ECOGEN Documentation

Release b1.0

code-mphi

Sep 27, 2019

CONTENTS:

Here is described the documentation of the ECOGEN open source plateform dedicated to numerical simulation of compressible multiphase flows.

ECOGEN is an open source project distributed under under GNU GPLv3 License and available on Git-Hub.

It is developped by code-mphi members in partenrship with AMU, Caltech and CNES.

In this document, ECOGEN's user will find all necessary information to use the software.

Interested developpers are encouraged to contribute in the project and can use the API documentation generated by Doxygen.

CHAPTER ONE

INTRODUCTION

ECOGEN is a CFD plateform written in C++ object oriented programming langage. It is dedicated to numerical simulation of compressible multiphase flows. It has the vocation to share academics researches in the multiphase flow field in direction to ohter academics but also for industrials, students, etc.

- multi-models (single phase, multiphase with or without equilibrium)
- multi-physics (thermal transfers, viscosity, surface tension, mass transfers)
- multi-meshes (Cartesian, unstructured, AMR)
- multi-CPU

ECOGEN stands for:

- **E**volutive: makes easier future developpements
- **C**ompressible: dedicated to compressible flows
- **O**penSource: Distributed under GPLv3 Licence
- **G**enuine: Uses the "Diffuse Interface Method" (DIM)
- **E**asy: simple to install and use (C++ compiler and MPI)
- **N**-phase: liquids, vapors, inert gases and/or reactives

1.1 What kind of physical problems ?

ECOGEN is designed for following applications:

- Solving interface problems between pure or multicomponents fluids and mixtures of multiphase flows.
- Treating surface tension, heat and mass transfers for evaporating and condensing flow, cavitation.
- Computing wave propagation in strongly unsteady situations using a specific Adaptative Mesh Reffinement .
- Computing on unstructured grids to simulate complex geometries.
- Parallel computing using open MPI libraries.

1.2 What about the engine ?

ECOGEN is a receptacle of a story of diffuse interface method (DIM) theory that started in the late 90s. DIM summarizes more than 20 years of researches on multiphase flow modelling with the objective to develop mathematical models as well as their associated numerical methods.

What is a diffuse interface? In DIM theory, the interfaces between pure phases are captured as diffuse numerical zones meaning that one goes continuously from one phase to another.

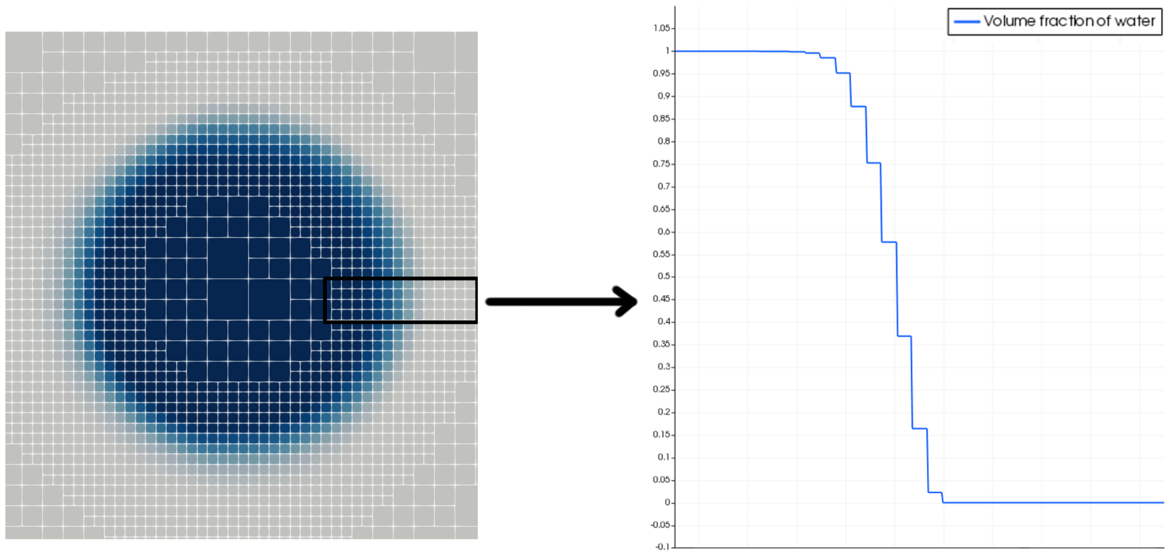


Fig. 1.1: 1D extraction of the diffuse interface zone of a water droplet.

This way is possible thanks to a thermodynamical consistency. Then, the flow solution does no longer requires interface tracking algorithms: It became easy to simulate complex topological shape evolutions between miscible or non miscible fluid. Moreover, pressure waves (shock waves, acoustic waves) can propagate and interact properly in the whole flow.

The basic research on DIM is now matured enough to propose ECOGEN, a numerical tool that can be largely cast and use to solve industrial as well as research multiphase flow problems.

CHAPTER

TWO

INSTALLATION INSTRUCTIONS

This section contains basic instruction for ECOGEN installation on windows / ubuntu systems.

2.1 Prerequisities

ECOGEN must be compiled with C++. It also requires a functional system implementation of MPI library (not provided in this package). Depending on your operating system, you can follow the instructions below to set a full open source installation:

2.1.1 Installing prerequisities on Ubuntu system

ECOGEN required two mandatory components to be installed on your Ubuntu system : a C++ compiler and an effective implementation of MPI.

Installing C++ compiler

Nothing is more easy than installing C and C++ compiler on Ubuntu. In your terminal just enter the following commands:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install gcc
sudo apt-get install g++
sudo apt-get install build-essential
```

More information on the ubuntu doc page https://doc.ubuntu-fr.org/gcc

Installing openMPI

Dowload the latest stable version of openMPI under compressed format. At the time this page is written, it corresponds to the compressed file : openmpi-4.0.1.tar.gz. Uncompresse and move into the directory:

```
tar -xvf openmpi-4.0.1.tar.gz
cd openmpi-4.0.1/
```

Prepare the environnement for using your favorite compiler:

```
export CC=gcc
export CXX=g++
```

Configure and proceed to the installation (you can choose a different directory). The "make" step should take some times (coffee time ?):

```
./configure --prefix=/opt/openmpi
make
sudo make install
```

Cleaning

```
cd ..
rm -rf openmpi-4.0.1/
```

Modify the /etc/bash.bashrc by adding the line:

export PATH=/opt/openmpi/bin:$PATH

Then source the file to take into consideration the modifications:

```
source /etc/bash.bashrc
```

If the installation succeed you should use the mpicxx command in your terminal. Then proceed to the download step below.

2.2 Download

The last ECOGEN version can be downloaded from Git-hub. The source files are available at the following address: https://github.com/code-mphi/ECOGEN.

The package includes:

- ECOGEN/src/ folder including C++ source files.
- ECOGEN/libMeshes/ folder including examples of unstructured meshes in *.geo* format (gmsh files version 2). See section *Generating Meshes* for details.
- ECOGEN/libEOS/ folder including some possible parameters for Equation of State in XML files. See section *Materials* for details.
- ECOGEN/libTests folder including:
  - ECOGEN/libTests/referenceTestCases/ folder organized in a test cases library according the flow model (Euler Equations ECOGEN solver, Kapila's model for multiphase flow ECOGEN solver, Homogeneous Euler Equation ECOGEN solver, etc.). A detailed list of available test cases is proposed in section *Test cases*.
  - 4 quick-manual XML files to create a new flow calculation with ECOGEN.
- *ECOGEN.xml* main entry file to select running cases.
- *Makefile*: for compilation in Unix environment. This file may require some adaptation to the user's environment.
- *LICENSE*, *COPYRIGHT* and *AUTHORS*: Information files about authors and licensing.
- *README.md*: Information file.
- *ECOGEN_documentation.pdf*: The full documentation for ECOGEN.

2.3 Compilation/Execution on bash

Use the Makefile (can be adapted if necessary) to compile ECOGEN sources directly on bash (XX is the number of CPU required for compilation):

```
make -j XX
```

Executing ECOGEN is really easy on bash (XX is the number of CPU required for execution):

```
mpirun -np XX ECOGEN
```

2.4 Testing

Once preceding compiling of the code succeed, the better way to test ECOGEN's installation is to run successively the two simple following commands:

```
./ECOGEN
mpirun -np 2 ECOGEN
```

This will run the default test case included in the package two times:

- In sequential (single CPU).
- In parallele using 2 CPU.

This should print informations in the terminal on the running default test case. If no error message appears, then your installation should be OK. You should use ECOGEN for your own applications.

ECOGEN is including a given number of simple prebuild test cases. Each test can be used as a basis for a new one. Visit the tutorial section *Tutorials* for more informations.

CHAPTER

THREE

I/O DESCRIPTIONS

ECOGEN settings are managed via INPUT FILES only. The global INPUT FILES and OUTPUT FILES structure is depicted in Fig. 3.1.

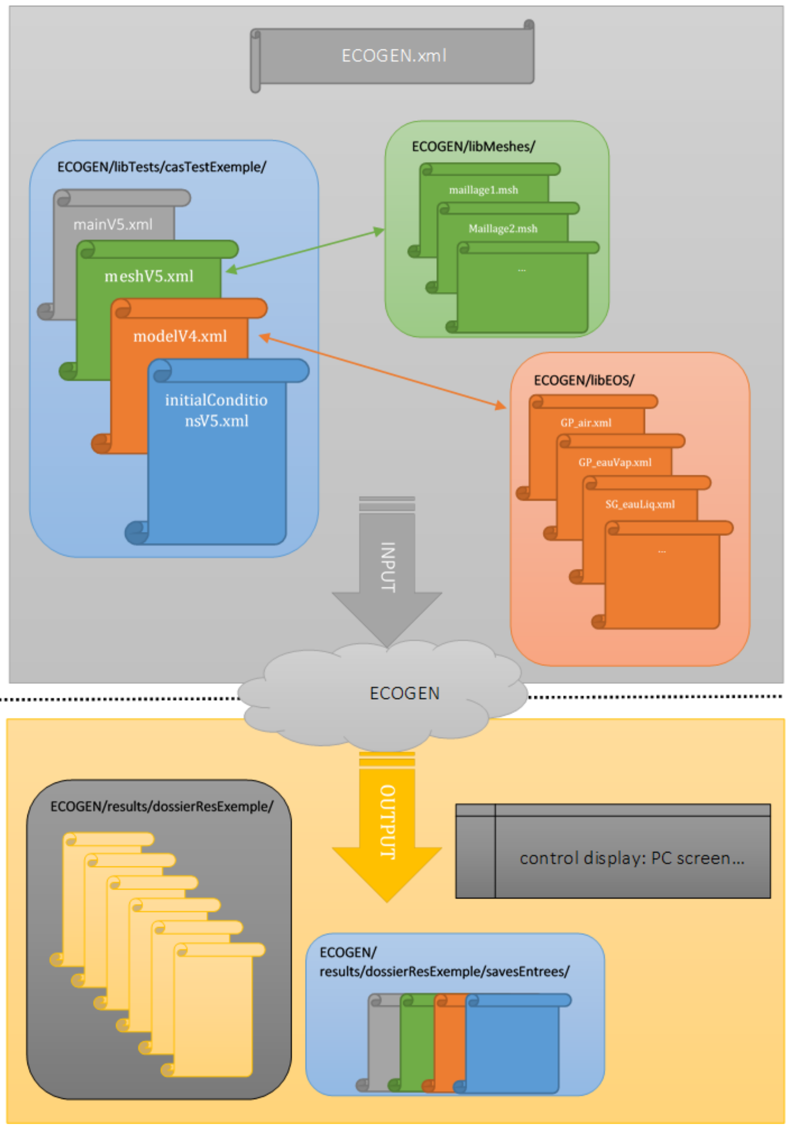


Fig. 3.1: Structure of input and output files in ECOGEN

3.1 Input Files

The standard XML format is used for ECOGEN input files. This data format is using markups that gives to the input files some interesting flexibility. ECOGEN is using the TinyXML-2 parser to read/write xml files.

More information about XML format can be found at: https://www.w3schools.com/xml/

ECOGEN package includes a sample of test cases. Each of them is independent, ready to use and can also be adapted and enriched to develop a new configuration. The test case to be run is chosen in the main input file: *ECOGEN.xml*. The minima structure of this file is:

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<ecogen>
  <testCase>./libTests/referenceTestCases/euler/1D/transport/positiveVelocity/</
↪testCase>
</ecogen>
```

In this file, the `<testCase>` markup indicates the folder containing the test case to be run. It is then possible to run successively several cases by adding as many `<testCase>` markup as necessary. Each folder indicated in a `<testCase>` markup must contain 4 input files:

- *mainV5.xml*
- *meshV5.xml*
- *modelV4.xml*
- *initialConditionsV4.xml*

Additional input files depending on the test case are necessary. They are placed in the following folders:

- **ECOGEN/libEOS/**: contains the files defining the material used in the test case.
- **ECOGEN/libMeshes/**: contains the files defining the mesh used in the test case.

