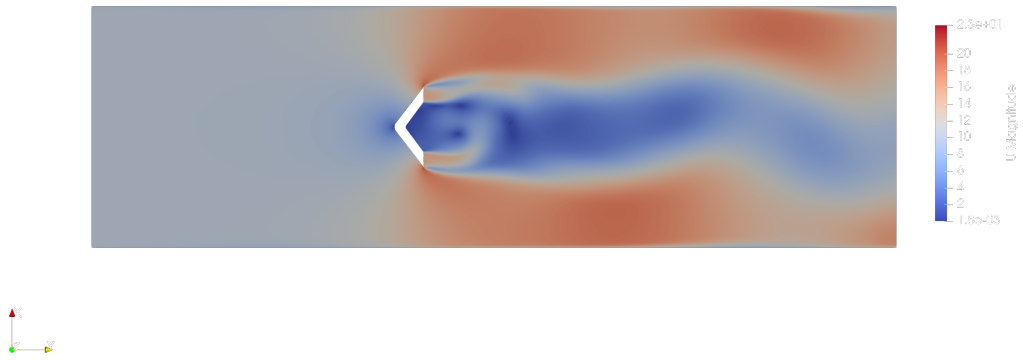

Figure 7: Velocity profile for combustor with CFL < 15.

3 Compressible flow - `Lab04`

In C are described the main equations, concepts and possible OpenFOAM implementation for compressible flows.

3.1 Problem setup

3.1.1 Boundary conditions

The main changes in the boundary conditions in `0/` folder are:

p Due to the presence of pressure waves, because the hyperbolic formulation of the problem, the outlet boundary condition has to be changed. As result of this, the outlet allows pressure waves to pass through and it allows also to not over-constraint the problem. So the outlet is set as `waveTransmissive` type and with the pressure field at infinite equal to `internalField`. Of course, the dimension of p has to change because now the problem is no more formulated as $\frac{p}{\rho}$.

α_t Due to the compressibility, the problem has to deal with turbulent heat transfer coefficient α_t. All the α_t are set as `calculated` except at the walls where it is used a wall function `compressible::alphatWallFunction`[12].

T Setting T boundary conditions is like setting h boundary conditions, because their relation. The `T` file changes for each problem because different boundary conditions at the walls and the flow field temperature.

3.1.2 Schemes

`fvSchemes` New schemes are added such `div(phi,h)` and `div(phi,K)` that allow to compute $\boldsymbol{\nabla} \cdot \left(\rho \, \boldsymbol{u} \, h\right)$ and $\boldsymbol{\nabla} \cdot \left(\rho \, \boldsymbol{u} \, K\right)$ respectilvely.

`wallDist` is a new function that describes the method used for the `waveTransmissive` type in `p`.

`fvSolution` The compressible case has to deal also with how to solve the energy equations, so solution schemes and tolerances are declared as well as p and U for h.

[12] The use of wall functions in κ, ε, ν_t and α_t is related to the fact that the centres of all the nearest control volumes to the walls are at $y^+ > 10$ and so out of the **viscous subregion**. Being out of this region it is necessary compute with the wall function these values. This will result in exploiting the results of theory: **log-law** model adoption.

3.1.3 `thermophysicalProperties`

This file allows to describe the way the equation of energy is solved and how to compute ρ from the available fields.

`thermoType` This part of the dictionary declares how to solve energy and ρ through `hePsiThermo`.

- `hePsiThermo` - `type` - allows the usage of the equation of energy base on $h-e$ and the usage of ψ based method for the computation of ρ (so explicitly related to p).
- `pureMixture` - `mixture` - allows treating a single mixture in gaseous phase.
- `perfectGas` - `equationOfState` - allows using the perfect gas equation $\rho = \frac{p}{RT}$ for ψ.
- `sensibleEnthalpy` - `energy` - allows using the h formulation for the energy equation.

`mixture` This part of the dictionary declares the mixture properties.

The following `mixture` properties are related to gas propeties:

- `specie` sets the molecular weight of the substance - air in this case -.
- `thermodynamics` sets the thermodynamics relations between T and h; based on constants as expressed in `thermoType -> thermo hConst;`
- `transport` sets transport properties of the mixture and are set to `const` in `thermoType -> transport const;`.

`dpdt yes;` enables the usage of $\frac{\partial p}{\partial t}$ used in the h formulation.

3.2 Post-processing

All the post-processing functions, except the residual function, print the field with time interval $\Delta t = 0.1s$ in order to save disk space.

3.2.1 `MachNo`

This is a post-process function that enables the computation of Mach field using $Ma = \frac{U}{\sqrt{\gamma RT}}$.

3.3 Results

3.3.1 `combustorRhoPimple_lowT_lowRe`

This simulation was set `maxCo 5.0;`. Vortex shedding is present in the field. With respect to the other fields, pressure waves travel at lower speed, this brings to a longer time for the velocity to change.

3.3.2 `combustorRhoPimple_highT_lowRe`

This simulation starts to show instability effects from the last few steps of the simulation. The pressure wave speed[13] is higher than in the `combustorRhoPimple_lowT_lowRe` case.

3.3.3 `combustorRhoPimple_hightT_highRe`

As the previous simulation, flow instability effects start to be present in the final solution steps.

[13] $c = \sqrt{\gamma\ R\ T}$.

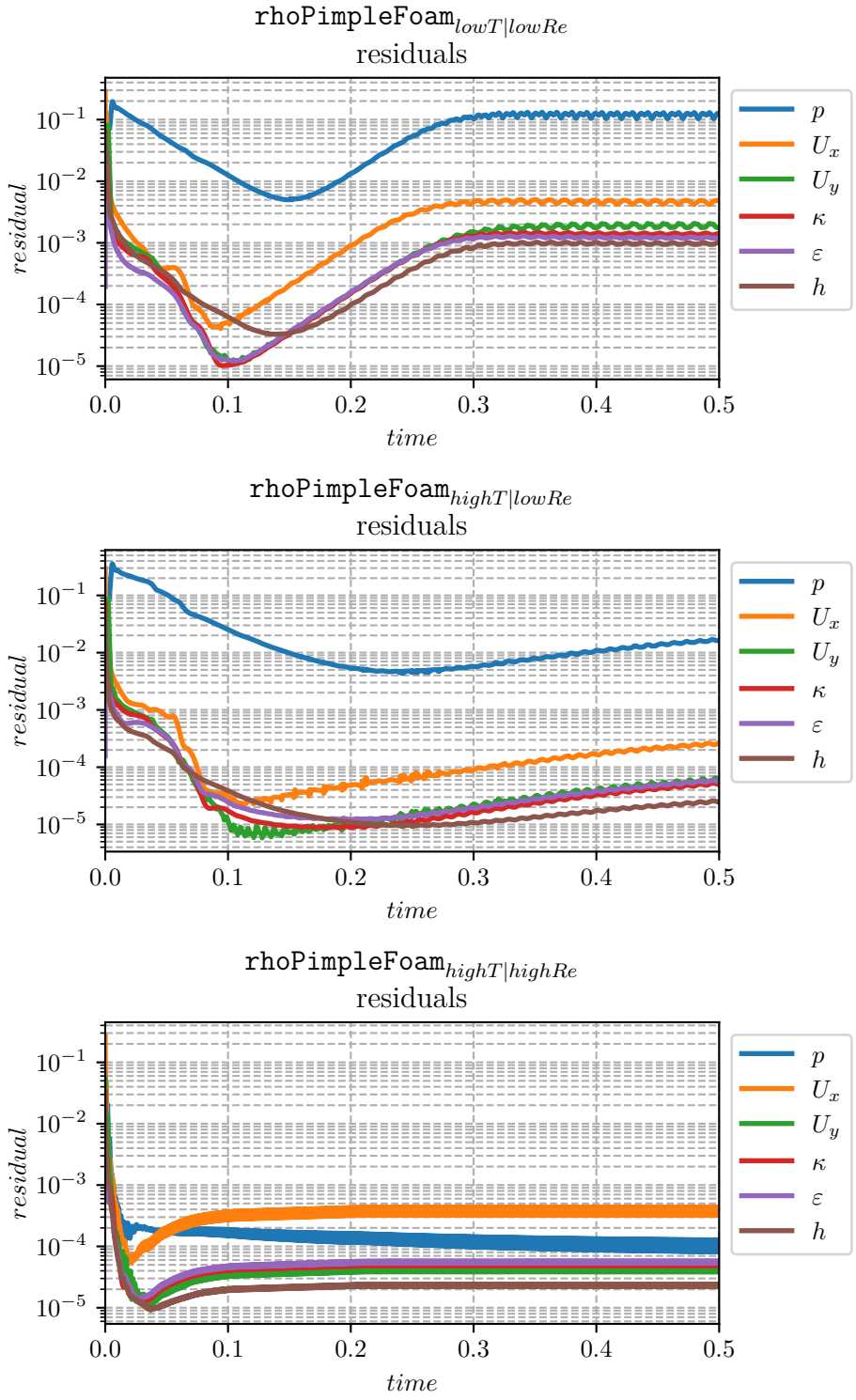

Figure 8: Compressible cases: residuals.

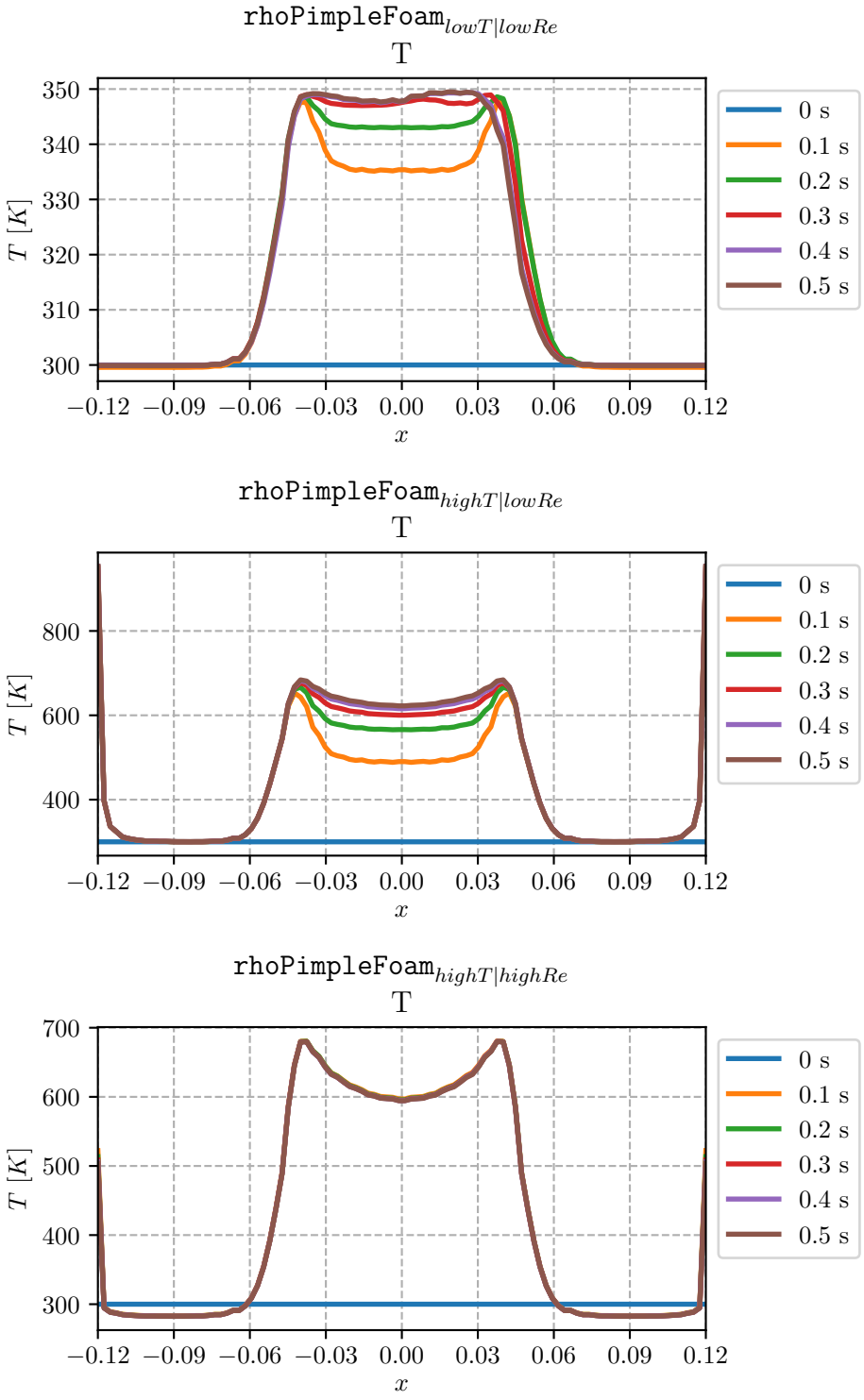

Figure 9: Compressible cases: T profile at $y = 100mm$.

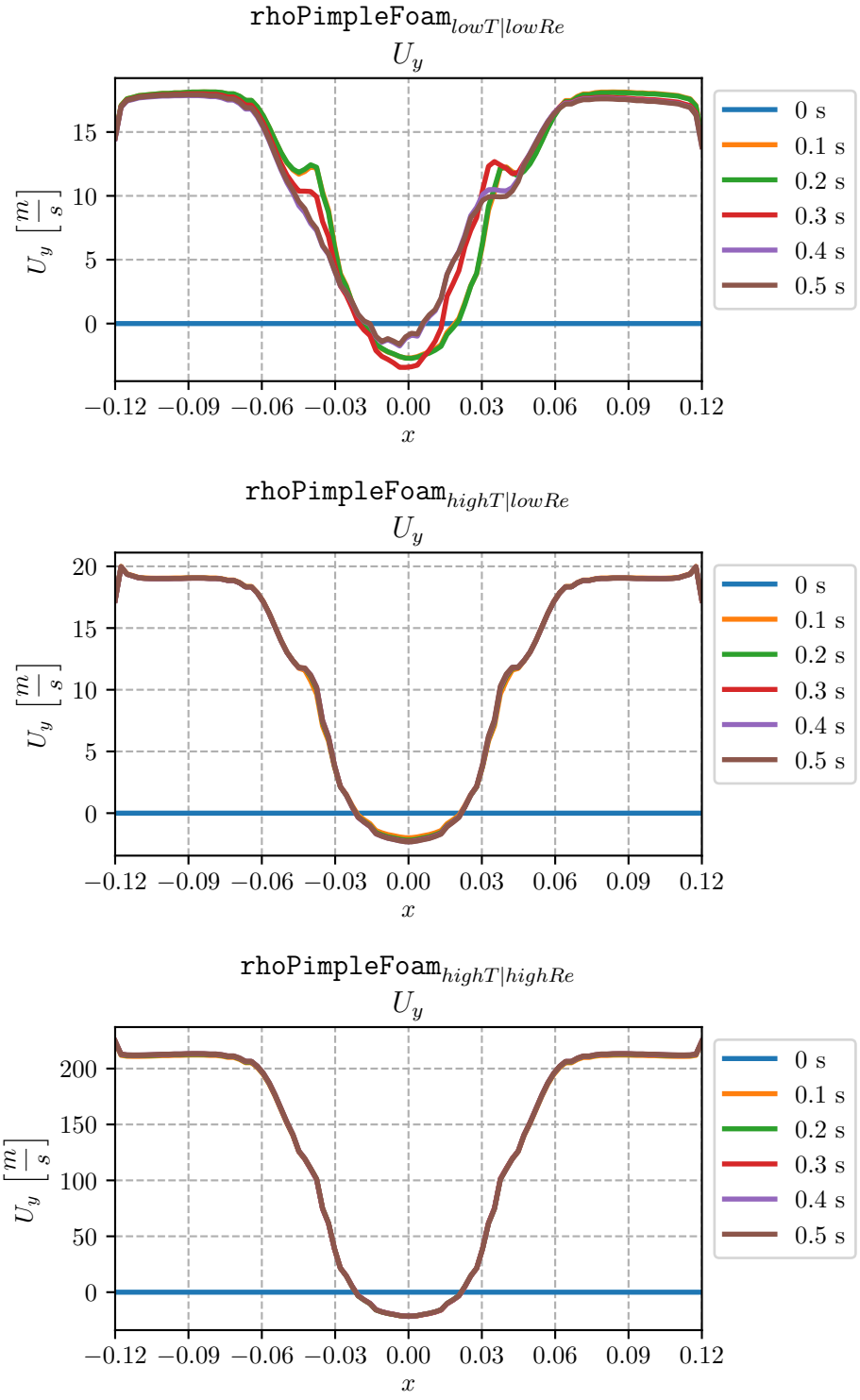

Figure 10: Compressible cases: U_y profile at $y = 100mm$.

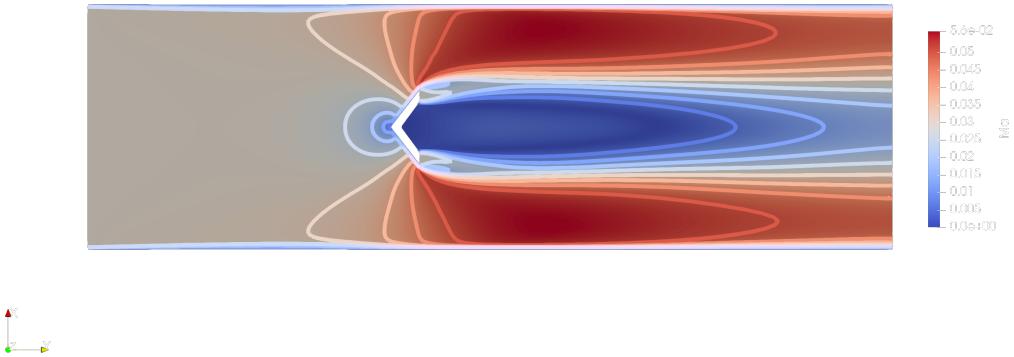

Figure 11: Mach isolines for `combustorRhoPimple_highT_lowRe` at $t = 0.4s$.

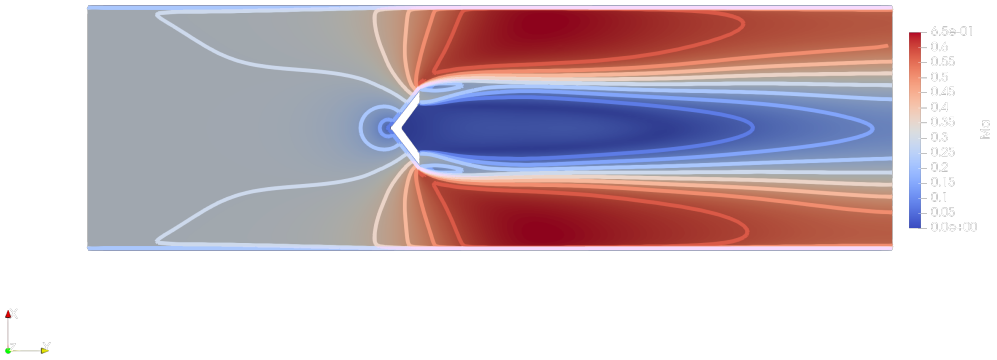

Figure 12: Mach isolines for `combustorRhoPimple_highT_highRe` at $t = 0.4s$.

4 Lagrangian particles tracking and wall-film modeling - `Lab05` and `Lab06`

Spray modeling and wall-film theory are described in appendix D and appendix E.

4.1 Problem setup

It is possible to see this new case as the previous compressible case plus two **additional layers** of modeling: the lagrangian particles evolution and the surface wall-film modeling. These new layers are described by defined dictionaries. The `fvModels` allows to enable these dictionaries and to **wrap** them into the *main* case. The wall-film region is generated through the `topoSet` dictionary. This dictionary coverts patch of the system into `faceSet` or `faceZoneSet` objects; these objects allow to make new operations such as wall-film region generation. The wall-film region is made using the `extrudeToRegionMesh` dictionary. This dictionary makes a one layer 2D mesh from the patch converted by `topoSet` firstly into `faceSet` and then into `faceZoneSet`. As additional step, it is necessary to change the boundary conditions; this means that the *main* field and the wall-film region field (in `0/wallFilmRegion`) have to set their boundary conditions on the `faceZoneSet` generated[14].

4.1.1 `cloudProperties`

This dictionary is a describer of the injector properties. The injector properties are summed up in parcels number, velocity, temperature, distribution and timing. These properties are assigned in `subModels -> injectionModels`. The force applied to the parcels are described in `particleForces`; only drag forces are present. The distribution model used is the `RosinRammler`. The interaction between parcel and wall is set as `rebound` (particle bumps on the surface). Particles diameter is allowed to change with an evaporation model set as `liquidEvaporationBoil`[15]. `breakupModel` describes the particle break up interaction into the simulation.

4.1.2 `surfaceFilmProperties`

The liquid film modeling is described by a complicated dictionary that containts in itself many submodels. These submodels are describers of the interaction of the wall-film with the *main* flow. `viscosity`, `momentumTransport`, `forces` and

[14] This is done through the `bash` script `AllSplitterRegion`.

[15] Every chemical property of the liquid is described in `chemkin` folder and converted into OpenFOAM syntax with `chemkinToFoam` command in `bash`.

`ejection` are the main subdictionaries. The `ejection` subdictionary describes the interaction of the liquid film with the splitter geometry with respect to its **shape**.

4.2 Post-processing

The post-processing setup is different from the previous cases. There exist different ways to get the results. Using the python code `latexData.py`, data from parcels size distribution and liquid penetration are available. A useful post-processing tool is `foamToVTK`. This command allows to reconstruct the case and all its properties in `VTK` format.

4.3 Result

The main differences between the different cases consist into particles breakup model - in `cloudProperties` - and *main* flow temperature.

4.3.1 `combustorSprayWallFilmReitzKHRT`

This case uses the `ReitzKHRT` breakup model. The effect of `rebound` setting in `cloudProperties` is very evident in the final solution. The particle jet reaches its maximum amplitude at the middle of the simulation due the the `sine` flow rate setting. Most of the particles impinges on the splitter. There are particles that are still not fully evaporated after passed the splitter.

4.3.2 `combustorSprayWallFilmReitzDiwakar`

This case uses the `ReitzDiwakar` breakup model. The jet behaviour is similar to the `ReitzKHRT`. There are changes in the particles distribution over time and of the liquid penetration. The reason relies in the way the breakup of the parcels is modeled; it is like the `ReitzDiwakar` treats the particles as less interactive with the environment (this explains why the liquid jet penetration is a bit higher than `ReitzKHRT`).