In this section the structure of each input file is detailed. This information is also provided in a condensed form in the "handbook" files at the root folder of test cases library **ECOGEN/libTests/**. These files are named and constitute quick-reference manuals:

- *manualMainV5.xml*
- *manualMeshV5.xml*
- *manualModelV4.xml*
- *manualInitialConditionsV4.xml*

3.1.1 MainV5.xml

mainV5.xml is the main input file needed to define the test case. It **must be** in the current test case folder. Its minimal structure is:

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<computationParam>
  <run>example</run>
  <outputMode format="XML" binary="false" precision="10"/>
  <timeControlMode iterations="false">
    <iterations number="3" iterFreq="1"/>
    <physicalTime totalTime="8.e-3" timeFreq="8.e-4"/>
  </timeControlMode>
```

(continues on next page)

(continued from previous page)

```
  <computationControl CFL="0.8"/>
</computationParam>
```

It contains general parameters for the computation listed below. Some are mandatory, others are optional.

Run name

```
<run>example</run>
```

The `<run>` markup is mandatory. It will be used to create the folder containing the test case results in **ECOGEN/results/**.

Output format

```
<outputMode format="XML" binary="false" precision="10"/>
```

The `<outputMode>` markup is mandatory. The user can choose the writing output format. Attributes are:

- `format`: can take the value *GNU* (standard writing in column) or *XML* (XML VTK format).
- `binary`: can take the value true or false. *Binary* (true) or *ASCII* (false) format can be chosen.

Output results files will be placed in the folder **ECOGEN/results/** into a specific subfolder with the name of the run.

Time evolution control

```
<timeControlMode iterations="false">
  <iterations number="3" iterFreq="1"/>
  <physicalTime totalTime="8.e-3" timeFreq="8.e-4"/>
</timeControlMode>
```

ECOGEN is a CFD tool based on an explicit integration scheme in time. The `<timeControlMode>` markup is mandatory and defines the temporal evolution of the current simulation. it contains the `iterations` attribute that can take the two values:

- *true*: the time control is done thanks to the total number of timesteps and the <iterations> node must be present.
- *false*: the time control is done thanks to the physical final time and the <physicalTime> node must be present.

The `<iterations>` markup:

```
<iterations number="3" iterFreq="1"/>
```

ECOGEN automatically computes the timestep value thanks to a numerical stability criterion (CFL criterion). This markup is defined with following attributes:

- `number`: Integer equals to the total number of temporal timesteps.
- `iterFreq`: Integer equal to the frequency of results writing (results are written every iterFeq timestep)

The `<physicalTime>` markup:

```
<physicalTime totalTime="8.e-3" timeFreq="8.e-4"/>
```

If this markup is used ECOGEN automatically determines the total amount of timestep to compute to reach the chosen physical time. Attributes are:

- `totalTime`: Real number equals to the physical final time of the simulation. (unit: s (SI)).
- `timeFreq`: Real number equals to the frequency of results writing (results are written every timeFreq seconds).

CFL criterion

```
<computationControl CFL="0.8"/>
```

The `<computationControl>` markup is mandatory. It specifies the value of the attribute `CFL` which ensures the stability of the temporal integration scheme: the value (real number) must be less than 1.

Global accuracy order of the numerical scheme

When it is possible (according to the mesh or to the flow model) ECOGEN can use a second-order scheme (based on MUSCL approach with a TVD slope limiter). In this case the optional markup `<secondOrder>` can be inserted in the *mainV5.xml* input file as in the following example:

```
<secondOrder>
  <globalLimiter>minmod</globalLimiter>
  <interfaceLimiter>superbee</interfaceLimiter>                           <!--␣
↪optionnal node -->
  <globalVolumeFractionLimiter>minmod</globalVolumeFractionLimiter>       <!--␣
↪optionnal node -->
  <interfaceVolumeFractionLimiter>thinc</interfaceVolumeFractionLimiter>  <!--␣
↪optionnal node -->
</secondOrder>
```

The `<secondOrder>` markup must contain the node `<globalLimiter>`. The other nodes are optional. The slope limiters available in ECOGEN are the following: minmod [Roe86], vanleer [VL74], vanalbada [VAVLR97], mc [VL77], superbee [Roe86]. Markups significations are:

- `<globalLimiter>`: applied everywhere and on all variables unless it is overwrite by the following optional limiters.
- `<interfaceLimiter>`: applied on all variables but only at the interface location. By default is equal to the global limiter.
- `<globalVolumeFractionLimiter>`: applied everywhere but only on the volume-fraction and transport equations (THINC is only applied on the volume fractions) unless it is overwrite by the interface volume-fraction limiter. By default is equal to the global limiter.
- `<interfaceVolumeFractionLimiter>`: applied only at the interface location and on the volume-fraction and transport equations (THINC [SX14] is only applied on the volume fractions). By default is equal to the interface limiter.

Probes

It is possible to record flow variables at given locations in the computational domain against time. This is done by including to the *mainV5.xml* input file the optional `<probe>` markup.

```
<probe name="capteur1">
  <vertex x="0.51" y="0.51" z="0.51"/>
  <timeControl acqFreq="-1."/>        <!-- if negative or nul, recording at each time␣
↪step -->
</probe>
```

The two following nodes must be included in the `<probe>` markup: - `<vertex>`: Specifies the location of the probe into the computational domain in each physical direction corresponding to the attributes: `x`, `y` and `z` (unit: m (SI)). Values must be real numbers. - `<timeControl>`: permits to specify the probe acquisition frequency (unit: s (SI)). If the value is set to zero or negative, flow values at the probe location are recorded at each time step.

Probes output results files will be placed in the specific subfolder **ECOGEN/results/XXX/probes/** where *XXX* represent the name of the run specified in `<run>` markup.

Remarks:

1. Recording probe with a high frequency could have a significant impact on computation performances due to the computer memory time access. To prevent that, one should fix a reasonable acquisition frequency.
2. Several probes can be added simultaneously. For that, place as many as wanted `<probe>` markups in the *mainV5.xml* input files.

3.1.2 ModelV4.xml

Fluid mechanics models used in the computation are specified in *modelv4.xml* file. It is **mandatory** located in the folder of the current case. A typical form of this file is:

```xml
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<model>
  <flowModel name="Kapila" numberPhases="2" alphaNull="false"/>
  <EOS name="IG_air.xml"/>
  <EOS name="SG_water.xml"/>
</model>
```

Flow model

```xml
<flowModel name="Kapila" numberPhases="2" numberTransports="1" alphaNull="false"/>
```

The `<flowModel>` markup is **mandatory** to specify the mathematical model to solve during the computation. This markup may contains the following attributes:

- `name`: name of the mathematical flow model. This attribute can take the values: *Euler*, *Kapila*, *multip*, *ThermalEq* or *EulerHomogeneous*.
- `numberPhases`: Integer number corresponding to the number of phases present in the simulations. The total amount of equations is related to this number. This attribute is not necessary for the values of name: *Euler*, *EulerHomogeneous*.
- `numberTransports`: this attribute is optionnal and is set to 0 as default. It can be used to add specific variable advected in the flow (color function).
- `alphaNull`: For *Kapila*'s model, the volume fraction can either be null or not and this choice is determined by the parameter alphaNull. default value is *false*.

Remark:

if *EulerHomogeneous* model is chosen, two additional attributes may be used: `liquid` and `vapor` to specify the number corresponding to the liquid phase and the vapor phase. It is phase 0 (for the first) or 1 (for the second).

```xml
<flowModel name="EulerHomogeneous" liquid="0" vapor="1"/>
<EOS name="SG_waterLiq.xml"/>
<EOS name="IG_waterVap.xml"/>
```

Equations of state (EOS)

```
<EOS name="IG_air.xml"/>
```

The *modelV4.xml* input fle **must contain** as many `<EOS>` markups as many number of phases specified in the *Flow model* markup. Each phase is described thanks to relations and parameters. The values of these parameters are specified in a separate file: the attribute name contains the name of this file that must be placed in the folder **ECOGEN/libEOS/**. Some fluid files are already present in the ECOGEN package.

Advected additional variables

```
<transport name="color"/>
```

The *modelV4.xml* input fle **must contain** as many `<transport>` markups as many number of transport specified in the *Flow model* markup. Each transported variable is described by its name. The default number of advected variable is 0.

Relaxation procedures

```
<relaxation type="PT"/>
```

An additional markup `<relaxation>` may be used to impose some specific equilibrium between the phases depending on the flow model used. The attribute `type` specifies the kind of equilibrium:

- *P*: a pressure equilibrium is imposed at every location of the flow. It does not require additional attributes.
- *PT*: Both pressure and thermal equilibrium are imposed at every location of the flow. It does not require additional attributes.
- *PTMu*: a thermodynamical equilibrium is imposed at every location of the flow. It must be associated to the node `<dataPTMu>` with attributes `liquid` and `vapor` to specify the name of the EOS of the liquid and the vapor phase. Hereafter the complete node when PTMu is used:

```
<relaxation type="PTMu">
  <dataPTMu liquid="SG_waterLiq.xml" vapor="IG_waterVap.xml"/>
</relaxation>
```

Source terms

The additional `<sourceTerms>` markup can be used to numerically integrate some source terms in the equations. The attribute `type` selects the source term:

- *heating*: related to a thermal energy heating/cooling. This attribute requires the `<dataHeating>` node with the attribute `volumeHeatPower`: a real number corresponding to the power by volume unit added to the flow (unit: W/m3 (SI)).

```
<sourceTerms type="heating">
  <dataHeating volumeHeatPower="1.e6"/>
</sourceTerms>
```

- *gravity*: if the gravity is considered. The node `<dataGravity>` with the following attributes must be present with the attributes `x`, `y` et `z` giving the coordinates for the gravity acceleration vector in real numbers (unit: m/s2 (SI))

```
<sourceTerms type="gravity">
  <gravity x="0." y="-9.81" z="0."/>
</sourceTerms>
```

- *MRF*: for a simulation in the moving reference frame. Allow to compute solution in a rotating frame. The node `<omega>` requires the attributes `x`, `y` et `z` giving the coordinates for the rotating vector in real numbers (unit: rad/s (SI)). The node `<timeToOmega>` is optional and allow to specify a progressing acceleration (linear) to the final rotating velocity (requires the attribute `tf` for acceleration time).

```
<sourceTerms type="MRF">
  <omega x="0." y="0." z="1."/>
  <timeToOmega tf="1.e-3"/>  <!-- Optional: If activated, the angular velocity␣
↪increase linearly to omega in during tf -->
</sourceTerms>
```

Symmetry terms

Both cylindrical (2D) and spherical (1D) symmetries are implemented. The additional `<symmetryTerm>` markup can be used. It requires the attribute `type` that can take the value *cylindrical* ar *spherical*. It also requires an additional node depending on the symmetry terms:

- cylindrical:

```
<symmetryTerm type="cylindrical">
  <dataSymCyl radialAxe="X"/>
</symmetryTerm>
```

- spherical:

```
<symmetryTerm type="spherical">
  <dataSymSpher radialAxe="X"/>
</symmetryTerm>
```

Additional physics (dev)

Depending on the model chosen in section *Flow model*, tension surface effects can be added. This is the case for surface tension, viscosity and conductive heat transfers. These additional physical effects are obtained thanks to the additional markup `additionalPhysics` with the attribute `type` that can take different value according to the chosen effect.

Surface tension

This physical effect is obtained by using the type *surface tension*. Then it requires the node `dataSurfaceTension` with following attributes:

- `transport`: this is the name of advected variable used as color function for surface-tension terms. This advected variable has been precised in the section *Advected additional variables*. The name should be the same.
- `sigma`: a real number for the surface-tension coefficient in N/m.

```
<additionalPhysic type="surfaceTension" >
  <dataSurfaceTension transport="color" sigma="72.e-3"/>
</additionalPhysic>
```

Others

in dev...

3.1.3 MeshV5.xml

Input file *meshV5.xml* is necessary to specify the geometrical characteristics of the computational domain and the kind of mesh used. This file is **mandatory** and must be present in the folder of the current case. The minimalist content of this file is:

```xml
<?xml version="1.0" encoding="UTF-8" standalone = "yes"?>
<mesh>
  <type structure="cartesian"/>
  <cartesianMesh>
    <dimensions x="1.e-1" y="5.e-2" z="1."/>
    <numberCells x="50" y ="25" z="1"/>
  </cartesianMesh>
</mesh>
```

The `<type>` markup is mandatory and specifies via the attribute `structure` the kind of mesh used in the current computation:

- *cartesian*: ECOGEN automatically generates its own Cartesian mesh. See section *Cartesian mesh* for details.
- *unstructured*: a specific external mesh generator (not provide with ECOGEN) must be used to generate the mesh. See section *Unstructured mesh* for details.

Cartesian mesh

```xml
<cartesianMesh>
  <dimensions x="1.e-1" y="5.e-2" z="1."/>
  <numberCells x="50" y ="25" z="1"/>
</cartesianMesh>
```

ECOGEN is able to automatically generate a cartesian mesh according the markup `<cartesianMesh>` which must contain the two following nodes:

- `<dimensions>`: Specifies the physical dimensions of the computational domain in each physical direction corresponding to the attributes: `x`, `y` and `z` (unit: m (SI)). Values must be real numbers.
- `<numberCells>`: Specify the number of cells in each direction corresponding to the `x`, `y` and `z` attributes. The values are integer numbers.

Optional Stretching

```xml
<meshStretching>    <!-- Optionnal node -->
  <XStretching>
    <stretch startAt="0." endAt="0.5" factor="0.9" numberCells="20"/>
    <stretch startAt="0.5" endAt="1." factor="1.1" numberCells="10"/>
  </XStretching>
</meshStretching>
```

Stretching can be set optionally adding the `<XStretching>` node to the `<cartesianMesh>` parent markup for X stretching. It should contain one or more `<stretch>` node(s) equipped with the following attributes:

- `startAt`: real number giving the beginning position of the stretched zone.
- `endAt`: real number giving the ending position of the stretched zone.
- `factor`: real number for the stretch factor (lower than 1 for shrinking, greater than 1 for stretching).
- `numberCells`: integer for the cell number attributed to the stretched zone.

Remark:

1. Stretching can be set in each directions using `<YStretching>` and `<ZStretching>`.
2. For each stretched direction, the sum of stretched zones should exactly recover the entire domain without overlaping, but the number of cell can differ than those precised in the `<numberCells>` initial markup.
3. A particular attention should be paid to the linking between zones that possibily present a bad quality.

Optional AMR

```
<AMR lvlMax="2" criteriaVar="0.2" varRho="true" varP="true" varU="false" varAlpha=
↪"false" xiSplit="0.11" xiJoin="0.11"/> <!-- Optionnal node -->
```

An efficient Adaptive Mesh refinement (AMR) technology is embedded in ECOGEN. To do that *meshV5.xml* file must content the optional node `<AMR>` of the `<cartesianMesh>` markup to define the following attributes:

- `lvlMax`: integer to define the maximal number of refinements.
- `criteriaVar`: real number that controls the detection of gradients for the location of the refinement.
- `varRho`, `varP`, `varU`, `varAlpha`: boolean (*true* or *false*) to select the flow quantity on which the gradient operator is applied to detect large gradient.
- `xiSplit`, `xiJoin`: normalized real numbers to control if a computational cell, selected by its high gradient value, must be refined or un-refined (values are in the range *0-1.*).

Remark:

The global efficiency of the method is greatly depending on the chosen values for the `criteriaVar`, `xiSplit` and `xiJoin` attributes. These values depend on the physical problem and required a real *know-how*. More details about these criterion values can be found in [SPD19].

Unstructured mesh

```
<unstructuredMesh>
  <file name="unstructured2D/testUS.msh"/>
  <parallel GMSHPretraitement="true"/>  <!-- Optionnal node if multiCPU -->
</unstructuredMesh>
```

When dealing with unstructured meshes, the `<unstructuredMesh>` markup **must be** present in the *meshV5.xml* input file and contains the following nodes:

- `<file>`: this **mandatory** node specifies the path of the mesh file via the attribute `name`. The file must be located in the folder **ECOGEN/libMeshes/**.
- **`<modeParallele>`** [This node is required only if the file mesh is a multi-CPU file. The attribute `GMSHPretraitement` can take the following values:]
  - *true*: ECOGEN automatically splits the given mesh file in as many as necessary files according to the number of available CPUs.
  - *false*: do not redo the split of the given mesh (which has already been split in a precedent simulation).

Remarks:

1. The attribute `GMSHPretraitement` must be set as true if it is the first run with the given mesh file.
2. In the current version of ECOGEN, only mesh files generated with the opensource Gmsh software under file format *version 2* can be used.

Please refer to the section *Generating Meshes* for learning how to generate a mesh adapted to ECOGEN.

3.1.4 InitialConditionsV4.xml

The *initialConditionsV4.xml* input file includes the initial conditions and the boundary conditions of the flow simulation. It is **mandatory** located in the folder of the current case. The typical structure of this file is:

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<CI>
  <!-- LIST OF GEOMETRICAL DOMAINS  -->
  <physicalDomains>
    <domain name="base" state="leftChamber" type="entireDomain"/>
  </physicalDomains>

  <!-- LIST OF BOUNDARY CONDITIONS -->
  <boundaryConditions>
    <boundCond name="CLXm" type="abs" number="1"/>
    <boundCond name="CLXp" type="abs" number="2"/>
  </boundaryConditions>

  <!--  LIST OF STATES  -->
  <state name="leftChamber">
    <material type="fluide" EOS="IG_air.xml">
      <dataFluid alpha="0.5" density="1.0"/>
    </material>
    <material type="fluide" EOS="SG_water.xml">
      <dataFluid alpha="0.5" density="1000.0"/>
    </material>
    <mixture>
      <dataMix pressure = "1.e5"/>
      <velocity x="0." y="0." z="0."/>
    </mixture>
    <transport name="color" value="32."/>
  </state>
</CI>
```

Physical domains

The `<physicalDomains>` markup is mandatory. Some different initial conditions can be specified at different zones of the computational domain. This markup must contain as many nodes `<domain>` as necessary to correctly initialize the computational domain (overlaps are possible). A `<domain>` node contains the following attributes:

- `name`: a name for the domain. This name has no influence on the choices remaining in this file.
- `state`: This is the state of the fluid which will be specified with the `<state>` markup further in the file.
- `type`: to specify the kind of geometrical domain on which the state of the fluid must be attribute: *entireDomain*, *halfSpace*, *disc*, *Rectangle*, *Pavement*, *sphere*.

Important remark:

The initial conditions are attributed on each domain by using a superposition principle. The order is important: in the case of overlapping, the last attributed data are considered in the flow computation. Hence, it is important to attribute at least the entire domain at the first-place thanks to the value entireDomain.

According to the geometrical shape, additional information is required thanks to the use of the nodes among the list:

entireDomain

Set the initial condition on the entire domain. No more information required.

```
<domain name="base" state="leftChamber" type="entireDomain"/>
```

halfSpace

Set the initial condition on a half-domain. The node `<dataHalfSpace>` must be included with the following attributes:

- `axe`: can take the value *x*, *y* or *z*.
- `origin`: real number, indicates the location of the edge between the two subdomains on the specified axis.
- `direction`: can take the value positive or negative on the specified axis.

```
<domain name="HP"  state="rightChamber" type="halfSpace">
  <dataHalfSpace axe="x" origin="0.5" direction="positive"/>
</domain>
```

disc

In 2D allows to define a disc on a plane, in 3D a cylinder with an infinite length is defined. The node `<dataDisc>` must be added with the following attributes:

- Attributes `axe1`, `axe2`: The name of 2 axes to define the plane on which the disc is defined. Can take two different values among *x*, *y* or *z*.
- Attribute `radius`: Real number of the radius disc (unit: m (SI)).
- Node `<center>`: requires the attributes `x`, `y` et `z` giving the location of the center of the disc in the plan (axe1, axe2) in real numbers (unit: m (SI)).

```
<domain name="HP"  state="rightChamber" type="disc">
  <dataDisc axe1="x" axe2="y" radius="0.5">
    <center x="0." y="0." z="0."/>
  </dataDisc>
</domain>
```

Rectangle

In 2D allows to define a rectangle on a plane, in 3D a rectangular beam with an infinite length is defined. The node `< dataRectangle >` must be added with the following attributes:

- Attributes `axe1`, `axe2`: The name of 2 axes to define the plane on which the disc is defined. Can take two different values among *x*, *y* or *z*.
- Attributes `lAxe1`, `lAxe2`: Length of both sides along (axe1,axe2).
- Node `<posInferiorVertex>`: equipped with the attributes `x`, `y` and `z`, real numbers giving the location of the inferior corner in the plane (axe1, axe2).

```
<domain name="HP"  state="rightChamber" type="rectangle">
  <dataRectangle axe1="x" axe2="y" lAxe1="0.3" lAxe2="0.2">
    <posInferiorVertex x="0.4" y="0.5" z="0."/>
  </dataRectangle>
</domain>
```

Pavement

Set the initial condition in a pavement. The additional node `<dataPavement>` must be added with the attributes:

- Attributes `lAxeX`, `lAxeY`, `lAxeZ`: Real numbers for length of each side of the pavement along axes (unit: m (SI)).
- Node `<posInferiorVertex>`: with the des attributes `x`, `y` and `z`, real numbers corresponding to the location of the inferior corner (unit: m (SI)).

```
<domain name="HP"  state="rightChamber" type="pavement">
  <dataPavement lAxeX="1." lAxeY="1." lAxeZ="0.5">
    <posInferiorVertex x="1." y="0.5" z="0.5"/>
  </dataPavement>
</domain>
```

sphere

Set the initial condition in a sphere. The additional node `<dataSphere>` is required with the attributes or nodes:

- Attribute radius: real number giving the radius of the sphere (unit: m (SI)).
- Node `<center>`: with the attributes `x`, `y` et `z` real numbers giving the ocation on the ceter of the sphere (unit: m (SI)).

```
<domain name="HP"  state="rightChamber" type="sphere">
  <dataSphere radius="0.5">
    <center x="1." y="0.5" z="0.5"/>
  </dataSphere>
</domain>
```

Initializing using physical Identity

Additional feature for geometrical domain: It is possible to use physicalIdentity number coming from mesh software to initialize a geometrical domain.

Example:

```
<domain name="base" state="leftChamber" type="entireDomain" physicalEntity="10"/>
```

In this example the entire computation domain will be initialize accordingly to the physicalEntity 10.

Boundary conditions

The `<boundaryConditions>` markup is mandatory. The boundary conditions are specified at the boundary of the computational domain. This markup must contain as many nodes `<boundCond>` as necessary to recover the entire boundary. Each `<boundCond>` node contains the following attributes:

- name: a name for the boundary condition. This name has no influence on the choices remaining in this file.
- type: the kind of boundary condition, to choose among *abs*, *wall*, *tank*, *outflow*
- numero: integer number that correspond to the number of the boundary.

According `<type>`, additional information is required thanks the use of the nodes among the list:

abs

The numerical treatment corresponds to an outgoing flow with no wave reflection. No more information required.

```
<boundCond name="exit" type="abs" number="1" />
```

wall

The numerical treatment corresponds to a wall boundary condition. No more information required.

```
<boundCond name="wall" type="wall" number="3" />
```

tank

The numerical treatment corresponds to the link between the boundary with an infinite tank. An infinite tank is characterized by a null velocity while pressure and temperature are constant). `<tank>` requires the `<dataTank>` node with the following attributes:

- `p0`: Stagnation pressure, real number (unit: Pa(SI)).
- `T0`: Stagnation pressure, real number (unit: K (SI)).
- **An additional `<fluidsProp>` node is necessary to define the presence of each phase in the tank. It must contain as many**

  - `EOS`: the name of the file corresponding to the choice of the EOS for the phase in the tank. This file must correspond to the one specified in modelV4.xml input file for every fluid.
  - `alpha`: The volume fraction of the fluid in the tank, real number in the range]0.,1.[

```
<boundCond name="entrance" type="tank" number="3">
  <dataTank p0="4.e6" T0="93.3"/>
  <fluidsProp>
    <dataFluid EOS="IG_oxyVap.xml" alpha="0.0001"/>
    <dataFluid EOS="SG_oxyLiq.xml" alpha="0.9999"/>
  </fluidsProp>
</boundCond>
```

outflow

In the case of a subsonic flow, the pressure is set equal to the ambient pressure at the boundary. The additional <dataOutflow> node is required with the attributes:

- p0: outside pressure, real number (unit: Pa(SI)).

```
<boundCond name="exit" type="outflow" number="5">
  <dataOutflow p0="1.e5">
    <transport name="color" value="1.e-6"/>
  </dataOutflow>
</boundCond>
```

Important remark

The choice of the boundary condition number is made according to the kind of mesh given in meshV5.xml input file according to the following rules:

- **cartesian: The boundaries are ordered and labeled from 1 to 6 (in 3D) according to:**
    1. boundary condition at the minimal x location
    2. boundary condition at the maximal x location
    3. boundary condition at the minimal y location
    4. boundary condition at the maximal y location
    5. boundary condition at the minimal z location
    6. boundary condition at the maximal z location
- unStructured: When an unstructured mesh is used, the number of the boundary condition must correspond to the number specified in the mesh file .geo (see example in section *Generating Meshes*).

Remark

The boundary conditions are dependent on the flow model specified in modelV4.xml input file. Some boundary conditions may be not available for the flow model considered.

Mechanical and thermodynamical states of the fluid

For each physical domain in the <physicalDomains> markup, a fluid state must correspond. It implies an additional <state> markup for each state of fluid. This <state> markup contains:

- as many <material> nodes as the number of phases involved in the simulation.
- A <mixture> node is required if a multiphase model is used.

Each <material> node corresponds to a phase and contains the following attributes or nodes:

- Attribute type: Only the value *fluide* is available in the current ECOGEN version.
- Attribute EOS: the name of the file corresponding to the fluid Equation of State parameters. This file must correspond to the one specified in *modelV4.xml* input file for each phase (see section *Flow model*).
- Node <dataFluid>: contains data related to the considered state of the fluid in the current phase.

This last node <dataFluid> as well as the <mixture> node are dependent on the flow model according to:

Euler

Single phase flow. In this case, the <mixture> node is absent and the <dataFluid> node contains the following attributes or nodes:

- Attribute temperature: Initial temperature of the fluid, real number (unit: K (SI)).
- Attribute pressure: Initial pressure of the fluid, real number (unit: Pa(SI)).
- Node <velocity>: with x, y and z attributes setting the initial values for the components of the velocity vector, real numbers (unit: m/s (SI)).

```
<material type="fluide" EOS="IG_air.xml">
  <dataFluid density="10.0" pressure="1.e5">
    <velocity x="1000." y="1000." z="0."/>
  </dataFluid>
</material>
```

Kapila

Multiphase flow at pressure and velocity equilibrium (same velocity and same pressure for every phase). Each <dataFluid> node corresponds to a phase with the following attributes:

- alpha: Volume fraction of the phase, real number in the range]0.,1.[.
- density: Initial specific mass of the fluid, real number (unit: kg/m3 (SI)) or temperature: Initial temperature (unit: K).

Moreover, in this case, the <mixture> node contains:

- the <dataMix> node with pressure attribute for initial pressure of the fluid, real number (unit: Pa(SI)).
- the <velocity> node with x, y and z attributes setting the initial values for the components of the velocity vector, real number (unit: m/s (SI)).

```
<material type="fluide" EOS="IG_air.xml">
  <dataFluid alpha="0.5" density="1.0"/>
</material>
<material type="fluide" EOS="SG_water.xml">
  <dataFluid alpha="0.5" density="1000.0"/>
</material>
<mixture>
  <dataMix pressure = "1.e5"/>
  <velocity x="0." y="0." z="0."/>
</mixture>
```

ThermalEq

Multiphase flow at pressure, velocity and thermal equilibrium (same velocity, same pressure and same temperature for every phase). In this case, every <dataFluid> node corresponds to a phase with only one attribute alpha setting the volume fraction real number in the range]0.,1.[.

The <mixture> node contains the following attributes and nodes:

- the <dataMix> node with temperature and pressure attributes for initial temperature of the fluid, real number (unit: K (SI)) and initial pressure, real number (unit: Pa).
- Attribute pressure: Initial pressure of the mixture, real number (unit: Pa(SI)).

- Node `<velocity>`: with `x`, `y` and `z` attributes setting the initial values for the components of the velocity vector of the mixture, real numbers (unit : m/s (SI)).

```
<material type="fluide" EOS="IG_waterVap.xml">
  <dataFluid alpha="0.2"/>
</material>
<material type="fluide" EOS="SG_waterLiq.xml">
  <dataFluid alpha="0.8"/>
</material>
<mixture>
  <dataMix pressure = "1.e5" temperature ="300."/>
  <velocity x="0." y="0." z="0."/>
</mixture>
```

EulerHomogeneous

Multiphase flow at mechanical and thermodynamical equilibrium. In this case, every `<dataFluid>` node corresponds to a phase with only one attribute `alpha` setting the volume fraction real number in the range]0.,1.[.