4.3.3 `combustorSprayWallFilmReitzKHRT450`

This last case uses the `ReitzKHRT` breakup model and `fixedValue -> value 450;` as boundary condition for the `T` at the `inlet`[16]. The solution is completelly different from the previous ones. This because the higher temperature of the *main* flow that allows faster evaporation of the parcels into the system.

[16] The other two cases used `fixedValue -> value 300;` as inlet temperature boundary condition.

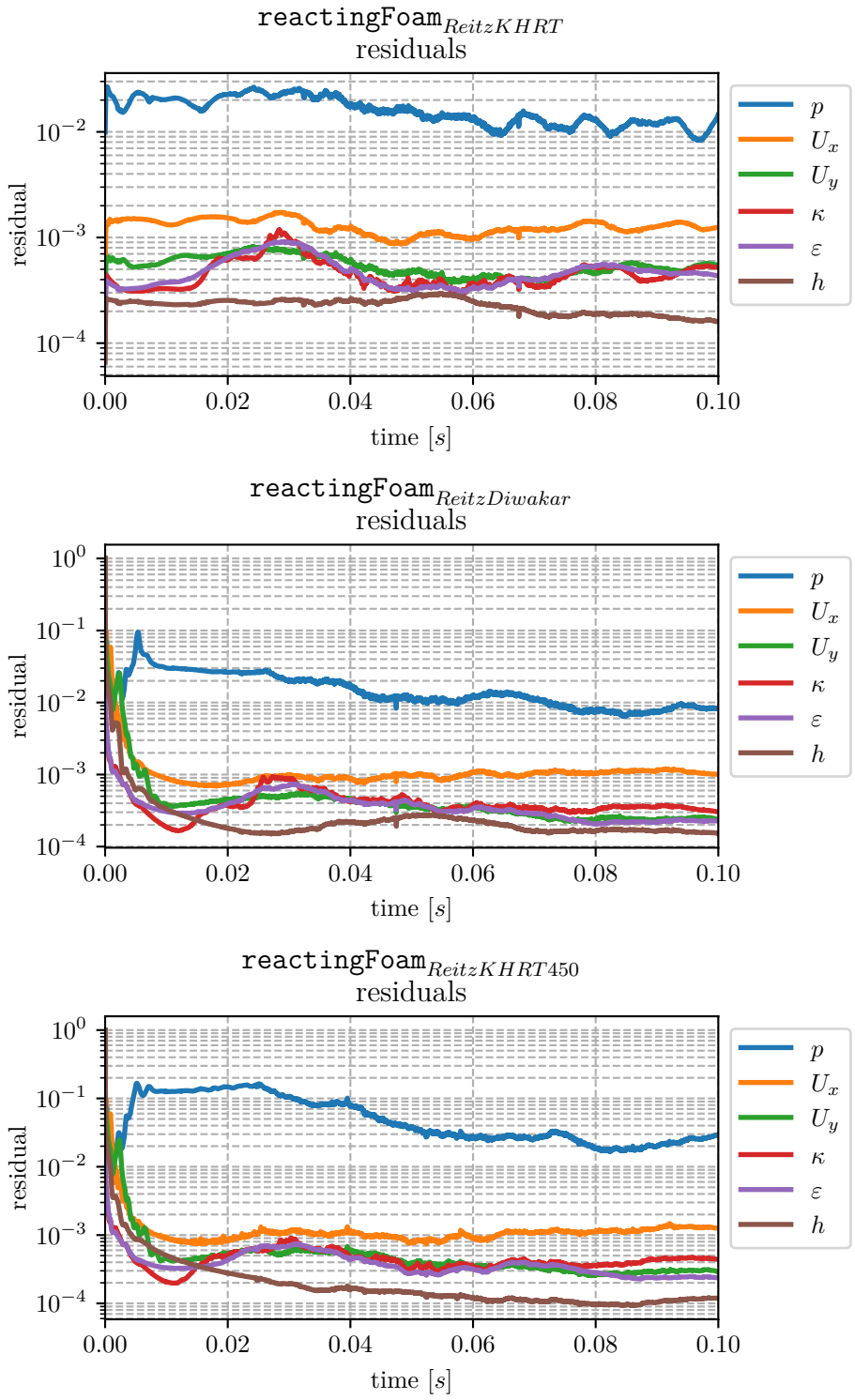

Figure 13: Surface & wall-film cases: residuals.

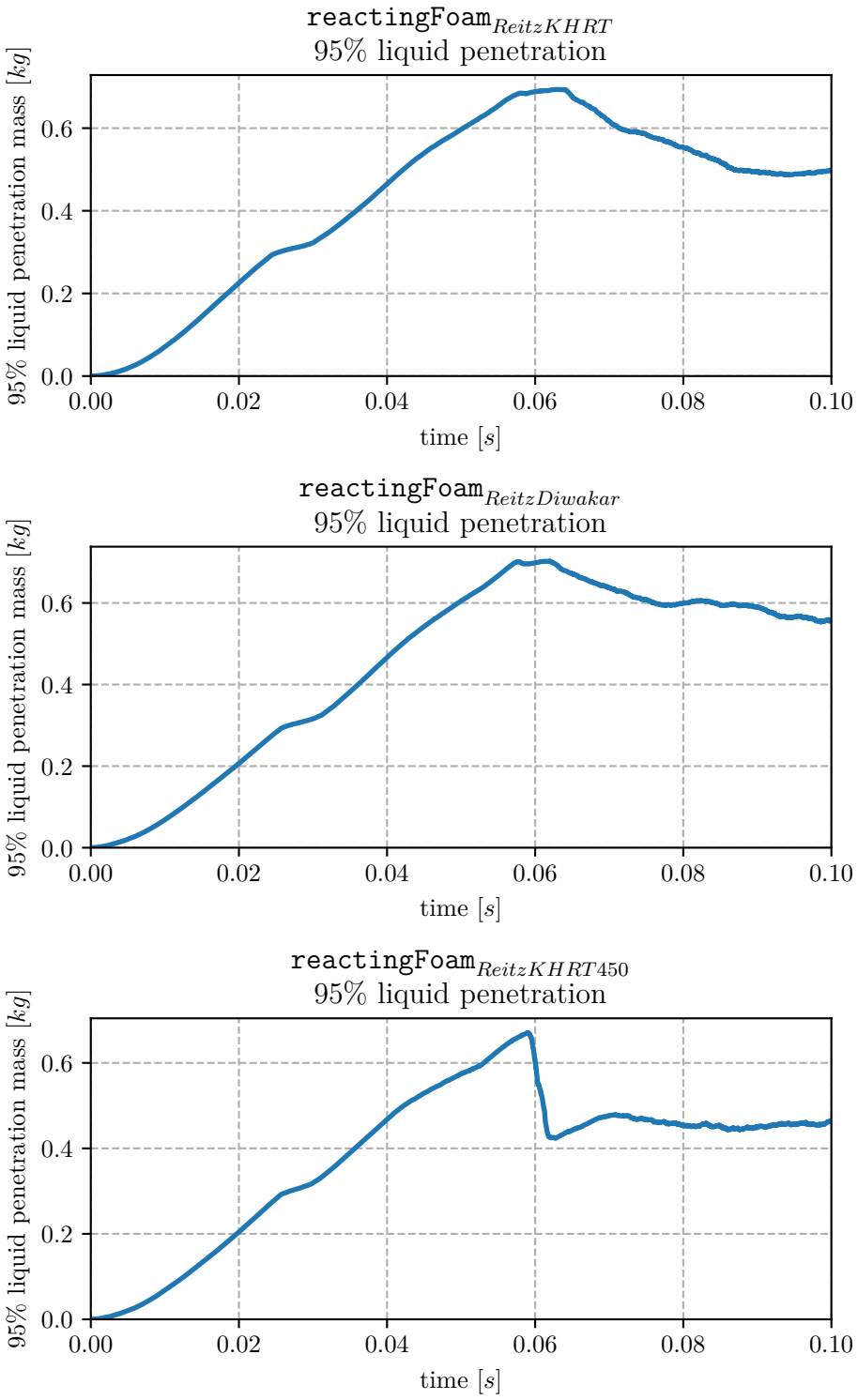

Figure 14: Surface & wall-film cases: liquid penetration.

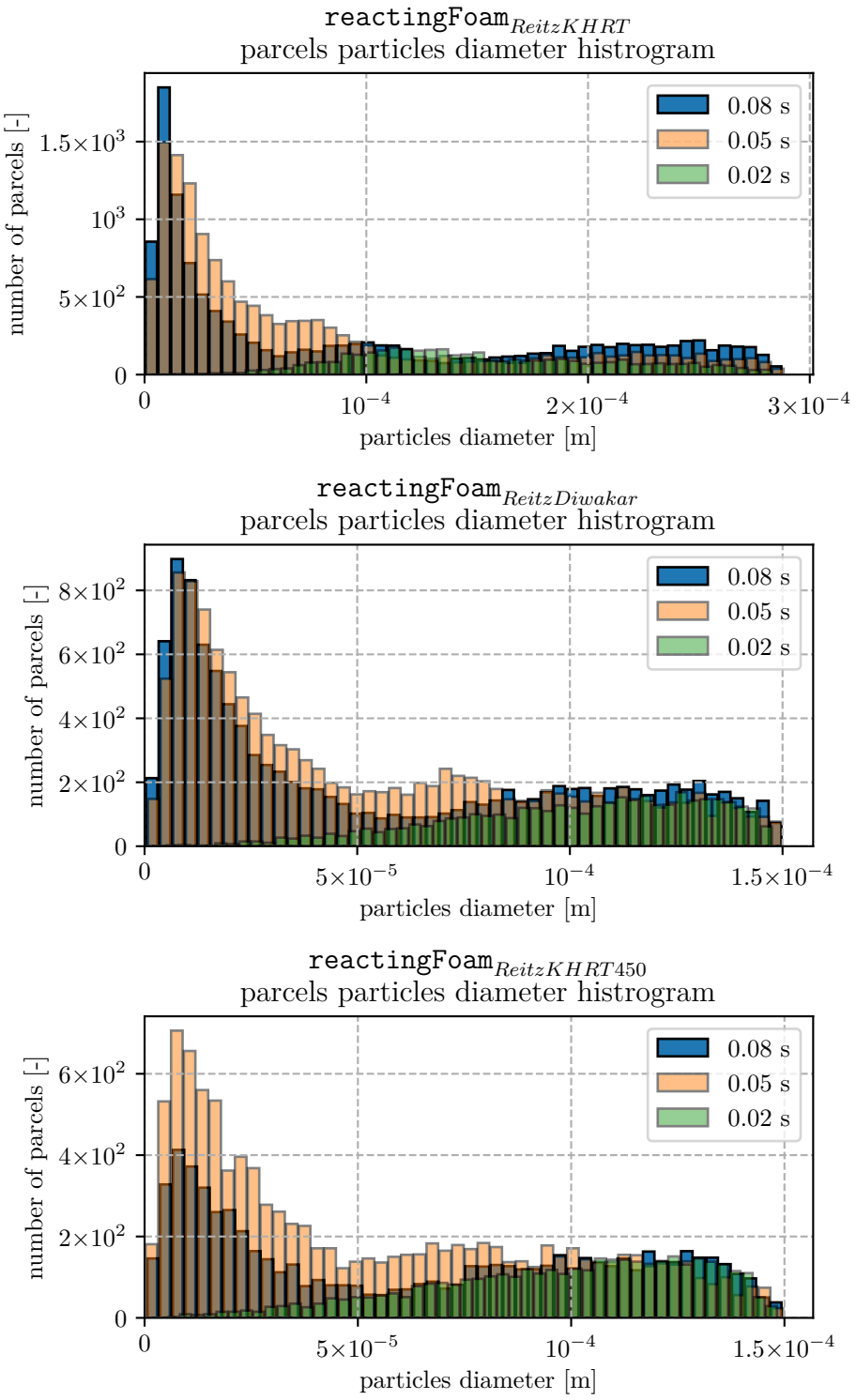

Figure 15: Surface & wall-film cases: parcels diameter.