Moreover, in this case, the `<mixture>` contains the following attributes and nodes:

- the `<dataMix>` node with `pressure` attribute for initial pressure of the mixture, real number (unit: Pa(SI)).
- Node `<velocity>`: with `x`, `y` and `z` attributes setting the initial values for the components of the velocity vector of the mixture, real numbers (unit: m/s (SI)).

```
<material type="fluide" EOS="SG_waterLiq.xml">
  <dataFluid alpha="0.99"/>
</material>
<material type="fluide" EOS="IG_waterVap.xml">
  <dataFluid alpha="0.01"/>
</material>
<mixture>
  <dataMix pressure = "1.e6"/>
  <velocity x="0." y="0." z="0."/>
</mixture>
```

Remark

Be careful to set the volume fraction in the range]0,.1.[as well as the sum over the phases equal to 1.

3.2 Materials

For each phase, an Equation of State is required. The data for a given phase are gathered in a file. Every file must be in the folder **ECOGEN/libEOS/** and must follow rules depending on the EOS. Two equations of states are implemented in ECOGEN:

- *Ideal gas*: for gaseous phase only.
- *Stiffened gas*: for condensed matter (liquid, solid) in a pressure range where the compressible assumption is reasonable.

3.2.1 Ideal Gas

It is a simple relation linking the pressure (p), the temperature (T) and the density (ρ) or the specific volume ($v = 1/$):

$$pv = rT$$

r is the universal gas constant divided by the molar weight of the gas (unit: $J/(K.kg)$ (SI)). The Gibbs relation as well as Maxwell relations yield the definition of the internal energy:

$$e(p,v) = \frac{p}{\gamma - 1} v + e_{ref}$$

One can easily obtain the entropy:

$$s(p,T) = c_v \ln\left(\frac{T^\gamma}{p^{\gamma-1}}\right) + s_{ref}$$

The following parameters are assumed constant and γ, c_v, e_{ref} and s_{ref} must be specified in the material file, as in the following example :

```xml
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<parametersEOS>
        <EOS fichier="IG_air.xml" type="IG"/>
        <parameters
                gamma="1.4"
                cv="800" cv_save="717.46"
                energyRef="0."
                entropyRef="0.">
        </parameters>
        <physicalParameters
                mu="1.85e-5"
                lambda="0.0262">
        </physicalParameters>
</parametersEOS>
```

3.2.2 Stiffened Gas

This Equation of state was first presented in 1971 by Harlow & Amdsen [HA68]:

$$e = \left(\frac{p + \gamma p_\infty}{\gamma - 1}\right) v + e_{ref}$$

This EOS takes into account for the molecular attraction within condensed matter. This quite simple EOS is largely used in numerical simulation based on diffuse interface methods because it is able to reproduce shock relation as well as saturation curve for a liquid-vapor mixture when the phase transition is modeled. Hereafter some useful relations for the specific volume, the enthalpy and the entropy:

$$v(p,T) = \frac{(\gamma - 1)c_v T}{p + p_\infty}$$

$$v(p,T) = \frac{(\gamma - 1)c_v T}{p + p_\infty}$$

$$h(T) = \gamma c_v T + e_{ref}$$

$$s(p,T) = c_v \ln\left(\frac{T^\gamma}{(p + p_\infty)^{\gamma-1}}\right) + s_{ref}$$

A usefull description of these relations can be found in [LeMetayerMS04].

The ideal gas law is recovered if $p_\infty = 0$. The following parameters are assumed constant γ, p_∞, c_v, e_{ref} and s_{ref}. These parameters must be specified in the material file, as in the following example:

```xml
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<parametersEOS>
        <EOS fichier="SG_water.xml" type="SG"/>
        <parameters
                gamma="4.4"
                pInf="6.e8"
                cv="1000.0"
                energyRef="0."
                entropyRef="0.">
        </parameters>
        <physicalParameters
                mu="1.e-3"
                lambda="0.6">
        </physicalParameters>
</parametersEOS>
```

3.3 Output Files

Writing the result files is done according the user's choice in *mainV5.xml* INPUT FILE of the current test (see section *MainV5.xml*). For each run, the results are recorded in the specified folder in **ECOGEN/results/XXX/** where XX is the test case *name*. One can select the following format:

- *GNU*: Format in ASCII, results are given in column.
- *XML*: VTK file format (ASCII or binary).

The name of the results files follows the rules:

result(format)_CPU(proc)_AMR(level)_TIME(instant). (ext)

that can select results files according:

- (format) : data format (empty for ASCII, B64 for binary).
- (instant) : time of writing results (depends on the selected frequency for writing.
- (level) : AMR level (in the case of an AMR simulation).
- (proc) : the number of the processor where the results are from (in the case of a parallel simulation).
- (ext) : kind of mesh.

3.3.1 Using GNU format

This format lead to results files with the (ext)=out extension. This format in colon is very useful for a quick visualization of the results when the freeware gnuplot or any other tool. When this format is selected, ECOGEN automatically creates a script file visualisation.gnu at the root of the result folder. This allows a very quick and efficient use for 1D runs.

3.3.2 XML VTK file format

ECOGEN can provide output using XML files in VTK file format (ASCII or BINARY). Writing files according VTK file format leads to files with the extension:

- (ext)=vtr : cartesian mesh.
- (ext)=vtu : unstructured mesh.

ECOGEN also produces a file named *collection.pvd* in order to load only one pack of files when the software PARAVIEW is used.

3.3.3 Saving input files

In addition to the results, a copy of the INPUT FILES of the current simulation is done in the subfolder **ECOGEN/results/dossierResExemple/savesEntrees/** to ensure a safe reproduction of the results.

3.3.4 Screen output

In real time, some data are available during the simulation directly at the terminal screen as soon as ECOGEN starts. hereafter an example of a screenshot:

One reads:

- number of the results files.
- number of the last timestep.
- Real time in the simulation.
- Value of the last timestep.
- CPU time since the beginning of the simulation.

CHAPTER

FOUR

TUTORIALS

Here are presented a collection of tutorials to help using efficiently ECOGEN. One should began with the section *Start with ECOGEN* when first use of ECOGEN.

4.1 Start with ECOGEN

Here is described your first use of ECOGEN: running the *default test case*. This is what user should observe without any change in the downloaded ECOGEN package.

Important: Before beginning with ECOGEN, you should have succeed all steps of installation instructions of section *Installation instructions*.

4.1.1 The conductor input file

ECOGEN is mainly controled thanks to the input file named *ECOGEN.xml*. This file looks like :

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes"?>
<ecogen>
  <!-- Euler reference test cases -->
  <!-- ---------------------------- -->
  <testCase>./libTests/referenceTestCases/euler/1D/transport/positiveVelocity/</
↪testCase>
  <!-- <testCase>./libTests/referenceTestCases/euler/1D/transport/negativeVelocity/</
↪testCase> -->
  <!-- <testCase>./libTests/referenceTestCases/euler/1D/shockTubes/HPLeft/</testCase>␣
↪-->
</ecogen>
```

Each test case corresponds to specific input files organised in different folders and associated to a markup `<testCase>`.

When executing ECOGEN, it will run each test case corresponding to the uncommented lines present in the markup `<ecogen>`.

A unique line is uncommented in the original file and corresponds to the *default test case*. One should modify the *ECOGEN.xml* input file to run other provided test by uncommenting / commenting lines in this file. New lines can also be added when creating new test cases.

4.1.2 Running the default test case

The default test case provided with ECOGEN package is a single flow test which simply advect a density discontinuity with a positive velocity in 1D. Input files for this test case are present in the folder *./libTests/referenceTestCases/euler/1D/transport/positiveVelocity/*

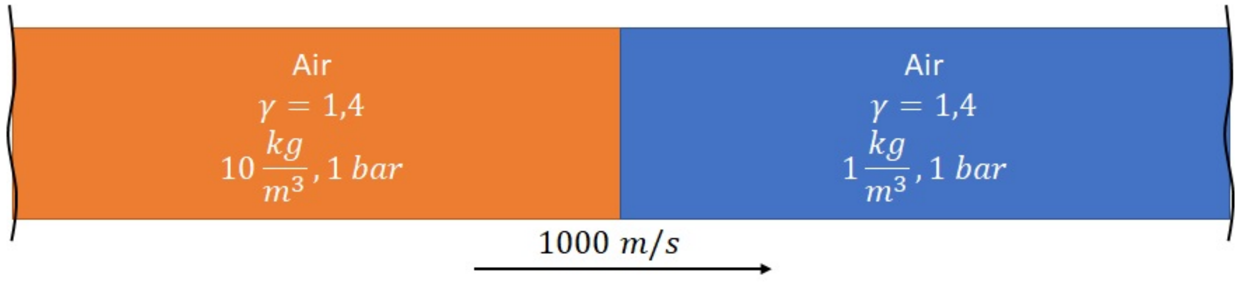


Fig. 4.1: Initial condition for single phase 1D transport test case.

The initial characteristics of the run are:

Characteristic	value
dimension	1 m
mesh size	100
AMR max level	3
discontinuity position	0,5 m
boundary conditions	transmittive
final solution time	0.36 ms
solution printing frequency	0.036 ms

This test can be executed on single CPU or on XX CPU by one of the commands:

```
./ECOGEN
mpirun -np XX ECOGEN
```

Note: Informations on available CPU can be obtained under linux system using the command:

```
/usr/bin/nproc
```

The code is running and at the top of the console output one can read :

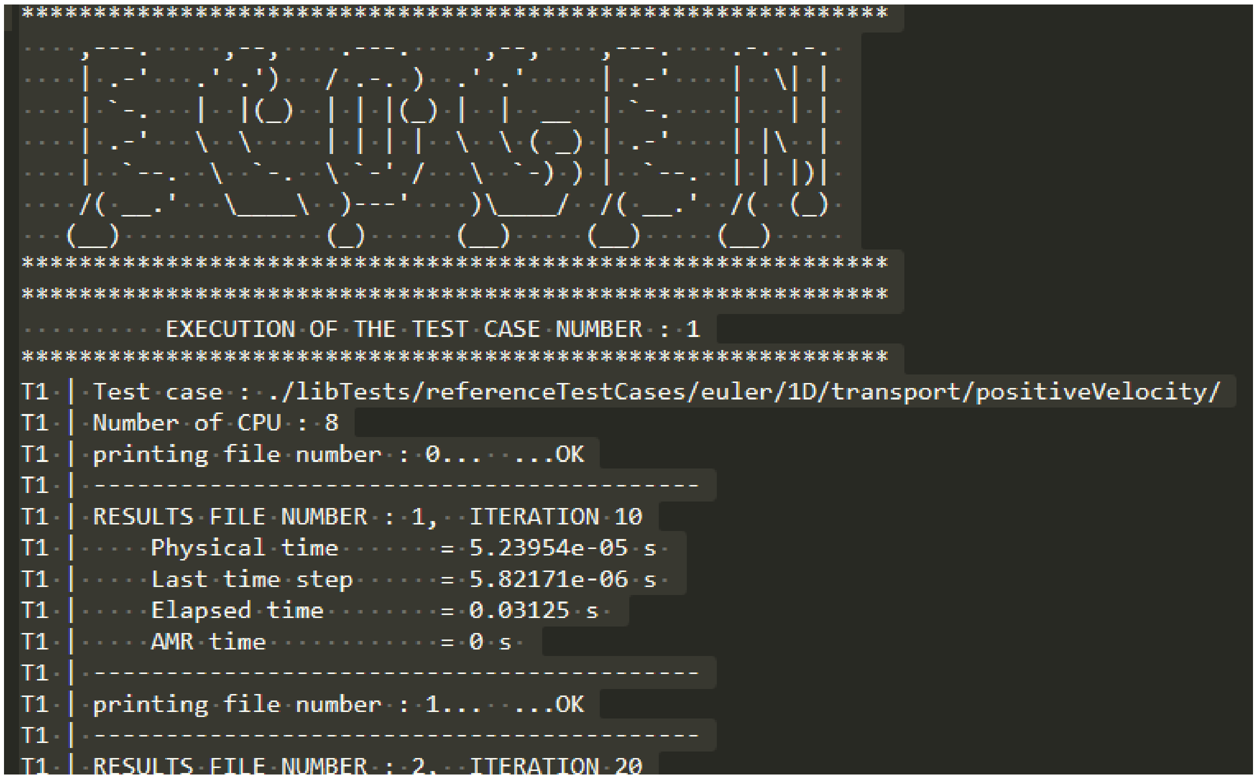
- The console logo of ECOGEN
- The name of the test case including the full path of the test case : *./libTests/referenceTestCases/euler/1D/transport/positiveVelocity/*
- Information concerning the number of iterations, the elapsed time…

euler1DTransportPositiveVelocity is the actual name of the default run.

The code ends and the following information comes:

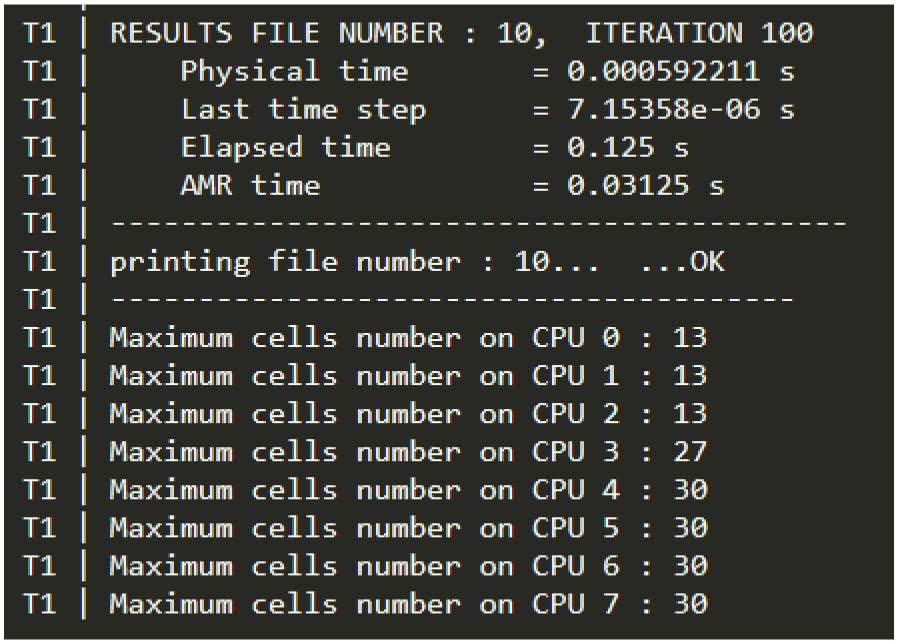
A new folder *results* is created at the first run, (unusefull to remove it). This folder contains a folder named *euler1DTransportPositiveVelocity* containing output files of our test case. Are included in :

- *collection.pvd* used in *Paraview* software and the associated *vtu* files



```
************************************************************
    ,---.     ,--,    .---.     ,--,    ,---.    .-. .-.
    | .-'   .' .')   / .-. )  .' .'     | .-'    |  \| |
    | `-.   |  |(_)  | | |(_) |  |  __  | `-.    |   | |
    | .-'   \  \     | | | |  \  \ ( _) | .-'    | |\  |
    |  `--.  \  `-.  \ `-' /   \  `-) ) |  `--.  | | |)|
    /( __.'   \____\  )---'    )\____/  /( __.'  /(  (_)
   (__)              (_)      (__)     (__)     (__)
************************************************************
************************************************************
          EXECUTION OF THE TEST CASE NUMBER : 1
************************************************************
T1 | Test case : ./libTests/referenceTestCases/euler/1D/transport/positiveVelocity/
T1 | Number of CPU : 8
T1 | printing file number : 0...  ...OK
T1 | -----------------------------------------
T1 | RESULTS FILE NUMBER : 1,  ITERATION 10
T1 |     Physical time      = 5.23954e-05 s
T1 |     Last time step     = 5.82171e-06 s
T1 |     Elapsed time       = 0.03125 s
T1 |     AMR time           = 0 s
T1 | -----------------------------------------
T1 | printing file number : 1...  ...OK
T1 | -----------------------------------------
T1 | RESULTS FILE NUMBER : 2,  ITERATION 20
```

Fig. 4.2: : Screenshot of the top of ECOGEN default run console. In this particular run, 8 CPU have been used.



```
T1 | RESULTS FILE NUMBER : 10,  ITERATION 100
T1 |     Physical time       = 0.000592211 s
T1 |     Last time step      = 7.15358e-06 s
T1 |     Elapsed time        = 0.125 s
T1 |     AMR time            = 0.03125 s
T1 | ------------------------------------------
T1 | printing file number : 10...  ...OK
T1 | ---------------------------------------
T1 | Maximum cells number on CPU 0 : 13
T1 | Maximum cells number on CPU 1 : 13
T1 | Maximum cells number on CPU 2 : 13
T1 | Maximum cells number on CPU 3 : 27
T1 | Maximum cells number on CPU 4 : 30
T1 | Maximum cells number on CPU 5 : 30
T1 | Maximum cells number on CPU 6 : 30
T1 | Maximum cells number on CPU 7 : 30
```

Fig. 4.3: : Screenshot of the end of the default run console with 8 CPU used.

- *infoCalcul.out*
- *infoMesh* folder
- *probes* folder
- *savesInput* folder: a kind of log folder that contains the *xml* files used for this run.

By default, output files are recorder in VTK XML format in separate files for each CPU, AMR level and TIME. A way to post-treat this output files is to open the *collection.pvd* file using Paraview software.

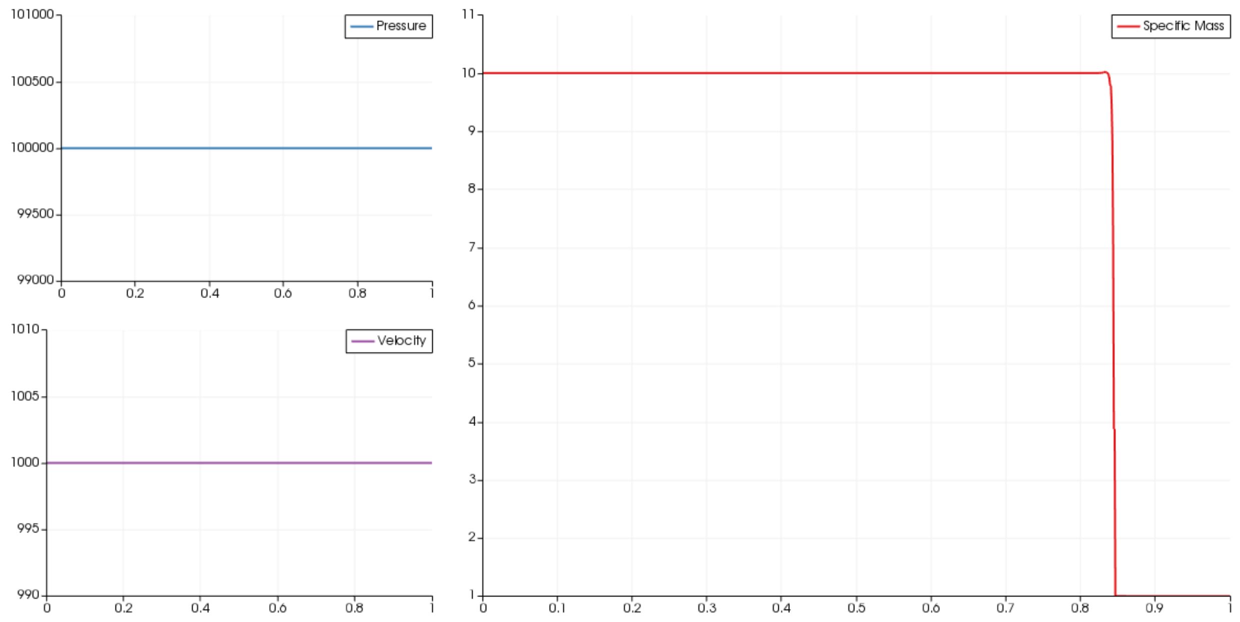


Fig. 4.4: Results for the single phase transport test

This basic test shows advection of a contact discontinuity while preserving pressure and velocity uniform conditions.

Editing input files

Input files for this test case are located in the following folder: *./libTests/referenceTestCases/euler/1D/transport/positiveVelocity/*.

Computation parameters will easily be modified according to the input file description of section *Input Files*

For example, one can prefer to visualise results under *gnu* file format. For that simply turn the *xml* option in the xml file *libTestsreferenceTestCaseseuler1DtransportpositiveVelocitymainV5.xml* into *gnu* and re-run the test case:

```
<outputMode format="GNU" binary="false" precision="10"/>
```

The results can be drawn by loading in the Gnuplot software the file *visualisation.gnu*.

4.2 Generating Meshes

ECOGEN is able to generate its own Cartesian meshes (see section *Cartesian mesh*), but for structured non Cartesian or unstructured meshes, as seen in section *Unstructured mesh*, an exeternal mesh software is required to generate mesh files.

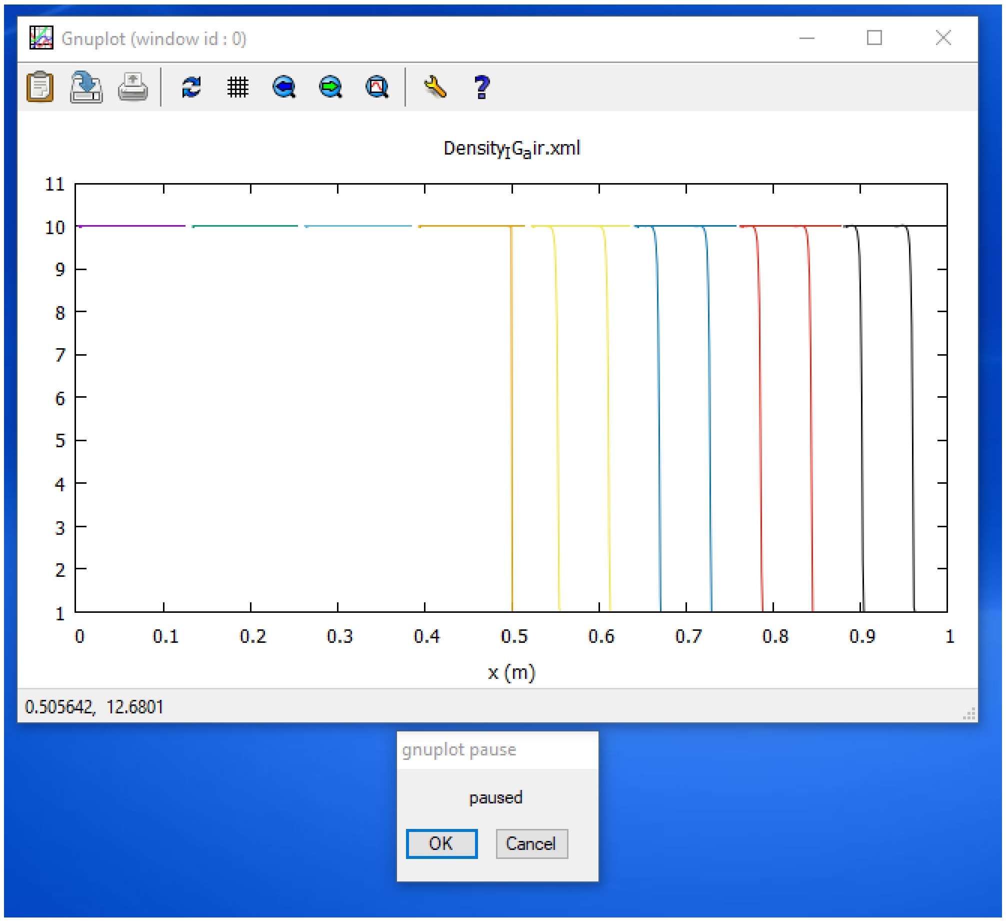


Fig. 4.5: : Screenshot of results in *Gnuplot*

4.2.1 Mesh files with Gmsh

ECOGEN can use mesh files for mono- or multiprocessors computations generated with the opensource Gmsh software [GR09] with some specific precaution when editing the geometry file (.geo). Only the MSH file format version 2 can be used in the current released version of ECOGEN.

Download binaries of Gmsh in version 3.0.6 or lower : http://gmsh.info/bin/.

Here are the restrictions that should be used when generating a geometry with Gmsh:

- Each part of the domain occupied by the fluid should correspond to a physical surface or a physical volume which is attribute to the value *10*.
- Each boundary condition must correspond to a physical line or a physical surface. The values are successively taken from *1* to maximum *9*. This is an important point that will be used to define boundary conditions physical treatment in the *initialConditionsV4.xml* input file described in section *InitialConditionsV4.xml*.

Below are presented examples:

Generating a nozzle structured mesh

Consider the geometry file of a simple nozzle depicted below:

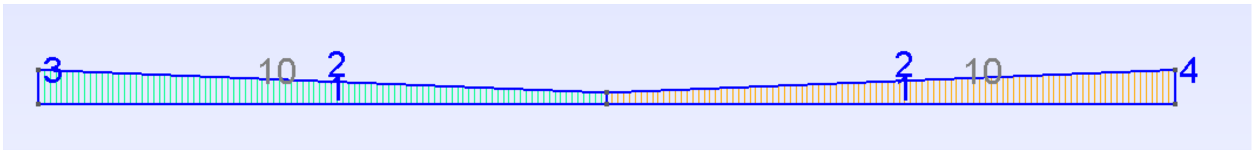


Fig. 4.6: Example of geometrical data file - nozzle2D_simple2.geo – for generating a mesh file at .msh format using Gmsh software and usable with ECOGEN.

This correponds to the geometry file available in ECOGEN/libMeshes/nozzles/nozzle2D_simple2.geo.

The computational domain is a nozzle, the mesh is unstructured with quadrangles. In that case, one should take care that the fluid surface is defined by the value *10* and that 4 boundaries conditions are set with the following numerotation:

- Symetrical axis: 1 (condLimAxe =1)
- Wall: 2 (condLimParoi =2)
- Inflow: 3 (condLimEntree =3)
- Outflow: 4 (condLimSortie =3)

The corresponding *initialConditionV4.xml* should then contains for example the following markup:

```
<!-- LIST OF BOUNDARY CONDITIONS -->
<boundaryConditions>
        <boundCond name="axe" type="wall" number="1" />
        <boundCond name="wall" type="wall" number="2" />
        <boundCond name="entrance" type="tank" number="3">
                <dataTank p0="2.e5" T0="187.5"/>
        </boundCond>
        <boundCond name="exit" type="outflow" number="4">
                <dataOutflow p0="1.e5"/>
        </boundCond>
</boundaryConditions>
```

That correponds to initialization of a nozzle connected to a tank on the left and to an outflow at imposed pressure to the right.

CHAPTER

FIVE

TEST CASES

ECOGEN is provided including several test cases. Major part of these tests can be found in companion papers. The full list of available test cases in the package are listed in the conductor input file *ECOGEN.xml*. Description of some provided test cases is presented in this section. We will refer to the test case following the line format to uncomment in *ECOGEN.xml* input file.

Example

For example, the following line in *ECOGEN.xml* input file corresponds to the default test:

```
<testCase>./libTests/referenceTestCases/euler/1D/transport/positiveVelocity/</
↪testCase>
```

It is presented in details in the tutorial section *Running the default test case* :

5.1 Single phase test cases

Test cases presented in this section are dealing with single phase compressible problems. In this part, ECOGEN solves the Euler equations [Eul57]:

$$
\begin{aligned}
&\frac{\partial \rho}{\partial t} + div(\rho \mathbf{u}) = 0\\
&\frac{\partial \rho \mathbf{u}}{\partial t} + div(\rho \mathbf{u} \otimes \mathbf{u} + p\mathbf{I}) = \mathbf{0}\\
&\frac{\partial \rho E}{\partial t} + div((\rho E + p)\mathbf{u}) = 0
\end{aligned}
$$

where ρ represents the density, $\mathbf{u}$ the velocity vector, p the pressure and $E = e + \frac{\mathbf{u}^2}{2}$ the total energy, with e the internal energy. The closure relation for this model is ensured by any convex equation of state (EOS) $p = p(\rho, e)$ (see section *Materials* for details about implemented EOS in ECOGEN). Euler equations are solved thanks to an explicit finite volume Godunov-like scheme [GZI+79] that is coupled with HLLC Riemann [Tor13] solver for fluxes computation.

5.1.1 Advection test cases

Basics advection test cases are proposed:

```
<testCase>./libTests/referenceTestCases/euler/1D/transport/positiveVelocity/</
↪testCase>
<testCase>./libTests/referenceTestCases/euler/1D/transport/negativeVelocity/</
↪testCase>
<testCase>./libTests/referenceTestCases/euler/2D/transports/rectangleDiagonal/</
↪testCase>
```

The first one is the default test case fully described in tutorial section *Running the default test case*. The second one is the reverse test advecting the contact discontinuity in the opposite direction.

rectangleDiagonal

This test is a 2D Cartesian test case with advection of a rectangle of high density air into low density air environnement. The computation uses the AMR techniques of [SPD19]. This test is referenced in ./libTests/referenceTestCases/euler/2D/transports/rectangleDiagonal/. A sketch of initial conditions for this test is presented in figure Fig. 5.1.

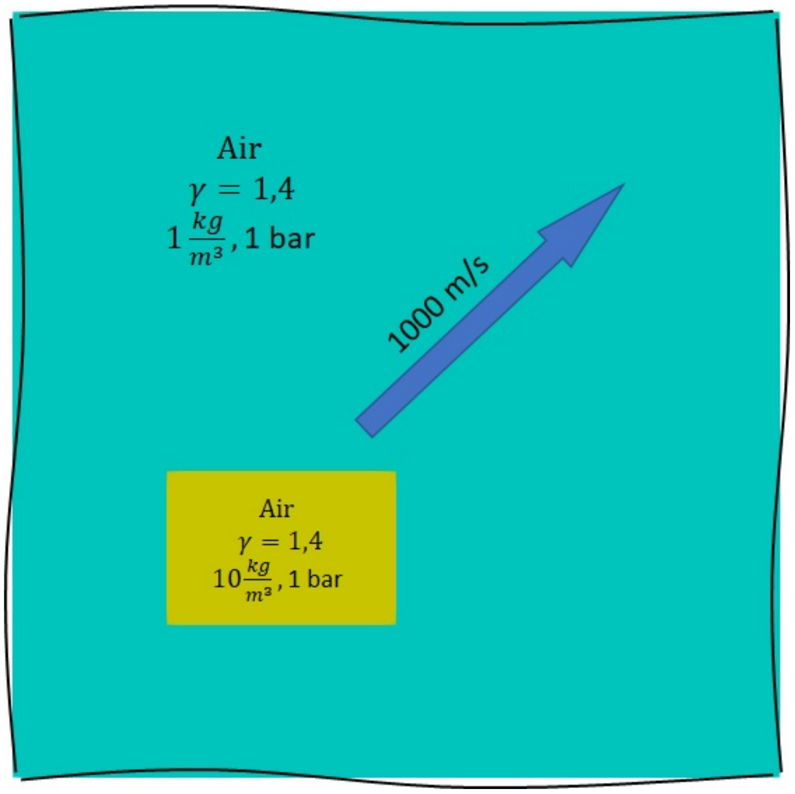


Fig. 5.1: Initial condition for single phase 2D transport test case.

The initial characteristics of the run are:

Characteristic	value
dimensions	1 m x 1 m
initial mesh size	50 x 50
AMR max level	2
rectangle position	0.2 m, 0.2 m
boundary conditions	transmittive
final solution time	0.36 ms
solution printing frequency	0.036 ms
precision	2nd order (superbee)

Bcause of adaptative mesh refinement, the final number of computational cells is 5383. Results are shown on figure Fig. 5.2. The left picture shows the evolution of mesh during simulation (red color is for high gradients of density).

Right top picture represents density contours and bottom right figure is to check for constant pressure preservation along the diagonal.

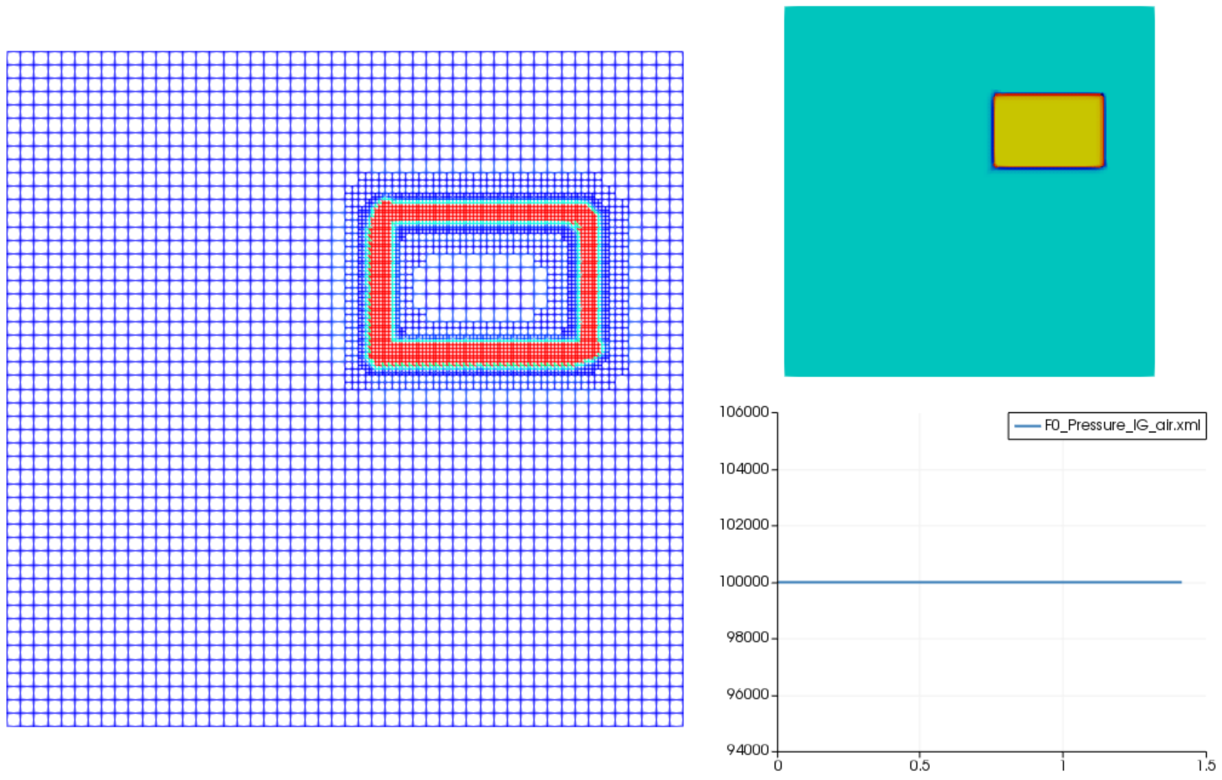


Fig. 5.2: Advection of a high density rectangle of air. Visualization using Paraview software.

5.1.2 Shock tubes

Single phase shock tubes are proposed in following test cases:

```
<testCase>./libTests/referenceTestCases/euler/1D/shockTubes/HPLeft/</testCase>
<testCase>./libTests/referenceTestCases/euler/1D/shockTubes/HPRight/</testCase>
<testCase>./libTests/referenceTestCases/euler/2D/HPCenter/</testCase>
<testCase>./libTests/referenceTestCases/euler/2D/HPUnstructured/</testCase>
```

HPLeft

This is a classical single phase shock tube filled with air. The test is available in the folder ./libTests/referenceTestCases/euler/1D/shockTubes/HPLeft/

The initial characteristics of the run are:

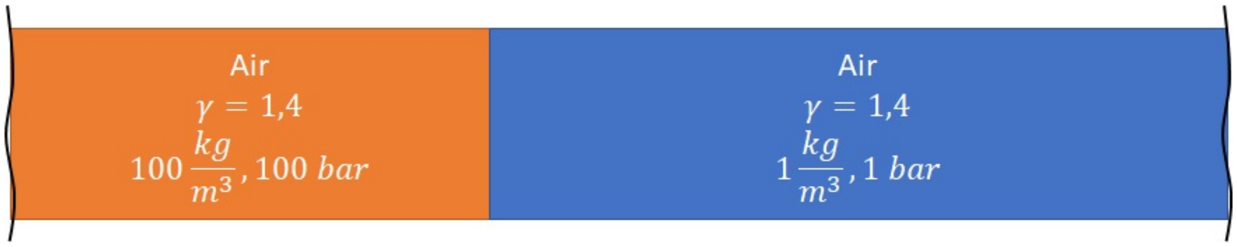


Fig. 5.3: Initial condition for single phase shock tube.

Characteristic	value
dimension	1 m
initial mesh size	100
AMR max level	3
diaphragm position	0.4 m
boundary conditions	transmittive
final solution time	0.6 ms
solution printing frequency	0.06 ms
precision	2nd order (vanleer)

Solution of this Riemann problem induces 3 waves:

- A fan of rarefaction waves propagating in high pressure chamber
- A Right-facing shock wave propagating in low pressure chamber
- A contact discontinuity

This three waves are clearly visible on the results:

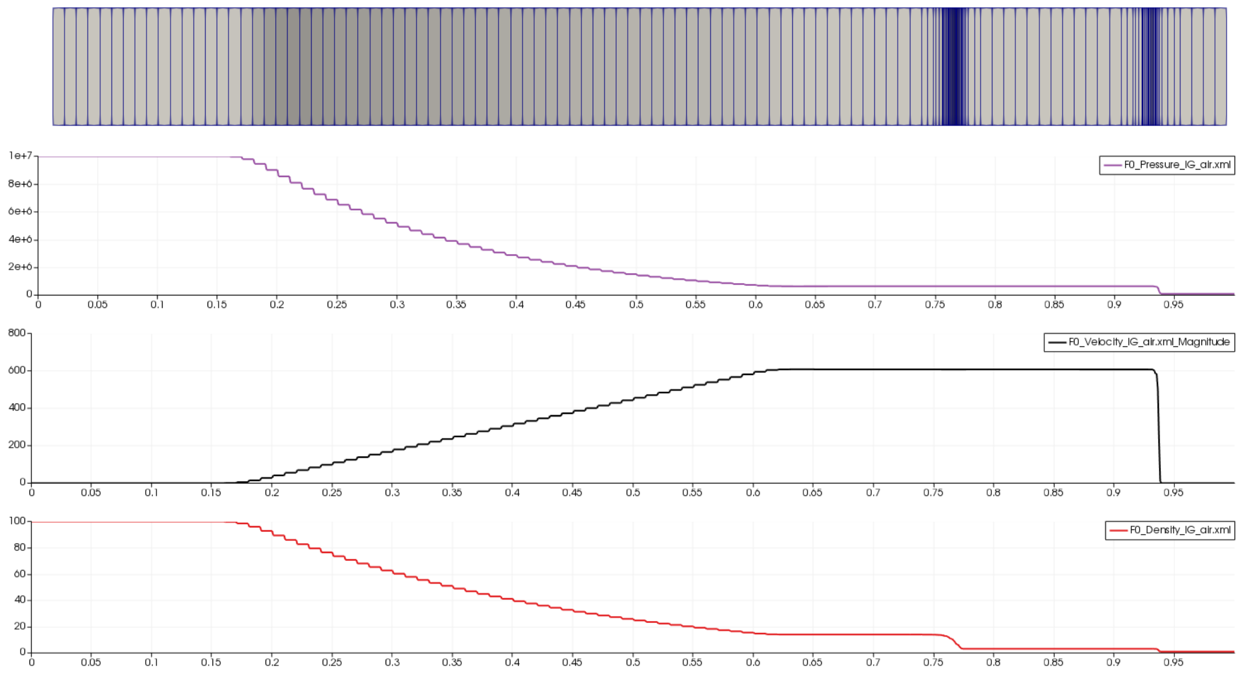


Fig. 5.4: Shock tube filled with air. Visualization using Paraview software.

This test is also equipped with 3 Eulerian sensors. For example, two sensors are positionned at $x = 0.6m$ and $x = 0.8m$. They record the following pressures:

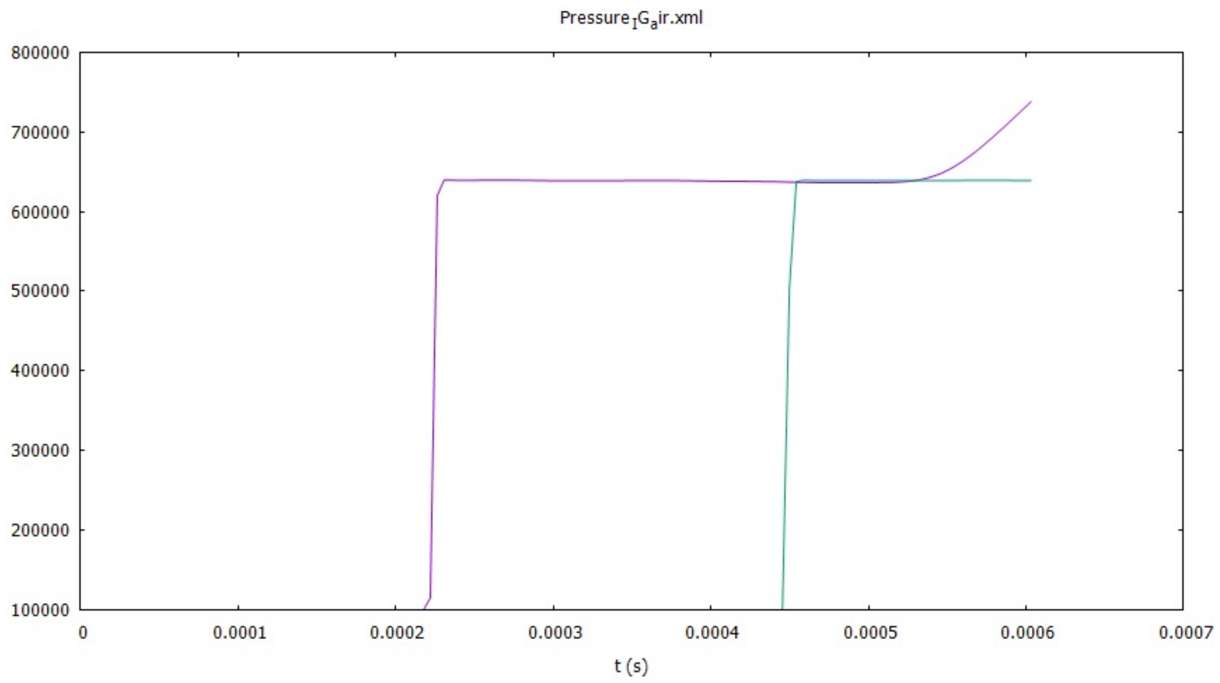


Fig. 5.5: Pressure recorded by sensors at $x = 0.6m$ (pink color) and $x = 0.8m$ (green color). Visualization using gnuplot software.

The first sensor see its pressure rising because of the shock wave before the second one. Because this Riemann problem generates a supersonic flow after the shock wave, the tail of the rarefaction waves fan is seen by the sensor after 0.5 ms.

5.1.3 Other test cases

Other tests are provided with ECOGEN package. They will be described in details soon.