Figure 16: C7H16 evaporation contour tracking, $t = 0.02s$. combustorSprayWallFilmReitzDiwakar

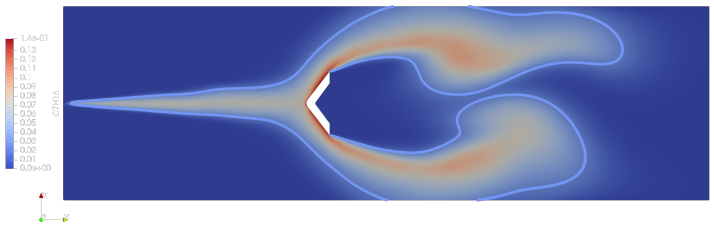

Figure 17: C7H16 evaporation contour tracking, $t = 0.05s$. combustorSprayWallFilmReitzDiwakar

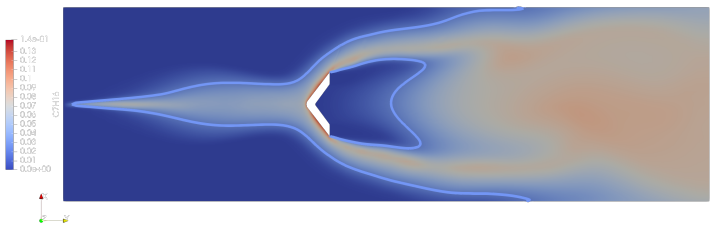

Figure 18: C7H16 evaporation contour tracking, $t = 0.08s$. combustorSprayWallFilmReitzDiwakar

5 Reacting flows - Lab07

This section treats the numerical modeling of chemical reactions. Reactions are treated as well as the spray modeling and the wall-film surface modeling, they are seen as an **additional layer** to the FVM problem. This additional modeling layer has to be blended into the *main* flow solution as it happened with lagrangian spray modeling and wall-film modeling. The important concepts related to combustion theory and modeling are explained in F.

5.1 Problem setup

5.1.1 Boundary conditions

Because the presence of different species - described by Y_i -, it is necessary to know the control volume species composition at time $t = 0$. As result of this, species dictionaries are created as many as the species in the system[17]. Since this simulation treats the combustion of CH4 with air, the fields dictionary set are CH4, O2, H2O, N2 and CO2. The initial values for the O2 and N2 field are computed from a previous simulation and then **mapped** into the combustorReaction case. The new main property in the boundary conditions setup is the inletOutlet type. This condition allows treating the field in a different manner with respect to inflow or outflow at the outlet boundary[18].

5.1.2 chemistryProperties

This dictionary describes the way chemistry is solved. Chemistry is activated through chemistry on; and it is used an implicit method to solve it:

Listing 1: combustorReaction/constant/chemistryProperties implicit chemistry model setup.

```
chemistryType
{
    solver          EulerImplicit;
    chemistryThermo psi;
}

chemistry               on;

initialChemicalTimeStep 1e-08;
```

[17] Ydefault dictionary can be found if the number of dictionaries does not match the number of species defined in the reaction dictionary. Ydefault dictionary is used only at $t = 0$, this because as the time step increases new species field are generated automatically by OpenFOAM.

[18] At $t = 0$ the field takes the value parameters, then it switches the cell center value with respect to the flow field direction at the outlet; if there is an inflow, the cell center takes 0 value.

The ODE implicit solver uses `seulex`[19] solver and the initial time step for the chemistry solver is set at `1e-08`.

5.1.3 `combustionProperties`

In this dictionary there are set information regarding how to treat the combustion as **laminar** flow or as **partial stirred reactor**. These two options are very important because they allow the *talk* between the combustion model and the turbulence model (in this case `RAS -> kEpsilon`). The `PaSR` modeling allows using a much rapid dispersion of the species into the system, this due to the turbulence model relation[20]. The `laminar` setting is most used for supersonic combustion[21] and for laminar flames[22].

5.1.4 `speciesThermo`

This dictionary containts the information of all the species in the system. Every species is described with the atomic composition (name and number), molecular weight, transport properties and thermodynamics. Thermodynamic properties are described with temperature range in which thermodynamic properties **fit best** and the coefficients that are used to generate a polynomial fitting of the thermodynamic properties with respect to temperature.

5.1.5 `reactions`

This dictionary cointaints the rules to follow for the reaction modeling. In this case the reaction is just *one way* and it is expressed with `irreversibleArrheniusReaction`. There are expressed the reaction formula and all the needed parameters for describing Arrhenius law, equation (11).

5.1.6 `thermophysicalProperties`

This dictionary is present for all the compressible flows. In order to make it work for the combustion simulation, it is necessary to link the thermodynamics of the species to this dictionary. These changes are in `thermo`, `mixture` and `transport`. The inert specie `N2` is declared as inert with `intertSpecie N2;` command.

[19] `seulex` is an extrapolation-algorithm, based on the linearly implicit Euler method with step size control and order selection [5]. This method increases convergence order of an implicit method; this can be achieved using two different time steps and a control parameter.

[20] Turbulence is expressed with ν_t in momentum equation diffusion term, the greater the turbulence the shorter the diffusion time. This concept affects also `PaSR` combustion modeling.

[21] This because turbulence diffusion has a longer characteristic time than that of the *main* flow.

[22] This because the turbulence is not present in the system and allows to have better and faster solutions.

Listing 2: `combustorReaction/constant/thermophysicalProperties`. This is the main link between the combustion model and the momentum transport equation.

```
thermoType
{
    type            hePsiThermo;
    mixture         multiComponentMixture;
    transport       sutherland;
    thermo          janaf;
    energy          sensibleEnthalpy;
    equationOfState perfectGas;
    specie          specie;
}
```

All the data `janaf` and `sutherland` use are taken from `thermo.mixture`[23].

5.2 Results

5.2.1 `combustorRhoPimple`

This is just the initialization of the flow. The main purpose of this case is to setup a developed flow in order to avoid the establishing of the flow in the `combustorReaction` case that would have been much more computationally demanding because the presence of combustion.

5.2.2 `combustorReaction`

The results show that there is a first mixing of `CH4` with the surrounding air. At a determined point combustion happens due to close to stoichiometric conditions of the mixture. Once the flame is generated, the combustion propagates much faster and the generation of `CO2` is much more evident than previously the flame generation[24]. Simulation results show huge fluctuations on the residuals, this may be related to the presence of combustion. A check has been made in the `log.reactingFoam` and no errors have been found.

[23] That is essentially a copy of `speciesThermo`.

[24] If `CH4` and air go in contact with proper temperature and pressure, reactoin happens but the heat generated from this reaction is related to the oxidizer-fuel ratio, `OF`, of the mixture. A proof of this is the presence of `CO2` before the flame develops.

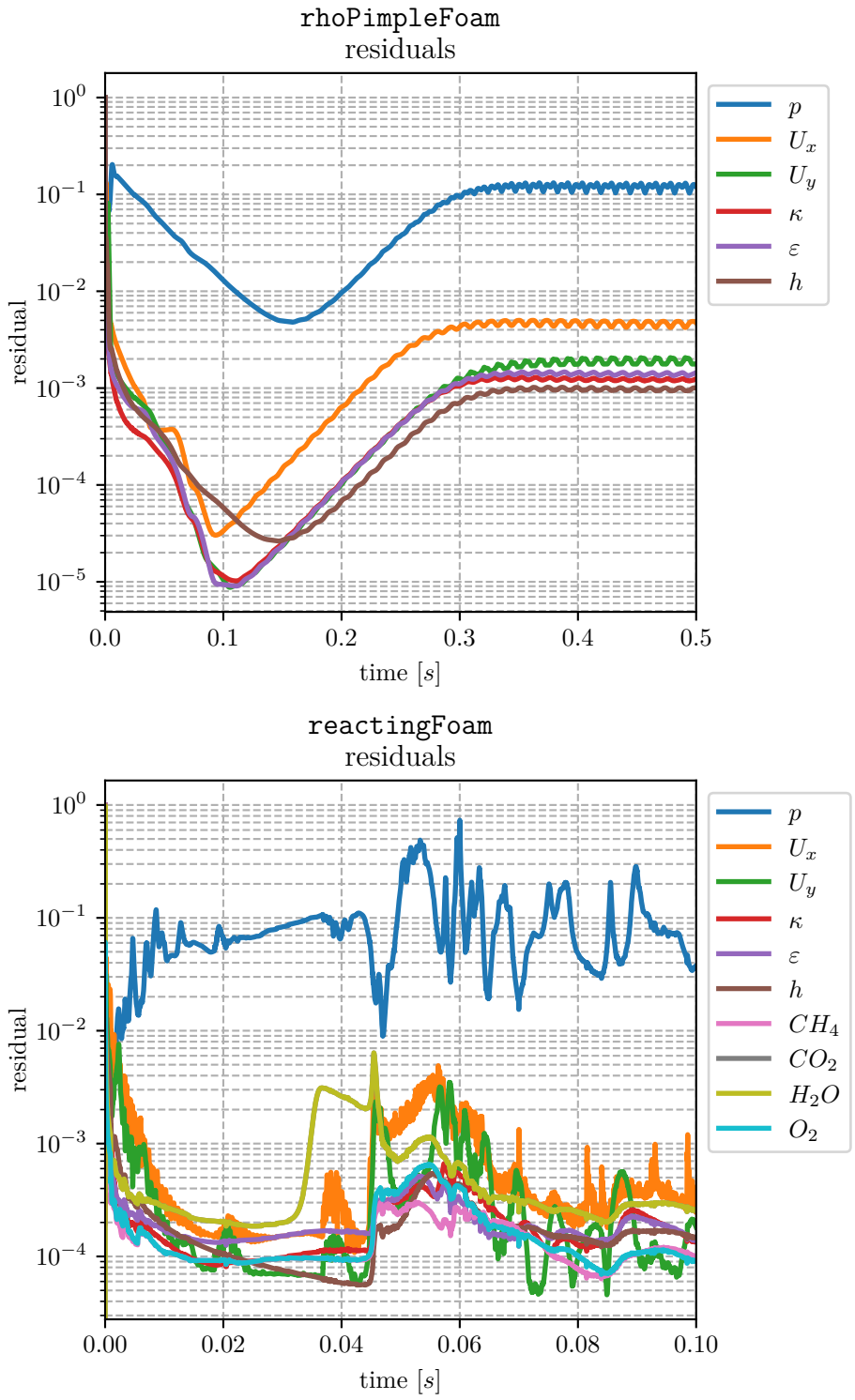

Figure 19: `rhoPimpleFoam` case and `reactingFoam` case: residuals.

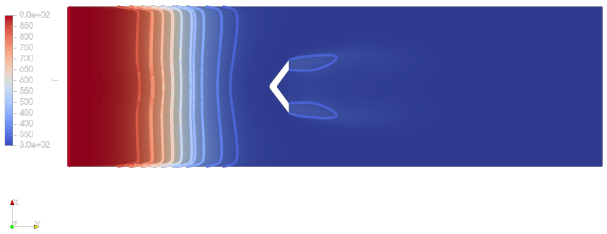

Figure 20: `T` contour plot, $t = 0.02s$.

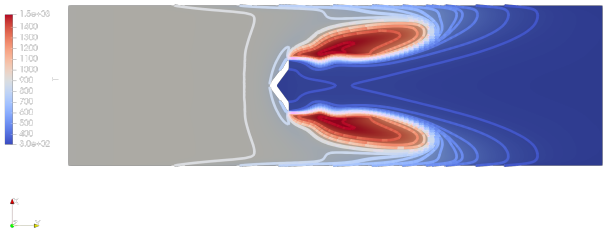

Figure 21: `T` contour plot, $t = 0.05s$.

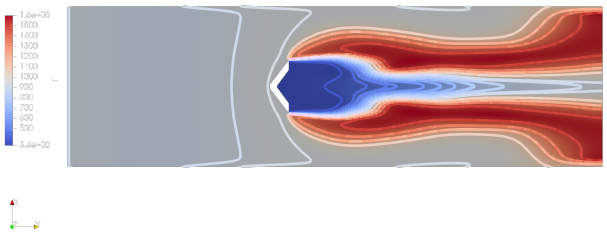

Figure 22: `T` contour plot, $t = 0.08s$.

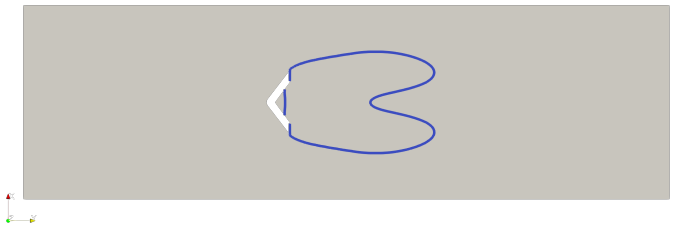

Figure 23: `A/F` stoichiometric contour plot, $t = 0.02s$.

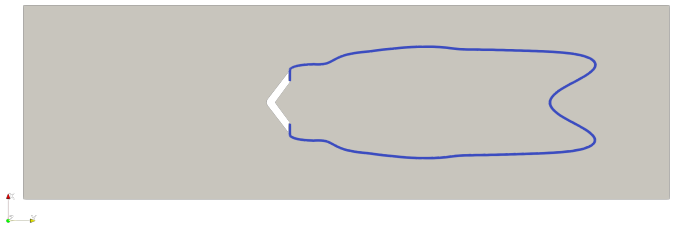

Figure 24: `A/F` stoichiometric contour plot, $t = 0.05s$.

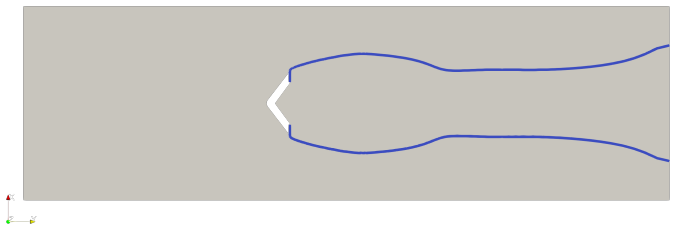

Figure 25: `A/F` stoichimetric contour plot, $t = 0.08s$.

6 Final assignment – `LabAssignment`

This last case treats the analysis of the 3D combustor.

The main difference among this case and the 2D cases is the computational cost related to the additional coordinate of study. Convergence parameters are set up in order to reduce the computational time and ensuring good results.

6.1 Problem setup

6.1.1 Boundary conditions

The main difference between the 2D case is the fact that there is no more the `frontAndBack` patch. The `frontAndBack` patch is changed with `outerWalls` patch. Because it is no more a 2D problem, the `outerWalls` does not use the `empty` type for the boundary conditions on every variables' dictionary.

6.1.2 `fvModels`

This dictionary activates *only* the lagrangian spray modeling. The liquid film modeling is not activated due to the computational resources available.

Listing 3: `combustorReaction/constant/fvModels` activates only lagrangian spray model and the bouyancy forces.

```
buoyancyForce
{
    type    buoyancyForce;
}

clouds
{
    type    clouds;
    libs    ("liblagrangianParcel.so");
}

//surfaceFilm
//{
//    type    surfaceFilm;
//    libs    ("libsurfaceFilmModels.so");
//}
```

6.1.3 `chemistryProperties`

As in `Lab07` chemistry solver is described in `chemistryProperties` and the solver used is the `seulex` solver. The reaction properties data are taken from `reactions` dictionary where the `C7H16`-air reaction is modeled as `irreversibleArrhenius`.

6.1.4 `combustionProperties`

As in 5.1.3, due to characteristic time of the reaction with respect to the *main* flow characteristic time, `PaSR` combustion model is used for the combustion modeling. The `PaSR` model considers turbulence for the combustion modeling.

6.1.5 `thermophysicalProperties`

This dictionary takes into account the presence of liquid `C7H16` into the system - because the presence of spray -, this will bring as setting `C7H16` in the `liquid` subdictionary. As in 5.1.4, species thermodynamics properties are taken from `speciesThermo`[25].

6.1.6 `fvSolution`

In order to reduce computational time for each time step, it has been used this setting:

Listing 4: `combustorReaction/system/fvSolution` convergence setting.

```
outerCorrectorResidualControl
{
    p
    {
        tolerance   5e-03;
        relTol      0.0;
    }

    U
    {
        tolerance   5e-04;
        relTol      0.0;
    }

    h
    {
        tolerance   5e-04;
        relTol      0.0;
    }
}

residualControl
{
    p   5e-03;
    U   5e-04;
    h   5e-04;
}
```

[25] For `Lab07`, `thermophysicalProperties` takes the thermodynamic properties of each species from the `thermo.mixture` dictionary, that is a copy of `speciesThermo`. In the `Lab-Assignemnt`, thermodynamic properties are recovered directly from `speciesThermo`.

The maximum number of number of outer correctors is 150 and the turbulence models are solved for each outer corrector step, `turbOnFinalIterOnly false;`, based on $\kappa - \varepsilon$ model.

6.2 Post-processing

6.2.1 `surfaceFieldValue`

In order to compute field average on `inlet` and `outlet` patches, the `surfaceFieldValue` function is used. Pressure and temperature are the averaged patch fields. The `writeInterval 1e-02;` is set in order to reduce the space occupied on the memory[26].

6.3 Results

6.3.1 `combustorRhoPimple`

This case is used for the initialization of the flow in the combustor case. It has been used a `maxCo 5;` for the simulation in order to get good initial conditions for the combustion in the shortest time possible. Of course, `maxCo 5;` can filter a bit too much the field but, with the computational power available, it has been preferred to use a bit finer mesh[27] to a low Courant number[28].

6.3.2 `combustorReaction`

The time spent for the solution of this case is around 3 days. Convergence is reached for every time step and the simulation results are shown below. Although the lack of the surface film modeling, spray interaction with the *main* flow and the `C7H16` combustion is present in the final solution. Combustion happens mainly at the end of the splitter after the mixing and evaporation of the `C7H16` spray with the air.

[26] The simulation occupies at the end of the simulation almost $3GB$.

[27] Around $1 \cdot 10^6$ cells.

[28] Having a low Courant number with coarse mesh is useless because a coarse mesh can be seen as a spacial filter of the results - as well as high Courant number can be seen as a temporal filter on the solution -. Although `RANS` are based on time filtering, an high Courant number (that is just related to numerics) can be seen as a **numerical model filter**.

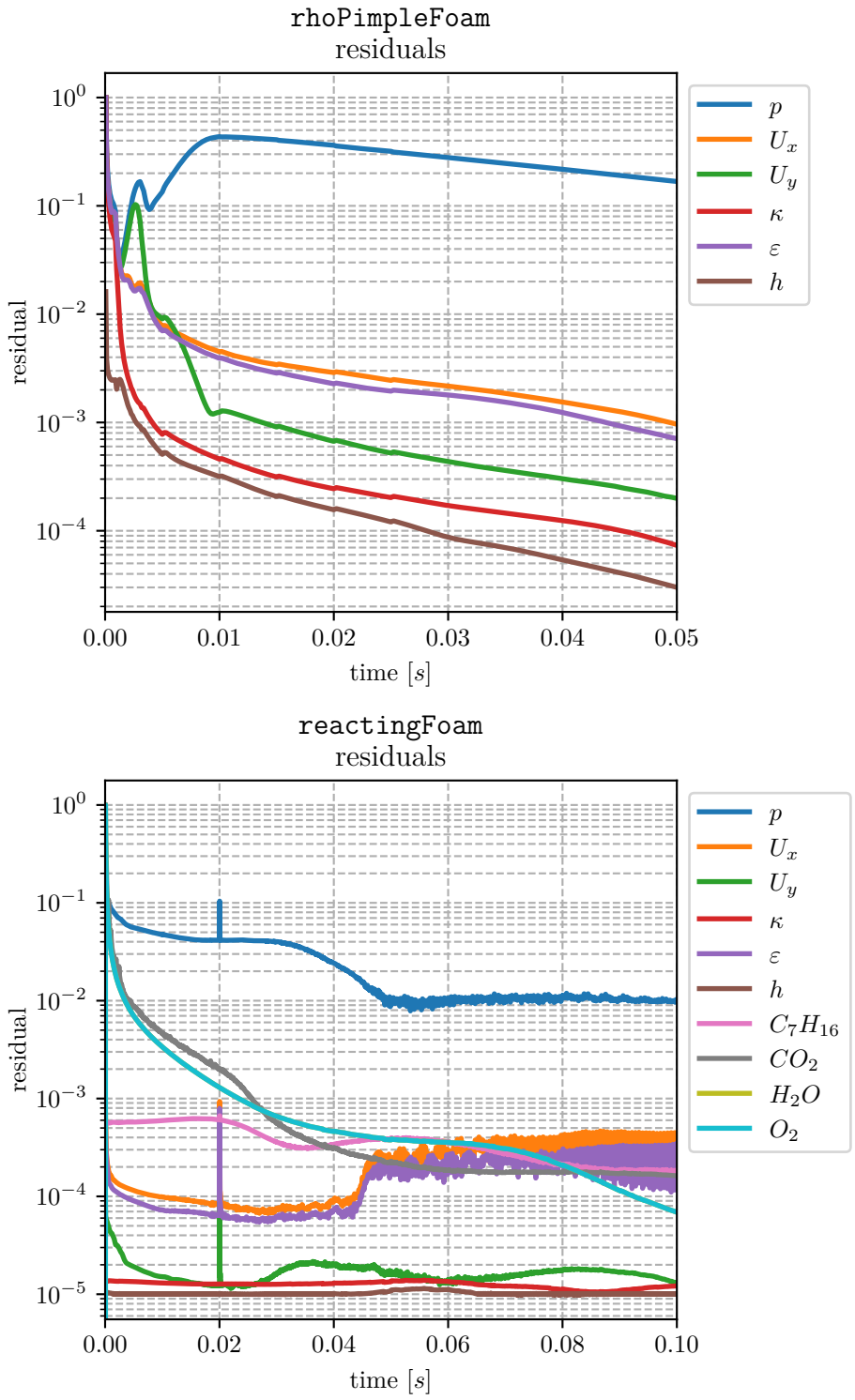

Figure 26: 3D case `rhoPimpleFoam` and `reactingFoam`: residuals.