```
<testCase>./libTests/referenceTestCases/euler/2D/HPCenter/</testCase>
<testCase>./libTests/referenceTestCases/euler/2D/HPUnstructured/</testCase>
<testCase>./libTests/referenceTestCases/euler/2D/nozzles/tankWithShock/</testCase>
<testCase>./libTests/referenceTestCases/euler/3D/LPCenter/</testCase>
```

5.2 Multiphase mechanical equilibrium flows

Mechanical equilibrium flows are solved in ECOGEN using the Kapila model [KMB+01]. In the particular case of 2 phases invilved, this model reads:

$$\left\{\begin{array}{ll} \dfrac{\partial \alpha_1}{\partial t} + \mathbf{u}\cdot\nabla\alpha_1 & = K div(\mathbf{u}), \\ \dfrac{\partial \alpha_1\rho_1}{\partial t} + div\left(\alpha_1\rho_1\mathbf{u}\right) & = 0, \\ \dfrac{\partial \alpha_2\rho_2}{\partial t} + div\left(\alpha_2\rho_2\mathbf{u}\right) & = 0, \\ \dfrac{\partial \rho\mathbf{u}}{\partial t} + div\left(\rho\mathbf{u}\otimes\mathbf{u} + p\mathbf{I}\right) & = \mathbf{0}, \\ \dfrac{\partial \rho E}{\partial t} + div\left(\left(\rho E + p\right)\mathbf{u}\right) & = 0, \end{array}\right. \tag{5.1}$$

where subscripts 1 and 2 correspond to one of the two phases, respectively. α_k and ρ_k are the volume fraction and density of phase k.

$\rho = \sum\limits_k \alpha_k\rho_k$, $\mathbf{u}$, p, $E = e + \dfrac{1}{2}\|\mathbf{u}\|^2$ and $e = \sum_k \alpha_k\rho_k e_k$ are the mixture density, velocity, pressure, total energy and internal energy, respectively.

The term $Kdiv(\mathbf{u})$ accounts for the differences in the acoustic behavior of both phases or in other words, for the differences in expansion and compression of each phase in mixture regions. K is given by:

$$K = \frac{\rho_2 s_2^2 - \rho_1 s_1^2}{\dfrac{\rho_2 s_2^2}{\alpha_2} + \dfrac{\rho_1 s_1^2}{\alpha_1}},$$

s_k being the speed of sound of phase k.

This model is solved thanks to the numerical method presented in [SPB09].

5.2.1 Advection test cases

The code is provided with following test cases for advections:

```
<testCase>./libTests/referenceTestCases/kapila/1D/transports/interfaceWaterAir/</
↪testCase>
<testCase>./libTests/referenceTestCases/kapila/2D/transportWaterSquareInAir/</
↪testCase>
```

interfaceWaterAir

This first test case is important since it validates the capacity of the numerical method to treat simple advection of an interface between pure fluid without creating spurious oscillations on pressure or velocity profiles. Input files for this test are available in ./libTests/referenceTestCases/kapila/1D/transports/interfaceWaterAir/

The initial characteristics of the run are:

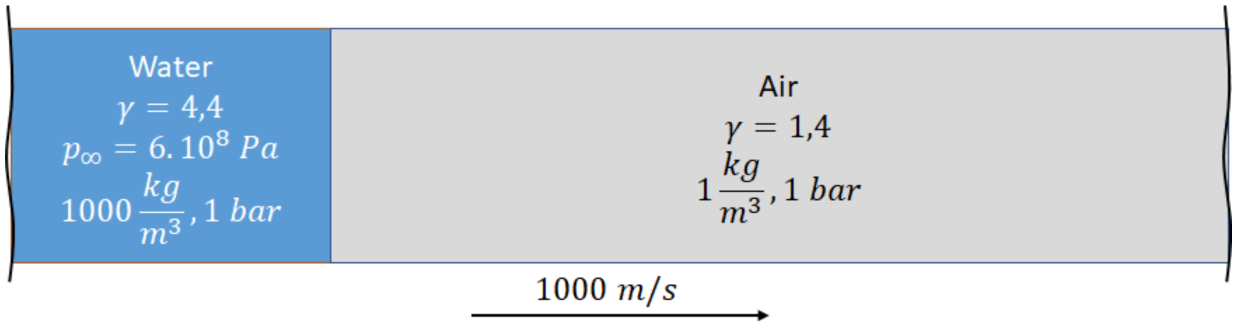


Fig. 5.6: Initial condition for 1D advection of water/air interface.

Characteristic	value
dimension	1 m
Mesh size	800
interface position	0.3 m
boundary conditions	transmittive
final solution time	0.7 ms
solution printing frequency	0.025 ms
precision	1st order

In the default case, the computation is performed with a 1st order scheme. We compare this solution with those obtained using the 2nd order scheme with THINC limiter [SX14].

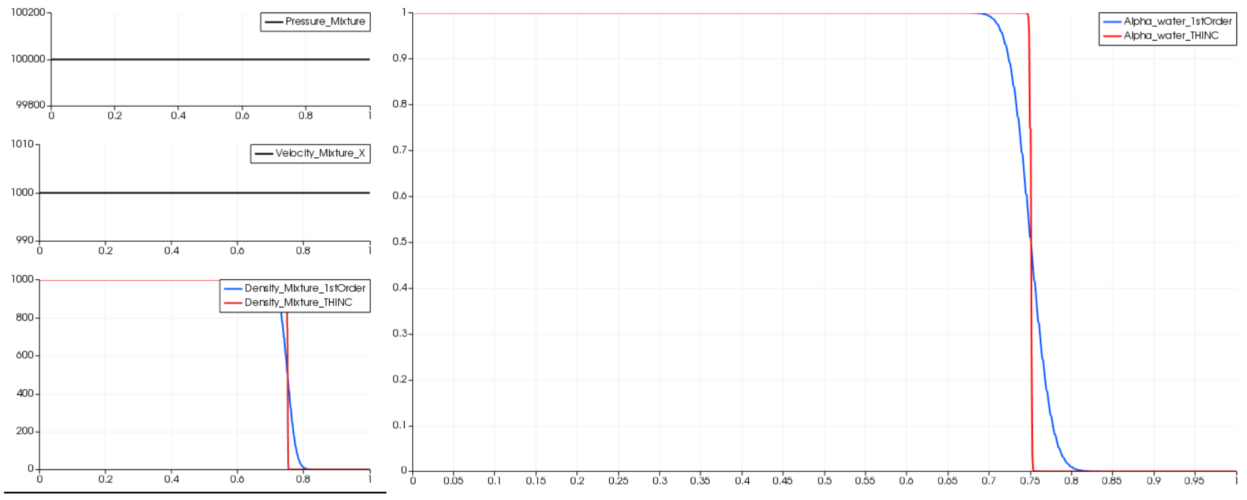


Fig. 5.7: Advection of a water/air interface. Visualization using Paraview software.

5.2.2 Shock tubes

The test cases relative to Kapila model are those presented in [SPB09]. They are here reproduced using ECOGEN.