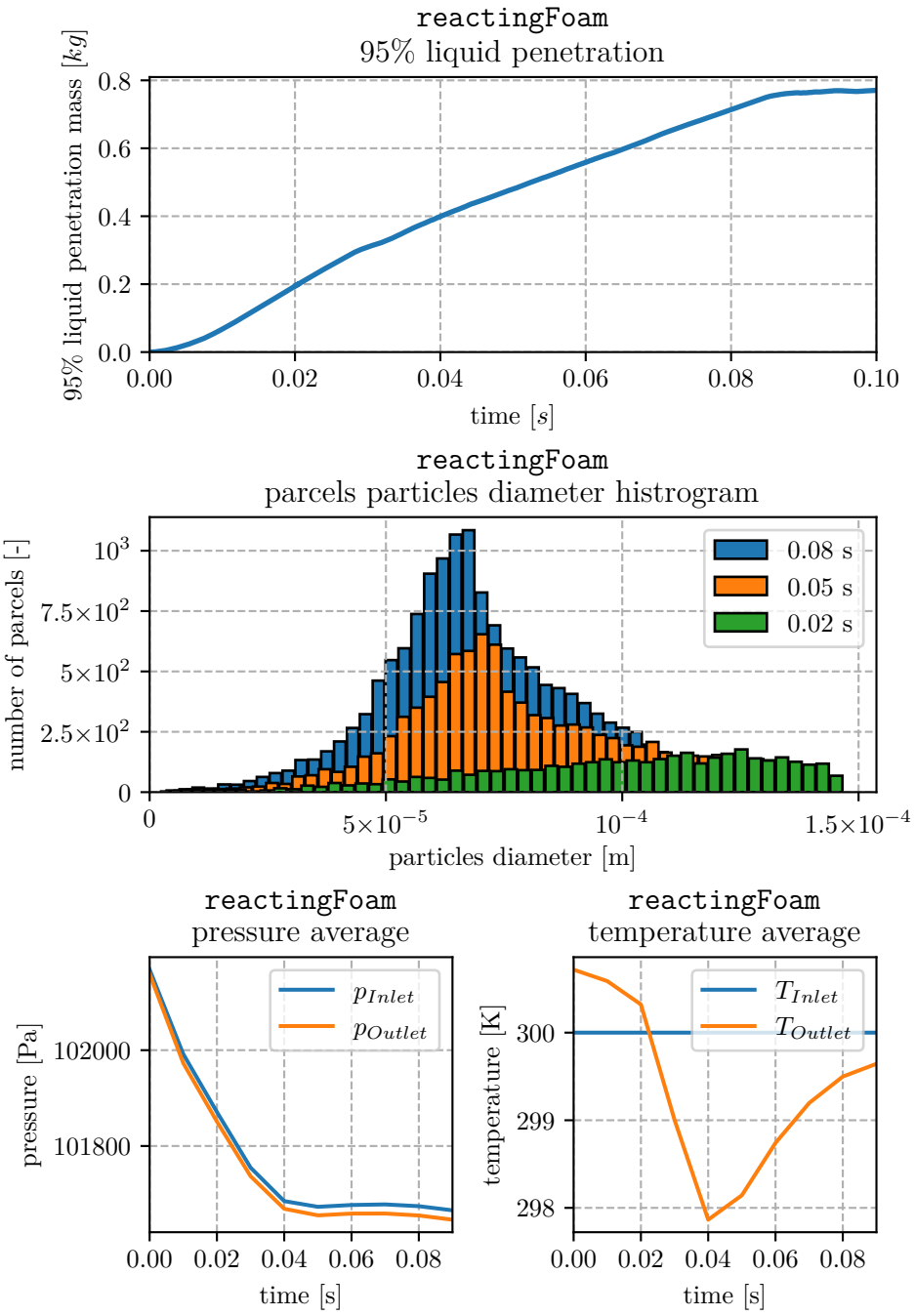

Figure 27: `reactingFoam` case: liquid penetration, parcels diameter and (p, T) average at inlet.

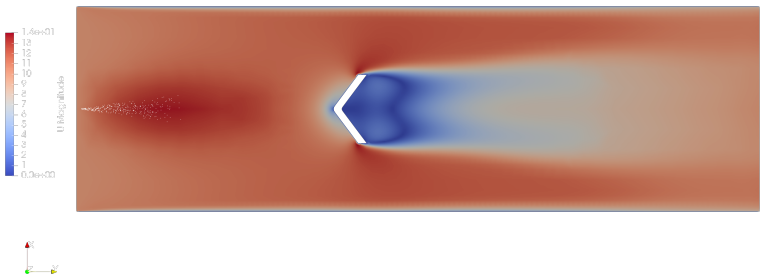

Figure 28: 3D reaction case: $|U|$ profile, $t = 0.02s$.

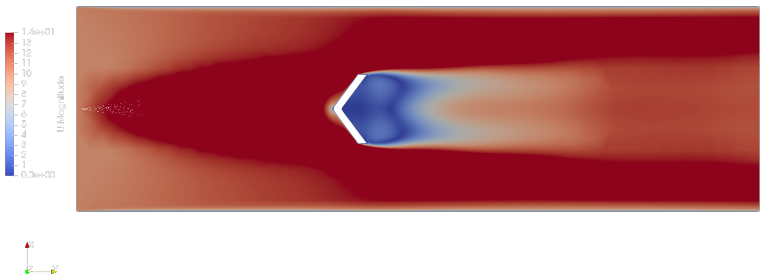

Figure 29: 3D reaction case: $|U|$ profile, $t = 0.05s$.

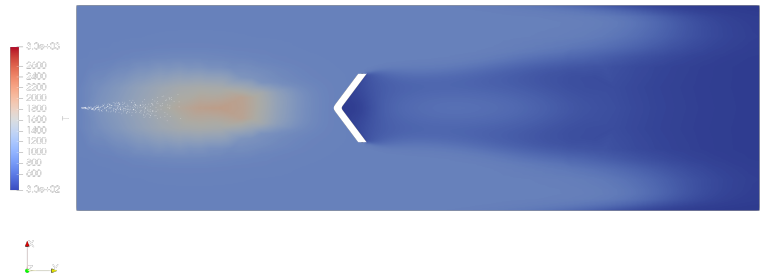

Figure 30: 3D reaction case: T profile, $t = 0.02s$.

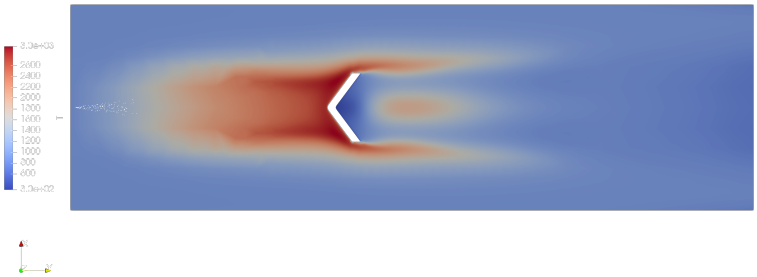

Figure 31: 3D reaction case: T profile, $t = 0.05s$.

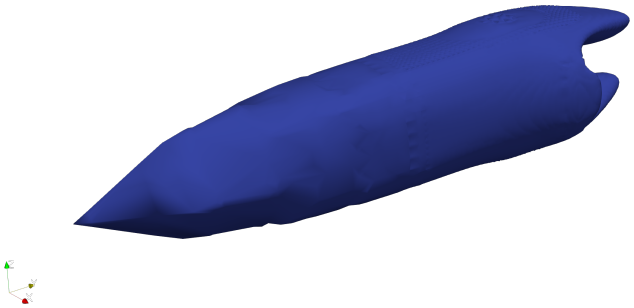

Figure 32: 3D reaction case: `A/F` iso-surface, $t = 0.05s$.
${C_7H_{16}}^{0.25} + 11{O_2}^{1.5} \rightarrow 7CO_2 + 8H_2O$
$\frac{A}{F} = \frac{O_2}{C_7H_{16}} \rightarrow surf\left(\frac{11 \cdot {O_2}^{1.5}}{{C_7H_{16}}^{0.25}}\right) = 1$

Appendices

A Mesh properties

A.1 Non orthogonality

The non orthogonality is a mesh **property** (Figure 33) based on the *vectorial* difference between the face normal versor $\boldsymbol{n}$ - at face middle point P - and the versor between the control volume and its neighbour cell, $\overline{C_1C_2}$. This property is expressed through θ_{no}: the closer to 0 the better. The non orthogonality importance is related to the `FVM` that bases the fluxes computation on vector $\boldsymbol{n}$.

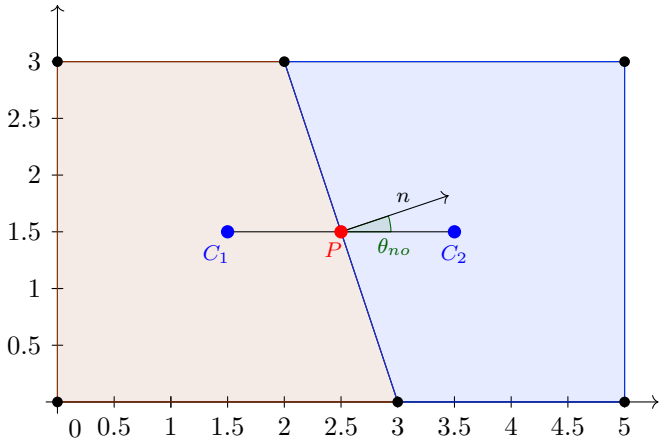

Figure 33: Non orthogonality.

There are many correctors for this topologic issue that are related on flux corrections. One of the main numeric operator that suffers a lot of bad non orthogonality values is the laplacian, $\boldsymbol{\Delta}$. One of the most important equation related to $\boldsymbol{\Delta}$ is the **Helmolts** equation - **Poisson** equation in the incompressible cases - that is about mass conservation (one of the most important principles to satisfy). Good non orthogonality values (in OpenFOAM syntax) are: $75 - 80$.

A.2 Skewness

As the non orthogonality, the skewness is a mesh **property** (Figure 34). This property is extremelly dependent on mesh topology and it has a very important effect on the simulation. This property is based on the computation of a quantity at face center. To compute this quantity, the `FVM` uses an interpolatory scheme that is based on control volumes' center distances. If the approximation of the field - and then the interpolation of fluxes at face center based on this field - is

wrong, the whole simulation will suffer of sources/sinks like presence in the domain that will bring to errors in the results and also to having bad effects on covergence rates.

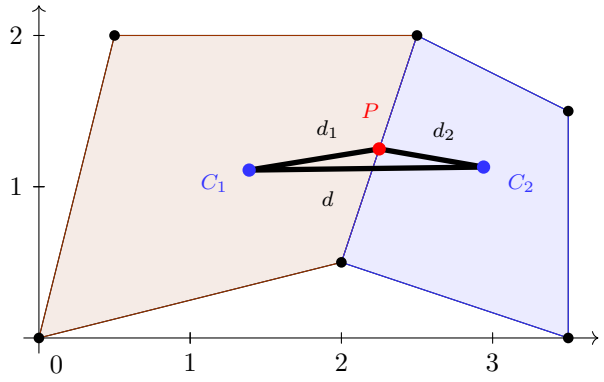

Figure 34: Skewness.

In contrast to the non orthogonality property, this property cannot be numerically **fixed**. As consequence, it is necessary to have meshes with the lowest skewness possible. It can happen that bad skewness brings to remeshing all the domain. Good skewness values (in OpenFOAM syntax) are: $1 - 8$.

B Incompressible solvers properties

B.1 t discretization in OpenFOAM

In `PISO/PIMPLE` the time advancing is made in order to discretize the time derivative. As result, the t advancing in the `PISO/PIMPLE` solver has a physical meaning. The contrary is for the `SIMPLE` solver where, in the `fvSchemes`, the time discretization is absent (Listing 5). In `SIMPLE`, the t advancement is essentially a controller for an additional loop over the predictor-corrector steps. As result, the previous step gives us a better estimate of $p^* - \boldsymbol{u}^*$ used for the numerical problem assembly.

Listing 5: `combustorSimple/system/fvSchemes` time discretization.

```
ddtSchemes  // time derivative discretization
    {       // no need of temporal discretization in SIMPLE algorithm
        default steadyState; // => no time stencil
    }
```

B.2 `PISO` vs `PIMPLE`

The main difference between these two models consists in the **outer** correction. `PISO` does not have outer correctors. The outer corrector loop consists in a correction of $p^* - \boldsymbol{u}^*$ used in the predictor step[29]. This allows, relating all the process to the $\boldsymbol{u}$ at the previous outer corrector step, to find a much more correct estimate of $\boldsymbol{u}$ at the new time step due to the fact that we have a better system formulation (based on $p - \boldsymbol{u}$ already estimated at the end of the previous predictor-corrector steps) that makes the problem treatment much like an **implicit** method. The **outer** correction step is very computational demanding but, at the same time, it is possible using Δt such that $Co > 1$. So choosing between `PISO` and `PIMPLE`, other than convergence and numerics, can be described with time step iterations[30].

B.2.1 Courant-Friedrichs-Lewy

In time marching problems, the CFL test allows to find a proper time step in order to link the **spatial** stencil with the **temporal** stencil, Figure 35.

As result, it is ensured that the $p-\boldsymbol{u}$ fields are physically related to the previous time steps. Of course, using $Co < 1$ does not guarantee achieving convergence;

[29] Solved if `momentumPredictor yes;` in `applications/solvers/incompressible/simpleFoam` with check on `simple.momentumPredictor()`.

[30] It is important to keep in mind that `pisoFoam` in OpenFOAM does not change the Δt. In order to solve this issue - so keeping $Co < 1$ and allowing Δt to change -, it is used running `pimpleFoam` in `PISO` mode. Of course, the number of outer corrector for `pimpleFoam` in `PISO` mode is 0.

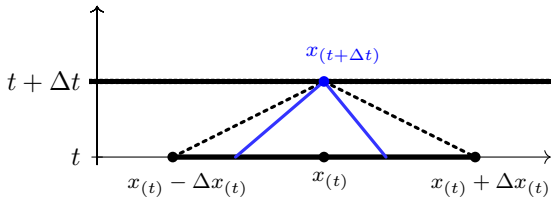

Figure 35: CFL, spatial and temporal stencils.

the convergence is related also to the discretization schemes and the correctors - so mainly, how the problem is solved -. The CFL condition is expressed as [1, Ch. 13]:

$$max_{\Omega}\left(\frac{\left|\boldsymbol{u}\right| \Delta t}{\Delta x}\right) < 1.0 \tag{1}$$

From (1), Δt is computed. There are methods, such as `PIMPLE`, that allow relaxation on this stencils constraint; this is mainly related to the way the solution is computed. $Co > 1$ acts as a filter on the solution: physics with time scales shorter than Δt are not seen in the final solution e.g. vortex shedding. In addition, for $Co > 1$ the time discretization error propagates on the solution; the last time step is the initial condition for the new time step. In OpenFOAM the Courant-Friedrichs-Lewy conditions is expressed through `maxCo`.

B.3 Under relaxation

The under relaxation consists in changing the value of a computed field in order to smooth out convergence. The under relaxation process followed in many codes is the **Patankar** model [2, Ch. 14.1], that consists in an explicit (2) and an implicit (3) formulation:

$$\phi_{j+1}^{relaxed} = \phi_j + \alpha_\phi \left(\phi_{j+1} - \phi_j\right) \tag{2}$$

$$\frac{a_C}{\lambda_\phi}\ \phi_{C_{j+1}} + \sum_F a_F\ \phi_{F_{j+1}} = b_C + \frac{1-\lambda_\phi}{\lambda_\phi}\ a_C\ \phi_{C_j} \tag{3}$$

This model relies on α_ϕ that is the relaxation parameter of the ϕ field. The most used α_ϕ values are: $\alpha_{\boldsymbol{u}} \approx 0.7$ and $\alpha_p \approx 0.3$. Now, having explained the relaxation model, it is necessary to relate it to the different algorithms. The under relaxation is not present in the `PISO` algorithm; this because the under relaxed field without outer corrector loop generates a not physical field, so it should not became an output of the solver. As this said, the `PISO` algorithm outputs directly the p field

and the corrected $\boldsymbol{u}$ field ($\boldsymbol{u} = \boldsymbol{u}_{(p,\ \boldsymbol{u}^*)}$). The other algorithms, `SIMPLE` and `PIMPLE`, allow using under relaxation. The under relaxation parameters are expressed in `system/fvSolution` with `relaxationFactors` and it can be explicit on a field with `fields` setting or implicit in the equations with `equations` setting.

C Compressible solvers properties

C.1 Energy equation

With respect to the incompressible case, the compressible case enables the changes in ρ of the fluid. This new degree of freedom in the system has to be treated with care using the energy equation. The energy equation has different forms that suit best for defined physical behaviours in the system[31]:

$$\frac{\partial \rho e}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u} e) + \frac{\partial \rho K}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u} K) + \boldsymbol{\nabla} \cdot (\boldsymbol{u} p) = -\boldsymbol{\nabla} \cdot \boldsymbol{q} + \boldsymbol{\nabla} \cdot (\boldsymbol{\tau} \cdot \boldsymbol{u}) + \rho r + \rho \boldsymbol{g} \cdot \boldsymbol{u} \tag{4}$$

$$\frac{\partial \rho h}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u} h) + \frac{\partial \rho K}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u} K) - \frac{\partial p}{\partial t} = -\boldsymbol{\nabla} \cdot \boldsymbol{q} + \boldsymbol{\nabla} \cdot (\boldsymbol{\tau} \cdot \boldsymbol{u}) + \rho r + \rho \boldsymbol{g} \cdot \boldsymbol{u} \tag{5}$$

$$\frac{\partial \rho E}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u} E) + \boldsymbol{\nabla} \cdot (\boldsymbol{u} p) = -\boldsymbol{\nabla} \cdot \boldsymbol{q} + \boldsymbol{\nabla} \cdot (\boldsymbol{\tau} \cdot \boldsymbol{u}) + \rho r + \rho \boldsymbol{g} \cdot \boldsymbol{u} \tag{6}$$

.

One of the most used energy equation is in the h formulation [4, Ch. 11]. The h formulation uses $h = e + \frac{p}{\rho}$ and K as *thermodynamic* and *mechanical* energy describers. In both $h - e$ formulations, K is always computed; instead in the total energy formulation, K is hidden in $E = e + K$. Most of the energy formulations in OpenFOAM neglect the mechanical part $\boldsymbol{\nabla} \cdot \left(\boldsymbol{\tau} \cdot \boldsymbol{u}\right)$ and $\rho \boldsymbol{g} \cdot \boldsymbol{u}$.

In the whole assignment, the compressibility is treated with the h formulation. The reason about this choice relies on the physics of the problem: for a problem with heat flux and with combustion of a non premixed mixture, the h formulation is preferred.

C.1.1 h equation in OpenFOAM

There are different ways for solving the compressible flow problems. The philosophy stays the same. One of the possible h formulations is described in `rhoPimpleFoam` and the main codes used in it are `pEqn.H`, `UEqn.H`, `EEqn.H` and `rhoPimpleFoam.C`.

The main parts of `rhoPimpleFoam.C` are the following:

- **Outer loop**
  - ρ **computation** - `rhoEqn.H` - Since ρ is a derived field, it needs to be computed.

[31] h is the specific enthalpy, e is the specific internal energy, K is the total kinetic energy and E is the total energy.

- $\boldsymbol{u}$ **predictor** - `UEqn.H` - Computation of a first $\boldsymbol{u}$ approximation.
- h **equation** - `EEqn.H`[32] - Computation of the $h-T$ field through guessed $\rho - p^*$ and the previous predicted $\boldsymbol{u}$.
- **Inner loop**
  - p **corrector** - `pEqn.H`[33] - Guarantee continuity with Helmolts equation.
    - $\frac{\partial p}{\partial t}$ **computation** Computation of $\frac{\partial p}{\partial t}$ needed in h formulation.
  - ρ **computation/correction** - `rhoEqn.H` - Correcting again ρ field with new p field and the T field from the h equation solution.

- **Convergence check**
- **Time step advancement**
  - **Guessed fields for the new time step** The new time step gets the guessed values from the last time step solutions.

C.1.2 ψ and ρ based model

In CFD there are many ways for coupling the energy, pressure, velocity equations and the equation of state. All these ways for the computation of ρ can be separated in 2 families: the pressure based ψ equation and the ρ based equations.

ψ The ψ based equations uses an explicit relation between ρ and p through ψ. An example is the ψ equation that is based on the equation of the perfect gases, $\psi = \frac{1}{RT}$ such that $\rho = \psi \cdot p$. This family of models are best suitable for the segregated (semi-implicit) method because it is a p **based** algorithm.

ρ The ρ based equations are another set of equations that allow the computation of ρ through a ρ dependent PDE or through an equation that does not explicitly relate ρ to p such as the **Boussinesque approximation** $\Delta\rho = \beta\Delta T$.

[32] `EEqn.H` uses `dpdt`. `dpdt` is computed in `pEqn.H` with `dpdt = fvc::ddt(p)`.

[33] The Helmoltz equation can be seen as a Poisson equation with a $\frac{\partial \rho}{\partial t}$ term that is converted in p terms and then treated implicitly.

D Spray modeling

D.1 Spray regions description

Sprays are largely used in the combustion field. The aim is to let the fuel evaporate in the shortest time possible to reach the best performances. Engineers have divided the spray in 3 regions and one of the main describers can be the void fraction[34]:

- **Dense region** In this region, there is the first break up of the jet core in many string shaped drops. The fuel has a low surface to volume ratio, this implies the fuel is a dense and confined region in the system. This region has to be treated with care and it is very computationally demanding. The models required to solve this region are called **continuous phase models**, `CPM`. The void fraction for this region is around unity.