```
<testCase>./libTests/referenceTestCases/kapila/1D/shockTubes/interfaceWaterAir/</
↪testCase>
<testCase>./libTests/referenceTestCases/kapila/1D/shockTubes/epoxySpinel/</testCase>
```

interfaceWaterAir shock tube

A shock tube between a high pressure chamber filled with water and a low pressire chamber filled with air is released. Input files for this test are available in ./libTests/referenceTestCases/kapila/1D/shockTubes/interfaceWaterAir/.

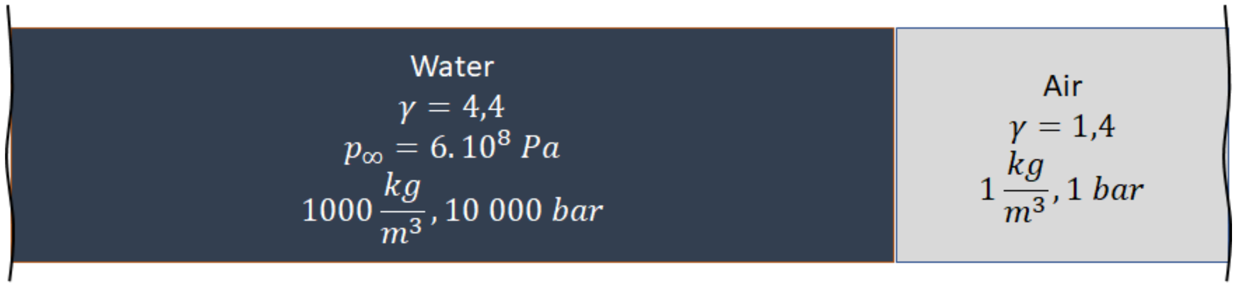

Water
$\gamma = 4{,}4$
$p_\infty = 6.10^8\ Pa$
$1000\,\frac{kg}{m^3}, 10\,000\ bar$
Air
$\gamma = 1{,}4$
$1\,\frac{kg}{m^3}, 1\ bar$

Fig. 5.8: Initial condition for 1D advection of water/air interface.

The initial characteristics of the run are:

Characteristic	value
dimension	1 m
Initial mesh size / max size	100 / 230
number of refinement level	4
diaphragm position	0.7 m
boundary conditions	transmittive
final solution time	0.240 ms
solution printing frequency	0.012 ms
precision	2nd order (Vanleer/THINC)

AMR technique of [SPD19] is used with 4 refinement level such that a maximum of 230 computational cells are used for this run.

epoxySpinel

This test deals with shocks in mixture of materials. Epoxy and Spinel are supposed mixed such that caracteristic times for wave propagation and drag effects are very small allowing to consider the mixture as evolving in mechanical equilibrium. Input files for this test are available in ./libTests/referenceTestCases/kapila/1D/shockTubes/epoxySpinel/.

The initial characteristics of the run are:

Characteristic	value
dimension	1 m
Initial mesh size / max size	200 / 237
number of refinement level	2
diaphragm position	0.6 m
boundary conditions	transmittive
final solution time	0.1 ms
solution printing frequency	0.025 ms
precision	2nd order (Vanleer)

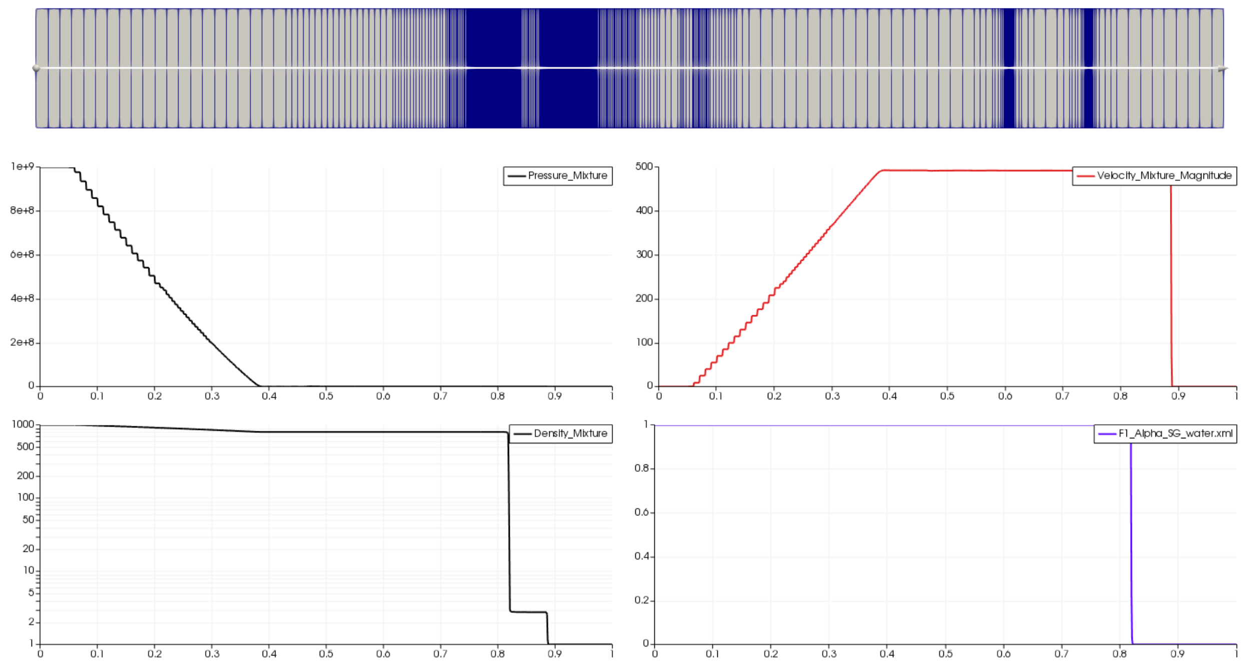
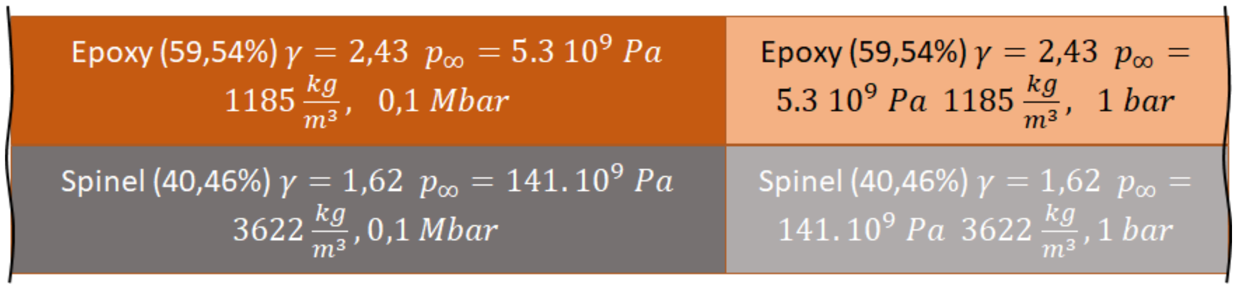


Fig. 5.9: Shock tube with water and air. Visualization using Paraview software.



Epoxy (59,54%) $\gamma = 2{,}43$ $p_\infty = 5.3\ 10^9\ Pa$ $1185\,\frac{kg}{m^3}$, $0{,}1\ Mbar$	Epoxy (59,54%) $\gamma = 2{,}43$ $p_\infty = 5.3\ 10^9\ Pa$ $1185\,\frac{kg}{m^3}$, $1\ bar$
Spinel (40,46%) $\gamma = 1{,}62$ $p_\infty = 141.10^9\ Pa$ $3622\,\frac{kg}{m^3}$, $0{,}1\ Mbar$	Spinel (40,46%) $\gamma = 1{,}62$ $p_\infty = 141.10^9\ Pa$ $3622\,\frac{kg}{m^3}$, $1\ bar$

Fig. 5.10: Initial condition for mixture shock tube with epoxy and spinel.

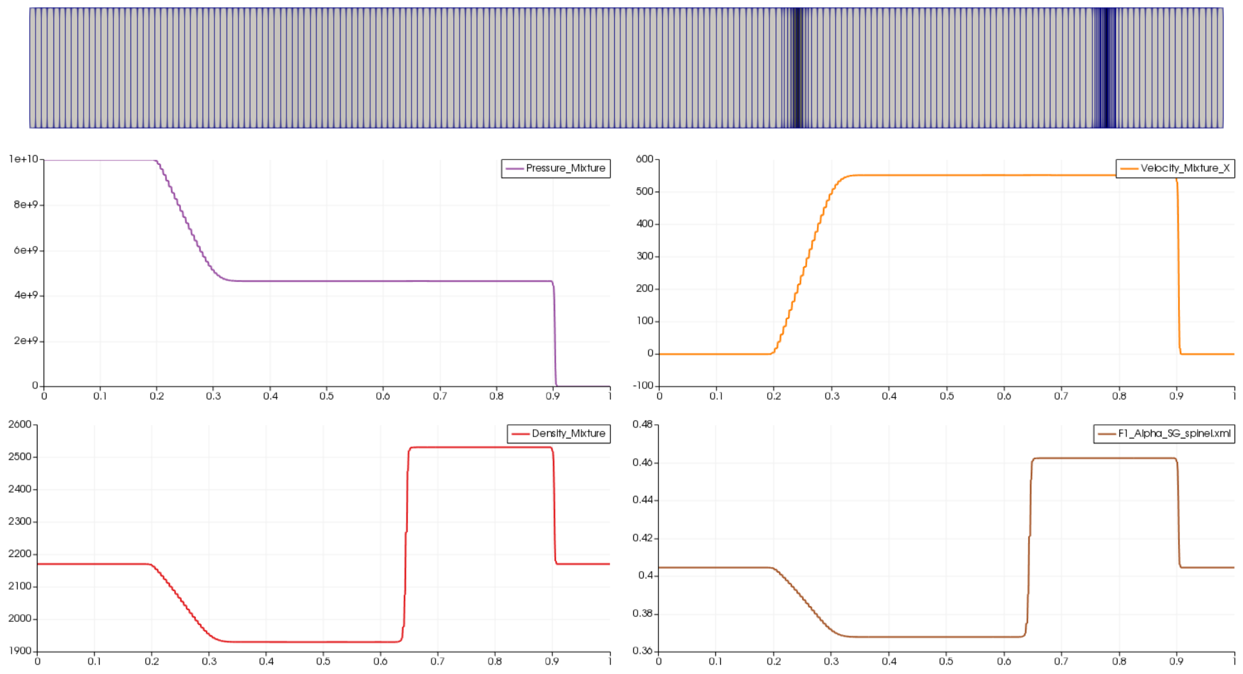


Fig. 5.11: Mixture shock tube with expoxy and spinel. Visualization using Paraview software.

5.2.3 Other tests cases

Other tests are provided with ECOGEN package. They will be described in details soon.

```
<testCase>./libTests/referenceTestCases/kapila/1D/cavitation/</testCase>
<testCase>./libTests/referenceTestCases/kapila/2D/transportWaterSquareInAir/</
↪testCase>
<testCase>./libTests/referenceTestCases/kapila/2D/squareWaterExplosion/</testCase>
<testCase>./libTests/referenceTestCases/kapila/2D/shockBubble/heliumAir/</testCase>
<testCase>./libTests/referenceTestCases/kapila/2D/richtmyerMeshkov/</testCase>
<testCase>./libTests/referenceTestCases/kapila/2D/testUnstructured/</testCase>
<testCase>./libTests/referenceTestCases/Kapila/AddPhysicalEffects/evap/evapShockTube/
↪</testCase>
<testCase>./libTests/referenceTestCases/Kapila/AddPhysicalEffects/evap/
↪dodEvapShockTube/</testCase>
<testCase>./libTests/referenceTestCases/kapila/AddPhysicalEffects/surfaceTension/
↪squareToCircle/</testCase>
<testCase>./libTests/referenceTestCases/kapila/AddPhysicalEffects/surfaceTension/
↪squareToCircleSymmetry/</testCase>
<testCase>./libTests/referenceTestCases/kapila/AddPhysicalEffects/surfaceTension/
↪waterCylinderInAir/</testCase>
<testCase>./libTests/referenceTestCases/kapila/AddPhysicalEffects/surfaceTension/
↪waterDropletInAir/</testCase>
<testCase>./libTests/referenceTestCases/kapila/AddPhysicalEffects/surfaceTension/
↪dropletImpact/</testCase>
<testCase>./libTests/referenceTestCases/kapila/AddPhysicalEffects/gravity/</testCase>
<testCase>./libTests/referenceTestCases/kapila/3D/unstructured/</testCase>
<testCase>./libTests/referenceTestCases/kapila/3D/shockBubble/heliumAir/</testCase>
```

CHAPTER

SIX

DEVELOPERS

ECOGEN is open source, so interested developers are welcome to collaborate improving the software. This section includes various informations on how to easily and efficiently contribute to the project.

Access to the API documentation

6.1 Contribute to ECOGEN

Before contributing, please read the section *License* informations.

6.1.1 API documentation

An API documentation is available. It is generated thanks to Doxygen and special comments in the source code. Interested developer will find informations on the different classes and functions used in ECOGEN.

When contributing to the project and to automatise code documentation using Doxygen, comments should be insert as follow in header files.

At the head of files

```
//! \file      file name
//! \author    authors names
//! \version   1.0
//! \date      12 Novembre 2009
//! \brief     brief description
```

Before class definition

```
//! \class     class name
//! \brief     brief descritption
//! \details    Detailed description
//!                  continue description
```

Before function and method prototypes

```
//! \brief      brief description
//! \details    Detailed description
//!                  continue description
//! \param      parameter name         description
//! \param      parameter name         description
//! \return     return description
```

Class member descrption

```
type m_variable; //!< member description
```

6.1.2 Coding constraints

Developers are thanks to respect some constraints when contributing to the project.

Variable and Classe names

1. Variable name should began with lowercase letter and be self-understandable. Each new word began with a uppercase letter.

```
int myInteger; vector<double *> vectorOfDoublePointer; etc.
```

2. Class attribute should began by "m_".

```
int m_myInteger; double * m_doublePointer; etc.
```

3. Class name should began with an uppercase Letter.

```
class MyClass;
```

Developer personnal comments - flags

Developer personnal comments should be included using the following template:

//Developer//KeyWord// comments

```
//FP//DEV// comment, description
```

Here is the list of keyword to use :

```
//DEV//     in developement
//Q//       question to dig
//TODO//    should be done in the future
//ERR//     error : to correct ASAP
//ID//      idea
//ICI//     Stop developement position
//VERIF//   to verify : is it needed ?
//TEST//    test : To delete ASAP
```

6.1.3 Git-hub submit

For each modification, a comment should be prepared to be included to the commit message for Git.

6.2 Building Sphinx Doc

6.2.1 Prerequisities

This documentation uses Sphinx third-party Python package. To generate it locally from the **docs/sphinx_docs/** folder, you need to install additional components.

Install Python

To install python, you can install for example the Anaconda package or the python offical package. You can also install the packages directly from ubuntu terminal:

```
apt-get install python3
apt-get install python-pip
```

Install Sphinx

From the prompt, it is possible to install sphinx, as well as additional libraries required for building this documentation using pip:

```
pip install sphinx
pip install sphinx_rtd_theme
pip install sphinx-numfig
pip install sphinxcontrib-bibtex
```

Install Latex

If you want to build pdf, you will need Latex installed

```
apt-get install texlive
apt-get install texlive-latex-extra
apt-get install latexmk
```

6.2.2 Building html doc

To build the docuentation as a webpage (as shown on ECOGEN webSite), move to the *docs* folder and run under prompt:

```
make html
```

6.2.3 Building pdf doc

To build the docuentation as a PDF, move to the *docs* folder and run under prompt:

```
make latexpdf
```

This will generate a folder *docs/build/latex* containing source files in *LaTeX* that can be used to generate a pdf.

6.2.4 Learning Sphinx

To learn how to developp a documentation using sphinx, here are some usefull links:

- Sphinx documentation
- Hosting documentation and read the docs theme : Read the docs website

CHAPTER

SEVEN

LICENSE

ECOGEN is the legal property of its developers, whose names are listed in the copyright file included with the source distribution. Contributors' names for version 1.0 are listed below:

- Eric Daniel,
- Sébastien Le Martelot,
- Kevin Schmidmayer,
- Fabien Petitpas

CHAPTER

EIGHT

BIBLIOGRAPHY

A list of the references used in this documentation:

CHAPTER

NINE

TABLE OF CONTENTS

BIBLIOGRAPHY

[Eul57] Leonhard Euler. Principes généraux du mouvement des fluides. *Mémoires de l'Académie des Sciences de Berlin*, pages 274–315, 1757.

[GR09] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[GZI+79] S.K. Godunov, A. Zabrodin, M. Ivanov, A. Kraiko, and G. Prokopov. *Resolution numerique des problemes multidimensionnels de la dynamique des gaz*. Editions Mir, Moscou, 1979.

[HA68] F. H. Harlow and A. A. Amsden. Numerical calculation of almost incompressible flow. *Journal of Computational Physics*, 3(1):80–93, 1968.

[KMB+01] A. Kapila, R. Menikoff, J. Bdzil, S. Son, and D. Stewart. Two-phase modeling of \textsc DDT in granular materials: \textsc Reduced equations. *Physics of Fluids*, 13:3002–3024, 2001.

[Roe86] P. L. Roe. Characteristic-based schemes for the Euler equations. *Annual review of fluid mechanics*, 18(1):337–365, 1986.

[SPB09] R. Saurel, F. Petitpas, and R.A. Berry. Simple and efficient relaxation methods for interfaces separating compressible fluids, cavitating flows and shocks in multiphase mixtures. *Journal of Computational Physics*, 228(5):1678–1712, 2009.

[SPD19] K. Schmidmayer, F. Petitpas, and E. Daniel. Adaptive mesh refinement algorithm based on dual trees for cells and faces for multiphase compressible flows. *Journal of Computational Physics*, 2019.

[SX14] K.M. Shyue and F. Xiao. An Eulerian interface sharpening algorithm for compressible two-phase flow: the algebraic THINC approach. *Journal of Computational Physics*, 268:326–354, 2014.

[Tor13] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[VAVLR97] G.D. Van Albada, B. Van Leer, and W. Roberts. A comparative study of computational methods in cosmic gas dynamics. In *Upwind and High-Resolution Schemes*, pages 95–103. Springer, 1997.

[VL74] B. Van Leer. Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme. *Journal of Computational Physics*, 14(4):361–370, 1974.

[VL77] Bram Van Leer. Towards the ultimate conservative difference scheme. iv. a new approach to numerical convection. *Journal of computational physics*, 23(3):276–299, 1977.

[LeMetayerMS04] O. Le Métayer, J. Massoni, and R. Saurel. Elaborating equations of state of a liquid and its vapor for two-phase flow models. *International Journal of Thermal Sciences*, 43:265–276, 2004.