- **Diluted region** In this region, the dense region starts to break up into smaller pieces; these pieces have a larger surface to volume ratio but not high enough to be modelled as spheric particles. This region is very important to lagrangian particles tracking methods because it is the region of transition between dense to dispersed region. This region can be treated as well with `CPM`. The final aim of the `CPM` in this case is to get a good description of the spray properties for the dispersed region modeling. The void fraction is around 0.5 - this because $\frac{V_{liquid}}{V_{gas}} \approx 1$.

- **Dispersed region** This region consists of many dispersed particles where the surface to volume ratio is high enough to consider these particles as spheres. This **huge** simplification allows using the lagragian spray method for the modeling of the development of the spray in the dispersed region[35]. In the following CFD cases, lagrangian particles models are used to discretize the spray evolution (dispersed region only) into the system. As result of dispersed fuel into the system, the void fraction is lower than 1.

D.2 Parcels in spray modeling

The number of particles in the dispersed region can be of order of billions; this will bring a lot of computational effort to model all the particles into the system and also to store all the data related to particles' properties, position and dynamics.

[34] $\alpha = \frac{V_{liquid}}{V_{cell}}$

[35] It is possible to model the dispersed region with `CPM`. Using `CPM` in the dispersed region means not considering the particles as spheres. `CPM` in dispersed region is very computational demanding and very difficult to handle.

In order to solve this problem, lagrangian sprays adopted a new way of modeling for these particles. All these particles are grouped into objects called parcels. The grouping process is related to the particles' diameter. If a particle has a certain diameter, this particle will be grouped with other particles that fall inside a defined diameter range. Particles' diameter is defined by experimental results, by results analysis of dense and diluted region simulations and by statistics. After the grouping process, every parcel in the system has defined properties. The lagrangian spray modeling will compute the parcels evolution into the system.

D.2.1 Parcels interaction

Since parcels represent the properties of group of particles, the evolution of parcels represents the evolution of those particles. Parcels and gaseous flow can interact in different ways. This depends on the nature of the spray, on parcels' properties and on computational power available. It is possible to model the spray as:

- **One way coupling** The one way coupling allows only a *one way talk* from the continous phase to the discrete phase. This allows to compute the parcels motion directly from the continuous phase field. Of course, parcel motion depends on the forces applied to it; these forces are computed using forces equations applied to spherical shaped particels with the help of experimental results. The forces' models depend on particles diameter - expressed through parcel object -, the particles diameter on the other hand depends on the thermodynamics (because of evaporation).

- **Two ways coupling** The two way coupling allows the *talk back* of the discrete phase evolution to the continous phase. This happens through additional source terms in the continuous phase equations. As result, this method is more computational demanding[36].

- **Four ways coupling** The four ways coupling accounts also to the parcels interaction. Usually this method is used when $\alpha \geq 10^{-2}$. The continuous phase treatment remains almost the same as the two ways coupling but the parcels interaction has an additional modeling step that takes into account the changes parcels make into the continuous field because their distance and also the parcels interaction as parcels agglomeration and parcels break-up.

[36] All the modeling aspects for the parcel's evolution are the same as the one way coupling

E Wall-film modeling

The particles-wall interaction has to be treated by the simulation. This interaction is very important because it can change a lot the behaviour of the solution. Wall-film in OpenFOAM are mainly treated using these hypothesis:

- **Wall-film thickness hypthesis** The wall-will thickness is supposed to be very thin; this brings to study the liquid film evolution as a 2D problem
- **Particles temperature change** The temperature change of the particles impinging the wall-film is slow; this allows direct analytical time integration istead of numerical integration
- **Conductivity hypothesis** Since the wall-film region is a layer above the solid wall, heat is transmitted from the wall to the liquid film through conductive mechanism
- **Temperature limits hypothesis** Due to modeling reasons, the wall and wall-film temerature is supposed to be lower than the liquid boiling temperature

E.1 Mesh and boundary conditions

Because the wall-film region can be seen as an additional model to the usual `FVM` case, a mesh has to be built in order to study the wall-film region. The only purpose of the wall-film mesh is to track the development of a 3D wall-film using a 2D mesh. The wall-film has to talk back to the *global* mesh in order to set up correct boundary conditions for the main flow. As result of this modeling choices, the main parameters that describe the wall-film region are:

- δ Liquid film height. Since the mesh is 2D, the wall film height is described by δ but at the same time δ describes the presence of the liquid film on the wall-film region surface
- $\boldsymbol{u}$ Liquid film velocity
- T Liquid film temperature

Transport equations (7) are used for tracking these quantities:

$$\begin{cases} \frac{\partial \rho \delta}{\partial t} + \boldsymbol{\nabla} \cdot \left(\rho \ \delta \ \boldsymbol{u} \right) & = \boldsymbol{S}_{\delta} \\ \frac{\partial \rho \delta \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot \left(\rho \ \delta \boldsymbol{u} \ \boldsymbol{u} \right) & = -\delta \ \boldsymbol{\nabla} p + \boldsymbol{S}_{\delta \boldsymbol{u}} \\ \frac{\partial \rho \delta h}{\partial t} + \boldsymbol{\nabla} \cdot \left(\rho \ \delta h \ \boldsymbol{u} \right) & = \boldsymbol{S}_{\delta h} \end{cases} \tag{7}$$

F Reactions

Before starting modeling reactions it is necessary to know the species inside the system, the possible reactions that can happen and all the parameters related to transport and diffusion of these species into the system. All these properties are mainly taken from experiments or they come from chemistry modeling (for the more complicated species).

F.1 Govering equations

The main changes in the governing equation are made on the continuity equation. The continuity equation expresses the conservation of mass in the system. Because the presence of reactions, it is needed to track the species' evolution in the system. The continuity equations for each species are written as:

$$\frac{\partial \rho Y_i}{\partial t} + \boldsymbol{\nabla} \cdot \left(\rho \boldsymbol{u}\ Y_i\right) = \boldsymbol{\nabla} \cdot \left(\rho \boldsymbol{v}_i\ Y_i\right) + \dot{\omega}_i \tag{8}$$

Where Y_i is the mass fraction of *ith* species, $\boldsymbol{v}_i$ is the *ith* species diffusion velocity[37] and $\dot{\omega}_i$ is the *ith* species production due to reactions[38]. As result, $\boldsymbol{v}_i$ are new degree of freedom into the system, so it is necessary to find equations that allow to treat them. The complete equation for the $\boldsymbol{v}_i$ computation is very hard to solve[39], so it is necessary to simplify this equation. Possible formulations are the **Fick**'s law, equation (9), and the **Hirshfelder** model, equation (10).

$$\boldsymbol{v}_i = -\mathcal{D} \boldsymbol{\nabla} \log Y_p \tag{9}$$

$$\boldsymbol{v}_i\ X_k = \mathcal{D}_k \boldsymbol{\nabla} X_k \tag{10}$$

All these simplified submodels imply a cost, mass conservation is not guaranteed. In order to conserve the mass, it is necessary to correct these fields through:

- Reducing the number of equations for the species from N to $N-1$ using the mass conservation formula $\sum_i Y_i = 1$ for the computation of the *Nth* species. This procedure can work but only if the Y_N is high enough to neglect these modeling errors.
- A much better solution can be achieved changing the fluctuation field $\boldsymbol{v}_k$ in order to conserve total mass for all the species.

[37] $\sum_i \boldsymbol{\nabla} \cdot \left(\rho Y_i \boldsymbol{v}_i\right) = 0$, where $\boldsymbol{v}_i$ is the species velocity fluctuation, $\sum_i Y_i = 1$ and $\boldsymbol{\nabla} \cdot \left(\rho \boldsymbol{u}\right) = 0$. $\sum_i \boldsymbol{\nabla} \cdot \left[Y_i \rho \left(\boldsymbol{u} + \boldsymbol{v}_i\right)\right] = \sum_i \boldsymbol{\nabla} \cdot \left(Y_i \rho \boldsymbol{u}\right) + \sum_i \boldsymbol{\nabla} \cdot \left(Y_i \rho \boldsymbol{v}_i\right) = \boldsymbol{\nabla} \cdot \left(\rho \boldsymbol{u} \sum_i Y_i\right) + \sum_i \boldsymbol{\nabla} \cdot \left(Y_i \rho \boldsymbol{v}_i\right) = 0$.

[38] $\sum_i \dot{\omega}_i = 0$ because the reaction process does not create mass, it just *moves* atoms in order to create new species, conserving mass.

[39] $\boldsymbol{v}_p$ equation: $\boldsymbol{\nabla} X_p = \sum_k \frac{X_p\ X_k}{\mathcal{D}_{p,k}} \left(\boldsymbol{v}_k - \boldsymbol{v}_p\right) + \left(Y_p - Y_k\right) \frac{\boldsymbol{\nabla} p}{p} + \frac{\rho}{p}\ \sum_k Y_p\ Y_k \left(f_p - f_k\right)$.

For the remaining governing equations, the main changes are relative the new source terms in the energy equation[40]. The momentum equation is affected most through the $\nu_{eff} = \nu + \nu_t$ due the dependence of viscosity and turbulence characteristics on the flow composition.

F.2 Chemistry

Having introduced the way reactions talk with the *main* flow governing equations, it is necessary to study the chemistry modeling in the system. This part is essential because it allows studing:

- **Combustion presence** If the energy activation threshold is overcomed, reaction takes place.
- **Chemical kinetics** Allows to treat the species evolution - $\dot{\omega}_i$ in mass conservation equation -, control volume cell species composition - for ν_{eff} assembly in momentum equation.
- **Chemical thermodynamics** Allows to treat heat transfered, h_{reac}, in energy equation.

Most of the chemistry models are based on the **Arrhenius** law:

$$k = B\ T_a\ e^{-\frac{E_a}{R_u\ T}} \tag{11}$$

The equation (11) is a semi-empirical law[41] and it is used to determine the chemical kinetics of a reaction like the following:

$$\frac{\partial [C]}{\partial t} = (k_{forward} - k_{backward})\ [A]\ [B];\ \text{for the reaction}^{42}\ A + B \leftrightarrow C \tag{12}$$

Equation (12) is an ODE that can be solved in an implicit or explicit way. Due to stiffness of equation (12) - related to the exponential nature of k - an implicit formulation allows using less timesteps to reach the *main* flow time step lenght but at the same time it is more computationally expensive. On the other hand, the explicit formulation in much more direct that the implicit one but it needs more iterative steps to reach the *main* flow iterative time step lenght.

As result, chemistry can be seen as a computational burden for the simulation because it is related to small time steps due to the chemistry ODE stiffness properties.

[40] The source term sign depends on the nature of the reaction.

[41] B is the Boltzman constant, $B\ T_a$ is the collision frequency and E_a is the energy activation for the reaction.

[42] The probability of having a reaction made by the interaction of more than three species is extremelly low.

Bibliography

[1] Alfio Quarteroni and Silvia Quarteroni. *Numerical models for differential problems*. Vol. 8. Springer, 2012.

[2] Fadl Moukalled, L. Mangani, Marwan Darwish, et al. *The finite volume method in computational fluid dynamics*. Vol. 113. Springer, 2016.

[3] Stephen B. Pope. *Turbulent flows*. Cambridge university press, 2000.

[4] Joel H. Ferziger, Milovan Perić, and Robert L. Street. *Computational methods for fluid dynamics*. Vol. 4. Springer, 2002.

[5] Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II*. Vol. 375. Springer Berlin Heidelberg, 1996.